USER GUIDE

Essential Studio

for ASP.NET MVC

Version - v18.1.0.36 | Release Date - March 19, 2020

Syncfusion

Welcome to Syncfusion Essential ASP.NET MVC 100
How to best read this User Guide 100
Additional Help Resources 100
Create a Support Incident 100
System Requirement 100
Operating Systems 100
Hardware Environment 101
Web browsers 101
Development Environment 101
Download Installer 101
Download the Trial Version 101
Free Trial Page 101
Start Trial Page 102
Download the License Version 102
Installation and Upgrade 103
Installation 103
Uninstallation 113
Installation and Uninstallation 118
Installation using Offline Installer 127
Installing with UI 127
Installing in silent mode 133
Installation FAQ 135
Upgrade from one version to another version. 135
Upgrade from major version to service pack version 135
Upgrade from trial version to license version 135
Configuration and Deployment 135
Configuring Syncfusion Bower Packages 135
Overview 135
Bower Installation 135
Visual Studio Integration 136
Visual Studio Extensions 136
CDN 167
Getting Started 167
What is Html.EJ() ? 167
Getting started with Syncfusion MVC 167

Through Syncfusion Project Template ... 167
Through Syncfusion Project Conversion ... 172
Through Syncfusion NuGet Packages ... 177
Manual Integration ... 179
Creation of First ASP.NET MVC Application ... 179
For Existing Applications ... 180
Adding the required JavaScript files ... 182
CDN Link reference ... 185
Assembly Reference ... 185
Registering Syncfusion Assemblies within the Web.config ... 188
Version compatibility with respect to framework ... 189
Getting Started ... 190
Configuring EJ MVC in Visual Studio 2019 ... 190
Utilities ... 198
Project Conversion ... 198
Convert into Syncfusion ASP.NET MVC (Web) project ... 198
Rendering Control after Syncfusion ASP.NET MVC (Web) Conversion: ... 203
Project Migration ... 204
Sample Creator ... 206
Create Syncfusion MVC Project from Sample Creator ... 206
Core Concepts ... 213
Core Concepts ... 213
Basics ... 213
Localization ... 215
Globalization ... 216
Dynamic Globalization ... 218
Unobtrusive ... 219
Enabling Client Side Validation ... 221
Validation ... 224
jQuery Validation ... 224
Accessibility ... 225
Creating accessible widgets using WAI-ARIA ... 225
Usage of Accessibility in Syncfusion Widgets ... 226
IntelliSense ... 228
Exporting Multiple Syncfusion Controls ... 229

Strongly-Typed HTML Helper 232
Client Side Validation 234
Server Side Validation 236
Complex Model Binding 237
Syntax for VB 239
Accordion 240
Accordion 240
Key Features 240
Getting Started 240
Create your first Accordion in ASP.NET 241
Configure Accordion 241
Create a Simple Accordion 242
Configure Multiple Open 244
Set Rounded corner 247
Customize Icon 248
Header customization 250
Collapsible 250
Enable Collapsible settings 250
Single selection 253
Multiple selection 254
AJAX settings 256
Populate accordion with AJAX content 256
View multiple contents 257
Enabling multiple panel open 257
Accordion Panel enabler 258
Enable or Disable control 258
Enable panel items 259
State Persistence 261
Configure state persistence of Accordion panel 261
Appearance and Styling 263
Adjusting Accordion size 263
Rounded corner 264
Customize Accordion icon 265
Animation Options 266
Theme 268

CSS class ... 268
RTL Support ... 270
Enabling RTL Support ... 270
Keyboard Navigation ... 271
Configure keyboard interaction ... 271
Template Support ... 273
Content Template ... 273
Header Template ... 274
How To ... 275
Render Accordion from code behind ... 275
AutoComplete ... 276
AutoComplete ... 276
Key Features ... 276
Getting Started ... 277
Create your first AutoComplete in MVC ... 277
Data-Binding ... 281
Fields ... 281
Local data ... 281
Remote data ... 283
Behavior Settings ... 286
Filtering Type ... 286
AutoFill ... 287
Sorting Items ... 288
Distinct List items ... 288
Show Popup button ... 289
Restrict editing ... 290
Empty Result settings ... 290
Search Customization ... 291
HighlightSearch ... 291
Case-sensitive Search ... 291
MultiSelection modes ... 292
Configuring MultiSelection Mode ... 292
Multiple Columns ... 293
Configuring Multiple Columns ... 294
Grouping ... 294

Configuring Grouping for AutoComplete 294
State Persistence 296
Configure state persistence of AutoComplete 296
Templates 297
Configuring Templates 297
Appearance and Styling 299
Adjusting AutoComplete size 299
Rounded corner 300
Watermark text 300
Adjusting Suggestion list size 301
PopUp Time Delay 302
Theme 302
CSS Class 302
RTL support 303
Enabling RTL Support 303
Keyboard Interaction 304
Configure Keyboard Interaction 304
How to 305
Render Autocomplete from Code behind 305
How to set autocomplete default value 306
How to show text on autocomplete using key value 309
Barcode 311
Barcode 311
Key Features 311
Getting Started 312
Control Structure 312
Create your first Barcode in MVC 312
Barcode types and supported characters 314
Two Dimensional Barcodes 315
QR Barcode 315
DataMatrix 315
One Dimensional Barcodes 316
Code 39 316
Code 39 Extended 316
Code 11 316

Code 32 317
Code 93 317
Code 93 Extended 317
Code 128 317
Code 128 Code Sets 317
Code 128 Special characters 318
Customizing the appearance 318
BulletGraph 319
BulletGraph 319
Key Features 319
Getting Started 319
Create your first BulletGraph in MVC 319
Bullet Graph Elements 328
Bullet Graph Dimensions 329
Size 329
Value for performance bar 330
Comparative measure value 330
Theme 330
Orientation 331
Flow direction 331
Qualitative range size 331
Quantitative scale length 331
Bullet Graph Caption 332
Subtitle 333
Indicator 334
Trim 335
Text Placement 335
Data Binding 338
Local Data 338
Remote Data 339
Quantitative Scale 340
Range for Quantitative Scale 340
Quantitative scale location 340
Major ticks 341
Minor ticks 341

Tick position 342
Tick Placement 342
Quantitative scale labels 343
Label Placement 343
Performance measure bar 344
Comparative measure symbol 344
Multiple performance measures comparison 345
Qualitative Range 345
User Interaction 346
Animation 346
Responsiveness during browser resize 346
Applying same color to all ticks and labels in a range 347
Tooltip 347
Methods 348
Events 349
Button 352
Button 352
Key Features 352
Getting Started 353
Create your first Button in MVC 353
Create Button Control in MVC 353
Easy Customization 357
Button Size 357
Content Type 358
Prefix and Suffix icons 359
Image Position 360
Button Type 364
Repeat Button 365
RTL Support 366
Icons 366
Adding icon in Button 367
List of Icons 367
Miscellaneous 371
Text 371
Show Rounded Corner 371

How to? ... 372
Render Button from Code behind ... 372
Captcha ... 373
Captcha ... 373
Key Features ... 373
Getting Started ... 373
Create your first Captcha in ASP.NET MVC ... 373
Behavior Settings ... 378
Regenerate Captcha ... 378
Audio Accessibility ... 379
Enabling Validation ... 379
Auto Validation ... 379
Validation by Method ... 380
Case Sensitive Validation ... 381
Appearance and Styling ... 382
Hatch Styles ... 382
Pattern ... 383
Background and Font color ... 383
Adjusting Captcha Size ... 384
Theme ... 384
Maximum and Minimum length ... 385
Character Set Customization ... 385
Localization ... 386
RTL Support ... 386
Chart ... 387
Overview ... 387
Getting Started ... 387
Installation and Deployment ... 387
Initialize chart ... 389
Populate chart with data ... 389
Add Data Labels ... 392
Enable Legend ... 394
Enable Tooltip ... 395
Add Chart Title ... 396
Working with Data ... 397

Local Data 397
Remote Data 399
AngularJS Data Binding 400
Chart Dimensions 402
Set size for the container 402
Set size in pixels 402
Setting size relative to the container size 403
Responsive chart 403
Axis 404
Category Axis 404
Numeric Axis 409
DateTime Axis 417
DateTime Category Axis 424
Logarithmic Axis 427
Label Format 431
Common axis features 435
Multiple Axis 455
Smart Axis Labels 457
Multi-level Labels 464
Multiple panes 469
Column Definitions 472
ChartTypes 474
Line Chart 474
Step Line Chart 480
Area Chart 485
Range Area Chart 486
Step Area Chart 487
Spline Area Chart 488
Stacked Area Chart 489
100% Stacked Area Chart 490
Column Chart 491
RangeColumn Chart 497
Stacked Column Chart 499
100% Stacked Column Chart 502
Bar Chart 505

Stacked Bar Chart 507
100% Stacked Bar Chart 510
Spline Chart 513
Pie Chart 516
Doughnut Chart 522
Multiple Pie Chart 529
Pyramid Chart 533
Funnel Chart 537
Bubble Chart 540
Scatter 541
HiLoOpenClose Chart 543
Candle 546
HiLo 548
Polar 549
Radar Chart 552
Waterfall Chart 555
ErrorBar Chart 558
Box and Whisker Chart 563
Pie Of Pie Chart 566
Chart Series 570
Multiple Series 570
Combination Series 572
Data Markers 574
Add Shapes 575
Add image as marker 576
Add labels 577
Binding label from the datasource 585
Binding fill color to the points from the datasource 585
Customize specific points 586
Connect Line 589
Smart labels 590
Legend 591
Legend Visibility 591
Legend title 592
Position and Align the Legend 593

Arrange legend items in the rows and columns 595
Customization 596
Handle the legend item clicked 606
Series selection on legend item click 606
Collapsing legend item 607
Empty Points 608
EmptyPointSettings 609
Customizing Styles 611
Chart Title & Subtitle 612
Title 612
Add Subtitle to the chart 615
Striplines 618
Horizontal Stripline 618
Vertical Stripline 619
Customize the Text 620
Customize the Stripline 622
Change the Z-order of the stripline 623
User Interactions 624
Tooltip 624
Zooming and Panning 629
Crosshair 633
Trackball 635
Highlight 639
Selection 645
Data Editing 655
Performance 657
Lazy Loading 658
Trendlines 658
Customize the trendline styles 659
Types of Trendline 660
Forecasting 666
Trendlines Legend 668
Technical Indicators 669
Bind data to render the indicator 669
Indicator Types 670

Enable Tooltip 681
Annotations 682
Rotate the annotation template 683
Positioning Annotation 684
Annotation alignments 685
Appearance 686
Custom Color Palette 686
Built-in Themes 687
Point level customization 688
Series border customization 689
Chart area customization 690
Rendering Modes 696
VML 696
SVG 696
Canvas 696
Real-Time Chart 697
Exporting Chart 698
Client-side Exporting 698
Server-side Exporting 699
Multiple Chart Excel Exporting 702
Naming the exported file 703
Rotating the chart 703
Setting orientation for the document 704
Printing Chart 704
Printing Multiple chart 705
Page Setup 706
3D Chart 706
3D Column Chart 706
3D Bar Chart 707
3D Stacked Column Chart 708
3D 100% Stacked Column Chart 709
3D Stacked Bar Chart 710
3D 100% Stacked Bar Chart 711
3D Pie Chart 712
3D Doughnut Chart 713

Configure 3D Chart 714
Localization 720
Checkbox 721
Checkbox 721
Key Features 721
Getting Started 721
Create your first Checkbox in MVC 721
Easy customization 726
Checked state 726
Enable Tri-State 727
Check State 727
Checkbox Size 728
Text 729
Rounded corner for checkbox 729
Styles Customization 729
Miscellaneous 730
Checkbox Id 730
Checkbox Id Prefix 730
Checkbox Name 731
Checkbox Value 731
How to? 731
Render Checkbox from Code behind 731
CircularGauge 732
CircularGauge 732
Key Features 732
Getting Started 733
Create a Circular Gauge 733
Set Height and Width 734
Set Background Color 735
Provide Scale Values 736
Add Label Customization 737
Add Pointers 738
Add Tick Details 739
Add Range Values 740
Add Indicator Details 742

Add Custom Label Details 744
Interaction and Animation 745
Interaction 745
Animations 746
Gradient 747
Distance From Corner 747
Resize 747
Localization 748
Themes 748
Circular Gauge Values 748
Scales 749
Adding Scale Collection 749
Scale Customization 749
Multiple Scales 753
Pointers 754
Adding Pointer Collection 754
Adding Pointer Value 755
Pointer Styles 756
Appearance 757
Pointer Value 764
Appearance 765
Font Options 767
Labels 768
Adding Label Collection 768
Label Customization 769
Multiple Labels 771
Ticks 772
Adding Tick Collection 772
Tick Customization 773
Indicators 774
Adding Indicator Collection 774
Basic Customization 775
State Ranges 776
Multiple Indicators 778
Ranges and Frames 779

Add Range Collection 779
Multiple Ranges 783
Frames 784
Legend 785
Legend Visibility 785
Position and Align the Legend 786
Customization 788
Events 794
Custom labels 795
Adding Custom Label Collection 795
Multiple Custom Labels 796
Outer Custom Label 797
Tooltip 798
Default Tooltip 799
Tooltip Template 800
Sub Gauges 801
Adding SubGauges 801
Multiple SubGauges 802
Gauge Position 804
Positioning 804
Exporting 806
Methods and Events 806
Public Methods 806
Events 825
MVVM 829
AngularJS 829
Rendering the Circular Gauge 829
Adding Scale Collection 830
Adding Pointer Collection 831
Adding Label Collection 832
Adding Tick Collection 833
Adding Range Collection 834
Two Way Binding 835
KnockoutJS 837
ColorPicker 838

Overview 838
Getting Started 838
Create your first ColorPicker in ASP.NET MVC 838
Behavior Settings 843
ShowPreview 843
showRecentColors 844
EnableOpacity 844
Columns 845
Configure Values 846
OpacityValue 846
Button and TooltipText 846
Appearance and Styling 848
modelType 848
Palette 849
BasicPalette 849
PresetType 849
CustomPalette 850
DisplayInline 851
Theme Support 852
CustomCss 853
KnockoutJS Binding 854
AngularJS Binding 856
Keyboard Interaction 858
Configure Keyboard Interaction 859
Miscellaneous 859
getValue 859
setValue 860
getColor 860
How to? 861
Render ColorPicker from Code behind 861
ComboBox 861
Overview 861
Key features 862
Getting Started 862
Initialize the ComboBox 862

Binding data source 863
Custom values 864
Configure the popup list 865
ComboBox Dependencies 866
Data Binding 866
Binding local data 867
Binding remote data 869
SQL Data Binding 869
LINQ Data Binding 870
XmlDataSource 871
EntityDataSource 872
AccessDataSource 873
LINQ to SQL Data Binding 874
Remote data 876
OData Version 4 876
WebAPI 876
Templates 877
Item template 877
Group template 879
Header template 880
Footer template 882
No records template 884
Action failure template 885
Grouping 885
Customization 887
Sorting 887
Filtering 888
Limit the minimum filter character 889
Change the filter type 890
Case sensitive filtering 892
Localization 893
Loading translations 893
Accessibility 894
ARIA attributes 894
Keyboard interaction 895

How To 897
Configure the Cascading ComboBox 897
Show the list items with icons 899
AutoFill supported with ComboBox 901
Validation of ComboBox using jQuery Validator 902
Render ComboBox from code behind 903
CurrencyTextBox 904
Overview 904
Key Features 904
Getting Started 904
Create your first CurrencyTextBox in MVC 904
Behavior Settings 910
Decimal Places 910
Persistence Support 910
Strict Mode Support 911
Enabled or Disabled 911
Adjusting Textbox Size 912
Increment Step 912
Define Name 912
Define Value 913
Define MaxValue and MinValue 913
Read Only Support 913
Appearance 914
Rounded Corner Support 915
Spin Button Support 915
Water Mark Text Support 916
Globalization Support 916
Configure Localization 917
RTL Support 917
Enable RTL 917
Keyboard Interaction 917
Configuring Keyboard Navigation 918
DataManager 918
DataManager 918
DataManager supports different types of data binding such as: 918

Key Features ... 918
Getting Started ... 919
Create your DataManager in the ASP.NET MVC ... 919
Create Connection ... 919
Binding with Grid Control ... 920
Filter ... 920
Sort ... 921
Page ... 922
Data Binding ... 923
Local ... 923
REST Services ... 924
OData binding ... 924
OData V4 ... 925
WebAPI binding ... 926
Other Web Services binding ... 927
Complex Data Sorting ... 929
Data Adaptors ... 930
JSON Adaptor ... 930
URL Adaptor ... 932
OData Adaptor ... 935
WebAPI Adaptor ... 937
RemoteSave Adaptor ... 939
Custom Adaptor ... 942
Cache Adaptor ... 943
Query ... 947
Select ... 947
From ... 947
Clone ... 948
Expand ... 949
Sorting ... 950
Default ... 950
SortByDesc ... 951
Dynamic sorting ... 951
Multi sorting ... 952
Searching ... 953

Filtering 954
Filter Operators 954
lessThan 954
greaterThan 955
lessThanOrEqual 956
greaterThanOrEqual 956
equal 957
notEqual 958
contains 958
startswith 959
endswith 960
and predicate 960
or predicate 961
Containment Operators 962
Lambda Operators 965
Grouping 967
CRUD Data Operations 968
Local CRUD Operations 968
Remote CRUD Operations 973
Batch Edit 979
Paging 980
Default 980
Dynamic Paging 981
Custom paging 982
Skip 983
Take 984
RequiresCount 985
Range 985
Summary 986
Sum 986
Min 987
Max 988
Avg 989
Distinct 990
Hierarchical Query 991

foreignKey 991
Helper Functions 992
KO ResultSet 994
Table Model 996
CORS Data Fetching 998
SAME-ORIGIN POLICY 998
Access-Control-Allow-Origin 998
Access-Control-Request-Method 999
Access-Control-Request-Headers 999
Advanced Functionalities 999
Offline Support 999
Load on demand 1000
Custom Request Headers 1002
Cross domain & JSONP 1005
HTML Table 1006
WebAPI Batch Edit 1007
Configuration 1007
IEnumerable 1012
Data Binding 1012
Filtering Sorting and Searching with IEnumerable 1013
IQueryable 1014
IQueryable support in DataManager 1014
Anti-forgery Token 1015
DatePicker 1017
Overview 1017
Key features 1017
Getting Started 1018
A new MVC application and required assemblies with dependent files 1018
DatePicker Initialization 1019
Get / Set value 1019
Client Side Events 1020
Strongly Typed HTML Helper 1020
Behavior Settings 1020
Selected Date 1020
Date Range 1021

Start and Depth Level 1021
Display Inline Mode 1021
Strict Mode 1022
Formatting 1022
Date Format 1023
Header Format 1024
Day Header 1025
Tooltip with Formatting 1025
Globalization 1026
Watermark and Today Button Text 1027
Customization 1027
Set Dimension 1027
Show Footer 1028
Show Popup Button 1028
Show Other Months 1028
Highlight Special Date 1028
Validation 1029
State Persistence 1030
Miscellaneous 1030
Start Day 1030
Step Months 1030
Read Only 1031
Enable or Disable 1031
How to? 1031
Customizing template with range selection between two DatePicker 1031
Validating the date value in server side 1031
Localize DatePicker with browser specific culture 1031
Disable specific dates to restrict user 1031
How to add clear button with DatePicker? 1032
How to integrate with bootstrap grid system? 1032
Render Datepicker from Code behind 1033
DateRangePicker 1033
DateRangePicker 1033
Key Features 1033
Getting Started 1034

Create your first DateRangePicker in MVC 1034
Create a DateRangePicker 1034
Start Date 1035
End Date 1036
Behavior-Settings 1036
Value 1036
Separator 1037
Localization 1038
Customization 1040
Setting Dimension 1040
Rounded Corners 1040
Persistence 1040
Preset Ranges 1040
TimePicker Option 1041
Allow Editing 1042
Enable/Disable 1042
Clear Ranges 1042
How to? 1043
Render DateRangePicker from Code behind 1043
DateTimePicker 1044
DateTimePicker 1044
Key Features 1044
Getting Started 1044
Create your first DateTimePicker in MVC 1044
Create a DateTimePicker 1045
Set the Min/Max Date and Time Intervals 1045
Display format 1046
DateTime format 1046
Day Header Format 1047
Date Range 1048
Start and Depth level 1050
Start Level 1050
Depth Level 1051
Date in other months 1051
Time Interval 1052

Localization 1053
Right-to-Left 1055
Appearance and Styling 1056
Theme 1056
CSS Class 1057
How to? 1058
Render DateTimePicker from Code behind 1058
Diagram 1059
Diagram 1059
Getting Started 1060
Flow Diagram 1060
Automatic organization chart 1064
Model 1066
Node 1066
Create Node 1067
Update Node at runtime 1068
Position 1069
Types 1069
Appearance 1069
Shadow 1071
Interaction 1072
Constraints 1073
Connector 1073
Create Connector 1073
Update Connector at runtime 1074
Connect nodes 1075
Segments 1077
Decorator 1083
Padding 1084
Bridging 1087
Corner radius 1089
Appearance 1090
Interaction 1091
Constraints 1091
Port 1091

Create Port 1092
Connect with ports 1093
Appearance 1093
Constraints 1094
Label 1094
Create Label 1094
Update Label at runtime 1095
Alignment 1096
Wrapping 1098
Appearance 1099
Drag 1100
Rotate 1101
Edit 1101
Multiple labels 1103
Limitation 1104
Group 1104
Create Group 1104
Container 1105
Difference between a basic group and containers 1108
Interaction 1109
Swim lane 1109
Initialize the Diagram model 1109
Create a swimlane 1109
Add swimlane into diagram 1109
Headers 1110
Lane 1111
Phase 1114
Limitations 1116
Gridlines 1116
Customize the gridlines visibility 1116
Appearance 1117
Snapping 1119
Snap To Lines 1119
Customization of Snap Intervals 1119
Snap To Objects 1120

Page Settings 1121
Page size and appearance 1121
MultiplePage and PageBreaks 1123
Boundary Constraints 1124
Rulers 1125
Adding Rulers to the Diagram 1125
Customizing the Ruler 1125
Interaction 1126
Selection 1126
Drag 1128
Resize 1128
Rotate 1128
Connection editing 1128
Drag and drop nodes over other elements 1129
User handles 1130
Zoom pan 1131
Keyboard 1132
History Manager 1132
Undo and Redo 1132
Track custom changes 1133
Group multiple changes 1134
Serialization 1134
Save 1134
Load 1135
Limitation 1135
Commands 1136
Align 1136
Space 1138
Sizing 1138
Clipboard 1139
Grouping 1139
Z-Order Command 1139
Zoom 1141
Nudge Command 1141
BringIntoView 1142

BringToCenter 1142
FitToPage command 1143
Command Manager 1143
Data Binding 1145
Local Data 1145
Remote Data 1147
HTML Table Data 1148
CRUD 1151
Read DataSource 1151
How to perform Editing at runtime 1152
InsertData 1152
UpdateData 1153
DeleteData 1154
Automatic Layout 1155
Hierarchical Layout 1155
Radial Tree Layout 1157
Organizational Chart 1159
Customize Layout 1167
Overview Control 1171
Create overview 1171
Zoom and Pan 1172
Symbol Palette 1172
Create symbol palette 1172
Add palettes to SymbolPalette 1173
Add symbols to palette 1175
Symbol Previews 1178
Show/hide the symbol Text 1181
Exporting 1182
Exporting options 1182
Localization 1184
Tools 1186
Drawing Tools 1186
Tool Selection 1188
Tooltip 1189
Default tooltip 1189

Common tooltip for all nodes and connectors 1190
Tooltip for a specific node/connector 1191
Tooltip alignments 1191
Constraints 1193
DiagramConstraints 1194
NodeConstraints 1194
ConnectorConstraints 1195
PortConstraints 1195
SelectorConstraints 1196
SnapConstraints 1196
Inherit behaviors 1197
Bitwise Operations 1197
Scroll Settings 1198
Get current scroll status 1198
Define scroll status 1198
Update scroll status 1199
AutoScroll 1199
Autoscroll border 1200
Scroll limit 1200
Scrollable Area 1200
Context Menu 1201
Default Context Menu 1201
Customize Context Menu 1202
Context Menu Events 1202
How To 1203
Bitwise Operations 1203
Dialog 1204
Overview 1204
Key Features 1204
Getting Started 1204
Create a Dialog 1204
Load content 1207
Action Buttons 1208
Customizing Action Buttons 1209
Animation 1209

Keyboard interaction 1210
How To? 1211
Create Multiple Dialogs 1211
Create Nested Dialog 1212
Create a Confirmation Dialog with Footer section 1213
Render Dialog from Code behind 1214
DigitalGauge 1215
DigitalGauge 1215
Key Features 1216
Getting Started 1216
Create a Digital Gauge 1216
Set Height and Width values 1217
Set Items Property 1217
Add Background Image 1218
Add Location 1219
Add Items Collection 1219
Basic Settings 1220
Height and Width Customization 1220
Responsive Layout 1221
Themes 1221
Setting Text for Digital Gauge 1222
Frames 1222
Inner and Outer Customization 1222
Setting Background Image 1223
Digital Elements 1223
Text Customization 1223
Segment Settings 1224
Appearance 1224
Dimension Modification 1225
Character Settings 1225
Appearance 1225
Count and Type 1226
Text Positioning 1227
Shadow Effects 1228
Font Customization 1228

Multiple Items 1229
Exporting the Digital Gauge 1230
MVVM 1230
AngularJS 1230
Rendering the Digital Gauge 1230
Adding the Digital Gauge Items 1231
Two Way Binding 1231
KnockoutJS Binding 1232
Methods 1233
Events 1235
DropDownList 1237
DropDownList 1237
Key Features 1237
Getting Started 1237
Creating your first DropDownList in MVC application 1237
Populating data 1238
Setting Dimensions 1238
Setting and Getting Value 1240
DropDownList Dependencies 1240
Rendering Mode 1241
Target Rendering 1241
Data Binding 1243
Fields 1243
Local Data 1244
SQL Data Binding 1245
LINQ Data Binding 1246
XmlDataSource 1247
EntityDataSource 1248
AccessDataSource 1249
LINQ to SQL Data Binding 1250
Remote data 1252
OData Version 4 1252
WebAPI 1253
Virtual Scrolling 1253
Checkbox 1254

Selection Modes 1256
Select multiple values from controller 1258
Functionalities 1259
Selection 1259
Grouping 1265
Sorting 1267
Cascading 1268
Validations 1273
Using jQuery Validator 1273
Client Side Validation 1274
Server side Validation 1276
Template Support 1277
Header Template 1277
Template Field 1277
Customization 1279
Adding watermark text 1279
Applying Rounded Corner 1280
Enable/Disable the control 1281
Applying HTML Attributes 1282
Setting dimensions 1282
Control Sizing 1282
Popup resizing 1284
Search 1285
Incremental Search 1285
Filter search 1286
Server Filtering 1287
State Persistence 1288
Accessing currently stored state 1289
Strongly Typed HTML Helper 1289
Localization 1292
Load On Demand 1293
Accessibility 1293
Keyboard Navigation 1293
How To 1296
Add check all option in popup list? 1296

Render the DropDownList in Partial View using normal mode ... 1297
Render the DropDownList in Partial View using Unobtrusive mode ... 1298
Dynamically bind the data to ejDropDownList on AJAX call ... 1299
Remove validation error message on item selection ... 1300
Render DropDownList from code behind ... 1301
FileExplorer ... 1301
Overview ... 1301
Key features ... 1302
Getting Started ... 1302
A new MVC Application and required assemblies with dependent files ... 1302
Adding the references ... 1302
FileExplorer using helper ... 1304
Behavior Settings ... 1305
File type restriction ... 1305
Customize the AJAX request settings ... 1306
User Interface ... 1306
File Actions ... 1307
File action through Toolbar ... 1307
File action through Context menu ... 1307
Access Control ... 1308
FileAccessOperations ... 1309
Toolbar ... 1318
Toolbar items ... 1318
Toolbar Visibility ... 1321
Toolbar Configuration ... 1321
Search bar ... 1322
Custom Tool in Toolbar ... 1322
Enable / Disable the Toolbar Item ... 1323
Customizing the Upload Functionality ... 1324
Context Menu ... 1325
Context menu items ... 1326
Context menu Visibility ... 1327
Enable / Disable the Context menu Item ... 1327
Context Menu Customization ... 1328
Context Menu Events ... 1330

Customization 1331
Dimension Customization 1331
Customizing the Navigation pane 1331
Customizing the Content pane 1331
Footer Customization 1333
Multiple Selection 1334
Checkbox option 1334
Resizing 1334
Responsiveness 1335
Restriction on Resize 1335
Drag and Drop Support 1335
Thumbnail Image Compression Support 1336
Access shared folder using FileExplorer 1337
Localization 1338
Change the Culture 1340
Keyboard Navigation 1343
Virtual Scrolling 1344
How To 1345
File handling operations 1345
Customizing AJAX handling functions 1345
Managing files that is available in SQL database 1354
Cross-origin resource sharing support for FileExplorer 1355
Access static physical path using FileExplorer control 1358
Adding Custom column in grid view of FileExplorer 1359
Create FileExplorer in older version 1361
Customize the Root Folder name in FileExplorer 1364
Use the FTP connection using FileExplorer 1364
Gantt 1366
Overview 1366
Getting Started 1367
Create your first Gantt in MVC 1367
Enable Toolbar 1371
Enable sorting 1372
Enable Editing 1372
Enable Context Menu 1373

Enable Column Menu 1374
Provide tasks relationship 1375
Provide Resources 1375
Highlight Weekend 1377
Define dimension of Gantt 1377
Add notes in tasks 1378
Milestones in Gantt 1380
Data Binding 1382
Local Data Binding 1382
Self-Referential Data Binding (Flat Data) 1384
Remote data 1387
Virtualization mode in Gantt 1389
Resource Allocation View 1390
Data Binding 1390
Grouping 1393
CRUD data operations 1395
Unassigned Tasks and Resources 1400
Resource Overallocation 1402
Resource Histogram 1402
Databinding 1402
Synchronize resource histogram view Gantt with other views 1403
Editing 1408
Cell Editing 1408
Normal Editing 1410
Taskbar Editing 1414
Dependency Editing 1415
Update Gantt record value dynamically 1417
Delete confirmation message 1418
Task Indent 1419
Task Outdent 1421
Task Delete 1422
Read-only Gantt 1423
Timescale customization 1423
Schedule Header Types 1424
Week start day customization 1428

Rounding off timescale (schedule) start date 1428
Customize automatic timescale update action 1429
Critical Path 1431
Using Toolbar Icon 1431
Using Method 1432
Customizing critical path background 1433
Gantt Columns 1434
Column Edit Types 1434
Format Column 1434
Column Resizing 1435
Column Template 1435
Column Menu 1437
Change visibility of the columns dynamically 1441
Change Tree/Expander Column 1441
Sequencing Tasks 1442
How to enable serial number column in Gantt? 1442
Rows 1443
Adding a row 1443
Row drag and drop 1446
Alternate row background 1449
Row height 1450
Expand/Collapse Row 1451
Task Dependencies 1454
Task Relationship Types 1455
Define task relationship 1455
Predecessor offset with duration units 1457
Enable/disable predecessor validation 1459
Validate predecessor links on editing 1460
Time Options 1462
Validating Schedule Dates 1465
Task Scheduling Modes 1466
Automatically Scheduled Tasks 1466
Manually Scheduled Tasks 1467
Custom 1467
Unscheduled Tasks 1469

Unscheduled Task Types 1469
Define unscheduled tasks in data source 1469
Show/hide null text in Gantt columns 1471
Context Menu 1472
Default Menu Items 1472
Custom Menu Item 1472
Customizations 1474
Taskbar template 1475
Task label template 1476
Tooltip template 1478
Selection 1484
Row selection 1484
Cell selection 1487
MultiSelection – Touch Option 1489
Export 1490
Server dependencies 1492
Supported Export Types 1492
Export Mapper in MVC 1492
Multiple exporting 1493
Export Theme 1494
Customization 1495
Sorting 1500
Sorting Columns 1500
Multicolumn sorting 1500
Sorting column on Gantt initialization 1501
Sorting column dynamically 1502
Sorting multiple columns dynamically 1503
Searching 1504
Filtering 1505
Filter columns at initial load 1505
Filtering a specific column by public method 1506
Clearing the filter applied to Gantt 1506
Toolbar 1507
Default Toolbar Items 1507
Custom Toolbar Items 1508

Baseline 1509
Work Breakdown Structure 1510
Resources 1511
Resource Collection 1511
Assign Resource 1512
Assign Resources with Unit 1513
Edit Resource Collection 1514
Work 1517
Task type 1518
Effort driven tasks 1518
Stripline 1519
Holidays 1520
Localization 1521
Date format 1526
Keyboard navigation 1527
Tooltip 1527
Taskbar and dependency line tooltip 1527
Taskbar drag tooltip 1529
Cell tooltip 1529
Tree column (Expander column) tooltip 1530
Duration Units 1530
Mapping the Duration Unit field 1531
Defining duration unit along with duration field 1532
Localizing the duration unit 1533
Working Time Range 1534
Highlight working time range 1534
Appearance and Styling 1535
Schedule Header Customization 1535
Taskbar Customization 1537
Dependency line customization 1541
Weekend background 1542
Themes 1542
Configuring CSS Class 1543
Customize rows and cells 1544
How to 1545

Destroy Gantt Control ... 1548
Display custom fields in general tab ... 1548
Change week start day in month timescale mode ... 1550
Rendering the Gantt control using GanttProperties in server-side ... 1552
Update task id value ... 1553
Get taskbar information on click action ... 1554
Grid ... 1555
Grid ... 1555
Key features ... 1555
Getting Started ... 1555
Create your first grid in MVC ... 1556
Data binding ... 1557
Enable paging ... 1558
Enable filtering ... 1560
Enable grouping ... 1561
Add summaries ... 1563
Grid Dependencies ... 1565
Data Binding ... 1566
IEnumerable ... 1566
DataTable ... 1567
Entity Framework ... 1569
LINQ to SQL ... 1570
ITypedList Binding ... 1572
Complex Binding ... 1574
WCF DataService / OData Service ... 1575
ODataV4 Service ... 1576
Web API Service ... 1577
HTML Table binding ... 1578
Miscellaneous ... 1580
Data Adaptors ... 1586
JSON Adaptor ... 1586
Url Adaptor ... 1587
OData Adaptor ... 1589
ODataV4 Adaptor ... 1590
WebAPI Adaptor ... 1591

RemoteSave Adaptor 1592
Foreign Key Adaptor 1595
Custom Adaptor 1596
Data Annotation 1597
Columns 1600
Auto Generation 1600
Headers 1603
Text alignment 1607
Format 1608
Width 1609
Resizing 1610
Resize to fit 1611
Reorder 1613
Visibility 1614
Unbound Column 1615
Column Template 1617
Controlling Grid actions 1618
Read only 1619
Expression Column 1620
Command Column 1621
Column Chooser 1625
Foreign Key Column 1626
Customize column 1628
Type 1630
Column Layout 1632
Row 1633
Row hover 1633
Details template 1635
Row Template 1637
Alternate row styling 1639
Drag-and-Drop 1641
Cell 1649
Auto wrap 1649
Both 1649
Cell Merging 1653

Custom Attribute 1655
Displaying HTML content 1656
Tooltip 1657
ClipMode 1659
Sorting 1663
Initial sorting 1665
Multi-column sorting 1666
Stable sorting 1668
Touch options 1670
Filtering 1671
Menu filter 1673
Excel-like filter 1676
Filter bar 1682
Filter Operators 1686
FilterBar template 1687
Editing 1689
Toolbar with edit option 1690
Cell edit type and its edit options 1691
Cell Edit Template 1693
Edit Modes 1695
Confirmation messages 1718
Column Validation 1722
Persisting data in Server 1726
Adding New Row Position 1739
Render with blank row for easy add new 1740
Default column values on add new 1742
Grouping 1744
Initial Grouping 1745
Multi-Column Grouping 1747
Group Buttons 1749
Hide Ungroup Button 1750
Hide Grouped Column 1752
AutoSize Drop Area 1753
Hide Drop Area 1755
Group Caption Format/Group Caption Template 1756

Handling grouped records count in server-side 1758
Summary 1760
Supported aggregates 1762
Custom summary 1766
Group summary 1768
Group caption summary 1770
Summary template 1772
Format 1773
Handling aggregation in server-side 1775
Paging 1777
Pager with query string 1778
Pager template 1779
Pager with pageSize drop down 1781
Pager with pageSettings 1782
Scrolling 1783
Set width and height in pixel 1785
Set width and height in percentage 1786
Set width as auto 1787
Frozen columns 1788
Frozen rows 1792
Touch scroll 1793
Virtual scrolling 1794
Responsive 1798
Mobile layout 1799
Tablet layout 1808
Width 1810
Min width 1811
Priority for columns 1812
Selection 1812
Types of selection 1812
Row selection 1813
Multiple row selection using checkbox column 1814
Cell selection 1815
Column selection 1818
Touch options 1819

Toggle selection 1821
Toolbar 1822
Default toolbar items 1822
Custom toolbar items 1824
Web Accessibility 1826
Keyboard Navigation 1826
WAI - ARIA 1827
Styling 1828
List of classes and its purposes 1828
Toolbar customization 1829
Context Menu 1829
Default Context Menu items 1829
Custom Context Menu 1833
Sub Context Menu 1835
Sub Context Menu with Template 1837
Hierarchical Bindings 1839
Expand or collapse all child's 1841
Stacked Headers 1842
Adding stacked header columns 1842
Exporting 1844
Hierarchy grid exporting 1846
Server dependencies 1848
Support export types 1848
Server side handlers 1848
Export mapper in MVC 1850
ColumnTemplate exporting 1851
DetailTemplate exporting 1854
Exporting with Custom Summary 1857
Multiple exporting 1859
DataTable exporting 1862
Export required grid in single file 1863
List of properties ignored on export 1863
Export only visible records 1864
Local data 1864
Themes 1865

AutoFormat class ... 1867
Exporting server events ... 1870
Print ... 1872
Page setup ... 1873
Print on external button click ... 1874
Print visible page ... 1876
State Persistence ... 1877
List of properties are not persisted by default ... 1877
Accessing currently stored state ... 1878
Globalization and localization ... 1878
Localization ... 1878
Pager Localization ... 1881
Excel-filter Localization ... 1883
Globalization ... 1887
Right to left - RTL ... 1888
How to ... 1890
Apply formatting for columns dynamically ... 1890
Disable double-click edit ... 1890
Customize drag and drop element while grouping ... 1891
Display tooltip ... 1893
Set dynamic dataSource to grid ... 1893
Custom binding for grid CRUD operation ... 1895
Binding SignalR endpoint ... 1897
Copy data from Excel to grid ... 1899
Use NHibernate with grid ... 1899
Adding comments in the exported file (Excel, Word or PDF) ... 1902
Adding header and footer in the exported file (Excel, Word or PDF) ... 1906
External search in grid ... 1909
Configure grid through the grid model properties ... 1911
Work with partial views ... 1912
Rendering the grid using grid properties in server-side ... 1914
Export grid to target location without download prompt(Excel, Word or PDF) ... 1916
Hierarchy grid with different foreignKeyField in parent and child table ... 1917
Display other Syncfusion controls in grid columns ... 1918
Perform grid actions on external button click ... 1920

Getting Datasource of Grid in Sorted Order 1926
GroupButton 1927
Overview 1927
Key Features 1927
Getting Started 1928
Dependencies 1928
Create a simple GroupButton control 1928
Behavior settings 1928
Groupbutton Mode 1928
Easy Customization 1929
Button Size 1929
Orientation 1930
DataSource 1931
Local data 1931
RTL Support 1932
Miscellaneous 1933
Text 1933
Show Rounded Corner 1933
How to? 1933
Render GroupButton from Code behind 1933
HeatMap 1934
Overview 1934
Key features: 1935
Getting Started 1935
Initialize the HeatMap 1935
Prepare and Populate data 1935
Map data into HeatMap 1936
Color Mapping 1937
Legend 1938
Items Mapping 1940
Color Mapping 1942
Legend 1942
Create Legend 1943
Legend Mode 1943
Orientation 1944

Kanban Board 1945
Overview 1945
Getting Started 1945
Create your first Kanban in MVC 1945
Data Binding 1946
Mapping Values 1948
Enable Swimlane 1950
Adding Filters 1952
Columns 1954
Key Mapping 1954
Multiple Key Mapping 1956
Headers 1957
Width 1958
Visibility 1960
Toggle 1961
Allow Dragging 1962
Allow Dropping 1963
Items Count 1964
Customize Items Count Text 1965
Display Empty Column 1966
Data Binding 1968
IEnumerable 1968
Entity Framework 1970
LINQ to SQL 1971
WCF DataService / OData Service 1972
ODataV4 Service 1973
WebAPI 1974
URL Adaptor 1975
Workflows 1977
Swim lanes 1978
Customized swimlane header text 1980
Empty swimlane row on Kanban board 1982
Drag And Drop between swim lanes 1984
Unassigned swim lane group 1985
Drag and Drop 1989

Drag and Drop to external control 1989
Cards 1995
Customization 1995
Template 1997
Tooltip 1999
Collapsible Cards 2003
Prioritization of cards 2004
Editing 2006
Configuring Edit Items 2006
Edit modes 2007
Column Validation 2020
Persisting data in server 2023
Filtering 2028
Scrolling 2029
Set width and height in pixel 2030
Set height and width in percentage 2031
Set width as auto 2032
Enabling freeze swim lane 2033
Selection & Hovering 2035
Types of Selection 2035
Responsive 2037
Mobile Layout 2038
Width 2045
Min Width 2046
Stacked Headers 2047
Adding Stacked header columns 2048
Context Menu 2049
Default Context Menu items 2049
Custom Context Menu 2051
Sub Context Menu 2053
Print 2054
Localization 2056
Right to Left (RTL) 2059
Styling 2061
List of classes and its purposes 2061

Web Accessibility ... 2062
Keyboard Navigation ... 2062
How to ... 2063
Render Kanban from Code behind ... 2063
LinearGauge ... 2064
Overview ... 2064
Getting Started ... 2065
Interaction and Animation ... 2077
Scales ... 2078
Adding scale collection ... 2079
Scale Customization ... 2080
Scale Types ... 2083
Thermometer ... 2085
Marker Pointers ... 2087
Adding marker pointer collection ... 2088
Add marker pointer value ... 2088
Pointer Styles ... 2089
Positioning the pointer ... 2090
Types ... 2091
Bar Pointers ... 2093
Adding bar pointer collection ... 2093
Adding bar pointer value ... 2094
Pointer Styles ... 2095
Positioning the pointer ... 2096
Multiple Bar Pointers ... 2097
Labels ... 2098
Adding label collection ... 2098
Label Customization ... 2100
Unit text and Positioning ... 2101
Ticks ... 2102
Tick Customization ... 2103
Ranges ... 2107
Adding range collection ... 2107
Range Customization ... 2108
Positioning the ranges ... 2110

Multiple Ranges ... 2111
Custom labels ... 2112
Adding Custom label collection ... 2112
Basic Customization ... 2114
Locating the Custom Labels ... 2115
Multiple Custom Labels ... 2116
Indicators ... 2118
Setting Dimension ... 2118
State Ranges ... 2119
Font options ... 2122
Multiple Indicator ... 2124
Exporting ... 2126
MVVM ... 2128
Rendering the Linear gauge ... 2128
Adding Scale collection ... 2128
Adding Marker Pointer collection ... 2129
Adding label collection ... 2130
Adding Tick collection ... 2131
Adding Range collection ... 2132
Two Way Binding ... 2133
KnockoutJS ... 2134
Events ... 2135
Methods ... 2138
ListBox ... 2152
ListBox ... 2152
Key Features ... 2152
Getting Started ... 2153
Create your first ListBox in MVC ... 2153
Data-binding ... 2161
Data fields and configuration ... 2161
Local data ... 2162
Remote data ... 2163
Behavior Settings ... 2164
Target ID ... 2164
Select the value by index ... 2165

Enable or Disable the ListBox Widget 2166
Persistence support 2167
Multiple Selection 2168
Allow multiple selection 2168
Multiple selection through index 2169
Checkbox Support 2170
Appearance and Styling 2173
Adjusting ListBox size 2173
Sorting 2176
Theme 2177
Custom class with ListBox 2178
Tooltip Support 2180
Cascading Support 2181
VirtualScrolling support 2184
Drag and Drop Support 2185
Template Support 2187
RTL Support 2189
Keyboard interaction 2190
How to 2192
Render ListBox from Code behind 2192
ListView 2193
ListView 2193
Key Features 2193
Getting Started 2193
Create your first ListView in MVC 2193
Create a simple ListView 2194
Add header 2195
Grouped List 2196
First Level Group List 2196
Nested Child Group List 2197
Selection 2198
MultiSelection 2198
PreventSelection 2199
PersistSelection 2200
Customize Header 2200

Data Binding 2201
Local Data Binding 2201
Remote Data Binding 2203
FieldSettings 2207
Filtering 2208
Dimensions 2209
Virtual Scrolling 2210
Normal Mode 2210
Continuous Mode 2210
How To 2211
Get the selected values on post back. 2211
Render ListView from Code behind 2212
Maps 2213
Maps 2213
Use Case Scenarios 2213
Key Features 2213
Getting Started 2213
Create your first Map in MVC 2213
Initialize Map 2217
Data Binding in Map 2218
Enable Tooltip 2221
Legend 2222
Populate Data 2224
Shape Data 2224
Data Binding 2225
Customization 2228
Shapes Color Customization 2228
Color Mapping 2229
Binding color from the datasource 2234
ColorPalette 2236
Tooltip 2238
Map Elements 2241
Markers 2241
Bubbles 2244
Add bubbles to the map 2246

Legend ... 2248
User Interaction ... 2252
Map Selection ... 2252
MultiSelection ... 2253
Dragging On Selection ... 2254
Zooming ... 2255
Navigation Control ... 2258
Layers ... 2261
Multilayer ... 2261
SubLayer ... 2261
Map Providers ... 2263
Open Street Map ... 2263
Bing Map ... 2264
AngularJS Support ... 2265
RTL ... 2266
Interactive legend ... 2267
Tooltip ... 2267
Methods ... 2268
Events ... 2269
MaskEdit ... 2274
MaskEdit ... 2274
Key Features ... 2274
Getting Started ... 2274
Create your first MaskEdit Widget in MVC ... 2274
Set Mask value to Mobile Number textbox ... 2276
Create a MaskEdit control for Product Key Validation ... 2277
Set Mask value to Product key textbox ... 2278
Behavior Settings ... 2278
Persistence Support ... 2278
Enabled or Disabled ... 2279
Adjusting Textbox Size ... 2279
Define Value ... 2279
Read Only Support ... 2280
Error Visibility ... 2280
Mask Edit Properties ... 2281

Appearance 2282
Globalization Support in MaskEdit 2285
Menu 2286
Menu 2286
Key Features 2286
Getting Started 2286
Create Syncfusion Menu in MVC 2286
Create a Menu 2287
Configure Parent Menu items 2287
Initialize sub-level Menu items 2288
Define multiple level Menu items 2290
Orientation 2292
Horizontal Menu 2292
Vertical Menu 2293
Data binding 2294
Field Members 2294
Local data 2295
Remote data 2296
Icons and navigation 2297
Navigation 2300
Customizing the Submenu direction 2301
Look and feel 2305
CssClass 2305
Background Template 2306
Context Menu 2310
HideContextMenu 2311
ShowContextMenu 2312
Center Menu 2312
RTL Support 2313
Separators 2314
Responsive Layout 2316
Responsive in Mobile or Tablet: 2317
Keyboard Navigation 2318
Miscellaneous 2320
Height 2320

Width 2320
Open on click 2321
Animation 2322
Title text 2323
Show root level arrows 2325
Show sub level arrows 2326
How To 2327
Render Menu from code behind 2327
NavigationDrawer 2328
Overview 2328
Key Features 2328
Getting Started 2328
Create your first Navigation Drawer control in MVC 2328
Customize Direction 2334
Customize Position 2335
TargetId 2336
Animations 2337
How To 2339
Render NavigationDrawer from Code behind 2339
NumericTextBox 2341
NumericTextBox 2341
Key Features 2341
Getting Started 2341
Create your first NumericTextBox in MVC 2341
Create Editors 2342
Set MinValue, MaxValue and value in NumericTextBox 2344
Setting the Strict Mode Option 2345
Set Calculation process with NumericTextBox Widgets 2346
Behavior Settings 2347
Decimal Places 2347
Persistence Support 2348
Strict Mode Support 2348
Enabled or Disabled 2349
Adjusting Textbox Size 2349
Increment Step 2349

Define Name ... 2350
Define Value ... 2350
Define MaxValue and MinValue ... 2350
Read Only Support ... 2351
Appearance ... 2351
Water Mark Text Support ... 2353
Globalization Support ... 2353
Configure Localization ... 2353
RTL Support ... 2354
Enable RTL ... 2354
Keyboard Interaction ... 2354
Configuring Keyboard Navigation ... 2354
How To ... 2355
Use Normal Textboxes as Syncfusion textboxes ... 2355
Pager ... 2355
Overview ... 2355
Getting Started ... 2356
Adding scroller control in your MVC application ... 2356
Behavior Settings ... 2357
Page Size List ... 2357
Page Size Message ... 2357
Responsive ... 2358
Current Page ... 2358
Enable/Disable ... 2358
Customization ... 2358
Show Go To Page ... 2358
Show Page Info ... 2359
Custom Text ... 2359
External Message ... 2359
Custom CSS ... 2360
Localization ... 2361
Setting different culture to control ... 2361
How to? ... 2361
Render Pager from Code behind ... 2361
PivotChart ... 2362

Overview 2362
Key Features 2362
Getting Started 2362
Getting Started 2362
Getting Started 2376
Data Binding 2386
Data Binding 2386
DataBinding 2392
Dimensions 2398
Set size in percentage 2398
Set size in pixels 2399
Responsive 2399
Chart Types 2400
Column Chart 2401
Stacking Column Chart 2402
Bar Chart 2402
Stacking Bar Chart 2403
Pie Chart 2404
Pyramid Chart 2405
Funnel Chart 2406
Line Chart 2407
Step Line Chart 2408
Spline Chart 2409
Area Chart 2410
Step Area Chart 2411
Spline Area Chart 2412
Stacking Area Chart 2413
Doughnut Chart 2414
Scatter Chart 2415
Bubble Chart 2415
WaterFall chart 2416
Combination Chart 2417
Drill Operation 2417
Legend 2418
Legend Visibility 2418

Legend Shape 2419
Legend Position 2420
Legend Title 2421
Legend Alignment 2422
Legend Items - Size and Border 2422
Legend Border 2423
Legend Text 2424
Axes 2425
Label Format 2425
Common Axis Features 2427
Smart Axis Labels 2434
Title 2435
Title Text 2435
Title Alignment 2436
Title Customization 2437
Series 2438
Series Point customization 2438
Multiple Axes 2439
Customizing axes at row index of zero 2440
Customizing axes at row index one 2440
Customizing axes at column index of zero 2441
Customizing axes at column index of one 2442
Customizing series 2443
Multiple axes support by series index 2443
Customizing PrimaryYAxis and axes properties 2445
Multi-level Labels 2446
Relational 2446
OLAP 2447
3D Visualization 2447
3D Column Chart 2447
3D Bar Chart 2448
3D Stacking Bar Chart 2449
3D Stacking Column Chart 2450
3D Pie Chart 2451
Rotating 3D Chart 2452

User Interactions ... 2453
Tooltip ... 2453
Zooming and Panning ... 2457
Marker and Crosshair ... 2459
Trackball ... 2462
Highlight ... 2464
Selection ... 2467
Exporting ... 2471
Excel Export ... 2473
Word Export ... 2474
PDF Export ... 2475
Image Export ... 2476
PivotChart format selection ... 2477
Exporting Customization ... 2478
Appearance ... 2484
Built-in Themes ... 2484
PivotChart - Area Customization ... 2485
Localization ... 2488
Localization in PivotChart ... 2488
Localization and Globalization of Cube Info (Client Mode) ... 2489
Localization and Globalization of Cube Info ... 2490
How to ... 2491
Configure pivot chart through the model properties ... 2491
PivotClient ... 2493
PivotClient ... 2493
Key Features ... 2493
Getting Started ... 2493
Getting Started ... 2493
Getting Started ... 2511
Data Binding ... 2529
Data Binding ... 2529
Data Binding ... 2540
PivotClient: Elements ... 2553
Cube Selector ... 2553
Cube Dimension Browser ... 2554

Axis Element Builder 2557
Elements Editor 2558
Toolbar 2559
Report Manipulation 2559
PivotGrid and PivotChart 2565
Relational 2565
Data Exploration 2565
Exporting 2568
Value Sorting 2576
Summary Types 2578
OLAP 2581
Data Exploration 2581
Exporting 2588
Defer Update 2595
KPI 2597
Measure Groups 2599
Paging 2601
Drill Through 2604
Calculated Members 2608
Common 2611
Save and Load Report 2611
Layout Customization 2618
Advanced Filtering and Sorting 2632
Member Editor 2634
Toggle Axis 2637
Localization and Globalization 2638
Responsive Layout 2648
Splitter 2649
How to 2651
Configure pivot client through the model properties 2651
PivotGauge 2654
Overview 2654
Getting Started 2654
Getting Started 2654
Getting Started 2667

Data Binding 2678
DataBinding 2678
DataBinding 2684
Frame Type 2690
Full Circle 2690
Half Circle 2691
Scale 2691
Adding Scale 2691
Scale Customization 2692
Pointers 2694
Pointer Types 2694
Adding Pointer Collection 2696
Appearance Customization 2696
Pointer Position 2697
Pointer Image 2698
Pointer Value Text 2699
Ticks 2700
Adding Tick Collection 2700
Tick Customization 2700
Ranges 2701
Adding Range Collection 2701
Appearance Customization 2701
Multiple Ranges 2702
Layout 2703
Row-wise Layout 2703
Column-wise Layout 2704
Labels 2705
Adding Label Collection 2705
Appearance Customization 2706
Unit Text 2707
Custom labels 2708
Adding Custom Label Collection 2708
Appearance Customization 2708
Multiple Custom Labels 2709
Label Format 2710

Tooltip ... 2711
Animation ... 2712
Localization ... 2712
Localization in PivotGauge Control ... 2712
Localization and Globalization of Cube Info ... 2713
Responsive Layout ... 2714
How to ... 2716
Configure pivot gauge through the model properties ... 2716
PivotTreeMap ... 2718
Overview ... 2718
Key Features ... 2718
Getting Started ... 2718
Creating a simple application with PivotTreeMap and OLAP datasource (Client Mode) ... 2719
Creating a simple application with PivotTreeMap and OLAP datasource (Server Mode) ... 2722
DataBinding ... 2729
Binding PivotTreeMap to Offline Cube ... 2729
Binding PivotTreeMap to Cube in local SQL Server ... 2729
Binding PivotTreeMap to Cube in online SQL Server ... 2729
Binding PivotTreeMap to Cube in online Mondrian Server ... 2730
Binding PivotTreeMap to Cube in online ActivePivot Server ... 2730
WCF ... 2730
Drill Operation ... 2735
Named Sets ... 2735
Client Mode ... 2735
Server Mode ... 2736
Color Mapping ... 2737
Legend ... 2739
Legend Visibility ... 2739
Localization ... 2741
Localization and Globalization of Cube Info (Client Mode) ... 2741
Localization and Globalization of Cube Info (Server Mode) ... 2742
How to ... 2743
Configure pivot tree map through the model properties ... 2743
PdfViewer ... 2744
Overview ... 2744

Key Features ... 2744
Getting Started ... 2744
Toolbar Customization ... 2757
Localization ... 2767
Text Markup Annotation ... 2773
Handwritten Signature ... 2778
Interaction Mode ... 2785
Appearance ... 2786
Supported and Unsupported Elements ... 2787
PercentageTextBox ... 2790
PercentageTextBox ... 2790
Key Features ... 2790
Getting Started ... 2790
Create your first PercentageTextBox in MVC ... 2790
Set MinValue, MaxValue and value in PercentageTextBox ... 2793
Setting the Strict Mode Option ... 2794
Set Calculation process with PercentageTextBox Widgets ... 2795
Behavior Settings ... 2797
Decimal Places ... 2797
Persistence Support ... 2797
Strict Mode Support ... 2797
Enabled or Disabled ... 2798
Adjusting Textbox Size ... 2798
Increment Step ... 2798
Define Name ... 2799
Define Value ... 2799
Define MaxValue and MinValue ... 2799
Read Only Support ... 2800
Appearance ... 2800
CSS Class ... 2800
Rounded Corner Support ... 2801
Water Mark Text Support ... 2802
Globalization Support ... 2802
Configure Localization ... 2803
RTL Support ... 2803

Enable RTL ... 2803
Keyboard Interaction ... 2803
PivotGrid ... 2804
Overview ... 2804
Getting Started ... 2804
Getting Started ... 2804
Getting Started ... 2824
Data Binding ... 2837
Data Binding ... 2837
Data Binding ... 2848
Relational ... 2858
Grouping Bar ... 2858
PivotTable Field List ... 2864
Summary Type ... 2874
State persistence ... 2875
Value Sorting ... 2875
Calculated Field ... 2876
Cell Editing ... 2878
Defer Update ... 2879
Sub Total Hiding ... 2880
Exporting ... 2881
Collapse by default ... 2893
Group by date ... 2894
OLAP ... 2895
KPI ... 2895
Named Set ... 2896
Grouping Bar ... 2898
PivotTable Field List ... 2902
Write-back ... 2908
Defer Update ... 2909
Exporting ... 2911
Common ... 2923
PivotGrid: Elements ... 2923
Number format ... 2927
Save and Load Report ... 2930

Grid Layout 2941
Grand Total Hiding 2943
Advanced Filtering and Sorting 2945
Paging 2947
Member Editor 2950
Frozen Header 2952
Drill Through 2954
Resizing Column 2959
Styling 2959
Localization and Globalization 2963
Responsive 2975
ToolTip 2978
How to 2978
Configure pivot grid through the model properties 2978
ProgressBar 2981
Progress Bar 2981
Key Features 2981
Getting Started 2981
Create your first Progress Bar in MVC 2981
Create a Textbox and Progress Bar 2982
Define value 2984
Value 2984
Percentage 2985
Setting Range 2986
Appearance and Styling 2986
Adjusting ProgressBar size 2986
Enabling the ProgressBar 2989
State Maintenance 2989
RTL Support 2990
How To 2990
Render ProgressBar from code behind 2990
RadialMenu 2991
RadialMenu 2991
Key Features 2991
Getting Started 2991

Creating your first RadialMenu in MVC application 2991
Create a Simple RadialMenu with RTE content 2992
RadialMenu Dependencies 2994
Dimension 2995
Item Customization 2997
Image Customization 3003
Appearance and Styling 3006
Theme 3006
CSS Class 3006
Template Support 3008
How to 3012
Render RadialMenu from Code behind 3012
RadialSlider 3013
Overview 3013
Getting Started 3013
Creating your first RadialSlider in MVC application 3013
RadialSlider Dependencies 3014
Dimension 3014
Stroke Width 3014
Setting radius 3015
Behavior settings 3015
Show/hide RadialSlider on initial rendering 3015
Value Accuracy 3016
Display inline 3016
Modifying Label Space 3016
Show/Hide inner circle 3017
Values customization 3018
Animation Effect 3019
Display Angle Settings 3019
Start Angle 3019
End Angle 3020
Image customization 3021
Customizing inner circle 3021
Appearance and Styling 3023
Theme 3023

CssClass ... 3024
How to ... 3025
Render RadialSlider from Code behind ... 3025
RadioButton ... 3026
RadioButton ... 3026
Key Features ... 3026
Getting Started ... 3026
Create your first RadioButton in MVC ... 3026
Easy Customization ... 3033
Checked status ... 3033
Text ... 3034
Size ... 3034
RTL Support ... 3036
Styles Customization ... 3037
Miscellaneous ... 3038
Radio Button ID ... 3038
Radio Button Prefix id ... 3038
Radio Button Name ... 3038
Radio Button Value ... 3038
How to? ... 3039
Render RadioButton from Code behind ... 3039
RangeNavigator ... 3040
RangeNavigator ... 3040
Key Features ... 3040
Getting Started ... 3040
Create your first RangeNavigator in MVC ... 3040
Range Types ... 3051
Numeric Type ... 3051
DateTime ... 3051
DateTime Intervals ... 3052
Range Padding ... 3052
Numeric ... 3052
None: ... 3052
Additional: ... 3053
Normal: ... 3053

Round: 3053
DateTime 3053
None: 3054
Round: 3054
Padding 3054
AllowSnapping 3054
Responsive 3054
Auto Resizing 3055
Customize range Navigator border 3055
Customize size of range navigator 3055
Customize axis range of navigator 3055
Populate Data 3056
Add series to the RangeNavigator 3056
Behavior Customization 3057
Deferred update 3057
Methods 3057
Handle Events 3058
Use of ZoomCoordinates 3061
Thumb Template 3062
Value Axis Settings 3063
Selected Range Settings 3063
Appearance and Styling 3063
Customize labels 3064
Label Placement: 3065
Customize RangeNavigator 3065
Themes 3066
Tooltip 3067
Customization 3067
Label Format 3068
Tooltip display mode 3068
Globalization & Localization 3069
Localization 3069
RTL 3070
User Interactions 3070
Highlight 3070

Selection ... 3071
Scrollbar ... 3072
Rating ... 3073
Rating ... 3073
Key Features ... 3073
Getting Started ... 3073
Create your first Rating control in MVC ... 3073
Rating Customization ... 3076
Setting Value ... 3076
Min Value ... 3077
Max Value ... 3077
Set Precision ... 3078
Increment Step ... 3079
Resetting values ... 3079
Read only ... 3080
Enable or Disable ... 3081
Appearance and Styling ... 3081
Show ToolTip ... 3081
Adjusting Rating Size ... 3082
Orientation ... 3085
How To ... 3086
Render Rating from code behind ... 3086
ReportViewer ... 3086
ReportViewer ... 3086
Key Features ... 3086
Getting Started ... 3087
Create your first ReportViewer in MVC ... 3087
Load SSRS Server Reports ... 3093
Load RDLC Reports ... 3094
Report Controller ... 3096
IReportController ... 3096
ReportHelper ... 3097
SSRS Configuration ... 3098
Network Credentials for SSRS ... 3098
Report DataSources ... 3098

DataSource Credentials ... 3100
Report Parameters ... 3101
Toolbar Customization ... 3101
Events ... 3102
Limitations ... 3103
RDL Specification support ... 3103
Layout process ... 3103
Unsupported expression ... 3103
ReportDesigner ... 3103
Overview ... 3103
Key features ... 3103
Getting Started ... 3103
Create your first ReportDesigner application in ASP.NET MVC ... 3103
Add Assembly References ... 3105
Add Controller ... 3107
Add Scripts and Styles ... 3112
ReportDesigner Dependencies ... 3121
API Service ... 3122
ReportWriter ... 3123
Overview ... 3123
Getting Started with ASP.NET MVC Application ... 3124
Project Creation ... 3124
ReportWriter Initialization ... 3129
Generate RDL Reports ... 3135
Generate RDLC Reports ... 3135
Generate SSRS Reports ... 3137
Exporting Reports ... 3138
ReportWriter API ... 3143
Constructors ... 3143
Properties ... 3144
Methods ... 3144
Events ... 3145
How To ... 3145
How to ... 3145
Ribbon ... 3145

Overview 3145
Key Feature 3145
Getting Started 3146
Control Initialization 3146
Adding Tabs 3146
Configuring Groups 3147
Adding Controls to Group 3148
User Interface 3150
Ribbon Dependencies 3150
Application Tab 3151
Application Menu 3151
Backstage Page 3154
Tab 3156
Group 3157
Adding Tab Groups 3157
Group Expander 3161
Controls Support 3162
Built in Controls 3162
Custom 3165
Contextual Tab And Tab Set 3166
Gallery 3168
Gallery Items 3168
Custom Gallery Items 3170
Resize 3173
Tablet Layout 3173
Mobile Layout 3174
Mobile Toolbar Customization 3176
Group Button Customization 3182
Screen Tips 3186
HTML Tooltip 3186
Custom Tooltip 3187
Quick Access Toolbar 3191
Appearance and Styling 3194
CssClass 3194
Themes 3195

Customize Styles ... 3196
AngularJS-Support ... 3197
Tabs and Contextual Tabs ... 3197
Gallery and Custom Tooltip ... 3198
Load on Demand ... 3200
Initially Collapsible ... 3203
How to ... 3206
Get Ribbon object ... 3206
RichTextEditor ... 3206
Overview ... 3206
Getting Started ... 3206
Create your first RichTextEditor in MVC ... 3206
Toolbar–Configuration ... 3206
Setting and Getting Content ... 3207
Toolbar Configuration ... 3208
Toolbar Items ... 3208
Rearrange Group ... 3210
Undo and Redo ... 3211
Clipboard Operations ... 3211
Specify Custom Tools ... 3212
Types of responsive toolbar ... 3213
Working with Content ... 3214
IFrame Attributes ... 3214
Adding CSS File ... 3214
Content Editable ... 3215
Submit Content ... 3216
Refresh ... 3217
Persistence ... 3217
Set Default Font Name and Size ... 3217
Adding Font names and size ... 3219
Apply Font color and Background color ... 3220
Insert the content at cursor ... 3220
Footer ... 3221
Source View ... 3221
HTML Tag Info ... 3222

Characters Count/Word Count 3222
Clear Format 3223
Resize Handle 3223
Context Menu 3225
Adding an item with the context menu 3226
Removing an item from the context menu 3226
Image and File browser 3227
Insert an Image from Online Source 3227
Insert an Image from Your Computer 3228
Image Properties 3230
Resize an Image 3231
Suppression of the Image Browser 3233
Working with Hyperlinks 3234
Add and Edit a hyperlink 3234
Remove a hyperlink 3235
Working with Selection 3235
Select All 3235
Select a Range 3235
Get Selection 3236
Working with Tables 3236
Create a Table 3237
Insert a Table 3237
Insert and Delete a Row or Column 3238
Format a Table 3239
Delete a table 3240
Working with Lists 3241
Create a Lists 3241
Custom Lists 3241
Clean unwanted elements and styles when copy paste from Microsoft Word 3244
Zoom 3246
Print 3247
Find and Replace 3248
Localization 3249
Keyboard Support 3254
Strongly typed HTML helper 3255

XHTML Validation 3256
Import 3257
Import Server Dependencies 3258
Export 3259
Export Server Dependencies 3261
Validation 3262
jQuery Validation Methods 3262
Client Side Validation 3263
Server side Validation 3265
How to 3267
Render the RichTextEditor in Partial View 3267
Import word document to the RichTextEditor and Export to word document/pdf 3268
Enable the Browser Spell checker in RTE 3273
Pass the RTE from partial view to main view using AJAX call 3274
Render RichTextEditor from code behind 3275
Display content from db and post content to db in RTE 3275
Rotator 3276
Rotator 3276
Key Features 3276
Getting Started 3277
Create your first Rotator in MVC 3277
Create a Rotator 3277
Data Binding 3278
Data Fields and Configuration 3279
Local data binding 3279
Behavior settings 3280
Enabling rotator 3280
Responsive rotator 3281
EnableAutoPlay 3281
StopOnHover 3281
Pager settings 3282
ShowPager 3282
Show options 3283
ShowNavigateButton 3284
ShowCaption 3285

Thumbnail 3285
Orientation 3287
Horizontal 3287
Vertical 3288
Appearance and Styling 3288
Adjusting rotator item size 3288
Image with Contents 3288
Display items 3292
NavigateSteps 3292
StartIndex 3293
FrameSpace 3293
AnimationType 3294
AnimationSpeed 3294
Delay 3295
Theme 3295
CssClass 3295
Template 3296
RTL Support 3297
Keyboard interaction 3298
How to 3300
Render Rotator from Code behind 3300
Schedule 3301
Overview 3301
Getting Started 3301
Data Binding 3304
Appointment Fields 3304
Appointment Field Validation 3307
Binding to Simple List Data Array 3309
Binding Remote Data Service 3309
OData V4 3310
WebAPI Binding 3310
Data Binding using OLEDB 3311
Controller Action Binding with CRUD operation 3315
Loading Data on Demand 3318
Entity Framework Data Binding 3320

SQL Data Binding 3322
Views 3325
Day 3326
Week 3326
Work Week 3327
Month 3327
Customizing number of days (Custom View) 3328
Agenda 3329
Restriction on View Navigation 3330
Timeline View 3330
Working with Appointments 3331
Appointment Types 3331
CRUD operation 3332
Handling Appointment Actions 3340
Read Only 3341
Drag and Drop 3342
Resize 3346
Categorization 3348
Priority 3350
Search or Filter Appointments 3351
Recurrence Options 3354
Reminder 3358
Block Time Intervals 3359
Context Menu 3363
Default Menu Options 3363
Custom Menu Options 3365
Handling Menu Actions 3366
Adding Categorize Option 3368
Resources 3369
Fields of Resources 3369
Data Binding 3371
Multiple Resources (Without Grouping) 3373
Grouping 3374
Customization 3377
Hour Customization 3377

Hide Weekend days ... 3379
TimeScale ... 3379
Date Customization ... 3380
Appointment Window Customization ... 3381
Scheduler Customization using QueryCellInfo ... 3388
Navigation ... 3390
View Navigation ... 3390
Date Navigation ... 3391
Appointment Navigation ... 3392
Template ... 3392
Appointment Template ... 3393
Cells Template ... 3394
Date Header Template ... 3395
Resource Header Template ... 3396
TimeScale with Template ... 3398
Priority Settings Template ... 3399
Tooltip Template ... 3401
Agenda View Templates ... 3402
Globalization and Localization ... 3403
Globalization ... 3403
Localization ... 3404
Time Zone ... 3407
Time Mode ... 3410
Date Format ... 3411
First Day of Week ... 3411
Keyboard Navigation ... 3412
Setting Dimension ... 3413
Scheduler Dimension ... 3413
Scheduler Cell Dimensions ... 3413
Cell Auto-Height ... 3414
Responsiveness ... 3415
Auto-Resizing Scheduler ... 3415
Scheduler in Mobile/Tablets ... 3416
Persistence ... 3417
Export and Print ... 3418

Export Appointments to ICS file 3418
PDF Export 3421
Print 3425
Import Appointments 3427
Miscellaneous 3428
Time Indicator 3428
Show/Hide All-Day Row 3429
Show/Hide Header bar 3430
Show/Hide Location Field 3430
Recurrence Editor 3431
Getting Started 3431
Generating Recurrence Rule 3431
How To 3432
Validate the Custom Appointment Window Fields 3432
Highlight Different Work Hours for Each Resources 3433
Display Scheduler with Appointments Filtered by Subject 3436
Customize the Default Appointment Window 3437
SignalR Implementation 3438
Synchronize the Schedule with Outlook 3440
Scroller 3443
Overview 3443
Key Features 3443
Control Structure 3444
Getting Started 3444
Create your first Scroller in MVC 3444
Scroller Styles 3445
Thumb Scrolling 3447
Customizing the scroll Step 3447
RTL 3448
How to? 3449
Render Scroller from Code behind 3449
Signature 3450
Overview 3450
Key Features 3450
Getting Started 3451

Creating your first Signature in MVC application 3451
Adjusting Signature Size 3452
Signature Customization 3453
Stroke color 3453
Stroke Width 3454
Background color 3454
Background Image 3455
How To 3456
Save signature image with user defined format 3456
To clear the Signature 3457
Make signature as responsive 3457
Render Signature from Code behind 3459
Slider 3460
Slider 3460
Key Features 3460
Getting Started 3460
Create your first Slider in MVC 3461
Behavior Settings 3466
Persistence Support 3467
Orientation 3468
Slider Types 3469
Updating slider value 3470
Scale Settings 3472
Show Scale 3472
Enable Small Ticks 3472
Appearance and Styling 3474
CSS Class 3475
Animation 3477
Enable/Disable the Slider 3478
RTL support 3478
Enabling RTL 3478
Keyboard Interaction 3479
Configure keyboard interaction 3479
SpellCheck 3480
SpellCheck 3480

Key Features ... 3480
Getting Started ... 3480
Create your first SpellCheck in MVC ... 3480
SpellCheck Using Helper element ... 3480
Service Reference ... 3481
Spell Checking ... 3481
Dependencies ... 3482
Dictionary ... 3483
Default Dictionary ... 3483
Custom Dictionary ... 3483
Operations ... 3484
Dialog Mode ... 3484
Open Dialog Mode ... 3484
Handling Dialog Actions ... 3485
Context Menu Mode ... 3487
Default Menu Options ... 3487
Custom Menu Options ... 3488
Handling Menu Actions ... 3488
Functionalities ... 3489
Check Spelling ... 3489
Ignore Words ... 3489
Ignore ... 3489
IgnoreAll ... 3490
Word Collection to Ignore ... 3490
Ignore Settings ... 3491
Change Words ... 3492
Change ... 3492
ChangeAll ... 3493
Custom Words ... 3494
Checking content on typing ... 3494
Suggestion Words ... 3495
Synchronous request ... 3496
Customization ... 3496
Misspell Word Appearance ... 3497
Restrict Suggestion Count ... 3498

Localization 3499
Multiple Targets 3502
Dialog Mode 3503
Context Menu Mode 3504
Responsiveness 3504
SplitButton 3506
SplitButton 3506
Key Features 3506
Getting Started 3506
Create your first Split Button in MVC 3507
Easy customization 3507
Content for Split button 3507
Button Size 3508
ContentType 3510
Prefix and Suffix icons 3511
Prefix Icon 3511
Suffix Icon 3511
Image Position 3513
Theme support 3516
Custom CSS 3516
Dropdown Button 3519
Arrow Position 3519
RTL Support 3521
Miscellaneous 3522
Text 3522
ShowRoundedCorner 3522
How to? 3523
Render SplitButton from Code behind 3523
Splitter 3523
Splitter 3523
Key Features 3524
Getting Started 3524
Create your first Splitter in MVC 3524
Configure Tree View 3526
Set Actions to Splitter control to view the image 3526

Enabling Collapsible ... 3527
Configure Collapsible ... 3527
Splitter Orientation ... 3528
Configure Splitter Orientation ... 3529
Nested Splitter Support ... 3530
Configure Nested Splitter ... 3530
Splitter Integration ... 3532
Configuring other widgets in Splitter ... 3532
Appearance and Styling ... 3534
Responsive ... 3534
Animation Support ... 3537
Adjusting Splitter Size ... 3538
Theme ... 3540
CSS Class ... 3540
Keyboard Navigation ... 3542
Configuring Keyboard Navigation ... 3542
RTL Support ... 3543
Enable RTL ... 3543
How To? ... 3545
Render Splitter from Code behind ... 3548
Spreadsheet ... 3549
Overview ... 3549
Getting Started ... 3550
Initialize Spreadsheet ... 3550
Populate Spreadsheet with Data ... 3550
Apply Conditional formatting ... 3552
Export Spreadsheet as Excel File ... 3553
Worksheet ... 3554
List of Sheet Operations ... 3554
Headers ... 3559
Show / Hide Sheets ... 3560
Show / Hide Gridlines ... 3561
Rows and Columns ... 3562
Rows ... 3562
Columns ... 3562

List of operations 3563
Insert 3563
Delete 3564
Show and Hide 3567
Resizing 3571
Freeze Pane 3573
Freeze Rows 3577
Freeze Columns 3580
Unfreeze Pane 3583
Open and Save 3586
Open 3586
Save 3591
Server dependencies 3592
Data Binding 3592
Local Data 3593
Remote Data 3594
HTML Table Data 3596
Ways to bind data in Spreadsheet 3598
Cell Range 3605
Comment 3605
Cell Navigation 3607
Data Validation 3608
Drag and Drop 3612
Auto Fill 3614
Hyperlink 3618
Merge Cell 3620
Editing 3624
Edit cell content 3624
Save cell content 3624
Read-Only Cells 3626
Events 3628
Data binding 3628
Selection 3629
Selection Modes 3630
Selection Type 3630

Context Menu 3634
Default Context Menu items 3634
Data Presentation 3639
Cell Type 3639
Chart 3641
Conditional Formatting 3644
Filtering 3648
Picture 3652
Pivot Table 3652
Sorting 3655
Table 3657
Formulas 3660
User Defined Functions 3664
Named Ranges 3667
Formula Bar 3670
Auto Sum 3670
Formatting 3670
Number Formatting 3670
Text Formatting 3673
Cell Formatting 3676
Scrolling 3678
Set height and width for Scrolling 3678
Responsive 3679
Scroll Mode 3682
Virtual Scrolling 3682
Clipboard 3682
Cut 3682
Copy 3683
Paste 3683
Searching 3685
Find 3685
Replace 3687
Go To 3689
GoTo Special 3691
Undo and Redo 3692

Undo the last action ... 3692
Redo the action ... 3692
Print ... 3692
Print Sheet / Print Selected Area ... 3693
Print with Gridlines ... 3696
Print with Headings ... 3698
Keyboard Shortcuts ... 3700
Sparkline ... 3702
Types of Sparkline ... 3702
Sparkline Customization ... 3702
Sparkline ... 3705
Overview ... 3705
Getting Started ... 3705
Installation and Deployment ... 3705
Initialize Sparkline ... 3707
Working with Data ... 3707
Local Data ... 3707
Sparkline Dimensions ... 3708
Set size for the container ... 3709
Set size in pixels ... 3709
Responsive Sparkline ... 3709
Sparkline Types ... 3709
Line Type ... 3709
Column Type ... 3710
Area Type ... 3710
WinLoss Type ... 3710
Pie Type ... 3711
Axis Customize ... 3711
Marker Customization ... 3711
Point Customization ... 3712
Tooltip ... 3712
Tooltip Customization ... 3712
Tooltip Template ... 3713
Range Band ... 3713
Sparkline Customization ... 3713

Sparkline background 3714
Stroke color and width 3714
Sparkline border 3714
Opacity 3715
Padding for Sparkline 3715
Canvas support 3715
Themes 3716
Methods and Events 3716
Methods 3716
Events 3716
SunburstChart 3720
Overview 3720
Getting Started 3720
Installation and Deployment 3720
Initialize Sunburst Chart 3722
Populate Data source: 3722
Add Title to the Sunburst Chart 3725
Enable Legend 3725
Add Data Labels 3725
Sunburst Elements 3726
Start and End Angle 3726
Sunburst Radius 3727
Sunburst Inner Radius 3727
Levels 3728
GroupMemberPath 3728
Legend 3729
Legend Icon 3730
Positioning the Legend 3731
Legend Item Size and border 3732
Legend Size 3733
Legend Row and Columns 3734
LegendInteractivity 3735
ToggleSegmentSelection 3735
Toggle Segment Visibility 3736
Data Labels 3737

Label Overflow mode ... 3738
Label Rotation Mode ... 3740
Customizing the data labels ... 3742
Tooltip ... 3743
Tooltip Template ... 3744
Highlight ... 3745
Highlight Display mode ... 3746
Highlight Mode ... 3747
Selection ... 3751
Selection Display mode ... 3752
Selection Mode ... 3753
Zooming ... 3757
Zooming toolbar ... 3758
Animation ... 3759
Animation Types ... 3759
Appearance ... 3761
Palette ... 3761
Built- in Themes ... 3762
Methods ... 3763
Events ... 3764
Tab ... 3769
Tab ... 3769
Key Features ... 3769
Getting Started ... 3769
Create your first Tab Control in MVC ... 3769
Behavior Settings ... 3779
Close Button ... 3779
Orientation ... 3779
State Maintenance ... 3780
Appearance and Styling ... 3781
Header Image Customization ... 3781
Rounded corner ... 3783
Enable/Disable ... 3783
Enabling Reload Icon ... 3784
Collapsible Tabs ... 3785

Adjusting Tab Size ... 3786
Integration with other widgets ... 3789
AJAX Content Load (Load on Demand) ... 3790
Sub Tab with AJAX Content ... 3790
RTL Support ... 3792
Keyboard Navigation ... 3793
Scroll Support ... 3794
Template Support ... 3795
ASP.NET MVC ... 3795
Render Tab from code behind ... 3796
How To ... 3797
Add a tab item dynamically based on condition ... 3797
TagCloud ... 3798
TagCloud ... 3798
Key Features ... 3798
Getting Started ... 3798
Create your first TagCloud in MVC ... 3798
Create a TagCloud ... 3798
Set the Min and Max Font Size ... 3799
Set an event to perform operation ... 3800
Data-Binding ... 3802
Fields ... 3802
Local Binding ... 3802
Remote Binding ... 3804
Title Customization ... 3805
ShowTitle ... 3805
Title ... 3805
TitleImage ... 3806
Appearance and Styling ... 3807
Minimum and maximum Font size ... 3807
Display Formats ... 3808
Theme ... 3809
CssClass ... 3810
RTL Support ... 3811
Enabling RTL Support ... 3811

Render TagCloud from Server Side ... 3811
Tile ... 3812
Tile ... 3812
Getting Started ... 3812
Create your first Tile in MVC ... 3813
Tile dependencies ... 3814
Image Configuration ... 3815
Text Configuration ... 3816
Template Support ... 3816
Configure Badge ... 3818
LiveTile Configuration ... 3818
Customize size ... 3819
Add Group Tiles ... 3820
Styles and Appearance ... 3822
Customize styles ... 3822
How to ... 3823
Render TileView from Code behind ... 3823
TimePicker ... 3823
TimePicker ... 3823
Key Features ... 3823
Getting Started ... 3824
Create your first TimePicker in MVC ... 3824
TimePicker Customization ... 3829
Creating TimePicker Widget ... 3830
Localization ... 3830
Enabling Localization Support ... 3830
Behavior Settings ... 3831
Set value of the TimePicker widget ... 3831
Enable/Disable TimePicker widget ... 3831
Restrict editing ... 3832
Rounded Corner ... 3832
Scaling ... 3832
State persistence ... 3833
Strict mode of the TimePicker ... 3834
Interval ... 3834

Range 3834
Steps to change MinTime & MaxTime of the TimePicker 3834
Time Format 3835
Steps to change Time Format of TimePicker widget 3835
RTL 3835
Enabling Right-To-Left Support 3835
Keyboard Interaction 3836
When popup is open 3836
Configure Keyboard Interaction 3837
How to? 3837
Render TimePicker from Code behind 3837
ToggleButton 3838
ToggleButton 3838
Key Features 3838
Getting Started 3838
Create your first Toggle Button in MVC 3838
Easy Customization 3839
Toggle State 3839
ToggleState with icons 3840
Toggle button size 3842
Content type 3843
Theme support 3848
Custom CSS 3848
Button types 3850
RTL support 3852
Miscellaneous 3852
ShowRoundedCorner 3852
PreventToggle 3853
How to? 3853
Render ToggleButton from Code behind 3853
Toolbar 3854
Toolbar 3854
Key Features 3854
Getting Started 3854
Create your first Toolbar in MVC 3854

Data Binding ... 3862
Data fields and configuration ... 3862
Local data ... 3863
Remote data ... 3865
Behavior settings ... 3865
Enabling Toolbar ... 3865
Hiding Toolbar ... 3866
Disable Or Enable Separate Toolbar Item ... 3866
Orientation ... 3868
Horizontal ... 3868
Vertical ... 3869
Appearance and Styling ... 3869
Adjusting Toolbar size ... 3869
Enabling Rounded Corner ... 3870
Enabling Separator ... 3870
Themes ... 3871
CssClass ... 3871
Template Support ... 3872
RTL ... 3872
Keyboard Navigation ... 3873
Responsive Layout ... 3874
responsiveType:Inline ... 3877
responsiveType:Popup ... 3878
How To ... 3880
Render Toolbar from code behind ... 3880
Tooltip ... 3880
Overview ... 3880
Getting started ... 3881
Create a Tooltip ... 3881
Setting Dimensions ... 3882
Tooltip Appearance ... 3882
Customization ... 3883
Template Support ... 3883
Animation Effects ... 3886
Modernize the tooltip’s content ... 3887

Closing Mode ... 3887
Position ... 3889
Containment ... 3890
Associates ... 3890
Collision ... 3892
How To ... 3892
Use AJAX to generate the Tooltip's content ... 3892
Integration with the Slider control ... 3895
Tip(arrow) customization ... 3896
Initialize Tooltip for the target element ... 3898
Interact with the Tooltip ... 3900
Render Tooltip from code behind ... 3900
TreeMap ... 3901
TreeMap ... 3901
Key Features ... 3901
Use Case Scenarios ... 3901
Getting Started ... 3901
Configure a TreeMap ... 3901
Create a simple MVC Application ... 3902
Add Data ... 3902
Initialize TreeMap ... 3903
Data Source ... 3904
Weight Value Path ... 3904
GroupTreeMap Items using Levels ... 3905
Customize TreeMap Appearance by Range ... 3906
Enable Tooltip ... 3907
Legend ... 3908
DataBinding ... 3909
TreeMapLevels ... 3910
Flat Level ... 3911
Hierarchical Level ... 3912
Layout ... 3914
Squarified Layout ... 3914
SliceAndDiceAuto Layout ... 3915
SliceAndDiceHorizontal Layout ... 3916

SliceAndDiceVertical Layout ... 3916
Customization ... 3917
Color ... 3917
Binding color from the datasource ... 3917
UniColorMapping ... 3918
Range Color Mapping ... 3918
Desaturation Color Mapping ... 3919
Tooltip ... 3920
Leaf Item Setting ... 3920
Border Brush ... 3922
Border Thickness ... 3922
Dock Position ... 3922
Clicking and Dragging ... 3923
Fill with Gradient ... 3923
Responsive Treemap ... 3923
GroupColorMapping ... 3924
GroupSelectionMode ... 3924
Header ... 3924
Specifying HierarchicalDatasource ... 3924
Treemap Items ... 3924
TreeMap Elements ... 3924
Legend ... 3925
Header ... 3927
Customizing the header ... 3928
Label ... 3929
Customizing the Overflow labels ... 3931
Palette Color Mapping ... 3931
Drill Down Support ... 3931
Enabling Drill Down ... 3932
Methods and Events ... 3933
Method ... 3933
Refresh() ... 3933
Events ... 3934
TreeMapItemSelected ... 3934
DrillStarted ... 3934

DrillDownItemSelected 3934
HeaderTemplateRendering 3935
Refreshed 3935
TreeMapGroupSelected 3935
ItemRendering 3936
LegendItemRendering 3936
AngularJS Support 3937
TreeGrid 3939
TreeGrid 3939
Getting Started 3940
Create your first TreeGrid in MVC 3940
Enable Toolbar 3943
Enable Sorting 3944
Enable Editing 3944
Enable Context Menu 3945
Enable Column Menu 3946
Define dimension of TreeGrid 3946
Data Binding 3947
Local Data Binding 3947
Remote data binding 3954
Virtual rendering 3957
Columns 3957
Editing type 3958
Formatting 3958
Defining column width 3959
Headers 3960
Frozen Columns 3962
Resizing 3963
Checkbox column 3964
Column Template 3965
Column Menu 3966
Show/Hide columns using method 3970
Command Column 3970
Tree column/ Expander column 3972
Visibility 3973

Read-only 3973
Validation Rules 3974
Column Reorder 3975
Text Alignment 3976
Get Column Object 3978
Rows 3978
Row Template 3978
Row Height 3982
Alternate row styling 3982
Row Drag and Drop 3983
Customizing Drag tooltip 3984
Multiple row drag and drop 3985
Details row 3986
Expand/Collapse Row 3991
Cell 3995
Tooltip 3995
Clip Mode 3999
Text Wrap 4001
Editing 4003
Add new record 4003
Add row position 4004
Edit modes 4006
Edit type and its params 4011
Dialog Template 4014
BeginEditAction - edit cell/row by single/double click 4016
Cell Edit Template 4017
Delete record 4018
Delete confirmation message 4019
Filtering 4020
Filtering Modes 4021
Filtering types 4027
Filter columns at initial load 4029
Disabling filtering for a specific column 4030
Toolbox searching 4030
Searching 4033

Search Hierarchy Modes 4033
Summary Row 4034
Customize height of total summary 4036
Expand/collapse total summary row 4036
Custom Summary 4038
Responsive 4039
Layout 4041
Mobile Layout 4041
Tablet Layout 4045
Changing responsive width using method 4048
Priority for columns 4049
Selection 4050
Row selection 4050
Get record details 4053
Cell selection 4053
Checkbox selection 4056
MultiSelection – Touch Option 4059
Clear selection using method 4059
Export 4060
Server dependencies 4061
Supported Export Types 4062
Export Mapper in MVC 4062
Multiple exporting 4063
Export Theme 4065
Paging 4065
Paging settings 4066
Pager Template 4067
Paging - Touch Option 4069
Toolbar 4069
Default Toolbar Items 4069
Custom Toolbar Items 4070
Sorting 4072
Sorting Columns 4072
Multicolumn sorting 4072
Disable sorting for specific column 4073

Sort column at initial load 4073
Stacked Headers 4074
Adding Stacked header columns 4074
Stacked header column customization 4075
Context Menu 4078
ContextMenu Customization 4079
Header Context Menu 4080
Printing 4081
Print Mode 4082
BeforePrint Event 4082
Appearance and Styling 4083
Configuring CSS class 4084
Customize rows and cells 4085
Keyboard navigation 4086
How to 4087
Rendering the TreeGrid control using TreeGridProperties in server-side 4087
TreeView 4088
Overview 4088
Key features 4088
Getting started 4088
Create your first TreeView in MVC 4088
TreeView using Helper element 4088
TreeView using Data Binding 4089
Create Instance for TreeView 4090
TreeView events 4090
Populate Data 4091
Fields 4091
Local data 4094
Remote Data 4096
Load on demand 4100
Tree Node 4104
Get / Set Node Value 4105
Get Parent Node 4106
Get Node Index 4106
Node Manipulations 4107

Editing 4109
Selection 4109
Ensure Visibility 4109
Checkboxes 4110
Indeterminate Checkboxes 4110
Auto Checkable 4110
Check or Uncheck Node 4111
Get Checked Nodes 4111
Drag and Drop 4111
Position Indicators 4112
Restriction 4112
Drag and drop between Trees 4112
Auto node structuring 4114
Multiple Selection 4114
Select Nodes 4115
Get Selected Nodes 4116
Drag and Drop Multiple Nodes 4116
Full Row Selection 4117
Customization 4119
Template Option 4121
State Persistence 4124
Right to Left Align 4125
How To 4125
Update the modified data from tree to database. 4125
TreeView context menu to process node operations 4128
Sorted data using refresh method 4130
Persist updated data after edit, add and remove node 4131
Filtering nodes in TreeView 4134
AngularJS data binding to update data while add and remove node 4136
Set tooltip for TreeView nodes 4138
Auto hide/show the expand/collapse icon of TreeView 4140
Customize the expand/collapse icons of TreeView 4141
In form post, how to get all checked nodes details of TreeView in server end 4142
Populate data for TreeView using AJAX 4144
Populate data for TreeView using XML data source 4146

UploadBox 4148
UploadBox 4148
Key Features 4148
Getting Started 4148
Create your first UploadBox in MVC 4148
Set Restriction for File Extension 4150
File Actions 4154
Save File Action 4154
Remove File Action 4155
AutoUpload 4156
Restricting uploading files based on its extension 4157
Allow Extension 4157
Deny Extension 4157
File Size 4158
Maximum File Size for UploadBox 4158
Enable or Disable the UploadBox 4160
Asynchronous Upload 4160
Synchronous Upload 4160
Multiple files upload 4162
Drag and Drop Support 4163
Appearance and styling 4167
Customizing Button Text 4167
Customizing Upload Dialog 4168
Show or Hide File details 4169
Theme 4169
Custom CSS 4170
RTL Support 4170
How To 4171
Render UploadBox from code behind 4171
Upload a file using pushFile property 4172
Use Synchronous upload with drag and drop 4174
WaitingPopup 4175
WaitingPopup 4175
Getting Started 4176
Create your first Waiting Popup in MVC 4176

Behavior and Settings 4179
Automatic Initializing WaitingPopup widget 4179
Show / Hide WaitingPopup 4181
Appearance and Styling 4182
Custom Text 4182
How To 4186
Render WaitingPopup from code behind 4186

Welcome to Syncfusion Essential ASP.NET MVC

The **Syncfusion Essential Studio for ASP.NET MVC** is a set of server-side wrappers that is empowered by Essential JavaScript. The user can add the Syncfusion MVC components into their application with the help of the server-side wrapper helper classes. The MVC helper's values are passed from the server-side wrapper classes to browser via ScriptManager, the control creation and manipulations takes place in the client-side.

How to best read this User Guide

- The best way to get started is to read the common "Getting Started" document for the component that you would like to start using first. The "Getting Started" guide gives enough information that you need to know before starting to write code. This is the only section that is recommended to read end-to-end before starting to write code, all the other information can be referred when needed.
- Once you get familiar with the basics of using the component, the next step will be to start integrating the component into your application. A good starting point will be to refer to the code snippets in the online sample browser that contains hundreds of code samples, it is very likely that you will find a code sample that resembles your intended usage scenario.
- After integrating the component into your application using one of the sample code snippets as a starting point, it is likely that you would need additional information on specific features and API. The best option will be to search a specific topic using the search box that is available at the top of the user guide.
- Another valuable resource is the API reference that provides detailed information on the object hierarchy as well as the settings available on every object.

Additional Help Resources

- The Knowledge Base section contains responses to some of the most common questions that other customers have asked us before. So this will be a good place to search for topics that are not covered in the user guide.
- Similar to the Knowledge Base, the forum section also contains responses to questions that other customers have asked us before.

Create a Support Incident

If you are still not able to find the information that you are looking for in the self-help resources mentioned above, please contact us by creating a support ticket.

System Requirement

he following topic describes the system requirements for ASP.NET MVC platform.

Operating Systems

- Windows 10
- Windows 8, 8.1
- Windows 7
- Windows vista
- Windows server 2008 and later

Copyright © 2001 - 2020 Syncfusion Inc.

Hardware Environment

- Processor: x86 or x64
- RAM : 512 MB (minimum), 1 GB (recommended)
- Hard disk: up to 5 GB of available space may be required. However, 500 MB free space is required in boot drive even if you are installing in other drive.

Web browsers

Syncfusion components are also mostly compatible with all the major browsers that we use on these days that includes the following.

- Internet Explorer 8+
- Microsoft Edge
- Mozilla Firefox 22 +
- Chrome 17 +
- Opera 12 +
- Safari 5 +

Development Environment

- Microsoft Visual Studio 2010/2012/2013/2015/2017/2019
- .NET Framework 3.5/4.0/4.5/4.5.1/4.6
- MVC Version: MVC3/MVC4/MVC5
- Microsoft SQL Server 2005 or higher
- Web browsers with JavaScript enabled

Download Installer

You can download the installer from Syncfusion.com website.

Download the Trial Version

There are two ways to download our 30-day trial.

Free Trial Page

1. You can evaluate our 30-day free trial from Free Trial page.
2. Once you fill the required form or made the login using the your Syncfusion registered account you can download the trial installer in the confirmation page.
3. You can download the latest version trial installer.
4. You can unlock the installer using the unlock key, also you can unlock the installer using the Syncfusion registered login credential.
5. You can download the trial installer using the Trials & Downloads page under your registered account at any time before the trial expire. (Refer the below screenshot).

Copyright © 2001 - 2020 Syncfusion Inc.



Start Trial Page

1. You can evaluate our 30-day free trial from Start Trial page.
2. You should login using your Syncfusion account to access this page.
3. You can start your trail by clicking on the required product.

Note: If you already using the trail products and it's not expired, you couldn't start the trial again for same product.

4. After you started the trial, you can download the latest version trial installer using the Trials & Downloads page.
5. In Trials & Downloads page, you can find your current active trial products. Trials, which you done in both Free Trial Page and Start trial pages are listed here.
6. Use the Download (element 1 in below screenshot) button to download the installer of respective product.
7. Online installer can be downloaded from this page.
8. No need of unlock key to unlock the online installer.
9. You can unlock the installer using the Syncfusion registered login credential.



Note: You can generate the license key for your active trial products from Trials & Downloads page. This license key will be mandatory to use our trial products in your application. To know more about License key, refer this help topic.

Download the License Version

1. You can find your available licensed products which under your registered Syncfusion account in License & Downloads page.
2. You can find all the licenses (both active licenses and expired licenses) which are under your account.
3. You can find the licenses listed based on SKU names.

Copyright © 2001 - 2020 Syncfusion Inc.

4. Use the Download (element 1 in below screenshot) button to download the installer of respective product.
5. Latest version installer will be downloaded from this page.
6. You can navigate to Downloads Older Versions (element 2 in below screenshot) to download older version installers.
7. From 16.2 version online installer will be downloaded by default, and earlier versions offline installer will be downloaded.

Note: Online Installer have been introduced from the release version 16.2.

8. You can navigate to More Downloads Options (element 3 in below screenshot) to download other installers.
9. EXE and Zip format available to download for Windows OS. Both are Offline Installer.
10. No need of unlock key to unlock the online installer.
11. You can unlock the installer using the unlock key for versions earlier to 16.2, also you can unlock the installer using the Syncfusion registered login credential.



Note: You can generate the license key for your licensed products from License & Downloads page. This license key will be required only from release version 16.2. To know more about License key, refer this help topic.

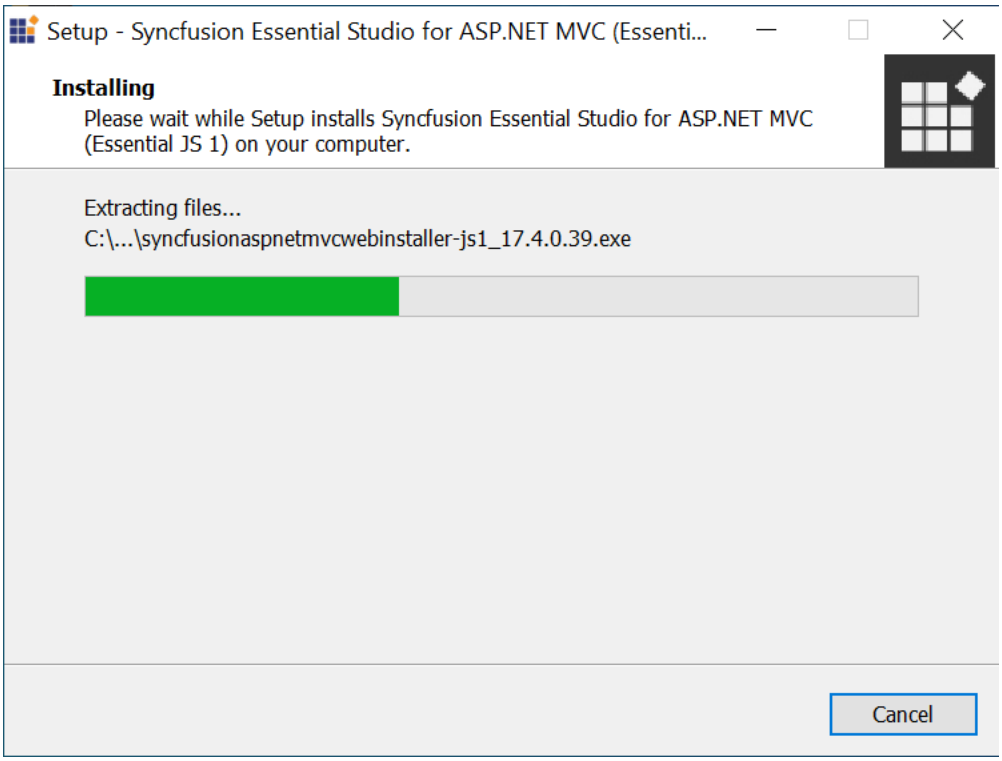
Installation and Upgrade

Installation

The following procedure illustrates how to install Essential Studio for ASP.NET MVC Online Installer.

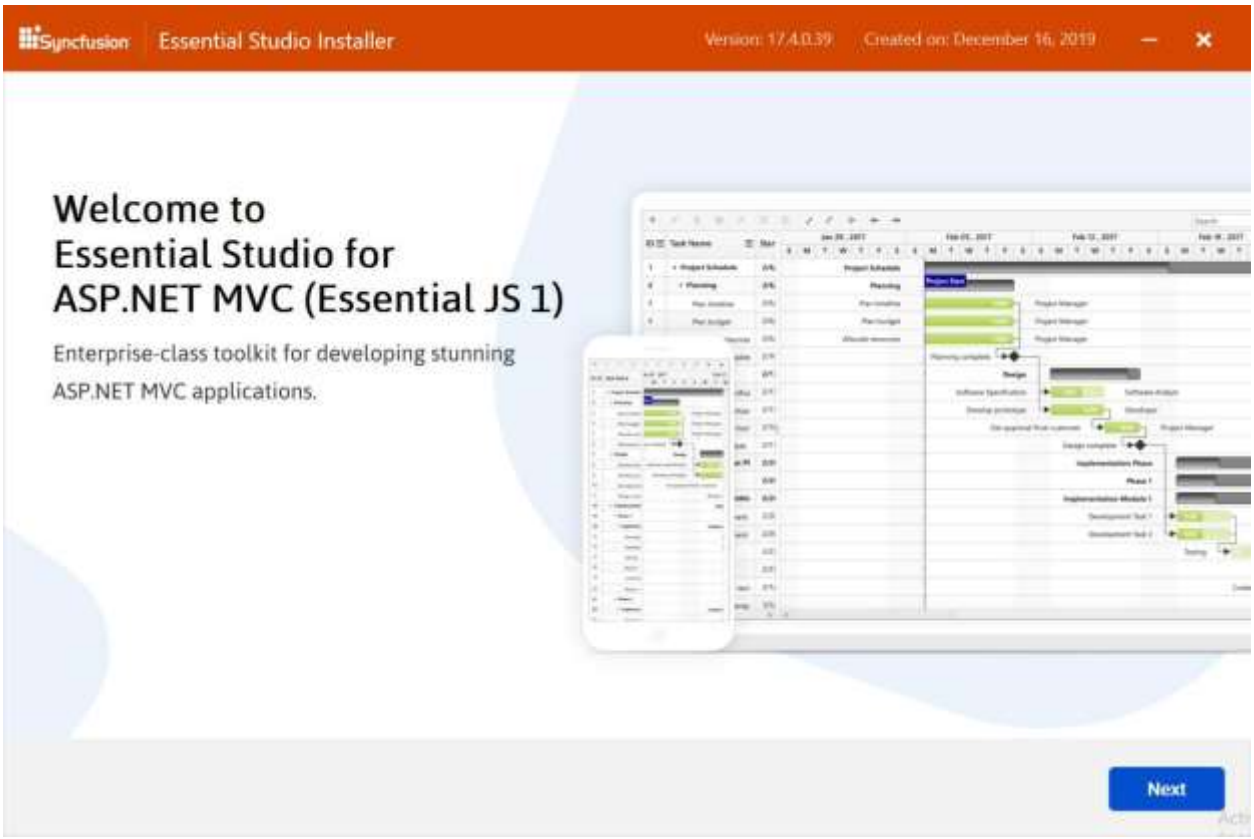
1. Double-click the Syncfusion ASP.NET MVC Online Installer file. The installer Wizard opens and extracts the package automatically.

Copyright © 2001 - 2020 Syncfusion Inc.



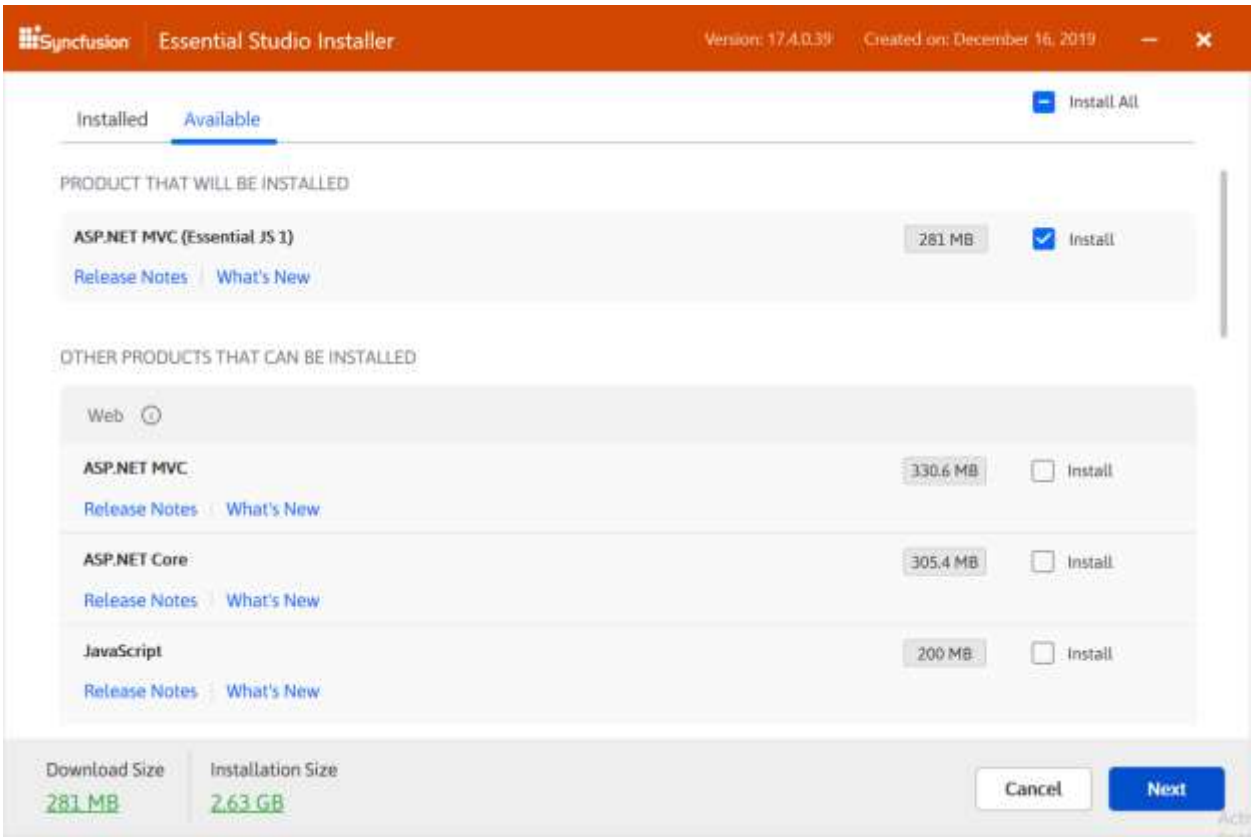
Note: The installer wizard extracts the syncfusionasnpetmvcwebinstaller_{version}.exe dialog, displaying the unzip operation of the package.

2. Welcome wizard of the Syncfusion ASP.NET MVC Online Installer will be displayed. Click Next.

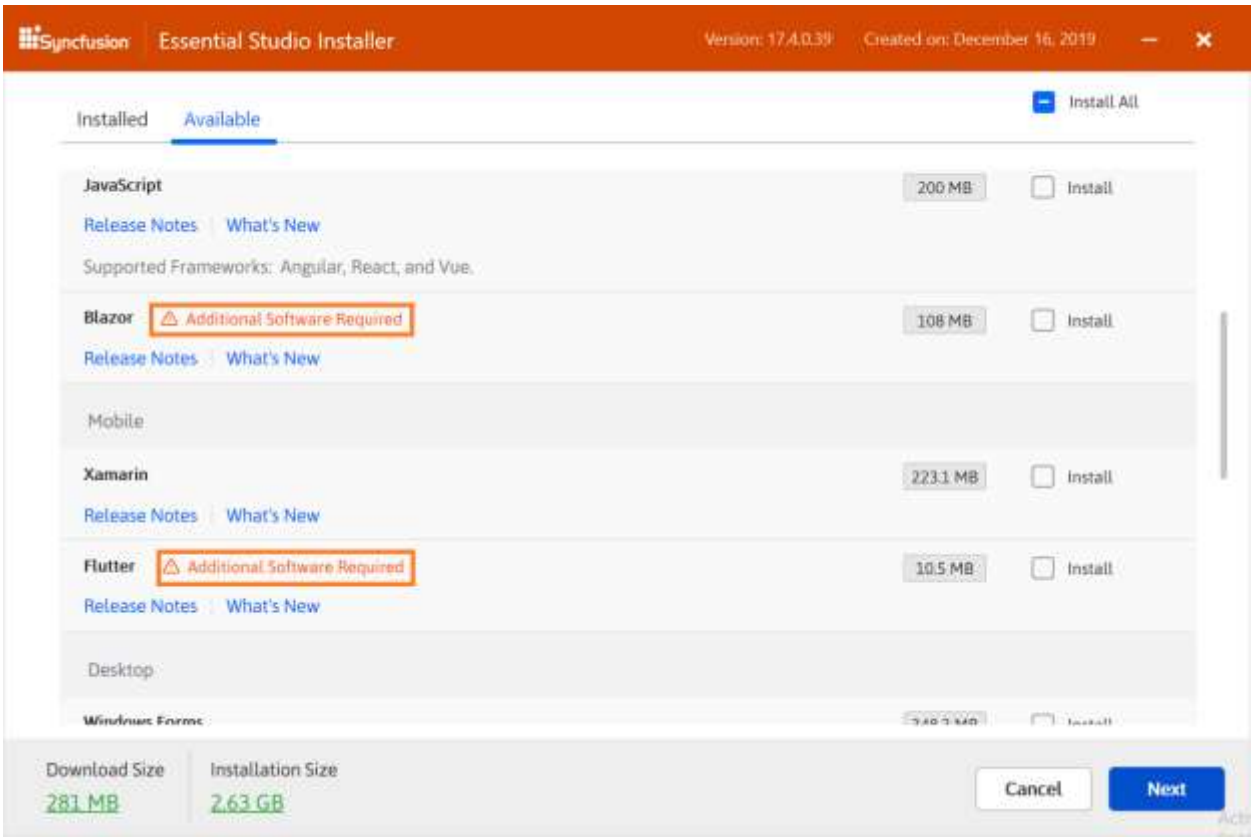


3. Platform Selection wizard will be displayed. Select the platforms to be installed from the **Available** tab. Select **Install All** checkbox to select all the platforms. Click Next.

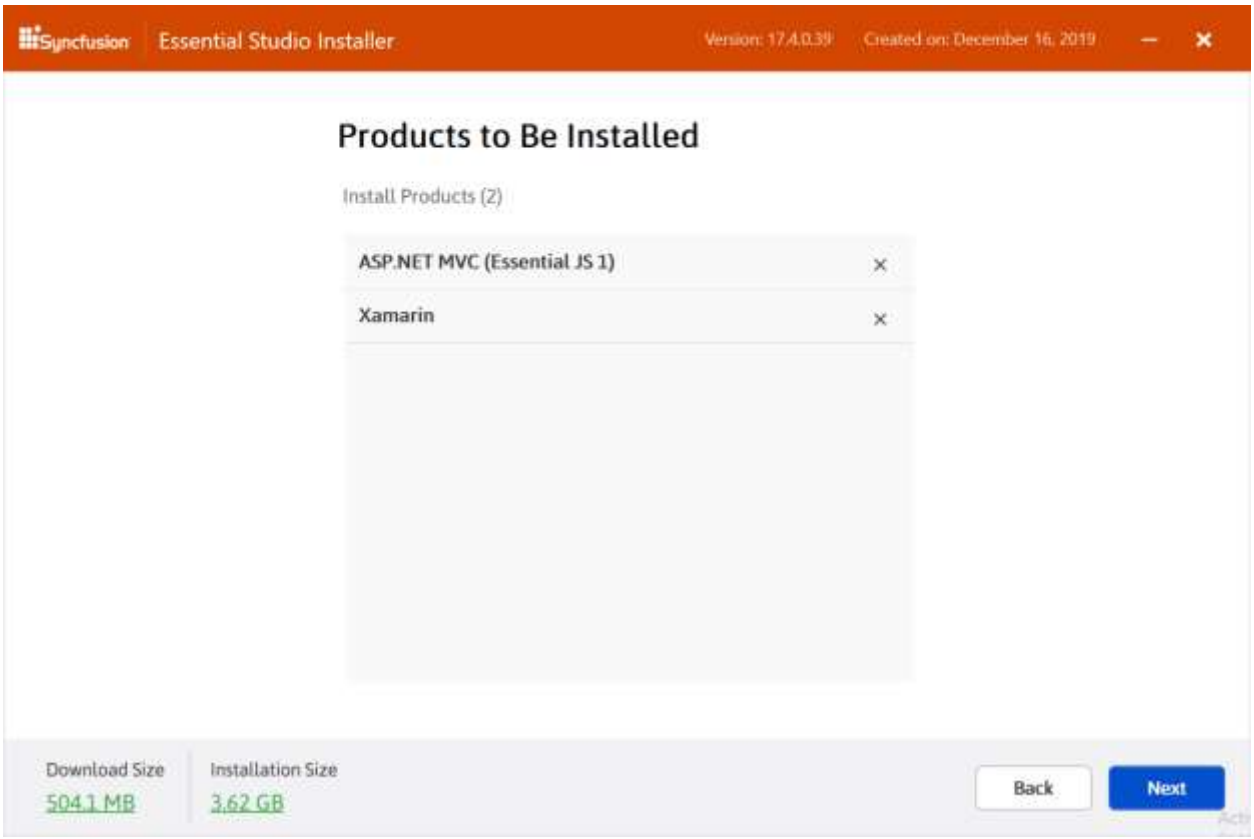
Copyright © 2001 - 2020 Syncfusion Inc.



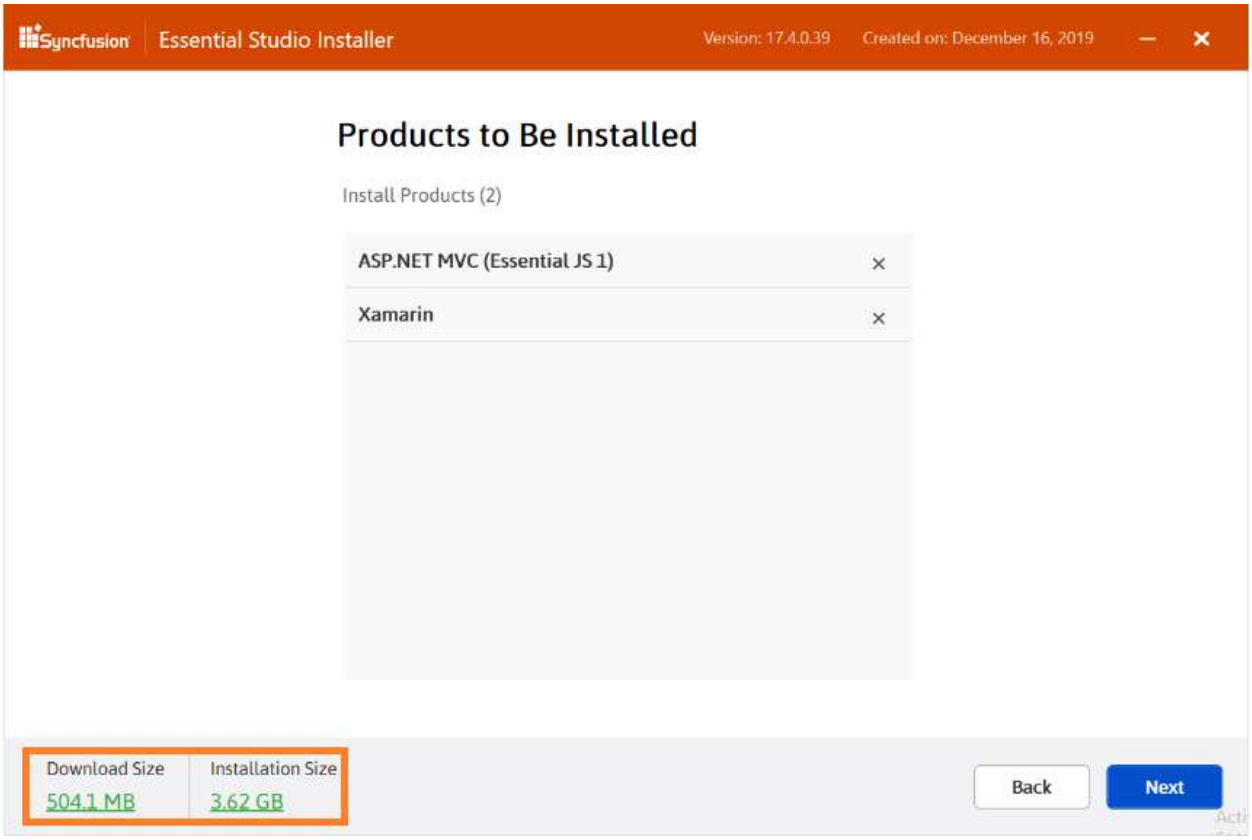
Information: If the required software of the selected platform was not already installed, **Additional Software Required** alert will be displayed. However, you can continue the installation and install the required software later.



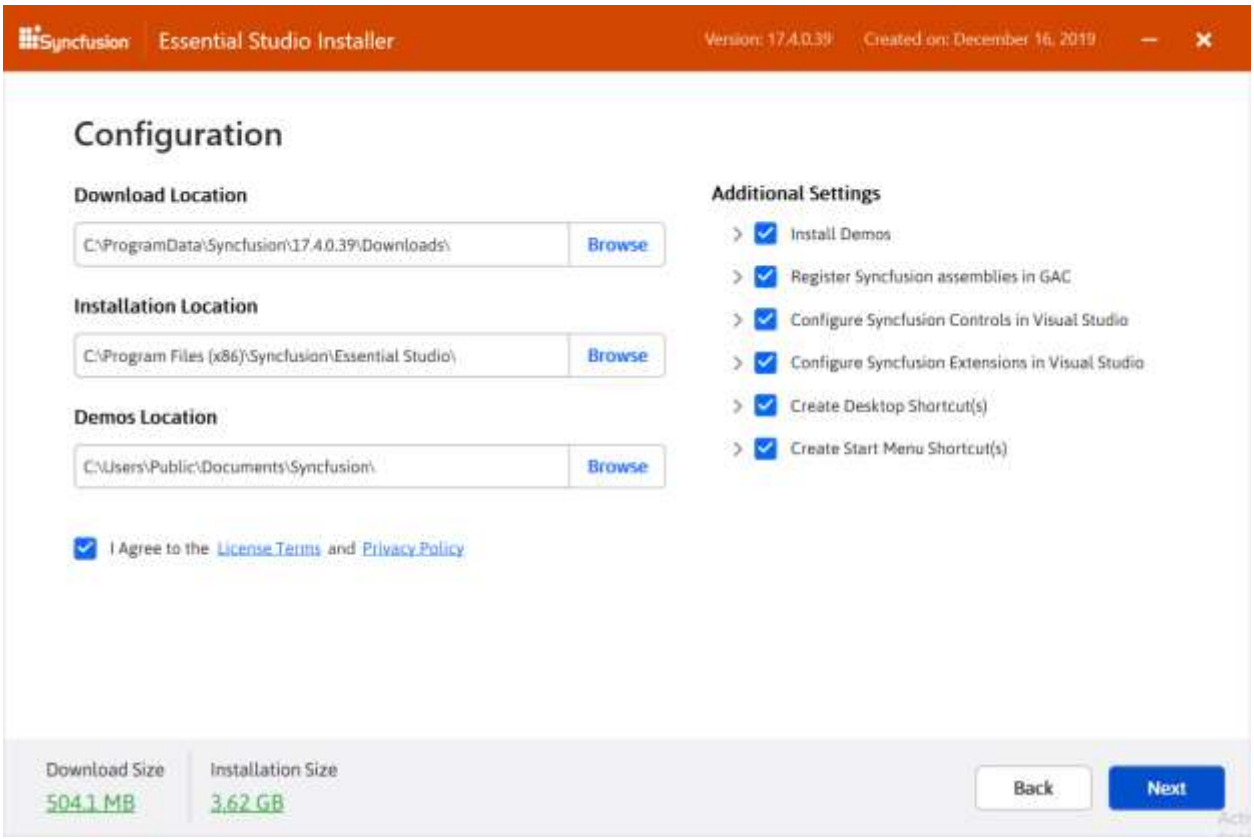
4. Confirmation wizard will be displayed. Here you can view and modify the list of platforms that will be installed.



Note: You can check the Estimated size of the Download and Installation by clicking the **Download Size and Installation Size** link.



5. Configuration wizard will be displayed. Here you can change the Download, Install and samples location. Also, you can change the Additional settings by platform basis. To install using the default configuration, click Next.



Information: From version 17.3 (2019 Vol 3), Syncfusion provides option to provide custom download location.

Note: From the 2018 Volume 2 release, Syncfusion has changed the install and samples location

Default Install location: {ProgramFilesFolder}\Syncfusion\{Platform}\{version}

Default Samples location: C:\Users\Public\Documents\Syncfusion\{platform}\{version}

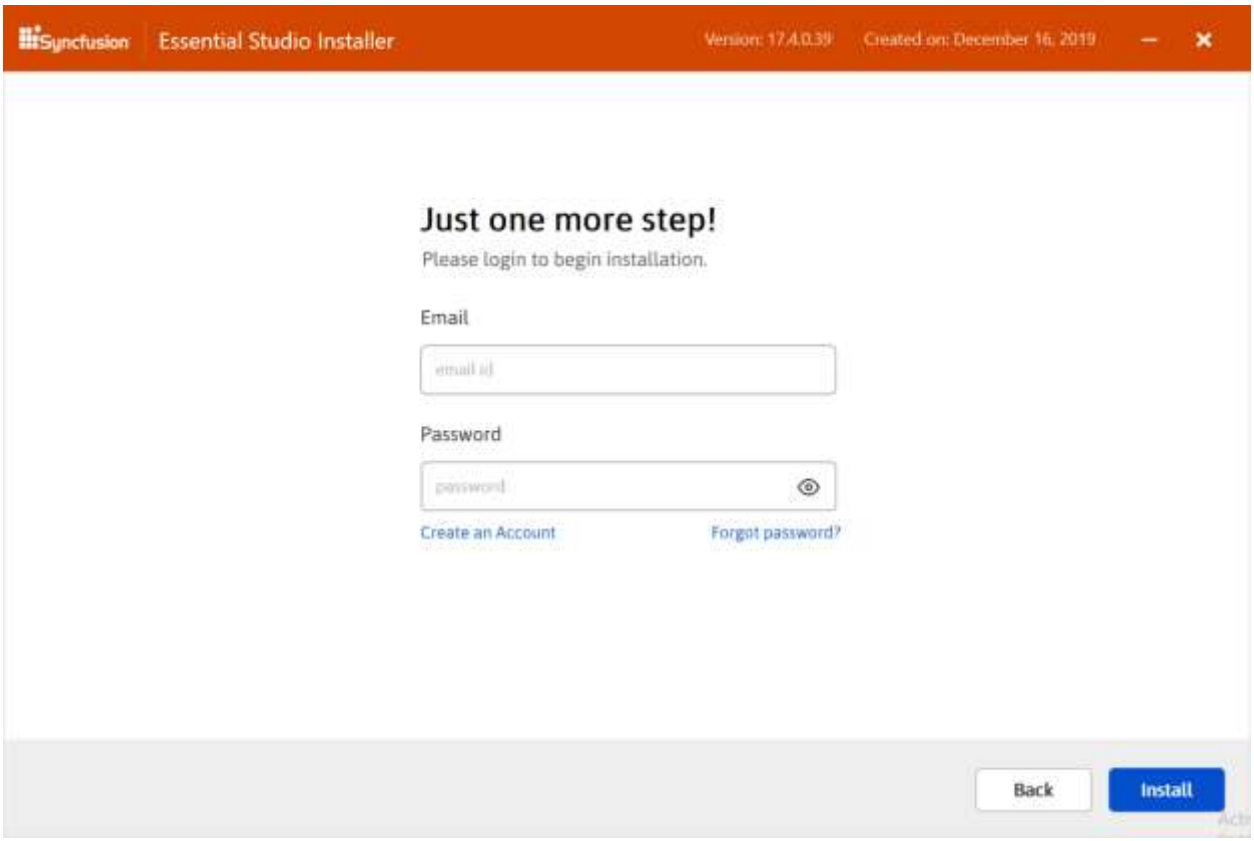
However, you can change the locations by clicking browse button.

- Select the **Install Demos** check box to install Syncfusion samples, or leave the check box clear, when you do not want to install Syncfusion samples.
- Select the **Register Syncfusion Assemblies in GAC** check box to install the latest Syncfusion assemblies in GAC, or clear this check box when you do not want to install the latest assemblies in GAC.
- Select the **Configure Syncfusion Extensions in Visual Studio** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.
- Select the **Create Desktop Shortcut** checkbox to create the desktop shortcut for Syncfusion Control Panel.

6. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click Next.
7. Login wizard will be displayed. You should enter your Syncfusion Direct-Trac login credentials. If you don't have Syncfusion Direct-Trac login credentials, then you can click on **Create an**

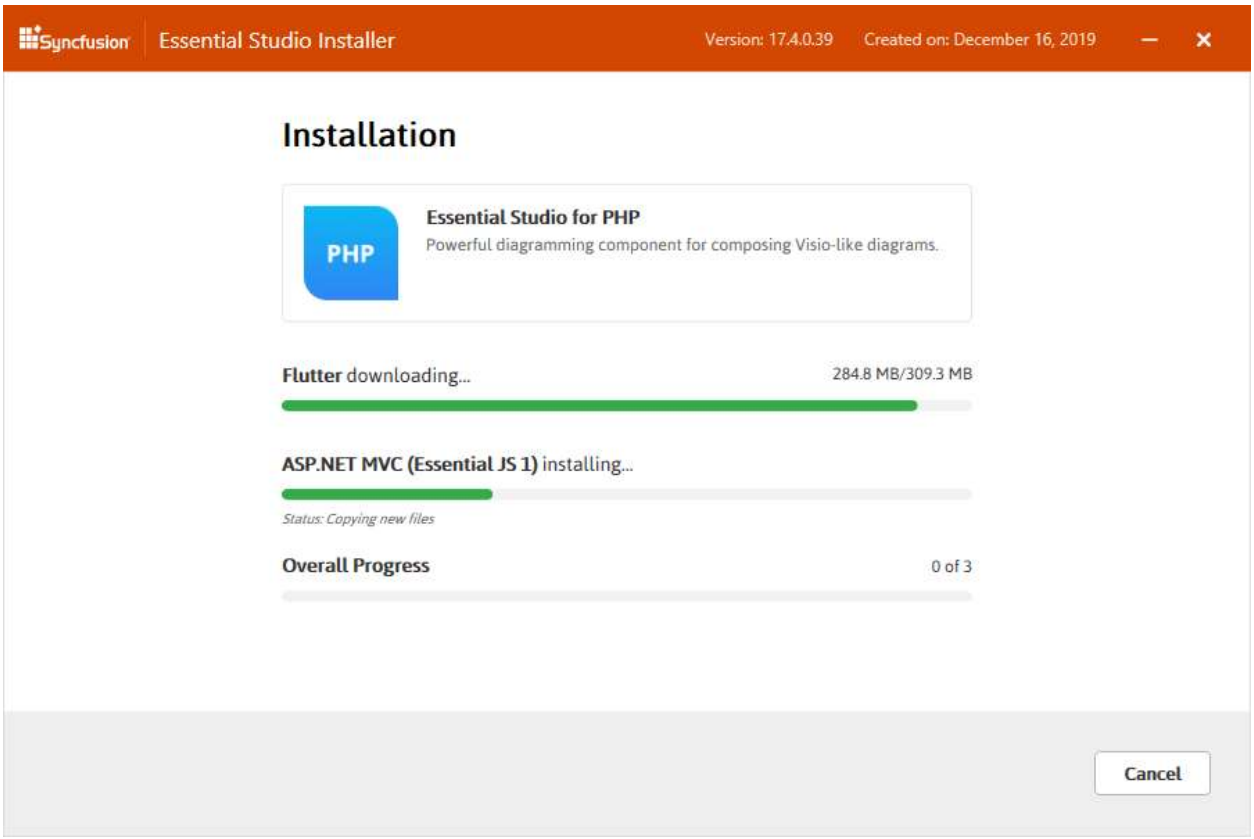
Copyright © 2001 - 2020 Syncfusion Inc.

Account. Else if you forgot your password, click on **Forgot Password** to create new password. Click Install.



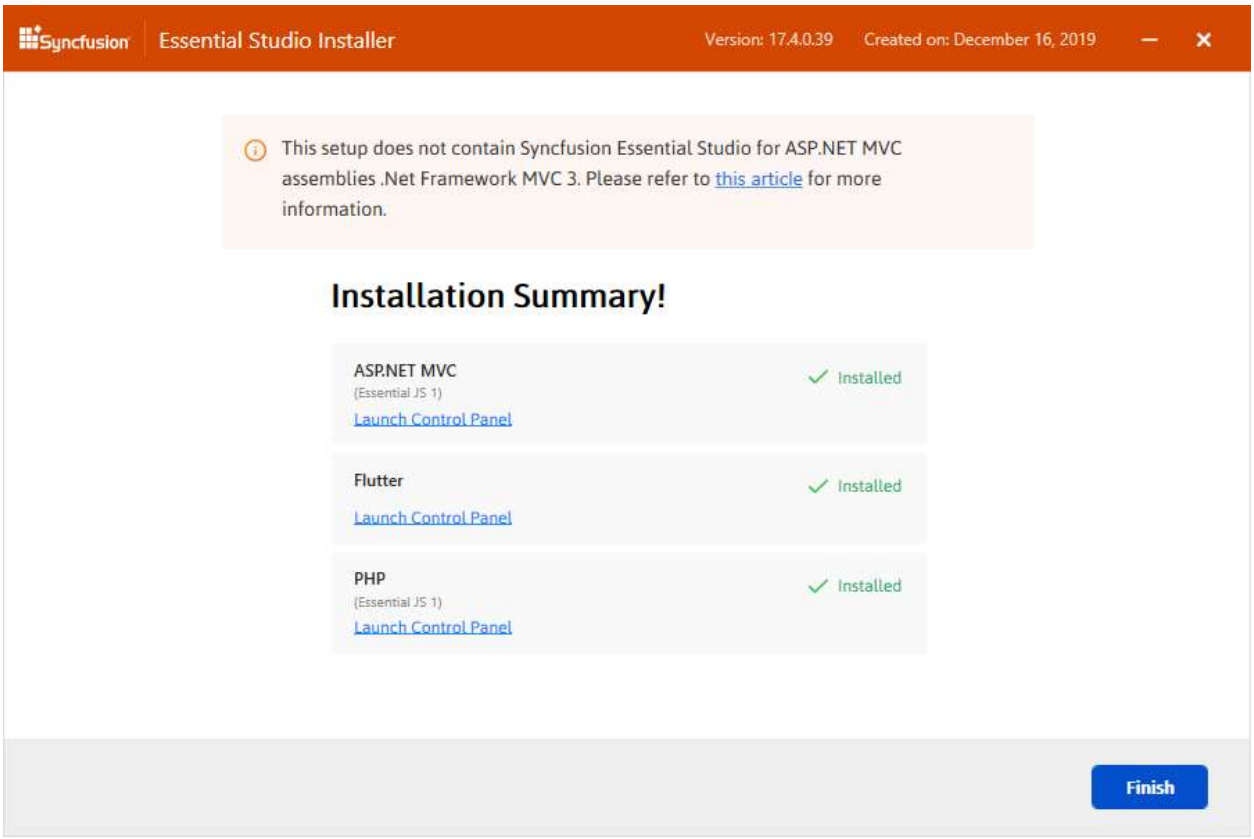
Information: The selected platforms will be installed based on your Syncfusion License (Trial or Licensed).

8. Download and Installation progress will be displayed.



9. Once the Installation is complete, **Summary** wizard will be displayed. Here you can check the list of platforms which are installed successfully and failed. Click Finish to exit the Summary wizard.

Copyright © 2001 - 2020 Syncfusion Inc.



- Click **Launch Control Panel** to open the Syncfusion Control Panel.

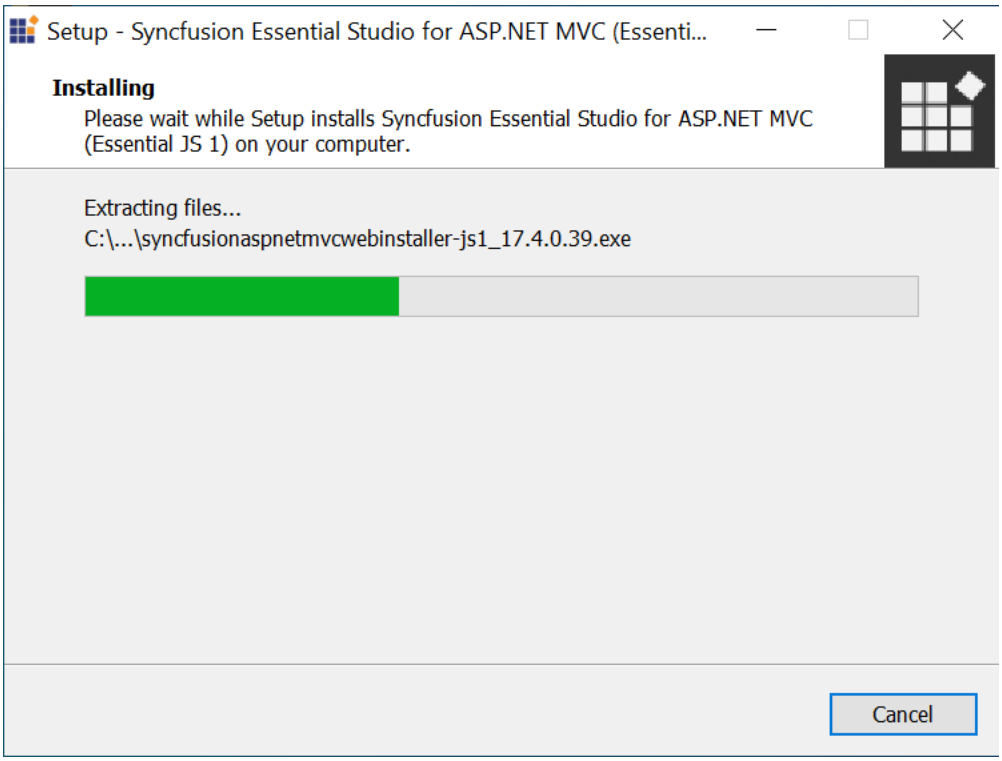
Uninstallation

Till version 17.2, Syncfusion Web Installer had option for installation alone. Starting with the version 17.3 (2019 Vol 3), Syncfusion provides option for uninstalling the platforms of the same version from the Web Installer application itself. Select the list of the platforms to be uninstalled and Web Installer will uninstall those platforms one by one.

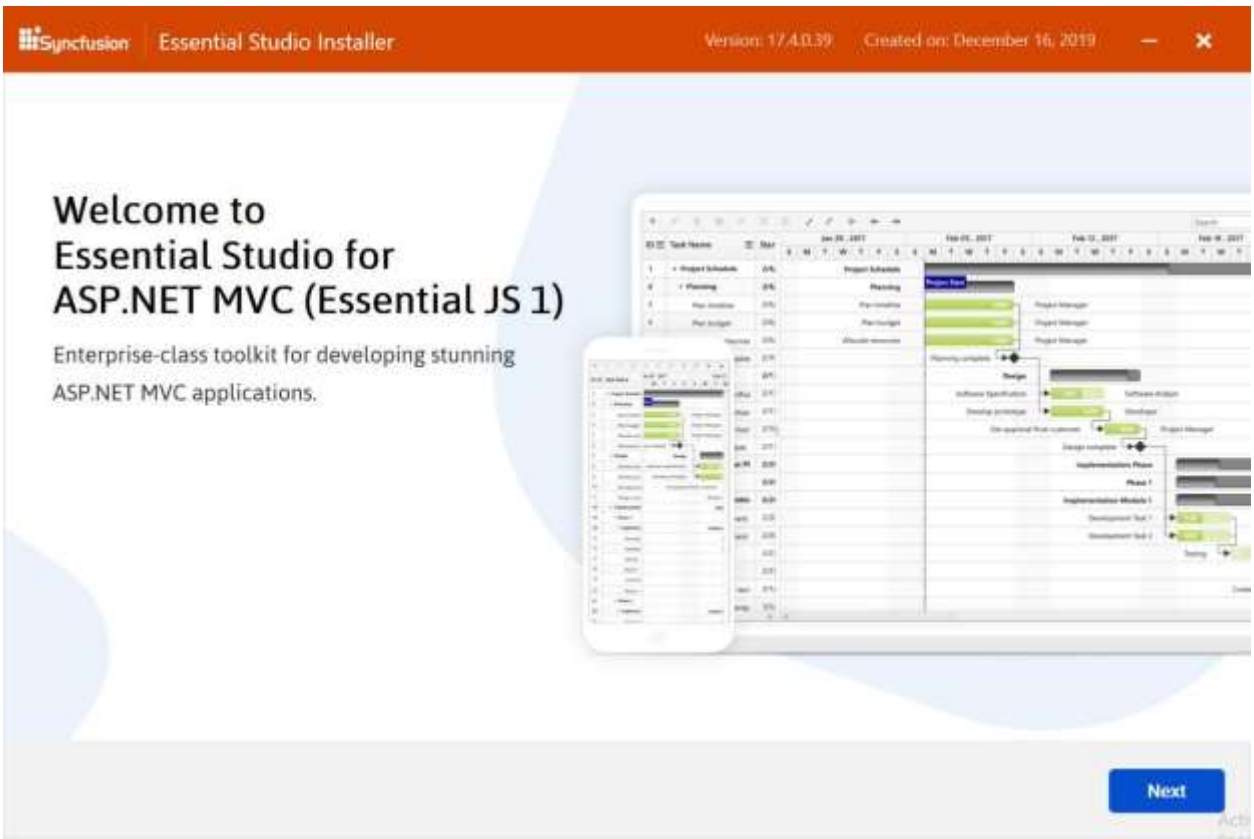
The following procedure illustrates how to uninstall Essential Studio ASP.NET MVC from Web Installer.

1. Double-click the Syncfusion ASP.NET MVC Online Installer file. The installer Wizard opens and extracts the package automatically.

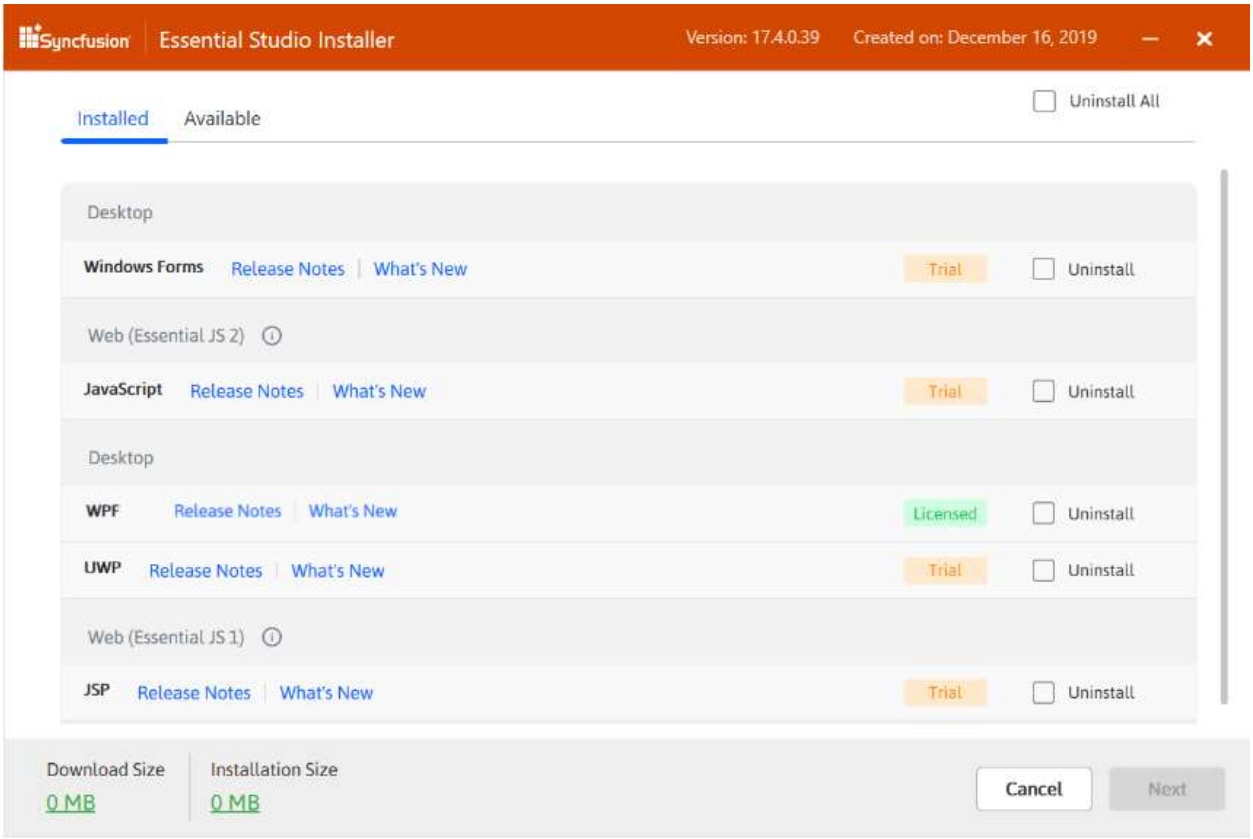
Copyright © 2001 - 2020 Syncfusion Inc.



2. Welcome wizard of the Syncfusion ASP.NET MVC Online Installer will be displayed. Click Next.

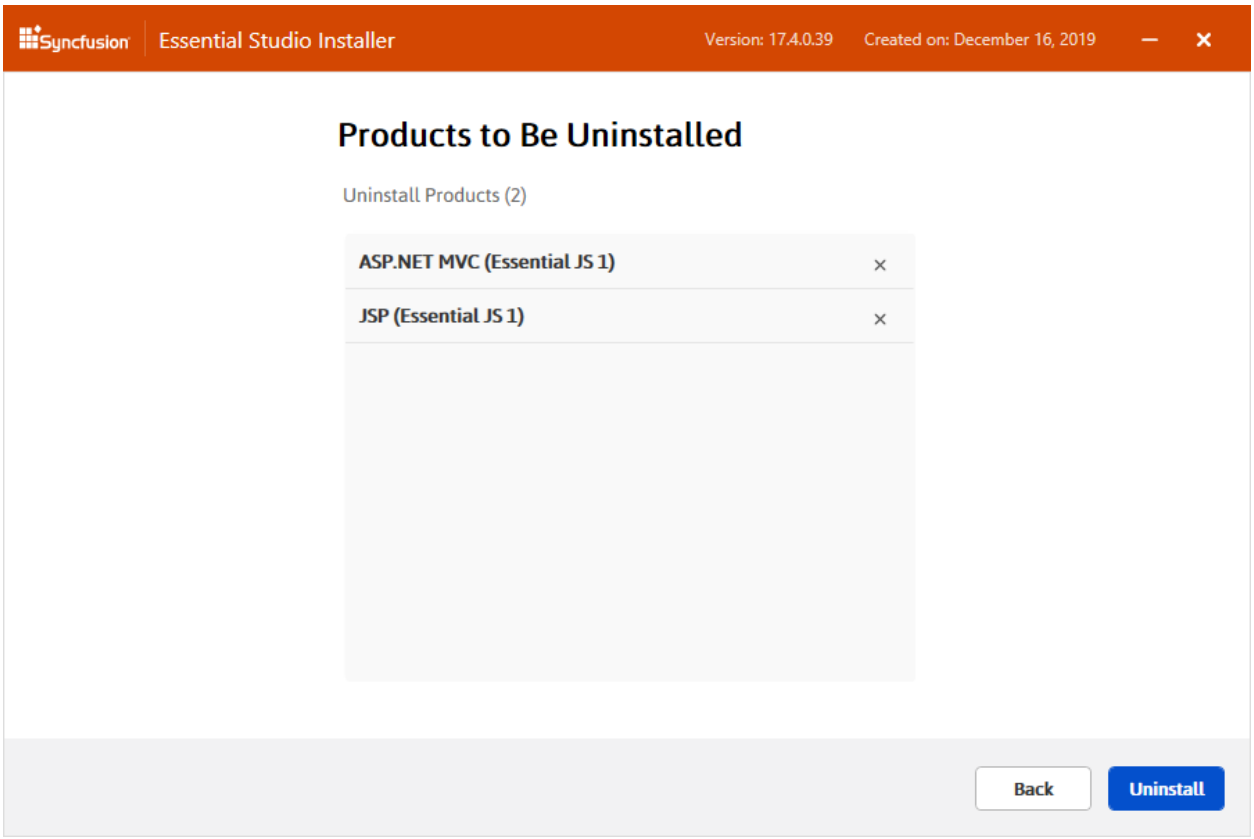


3. Platform Selection wizard will be displayed. Select the platforms to be uninstalled from the **Installed** tab. Select **Uninstall All** checkbox to select all the platforms. Click Next.



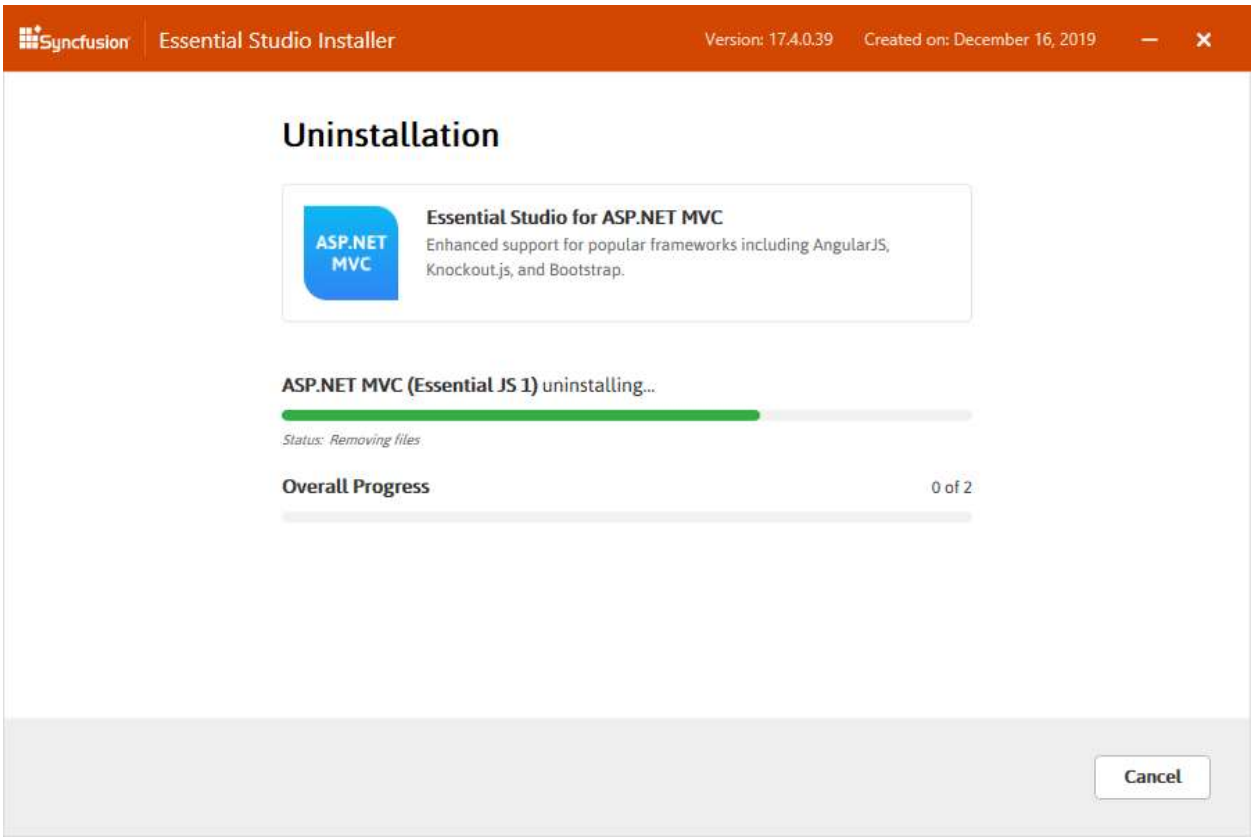
4. Confirmation wizard will be displayed. Here you can view and modify the list of platforms that will be uninstalled. Click Uninstall.

Copyright © 2001 - 2020 Syncfusion Inc.



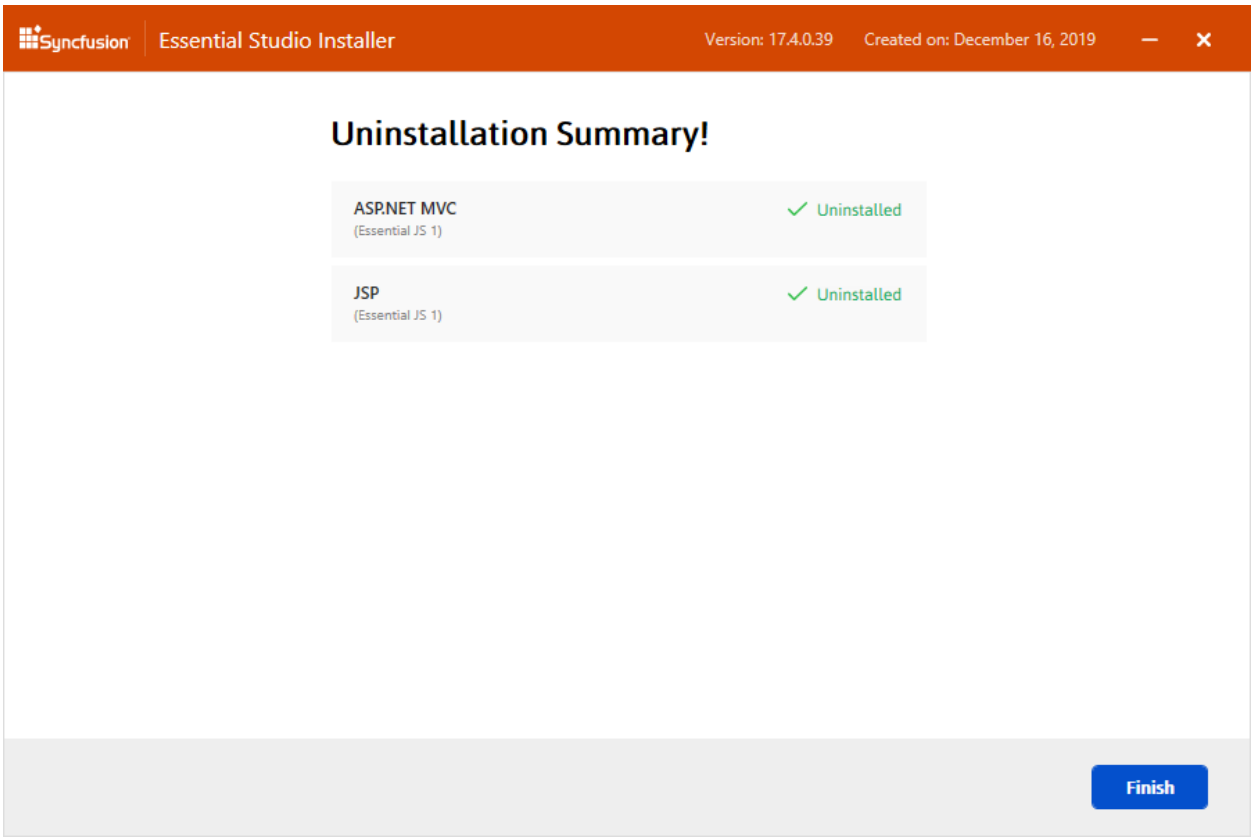
5. Uninstallation progress will be displayed.

Copyright © 2001 - 2020 Syncfusion Inc.



6. Once the Uninstallation is complete, **Summary** wizard will be displayed. Here you can check the list of platforms which are uninstalled successfully and failed. Click Finish to exit the Summary wizard.

Copyright © 2001 - 2020 Syncfusion Inc.



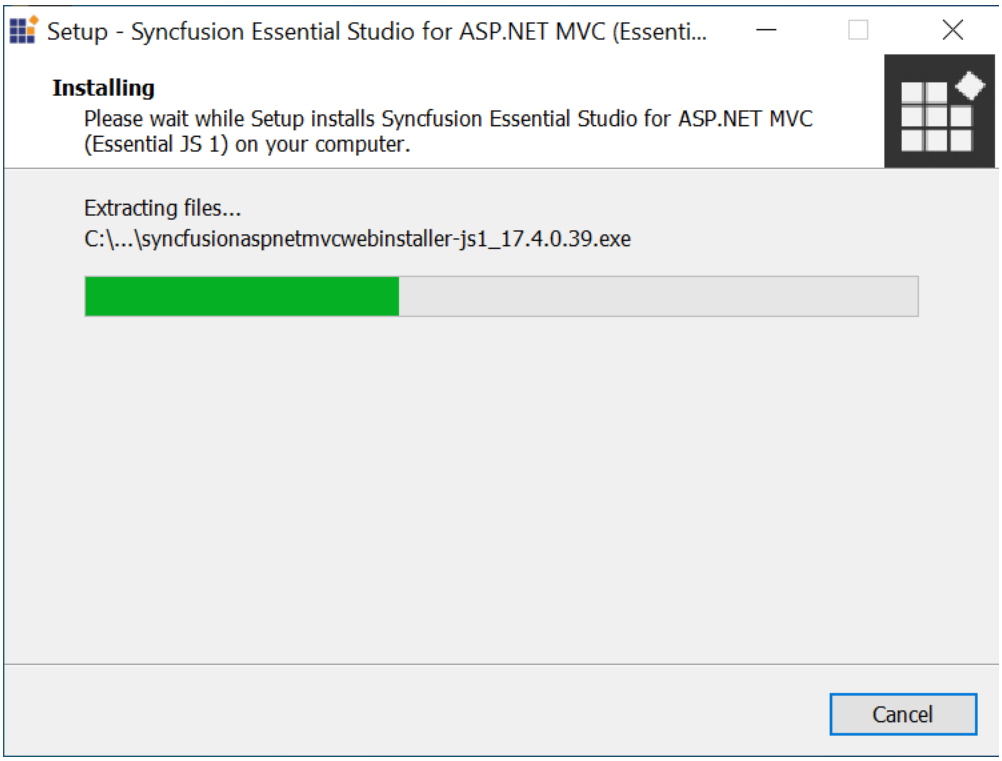
Installation and Uninstallation

Till version 17.2, Syncfusion Web Installer had option for installation alone. Starting with the version 17.3 (2019 Vol 3), Syncfusion provides option for both install and uninstall the platforms of the same version from the Web Installer application itself.

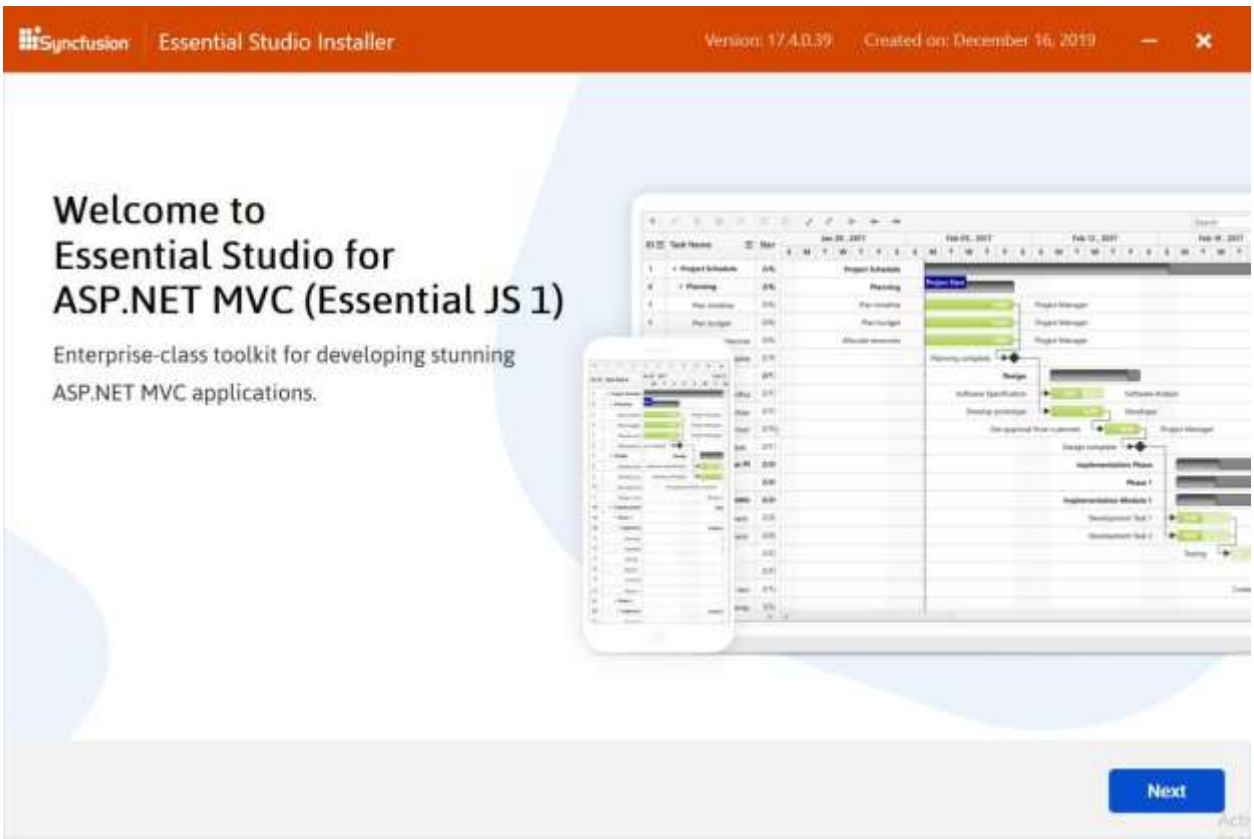
The following procedure illustrates how to install/uninstall Essential Studio ASP.NET MVC from Web Installer.

1. Double-click the Syncfusion ASP.NET MVC Online Installer file. The installer Wizard opens and extracts the package automatically.

Copyright © 2001 - 2020 Syncfusion Inc.

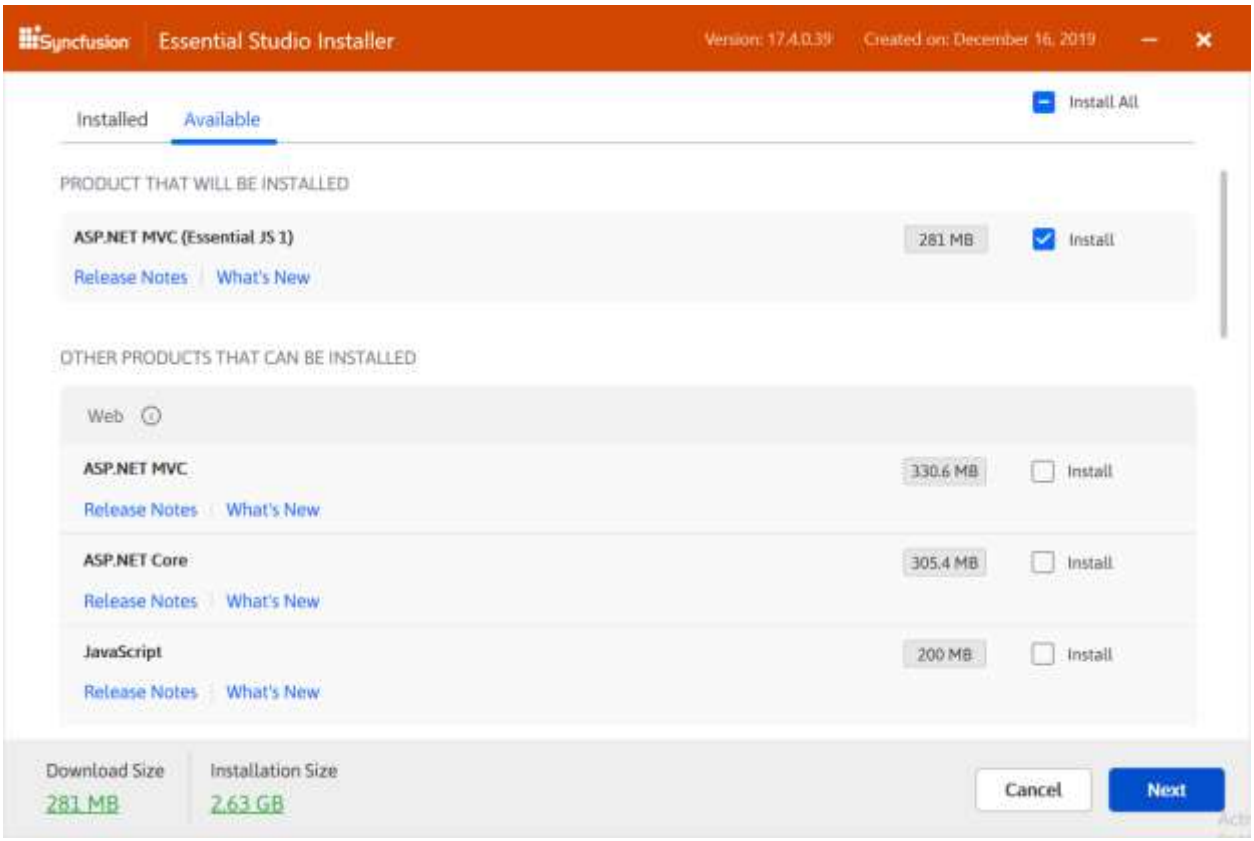


2. Welcome wizard of the Syncfusion ASP.NET MVC Online Installer will be displayed. Click Next.

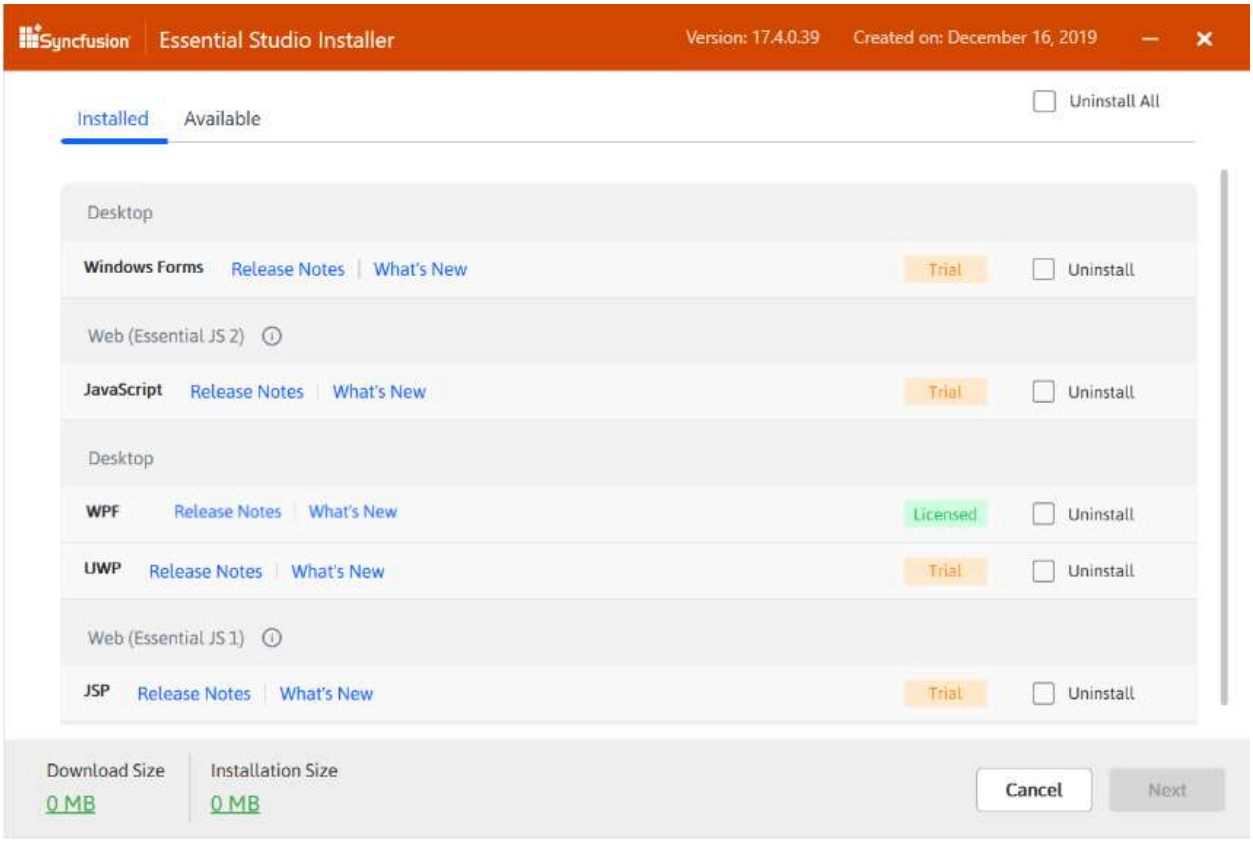


3. Platform Selection wizard will be displayed. Select the platforms to be installed from the **Available** tab and platforms to be uninstalled from the **Installed** tab. Click Next.

Available

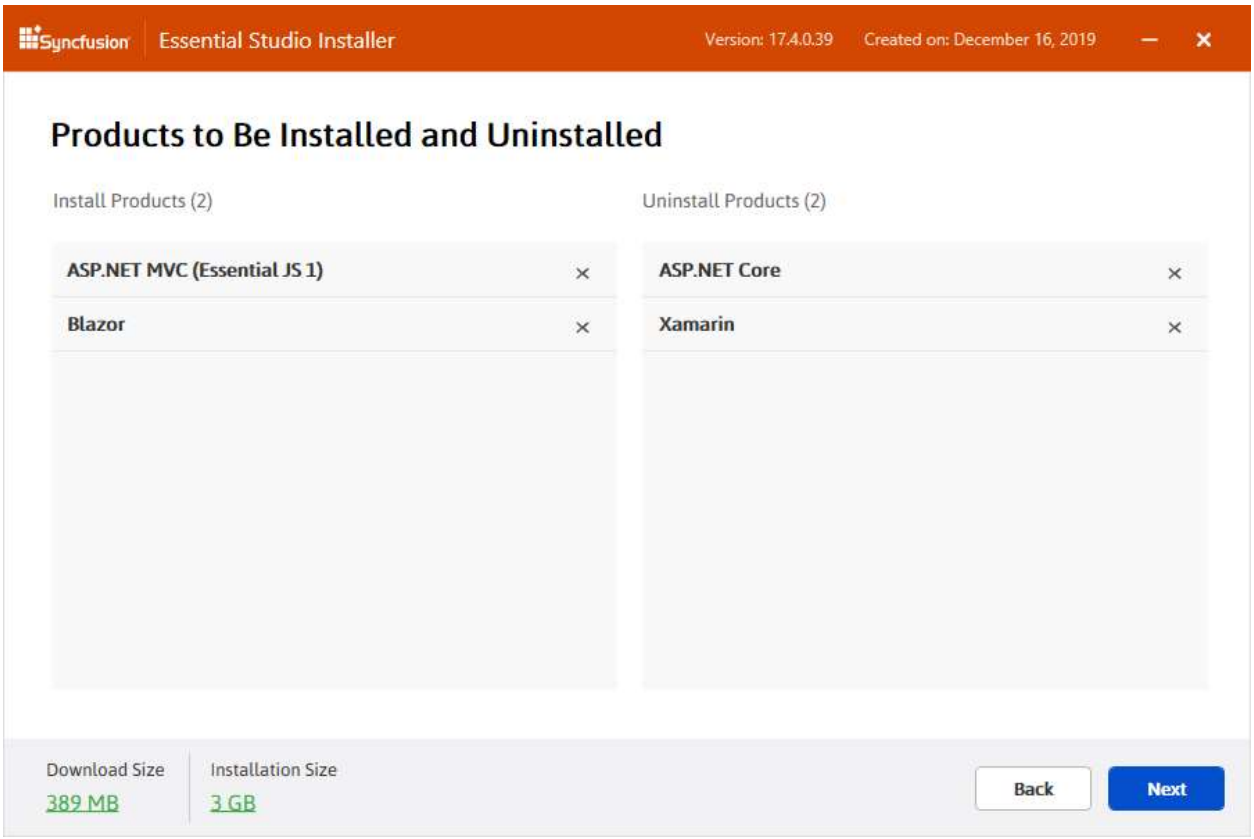


Installed

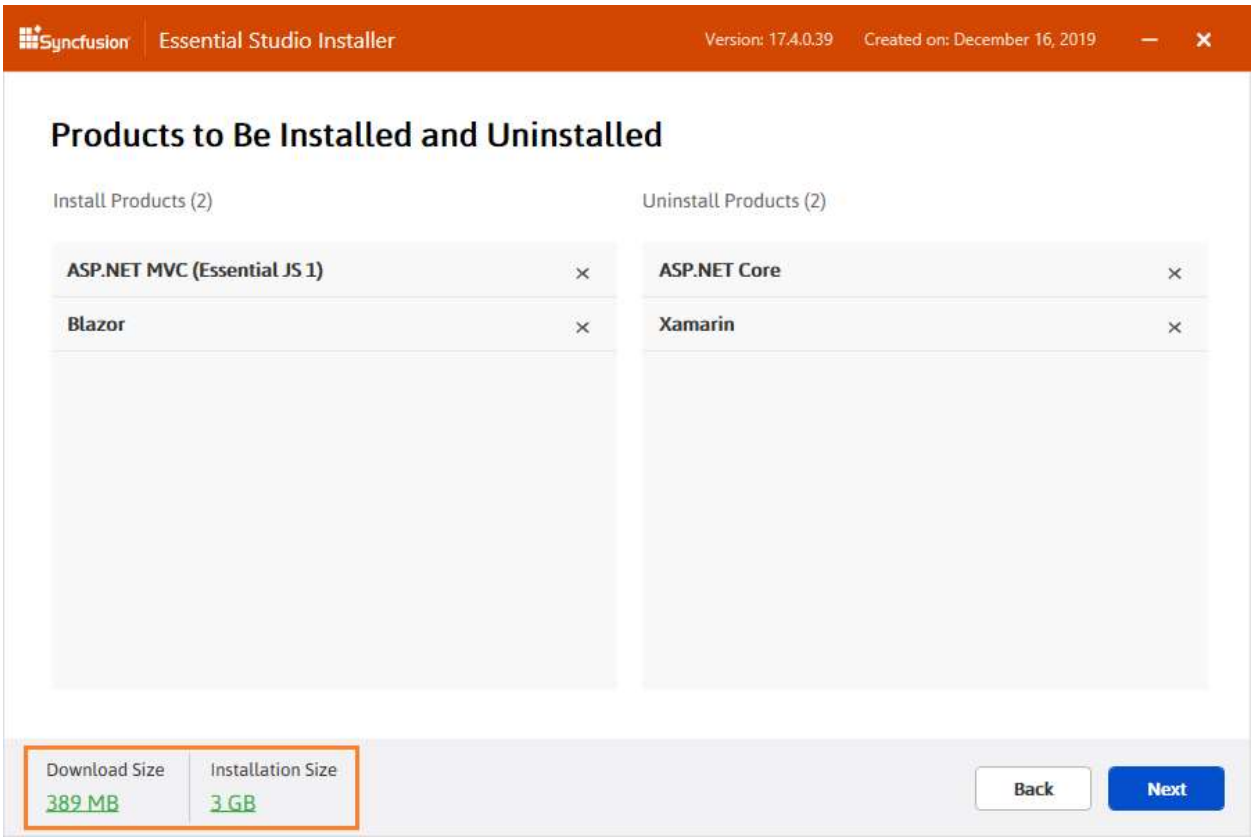


4. Confirmation wizard will be displayed. Here you can view and modify the list of platforms that will be installed/uninstalled.

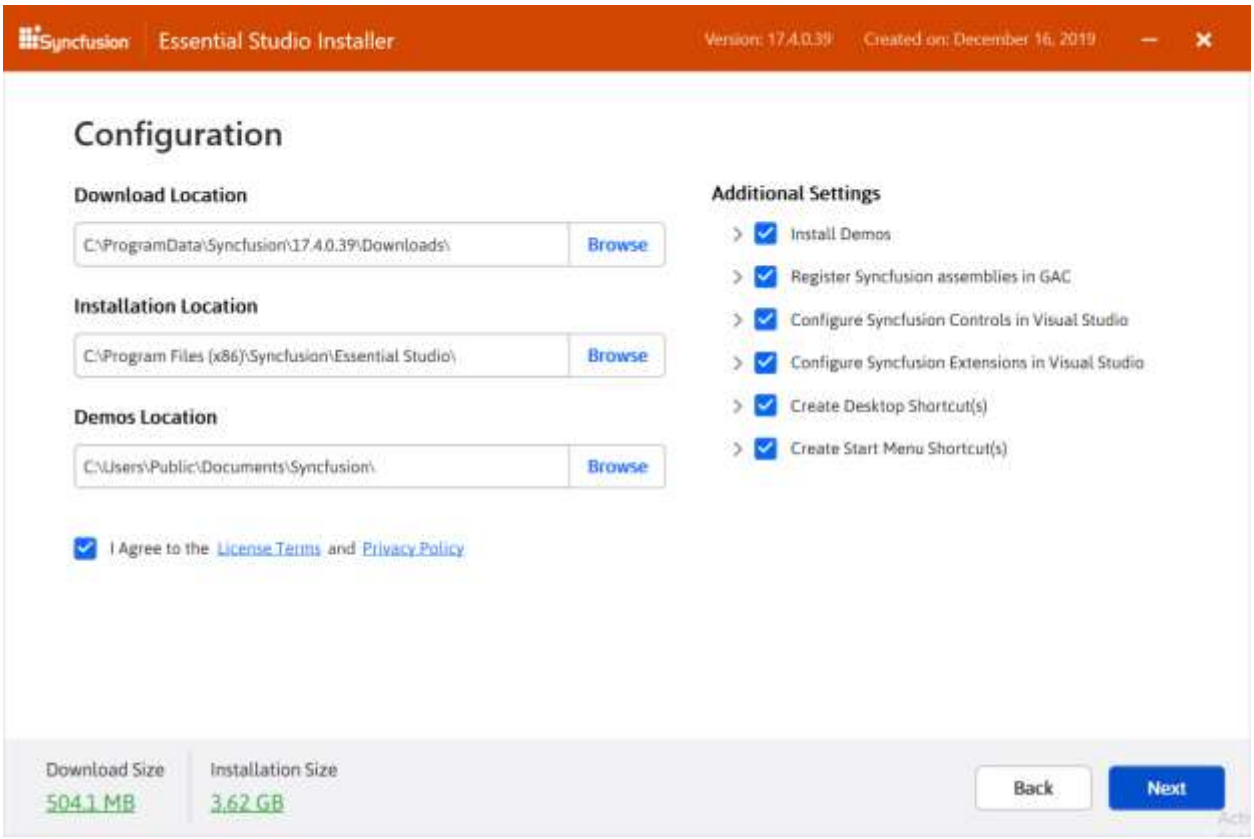
Copyright © 2001 - 2020 Syncfusion Inc.



Note: You can check the Estimated size of the Download and Installation by clicking the **Download Size and Installation Size** link.



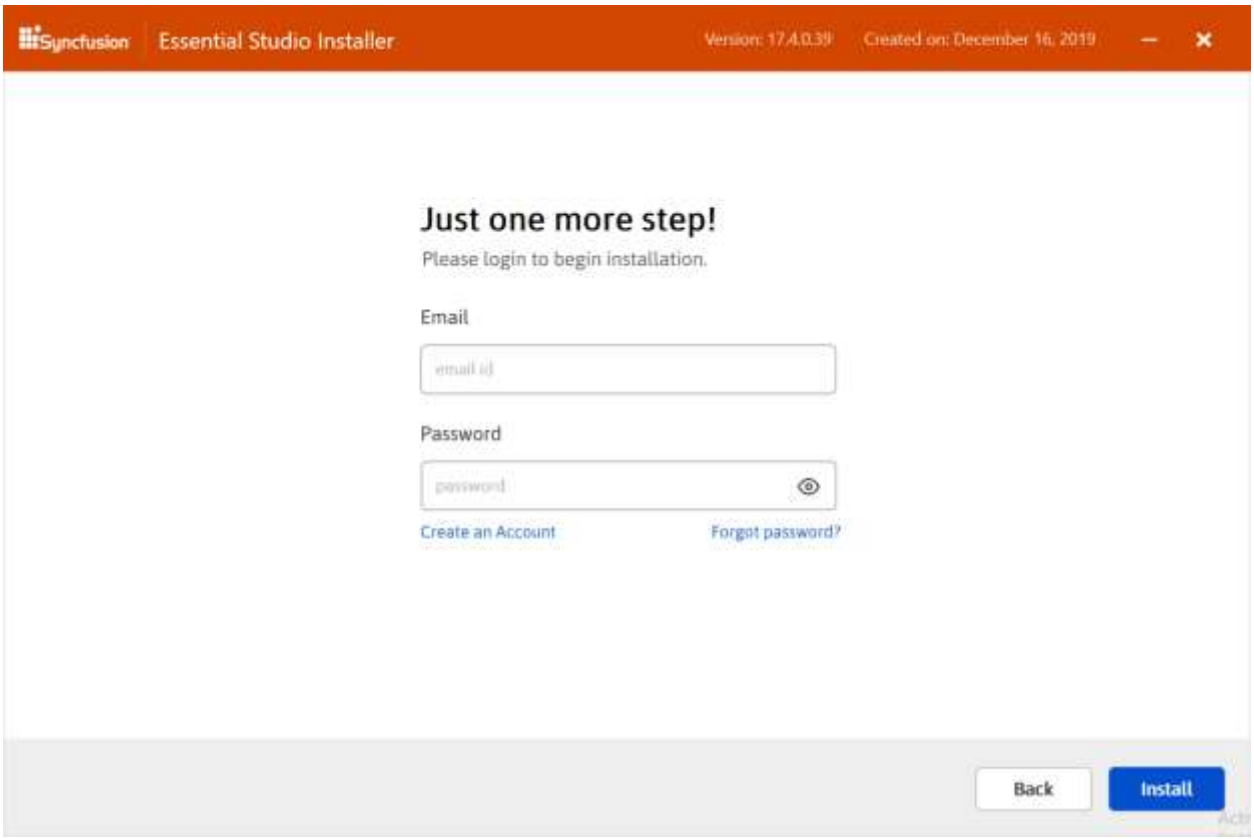
5. Configuration wizard will be displayed. Here you can change the Download, Install and samples location. Also, you can change the Additional settings by platform basis. To install using the default configuration, click Next.



Information: From version 17.3 (2019 Vol 3), Syncfusion provides option to provide custom download location.

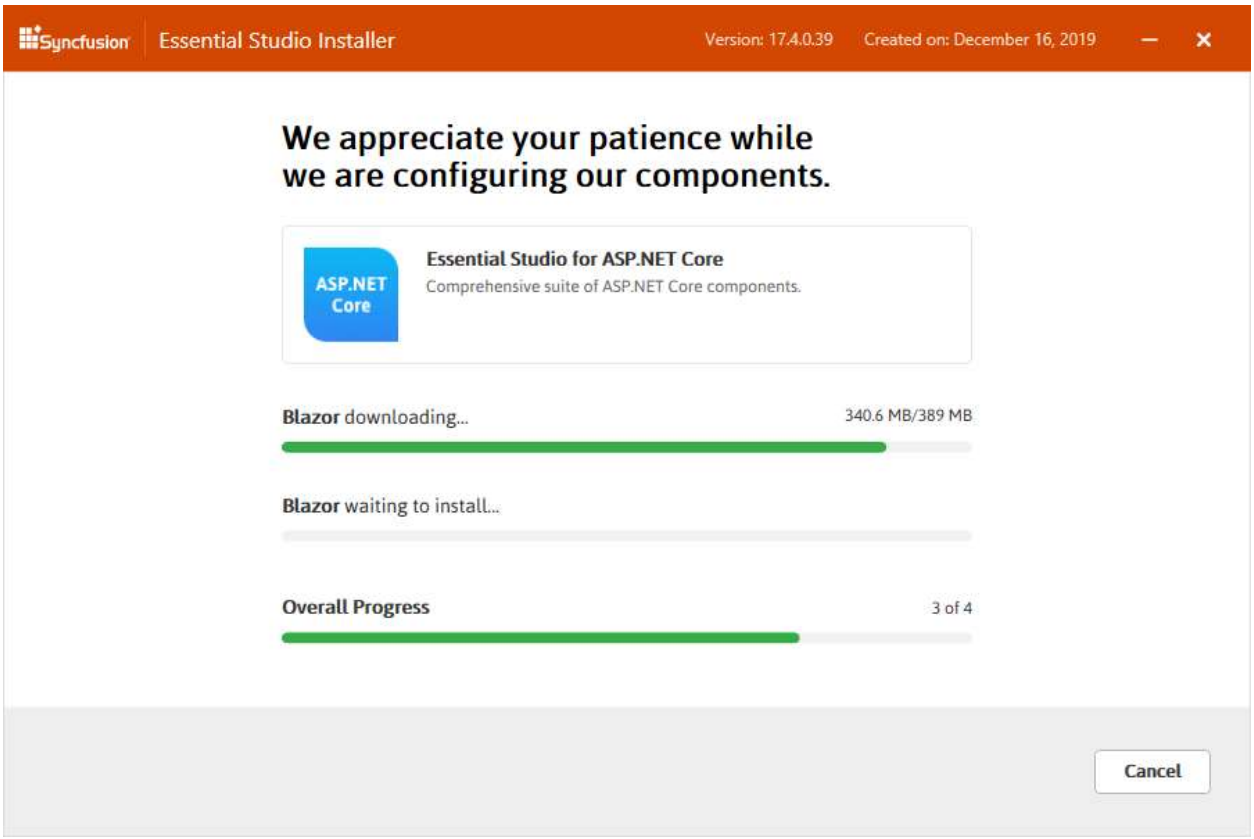
6. After reading the License Terms and Conditions, check the **I agree to the License Terms and Privacy Policy** check box. Click Next.
7. Login wizard will be displayed. You should enter your Syncfusion Direct-Trac login credentials. If you don't have Syncfusion Direct-Trac login credentials, then you can click on **Create an Account**. Else if you forgot your password, click on **Forgot Password** to create new password. Click Install.

Copyright © 2001 - 2020 Syncfusion Inc.



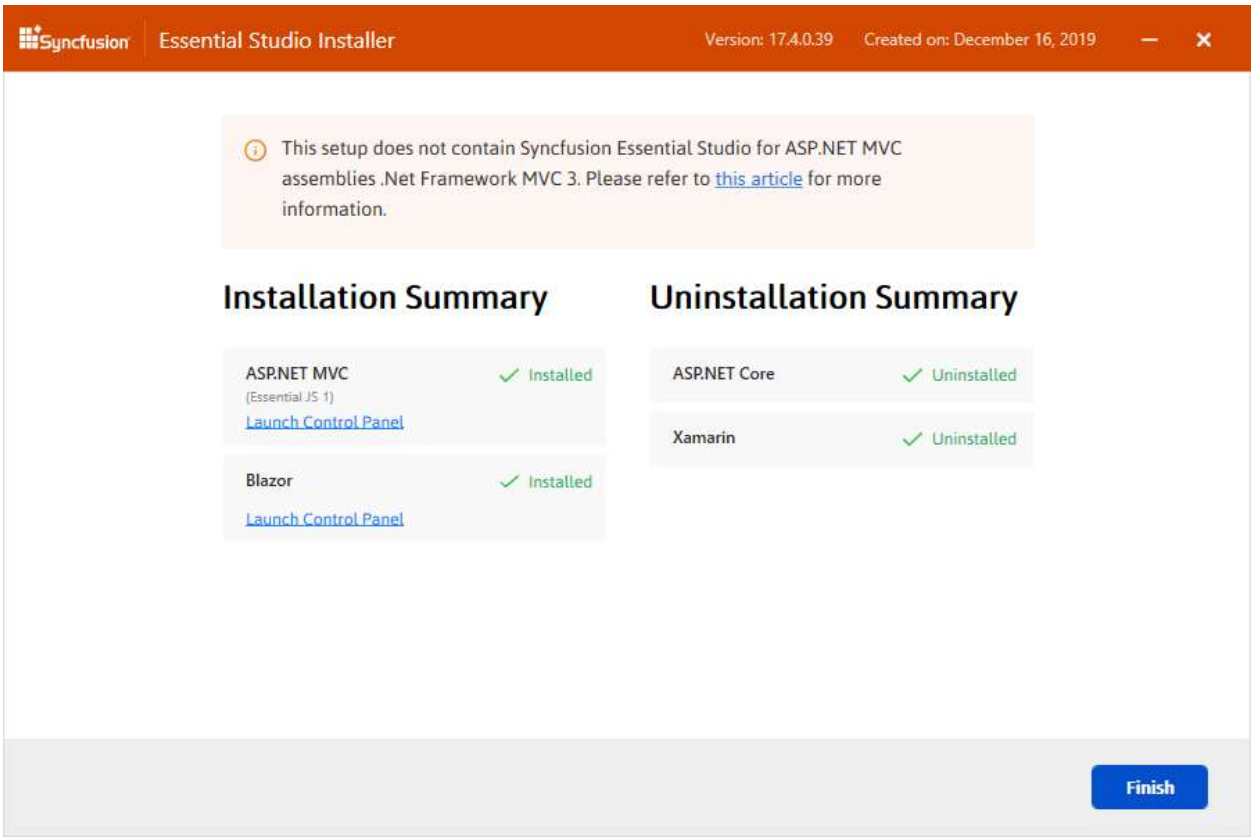
Information: The selected platforms will be installed based on your Syncfusion License (Trial or Licensed).

8. Download, Installation and Uninstallation progress will be displayed.



9. Once the Installation is complete, **Summary** wizard will be displayed. Here you can check the list of platforms which are **installed/uninstalled** successfully and failed. Click Finish to exit the Summary wizard.

Copyright © 2001 - 2020 Syncfusion Inc.



- Click **Launch Control Panel** to open the Syncfusion Control Panel.

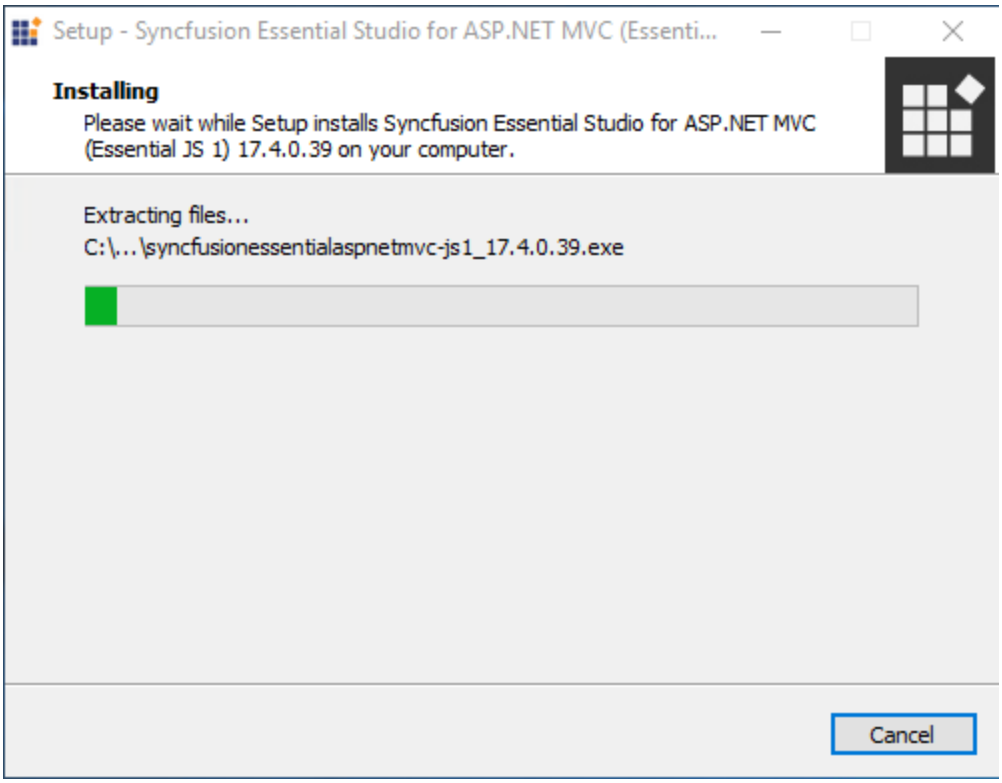
Installation using Offline Installer

Installing with UI

The following procedure illustrates how to install Essential Studio ASP.NET MVC platform.

1. Close all the running Visual Studio instances.
2. Double-click the Syncfusion ASP.NET MVC platform installer file. The installer Wizard opens and extracts the package automatically.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: The installer wizard extracts the syncfusionessentialaspnetmvc_(version).exe dialog, displaying the unzip operation of the package.

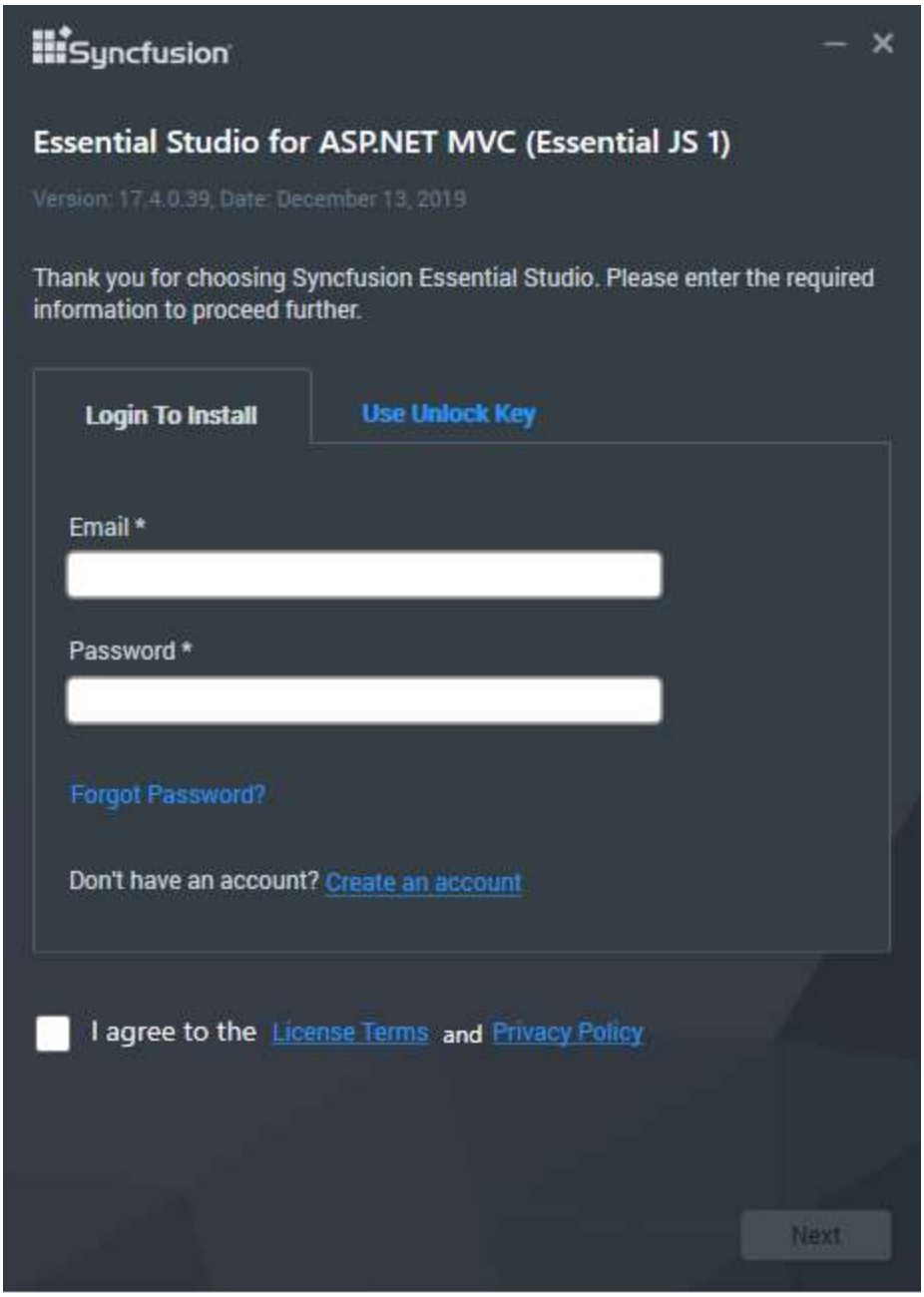
3. You have two options to unlock the Syncfusion installer:

- *Login To Install*
- *Use Unlock Key*

Login To Install

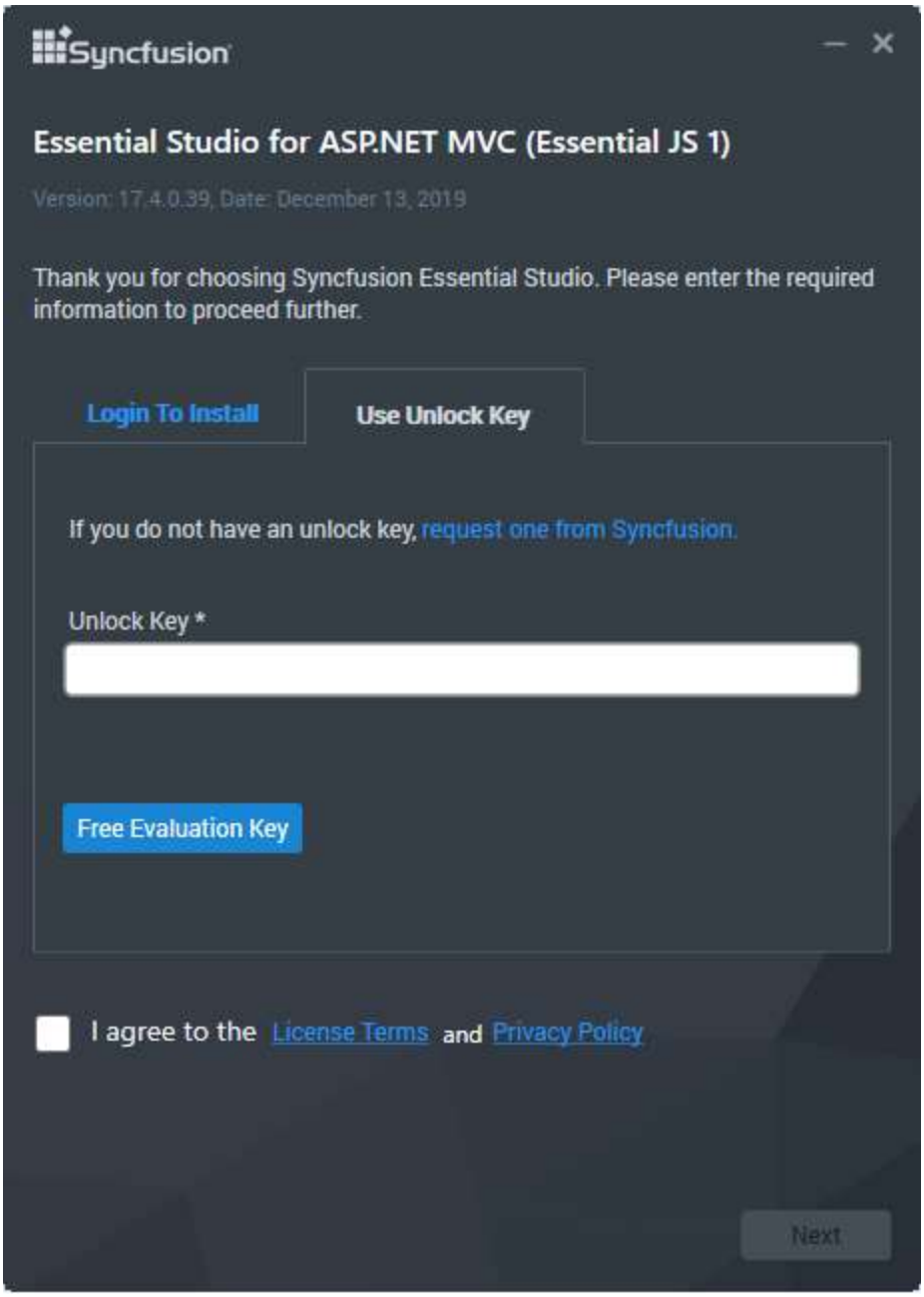
You should enter your Syncfusion Direct-Trac login credentials. If you don't have Syncfusion Direct-Trac login credentials, then you can click on Sign Up to a create new account. Else if you forgot your password, click on Reset Password to create new password. Here Email address and Password is validated and the Platform Selection window opens.

Copyright © 2001 - 2020 Syncfusion Inc.



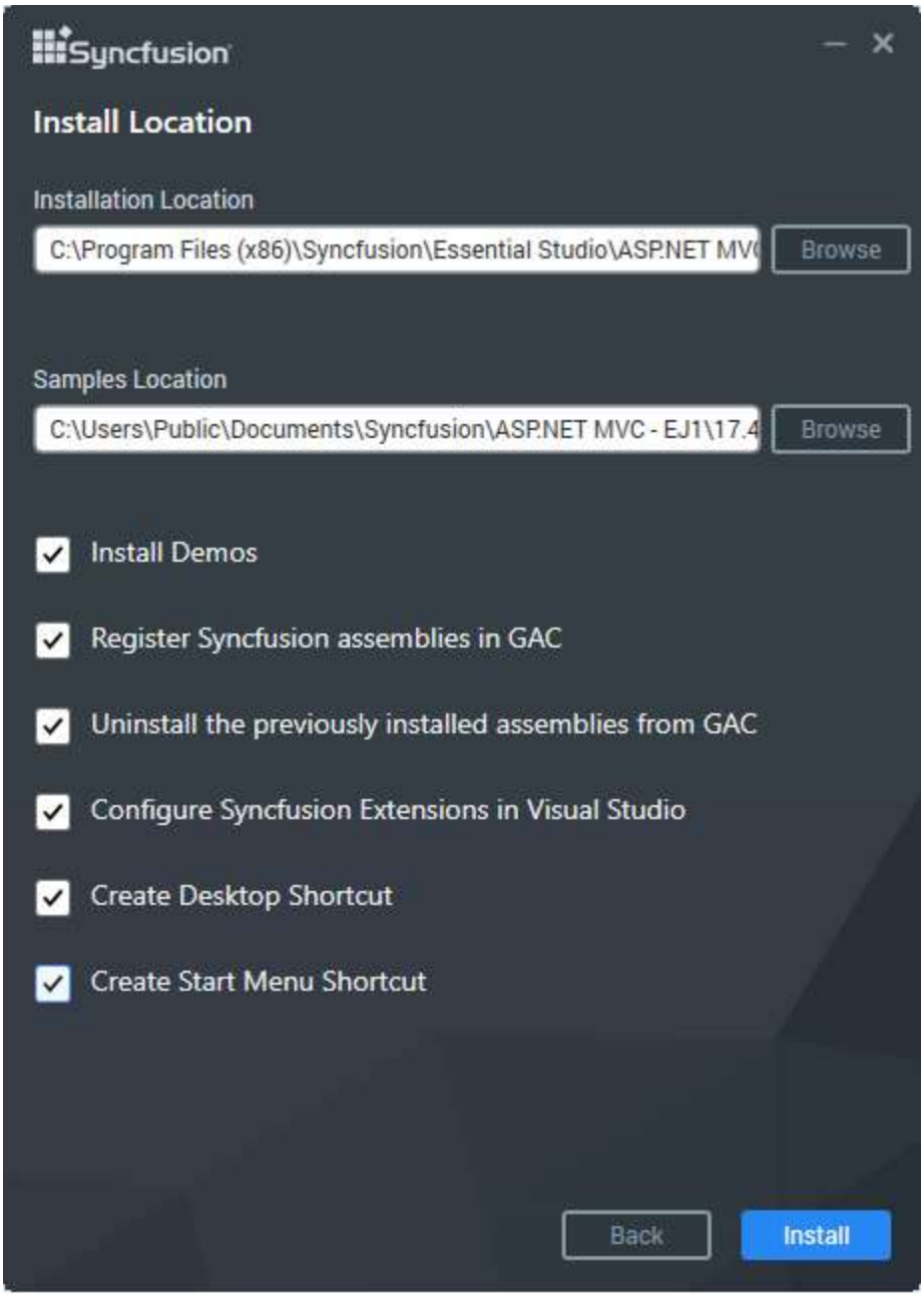
Use Unlock Key

You should use the Syncfusion License/Trial key. Trial key is valid for 30 days and the installer won't accept the expired trial key. Licensed customer can generate key from here.



4. After reading the License Terms and Conditions, check the **I agree to the License Terms and Conditions** check box.

5. Click Next. Select the Installation, Samples Folder and Advanced Options screen opens. To install in the displayed default location, click Install



Note: From the 2018 Volume 2 release, Syncfusion has changed the install and samples location

Default Install location: {ProgramFilesFolder}\Syncfusion\{Platform}\{version}

Default Samples location: C:\Users\Public\Documents\Syncfusion\{platform}\{version}

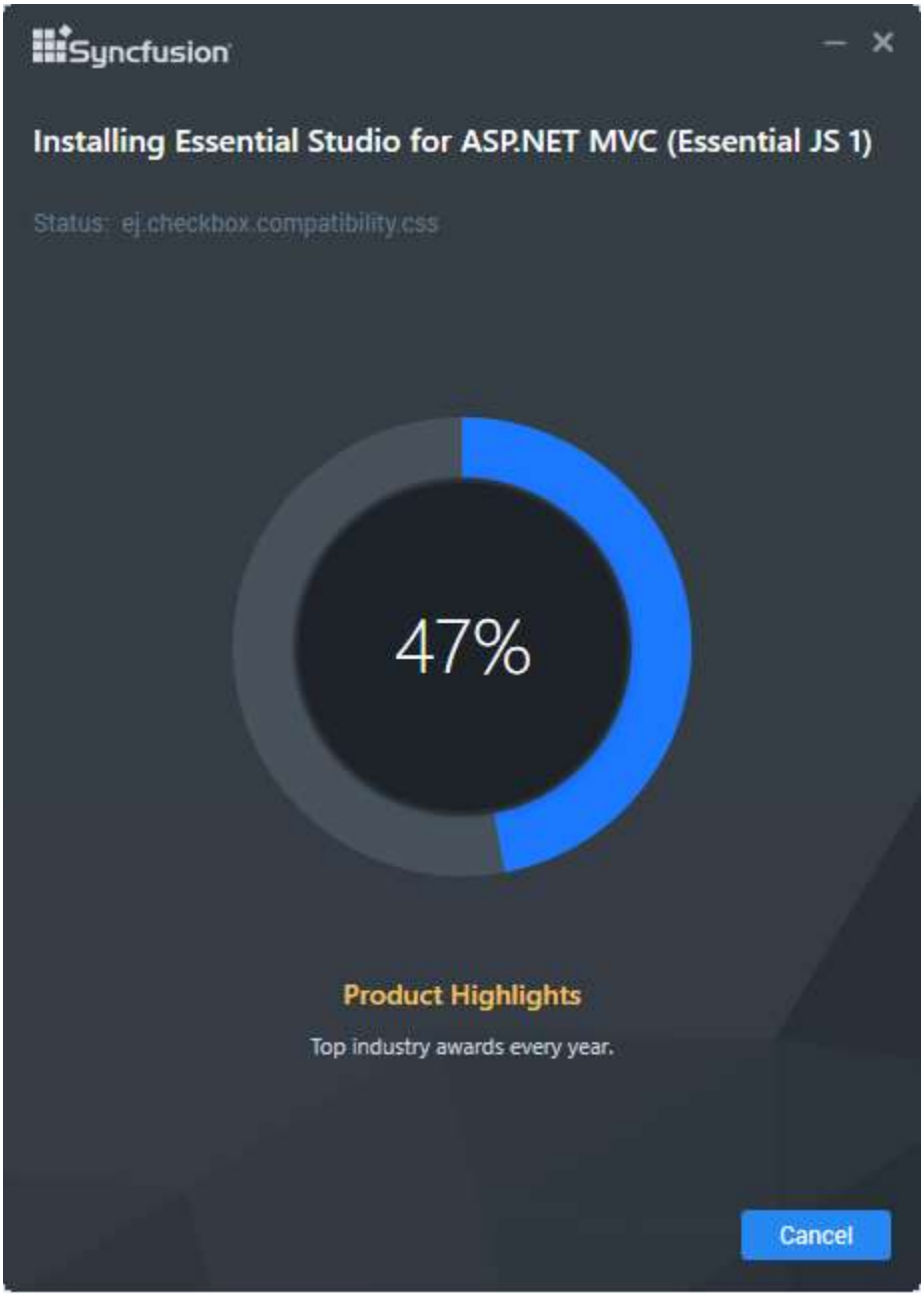
However, you can change the locations by clicking browse button.

- Select the **Install Syncfusion Samples** check box to install Syncfusion samples, or leave the check box clear, when you do not want to install Syncfusion samples.
- Select the **Register Syncfusion Assemblies in GAC** check box to install the latest Syncfusion assemblies in GAC, or clear this check box when you do not want to install the latest assemblies in GAC.

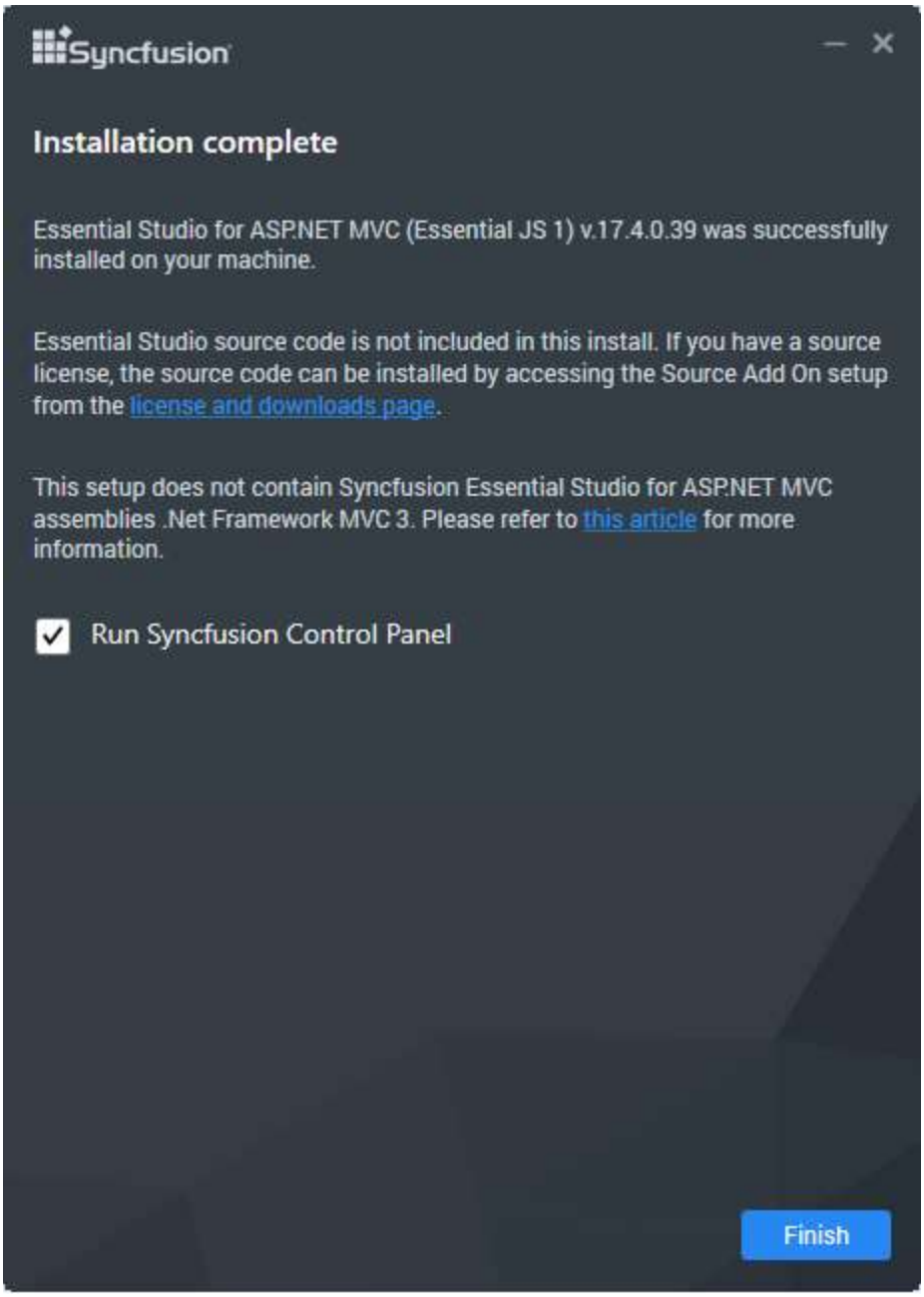
Copyright © 2001 - 2020 Syncfusion Inc.

- Select the **Uninstall the previously installed Syncfusion assemblies from GAC** check box to uninstall the previously installed Syncfusion assemblies from GAC, or clear this check box to maintain the previously installed assemblies.
- Select the **Install Syncfusion Extensions** checkbox to configure the Syncfusion Extensions in Visual Studio or clear this check box when you do not want to configure the Syncfusion Extensions in Visual Studio.

6. Click Install.



7. The Completed screen is displayed once the ASP.NET MVC platform is installed.



8. Select the **Run Syncfusion Control Panel** check box to launch the Syncfusion Control Panel after installing.

9. Click Finish. Syncfusion ASP.NET MVC platform is installed in your system and the Syncfusion Essential Studio Syncfusion Control Panel is launched automatically.

Installing in silent mode

The Syncfusion Essential Studio Platform Installer supports installing/uninstalling through Command Line. The following sections illustrate this ability.

Command Line Installation

Follow the steps below to install through Command Line in Silent mode.

1. Double-click the Syncfusion Essential Studio platform installer file. The installer Wizard opens and extracts the package automatically.
2. The syncfusionessentialaspnetmvc_(version).exe file is extracted into the Temp folder.
3. Run %temp%. The Temp folder will open. The syncfusionessentialaspnetmvc_(version).exe file is available in one of the folders.
4. Copy the extracted syncfusionessentialaspnetmvc_(version).exe file in local drive.
5. Cancel the Wizard.
6. Open the Command Prompt in administrator mode and pass the following arguments.

Arguments: “Installer file path\SyncfusionEssentialStudio(platform)_(version).exe” /Install silent /PIDKEY:“(product unlock key)” [/log “{Log file path}”] [/InstallPath:{Location to install}] [/InstallSamples:{true/false}] [/InstallAssemblies:{true/false}] [/UninstallExistAssemblies:{true/false}] [/InstallToolbox:{true/false}]

Note: [..] – Arguments inside the square brackets are optional.

Example: “D:\Temp\syncfusionessentialaspnetmvc*x.x.x.x.exe” /Install silent /PIDKEY:“product unlock key” /log “C:\Temp\EssentialStudio*Platform.log” /InstallPath:C:\Syncfusion\x.x.x.x /InstallSamples:true /InstallAssemblies:true /UninstallExistAssemblies:true /InstallToolbox:true

7. Essential Studio for ASP.NET MVC (Essential JS1) is installed.

Note: x.x.x.x needs to be replaced with the Essential Studio version and the Product Unlock Key needs to be replaced with the Unlock Key for that version.

Command Line Uninstallation

Syncfusion Essential Studio supports uninstalling through Command Line in Silent mode. The following steps illustrate this.

1. When you do not have the extracted installer (syncfusionessentialaspnetmvc_(version).exe) then follow the steps from 2 to 7.
2. Double-click the Syncfusion Essential Studio platform installer file. The installer Wizard opens and extracts the package automatically.
3. The syncfusionessentialaspnetmvc_(version).exe file is extracted into the Temp folder.
4. Run %temp%. The Temp folder will open. The syncfusionessentialaspnetmvc_(version).exe file is available in one of the folders.
5. Copy the syncfusionessentialaspnetmvc_(version).exe file in local drive.
6. Cancel the Wizard.
7. Open the Command Prompt in administrator mode and pass the following arguments.

Arguments: “Copied installer file path\syncfusionessentialaspnetmvc_(version).exe” /uninstall silent

Example: “D:\Temp\syncfusionessentialaspnetmvc_x.x.x.x.exe" /uninstall silent

8. Essential Studio for ASP.NET MVC (Essential JS1) is uninstalled.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: x.x.x.x needs to be replaced with the Essential Studio version installed in your machine.

Installation FAQ

Refer this topic for more information regarding the issues related to installation.

Upgrade from one version to another version.

You can upgrade to the latest version by downloading and installing the platforms you require from this link. After upgrading to the latest version, instead of upgrading your projects manually, you can use the Syncfusion Project Migration for migrating your projects.

ASP.NET MVC	https://help.syncfusion.com/extension/aspnet-mvc-extension/project-migration

Upgrade from major version to service pack version

Syncfusion provides a new Volume release once in every three months which has exciting new features. For that Volume release, there may be one or two Service Pack releases. The issues in the Volume release will be addressed in the Service Pack releases. You can download and install the latest Service Pack installer here.

It is not required to install the Volume release before installing the Service Pack release. As Volume and Service Packs releases works independently, you can directly install the latest Service Pack which contains major issue fixes.

Upgrade from trial version to license version

To upgrade from trial version, there are two possible solutions.

- Uninstall the trial version and install the fully licensed build from the License & Downloads section of our website.
- Replace the currently used trial license key with a paid license key that can be generated from the License & Downloads section of our website. Refer to this topic for more information regarding registering the license in the application.

Note: License registration is not required if you reference Syncfusion assemblies from Licensed installer. These licensing changes applicable to all evaluators who refers the Syncfusion assemblies from evaluation installer and those who use Syncfusion NuGet packages from nuget.org.

Configuration and Deployment

Configuring Syncfusion Bower Packages

Overview

The Bower is a package manager for the Web. Syncfusion Bower package allows to use the Syncfusion JavaScript Widgets in an efficient way.

Information: Syncfusion JavaScript Bower package is available in public Git Repository and also registered as Syncfusion-JavaScript in the Bower registry.

Bower Installation

To configure the Bower in your machine you need to install node, npm and git. For more information to configure the Bower package, please refer to the official site for bower.

Syncfusion JavaScript Bower package can be configured in the following ways.

1. Using command prompt.
2. Using bower.json file.
3. From local directory.

Using command prompt

Perform the following steps to install the Syncfusion Bower Package via command prompt in your web application.

1. Open your web project's location in a command prompt window.
2. Then run the command Bower install .

Visual Studio Integration

Visual Studio Extensions

HTML

```
<!doctype html>
<html>
<head>
<title>Essential Studio for JavaScript : EJ Icon Package</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0"
charset="utf-8"  />
<link href="Content/ej/web/default-theme/ej.web.all.min.css"
rel="stylesheet" />
</head>
<body>
<div class="icons">
<ul>
<li>
<div class="e-icon e-search"></div>
</li>
<li>
<div class="e-icon e-settings"></div>
</li>
<li>
<div class="e-icon e-upload"></div>
</li>
<li>
<div class="e-icon e-font"></div>
</li>
</ul>
<ul>
<li>
<div class="e-icon e-indent"></div>
</li>
<li>
<div class="e-icon e-outdent"></div>
</li>
<li>
<div class="e-icon e-redo"></div>
</li>
<li>
<div class="e-icon e-undo"></div>
</li>
```

```
</ul>
</div>
</body>
</html>
```

CSS

```
<style type="text/css" class="cssStyles">
div.icons li
{
display: inline-block;
vertical-align: middle;
padding: 20px;
list-style: none;
}
</style>
```

Note: Make sure the CSS file (ej.widgets.all.min.css/ej.web.all.min.css) is referred in your application.

Execute the above code to render the following output.



Step 3:

The complete list of EJ icons are listed in the following table. You have to use the following class names prefix with e- in your application.

<style type="text/css" class="cssStyles">

table {

width: 730px;

}

.txt{

margin-top: 5px;

text-align:center;

}

.spriteimage {

background-image: url('IconLibrary_images/spritesheet.png');

height: 50px;

width: 50px;

```
background-repeat: no-repeat;
display: block;
margin: 0 auto;
}
.spriteimage.shrink {
background-position: -13px -6px;
}
.spriteimage.enlarge {
background-position: -576px -389px;
}
.spriteimage.key {
background-position: -859px -830px;
}
.spriteimage.list {
background-position: -489px -54px;
}
.spriteimage.listnum {
background-position: -890px -675px;
}
.spriteimage.listnum_01 {
background-position: -213px -885px;
}
.spriteimage.indent {
background-position: -844px -896px;
}
.spriteimage.indent_01 {
background-position: -963px -501px;
}
.spriteimage.outdent {
background-position: -3px -950px;
}
.spriteimage.outdent_01 {
background-position: -76px 3px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.spriteimage.close {
background-position: -661px -2px;
float: none;
opacity: 3.2;
}
.spriteimage.close_01 {
background-position: 5px -103px;
}
.spriteimage.undo {
background-position: -650px -103px;
}
.spriteimage.undo_01 {
background-position: -337px -159px;
}
.spriteimage.redo {
background-position: -146px -216px;
}
.spriteimage.redo_01 {
background-position: -638px -213px;
}
.spriteimage.video {
background-position: -800px -263px;
}
.spriteimage.video_01 {
background-position: -472px -320px;
}
.spriteimage.cross {
background-position: -197px -379px;
}
.spriteimage.brush {
background-position: -754px -380px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.dcol {
background-position: 0px -449px;
}
.spriteimage.dcol_01 {
background-position: -218px -495px;
}
.spriteimage.drow {
background-position: -720px -561px;
}
.spriteimage.drow_01 {
background-position: -379px -615px;
}
.spriteimage.icolleft {
background-position: -108px -658px;
}
.spriteimage.icolleft_01 {
background-position: -171px -720px;
}
.spriteimage.icolleft_02 {
background-position: -3px -779px;
}
.spriteimage.icolright {
background-position: -652px -775px;
}
.spriteimage.icolright_01 {
background-position: -721px -831px;
}
.spriteimage.icolright_02 {
background-position: -912px 1px;
}
.spriteimage.irowbelow {
background-position: -904px -45px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.irowbelow_01 {
background-position: -908px -98px;
}
.spriteimage.irowbelow_2 {
background-position: -910px -157px;
}
.spriteimage.irowabove {
background-position: -66px -47px;
}
.spriteimage.irowabove_01 {
background-position: -910px -213px;
}
.spriteimage.irowabove_02 {
background-position: -916px -274px;
}
.spriteimage.table {
background-position: -170px -58px;
}
.spriteimage.table_01 {
background-position: -911px -447px;
}
.spriteimage.table_02 {
background-position: -911px -341px;
}
.spriteimage.sigma {
background-position: -906px -504px;
}
.spriteimage.sigma_01 {
background-position: -880px -557px;
}
.spriteimage.uppercase {
background-position: -264px -45px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.uppercase_01 {
background-position: -486px -152px;
}
.spriteimage.lowercase {
background-position: -766px -45px;
}
.spriteimage.lowercase_01 {
background-position: -910px -389px;
}
.spriteimage.bgcolor {
background-position: -323px -264px;
}
.spriteimage.superscript {
background-position: -880px -613px
}
.spriteimage.superscript_01 {
background-position: -153px -661px;
}
.spriteimage.subscript {
background-position: 1px -666px;
}
.spriteimage.subscript_01 {
background-position: -432px -724px;
}
.spriteimage.restore {
background-position: -915px -837px;
}
.spriteimage.upload {
background-position: -267px -661px;
}
.spriteimage.download {
background-position: -490px -724px;
}
```

```
.spriteimage.save {
background-position: -914px -726px;
}
.spriteimage.save_01 {
background-position: -4px -893px;
}
.spriteimage.save_02 {
background-position: -61px -893px;
}
.spriteimage.mail {
background-position: -316px -661px;
}
.spriteimage.arrowhead-right {
background-position: -367px -659px;
}
.spriteimage.arrowhead-right_01 {
background-position: -163px -886px;
}
.spriteimage.arrowhead-left {
background-position: -270px -893px;
}
.spriteimage.arrowhead-left_01 {
background-position: -314px -886px;
}
.spriteimage.arrowhead-up {
background-position: -365px -896px;
}
.spriteimage.arrowhead-down {
background-position: -407px -888px;
}
.spriteimage.arrow-sans-right {
background-position: -455px -891px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.arrow-sans-right_01 {
background-position: -203px -553px;
}
.spriteimage.arrow-sans-left {
background-position: -624px -896px;
}
.spriteimage.arrow-sans-left_01 {
background-position: -673px -888px;
}
.spriteimage.arrow-sans-up {
background-position: -706px -616px;
}
.spriteimage.arrow-sans-down {
background-position: -485px -552px;
}
.spriteimage.datetime {
background-position: -723px -896px;
}
.spriteimage.datetime_01 {
background-position: -911px -888px;
}
.spriteimage.calendar {
background-position: -967px 4px;
}
.spriteimage.calendar_01 {
background-position: -963px -51px;
}
.spriteimage.calendar-plus {
background-position: -966px -105px;
}
.spriteimage.calendar-edit {
background-position: -966px -156px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.chevron-right {
background-position: -966px -214px;
}
.spriteimage.chevron-right_01 {
background-position: -969px -331px;
}
.spriteimage.chevron-right_02 {
background-position: -969px -267px;
}
.spriteimage.chevron-left {
background-position: -969px -393px;
}
.spriteimage.chevron-left_01 {
background-position: -969px -446px;
}
.spriteimage.chevron-left_02 {
background-position: -956px -550px;
}
.spriteimage.chevron-up {
background-position: -756px -608px;
}
.spriteimage.chevron-up_01 {
background-position: -570px -107px;
}
.spriteimage.chevron-down {
background-position: -936px -617px
}
.spriteimage.chevron-down_01 {
background-position: -946px -653px;
}
.spriteimage.chevron-circle-right {
background-position: -117px -54px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.chevron-circle-left {
background-position: -313px -53px;
}
.spriteimage.font {
background-position: -968px -715px;
}
.spriteimage.strikethrough {
background-position: -915px -788px;
}
.spriteimage.strikethrough_01 {
background-position: -67px -944px;
}
.spriteimage.bold {
background-position: -969px -756px;
}
.spriteimage.bold_01 {
background-position: -969px -892px;
}
.spriteimage.italic {
background-position: -972px -811px;
}
.spriteimage.italic_01 {
background-position: -120px -954px;
}
.spriteimage.underline {
background-position: -178px -944px;
}
.spriteimage.underline_01 {
background-position: -228px -951px;
}
.spriteimage.reply {
background-position: -697px -48px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.forward {
background-position: -285px -946px;
}
.spriteimage.export {
background-position: -338px -946px;
}
.spriteimage.user {
background-position: -125px 1px;
}
.spriteimage.clipboard {
background-position: -173px 1px;
}
.spriteimage.home {
background-position: -218px 3px;
}
.spriteimage.clear {
background-position: -268px -4px;
}
.spriteimage.resize-handle {
background-position: -325px 4px;
}
.spriteimage.link {
background-position: -385px 9px;
}
.spriteimage.link_01 {
background-position: -438px 9px;
}
.spriteimage.unlink {
background-position: -494px 4px;
}
.spriteimage.unlink_01 {
background-position: -548px 11px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.external-link {
background-position: -601px 10px;
}
.spriteimage.external-link_01 {
background-position: -712px 10px;
}
.spriteimage.clock {
background-position: -770px 3px;
}
.spriteimage.settings {
background-position: -818px 3px;
}
.spriteimage.cut {
background-position: -219px -45px;
}
.spriteimage.cut_01 {
background-position: -379px -44px;
}
.spriteimage.copy {
background-position: -438px -44px;
}
.spriteimage.copy_01 {
background-position: -541px -37px;
}
.spriteimage.copy_02 {
background-position: -594px -45px;
}
.spriteimage.paste {
background-position: -652px -43px;
}
.spriteimage.paste_01 {
background-position: -821px -43px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.star {
background-position: -44px -93px;
}
.spriteimage.pointer {
background-position: -96px -104px;
}
.spriteimage.sortdirect {
background-position: -149px -106px;
}
.spriteimage.shoppingcart {
background-position: -196px -103px;
}
.spriteimage.shoppingcart_01 {
background-position: -239px -94px;
}
.spriteimage.cursor {
background-position: -298px -107px;
}
.spriteimage.warning {
background-position: -350px -98px;
}
.spriteimage.zoom-out {
background-position: -393px -98px;
}
.spriteimage.zoom-out_01 {
background-position: -455px -98px;
}
.spriteimage.zoom-out_02 {
background-position: -513px -98px;
}
.spriteimage.zoom-in {
background-position: -697px -97px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.zoom-in_01 {
background-position: -747px -108px;
}
.spriteimage.zoom-in_02 {
background-position: -800px -108px;
}
.spriteimage.arrow-circle-left {
background-position: -853px -108px;
}
.spriteimage.arrow-circle-left_01 {
background-position: -2px -153px;
}
.spriteimage.arrow-circle-left_02 {
background-position: -63px -153px;
}
.spriteimage.arrow-circle-right {
background-position: -116px -153px;
}
.spriteimage.arrow-circle-right_01 {
background-position: -176px -156px;
}
.spriteimage.arrow-circle-right_02 {
background-position: -230px -156px;
}
.spriteimage.arrow-circle-up {
background-position: -281px -167px;
}
.spriteimage.arrow-circle-down {
background-position: -387px -159px;
}
.spriteimage.edit {
background-position: -575px -158px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.edit_01 {
background-position: -627px -158px;
}
.spriteimage.edit_02 {
background-position: -435px -158px;
}
.spriteimage.edit_03 {
background-position: -686px -158px;
}
.spriteimage.edit_04 {
background-position: -805px -158px;
}
.spriteimage.edit_05 {
background-position: -857px -158px;
}
.spriteimage.notification {
background-position: 5px -211px;
}
.spriteimage.notification_01 {
background-position: -44px -216px;
}
.spriteimage.info {
background-position: -94px -219px;
}
.spriteimage.smiley {
background-position: -196px -210px;
}
.spriteimage.checkmark {
background-position: -488px -202px;
}
.spriteimage.checkmark_01 {
background-position: -739px -206px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.media-play {
background-position: -248px -214px;
}
.spriteimage.media-pause {
background-position: -793px -207px;
}
.spriteimage.media-eject {
background-position: -300px -212px;
}
.spriteimage.media-next {
background-position: -351px -220px;
}
.spriteimage.media-previous {
background-position: -402px -219px;
}
.spriteimage.media-forward {
background-position: -537px -219px;
}
.spriteimage.media-forward_01 {
background-position: -589px -212px;
}
.spriteimage.media-forward_02 {
background-position: -843px -212px;
}
.spriteimage.media-forward_03 {
background-position: -152px -264px;
}
.spriteimage.media-forward_04 {
background-position: -213px -270px;
}
.spriteimage.media-backward {
background-position: -268px -264px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.media-backward_01 {
background-position: -402px -262px;
}
.spriteimage.media-backward_02 {
background-position: -458px -266px;
}
.spriteimage.media-backward_03 {
background-position: -514px -266px;
}
.spriteimage.media-backward_04 {
background-position: -642px -266px;
}
.spriteimage.play-circle {
background-position: -692px -277px;
}
.spriteimage.full-screen-expand {
background-position: -745px -263px;
}
.spriteimage.full-screen-expand_01 {
background-position: -1px -317px;
}
.spriteimage.full-screen-collapse {
background-position: -59px -317px;
}
.spriteimage.full-screen-collapse_01 {
background-position: -117px -317px;
}
.spriteimage.bullets {
background-position: -855px -277px;
}
.spriteimage.bullets_01 {
background-position: -173px -328px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.filter {
background-position: -230px -335px;
}
.spriteimage.filter_01 {
background-position: -283px -324px;
}
.spriteimage.filternone {
background-position: -325px -324px;
}
.spriteimage.filternone_01 {
background-position: -372px -324px;
}
.spriteimage.filter-settings {
background-position: -415px -324px;
}
.spriteimage.align-right {
background-position: -518px -324px;
}
.spriteimage.align-right_01 {
background-position: -572px -316px;
}
.spriteimage.align-left {
background-position: -623px -321px;
}
.spriteimage.align-left_01 {
background-position: -752px -321px;
}
.spriteimage.align-justify {
background-position: -802px -321px;
}
.spriteimage.align-justify_01 {
background-position: -850px -325px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.align-center {
background-position: 4px -383px;
}
.spriteimage.align-center_01 {
background-position: -47px -383px
}
.spriteimage.align-none {
background-position: -99px -383px;
}
.spriteimage.search {
background-position: -146px -379px;
}
.spriteimage.search_01 {
background-position: -244px -382px;
}
.spriteimage.image {
background-position: -296px -385px;
}
.spriteimage.image_01 {
background-position: -358px -375px;
}
.spriteimage.plus {
background-position: -469px -382px;
}
.spriteimage.plus_01 {
background-position: -691px -213px;
}
.spriteimage.minus {
background-position: -521px -373px;
}
.spriteimage.minus_01 {
background-position: -860px 13px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.code {
background-position: -421px -377px;
}
.spriteimage.code_01 {
background-position: 4px -264px;
}
.spriteimage.code-hexagon {
background-position: -683px -375px;
}
.spriteimage.file-code {
background-position: -802px -384px;
}
.spriteimage.file-html {
background-position: -856px -389px;
}
.spriteimage.palette {
background-position: -101px -433px;
}
.spriteimage.reload {
background-position: -156px -438px;
}
.spriteimage.delete {
background-position: -623px -377px;
}
.spriteimage.delete_01 {
background-position: -206px -438px;
}
.spriteimage.delete_02 {
background-position: -356px -442px;
}
.spriteimage.delete_03 {
background-position: -405px -442px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.delete_04 {
background-position: -458px -439px;
}
.spriteimage.delete_05 {
background-position: -507px -443px;
}
.spriteimage.pin {
background-position: -554px -445px;
}
.spriteimage.unpin {
background-position: -601px -439px;
}
.spriteimage.stop {
background-position: -653px -439px;
}
.spriteimage.power-cord {
background-position: -698px -443px;
}
.spriteimage.fullborders {
background-position: -853px -443px;
}
.spriteimage.threed {
background-position: 2px -503px;
}
.spriteimage.file-excel {
background-position: -52px -499px;
}
.spriteimage.file-text {
background-position: -755px -441px;
}
.spriteimage.file-text_01 {
background-position: -94px -499px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.file-mdx {
background-position: -159px -497px;
}
.spriteimage.file-empty {
background-position: -270px -499px;
}
.spriteimage.file-list {
background-position: -329px -503px;
}
.spriteimage.file-delete_01 {
background-position: -383px -502px;
}
.spriteimage.file-settings {
background-position: -432px -503px;
}
.spriteimage.circle-square {
background-position: -485px -496px;
}
.spriteimage.diagonal-square {
background-position: -543px -493px;
}
.spriteimage.hexagon-square {
background-position: -603px -496px;
}
.spriteimage.pentagon-square {
background-position: -659px -496px;
}
.spriteimage.globe {
background-position: -811px -502px;
}
.spriteimage.globe_01 {
background-position: -151px -557px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.vertical-barchart {
background-position: -773px -550px;
}
.spriteimage.vertical-barchart_01 {
background-position: -826px -550px;
}
.spriteimage.horizontal-barchart {
background-position: 2px -605px;
}
.spriteimage.horizontal-barchart_01 {
background-position: -50px -603px;
}
.spriteimage.pie-chart {
background-position: -101px -617px;
}
.spriteimage.triangle {
background-position: -153px -617px;
}
.spriteimage.inverted-triangle {
background-position: -210px -600px;
}
.spriteimage.pyramid {
background-position: -268px -612px;
}
.spriteimage.inverted-pyramid {
background-position: -857px -494px;
}
.spriteimage.comments {
background-position: -323px -613px;
}
.spriteimage.folder {
background-position: -272px -555px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.folder_01 {
background-position: -432px -600px;
}
.spriteimage.folder-open {
background-position: -488px -600px;
}
.spriteimage.folder-open_01 {
background-position: -539px -600px;
}
.spriteimage.folder-add {
background-position: -601px -600px;
}
.spriteimage.signal {
background-position: -657px -600px;
}
.spriteimage.sort-alpha-desc {
background-position: -820px -600px;
}
.spriteimage.sort-alpha-desc_01 {
background-position: -50px -661px;
}
.spriteimage.sort-alpha-asc {
background-position: -437px -668px;
}
.spriteimage.sort-alpha-asc_01 {
background-position: -492px -661px;
}
.spriteimage.print {
background-position: -212px -668px;
}
.spriteimage.print_01 {
background-position: -551px -661px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.print_02 {
background-position: -607px -661px;
}
.spriteimage.print_03 {
background-position: -664px -661px;
}
.spriteimage.word {
background-position: -721px -661px;
}
.spriteimage.word_01 {
background-position: -771px -661px;
}
.spriteimage.word-export {
background-position: -827px -661px;
}
.spriteimage.pdf {
background-position: -2px -722px;
}
.spriteimage.pdf_01 {
background-position: -53px -722px;
}
.spriteimage.pdf-export {
background-position: -106px -722px;
}
.spriteimage.excel {
background-position: -226px -714px;
}
.spriteimage.excel_01 {
background-position: -284px -733px;
}
.spriteimage.excel-export {
background-position: -338px -724px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.powerpoint-export {
background-position: -548px -718px;
}
.spriteimage.ie {
background-position: -604px -727px;
}
.spriteimage.exit {
background-position: -661px -728px;
}
.spriteimage.document {
background-position: -713px -723px;
}
.spriteimage.documents {
background-position: -766px -723px;
}
.spriteimage.question {
background-position: -381px -719px;
}
.spriteimage.film {
background-position: -818px -723px;
}
.spriteimage.volume-up {
background-position: -60px -775px;
}
.spriteimage.circle-one {
background-position: -120px -782px;
}
.spriteimage.circle-two {
background-position: -174px -782px;
}
.spriteimage.circle-three {
background-position: -337px -782px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.circle-four {
background-position: -393px -782px;
}
.spriteimage.arrow-right {
background-position: -449px -782px;
}
.spriteimage.arrow-left {
background-position: -501px -782px;
}
.spriteimage.arrow-left_01 {
background-position: -559px -782px;
}
.spriteimage.arrow-up {
background-position: -231px -782px;
}
.spriteimage.arrow-down {
background-position: -607px -782px;
}
.spriteimage.arrow-down_01 {
background-position: -252px -442px;
}
.spriteimage.sync {
background-position: -711px -782px;
}
.spriteimage.sync-disabled {
background-position: -759px -782px;
}
.spriteimage.paperclip {
background-position: -808px -782px;
}
.spriteimage.paperclip_01 {
background-position: -58px -837px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.spriteimage.circle {
background-position: -857px -782px;
}
.spriteimage.circle_01 {
background-position: -233px -833px;
}
.spriteimage.th-list {
background-position: -448px -832px;
}
.spriteimage.th-large {
background-position: -657px -829px;
}
.spriteimage.th {
background-position: -111px -838px;
}
.spriteimage.th-small {
background-position: -506px -838px;
}
.spriteimage.view-details {
background-position: -558px -849px;
}
.spriteimage.file-resize-four-direction {
background-position: -781px -837px;
}
.spriteimage.file-resize-horizontal {
background-position: -281px -833px;
}
</style>
```

shrink	enlarge	key	list	list-numbered
list-numbered_01	indent	indent_01	outdent	outdent_01
close	close_01	undo	undo_01	redo
redo_01	video	video_01	cross-circle	clean-brush

delete-column	delete-column_01	delete-row	delete-row_01	insert-column-left
insert-column-left_01	insert-column-left_02	insert-column-right	insert-column-right_01	insert-column-right_02
insert-row-below	insert-row-below_01	insert-row-below_02	insert-row-above	insert-row-above_01
insert-row-above_02	table	table_01	table_02	sigma
sigma_01	uppercase	uppercase_01	lowercase	lowercase_01
background-color	superscript	superscript_01	subscript	subscript_01
restore	upload	download	save	save_01
save_02	mail/message	arrowhead-right	arrowhead-right_01	arrowhead-left
arrowhead-left_01	arrowhead-up	arrowhead-down	arrow-sans-right	arrow-sans-right_01
arrow-sans-left	arrow-sans-left_01	arrow-sans-up	arrow-sans-down	datetime
datetime_01	calendar	calendar_01	calendar-plus	calendar-edit
chevron-right	chevron-right_01	chevron-right_02	chevron-left	chevron-left_01
chevron-left_02	chevron-up	chevron-up_01	chevron-down	chevron-down_01
chevron-circle-right	chevron-circle-left	font	strikethrough	strikethrough_01
bold	bold_01	italic	italic_01	underline
underline_01	reply	forward	export	user
clipboard	home	clear	resize-handle	link
link_01	unlink	unlink_01	external-link	external-link_01
clock	settings	cut	cut_01	copy
copy_01	copy_02	paste	paste_01	star
pointer	sort direct	shopping cart	shopping cart_01	cursor
warning	zoom-out	zoom-out_01	zoom-out_02	zoom-in
zoom-in_01	zoom-in_02	arrow-circle-left	arrow-circle-left_01	arrow-circle-left_02
arrow-circle-right	arrow-circle-right_01	arrow-circle-right_02	arrow-circle-up	arrow-circle-down
edit	edit_01	edit_02	edit_03	edit_04
edit_05	notification	notification_01	info	smiley
check mark	check mark_01	media-play	media-pause	media-eject

Copyright © 2001 - 2020 Syncfusion Inc.

media-next	media-previous	media-forward	media-forward_01	media-forward_02
media-forward_03	media-forward_04	media-backward	media-backward_01	media-backward_02
media-backward_03	media-backward_04	play-circle	full-screen-expand	full-screen-expand_01
full-screen-collapse	full-screen-collapse_01	bullets	bullets_01	filter
filter_01	filter none	filternone_01	filter-settings	align-right
align-right_01	align-left	align-left_01	align-justify	align-justify_01
align-center	align-center_01	align-none	search	search_01
image	image_01	plus	plus_01	minus
minus_01	code	e-code_01	code-hexagon	file-code
file-html	palette	reload	delete	delete_01
delete_02	delete_03	delete_04	delete_05	pin
unpin	stop	power-cord	full borders	3d
file-excel	file-text	file-text_01	file-mdx	file-empty
file-list	file-delete_01	file-settings	circle-square	diagonal-square
hexagon-square	pentagon-square	globe	globe_01	vertical-bar chart
vertical-bar chart_01	horizontal-bar chart	horizontal-bar chart_01	pie-chart	triangle
inverted-triangle	pyramid	inverted-pyramid	comments	folder
folder_01	folder-open	folder-open_01	folder-add	signal
sort-alpha-desc	sort-alpha-desc_01	sort-alpha-asc	sort-alpha-asc_01	print
print_01	print_02	print_03	word	word_01
word-export	PDF	PDF_01	pdf-export	excel
excel_01	excel-export	powerpoint-export	IE	exit
document	documents	question	film	volume-up
circle-one	circle-two	circle-three	circle-four	arrow-right
arrow-left	arrow-left_01	arrow-up	arrow-down	arrow-down_01
sync	sync-disabled	paper clip	paper clip_01	circle
circle_01	th-list	th-large	th	th-small

Copyright © 2001 - 2020 Syncfusion Inc.

view-details	file-resize-four-direction	file-resize-horizontal		

CDN

In case, if the users need to make use of the CDN links instead of the normal above specified scripts and stylesheet references in their application, then have a look at the link here.

Getting Started

Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Please refer to this link to know about registering Syncfusion license key in your ASP.NET MVC application to use our components.

What is Html.EJ() ?

EJ refers to Essential JavaScript or Syncfusion Essential JS 1. It is a comprehensive collection of over 80 enterprise-grade HTML5 JavaScript components for building modern web applications. Refer to this JavaScript Package for more information. This is inherited from Syncfusion.EJ Assembly and Syncfusion.JavaScript namespace. Syncfusion.EJ and Syncfusion.EJ.MVC assemblies have to be referred in the project to use this, and these namespaces have to be added in the Web.config file as shown in this KB article.

Getting started with Syncfusion MVC

This section describes how to configure the Syncfusion ASP.NET MVC components into the ASP.NET MVC applications. There are four ways for embedding our controls into ASP.NET application:

1. Through Syncfusion Project Template
2. Through Syncfusion Project Conversion
3. Through Syncfusion NuGet Packages
4. Through Manual Integration into the new/existing Application

The procedures that are followed in manual integration process are entirely automated, when we create an application using Syncfusion Project template.

The similar steps are followed for integrating the Syncfusion controls into MVC 3, MVC 4, MVC 5, and MVC6 applications, the only thing that makes it a little bit different is the reference assemblies version chosen for each of the target MVC application.

Through Syncfusion Project Template

Syncfusion provides the Visual Studio Project Templates for the Syncfusion ASP.NET MVC platform to create a Syncfusion MVC application.

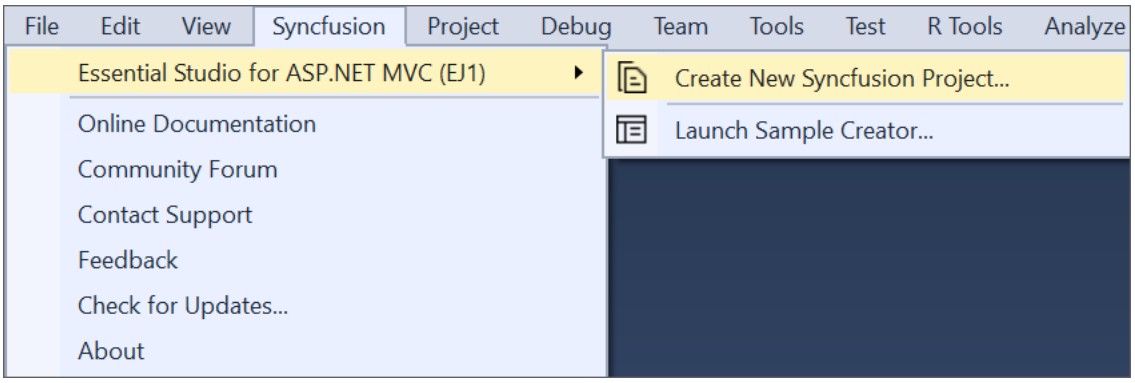
The **Project Configuration Wizard** automates the process of configuring the required Syncfusion assemblies, scripts and their styles within the newly created application. Lets look onto these topics in detail in the following sections.

1. To create a Syncfusion ASP.NET MVC (Essential JS 1) project, follow either one of the options below:

Option 1:

Copyright © 2001 - 2020 Syncfusion Inc.

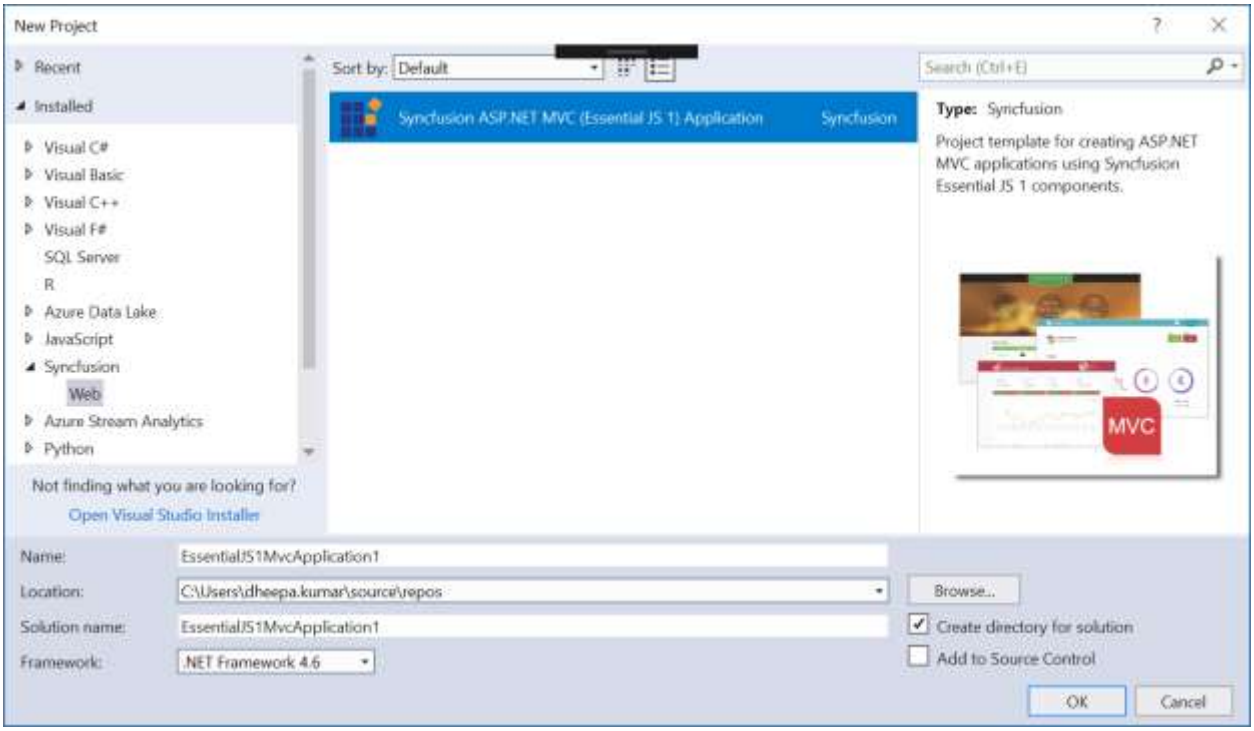
Click **Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ1) > Create New Syncfusion Project…** in **Visual Studio**.



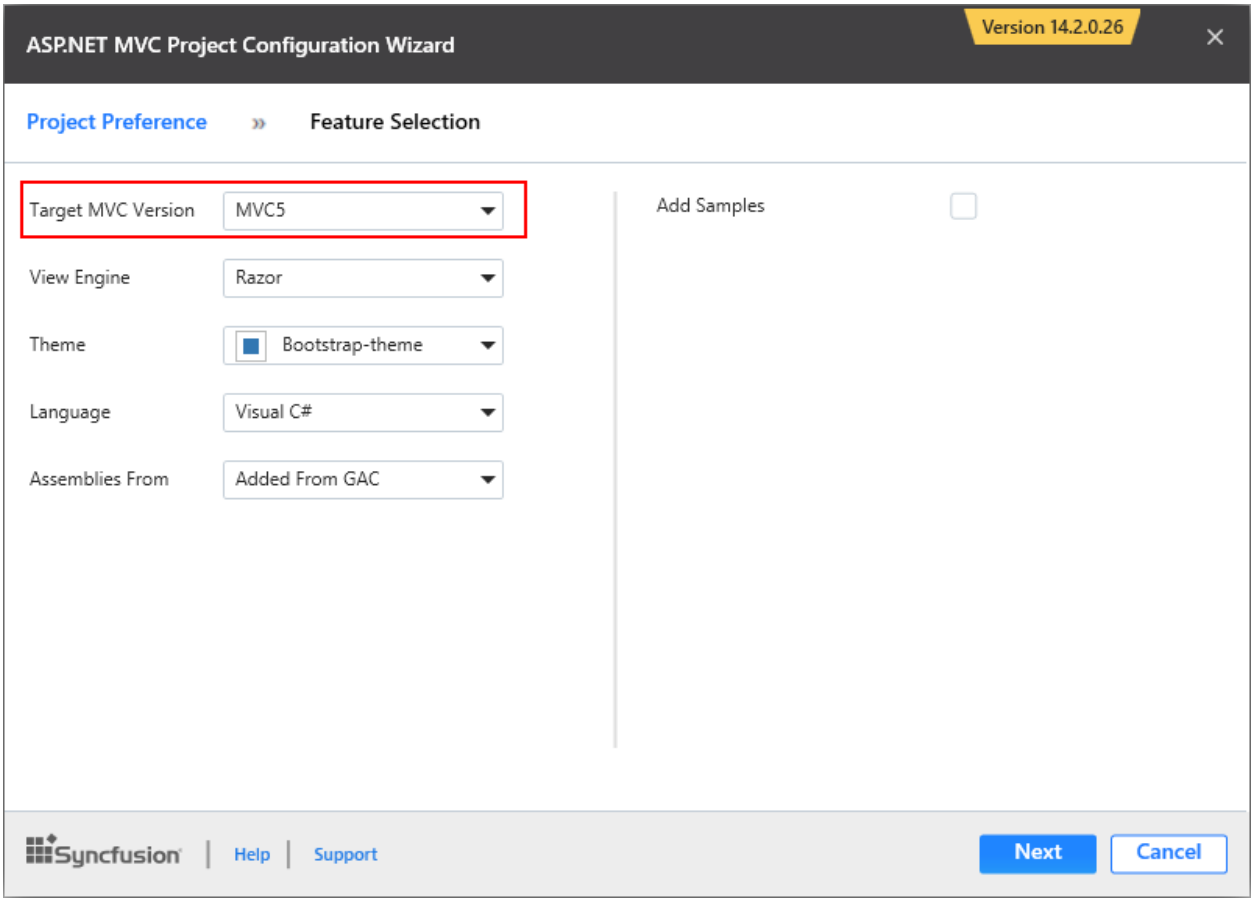
Note: In Visual Studio 2019, Syncfusion menu available under Extension in Visual Studio menu.

Option 2:

Choose **File > New > Project** and navigate to **Syncfusion > Web > Syncfusion ASP.NET MVC (Essential JS 1) Application** in **Visual Studio**.

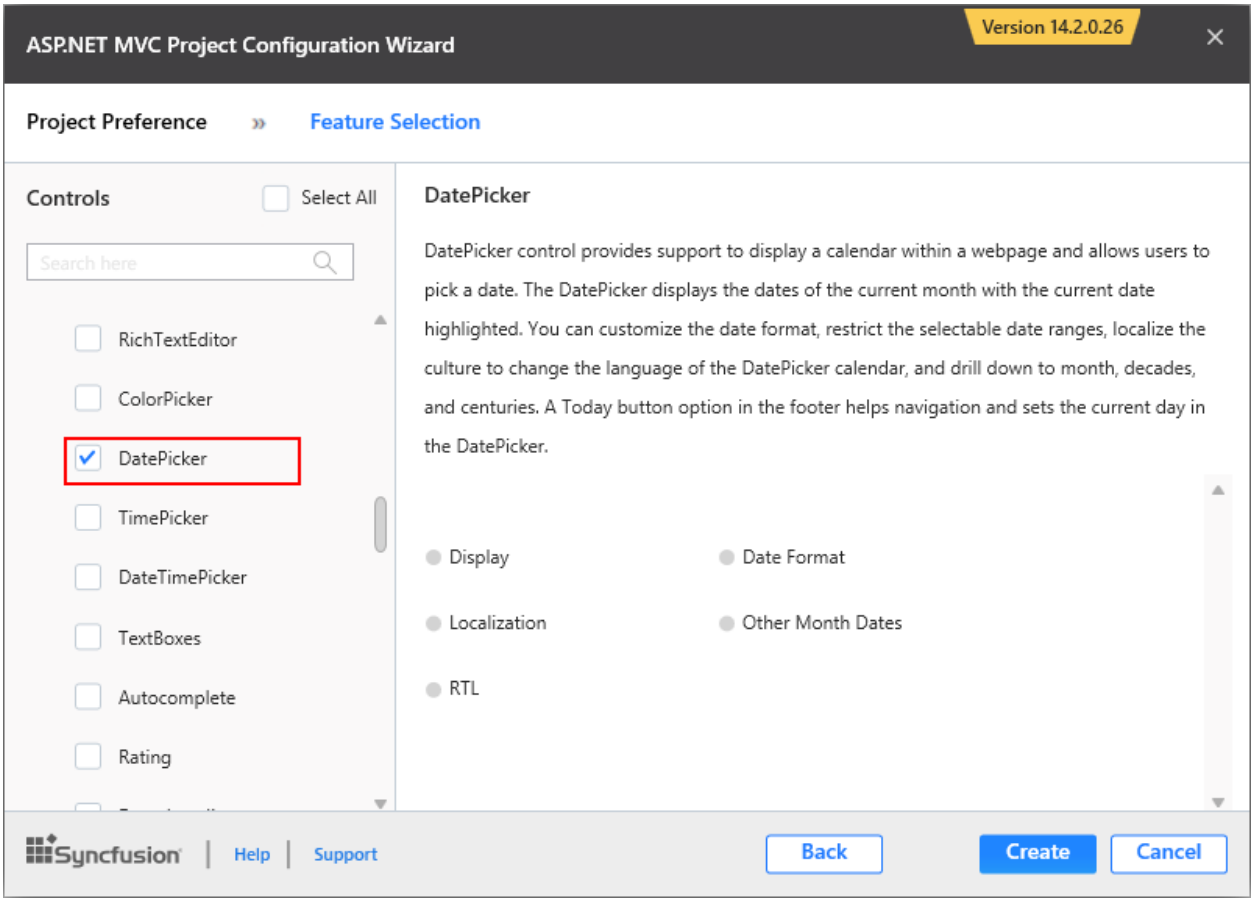


2. Then it opens **Project Configuration Wizard** as shown below. In this Wizard, select **Target MVC Version** as **MVC5** and keep the **other** options as default. Click **Next**.



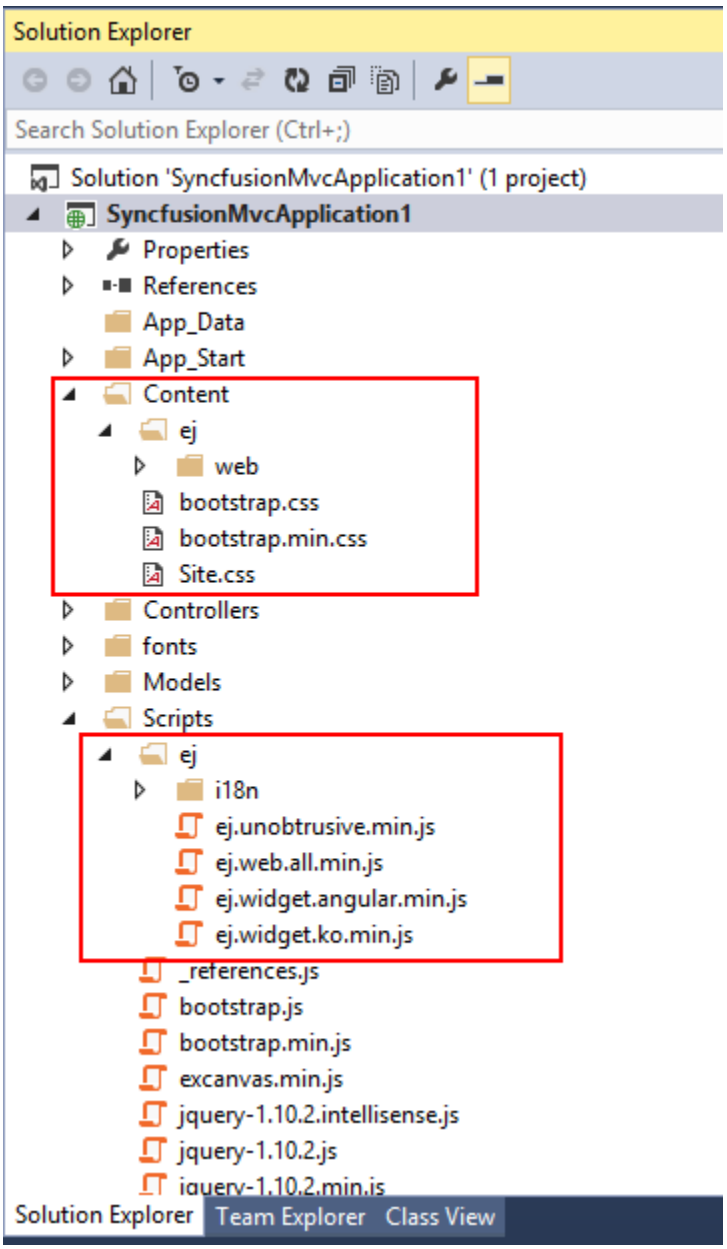
3. Next window will be shown with the list of Syncfusion MVC controls. Choose the required controls and then click **Create**.

Copyright © 2001 - 2020 Syncfusion Inc.

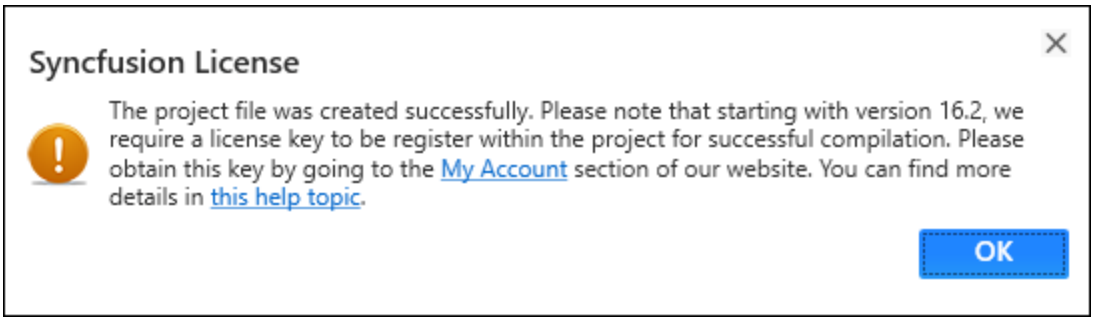


4. Now you can notice the **Syncfusion MVC 5 References**, **Scripts** and **Styles** are configured into Scripts and Content folders. Also it configures the **web.config** and **_Layout.cshtml** files.

Copyright © 2001 - 2020 Syncfusion Inc.



5. Then, Syncfusion licensing registration required message box will be shown as follow, if you are installed the trial setup or NuGet packages since Syncfusion introduced the licensing system from 2018 Volume 2 (v16.2.0.41) Essential Studio release. Please navigate to the help topic which is shown in the licensing message box to generate and register the Syncfusion license key to your project. Refer to this blog post for understanding the licensing changes introduced in Essential Studio.

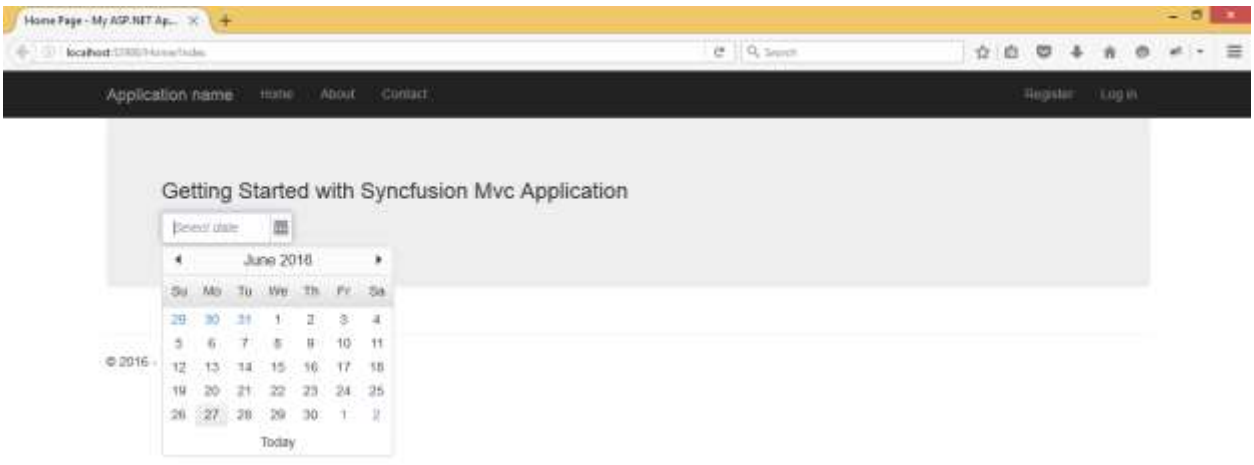


6. Now you can add the control **DatePicker** in the **Index.cshtml** file present within **~/Views/Home** folder.

ASPX-CS

```
@Html.EJ().DatePicker("MyFirstDatepicker")
```

7. Compile and execute the application. You can able to see the following output in the browser.



For more information about Project Configuration Templates and their options details, please visit here

Note: Ensure whether your project has only a single reference to jQuery. Because, multiple reference to jQuery in same project will throws console error and control will not be rendered. For more details refer to the KB here

Through Syncfusion Project Conversion

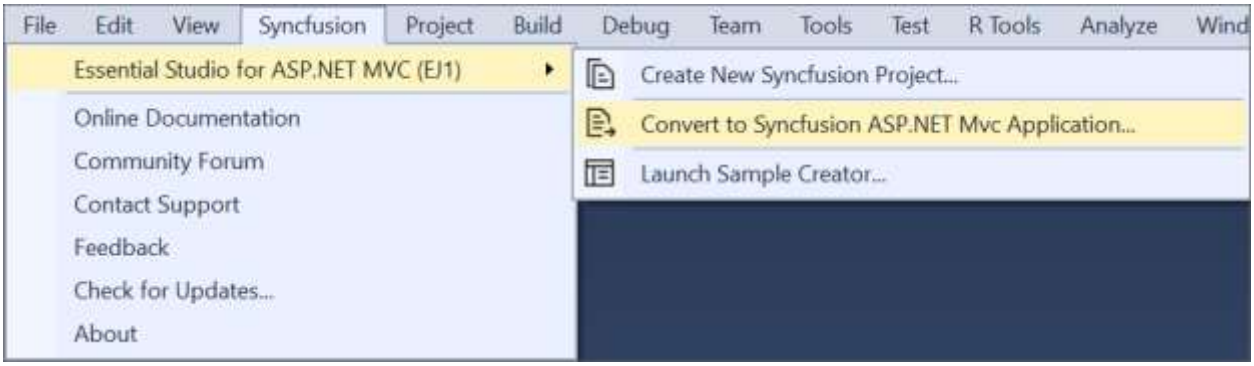
Syncfusion Project Conversion is a Visual Studio add-in that converts an existing ASP.NET MVC Project into a Syncfusion ASP.NET MVC Project by adding the required assemblies and resource files.

The following steps help you use the Syncfusion Project Conversion in the existing ASP.NET MVC (Web) Project.

1. Open an existing Microsoft MVC Project or create a new Microsoft MVC Project.
2. To open Project Conversion Wizard, follow either one of the options below:

Option 1:

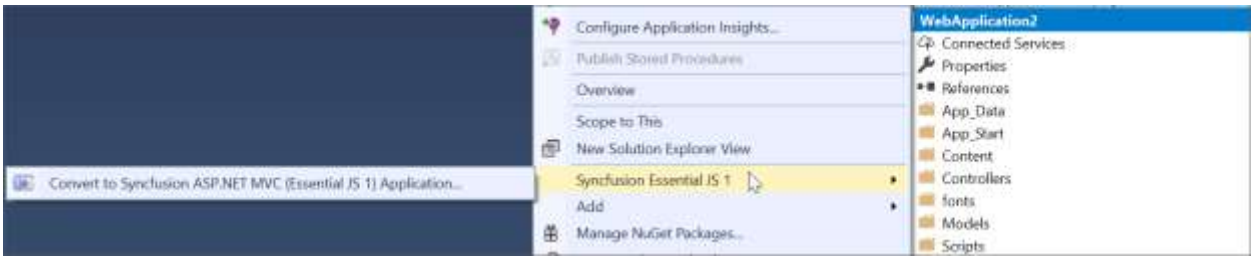
Click **Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ1) > Convert to Syncfusion ASP.NET MVC Application…** in **Visual Studio**.



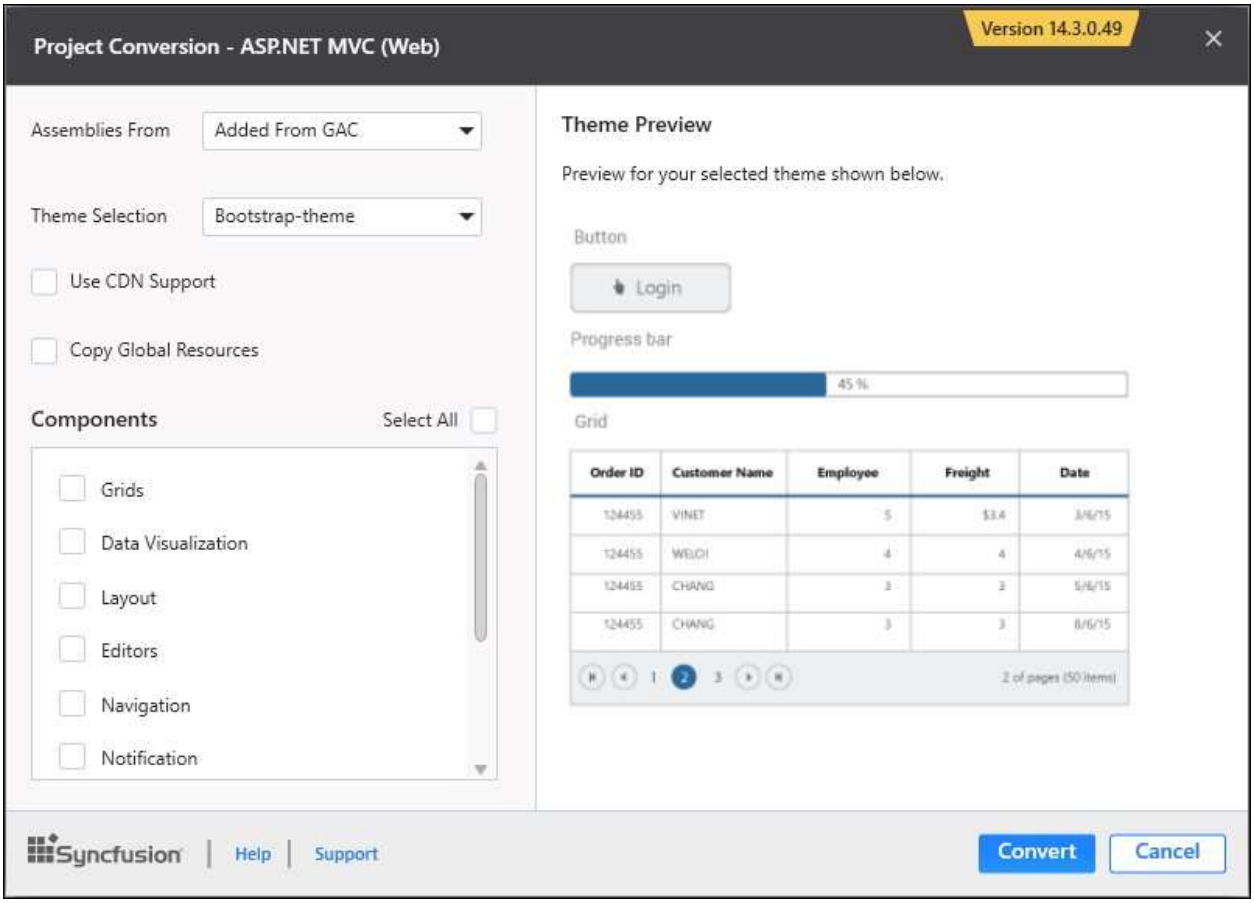
Note: In Visual Studio 2019, Syncfusion menu available under Extension in Visual Studio menu.

Option 2:

Right-click the Project from Solution Explorer, select **Syncfusion Essential JS 1**, and choose **Convert to Syncfusion MVC (Essential JS 1) Application...** Refer to the following screenshot for more information.



3. Project Conversion Wizard opens so that you can configure the project.

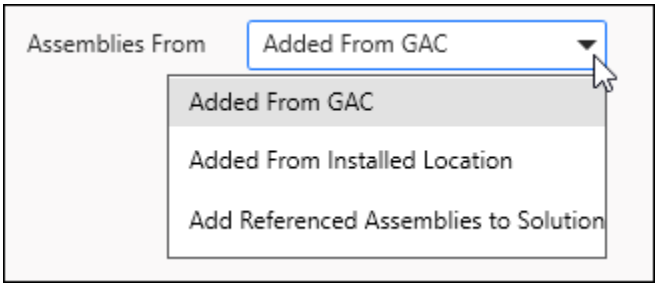


The following configurations are used in the Project Conversion Wizard.

Assemblies From:

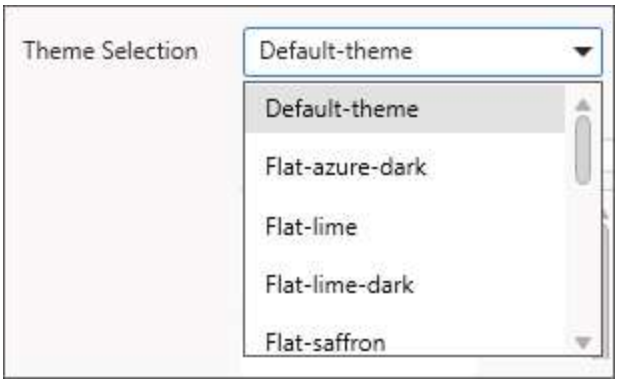
Choose the assembly location:

1. Added From GAC: Refer the assemblies from the Global Assembly Cache
2. Added from Installed Location: Refer the assemblies from the Syncfusion Installed locations.
3. Add Referenced Assemblies to Solution: Copy and refer to the assemblies from project's solution file lib directory.



Choose the Theme:

The master page of project will be updated based on the selected theme. The Theme Preview section shows the controls preview before convert into a Syncfusion project.



Choose CDN Support:

The master page of the project will be updated based on the required Syncfusion CDN links.

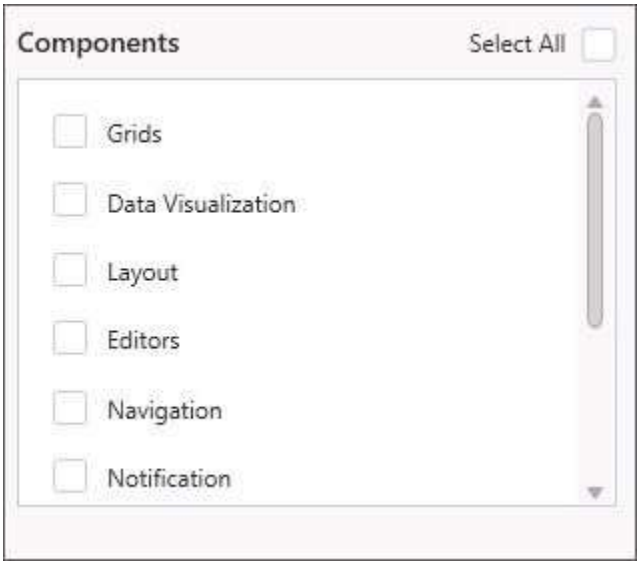


Choose Copy Global Resources:

The localization culture files will be shipped into Scripts\ej\i18n directory of the project.

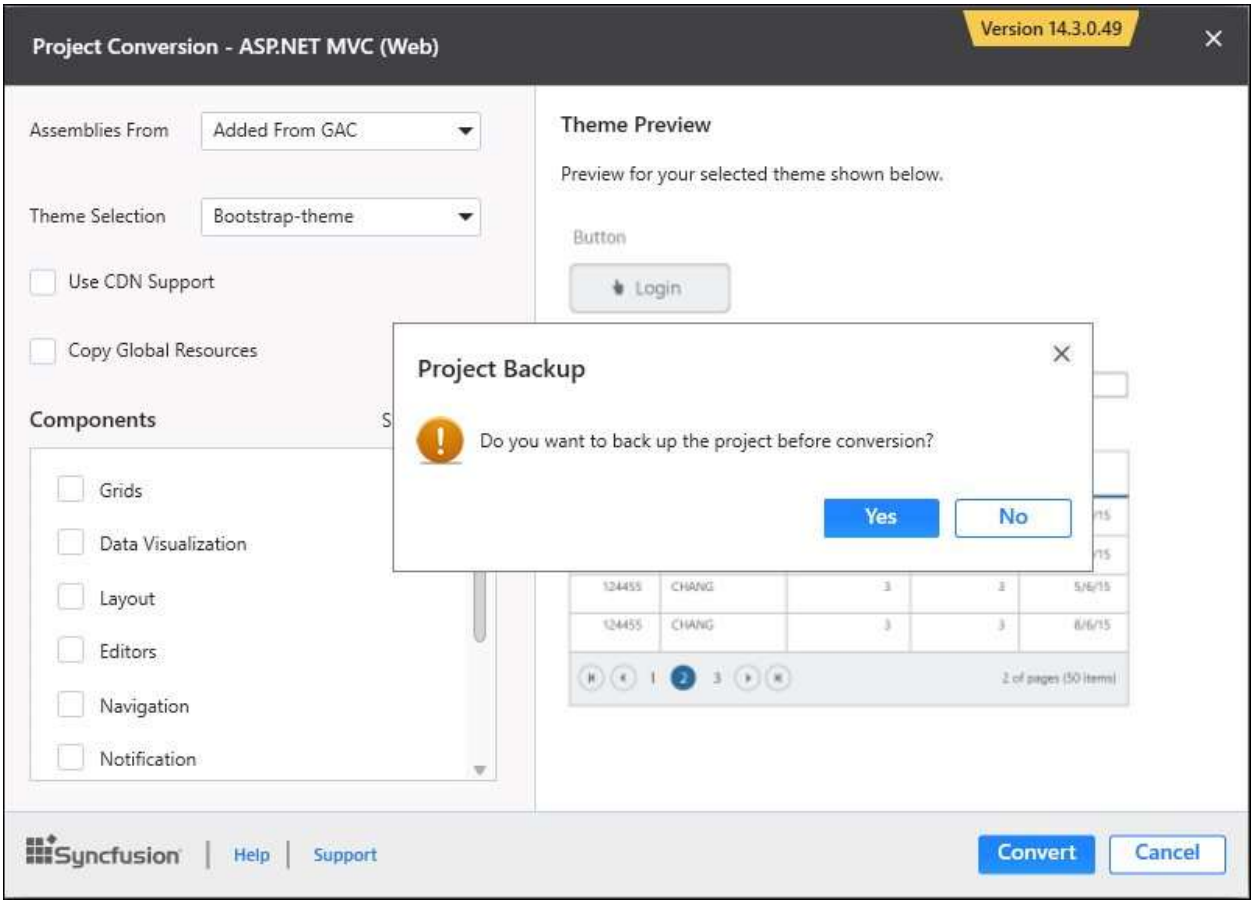


4. Choose the required controls from Components section and Click the **Convert** button to convert it into a Syncfusion Project.

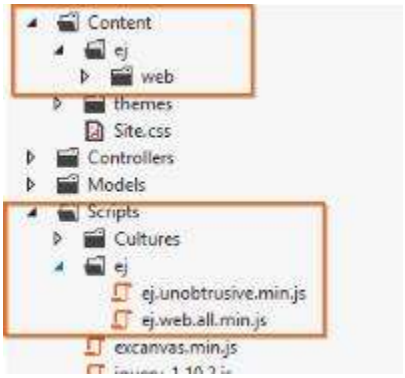


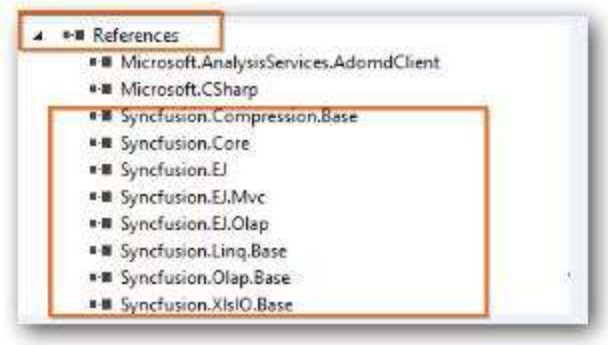
5. The **Project Backup** dialog will be opened. If Yes is clicked, it will backup the current project before converting it to Syncfusion project. If No is clicked it will convert the project to Syncfusion project without backup.

Copyright © 2001 - 2020 Syncfusion Inc.



6. The required Syncfusion Reference Assemblies, Scripts and CSS are included in the MVC Project. Refer to the following screenshots for more information.





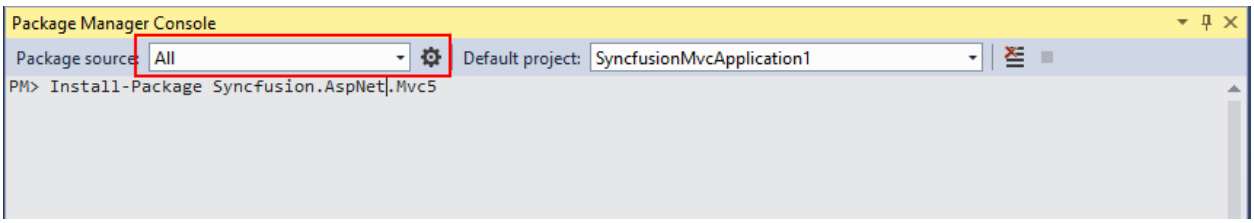

```
<compilation debug="true" targetFramework="4.5">
  <assemblies>
    <add assembly="Syncfusion.EJ.Export, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Compression.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"
    <add assembly="Syncfusion.EJ, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.EJ.Mvc, Version=14.3500.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Linq.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.XlsIO.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.EJ.Pivot, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Olap.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
  </assemblies>
</compilation>
```

Through Syncfusion NuGet Packages

To add our Syncfusion MVC controls into the new ASP.NET MVC5 application by making use of the **Syncfusion Nuget Packages**, refer to the following steps.

1. The steps to download and configure the required **Syncfusion NuGet Packages** in Visual Studio is mentioned here
2. Once Configured the Packages source, search and install the **Syncfusion.AspNet.Mvc5** from **Package Manager console** by using following commands.

PM>Install-Package Syncfusion.AspNet.Mvc5



3. While initial creation, by default, the **Unobtrusive** setting is enabled in your applications web.config file. You need to change its value to **false** in your application as shown in the following:

ASPX-CS

```
<appSettings>
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
```

4. You need to add the script manager code in the ***Layout.cshtml file present within the ~/Views/Shared folder of your application. Add it before the closing body tag in the*** Layout.cshtml file.

ASPX-CS

```
<body>
@RenderSection("scripts", required: false)
@Html.EJ().ScriptManager()
</body>
```

Note: The main reason for referring the Script manager in *Layout file is that, it can be referred as common by all the View files present within your application. If unobtrusive is set to true in the application, the script manager can be excluded, as the control is initialized using HTML5 attributes. Also for control rendering, you need to include ej.unobtrusive.min.js file along with ej.web.all.min.js in* **Layout.cshtml** page. Refer to the unobtrusive section.

5. Syncfusion specific stylesheets are loaded into the **Content** folder of your application, include the below specified theme reference **(bootstrap-theme/ej.web.all.min.css)** file in the **~/Views/Shared/_Layout.cshtml** file, within the head section as this file contains the default theme styles applied for all the Syncfusion MVC controls.

ASPX-CS

```
<head>
<title>@ViewBag.Title</title>
@Styles.Render("~/Content/ej/web/bootstrap-theme/ej.web.all.min.css")
</head>
```

6. It is mandatory to include the reference to the required JavaScript files in your **_Layout.cshtml**, so as to render the Syncfusion MVC controls properly.

ASPX-CS

```
<head>
<meta charset="utf-8" />
<title>@ViewBag.Title - My ASP.NET MVC Application</title>
@Styles.Render("~/Content/ej/web/bootstrap-theme/ej.web.all.min.css")
</head>
<body>
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@Scripts.Render("~/Scripts/jsrender.min.js")
@Scripts.Render("~/Scripts/ej/web/ej.web.all.min.js")
@RenderSection("scripts", required: false)
@Html.EJ().ScriptManager();
</body>
```

The order of the reference to the script files made in the above section should be maintained in the same manner as mentioned above.

Copyright © 2001 - 2020 Syncfusion Inc.

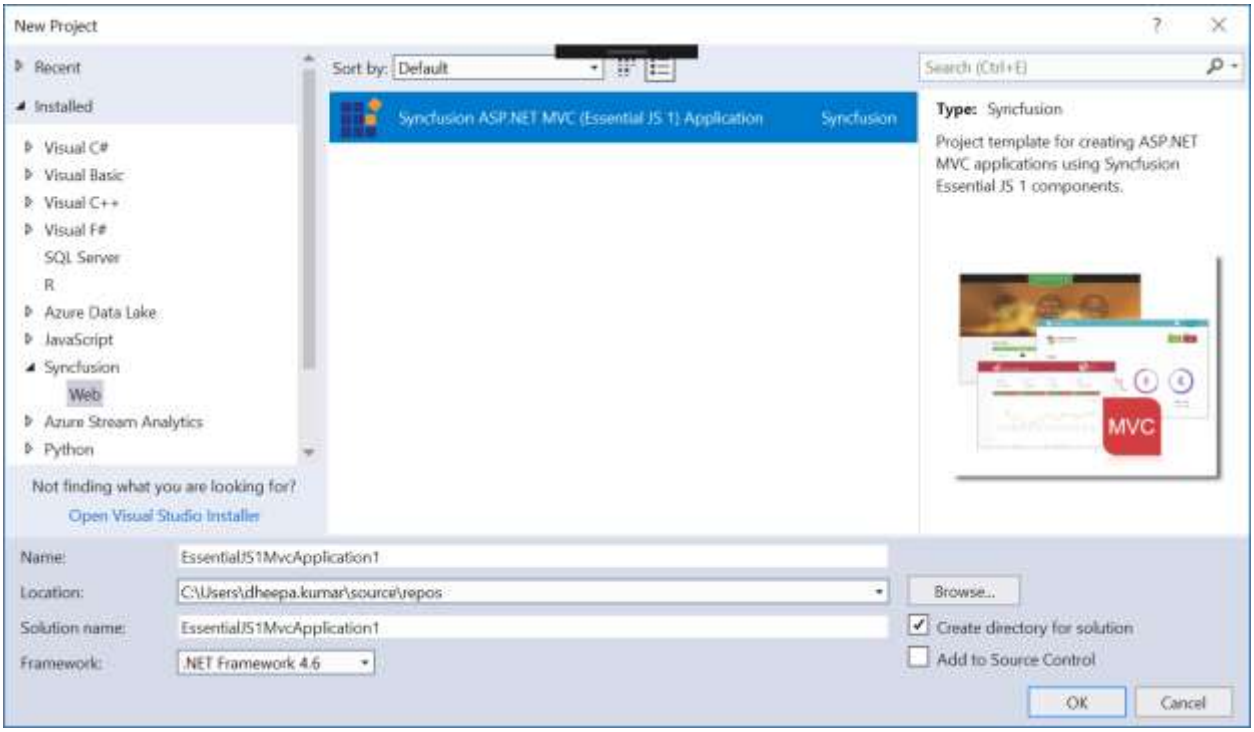
If your application contains duplicate/multiple references to the jQuery files, remove it as the explicit reference to the **jquery-1.10.2.min.js** script file that is added to the application as specified above.

7. Now you can add the control **DatePicker** in the **Index.cshtml** file present within **~/Views/Home** folder.

ASPX-CS

```
@Html.EJ().DatePicker("MyFirstDatepicker")
```

8. Compile and execute the application. You can able to see the below output in the browser.



For more details - View

Manual Integration

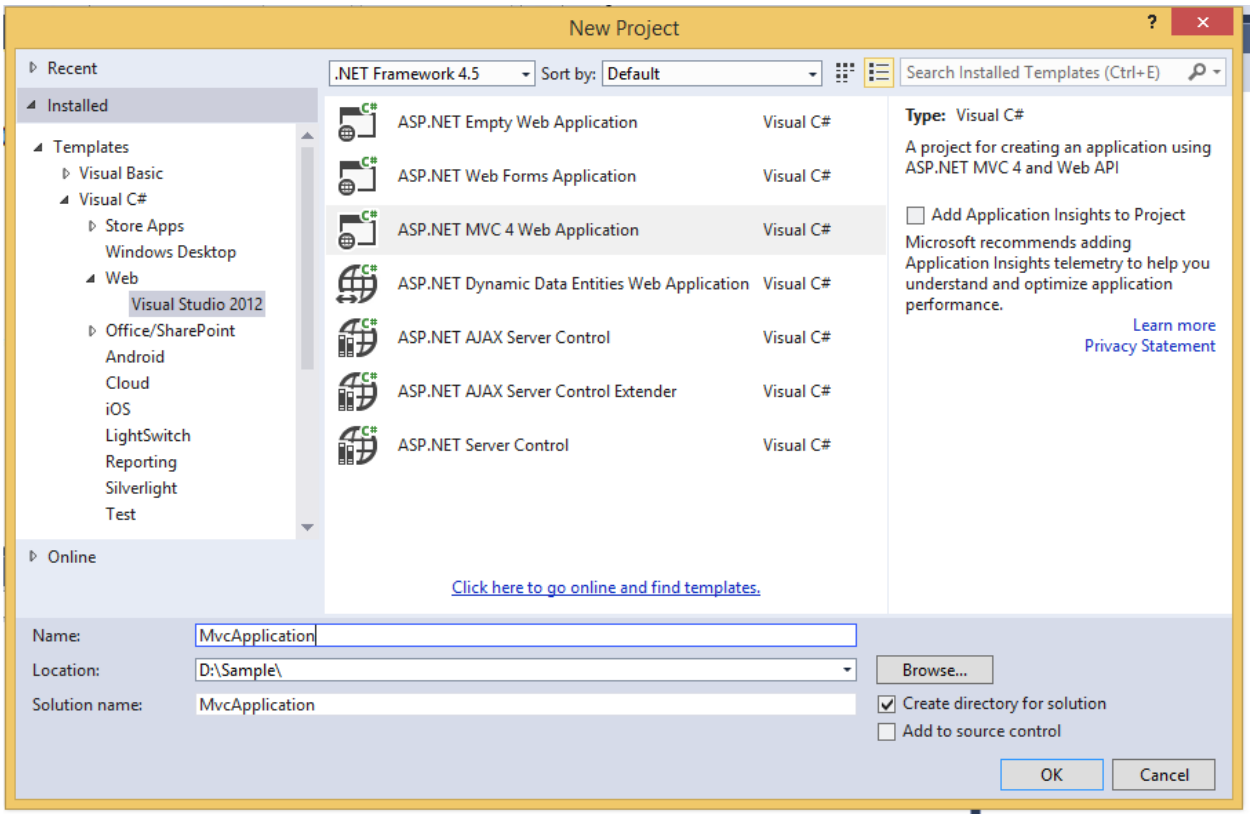
This topic mainly focuses on how to integrate the Syncfusion ASP.NET MVC controls manually into the newly created/existing ASP.NET MVC application. The procedure for making use of any of our ASP.NET MVC controls within the ASP.NET MVC application are explained in the following.

Creation of First ASP.NET MVC Application

Follow the below steps to create a normal ASP.NET MVC application.

1. Start the Visual Studio. Create a new MVC application by selecting File -> New -> Project and save it with a meaningful name as shown in the following.

Copyright © 2001 - 2020 Syncfusion Inc.



2. Build and run your application by pressing Ctrl+F5.

It is time to add some other essential things to your application that allows you to make use of our Syncfusion ASP.NET MVC controls. For that, follow steps explained in the Existing Application section.

For Existing Applications

To add our Syncfusion ASP.NET MVC controls to your existing application, open your existing application and proceed with the following steps.

Adding the required StyleSheets

To render the Syncfusion ASP.NET MVC controls with its unique style and theme, it is necessary to refer to the required CSS files into your application. Copy all the required CSS files into your application from the following location.

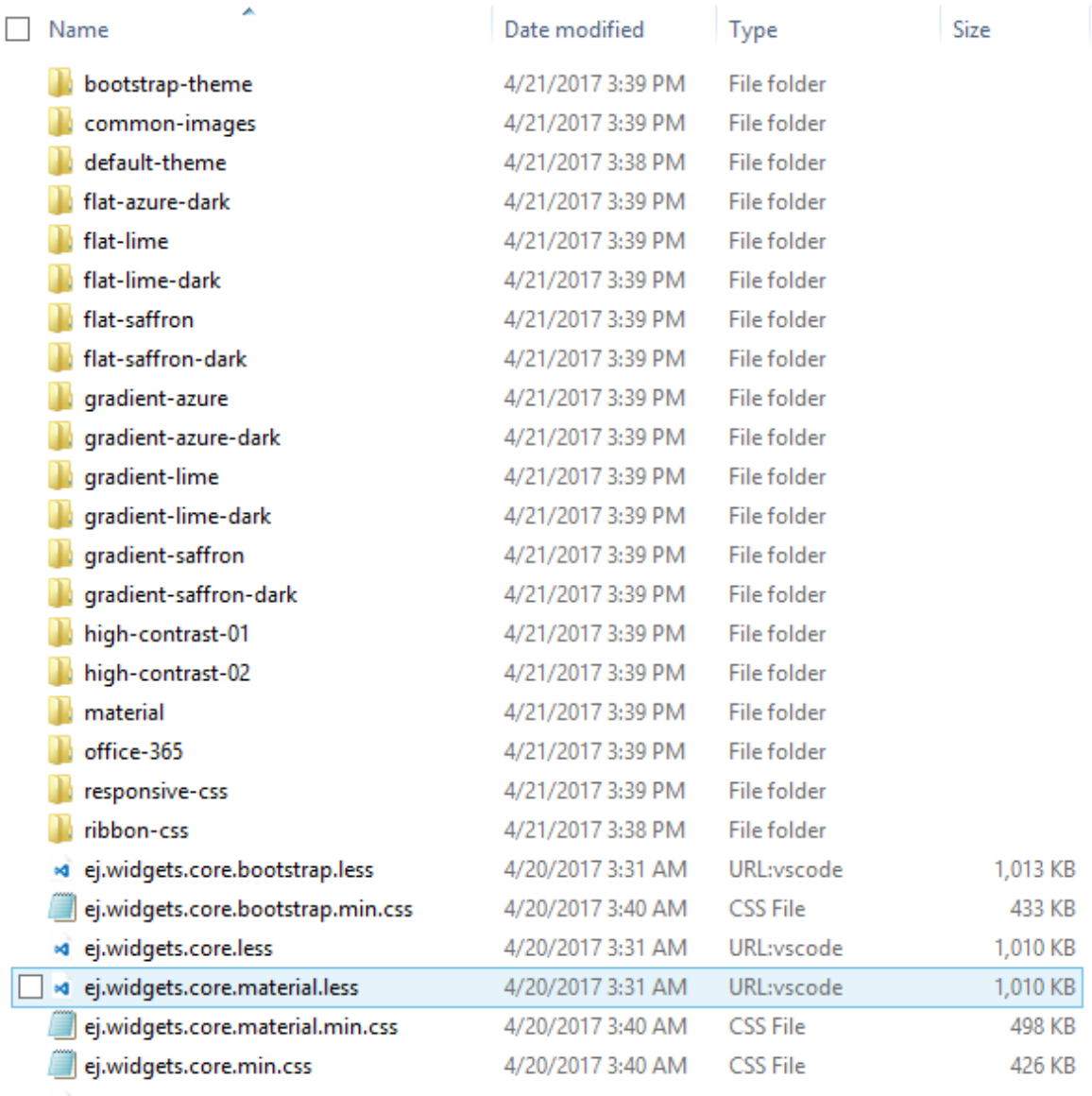
Note: <installed location>\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\css\web

For example, If you have installed the Essential Studio within C:\Program Files (x86), then navigate to the below location,

C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\css\web

Navigate to above mentioned location, find the files as shown in the below image. Copy entirely and paste it into your root application.

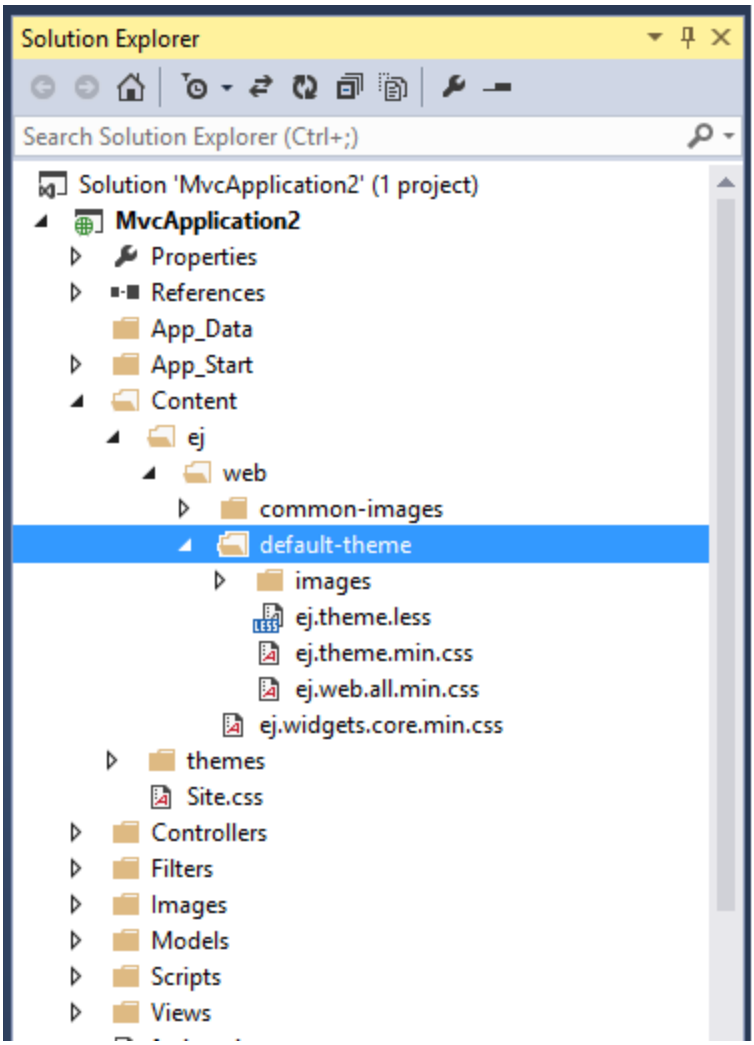
Copyright © 2001 - 2020 Syncfusion Inc.



Name	Date modified	Type	Size
bootstrap-theme	4/21/2017 3:39 PM	File folder	
common-images	4/21/2017 3:39 PM	File folder	
default-theme	4/21/2017 3:38 PM	File folder	
flat-azure-dark	4/21/2017 3:39 PM	File folder	
flat-lime	4/21/2017 3:39 PM	File folder	
flat-lime-dark	4/21/2017 3:39 PM	File folder	
flat-saffron	4/21/2017 3:39 PM	File folder	
flat-saffron-dark	4/21/2017 3:39 PM	File folder	
gradient-azure	4/21/2017 3:39 PM	File folder	
gradient-azure-dark	4/21/2017 3:39 PM	File folder	
gradient-lime	4/21/2017 3:39 PM	File folder	
gradient-lime-dark	4/21/2017 3:39 PM	File folder	
gradient-saffron	4/21/2017 3:39 PM	File folder	
gradient-saffron-dark	4/21/2017 3:39 PM	File folder	
high-contrast-01	4/21/2017 3:39 PM	File folder	
high-contrast-02	4/21/2017 3:39 PM	File folder	
material	4/21/2017 3:39 PM	File folder	
office-365	4/21/2017 3:39 PM	File folder	
responsive-css	4/21/2017 3:39 PM	File folder	
ribbon-css	4/21/2017 3:38 PM	File folder	
ej.widgets.core.bootstrap.less	4/20/2017 3:31 AM	URL:vscode	1,013 KB
ej.widgets.core.bootstrap.min.css	4/20/2017 3:40 AM	CSS File	433 KB
ej.widgets.core.less	4/20/2017 3:31 AM	URL:vscode	1,010 KB
ej.widgets.core.material.less	4/20/2017 3:31 AM	URL:vscode	1,010 KB
ej.widgets.core.material.min.css	4/20/2017 3:40 AM	CSS File	498 KB
ej.widgets.core.min.css	4/20/2017 3:40 AM	CSS File	426 KB

Before pasting it into your application, create a folder structure with names ej/web within the Content folder of your application and place all the copied files into it as shown in the following.

Copyright © 2001 - 2020 Syncfusion Inc.



Solution Explorer - Project with CSS files copied into the Content folder

Note: The common-images folder is needed to be copied into your application mandatorily, as it includes all the common font icons and other images required for the control to render.

Once the CSS files are added in your application, include the reference to "ej.web.all.min.css" file in the _Layout.cshtml page, within the head section.

HTML

```
<link href="~/Content/ej/web/default-theme/ej.web.all.min.css"
rel="stylesheet" />
```

Adding the required JavaScript files

Adding the required JavaScript files into your application plays an important role, without that the Syncfusion controls cannot be created. It requires the following mandatory common script files.

- jquery.min.js (1.7.1 and later versions)
- jsrender.min.js

Copyright © 2001 - 2020 Syncfusion Inc.

Note: jQuery-2.1.4, jQuery-3.0.0 support has been given from ejVersion 13.2.0.29, 14.3.0.49 onwards respectively.

- ej.globalize.min.js library avails as built-in within ej.web.all.min.js file, therefore it is not necessary to externally refer it in your application (applicable for version 13.4.0.53 and higher). For version lower than 13.4.0.53, refer jQuery.globalize.min.js along with ej.web.all.min.js

Apart from the above common scripts, it is also necessary to refer the ej.web.all.min.js file in your application that plays a major role in control creation.

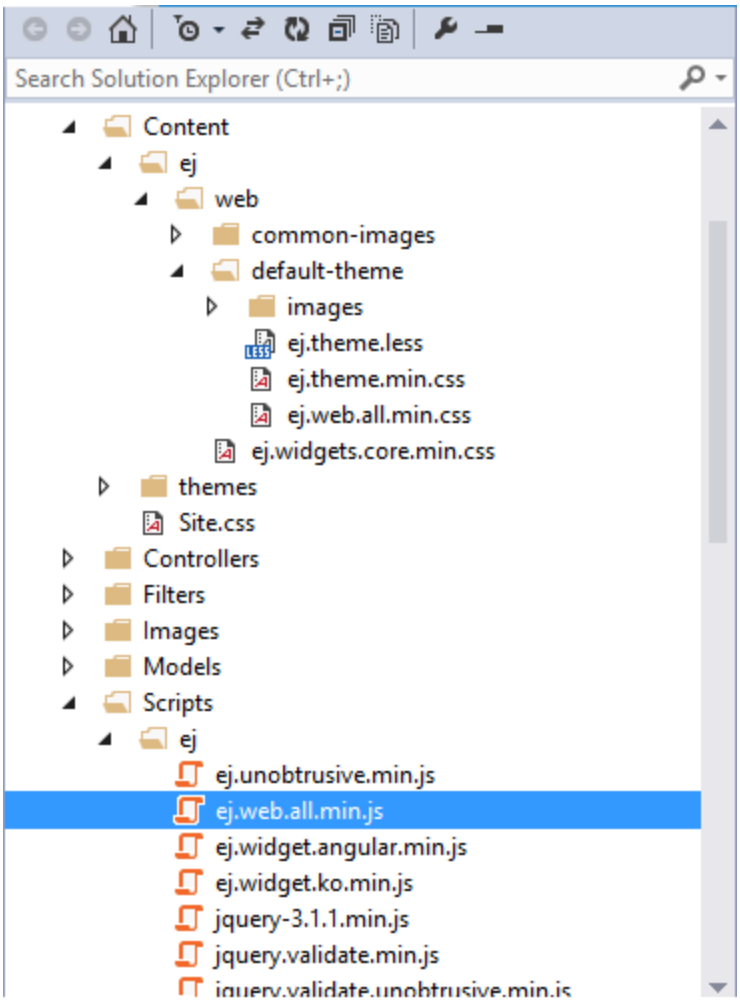
The dependencies are available in the following locations of your machine. Please copy these files from location given.

Files	Location
jquery-1.10.2.min.js jsrender.min.js	<Syncfusion Installed Location>\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\external
ej.web.all.min.js	<Syncfusion Installed Location>\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\web

Note: Example for "Syncfusion Installed location" is "C:\Program Files (x86)\Syncfusion"

Now, create a folder named ej, under the Scripts folder of your application and place the copied files ej.web.all.min.js into it as shown in the following.

Copyright © 2001 - 2020 Syncfusion Inc.



Solution Explorer - Script files copied into the Scripts folder of the project

Once the scripts are added in your application, now it is necessary to include the reference to it in your application. This should be done within the _Layout.cshtml page, as we did previously for CSS files.

Add the following script references in the _Layout.cshtml file within the head section.

HTML

```
<link href="Content/ej/web/default-theme/ej.web.all.min.css"
rel="stylesheet" />
<script src='<%= Page.ResolveClientUrl("~/Scripts/jquery-1.10.2.min.js")%>'
type="text/javascript"></script>
<script src='<%= Page.ResolveClientUrl("~/Scripts/jsrender.min.js")%>'
type="text/javascript"></script>
<script src='<%= Page.ResolveClientUrl("~/Scripts/ej/ej.web.all.min.js")%>'
type="text/javascript"></script>
```

While initial creation, by default, the **Unobtrusive** setting is enabled in your applications web.config file. You need to change its value to **false** in your application as shown in the following.

HTML

```
<appSettings>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
```

You need to add the script manager code in the ***Layout.cshtml file present within the ~/Views/Shared folder of your application. Add it before the closing body tag in the*** Layout.cshtml file.

HTML

```
<body>
@RenderSection("scripts", required: false)
@Html.EJ().ScriptManager()
</body>
```

Note: The main reason for referring the Script manager in *Layout file is that, it can be referred as common by all the View files present within your application. If unobtrusive is set to true in the application, the script manager can be excluded, as the control is initialized using HTML5 attributes. Also for control rendering, you need to include ej.unobtrusive.min.js file along with ej.web.all.min.js in* **Layout.cshtml** page. Refer to the unobtrusive section.

CDN Link reference

If you want to refer the CDN links instead of the direct script and CSS references in your application, you need to make use of the following references in the _Layout.cshtml page.

HTML

```
<head>
<meta charset="utf-8" />
<title><%: Page.Title %> - My ASP.NET Application</title>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/default-theme/ej.web.all.min.css" rel="stylesheet" />
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.easing.1.3.min.js">
</script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.globalize.min.js"><
/script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
</head>
```

Assembly Reference

Refer the following assemblies in your newly created ASP.NET MVC application that allows you to use any of the Syncfusion ASP.NET MVC controls within it.

- Syncfusion.EJ
- Syncfusion.EJ.MVC

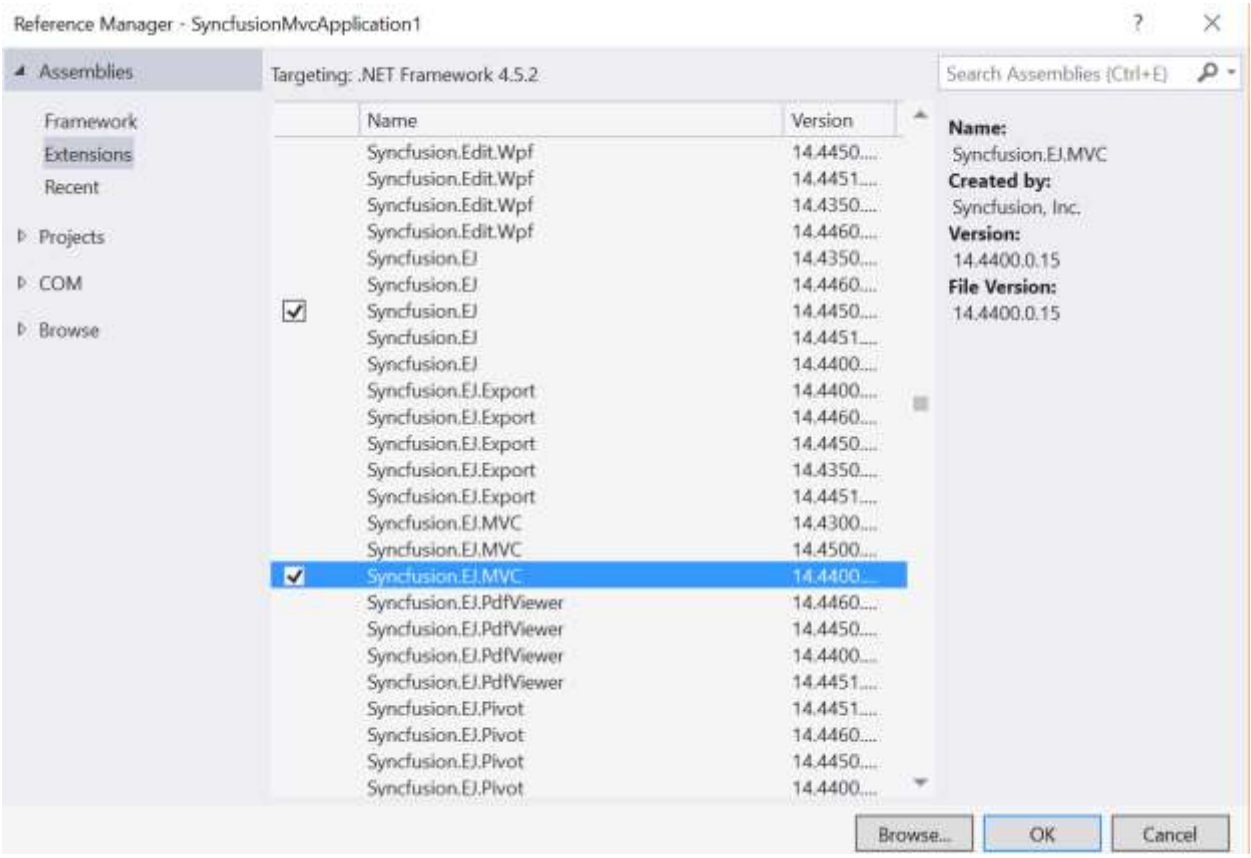
Copyright © 2001 - 2020 Syncfusion Inc.

The reference to the Syncfusion assemblies can be added to your application in either of the following ways.

- Referring from GAC - Referring from the installed location

Referring from GAC

1. Once you have installed the Essential Studio package in your system, the Syncfusion assemblies are automatically registered in the GAC. You can easily add the reference assemblies to your project by choosing Add Reference option. 2. Now the Reference Manager pop-up will appear on the screen. In that pop-up, select the required assemblies from the Extensions tab as below, by choosing the appropriate versions (13.1450.0.21). The version to be chosen for the reference assemblies is based on the Framework used in the application.



Reference Manager Pop-up

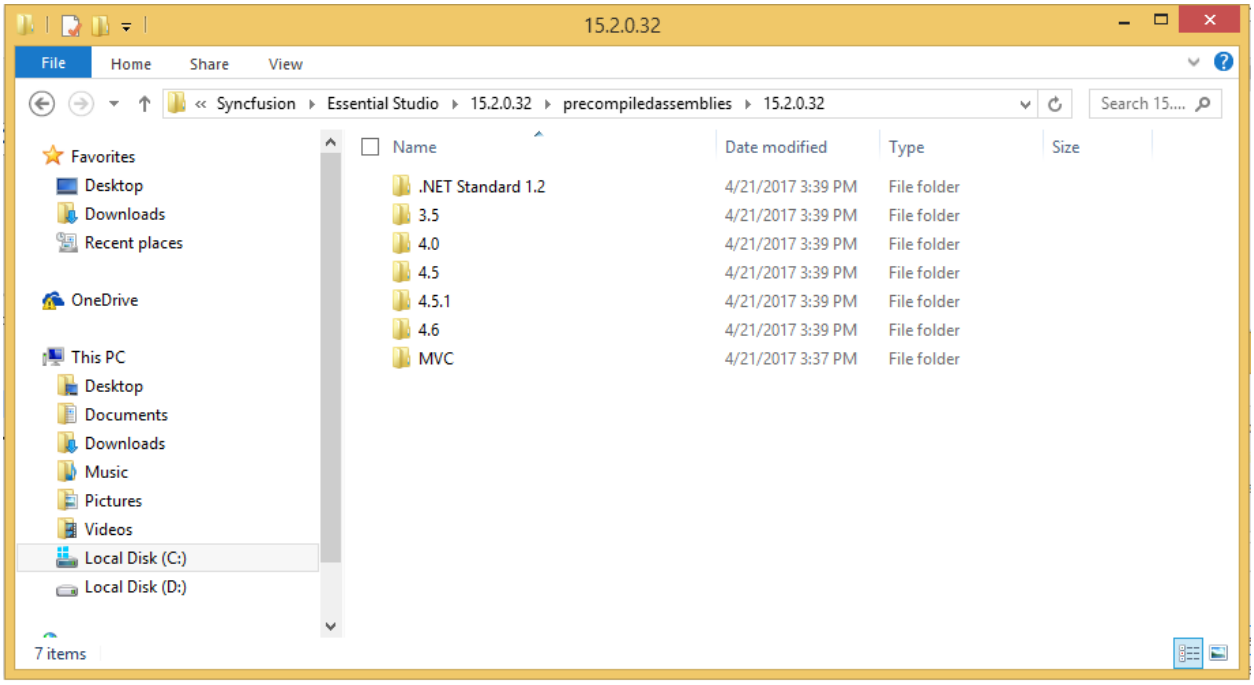
Referring from the installed location

1. Add the reference assemblies to your project by choosing Add Reference option, 2. Now the Reference Manager pop-up will appear on the screen. Select the Browse tab in it and navigate to the installed location of the Syncfusion Essential Studio package in your system. (As depicted in the below image.)

Note: <installed location>\Syncfusion\Essential Studio\{{ site.releaseversion }}\precompiledassemblies\14.4.0.15

For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the following location:

C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\precompiledassemblies\14.4.0.15



Reference Manager Pop-up with Browse button clicked

Note: In the above image, the folders 3.5, 4.0, 4.5, 4.5.1 denotes the .NET Framework version. Based on the Framework version used in your application, you can choose assemblies from the appropriate folders. The Syncfusion.EJ.MVC and other core assemblies like Syncfusion.Core, Syncfusion.EJ are available within these folders.

3. Add the Syncfusion.EJ, Syncfusion.EJ.MVC, and Syncfusion.Core assemblies to your application from the following specified location.

Note: <installed location>\Syncfusion\Essential Studio\{{ site.releaseversion }}\precompiledassemblies\{{ site.releaseversion }}\4.5

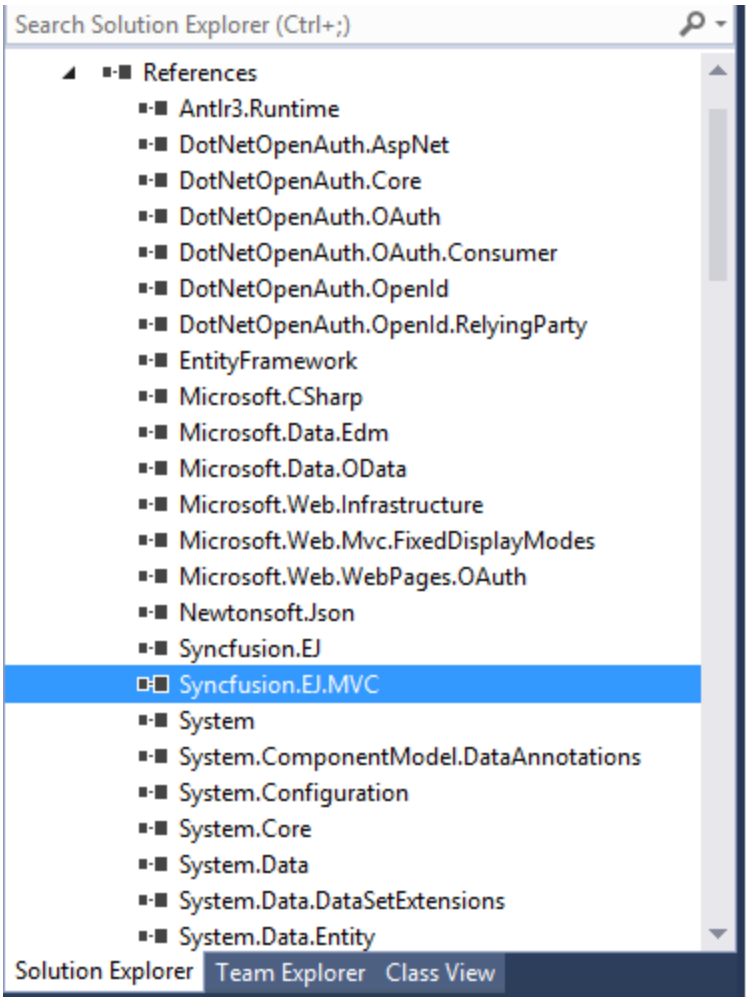
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location,

C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\precompiledassemblies\{{ site.releaseversion }}\4.5

Note: Syncfusion.Core dependency has been removed from 13.2.0.29 and it is not required to refer this assembly when you are using 13.2.0.29 and later versions.

4. Once the assembly selection is done, click OK to add the selected references to your project. You can view the assembly references added to your application, in the solution explorer as shown in the following.

Copyright © 2001 - 2020 Syncfusion Inc.



Selected Assemblies added to the Project reference

Registering Syncfusion Assemblies within the Web.config

In your application's web.config file, add the below assembly information within the <assemblies> tag.

XML

```
<system.web>
<compilation debug="true" targetFramework="4.5">
<assemblies>
<add assembly="Syncfusion.EJ, Version=15.4450.0.20, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version=15.4500.0.20, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
<authentication mode="Forms">
…
</system.web>
```

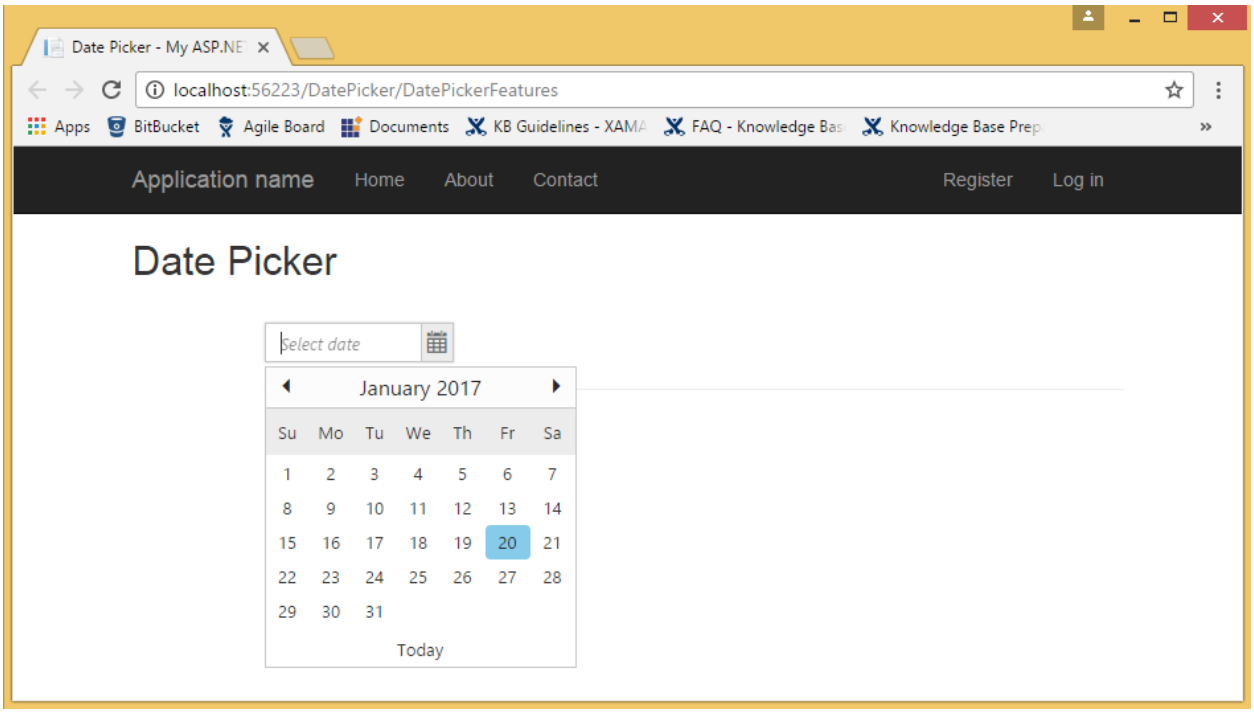
1. Add the below DatePicker code in your view page as shown in the following.

HTML

```
@Html.EJ().DatePicker("DatePick").DateFormat("MM/dd/yyyy").ShowOtherMonths(f
alse).EnableRTL(false).Locale("en-US")
```

Note: Add the DatePicker code within the Content section, by removing the unwanted code within it.

2. Finally build and run the project by pressing F5, so that you can now see the output similar to the following screenshot in your web browser.



DatePicker control displaying on the web browser

Thus the DatePicker control is rendered successfully with its default appearance. You can then use its various properties to set its value and also make use of its available events to trigger when necessary.

Version compatibility with respect to framework

The Syncfusion.EJ.MVC assembly supports only till framework 4.5, so the dependent Syncfusion.EJ assembly will also be installed in that version even if the framework chosen is 4.6 or above. Find the supported MVC versions for the target framework in the following table.

MVC Version	Assembly	Version	Example Version
MVC 3	Syncfusion.EJ Syncfusion.EJ.Mvc	XX.X400.X.XX XX.X300.X.XX	16.4400.0.52 16.4300.0.52
MVC 4	Syncfusion.EJ Syncfusion.EJ.Mvc	XX.X400.X.XX XX.X400.X.XX	16.4400.0.52 16.4400.0.52
MVC 5	Syncfusion.EJ Syncfusion.EJ.Mvc	XX.X450.X.XX XX.X500.X.XX	16.4400.0.52 16.4500.0.52

Copyright © 2001 - 2020 Syncfusion Inc.

Note: For Framework 4.6 and above, support has been provided in ASP.NET Core. So, the assembly version “XX.X460.X.XX” cannot be used in MVC platform.

Getting Started

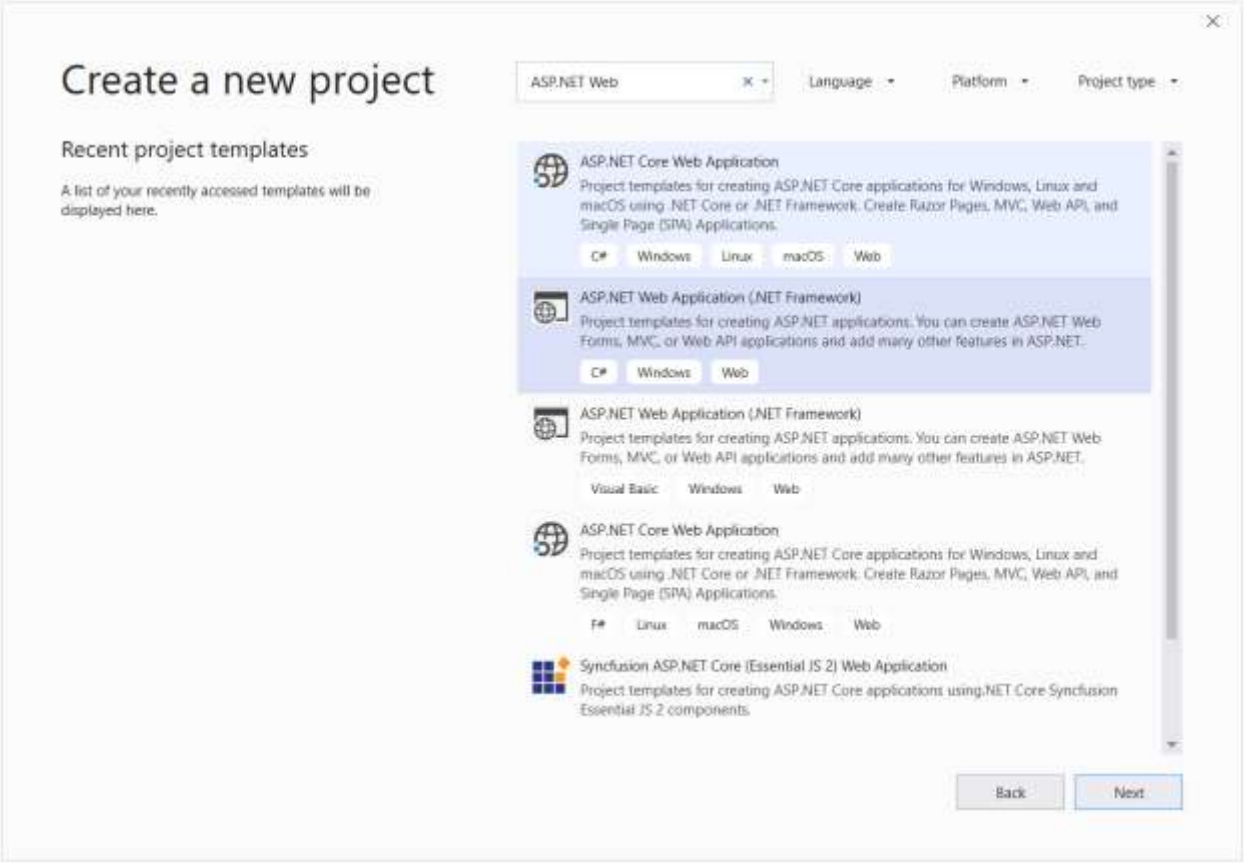
Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Please refer to this link to know about registering Syncfusion license key in your ASP.NET MVC application to use our components.

Configuring EJ MVC in Visual Studio 2019

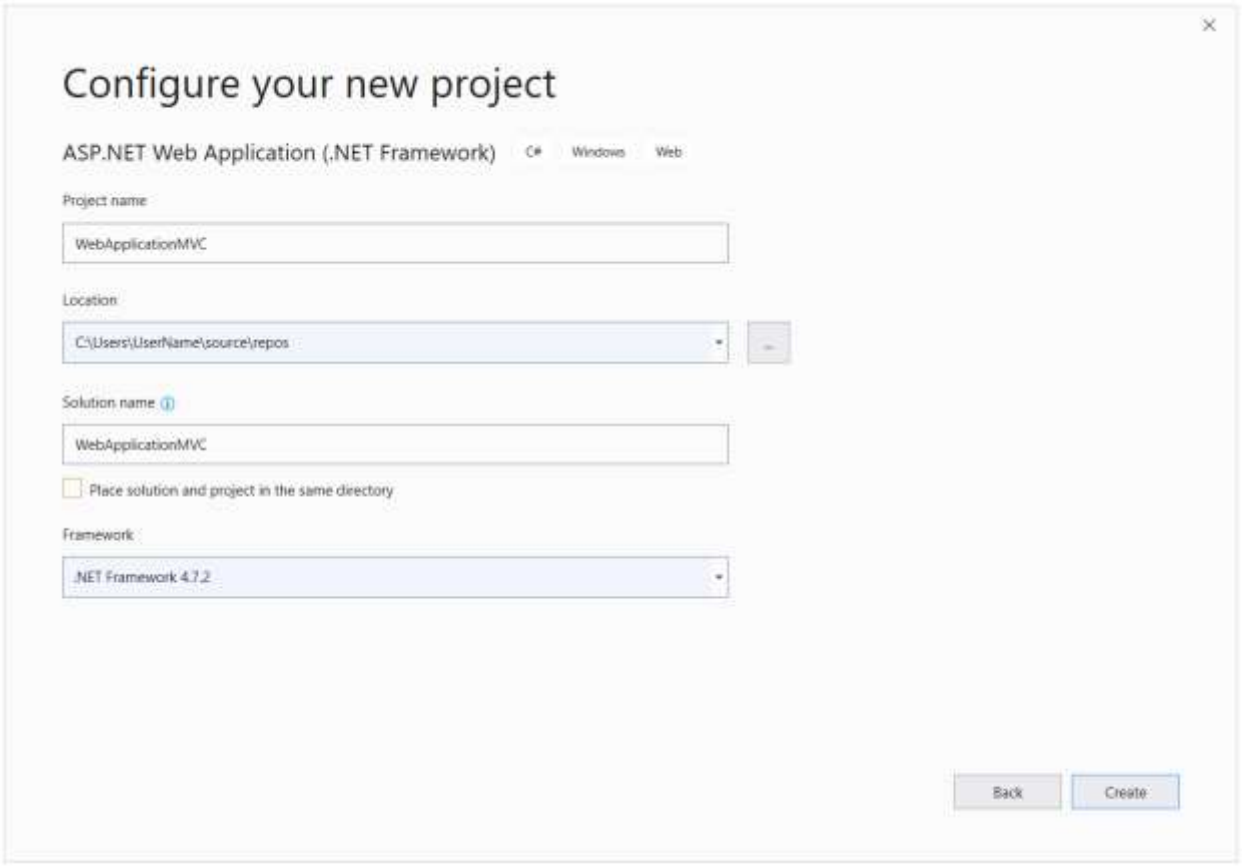
This section describes how to configure the Syncfusion ASP.NET MVC components into the ASP.NET MVC applications using Visual Studio 2019.

To add our Syncfusion MVC controls into the new ASP.NET MVC5 application by making use of the **Syncfusion NuGet Packages**, refer to the following steps.

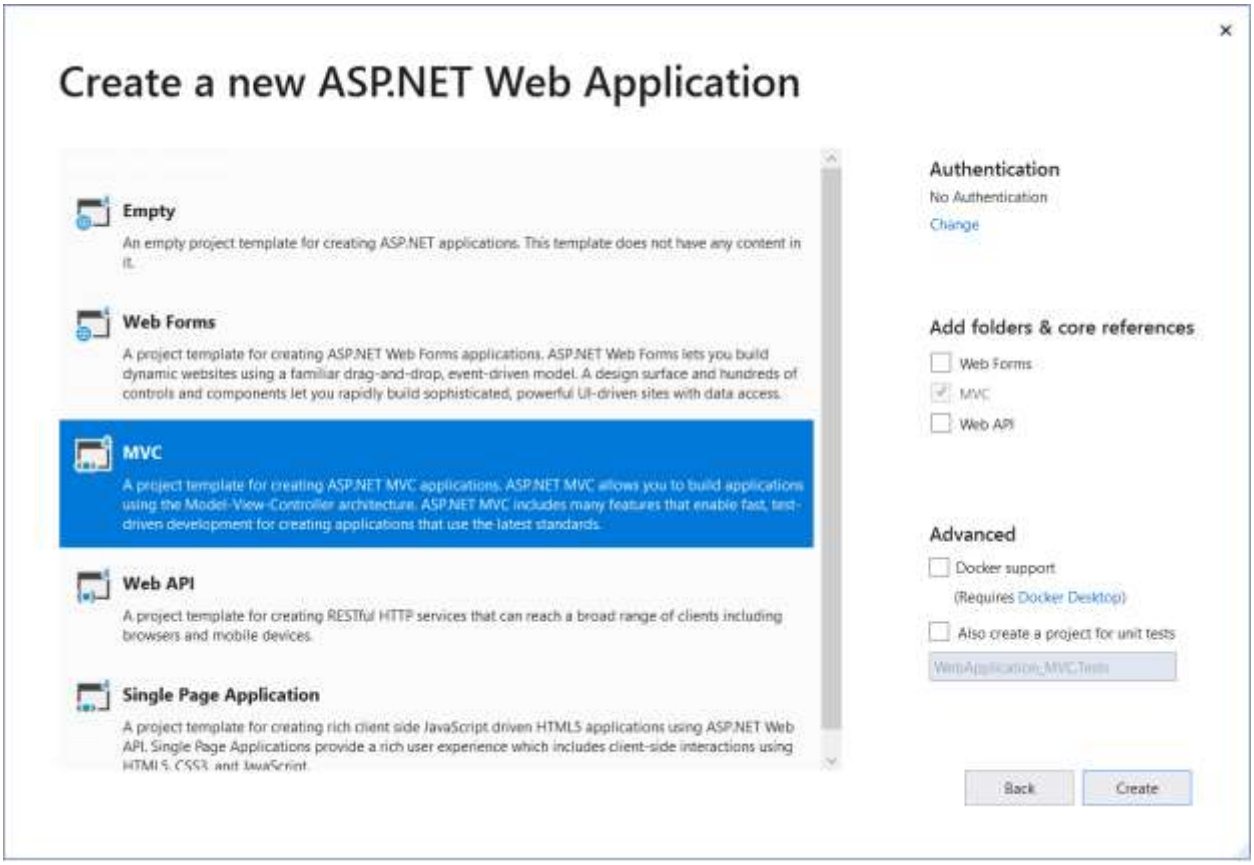
- Open Visual Studio 2019 and Create a new ASP.NET Web Application project and click on the **Next** button.



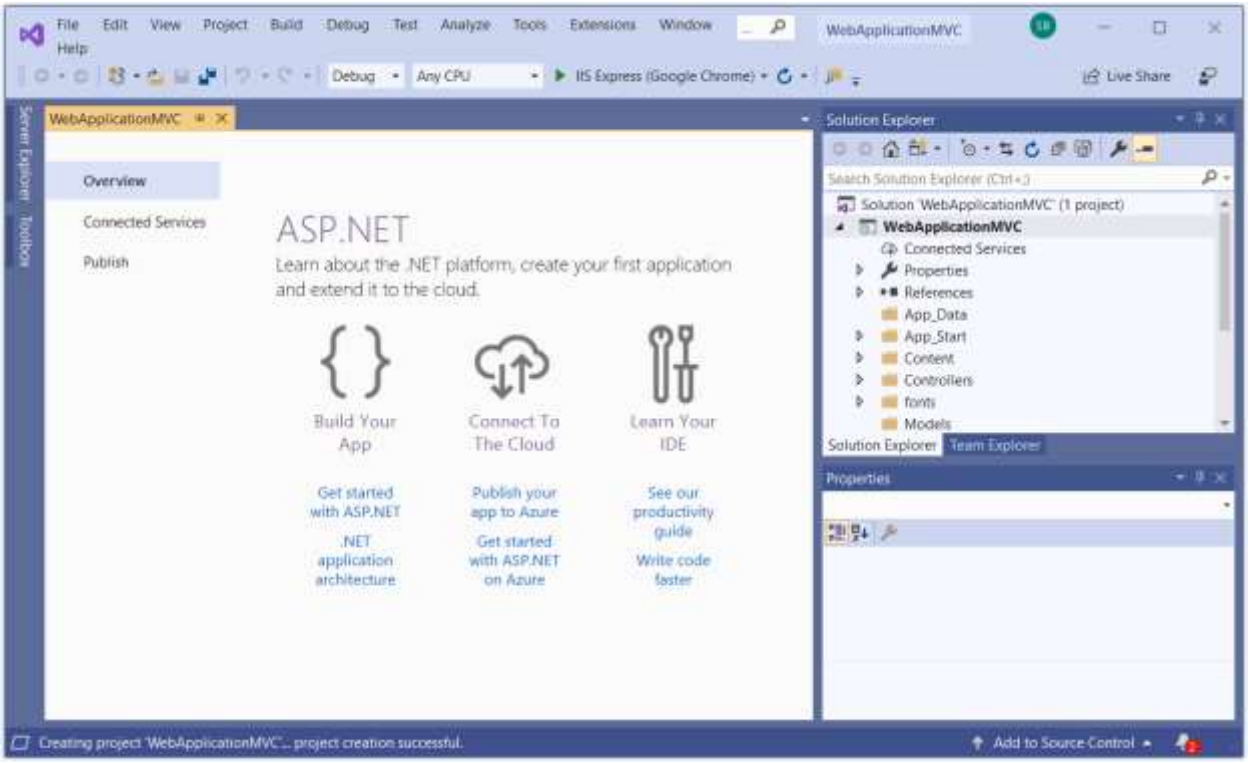
- Name your project and configure its settings and after click on the **Create** button.



- Select **MVC** project template, then click on **Create** button.



- A new project MVC application has been created successfully using Visual Studio 2019 (IDE).

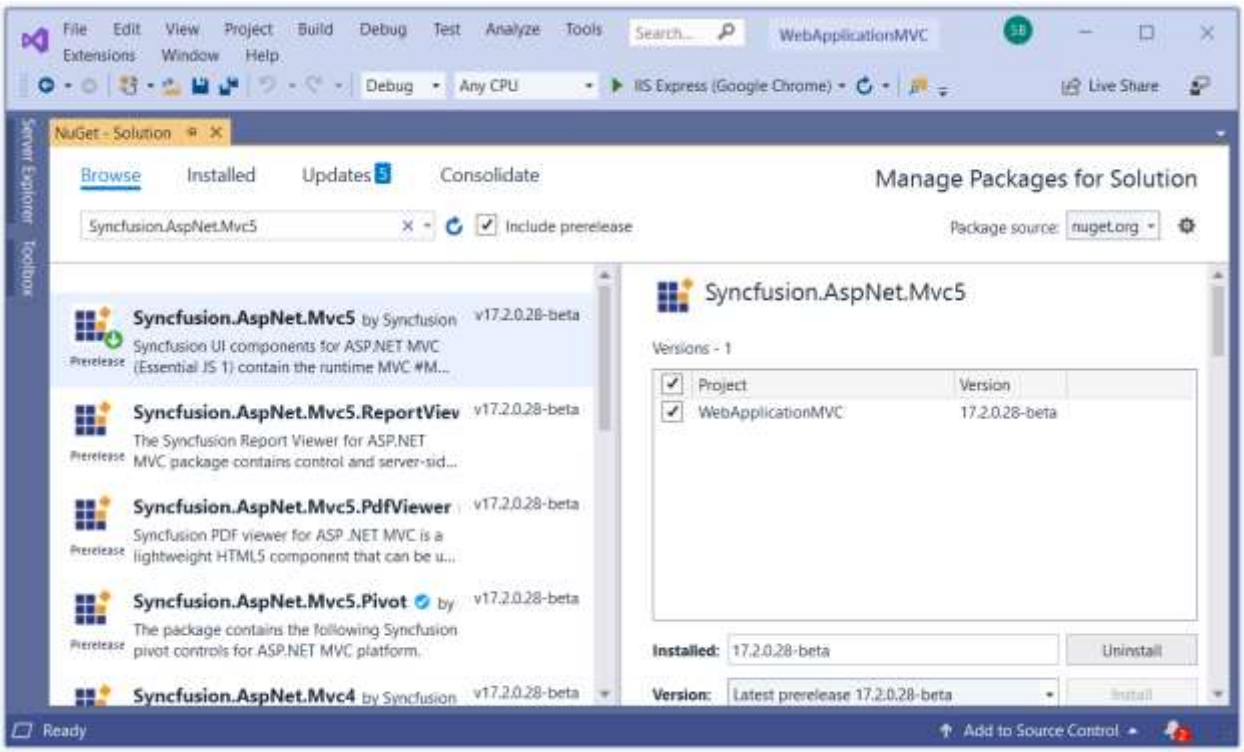


- After project creation, install the Syncfusion Packages in your application.
- Select the **Tools->NuGet Package Manager->Package Manager settings** the dialog window will open.
- Navigate to the **NuGet Package Manager->Package Sources** from the options dialog.
- Click the Add button to create the new package sources.
- Select the newly created Package Source and rename the source name using the Name input box.

Name: Name of the package that listed in Available package sources

Source: Syncfusion ASP.NET MVC NuGet Package feed URL

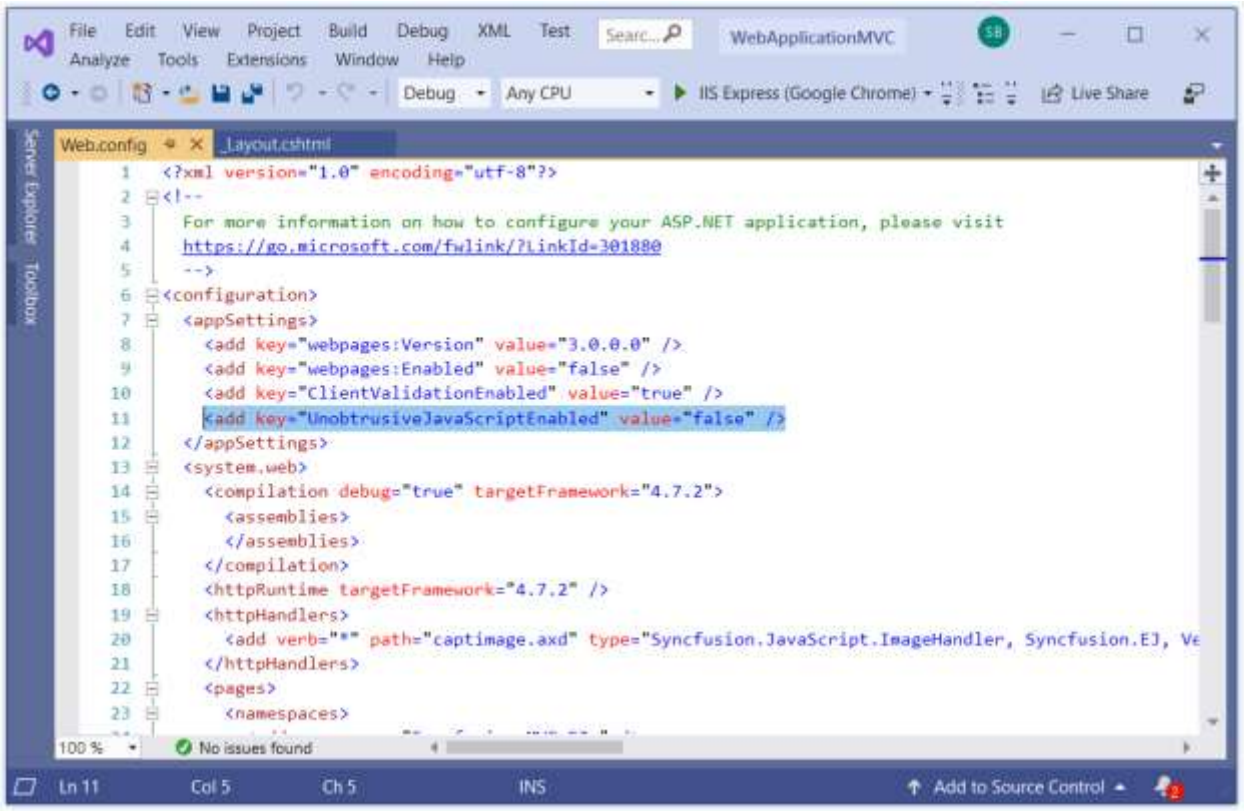
- The Syncfusion ASP.NET MVC NuGet Package can be installed from https://www.nuget.org/packages/Syncfusion.AspNet.Mvc5
- Select the Update and then click the OK button. The package's source get added to the list of available package sources.
- Right click your project references and then select "**Manage NuGet Package**" option. In this window choose the "**Syncfusion Packages registered name**" from the package source dropdown. And check the "**include prerelease**" option.
- Now, our Syncfusion Packages will list in this window. Select and install the "**Syncfusion.EJ.AspNet.Mvc5**" package from this list.



- Then the packages will get installed and it will be automatically referred to your application.
- While initial creation, by default, the **Unobtrusive** setting is enabled in your applications **web.config** file. Change its value to **false** in your application as shown in the following:

ASPX-CS

```
<appSettings>
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
```

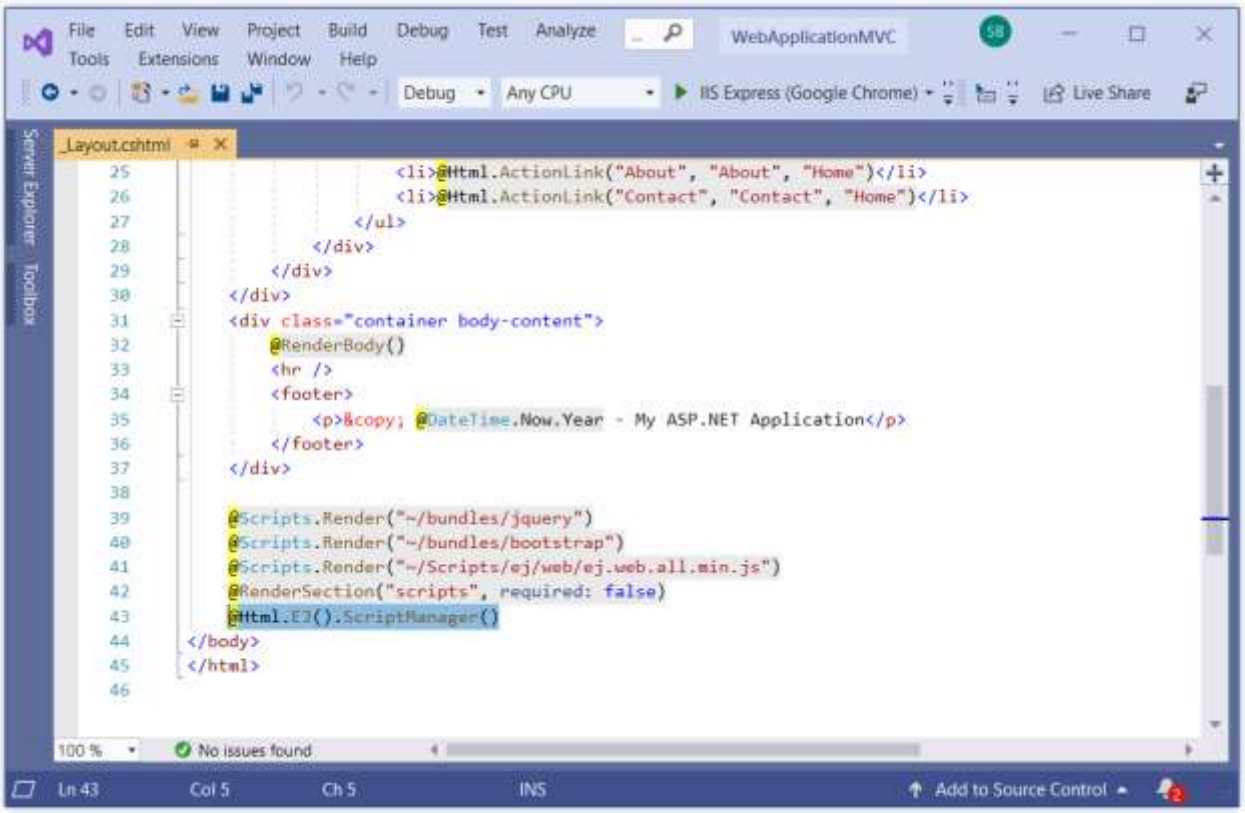


- Add the script manager code in the ***Layout.cshtml file present within the ~/Views/Shared folder of your application. Add it before the closing body tag in the*** Layout.cshtml file.

ASPX-CS

```
<body>
@RenderSection("scripts", required: false)
@Html.EJ().ScriptManager()
</body>
```

Copyright © 2001 - 2020 Syncfusion Inc.



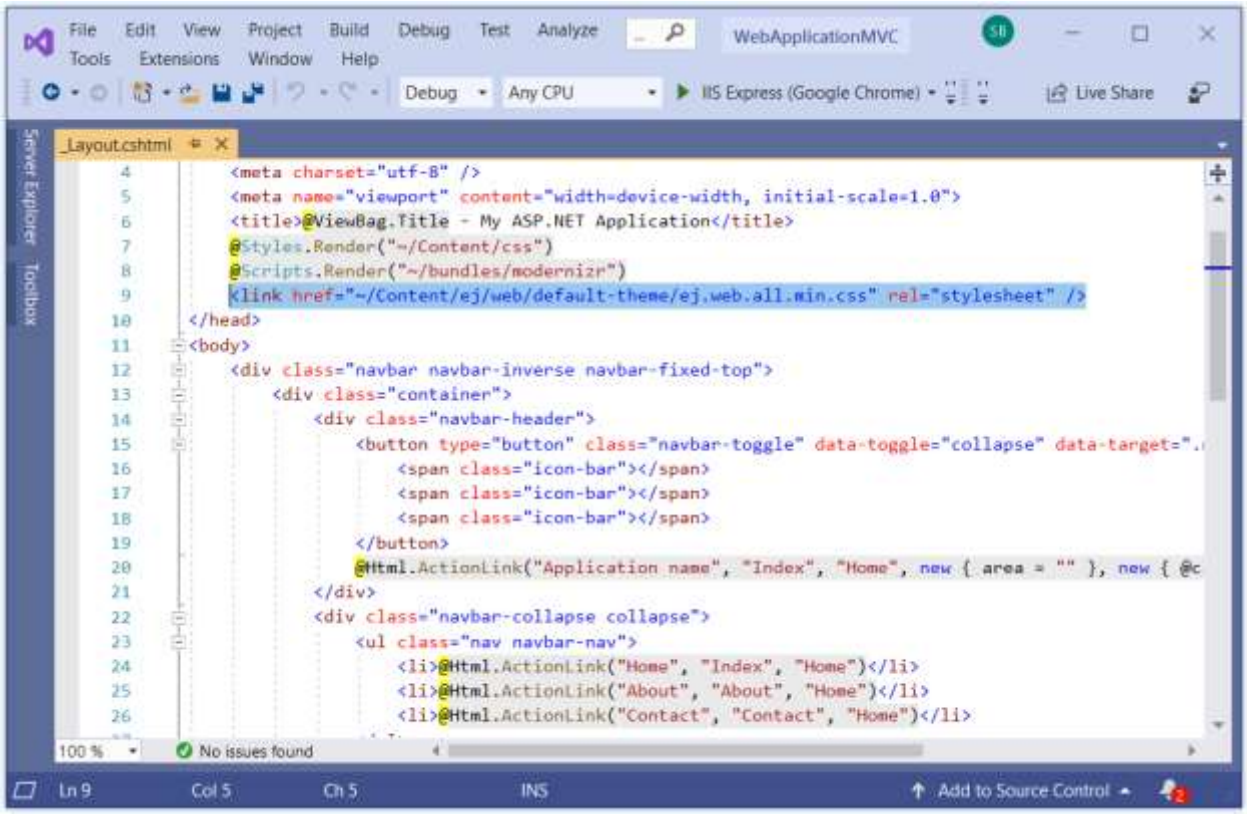
If **unobtrusive** is set to true in the application, the script manager can be excluded, as the control is initialized using HTML5 attributes. Also for control rendering, you need to include **ej.unobtrusive.min.js** file along with ej.web.all.min.js in **_Layout.cshtml** page. Refer to the unobtrusive section.

- Syncfusion specific stylesheets are loaded into the **Content** folder of your application, include the below specified theme reference **(default-theme/ej.web.all.min.css)** file in the **~/Views/Shared/_Layout.cshtml** file, within the head section as this file contains the default theme styles applied for all the Syncfusion MVC controls.

ASPX-CS

```
<head>
<title>@ViewBag.Title</title>
@Styles.Render("~/Content/ej/web/default-theme/ej.web.all.min.css")
</head>
```

Copyright © 2001 - 2020 Syncfusion Inc.



- It is mandatory to include the reference to the required JavaScript files in your **_Layout.cshtml**, so as to render the Syncfusion MVC controls properly.

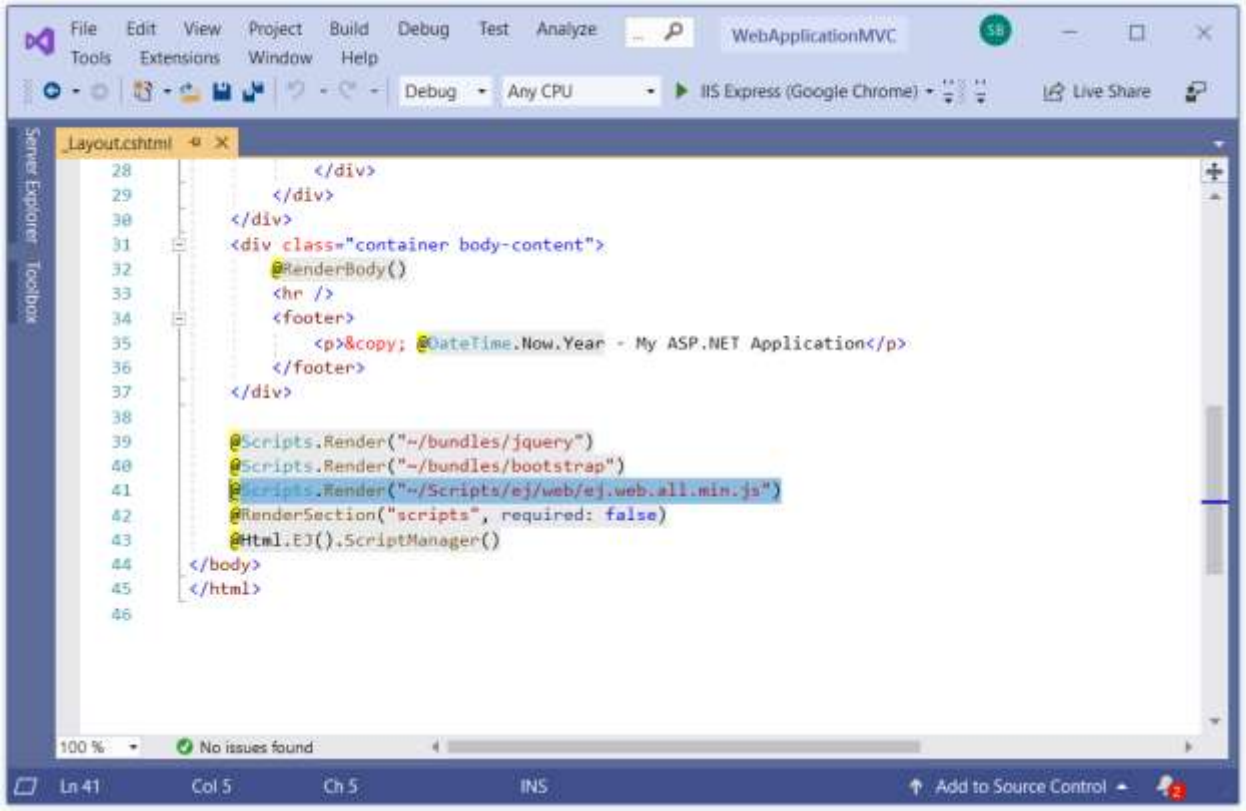
ASPX-CS

```
<head>
<meta charset="utf-8" />
<title>@ViewBag.Title - My ASP.NET MVC Application</title>
@Styles.Render("~/Content/ej/web/bootstrap-theme/ej.web.all.min.css")
</head>
<body>
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@Scripts.Render("~/Scripts/jsrender.min.js")
@Scripts.Render("~/Scripts/ej/web/ej.web.all.min.js")
@RenderSection("scripts", required: false)
@Html.EJ().ScriptManager();
</body>
```

The order of reference for the script files made in the above section should be maintained. Any change in order reference may cause rendering issues in EJ Controls.

If your application contains duplicate/multiple references to the jQuery files, remove it as the explicit reference to the **jquery-{version}.js** script file that is added to the application as specified above.

Copyright © 2001 - 2020 Syncfusion Inc.



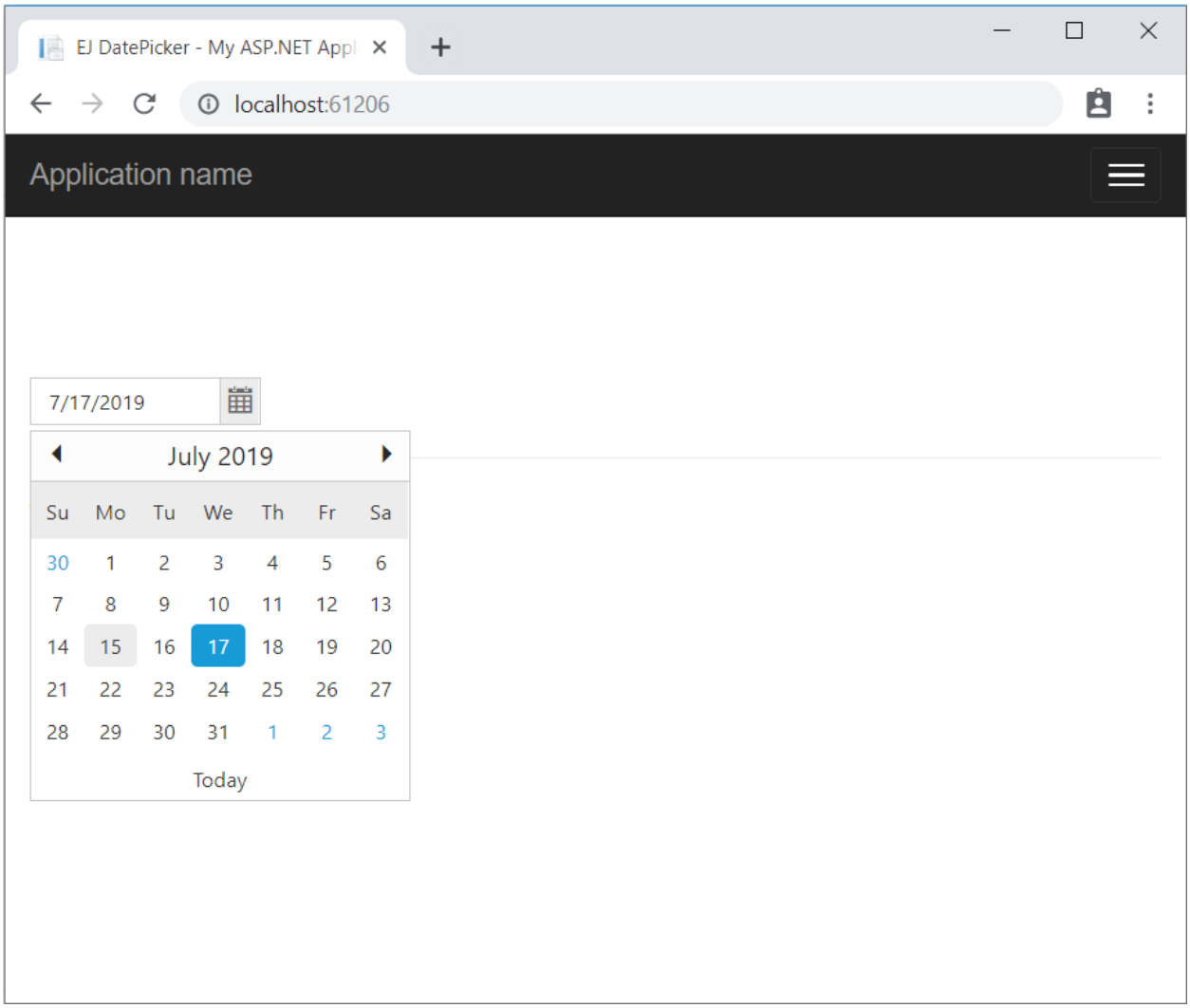
- Now you can add the control **DatePicker** in the **Index.cshtml** file present within **~/Views/Home** folder.

ASPX-CS

```
@Html.EJ().DatePicker("MyFirstDatepicker")
```

- Compile and execute the application. You can able to see the below output in the browser.

Copyright © 2001 - 2020 Syncfusion Inc.



Utilities

The Syncfusion MVC Extension provides you quick access, so that you can create or configure the Syncfusion MVC projects. The Syncfusion ASP.NET MVC Extensions has the following features.

- Syncfusion Sample Creator for ASP.NET MVC
- Syncfusion Project Conversion for ASP.NET MVC
- Syncfusion Project Migration for ASP.NET MVC

Project Conversion

The Project Conversion is a Visual Studio add-in that converts an existing ASP.NET MVC Project into a Syncfusion ASP.NET MVC Project by adding the required assemblies and resource files.

Convert into Syncfusion ASP.NET MVC (Web) project

The following steps help you to use the Syncfusion Project Conversion in the existing ASP.NET MVC (Web) Project.

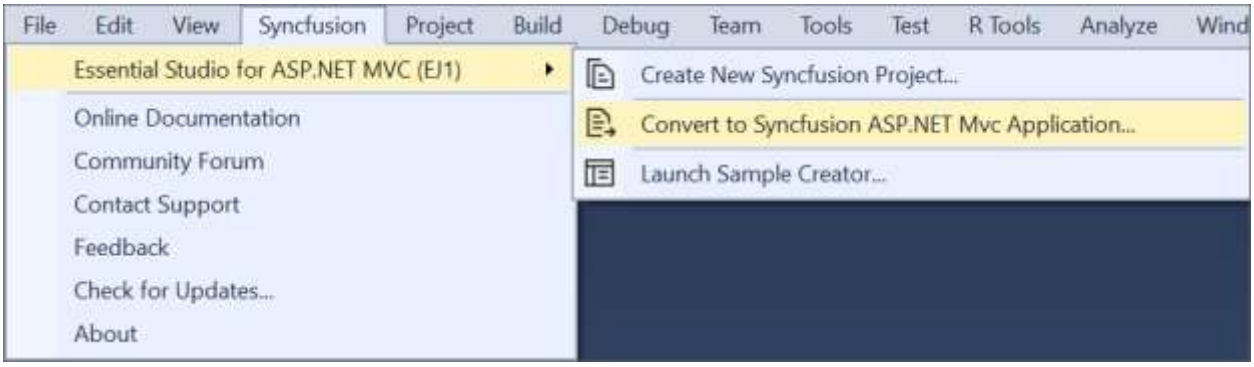
1. Open an existing Microsoft MVC Project or create a new Microsoft MVC Project.

Copyright © 2001 - 2020 Syncfusion Inc.

2. To open Project Conversion Wizard, follow either one of the options below:

Option 1:

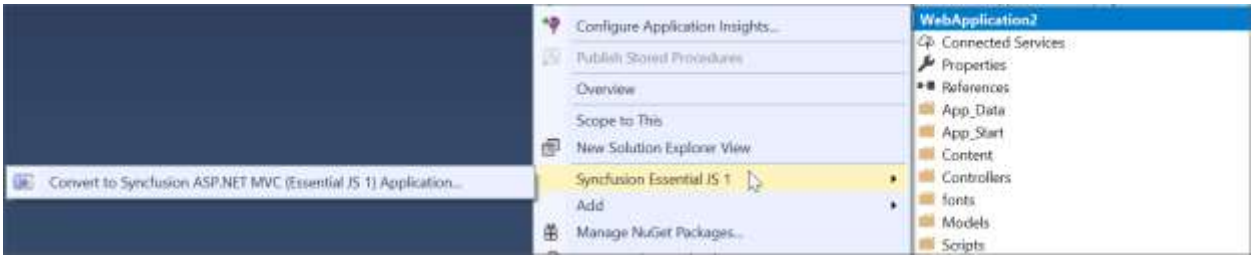
Click **Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ1) > Convert to Syncfusion ASP.NET MVC Application…** in **Visual Studio**.



Note: In Visual Studio 2019, Syncfusion menu available under Extension in Visual Studio menu.

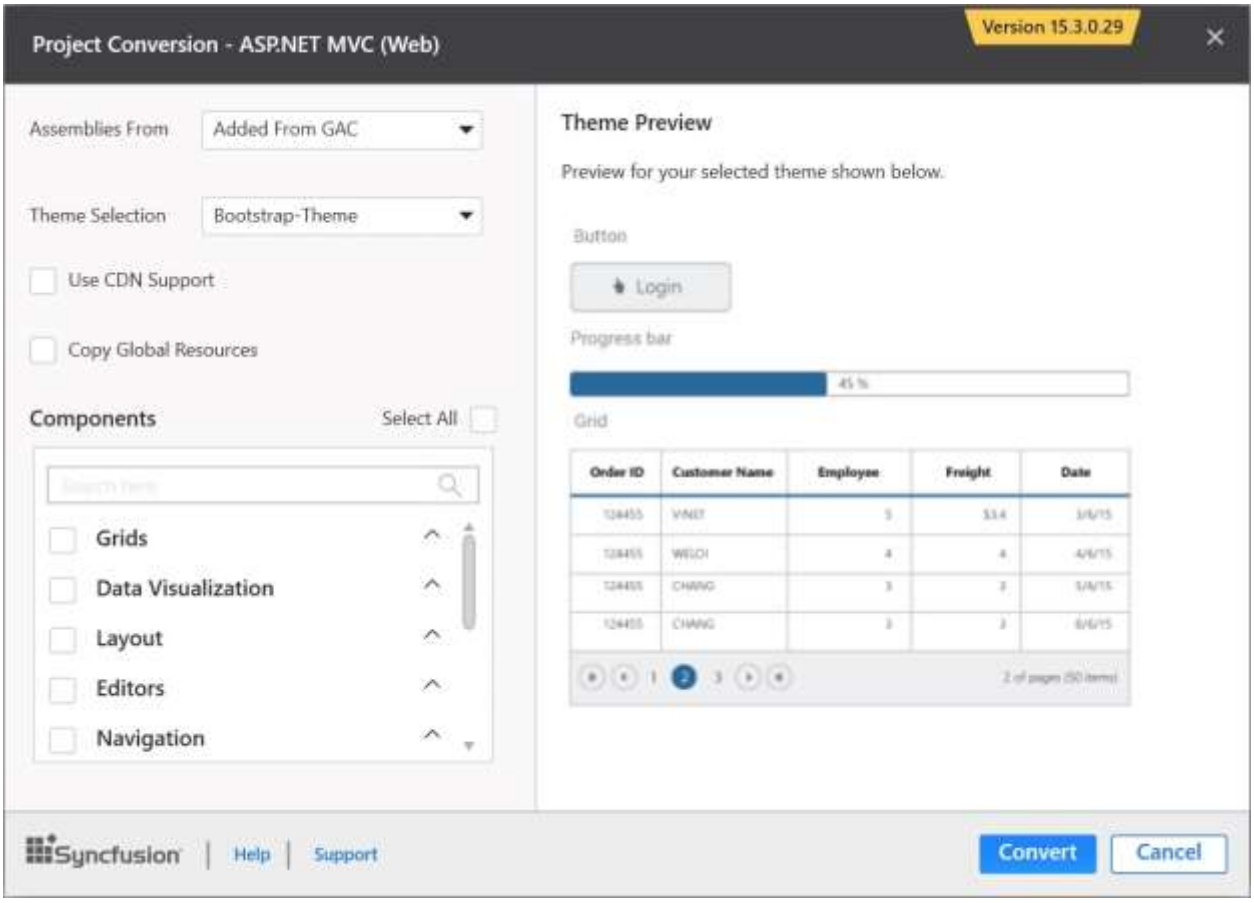
Option 2:

Right-click the Project from Solution Explorer, select **Syncfusion Essential JS 1**, and choose **Convert to Syncfusion MVC (Essential JS 1) Application...** Refer to the following screenshot for more information.



3. Project Conversion Wizard opens so that you can configure the project.

Copyright © 2001 - 2020 Syncfusion Inc.

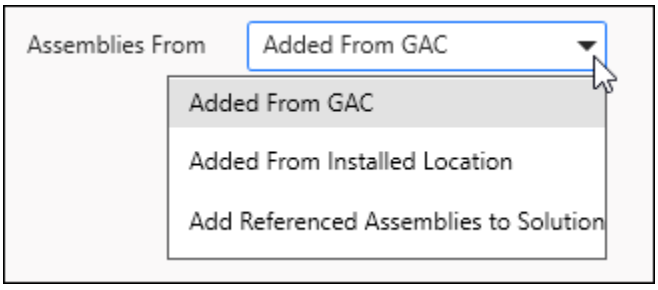


The following configurations are used in the Project Conversion Wizard.

Assemblies From:

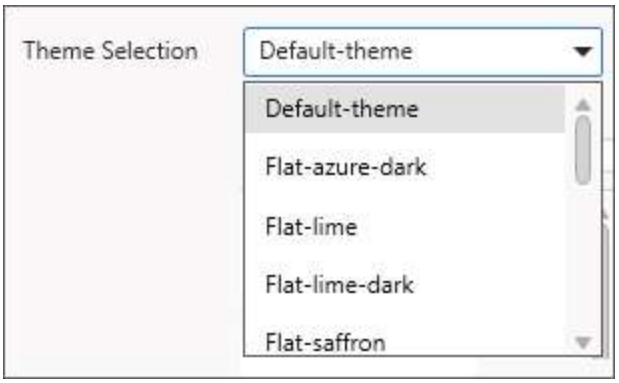
Choose the assembly location:

1. Added From GAC: Refer to the assemblies from the Global Assembly Cache.
2. Added from Installed Location: Refer to the assemblies from the Syncfusion Installed locations.
3. Add Referenced Assemblies to Solution: Copy and refer to the assemblies from project's solution file lib directory.



Choose the Theme:

The master page of project will be updated based on the selected theme. The Theme Preview section shows the controls preview before converting to a Syncfusion project.



Choose CDN Support:

The master page of the project will be updated based on the required Syncfusion CDN links.

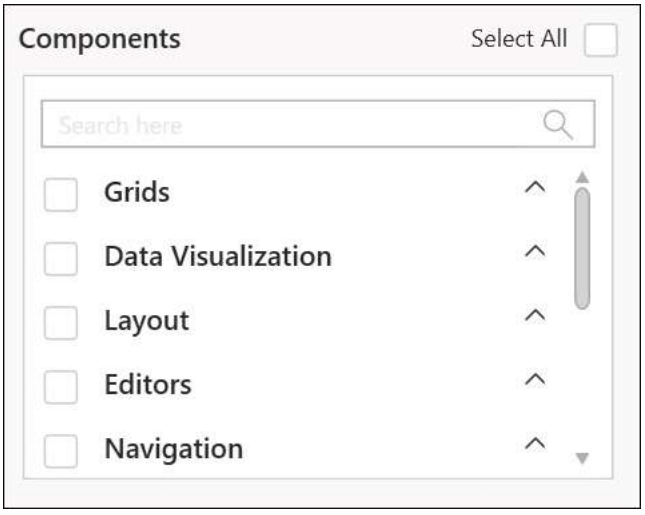


Choose Copy Global Resources:

The localization culture files will be shipped into Scripts\ej\i18n directory of the project.

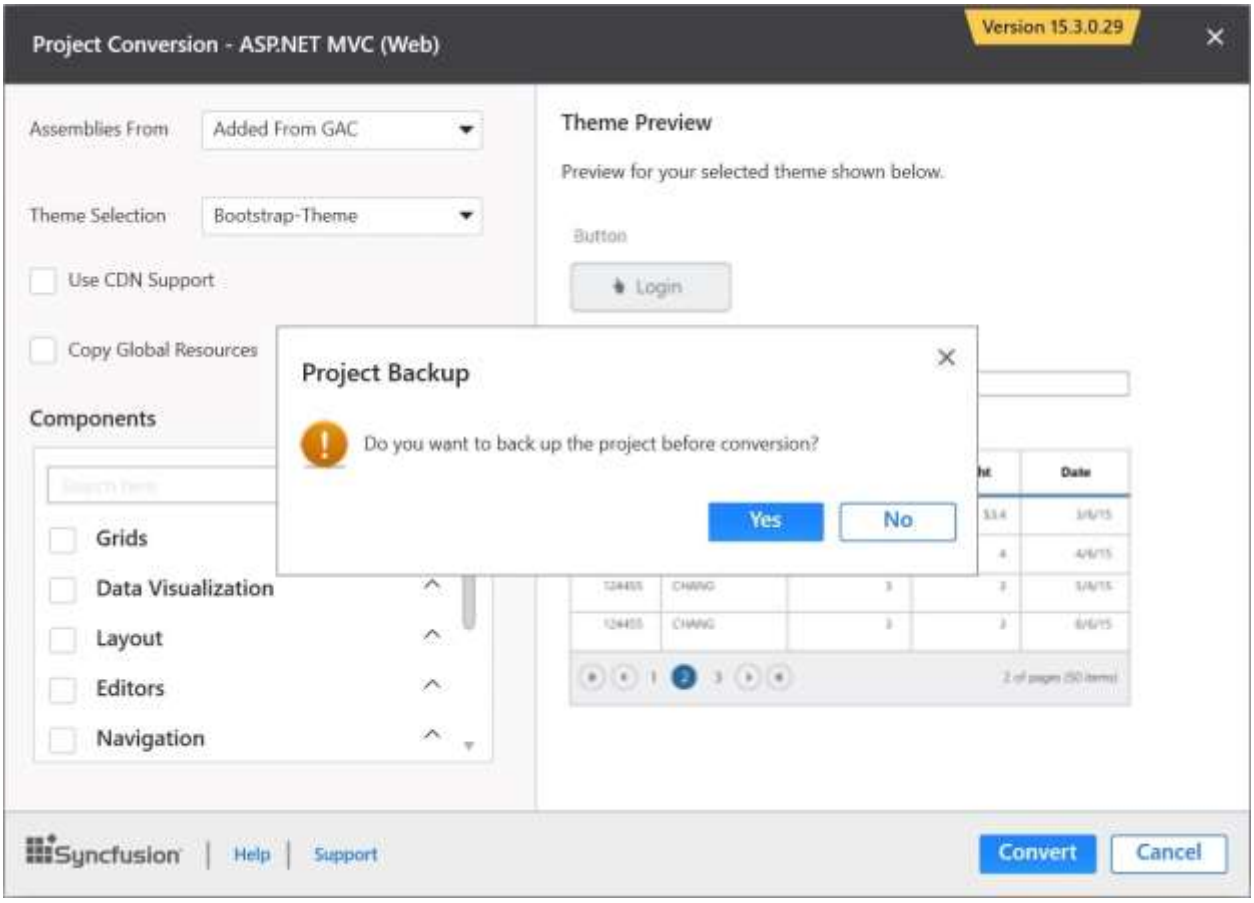


4. Choose the required controls from Components section and Click the **Convert** button to convert it into a Syncfusion Project.

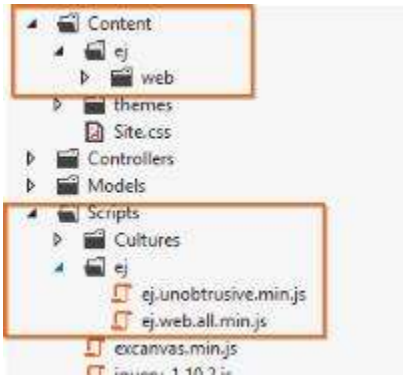


5. The **Project Backup** dialog will be opened. Click yes, to backup the current project before converting it to Syncfusion project. Click No to convert the project to Syncfusion project without backup.

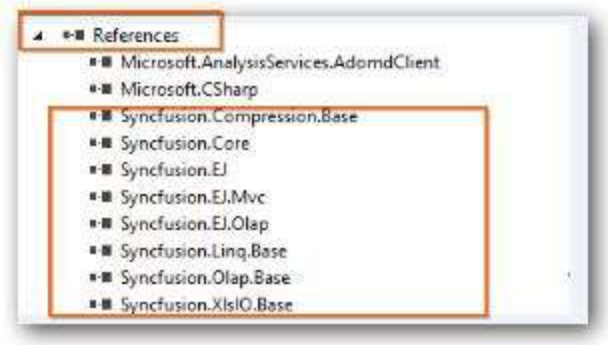
Copyright © 2001 - 2020 Syncfusion Inc.



6. The required Syncfusion Reference Assemblies, Scripts and CSS are included in the MVC Project. For more information, refer to the following screenshots.



Copyright © 2001 - 2020 Syncfusion Inc.




```
<compilation debug="true" targetFramework="4.5">
  <assemblies>
    <add assembly="Syncfusion.EJ.Export, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Compression.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"
    <add assembly="Syncfusion.EJ, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.EJ.Mvc, Version=14.3500.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Linq.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.XlsIO.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.EJ.Pivot, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Olap.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
  </assemblies>
</compilation>
```

Rendering Control after Syncfusion ASP.NET MVC (Web) Conversion:

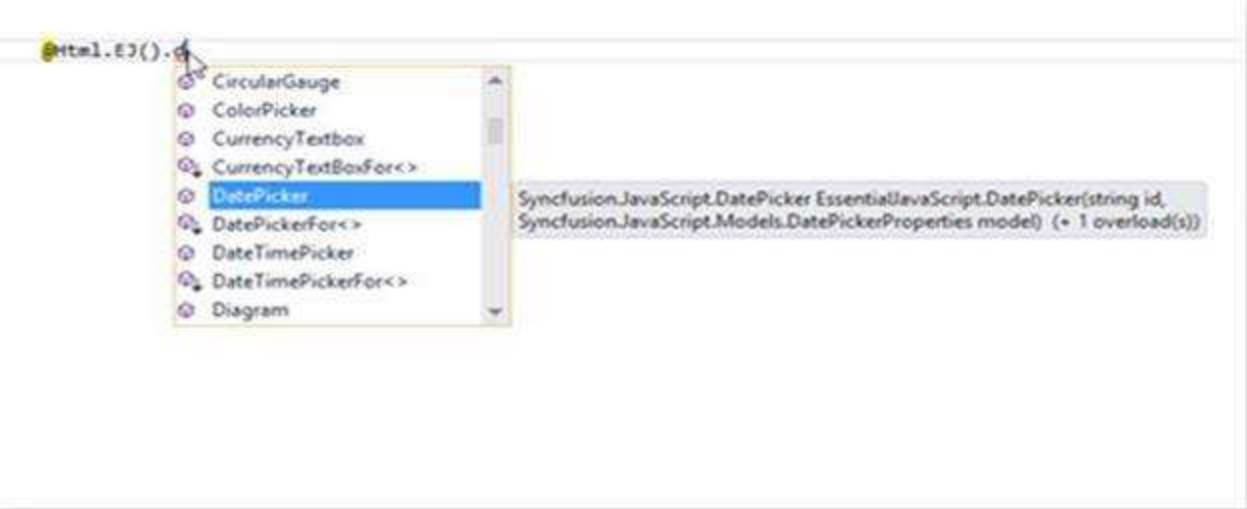
Once you convert your ASP.NET MVC project to Syncfusion ASP.NET MVC Project, perform the following steps to render the Syncfusion Controls to your project.

1. The CSS, Scripts, Syncfusion References, and required Web.config file entries are added to your project by Syncfusion ASP.NET MVC Conversion.
2. Add the required Script and CSS files references in the master page (_Layout.cshtml/Layout.vbhtml file). For more information, please refer to the following screenshot.

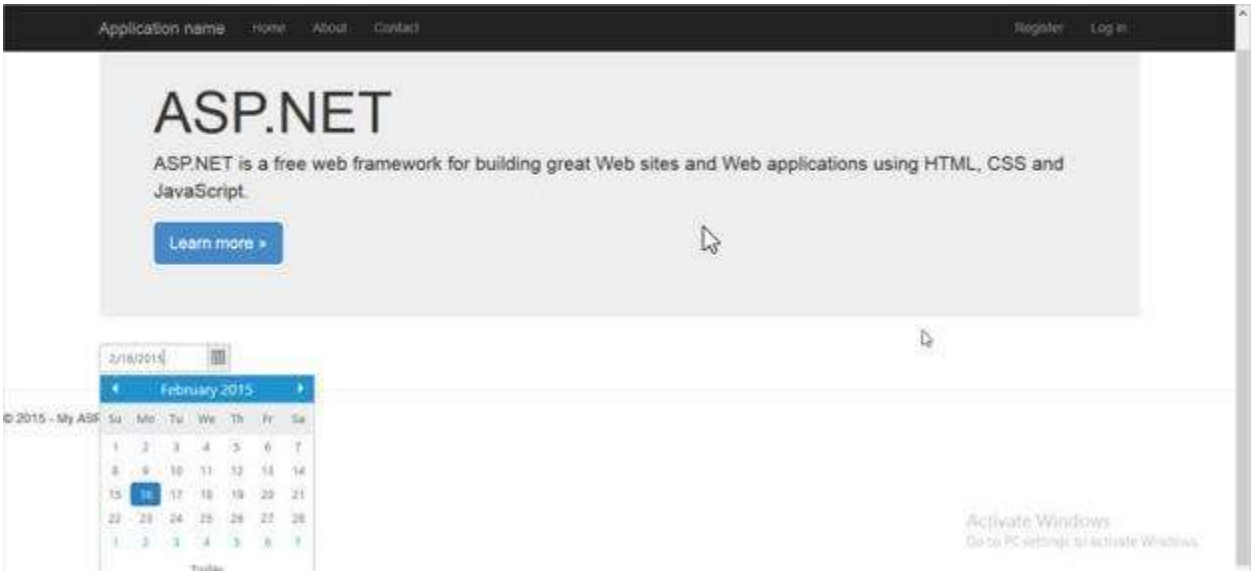
```
<title>@ViewBag.Title - My ASP.NET Application</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
<link href="@Url.Content("~/Content/ej/web/Default-theme/ej.widgets.all.min.css")" rel="stylesheet" />
<script src="@Url.Content("~/Scripts/jquery-1.10.2.min.js")"></script>
<script src="@Url.Content("~/Scripts/jsrender.min.js")"></script>
<script src="@Url.Content("~/Scripts/jquery.easing-1.3.min.js")"></script>
<script src="@Url.Content("~/Scripts/jquery.globalize.min.js")"></script>
<script src="@Url.Content("~/Scripts/ej/ej.web.all.min.js")"></script>
<script src="@Url.Content("~/Scripts/ej/ej.unobtrusive.min.js")"></script>
```

3. Now, include the Syncfusion controls to your project. For more information, refer to the following screenshot.

Copyright © 2001 - 2020 Syncfusion Inc.



4. Run the project and the following output will be displayed.



Project Migration

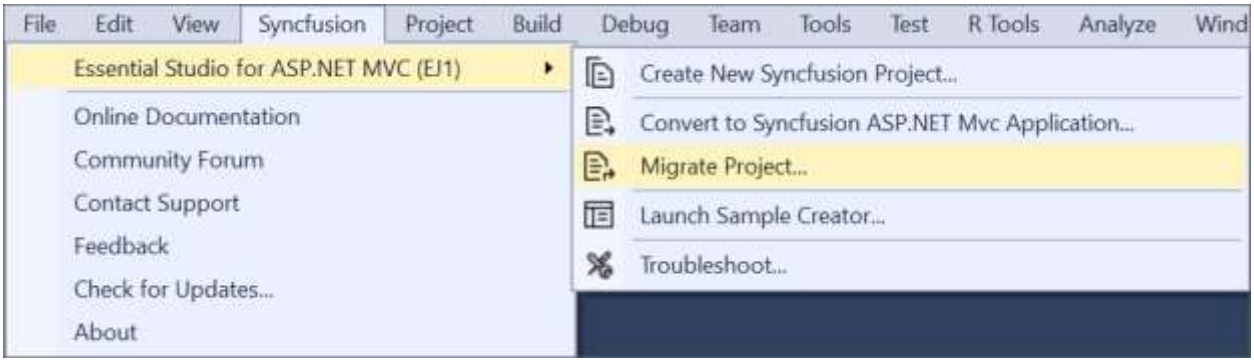
The Project Migration is a Visual Studio add-in that helps to migrate the existing Syncfusion ASP.NET MVC (Web) project from one Syncfusion version to another Syncfusion version.

The following steps help you to migrate from one version to another version of your existing Syncfusion ASP.NET MVC(web) application.

1.To open Migration Wizard, follow either one of the options below:

Option 1:

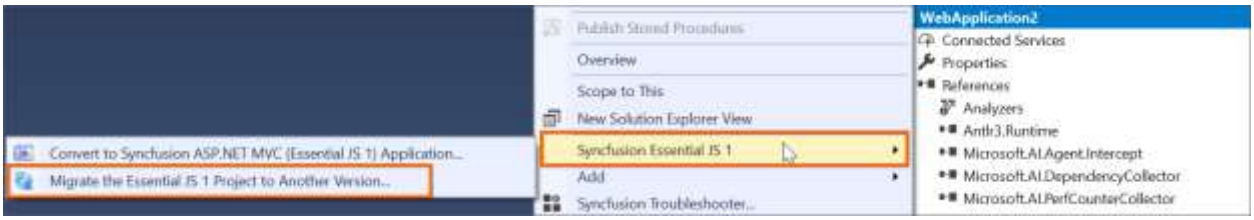
Click **Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ1) > Migrate Project…** in **Visual Studio**.



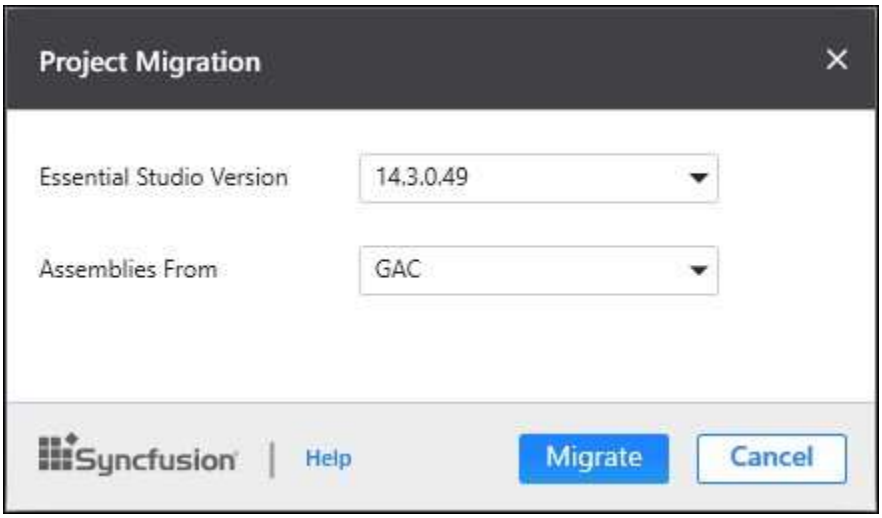
Note: In Visual Studio 2019, Syncfusion menu available under Extension in Visual Studio menu.

Option 2:

Right-click the **Syncfusion ASP.NET MVC Application** from Solution Explorer and select **Syncfusion Essential JS 1**. Choose **Migrate the Essential JS 1 Project to Another Version...**



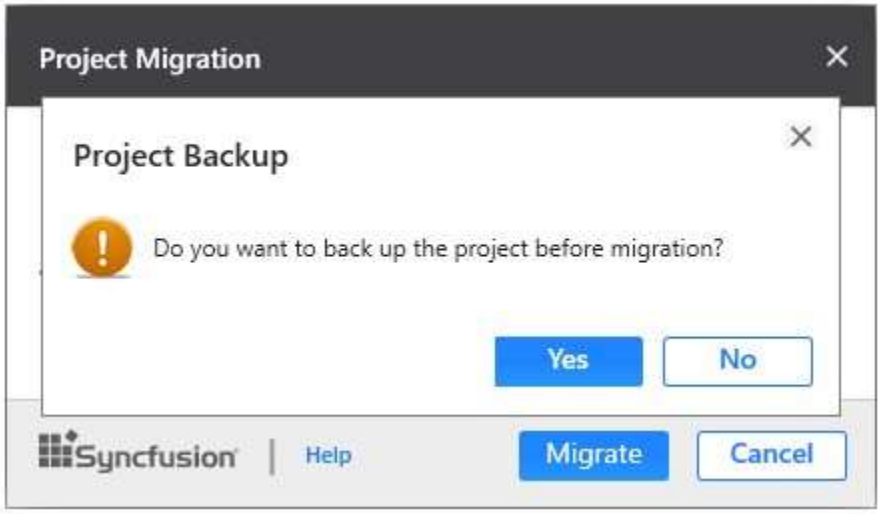
2. The Project Migration window appears. You can choose the required Syncfusion version that is installed in the machine, that can be either Syncfusion ASP.NET MVC.



3. The Project Migration window allows you to configure the following options:
    - Essential Studio Version: Select any version from the list of Installed Versions.
    - Assemblies From: Add the assembly to project from the following locations.
4. Added From GAC: Refer the assemblies from the Global Assembly Cache.
5. Added from Installed Location: Refer to the assemblies from the Syncfusion Installed locations.
6. Add Referenced Assemblies to Solution: Copy and refer to the assemblies from project's solution file lib directory.

Copyright © 2001 - 2020 Syncfusion Inc.

7. Click the Migrate Button and the **Project Backup** dialog will be opened. Click Yes to backup the current project before migrating the Syncfusion project. Click No to migrate the project to required Syncfusion version without backup.



5. The Syncfusion Reference Assemblies, Scripts and CSS are updated to the corresponding version in the project.

Sample Creator

Sample Creator is the utility that allows you to create Syncfusion ASP.NET MVC Projects along with the samples based on Controls and Features selection.

Create Syncfusion MVC Project from Sample Creator

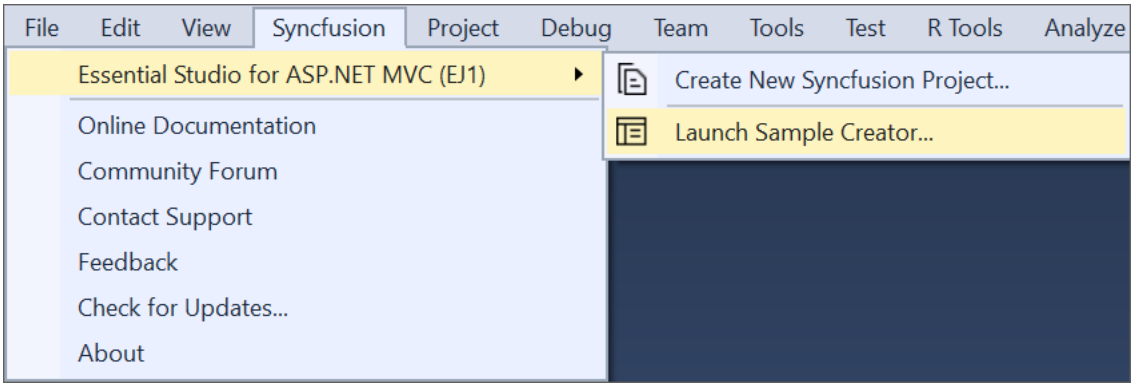
Sample Creator can be download from the Syncfusion Dashboard. After installing the complete Essential Studio suite or ASP.NET MVC setup, follow the given steps:

1. To launch ASP.NET MVC (Essential JS 1) Sample Creator application, follow either one of the options below:

Option 1:

Click **Syncfusion Menu** and choose **Essential Studio for ASP.NET MVC (EJ1) > Launch Sample Creator…** in **Visual Studio**.

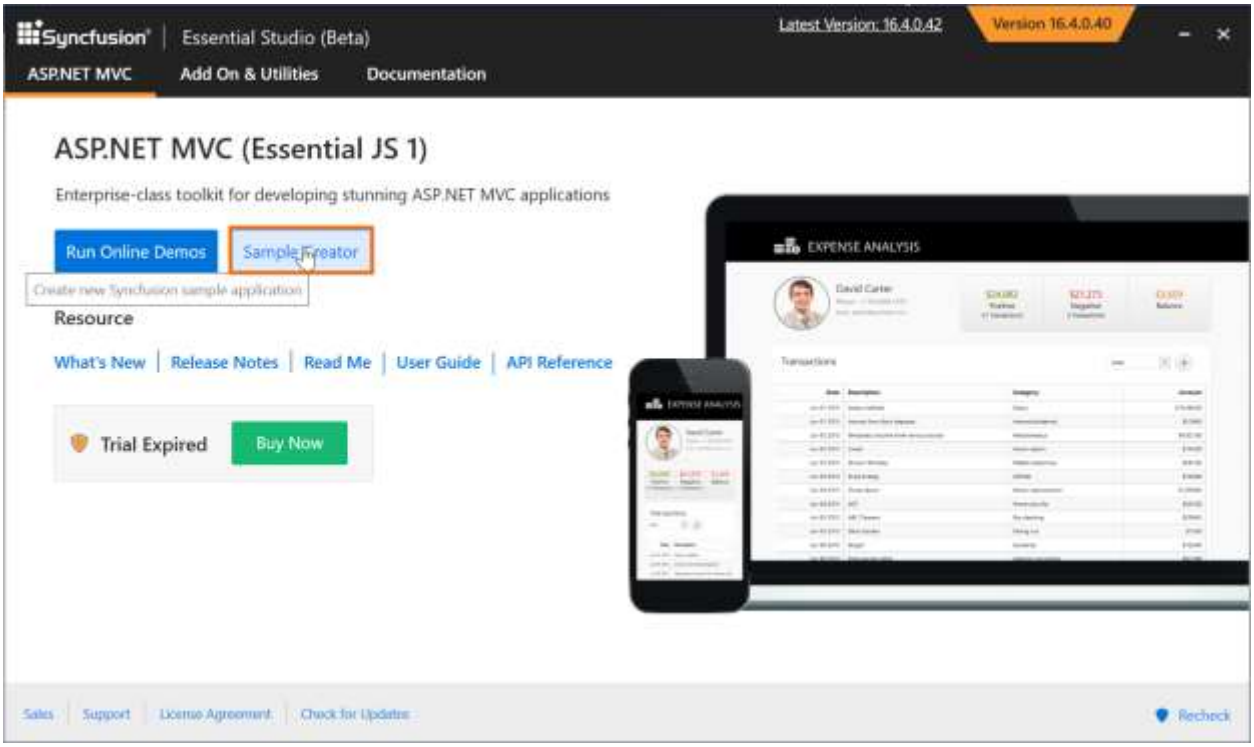
Copyright © 2001 - 2020 Syncfusion Inc.



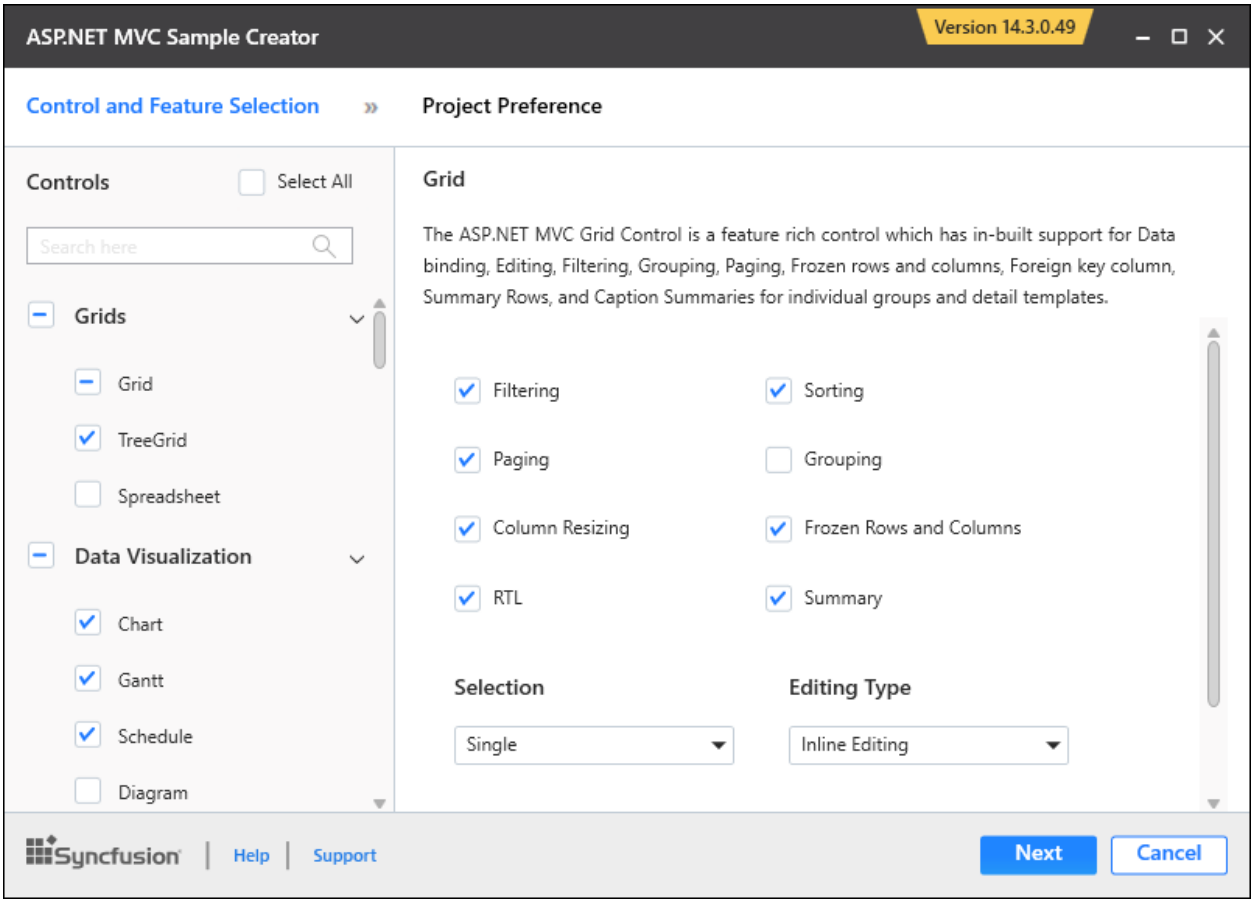
Note: In Visual Studio 2019, Syncfusion menu available under Extension in Visual Studio menu.

Option 2:

Launch the Syncfusion Essential Studio Dashboard and select the ASP.NET MVC (Essential JS 1)/ASP.NET MVC (Classic) platform. Select the Sample Creator button to launch the ASP.NET MVC (Essential JS 1) Sample Creator application. Refer to the following screenshot for more information.

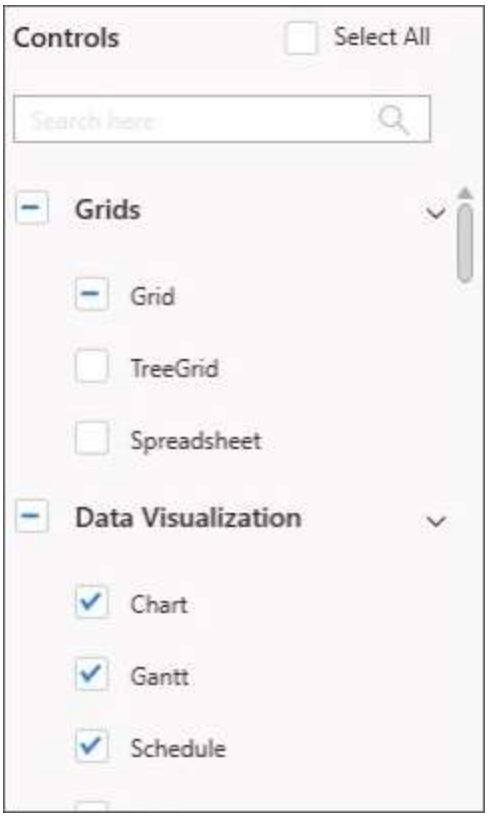


2. Syncfusion Sample Creator Wizard displays the **Controls and its Feature Selection** section.



Controls Selection

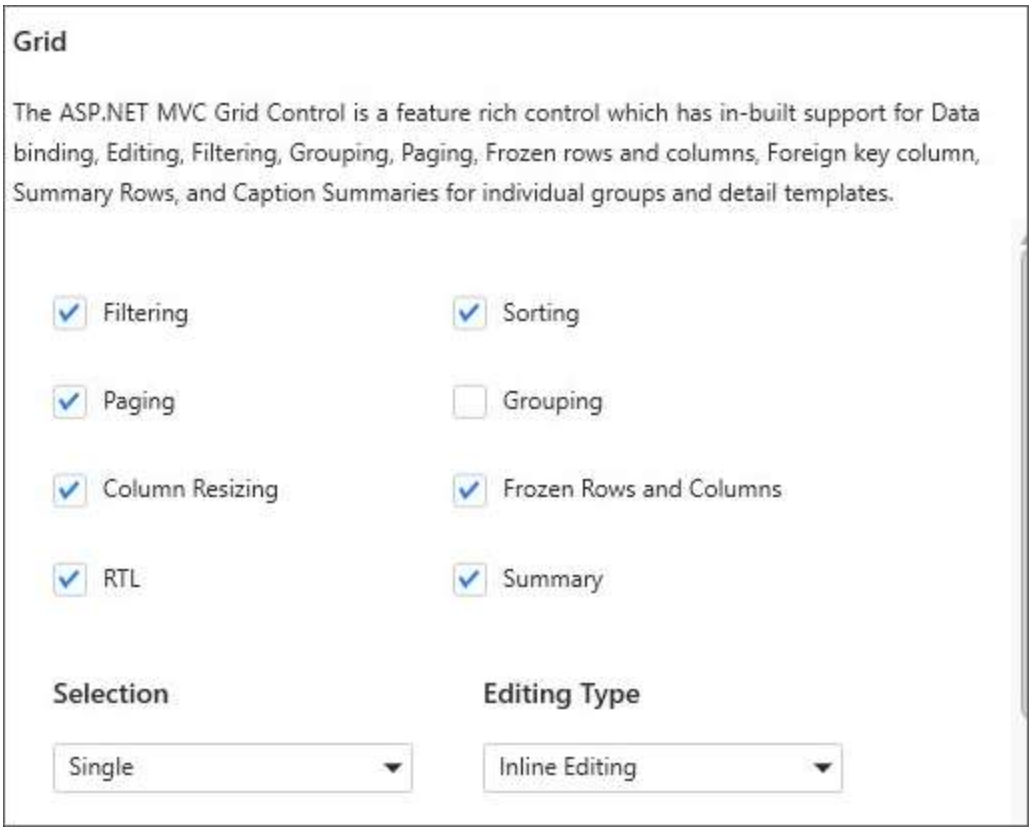
The Syncfusion ASP.NET MVC controls are listed here and you can choose the required controls. The controls are grouped product wise.



Feature Selection

Based on the controls, the Feature is enabled to choose the features of the corresponding controls.

Copyright © 2001 - 2020 Syncfusion Inc.

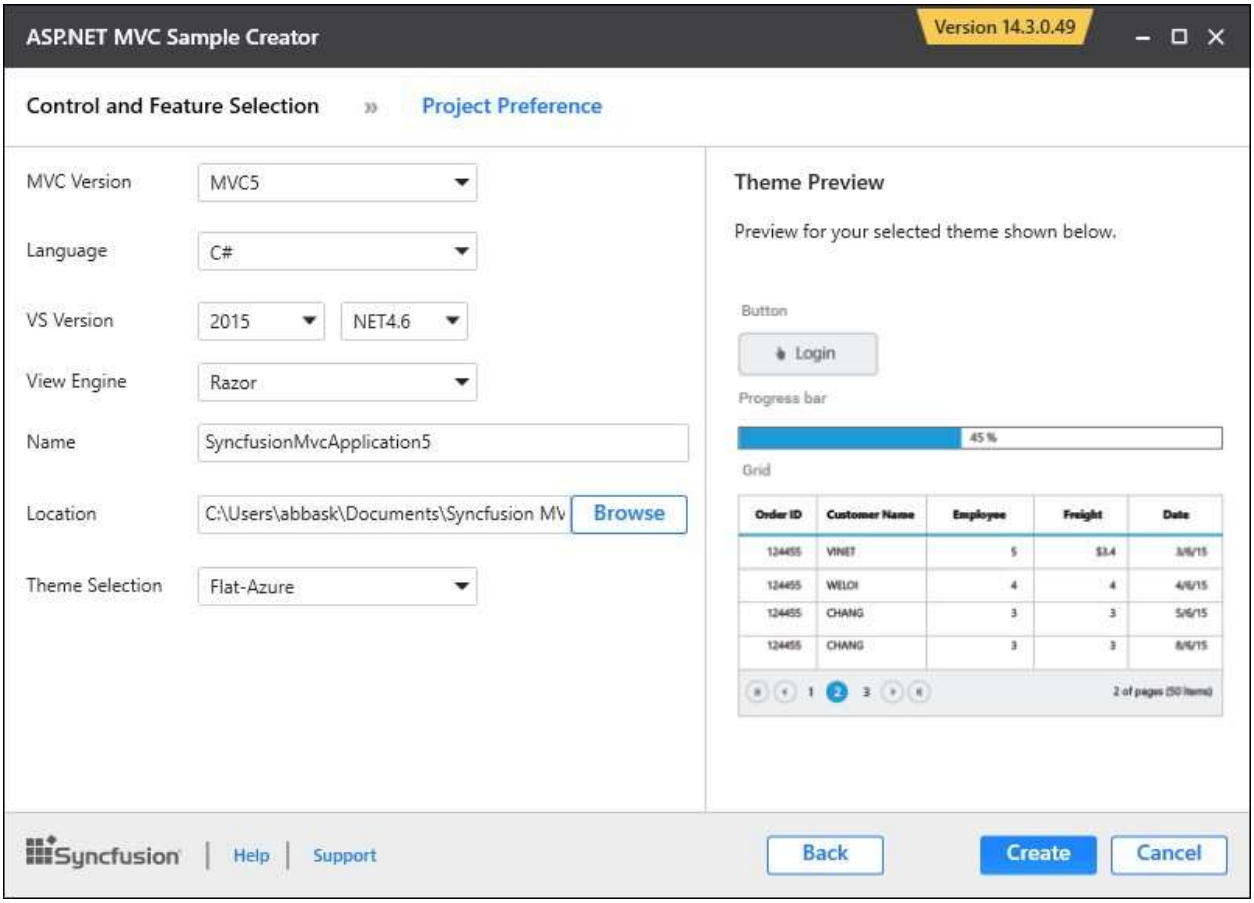


Project Configuration

You can configure the following project details in the Sample Creator.

- MVC Version: Choose the required MVC Version.
- Language: Select the language, either C# or VB.
- VS Version: Choose the Project version.
- .NET Framework: Choose the .NET Framework version.
- View Engine: Select either Razor or ASPX. By default, Syncfusion supports only Razor view engine for ASP.NET MVC projects.
- Compress Style Sheets: Option to compress style sheets.
- Compress Scripts: Option to compress the scripts.
- Name: Name your Syncfusion MVC Application.
- Location: Choose the target location of your project.
- Theme Selection: Choose the required theme. The Theme Preview section shows the controls preview before creating the Syncfusion project.

Copyright © 2001 - 2020 Syncfusion Inc.

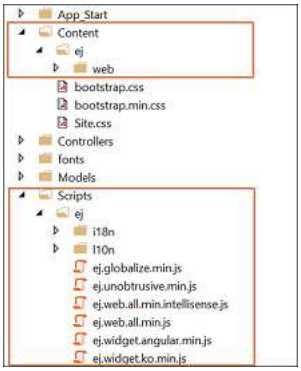


When you click the Create button, the new Syncfusion ASP.NET MVC project is created. The following is added in the project:

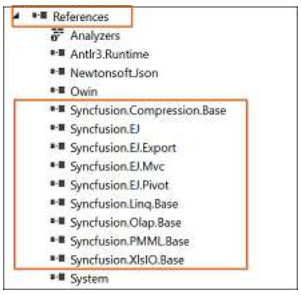
- Added the required Controller and View files in the project.



- Included the required Syncfusion ASP.NET MVC scripts and themes files.



- The required Syncfusion assemblies are added for selected controls under Project Reference.

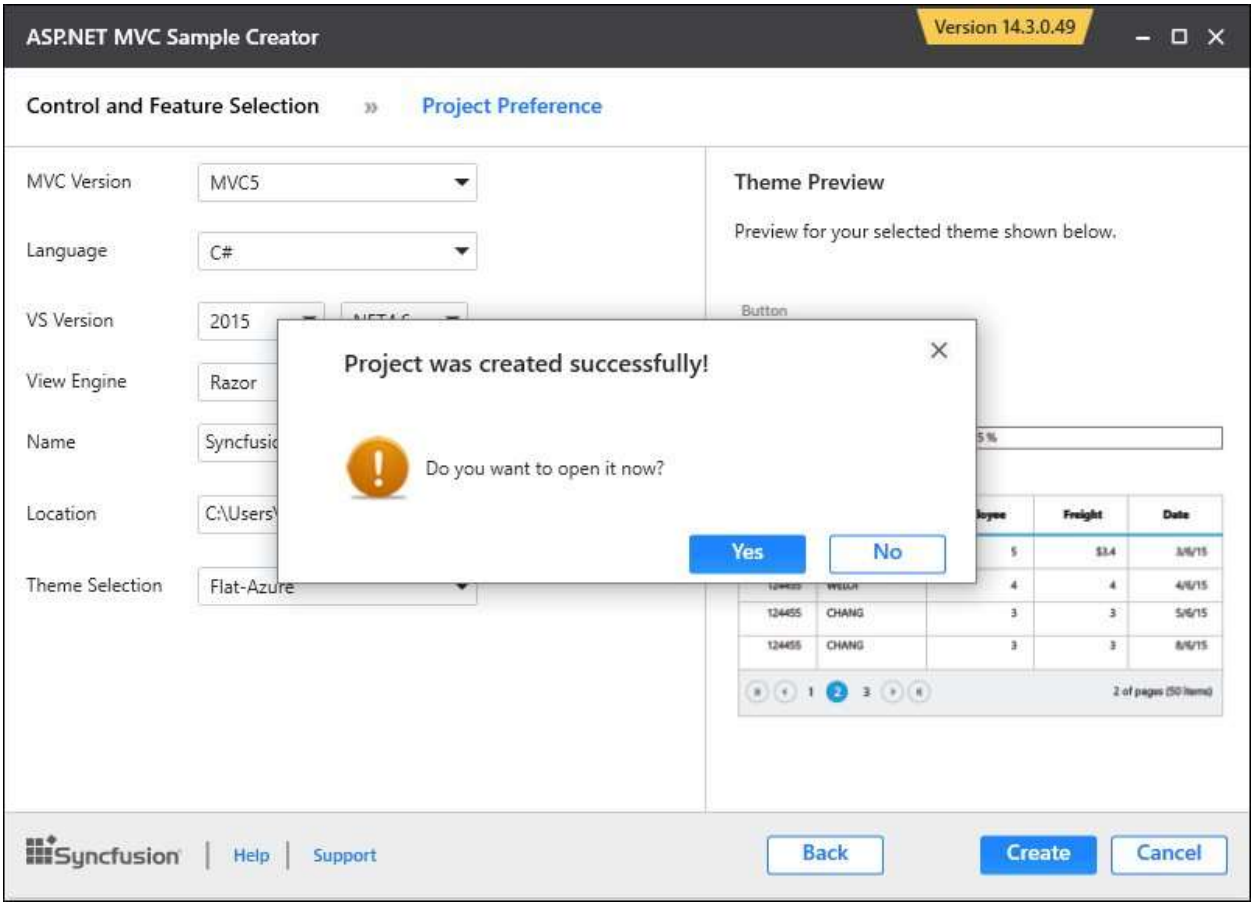


- Configure the Web.Config file by adding the Syncfusion reference assemblies.

```
<compilation debug="true" targetFramework="4.5">
  <assemblies>
    <add assembly="Syncfusion.EJ.Export, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Compression.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"
    <add assembly="Syncfusion.EJ, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.EJ.Mvc, Version=14.3500.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Linq.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.XlsIO.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.EJ.Pivot, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
    <add assembly="Syncfusion.Olap.Base, Version=14.3450.0.49, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
  </assemblies>
</compilation>
```

- Once the project is created you can open the project by clicking the Yes button. Refer to the following screenshot for more information.

Copyright © 2001 - 2020 Syncfusion Inc.



Core Concepts

Core Concepts

Basics

API Configuration

Set the values to Syncfusion Widget properties during or after the creation by using fluent helper methods.

```
<div class="jumbotron">
    <h3>Syncfusion MVC Application</h3>
    @(Html.EJ().DatePicker("MyFirstDatepicker")
                .MinDate(new DateTime(2016,5,1))
                .MaxDate(new DateTime(2016,12,31))
                .
    )
</div>
```

TagName
ToHtmlString
TooltipFormat
ToString
ValidationMessage
ValidationRules
Value
WatermarkText
Width

DatePickerPropertiesBuilder DatePickerPropertiesBuilder.Value(string value) (+ 1 overload(s))
Sets current value of DatePicker in string return type.

In the above example, minimum and maximum ranges are defined for DatePicker component.

Wiring Client Side Events

Client side events can be wired in the same way as properties. But here all the client side events are grouped into **"ClientSideEvents"** Action method.

1. Specify the client side event name in string format.

For example:

ASPX-CS

```
@(Html.EJ().DatePicker("MyFirstDatepicker")
.MinDate(new DateTime(2016,5,1))
.MaxDate(new DateTime(2016,12,31))
.Value(DateTime.Now.AddDays(1) )
.ClientSideEvents( events =>
events.Change("datepicker_change") // To handle datepicker change event
.Select("datepicker_select") // To handle datepicker select event
)
)
```

2. Declare the client side events in Script sections.

ASPX-CS

```
@section scripts{
<script>
function datepicker_change(args) {
// event handler
//args contains entire model of DatePicker to get the value of all
properties
}
function datepicker_select(args) {
// event handler
//args contains entire model of DatePicker to get the value of all
properties
}
</script>
}
```

Invoking methods

Syncfusion JavaScript widget client side methods can be accessed via their client side object. You can access the client side object via jQuery.data() utility method.

1. Create server side component in CSHTML.

ASPX-CS

```
@(Html.EJ().DatePicker("MyFirstDatepicker")
.MinDate(new DateTime(2016,5,1))
.MaxDate(new DateTime(2016,12,31))
.Value(DateTime.Now.AddDays(1) )
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

2. Access the client side object using jQuery.data() as shown in the following:

ASPX-CS

```
<script>
$(document).ready(function () {
var datepickerObject = $("#MyFirstDatepicker").data("ejDatePicker");
});
</script>
```

3. Access the method from client side object like properties access, as shown in the following.

ASPX-CS

```
<script>
$(document).ready(function () {
var datepickerObject = $("#MyFirstDatepicker").data("ejDatePicker");
datepickerObject.disable(); // to disable the datepicker object
var selectedDate = datepickerObject.getValue(); // to access the selected
value
});
</script>
```

Localization

Localization is the process of customizing an application for given language and region.

Configure the Syncfusion Components to a particular language by the following steps.

1. Specify the target culture in **web.config** file under root.

XML

```
<system.web>
<globalization uiCulture="fr-FR" culture="fr-FR"
enableClientBasedCulture="true"/>
</system.web>
```

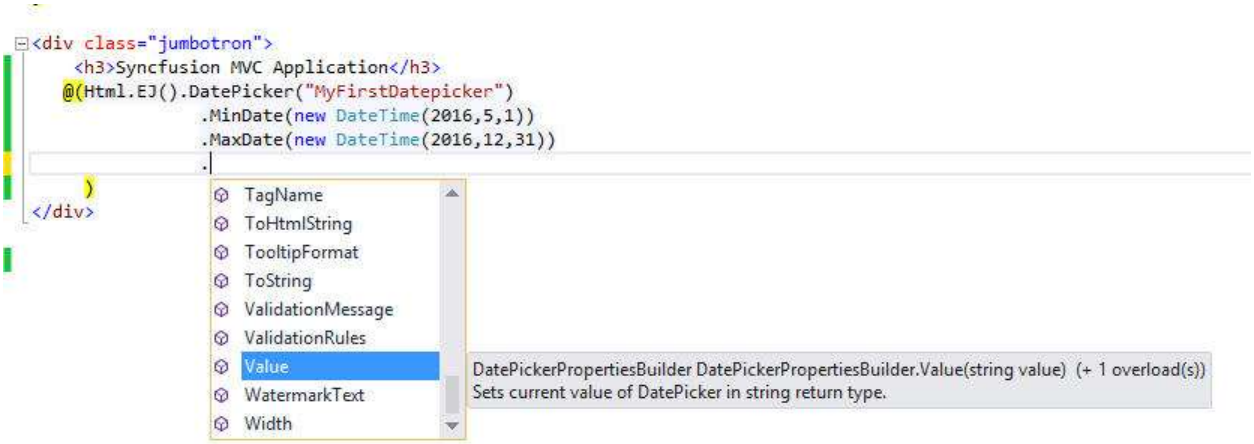
2. Load the globalize culture file from **i18n** folder dynamically by using the following codes.

ASPX-CS

```
@Scripts.Render("~/Scripts/ej/i18n/ej.culture." +
System.Globalization.CultureInfo.CurrentCulture.Name.ToString() + ".min.js")
```

3. Download and refer to the Syncfusion UI component localized text from ej-global repository.

Copyright © 2001 - 2020 Syncfusion Inc.



4. Load the Localized text file from **localetexts** folder like the previous step.

ASPX-CS

```
@Scripts.Render("~/Scripts/ej/localetexts/ej.localetexts." +
System.Globalization.CultureInfo.CurrentCulture.Name.ToString() + ".js")
```

5. Set the culture to Syncfusion UI components using the **Locale** helper method as shown in the following:

ASPX-CS

```
@(Html.EJ().DatePicker("MyFirstDatepicker")
.MinDate(new DateTime(2016, 5, 1))
.MaxDate(new DateTime(2016, 12, 31))
.Locale(System.Threading.Thread.CurrentThread.CurrentCulture.Name) //
Specify the UI culture
.ClientSideEvents(events =>
events.Change("datepicker_change") // To handle datepicker change event
.Select("datepicker_select") // To handle datepicker select event
)
)
```

6. Compile and execute the application. You can be able to see the following output in the browser.

Globalization

As per MSDN, “Globalization involves designing and developing a world-ready app that supports localized interfaces and regional data for users in the multiple cultures”.

The **ejGlobalize** plugin provides options parse numeric, currency, and date values by using culture file. The **assets/scripts/i18n** contains more than 350 cultures currently. You can load the particular culture into application by using the **preferredCulture** method.

Copyright © 2001 - 2020 Syncfusion Inc.

To Internationalize any of our Syncfusion components/page into particular culture, please follow the below steps:

1. Add the reference ej.globalize.min.js script into your application.

Note: The **ej.globalize.min.js** library already avails in within **ej.web.all.min.js** file, therefore it is not necessary to externally refer it in your application if you have referred **ej.web.all.min.js** already. For version lower than 13.4.0.53 refer to the **jQuery.globalize.min.js** along with the **ej.web.all.min.js**.

2. Add the required culture file into the page which is located in **assets/scripts/i18n**.

ASPX-CS

```
@Scripts.Render("~/Scripts/ej/common/ej.globalize.js")
@Scripts.Render("~/Scripts/ej/i18n/ej.culture.fr-FR.min.js")
```

3. Set the culture by using the **preferredCulture()** method. Here, French (fr-FR) culture is used as current culture.

ASPX-CS

```
<script>
if (typeof (ej.globalize) != 'undefined') {
ej.globalize.preferredCulture("fr-FR");//set Culture fr-FR
}
</script>
```

4. Use **format()** method to format the object like numbers, date, price, etc.

ASPX-CS

```
<script>
if (typeof (ej.globalize) != 'undefined') {
console.log("en-US Culture : ");
ej.globalize.preferredCulture("en-US");  //set Culture. Default culture is
en-US
console.log(ej.globalize.format(53422, 'C', "en-US")); // Formatting Price
console.log(ej.globalize.format(new Date(2016,07,15), 'D', "en-US"));
//Formatting date
console.log(ej.globalize.format(123123.576, 'n2', "en-US")); // Formatting
number
console.log("fr-FR Culture : ");
ej.globalize.preferredCulture("fr-FR");//set Culture fr-FR
console.log(ej.globalize.format(53422, 'C', "fr-FR"));// Formatting Price
console.log(ej.globalize.format(new Date(2016, 07, 15), 'D', "fr-FR"));
//Formatting date
console.log(ej.globalize.format(123123.576, 'n2', "fr-FR")); // Formatting
number
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Dynamic Globalization

The steps to globalize the MVC application are as follows:

1. Specify the target culture in **web.config** file under root.

XML

```
<system.web>
<globalization uiCulture="fr-FR" culture="fr-FR"
enableClientBasedCulture="true"/>
</system.web>
```

2. Load the culture file based on current culture that is specified in **web.config** as shown in the following.

ASPX-CS

```
@Scripts.Render("~/Scripts/ej/i18n/ej.culture." +
System.Globalization.CultureInfo.CurrentCulture.Name.ToString() + ".min.js")
```

3. Access the current culture using System.Globalization.CultureInfo and set the culture to ej.globalize widget.

ASPX-CS

```
<script>
if (typeof (ej.globalize) != 'undefined') {
ej.globalize.preferredCulture('@System.Threading.Thread.CurrentThread.Curren
tCulture.Name');//access the current culture from web.config and set it
console.log(ej.globalize.format(53422, 'C'));// Formatting Price
console.log(ej.globalize.format(new Date(2016, 07, 15), 'D')); //Formatting
date
console.log(ej.globalize.format(123123.576, 'n2')); // Formatting number
}
</script>
```

Please refer the below DatePicker sample which uses globalization,

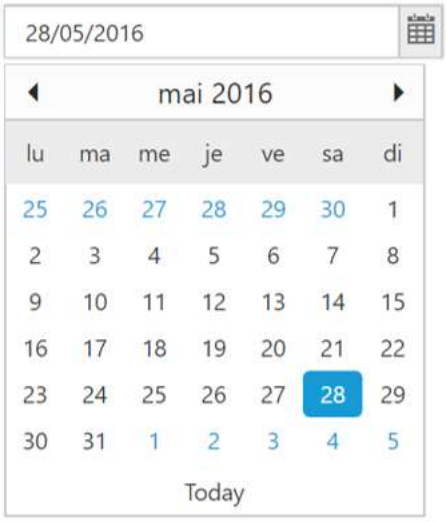
Follow the above steps 1 and 2 and then add the control in the view page

HTML

```
@Html.EJ().DatePicker("DatePick").Value("28/05/2016").Width("100%")
```

The following screenshot exhibits the output of the above code,

Copyright © 2001 - 2020 Syncfusion Inc.



Unobtrusive

Many uncertainties and difficulties are involved in a usual **JavaScript programming** environment, like some of the browsers may ignore the JavaScript codes under the scripts section completely or partially due to its complexity, and so on. To overcome all such inconveniences, the Unobtrusive JavaScript support has been introduced to make it easier for the users to create all our Syncfusion components with basic level HTML tag-like structure.

One of the main goal of the unobtrusive support is to achieve the clear separation of both the HTML content and behavior, so as to enhance the page loading time and to make the code updation easier. The **Essential JavaScript** have separate integration library to achieve the **Unobtrusive JS** support. To make use of Unobtrusive support with our Essential JavaScript components, it is necessary to refer to the **ej.unobtrusive.min.js** file in your application.

The **ej.unobtrusive.min.js** file can be accessed from the following location, that can then be copied and referred in your application.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\common
For example, If you have installed the Essential Studio package within C:\Program Files (x86), navigate to the following location. C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\common

When you enable the **unobtrusive** in your application, refer to the **ej.unobtrusive.min.js** file in the application.

You have to set the value of **UnobtrusiveJavaScriptEnabled** to **true** in Root directory web.config file as shown in the following code example.

HTML

```
<appSettings>
```

```
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

After setting the value as true, refer to the unobtrusive script file in the _Layout page as shown in the following code example.

HTML

```
<script src="~/Scripts/ej/ej.unobtrusive.min.js"></script>
```

Note: The ej.unobtrusive.min.js script should be referred next to the ej.web.all.min.js file reference, since it is dependent on it.

By default, it is necessary to add Script Manager code in the _Layout.cshtml file present within the ~/Views/Shared folder of your application.

HTML

```
<body>
@Html.EJ().ScriptManager()
@RenderSection("scripts", required: false)
</body>
```

Note: The main reason for referring the Script manager in _Layout file is that, it can be referred as common by all the View files present within your application.

If unobtrusive is set to true in the application, the script manager can be excluded, as the control is initialized by using HTML5 attributes.

A simple example by using the DropDownList control is as follows:

Render the DropDownList control as follows in your view page and enable the unobtrusive mode in your application.

HTML

```
@Html.EJ().DropDownList("selectCar").TargetID("carsList").Width("100%").Wate
rmarkText("Select a Car")
<div id="carsList">
<ul>
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
</ul>
</div>
```

When the above code is executed on the browser, the DropDownList control will be rendered with the following equivalent HTML DOM attributes created for it.

Copyright © 2001 - 2020 Syncfusion Inc.



Enabling Client Side Validation

Refer to the required jQuery validation scripts to perform Client-side Validation before postback, when the form is submitted. This can be enabled or disabled by ClientValidationEnabled key in app settings of web.config file.

In this section, how to perform Client-Side Validation for DropDownList control with DataAnnotations attribute is explained.

Define the view model for DropDownList control as follows.

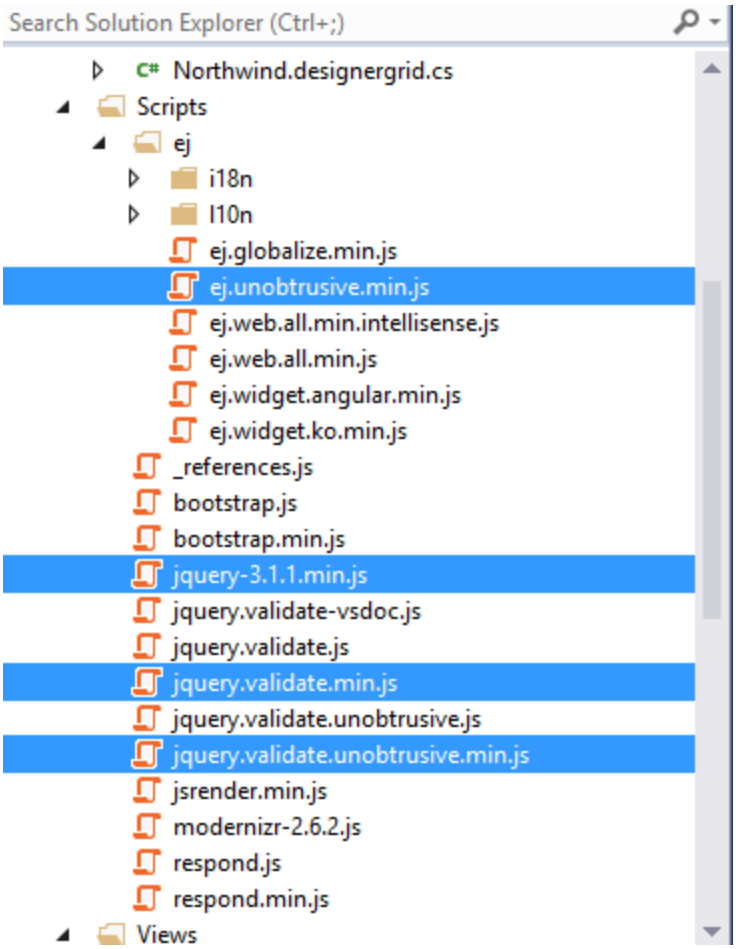
HTML

```
using System.ComponentModel.DataAnnotations;
namespace Dropdown
{
public class DropValue
{
[Required(ErrorMessage = "DropDownList value is Required")]
public List<DropDownValue> DropData { get; set; }
}
public class DropDownValue
{
public string Text { get; set; }
public string Value { get; set; }
}
}
```

Four sets of files are needed to implement the Unobtrusive.

- The jQuery library
- The jQuery Validate plugin
- The Unobtrusive extensions to Validate
- Ej unobtrusive library

Copyright © 2001 - 2020 Syncfusion Inc.



Refer the above mentioned necessary scripts files in your layout.cshtml page.

HTML

```
<script src="~/Scripts/jquery-3.1.1.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
<script src="~/Scripts/ej/ej.web.all.min.js"></script>
<script src="~/Scripts/ej/ej.unobtrusive.min.js"></script>
```

Note: While creating a sample through Syncfusion ProjectTemplate or SampleCreator, the "ej.unobtrusive.min.js" file will be found in the "Scripts/ej" folder, whereas ,installing NuGet will ship the "ej.unobtrusive.min.js" file inside the "Scripts/ej/common" folder. Modify the script reference in the layout page based on this folder structure.

After that you need to create the controller's action methods. These render views on the UI and bind a model with the view. So let us create a controller as follows:

HTML

```
using Syncfusion.JavaScript.Models;
namespace Dropdown
{
public partial class DropdownlistController: Controller
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult DropdownlistFeatures()
{
BindingData();
return View();
}
public void BindingData()
{
List<DropDownValue> data = new List<DropDownValue>() { };
data.Add(new DropDownValue() { Value = "item1", Text = "List Item 1" });
data.Add(new DropDownValue() { Value = "item2", Text = "List Item 2" });
data.Add(new DropDownValue() { Value = "item3", Text = "List Item 3" });
data.Add(new DropDownValue() { Value = "item4", Text = "List Item 4" });
data.Add(new DropDownValue() { Value = "item5", Text = "List Item 5" });
DropDownListProperties obj = new DropDownListProperties();
obj.DataSource = data;
DropDownListFields fields = new DropDownListFields();
fields.Text = "Text";
fields.Value = "Value";
obj.DropDownListFields = fields; ViewData["properties"] = obj;
}
[HttpPost]
public ActionResult DropdownlistFeatures(DropDownListModel model)
{
BindingData();
return View(model);
}
}
}
```

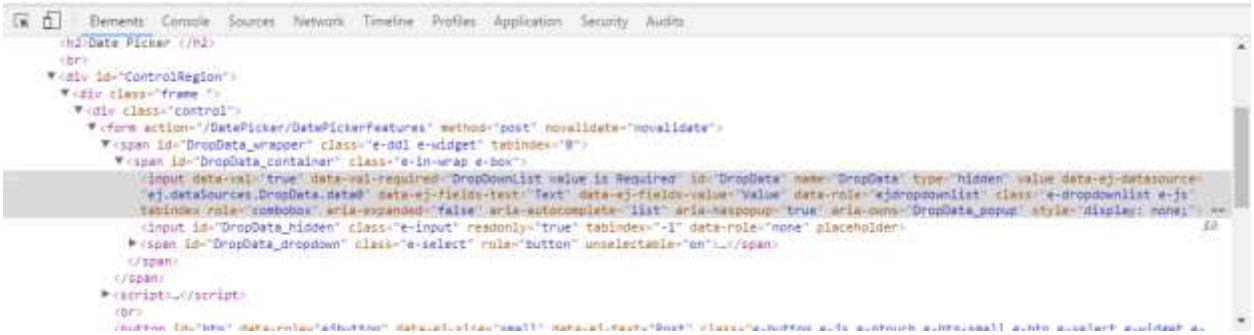
The view is created as in the following code snippet:

HTML

```
@using (Html.BeginForm())
{
@Html.ValidationSummary(true)
@Html.EJ().DropDownListFor(Model => Model.DropData,
(Syncfusion.JavaScript.Models.DropDownListProperties)ViewData["properties"])
<br />
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Post").Type(ButtonType
.Submit)
}
```

When the above code is executed on the browser, the DropDownList control will be rendered with the following equivalent HTML DOM attributes created for it.

Copyright © 2001 - 2020 Syncfusion Inc.



Now, some interesting new attributes are there to look at:

- data-val specifies that the DropDownList needs validation.
- data-val-required is the error message to be displayed, if a value is not provided.

This is part of the magic of Unobtrusive: it uses HTML5-compatible "data-" attributes to store all the information it needs to perform validation. This is why you don't need to use any other code besides attributes to enable client-side validation with this library.

Validation

jQuery Validation

To perform the jQuery validation for EJ form controls, refer to the following steps.

Step 1: First you need to include the jquery.validate (1.15.0) script in your HTML page.

Step 2: After adding this script file, you can validate the EJ form controls in same way as you have performed the jQuery validation for HTML form elements. But, here you have to do some little bit configuration along with that. It is discussed in the following.

If validation gets fail, you have to place the error message in proper position using the “errorPlacement” API that is available in the default settings of jQuery validation. Also, built-in “error” class will be added to the corresponding form element. Here, you have to specify a custom class with your own style using the “errorClass” API. These are the common settings for all of our EJ form controls.

And some of our EJ form controls contains the form values in hidden element for its functionality purpose and by default, these hidden elements are not validated. If you want to include the hidden elements in jQuery validation, set “[]” in “ignore” API of “$.validator.setDefaults”. Refer to the following list to find out the EJ form controls that contains the values in hidden element.

- Checkbox
- MaskEdit
- NumericTextbox
- CurrencyTextbox
- PercentageTextbox
- RTE
- Dropdownlist

Refer to the following code block to configure the above mentioned jQuery validation settings.

JAVASCRIPT

```
$.validator.setDefaults({
```

```
//to validate all fields(hidden also)
ignore: [],
//if we don't set custom class, default "error" class will be added
errorClass: 'e-validation-error',
//it specifies the error message display position
errorPlacement: function (error, element) {
$(error).insertAfter(element.closest(".e-widget"));
}
});
```

Up to 13.4.0.58 version, the above specified jQuery validation settings were configured in source level of EJ form controls. Since 14.1.0.41 release onwards, the above specified settings has been removed from the source and it needs to be specified in sample level to improve the customization of this validation.

The EJMVC DatePicker provides cool way to validate the value by custom rules and messages before it get processed. Enhanced APIs and built-in validation support in EJWEB DatePicker provides the quick validation in client side itself, that provides easy customization and better user experience than validating the values in server side.

HTML

```
@using (Html.BeginForm())
{
//Sets the field to be required with validation message
@Html.EJ().DatePicker("datePicker").ValidationRules(rule =>
rule.AddRule("required", true)).ValidationMessage(msg =>
msg.AddMessage("required", "Required Date value"));
<input type="submit" value="click here!" />
}
```

Accessibility

The term Accessibility in general, refers to the process of designing a product that is easily accessible to all kinds of people irrespective of their disabilities. The developer should keep in mind the concept of Accessibility as their prime intention while creating any of the web applications, so that the application and the web content will be easily accessible to the people with disabilities. Such applications created with Accessibility support is generally termed as Accessible Rich Internet Applications (ARIA).

Sometimes, the content and the functionalities used in the websites cannot be conveyed to the users who mainly relies on screen readers and one who cannot use the mouse. To overcome such circumstances and to address these accessibility challenges, Web Accessibility Initiative (WAI-ARIA) has defined a new way/specification to achieve those functionalities with the use of some assistive technologies. The assistive technologies will make use of the widget's attribute information (roles, states, and properties) and then convey the appropriate information to the persons with disabilities.

The WAI-ARIA specification suite defines a set of roles, states, and properties for the user-accessible DOM elements that makes the accessibility and interoperability of the web content and applications easier than before.

Creating accessible widgets using WAI-ARIA

This section describes about the approaches followed to make the widgets accessible using the Roles, States, and Properties defined by WAI-ARIA specifications. The WAI-ARIA specifications recommends several attributes to be used along with the DOM element structure to make it accessible.

Copyright © 2001 - 2020 Syncfusion Inc.

The common widgets that can be made accessible by using the WAI-ARIA are listed here. The various ARIA attributes that are used to make the widgets accessible are listed here.

The basic steps for converting a common widget into accessible are as follows:

1. Choose the appropriate role for your widget that defines the general component type. For example, if you need to create a button, set the role to the button.

HTML

```
<button id="button1" role="button">Submit</button>
```

2. Once the role is selected, now you can set its state/properties avails for it from the link given here.

HTML

```
<button id="button1" role="button" aria-disabled="false" aria-
describedby="Submit the Form" >Submit</button>
```

i. The above code will make the button widget accessible, as it was defined with the aria-disabled state as false. It states that if none of that button actions are currently available, then the button is said to be in an aria-disabled state.

ii. The property aria-describedby has been assigned with some related button text that denotes the button description to be notified to the user.

Note: Note: If your button is a togglebutton, it can be defined with an additional aria-pressed state, that returns the value true if the button undergoes toggling, else the state remains as false.

An example with button control that has been made accessible can be referred from this link.

Usage of Accessibility in Syncfusion Widgets

All our Syncfusion Widgets are developed based on the standards mentioned in the ARIA specifications, therefore it supports the full-fledged feature of Web Accessibility Initiative (WAI). Since our Essential JavaScript has provided the built-in support for accessibility in all its UI widgets, therefore it allows the user to create the widgets that are easily used by the assistive technologies. The two major concepts that has been mainly focused for providing such accessibility support are as follows.

- Keyboard navigation
- Defining ARIA attributes to the required DOM elements

As of now, the keyboard navigation functionality is provided as a default feature for all the Syncfusion widgets which allows all the keyboard enabled widgets to interact with the user.

Keyboard Navigation

Any Syncfusion widgets used within an application are easily accessible to all, especially for the visually disabled peoples and those who cannot use mouse for navigating through the application. To make it more accessible, the built-in keyboard navigation support has been provided for almost all the components available within the Essential Studio package. Simply by pressing the TAB key will move the focus from one element to another within the application.

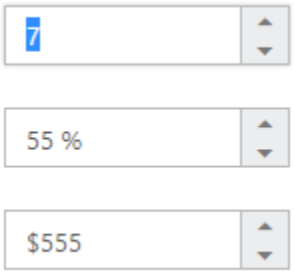
With this support enabled for the widgets, the various elements present within that particular widget can be easily focused by pressing the appropriate keyboard shortcuts. The currently focused element

Copyright © 2001 - 2020 Syncfusion Inc.

within the widget is clearly indicated with its unique visual style that denotes that it is now ready to accept the user inputs.

Such visual indications through keyboard navigation are very useful for the accessibility support, as it needs to be properly conveyed to the assistive technologies that the focus has been changed.

Each and every widget available within the Syncfusion Essential JavaScript suite has its unique behavior and responds appropriately to the keyboard actions. As an example, we will look onto the following 3 editors placed on the same page and how to navigate between them through the keyboard.



Initially, the focus is on the first editor (numeric textbox) that is indicated with the text selection and the border effect is provided for the widget's outline through the CSS.

Having the current focus on the numeric text box, now pressing the up arrow will increase the numeric value and down arrow will decrease the value. Pressing tab key will advance the focus to the next widget present in the page - here the percentage textbox that is placed as second will get focused on pressing the tab key.

To enable the default tab key navigation, all the Syncfusion widgets are set with appropriate tabIndex property with the provided value as 0 (zero). The element with keyboard focus is essential because it communicates essential information about the widget to the assistive technologies like screen readers.

Pressing Tab key will traverse the focus in a forward direction and if the shift + Tab combination of keys are pressed, the focus will be moved to the previous elements (focus traverse through backward direction).

Defining ARIA attributes to the required DOM elements

As discussed in the previous topics - To make a component more accessible by all kind of users, it needs to be defined with some essential information along with its initial element creation. Here, the essential information to be defined for the components are its suitable role, state, and its properties. Usually the assistive technologies will make use of these information retrieved from the specific attributes and use it to convey the appropriate information to the users.

Roles, States and Properties

As the widget role defines the component type, therefore all the Syncfusion widgets are provided with corresponding roles. The roles can be anything like button, toolbar, checkbox, radiobutton, and so on.

All the Syncfusion widgets are assigned with built-in States and properties based on its role. Usually the states and properties are defined for a widget using the ARIA attributes that will convey the appropriate current state and action of the control to the assistive technologies.

Copyright © 2001 - 2020 Syncfusion Inc.

For example, our Syncfusion button widget, when created through JavaScript code will be rendered in the web browser with HTML DOM elements (along with the assigned built-in ARIA attributes) as shown in the following.

HTML

```
@Html.EJ().Button("button11").Text("Login").Size(ButtonSize.Normal).ShowRoun
dedCorner(true)
```

When the above code is executed on the browser, the Syncfusion button widget will be rendered with the following equivalent HTML DOM attributes created for it.

HTML

```
<button id="button11" class="e-button e-js e-btn-normal e-btn e-select e-
widget e-corner-all" tabindex="" type="submit" role="button" aria-
describedby="login" aria-disabled="false"></button>
```

Here, the aria-describedby is the ARIA property assigned to the Syncfusion button widget as built-in, so that the button description is notified to the user through assistive technologies.

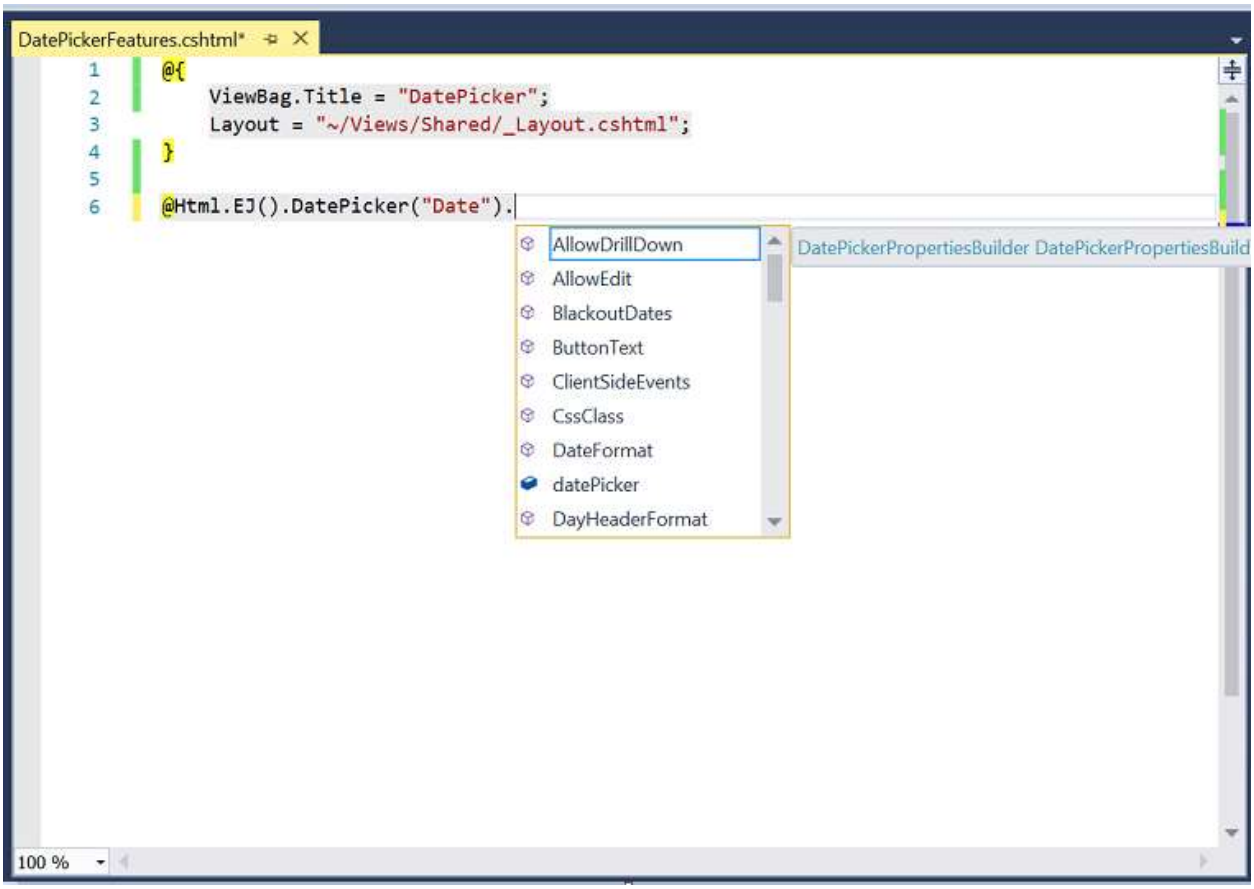
Likewise, the aria-disabled is one of the ARIA state assigned to the button as built-in, that states that if none of that button actions are currently available, the button is said to be in an aria-disabled state.

IntelliSense

The IntelliSense support is provided like the default ASP.NET MVC controls. It has the object list that holds the properties, public methods and event arguments of respective controls. On pressing the (.) Dot operator it displays those properties, public methods and event arguments of the corresponding control in the dropdown.

Refer to the following screenshot where the DatePicker control is used in the sample.

Copyright © 2001 - 2020 Syncfusion Inc.



Exporting Multiple Syncfusion Controls

We can export multiple Syncfusion controls like Grid, Chart, TreeGrid, etc., in a single document.

This can be achieved by calling ej.exportAll method. exportAll method accepts two necessary parameters; one for routing URL (either to export Excel or PDF) and other for passing multiple controls ID's as an array.

In the server side we can handle exporting multiple control either Excel or PDF exporting by calling ExportAll method

The following code example describes the above behavior.

RAZOR

```
@Html.EJ().Button("ExcelExport").Size(ButtonSize.Medium).ShowRoundedCorner(t
rue).Text("Excel Export").ClientSideEvents(eve=>eve.Click("ExcelExport"))
@Html.EJ().Button("PdfExport").Size(ButtonSize.Medium).ShowRoundedCorner(tru
e).Text("Pdf Export").ClientSideEvents(eve => eve.Click("PdfExport"))
@(Html.EJ().Grid<EmployeeView>("MasterGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(100).Add();
col.Field("FirstName").HeaderText("First Name").Width(100).Add();
col.Field("LastName").HeaderText("Last Name").Width(100).Add();
col.Field("Title").HeaderText("Title").Width(90).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("BirthDate").HeaderText("Birth
Date").TextAlign(TextAlign.Right).Width(100).Format("{0:MM/dd/yyyy}").Add();
col.Field("Country").Width(120).HeaderText("Country").Add();
})
)
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
.TreeColumnIndex(1)
.Columns(co =>
{
co.Field("TaskId").HeaderText("Task Id").Width(80).Add();
co.Field("TaskName").HeaderText("Task Name").Add();
co.Field("StartDate").HeaderText("Start Date").Add();
co.Field("EndDate").HeaderText("End Date").Add();
co.Field("Duration").HeaderText("Duration").Add();
})
.EditSettings(es => es.AllowDeleting(false))
.SizeSettings(ss => ss.Width("100%").Height("350px"))
.Datasource(ViewBag.data)
)
@(Html.EJ().Chart("ChartContainer").ChartArea(cr => cr.Border(ab =>
ab.Color("transparent")))
.Border(border => border.Opacity(1))
.Series(sr =>
{
sr.Points(pt =>
{
pt.X("John").Y(10).Add();
pt.X("Jake").Y(12).Add();
pt.X("Peter").Y(18).Add();
pt.X("James").Y(11).Add();
pt.X("Mary").Y(9.7).Add();
}).Name("Person").Type(SeriesType.Column).XName("XValue").YName("YValue1").E
nableAnimation(true).Tooltip(d => d.Visible(true)).Add();
})
.EnableCanvasRendering(true))
```

C#

```
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var MasterData = new NorthwindDataContext().EmployeeViews.Take(5).ToList();
ViewBag.dataSource = MasterData;
var data = this.GetDefaultData();
ViewBag.data = data;
return View();
}
public void ExcelExport (FormCollection controlModel)
{
Dictionary<string, string> val =controlModel.AllKeys.ToDictionary(k => k,
v=>controlModel[v]);
ExcelExport exp = new ExcelExport();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var EmployeeData = new
NorthwindDataContext().EmployeeViews.Take(5).ToList();   //DataSource for
Grid.
var treeData = this.GetDefaultData();
//DataSource for TreeGrid.
List<ExportChartData> chartData = new List<ExportChartData>();
chartData.Add(new ExportChartData("John", 10));
chartData.Add(new ExportChartData("Jake", 12));
chartData.Add(new ExportChartData("Peter", 18));
chartData.Add(new ExportChartData("James", 11));
chartData.Add(new ExportChartData("Mary", 9.7));
var chartDataSource = chartData;
//DataSource for Chart
Dictionary<string, object> items = new Dictionary<string, object>();
items.Add("ejGrid",EmployeeData);
items.Add("ejTreeGrid", treeData);
items.Add("ejChart", chartDataSource);
exp.ExportAll(val, items, "Export.xlsx", ExcelVersion.Excel2010, "flat-
saffron");
}
public void PdfExport (FormCollection controlModel)
{
Dictionary<string, string> val = controlModel.AllKeys.ToDictionary(k => k, v
=> controlModel[v]);
PdfExport exp = new PdfExport();
var EmployeeData = new
NorthwindDataContext().EmployeeViews.Take(5).ToList();    //DataSource for
TreeGrid.
var treeData = this.GetDefaultData();
//DataSource for TreeGrid.
List<ExportChartData> chartData = new List<ExportChartData>();
chartData.Add(new ExportChartData("John", 10));
chartData.Add(new ExportChartData("Jake", 12));
chartData.Add(new ExportChartData("Peter", 18));
chartData.Add(new ExportChartData("James", 11));
chartData.Add(new ExportChartData("Mary", 9.7));
var chartDataSource = chartData;
//DataSource for Chart
Dictionary<string, object> items = new Dictionary<string, object>();
items.Add("ejGrid", EmployeeData);
items.Add("ejTreeGrid", treeData);
items.Add("ejChart", chartDataSource);
exp.ExportAll(val, items, "Export.pdf", "flat-saffron");
}
}
```

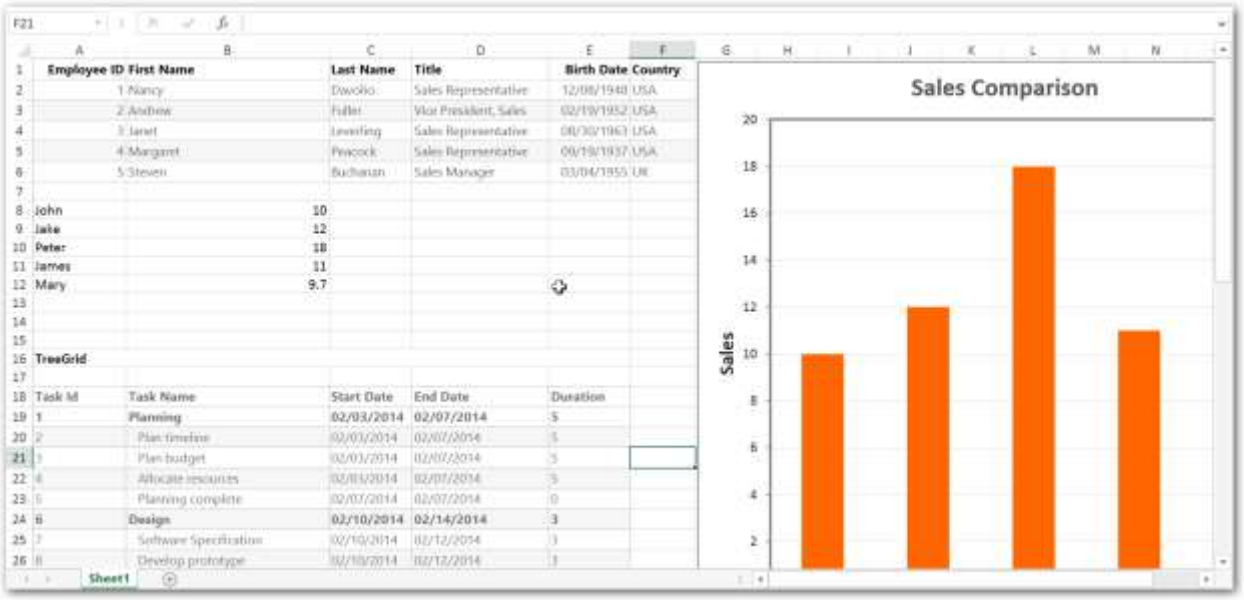
JS

```
<script>
function ExcelExport(args) {
ej.exportAll("/Grid/ExcelExport", ['MasterGrid', 'ChartContainer',
'TreeGridContainer'])
}
function PdfExport(args) {
ej.exportAll("/Grid/PdfExport", ['MasterGrid','TreeGridContainer' ]);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

The following output is displayed as a result of the above code example.



Strongly-Typed HTML Helper

The Syncfusion editor controls supports the strongly typed HTML helpers represented by lambda expressions that have the model or template passed into the view. The Extension method is used to get a value from the model.

The Strongly-Typed HTML helper (i.e., NumericTextBox) takes lambda as a parameter that tells the helper, which element of the model to be used in the typed view.

The Strongly typed views are used for rendering specific types of model objects, instead of using the general ViewData structure.

The following list of controls supports the Strongly-Typed HTML Helper:

- Autocomplete
- Checkbox and Radio Button
- DatePicker
- DateTimePicker
- DropDownList
- Mask Edit
- Numeric, Currency, and Percentage Textbox
- RichTextEditor
- TimePicker

The following steps explain how to use the strongly typed helpers to create a NumericTextBox.

The NumericTextBox control supports strongly typed HTML helpers that uses lambda expression to refer to the models or view models passed to a view template. These helpers allows you to define the value of the NumericTextBoxFor from the model.

Copyright © 2001 - 2020 Syncfusion Inc.

Add a class name “EditorValue” in the Models folder and replace the code with the following code:

HTML

```
public class EditorValue
{
public int number { set; get; }
public EditorValue(int value)
{
number = value;
}
public EditorValue (){}
}
```

Create an action method that renders the NumericTextBox on the view page, and passes the model to be bound to the view page.

C#

```
using Syncfusion.JavaScript.Models;
using MVCSampleBrowser.Models;
public ActionResult EditorFor()
{
//Editor For
EditorProperties editor = new EditorProperties();
editor.EnableStrictMode = true;
ViewData["editmodel"] = editor;
return View(new EditorValue(66));
}
```

In View, invoke the strongly typed NumericTextBoxFor helper with the lambda expression to set the default value.

RAZOR

```
@model MVCSampleBrowser.Models.EditorValue
@using (Html.BeginForm())
{
@Html.EJ().NumericTextBoxFor(model => model.number,
(Syncfusion.JavaScript.Models.EditorProperties)ViewData["editmodel"])
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("PayBill").Type(ButtonT
ype.Submit)
}
```



The following steps explain how to get the values by using the Scaffolding methods in Post back.

Copyright © 2001 - 2020 Syncfusion Inc.

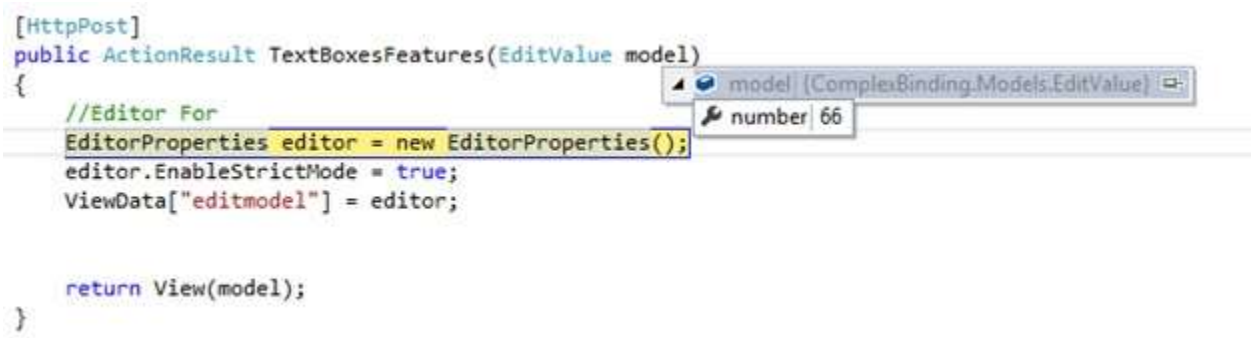
1. Create an action method, FormPost that handles the Post request and processes the data.

In the action method, you can pass the model as the parameter and that model has the NumericTexBox's value.

C#

```
[HttpPost]
public ActionResult EditorFor(EditorValue model)
{
//Editor For
EditorProperties editor = new EditorProperties();
editor.EnableStrictMode = true;
ViewData["editmodel"] = editor;
return View(model);
}
```

On clicking the button, the Post method will be triggered. In that, the selected value will be obtained as follows.



Client Side Validation

With the client-side validation, the input data is checked as soon as they are submitted, so there is no postback to the server and no page refresh.

By default, client-side validation is enabled. But it can be easily enabled or disabled by the writing of the following app setting code snippet in the web.config file.

HTML

```
<appSettings>
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

Note: Enable client side validation for specific view only by adding the Html.EnableClientValidation(true) at the top in view page.

After setting the value as true, refer to the jQuery validation script file in the _Layout page as shown in the following code example.

HTML

```
<script src="~/Scripts/jquery.validate.min.js"></script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script src="~/Scripts/jquery.validate.unobtrusive.js"></script>
<script src="~/Scripts/ej/ej.unobtrusive.min.js"></script>
```

The jQuery validation plug-in takes the advantage of the Data Annotation attributes defined in the model.

Step 1: Add the following namespace to the “EditorValue” model.

HTML

```
using System.ComponentModel.DataAnnotations;
```

The Data Annotations allows to decorate model classes with the metadata. This metadata describes a set of rules that are used to validate a property.

The following Data Annotation attributes are used for the Numeric Textbox.

Required: Indicates that the property is a required field.

Step 2: Next, Update the number property of the “EditorValue” class as “Required Field” by adding the following line.

C#

```
using System.ComponentModel.DataAnnotations;
public class EditorValue
{
[Required(ErrorMessage = "Numeric Textbox is Required")]
public int number { set; get; }
public EditorValue(int value)
{
number = value;
}
}
```

Step 3: Modify the view page as follows:

RAZOR

```
@model ComplexBinding.Models.EditorValue
@using (Html.BeginForm())
{
@Html.ValidationSummary(true)
@Html.EJ().NumericTextBoxFor(model => model.number,
(Syncfusion.JavaScript.Models.EditorProperties)ViewData["editmodel"])
@Html.ValidationMessageFor(model => model.number)
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("PayBill").Type(ButtonT
ype.Submit)
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Server Side Validation

In the server-side validation, the page must be submitted via a postback to validate on the server and if the model data is not valid, the server sends a response back to the client.

The best way to validate a model is by using the Data Annotations that has a set of attributes and classes defined in the System.ComponentModel.DataAnnotations assembly.

Step 1: Add the following namespace to the “EditorValue” model.

HTML

```
using System.ComponentModel.DataAnnotations;
```

The Data Annotations allows to decorate model classes with the metadata. This metadata describes a set of rules that are used to validate a property.

The following Data Annotation attributes are used for the Numeric Textbox.

Required: Indicates that the property is a required field.

Step 2: Next, Update the number property of the “EditorValue” class as “Required Field” by adding the following line

C#

```
using System.ComponentModel.DataAnnotations;
public class EditorValue
{
[Required(ErrorMessage = "Numeric Textbox is Required")]
public int number { set; get; }
public EditorValue(int value)
{
number = value;
}
}
```

Step 3: Modify the view page as follows:

RAZOR

```
@model ComplexBinding.Models.EditorValue
@using (Html.BeginForm())
{
@Html.ValidationSummary(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().NumericTextBoxFor(model => model.number,
(Syncfusion.JavaScript.Models.EditorProperties)ViewData["editmodel"])
@Html.ValidationMessageFor(model => model.number)
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("PayBill").Type(ButtonT
ype.Submit)
}
```

When you press the “PayBill” button on this page then it will post the data to the server and the code written with in EditorFor action will validate the NumericTextBox value by checking the ModelState.IsValid property. If the NumericTextBox value is not selected, the ModelState.IsValid will return false and display error message.



Complex Model Binding

The lambda expressions are quite powerful, allows you to build quite complex edit models and have model binding that put everything back together again. A complex view model type such as:

C#

```
public class Person
{
public string Name { get; set; }
public BillingInfo BillingInfo { get; set; }
public Person(BillingInfo info){
BillingInfo = info;
}
public Person() { }
}
public class BillingInfo
{
public int BillingNo { get; set; }
public BillingInfo(int no)
{
BillingNo = no;
}
public BillingInfo() { }
}
```

Create an action method that renders NumericTextbox on the view page, and passes the model to be bound to the view page.

C#

```
using ComplexBinding.Models;
using Syncfusion.JavaScript.Models;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Complex()
{
//Editor For
EditorProperties editor = new EditorProperties();
editor.EnableStrictMode = true;
ViewData["editmodel"] = editor;
return View(new Person(new BillingInfo(1121)));
}
```

In View, invoke the strongly typed NumericTextBoxFor helper with the lambda expression to set the default value.

RAZOR

```
@model ComplexBinding.Models.Person
@using (Html.BeginForm())
{
@Html.EJ().NumericTextBoxFor(x => x.BillingInfo.BillingNo,
(Syncfusion.JavaScript.Models.EditorProperties)ViewData["editmodel"])
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("PayBill").Type(ButtonT
ype.Submit)
}
```



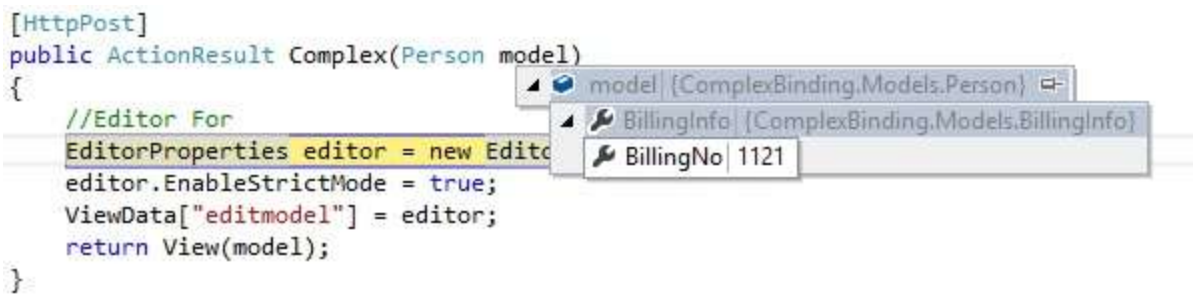
When the form is submitted, it will perform an HTTP Post request to the controller. The action with the HttpPost attribute will handle the request.

C#

```
[HttpPost]
public ActionResult Complex(Person model)
{
//Editor For
EditorProperties editor = new EditorProperties();
editor.EnableStrictMode = true;
ViewData["editmodel"] = editor;
return View(model);
}
```

In the above code, the selected value of NumericTextBox will be obtained as follows:

Copyright © 2001 - 2020 Syncfusion Inc.



Syntax for VB

Our Syncfusion ASP.NET MVC components can render the control either by using **C#** or **VB**. As only the control declaration syntax varies across languages and other functionalities are usually performed in the client-side.

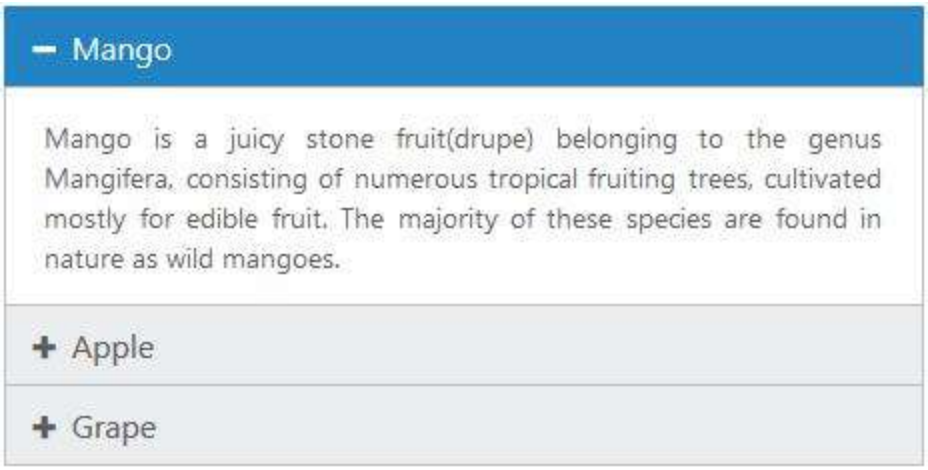
The following example code illustrates the VB syntax to be used within the server-side wrapper declaration – **to create an Accordion control** using the lambda expression.

VB.NET

```
@Code
Html.EJ().Accordion("basicAccordion").Items(Sub(data)
data.Add().Text("Mango").ContentTemplate(Sub()@<div> Mango is a juicy stone
fruit(drupe) belonging to the genus Mangifera, consisting of numerous
tropical fruiting trees, cultivated mostly for edible fruit. The majority of
these species are found in nature as wild mangoes.</div>End Sub)
data.Add().Text("Apple").ContentTemplate(Sub()@<div>Apple is a sweet,
pomaceous fruit originated in Central Asia, where its wild ancestor, Malus
sieversii, is still found today. Apples have been grown for thousands of
years in Asia and Europe, and were brought to North America by European
colonists.</div> End Sub)
data.Add().Text("Grapes").ContentTemplate(Sub()@<div>Grapes are a non-
climacteric type of fruit, generally occurring in clusters. It is a fruiting
berry of the deciduous woody vines of the botanical genus Vitis. The
cultivation of the domesticated grape began 6,000–8,000 years ago in the
Near East.</div> End Sub)
End Sub).Render()
End Code
```

The output for the above code will be like the following:

Copyright © 2001 - 2020 Syncfusion Inc.



Accordion

Accordion

The Accordion control is an interface where lists of items can be collapsed or expanded. It has several collapsible panels where only one/more can be expanded at a time that is useful for dashboards where space is limited. Each Accordion control has a template for its header and its content.

Key Features

- Collapsible Header: All headers are collapsible.
- AJAX Load: Load AJAX content in the Accordion content panel.
- Icon Customization: Expand and collapse icons can be customized.
- Custom Event for Expand Header: By default, Accordion panels can be expanded and collapsed on a single click. However, this action can be customized to occur on mouse-over or mouse-up through custom events.
- Multiple Open: Multiple items can be expanded.
- Auto Size: The content panel height can be predefined with values such as content, auto, and fill.
- Theme: Essential ASP.NET MVC controls feature 13 built-in themes (six flat themes and six with gradient effects and a bootstrap theme), and also supports custom skin options to set user-defined themes.
- RTL: Accordion headers and content text can display RTL languages.
- Keyboard Navigation: You can expand and collapse panels using the keyboard.

Getting Started

This section explains briefly about how to create an Accordion in your application with ASP.NET MVC.

Copyright © 2001 - 2020 Syncfusion Inc.



Create your first Accordion in ASP.NET

The ASP.NET MVC Accordion control allows you to provide multiple panes and display them one/more at a time. In this section, you can learn how the Accordion control is configured and how to use it in your application.

Configure Accordion

This section explains you the details on how to configure the Accordion control in your application and how to customize it with various properties such as multiple open, rounded corner and icons for the Accordion header according to your requirement.

The following screenshot illustrates you the usage of Accordion control in listing the controls under the Essential Studio products.

Copyright © 2001 - 2020 Syncfusion Inc.



In the above screenshot , the Accordion contains a template for its Header and its content. In this Accordion application, you can list the controls under Essential Studio.

Create a Simple Accordion

ASP.NET MVC Accordion basically renders using a div element. The following step describes the creation of Accordion control.

- You can create a MVC Project and add necessary Dll's with the help of the given MVC-Getting Started Documentation.
- Please add the below code in layout._cshtml file head section to add the necessary script and CSS files to render the Accordion control.

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/13.1.0.21/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<!--Scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.easing.1.3.min.js">
</script>
<script src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.web.all.min.js">
</script>
</head>
```

Copyright © 2001 - 2020 Syncfusion Inc.

- Add the following code example to the corresponding view page for Accordion rendering.

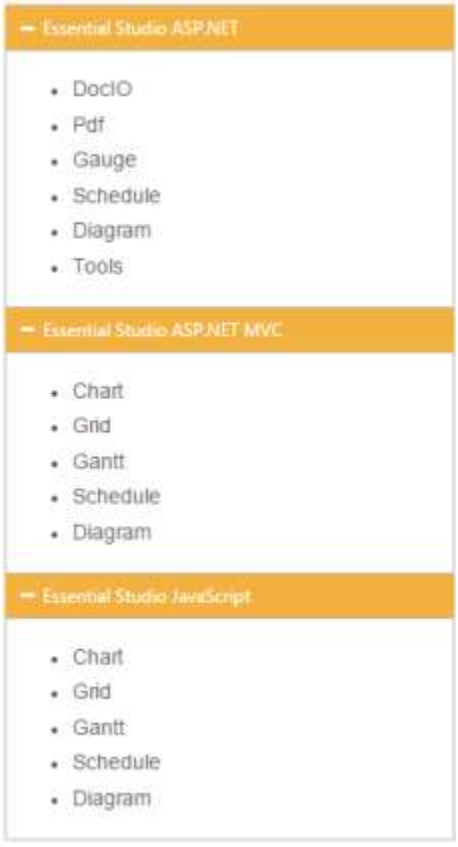
ASPX-CS

```
@{Html.EJ().Accordion("basicAccordion").Width("400px").EnableMultipleOpen(tr
ue).Items(data =>
{
data.Add().Text("Essential Studio ASP.NET").ContentTemplate(@
<div>
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>PDF  </h4>
</li>
<li>
<h4>Gauge  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio ASP.NET MVC").ContentTemplate(@
<div>
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio JavaScript").ContentTemplate(@
<div>
<ul>
<li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
</ul>
</div>);
}).Render();}
```

Execute the above code example to display the Accordion control with simple control list.



You can customize the Accordion control using various properties. The Accordion control properties and its default values are described in the following section.

Configure Multiple Open

You can open multiple Accordion tabs to view all products at a time. To render this set the EnableMultipleOpen property of the Accordion control to true.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: EnableMultipleOpen property is false by default.

You can also open all the panels during initialization using the SelectedItems property of the Accordion control. The following code sample illustrates the opening of multiple tabs by passing the tab index values of tab.

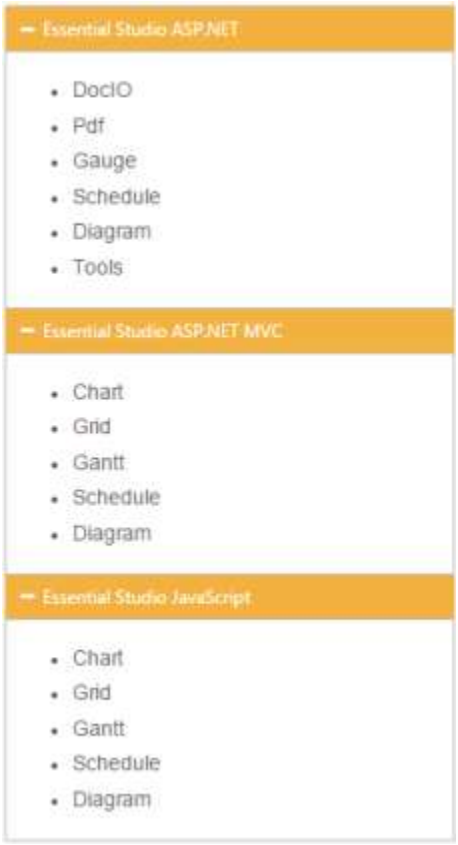
HTML

```
@{
List
<int> selected = new List
<int>() { 0, 1, 2 };
}
@{Html.EJ().Accordion("basicAccordion").EnableMultipleOpen(true).SelectedIte
ms(selected).Items(data =>
{
data.Add().Text("Essential Studio ASP.NET").ContentTemplate(@
<div>
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>PDF  </h4>
</li>
<li>
<h4>Gauge  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio ASP.NET MVC").ContentTemplate(@
<div>
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</ul>
</div>);
data.Add().Text("Essential Studio JavaScript").ContentTemplate(@
<div>
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
</ul>
</div>);
}).Render();
}
```

Accordion control with EnableMultipleOpen property is illustrated in the following screenshot.



Copyright © 2001 - 2020 Syncfusion Inc.

Set Rounded corner

Accordion control by default is rendered in a regular rectangle. You can modify the regular rectangles with rounded corners by setting the ShowRoundedCorner property to true.

Note: ShowRoundedCorner property is false by default.

ASPX-CS

```
<div id="ProductsAccordion" style="width: 400px">
@{
List<int> selected = new List<int>() { 0, 1, 2 };
}
@{Html.EJ().Accordion("basicAccordion").EnableMultipleOpen(true).SelectedIte
ms(selected).ShowRoundedCorner(true).Items(data =>
{
data.Add().Text("Essential Studio ASP.NET").ContentTemplate(@<div>
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>PDF  </h4>
</li>
<li>
<h4>Gauge  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio ASP.NET MVC").ContentTemplate(@<div>
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio JavaScript").ContentTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
</ul>
</div>);
}).Render();
}
```

The following screenshot illustrates the Accordion control with rounded corners.



Customize Icon

You can customize the Header icon using Custom Icon property. This property is having two features such as Header and SelectedHeader. By default, the classes of Header and SelectedHeader are e-collapse and e-expand respectively.

You can change the + and - symbols in the Accordion header, that are default icons with Up and Down arrow icons.

Up and Down arrow icons are available in e-arrowheadup and e-arrowheaddown classes respectively in the ej.widgets.core.min.css stylesheets from the sample.

You can set the Up/Down arrow icon to Accordion header, by adding e-arrowheadup and e-arrowheaddown class to SelectedHeader and Header properties respectively.

ASPX-CS

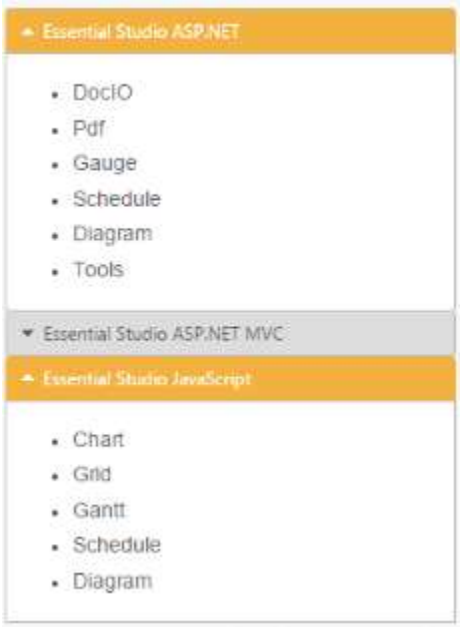
```
@{Html.EJ().Accordion("basicAccordion").EnableMultipleOpen(true).ShowRounded
Corner(true).CustomIcon(icon => icon.SelectedHeader("e-
arrowheadup").Header("e-arrowheaddown")).Items(data =>
```

```
{
data.Add().Text("Essential Studio ASP.NET").ContentTemplate(@<div>
<ul>
<li>
<h4>DocIO</h4>
</li>
<li>
<h4>PDF  </h4>
</li>
<li>
<h4>Gauge  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
<li>
<h4>Tools </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio ASP.NET MVC").ContentTemplate(@<div>
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
<h4>Diagram  </h4>
</li>
</ul>
</div>);
data.Add().Text("Essential Studio JavaScript").ContentTemplate(@<div>
<ul>
<li>
<h4>Chart </h4>
</li>
<li>
<h4>Grid  </h4>
</li>
<li>
<h4>Gantt  </h4>
</li>
<li>
<h4>Schedule  </h4>
</li>
<li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<h4>Diagram  </h4>
</li>
</ul>
</div>);
}).Render();
}
```

The following screenshot illustrates the customization of SelectedHeader and Header of the Accordion control.



Header customization

Collapsible

Accordion widget allows you to set Collapsible state for an Accordion header. Thus you can expand and collapse accordion contents. By default Collapsible is set to false.

Enable Collapsible settings

The following code explains to enable Collapsible state for Accordion using MVC helper.

ASPX-CS

```
// In the View page, render Accordion with corresponding data and enable
Collapsible property.
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).Collapsible(true).Render();}
</div>
```

Output for Accordion control with collapsible headers.



Enable Header expand

Accordion widget provides you support to set the event, where the headers should expand and collapse. The events properties takes default events like mouseout, mouseover, and click.

Configure header expand event

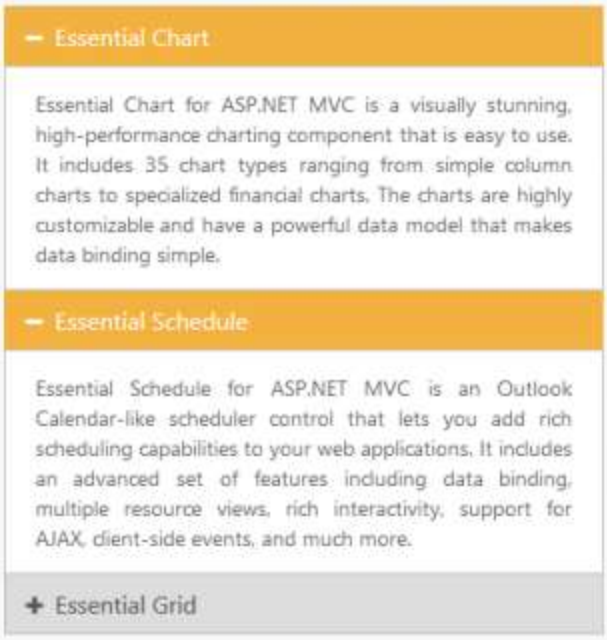
The following code explains you to configure header expand event for Accordion.

ASPX-CS

```
// In the View page, render Accordion with corresponding data and configure
the Events property.
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).Events("mouseout").Render();}
</div>
```

Output for Accordion control that expands header on mouseout event is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Single selection

BY using SelectedItemIndex property you can modify the expanded panel when the control is rendered. By default SelectedItemIndex is '0' that always activate the first Accordion panel.

Specify the selected item in Accordion panel

Using SelectedItemIndex property you can modify the expanded panel when the control is rendered. By default SelectedItemIndex is '0' that always activate the first Accordion panel.

Specify the selected item in Accordion panel

The following code explains you to configure selected item for Accordion.

ASPX-CS

```
// In the View page, render Accordion with corresponding data and configure
selected item for Accordion based on the index.
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
```

```
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).SelectedItemIndex(2).Render();}
</div>
```

Output for Accordion control with the selected item by index is as follows.



Multiple selection

In Accordion widget you can select multiple panel items using SelectedItems property. It takes array of indices that needs to be selected on rendering the control. As you need to select multiple items, you can set EnableMultipleOpen to true.

Configure multiple selection in Accordion panel

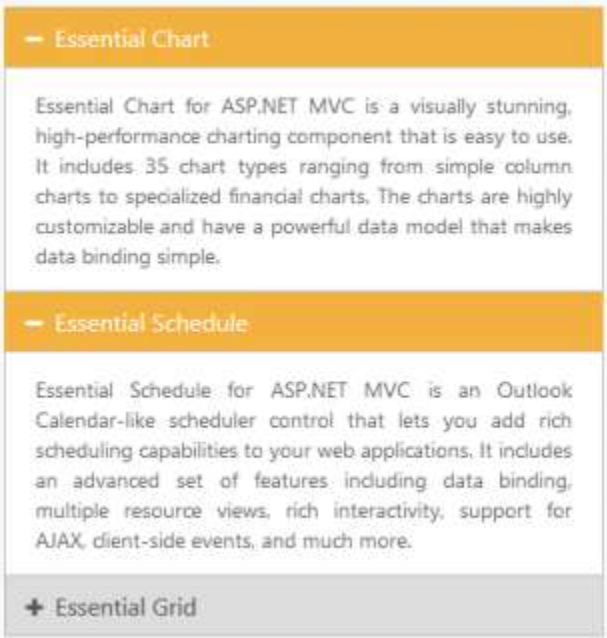
The following code explains to configure selected items for Accordion.

ASPX-CS

```
// In the View page, render Accordion with corresponding data and configure
multiple items selection for Accordion.
<div style="width: 400px">
@{      //List of integer array with index values.
List<int> selected = new List<int>() { 0, 1, 2 };
}
```

```
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).SelectedItems(selected).EnableMultipleOpen(true).Render();}
</div>
```

Output for Accordion control with the multiple selected items is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.

AJAX settings

Accordion widgets allow you to load content for the Accordion panel using AJAX. This renders content from the specified URL location that is set to the anchor tag. You can set the destination file URL string by using AjaxUrl property AJAX contents enables you to load the content of the Accordion panel when it is expanded. This enhances the Accordion control efficiency when a large content is to be loaded into the panel.

Populate accordion with AJAX content

The following code explains about loading accordion content using AjaxUrl property.

ASPX-CS

```
// In the View page, render Accordion with corresponding data and set
AjaxUrl property for the corresponding headers.
<div style="width: 800px; float:left;">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("MVC").AjaxUrl("../../Content/mvccontent.html");
data.Add().Text("WPF").AjaxUrl("../../Content/wpfcontent.html");
data.Add().Text("WCF").ContentTemplate(@<div>
WCF is a tool often used to implement and deploy a service-oriented
architecture (SOA).It is designed using service-oriented architecture
principles to support distributed computing where services have remote
consumers. Clients can consume multiple services; services can be consumed
by multiple clients. Services are loosely coupled to each other. Services
typically have a WSDL interface (Web Services Description Language) that any
WCF client can use to consume the service, regardless of which platform the
service is hosted on. WCF implements many advanced Web services (WS)
standards such as WS-Addressing, WS-ReliableMessaging and WS-Security. With
the release of .NET Framework 4.0, WCF also provides RSS Syndication
Services, WS-Discovery, routing and better support for REST
services.</div>);
}).Render();}
</div>
```

Output for Accordion control with loaded AJAX content is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



View multiple contents

By default Accordion allows only one panel to be in expanded state. You can enable multiple panels in expand state by setting EnableMultipleOpen to true.

Enabling multiple panel open

The following code explains you to enable multiple panel for Accordion.

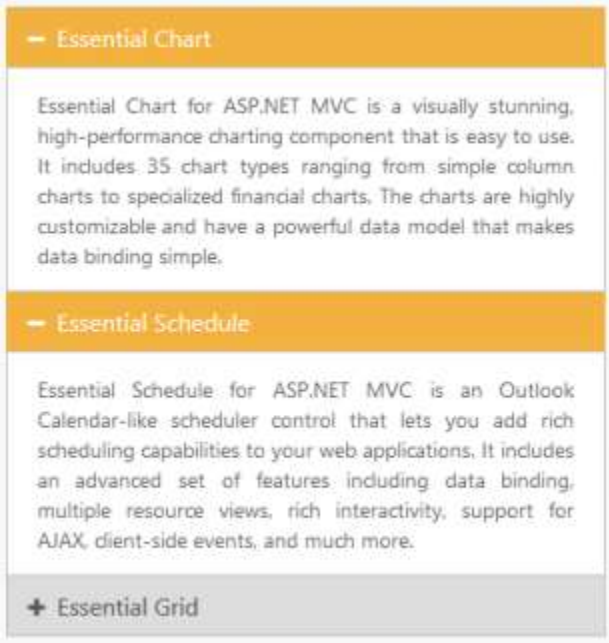
ASPX-CS

```
// In the View page, render Accordion with corresponding data and set
EnableMultipleOpen property as true.
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).EnableMultipleOpen(true).Render();}
</div>
```

Following screenshot is the output for Accordion control on EnableMultipleOpen set to true.



Accordion Panel enabler

Enable or Disable control

You can enable or disable the Accordion widget on initial rendering using the Enabled property. By default Enabled property is set to true and the Accordion panels are active always.

The following code explains you on how to enable or disable the Accordion widget

ASPX-CS

```
// In the View page, render Accordion with corresponding data and disable
the control actions.
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
```

```
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).Enabled(false).Render();}
</div>
```

Output for disabled Accordion control is as follows.



Enable panel items

You can enable the Accordion widget items on initial loading using EnabledItems property. This property takes array of indices whose panel needs to be enabled in Accordion widget.

The DisabledItems property disables the Accordion items based on the index. This takes array of indices whose panel is to be disabled.

Enabling accordion panel items

The following code explains you on how to enable the panel items in Accordion widget.

ASPX-CS

```
// In the View page, configure Accordion with the corresponding data, and to
enable and disable Accordion panel items set the values for EnabledItems and
DisabledItems.
@{      //List of integer array with index values.
List<int> enabled = new List<int>() { 1, 2 };
List<int> disabled = new List<int>() { 0 };
}
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).SelectedItemIndex(1).EnabledItems(enabled).DisableItems(disabled).EnableM
ultipleOpen(true).Render();}
</div>
```

Output for Accordion control with some enabled and disabled items, where first panel is disabled and it can’t be expanded or collapsed is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



State Persistence

Accordion widget can store the model value in the browser cookies and on every time after initial rendering, the control get the model from the cookie only. Using EnablePersistence property you can store the model value in cookies. Thus when any changes are made dynamically then those values are updated in cookie. On refreshing the page the past state of the Accordion control is maintained in cookie and control is rendered from it.

Configure state persistence of Accordion panel

The following code explains to enable state maintenance for Accordion.

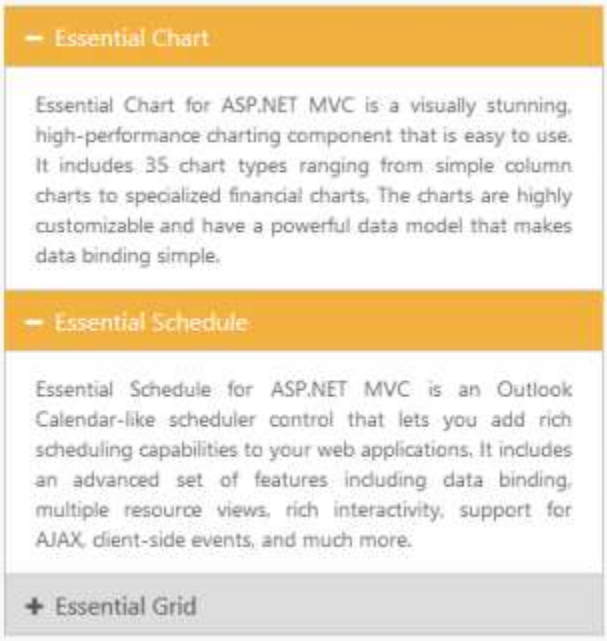
ASPX-CS

```
// In the View page, configure Accordion with corresponding data and enable
the state persistence.
<div style="width: 400px">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
```

Copyright © 2001 - 2020 Syncfusion Inc.

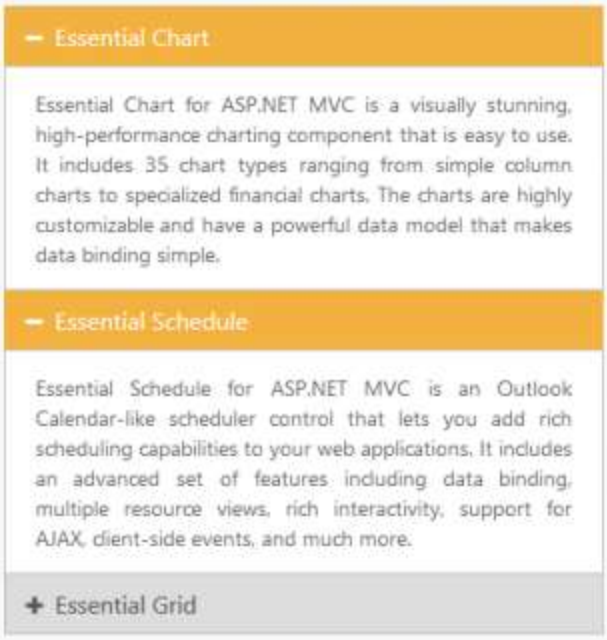
```
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).EnablePersistence(true).Render();}
</div>
```

Output after page refresh maintaining the previous state of Accordion widget is as follows.



1. Accordion Selected Item changed

Copyright © 2001 - 2020 Syncfusion Inc.



2. Accordion state persisted after page refresh

Appearance and Styling

Adjusting Accordion size

You can customize the Accordion panel height using HeightAdjustMode property. It can be set to enum values like content, fill or auto. By default HeightAdjustMode is set to content so the panel height is adjusted to the content size.

Configure Height of Accordion panel

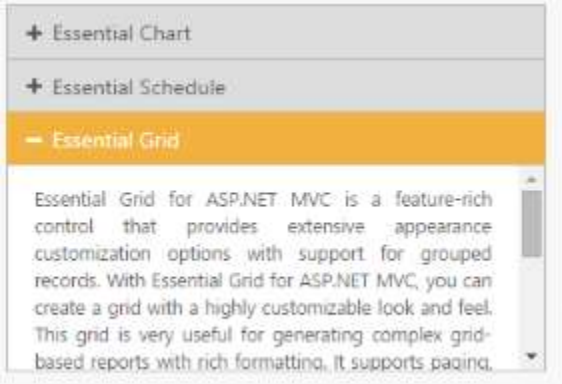
The following code explains to configure the height of the Accordion content panel.

ASPX-CS

```
// In the View page, configure Accordion with corresponding data and define
HeightAdjustMode.
<div style="width:850px">
<div style="width: 400px; float:left;">
<div style="width: 400px; float:right;">
@{Html.EJ().Accordion("accordionFill").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
```

```
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).HeightAdjustMode(HeightAdjustMode.Fill).Render();}
</div>
</div>
```

Output for Accordion control when panel height is set to auto so that the maximum content height and Fill for minimum content height in all the panels is as follows.



Rounded corner

You can customize the shape of the Accordion widget from regular rectangular shape to rounded rectangle shape enabling RoundedCorner property that is set to false by default.

Enabling Rounded corner property

The following code explains you in enabling the ShowRoundedCorner property for an Accordion control.

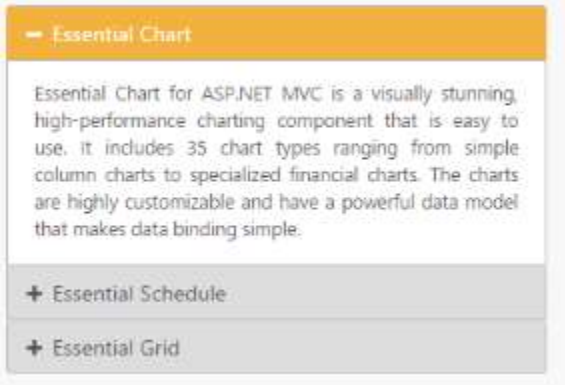
ASPX-CS

```
// In the View page, configure Accordion with corresponding data and enable
the ShowRoundedCorner property for it.
<div style="width: 400px; float:left;">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).ShowRoundedCorner(true).Render();}
</div>
```

Output for accordion widget when “ShowRoundedCorner” is set to “true” is as follows.



Customize Accordion icon

Accordion widget allows you to customize the icons using CustomIcon option that has two properties header and SelectedHeader. By default, the classes of header and selectedHeader are e-collapse and e-expand respectively. By setting the desired CSS class names for these properties as required overrides the default icons with customized icons.

Configuring custom icon for Accordion

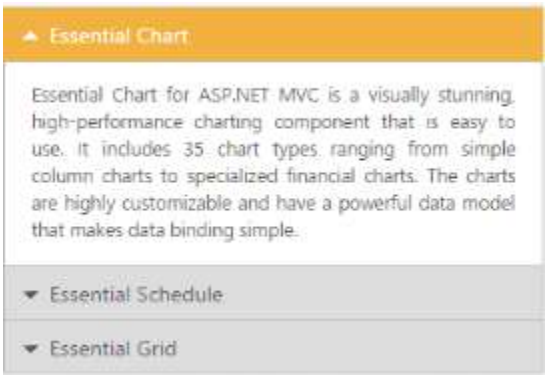
The following code explains you the configuration of icon for an Accordion control.

ASPX-CS

```
// In the View page, configure Accordion with corresponding data and set the
“e-arrowheaddown” and “e-arrowheadup” classes to Header and SelectedHeader
properties.  “E-arrowheaddown” and “e-arrowheadup” are available in
ej.widgets.core.min.css file.
<div style="width: 400px; float:left;">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
```

```
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).CustomIcon(icon =>icon.SelectedHeader("e-arrowheadup").Header("e-
arrowheaddown")).Render();}
</div>
```

Output for Accordion widget with customized icons is as follows.



Animation Options

Set animation

By default the Animation for expanding and collapsing is enabled. To remove the Animation you can set the EnableAnimation property to false. This restricts customizing animations as well. By default EnableAnimation is set to true.

Following code disables Animation for Accordion.

ASPX-CS

```
@{Html.EJ().Accordion("accordion").Items(data =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).EnableAnimation(false).Render();}
```

Expand and collapse speed

This feature allows you to set the speed for expanding and collapsing the Accordion panels. By default it is set to 300 in milliseconds. By configuring the animation speed you can optimize the delay in loading the panel content.

The following code sample sets value for ExpandSpeed and CollapseSpeed properties,

ASPX-CS

```
// The following code example sets values for ExpandSpeed and CollapseSpeed
properties.
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).CollapseSpeed("1000").ExpandSpeed("600").Render();}
</div>
```

Theme

You can control the style and appearance of Accordion control based on CSS classes. In order to apply styles to the Accordion control, you can refer two files, ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.widgets.all.min.css file, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 13 themes support available for Accordion control namely

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap-theme

CSS class

CSS class can be used to customize the Accordion control appearance. Define a CSS class as you're your requirement and assign the class name to CssClass property. The custom CSS class must override the predefined CSS styles applied in the Accordion control.

Class or Element name	Description
.e-acrdn	Base class that is added to the Accordion wrapper div element where the custom CSS class will also be added
h3	Header element for the Accordion panel

Copyright © 2001 - 2020 Syncfusion Inc.

span .e-icon	Accordion icon in the header that change on expand and collapse panels
.e-content	Class for the div element that holds the Accordion panelâ€™s content.

Configure Accordion using CSS class

The following code allows you to configure CSS class for an Accordion widget.

ASPX-CS

```
// In the View page, configure Accordion with corresponding data and set the
CssClass property with corresponding CSS class name.
<div style="width: 400px; float:left;">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).CssClass("customCss").Render();}
</div>
```

Define CSS class for customizing the Accordion.

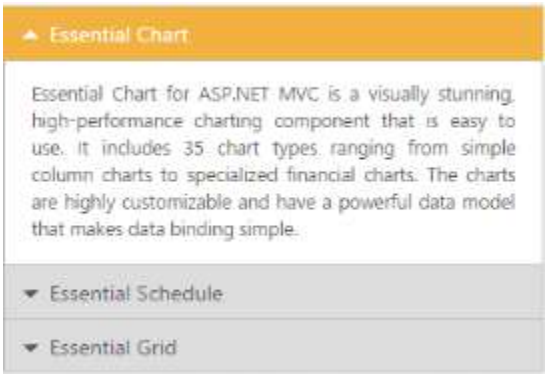
ASPX-CS

```
<style class="cssStyles">
.customCss
{
font-style: italic;
text-align:justify;
}
.customCss span.e-icon {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
display: none !important;
}
.customCss h3
{
text-decoration:underline;
text-align:center;
}
</style>
```

Output for Accordion with customized CSS property to hide the Accordion icon and format its content is as follows.



RTL Support

This feature supports to change the left-to-right alignment of the Accordion control to right-to-left (RTL).

Enabling RTL Support

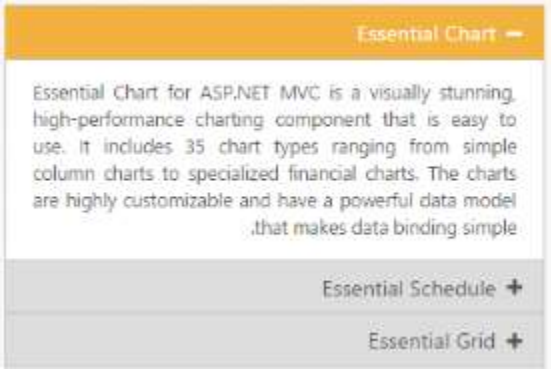
ASPX-CS

```
// In the View page, configure Accordion with corresponding data and set
EnableRTL property value as true.
<div style="width: 400px; float:left;">
@{Html.EJ().Accordion("accordion").Items(data =>
{
data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.</div>);
}).EnableRTL(true).Render();}
</div>
```

Output for accordion when “EnableRTL” is set to “true” is as follows,



Keyboard Navigation

You can use Keyboard shortcut keys as an alternative to the mouse on using Accordion control using AllowKeyboardNavigation property. However you will have to focus the control to enable the keyboard navigation. Accordion Control allows you to perform all kind of actions using keyboard shortcuts.

Shortcut Key	Description
Access keyÂ + j	Focuses into the accordion control
Up	Moves to previous panel
Down	Moves to next panel
Left	Moves to previous panel
Right	Moves to next panel
Home	Moves to the first accordion panel
End	Moves to the last accordion panel

Configure keyboard interaction

The following code explains you on how to enable keyboard interaction for an Accordion widget.

ASPX-CS

```
// In the View page, configure Accordion with corresponding data and
configure Keyboard navigation for Accordion by setting
AllowKeyboardNavigation property to true.
<div style="width: 400px; float:left;">
```

```
@{Html.EJ().Accordion("accordion").Items(data =>
{            data.Add().Text("Essential Chart").ContentTemplate(@<div>
Essential Chart for ASP.NET MVC is a visually stunning, high-performance
charting component that is easy to use. It includes 35 chart types ranging
from simple column charts to specialized financial charts. The charts are
highly customizable and have a powerful data model that makes data binding
simple.
</div>);
data.Add().Text("Essential Schedule").ContentTemplate(@<div>
Essential Schedule for ASP.NET MVC is an Outlook Calendar-like scheduler
control that lets you add rich scheduling capabilities to your web
applications. It includes an advanced set of features including data
binding, multiple resource views, rich interactivity, support for AJAX,
client-side events, and much more.
</div>);
data.Add().Text("Essential Grid").ContentTemplate(@<div>
Essential Grid for ASP.NET MVC is a feature-rich control that provides
extensive appearance customization options with support for grouped records.
With Essential Grid for ASP.NET MVC, you can create a grid with a highly
customizable look and feel. This grid is very useful for generating complex
grid-based reports with rich formatting. It supports paging, sorting,
grouping, filtering, and editing features. It also supports a JSON mode in
which you can handle all the operations like paging and sorting. The
performance of these operations in the JSON mode will be much faster than if
the grid were to handle them. Essential Grid generates clean HTML in
compliance with XHTML 1.0. It supports any kind of IEnumerable data source.
It uses LINQ data retrieval techniques for handling data sources, and offers
high performance.
</div>);
}).AllowKeyboardNavigation(true).Render();}</div>
```

JS

```
// Define script to focus into the Accordion control on Alt + J shortcut
keys.
<script>
$(function () {
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#accordion").focus();
}
});
});
</script>
```

Output for Accordion widget focused and navigated to last item using Keyboard navigation is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Template Support

Content Template

The Content template option provided in MVC is used to specify the HTML elements inside the Accordion control. You can use this option to load any HTML elements and display it in the Accordion panels as per your requirement.

The following code example explains how to use content template option in the Accordion control.

ASPX-CS

```
<div style="width:500px;">
@{Html.EJ().Accordion("pizzaMenu").Items(data =>
{
data.Add().Text("GARDEN FRESH (Veg)").ContentTemplate(@<div>
<img src="~/Content/accordion/garden-veggie.png" alt="garden-fresh" />
<div class="ingredients">
Rate     : $50
<br />
Ingredients : cheese, onions, green capsicums & tomatoes.
</div>
</div>);
data.Add().Text("CORN & SPINACH (Veg)").ContentTemplate(@<div>
<img src="~/Content/accordion/corn-and-spinach-05.png" alt="garden-fresh" />
<div class="ingredients">
Rate     : $70
<br />
Ingredients : cheese, sweet corn & green capsicums.
</div>
</div>);
data.Add().Text("CHICKEN DELITE (Non-veg)").ContentTemplate(@<div>
<img src="~/Content/accordion/chicken-delite.png" alt="garden-fresh" />
<div class="ingredients">
Rate     : $100
<br />
```

```
Ingredients : cheese, chicken chunks, onions & pineapple chunks.
</div>
</div>);
data.Add().Text("KEEMA LA JAWAB (Non-veg)").ContentTemplate(@<div>
<img src="@Url.Content("~/Images/accordion/chicken-delite.png")"
alt="garden-fresh" />
<div class="ingredients">
Rate     : $95
<br />
Ingredients : lamb keema, onions, garlic & tandoori seasoning.
</div>
</div>);
}).EnableMultipleOpen(true).Render();}
</div>
```

Output:



Header Template

The Header template option provided in MVC is used to specify the HTML elements inside the Accordion control.

The following code example explains how to use header template option in the Accordion control.

ASPX-CS

```
@{
ViewBag.Title = "Accordion";
Layout = "~/Views/Shared/_Layout.cshtml";
}
@model AccordionIcons.AccordionController.SI_MODULO
<h2>Accordion</h2>
<br/>
<br/>
<div id = "ControlRegion">
<div style="width: 400px;">
@{Html.EJ().Accordion("iconAccordion").Items(data =>
{
foreach (var m in Model.image)
{
var node = data.Add().Text(m.text).HeaderTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<image class='logos'
src='http://js.syncfusion.com/demos/web/content/images/Employees/@m.imageID'
/>
</div>).ContentTemplate(@<div> This is just a sample text.</div>);
}
}).Render();}
</div>
<style type="text/css">
.logos {
float: left;
height: 35px;
margin: -6px 1px 2px -5px;
width: 35px;
}
</style>
</div>
```

C#

```
List<ImagesID> list = new List<ImagesID>();
public ActionResult AccordionFeatures()
{
SI_MODULO model = new SI_MODULO();
list.Add(new ImagesID { imageID = "6.png", text = "Volkswagen" });
list.Add(new ImagesID { imageID = "7.png", text = "Mitsubishi" });
list.Add(new ImagesID { imageID = "8.png", text = "Mercedes-Benz" });
model.image = list;
return View(model);
}
public class ImagesID {
public string imageID { get; set; }
public string text { get; set; }
}
public class SI_MODULO {
public List<ImagesID> image { get; set; }
}
```

How To

Render Accordion from code behind

Accordion can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of Accordion using Accordion properties from code behind.

RAZOR

```
@{
Html.EJ().Accordion("multiAccordion",
(Syncfusion.JavaScript.Models.AccordionProperties)ViewData["accordion"]).Ren
der();
}
```

C#

```
public ActionResult Index()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
AccordionProperties obj = new AccordionProperties();
obj.Items.Add(new AccordionBaseItem() {Text = "ASP.NET", ContentTemplate =
new MvcTemplate<AccordionBaseItem> { RazorViewTemplate = (data) => { return
"Microsoft ASP.NET is a set of technologies in the Microsoft .NET Framework
for building Web applications and XML Web services. ASP.NET pages execute on
the server and generate markup such as HTML, WML, or XML that is sent to a
desktop or mobile browser. ASP.NET pages use a compiled, event-driven
programming model that improves performance and enables the separation of
application logic and user interface."; } } });
obj.Items.Add(new AccordionBaseItem() {Text = "ASP.NET MVC", ContentTemplate
= new MvcTemplate<AccordionBaseItem> { RazorViewTemplate = (data) => {
return " The Model-View-Controller (MVC) architectural pattern separates an
application into three main components: the model, the view, and the
controller. The ASP.NET MVC framework provides an alternative to the ASP.NET
Web Forms pattern for creating Web applications. The ASP.NET MVC framework
is a lightweight, highly testable presentation framework that (as with Web
Forms-based applications) is integrated with existing ASP.NET features, such
as master pages and membership-based authentication."; } } });
obj.Items.Add(new AccordionBaseItem() {Text = "JavaScript", ContentTemplate
= new MvcTemplate<AccordionBaseItem> { RazorViewTemplate = (data) => {
return "JavaScript (JS) is an interpreted computer programming language."; }
} });
obj.EnableAnimation = true;
ViewData["accordion"] = obj;
return View();
}
```

AutoComplete

AutoComplete

The Essential ASP.NET MVC AutoComplete control is a textbox control that provides a list of suggestions based on the user query. When users enter text into the text box, the control performs a search operation and provides a list of results. There are several filter types available to perform the search.

Key Features

- Delimiter: Supports search for multiple words using a delimiter symbol.
- Suggestion box customization: Supports customization of the suggestion box height and width, and also provides you option to customize the items count to be displayed in the suggestion list.
- minCharacter: Sets the minimum number of characters required to perform the search.
- Filter Types: Supports various search filters with case sensitivity search option.
- Watermark: Supports custom watermark text.
- Data binding: Supports data binding with the List of objects and remote data.
- Template: Supports templates for the suggestion list content.
- Highlight search: Supports highlighting the typed text in the suggestion list.
- Auto fill: Allows the control to automatically fill the autocomplete textbox based on the character typed.
- Keyboard navigation: Provides complete support to work with the control by using the keyboard keys.
- RTL: Supports displaying the text box and suggestion list from right to left.

Copyright © 2001 - 2020 Syncfusion Inc.

- Themes: The AutoComplete control features 12 built-in themes (six flat and six with gradient effects), and also supports custom skin options for user defined themes.

Getting Started

This section explains briefly about how to create an AutoComplete in your ASP.NET MVC application.

Create your first AutoComplete in MVC

This section helps you to configure the AutoComplete control in your application and also helps you to learn how to pass the required data to it and to customize its various options according to your requirements.

In this section, the flat-saffron theme (default) for the Autocomplete is explained.

The following screen shot illustrates the AutoComplete control that searches the list of components available in the database.



AutoComplete control with search

Create an AutoComplete

ASP.NET MVC AutoComplete textbox widget basically renders with built-in features like keyboard navigation with animations and flexible API's. You can easily create the AutoCompleteTextbox widget by the following steps.

1. You can create a MVC Project and add necessary Dll's and Scripts with the help of the given MVC-Getting Started Documentation.
2. Initialize the corresponding AutoComplete widget in the view page.

ASPX-CS

```
<div>
Select Component/s:
@Html.EJ().Autocomplete("ComponentList")
</div>
```

3. Execute the code to render a AutoComplete widget as follows



Output

Copyright © 2001 - 2020 Syncfusion Inc.

Populate Data to AutoComplete

You can provide either local data or remote data to the Autocomplete.

Local Data Binding

AutoComplete provides extensive data binding support to populate AutoComplete items, so that the values are mapped to the AutoComplete fields, namely Key and Text. DataBinding helps you bind a key value pair to AutoComplete textbox. Key field takes the unique id of the dataSource elements. Text field gets the value to be displayed in the AutoComplete textbox.

Defining the Local data for AutoComplete

The following steps explain local data binding to an AutoComplete textbox.

You need to add the class in the Models. Define the Class with key and text field. Then create a List of that class and add the data.

C#

```
public class ComponentsList
{
public int ComponentId { get; set; }
public string ComponentName { get; set; }
public static List<ComponentsList> GetComponentsList()
{
List<ComponentsList> component = new List<ComponentsList>();
component.Add(new ComponentsList { ComponentName = "Autocomplete" });
component.Add(new ComponentsList { ComponentName = "Accordion" });
component.Add(new ComponentsList { ComponentName = "BulletGraph" });
component.Add(new ComponentsList { ComponentName = "Chart" });
component.Add(new ComponentsList { ComponentName = "DatePicker" });
component.Add(new ComponentsList { ComponentName = "Dialog" });
component.Add(new ComponentsList { ComponentName = "Diagram" });
component.Add(new ComponentsList { ComponentName = "DropDown" });
component.Add(new ComponentsList { ComponentName = "Gauge" });
component.Add(new ComponentsList { ComponentName = "Schedule" });
component.Add(new ComponentsList { ComponentName = "Scrollbar" });
component.Add(new ComponentsList { ComponentName = "Slider" });
component.Add(new ComponentsList { ComponentName = "RangeNavigator" });
component.Add(new ComponentsList { ComponentName = "Rating" });
component.Add(new ComponentsList { ComponentName = "RichTextEditor" });
component.Add(new ComponentsList { ComponentName = "Tab" });
component.Add(new ComponentsList { ComponentName = "TagCloud" });
component.Add(new ComponentsList { ComponentName = "Toolbar" });
component.Add(new ComponentsList { ComponentName = "TreeView" });
return component;
}
}
```

In the controller page, you need to pass the model class to the corresponding view.

C#

```
public ActionResult Index()
{
return View(ComponentsList.GetComponentsList());
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

In the View page, add Autocomplete helper and map the local data list to corresponding DataSource and AutoCompleteFields. You need to refer the model class at the top of the page.

ASPX-CS

```
@model List<MvcApplication7.Models.ComponentsList>
<div>
Select Component/s:
@Html.EJ().Autocomplete("ComponentList").Datasource(Model).AutocompleteFields(f=> f.Text("ComponentName").Key("ComponentId")).Width("500")
</div>
```

Run this code to render the AutoComplete with components list.



AutoComplete with Component list

You can also set some common customization changes to the AutoComplete textbox like enabling multiple-selection, highlight search and add dropdown icon based on your requirement.

Configure Visual Mode with filter option

By default, the AutoComplete is rendered with single-value selection. For multiple-value selection using the property MultiSelectMode that allows you to select multiple Data. There are two types of multiple selection one is ‘delimiter’ and another one is ‘visual mode’. In ‘Delimiter’ mode, the multiple values chosen are separated by using the delimiter character specified. In ‘visual mode’, the values chosen are displayed as box model. Here, the ‘visual mode’ is shown. You can set the FilterType option as StartsWith to sort the suggestion list based on the starting character.

ASPX-CS

```
<div>
Select Component/s:
@Html.EJ().Autocomplete("ComponentList").Datasource(Model).AutocompleteFields(f=>
f.Text("ComponentName").Key("ComponentId")).Width("500").MultiSelectMode(MultiSelectModeTypes.VisualMode).FilterType(FilterOperatorType.StartsWith).Width("500")
</div>
```

The following screen shot displays the AutoComplete textbox with selection visual mode.



Copyright © 2001 - 2020 Syncfusion Inc.

AutoComplete textbox with selection visual mode

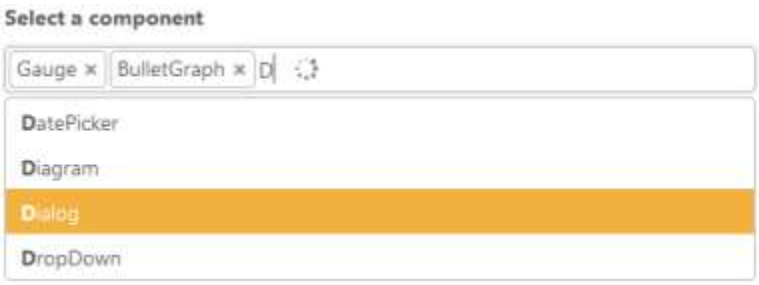
Configure Highlight Search and Rounded corners

When you set the HighlightSearch property to ‘true’, the characters typed in textbox gets highlighted in the suggestion list. To display textbox reforms from sharp ends to rounded ends, you can enable the ShowRoundedCorner property.

ASPX-CS

```
<div>
Select Component/s:
@Html.EJ().Autocomplete("ComponentList").Datasource(Model).AutocompleteFields(f=>
f.Text("ComponentName").Key("ComponentId")).Width("500").MultiSelectMode(MultiSelectModeTypes.VisualMode).FilterType(FilterOperatorType.StartsWith).HighlightSearch(true).ShowRoundedCorner(true).Width("500")
</div>
```

The following screen shot displays the AutoComplete textbox with highlight search enabled.



AutoComplete text box with highlight search enabled

Configure Popup button

To enable the DropDown button, you can set ShowPopupButton property to ‘true’ that displays the DropDown icon at the end of textbox. By default, search icon replaces other icons and so you need to override the CSS classes and replace the content to DropDown arrow icon available in core CSS file as follows.

CSS

```
<style>.e-icon.e-search:before
{
content:"\e63b";
}
</style>
```

ASPX-CS

```
<div>
Select Component/s:
@Html.EJ().Autocomplete("ComponentList").Datasource(Model).AutocompleteFields(f=>
f.Text("ComponentName").Key("ComponentId")).Width("500").MultiSelectMode(MultiSelectModeTypes.VisualMode).FilterType(FilterOperatorType.StartsWith).HighlightSearch(true).ShowRoundedCorner(true).Width("500").ShowPopupButton(true)
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screen shot displays the AutoComplete textbox with dropdown icon.



AutoComplete textbox with dropdown icon

Data-Binding

In order to render the AutoComplete control, the data needs to be bound to it in a proper way. The following sub-properties provide a way to bind either the local or remote data to the AutoComplete widget by binding the appropriate data fields to the corresponding options.

Fields

DataSource

This property assigns the List of objects to the AutoComplete control. Either local data or remote data can be specified.

Query

It accepts the data of object type, usually the query string to fetch the required data from a specific table based on certain condition. This property is optional hence when it is not specified the entire records that are initially assigned through DataSource are taken into consideration.

Key

It maps the corresponding key field name from the data table or the List of objects that is assigned to the DataSource with the Key property of the AutoComplete control. The Key value that is fetched from the table is unique for each records.

Text

It maps the corresponding Text field name from the data table or the List of Objects that is assigned to the DataSource with the Text property of the AutoComplete control. The text value that is fetched from the table represents the value to be displayed in the AutoComplete textbox.

GroupBy

It maps the groupBy field name from the data table or JSON data that is assigned to the dataSource. The groupBy value that is fetched from the table is made to group the values in the datasource.

HtmlAttributes

This allows you to map the CSS styles or classes to the corresponding data from table or the List of Objects data with the AutoComplete items. The HtmlAttributes value customizes the AutoComplete items based on HTML styling or class assigned to it.

Local data

AutoComplete provides extensive data binding support to populate AutoComplete items, so that the values are mapped to the AutoComplete fields, namely Key and Text. DataBinding helps you bind a key

Copyright © 2001 - 2020 Syncfusion Inc.

value pair to AutoComplete textbox. Key field takes the unique id of the dataSource elements. Text field gets the value to be displayed in the AutoComplete textbox.

Defining the Local data for AutoComplete

The following steps explain local data binding to an AutoComplete textbox.

1. In the Controller page, define the Class with key and text field. Then create a List of that class and add the data.

CSHARP

```
public class CarsList
{
public int UniqueKey { get; set; }
public string Text { get; set; }
}
public partial class AutocompleteController : Controller
{
List<CarsList> cars = new List<CarsList>();
public ActionResult AutocompleteFeatures()
{
cars.Add(new CarsList { UniqueKey = 1, Text = "Audi S6" });
cars.Add(new CarsList { UniqueKey = 2, Text = "Austin-Healey" });
cars.Add(new CarsList { UniqueKey = 3, Text = "Alfa Romeo" });
cars.Add(new CarsList { UniqueKey = 4, Text = "Aston Martin" });
cars.Add(new CarsList { UniqueKey = 5, Text = "BMW 7" });
cars.Add(new CarsList { UniqueKey = 6, Text = "Bentley Mulsanne" });
cars.Add(new CarsList { UniqueKey = 7, Text = "Bugatti Veyron" });
cars.Add(new CarsList { UniqueKey = 8, Text = "Chevrolet Camaro" });
cars.Add(new CarsList { UniqueKey = 9, Text = "Cadillac" });
cars.Add(new CarsList { UniqueKey = 10, Text = "Duesenberg J " });
cars.Add(new CarsList { UniqueKey = 11, Text = "Dodge Sprinter" });
cars.Add(new CarsList { UniqueKey = 12, Text = "Elantra" });
cars.Add(new CarsList { UniqueKey = 13, Text = "Excavator" });
cars.Add(new CarsList { UniqueKey = 14, Text = "Ford Boss 302" });
cars.Add(new CarsList { UniqueKey = 15, Text = "Ferrari 360" });
cars.Add(new CarsList { UniqueKey = 16, Text = "Ford Thunderbird" });
cars.Add(new CarsList { UniqueKey = 17, Text = "GAZ Siber" });
cars.Add(new CarsList { UniqueKey = 18, Text = "Honda S2000" });
cars.Add(new CarsList { UniqueKey = 19, Text = "Hyundai Santro" });
cars.Add(new CarsList { UniqueKey = 20, Text = "Isuzu Swift" });
cars.Add(new CarsList { UniqueKey = 21, Text = "Infiniti Skyline" });
cars.Add(new CarsList { UniqueKey = 22, Text = "Infiniti Skyline" });
cars.Add(new CarsList { UniqueKey = 23, Text = "Kia Sedona EX" });
cars.Add(new CarsList { UniqueKey = 24, Text = "Koenigsegg Agera" });
cars.Add(new CarsList { UniqueKey = 25, Text = "Lotus Esprit" });
cars.Add(new CarsList { UniqueKey = 26, Text = "Lamborghini Diablo" });
cars.Add(new CarsList { UniqueKey = 27, Text = "Mercedes-Benz" });
cars.Add(new CarsList { UniqueKey = 28, Text = "Mercury Coupe" });
cars.Add(new CarsList { UniqueKey = 29, Text = "Maruti Alto 800" });
cars.Add(new CarsList { UniqueKey = 30, Text = "Nissan Qashqai" });
cars.Add(new CarsList { UniqueKey = 31, Text = "Oldsmobile S98" });
cars.Add(new CarsList { UniqueKey = 32, Text = "Opel Superboss" });
cars.Add(new CarsList { UniqueKey = 33, Text = "Porsche 356" });
cars.Add(new CarsList { UniqueKey = 34, Text = "Pontiac Sunbird" });
cars.Add(new CarsList { UniqueKey = 35, Text = "Scion SRS/SC/SD" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
cars.Add(new CarsList { UniqueKey = 36, Text = "Saab Sportcombi" });
cars.Add(new CarsList { UniqueKey = 37, Text = "Subaru Sambar" });
cars.Add(new CarsList { UniqueKey = 38, Text = "Suzuki Swift" });
cars.Add(new CarsList { UniqueKey = 39, Text = "Triumph Spitfire" });
cars.Add(new CarsList { UniqueKey = 40, Text = "Toyota 2000GT" });
cars.Add(new CarsList { UniqueKey = 41, Text = "Volvo P1800" });
cars.Add(new CarsList { UniqueKey = 42, Text = "Volkswagen Shirako" });
cars.Add(new CarsList { UniqueKey = 43, Text = "Ford Boss 302" });
cars.Add(new CarsList { UniqueKey = 44, Text = "Ferrari 360" });
cars.Add(new CarsList { UniqueKey = 45, Text = "Ford Thunderbird" });
ViewBag.datasource = cars;
return View();
}
}
```

2. In the View page, add Autocomplete helper and map the Local data list to corresponding DataSource and AutoCompleteFields.

ASPX-CS

```
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
```

The following image is the output for AutoComplete control with local data binding.



AutoComplete with local data-binding

Remote data

AutoComplete provides remote data binding support to populate AutoComplete items, so that the values are mapped to the AutoComplete fields from a remote web service using DataManager and Query.

DataManager is used to manage relational data. It supports CRUD that is Create, Read, Update, and Destroy, in individual requests and batches. DataManager uses two different classes, ej.DataManager, for processing and ej.Query, for serving data. ej.DataManager communicates with dataSource and ej.Query generates data queries that are read by the DataManager.

Configuring remote data for AutoComplete

The below following types are used to bind remote datasource to an AutoComplete textbox.

Copyright © 2001 - 2020 Syncfusion Inc.

OData

OData is a standardized protocol for creating and consuming data. You can provide the OData service URL directly to the Datasource URL property.

HTML

```
@Html.EJ().Autocomplete("selectCar").Datasource("http://js.syncfusion.com/ej
services/Wcf/Northwind.svc/").Query("ej.Query().from('Customers').take(10)")
.AutocompleteFields(af => af.Text("CustomerID")).Width("300")
```

Run the code to get the following output



OData Version 4

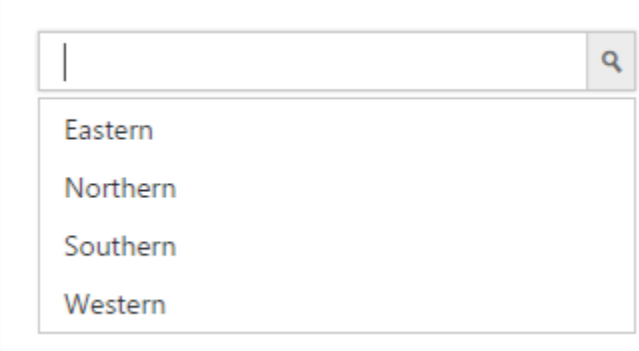
For OData Version 4 support ODataV4Adaptor should be used. By using URL property of Datasource, you can bind OData Version 4 Service link and specify Adaptor value as enum AdaptorType.ODataV4Adaptor.

For further details about OData service please refer the link.

HTML

```
@Html.EJ().Autocomplete("selectData").Datasource(dataSource =>
dataSource.URL("http://services.odata.org/V4/Northwind/Northwind.svc/Regions
/").Adaptor(AdaptorType.ODataV4Adaptor)).AutocompleteFields(af =>
af.Text("RegionDescription")).Width("300").ShowPopupButton(true)
```

Run the code to get the following output.



URL Adaptor

URL Adaptor of **DataManager** can be used when you are required to use remote service to retrieve data. It interacts with server-side for all **DataManager** Queries and **CRUD** operations.

Copyright © 2001 - 2020 Syncfusion Inc.

Now, in the following code example the data is retrieved from **Controller**.

HTML

```
@Html.EJ().Autocomplete("selectCar").Datasource(dataSource =>
dataSource.URL(Url.Action("DataSource",
"Autocomplete")).Adaptor(AdaptorType.UrlAdaptor)).Query("ej.Query()").Autoco
mpleteFields(af =>
af.Text("text")).Width("300").FilterType(FilterOperatorType.Contains)
```

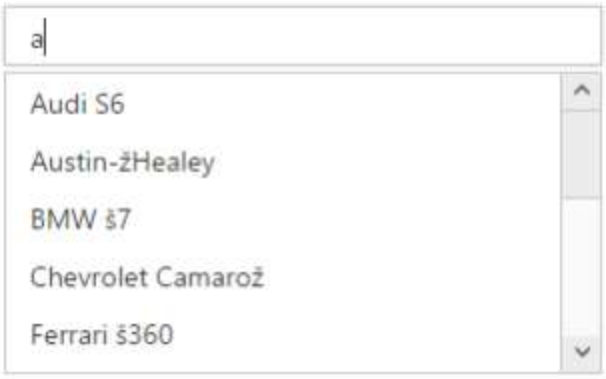
Controller:

C#

```
public JsonResult DataSource()
{
IEnumerable Data = setListSource();
return Json(Data, JsonRequestBehavior.AllowGet);
}
public static List<CarsList> setListSource()
{
List<CarsList> cars = new List<CarsList>();
cars.Add(new CarsList { uniqueKey = 1, text = "Audi S6", company = "Audi"
});
cars.Add(new CarsList { uniqueKey = 2, text = "Austin-žHealey", company =
"Austin" });
cars.Add(new CarsList { uniqueKey = 3, text = "BMW š7", company = "BMW" });
cars.Add(new CarsList { uniqueKey = 4, text = "Chevrolet Camarož", company =
"Chevrolet" });
cars.Add(new CarsList { uniqueKey = 6, text = "Ferrari š360", company =
"Ferrari" });
cars.Add(new CarsList { uniqueKey = 7, text = "Honda S2000", company =
"Honda" });
cars.Add(new CarsList { uniqueKey = 8, text = "Hyundai Santroš", company =
"Hyundai" });
cars.Add(new CarsList { uniqueKey = 9, text = "Isuzu Swift", company =
"Isuzu" });
cars.Add(new CarsList { uniqueKey = 10, text = "Jaguar XJS", company =
"Jaguar" });
cars.Add(new CarsList { uniqueKey = 11, text = "iLotus Esprit", company =
"Lotus" });
cars.Add(new CarsList { uniqueKey = 12, text = "Mercedes-Benz", company =
"Mercedes" });
cars.Add(new CarsList { uniqueKey = 13, text = "Toyota ž2000GT", company =
"Toyota" });
cars.Add(new CarsList { uniqueKey = 14, text = "Volvo P1800", company =
"Volvo" });
return cars;
}
```

Run the code to get the following output

Copyright © 2001 - 2020 Syncfusion Inc.



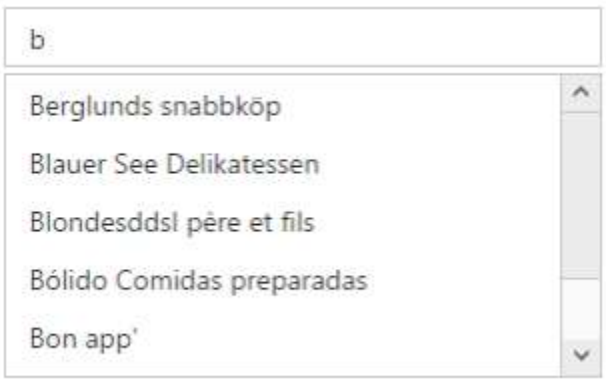
WebAPI

WebApi Adaptor, extended from **ODataAdaptor**, of **DataManager** is used for retrieving data from WebApi service. ASP.NET Web API is a Framework for building HTTP services. You can retrieve data from ASP.NET Web API by using DataManager. Using WebApiAdaptor, you can bind WebApi service's data to Autocomplete. The data from WebApi service must be returned as an object that has its value as data source and field property customerID with its value as Datasource total records count.

HTML

```
@(Html.EJ().Autocomplete("selectAuto").Datasource(dataSource =>
dataSource.URL(Url.Action("http://js.syncfusion.com/ejservices/Wcf/Northwind
.svc/")).Adaptor(AdaptorType.WebApiAdaptor)).Query("ej.Query().from('Custome
rs')").Width("300").AutocompleteFields(fieldSetting =>
fieldSetting.Text("CustomerID")))
```

Run the code to get the following output.



Behavior Settings

Filtering Type

AutoComplete textbox supports a wide range of filtering options to search items in the PopUp list. The FilterType takes enum or string values. The default value is StartsWith. Other supported values are Contains, EndsWith, LessThan, GreaterThan, GreaterThanOrEqual, LessThanOrEqual, Equal and NotEqual.

Copyright © 2001 - 2020 Syncfusion Inc.

Defining the Filter type

The following steps explain the configuration of the filtering conditions for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and add the filter type as contains.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.FilterType(FilterOperatorType.Contains)
```

The following image is the output for AutoComplete control that filters list items based on the ‘contains’ option.



AutoComplete using "contains" filterType

AutoFill

The AutoComplete textbox widget offers an AutoFill option. This feature is used to automatically fill the item when text is entered when EnableAutoFill is set to ‘true’. The first Item in the suggestions list that matches the entered text is automatically displayed in the AutoComplete textbox. The search text is selected in the AutoComplete textbox for identification.

This feature reduces the need to type the entire text and makes the search box more efficient. This is used only with filterType “StartsWith”, since text is filled automatically based on the text entered.

Configure AutoFill property in AutoComplete

The following steps explain how to enable the AutoFill property for an AutoComplete textbox.

1. In the View page, add the Autocomplete helper and enable the EnableAutoFill property

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@(Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>
)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.EnableAutoFill(true))
```

The following image is the output for AutoComplete when EnableAutoFill is set to ‘true’.

Copyright © 2001 - 2020 Syncfusion Inc.



AutoComplete with AutoFill

Sorting Items

AutoComplete widget allows you to sort the suggestions list items and set the sorting order. To enable sorting, set AllowSorting to 'true'. It is enabled by default. The SortOrder property takes enum values, either in ascending or descending order.

Steps to define sorting order and distinct

The following steps explain how to enable the sorting property for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and enable AllowSorting.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.AllowSorting(true).SortOrder(SortOrder.Descending)
```

The following image is the output for AutoComplete when "SortOrder" is configured.



AutoComplete PopUp sorted in descending order

Distinct List items

AutoComplete widget provides an option to extract repeating items in the PopUp list. By setting EnableDistinct property to 'true', you can prevent the duplicate items in the suggestions list.

Steps to enable distinct items

The following steps explain you how to enable this property.

1. In the View page, define the AutoComplete control and configure the EnableDistinct property.

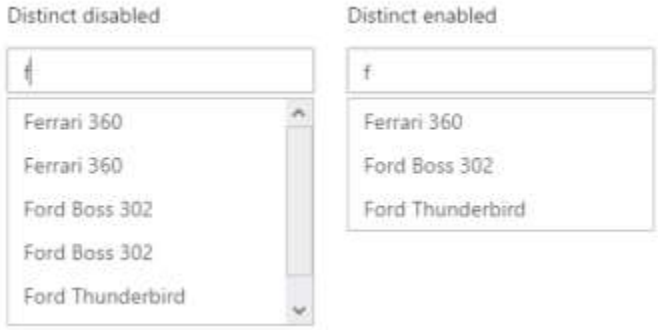
ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
<div style="width: 400px">
<div style="float: left;">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<h6><span style="font-style: italic; font-weight: normal; position:
absolute; margin-top: -30px;">
Distinct disabled</span></h6>
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.EnableDistinct(false)
</div>
<div id="binding" style="float: right;">
<h6><span style="font-style: italic; font-weight: normal; position:
absolute; margin-top: -30px;">
Distinct enabled</span></h6>
@Html.EJ().Autocomplete("autocompleteDistinct")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.EnableDistinct(true)
</div>
</div>
```

The following images are the outputs for AutoComplete when EnableDistinct is set to “true” and “false”.



AutoComplete PopUp items with Distinct property disabled and enabled

Show Popup button

Show Popup button property provides you with an option to display an icon, to show the popup list in the AutoComplete widget.

Enabling Popup button

The following steps explains you how to configure the Popup button for an AutoComplete textbox.

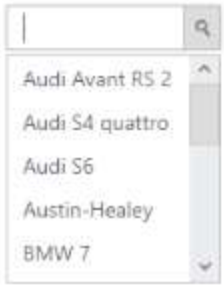
1. In the View page, define the AutoComplete control and enable ShowPopupButton property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.ShowPopupButton(true)
```

The following image is the output for AutoComplete when ShowPopupButton is enabled.

Copyright © 2001 - 2020 Syncfusion Inc.



AutoComplete with popup icon

Restrict editing

AutoComplete textbox widget provides ReadOnly property to disable editing in the control, so that the value set to AutoComplete can only be read and cannot be modified by the user. The value property allows you to set the default value for AutoComplete widget, when it is created.

Configure AutoComplete textbox to restrict editing

The following steps help you to disable editing in an AutoComplete textbox.

1. In the View page, define the AutoComplete control and configure Readonly property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.Readonly(true).Value("BMW 7")
.ShowPopupButton(true)
```

The following image is the output for the AutoComplete textbox configured to restrict editing.



AutoComplete with readOnly property

Empty Result settings

The AutoComplete widget allows you to configure the display message when the list doesn't return any values. By default, ShowEmptyResultText is set to 'true' and EmptyResultText is set to the string value "No suggestions".

Configure Empty result setting

The following steps allow you to set text for empty results of an AutoComplete textbox.

1. In the View page, define the AutoComplete control and set the desired text message to be shown in EmptyResultText.

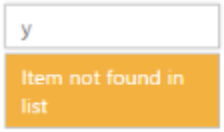
ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.ShowEmptyResultText(true).EmptyResultText("Item not found in list")
```

The following image is the output of the AutoComplete textbox when the list doesn't return any value.



AutoComplete with customized EmptyResultText

Search Customization

HighlightSearch

AutoComplete widget allows you to highlight the search text in the AutoComplete suggestions list using the HighlightSearch property. When this property is set to 'true', the suggestions list appear with the search text it contains is highlighted.

Enabling highlightSearch option

The following steps explain how you can enable the HighlightSearch property for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and enable the HighlightSearch property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.HighlightSearch(true).FilterType(FilterOperatorType.Contains)
.MinCharacter(2)
```

The following image is the output for AutoComplete when HighlightSearch is set to 'true'.



AutoComplete with highlightSearch enabled

Case-sensitive Search

AutoComplete allows you to enable case sensitivity to filter the suggest list items based on the entered text casing. This property enables strict filtering of list items based on entered text. To enable it, set CaseSensitiveSearch value to 'true'. It is false, by default.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure case sensitivity for AutoComplete

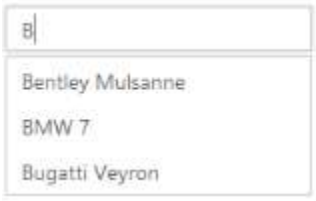
The following steps explain you how to enable the CaseSensitiveSearch property for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and enable the CaseSensitiveSearch property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.CaseSensitiveSearch(true)
```

The following image is the output for AutoComplete when CaseSensitiveSearch is set to ‘true’.



AutoComplete textbox with caseSensitiveSearch

MultiSelection modes

AutoComplete widget allows you to select multiple values from the suggestions list using the MultiSelectMode property. Multiple values are selected when MultiSelectMode value is set to VisualMode or Delimiter.

Delimiter mode separates multiple items using a separator character defined. When the delimiter mode is set, you can define the delimiter character using DelimiterChar. It takes a single character and it is a symbol.

VisualMode selects multiple items by enclosing the item in a rounded rectangle with a close icon to remove item from the selection.

Configuring MultiSelection Mode

The following steps explain the configuration of the MultiSelectMode for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and configure multiple selection mode.

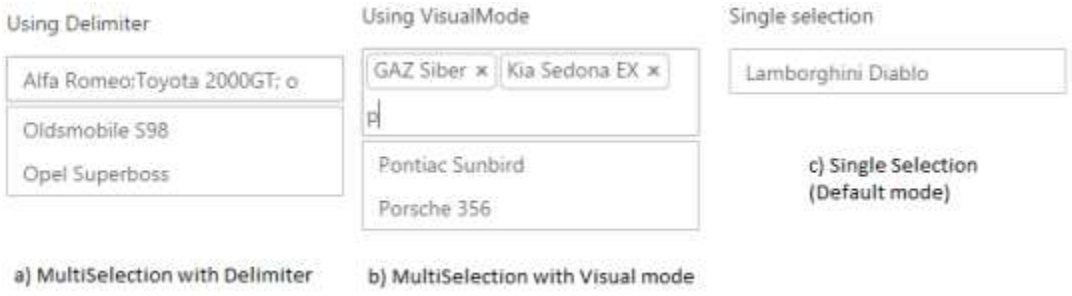
ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
<div style="width: 600px">
<div style="display:inline-block; float:left; margin-right:25px">
<span style="display:block; margin-bottom:12px">Using Delimiter</span>
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.MultiSelectMode(MultiSelectModeTypes.Delimiter).Delimiter(";")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
<div style="display:inline-block; float:left; margin-right:25px">
<span style="display:block; margin-bottom:12px">Using VisualMode</span>
@Html.EJ().Autocomplete("autocompleteVisual")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.MultiSelectMode(MultiSelectModeTypes.VisualMode)
</div>
<div style="display:inline-block; float:left;">
<span style="display:block; margin-bottom:12px">Single selection</span>
@Html.EJ().Autocomplete("autocompleteSingle")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field => field.Key("UniqueKey").Text("Text"))
.MultiSelectMode(MultiSelectModeTypes.None)
</div>
</div>
```

The following image is the output for AutoComplete control with configured multiple selection.



AutoComplete with MultiSelection

Multiple Columns

The Autocomplete allows list of data to be displayed in several columns and column collection can be defined and customized through the **MultiColumnSettings** property.

In AutoComplete Multiple Column search is based on **searchColumnIndices** property which allows user to search text for any number of fields in the suggestion list without modifying the selected text format.

In AutoComplete Multiple Column searched value is updated to autocomplete input box based on **StringFormat** property which specifies values to updated.

Note: StringFormat as “{0} ({1}) ({2})” means search based on 0, 1 and 2 columns data.

Name	Description
Enable	Allow list of data to be displayed in several columns.
ShowHeader	Allow header text to be displayed in corresponding columns.
StringFormat	Displayed selected value and autocomplete search based on mentioned column value specified in that format.
Columns	Field and Header Text collections can be defined and customized through this field.

Copyright © 2001 - 2020 Syncfusion Inc.

Field	Get or set a value that indicates to display the columns in the autocomplete mapping with column name of the dataSource.
HeaderText	Get or set a value that indicates to display the title of that particular column.

Configuring Multiple Columns

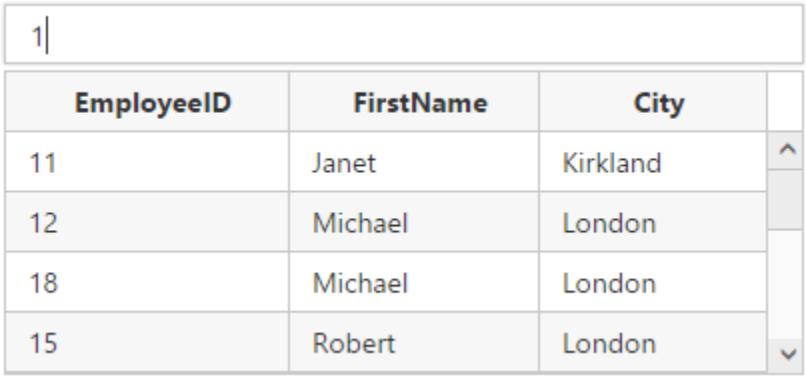
The following steps explain the configuration of the Multiple Columns for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and configure multicolumn.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
<div style="width: 600px">
<div style="display:inline-block; float:left; margin-right:25px">
<span style="display:block; margin-bottom:12px">Using Delimiter</span>
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.MultiColumnSettings(obj => obj.Enable(true).Columns(obj1 =>
{
obj1.Field("EmployeeID").HeaderText("EmployeeID").Add();
obj1.Field("FirstName").HeaderText("FirstName").Add();
obj1.Field("City").HeaderText("City").Add();
}).ShowHeader(true).SearchColumnIndices(new List<int> { 0,1,2
}).StringFormat("{0}"))
</div>
</div>
```

The following image is the output for AutoComplete control with configured multiple column.



AutoComplete with MultiColumn

Grouping

AutoComplete widget provides Grouping support for the suggestions list based on the category specified in the dataSource. By default AllowGrouping is set to ‘false’. To enable Grouping for your AutoComplete widget, set the value to ‘true’.

Configuring Grouping for AutoComplete

The following steps explain you how to configure Grouping for an AutoComplete textbox.

Copyright © 2001 - 2020 Syncfusion Inc.

1. In the Controller page, define the data list with Key, Text, and Category field for grouping.

CSHARP

```
public class CarsList
{
public int UniqueKey { get; set; }
public string Text { get; set; }
public string Category { get; set; }
}
public partial class AutocompleteController : Controller
{
List<CarsList> cars = new List<CarsList>();
public ActionResult AutocompleteFeatures()
{
cars.Add(new CarsList { UniqueKey = 1, Text = "Audi S6", Category = "Audi"
});
cars.Add(new CarsList { UniqueKey = 2, Text = "Austin-Healey", Category =
"Austin" });
cars.Add(new CarsList { UniqueKey = 3, Text = "BMW 7", Category = "BMW" });
cars.Add(new CarsList { UniqueKey = 4, Text = "Chevrolet Camaro", Category =
"Chevrolet" });
cars.Add(new CarsList { UniqueKey = 5, Text = "Mercedes-Benz", Category =
"Mercedes" });
cars.Add(new CarsList { UniqueKey = 6, Text = "Toyota 2000GT", Category =
"Toyota" });
cars.Add(new CarsList { UniqueKey = 7, Text = "Volvo P1800", Category =
"Volvo" });
cars.Add(new CarsList { UniqueKey = 8, Text = "Audi Avant RS 2", Category =
"Audi" });
cars.Add(new CarsList { UniqueKey = 9, Text = "Audi S4 quattro", Category =
"Audi" });
cars.Add(new CarsList { UniqueKey = 10, Text = "BMW M Roadster E85",
Category = "BMW" });
cars.Add(new CarsList { UniqueKey = 11, Text = "Mercedes-Benz Sprinter",
Category = "Mercedes" });
cars.Add(new CarsList { UniqueKey = 12, Text = "Corvette", Category =
"Chevrolet" });
cars.Add(new CarsList { UniqueKey = 13, Text = "Volvo P1800", Category =
"Volvo" });
ViewBag.datasource = cars;
return View();
}
}
```

2. In the View page, define the AutoComplete control and then give the Category filed for GroupBy field property.

HTML

```
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource).Width("205")
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following image is the output for AutoComplete control that provides Grouping.



AutoComplete popup panel with grouping

State Persistence

AutoComplete widget can store the model value in the browser's cookies. Every time after initial rendering, the control gets the model from the cookie only. Using EnablePersistence property you can store the model value in cookies. So when any changes are made dynamically, the values are updated in the cookie. On refreshing the page, the previous state of the AutoComplete control is maintained in the cookie and the control is rendered from it.

Configure state persistence of AutoComplete

The following steps explain you how to enable state maintenance for AutoComplete.

1. In the View page, define the AutoComplete control and configure state persistence by using EnablePersistence property.

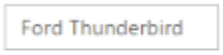
ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.EnablePersistence(true)
```

The following image is the output for AutoComplete when EnablePersistence is set to 'true'.



1. Autocomplete during value selection



2. Autocomplete after page refresh

AutoComplete with State maintenance

Templates

You can provide a Template for customizing the appearance of the AutoComplete textbox suggestions. This is achieved by assigning a string template to the Template property.

Configuring Templates

The following steps explain you how to define a Template to display a text and image for an AutoComplete textbox.

1. Define DataSource elements with required template fields by using the Sprite CSS class names.

CSHARP

```
public partial class AutocompleteController : Controller
{
List<Flags> flags = new List<Flags>();
public ActionResult AutocompleteFeatures()
{
flags.Add(new Flags { Text = "Algeria", Sprite = "flag-algeria" });
flags.Add(new Flags { Text = "Armenia", Sprite = "flag-am" });
flags.Add(new Flags { Text = "Brazil", Sprite = "flag-br" });
flags.Add(new Flags { Text = "Bangladesh", Sprite = "flag-bangladesh" });
flags.Add(new Flags { Text = "Canada", Sprite = "flag-ca" });
flags.Add(new Flags { Text = "Cuba", Sprite = "flag-cu" });
flags.Add(new Flags { Text = "China", Sprite = "flag-china" });
flags.Add(new Flags { Text = "Denmark", Sprite = "flag-denmark" });
flags.Add(new Flags { Text = "Estonia", Sprite = "flag-estonia" });
flags.Add(new Flags { Text = "Egypt", Sprite = "flag-eg" });
flags.Add(new Flags { Text = "France", Sprite = "flag-fr" });
flags.Add(new Flags { Text = "Finland", Sprite = "flag-fi" });
flags.Add(new Flags { Text = "Greenland", Sprite = "flag-gl" });
flags.Add(new Flags { Text = "India", Sprite = "flag-in" });
flags.Add(new Flags { Text = "Indonesia", Sprite = "flag-id" });
flags.Add(new Flags { Text = "Malaysia", Sprite = "flag-my" });
flags.Add(new Flags { Text = "Mexico", Sprite = "flag-mexico" });
flags.Add(new Flags { Text = "New Zealand", Sprite = "flag-new-zealand" });
flags.Add(new Flags { Text = "Netherlands", Sprite = "flag-nl" });
flags.Add(new Flags { Text = "Norway", Sprite = "flag-no" });
flags.Add(new Flags { Text = "Portugal", Sprite = "flag-pt" });
flags.Add(new Flags { Text = "Poland", Sprite = "flag-pl" });
flags.Add(new Flags { Text = "Qatar", Sprite = "flag-qatar" });
flags.Add(new Flags { Text = "Romania", Sprite = "flag-ro" });
flags.Add(new Flags { Text = "Spain", Sprite = "flag-es" });
flags.Add(new Flags { Text = "Singapore", Sprite = "flag-singapore" });
flags.Add(new Flags { Text = "Saudi Arabia", Sprite = "flag-sa" });
flags.Add(new Flags { Text = "Thailand", Sprite = "flag-th" });
flags.Add(new Flags { Text = "Turkey", Sprite = "flag-tr" });
flags.Add(new Flags { Text = "Ukraine", Sprite = "flag-ukraine" });
flags.Add(new Flags { Text = "United States", Sprite = "flag-us" });
flags.Add(new Flags { Text = "Uruguay", Sprite = "flag-uruguay" });
flags.Add(new Flags { Text = "Viet Nam", Sprite = "flag-viet-nam" });
ViewBag.datasource = flags;
return View();
}
}
public class Flags
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public string Text { get; set; }
public string Sprite { get; set; }
}
```

2. Configure template structure for AutoComplete control as follows, including a

   element with image and text in every row of the popup panel. Assign the corresponding variable name within ${} to map them into the list.

ASPX-CS

```
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<Flags>)ViewBag.datasource)
.AutocompleteFields(fields => fields.Text("Text"))
.Template("<div class='flag ${Sprite}'></div><div class='txt'> ${_Text_}
</div>")
```

3. Define the CSS classes for the sprite images. You can find the images in the following location:

[Installed Drive]:\Users\[user name]\AppData\Local\Syncfusion\EssentialStudio\X.X.X.X\MVC\Samples\web\Images\autocomplete\flags.png

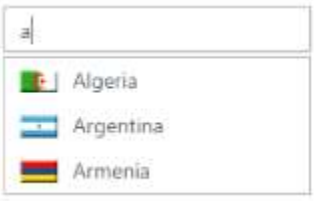
S

```
<style type="text/s">
/* Sprite s for country flags */
.flag
{
background: url("Styles/flags.png") no-repeat;
float: left;
height: 15px;
margin-right: 10px;
margin-top: 3px;
width: 25px;
}
.flag.flag-am {background-position: -25px 0}
.flag.flag-ar {background-position: -50px 0}
.flag.flag-bangladesh {background-position: -75px 0}
.flag.flag-br {background-position: -100px 0}
.flag.flag-ca {background-position: -125px 0}
.flag.flag-china {background-position: 0 -15px}
.flag.flag-cu {background-position: -25px -15px}
.flag.flag-denmark {background-position: -50px -15px}
.flag.flag-algeria {background-position: -75px -15px}
.flag.flag-estonia {background-position: -100px -15px}
.flag.flag-eg {background-position: -125px -15px}
.flag.flag-es {background-position: 0 -30px}
.flag.flag-fi {background-position: -25px -30px}
.flag.flag-fr {background-position: -50px -30px}
.flag.flag-gl {background-position: -75px -30px}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.flag.flag-id {background-position: -100px -30px}
.flag.flag-in {background-position: -125px -30px}
.flag.flag-mexico {background-position: 0 -45px}
.flag.flag-my {background-position: -25px -45px}
.flag.flag-nl {background-position: -50px -45px}
.flag.flag-no {background-position: -75px -45px}
.flag.flag-new-zealand {background-position: -100px -45px}
.flag.flag-pl {background-position: -125px -45px}
.flag.flag-pt {background-position: 0 -60px}
.flag.flag-qatar {background-position: -25px -60px}
.flag.flag-ro {background-position: -50px -60px}
.flag.flag-sa {background-position: -75px -60px}
.flag.flag-singapore {background-position: -100px -60px}
.flag.flag-th {background-position: -125px -60px}
.flag.flag-tr {background-position: 0 -75px}
.flag.flag-ukraine {background-position: -25px -75px}
.flag.flag-us {background-position: -50px -75px}
.flag.flag-uruguay {background-position: -75px -75px}
.flag.flag-viet-nam {background-position: -100px -75px}
.flag.flag-ye {background-position: -125px -75px}
.txt {
display: table-cell;
height: 20px;
vertical-align: middle;
}
</style>
```

The following image is the output for AutoComplete widget with Template support.



AutoComplete with Custom template

Appearance and Styling

Adjusting AutoComplete size

AutoComplete widget allows you to set the height and width of the textbox element in AutoComplete. The Height and Width properties take pixel values to set the dimension accordingly.

Define height and width for AutoComplete textbox

The following steps explain the dimensional properties of an AutoComplete textbox.

1. In the View page, define the AutoComplete control and set values for Height and Width property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.Height("50px").Width("250px")
.MultiSelectMode(MultiSelectModeTypes.Delimiter)
```

The following image is the output for AutoComplete textbox with customized dimensions.



AutoComplete with Customized dimensions

Rounded corner

By enabling the ShowRoundedCorner property, you can customize the shape of the AutoComplete widget from a regular rectangular shape to a rounded rectangle shape that is set to 'false' by default.

Enabling Rounded corner

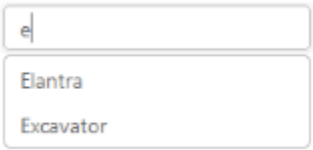
The following steps explain enabling the ShowRoundedCorner property for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and enable the ShowRoundedCorner property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.ShowRoundedCorner(true)
```

The following image is the output for AutoComplete when ShowRoundedCorner is set to "true".



AutoComplete with Rounded corners

Watermark text

WatermarkText property provides you with an option to display a faded text in the AutoComplete textbox when the textbox is empty.

Copyright © 2001 - 2020 Syncfusion Inc.

Defining Watermark text

The following steps explain you how to configure WatermarkText property for an AutoComplete textbox.

1. In the View page, define the AutoComplete control and enable the WatermarkText property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.Width("250px").WatermarkText("Select an item")
```

The following image is the output for AutoComplete when WatermarkText is defined.



AutoComplete loaded with watermark text

Adjusting Suggestion list size

AutoComplete widget provides you with a property to define the dimensions of the popup panel that holds the suggestions list items. The PopupHeight and PopupWidth properties allow you to set the maximum height and width of the popup element for use when the content exceeds the default dimensions.

Configure dimensions of PopUp panel

The following steps help you set height and width of the popup panel of an AutoComplete textbox.

1. In the View page, define the AutoComplete control and configure the Popup panel height and width properties.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.Width("250px").PopupHeight("80px").PopupWidth("350px")
```

The following image is the output for AutoComplete, after configuring the height and width of the popup panel.



Copyright © 2001 - 2020 Syncfusion Inc.

AutoComplete PopUp panel with configured dimensions

PopUp Time Delay

AutoComplete widget allows you to set the time delay to fetch the list items. The value of DelaySuggestionTimeout is set in milliseconds, so that data search time can be configured. This enhances the turnaround time to populate the list items.

Configure Time delay of PopUp panel

The following steps are used to set the time delay to load the popup panel of an AutoComplete textbox.

1. In the View page, add an Autocomplete helper and configure it.

2. Configure the delay time for popup panel in AutoComplete control as follows.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.Width("250px").DelaySuggestionTimeout(1000)
```

This takes 1000ms to display the popup panel list items.

Theme

AutoComplete control's style and appearance are controlled based on CSS classes. In order to apply styles to the AutoComplete control, you can refer 2 files namely, ej.widgets.core.min.s and ej.theme.min.s. When the file ej.widgets.all.min.s is referred, then it is not necessary to include the files ej.widgets.core.min.s and ej.theme.min.s in your project, as ej.widgets.all.min.s is the combination of these two.

By default, there are 12 theme supports available for AutoComplete control namely:

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

CSS Class

CSS class is used to customize the AutoComplete control's appearance. Define CSS class as per requirement and assign the class name to CssClass property.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure AutoComplete textbox using CSS class

The following steps allow you to configure CssClass for an AutoComplete textbox.

1. Define CSS class for customizing the AutoComplete control.

S

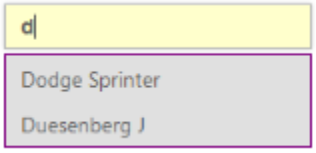
```
<style type="text/s" class="sStyles">
/* Customize the PopUp panel */
.customCss
{
border-color:Purple;
background-color: #E0E0E0;
}
/* Customize the AutoComplete input textbox */
.customCss .e-autocomplete
{
background-color: #FFFFCC;
font-weight:bold;
font-family: sans-serif;
}
</style>
```

2. In the View page, define the AutoComplete control and assign the class name to CssClass property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").Category("Category"))
.Width("250px").CssClass("customCss")
```

The following image is of an AutoComplete textbox configured based on CSS class.



AutoComplete widget configured with CSS class

RTL support

This feature allows you to change the alignment of the AutoComplete textbox widget from left-to-right to right-to-left (RTL). The custom template AutoComplete textbox also supports RTL.

Enabling RTL Support

The following steps explain how you can enable the right-to-left property for an AutoComplete textbox.

Copyright © 2001 - 2020 Syncfusion Inc.

1. In the View page, define the AutoComplete control and enable the RTL property.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@Html.EJ().Autocomplete("autocomplete")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.Width("250px").EnableRTL(true)
```

The following image is the output for AutoComplete when EnableRTL is set to “true”.



AutoComplete template with RTL support

Keyboard Interaction

You can use keyboard shortcut keys as an alternative to the mouse while using the AutoComplete widget. The AutoComplete widget allows you to perform all kinds of actions using keyboard shortcuts.

Keyboard shortcut keys

Shortcut Key	Description
Access keyÂ + j	Focuses into the AutoComplete text box
Up	Moves to previous item in pop up
Down	Moves to next item in pop up
Enter	Selects the focused item
Esc	Closes the popup

Configure Keyboard Interaction

The following steps explain how you can enable keyboard interaction for an AutoComplete textbox.

1. In the View page, define the AutoComplete control.

ASPX-CS

```
@*Refer to the DataSource defined in Local Data binding Step 1 *@
@{IDictionary<string, object> htmlAttribute = new Dictionary<string,
object>();
htmlAttribute.Add("accesskey", "j");
}
@(Html.EJ().Autocomplete("autocomplete").HtmlAttributes(htmlAttribute)
.Datasource((IEnumerable<CarsList>
```

```
)ViewBag.datasource)
.AutocompleteFields(field =>
field.Key("UniqueKey").Text("Text").GroupBy("Category"))
.Width("200px"))
```

2. Run the sample, press Access Key + J to focus in the AutoComplete widget and you can navigate using the arrow keys. Use the Escape key to close the popup.



AutoComplete focused with keyboard shortcut

How to

Render Autocomplete from Code behind

Autocomplete can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Autocomplete properties in the controller.

C#

```
//Namespace to get the JavaScript (Autocomplete) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class AutocompleteController : Controller
{
// GET: Autocomplete
public ActionResult AutocompleteFeatures()
{
//Initializing the Autocomplete model
AutocompleteProperties AutoObj = new AutocompleteProperties();
//Initializing the datasource and other properties
AutoObj.DataSource =
"http://js.syncfusion.com/ejServices/wcf/NorthWind.svc/";
AutoObj.Query = "ej.Query().from('Suppliers').select('SupplierID',
'ContactName')";
AutoObj.AutocompleteFields.Text = "ContactName";
AutoObj.AutocompleteFields.Key = "SupplierID";
AutoObj.Width = "500px";
AutoObj.ShowPopupButton = true;
AutoObj.ShowLoadingIcon = true;
//Passing Autocomplete properties using the ViewData
ViewData["AutoModel"] = AutoObj;
return View();
}
}
}
```

Binding the Autocomplete properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{
Html.EJ().Autocomplete("searchCustomer",(Syncfusion.JavaScript.Models.Autoco
mpleteProperties)ViewData["AutoModel"]).Render();
}
```

How to set autocomplete default value

You can set a value into autocomplete at initial rendering using



Value property. It is used to select a single value from the autocomplete widget at initial rendering state.

Refer to the following code.

RAZOR

```
@Html.EJ().Autocomplete("selectState").Datasource((IEnumerable<states>)ViewB
ag.datasource).AutocompleteFields(field =>
field.Key("index").Text("countryName")).Value("Arizona")
```

C#

```
public partial class AutocompleteController : Controller{
List<states> state = new List<states>();
public ActionResult AutocompleteFeatures()
{
state.Add(new states { index = "s1", countryName = "Alabama" });
state.Add(new states { index = "s2", countryName = "Alaska" });
state.Add(new states { index = "s3", countryName = "Arizona" });
ViewBag.datasource = state;
return View();
}
}
public class states
{
public string index { get; set; }
public string countryName { get; set; }
}
```

The following output is displayed as a result of the previous code example.

![howto](how-to_images/howto1.png)

If you set autocomplete Value property as a string that is present in the data source, the hidden input value holds the corresponding Key value. This is used for validation purpose in the autocomplete. For example, ‘Arizona’ text holds the key value of ‘s3’. The autocomplete default value ‘Arizona’ is present in the ‘countryName’ of data source and also this field is mapped for autocomplete

Copyright © 2001 - 2020 Syncfusion Inc.

```
<span id="selectState_wrapper" class="e-atc e-widget" style="width: 100%;">
  <span class="e-in-wrap e-box">
    <input id="selectState_hidden" type="hidden" name="selectState" value="s3"> == $0
    <input type="text" id="selectState" class="e-autocomplete e-js e-input" role="combobox" aria-label="Autocomplete textbox" aria-
    expanded="false" aria-autocomplete="list" tabindex="0" autocomplete="off" name="selectState" placeholder="Select a state" aria-owns=
    "selectState_suggestion">
  </span>
</span>
```

Text property. So, hidden input value holds the ‘s3’ value.

The following screenshot illustrates the previous hidden input state of code.

If you set autocomplete

```
<span id="selectState_wrapper" class="e-atc e-widget" style="width: 100%;">
  <span class="e-in-wrap e-box">
    <input id="selectState_hidden" type="hidden" name="selectState" value="New York"> == $0
    <input type="text" id="selectState" class="e-autocomplete e-js e-input" role="combobox" aria-label="Autocomplete textbox" aria-
    expanded="false" aria-autocomplete="list" tabindex="0" autocomplete="off" name="selectState" placeholder="Select a state" aria-owns=
    "selectState_suggestion">
  </span>
</span>
```

Value property as a string that is not present in the data source, the hidden input value holds the autocomplete value. For example, if autocomplete default value is ‘New York’ and not present in the ‘countryName’ of data source, the hidden input value holds the ‘New York’string.

Refer to the following code sample.

RAZOR

```
@Html.EJ().Autocomplete("selectState").Datasource((IEnumerable<states>)ViewB
ag.datasource).AutocompleteFields(field =>
field.Key("index").Text("countryName")).Value("New York")
```

The following screenshot illustrates the previous hidden input state of code.

Remote data

The remote data also allows you to set the default value into autocomplete using Value property.

Refer to the following code sample.

RAZOR

```
@Html.EJ().Autocomplete("selectCar").Datasource(dataSource =>
dataSource.URL(Url.Action("GetOutletsForAutocomplete",
"Autocomplete")).Adaptor(AdaptorType.UrlAdaptor)).AutocompleteFields(field
=> field.Text("Name").Key("Id")).Value("Two")
```

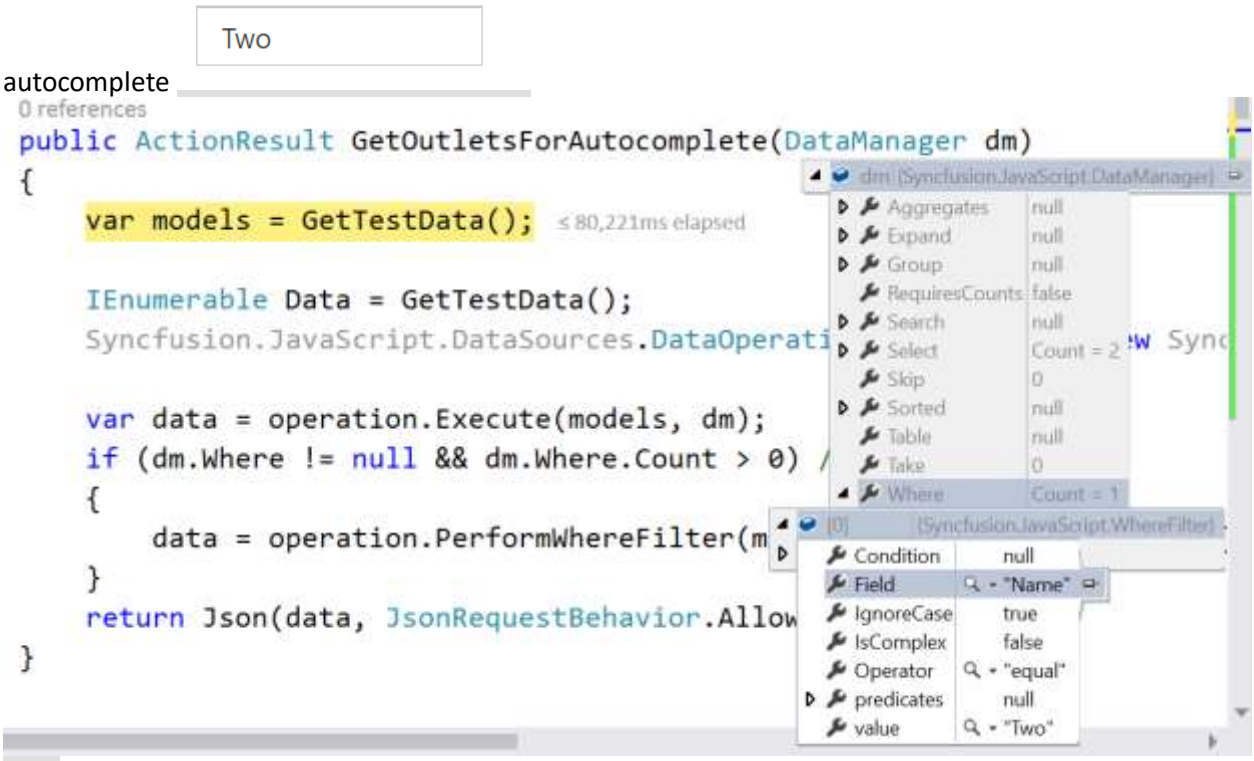
C#

```
public ActionResult GetOutletsForAutocomplete(DataManager dm)
{
var models = GetTestData();
IEnumerable Data = GetTestData();
Syncfusion.JavaScript.DataSources.DataOperations operation = new
Syncfusion.JavaScript.DataSources.DataOperations();
var data = operation.Execute(models, dm);
if (dm.Where != null && dm.Where.Count > 0) //Filtering
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
data = operation.PerformWhereFilter(models, dm.Where, dm.Where[0].Operator);
}
return Json(data, JsonRequestBehavior.AllowGet);
}
public class AutocompleteModel
{
[Display(Name = "Id")]
public string Id { get; set; }
[Display(Name = "Name")]
public string Name { get; set; }
}
private List<AutocompleteModel> GetTestData()
{
var list = new List<AutocompleteModel>();
list.Add(new AutocompleteModel(){ Id = "1", Name = "One"});
list.Add(new AutocompleteModel(){ Id = "2", Name = "Two"});
list.Add(new AutocompleteModel(){ Id = "3", Name = "Three"});
return list;
}
```

The autocomplete Value property triggers a server-side post when using remote data in the autocomplete. The server side data manger holds where query which contains field name as



Text property.

Find the following screenshot for the data manager where query.

![howto](how-to_images/howto6.png)

Find the following output for the previous code.

![howto](how-to_images/howto7.png)

Copyright © 2001 - 2020 Syncfusion Inc.

How to show text on autocomplete using key value

You can set the value of autocomplete text box based on a given key value. The SelectValueByKey property is used to select autocomplete value based on the specified key value.

Refer to the following code sample.

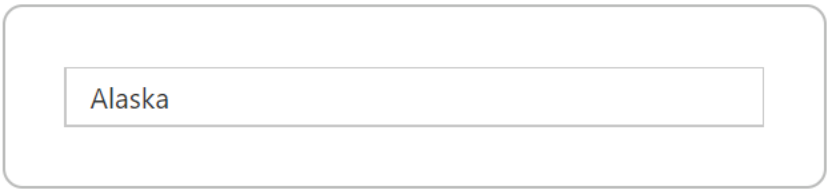
RAZOR

```
@Html.EJ().Autocomplete("selectState").Datasource((IEnumerable<states>)ViewB
ag.datasource).AutocompleteFields(field =>
field.Key("index").Text("countryName")).SelectValueByKey("s2")
```

C#

```
public partial class AutocompleteController : Controller{
List<states> state = new List<states>();
public ActionResult AutocompleteFeatures()
{
state.Add(new states { index = "s1", countryName = "Alabama" });
state.Add(new states { index = "s2", countryName = "Alaska" });
state.Add(new states { index = "s3", countryName = "Arizona" });
ViewBag.datasource = state;
return View();
}
}
public class states
{
public string index { get; set; }
public string countryName { get; set; }
}
```

For example, If SelectValueByKey property specified the value as ‘s2’, the corresponding Text value



Alaska is shown in the autocomplete text.

The following output is displayed as a result of the previous code example.



SelectValueByKey property into autocomplete control, the hidden input value holds a specified key value.

Refer to the following screenshot.

![howto](how-to_images/howto5.png)

Remote data

The remote data also allows you to set the default value into autocomplete based on a given key value using the SelectValueByKey property.

Refer to the following code snippet.

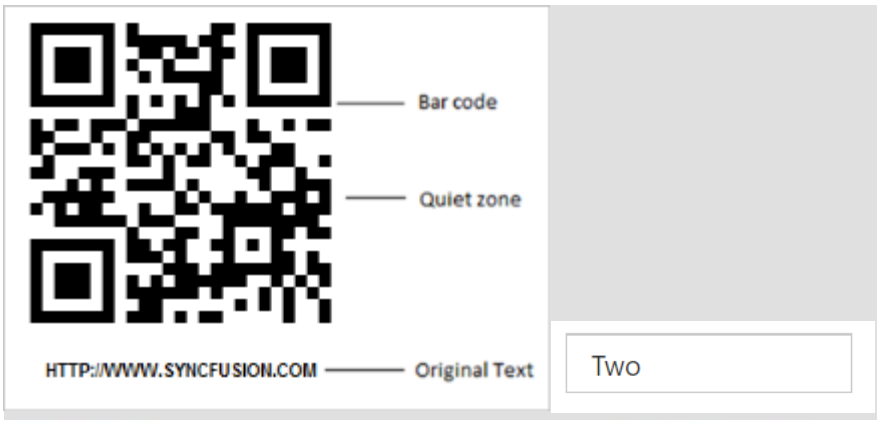
RAZOR

```
@Html.EJ().Autocomplete("selectCar").Datasource(dataSource =>
dataSource.URL(Url.Action("GetOutletsForAutocomplete",
"Autocomplete")).Adaptor(AdaptorType.UrlAdaptor)).AutocompleteFields(field
=> field.Text("Name").Key("Id")).SelectValueByKey("2")
```

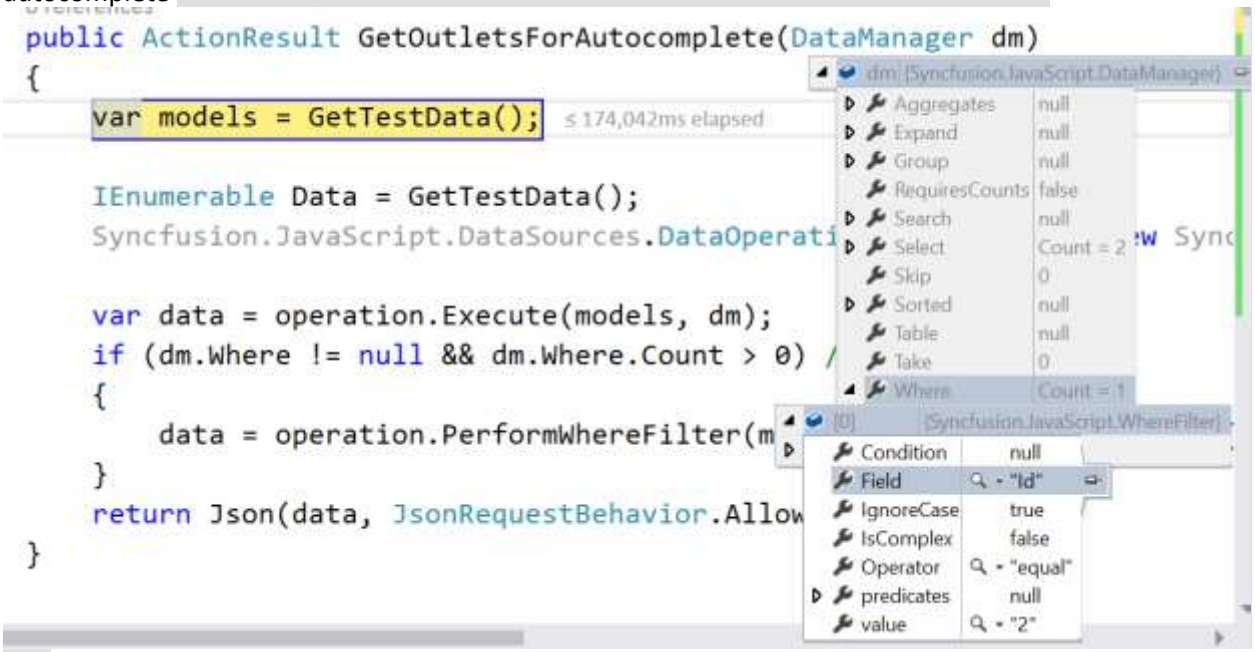
C#

```
public ActionResult GetOutletsForAutocomplete(DataManager dm)
{
var models = GetTestData();
IEnumerable Data = GetTestData();
Syncfusion.JavaScript.DataSources.DataOperations operation = new
Syncfusion.JavaScript.DataSources.DataOperations();
var data = operation.Execute(models, dm);
if (dm.Where != null && dm.Where.Count > 0) //Filtering
{
data = operation.PerformWhereFilter(models, dm.Where, dm.Where[0].Operator);
}
return Json(data, JsonRequestBehavior.AllowGet);
}
public class AutocompleteModel
{
[Display(Name = "Id")]
public string Id { get; set; }
[Display(Name = "Name")]
public string Name { get; set; }
}
private List<AutocompleteModel> GetTestData()
{
var list = new List<AutocompleteModel>();
list.Add(new AutocompleteModel(){ Id = "1", Name = "One"});
list.Add(new AutocompleteModel(){ Id = "2", Name = "Two"});
list.Add(new AutocompleteModel(){ Id = "3", Name = "Three"});
return list;
}
```

The autocomplete SelectValueByKey property triggers a server-side post when using remote data on autocomplete. The server side data manger holds where query which contains field name as



autocomplete



Key property.

Find the following screenshot for the data manager where query.

![howto](how-to_images/howto8.png)

Find the output for the previously given code as follows.

![howto](how-to_images/howto9.png)

Barcode

Barcode

The Barcode control enables rendering of one dimension and two dimension Barcodes in web page. Barcode provides you a simple and inexpensive method of encoding text information that can be easily read by inexpensive electronic readers.

Key Features

- Supports 10 one-dimensional Barcodes including Code 39 and Code 32 barcodes.
- Supports 2 two-dimensional Barcodes such as QR and DataMatrix barcodes.

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Started

This section explains you briefly on how to create a Barcode in your application with ASP.NET MVC.

Control Structure



Control structure

Create your first Barcode in MVC

The ASP.NET MVC Barcode provides you support to encode data and display it in your web page; it takes text and symbology as input and renders the encoded text as Barcode.

The following guideline explains you on how to create a QR Barcode for a real time scenario of encoding a URL. The following screenshot demonstrates a two dimensional barcode encoding the value “HTTP://WWW.SYNCFUSION.COM”.

QR Barcode for the URL “HTTP://WWW.SYNCFUSION.COM”

By default, ASP.NET MVC Barcode renders gray scale colors with predefined Quiet zone. Quiet zone is the blank margin on the side(s) that denotes the reader with the start and stop of the Barcode. It also provides flexible API’s to customize the appearance of the Barcode.

Refer Assemblies

You can create a MVC Project, add necessary assemblies and Scripts. Refer MVC-Getting Started Documentation.

Add the following assemblies in your MVC application for using Barcode control:

- Syncfusion.Core.dll
- Syncfusion.EJ.dll
- Syncfusion.EJ.MVC.dll

These Syncfusion assemblies can either be referenced from Global Assembly Cache (GAC) or from the installed folder.

[Installed Drive]:\Program Files (x86)\Syncfusion\Essential Studio\X.X.X.X\precompiledassemblies\X.X.X.X

Configure Web.Config

Web.config file is configured according to the referenced assemblies. The steps to configure the file are as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

- On the Solution Explorer, double-click the Web.config file in the root folder. The Web.config page appears.
- Add the following assemblies in the Web.config page under the tag:
- Syncfusion.Core
- Syncfusion.EJ
- Syncfusion.EJ.MVC

XML

```
<system.web>
<compilation>
<assemblies>
<add assembly="Syncfusion.Core, Version=x.x.x.x, Culture=neutral,
PublicKeyToken=632609b4d040f6b4"/>
<add assembly="Syncfusion.EJ, Version=x.x.x.x, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.EJ.Mvc, Version=x.x.x.x, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89"/>
</assemblies>
</compilation>
</system.web>
```

Note: X.X.X.X in the above code corresponds to the correct version number of the Essential Studio version that you are currently using.

- Add the following namespaces under the tag.
- Syncfusion.JavaScript
- Syncfusion.JavaScript.DataVisualization
- Syncfusion.Mvc.EJ

XML

```
<system.web>
<pages>
<namespaces>
<add namespace="Syncfusion.JavaScript"/>
<add namespace="Syncfusion.JavaScript.DataVisualization"/>
<add namespace="Syncfusion.MVC.EJ"/>
</namespaces>
</pages>
</system.web>
```

- Open the Index.cshtml file under Views->Home folder. Add the following code in the cshtml file for Barcode rendering.

ASPX-CS

```
<div>
<div>
@Html.EJ().Barcode("barcode")
.Text("HTTP://WWW.SYNCFUSION.COM")
.SymbologyType(BarcodeSymbolType.QRBarcode)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Render()
</div>
</div>
```

- Run the project. Refer the following screenshot for output.



QR Barcode generated in ASP.NET

Barcode types and supported characters

The following table contains the supported types and associated valid characters.

Table of Barcode types and supported character

Symbol	Enum Value	Supported characters	Length
QR Barcode	QRBarcode	0,1,2,3,4,5,6,7,8,90â€“9, Aâ€“Z (upper-case only), space, $, %, *, +, -,., /, :Shift JIS characters	variable
DataMatrix	DataMatrix	All ASCII characters	
Code 39	Code39	0-9, A-Z,a dash(-),a dot(.),$,/,+,%, SPACE	variable
Code 39 Extended	Code39Extended	0-9 A-Z a-z	variable
Code 11	Code11	0-9, a dash(-)	variable
Codabar	Codabar	0-9,-,$,:,/,a dot(.),+	variable
Code 32	Code32	0 1 2 3 4 5 6 7 8 9Text length should be 8	8
Code 93	Code93	1 2 3 4 5 6 7 8 9 0 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z - . $ / + % SPACE	variable
Code 93 Extended	Code93Extended	All 128 ASCII characters	variable
Code 128 A	Code128A	NUL (0x00) SOH (0x01) STX (0x02) ETX (0x03) EOT(0x04) ENQ (0x05) ACK (0x06) BEL (0x07) BS (0x08) HT (0x09) LF (0x0A) VT(0x0B) FF (0x0C) CR (0x0D) SO (0x0E) SI (0x0F) DLE (0x10) DC1 (0x11) DC2(0x12) DC3 (0x13) DC4 (0x14) NAK (0x15) SYN (0x16) ETB (0x17) CAN(0x18) EM (0x19) SUB (0x1A) ESC (0x1B) FS (0x1C)	variable

Copyright © 2001 - 2020 Syncfusion Inc.

		GS (0x1D) RS (0x1E) US(0x1F) SPACE (0x20) " ! # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ AB C D E F G H I J K L M N O P Q R S T U V W X Y Z [/]^ _	
Code 128 B	Code128B	SPACE (0x20) ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 :; < = > ? @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [/]^ _ ` ab c d e f g h i j k l m n o p q r s t u v w x y z { | } ~ DEL (â€¢)	variable
Code 128 C	Code128C	0 1 2 3 4 5 6 7 8 9	variable

Two Dimensional Barcodes

QR Barcode

A QR Barcode is a two-dimensional Barcode that consists of a grid of dark and light dots or blocks that form a square. The data encoded in the Barcode can be numeric, alphanumeric, or Shift Japanese Industrial Standards (JIS8) characters. The QR Barcode uses version from 1 to 40. Version 1 measures 21 modules x 21 modules, Version 2 measures 25 modules x 25 modules, and so on. The number of modules increases in steps of 4 modules per side up to Version 40 that measures 177 modules x 177 modules. Each version has its own capacity. By default, the Barcode control automatically set the version according to the length of the input text. The QR Barcodes are designed for industrial uses and also commonly used in consumer advertising.

Here is the code snippet to create a QR Barcode:

ASPX-CS

```
<div>
<div>
@*Sets the following properties:
Text – Text to encode
SymbologyType - Barcode Type (Default: QRBarcode)
XDimension - Size of each block (Optional, Default: 4)
QuietZone – Sets quiet zone on all sides (Optional, Default: All->1)
*@
@Html.EJ().Barcode("barcode")
.Text("B5330E8278BC4C797C49DD3ED5AD9715")
.XDimension(8)
.SymbologyType(BarcodeSymbolType.QRBarcode)
.QuietZone(qz => { qz.All(30); })
.Render()
</div>
</div>
```

DataMatrix

DataMatrix Barcode is a two dimensional Barcode that consists of a grid of dark and light dots or blocks forming square or rectangular symbol. The data encoded in the Barcode can either be numbers or alphanumeric. They are widely used in printed media such as labels and letters. You can read it easily with the help of a Barcode reader and mobile phones.

Copyright © 2001 - 2020 Syncfusion Inc.

One Dimensional Barcodes

Code 39

The Code 39 character set includes the digits 0-9, the letters A-Z (upper case only), and the following symbols: space, minus (-), plus (+), period (.), dollar sign ($), slash (/), and percent (%). A special start / stop character is placed at the beginning and ending of each Barcode. The Barcode can be of any length; even more than 25 characters begin to push the bounds. Code 39 is the only type of Barcode that does not require a checksum for common use.

Here is the code example to create Code 39 barcode:

HTML

```
<div>
<div>
@*Sets the following properties:
Text - Text to encode
SymbologyType - Barcode Type (Default: QRBarcode)
DisplayText - If original text should be displayed (Optional, Default: true)
QuietZone - Quiet zone on all sides (Optional, Default: All->1)
BarcodeToTextGapHeight - Height between barcode and text (Optional, Default:
10)
BarHeight - Height of bar (Optional, Default: 150)
DarkBarColor - Color of dark bar (Optional, Default: black)
EncodeStartStopSymbol - If start stop symbol should be encoded (Mandatory
for some types, Default: true)
LightBarColor - Color of light bar (Optional, Default: white)
NarrowBarWidth - Width of the narrow bar (Optional, 1)
TextColor - Color of the display text (Optional, black)
WideBarWidth - Width of the wide bar (Optional, 3)
*@
@Html.EJ().Barcode("barcode")
.Text("SYNCFUSION")
.SymbologyType(BarcodeSymbolType.Code39)
.DisplayText(true)
.QuietZone(qz => { qz.All(30);})
.BarcodeToTextGapHeight(10)
.BarHeight(150)
.DarkBarColor("black")
.EncodeStartStopSymbol(true)
.LightBarColor("white")
.NarrowBarWidth(1)
.TextColor("red")
.WideBarWidth(3)
.Render()
</div>
</div>
```

Code 39 Extended

Code 39 Extended is an extended version of Code 39 that supports ASCII character set. In Code 39 Extended, you can also code 26 lower letters (a-z) and the special characters in the keyboard.

Code 11

Code 11 is used primarily for labeling the telecommunication equipment's. The character set includes the digits 0 to 9, a dash (-), and a start / stop code.

Copyright © 2001 - 2020 Syncfusion Inc.

Codabar

Codabar is a variable length symbol that encodes the following 20 characters:

0123456789-$:/.+ABCD

The characters, A, B, C and D are used as start and stop characters. Codabar is used in libraries, blood banks, the package delivery industry and a variety of other information processing applications.

Code 32

It is mainly used for coding pharmaceuticals, cosmetics and dietetics. Code 32 is often used to encode Italian Pharmacode that has the following structure:

- 'A' character (ASCII 65), that is not really encoded
- 8 digits for Pharmacode (It generally begins with / and prefixed with 0)
- 1 digit for Checksum module 10, that is automatically calculated by Barcode

The value to be encoded must be8 digits Pharmacode (prefix it with '0' if necessary) and the 9th digit (the checksum) is automatically calculated by Barcode.

Code 93

Code 93 was designed to complement and improve upon Code 39. It can represent the entire ASCII character set by using combinations of 2 characters. Code 93 is a continuous, variable-length symbology and produces denser code.

- The Standard Mode (default implementation) can encode uppercase letters (A-Z), digits (0-9), and special characters like *, -, $, %, (Space), ., /, and +.

Code 93 Extended

The Code 93 ExtendedBarcode symbology is continuous, variable length, and self-checking. It is based on Code 93 Barcode. The Extended Version can encode all 128 ASCII characters.

Code 128

Code 128 is a variable length, high density, alphanumeric, linear bar code symbology, capable of encoding the full 128-character ASCII character set and extended character sets. This symbology includes a checksum digit for verification and the barcode can also be verified character-by-character by verifying the parity of each data byte.

Code 128 Code Sets

- Code Set A (or Chars Set A) includes all of the standard upper case U.S. alphanumeric keyboard characters and punctuation characters along with the control characters, (namely, characters with ASCII values from 0 to 95 inclusive), and seven special characters.
- Code Set B (or Chars Set B) includes all of the standard upper case alphanumeric keyboard characters and punctuation characters along with the lower case alphabetic characters (namely, characters with ASCII values from 32 to 127 inclusive), and seven special characters.
- Code Set C (or Chars Set C) includes the set of 100 digit pairs from 00 to 99 inclusive along with three special characters. This allows numeric data to be encoded as two data digits per symbol character, at effectively twice the density of standard data.

Copyright © 2001 - 2020 Syncfusion Inc.

Code 128 Special characters

The last seven characters of Code Sets A and B (character values 96 - 102) and the last three characters of Code Set C (character values 100 - 102) are special non-data characters with no ASCII character equivalents that have a particular significance to the Barcode reading device.

Customizing the appearance

A page or printed media with Barcode often appears colorful in the background and surrounding region with other contents. In such cases the Barcode can also be customized to suit the needs. You can achieve this by changing the darkBarColor property.

Note: This color customization is possible only for one dimensional barcodes and it is not supported for two dimensional barcodes.

ASPX-CS

```
<div>
<div
@Html.EJ().Barcode("barcode")
.Text("B5330E8278BC4C797C49DD3ED5AD9715")
.SymbologyType(BarcodeSymbolType.Code39)
.DisplayText(true)
.QuietZone(qz => { qz.All(30);})
.BarcodeToTextGapHeight(10)
.BarHeight(150)
.DarkBarColor("blue")
.EncodeStartStopSymbol(true)
.LightBarColor("white")
.NarrowBarWidth(1)
.TextColor("red")
.WideBarWidth(3)
.Render()
</div>
</div>
```

Execute the above code to render the following output.



Customized Barcode

The height of the barcode can be changed using the BarHeight property. The equivalent property to change the block size for two dimensional barcode is XDimension.

ASPX-CS

```
<div>
<div
@Html.EJ().Barcode("barcode")
.Text("B5330E8278BC4C797C49DD3ED5AD9715")
.SymbologyType(BarcodeSymbolType.Code39)
.DisplayText(false)
.BarcodeToTextGapHeight(10)
.BarHeight(45)
.DarkBarColor("#990099")
.Render()
</div>
</div>
```

Execute the above code to render the following output.



Barcode with customized height

BulletGraph

BulletGraph

BulletGraphs are easy to interpret and also it conveys much more information to the user by visualizing the data in a small amount of space. It is generally used to compare a primary measure to one or more other measures in the context of qualitative ranges of performance. Occasionally, the BulletGraphs are also used to compare the same measure across multiple categories.

Key Features

- Data sources: Bind the BulletGraph control with an array of JSON objects or ej.DataManager.
- Tooltip with template: Template for tooltip support has been included to customize the tooltip.
- FlowDirection: The control can be rendered either in a “forward” or “backward” direction.
- Orientation: The control can be oriented either in “horizontal” or “Vertical” mode.
- Binding Range Stroke to ticks and labels: The Range colors can be bound to its underlying ticks and labels when the properties “applyRangeStrokeToLabels” and “applyRangeStrokeToTicks” are enabled.
- Multiple Measures: Provided support to render multiple feature measure bars as well as multiple comparative measure symbols.
- Animation: Animation support provided for the feature measure bars and the comparative measure symbols.

Getting Started

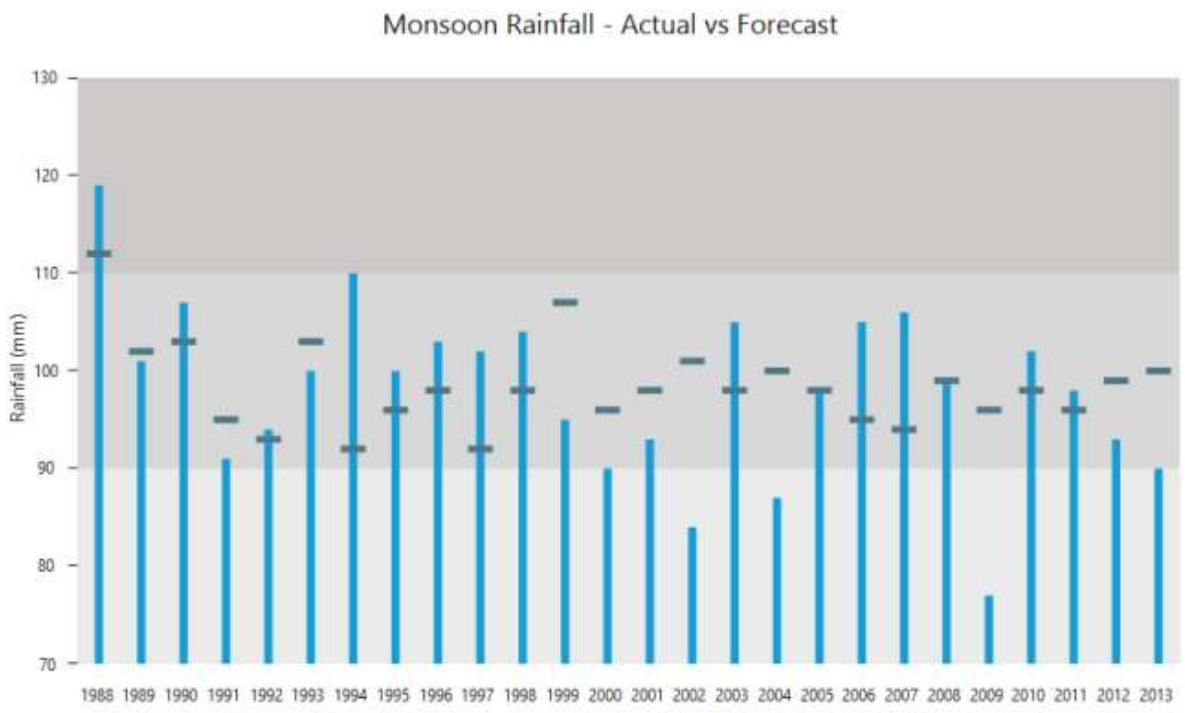
This section explains briefly about how to create a **BulletGraph** in your ASP.NET MVC application.

Create your first BulletGraph in MVC

This section encompasses the details on how to configure the **BulletGraph** control in your application. It also allows you to learn how to pass the required data to it and customize its various options according to your requirements.

Copyright © 2001 - 2020 Syncfusion Inc.

In the following screenshot, a **BulletGraph** is used to compare the actual monsoon rainfall received in a state versus its forecasted values for the years ranging from 1988 to 2013.



Bullet Graph

1. Create a

   tag.

HTML

```
<div>
</div>
```

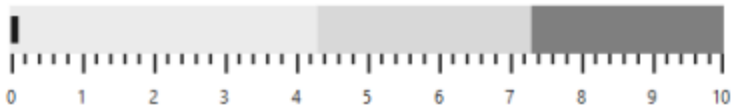
2. Add the following code in the index.cshtml file to create the BulletGraph control in the View page.

JAVASCRIPT

```
<div>
@(Html.EJ().BulletGraph("Bullet").Render())
</div>
```

3. Execute the above code to display the BulletGraph. To customize the measure bars in the BulletGraph, you can pass the data either locally or remotely.

Copyright © 2001 - 2020 Syncfusion Inc.



BulletGraph

Provide Required Data

You can customize the values of feature and comparative measure bars in a **BulletGraph**, either locally or remotely. The category data is optional, and it is used to display label values parallel to the measure bars.

Assign the data in BulletLocalDataBind variable to the DataSource property of BulletGraph as illustrated in the following code example.

C#

```
publicActionResult LocalDataBinding()
{
localDataBind.Add(newBulletLocalDataBind { category = "2013", value = "90",
comparativeMeasureValue = "100" });
localDataBind.Add(newBulletLocalDataBind { category = "2012", value = "93",
comparativeMeasureValue = "99" });
localDataBind.Add(newBulletLocalDataBind { category = "2011", value = "98",
comparativeMeasureValue = "96" });
localDataBind.Add(newBulletLocalDataBind { category = "2010", value = "102",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "2009", value = "77",
comparativeMeasureValue = "96" });
localDataBind.Add(newBulletLocalDataBind { category = "2008", value = "99",
comparativeMeasureValue = "99" });
localDataBind.Add(newBulletLocalDataBind { category = "2007", value = "106",
comparativeMeasureValue = "94" });
localDataBind.Add(newBulletLocalDataBind { category = "2006", value = "105",
comparativeMeasureValue = "95" });
localDataBind.Add(newBulletLocalDataBind { category = "2005", value = "98",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "2004", value = "87",
comparativeMeasureValue = "100" });
localDataBind.Add(newBulletLocalDataBind { category = "2003", value = "105",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "2002", value = "84",
comparativeMeasureValue = "100" });
localDataBind.Add(newBulletLocalDataBind { category = "2001", value = "93",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "2000", value = "96",
comparativeMeasureValue = "101" });
localDataBind.Add(newBulletLocalDataBind { category = "1999", value = "107",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "1998", value = "98",
comparativeMeasureValue = "101" });
localDataBind.Add(newBulletLocalDataBind { category = "1997", value = "92",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "1996", value = "98",
comparativeMeasureValue = "96" });
localDataBind.Add(newBulletLocalDataBind { category = "1995", value = "96",
comparativeMeasureValue = "107" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
localDataBind.Add(newBulletLocalDataBind { category = "1994", value = "92",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "1993", value = "103",
comparativeMeasureValue = "92" });
localDataBind.Add(newBulletLocalDataBind { category = "1993", value = "93",
comparativeMeasureValue = "98" });
localDataBind.Add(newBulletLocalDataBind { category = "1991", value = "95",
comparativeMeasureValue = "96" });
localDataBind.Add(newBulletLocalDataBind { category = "1990", value = "103",
comparativeMeasureValue = "92" });
localDataBind.Add(newBulletLocalDataBind { category = "1989", value = "102",
comparativeMeasureValue = "103" });
localDataBind.Add(newBulletLocalDataBind { category = "1988", value = "112",
comparativeMeasureValue = "93" });
ViewBag.datasource = localDataBind;
return View();
}
```

Once the **DataSource** property is assigned with the required values, you can bind the variable names used in the **JSON** data to the corresponding fields of the **BulletGraph** as shown in the following code example.

ASPX-CS

```
@(Html.EJ().BulletGraph("Bullet")
.Fields(fie =>
{
fie.Datasource((IEnumerable<BulletLocalDataBind>)ViewBag.datasource)
.Category("category").ComparativeMeasure("comparativeMeasureValue")
.FeatureMeasure("value");
})
.Render())
```

Set Default and Scale Values

You can plot any number of measure bars within the **BulletGraph** by increasing the height and width of the control to locate all the measure bars within the graph. Set the QualitativeRangeSize and QuantitativeScaleLength properties according to the following code example.

By default, the **BulletGraph** is rendered in the horizontal orientation with its flow direction set to **Forward**. In this example, to achieve the desired output, change the horizontal orientation to vertical orientation with the flow direction set to **Backward**.

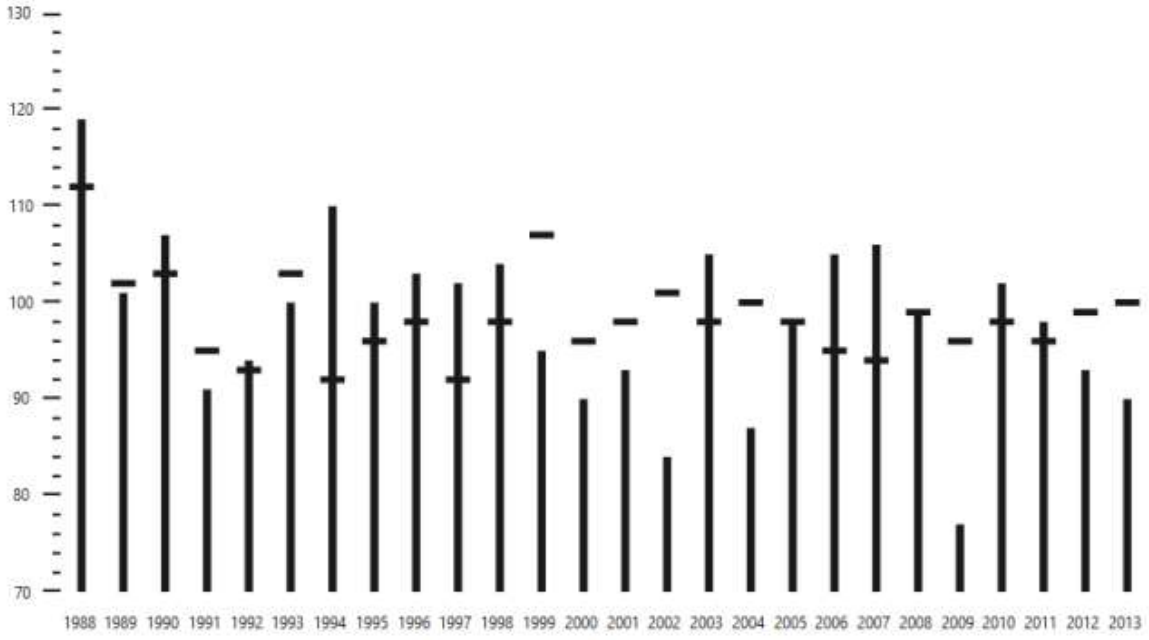
Minimum, Maximum and Interval values for the QuantitativeScale of the BulletGraph are set, as illustrated in the following code example. The ticks and labels within the scale are also positioned.

JS

```
@(Html.EJ().BulletGraph("Bullet").Width(850).Height(540)
.QualitativeRangeSize(800).QuantitativeScaleLength(425)
.Orientation(Orientation.Vertical).FlowDirection(FlowDirection.Backward)
.QuantitativeScaleSettings(scale=>scale.Minimum(70).Maximum(130).Interval(10
)
.TickPosition(TickPosition.Above)
.LabelSettings(lb=>lb.Position(LabelPosition.Above)))
.Fields(fie =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
fie.Datasource((IEnumerable<BulletLocalDataBind>)ViewBag.datasource)
.Category("category").ComparativeMeasure("comparativeMeasureValue")
.FeatureMeasure("value");
})
.Render())
```



BulletGraph

The above image illustrates the BulletGraph without any ranges displayed in the background.

Add Qualitative Ranges

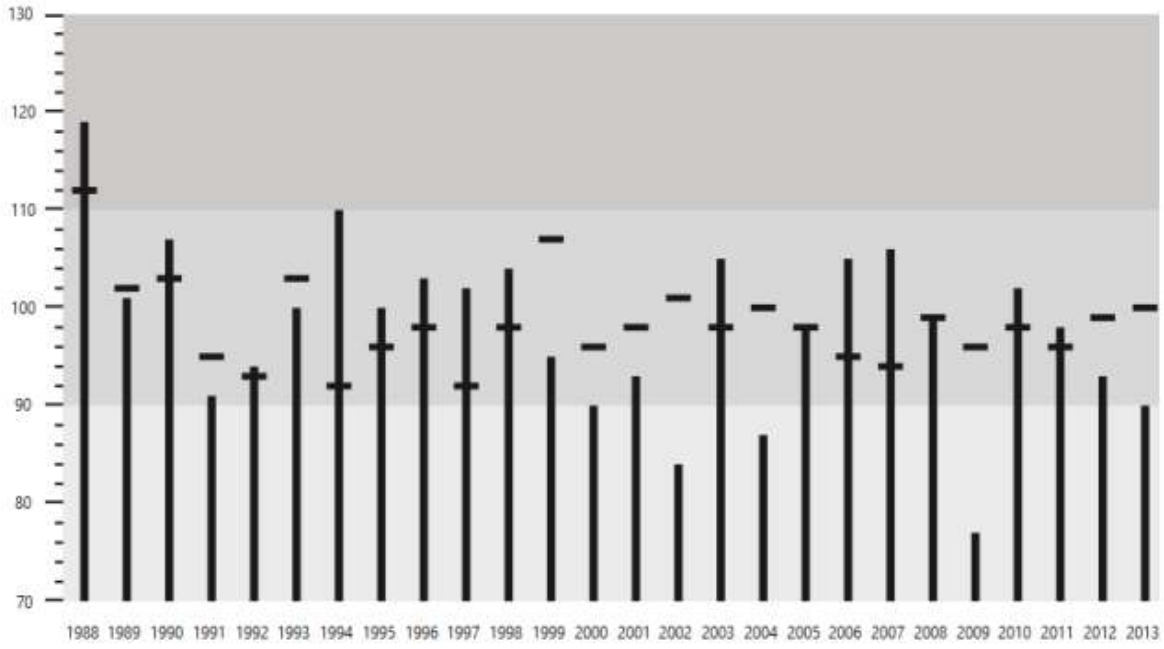
By default, 3 ranges are displayed in the **BulletGraph** control during the initial rendering of the control with its default values. To customize it, you can set appropriate values for the RangeEnd and RangeStroke properties. Any number of QualitativeRanges can be added to the control.

JS

```
@(Html.EJ().BulletGraph("Bullet").Width(850).Height(540)
.QualitativeRangeSize(800).QuantitativeScaleLength(425)
.Orientation(Orientation.Vertical).FlowDirection(FlowDirection.Backward)
.QuantitativeScaleSettings(scale=>scale.Minimum(70).Maximum(130).Interval(10
)
.TickPosition(TickPosition.Above)
.LabelSetting(lb=>lb.LabelPosition(LabelPosition.Above)))
.Fields(fie =>
{
fie.Datasource((IEnumerable<BulletLocalDataBind>)ViewBag.datasource)
.Category("category").ComparativeMeasure("comparativeMeasureValue")
.FeatureMeasure("value");
})
.QualitativeRanges(range =>
```

```
{
range.RangeEnd(90).Add();
range.RangeEnd(110).Add();
range.RangeEnd(130).RangeStroke(System.Drawing.Color.DarkBlue).Add();
})
.Render())
```

After adding **QualitativeRanges** to the **BulletGraph**, the control appears as follows.



BulletGraph

Ticks and Measure Bars Customization

You can do the following code changes in the quantitative scale to customize the tick size, the color of the feature bar and the comparative measure symbols.

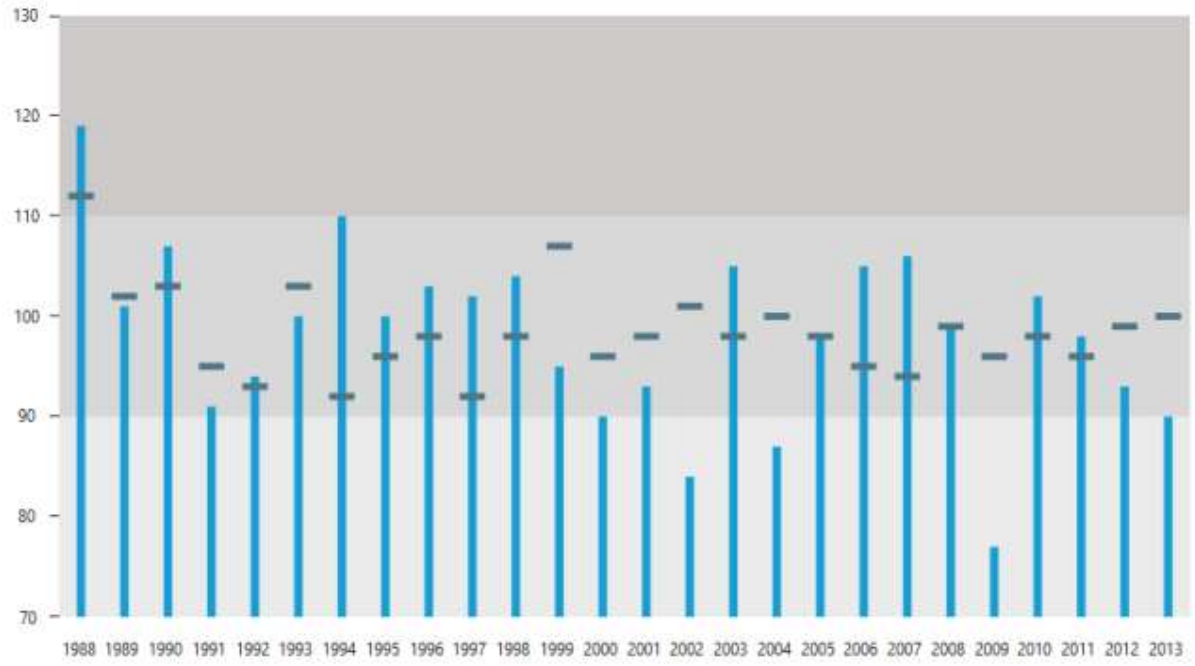
JS

```
@(Html.EJ().BulletGraph("Bullet").Width(850).Height(540)
.QualitativeRangeSize(800).QuantitativeScaleLength(425)
.Orientation(Orientation.Vertical).FlowDirection(FlowDirection.Backward)
.QuantitativeScaleSettings(scale=>scale.Minimum(70).Maximum(130).Interval(10
)
.TickPosition(TickPosition.Above)
.LabelSettings(lb=>lb.Position(LabelPosition.Above))
.MinorTickSettings(tick => tick. Width(1))
.MajorTickSettings(mat=> mat.Size(7).Width(1))
.ComparativeMeasureSettings(cm=>cm.Color(System.Drawing.Color.Blue))
.FeaturedMeasureSettings(measure=>measure.Color(System.Drawing.Color.CadetBl
ue)))
.Fields(fie =>
{
fie.Datasource((IEnumerable<BulletLocalDataBind>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Category("category").ComparativeMeasure("comparativeMeasureValue")
.FeatureMeasure("value");
})
.QualitativeRanges(range =>
{
range.RangeEnd(90).Add();
range.RangeEnd(110).Add();
range.RangeEnd(130).RangeStroke(System.Drawing.Color.DarkBlue).Add();
})
.Render())
```

When you customize the ticks and measure bar, the **BulletGraph** appears as follows.



BulletGraph

Add Caption and Subtitle

You can add the following code example to display an appropriate Caption and Subtitle to the **BulletGraph**.

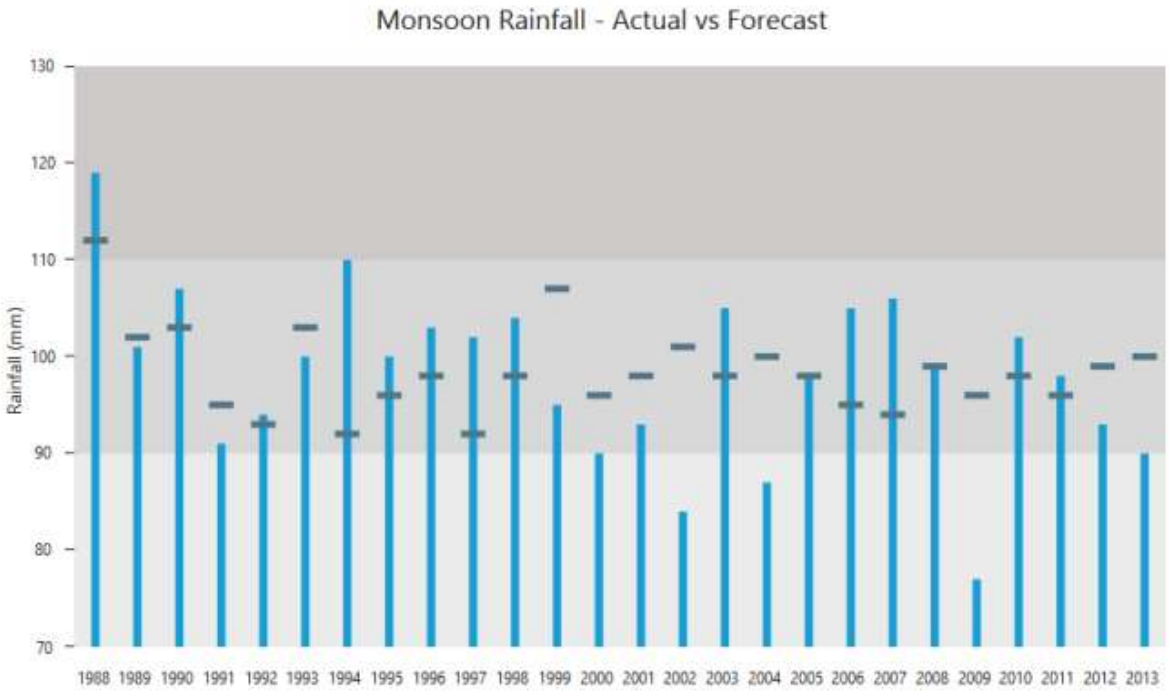
JS

```
@(Html.EJ().BulletGraph("Bullet").Width(850).Height(540)
.QualitativeRangeSize(800).QuantitativeScaleLength(425)
.Orientation(Orientation.Vertical).FlowDirection(FlowDirection.Backward)
.QuantitativeScaleSettings(scale=>scale.Minimum(70).Maximum(130).Interval(10
)
.TickPosition(TickPosition.Above)
.LabelSettings(lb=>lb.Position(LabelPosition.Above))
.MinorTickSettings(tick => tick.Width(1))
.MajorTickSettings(mat=> mat.Size(7).Width(1))
.ComparativeMeasureSettings(cm=>cm.Color(System.Drawing.Color.Blue))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.FeaturedMeasureSettings(measure=>measure.Color(System.Drawing.Color.CadetBl
ue)))
.Fields(fie =>
{
fie.Datasource((IEnumerable<BulletLocalDataBind>)ViewBag.datasource)
.Category("category").ComparativeMeasure("comparativeMeasureValue")
.FeatureMeasure("value");
})
.QualitativeRanges(range =>
{
range.RangeEnd(90).Add();
range.RangeEnd(110).Add();
range.RangeEnd(130).RangeStroke(System.Drawing.Color.DarkBlue).Add();
})
.CaptionSettings(ca=>ca.TextAngle(90).Location(location=>location.x(470).y(2
70))
.Text("Monsoon Rainfall - Actual vs Forecast")
.Font(font=>font.FontFamily("Segoe UI")
.FontSize("20px").FontWeight("regular").Opacity(1))
.SubTitle(title=>title.TextAngle(0).Text("Rainfall (mm)")
.Location(location=>location.x(180).y(4))
.Font(font=>font.FontFamily("Segoe UI")
.FontSize("14px").FontWeight("regular").Opacity(1))))
.Render())
```

The following screenshot displays a **BulletGraph** with a Caption and Subtitle.



BulletGraph with Caption and Subtitle

Show Tooltip

You can use a Tooltip in your application to display the values of forecasted rainfall, actual rainfall received in millimeter and also the appropriate year. The Tooltip Visible property is set to **true** to enable the Tooltip option. To set the template Tooltip, you can pass the template id to it as illustrated in the following code example.

JS

```
@(Html.EJ().BulletGraph("Bullet").Width(850).Height(540)
.TooltipSettings(t=>t.Visible(true).template("Tooltip"))
.QualitativeRangeSize(800).QuantitativeScaleLength(425)
.Orientation(Orientation.Vertical).FlowDirection(FlowDirection.Backward)
.QuantitativeScaleSettings(scale=>scale.Minimum(70).Maximum(130).Interval(10
)
.TickPosition(TickPosition.Above)
.LabelSettings(lb=>lb.Position(LabelPosition.Above))
.MinorTickSettings(tick => tick.Width(1))
.MajorTickSettings(mat=> mat.Size(7).Width(1))
.ComparativeMeasureSettings(cm=>cm.Color(System.Drawing.Color.Blue))
.FeaturedMeasureSettings(measure=>measure.Color(System.Drawing.Color.CadetBl
ue)))
.Fields(fie =>
{
fie.Datasource((IEnumerable<BulletLocalDataBind>)ViewBag.datasource)
.Category("category").ComparativeMeasure("comparativeMeasureValue")
.FeatureMeasure("value");
})
.QualitativeRanges(range =>
{
range.RangeEnd(90).Add();
range.RangeEnd(110).Add();
range.RangeEnd(130).RangeStroke(System.Drawing.Color.DarkBlue).Add();
})
.CaptionSettings(ca=>ca.TextAngle(90).Location(location=>location.x(470).y(2
70))
.Text("Monsoon Rainfall - Actual vs Forecast")
.Font(font=>font.FontFamily("Segoe UI")
.FontSize("20px").FontWeight("regular").Opacity(1))
.SubTitle(title=>title.TextAngle(0).Text("Rainfall (mm)")
.Location(location=>location.x(180).y(4))
.Font(font=>font.FontFamily("Segoe UI")
.FontSize("14px").FontWeight("regular").Opacity(1))))
.Render())
```

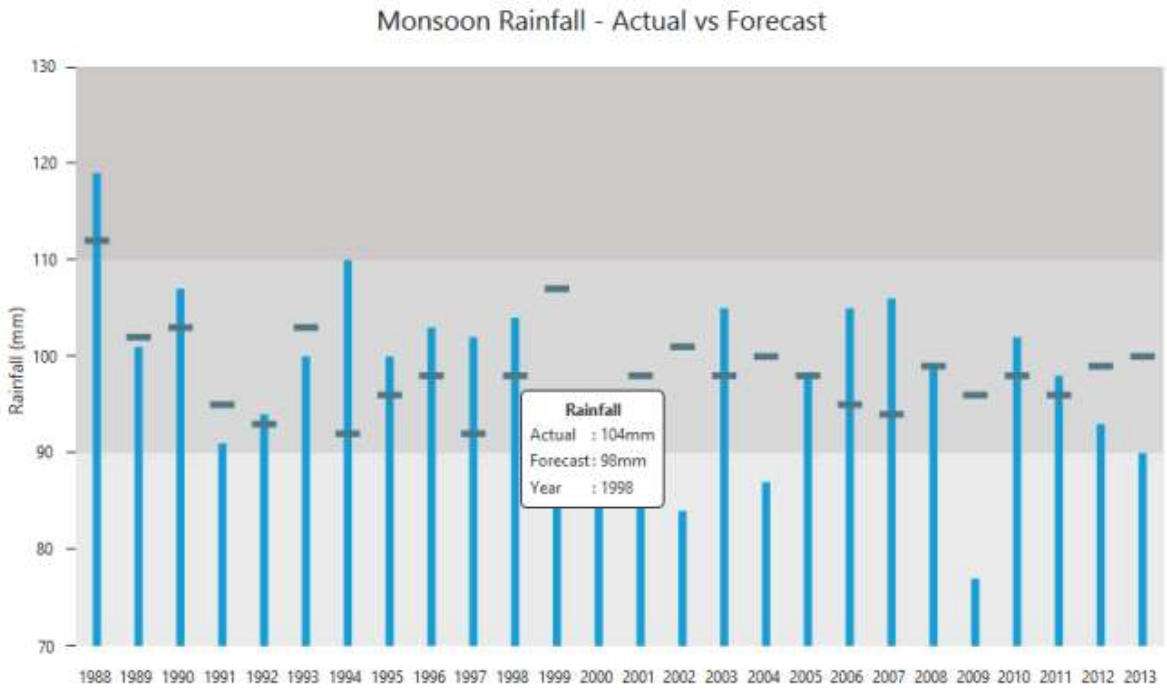
XML

```
<divid="Tooltip"style="display:none; width:125px;padding-top: 10px;padding-
bottom:10px">
<divalign="center"style="font-weight:bold">
Rainfall </div>
<table>
<tr><td>Actual</td>
<td>: {{:currentValue}}mm</td></tr>
<tr><td>Forecast</td>
<td>: {{:targetValue}}mm</td></tr>
<tr><td>Year</td>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<td>: {{:category}}</td></tr>
</table>
</div>
```

The following screenshot displays a customized BulletGraph.

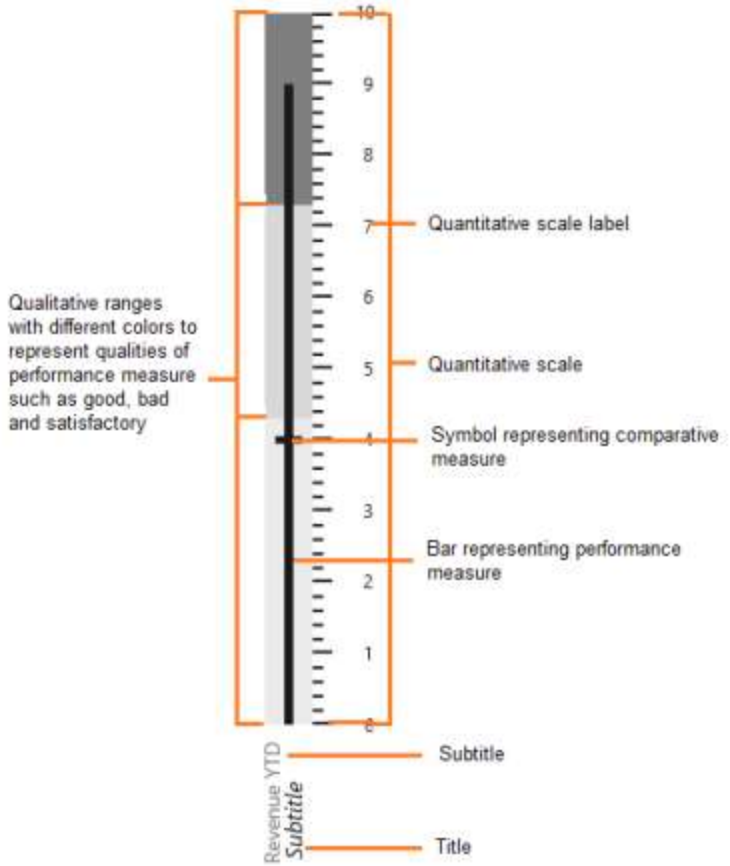


Customized BulletGraph

Bullet Graph Elements

Bullet Graph is composed of elements such as title, quantitative scale, qualitative range, etc. The following screenshot illustrates the element of BulletGraph

Copyright © 2001 - 2020 Syncfusion Inc.



Elements of the Bullet Graph

Bullet Graph Dimensions

This section explains you on how to change the dimensions of the **Bullet Graph**. You can change various dimensions and properties of Bullet Graph like width, height, quantitative scale length, qualitative range size etc. By default, Bullet Graph uses 595 pixel width and 90 pixel height. You can customize width and height of a Bullet Graph using Width and Height properties of Bullet Graph respectively.

Size

JS

```
@(Html.EJ().BulletGraph("Bullets").Width(500).Height(100))
```

In the above code example, width is set as 500 pixel and height is set as 100 pixel. The output of the above code example with dimension 500 * 100 is as follows.



Bullet Graph

Copyright © 2001 - 2020 Syncfusion Inc.

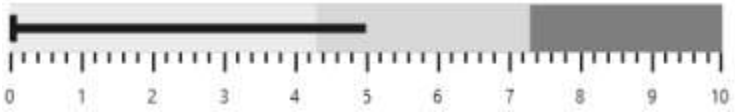
Value for performance bar

The feature measure bar value is customized using the Value property. Default value of this property is 0.

JS

```
@(Html.EJ().BulletGraph("Bullets").Value(5))
```

The following screenshot displays **Bullet Graph** with a performance measure value of 5



Bullet Graph with performance measure

Comparative measure value

The **Comparative measure value** is set using ComparativeMeasureValue property. The default value of this property is 0.

JS

```
@(Html.EJ().BulletGraph("Bullets").ComparativeMeasureValue(5))
```

The following screenshot displays **Bullet Graph** with comparative measure value of 5

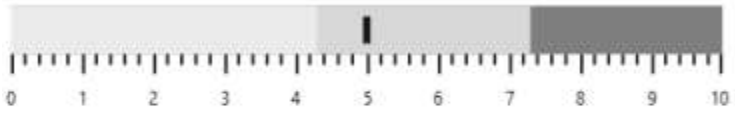


Figure: Bullet Graph with comparative measue

Bullet Graph with comparative symbol

Theme

Bullet Graph Theme is customized using Theme property. Default value is flatlight. **Bullet Graph** supports flatlight and flatdark themes. Flatdark theme improves **Bullet Graph** appearance when background of **Bullet Graph** container uses dark color like black.

JS

```
@(Html.EJ().BulletGraph("Bullets").Theme("flatdark”))
```

The following screenshot displays **Bullet Graph** with flatdark theme

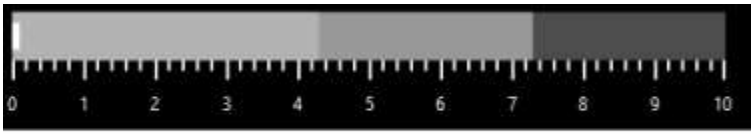


Figure: Bullet graph with flat dark theme

Bullet Graph with theme

Copyright © 2001 - 2020 Syncfusion Inc.

Orientation

Bullet Graph is oriented either horizontally or vertically using Orientation property. Default value of this property is Horizontal.

JS

```
@(Html.EJ().BulletGraph("Bullets").Orientation(Orientation.Vertical)
.Width(100)
.Height(550).FlowDirection(FlowDirection.Backward))
```

Flow direction

The Flow direction of Bullet Graph is customized using FlowDirection property. Default value of this property is Forward. Setting Forward renders Bullet Graph left to right and Backward renders from right to left.

JS

```
@(Html.EJ().BulletGraph("Bullets").ComparativeMeasureValue(2).FlowDirection(
FlowDirection.Backward))
```

The following screenshot displays **Bullet Graph** in a backward direction.

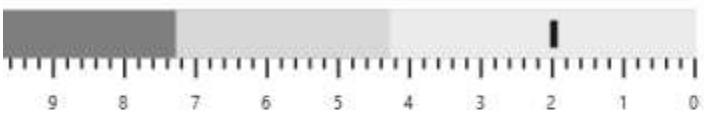


Figure: Bullet graph with backward flow (Right-to-left)

Bullet Graph in backward flow

Qualitative range size

Size of the qualitative range is customized using QualitativeRangeSize property. Default value of this property is 32.

JS

```
@(Html.EJ().BulletGraph("Bullets").QualitativeRangeSize(50))
```

The following screenshot displays **Bullet Graph** with Qualitative range of size 50

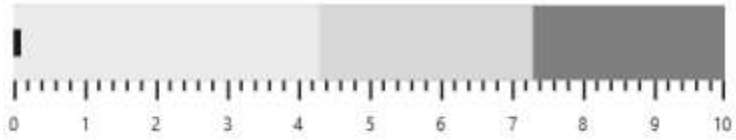


Figure: Bullet graph with qualitative range size of 50

Bullet Graph with customized qualitative range height

Quantitative scale length

Length of the quantitative scale is customized using QuantitativeScaleLength property. Default value of this property is 475.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleLength(500))
```

The following screenshot displays **Bullet Graph** with Quantitative scale length of 500

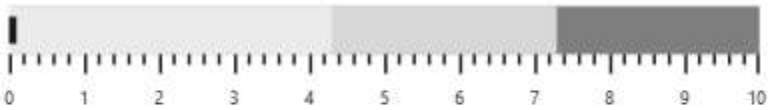


Figure: Bullet graph with quantitative scale length of 500

Bullet Graph with quantitative scale of length 500 pixels

Bullet Graph Caption

Bullet Graph supports title and subtitle to convey what is represented in **Bullet Graph**. They are customized using CaptionSettings property.

Title

Title is set to **Bullet Graph** using Text property in CaptionSettings. Caption settings also include properties like Location, Font, and TextAngle for customizing the caption of Bullet Graph.

Location

Using location option, you can set the X and Y position of caption text.

Font

Using font property, you can customize font color, font family, font style, font weight, opacity, size options.

ASPX-CS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.Location( loc=> loc.x(110).y(200))
.Minimum(0)
.Maximum(5)
.Interval(1)
.FeatureMeasure(measure=>measure.Value(4).ComparativeMeasureValue(3.5)))
.Height(700)
.Width(600)
.CaptionSettings(cs=>
cs.Text("Revenue YTD")
.Location(loc=>loc.y(220))
.Font(font=>font.FontStyle("bold").FontColor(System.Drawing.Color.Gray).Font
Size("14px")))
)
```

The following screenshot displays a **Bullet Graph** with customized caption using the above code

Copyright © 2001 - 2020 Syncfusion Inc.



Figure: Bullet graph with title

Bullet Graph with title

Subtitle

Subtitle is added to **Bullet Graph** using Text property of Subtitle in CaptionSettings. Subtitle also provides properties like Location, TextAngle and Font to customize subtitle similar to caption.

Location

Using location option, you can set the X and Y position of caption text.

Font

Using font property, you can customize font color, font family, font style, font weight, opacity, size options.

ASPX-CS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.Location( loc=> loc.x(110).y(200))
.Minimum(0)
.Maximum(5)
.Interval(1)
.FeatureMeasure(measure=>measure.Value(4).ComparativeMeasureValue(3.5)))
.Height(700)
.Width(600)
.CaptionSettings(cs=>
cs.SubTitle(subtitle=>
subtitle.Text("Subtitle")
.Location(loc=>loc.x(20).y(225))
.Font(font=>font.FontStyle("italic").FontSize("16px")))
.Text("Revenue YTD")
.Location(loc=>loc.y(210))
.Font(font=>font.FontStyle("bold").FontColor(System.Drawing.Color.Gray).Font
Size("14px")))
)
```

The following screenshot displays **Bullet Graph** with a subtitle

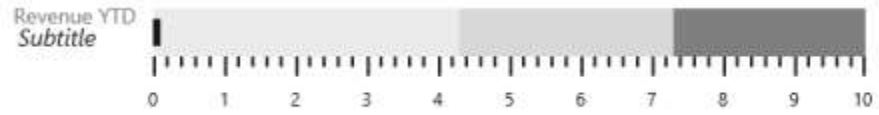


Figure: Bullet graph with subtitle

Bullet Graph with subtitle

Copyright © 2001 - 2020 Syncfusion Inc.

Indicator

You can add **Indicator** to bullet graph by enabling Visible and setting Text properties of Indicator in **CaptionSettings**. Indicator is used to represent whether target is achieved or not with text and symbol by comparing current and target values in bullet graph.

Indicator displays a symbol along with text which is different from caption and subtitle. Images like logos can be used in indicator instead of symbols. Indicator has properties such as Symbol, Text, TextSpacing, TextAngle, Location and Font.

Location

Using location option, you can set the X and Y position of caption text.

Font

Using font property, you can customize font color, font family, font style, font weight, opacity, size options.

Symbol

- Using symbol settings, you can customize the following symbol properties.
- You can customize the symbol border properties border color and border width using border

settings.

- You can specify the color of the symbol using color property.
- You can set the shape of the symbol using shape property.
- Instead of setting shape for symbol, you can use image also. To use image as symbol, you need to set image url for symbol.
- You can customize the size of the indicator symbol using width and height properties of

indicator's symbol size settings.

- To customize the opacity of the indicator symbol, you can use the opacity property of the

symbol.

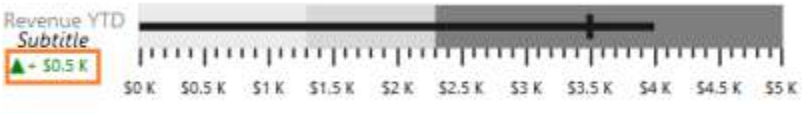
ASPX-CS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.Location( loc=> loc.x(110).y(200))
.Minimum(0)
.Maximum(5)
.Interval(1)
.LabelSettings(ls=>ls.LabelPrefix("$").LabelSuffix("K"))
.FeatureMeasure(measure=>measure.Value(4).ComparativeMeasureValue(3.5)))
.Height(700)
.Width(600)
.CaptionSettings(cs=>
cs.Indicator(indicator=>
indicator.Visible(true)
.Text("+ $0.5 K")
.Location(loc=>loc.x(15).y(240))
.Symbol(symbol=>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
symbol.Shape(SymbolShape.Triangle)
.Color("green")
.Border(border=>border.Width(1).Color("green")
)))
.SubTitle(subtitle=>
subtitle.Text("Subtitle")
.Location(loc=>loc.x(20).y(225))
.Font(font=>font.FontStyle("italic").FontSize("16px")))
.Text("Revenue YTD")
.Location(loc=>loc.y(210))
.Font(font=>font.FontStyle("bold").FontColor(System.Drawing.Color.Gray).Font
Size("14px")))
)
```

The following screenshot displays a **bullet graph** with indicator.



Bullet Graph with indicator

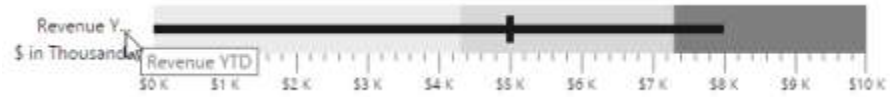
Trim

The title, subtitle and indicator text can be overlapped to the scale group. You can avoid the overlapped text by using the EnableTrim property of the CaptionSettings. The default value of the EnableTrim is true.

HTML

```
@(Html.EJ().BulletGraph("BulletGraph1")
.CaptionSettings(title=>title.Text("Bullet Graph Title").EnableTrim (true))
)
```

The following screenshot displays the BulletGraph with Trim.



BulletGraph with trim

Text Placement

All the caption group elements (caption, subtitle, and indicator) in the **Bullet Graph** support text positioning by using the property **TextPosition** available in all caption group elements. The properties, **TextAlignment** and **TextAnchor** are used to customize text placement further.

Text Position

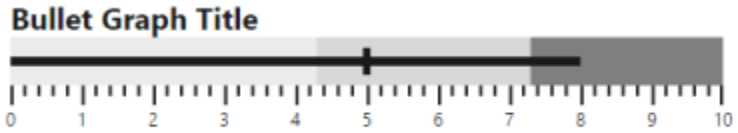
The property TextPosition, is used to position the text at the top, bottom, left, and right side of the quantitative scale. The default value of this property is **Float**. By default, text can be placed at any desired location by using the Location property.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().BulletGraph("BulletGraph1").Width(650).Height(150).Value(8).Comp
arativeMeasureValue(5)
.CaptionSettings(title=>title.Text("Bullet Graph Title")
.TextPosition(BulletTextPosition.Top)
.Font(font=>
font.FontSize("20px")
.FontWeight("bold")
))
.QuantitativeScaleSettings(scale=>scale.Location(loc=>loc.x(120).y(40)))
)
```

The following screenshot displays the **Bullet Graph** with the title positioned above.



Bullet Graph with title automatically positioned above the scale

Text Alignment

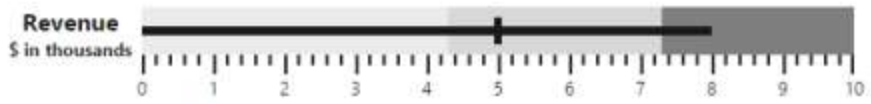
Alignment of text at different positions with respect to scale can be customized by using the TextAlignment property. Text can be aligned in the near, center, and far locations of the scale. Text alignment depends upon TextPosition property and is not applicable when the value of the TextPosition property is **Float**. The default value of the TextAlignment property is **Near**.

ASPX-CS

```
@(Html.EJ().BulletGraph("BulletGraph1").Width(650).Height(150).Value(8).Comp
arativeMeasureValue(5)
.CaptionSettings(title=>title.Text("Revenue")
.TextPosition(BulletTextPosition.Left)
.TextAnchor(BulletTextAnchor.Middle)
.Font(font=>
font.FontSize("16px")
.FontWeight("bold")
)
.SubTitle(subtitle=>subtitle.Text("$ in thousands")
.TextPosition(BulletTextPosition.Left)
.TextAlignment(BulletTextAlignment.Center)
.Font(font=>
font.FontSize("12px")
.FontWeight("bold")
)))
.QuantitativeScaleSettings(scale=>scale.Location(loc=>loc.x(120).y(40)))
)
```

The following screenshot displays the Bullet Graph with the title and subtitle at different alignments.

Copyright © 2001 - 2020 Syncfusion Inc.



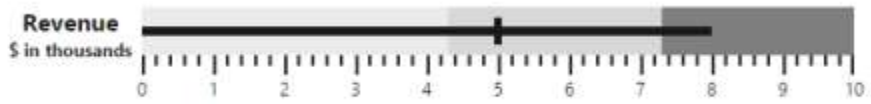
Bullet graph with the title aligned at near and sub title aligned at center positions

Text Anchor

Text elements aligned at the same position are anchored by using the TextAnchor property. These can be anchored at the Start, Middle, and End. The default value of this property is Start and applicable only when two or more text elements are aligned at the same position.

ASPX-CS

```
@(Html.EJ().BulletGraph("BulletGraph1").Width(650).Height(150).Value(8).Comp
arativeMeasureValue(5)
.CaptionSettings(title=>title.Text("Revenue")
.TextPosition(BulletTextPosition.Left)
.TextAnchor(BulletTextAnchor.Middle)
.Font(font=>
font.FontSize("16px")
.FontWeight("bold")
)
.SubTitle(subtitle=>subtitle.Text("$ in thousands")
.TextPosition(BulletTextPosition.Left)
.TextAlignment(BulletTextAlignment.Center)
.Font(font=>
font.FontSize("12px")
.FontWeight("bold")
)))
.QuantitativeScaleSettings(scale=>scale.Location(loc=>loc.x(120).y(40)))
)
```



Bullet Graph with title text anchored at the center with respect to subtitle text

Padding

The space required between text and quantitative scale is customized by using the Padding property. The default value of this property is 5 and not applicable when the value of the TextPosition property is Float.

ASPX-CS

```
@(Html.EJ().BulletGraph("BulletGraph1").Width(650).Height(150).Value(8).Comp
arativeMeasureValue(5)
.CaptionSettings(title=>title.Text("Profit in %")
.TextPosition(BulletTextPosition.Left)
.TextAlignment(BulletTextAlignment.Center)
.Padding(10)
.Font(font=>
font.FontSize("16px")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.FontWeight("bold")
))
.QuantitativeScaleSettings(scale=>scale.Location(loc=>loc.x(120).y(40)))
)
```



Bullet graph with 10 pixel padding between the title text and quantitative scale

Data Binding

Bullet Graph supports binding JSON data from a remote server or data created in client-side. You can use the Fields property to customize the data bound with **Bullet Graph**.

Local Data

Data available in client-side (local data) can be bound with **Bullet Graph** using Fields property. This property provides option to specify data source, fields representing progress measure bar value, comparative measure value and category value.

C#

```
public class BulletData
{
public double BulletValue
{ get; set; }
public double ComparisonValue
{ get; set; }
public string Category
{ get; set; }
}
List<BulletData> data = new List<BulletData>();
data.Add(new BulletData() { BulletValue=9.5, ComparisonValue=7.5,
Category="2001"  });
data.Add(new BulletData() { BulletValue = 9.5, ComparisonValue = 5, Category
= "2002" });
ViewBag.DataSource = data;
```

ASPX-CS

```
@(Html.EJ().BulletGraph("Bullets").QualitativeRangeSize(60).QuantitativeScal
eSettings( scale=>
scale.Location( loc=> loc.x(50).y(20))
.Interval(1))
.Height(120)
.Fields(field=>field.Category("Category").ComparativeMeasure("ComparisonValu
e")
.FeatureMeasure("BulletValue").Datasource(ViewBag.DataSource))
)
```

The following screenshot displays Bullet Graph with local data generated using JavaScript

Copyright © 2001 - 2020 Syncfusion Inc.

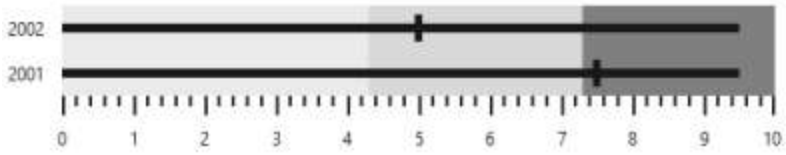


Figure: Bullet graph bound with local data

Bullet Graph bounded to JSON data

Remote Data

Bullet Graph provides option to bind data from a remote server using **ejDataManager** as data source in fields property. A query object should also be passed to query property when using data manager as data source.

JAVASCRIPT

```
@(Html.EJ().BulletGraph("Bullets").QualitativeRangeSize(60)
.QuantitativeScaleSettings(scale=>
scale.Location(loc=>loc.x(50).y(20))
.Minimum(5)
.Maximum(45)
.Interval(10))
.QualitativeRanges( qr=>
{
qr.RangeEnd(25).Add();
qr.RangeEnd(37).Add();
qr.RangeEnd(45).Add();
})
.Fields(fields=>
fields.Category("ProductID")
.ComparativeMeasure("Quantity")
.FeatureMeasure("UnitPrice"))
)
<script type="text/javascript">
//Creating data manager instance
var dataManger = new ej.DataManager({
url: "http://mvc.syncfusion.com/Services/Northwnd.svc/"
});
// Query creation
var query = ej.Query().from("Order_Details").take(3).where("UnitPrice",
ej.FilterOperators.greaterThan, 18, false)
.where("UnitPrice", ej.FilterOperators.lessThanOrEqual, 40, false)
.where("Quantity", ej.FilterOperators.greaterThan, 5, false)
.where("Quantity", ej.FilterOperators.lessThanOrEqual, 45, false);
dataManger.executeQuery(query).done(function (e) {
$("#Bullets").ejBulletGraph("option", "fields", {
dataSource: dataManger,
query: query,
});
});
</script>
```

The following screenshot displays a Bullet Graph bounded with data from a remote server

Copyright © 2001 - 2020 Syncfusion Inc.

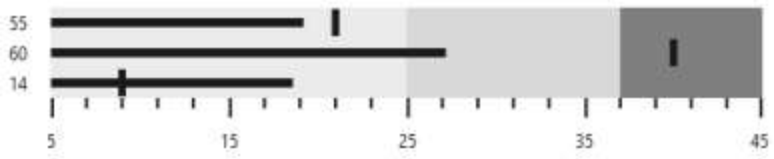


Figure: Bullet graph with data from remote server

Bullet Graph bounded to data from server

Quantitative Scale

The **Quantitative Scale** appearance is customized using QuantitativeScaleSettings property. It has properties to customize labels, major ticks, minor ticks, comparative measure and performance measure of the bullet graph

Range for Quantitative Scale

Quantitative Scale range is set using the properties Minimum, Maximum and Interval of **QuantitativeScaleSettings** property. **Minimum** specifies the start range of the scale, **Maximum** specifies the end range of scale and interval specifies the number of intervals between start and end range. Default values of Minimum, Maximum and Interval are 0, 10 and 1 respectively. The number of minor ticks (ticks between intervals) are specified using MinorTicksPerInterval property.

HTML

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.Minimum(0)
.Maximum(10))
)
```

The following screenshot displays a **Bullet Graph** with start range 0, end range 10 and interval 1 with 4 minor ticks per interval

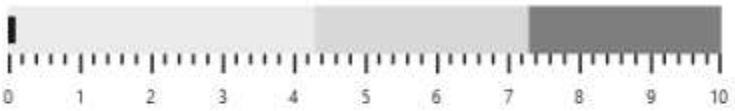


Figure: Bullet graph with default range in quantitative scale

Bullet Graph quantitative scale with customized range

Quantitative scale location

Bullet Graph does not position quantitative scale automatically based on its size or space required for caption text, etc. By default Quantitative scale is positioned at 10 pixels from left and 10 pixels from top. Quantitative scale location is customized as per the requirement using the Location property available in QuantitativeScaleSettings.

HTML

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.Location(loc=>loc.x(20).y(20))
)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays Bullet Graph with Quantitative scale at 20 pixels from left and 20 pixels from top

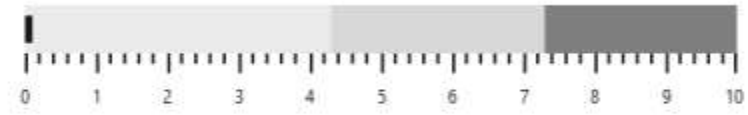


Figure: Bullet graph with customized quantitative scale location

Bullet Graph quantitative scale in a specified location

Major ticks

Color, size and width of **Major tick** lines are customized using MajorTickSettings property in **QuantitativeScaleSettings**. Default value of Size and Width properties are 13 and 2 respectively. Ticks are drawn in black color by default. The property Size represents the height of tick lines and Width represents the width of tick lines and ticks color are customized using Stroke property.

HTML

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.MajorTickSettings(mt=>
mt.Size(15).Width(3).Stroke(System.Drawing.Color.Gray))
)
)
```

The following screenshot displays **Major ticks** in **gray** color with a width of 3 pixels and height 15 pixels



Figure: Bullet graph with customized major ticks

Bullet Graph quantitative scale with customized major ticks

Minor ticks

Minor ticks can also be customized similar to major ticks. The properties Stroke, Width and Size of MinorTickSettings are used to customize Minor ticks in quantitative scale. Stroke specifies the color of ticks, Width specifies the width of ticks and Size specifies the height of the ticks.

HTML

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.MinorTickSettings(tick=>
tick.Size(7).Width(3).Stroke(System.Drawing.Color.Gray))
)
)
```

The following screenshot displays **Bullet Graph** with customized **Minor ticks** in quantitative scale

Copyright © 2001 - 2020 Syncfusion Inc.



Figure: Bullet graph with customized minor tick settings in quantitative scale

Bullet Graph quantitative scale with customized minor ticks

Tick position

Ticks are positioned below, above or inside the quantitative scale. By default ticks are positioned below the quantitative scale. The **TickPosition** property is used to customize the position of ticks in quantitative scale. Ticks can be placed inside the quantitative scale by setting **TickPosition** to **Cross**.

HTML

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.TickPosition(TickPosition.Above)
)
)
```

The following screenshot displays **Bullet Graph** with ticks positioned above quantitative scale

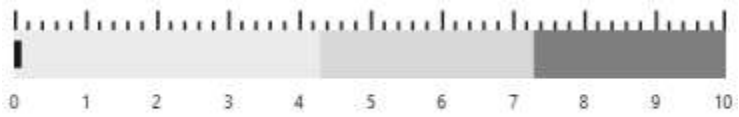


Figure: Bullet graph with ticks positioned above quantitative scale

Bullet Graph quantitative scale with ticks positioned above

Tick Placement

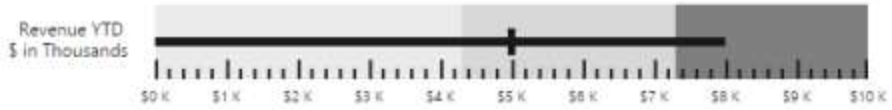
Quantitative scale ticks can be placed either inside or outside the scale using “TickPlacement” property. By default ticks are placed outside the scale.

HTML

```
@(Html.EJ().BulletGraph("Bullets") .QualitativeRangeSize(50)
.QuantitativeScaleSettings(scale =>
{
scale.Location(location => location.x(108).y(10))
.LabelSettings(ls=> ls.Offset(0).Size(10))
.TickPlacement(Syncfusion.JavaScript.DataVisualization.BulletTickPlacement.I
nside)
.FeatureMeasure(measure =>
{measure.ComparativeMeasureValue(6).Value(8).Add();});
})
.CaptionSettings(cs => cs.TextAngle(0).Text("Revenue YTD").Location(ls =>
ls.x(17).y(28))
.SubTitle(so => so.TextAngle(0).Text("$ in Thousands").Location(l =>
l.x(10).y(42))))
)
```

The following screenshot displays **Bullet Graph** ticks inside Quantitative Scale

Copyright © 2001 - 2020 Syncfusion Inc.



Bullet Graph with ticks inside quantitative scale

Quantitative scale labels

Quantitative scale labels are customized with prefix, suffix, font, color and size using **LabelSettings** property. By default, label text is displayed in black color with 12 pixel 'Segoe UI' font and there is a padding of 20 pixels space between quantitative scale and labels.

HTML

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings(scale=>
scale.LabelSettings(label=>
label.Stroke(System.Drawing.Color.Blue)
.LabelPrefix("$")
.LabelSuffix("K")
.Size(12)
.Offset(15)
.Font(font=>
font.FontFamily("Segoe UI")
.FontStyle("bold")
.FontWeight("regular")
.Opacity(0.8)
)
)
)
)
```

The following screenshot displays **Bullet Graph** labels in blue color



Figure: Bullet graph with customized labels

Bullet Graph quantitative scale with customized labels

Label Placement

Quantitative scale labels can be placed either inside or outside the scale using "LabelPlacement" property. By default labels are placed 15 pixels outside the scale.

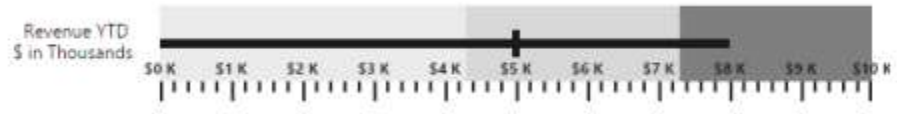
JS

```
@(Html.EJ().BulletGraph("Bullets").QualitativeRangeSize(50)
.QuantitativeScaleSettings(scale =>
{
scale.Location(loc => loc.x(108).y(10))
.LabelSettings(ls=> ls.Offset(0).Size(10)
.LabelPlacement(Syncfusion.JavaScript.DataVisualization.BulletLabelPlacement
.Inside))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.FeatureMeasure(measure =>
{measure.ComparativeMeasureValue(6).Value(8).Add();});
})
.CaptionSettings(cs => cs.TextAngle(0).Text("Revenue YTD").Location(ls =>
ls.x(17).y(28))
.SubTitle(so => so.TextAngle(0).Text("$ in Thousands").Location(l =>
l.x(10).y(42))))
)
```

The following screenshot displays **Bullet Graph** labels inside **Quantitative Scale**



Bullet Graph with labels inside quantitative scale

Performance measure bar

Performance measure bar is customized using FeaturedMeasureSettings in QuantitativeScaleSettings property. Color of the bar is customized using Stroke property and width using Width property. By default bar is drawn in black color with 6 pixels of width.

JS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.FeaturedMeasureSettings(measure=>
measure.Stroke(System.Drawing.Color.Blue).Width(4))
).Value(5))
```

The following screenshot displays **Bullet Graph** with customized **Performance measure bar**.

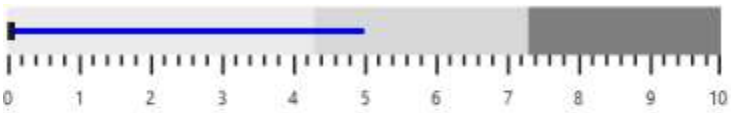


Figure: Bullet graph with customized performance measure bar

Bullet Graph quantitative scale with customized performance measure bar

Comparative measure symbol

Comparative symbol color and width are customized using ComparativeMeasureSettings through QuantitativeScaleSettings property. Color of the symbol is customized using Stroke property and width using Width property. By default Comparative measure symbol is displayed in black color with a width of 5 pixels.

JS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.ComparativeMeasureSettings(measure=>
measure.Stroke(System.Drawing.Color.Blue).Width(4))
).ComparativeMeasureValue(5)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays **Bullet Graph** with customized **Comparative measure value**.

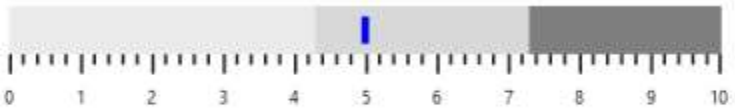


Figure: Bullet graph with customized comparative measure value

Bullet Graph with customized comparative symbol

Multiple performance measures comparison

Bullet Graph supports comparing more than one performance at a time, given that all the comparisons are related using FeatureMeasure in **QuantitativeScaleSettings** property.

JS

```
@(Html.EJ().BulletGraph("Bullets").QuantitativeScaleSettings( scale=>
scale.FeatureMeasure(measure=>
{
measure.Value(6).ComparativeMeasureValue(3).Category("2010").Add();
measure.Value(9).ComparativeMeasureValue(6).Category("2011").Add();
measure.Value(5).ComparativeMeasureValue(5).Category("2012").Add();
})
.Location(loc=>
loc.x(50).y(10))
).QualitativeRangeSize(60).Height(120)
)
```

The following screenshot displays **Bullet Graph** that compares 3 related performance measures.

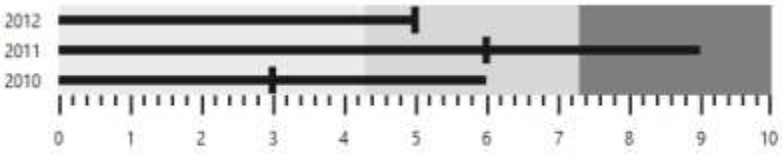


Figure: Bullet graph with multiple performance measures

Bullet Graph with multiple performance measures

Qualitative Range

Qualitative Range represents the quality of a specific range in quantitative scale like good, bad and satisfactory. Color for each qualitative range is customized using RangeStroke property. The RangeEnd property specifies the ending point of the qualitative range. Minimum value of quantitative scale is considered as the starting point of first qualitative range and previous end points are considered as starting point for other qualitative ranges.

JS

```
@(Html.EJ().BulletGraph("Bullets").QualitativeRanges(qr =>
{
qr.RangeEnd(35).RangeStroke(System.Drawing.Color.DarkRed).RangeOpacity(0.5).
Add();
qr.RangeEnd(50).RangeStroke(System.Drawing.Color.Red).RangeOpacity(1).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
qr.RangeEnd(75).RangeStroke(System.Drawing.Color.Blue).RangeOpacity(0.7).Add
();
qr.RangeEnd(90).RangeStroke(System.Drawing.Color.LightGreen).RangeOpacity(1)
.Add();
qr.RangeEnd(100).RangeStroke(System.Drawing.Color.Green).RangeOpacity(1).Add
();
}).QualitativeRangeSize(80)
.QuantitativeScaleSettings(scale =>
scale.FeatureMeasure(measure =>
{
measure.Value(55).ComparativeMeasureValue(75).Category("Year 1").Add();
measure.Value(65).ComparativeMeasureValue(70).Category("Year 2").Add();
measure.Value(80).ComparativeMeasureValue(65).Category("Year 3").Add();
})
.Location(loc =>
loc.x(50).y(20))
.Minimum(0)
.Maximum(100)
.Interval(10)
).QualitativeRangeSize(60).Height(120)
)
```

The following screenshot displays **Bullet Graph** with different qualitative ranges in different colors. In this image, range 0 to 35 represents bad performance, 35 to 50 represents average performance, 50 to 75 represents that the performance is above average, 75 to 90 represents good performance and above 90 represents excellent performance.

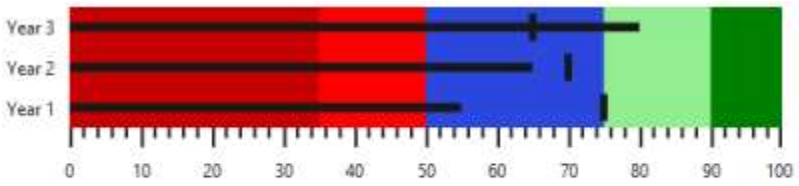


Figure: Bullet graph with different ranges to represent qualities of performance measure such as bad, average, above average, good and excellent

Bullet Graph with customized qualitative ranges

User Interaction

Animation

Bullet Graph supports animation that makes the performance measure bar to animate when rendering the **Bullet Graph**. **Animation** is enabled or disabled using EnableAnimation property. By default, **Animation** is enabled in **Bullet Graph**.

JS

```
@(Html.EJ().BulletGraph("Bullets").EnableAnimation(true).Value(8).Comparativ
eMeasureValue(5))
```

Responsiveness during browser resize

Bullet Graph is made responsive when resizing the browser by using IsResponsive property. By default the value of this property is **true** in **Bullet Graph**.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().BulletGraph("Bullets").IsResponsive(true).Value(8).ComparativeMe
asureValue(5))
```

Applying same color to all ticks and labels in a range

Background color for qualitative range is applied to major ticks and minor ticks of the **Bullet Graph** using ApplyRangeStrokeToTicks property. The range colors are applied to labels using ApplyRangeStrokeToLabels property. By default same colors are not applied to a qualitative range and its corresponding ticks or labels.

JS

```
@(Html.EJ().BulletGraph("Bullets").ApplyRangeStrokeToLabels(true).ApplyRange
StrokeToTicks(true)
.QualitativeRanges(qr =>
{
qr.RangeEnd(3.5).RangeStroke(System.Drawing.Color.DarkRed).RangeOpacity(0.5)
.Add();
qr.RangeEnd(5.0).RangeStroke(System.Drawing.Color.Red).RangeOpacity(1).Add()
;
qr.RangeEnd(7.5).RangeStroke(System.Drawing.Color.Blue).RangeOpacity(0.7).Ad
d();
qr.RangeEnd(9.0).RangeStroke(System.Drawing.Color.LightGreen).RangeOpacity(1
).Add();
qr.RangeEnd(10.0).RangeStroke(System.Drawing.Color.Green).RangeOpacity(1).Ad
d();
})
.Value(8).ComparativeMeasureValue(5))
```

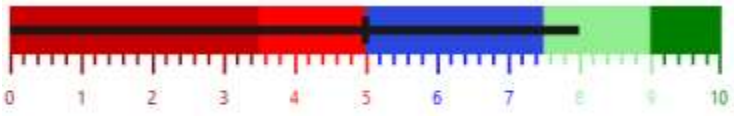


Figure: Bullet graph with same color for qualitative range, ticks and labels

Tooltip

By default **Bullet Graph** displays **Tooltip** when mouse is hovered over feature measure bar. **Tooltip** is enabled or disabled using Visible property in TooltipSettings.

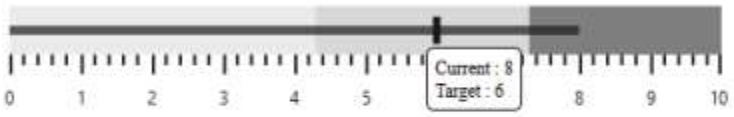


Figure: Bullet graph with default tooltip

Bullet Graph with tooltip

Bullet Graph supports Tooltip template instead of defaultTooltip to customize the appearance and contents of Tooltip. The Tooltip template should be a element with display set to ‘none’, so it is displayed only when mouse is placed on feature measure bar. The id value of the element should be provided as value to the Template property in TooltipSettings of Bullet Graph to display the

customized element as Tooltip instead of default Tooltip. The values displayed in default Tooltip such as current value, target value and category are accessed in template element by using {{currentValue}}, {{targetValue}} and {{category}} respectively.

JS

```
@(Html.EJ().BulletGraph("Bullets").Value(8).ComparativeMeasureValue(6).Height(150)
.TooltipSettings(ts=>
ts.Template("BulletGraphTooltip"))
)
```

The following screenshot displays Bullet Graph with a customized Tooltip including a header and contents such as current value and target value in different colors.

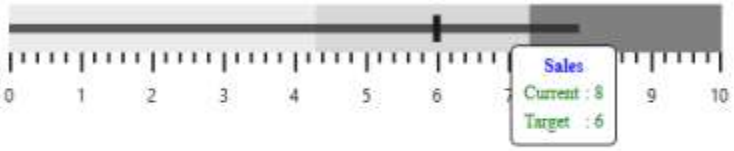


Figure: Bullet graph with customized tooltip using a template

Bullet Graph using a tooltip template

Methods

destroy ()

To destroy the bullet graph

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1"))
```

JS

```
var bullet = $("#bullet1").data("ejBulletGraph");
bullet.destroy();
```

redraw()

To redraw the bullet graph

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1"))
```

JS

```
var bullet = $("#bullet1").data("ejBulletGraph");
bullet.redraw();
```

setComparativeMeasureSymbol()

To set the value for comparative measure in bullet graph.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().BulletGraph("bullet1"))
```

JS

```
var bullet = $("#bullet1").data("ejBulletGraph");
bullet.setComparativeMeasureSymbol();
```

setFeatureMeasureBarValue()

To set the value for feature measure bar.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1"))
```

JS

```
var bullet = $("#bullet1").data("ejBulletGraph");
bullet.setFeatureMeasureBarValue();
```

Events

DrawCaption

Fires on rendering the caption of bullet graph.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawCaption("drawCaption")
)
```

JS

```
function drawCaption(){
// Do Something
}
```

DrawCategory

Fires on rendering the category.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawCategory("drawCategory")
)
```

JS

```
function drawCategory(){
// Do Something
}
```

DrawComparativeMeasureSymbol

Fires on rendering the comparative measure symbol.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawComparativeMeasureSymbol("drawComparativeMeasureSymbol")
)
```

JS

```
function drawComparativeMeasureSymbol(){
// Do Something
}
```

DrawFeatureMeasureBar

Fires on rendering the feature measure bar.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawFeatureMeasureBar("drawFeatureMeasureBar")
)
```

JS

```
function drawFeatureMeasureBar(){
// Do Something
}
```

DrawIndicator

Fires on rendering the indicator of bullet graph.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawIndicator("drawIndicator")
)
```

JS

```
function drawIndicator(){
// Do Something
}
```

DrawLabels

Fires on rendering the labels.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawLabels("drawLabels")
)
```

JS

```
function drawLabels(){
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// Do Something
}
```

DrawTicks

Fires on rendering the ticks.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawTicks("drawTicks")
)
```

JS

```
function drawTicks(){
// Do Something
}
```

DrawQualitativeRanges

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DrawQualitativeRanges("drawQualitativeRanges")
)
```

JS

```
function drawQualitativeRanges(){
// Do Something
}
```

Load

Fires on loading bullet graph.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.Load("load")
)
```

JS

```
function load(){
// Do Something
}
```

Click

Click event fires on clicking the bullet graph.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.Click("click")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

JS

```
function click(){
// Do Something
}
```

DoubleClick

DoubleClick event fires on double clicking the bullet graph.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.DoubleClick("doubleClick")
)
```

JS

```
function doubleClick(){
// Do Something
}
```

RightClick

RightClick event fires on right clicking the bullet graph.

ASPX-CS

```
@(Html.EJ().BulletGraph("bullet1")
.RightClick("rightClick")
)
```

JS

```
function rightClick(){
// Do Something
}
```

Button

Button

The Essential ASP.NET MVCButton control allows you to perform an action by clicking on it. The Button control has the feature of displaying both text and images. When the Button is clicked, it looks as if it is being pushed in and released. The Text property is used to display the Text content in Button. The Button control displays images using the sprite CSS and Image Position properties. The Buttons control has theme support. While using input type submit, reset and button to be render as button, has text only support.

Key Features

- Trendy Look: Rich Appearance with Theme Support

- RTL: Supports for Right to Left alignment
- Repeat-Button: Supports the rising of click event Repeatedly
- Text and Image: Supports both text and image as Button content
- Built-in Icon: Supports the built-in icon libraries
- Easy Customization: The customization of Button control to any form is made simple

Getting Started

This section explains you briefly on how to create a Button in your application with ASP.NET MVC.

Create your first Button in MVC

Essential ASP.NET MVC Button provides support to display a Button control within your web page and allows you to Click, Toggle Click, Reset, and Submit. Using the following guidelines, you can customize Button for a real-time Multimedia player scenario. This allows you to Play, Pause, Stop, Mute, Un-mute a music player.

The following screenshot illustrates the functionality of Button in Multimedia player control.



Multimedia Player

Create Button Control in MVC

Essential ASP.NET MVC Button control contains built-in features like Click and different display option. You can easily create the Button control by using HTML helper as follows.

1. You can create an MVC Project and add necessary assemblies, styles and scripts to it. Refer MVC-Getting Started.
2. Add the following code example to the corresponding view page to render Button.

HTML

```
<table>
<tr>
<td>
@Html.EJ().ToggleButton("playPause").Size(ButtonSize.Large).ShowRoundedCorne
r(true)
.ContentType(ContentType.TextAndImage).ToggleState(true).DefaultPrefixIcon("
e-icon e-mediaplay")
.ActivePrefixIcon("e-icon e-
mediapause").DefaultText("Play").ActiveText("Pause")
</td>
<td>
@Html.EJ().Button("mute").Size(ButtonSize.Large).ShowRoundedCorner(true).Tex
t("Mute")
</td>
<td>
@Html.EJ().Button("unmute").Size(ButtonSize.Large).ShowRoundedCorner(true).T
ext("unmute")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</td>
<td>
@Html.EJ().Button("stop").Size(ButtonSize.Large).ShowRoundedCorner(true).Tex
t("top")
</td>
</tr>
</table>
```

Create Multimedia Player

1. Add the Action Result for Audio control in HomeController.cs file.

CSHARP

```
public ActionResult MyAudio()
{
var file = Server.MapPath("../song.mp3");
return File(file, "audio/mp3");
}
```

2. Add the following code example to your view page to display Audio control and render the Button.

HTML

```
<div class="audiodiv">
<audio controls>
<source src="@Url.Action("MyAudio","Home")" type="audio/mp3" />
<p>Your browser does not support HTML 5 audio element</p>
</audio>
</div>
<div>
<table>
<tr>
<td>
@Html.EJ().ToggleButton("playPause").Size(ButtonSize.Large).ShowRoundedCorne
r(true)
.ContentType(ContentType.TextAndImage).ToggleState(true).ClientSideEvents(e
=> e.Create("play").Click("pause")
.Change("play")).DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediapause").DefaultText("Play").ActiveText("Pause")
</td>
<td>
@Html.EJ().Button("mute").Size(ButtonSize.Large).ShowRoundedCorner(true).Cli
entSideEvents(e => e.Click("mute")).Text("Mute")
</td>
<td>
@Html.EJ().Button("unmute").Size(ButtonSize.Large).ShowRoundedCorner(true).C
lientSideEvents(e => e.Click("unmute")).Text("unmute")
</td>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<td>
@Html.EJ().Button("stop").Size(ButtonSize.Large).ShowRoundedCorner(true).Cli
entSideEvents(e => e.Click("stop")).Text("stop")
</td>
</tr>
</table>
</div>
```

3. Add the following event function to script section in your view page to use Button control feature as the Multimedia player control.

JS

```
<script type="text/javascript">
//getting audio control access and stored in variable v
var v = document.getElementsByTagName("audio")[0];
//trigger the audio control using variable v
function play(e) {
if (e.isChecked) {
v.play();
}
else {
v.pause();
}
}
function start() {
v.play();
}
function stop() {
v.pause();
}
function mute()
{
v.volume = 0;
}
function unmute()
{
v.volume = 1;
}
</script>
```

4. The following screenshot displays Multimedia player control.



Multimedia Player Control

Create Office Ribbon

In this section, you can learn how to create a Microsoft Office Ribbon used to change the style of the selected text. You can achieve this by using ASP.NET MVC Toggle Button control. You can change the styles by toggling the Button. Add the following code example to your view page to display OfficeRibbon control and render the Button.

HTML

```
<table>
<tr>
<td>
@Html.EJ().ToggleButton("Bold").Size(ButtonSize.Mini).ShowRoundedCorner(true
).DefaultText("<b>B</b>").ActiveText("B").ClientSideEvents(e =>
e.Click("bold"))
</td>
<td>
@Html.EJ().ToggleButton("italic").Size(ButtonSize.Mini).ShowRoundedCorner(tr
ue).DefaultText("I").ActiveText("<li>I</li>").ClientSideEvents(e =>
e.Click("italic"))
</td>
<td>
@Html.EJ().ToggleButton("underline").Size(ButtonSize.Mini).ShowRoundedCorner
(true).DefaultText("U").ActiveText("<u>U</u>").ClientSideEvents(e =>
e.Click("underline"))
</td>
</tr>
</table>
```

1. Add the following code example in your view page for sample text.

HTML

```
<div class="sample">
<span id="sampletext">Hi Welcome!</span>
</div>
```

2. Add the following event function to the script section in your view page to use Button control feature as the Multimedia player control.

JAVASCRIPT

```
<script type="text/javascript">
function bold(e) {
if (e.isChecked) {
$(".sample span").wrap("<b></b>");
} else {
$(".sample span").unwrap("<b></b>");
}
}
function italic(e) {
if (e.isChecked) {
$(".sample span").wrap("<i></i>");
} else {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$(".sample span").unwrap("<i></i>");
}
}
function underline(e) {
if (e.isChecked) {
$(".sample span").wrap("<u></u>");
} else {
$(".sample span").unwrap("<u></u>");
}
}
</script>
```

3. The following screenshots display Office Ribbon control.



Office Ribbon before applying the styles



Office Ribbon after applying the styles

Easy Customization

Button is used in all applications. Button size, content and content location is varied according to each application. Here you can see some customizable option for button that can perform easily.

Button Size

You can render the button in different sizes. Here, you have some predefined size options for rendering a button with different sizes in easiest way. Each size option has different height and width. Mainly it avoids the complexity in rendering button with complex CSS class.

List of predefined button size

Normal	Creates button with content size.
Mini	Creates button with Built-in mini size height, width specified.
Small	Creates button with Built-in small size height, width specified.
Medium	Creates button with Built-in medium size height, with specified.
Large	Creates button with Built-in large size height, width specified.

Copyright © 2001 - 2020 Syncfusion Inc.

Apart from the above mentioned predefined size option, you can set your own width and height for button using Height and Width property.

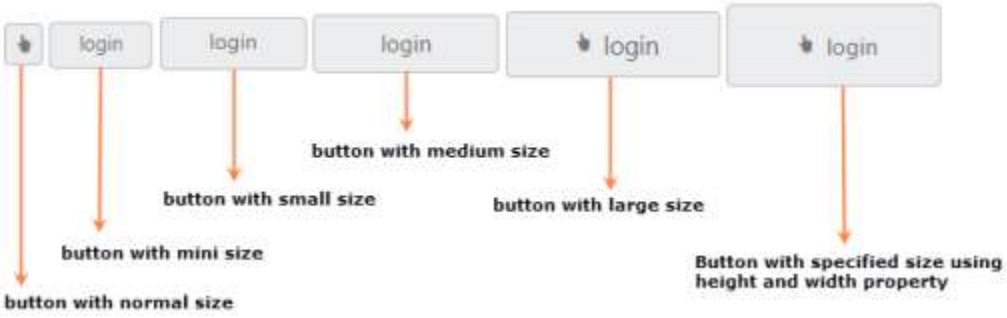
The following steps explains you the details about rendering the button with different size options.

1. In the CSHTML page, configure the button widget as follows.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@*Set the different size types for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_normal").Text("login").Size(ButtonSize.Normal).Sho
wRoundedCorner(true).ContentType(ContentType.ImageOnly).PrefixIcon("e-icon
e-handup")
@Html.EJ().Button("button_mini").Text("login").ShowRoundedCorner(true).Size(
ButtonSize.Mini)
@Html.EJ().Button("button_small").Text("login").ShowRoundedCorner(true).Size
(ButtonSize.Small)
@Html.EJ().Button("button_medium").Text("login").ShowRoundedCorner(true).Siz
e(ButtonSize.Medium)
@Html.EJ().Button("button_large").Text("login").ShowRoundedCorner(true).Size
(ButtonSize.Large).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon
e-handup")
@Html.EJ().Button("button_customSize").Text("login").ShowRoundedCorner(true)
.ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-
handup").Height("50").Width("150")
</div>
```

Execute the above code to render the following output.



Button with different sizes

Content Type

The content of the Button is mainly text and images. Instead of using complex CSS classes to render Button with different content types, you can use some predefined content type options provided for button control. Using this content types you can easily add different types of content for button. Button supports the following content types.

List of content types for button

TextOnly	Supports only for text content only.
ImageOnly	Supports only for image content only

ImageBoth	Supports image for both ends of the button.
TextAndImage	Supports image with the text content.
ImageTextImage	Supports image with both ends and middle in text.

Prefix and Suffix icons

Icons inside the Button is added easily using PrefixIcon and SuffixIcon property. Location of the icon in button is a necessary thing and you can easily customize it using the following mentioned options.

Button control also supports the Built-in icon libraries. The ej.widgets.core.minCSS contains definitions for important icons that can be used in buttons. Simply you can use these Built-in icons by mentioning the icon class name as value in PrefixIcon and SuffixIcon property. You can use any font icons that are defined in ej.widgets.core.minCSS. It avoids the complexity in specifying icon using sprite image and CSS.

For example the following mentioned Built-in CSS class are used to show the font icons that is used by media player.

e-mediaback

e-mediaforward

e-medianext

e-mediaprev

e-mediaeject

e-mediaclose

e-mediapause

e-mediaplay

Prefix Icon

It inserts the icon at the starting position of button. After this prefix icon, you can use text or suffix icon.

Suffix Icon

It inserts the icon at the ending position of button. Before this suffix icon, you can use text or prefix icon.

The following steps explains you the details about rendering the Button with above mentioned content type, prefix and suffix icon options

1. In the CSHTML page, configure the Button widget as follows.

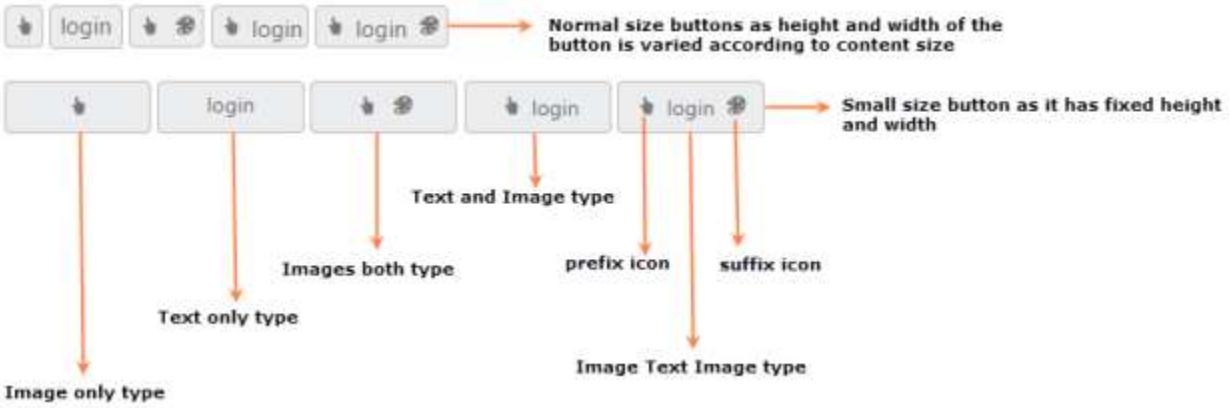
ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Set the different content types and use the necessary built-in icons for
button control as follows.*@
<div class="control">
@Html.EJ().Button("button_imageOnly").Text("login").ShowRoundedCorner(true).
ContentType(ContentType.ImageOnly).PrefixIcon("e-icon e-handup")
@Html.EJ().Button("button_textOnly").Text("login").ShowRoundedCorner(true).C
ontentType(ContentType.TextOnly)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Button("button_imageBoth").Text("login").ShowRoundedCorner(true).
ContentType(ContentType.ImageBoth).PrefixIcon("e-icon e-
handup").SuffixIcon("e-icon e-palette")
@Html.EJ().Button("button_textAndImage").Text("login").ShowRoundedCorner(tru
e).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-handup")
@Html.EJ().Button("button_imageTextImage").Text("login").ShowRoundedCorner(t
rue).ContentType(ContentType.ImageTextImage).PrefixIcon("e-icon e-
handup").SuffixIcon("e-icon e-palette")
<br />
<br />
@Html.EJ().Button("button_small_imageOnly").Text("login").Size(ButtonSize.Sm
all).ShowRoundedCorner(true).ContentType(ContentType.ImageOnly).PrefixIcon("
e-icon e-handup")
@Html.EJ().Button("button_small_textOnly").Text("login").ShowRoundedCorner(t
rue).Size(ButtonSize.Small).ContentType(ContentType.TextOnly)
@Html.EJ().Button("button_small_imageBoth").Text("login").ShowRoundedCorner(
true).Size(ButtonSize.Small).ContentType(ContentType.ImageBoth).PrefixIcon("
e-icon e-handup").SuffixIcon("e-icon e-palette")
@Html.EJ().Button("button_small_textAndImage").Text("login").ShowRoundedCorn
er(true).Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).Prefix
Icon("e-icon e-handup")
@Html.EJ().Button("button_small_imageTextImage").Text("login").ShowRoundedCo
rner(true).Size(ButtonSize.Small).ContentType(ContentType.ImageTextImage).Pr
efixIcon("e-icon e-handup").SuffixIcon("e-icon e-palette")
</div>
```

Execute the above code to render the following output.



Button with different content types

Image Position

To provide the best look and feel for Button, position of button images is an important customizable option. With ImagePosition property you can easily customize the position of images inside button without using any complex CSS. ImagePosition property is applicable only with the TextAndImageContentType property. This property supports the following values.

List of values supported by ContentType property

ImageLeft	Support for aligning text in right and image in left.
ImageRight	Support for aligning text in left and image in right.

Copyright © 2001 - 2020 Syncfusion Inc.

ImageTop	Support for aligning text in bottom and image in top.
ImageBottom	Support for aligning text in top and image in bottom.

The following steps explains you the details about rendering the Button with the above mentioned image Position options.

2. In the CSHTML page, configure the Button widget as follows.

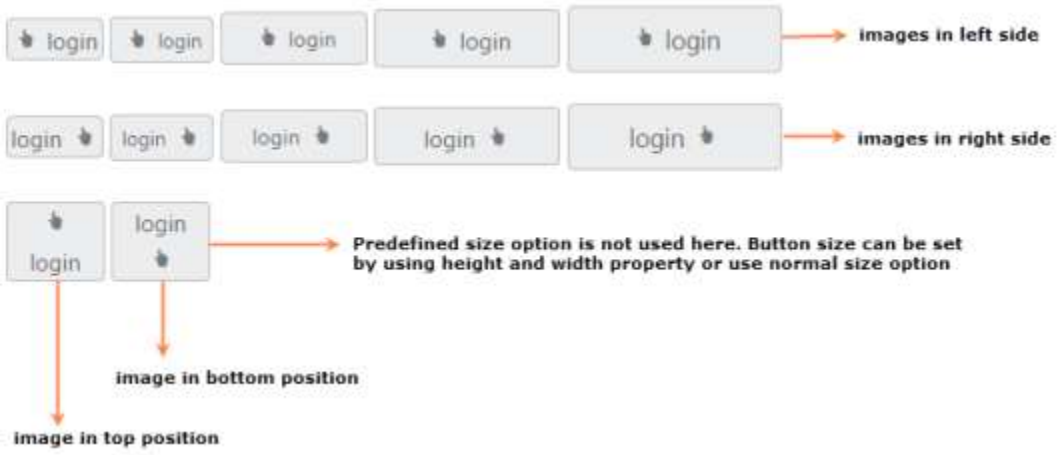
ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Set the different image position for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_normal_imageLeft").Text("login").ShowRoundedCorner
(true).Size(ButtonSize.Normal).ContentType(ContentType.TextAndImage).PrefixI
con("e-icon e-handup").ImagePosition(ImagePosition.ImageLeft)
@Html.EJ().Button("button_mini_imageLeft").Text("login").ShowRoundedCorner(t
rue).Size(ButtonSize.Mini).ContentType(ContentType.TextAndImage).PrefixIcon(
"e-icon e-handup").ImagePosition(ImagePosition.ImageLeft)
@Html.EJ().Button("button_small_imageLeft").Text("login").ShowRoundedCorner(
true).Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIco
n("e-icon e-handup").ImagePosition(ImagePosition.ImageLeft)
@Html.EJ().Button("button_medium_imageLeft").Text("login").ShowRoundedCorner
(true).Size(ButtonSize.Medium).ContentType(ContentType.TextAndImage).PrefixI
con("e-icon e-handup").ImagePosition(ImagePosition.ImageLeft)
@Html.EJ().Button("button_large_imageLeft").Text("login").ShowRoundedCorner(
true).Size(ButtonSize.Large).ContentType(ContentType.TextAndImage).PrefixIco
n("e-icon e-handup").ImagePosition(ImagePosition.ImageLeft)
<br />
<br />
@Html.EJ().Button("button_normal_imageRight").Text("login").ShowRoundedCorne
r(true).Size(ButtonSize.Normal).ContentType(ContentType.TextAndImage).Prefix
Icon("e-icon e-handup").ImagePosition(ImagePosition.ImageRight)
@Html.EJ().Button("button_mini_imageRight").Text("login").ShowRoundedCorner(
true).Size(ButtonSize.Mini).ContentType(ContentType.TextAndImage).PrefixIcon
("e-icon e-handup").ImagePosition(ImagePosition.ImageRight)
@Html.EJ().Button("button_small_imageRight").Text("login").ShowRoundedCorner
(true).Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIc
on("e-icon e-handup").ImagePosition(ImagePosition.ImageRight)
@Html.EJ().Button("button_medium_imageRight").Text("login").ShowRoundedCorne
r(true).Size(ButtonSize.Medium).ContentType(ContentType.TextAndImage).Prefix
Icon("e-icon e-handup").ImagePosition(ImagePosition.ImageRight)
@Html.EJ().Button("button_large_imageRight").Text("login").ShowRoundedCorner
(true).Size(ButtonSize.Large).ContentType(ContentType.TextAndImage).PrefixIc
on("e-icon e-handup").ImagePosition(ImagePosition.ImageRight)
<br />
<br />
@Html.EJ().Button("button_imageTop").Text("login").ShowRoundedCorner(true).C
ontentType(ContentType.TextAndImage).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageTop).Width("60")
@Html.EJ().Button("button_imageBottom").Text("login").ShowRoundedCorner(true
).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageBottom).Width("60")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
```

Execute the above code to render the following output.



Button with different image positions

Theme support

You can control the style and appearance of Button control based on CSS classes. In order to apply styles to the Button control, you can refer two files namely, ej.widgets.core.min.css and ej.theme.min.css. When you refer the ej.widgets.all.min.css file, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Button control.

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Custom CSS

You can customize the appearance of Button control using CSS class. Define a CSS class as per requirement and assign the class name to CssClass property.

The following steps explains you the details about rendering the Button with custom CSS.

3. In the CSHTML page, configure the Button widget as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Set Custom CSS class for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_customCss1").Text("login").ShowRoundedCorner(true)
.Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIcon("e-
icon e-
handup").ImagePosition(ImagePosition.ImageLeft).CssClass("customCss1")
@Html.EJ().Button("button_customCss2").Text("login").ShowRoundedCorner(true)
.Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIcon("e-
icon e-
handup").ImagePosition(ImagePosition.ImageLeft).CssClass("customCss2")
@Html.EJ().Button("button_customCss3").Text("login").ShowRoundedCorner(true)
.Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIcon("e-
icon e-
handup").ImagePosition(ImagePosition.ImageLeft).CssClass("customCss3")
@Html.EJ().Button("button_customCss4").Text("login").ShowRoundedCorner(true)
.Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIcon("e-
icon e-
handup").ImagePosition(ImagePosition.ImageLeft).CssClass("customCss4")
@Html.EJ().Button("button_customCss5").Text("login").ShowRoundedCorner(true)
.Size(ButtonSize.Small).ContentType(ContentType.TextAndImage).PrefixIcon("e-
icon e-
handup").ImagePosition(ImagePosition.ImageLeft).CssClass("customCss5")
</div>
```

4. Configure the CSS styles to apply on buttons.

CSS

```
<style type="text/css" class="cssStyles">
/* Customize the button background */
.e-button.customCss1 {
background-color: #121111;
}
.e-button.customCss2 {
background-color: #94bbd5;
}
.e-button.customCss3 {
background-color: #f3533c;
}
.e-button.customCss4 {
background-color: #d1eeed;
}
.e-button.customCss5 {
background-color: #deb66e;
}
/* Customize the button image & text color */
.e-button.customCss1.e-btn.e-select .e-icon, .e-button.customCss1.e-btn.e-
select .e-btntxt {
color: #94bbd5;
}
.e-button.customCss2.e-btn.e-select .e-icon, .e-button.customCss2.e-btn.e-
select .e-btntxt {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
color: #121111;
}
.e-button.customCss3.e-btn.e-select .e-icon, .e-button.customCss3.e-btn.e-
select .e-btntxt {
color: #cef6f7;
}
.e-button.customCss5.e-btn.e-select .e-icon, .e-button.customCss5.e-btn.e-
select .e-btntxt {
color: #534f4f;
}
</style>
```

Execute the above code to render the following output.



Button with Custom CSS

Button Type

Button is used as normal click able button, submitting form data, resetting the form data to its initial value. According to the usage of button, you can render the button in three types. Using the Type property, you can easily render the button in following types.

List of Button types

Button	The button is a click able button
Submit	The button is a submit button (submits form-data)
Reset	The button is a reset button (resets the form-data to its initial values)

The following steps explains you the details about rendering the Button with above mentioned button types.

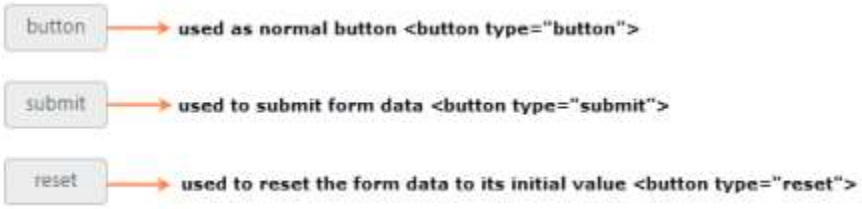
1. In the CSHTML page, configure the Button widget as follows.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Set the different types for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_button").Text("button").ShowRoundedCorner(true).Si
ze(ButtonSize.Mini).Type(ButtonType.Button)
<br />
<br />
@Html.EJ().Button("button_submit").Text("submit").ShowRoundedCorner(true).Si
ze(ButtonSize.Mini).Type(ButtonType.Submit)
<br />
<br />
@Html.EJ().Button("button_reset").Text("reset").ShowRoundedCorner(true).Size
(ButtonSize.Mini).Type(ButtonType.Reset)
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code to render the following output.



Different button types

Repeat Button

When you press button continuously, click event is raised at each specific time interval. This type of button is called Repeat Button. This functionality repeatedly raises the click event of button in both button click and from button in pressed state to the released state. TimeInterval property is used to specify the time Interval for triggering click event, when the button is in pressed state. RepeatButton property is used to set the button in repeat mode.

The following steps explains you the details about rendering the Repeat Button.

1. In the CSHTML page, configure the Button widget as follows.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Enable the button in repeat action mode and specifies time interval.*@
<div class="control">
<div class="align">
@Html.EJ().Button("repeatButton").Text("click").ShowRoundedCorner(true).Size
(ButtonSize.Mini).RepeatButton(true).TimeInterval("200").ClientSideEvents(e
=> e.Click("btnClick"))
</div>
<div class="align">
<div><b>Event Trace</b></div>
<div class="eventTrace"></div>
</div>
</div>
```

JAVASCRIPT

```
//Add this script section that is used to handle the click event
<script type="text/javascript">
function btnClick(e) {
$(".eventTrace").html("click event has been triggered..</br>" +
$(".eventTrace").html());
}
</script>
</table>
```

2. Configure the CSS styles to apply on button

CSS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<style>
.align {
display: table-cell;
padding-left: 50px;
}
</style>
```

Execute the above code to render the following output.



Output for repeat button

RTL Support

In some cases you can use right to left alignment. Here, RTL support is provided using EnableRTL property. In RTL mode when you have more than one content (image/text, image/image) in button, then these content are aligned in right to left format. For example, when text is in left and image is in right position, after applying right to left alignment these position are interchanged.

The following steps explains the details about rendering the button with Right to left alignment support.

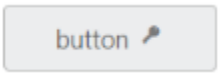
1. In the CSHTML page, configure the Button widget as follows.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Enable the alignment format for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_rtl").Text("button").ShowRoundedCorner(true).Size(
ButtonSize.Large).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon
e-login").EnableRTL(true)
</div>
```

In above mentioned code example PrefixIcon property is used and the icon that is to be on left side, (before text) is rendered on right side as EnableRTL property is used with PrefixIcon. Consequently, the alignment is changed in right to left order.

Execute the above code to render the following output.



Button in Right to left format

Icons

The Essential Studio for ASP.NET MVC provide icons library that contains the number of in-built icons that can be applied for CSS class names to elements and refer “ej.widgets.all.core.min.css” file. Use the following syntax to apply class names.

Copyright © 2001 - 2020 Syncfusion Inc.

Syntax: .e-icon .e-[icon description]

.e-icon .e-search

Adding icon in Button

For example, you can render the desired icon in the button by using the following table that contains the listed icons' CSS class names in the "PrefixIcon" property of button component. Also, use "ContentType" property to display the icon in the button. In the following code example, specify the "ContentType" of the button as ImageOnly.

Refer to the following link to know what are the values passed in the "ContentType" property

http://help.syncfusion.com/aspnetmvc/button/icons

Also in the button sample, you can use the icon class names as follows,

ASPX-CS

```
@Html.EJ().Button("button").Size(ButtonSize.Small).ContentType(ContentType.I
mageOnly).PrefixIcon("e-icon e-handup")
@Html.EJ().SplitButton("splitbutton").ContentType(ContentType.ImageOnly).Pre
fixIcon("e-icon e-calender")
@Html.EJ().ToggleButton("toggleButton").Size(ButtonSize.Small).ContentType(C
ontentType.ImageOnly).DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-mediapause")
```

Execute the above code to render the following output.



List of Icons

The complete list of icons is listed in the following table.

List of icons

e-unpin	
e-pin	
e-upload	
e-reload	
e-collapse	+
e-cancel	✖
e-expand	–

e-minimize	
e-login	
e-orientationlanscape	
e-alignleft	
e-aligncenter	
e-alignright	
e-alignjustify	
e-alignnone	
e-filterset	
e-filternone	
e-arrowheadup-2x	
e-arrowheaddown-2x	
e-arrowheadleft-2x	
e-arrowheadright-2x	
e-numbering	
e-bullets	
e-maximize	
e-delete	
e-scroll	
e-right-scroll	
e-search	
e-mediaback	
e-mediaforward	
e-medianext	
e-mediaprev	
e-mediaeject	
e-mediaclose	
e-mediapause	
e-mediaplay	

Copyright © 2001 - 2020 Syncfusion Inc.

e-righttick	
e-smile	
e-information	
e-left-arrow	
e-right-arrow	
e-file-delete	
e-file-percentage-success	
e-file-cancel	
e-file-percentage-failed	
e-file-retry	
e-resize-handle	
e-down-arrow	
e-time	
e-up-arrow	
e-date	
e-datetime	
e-collapse-arrow	
e-expand-arrow	
e-restore	
e-plus	
e-minus	
e-handup	
e-clock	
e-cursor	
e-hyperlink	
e-hyperlinkbreak	
e-settings	
e-shoppingcart	
e-palette	

Copyright © 2001 - 2020 Syncfusion Inc.

e-warningmessage	
e-cut	
e-copy	
e-paste	
e-edit	
e-swapleft	
e-swapright	
e-swapup	
e-swapdown	
e-zoomin	
e-zoomout	
e-star	
e-home	
e-clipboard	
e-userlogin	
e-dataexport	
e-arrowheadright	
e-arrowheaddown	
e-undo	
e-redo	
e-bold	
e-italic	
e-underline	
e-strikethrough	
e-font	
e-rarrowdown	
e-rarrowleft	
e-rarrowup	
e-rarrowright	

Copyright © 2001 - 2020 Syncfusion Inc.

e-calender	
e-save	

Miscellaneous

Text

You can display the user defined text for Button. Using Text property, you can easily set text content for button. This text property overwrites the text that is provided on input button element.

The following steps explains you the details about rendering the button with specified text.

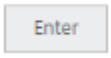
1. In the CSHTML page, configure the Button widget as follows.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Set the text for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_text").Text("Enter").Size(ButtonSize.Mini)
</div>
```

In the above code, the content of button “button” is replaced by the text value “Enter” that is given using text property.

Execute the above code to render the following output.



Button with new text

Show Rounded Corner

Specifies the corner of button in round shape. By default button doesn’t have rounded corner. To set rounded corner, you can enable ShowRoundedCorner property.

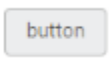
The following steps explains you the details about rendering the button with rounded corner.

2. In the CSHTML page, configure the Button widget as follows.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Enable the rounded corner for button control as follows.*@
<div class="control">
@Html.EJ().Button("button_roundedCorner").Text("button").Size(ButtonSize.Min
i).ShowRoundedCorner(true)
</div>
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Button with rounded corner

How to?

Render Button from Code behind

Button can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Button properties in the controller.

C#

```
//Namespace to get the JavaScript (Button) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class ButtonController : Controller
{
// GET: Button
public ActionResult ButtonFeatures()
{
//Initializing the Button model
ButtonProperties BtnObj = new ButtonProperties();
//Initializing the datasource and other properties
BtnObj.Text = "Button";
BtnObj.ShowRoundedCorner = true;
//Passing Button properties using the ViewData
ViewData["BtnModel"] = BtnObj;
return View();
}
}
}
```



Binding the Button properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Button("searchCustomer",(Syncfusion.JavaScript.Models.ButtonProper
ties)ViewData["BtnModel"]).Render();
}
```

Captcha

Captcha

Captcha is a popular technique that is used to prevent computer programs from sending automated requests and is used for meta-searching search engines, performing dictionary attacks in login pages, or sending spam using mail servers. Captcha is validated at the time of form submission.

Key Features

- Auto validation - It compares the input field with captcha image and thrown validation message. Also it prevents submission of forms when validation gets fail.
- Case sensitive text - It checks for case sensitive (Upper case and lower case) at the time of validation.
- Maximum length - It limits the captcha characters when exceeds the maximum limit.
- Minimum length - Sets minimum number of captcha characters to display.
- Customizable character set - You can customize the captcha characters.
- Image Refresh - When captcha is not readable you can use refresh to get new captcha.
- Audio Captcha - Captcha in the form of audio
- RTL - It provides audio and refresh button in left side of captcha

Getting Started

Create your first Captcha in ASP.NET MVC

This section enables you to configure the Captcha control in your ASP.NET MVC application and also in learning how to use captcha with auto validation in Web Forms.

Captcha control with AutoValidation

Create Captcha Control

The Captcha is one of the way to prevent dictionary attacks, it basically comes with the random text. The following steps are used to create Captcha control.

1. You can create an MVC Project and add necessary assembly and script with the given MVC-Getting Started Documentation.
2. Add the following code to the corresponding view page for Captcha rendering.

ASPX-CS

```
<div>
@Html.EJ().Captcha("SignUpCaptcha")
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

3. Add the following handler codes to the web.config file for Captcha image rendering.

HTML

```
<!--add the following code in <system.web>-->
<system.web>
<httpHandlers>
<add verb="*" path="captimage.axd" type="Syncfusion.JavaScript.ImageHandler,
Syncfusion.EJ, Version=XX.XXXX.X.XX, Culture=neutral,
PublicKeyToken=3D67ED1F87D44C89" />
</httpHandlers>
</system.web>
<!--add the following code in <system.webserver>-->
<system.webserver>
<handlers>
<add verb="*" path="captimage.axd" name="syncfusion_generatetools"
type="Syncfusion.JavaScript.ImageHandler, Syncfusion.EJ,
Version=XX.XXXX.X.XX, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</handlers>
</system.webserver>
```

Note: Version=XX.XXXX.X.X - It will vary depend up on .NET Framework version and Essential Studio version you have using. If you are using Essential Studio version as-12.3.0.36 and .NET Framework is 4.5 use like following Version=12.3450.0.36

The following screen shot displays the output of the above codes.



Captcha control with initial render

Enable Audio and Refresh

The Captcha control supports captcha in the form of audio and when you click the audio button it readouts the captcha characters. You can achieve this by setting “EnableAudio” property to true. Enable refresh is used to refresh or change the captcha image without full page refresh. This is achieved by adding “EnableRefreshImage” property to true. Also include RequestMapper for refresh support.

1. Add the following code example to view page to render captcha with Audio and Refresh.

ASPX-CS

```
<div>
@Html.EJ().Captcha("SignUpCaptcha").EnableAudio(true).EnableRefreshImage(tru
e).RequestMapper("Refresh")
</div>
```

2. Add the following code example to corresponding controller page to render captcha with Audio and Refresh.

CSHARP

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Refresh(CaptchaParams parameters)
{
return parameters.CaptchaActions();
}
```

The following screenshot displays the output of the above codes.



Captcha control with refresh and audio support

Auto Validation

The Captcha supports automatic validation by enabling the property EnableAutoValidation. When you set this property to true, captcha validation is done automatically. When the validation fails, CustomErrorMessage property supports to display the customized error message.

1. Add the following code example to view page for auto validation.

ASPX-CS

```
( CaptchaFeatures)
@using (Html.BeginForm("CaptchaFeatures", "Captcha", FormMethod.Post))
{
<div class="frame" style="width: 500px">
<div class="control" style="width: 560px;">
<table class="tableprop">
<tr>
<td>
<h2 style="text-align:center">
Sign Up</h2>
<br>
</td>
</tr>
<tr>
<td>
<span class="NodeText">Enter Your Name</span>
</td>
<td>
<input type="text" name="username" />
@if (!(bool)ViewBag.NameStatus)
{<span style="color: red">Invalid Value</span>}
</td>
</tr>
<tr>
<td>
<span class="NodeText">Email</span>
</td>
<td>
<input type="text" name="email" />
@if (!(bool)ViewBag.EmailStatus)
{<span style="color: red">Invalid Value</span>}
</td>
</tr>
```

```
<tr>
<td>
<span class="NodeText">Enter Your Password</span>
</td>
<td>
<input type="password" name="password" />
@if (!(bool)ViewBag.PasswordStatus)
{<span style="color: red">Invalid Password</span>}
</td>
</tr>
<tr>
<td>
<span class="NodeText">Re-Enter Your Password</span>
</td>
<td>
<input type="password" name="repassword" />
</td>
</tr>
<tr>
<td>
</td>
<td>
<div>
@Html.EJ().Captcha("myCaptcha").EnableRefreshImage(true).EnableAudio(true).E
nableAutoValidation(true).RequestMapper("Refresh").CustomErrorMessage("Inval
id captcha code entered. Please try again.").TargetButton("submit")
</div>
<br />
<br />
</td>
</tr>
<tr>
<td>
</td>
<td>
@Html.EJ().Button("submit").Size(ButtonSize.Large).Text("Submit").Type(Butto
nType.Submit)
</td>
</tr>
</table>
</div>
</div>
}
( CaptchaSuccess)
<div class="frame" style="height: 300px; width: 500px; border: 1px solid
gray; padding: 10px;
background: #f9f9f9">
<div class="control">
<table>
<tr>
<td style="padding: 50px;">
<br>
<br>
<h2>
Thank you for registering.</h2>
<br>
</td>
```

Copyright © 2001 - 2020 Syncfusion Inc.

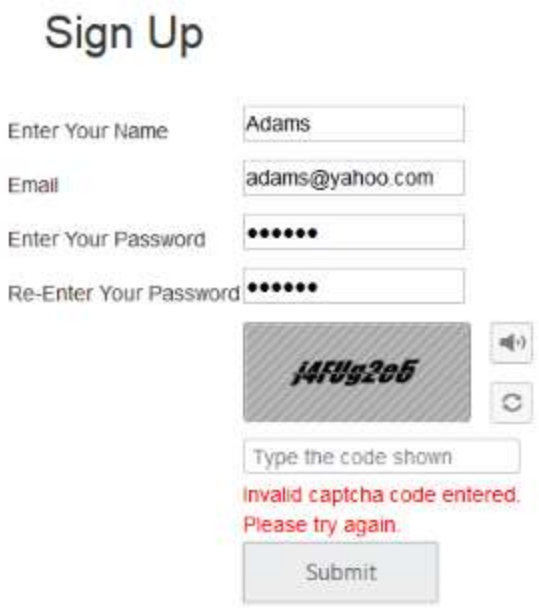
```
</tr>
</table>
</div>
</div>
```

2. Add the following code example to corresponding controller page for captcha with Auto-Validation support.

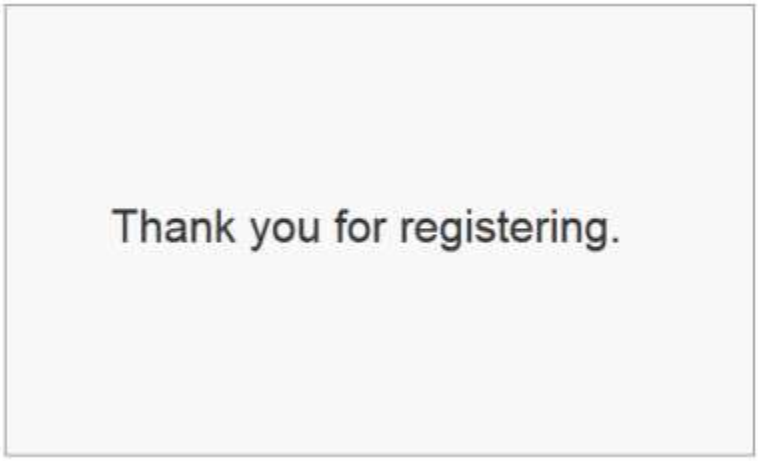
CSHARP

```
public ActionResult CaptchaFeatures()
{
ViewBag.NameStatus = true;
ViewBag.PasswordStatus = true;
ViewBag.EmailStatus = true;
return View();
}
public ActionResult CaptchaFeatures(FormCollection Values)
{
if (!string.IsNullOrEmpty(Values["username"].ToString()) &&
(Values["username"].ToString().Length > 3))
ViewBag.NameStatus = true;
else
ViewBag.NameStatus = false;
if (!string.IsNullOrEmpty(Values["email"].ToString()) &&
(Values["email"].ToString().Length > 3))
ViewBag.EmailStatus = true;
else
ViewBag.EmailStatus = false;
if ((!string.IsNullOrEmpty(Values["password"].ToString())) &&
(Values["password"].ToString().Length > 3) &&
(Values["password"].ToString().Equals(Values["repassword"].ToString())))
ViewBag.PasswordStatus = true;
else
ViewBag.PasswordStatus = false;
if ((bool)ViewBag.NameStatus && (bool)ViewBag.PasswordStatus &&
(bool)ViewBag.EmailStatus)
return View("CaptchaSuccess");
else
return View();
}
public ActionResult Refresh(CaptchaParams parameters)
{
return parameters.CaptchaActions();
}
public ActionResult CaptchaSuccess()
{
return View();
}
```

The following screenshot is the output for the above code example.



Captcha control with failed Auto Validation



Captcha control with successful AutoValidation

Behavior Settings

Regenerate Captcha

Captcha control supports regeneration of captcha image without full page refresh. You can achieve this by clicking refresh button. By default Captcha renders without refresh button. You can add the refresh button by setting EnableRefreshImage property to true.

Note: To refresh the Captcha image, include “RequestMapper”. It enables you to get or set name for the post action function.

The following code example is used to render the Captcha with Refresh support.

1. Add the following code example to the corresponding CSHTML page to render Captcha with Refresh button.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().Captcha("captcha").EnableRefreshImage(true).RequestMapper("Refres
h")
```

CSHARP

```
// Add the following code in controller page for Captcha with refresh image
public ActionResult Refresh(CaptchaParams parameters)
{
return parameters.CaptchaActions();
}
```

2. The following screenshot illustrates the Captcha with Refresh button.



Captcha with Refresh button

Audio Accessibility

Sometimes, Captcha characters are too hard to identify. In this case Captcha with audio helps to understand the Captcha character. Captcha supports captcha with audio. You can achieve this by enabling EnableAudio property to true. When this property is set true, captcha renders with audio button and when you click the audio button it readouts the captcha characters. By default Captcha renders without audio button.

The following code example is used to render the Captcha with Audio.

1. Add the following code example to the corresponding CSHTML page to render Captcha with audio button.

ASPX-CS

```
@Html.EJ().Captcha("captcha").EnableAudio(true)
```

2. The following screenshot illustrates the Captcha with audio button.



Captcha with Audio support

Enabling Validation

Auto Validation

You can validate the Captcha characters by default when you click the submit button without adding any validation code. You can achieve this by enabling EnableAutoValidation property to true. By default, this

property is set as false. When this property is set as true Captcha renders with textbox. Here, you can also customize the error message by using CustomErrorMessage property that accepts the string value.

Next, define the target button that performs validation. You can achieve this by using TargetButton property. The button id that performs the click action should be assigned in TargetButton.

Note: To validate the Captcha by default, include RequestMapper. It enables to get or set name for the post action function. The post action function is used to process the captcha values internally such as get values from client side for regeneration of captcha image and validation of the captcha.

The following code example is used to render the Captcha with Auto-Validation support.

1. Add the following code example to the corresponding CSHTML page to render Captcha with Auto-Validation support.

ASPX-CS

```
@(Html.EJ().Captcha("captcha").EnableAutoValidation(true).RequestMapper("Ref
resh").CustomErrorMessage("Invalid captcha code entered. Please try
again.").TargetButton ("submit"))<br /><br /><br />
@Html.EJ().Button("submit").Size(ButtonSize.Large).Text("Submit").Type(Butto
nType.Submit)
```

CSHARP

```
// Add the following code in controller page for Captcha with Auto-
Validation support
public ActionResult Refresh(CaptchaParams parameters)
{
return parameters.CaptchaActions();
}
```

2. The following screenshot illustrates the Captcha with Auto-Validation support.



Captcha with auto validation support

Validation by Method

Validation by method is used when EnableAutoValidation is set as false. Here, you can include TargetButton to validate Captcha. CaptchaService.IsValid() method is used to validate the Captcha. It requires three arguments namely as captcha, textbox, case sensitivity.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example is used to render the Captcha with manual validation.

1. Add the following code example to the corresponding CSHTML page to render Captcha with manual validation support.

ASPX-CS

```
@using (Html.BeginForm("Default", "Captcha", FormMethod.Post)){
@Html.EJ().Captcha("captcha").TargetButton("submit")<br />
@Html.TextBox("validateText")
@Html.ValidationMessage("myCaptcha")<br /><br />
@(Html.EJ().Button("submit").Size(ButtonSize.Large).Text("Submit")
.Type(ButtonType.Submit))}
```

CSHARP

```
// Add the following code in Controller page for Captcha with manual
validation support[HttpPost]
public ActionResult Default(FormCollection Values)
{
if (!CaptchaService.IsValid(Values["captcha"], Values["validateText "],
true))
ModelState.AddModelError("myCaptcha", "Invalid characters. Try again!");
return View();
}
```

2. The following screenshot illustrates the Captcha with manual validation support.



Captcha validation by method

Case Sensitive Validation

Captcha supports to check case sensitivity (Upper case and lower case) of the Captcha characters at the time of validation. You can achieve this by enabling EnableCaseSensitivity property to true. By default this value is set as true.

The following code example is used to render the Captcha with Case sensitive validation support.

1. Add the following code example to the corresponding CSHTML page to render Captcha with Case sensitive validation support.

ASPX-CS

```
@(Html.EJ().Captcha("captcha").EnableAutoValidation(true).RequestMapper("Ref
resh").CustomErrorMessage("Invalid captcha code entered. Please try
again.").TargetButton("submit ").EnableCaseSensitivity(true))<br /><br /><br
/>@Html.EJ().Button("submit").Size(ButtonSize.Large).Text("Submit").Type(But
tonType.Submit)
```

CSHARP

```
// Add the following code in controller page for Captcha with case sensitive
validation support
public ActionResult Refresh(CaptchaParams parameters)
{
return parameters.CaptchaActions();
}
```

2. The following screenshot illustrates the Captcha with Case sensitive validation support.



Captcha with Case sensitive validation support

Appearance and Styling

Hatch Styles

You can customize the captcha background styles by HatchStyle property.

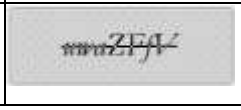
The following code example is used to render the Captcha with customized Hatch style.

1. Add the following code example to the corresponding CSHTML page to render Captcha with customized Hatch style.

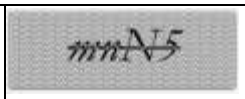
ASPX-CS

```
@Html.EJ().Captcha("captcha").HatchStyle(HatchStyle.Cross)
```

2. The following screenshot illustrates the Captcha with some of customized Hatch styles.

HatchStyle(HatchStyle.Cross)	
HatchStyle(HatchStyle.Percent90)	

Copyright © 2001 - 2020 Syncfusion Inc.

HatchStyle(HatchStyle.Wave)	
HatchStyle(HatchStyle.WideDownwardDiagonal)	
HatchStyle(HatchStyle.HorizontalBrick)	

Captcha with customized Hatch style

Pattern

You can customize the patterns of the Captcha by enabling EnablePattern property to true. When the property is set to true, Captcha characters renders with strikeout. By default the property is set as true.

The following code example is used to render the Captcha with hard visibility mode.

1. Add the following code example to the corresponding CSHTML page to render Captcha with visibility mode.

ASPX-CS

```
@Html.EJ().Captcha("captcha").EnablePattern(false)
```

2. The following screenshot illustrates the Captcha with disabled pattern.



Captcha without pattern

Background and Font color

You can customize the appearance of Captcha control by using the following property. PatternForeColor is used to set background pattern color. PatternBackColor is used to set background color for Captcha. ForeColor is used to set Captcha text color.

The following code example is used to render the Captcha with customized appearance.

1. Add the following code example to the corresponding CSHTML page to render Captcha with customized appearance.

ASPX-CS

```
@Html.EJ().Captcha("captcha").PatternForeColor(System.Drawing.Color.LightGra
y).PatternBackColor(System.Drawing.Color.Snow).ForeColor(System.Drawing.Colo
r.LightSeaGreen)
```

2. The following screenshot illustrates the Captcha with customized appearance.

Copyright © 2001 - 2020 Syncfusion Inc.



Captcha with customized appearance

Adjusting Captcha Size

Height and width Customization

The height of the Captcha widget can be customized using “Height” property that accepts only integer values. The width of the Captcha widget can be customized using “Width” property that also accepts only integer values.

The following code example is used to render the Captcha with customized Height and Width.

1. Add the following code example to the corresponding CSHTML page to render Captcha with customized Height and Width.

ASPX-CS

```
@Html.EJ().Captcha("captcha").Height(50).Width(150)
```

2. The following screenshot illustrates the Captcha with customized Height and Width.



Captcha with customized Height and Width

Theme

You can control the appearance of border color, refresh and audio button styles of the Captcha control based on CSS classes. In order to apply these styles, you can refer to the 2 files, ej.widgets.core.min.css and ej.theme.min.css. When ej.widgets.all.min.css file is referred, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Captcha control.

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Copyright © 2001 - 2020 Syncfusion Inc.

Appearance for gradient-azure,



Captcha with gradient-azure theme

Appearance for flat-lime,



Captcha with flat-lime theme

Maximum and Minimum length

You can limit the Captcha characters by MinimumLength and MaximumLength property. MaximumLength property is used to define the maximum number of characters to be displayed in the image. MinimumLength sets minimum number of captcha characters need to display.

The following code example is used to render the Captcha with Minimum and Maximum length support.

1. Add the following code example to the corresponding CSHTML page to render Captcha with Minimum and Maximum length support.

ASPX-CS

```
@Html.EJ().Captcha("captcha").MaximumLength(6).MinimumLength(5)
```

2. The following screenshot illustrates the Captcha with Minimum and Maximum length support.



Captcha with Minimum and Maximum length

Character Set Customization

The Captcha characters can be customized by CharacterSet property. Depending on this value, the Captcha characters are generated. This property accepts string values.

The following code example is used to render the Captcha with customized character set.

1. Add the following code example to the corresponding CSHTML page to render Captcha with customized character set.

ASPX-CS

```
@Html.EJ().Captcha("captcha").CharacterSet("qwertyuiop1234")
```

Copyright © 2001 - 2020 Syncfusion Inc.

2. The following screenshot illustrates the Captcha with customized character set.



Captcha with customized character set

Localization

The Captcha provides option to localize the place holder string to a particular local language. By default, the Captcha place holder text will use the US English (en-US) as its language.



Note: The culture name has to be specified in a standard format such as [Language Code]-[County/Region Code].

To localize the Captcha's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the Captcha in French language (“fr-FR”).

JAVASCRIPT

```
ej.Captcha.Locale["fr-FR"] =  {
placeHolderText: "Entrer le code indiqué",
};
```

Set the locale property of the Captcha to the new language.

HTML

```
@(Html.EJ().Captcha("captcha").EnableAutoValidation(true).RequestMapper("Ref
resh").Locale("fr-FR"))
```


RTL Support

Right to Left alignment of Captcha can be achieved by enabling EnableRTL property to true. When this property is set as true, refresh image and audio button are displayed in left side of captcha and also auto validation text box characters are entered from right to left. To achieve this, enable refresh, audio, and auto validation properties. By default RTL property value is set as false.

The following code example is used to render the Captcha with RTL support.

1. Add the following code example to the corresponding CSHTML page to render Captcha with RTL support.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Captcha("captcha").EnableRTL(true).EnableRefreshImage(true).Enab
leAudio(true).EnableAutoValidation(true))
```

2. The following screenshot illustrates the Captcha with RTL support.

Captcha with RTL support

Chart

Overview

Essential Chart for ASP.NET MVC is a visually stunning Charting component. It includes common Chart types ranging from line Charts to specialized financial Charts.

Some of the key features are,

- Supports highly interactive 28 chart types; 8 charts can be viewed in 3D view.
- Data Binding with local and remote data source.
- Supports multiple axes, and able to plot data with different data types such as numbers, date time and strings.
- Supports interactive features like zooming, panning, crosshair, trackball, tooltip and data point selection.
- Supports 10 types of technical indicators and trend lines.
- Easily customize each and every element of Chart.

Getting Started

This section explains you the steps required to populate the Chart with data, add data labels, tooltips and title to the Chart. This section covers only the minimal features that you need to know to get started with the Chart.

Installation and Deployment

It describes you on how to configure the Syncfusion dependencies in your ASP.NET MVC application to create a Chart.

Referencing Syncfusion Assemblies

The following assemblies need to be referenced in your application for using Essential Chart MVC

1. Syncfusion.EJ.dll
2. Syncfusion.EJ.MVC.dll

Follow the steps given below to deploy the application in the development server by referencing the assembly in

Copyright © 2001 - 2020 Syncfusion Inc.

```
<system.web>
  <compilation debug="true" targetFramework="4.5.1">
    <assemblies>
      <add assembly="Syncfusion.EJ, Version=13.3400.0.7, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
      <add assembly="Syncfusion.EJ.Mvc, Version=13.3400.0.7, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
    </assemblies>
  </compilation>
```

GAC.

- Web.config file should be configured according to the referenced assemblies.
- To deploy your application, you have to ensure that the above referenced assemblies (in your web.config files) are present in the GAC.

![](Getting-Started

```
<system.web.webPages.razor>
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="Syncfusion.JavaScript, Version=13.3400.0.7, Culture=neutral,
                                                PublicKeyToken=3d67ed1f87d44c89" />
      <add namespace="Syncfusion.JavaScript.DataVisualization, Version=13.3400.0.7,
                                                Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
      <add namespace="Syncfusion.MVC.EJ, Version=13.3400.0.7, Culture=neutral,
                                                PublicKeyToken=3d67ed1f87d44c89" />
    </namespaces>
  </pages>
</system.web.webPages.razor>
```

*images/Getting-Started*img1.png)

And, add Syncfusion namespace in the Web.config file of Views folder to generate Chart builder code in View page.

Note: If you are referring Syncfusion dll's manually from bin folder to create MVC Chart, then remove Culture, Version and PublicKeyToken attributes used in all, and nodes.

Adding script reference

Add the below scripts references and Syncfusion **ScriptManager** to execute the script as mentioned in the following code snippet in layout page.

ASPX-CS

```
<!--  jquery script  -->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<!-- Essential JS UI widget -->
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
<!--Add Syncfusion Script Manager -->
@Html.EJ().ScriptManager()
```

Note: If you are using the Essential Studio below 13.4.0.53 version, then you need to refer **jQuery.globalize.js** script file along with the above references to render the Chart control.

To render the MVC Chart in unobtrusive mode refer **ej.unobtrusive.js** script file. Otherwise set false to **UnobtrusiveJavaScriptEnabled** in Web.config file.

Copyright © 2001 - 2020 Syncfusion Inc.

In the above code, ej.web.all.min.js script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use CSG utility to generate a custom script file with the required widgets for deployment purpose.

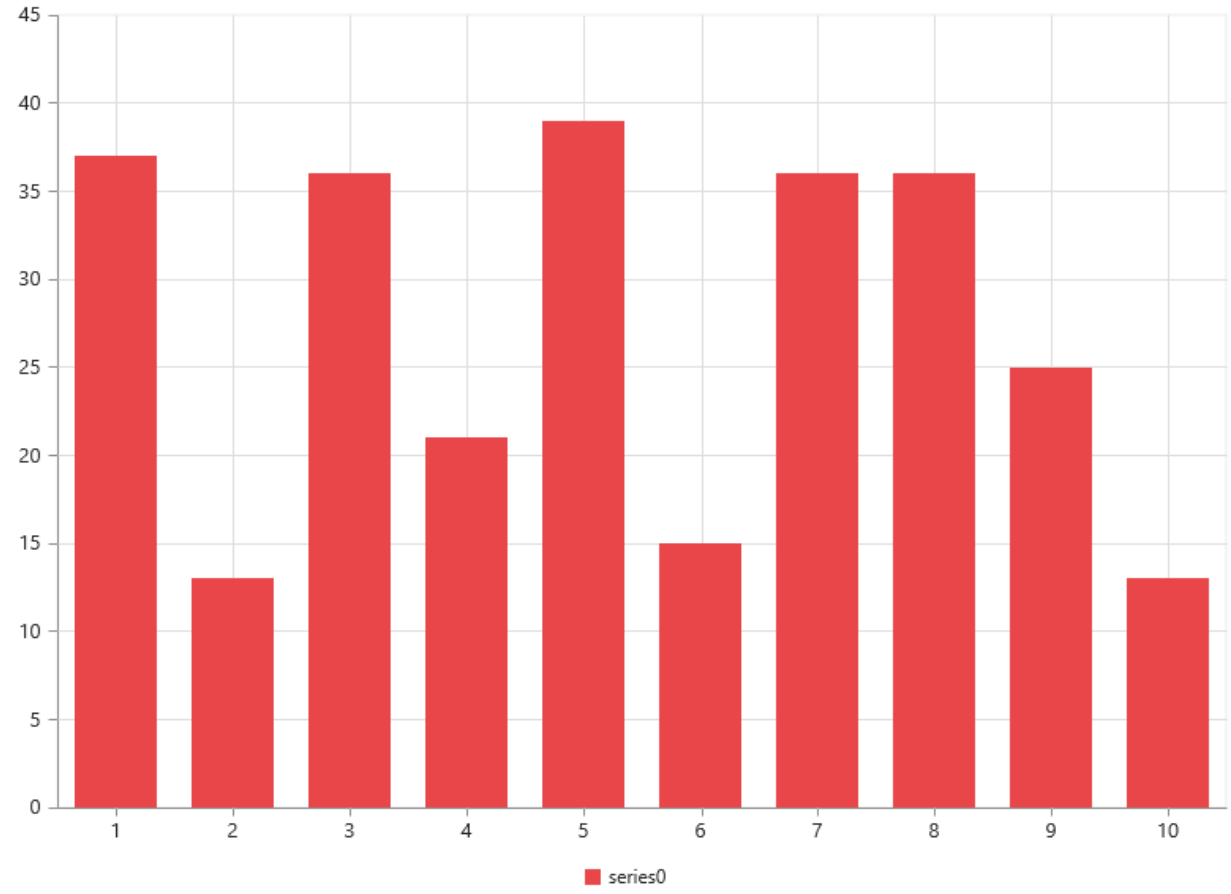
Initialize chart

Add the following code in the index.cshtml file to create the Chart control in View page.

ASPX-CS

```
<div>
@(Html.EJ().Chart("chartContainer"))
</div>
```

Now, the Chart is rendered with some auto-generated random values and with default Column chart type.



Initialize the chart by using the ejChart method. The chart is rendered to the size of its container, by default. You can also customize the chart dimension either by setting the width and height of the container element as in the above code example or by using the **Size** option of the Chart. Refer to the Chart Dimensions to know more about setting the size of the chart.

Populate chart with data

Now, let's see how to plot data source to the Chart. First, let us generate a data source containing following fields in controller– Month and Sales.

Copyright © 2001 - 2020 Syncfusion Inc.

CSHARP

```
public class HomeController : Controller
{
public ActionResult Index()
{
//// Create dataSource to chart
List<ChartData> data = new List<ChartData>();
data.Add(new ChartData("Jan", 35));
data.Add(new ChartData("Feb", 28));
data.Add(new ChartData("Mar", 34));
data.Add(new ChartData("Apr", 32));
data.Add(new ChartData("May", 40));
data.Add(new ChartData("Jun", 32));
data.Add(new ChartData("Jul", 35));
data.Add(new ChartData("Aug", 55));
data.Add(new ChartData("Sep", 38));
data.Add(new ChartData("Oct", 30));
data.Add(new ChartData("Nov", 25));
data.Add(new ChartData("Dec", 32));
///...
ViewBag.ChartData = data;
return View();
}
}
public class ChartData {
public string Month;
public double Sales;
public ChartData(string month, double sales)
{
this.Month = month;
this.Sales = sales;
}
}
```

Add a Series object to the chart using **Series** option and set the chart type as **Line** using **Type** option.

ASPX-CS

```
@using Syncfusion.JavaScript.DataVisualization
<div>
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//Change series type
sr.Type(SeriesType.Line).Add();
})
)
</div>
```

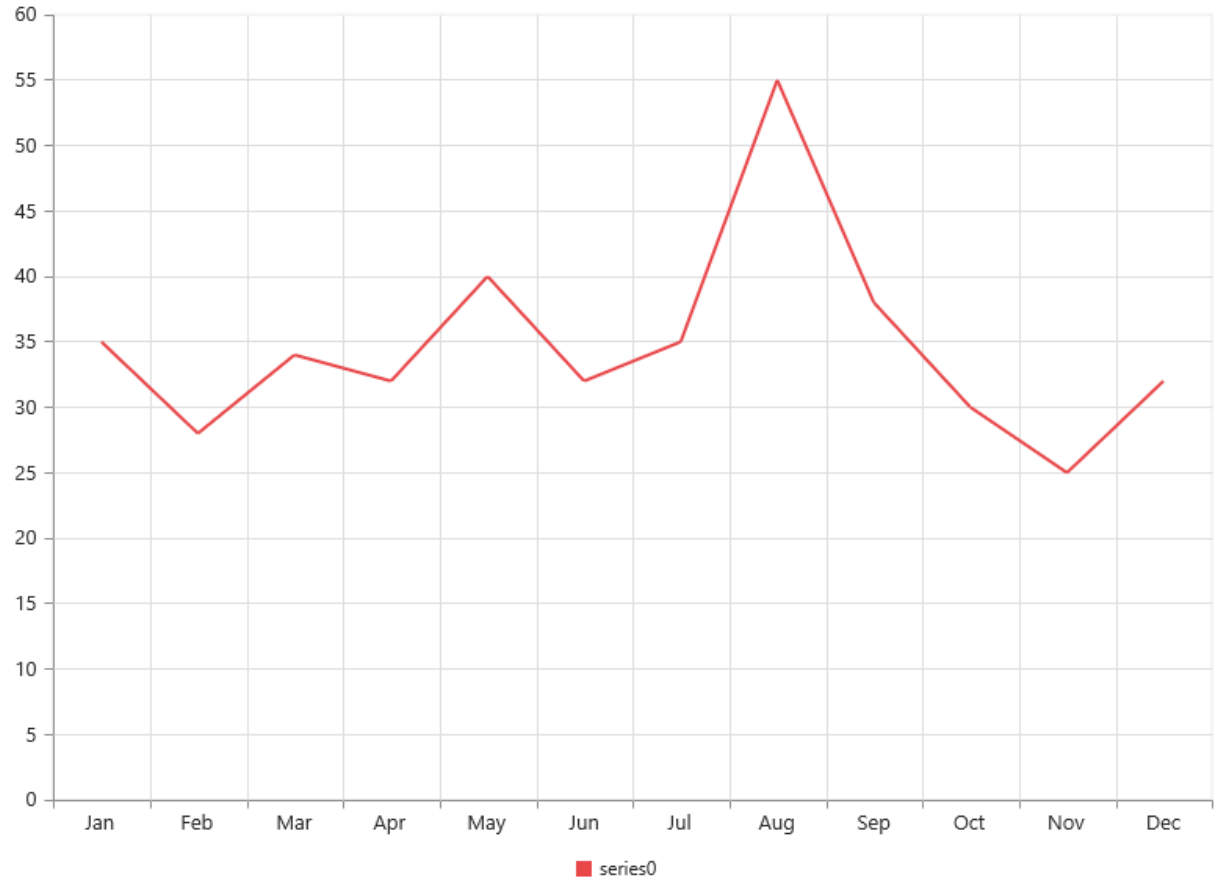
You can also add multiple series objects based on your requirement. Refer Chart Types and Chart Series sections to know more about chart types, how to add multiple series and customize series appearance.

Copyright © 2001 - 2020 Syncfusion Inc.

Next, map the Month and Sales values in the data source to the Line series by setting XName and YName with the field names respectively, and then set the actual data using DataSource option. Refer Data Binding section to know more about binding local and remote data to the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
sr.DataSource(ViewBag.ChartData)
.Type(SeriesType.Line)
.XName("Month").YName("Sales")
.Add();
})
)
```



Since the data is related to Sales, format the vertical axis labels by adding '$' as a prefix and 'K' as a suffix to each label. This can be achieved by setting the "${value}K" to the **LabelFormat** option of the axis. Here, {value} acts as a placeholder for each axis label, "$" and "K" are the actual prefix and suffix added to each axis label.

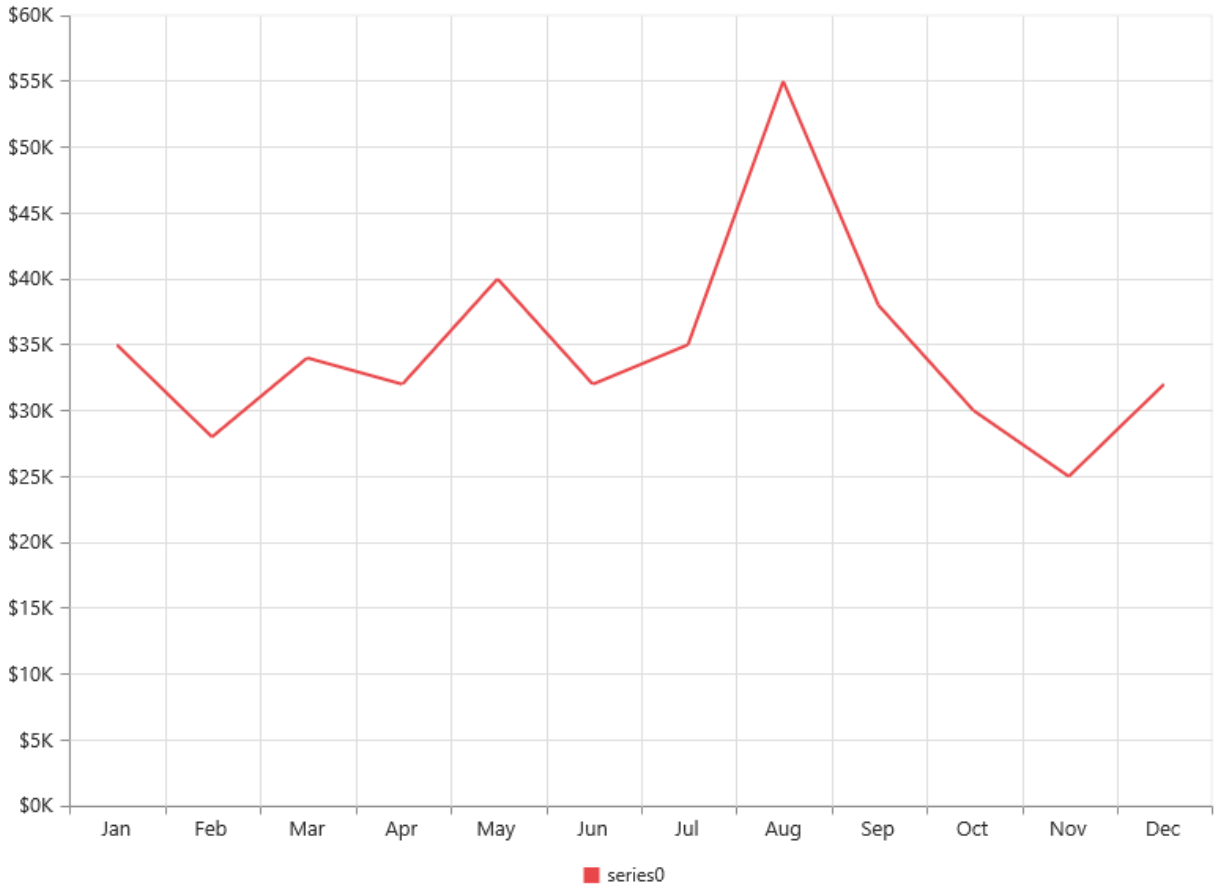
The following code example illustrates this,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Customize the axis label format.
.PrimaryYAxis(pry=>pry.LabelFormat("${value}K"))
//...
)
```



Refer Axis section to know more about axis types, adding multiple axes and other customization options.

Add Data Labels

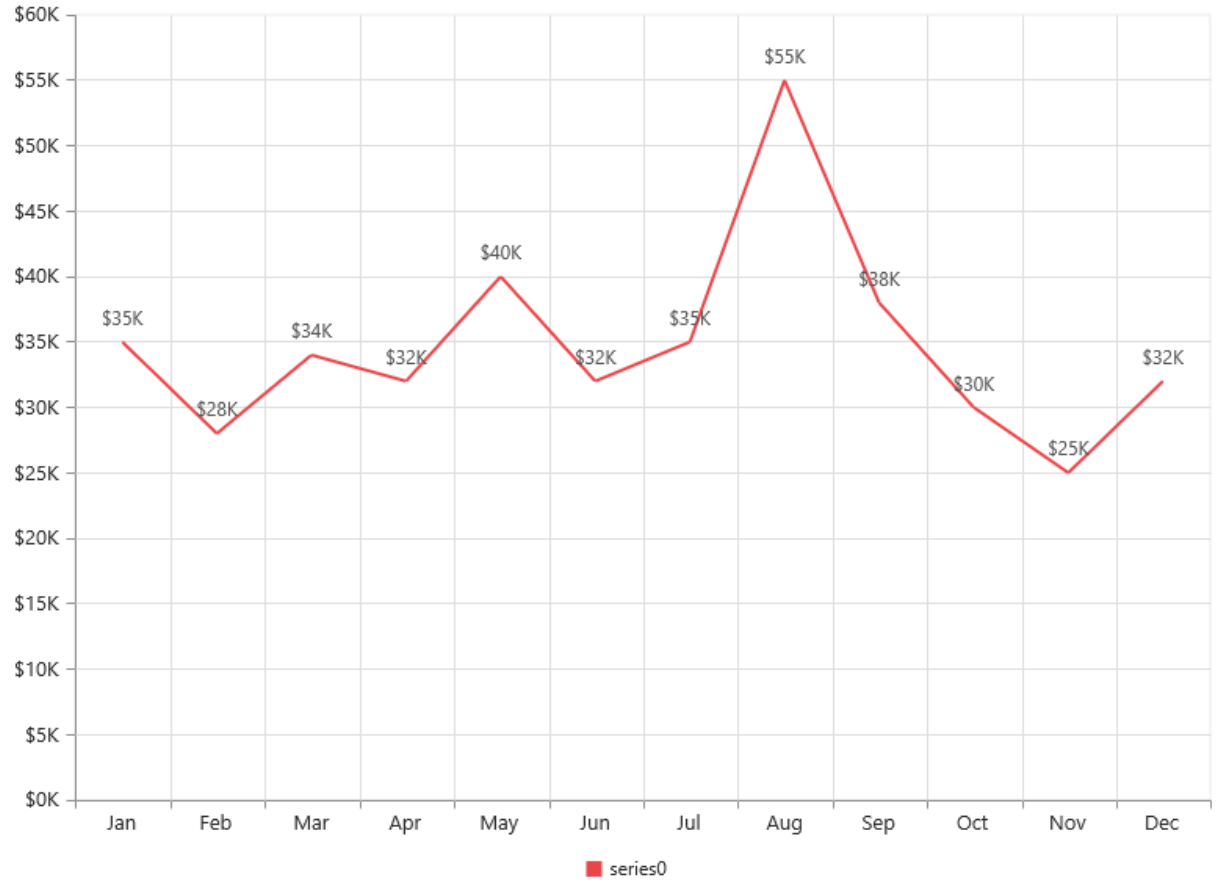
You can add data labels to improve the readability of the chart. This can be achieved by enabling the Visible option in the **DataLabel** option. Now, the data labels are rendered at the top of all the data points.

The following code example illustrates this,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//Display data label in chart
.Marker(mr=>mr.DataLabel(label=>label.Visible(true)))
//...
.Add();
})
//...
```

```
)
```



There are situations where the default label content is not sufficient to the user. In this case, you can use the **Template** option to format the label content with some additional information.

HTML

```
<!DOCTYPE html>
<html>
<body>
<div id="dataLabelTemplate" style="display:none; padding:3px;background-
color:#B9C5C9; opacity:0.8;">
<div id="point">#point.x#:$#point.y#K</div>
</div>
</body>
</html>
```

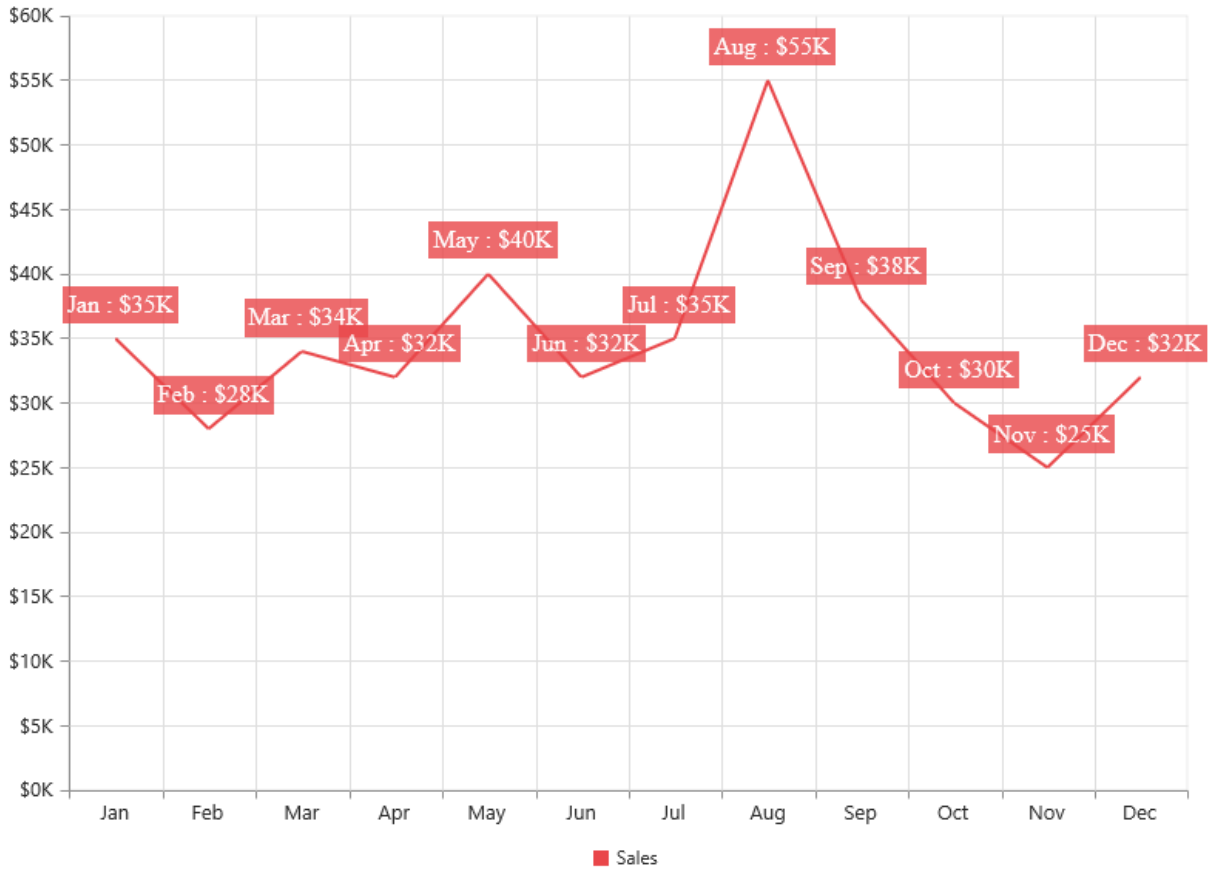
The above HTML template is used as a template for each data label. Here, “point.x” and “point.y” are the placeholder text used to display the corresponding data point’s x & y value.

The following code example shows how to set the id of the above template to Template option,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
```

```
.Series(sr =>
{
//Display data label in chart
.Marker(mr=>mr.DataLabel(label=>label.Visible(true).
//Set the id of HTML template to the chart series
Template("dataLabelTemplate"))
)
//...
.Add();
})
//...
)
```



Refer to the Data Markers section to know more about the options available to customize it.

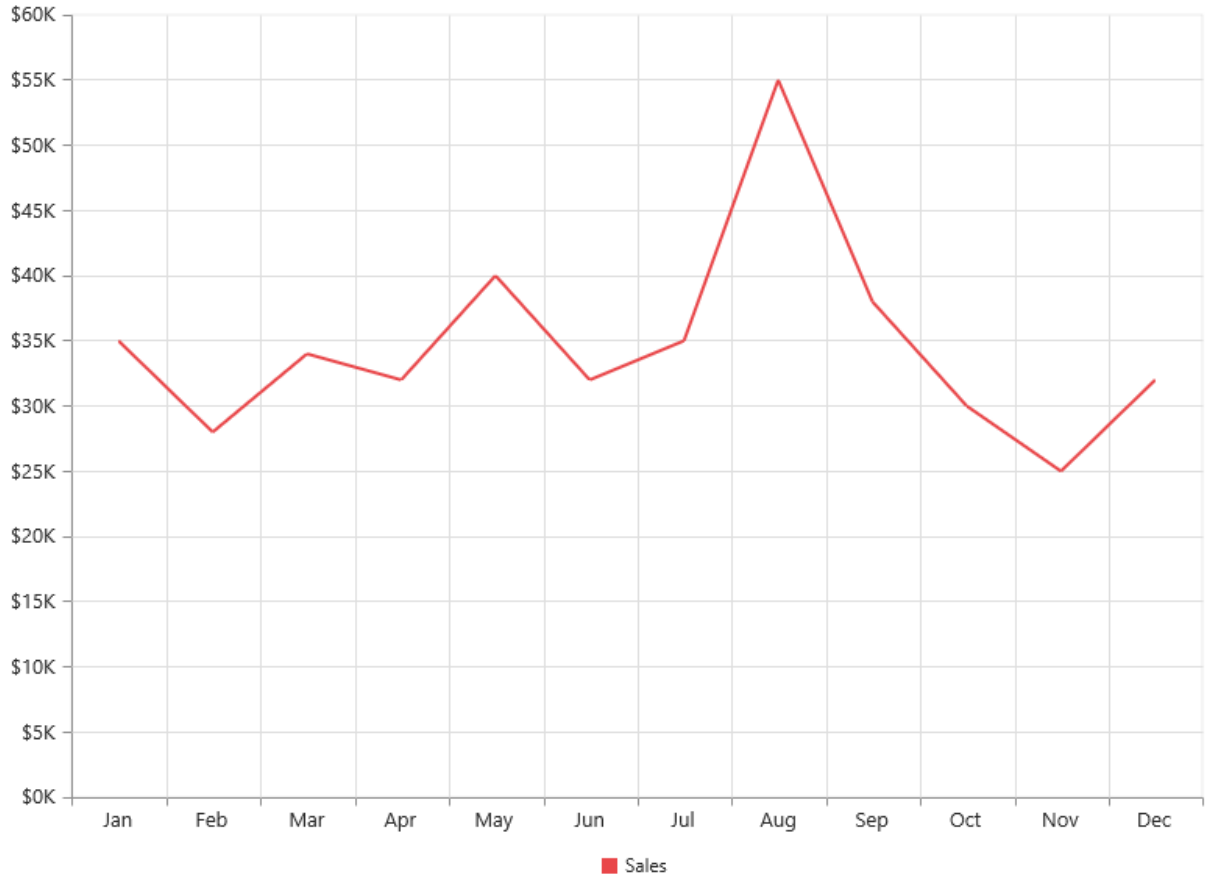
Enable Legend

You can enable or disable the legend by using the Visible option in the **Legend**. It is enabled in the chart, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//Add series name to display on the legend item
.Name("Sales")
```

```
.Add();
})
//Enable chart legend
.Legend(legend=>legend.Visible(true))
//...
)
```



Refer to the Legend section to know more about how to position legend and customize its appearance.

Enable Tooltip

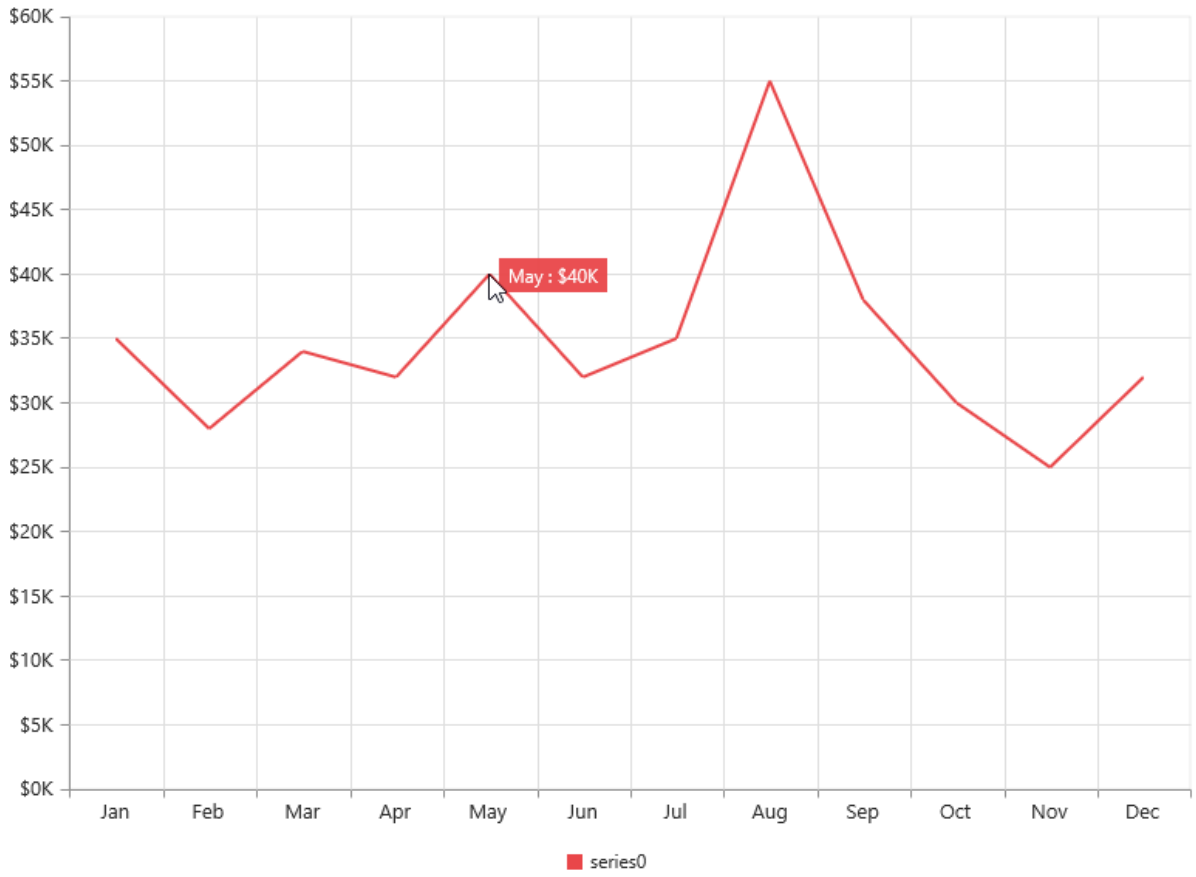
The Tooltip is useful when you cannot display information by using the **DataLabels** due to the space constraints. You can enable tooltip by using the Visible option of the **Tooltip** in the specific series.

The following code example illustrates this,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//...
//Enable tooltip in chart area
.Tooltip(tooltip=>tooltip.Visible(true))
.Add();
})
//...
```

```
)
```



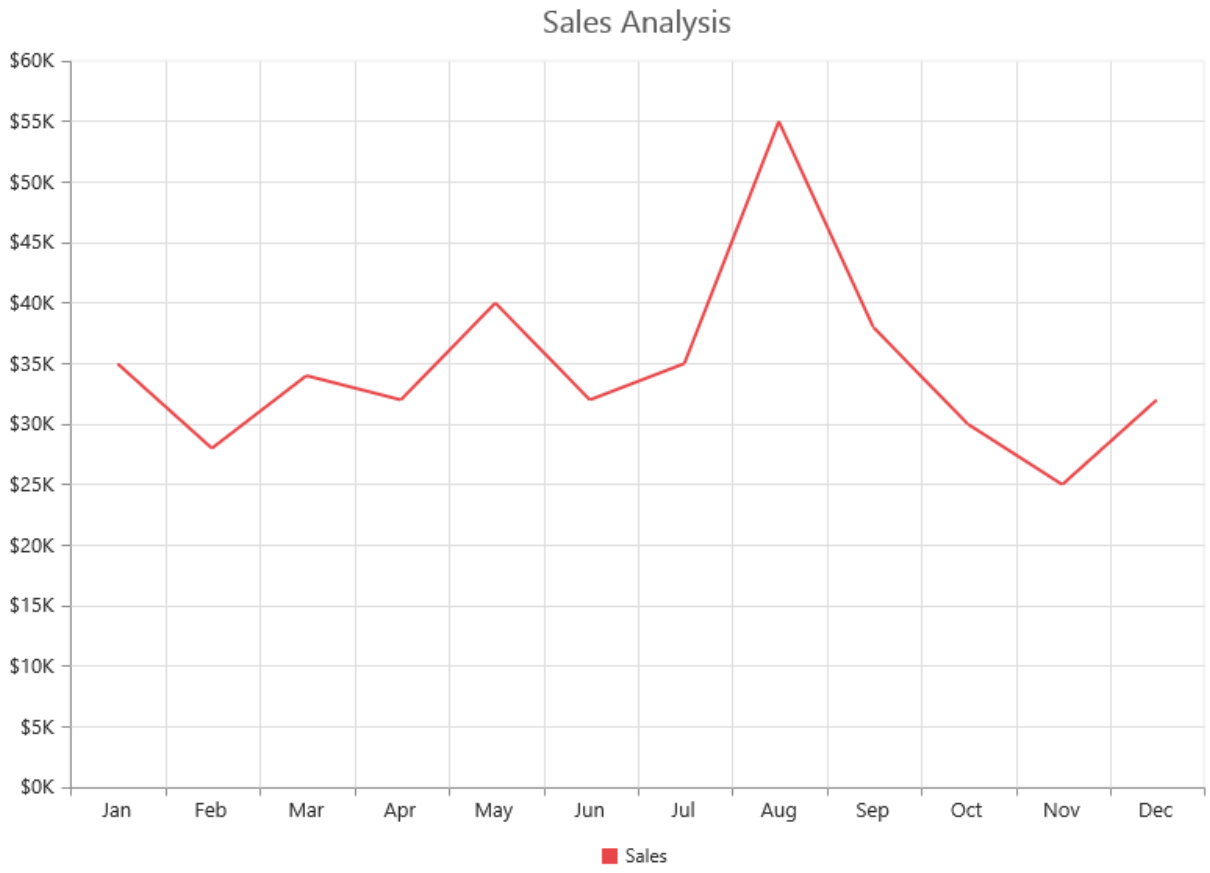
Refer to the Tooltip section to know more about formatting tooltip contents and customizing its appearance.

Add Chart Title

You need to add a title to the chart to provide quick information to the user about the data being plotted in the chart. You can add it by using the Text option of the **Title**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Add chart title
.Title(title=>title.Text("Sales Analysis"))
// ...
)
```



Refer to the Chart Title section to know more about aligning Title, customizing its appearance and adding SubTitle to the chart.

Working with Data

Local Data

There are two ways to provide local data to chart.

1. You can bind the data to the chart by using the **DataSource** property of the series and then you need to map the X and Y value with the *XName* and *YName* properties respectively.

Note: For the **OHLC** type series, you have to map four dataSource fields *High, Low, Open* and *Close* to bind the data source and for the **bubble** series you have to map the *Size* field along with the *XName* and *YName*.

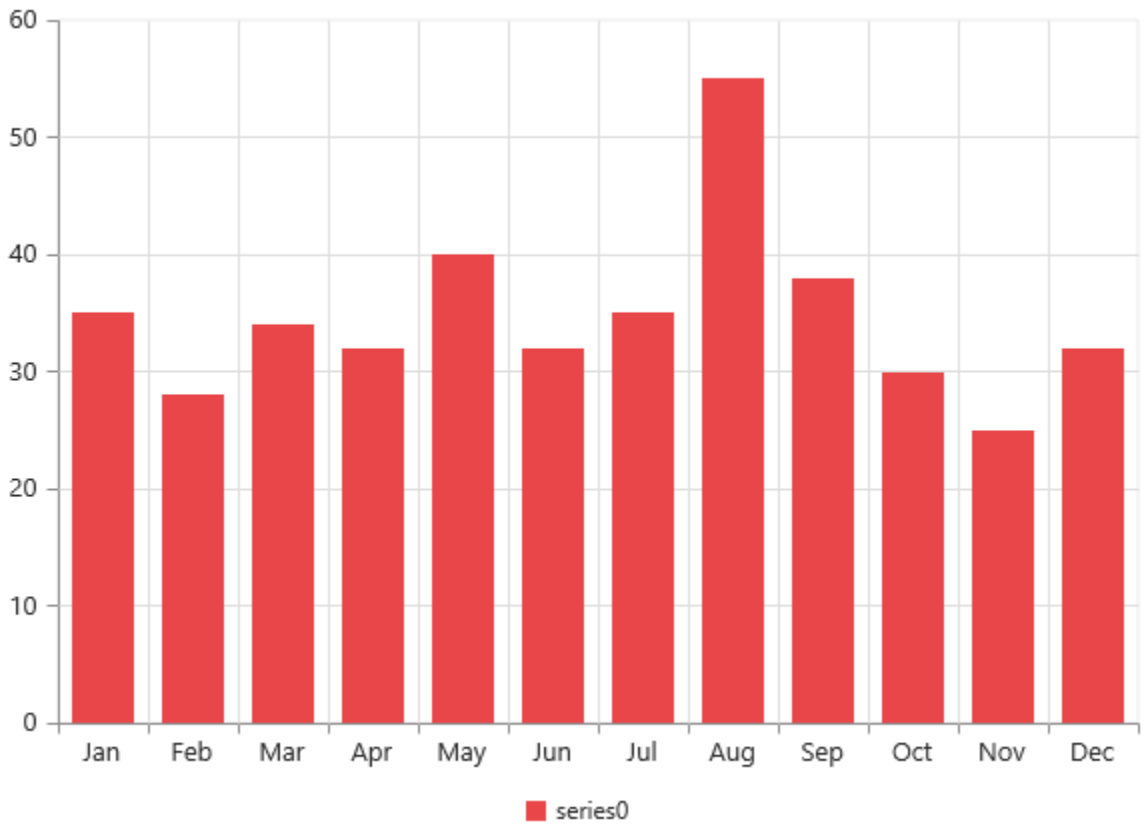
ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//Add series
sr.Type(SeriesType.Line).Add();
})
.Load("onChartLoad")
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
var chartData = [
{ month: 'Jan', sales: 35 }, { month: 'Feb', sales: 28 },  { month: 'Mar',
sales: 34 },
{ month: 'Apr', sales: 32 },{ month: 'May', sales: 40 },{ month: 'Jun',
sales: 32 },
{ month: 'Jul', sales: 35 },  { month: 'Aug', sales: 55 }, { month: 'Sep',
sales: 38 },
{ month: 'Oct', sales: 30 }, { month: 'Nov', sales: 25 }, { month: 'Dec',
sales: 32 }];
function onChartLoad(sender) {
var data = GetData();
sender.model.series[0].dataSource = chartData;
sender.model.series[0].xName = "month";
sender.model.series[0].yName = "sales";
}
```



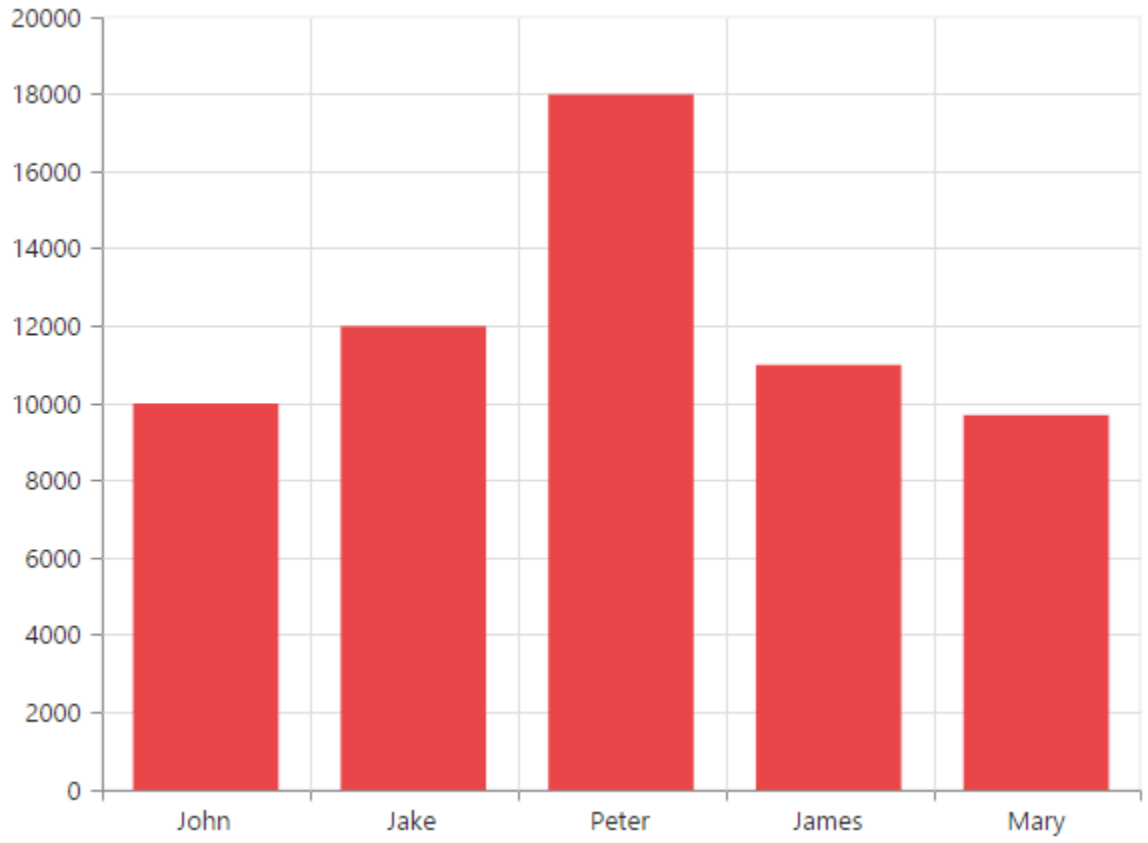
Click here to view the local data binding online demo sample.

2.You can also plot data to chart using **Points** option in the series. Using this property you can customize each and every point in the data.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
sr
//Adding data points using x and y field of points
.Points(pt =>
{
pt.X("John").Y(10000).Add();
pt.X("Jake").Y(12000).Add();
pt.X("Petter").Y(18000).Add();
pt.X("James").Y(11000).Add();
pt.X("Mary").Y(9700).Add();
}).Add();
})
//...
)
```



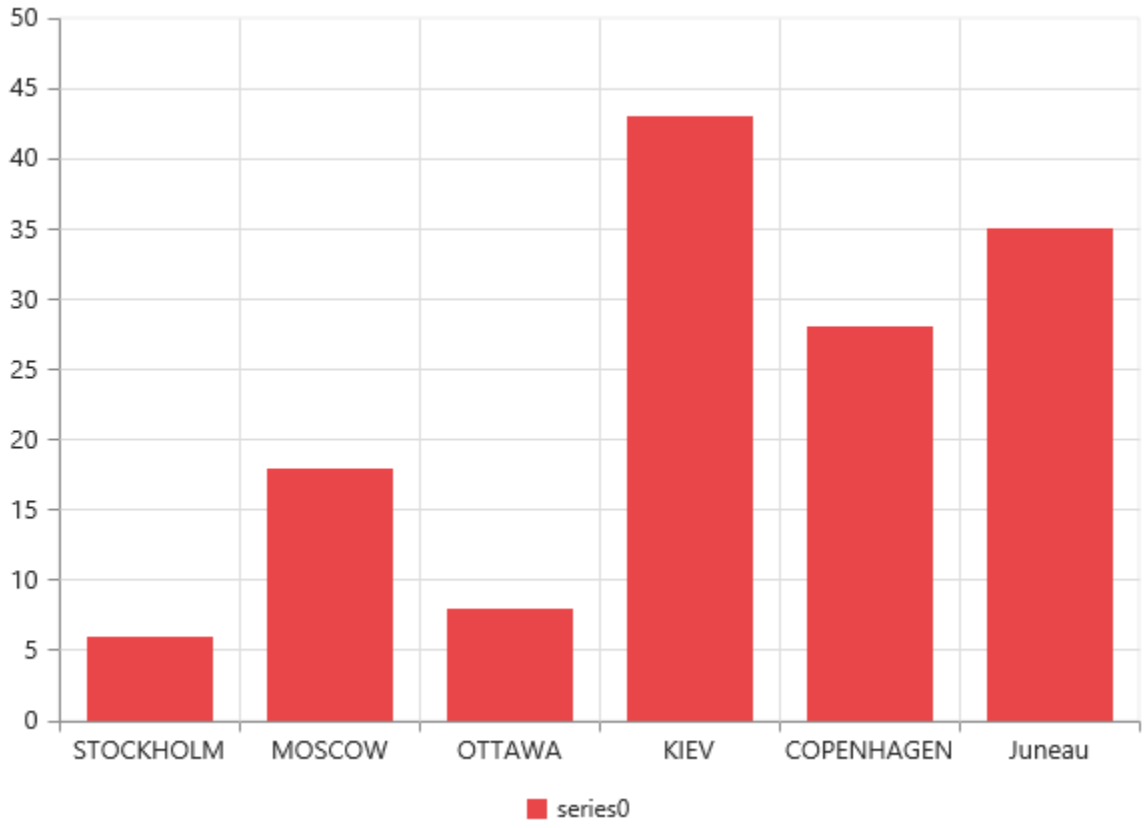
Remote Data

You can bind the remote data to the chart by using the **DataSource** and you can use the **Query** property of the series to filter the data from the dataSource.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
```

```
//...
.Series(ser =>
{
ser.DataSource(service =>
service.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/"));
ser.XName("ShipCity");
ser.YName("Freight");
ser.Query("ej.Query().from('Orders').take(10)").Add();
})
//...
)
```



Click here to view the remote data binding online demo sample.

AngularJS Data Binding

Typically, you will assign data directly to chart using **dataSource** property of the series. In AngularJS, you need to bind the variable, which contains data in the AngularJS scope object, to the dataSource property as illustrated in the following code example,

Information: Essential JS includes AngularJS directives for all controls in the **ej.widget.angular.min.js** script file.

Note: All the properties in EjChart supports one way AngularJS binding except inner array properties like **series.points[]**, **series.trendlines[]**. Click here to know more about Essential AngularJS and the properties which support two way AngularJS binding in chart.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<html ng-app="syncApp">
<head>
<script type="text/javascript"
src="http://cdn.syncfusion.com/js/assets/external/jquery-
2.1.4.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/angular.min.js"></script>
<script src="https://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
<script src="https://cdn.syncfusion.com/{{ site.releaseversion
}}/js/common/ej.widget.angular.min.js"></script>
</head>
<body ng-controller="Chart">
<div id="chartContainer" style="width:100%" ej-chart
e-size-width="800px" e-size-height="600px"
e-title-text="AngularJS Support" >
<e-series>
<e-series e-name="John" e-dataSource=dataSource e-xName="Day" e-
yName="John">
</e-series>
<e-series e-name="Hendry"  e-dataSource=dataSource e-xName="Day" e-
yName="Hendry">
</e-series>
</e-series>
</div>
</body>
</html>
```

JS

```
//Data source for chart.
var obj = [
{ "Day": 1, "John": 57, "Hendrik": 43 },
{ "Day": 2, "John": 73, "Hendrik": 27 },
{ "Day": 3, "John": 49, "Hendrik": 51 },
{ "Day": 4, "John": 63, "Hendrik": 37 },
{ "Day": 5, "John": 44, "Hendrik": 56 },
{ "Day": 6, "John": 49, "Hendrik": 51 },
{ "Day": 7, "John": 61, "Hendrik": 39 },
{ "Day": 8, "John": 35, "Hendrik": 65 },
{ "Day": 9, "John": 45, "Hendrik": 55 },
{ "Day": 10, "John": 37, "Hendrik": 63 }
];
angular.module('syncApp', ['ejangular'])
.controller('Chart', function ($scope) {
//Assigning data to the dataSource variable in the $scope object.
$scope.dataSource = obj;
});
```


Click here to view the AngularJS data binding online demo sample.

Copyright © 2001 - 2020 Syncfusion Inc.

Chart Dimensions

You can set the size of the chart directly on the chart or to the container of the chart. When you do not specify the size, it takes 450px as the height and window size as its width, by default.

Set size for the container

You can customize the chart dimension by setting the width and height for the container element.

ASPX-CS

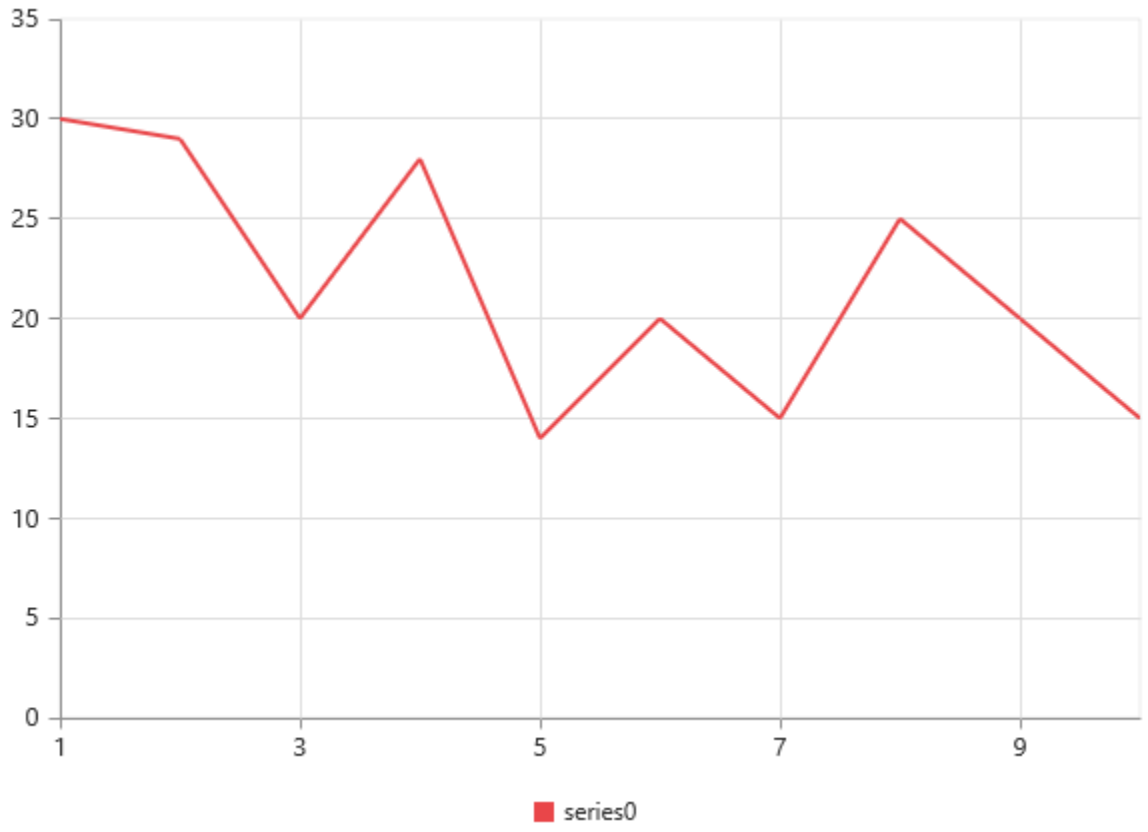
```
<div id="container" style="width:820px; height:500px;">
@(Html.EJ().Chart("chartContainer")
// ...
)
</div>
```

Set size in pixels

You can also set the chart dimension by using the **Size** property of the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Set size to chart container
.Size(size=>size.Height("450").Width("600"))
)
```



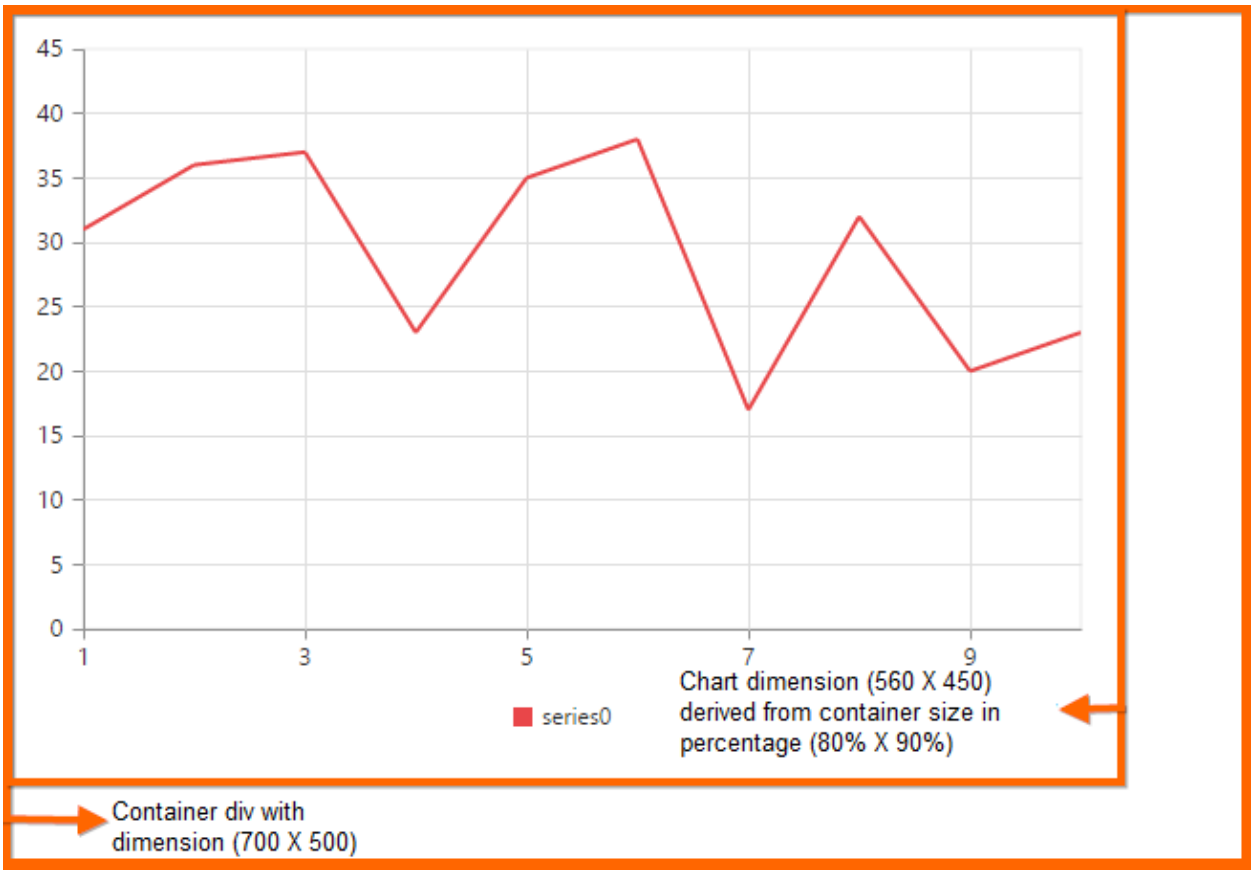
Copyright © 2001 - 2020 Syncfusion Inc.

Setting size relative to the container size

You can specify the chart size in percentage by using the Size property. The chart gets its dimension with respect to its container.

ASPX-CS

```
<div id="container" style="width:700px; height:500px">
@(Html.EJ().Chart("container")
// ...
//Set size to chart container
.Size(size=>size.Height("90%").Width("80%"))
)
</div>
```



Responsive chart

To resize the Chart when the browser or the chart container is resized, set the **IsResponsive** property to **true**, where the chart adapts to the changes in size of the container.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Enable IsResponsive to change the chart size dynamically.
.IsResponsive(true)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Axis

Charts typically have two axes that are used to measure and categorize data: a vertical (y) axis, and a horizontal (x) axis.

Vertical axis always uses numerical or logarithmic scale. Horizontal(x) axis supports the following types of scale:

- Category
- Numeric
- DateTime
- DateTime Category
- Logarithmic

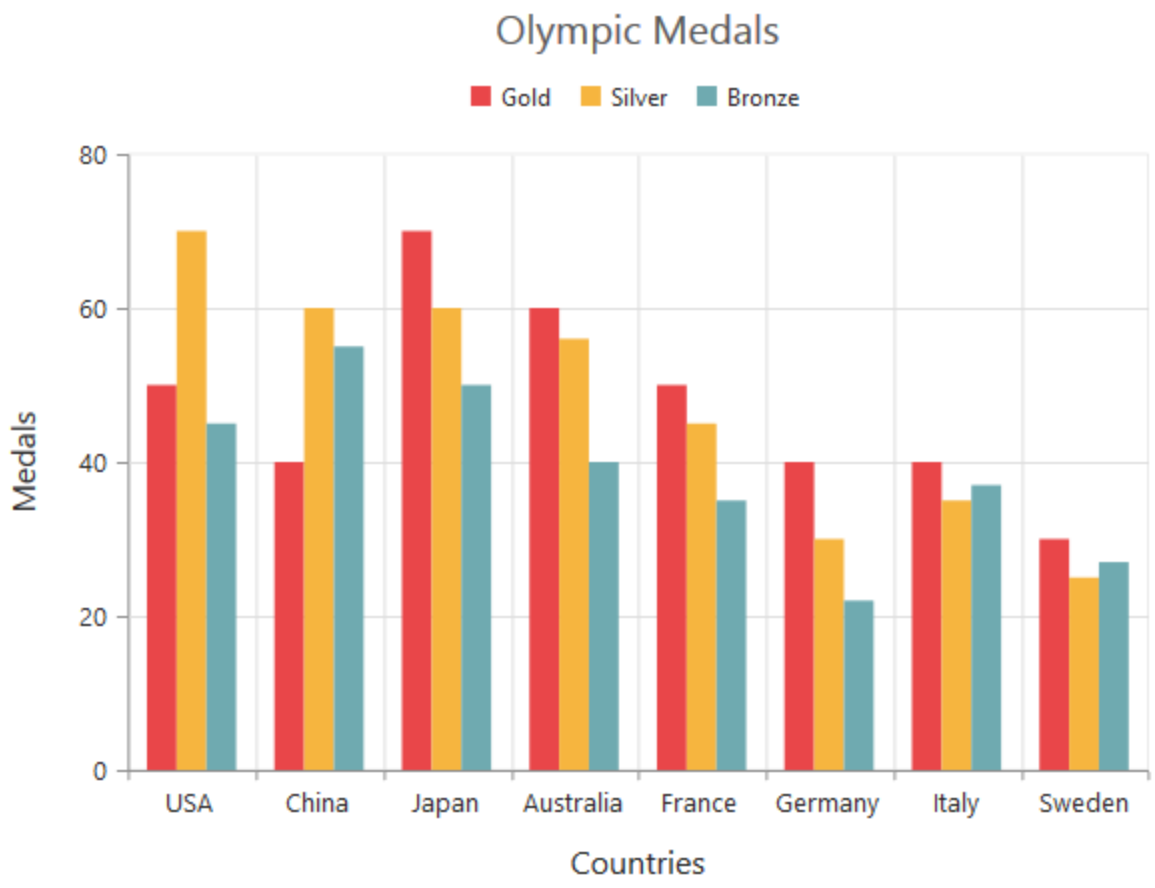
Category Axis

Category axis displays the text labels instead of numbers. To use the categorical axis, you can set the **ValueType** property of the axis to the **Category**. Default value of ValueType is **Double**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Use categorical scale in primary X axis
px=>px.ValueType(AxisValueType.Category)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view our online demo sample that uses Category axis.

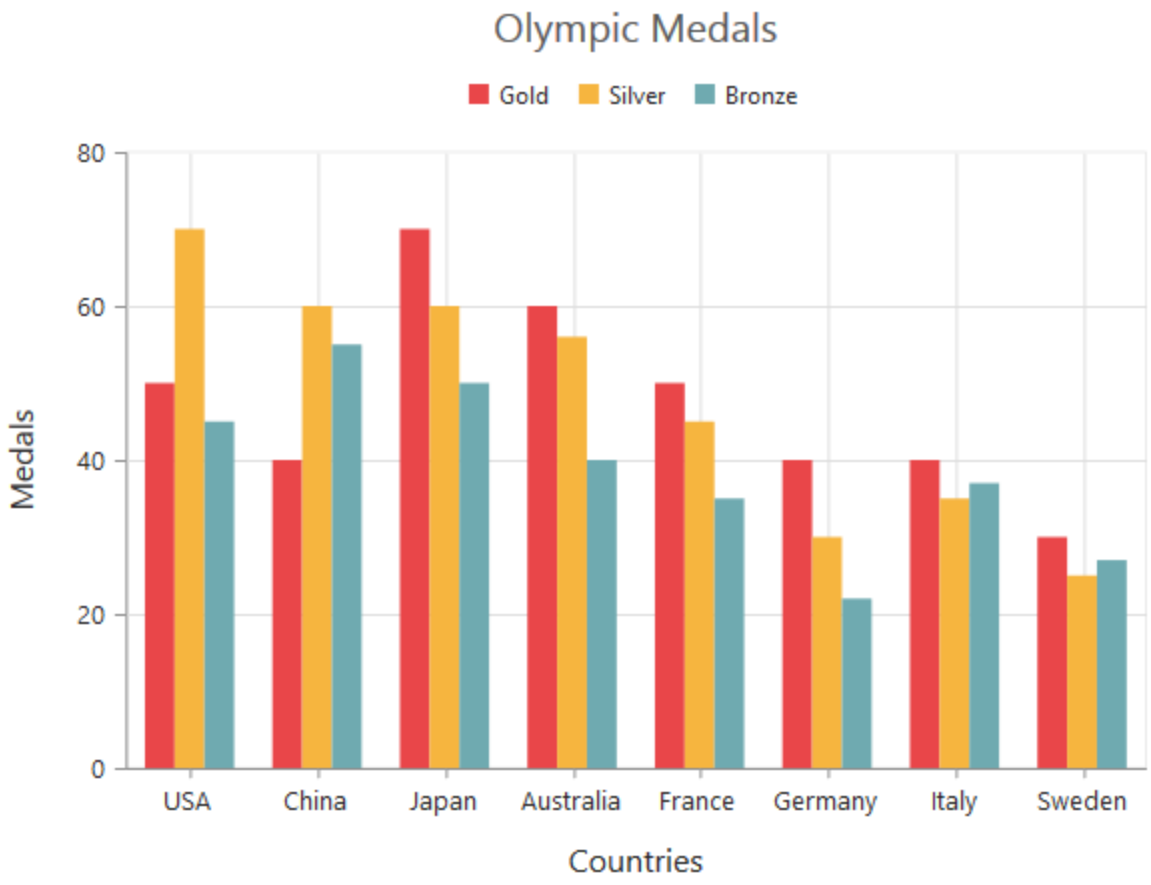
Place labels on ticks

Labels in the category axis can be placed on the ticks by setting the **LabelPlacement** property of axis to the onTicks. The default value of the LabelPlacement property is betweenTicks i.e. labels are placed between the ticks, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Placing X-axis labels on the ticks
px=>px.LabelPlacement("onTicks")
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



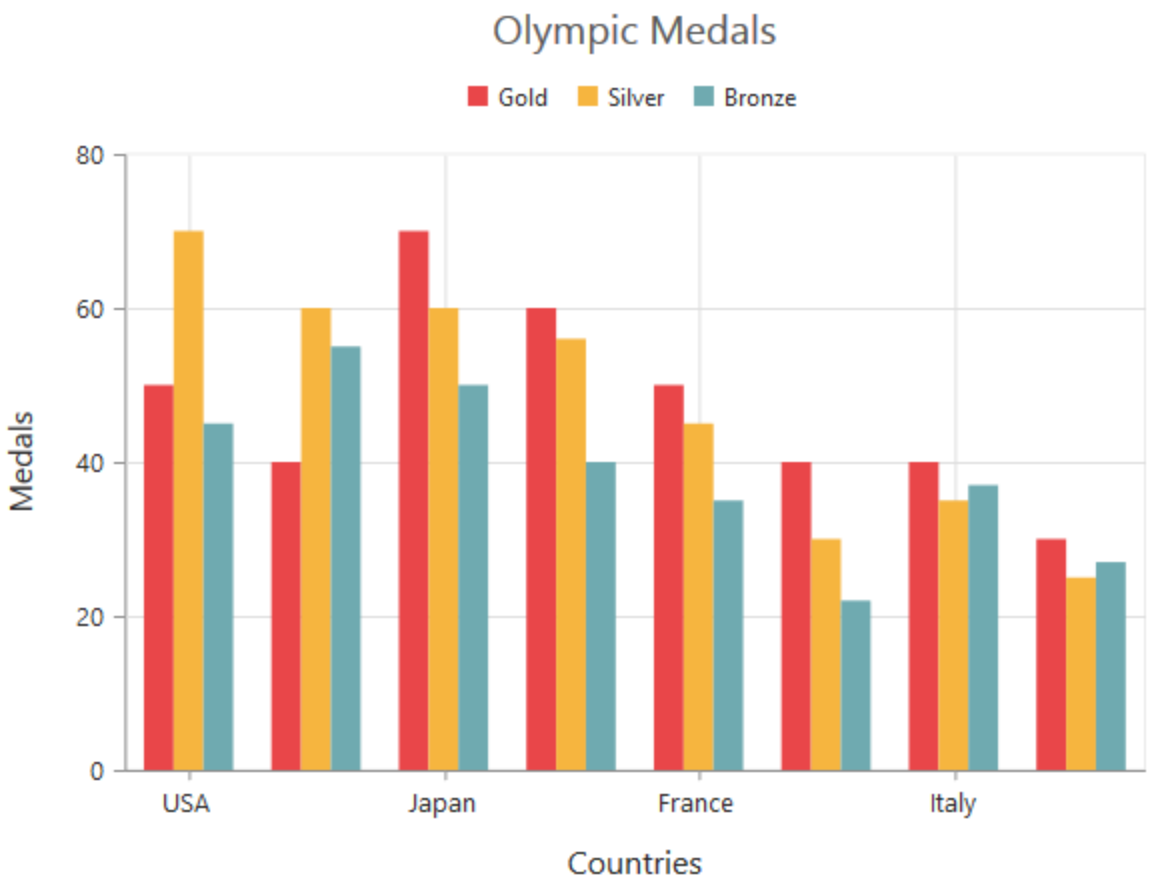
Display labels after a fixed interval

To display the labels after a fixed interval n, you can set the **Interval** property of the axis range as **n**. The default value of the interval is 1 i.e. all the labels are displayed.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Displaying labels after 2 intervals
axis=>axis.Range(range=>range.Interval(2))
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



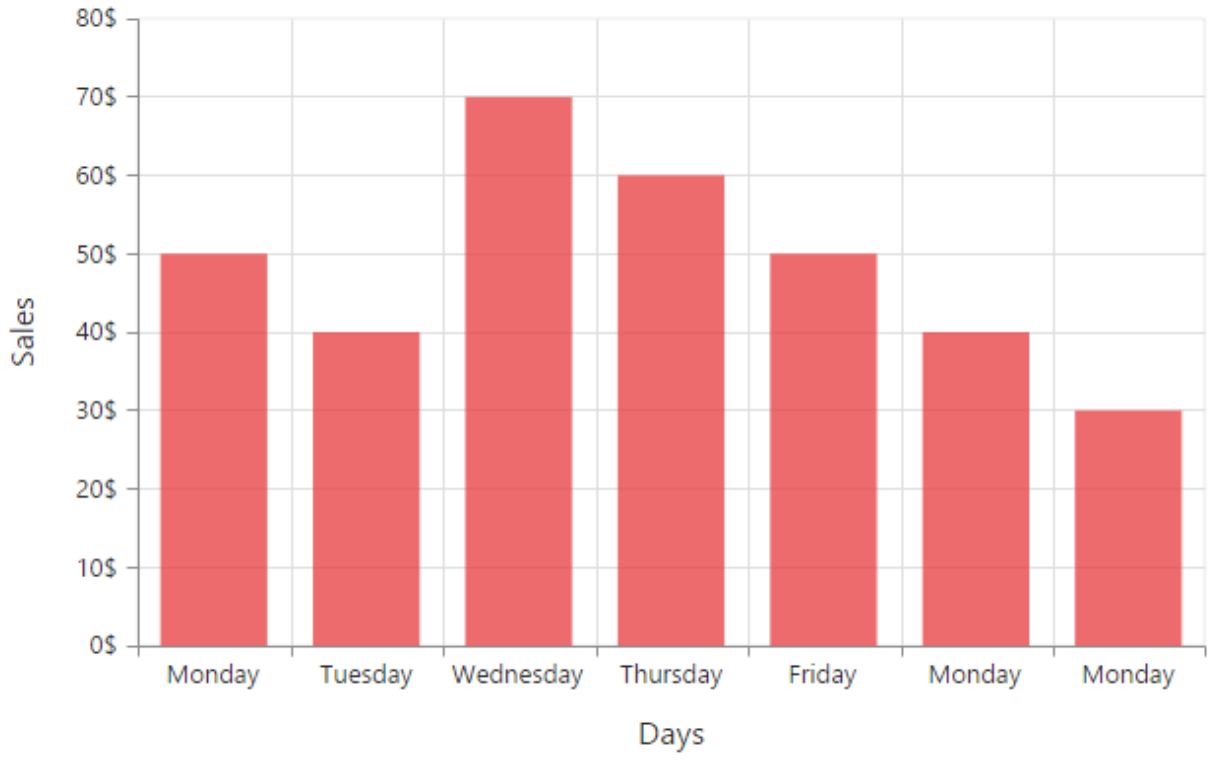
Indexed Category Axis

Category axis can also plot points based on index value of data points. Index based plotting can be enabled by setting **IsIndexed** property to true in the axis.

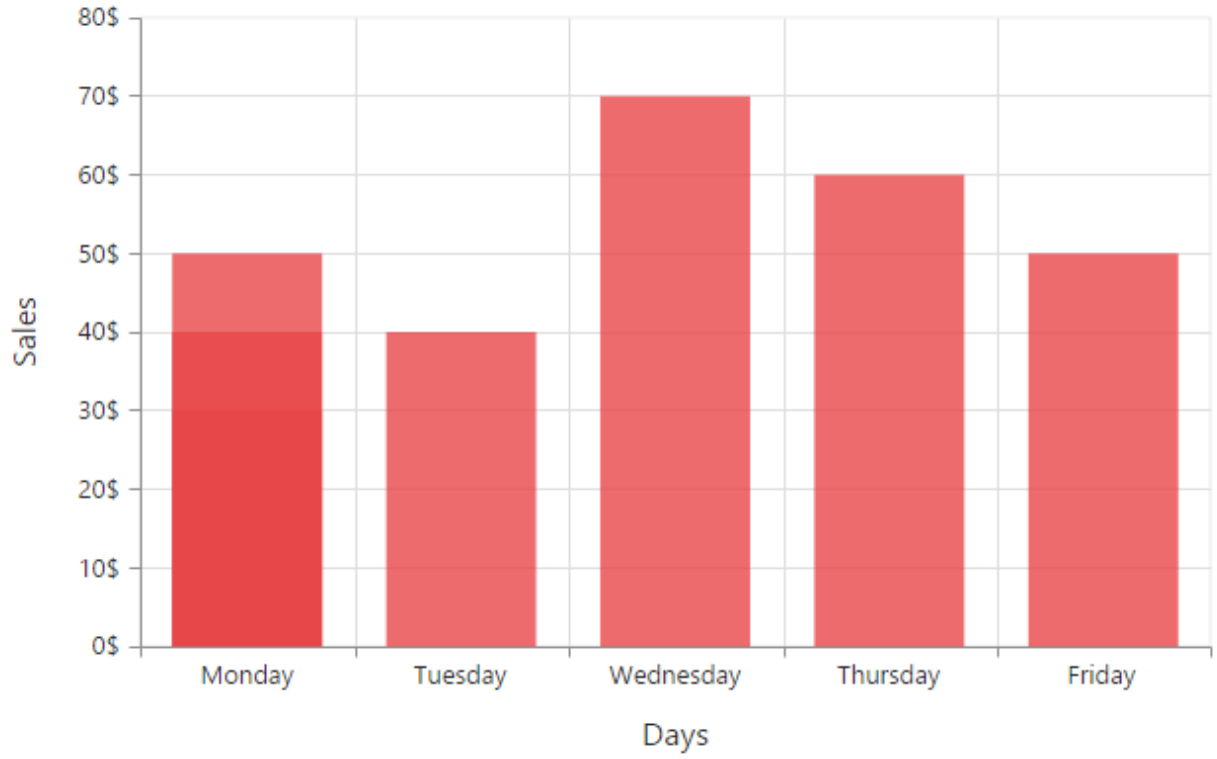
ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.PrimaryXAxis(px=>px.IsIndexed(true))
.Series(sr =>
{
//Adding Candle series
sr.Points(pt=>{
pt.X("Monday").Y(50).Add();
pt.X("Tuesday").Y(40).Add();
pt.X("Wednesday").Y(70).Add();
pt.X("Thursday").Y(60).Add();
pt.X("Friday").Y(50).Add();
pt.X("Monday").Y(40).Add();
pt.X("Monday").Y(30).Add();
}) .Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



While Category axis IsIndexed value false



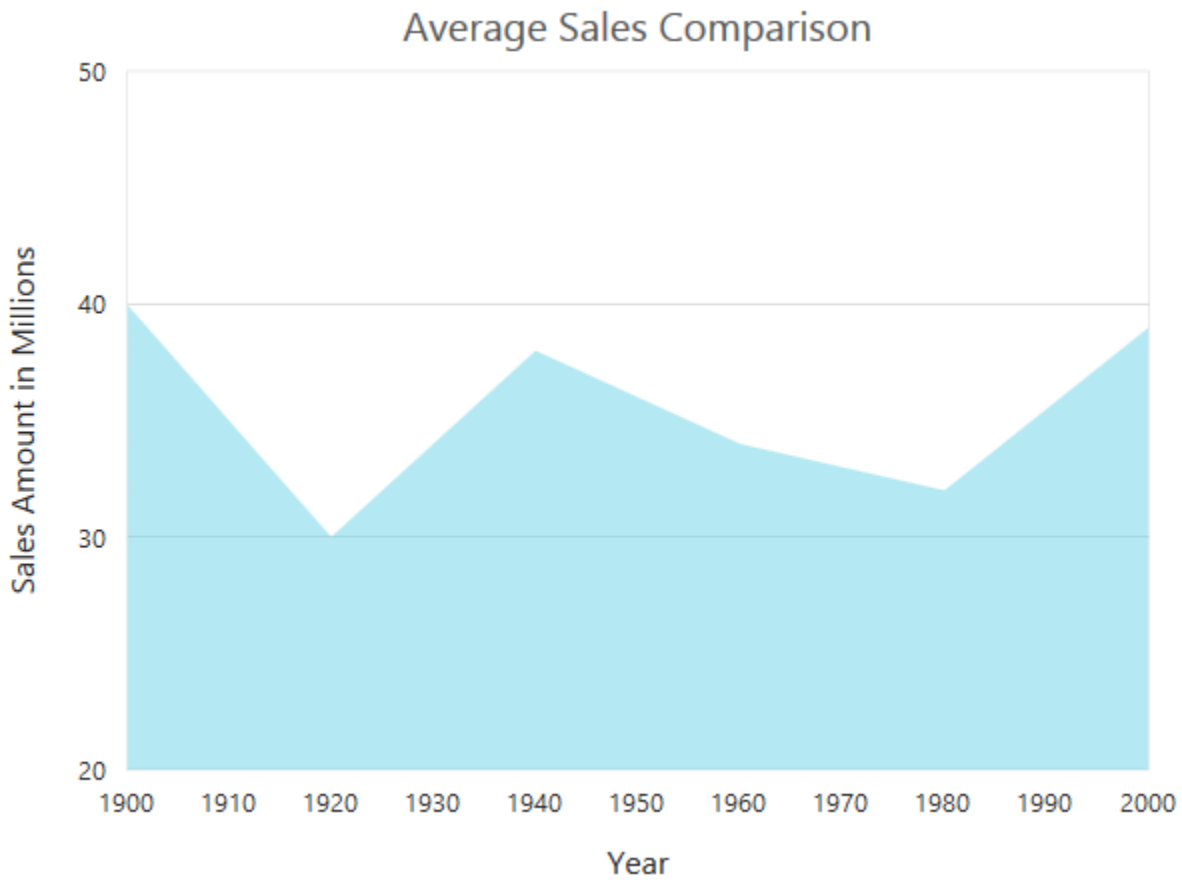
Copyright © 2001 - 2020 Syncfusion Inc.

Numeric Axis

Numeric axis uses numerical scale and displays numbers as labels. To use numeric axis, you can set the **ValueType** property of the axis to **Double**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Use numerical scale in primary Y axis
px=>px.ValueType(AxisValueType.Double)
)
// ...
)
```



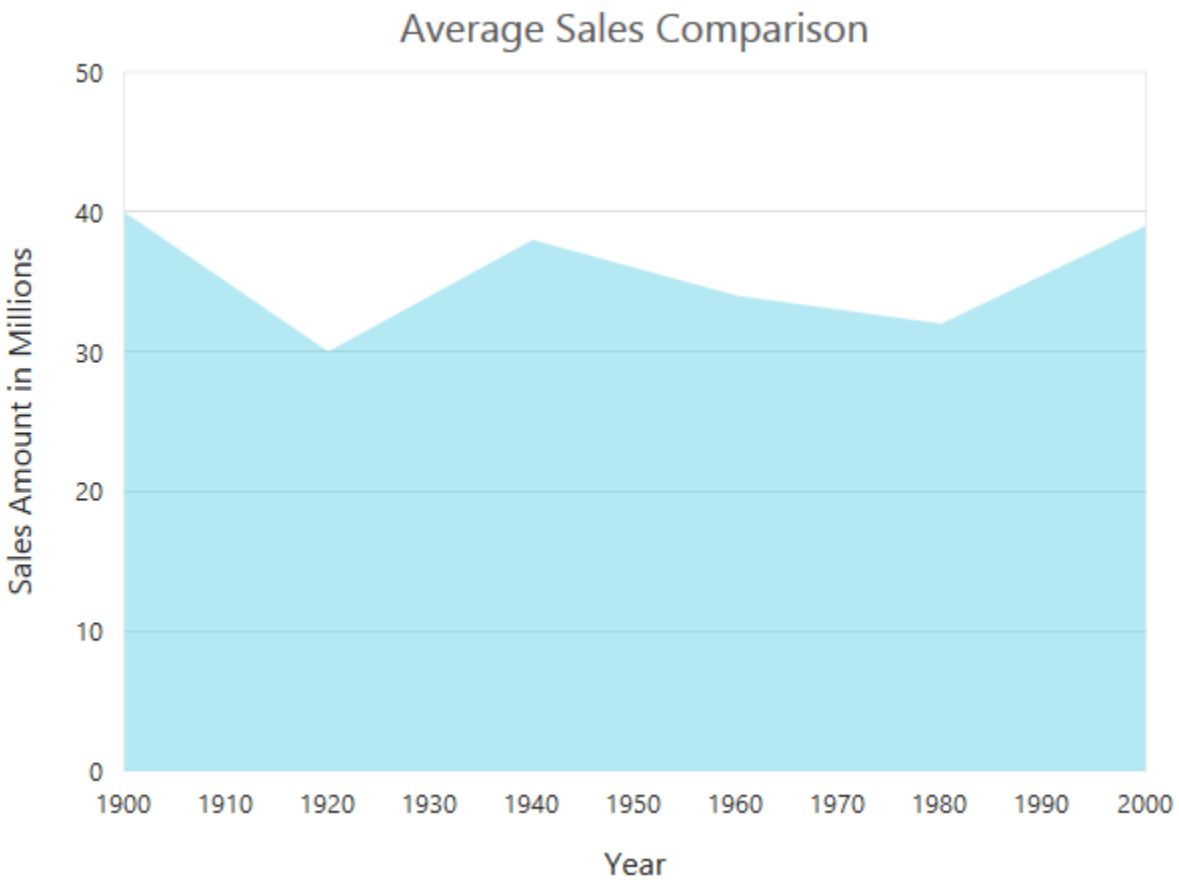
Customize numeric range

To customize the range of an axis, you can use the **Range** property of the axis to set the **Minimum**, **Maximum** and **Interval** values. Nice range is calculated automatically based on the provided data, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
```

```
//Customizing Y-axis range
px=>px.Range(range=>range.Min(0).Max(50))
)
// ...
)
```



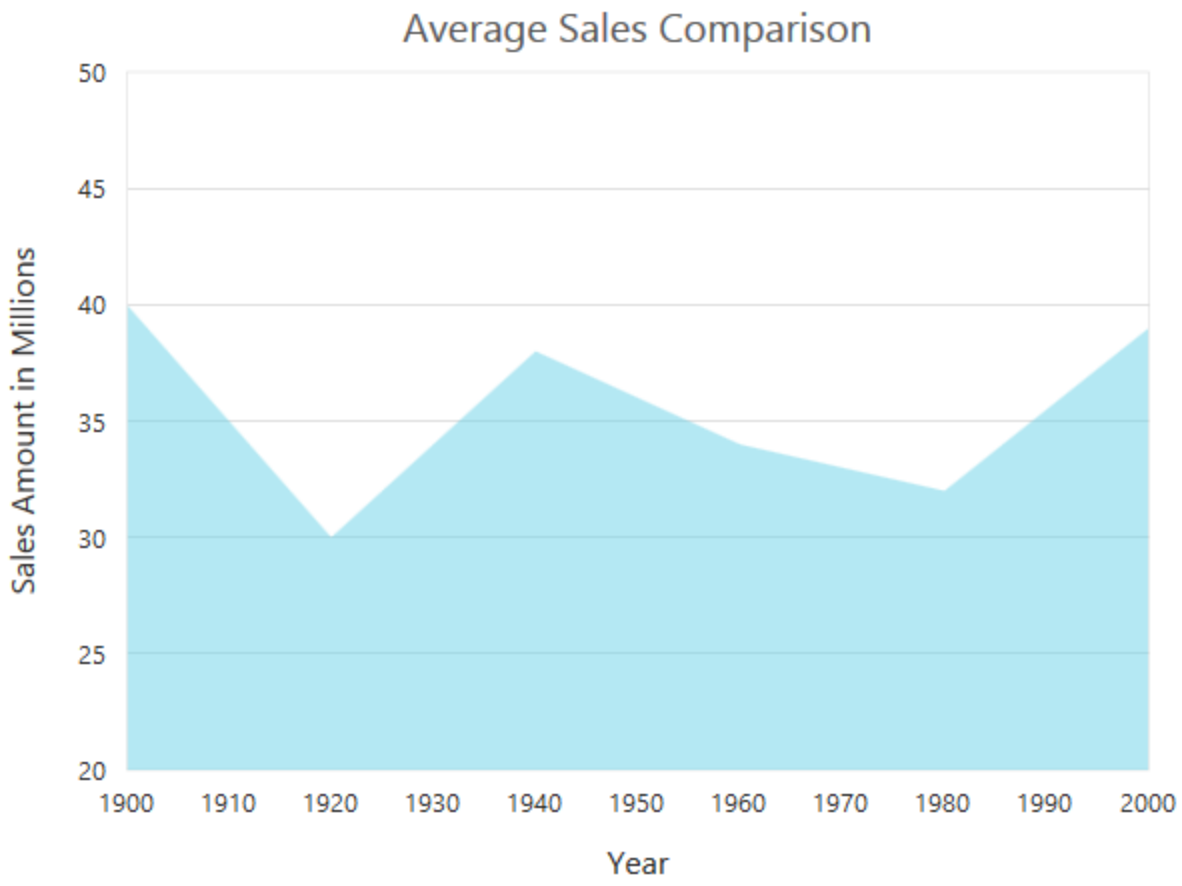
Customizing numeric interval

Axis interval can be customized by using the **Interval** property of the axis range. Nice interval is calculated based on the minimum and maximum value of the provided data, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Set interval to PrimaryYAxis
px=>px.Range(range=>range.Interval(5))
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Apply padding to the range

Padding can be applied to the minimum and maximum extremes of the axis range by using the **RangePadding** property. Numeric axis supports the following types of padding

- None
- Round
- Additional
- Normal

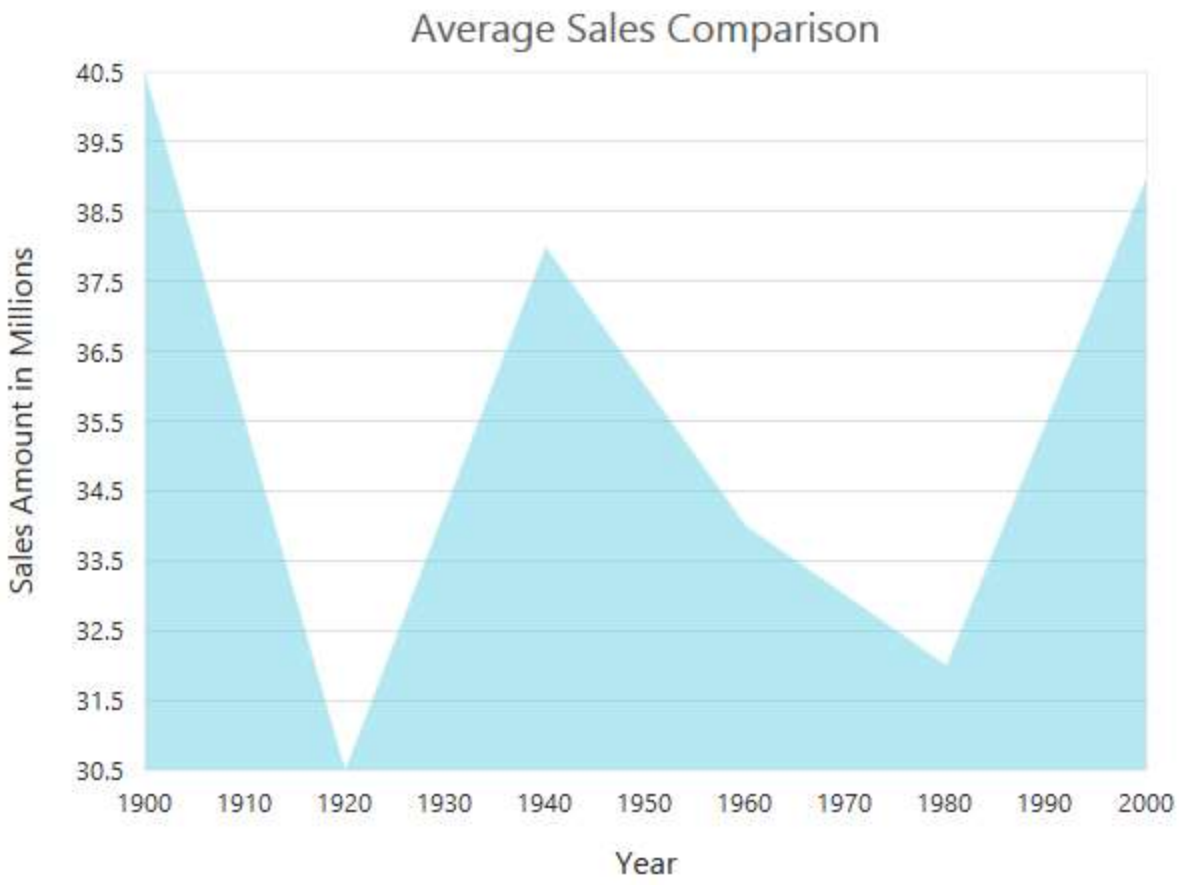
None

When the value of the RangePadding property is **None**, padding can not be applied to the axis. This is also the default value of the rangePadding.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Applying none as range padding
px=>px.RangePadding(ChartRangePadding.None)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Round

When the value of RangePadding property is **Round**, the axis range is rounded to the nearest possible value divided by the interval.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Applying round as range padding
px=>px.RangePadding(ChartRangePadding.Round)
)
// ...
)
```

Chart before rounding axis range

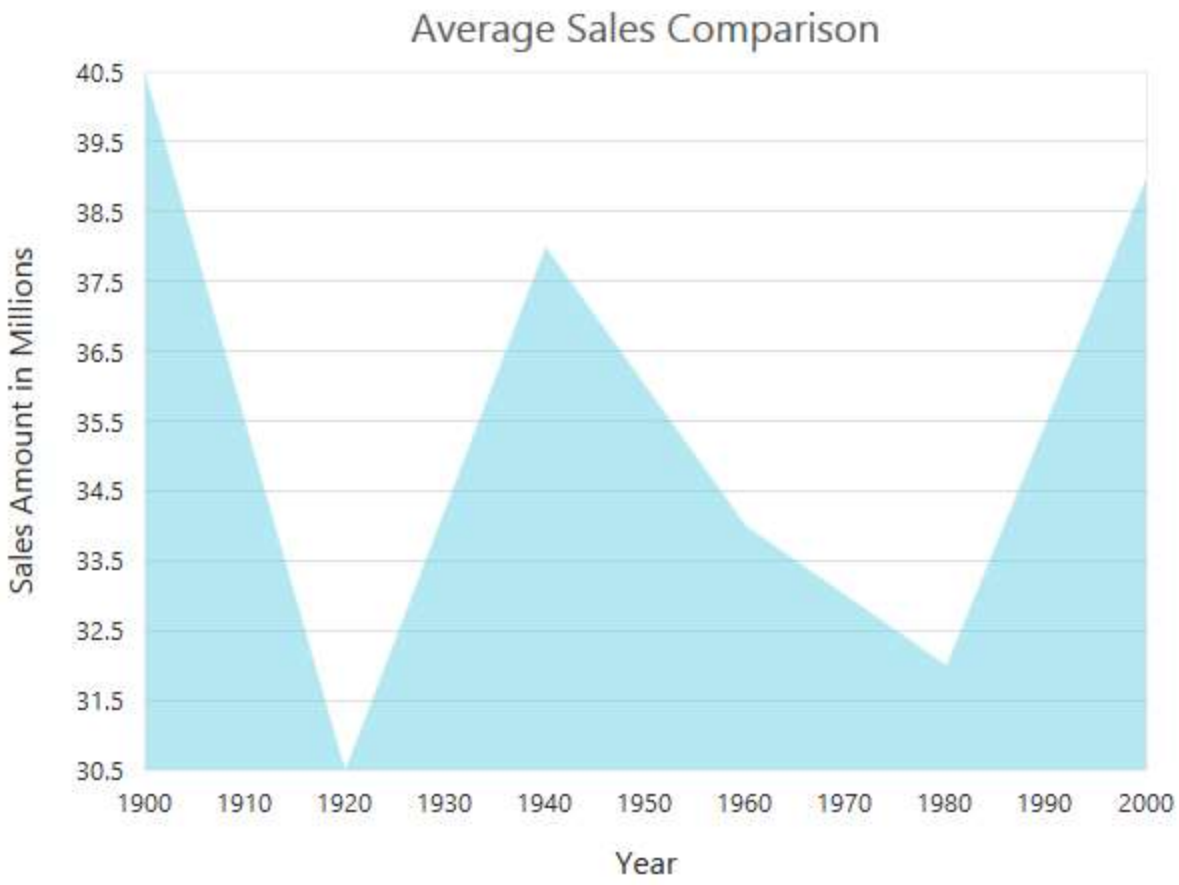
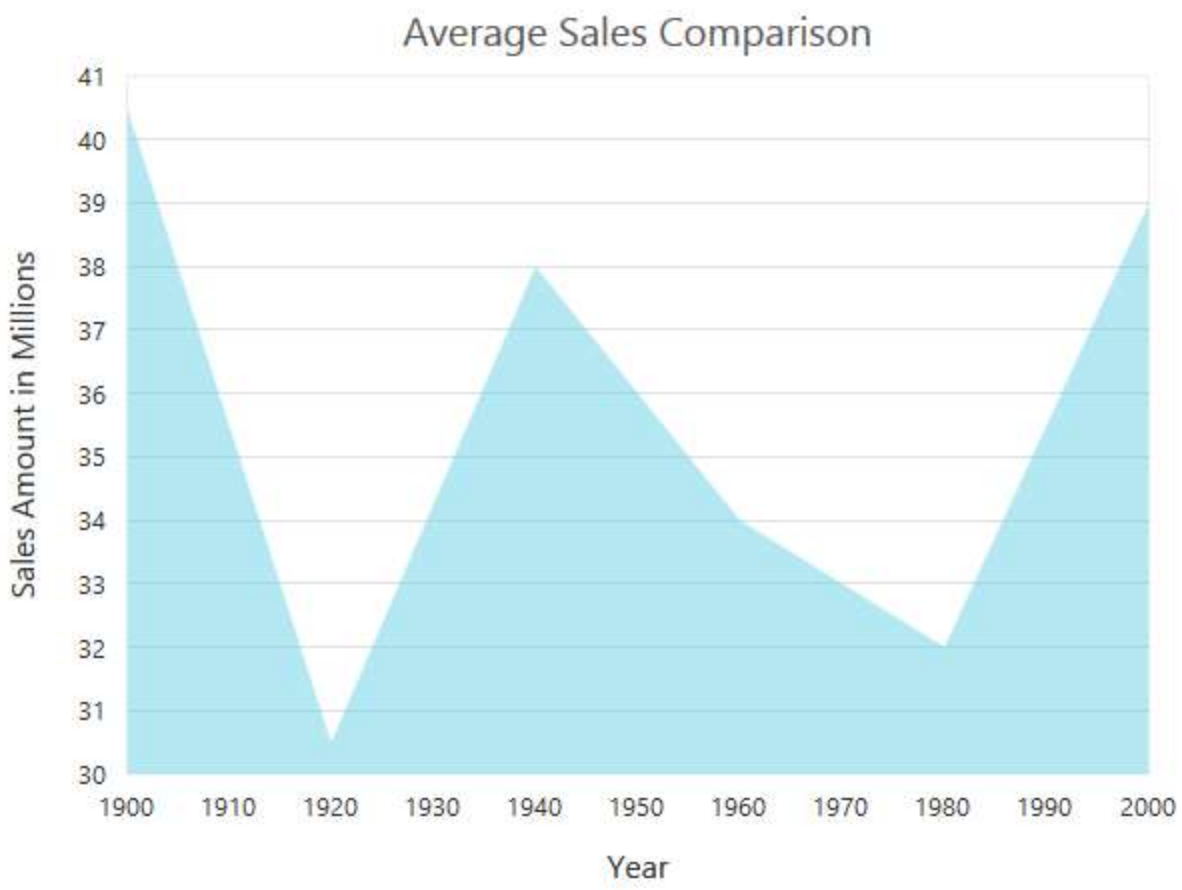


Chart after rounding axis range



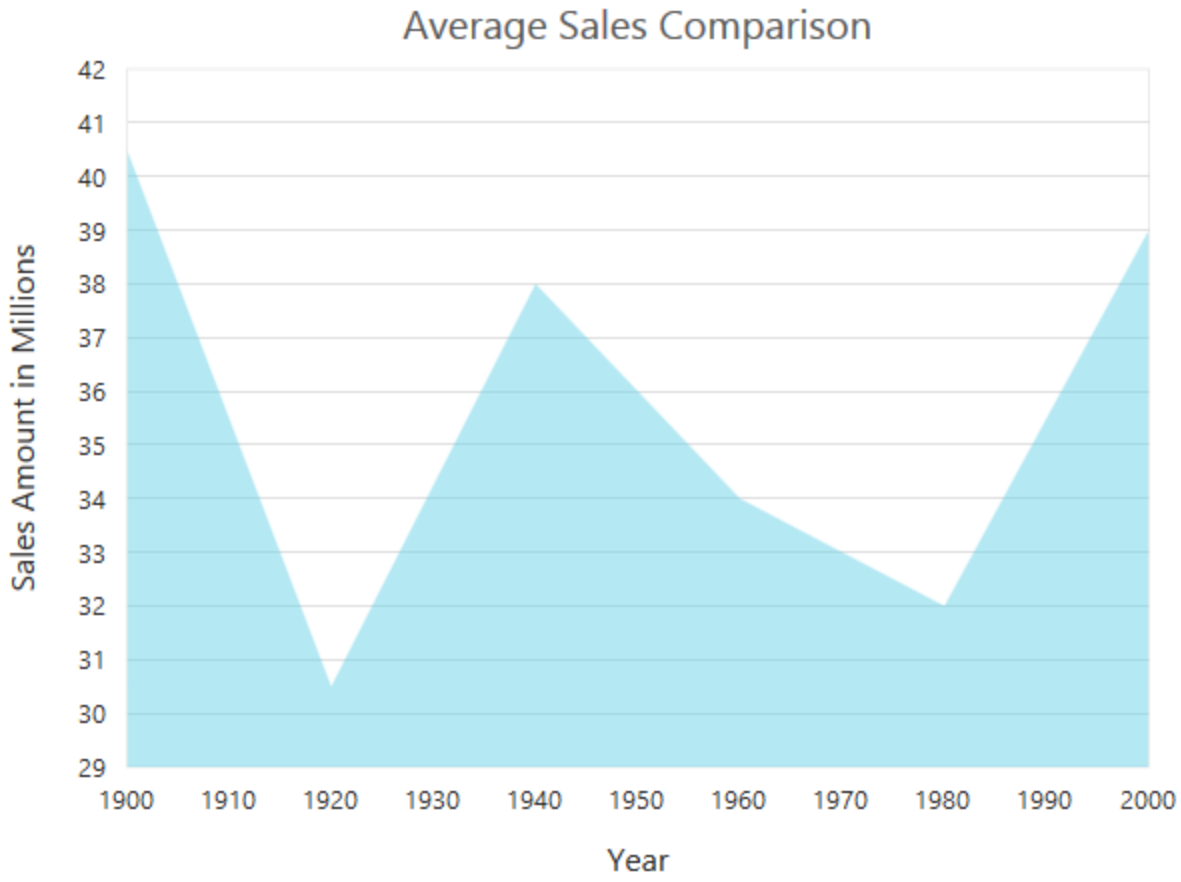
Additional

When the value of the RangePadding property is **Additional**, the axis range is rounded and an interval of the axis is added as padding to the minimum and maximum values of the range.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Applying additional range padding
px=>px.RangePadding(ChartRangePadding.Additional)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

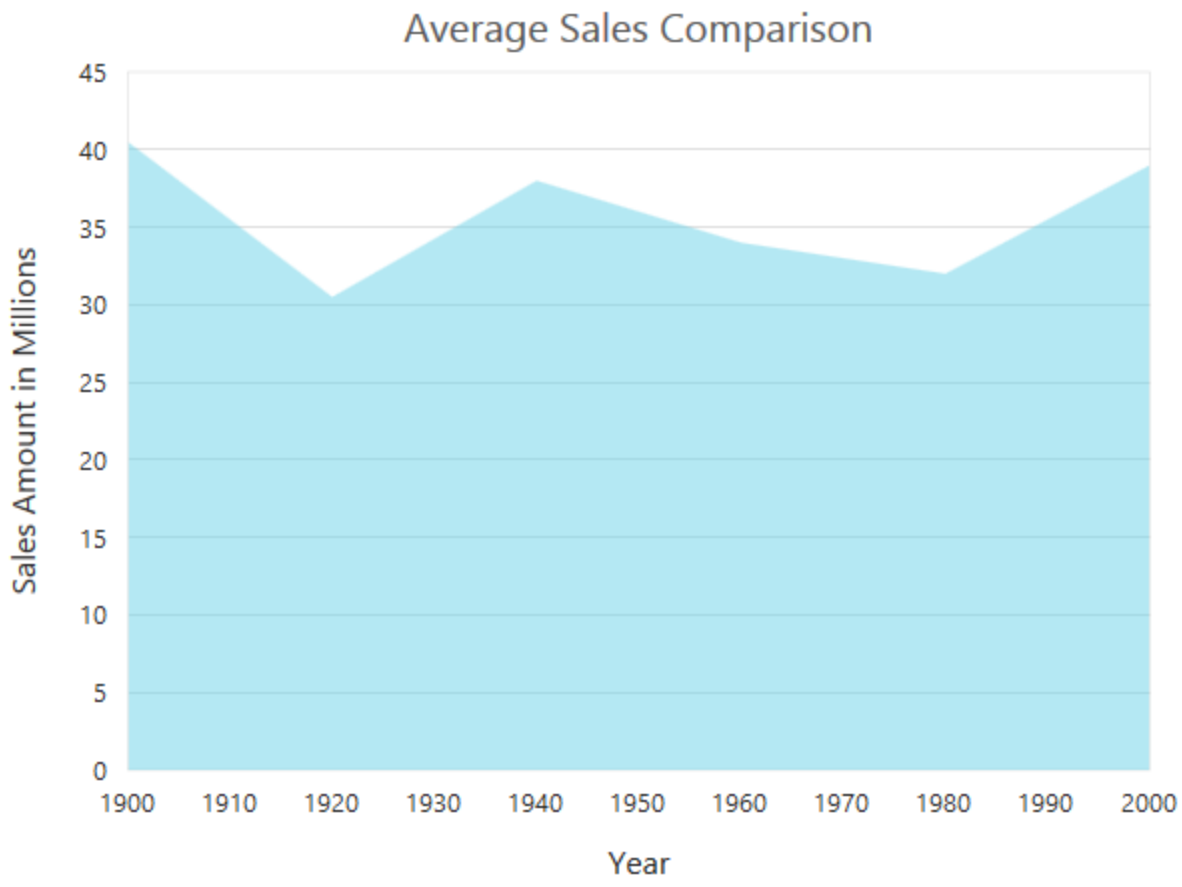


Normal

When the value of the RangePadding property is **Normal**, the padding is applied to the axis based on the range calculation.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Applying normal range padding
px=>px.RangePadding(ChartRangePadding.Normal)
)
// ...
)
```



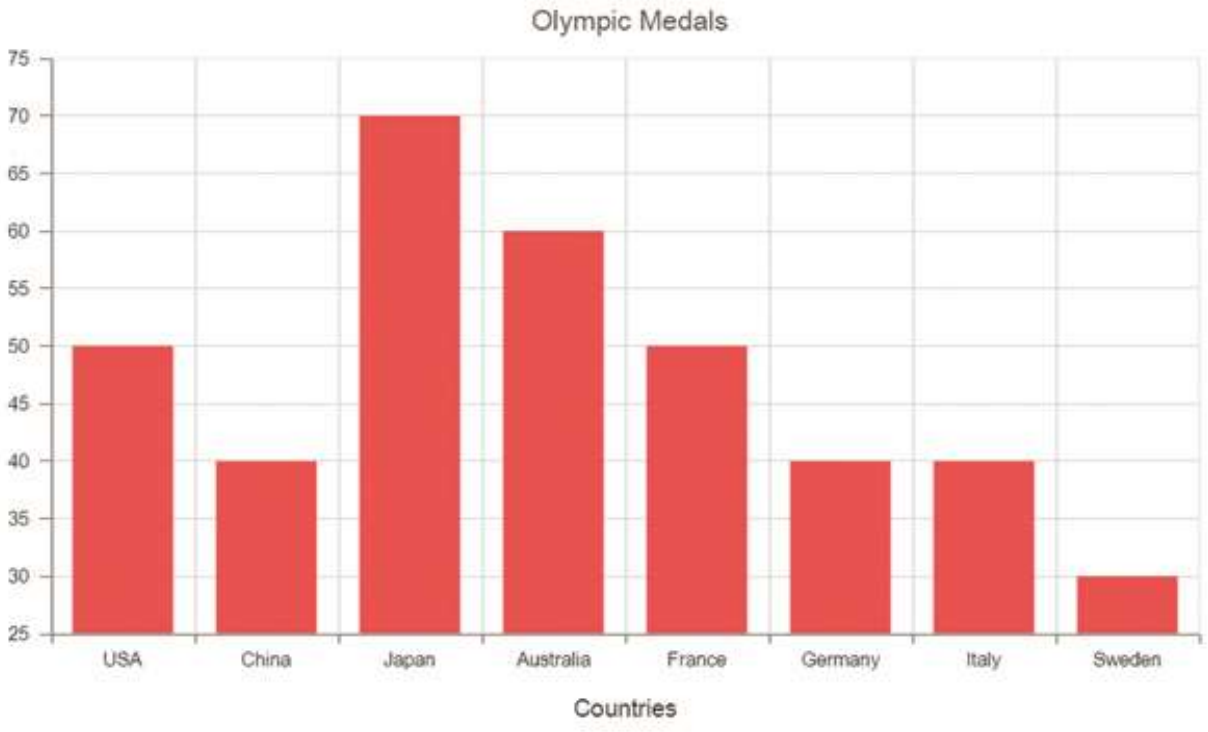
Customizing the starting range of the axis

By default the Y axis will be always calculated from the value 0 for column, bar, stacking column and stacking bar series types. You can modify this behavior by setting false to the property **StartFromZero** in the axis. On setting this the axis minimum value will be calculated based on the value for the data points.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryYAxis(axis =>
axis.RangePadding(ChartRangePadding.None).StartFromZero(false))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



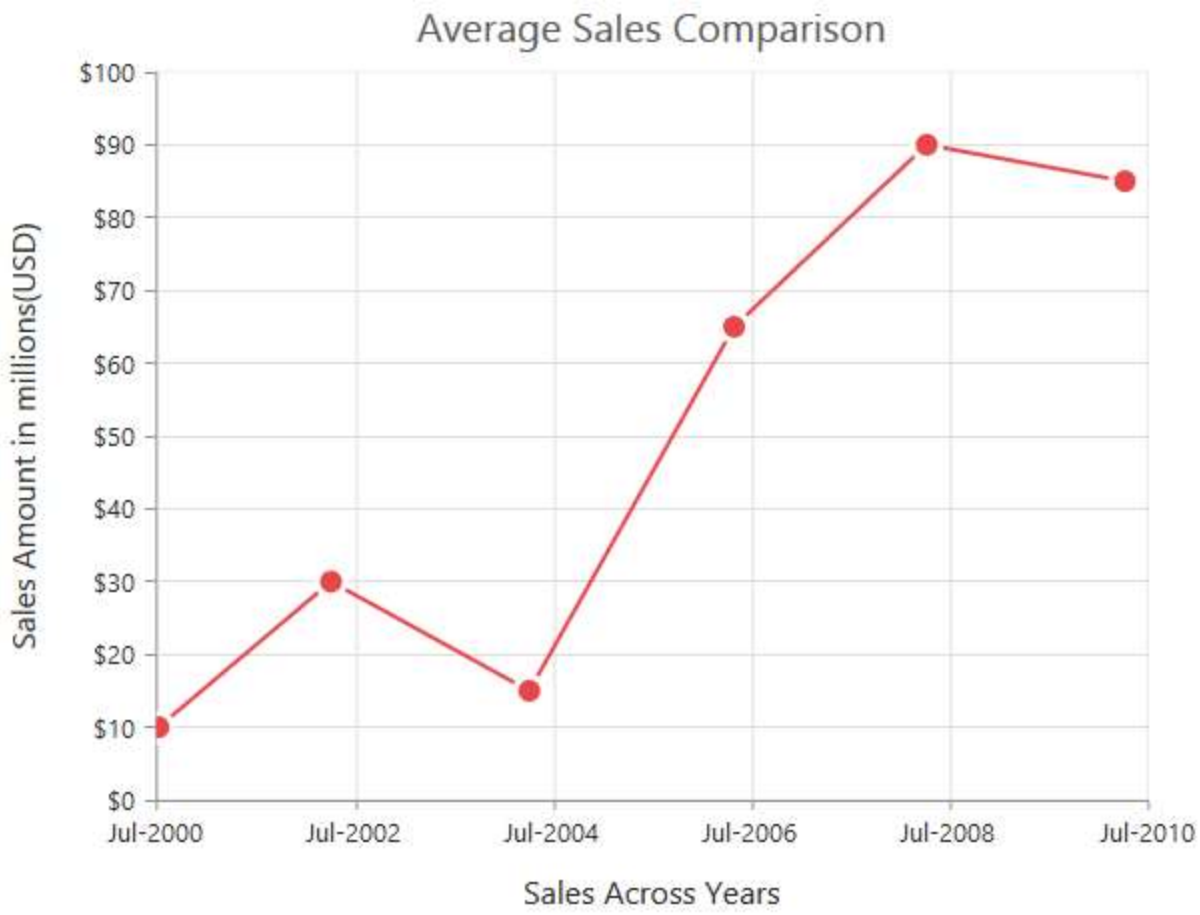
DateTime Axis

Date time axis uses date time scale and displays the date time values as axis labels in the specified format. To use date time axis, set the ValueType property of the axis to **Datetime**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Use date time scale in primary X axis
px=>px.ValueType(AxisValueType.Datetime)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



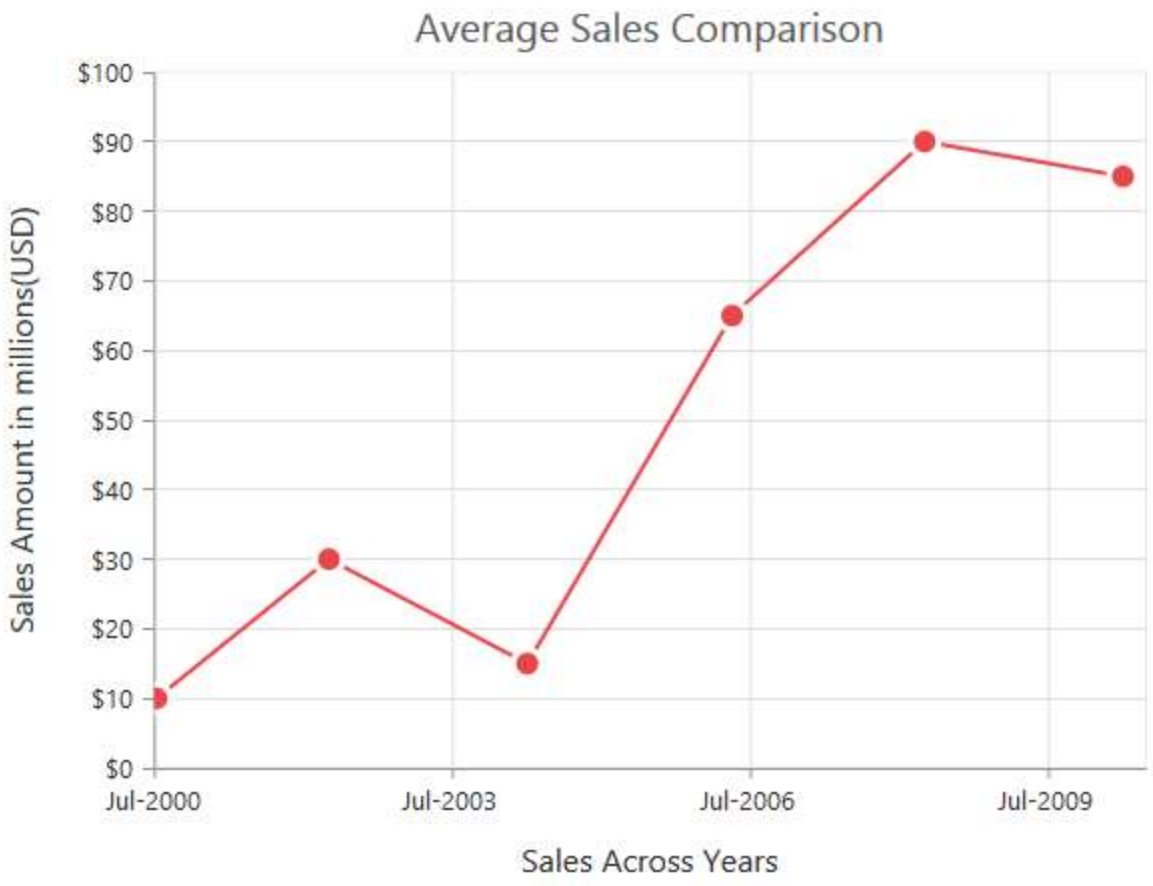
Click here to view our online demo sample for date time axis.

Customizing date time range

Axis range can be customized by using the Range property to set the **Minimum**, **Maximum** and **Interval** values. Nice range is calculated automatically based on the provided data, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing X-axis date time range
axis=>axis.Range(range=>range.Min("2000/6/1").Max("2010/6/1"))
)
// ...
)
```



Date time intervals

Date time intervals can be customized by using the **Interval** and **IntervalType** properties of the axis. For example, when you set Interval as **2** and IntervalType as **Years**, it considers the 2 years as interval.

Essential Chart supports the following types of interval for date time axis.

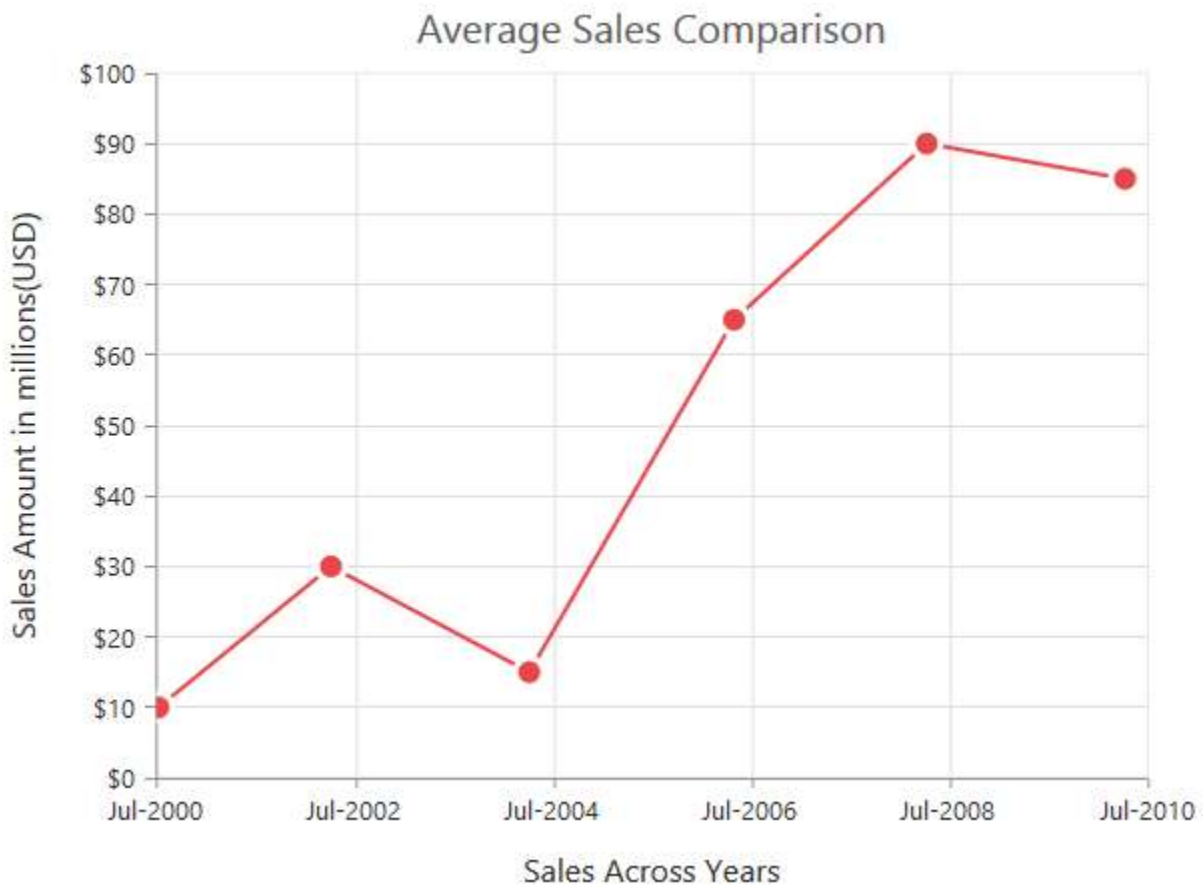
- Days
- Hours
- Milliseconds
- Minutes
- Months
- Seconds
- Years

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing X-axis date time range
axis=>axis.Range(range=>range.Interval(2)).IntervalType(ChartIntervalType.Ye
ars)
)
// ...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```



Apply padding to the range

Padding can be applied to the minimum and maximum extremes of the range by using the **RangePadding** property. Date time axis supports the following types of padding

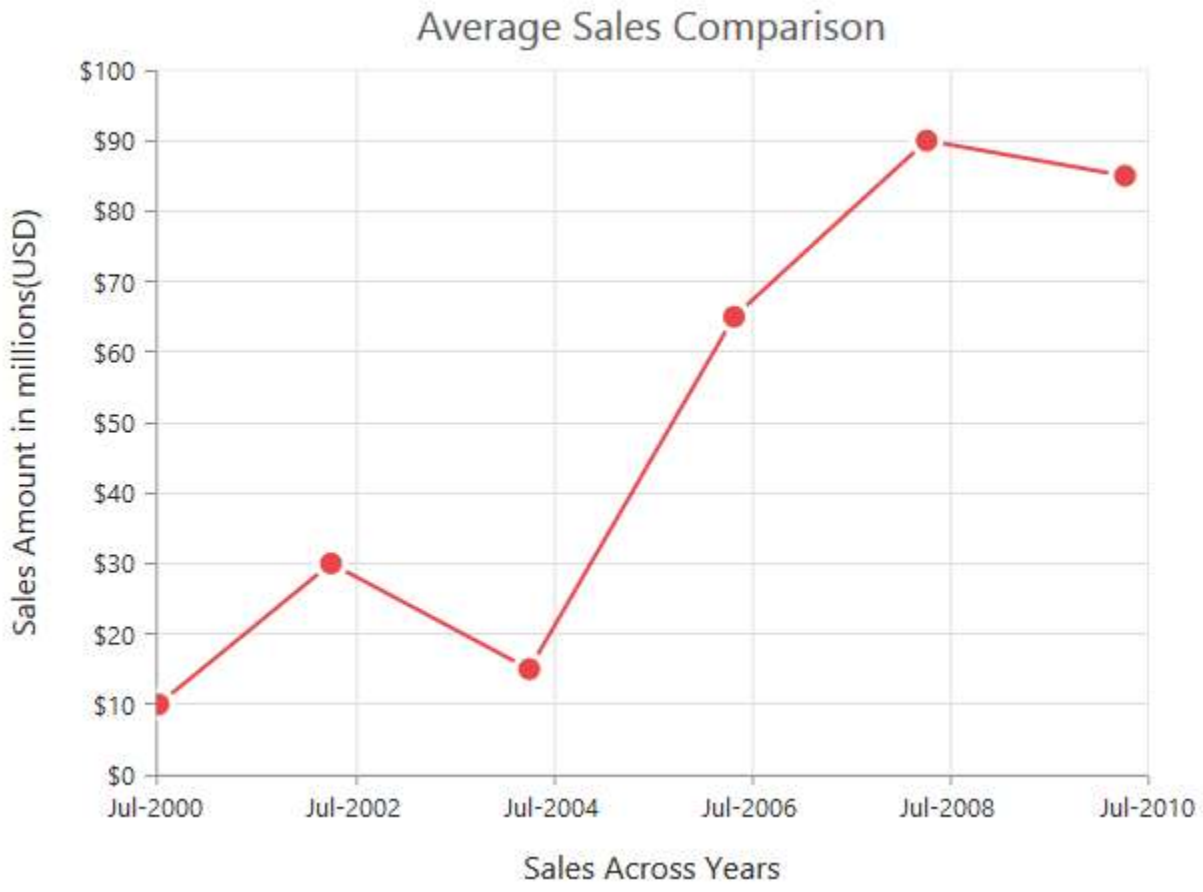
- None
- Round
- Additional

None

When the value of the RangePadding property is **None**, padding is applied to the axis. This is also the default value of the rangePadding.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Applying none as range padding
px=>px.RangePadding(ChartRangePadding.None)
)
// ...
)
```



Round

When the value of the RangePadding property is **Round**, the axis range is rounded to the nearest possible date time value.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Applying round as range padding
px=>px.RangePadding(ChartRangePadding.Round)
)
// ...
)
```

Chart before rounding axis range

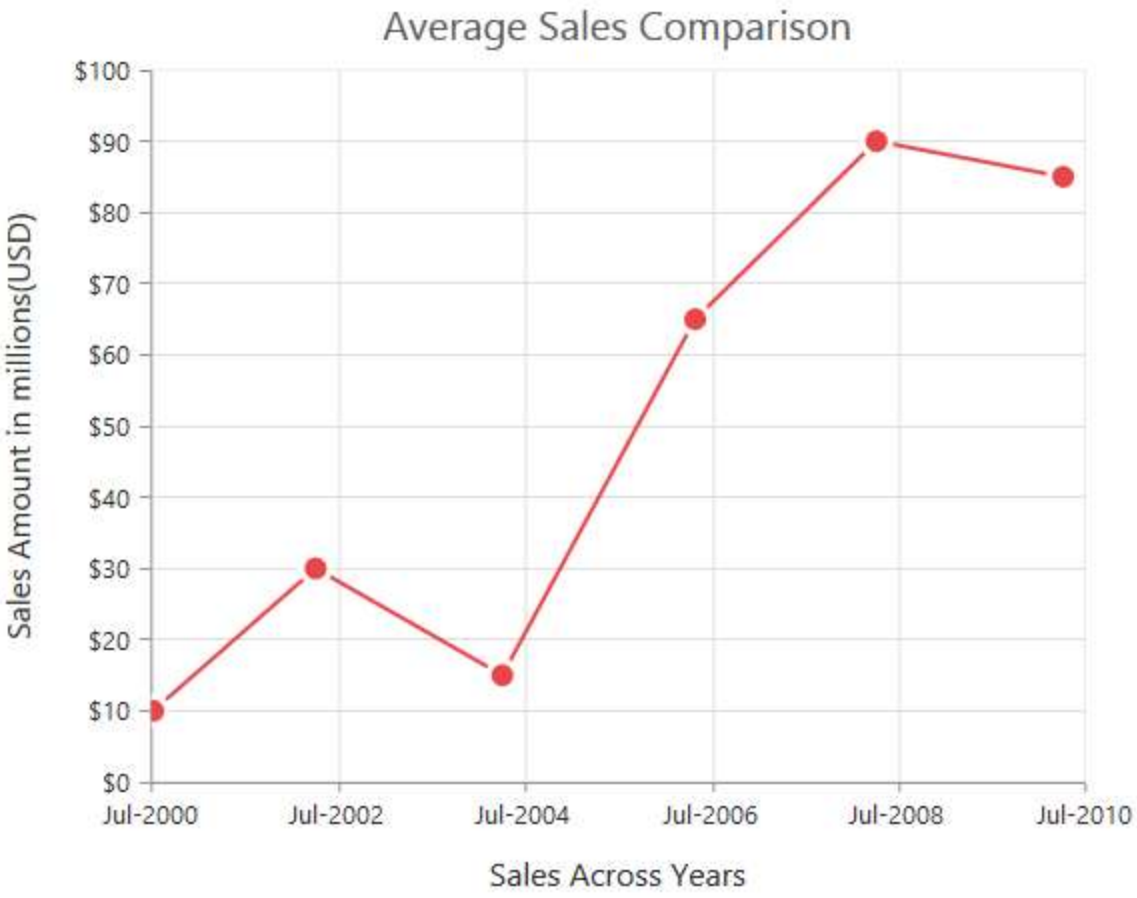
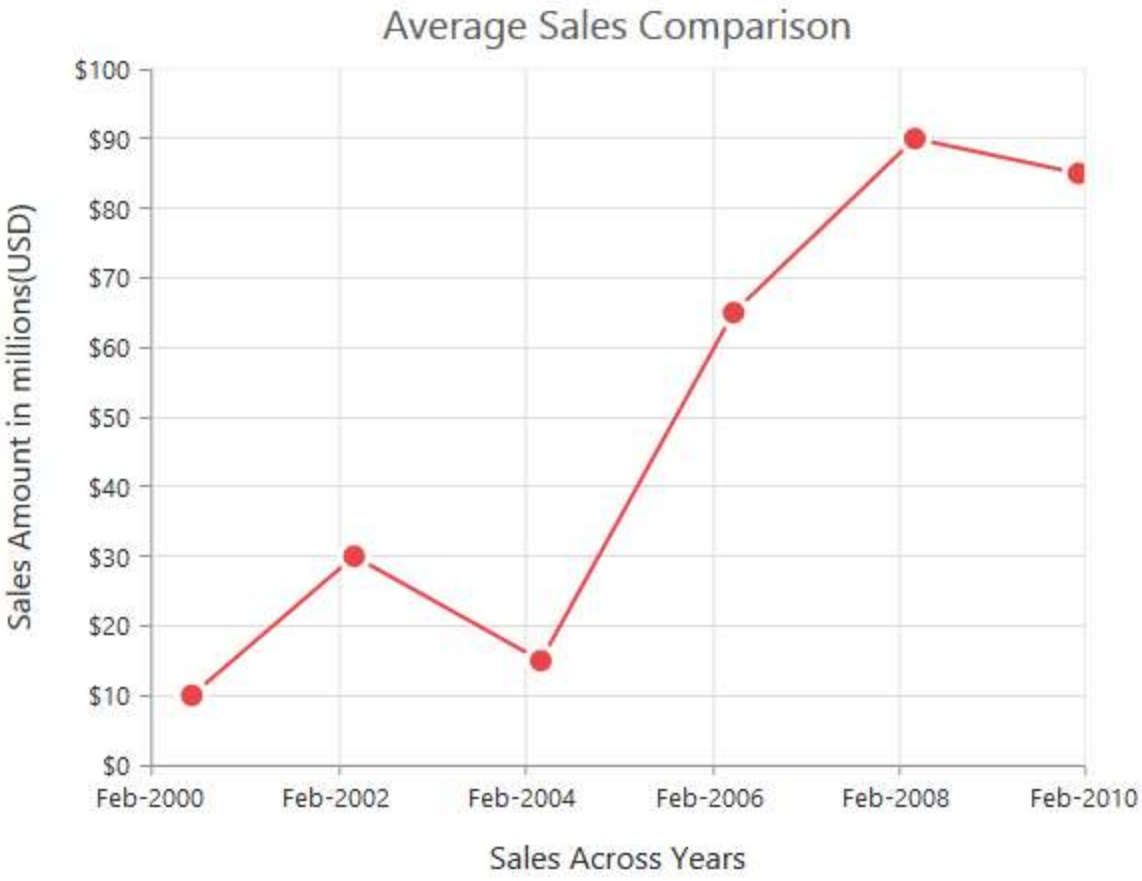


Chart after rounding axis range

Copyright © 2001 - 2020 Syncfusion Inc.



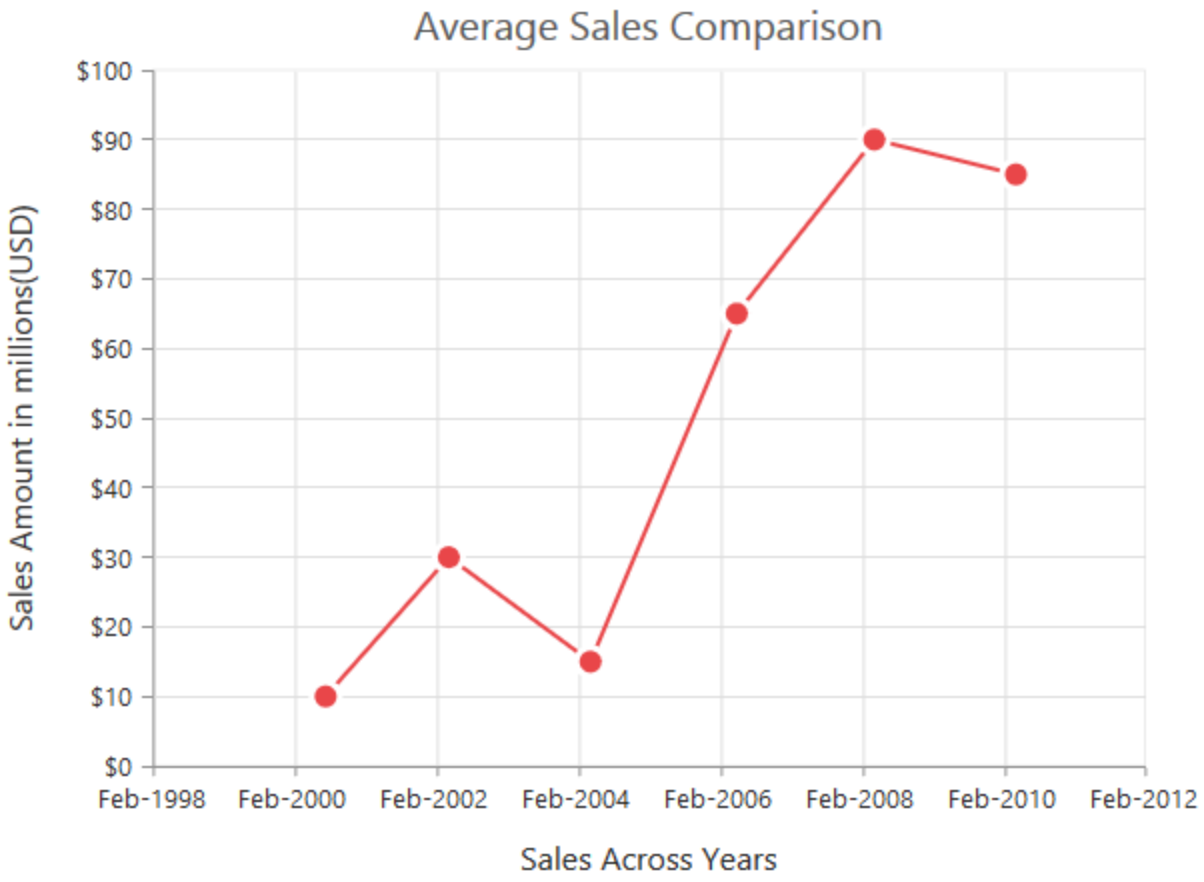
Additional

When the value of the RangePadding property is **Additional**, the range is rounded and date time interval of the axis are added as padding to the minimum and maximum extremes of the range.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Applying additional as range padding
px=>px.RangePadding(ChartRangePadding.Additional)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



DateTime Category Axis

DateTime category axis takes date time value as input but behaves like category axis. This is used to display the date time values with nonlinear intervals (used to depict the business days by skipping holidays). To use date time axis, set the **ValueType** property of the axis to **DatetimeCategory**.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis => axis.ValueType(AxisValueType.DateTimeCategory))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view our online demo sample for date time axis.

Customizing DateTime Category range

Axis range can be customized by using the *range* property to set the **Minimum**, **Maximum** and **Interval** values. Datetime category axis takes numeric input for minimum and maximum property.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(
//Customizing X axis date time category range
axis => axis.Range(range => range.Min(0).Max(4)))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



DateTime Category intervals

Date time category intervals can be customized by using the **Interval** and **IntervalType** properties of the axis. For example, when you set the intervalType as months, it displays only the first label of all the months from the data.

Essential Chart supports the following types of interval for date time category axis.

- Days
- Hours
- Milliseconds
- Minutes
- Months
- Seconds
- Years
- Auto

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis => axis.IntervalType(ChartIntervalType.Months))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



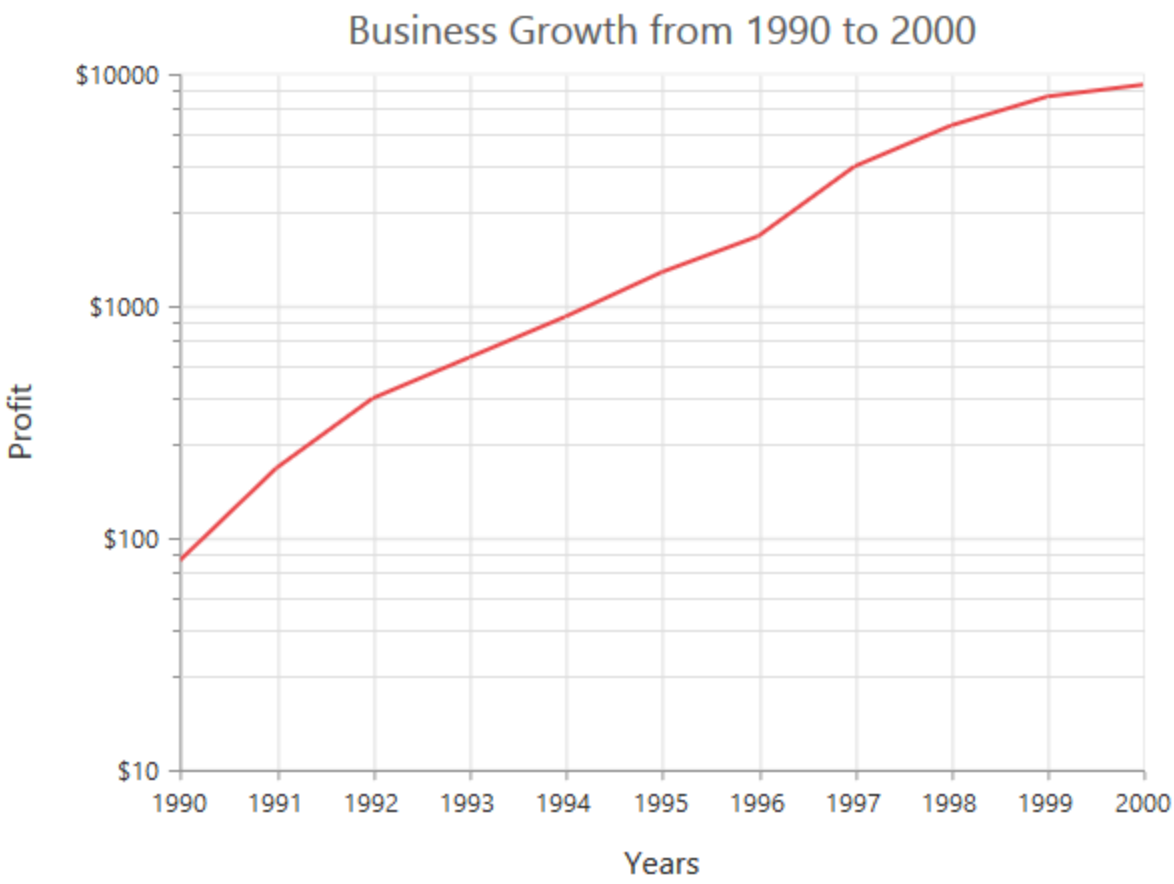
Logarithmic Axis

Logarithmic axis uses logarithmic scale and it is very useful in visualizing when the data has values with both lower order of magnitude **(eg: 10⁻⁶)** and higher order of magnitude **(eg: 10⁶)**. To use logarithmic axis, set the ValueType property of the axis to **Logarithmic**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Use logarithmic scale in primary X axis
px=>px.ValueType(AxisValueType.Logarithmic)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



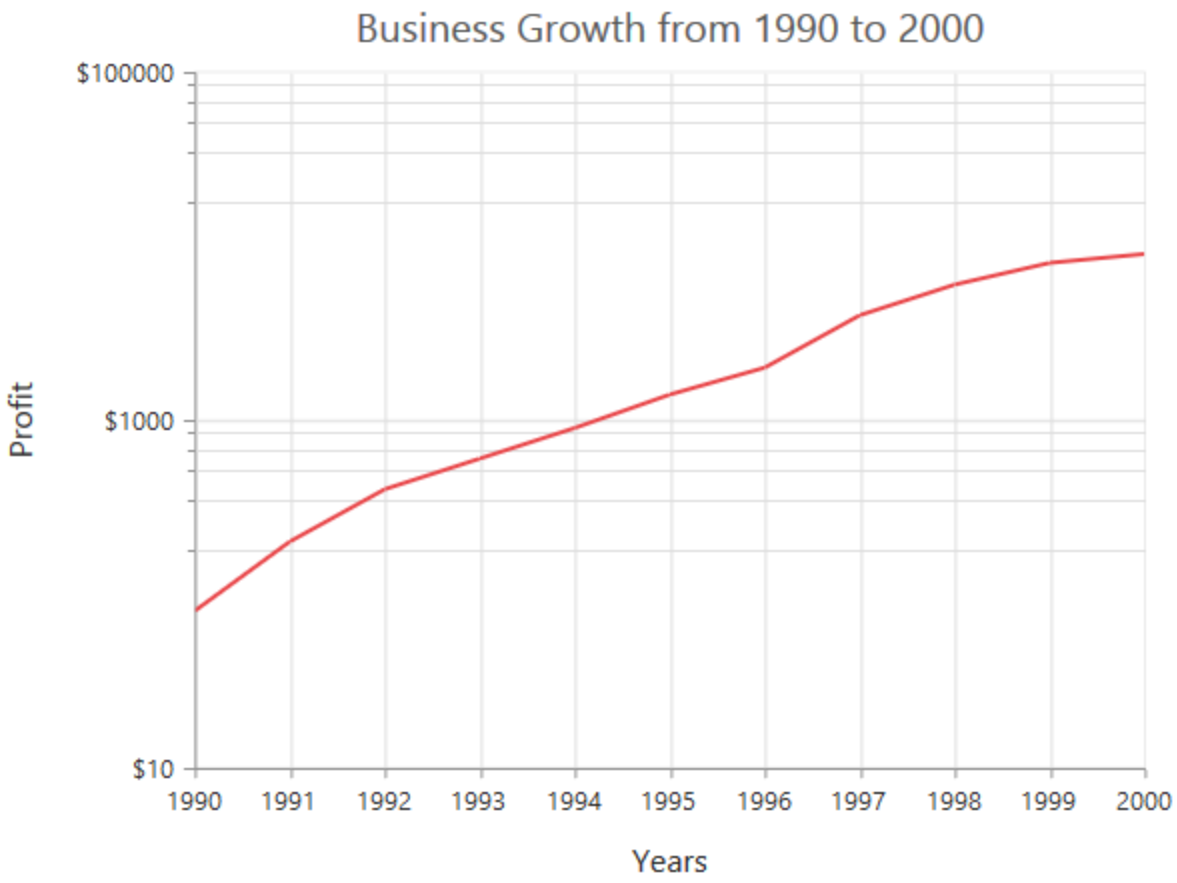
Click here to view our online demo sample link for logarithmic axis.

Customize Logarithmic range

Logarithmic range can be customized by using the Range property of the axis to change the Minimum, Maximum and Interval values. Nice range is calculated automatically based on the provided data, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Customizing logarithmic range
px=>px.Range(range=>range.Min(1).Max(5))
)
// ...
)
```



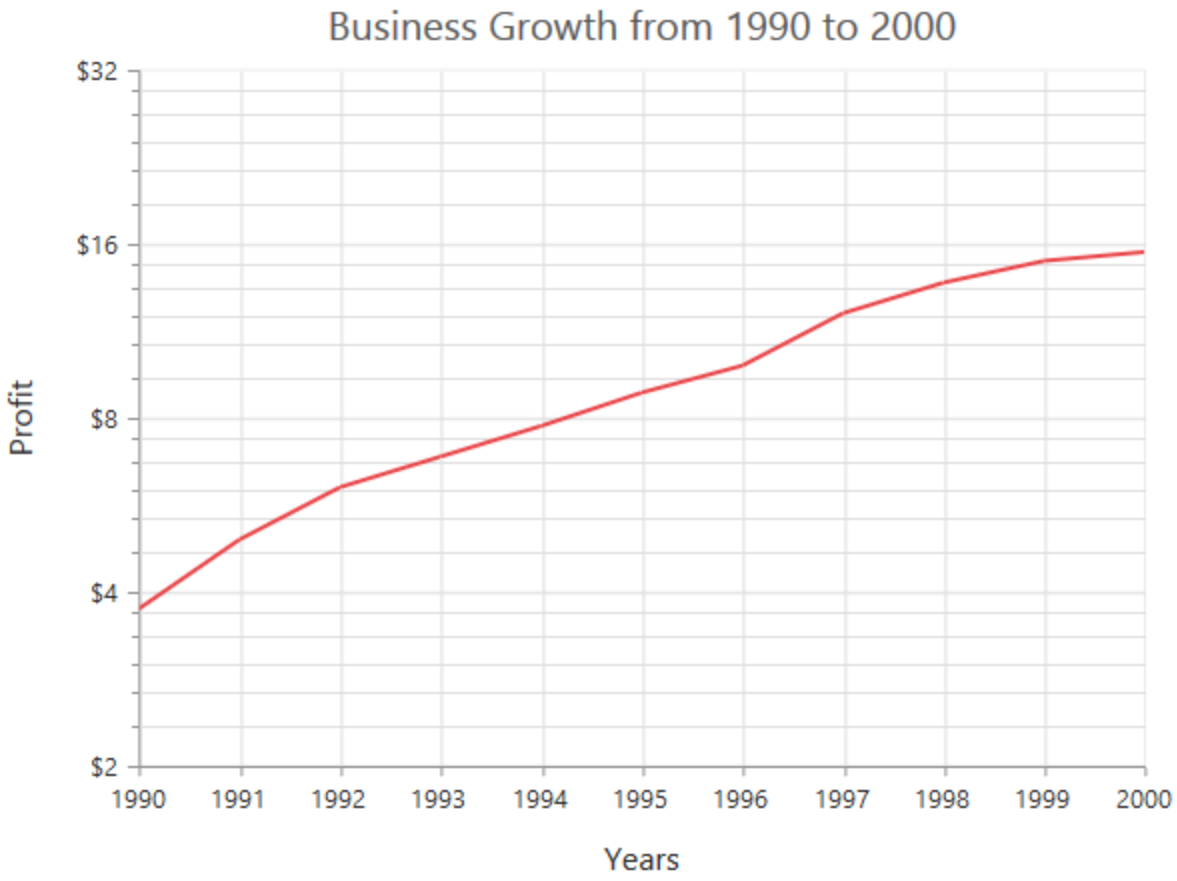
Logarithmic base

Logarithmic base can be customized by using the **LogBase** property of the axis. The default value of the LogBase is **10**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Customizing logarithmic base
px=>px.LogBase(2)
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



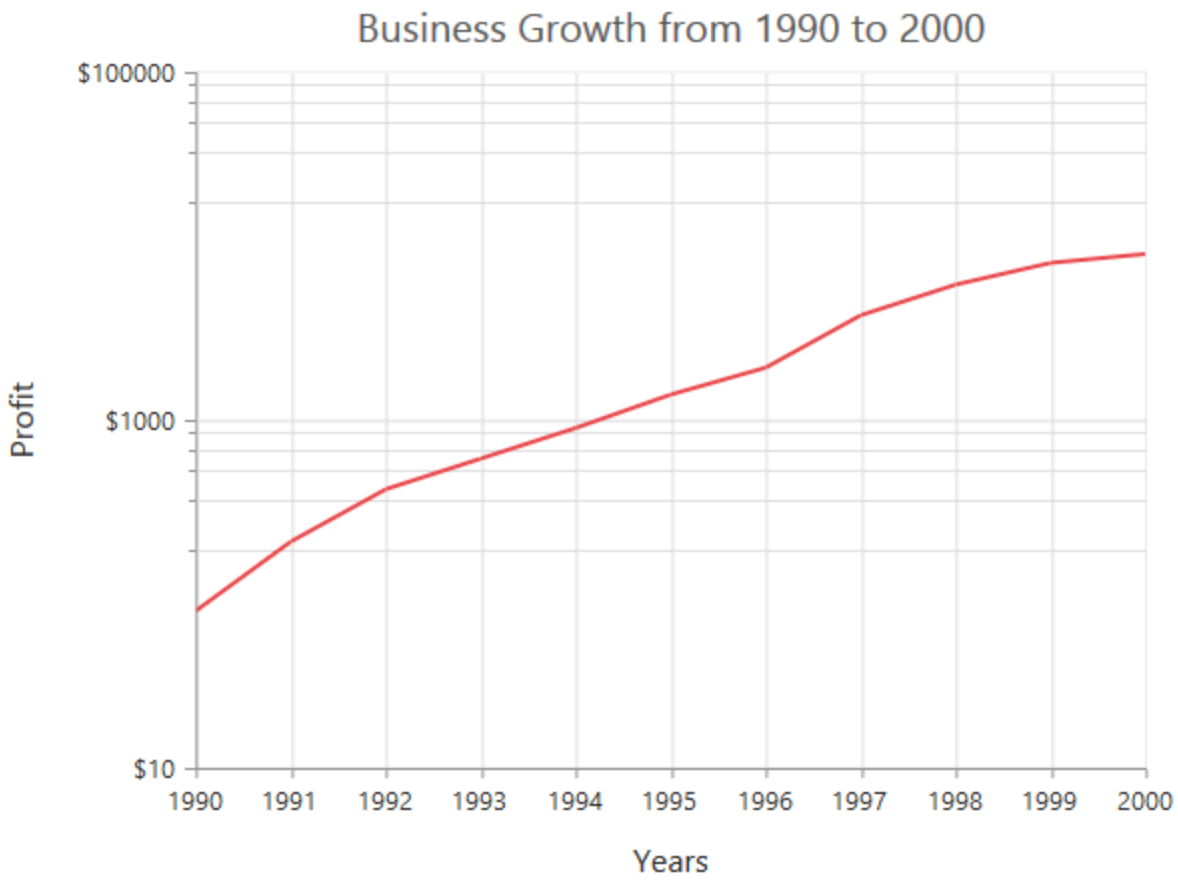
Logarithmic interval

Logarithmic axis interval can be customized by using the Interval property of the axis. When the logarithmic base is 10 and logarithmic interval is 2, then the axis labels are placed at an interval of 10². The default value of the interval is 1.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Customizing logarithmic interval
px=>px.Range(range=>range.Interval(2))
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Label Format

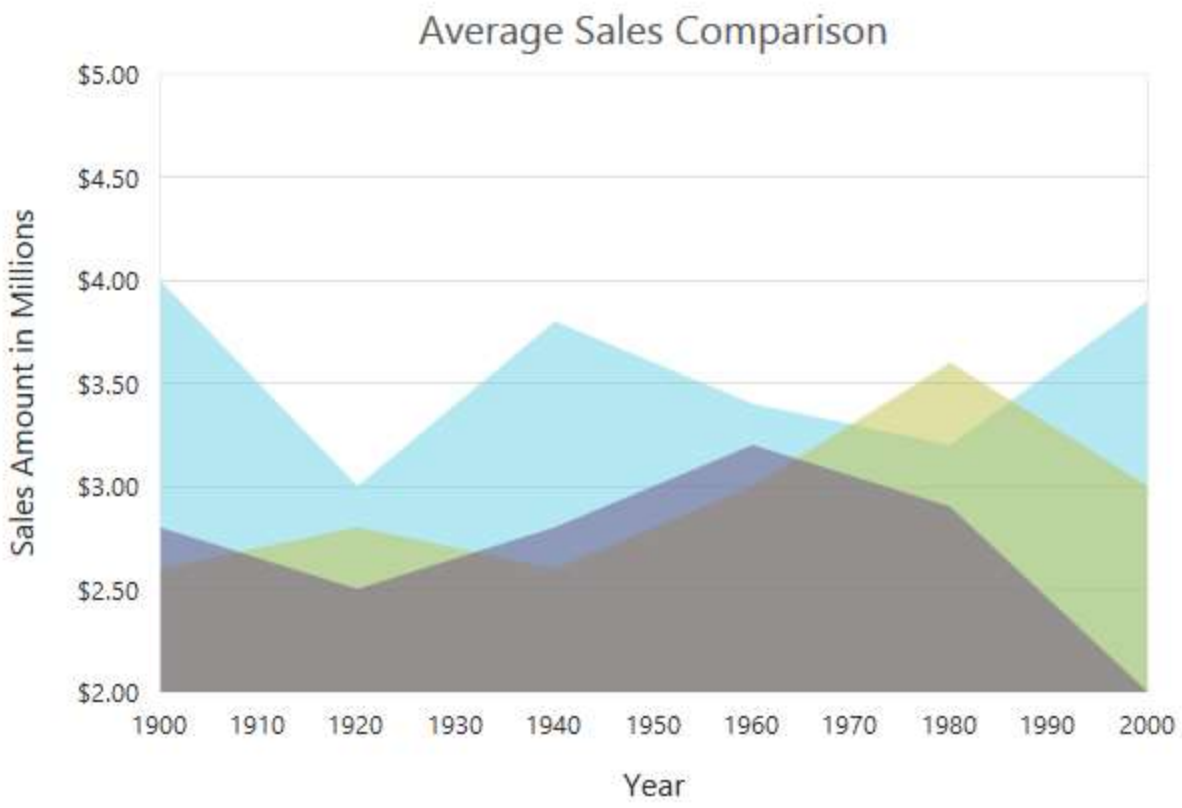
Format numeric labels

Numeric labels can be formatted by using the **LabelFormat** property. Numeric values can be formatted with n (number with decimal points), c (currency) and p (percentage) commands.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Applying currency format to axis labels
px=>px.LabelFormat("c")
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



The following table describes the result of applying some commonly used label formats on numeric values.

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

Copyright © 2001 - 2020 Syncfusion Inc.

Format date time values

Date time labels can be formatted by using the **LabelFormat** property of the axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Formatting date time labels in date/Month name/Year format
px=>px.LabelFormat("dd/MMMM/yyyy")
//....
)
// ...
)
```



The following table describes the result of applying some common date time formats to the labelFormat property

Label Value	Label Format Property Value	Result	Description
new Date(2000, 03, 10)	dddd	Monday	The Date is displayed in day format
new Date(2000, 03, 10)	MM/dd/yyyy	04/10/2000	The Date is displayed in month/date/year format

Copyright © 2001 - 2020 Syncfusion Inc.

new Date(2000, 03, 10)	n3	1000.000	The Number is rounded to 3 decimal place
new Date(2000, 03, 10)	MMM	Apr	The Shorthand month for the date is displayed
new Date(2000, 03, 10)	t	12:00 AM	Time of the date value is displayed as label
new Date(2000, 03, 10)	hh:mm:ss	12:00:00	The Label is displayed in hours:minutes:seconds format

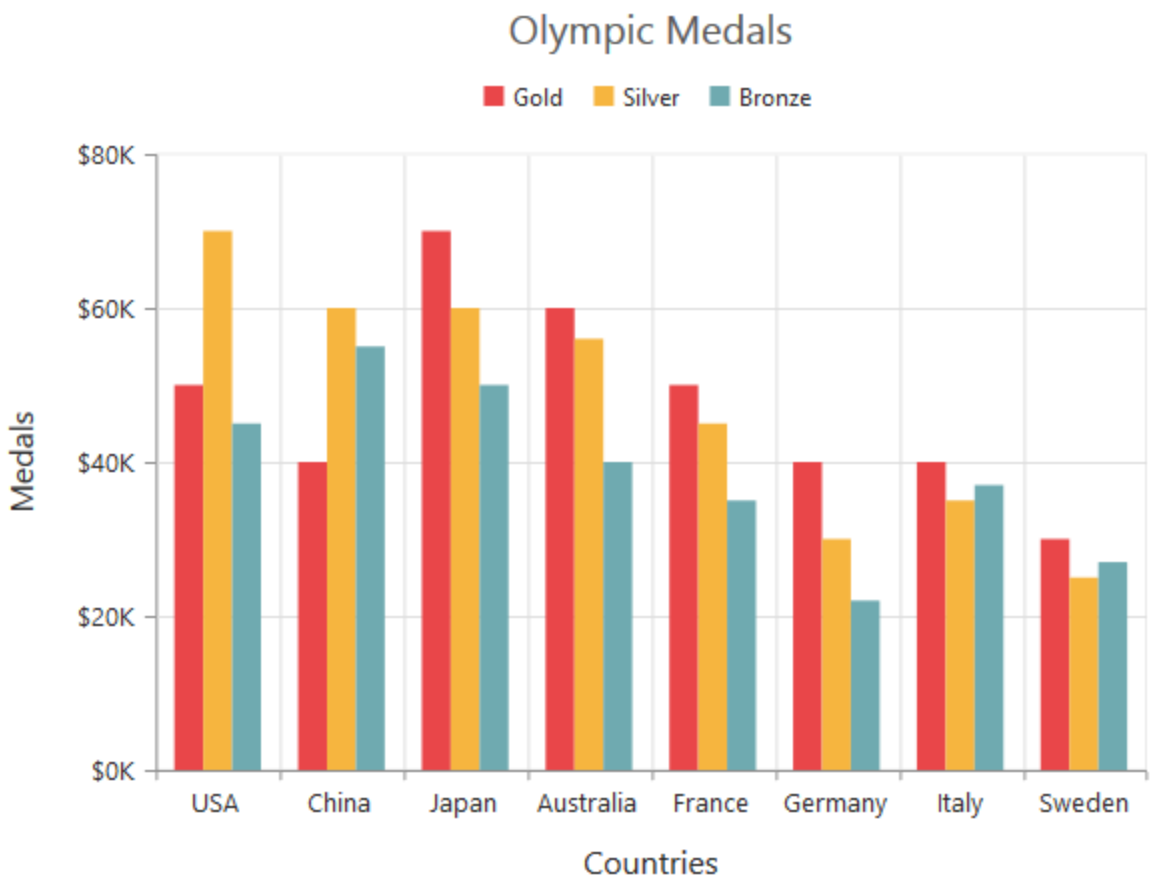
Custom label format

Prefix and suffix can be added to the category labels by using the LabelFormat property. You can use the *{value}* as placeholder text in your custom text, it is replaced with the corresponding axis label at the runtime.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Adding prefix and suffix to axis labels
px=>px.LabelFormat("${value} K")
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Common axis features

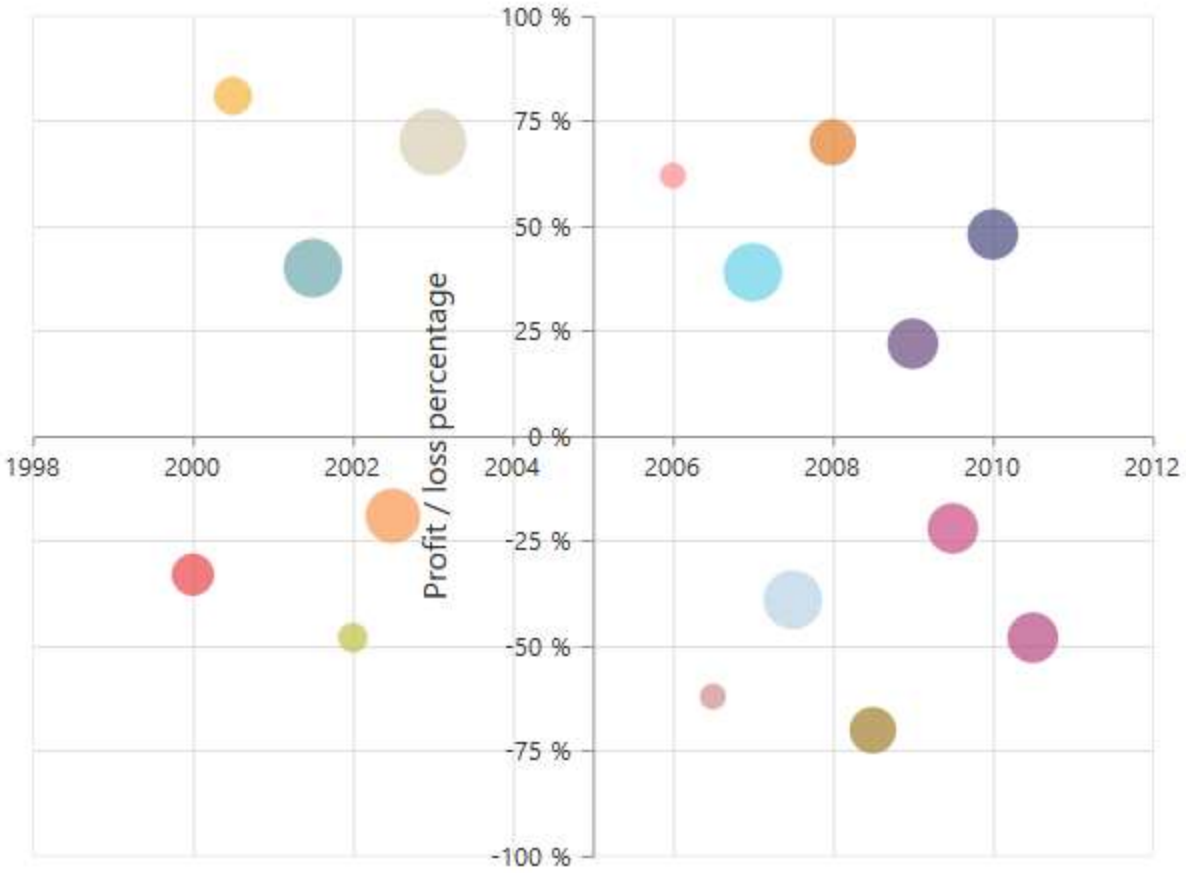
Customization of features such as axis title, labels, grid lines and tick lines are common to all the axis. Each of these features are explained in this section.

Axis Crossing

Axis can be positioned anywhere in chart area using the **CrossesAt** property of axis. This property specifies where the horizontal axis should intersect or cross the vertical axis and vice versa. Default value of *CrossesAt* property is null.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Crosses primary Y axis at 0
px=>px.CrossesAt(0)
//....
)
// ...
)
```



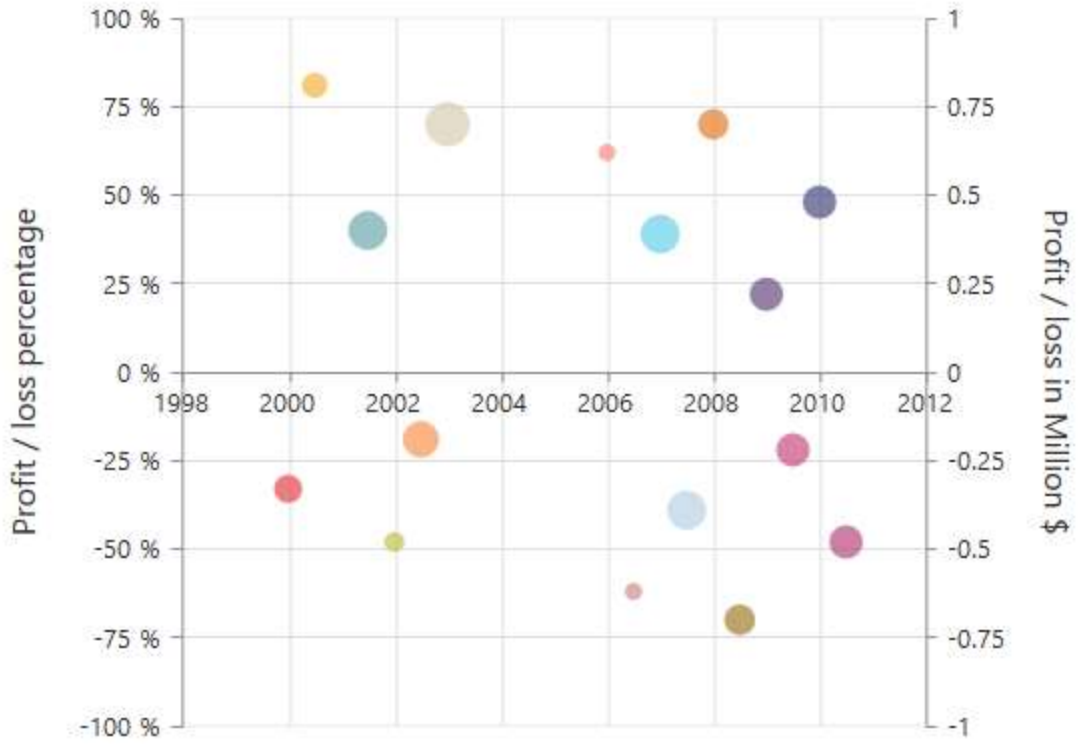
Crossing a specific Axis

The **CrossesInAxis** property takes axis name as input and determines the axis used for crossing. By default all the horizontal axes crosses in primary Y axis and all the vertical axes crosses in primary X axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Crosses secondary vertical axis at -0.2
px=>px.CrossesAt(-0.2)
.CrossesInAxis('SecondaryYAxis')
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

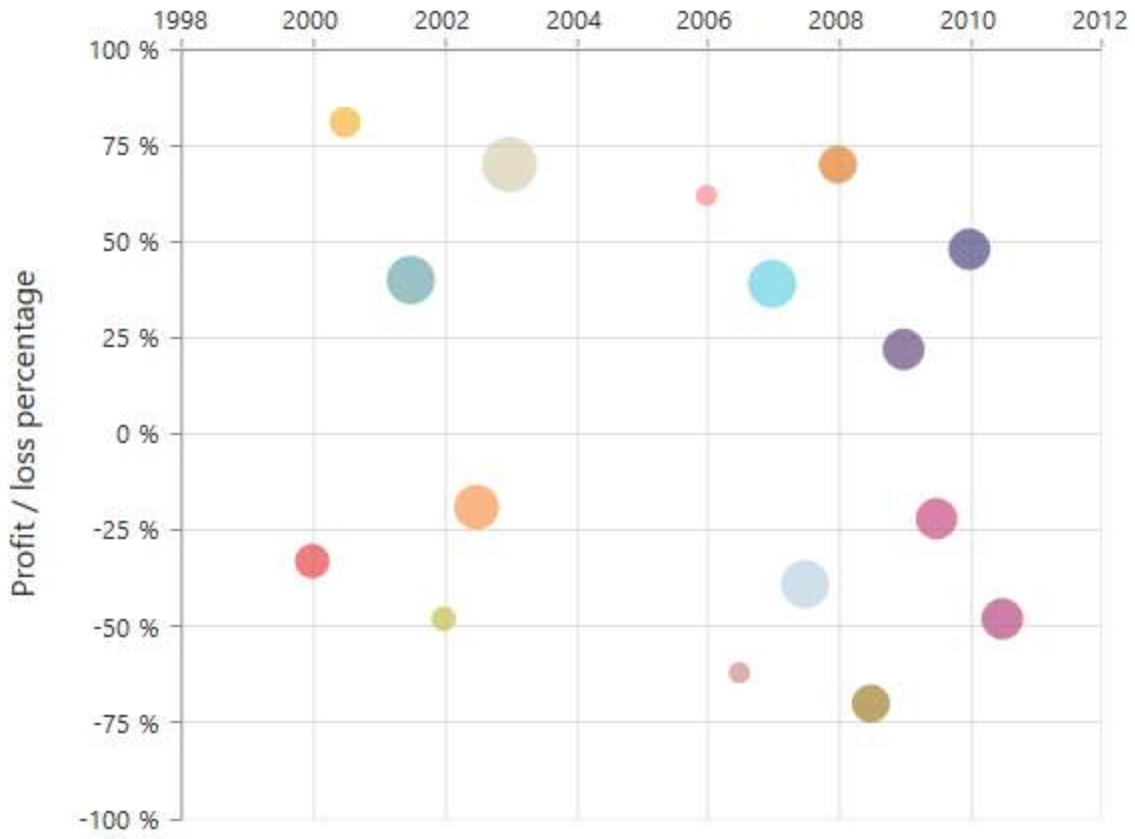


Axis will be placed in the opposite side if value of *CrossesAt* property is greater than the maximum value of crossing axis (axis name provided through *CrossesInAxis* property or primary Y axis for horizontal axis).

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Crosses primary Y axis at 200
px=>px.CrossesAt(200)
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Positioning the axis elements while crossing

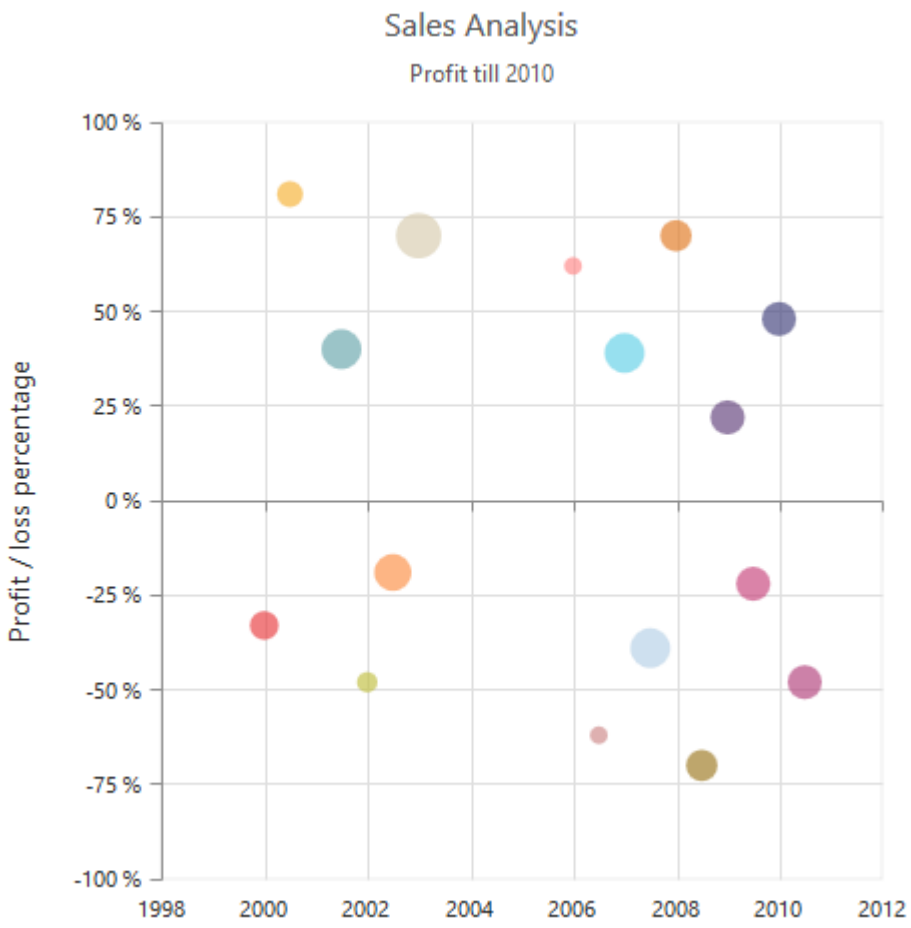
The ShowNextToAxisLine property is used for controlling the axis elements movement along with the axis line while axis crossing is performed. When the showNextToAxisLine is set as false only the axis line and the tick lines are placed at the crossing Value and the axis elements will be placed outside the chart area. The default value of ShowNextToAxisLine is **true**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Crosses primary Y axis at 0
px=>px.CrossesAt(0)
.ShowNextToAxisLine(false)
//....
)
// ...
)
```

The axis is placed at the crossing value without the axis elements

Copyright © 2001 - 2020 Syncfusion Inc.



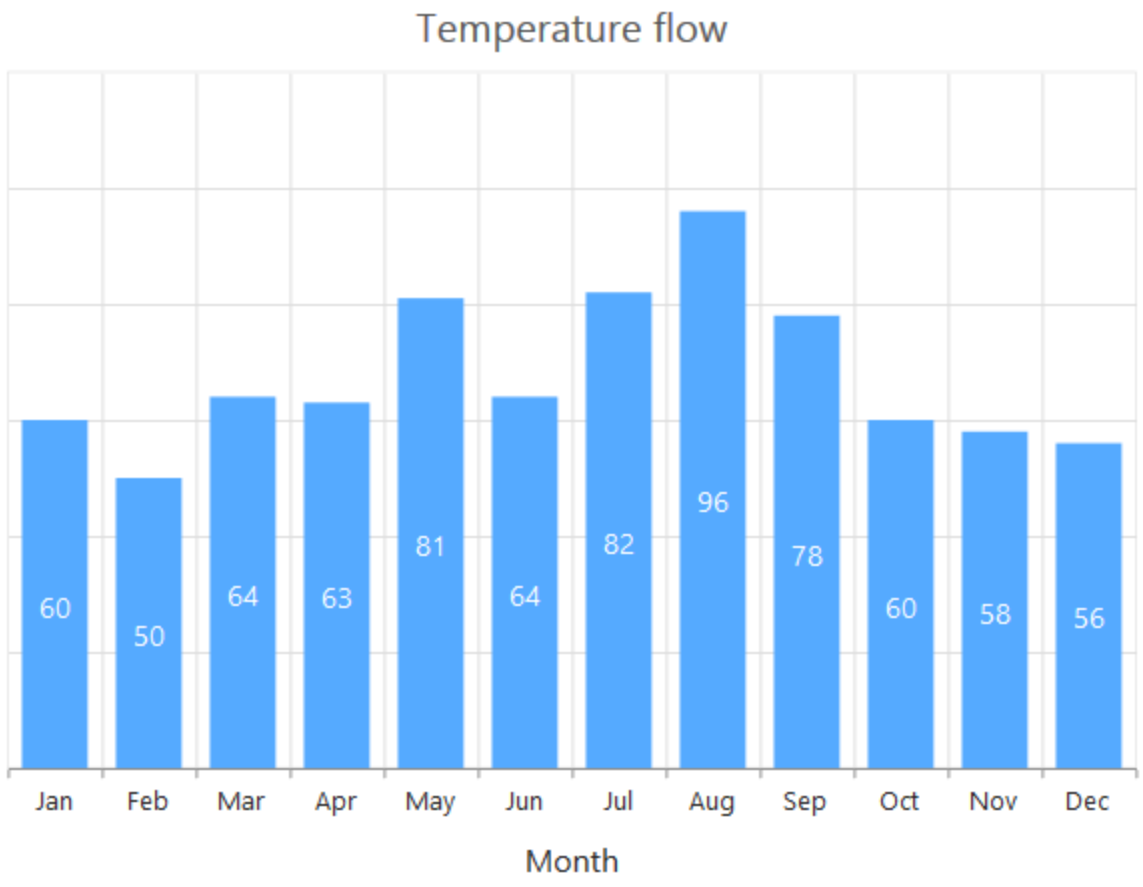
Axis Visibility

Axis visibility can be controlled by using the **Visible** property of the axis. The default value of the Visible property is **true**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(
//Disabling visibility of Y-axis
px=>px.Visible(false)
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



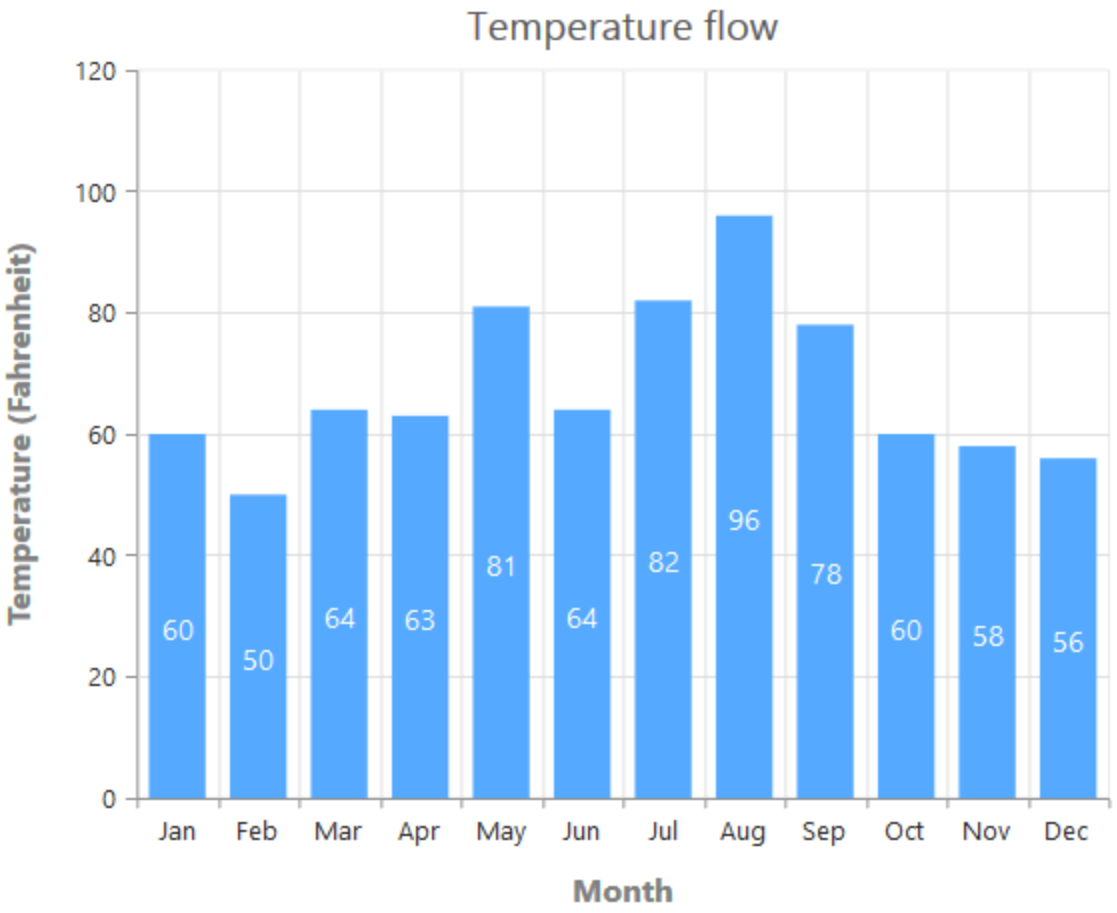
Axis title

The **Title** property in the axis provides options to customize the text and font of the axis title. Axis does not display the title, by default. Title text can also be trimmed based on the title text length or specified length.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing axis title
px=>px.Title(title=>title.EnableTrim(true)
.MaximumTitleWidth(80)
.Text("Month")
.Font(ft=>ft.Color("grey").
FontWeight(ChartFontWeight.Bold)
.FontFamily("Segoe UI")
.Size("16px")))
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

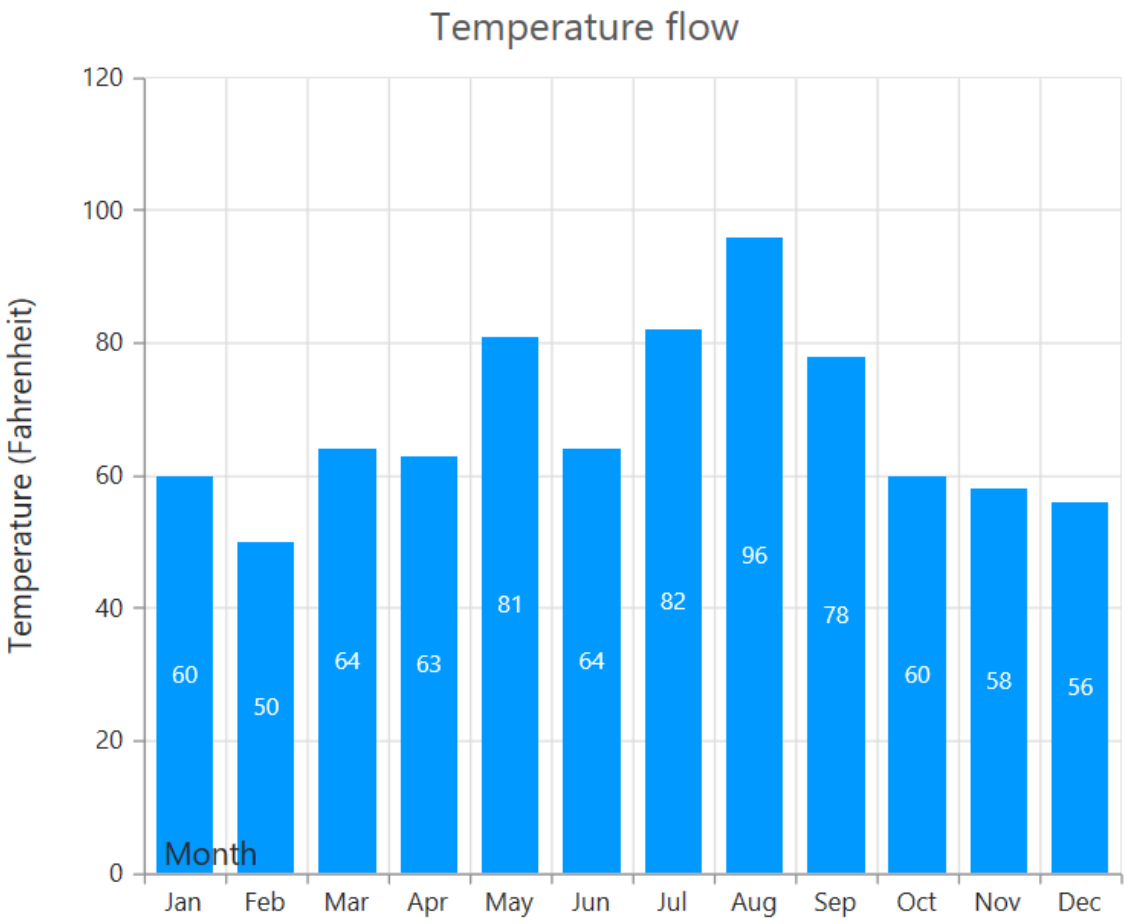


You can modify the position of the axis title either inside or outside the chart area using the property **Position**. By default, it will be placed outside the chart area. In addition, you can also change the alignment of the title to near, far and center by **Alignment** property, using **Offset** property you can change the position with respect to pixels.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(pr => pr.Title(title => title
.AxisTitlePosition(AxisTitlePosition.Inside).Alignment(Alignment.Near).Offse
t(10))
))
```

Copyright © 2001 - 2020 Syncfusion Inc.

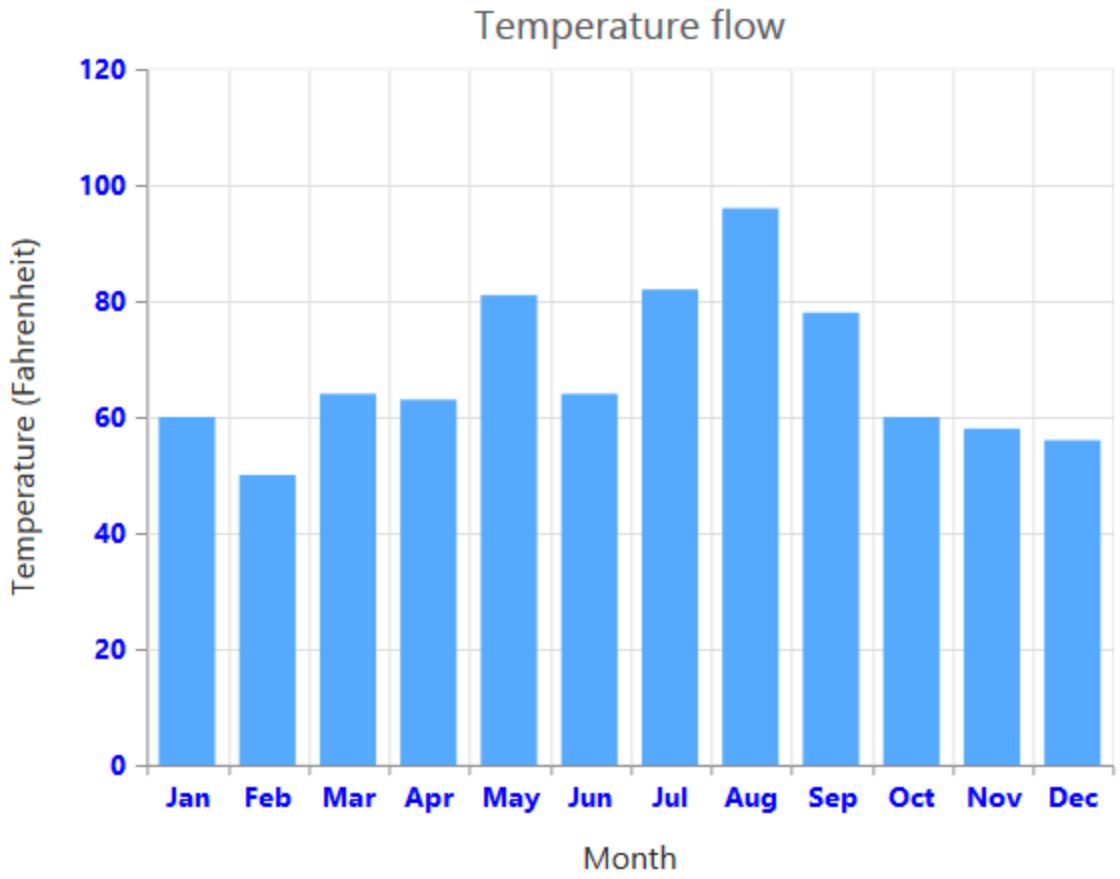


Label customization

The **Font** property of the axis provides options to customize the FontFamily, Color, Opacity, Size and FontWeight of the axis labels.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing label appearance
px=>px.Font(ft=>ft.Color("blue").
FontWeight(ChartFontWeight.Bold)
.FontFamily("Segoe UI")
.Size("14px"))
//....
)
// ...
)
```



Axis Labels Line Break

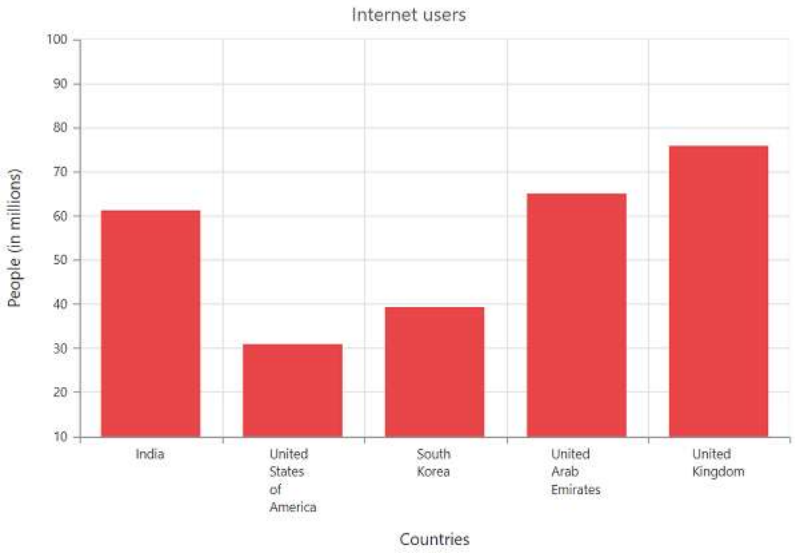
Axis Labels can be placed in multiple lines by specifying **
** for data points x value and in label format.

For category value type, **
** can be specified in x value of data points.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
sr
.Points(pt =>
{
pt.X("India").Y(61.3).Add();
pt.X("United<br>States<br>of<br>America").Y(31).Add();
pt.X("South<br>Korea").Y(39.4).Add();
pt.X("United<br>Arab<br>Emirates").Y(65.1).Add();
pt.X("United<br>Kingdom").Y(75.9).Add();
}).Add();
})
//...
)
```

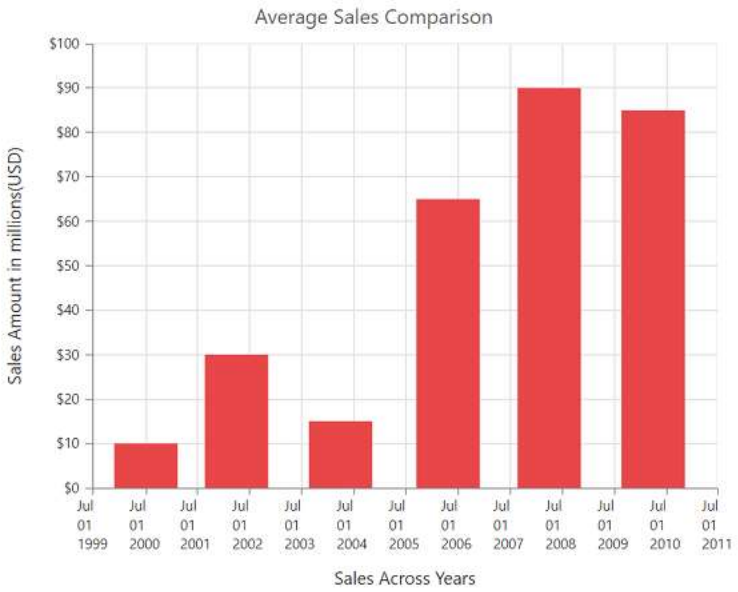
Copyright © 2001 - 2020 Syncfusion Inc.



For numeric, datetime and datetimeCategory value type, **
** can be specified in labelFormat.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
px=>px.LabelFormat("MMM<br>dd<br>yyyy").ValueType(AxisValueType.Datetime)
)
// ...
)
```



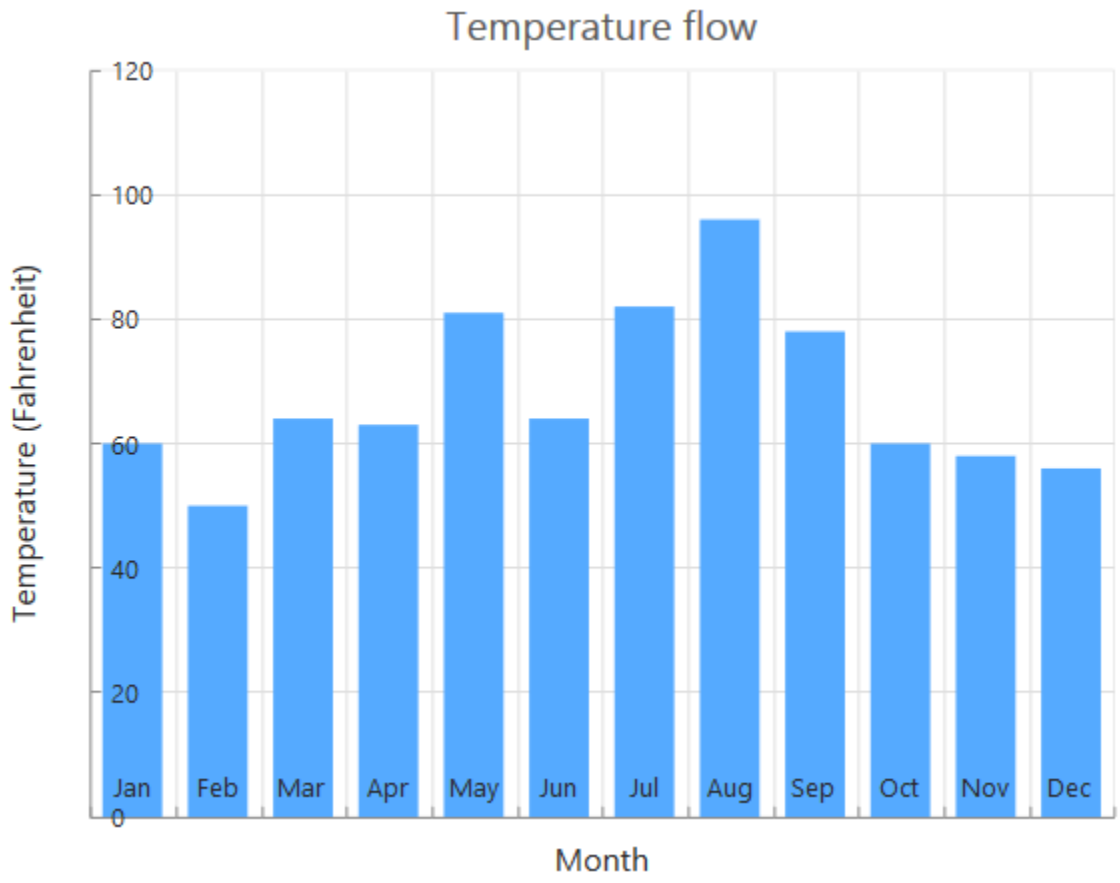
Label and tick positioning

Axis labels and ticks can be positioned inside or outside the chart area by using the **LabelPosition** and **TickPosition** properties. The labels and ticks are positioned outside the chart area, by default.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing label and tick positions
px=>px.AxisLabelPosition(AxisLabelPosition.Inside).TickLinesPosition(TickLin
esPosition.Inside)
//....
)
// ...
)
```



Edge labels placement

Labels with long text at the edges of an axis may appear partially outside the chart. The **EdgeLabelPlacement** property can be used to avoid the partial appearance of the labels at the corners.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing edge label placement
px=>px.EdgeLabelPlacement(EdgeLabelPlacement.Shift)
//....
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// ...
)
```

Chart before setting edge label placement to X-axis

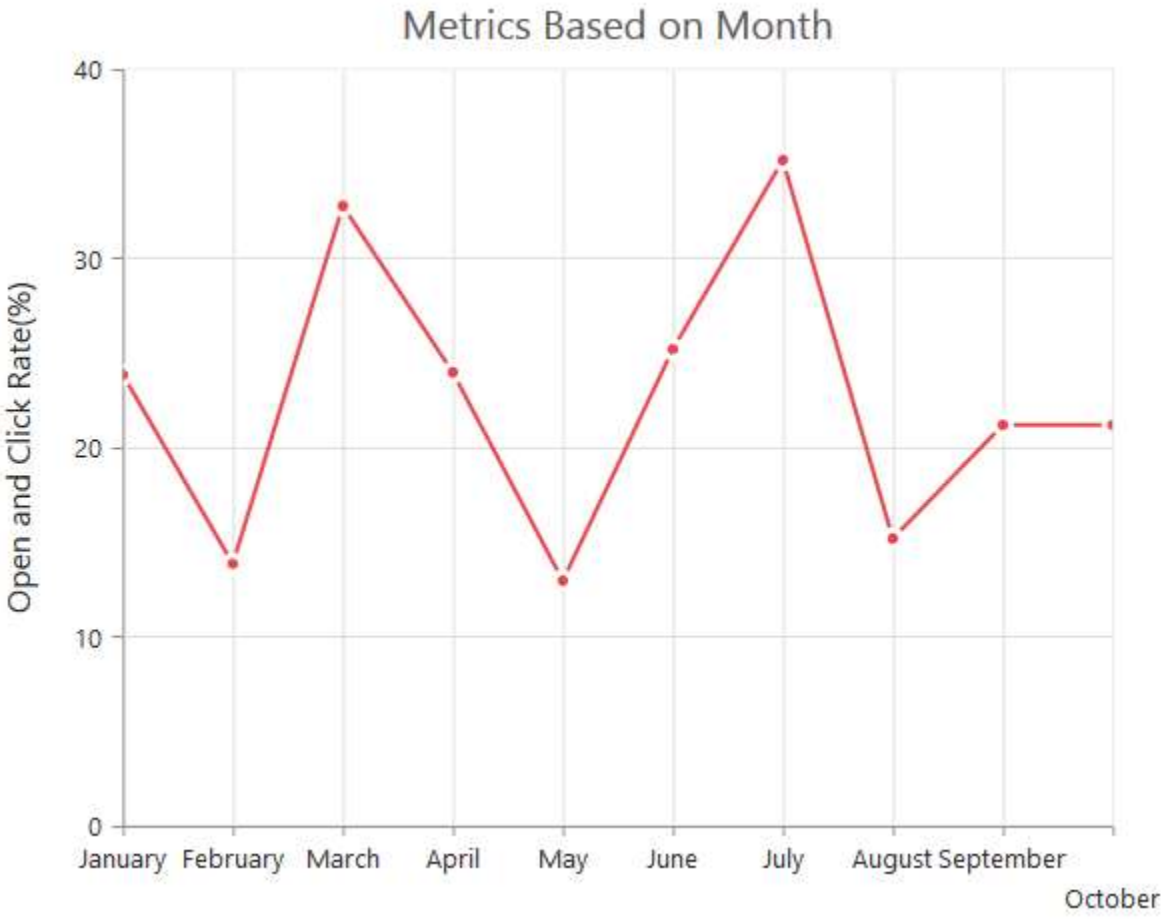
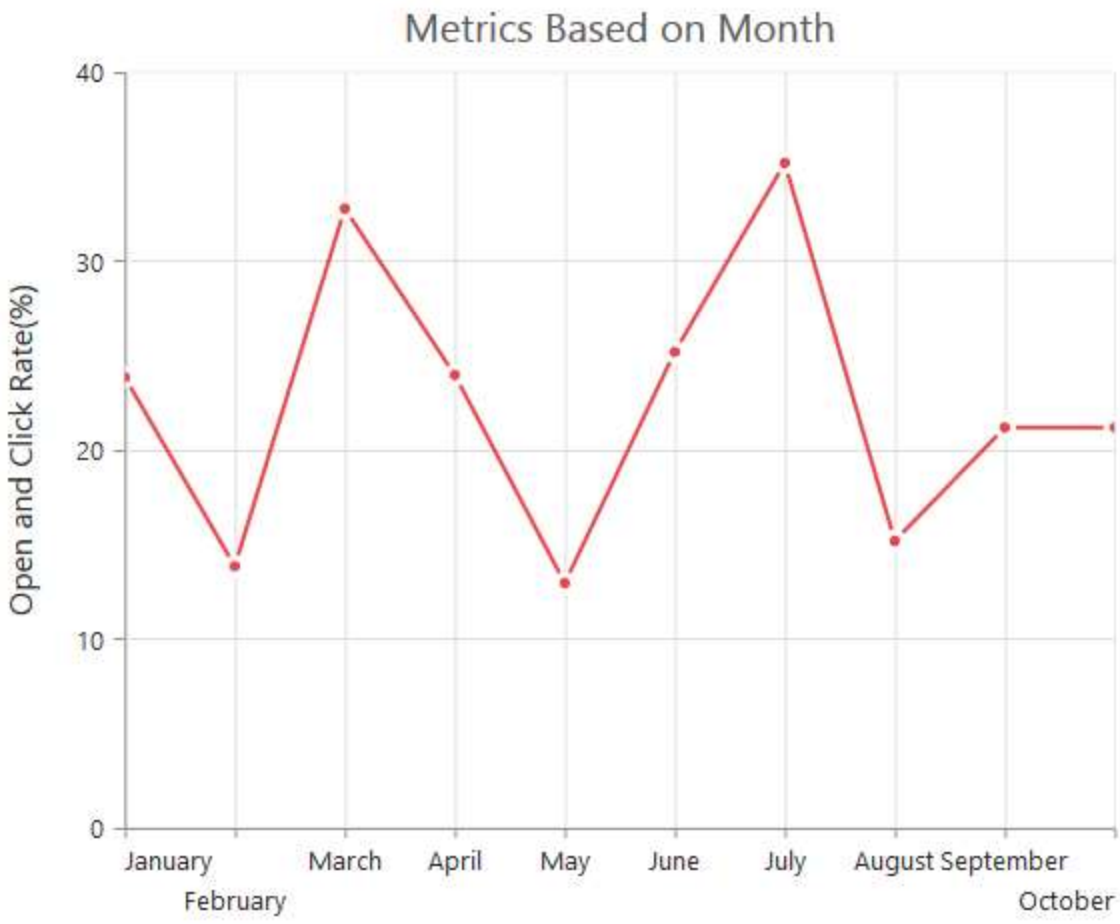


Chart after setting edge label placement to X-axis



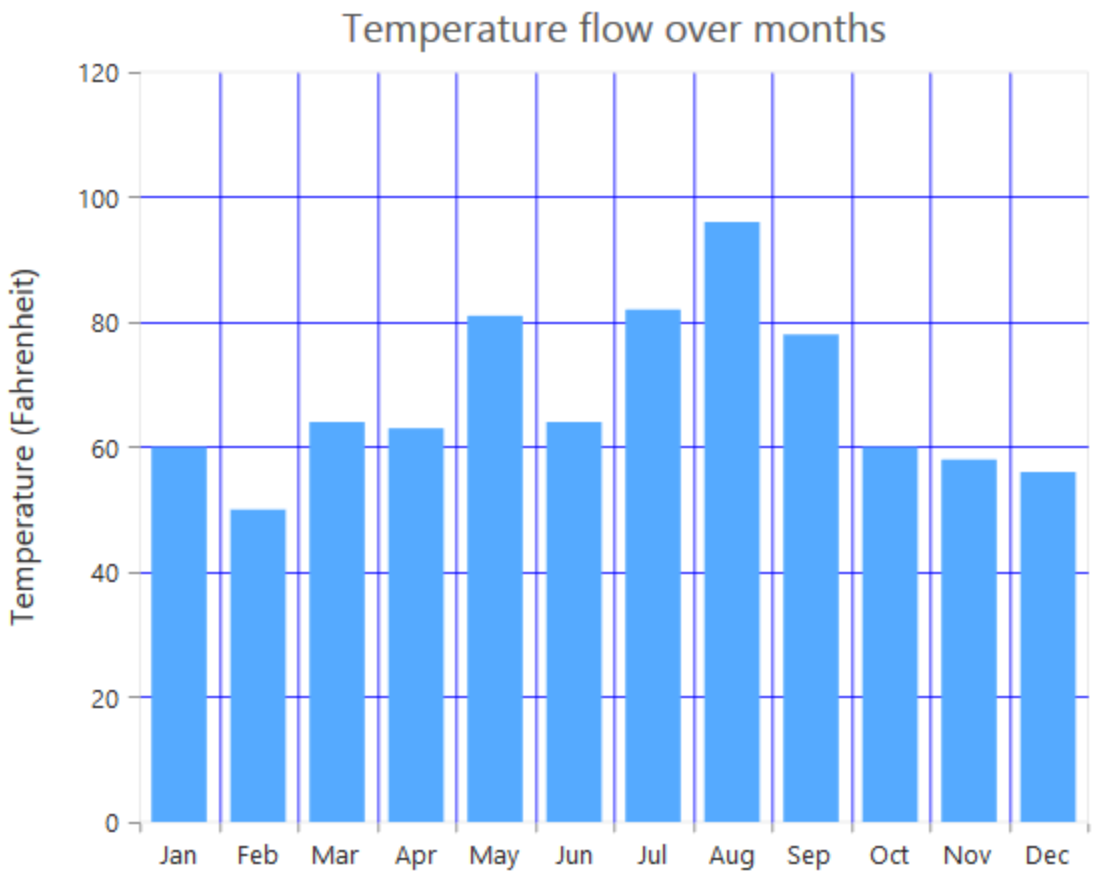
Grid lines customization

The **MajorGridLines** and **MinorGridLines** properties in the axis are used to customize the major grid lines and minor grid lines of an axis. They provide options to change the Width, Color, Visibility and Opacity of the grid lines. The minor grid lines are not visible, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing grid lines
px=>px.MajorGridLines(ma=>ma.Color("blue").Visible(true).Width(1))
.MinorGridLines(mi=>mi.Color("red").Visible(false).Width(1))
.MinorTicksPerInterval(0)
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



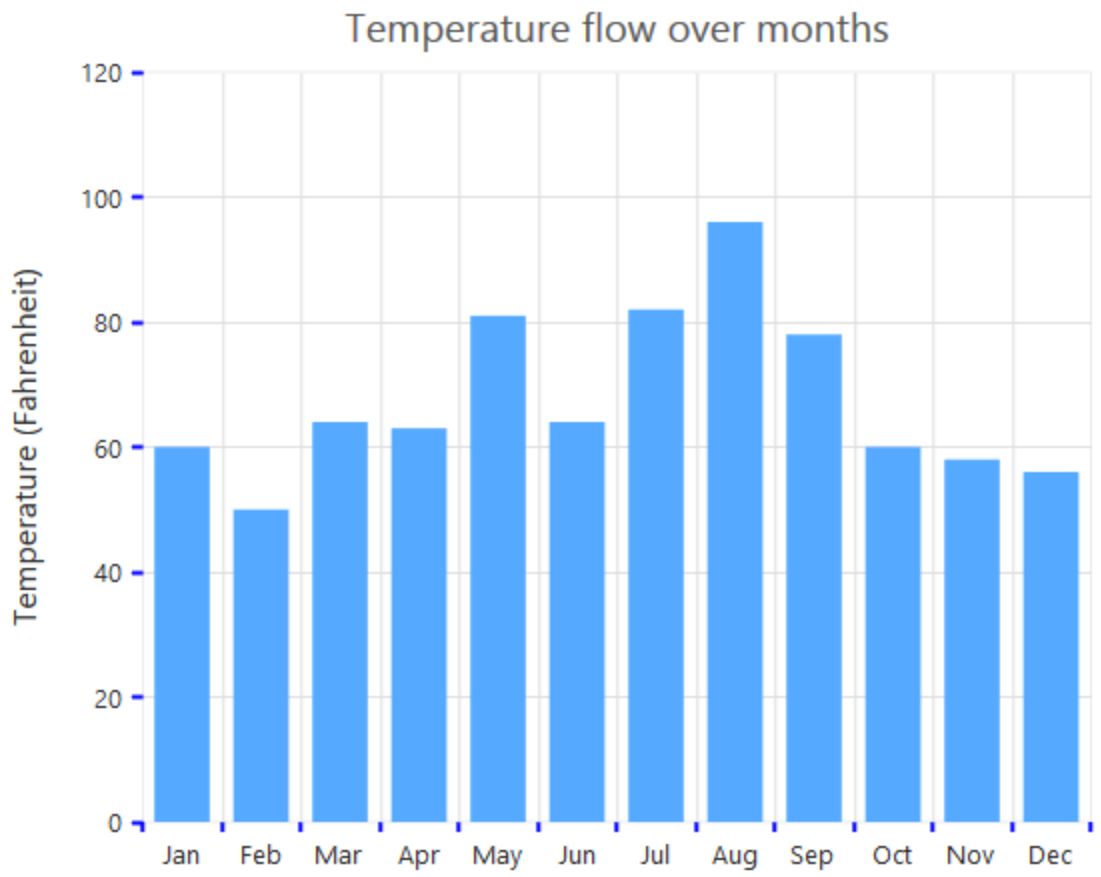
Tick lines customization

The **MajorTickLines** and **MinorTickLines** properties in the axis are used to customize the major tick lines of an axis and minor tick lines of an axis. They provide options to change the Width, Size, Color and Visibility of the grid lines. The minor tick lines are not visible, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Customizing tick lines
px=>px.MajorTickLines(ma=>ma.Color("blue").Visible(true).Width(1).Size(5))
.MinorTickLines(mi=>mi.Color("red").Visible(false).Width(1).Size(5))
.MinorTicksPerInterval(0)
//....
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Inversing axis

Axis can be inversed by using the **IsInversed** property of the axis. The default value of the IsInversed property is **false**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Inversing the X-axis
px=>px.IsInversed(false)
//....
)
// ...
)
```

Chart before inversing the axes

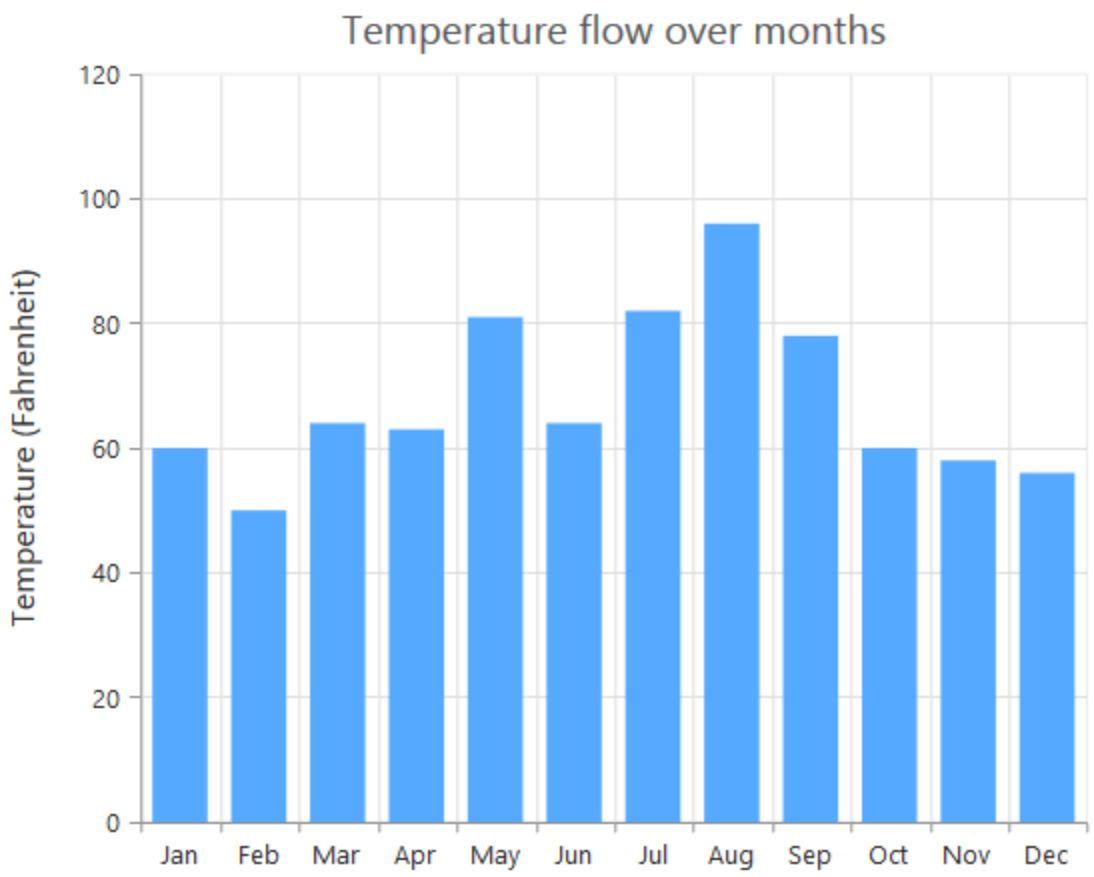
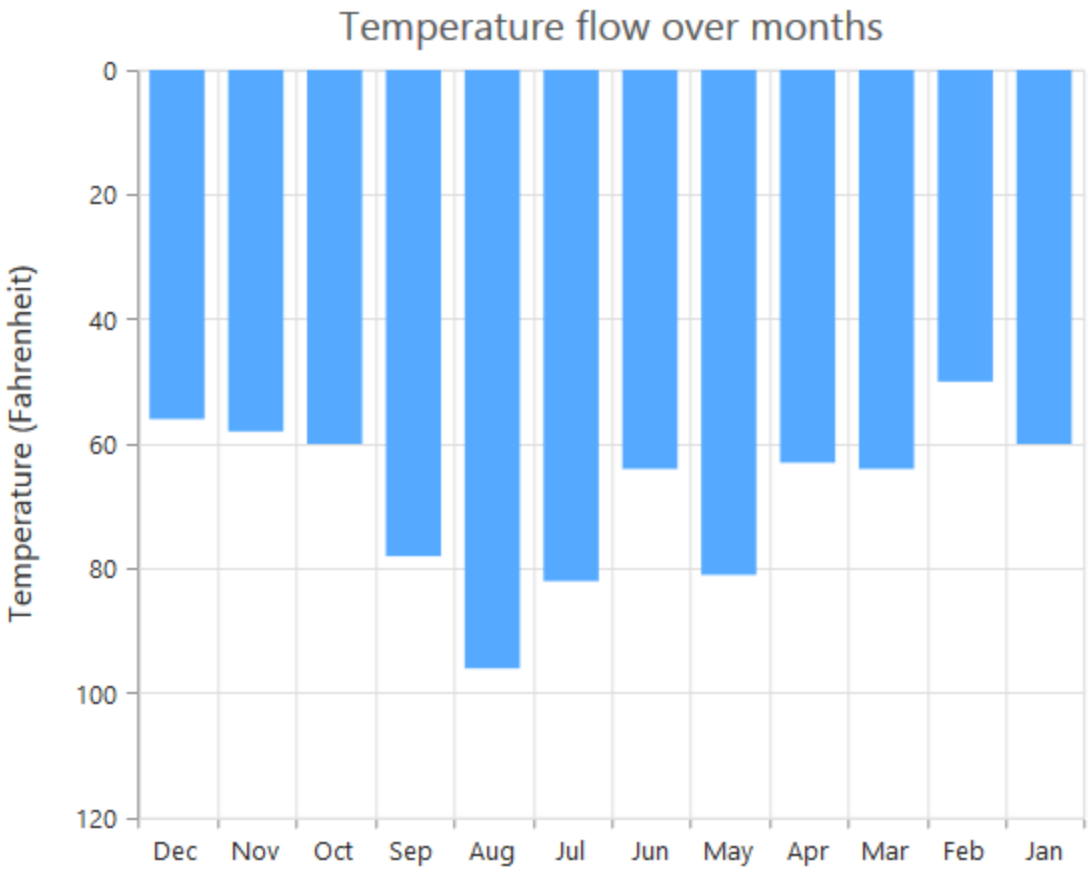


Chart after inversing the axes

Copyright © 2001 - 2020 Syncfusion Inc.



Place axes at the opposite side

The **OpposedPosition** property of axis can be used to place the axis at the opposite side of its default position. The default value of the OpposedPosition property is **false**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Placing axis at the opposite side of its normal position
px=>px.OpposedPosition(true)
//....
)
// ...
)
```

Chart with X and Y axes at normal position

Copyright © 2001 - 2020 Syncfusion Inc.

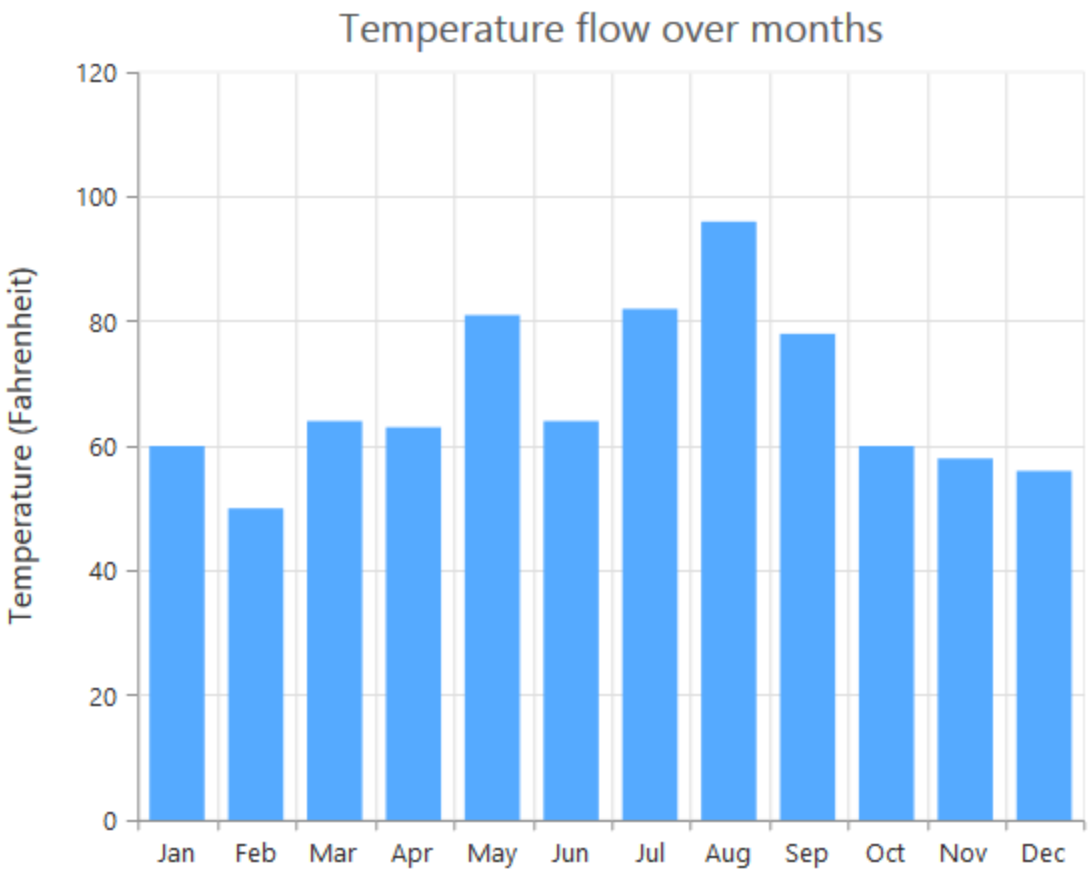
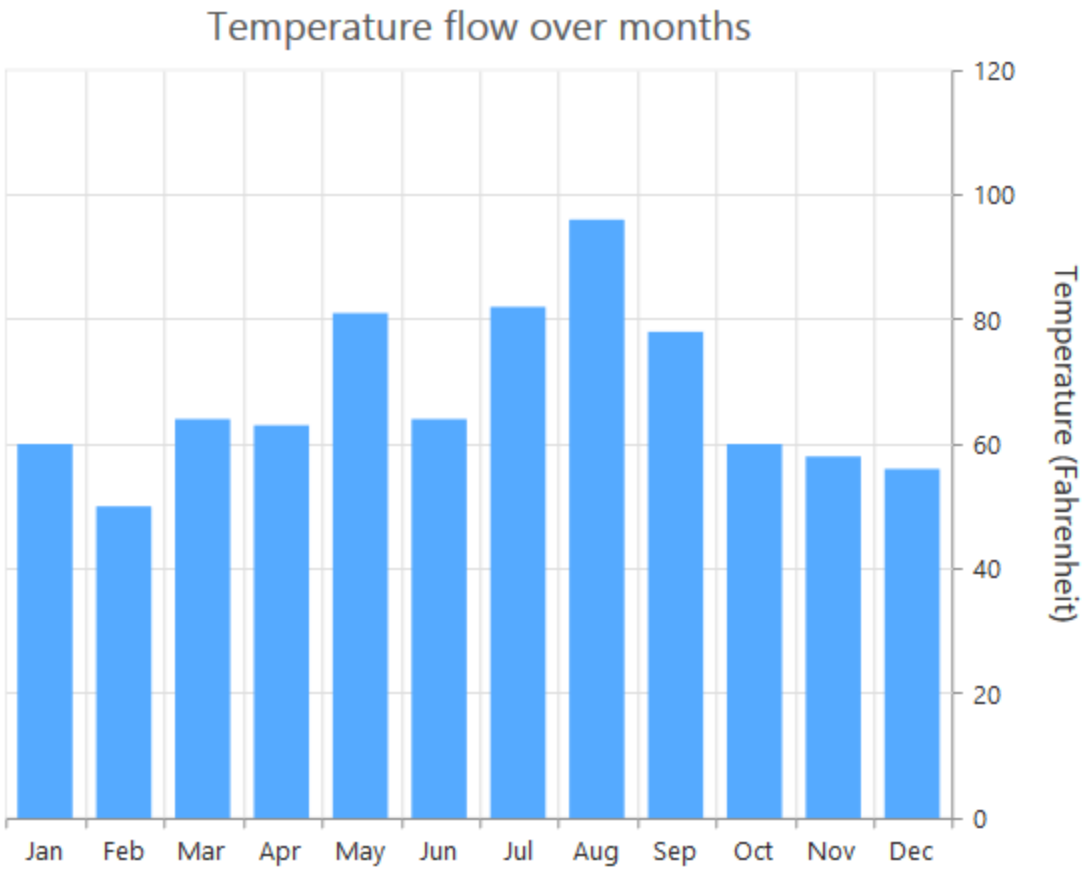


Chart with Y-axis at opposed position

Copyright © 2001 - 2020 Syncfusion Inc.



Maximum number of labels per 100 pixels

A maximum of 3 labels are displayed for each 100 pixels in the axis, by default. The maximum number of labels that is present within the 100 pixels length can be customized by using the **MaximumLabels** property of the axis. This property is applicable only for an automatic range calculation and it does not work when you set the value for Interval property in the axis range.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryXAxis(
//Maximum number of labels per 100 pixels
px=>px.MaximumLabels(1)
//....
)
// ...
)
```

Chart before setting maximum labels per 100 pixels

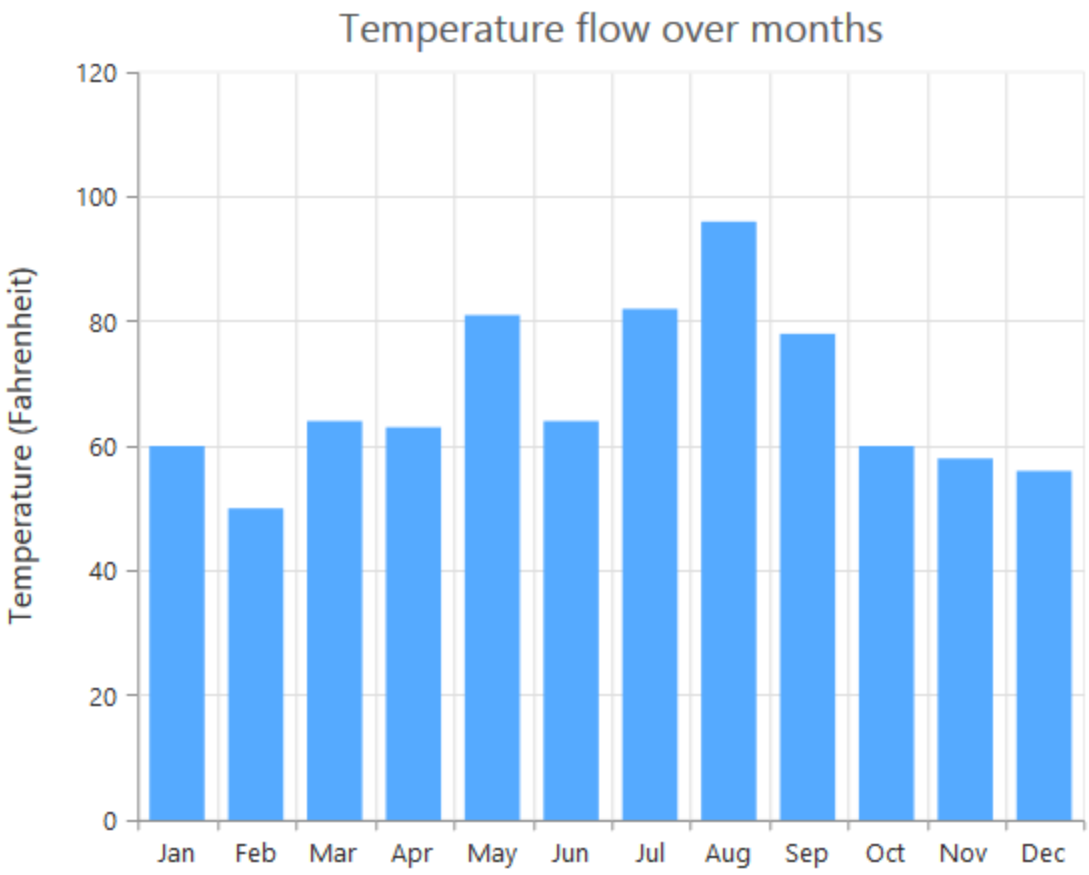
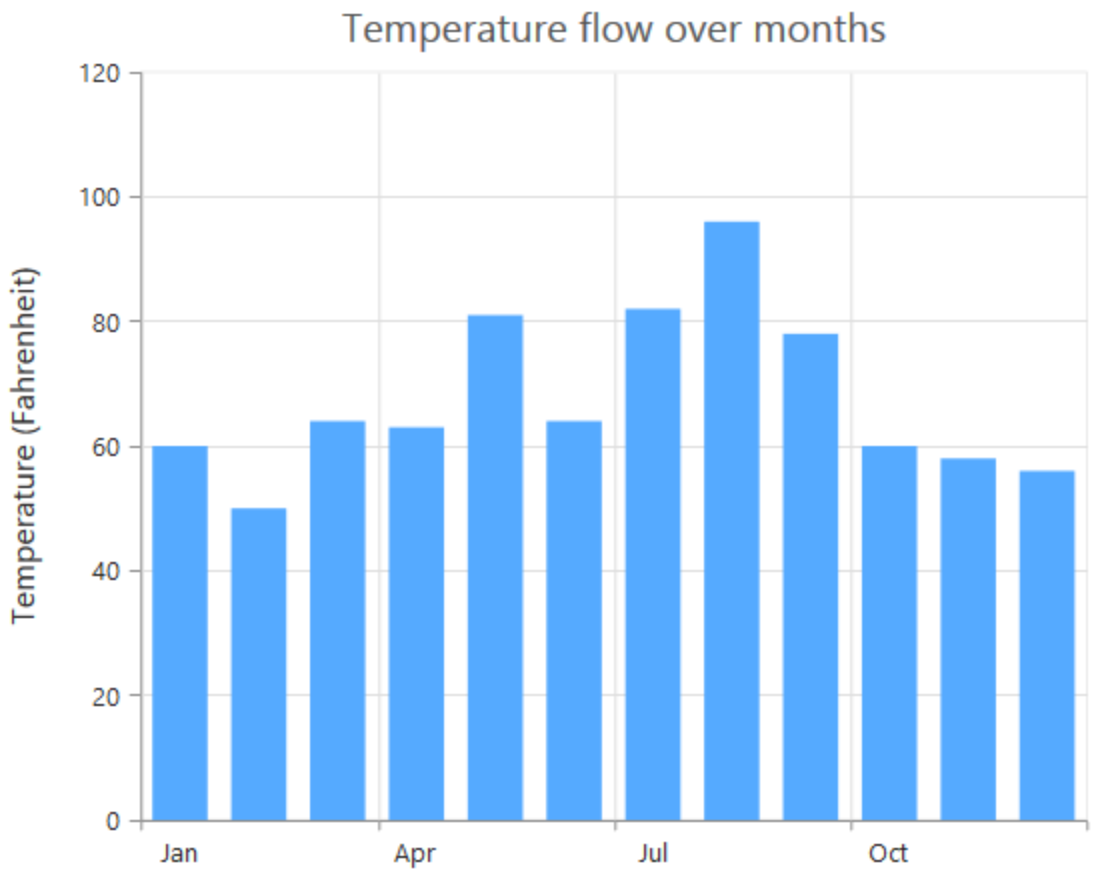


Chart after setting maximum labels one per 100 pixels

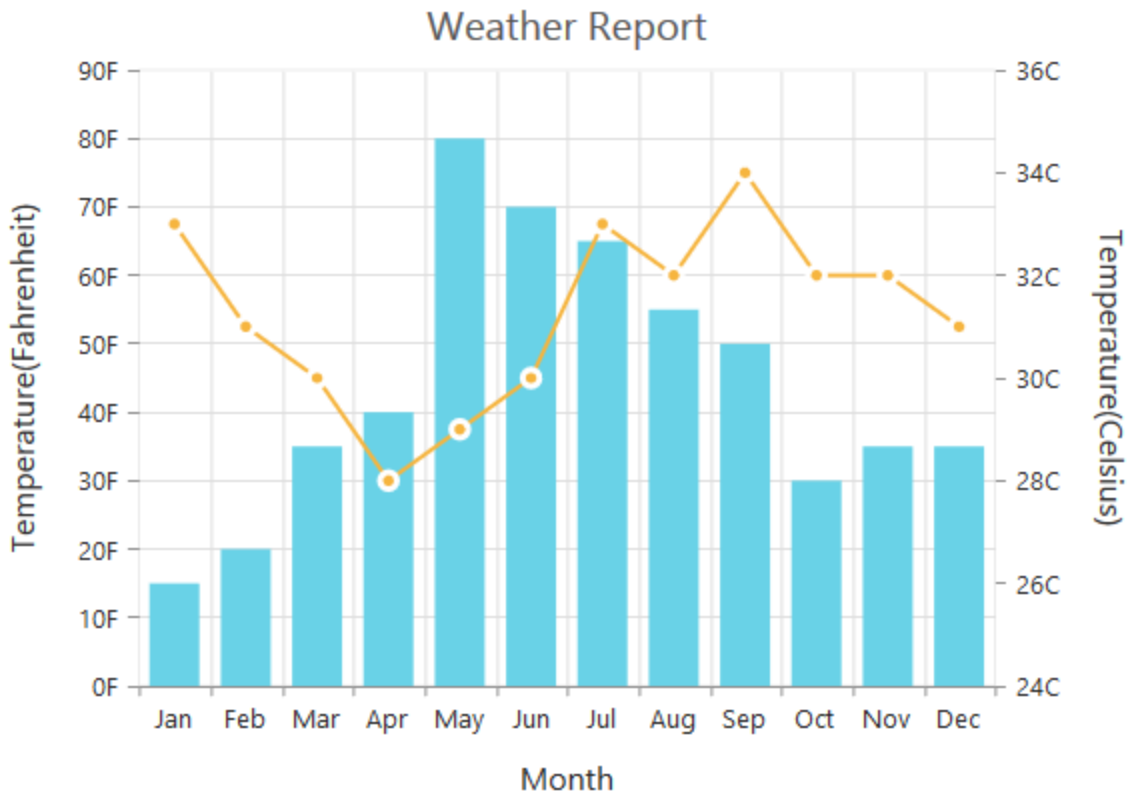
Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Axis

Multiple axes can be used in the Chart and chart area can be split into multiple panes to draw multiple series with multiple axes.

Copyright © 2001 - 2020 Syncfusion Inc.

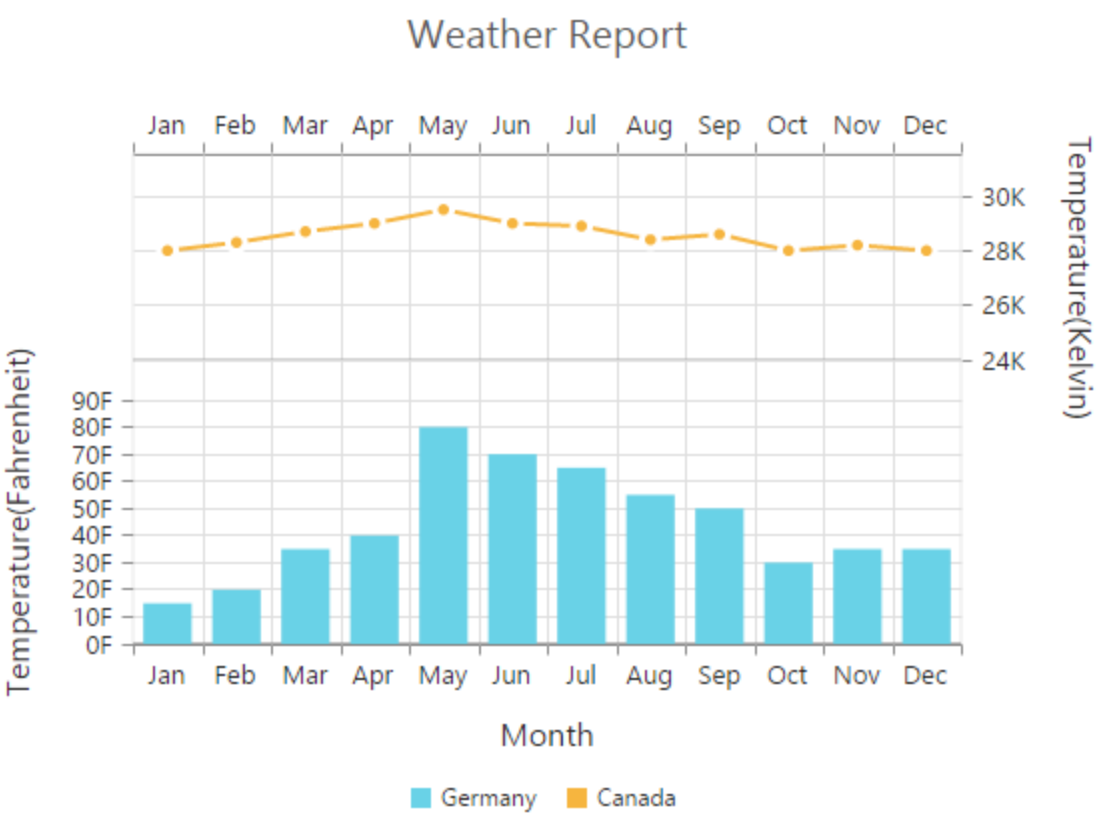


An additional horizontal or vertical axis can be added to the chart by adding an axis instance to the **Axes** collection and then you can associate it to a series by specifying the name of the axis to the **XAxisName** or **YAxisName** property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//  Creating a secondary horizontal axis
.Axes(ax => { ax.Name("SecondaryX").Add(); ax.Name("SecondaryY").Add(); })
.Series(sr =>
{
//  Binding secondary X-axis with a series
sr.XAxisName("SecondaryX")
//  Binding secondary Y-axis with a series
.YAxisName("SecondaryX").Add();
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the multiple axis online demo sample.

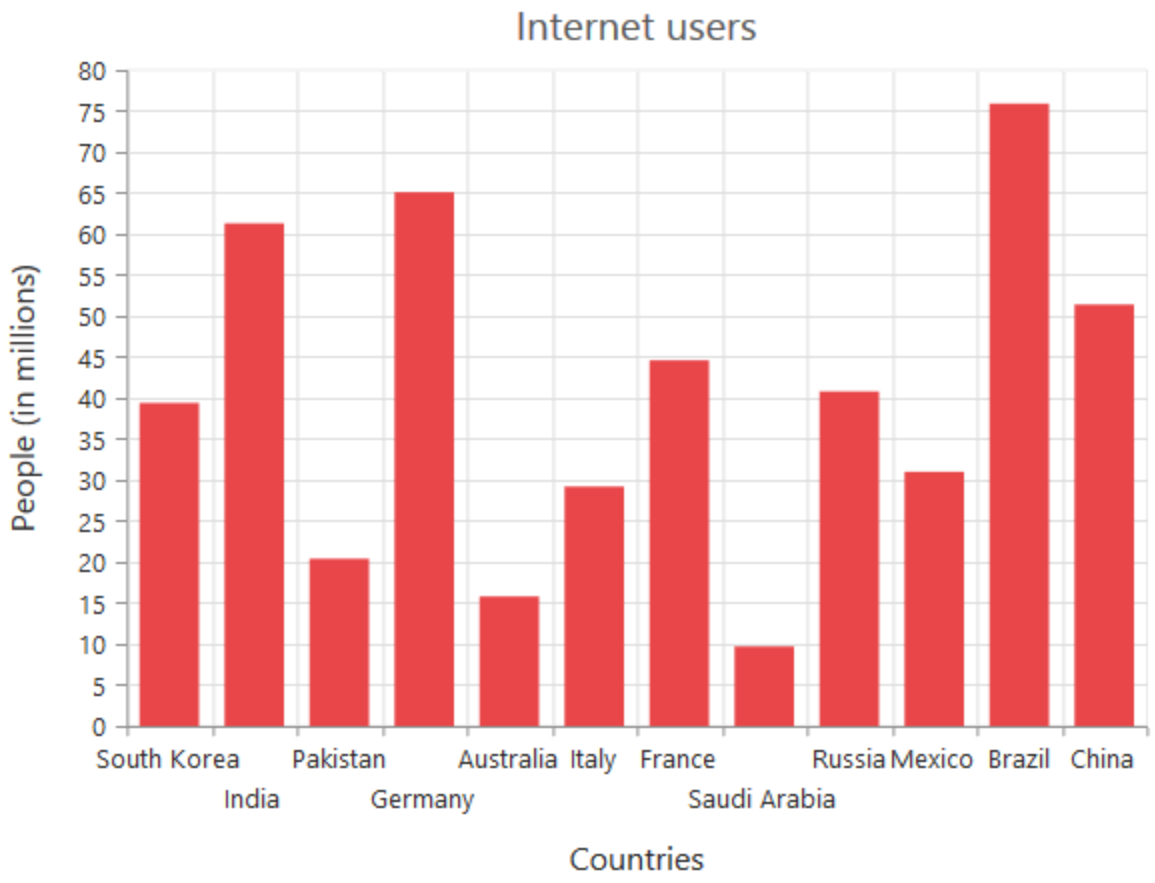
Smart Axis Labels

When the Axis labels overlap with each other based on the chart dimensions and label size, you can use the **LabelIntersectAction** property of the axis to avoid overlapping. The default value of the LabelIntersectAction is **None**. The other available values of the Label Intersect Actions are **Rotate45**, **Rotate90**, **Trim**, **MultipleRows**, **Wrap** and **Hide**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
// Avoid overlapping of x-axis labels
.PrimaryXAxis(px=>px.LabelIntersectAction(LabelIntersectAction.MultipleRows)
// ...
)
```

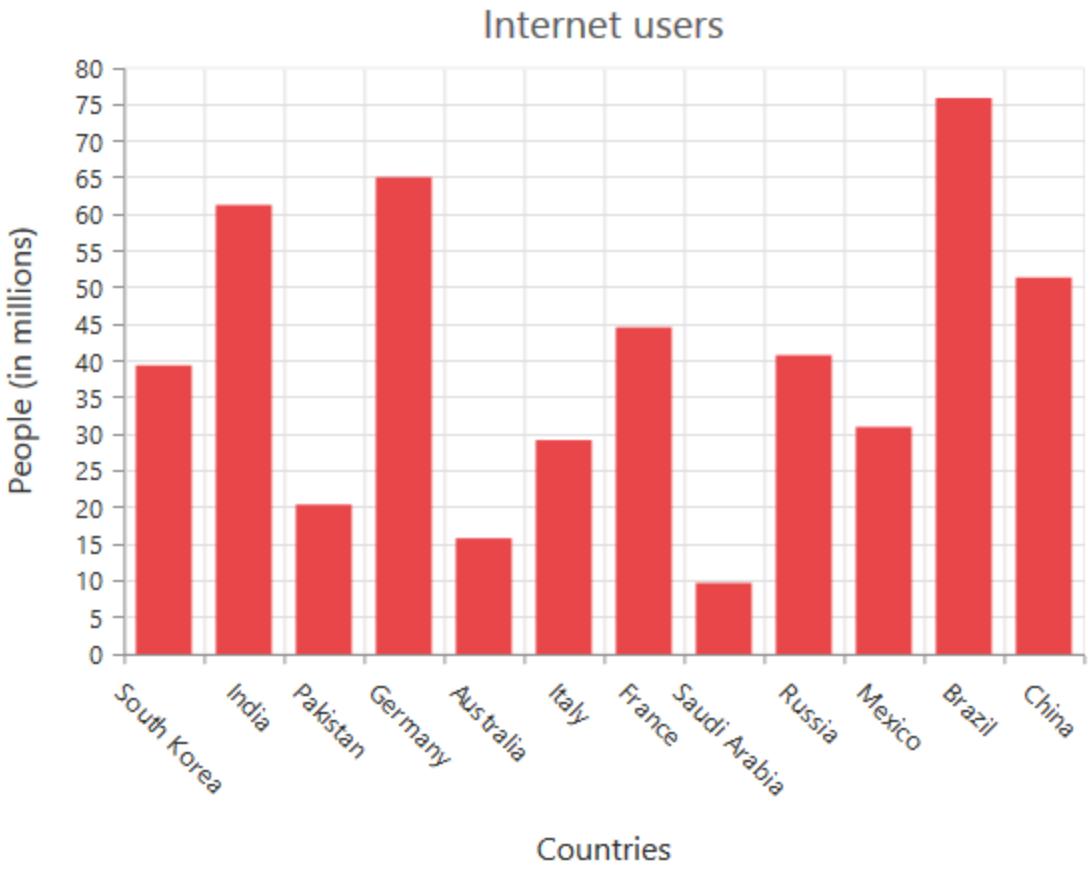
Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view our online demo sample for smart axis labels.

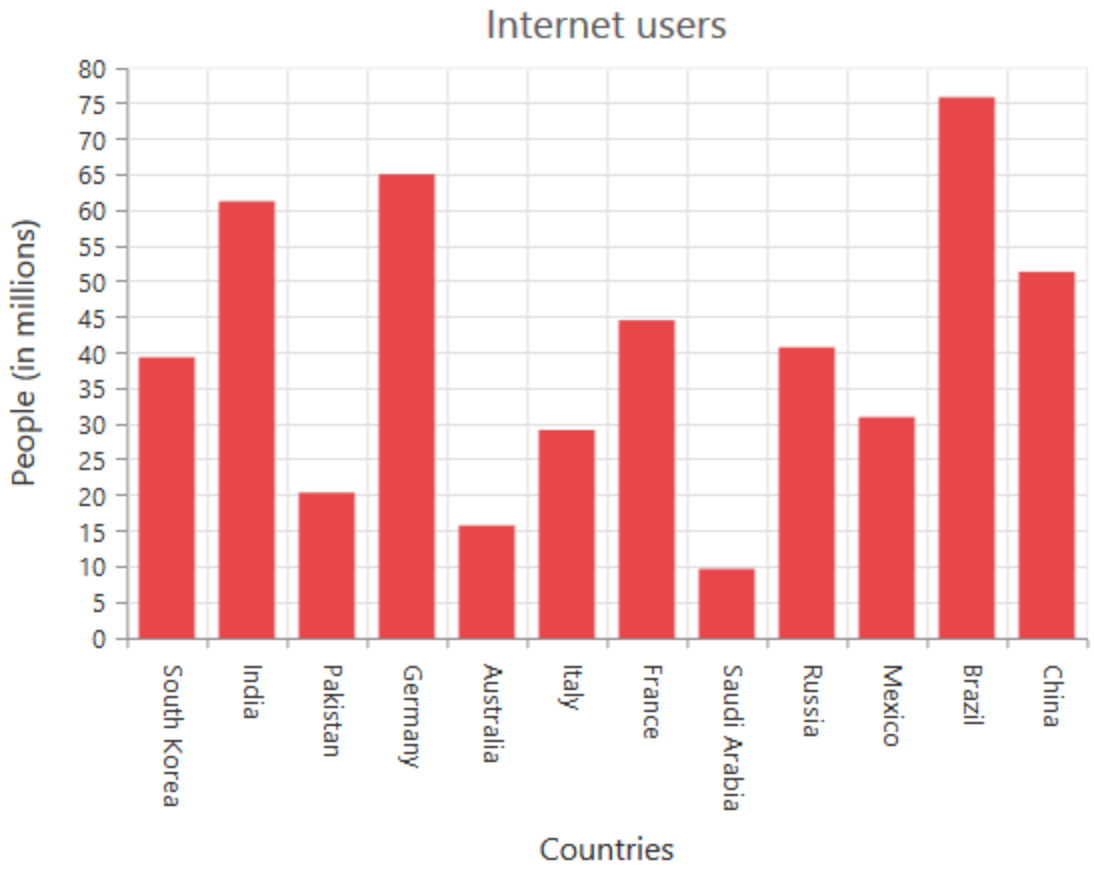
The following screenshot displays the result, when the LabelIntersectAction property is set as **Rotate45**.

Copyright © 2001 - 2020 Syncfusion Inc.



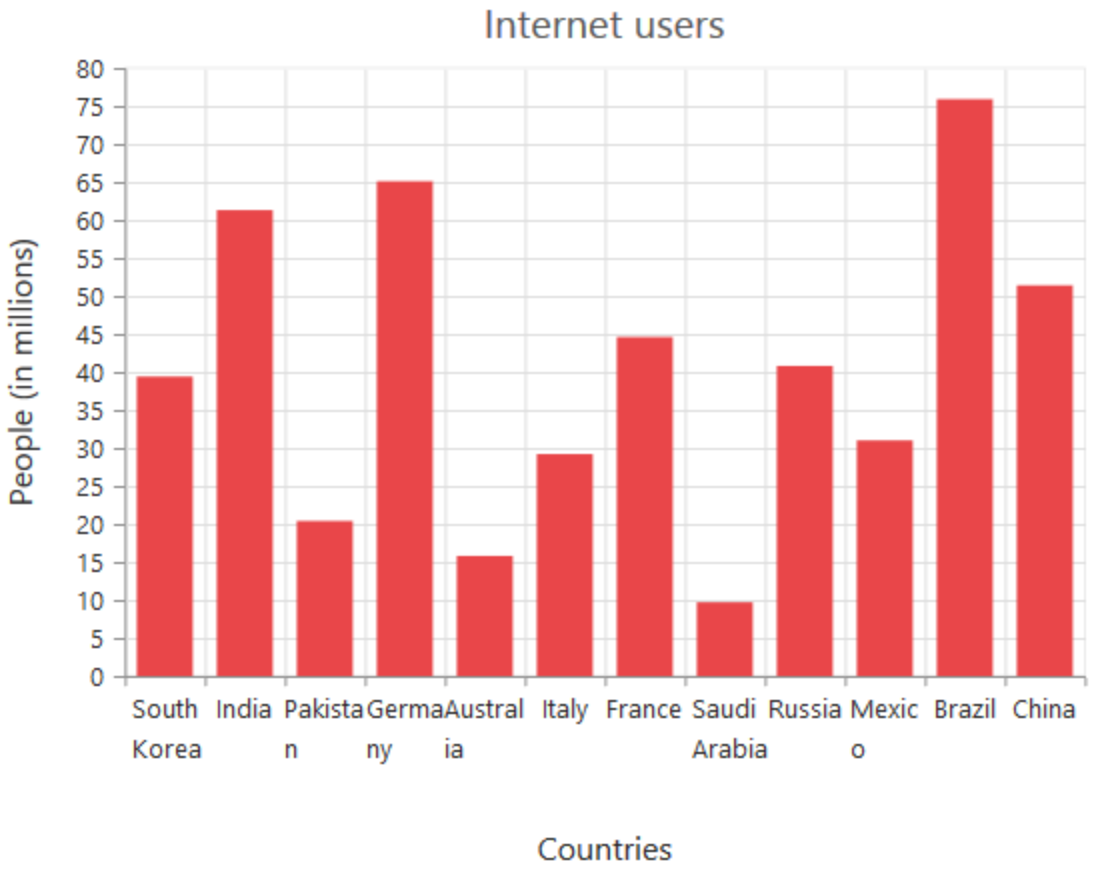
The following screenshot displays the result, when the LabelIntersectAction property is set as **Rotate90**.

Copyright © 2001 - 2020 Syncfusion Inc.



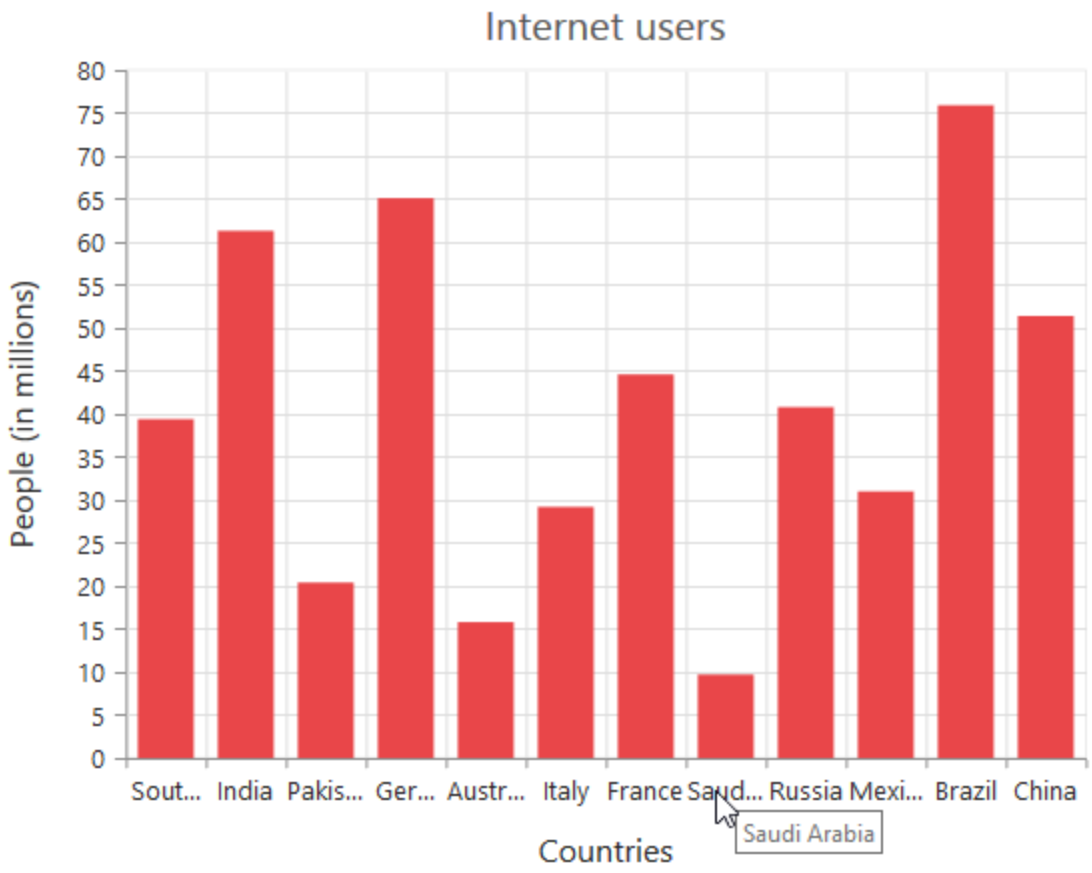
The following screenshot displays the result, when the LabelIntersectAction property is set as **Wrap**.

Copyright © 2001 - 2020 Syncfusion Inc.



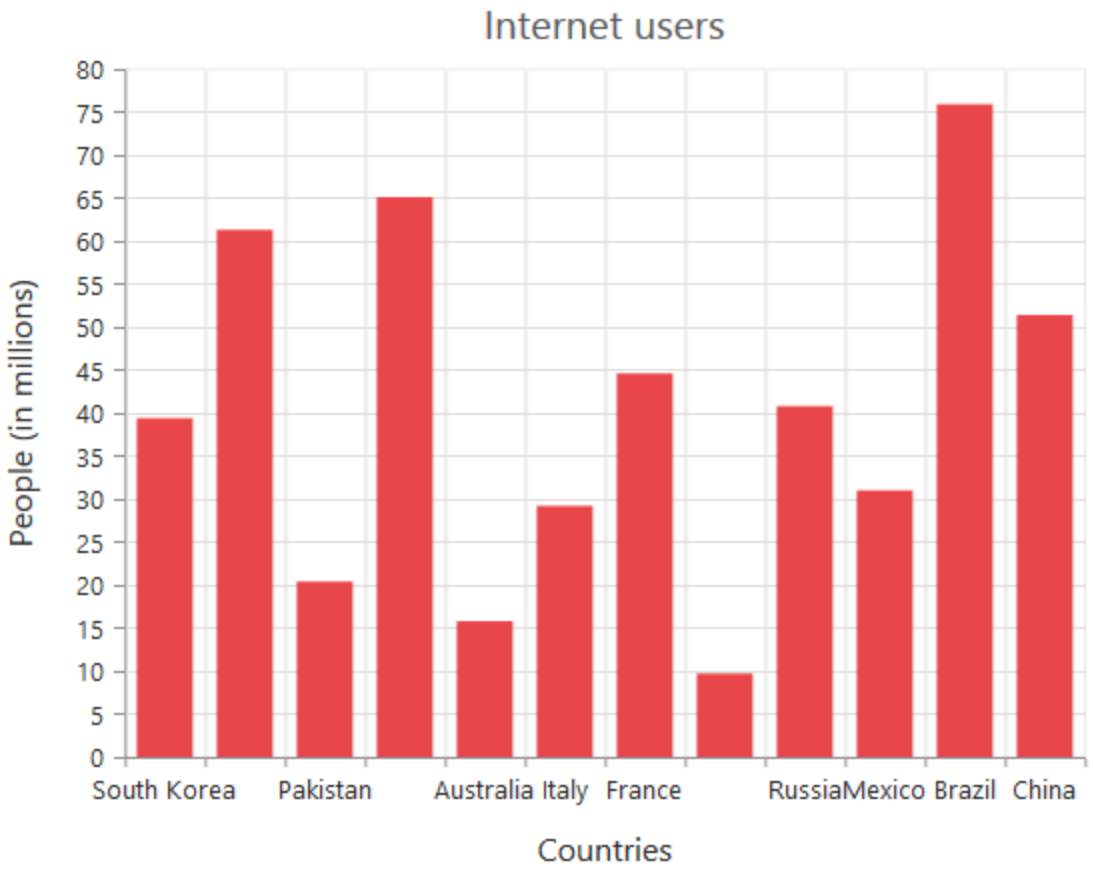
The following screenshot displays the result, when of setting the **trim** as value to the LabelIntersectAction property.

Copyright © 2001 - 2020 Syncfusion Inc.



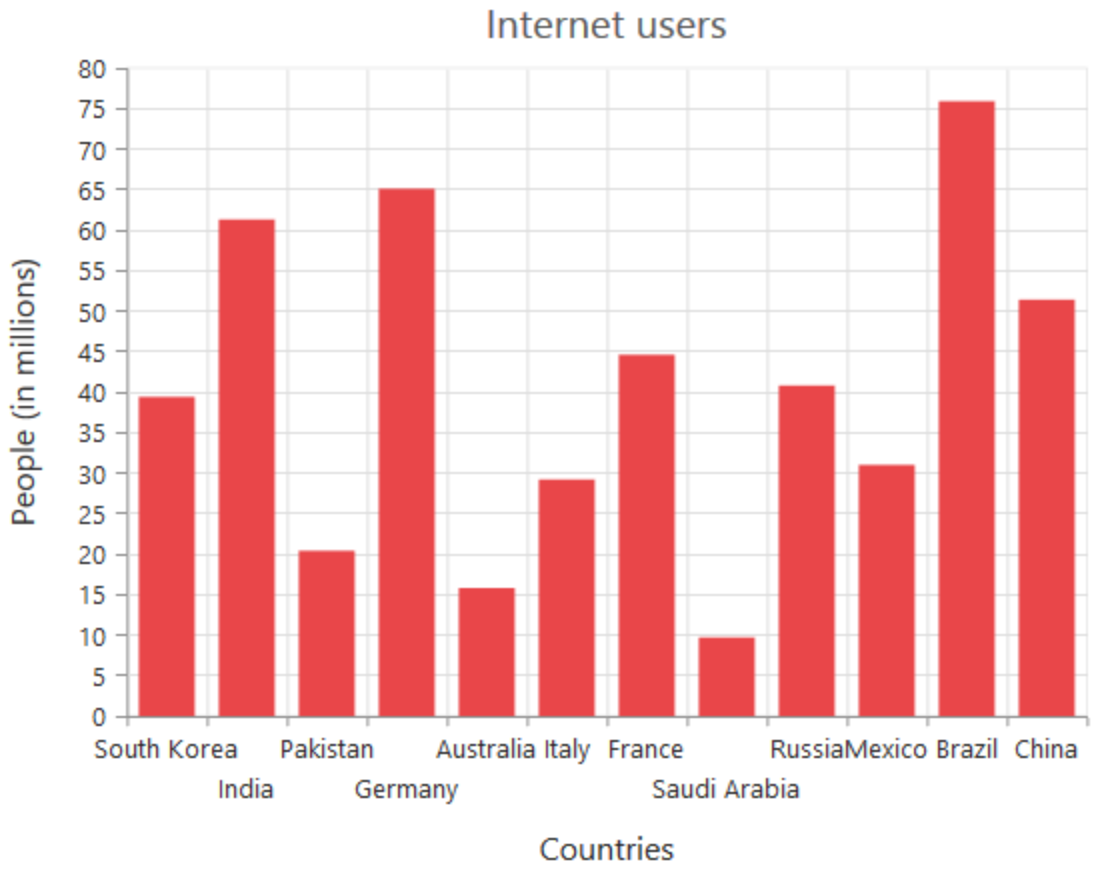
The following screenshot displays the result, when the LabelIntersectAction property is set as **Hide**.

Copyright © 2001 - 2020 Syncfusion Inc.



The following screenshot displays the result, when the LabelIntersectAction property is set as **MultipleRows** .

Copyright © 2001 - 2020 Syncfusion Inc.



The following screenshot displays the result, when the LabelIntersectAction property is set as **WrapByWord**.

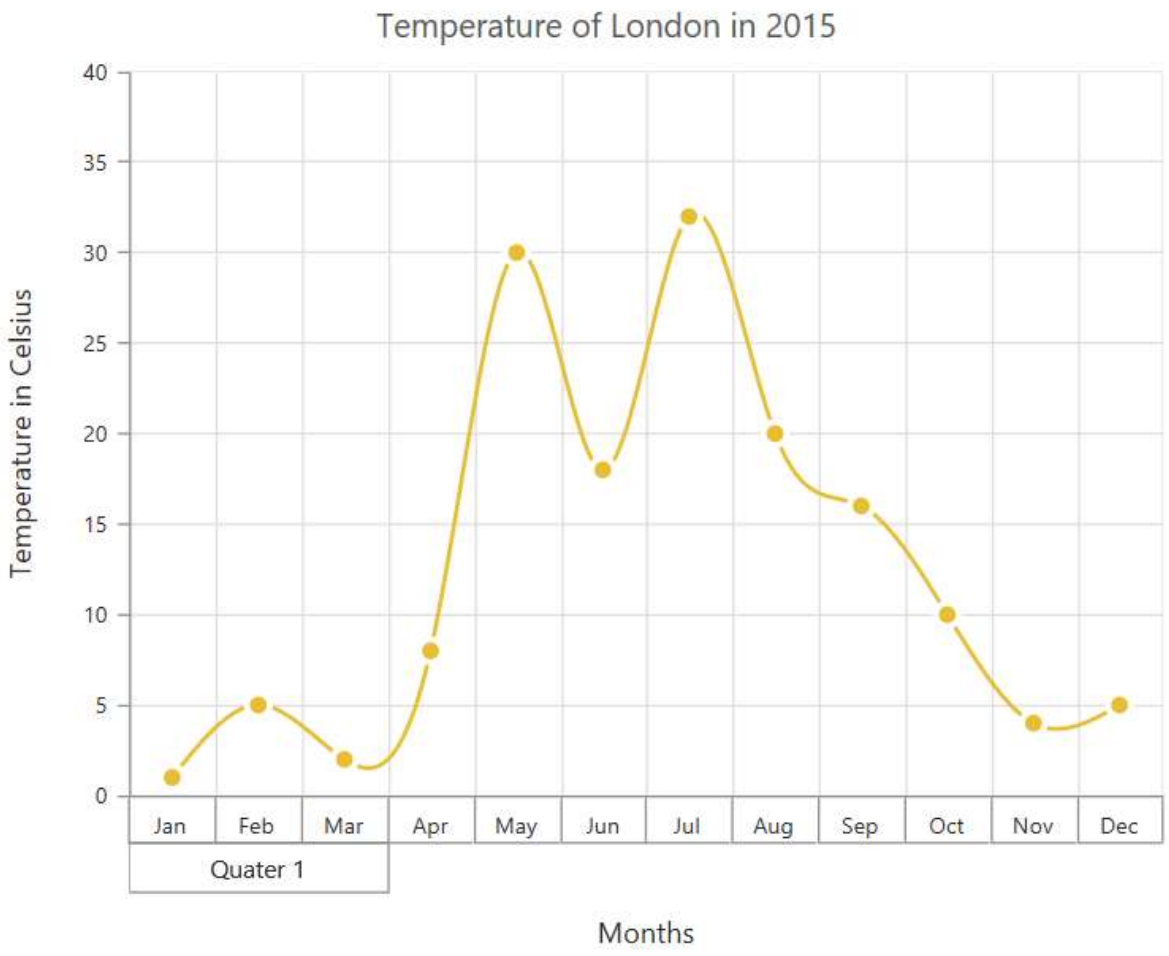
Multi-level Labels

Axis can be customized with multiple levels of labels using the **MultiLevelLabels** property. These labels are placed based on the start and end range values and we can add any number of labels to an axis.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis => axis.MultiLevelLabels(multiLevelLabels=> {
multiLevelLabels.Visible(true).Text("Quarter 1").Start(-0.5).End(2.5).Add();
})
))
```

Copyright © 2001 - 2020 Syncfusion Inc.

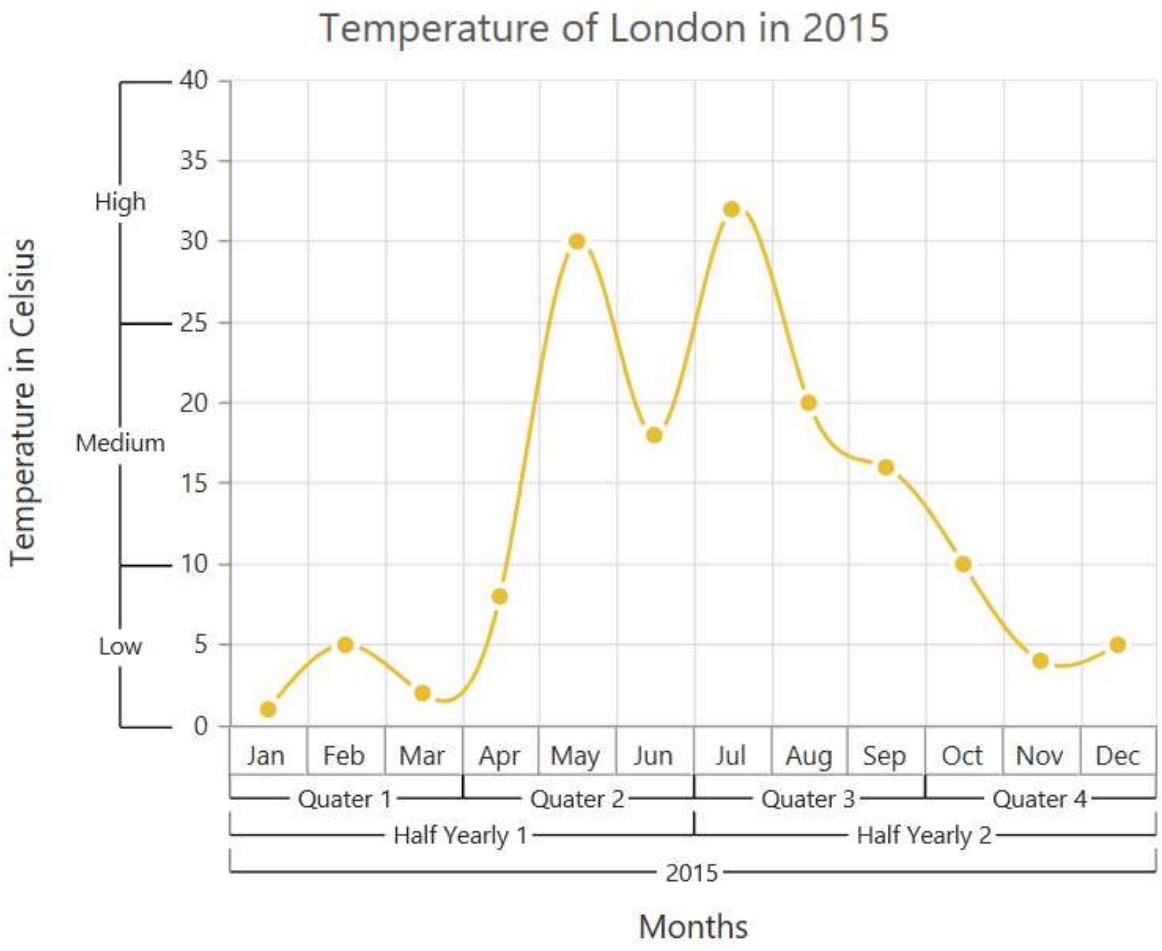


Customizing the multi-Level labels

The color, width and type of the border can be customized. The default border type is **Rectangle**. And the other supported border types are namely brace, curly brace, without top/bottom border and none.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis => axis.MultiLevelLabels(
multiLevelLabels=> {multiLevelLabels.
Border(br=>br.Type(MultiLevelLabelBorder.Brace).Width(2).Color("Black"))).Add();
})
))
```

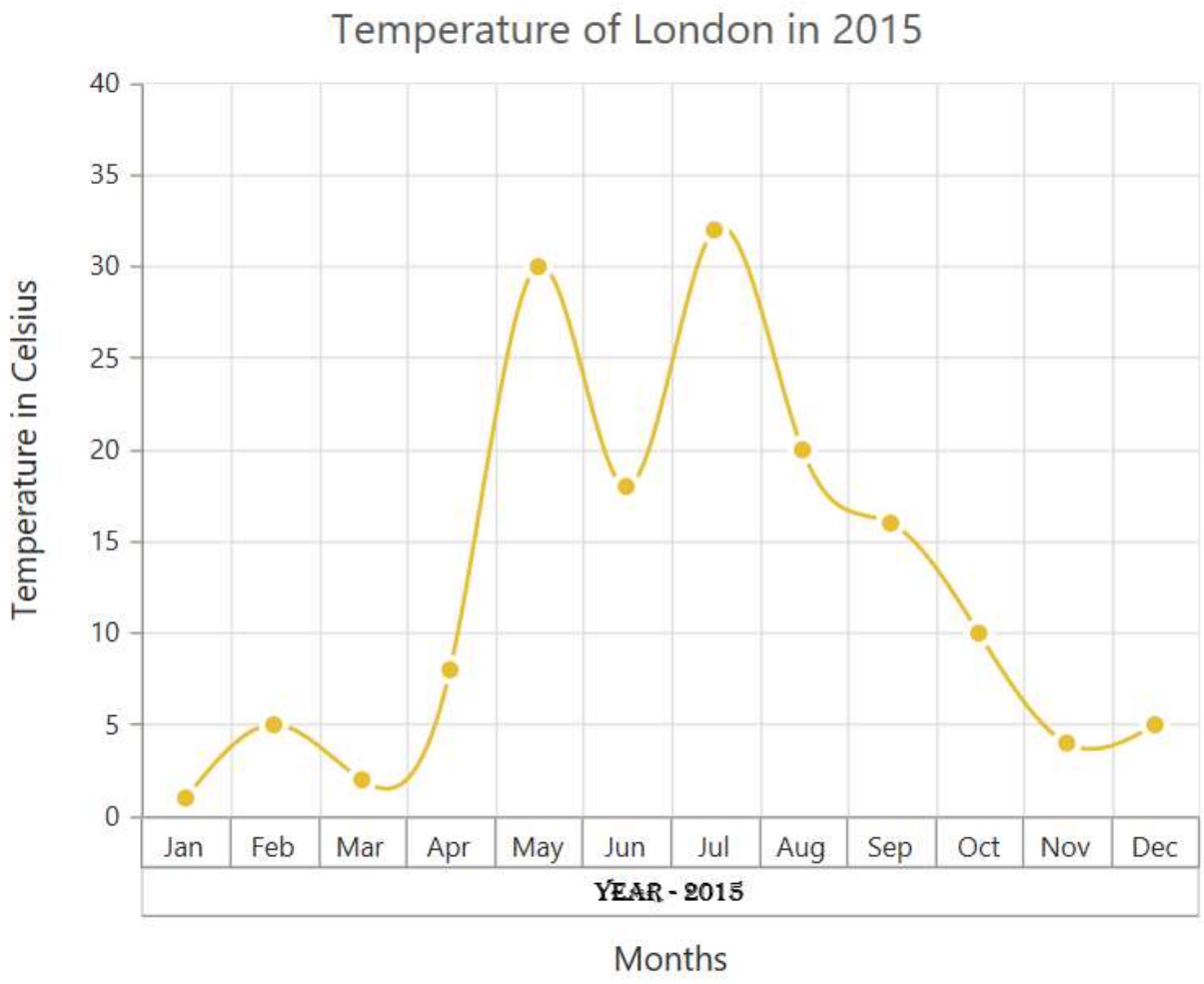


The text of the labels can be customized using the **Text** and **Font** properties

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis =>axis.MultiLevelLabels(multiLevelLabels=>
{multiLevelLabels.Text("Year - 2015")
.Font(ft=> ft.FontFamily("Algerian").Size("12px").Color("Black")).Add();
})
))
```

Copyright © 2001 - 2020 Syncfusion Inc.

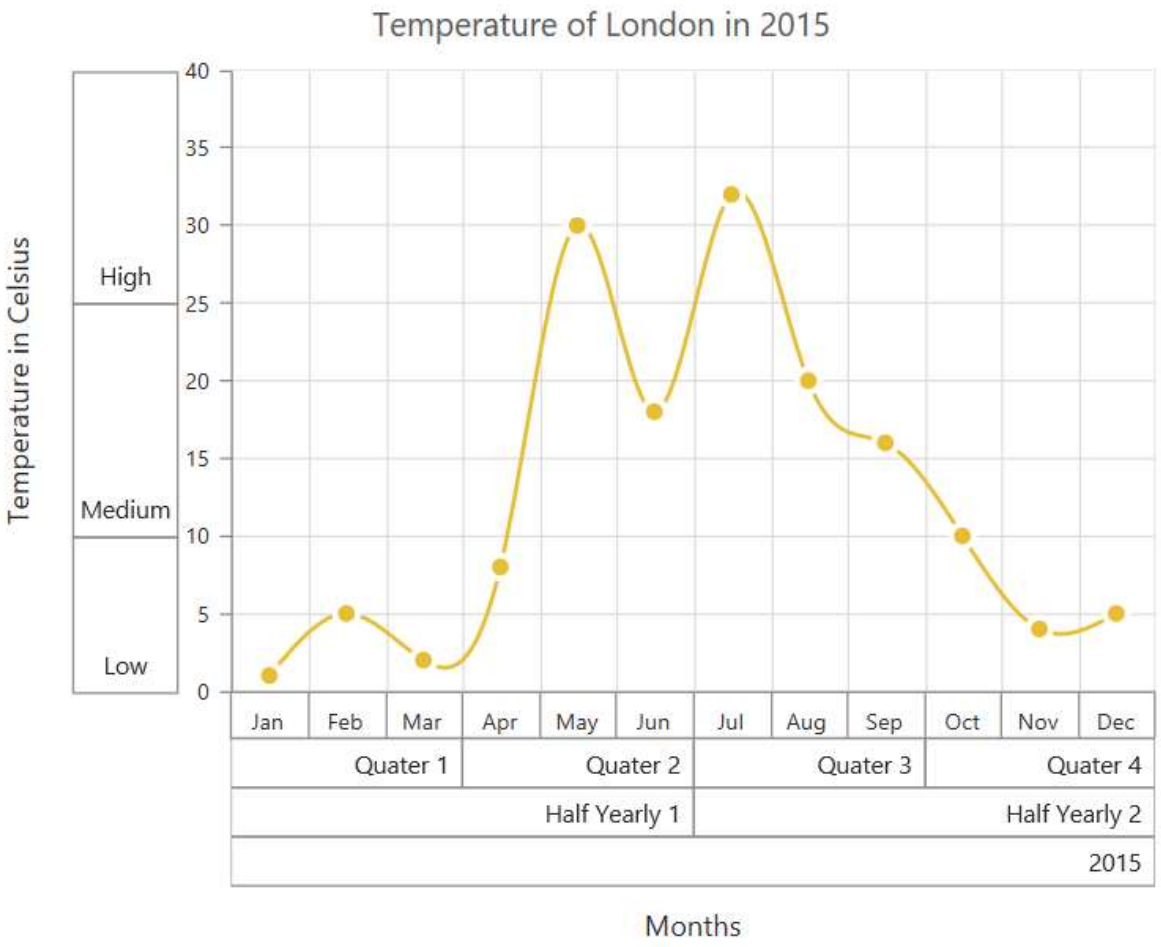


You can change the alignment of the text to far, near and center position using the **TextAlignment** property. By default, the text will be center aligned.

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis =>axis.MultiLevelLabels(multiLevelLabels=> {
multiLevelLabels.TextAlignment(Alignment.Far).Add();
})
))
```

Copyright © 2001 - 2020 Syncfusion Inc.



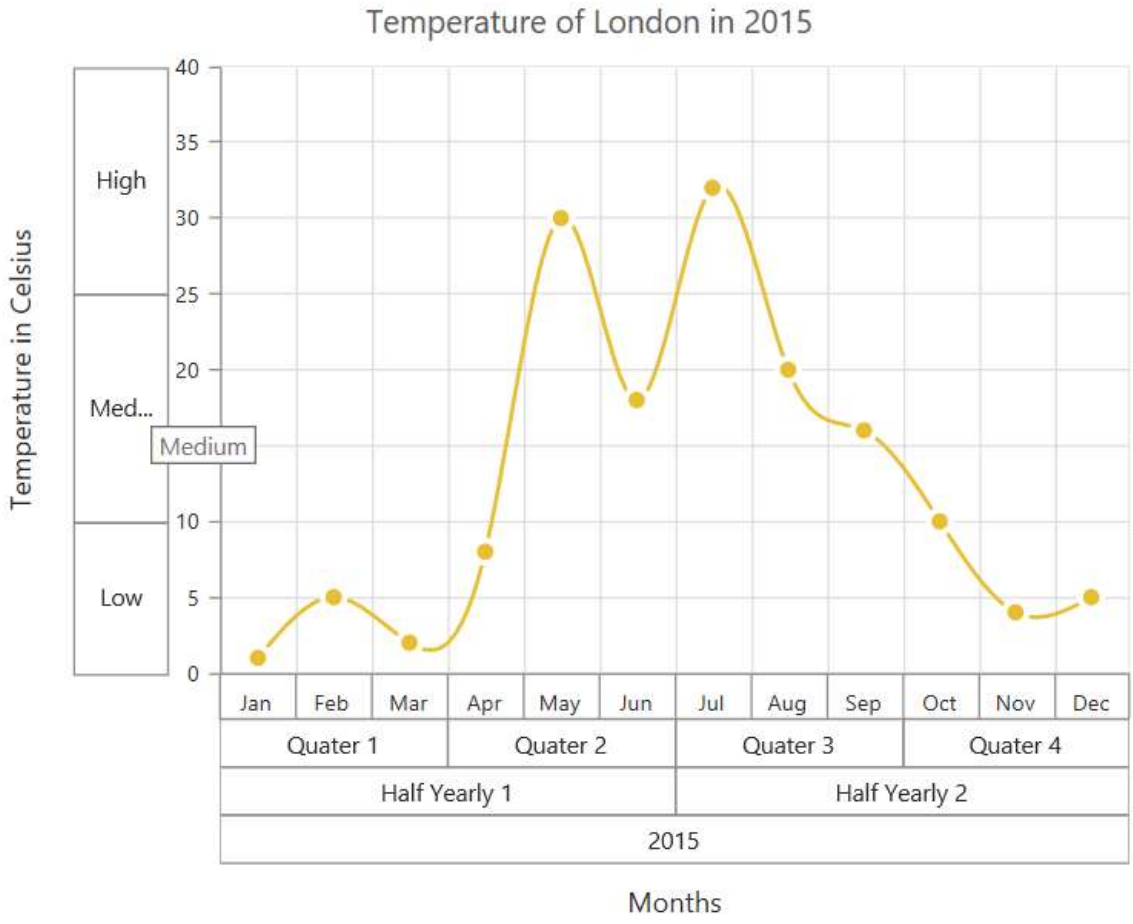
You can trim, wrap or wrapAndTrim the text if it exceeds the maximum text width value using the property **TextOverflow**

ASPX-CS

```
@(Html.EJ().Chart("container")
.PrimaryXAxis(axis =>axis.MultiLevelLabels(multiLevelLabels=> {
multiLevelLabels.TextOverflow(TextOverflow.Trim).MaximumTextWidth(40).Add();
})
))
```

The below screenshot shows the trimmed multi-level labels

Copyright © 2001 - 2020 Syncfusion Inc.



And these labels can be placed in various rows using the **Level** property.

Click here to view the multi-level labels online demo sample.

Multiple panes

Chart area can be divided into multiple panes using the **RowDefinitions** and **ColumnDefinitions** properties.

Row Definitions

To split the chart area vertically into a number of rows, use RowDefinitions of the chart.

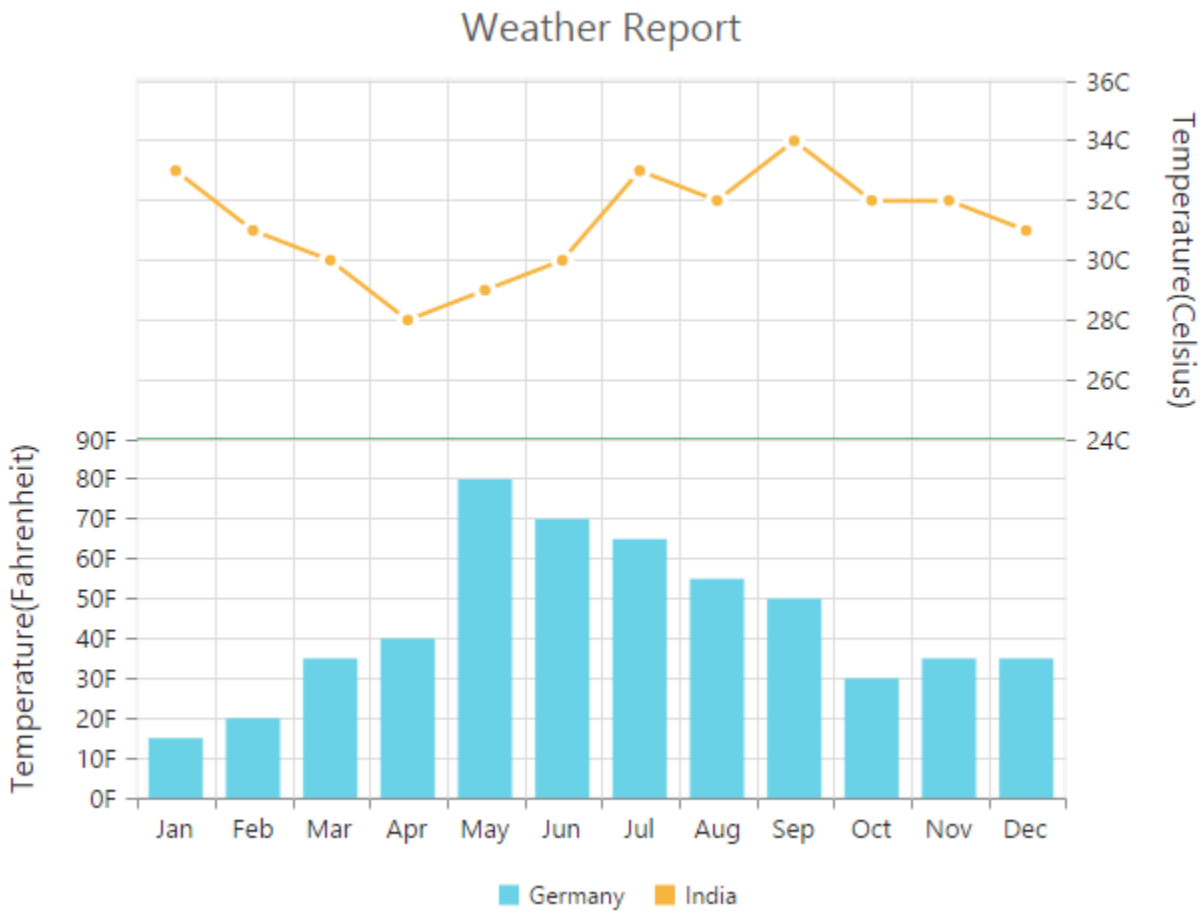
- You can allocate space for each row by using the **Unit** option that determines whether the chart area should be split by *percentage* or *pixels* for the given **RowHeight** value of the RowDefinitions.
- To associate a vertical axis to a row, specify the RowDefinitions **Index** value to the RowIndex property of the chart axis.
- To customize each row's horizontal line, use LineColor and LineWidth property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//  Splitting chart area into multiple rows
.RowDefinitions(rd =>
{
rd
//  Split first row of the chart area
.Unit("percentage").LineColor("Gray").LineWidth(0).RowHeight(50).Add();
//  Split second row of the chart area
rd.Unit("percentage").LineColor("green").LineWidth(0).RowHeight(50).Add();
})
.Axes(ax =>
{
//Create secondary axis and bind it to second row of chart area
ax.Name("yAxis1")
.RowIndex(1).Add();
})
.Series(sr =>
{
sr
//Binding vertical axis name
.YAxisName("yAxis1").Add();
})
//...
)
```



Click here to view the online demo sample for multiple panes.

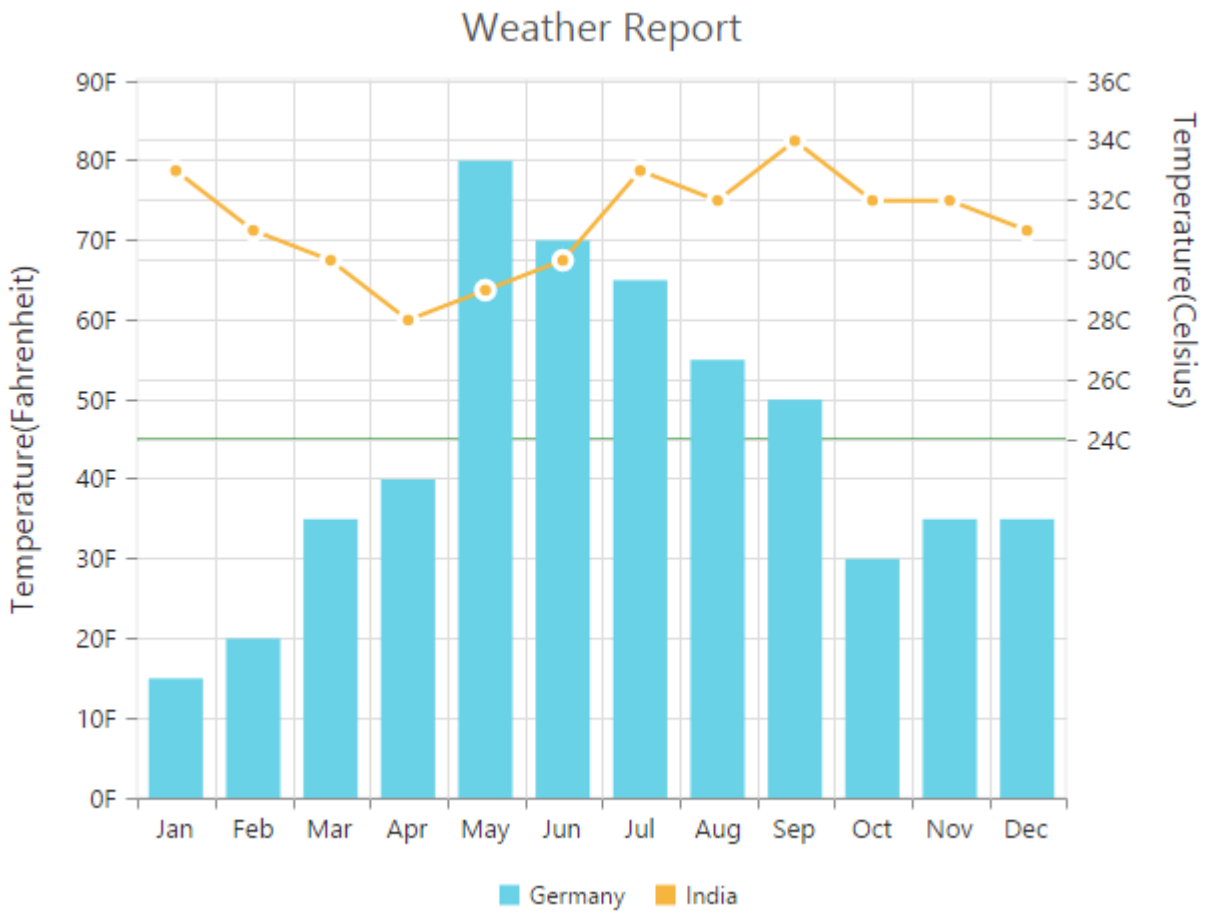
Row Span

For spanning the vertical axis along multiple panes vertically, you can use **RowSpan** property of axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//  Splitting chart area into multiple rows
.RowDefinitions(rd =>
{
//...
})
.Axes(ax =>
{
//..
})
.PrimaryYAxis(pry=>pry
//  Span the PrimaryYAxis
.RowSpan(2)
)
.Series(sr =>
{
//...
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Column Definitions

To split the chart area horizontally into a number of columns, use ColumnDefinitions of the chart.

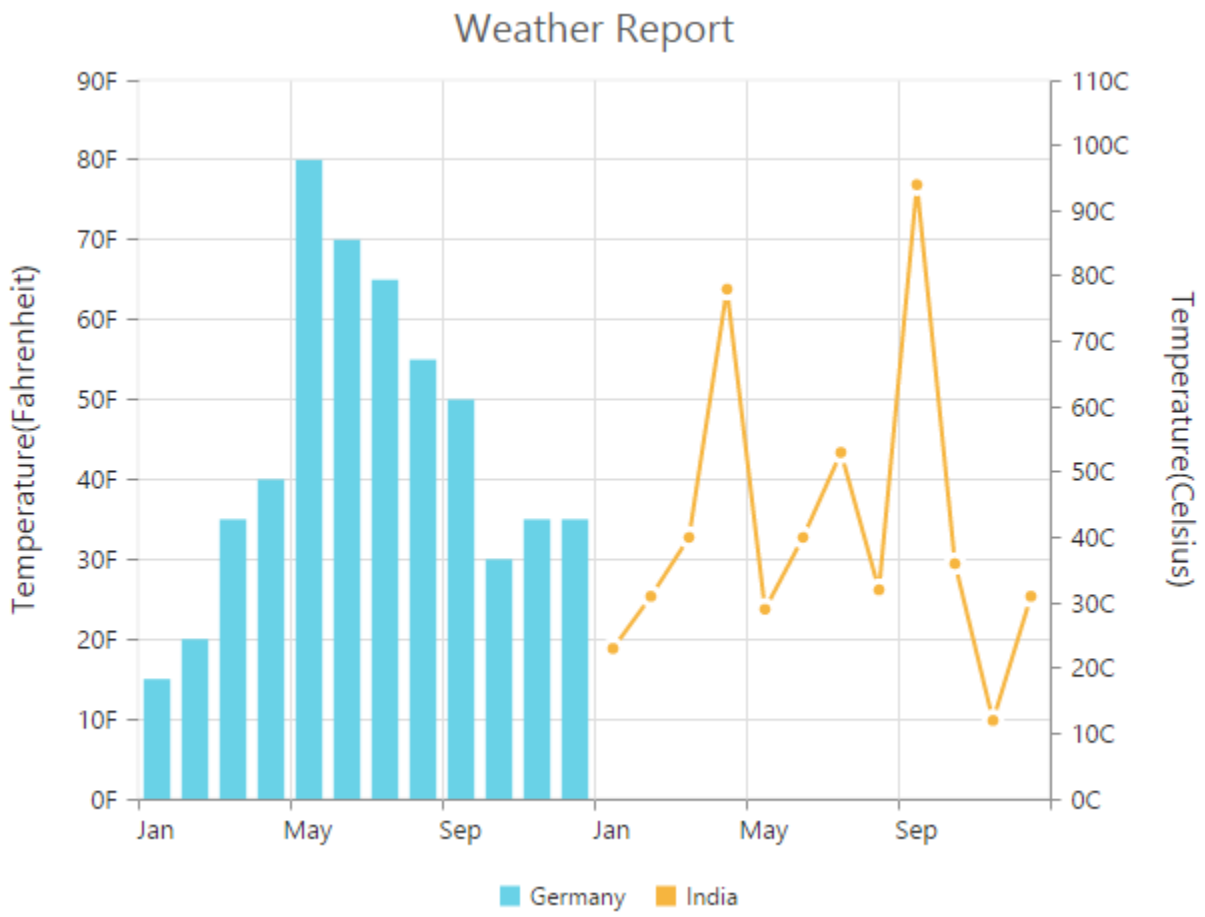
- You can allocate space for each column by using the **Unit** option that determines whether the chart area should be split by *percentage* or *pixels* for the given **ColumnWidth** value of the ColumnDefinitions.
- To associate a horizontal axis to a column, specify the ColumnDefinitions **Index** value to the **ColumnIndex** property of the chart axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//  Splitting chart area into multiple columns
.ColumnDefinitions(cdn =>
{
//  Split first column of the chart area
.Unit("percentage").ColumnWidth(50).Add();
//  Split second column of the chart area
cdn.Unit("percentage").ColumnWidth(50).Add();
})
.Axes(ax =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Create secondary axis and bind it to second row of chart area
ax.Name("xAxis1")
.ColumnIndex(1).Add();
})
.Series(sr =>
{
sr
//Binding horizontal axis name
.XAxisName("xAxis1").Add();
})
//...
)
```



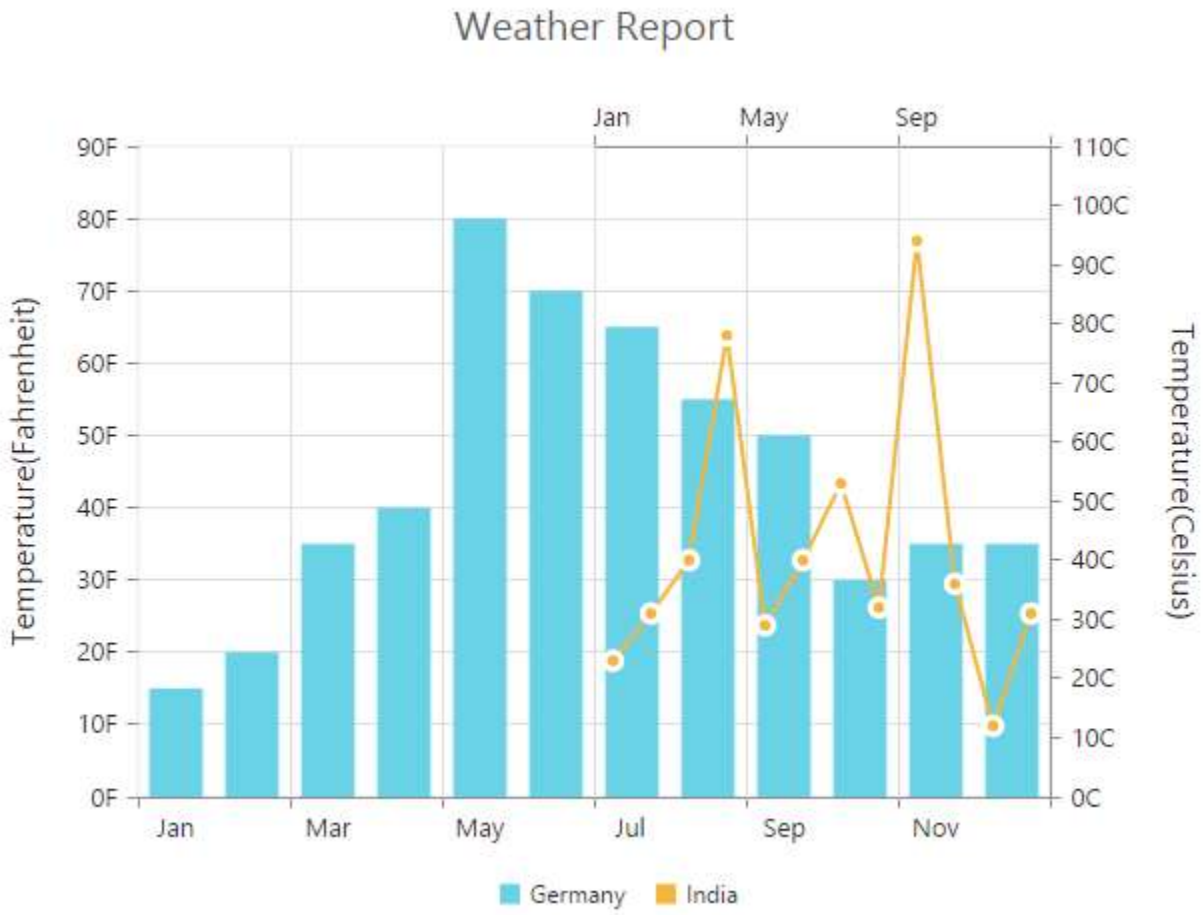
Column Span

For spanning the horizontal axis along multiple panes horizontally, you can use **ColumnSpan** property of axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//  Splitting chart area into multiple columns
.ColumnDefinitions(cdn =>
```

```
{
//...
})
.PrimaryXAxis(axis=>axis
//  Span the PrimaryXAxis
.Name("xAxis1").ColumnSpan(2)
)
.Axes(ax =>
{
//...
})
.Series(sr =>
{
//...
})
//...
)
```



ChartTypes

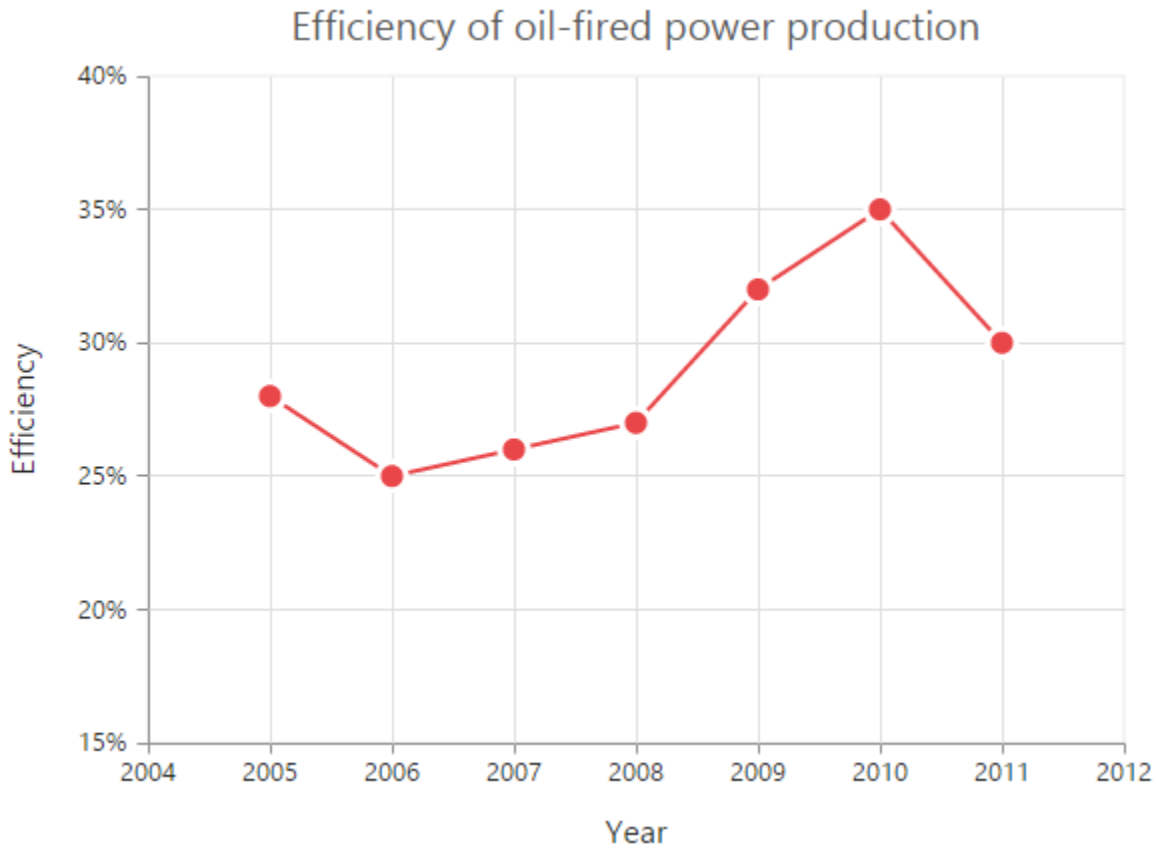
Line Chart

To render a Line Chart, set the series **Type** as **Line** in the chart series. To change the line segment color, you can use the **Fill** property of the series.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series.
sr.Type(SeriesType.Line).Fill("#E94649").Add();
})
//...
)
```



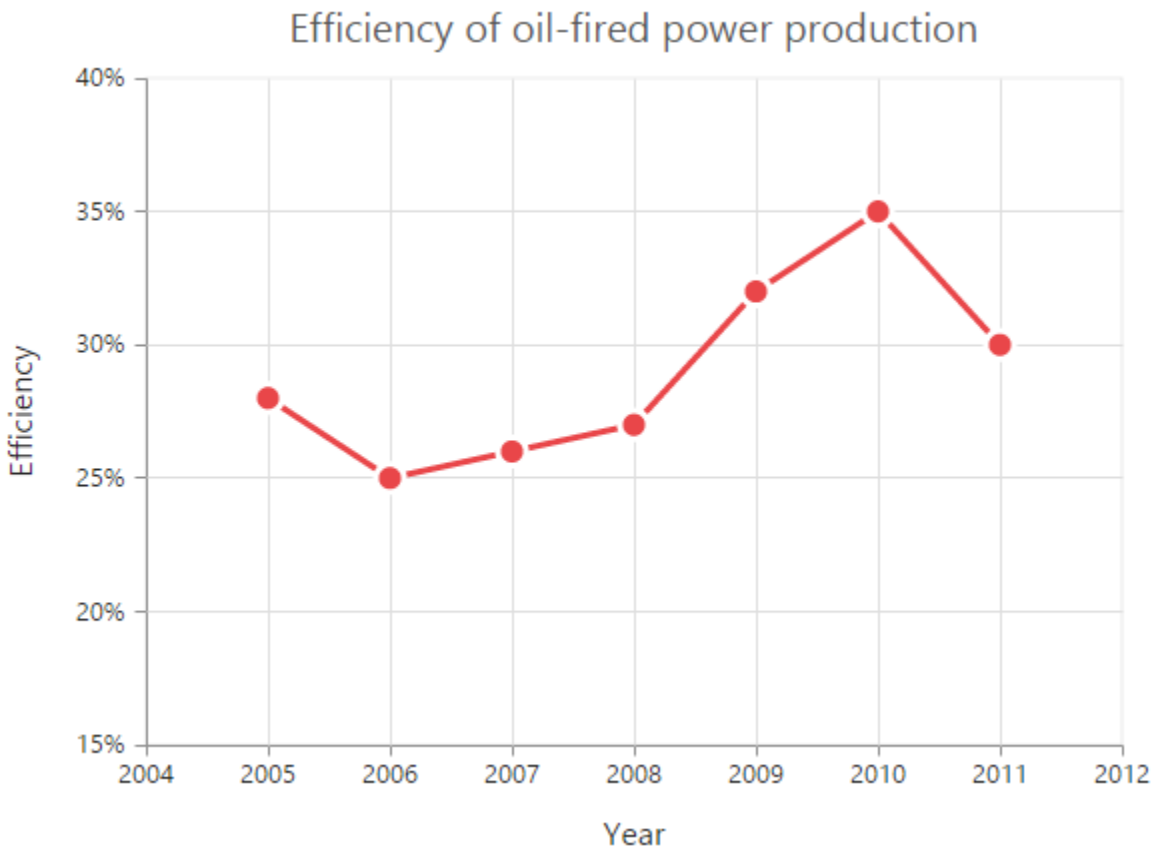
Click here to view the Line Chart online demo sample.

Change the line width

To change the width of the line segment, you can use the **Width** property in the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the width of line series
sr.Width(3).Add();
})
//...
)
```

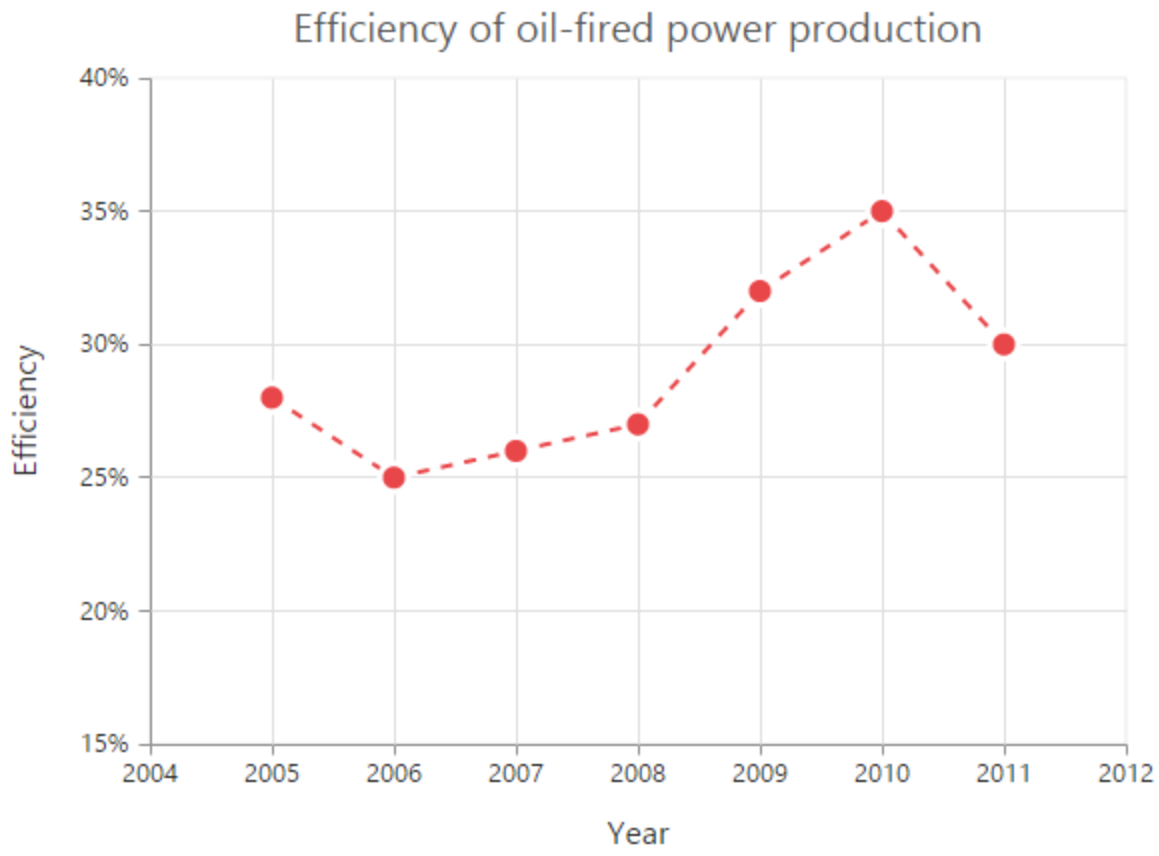


Dashed lines

To render the line series with dotted lines, you can use the **DashArray** option of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change dash array to display dotted or dashed lines
sr.DashArray("5,5").Add();
})
//...
)
```



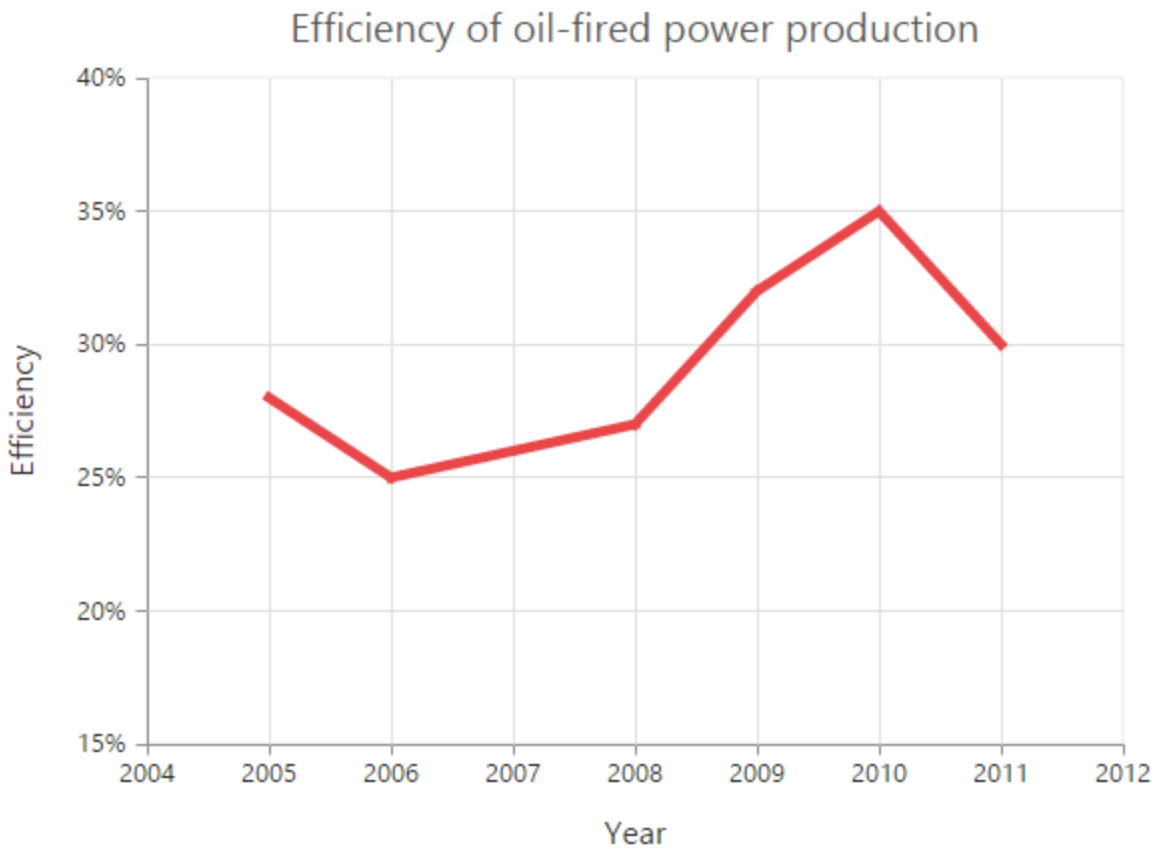
Changing the line cap

For customizing the start and end caps of the line segment, you can use the **LineCap** property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change line cap
sr.LineCap(ChartLineCap.Square).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



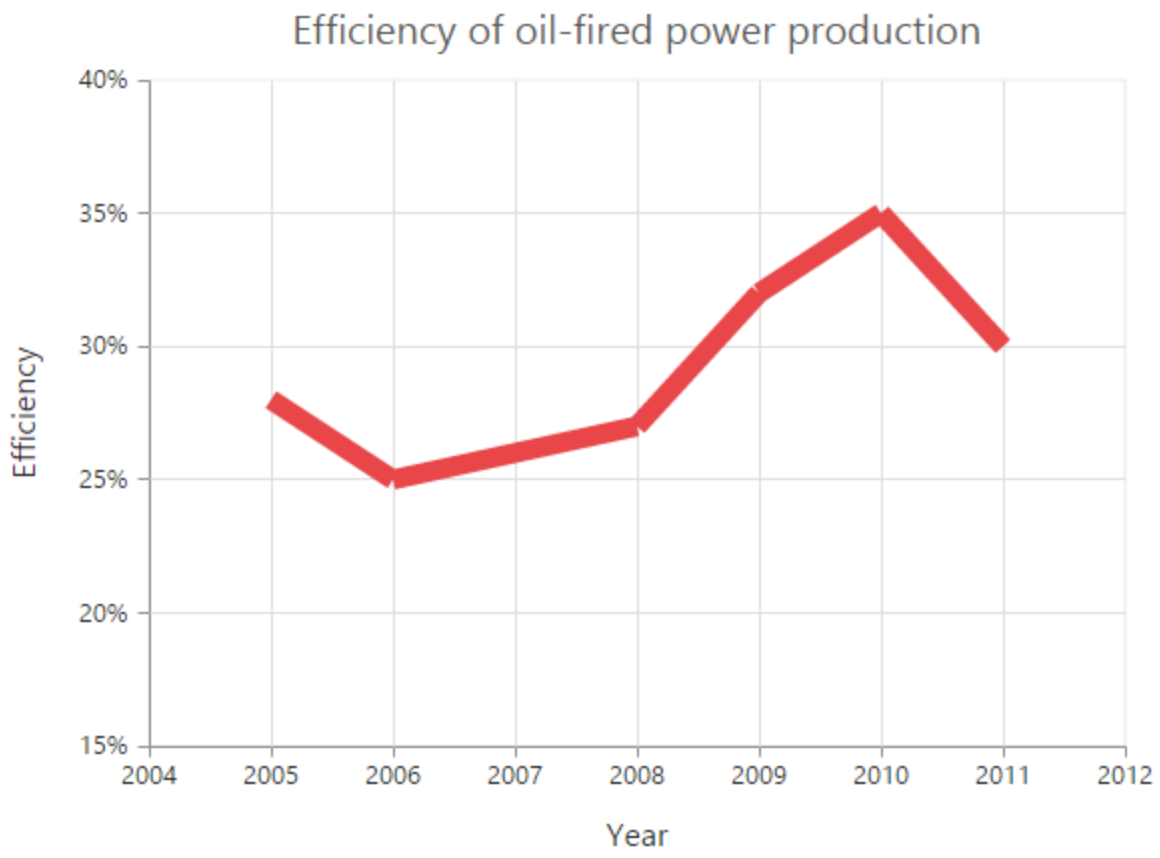
Changing the line join

You can use the **LineJoin** property to specify how two intersecting line segments should be joined.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change line join
sr.LineJoin(ChartLineJoin.Round).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

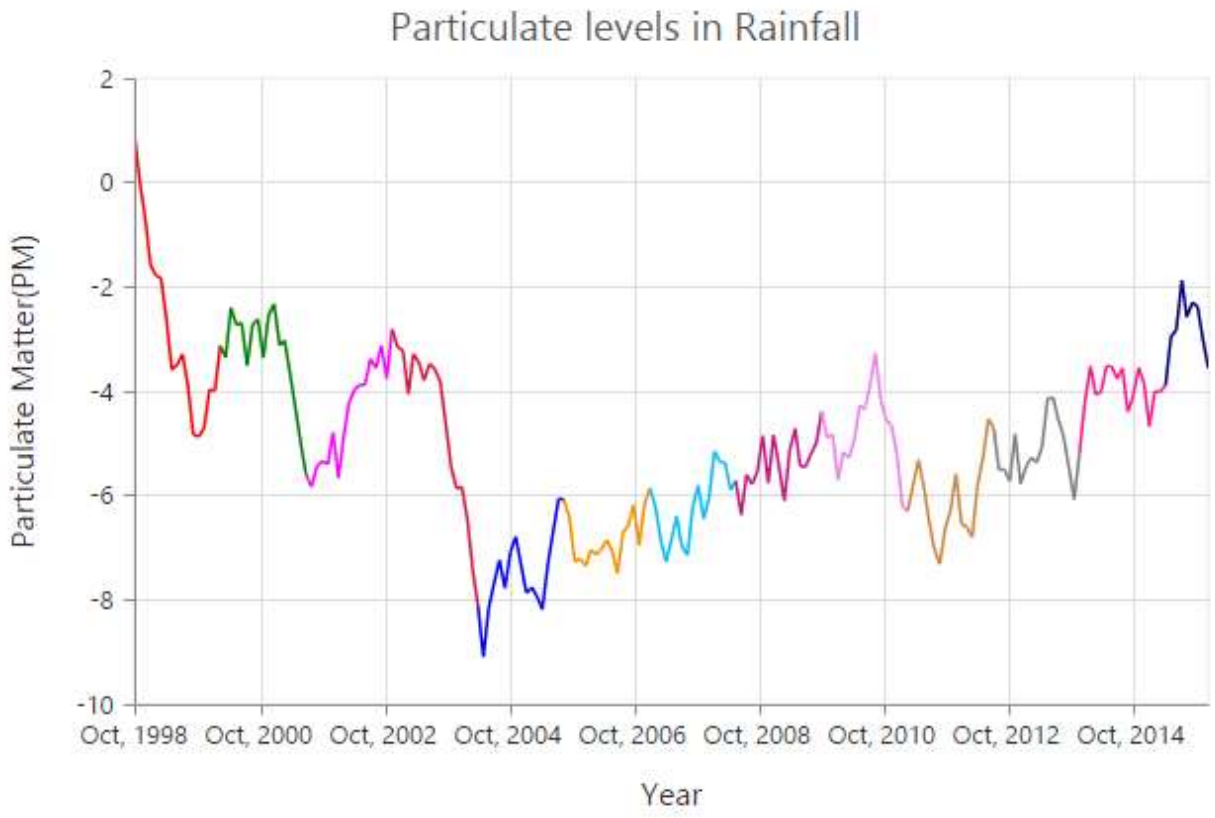


MultiColor Line

You can change the color of the line segments by using the *Fill* property of the each *Points* in the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding Candle series
sr.Points(pt=>{
pt.Fill("red").Add();
}) .Add();
})
//...
)
```



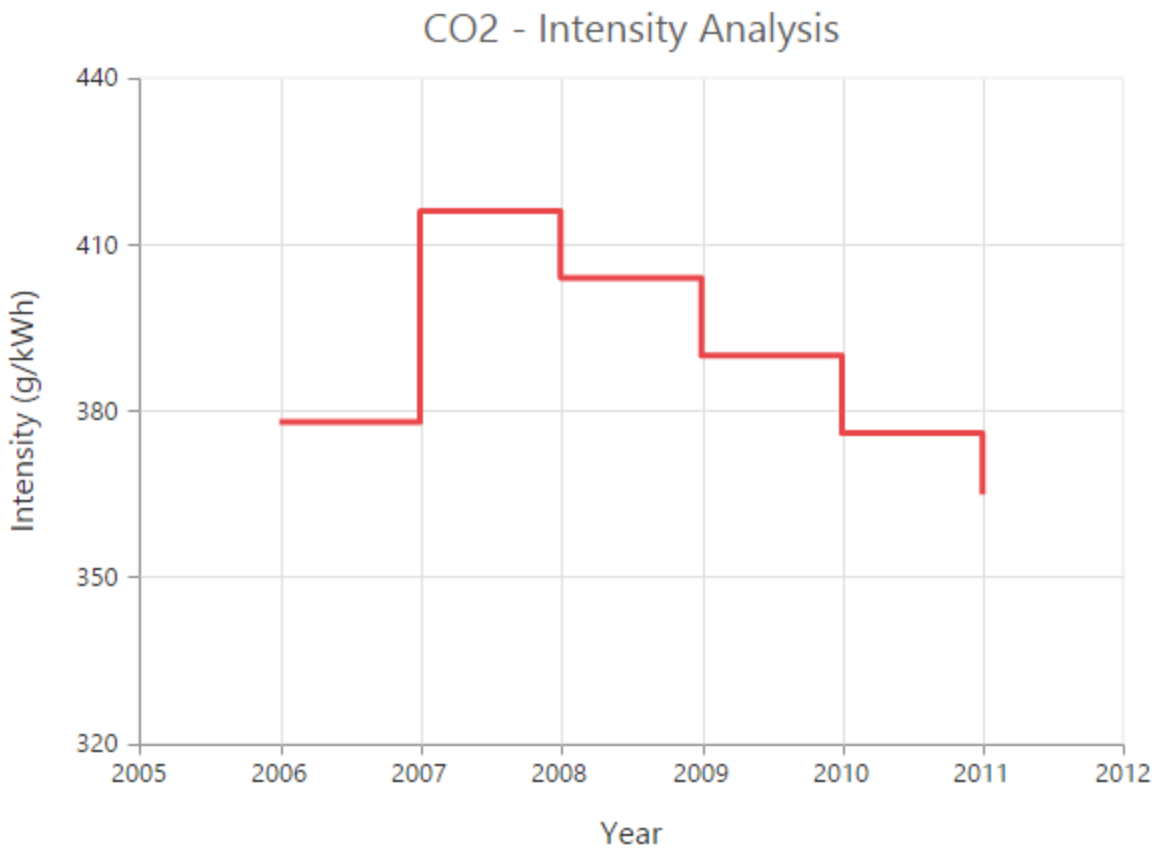
Click here to view the MultiColor Line Chart online demo sample.

Step Line Chart

To render a Step Line Chart, set the series Type as **StepLine"** in the chart series. To change the StepLine segment color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series.
sr.Type(SeriesType.StepLine).Fill("#E94649").Add();
})
//...
)
```



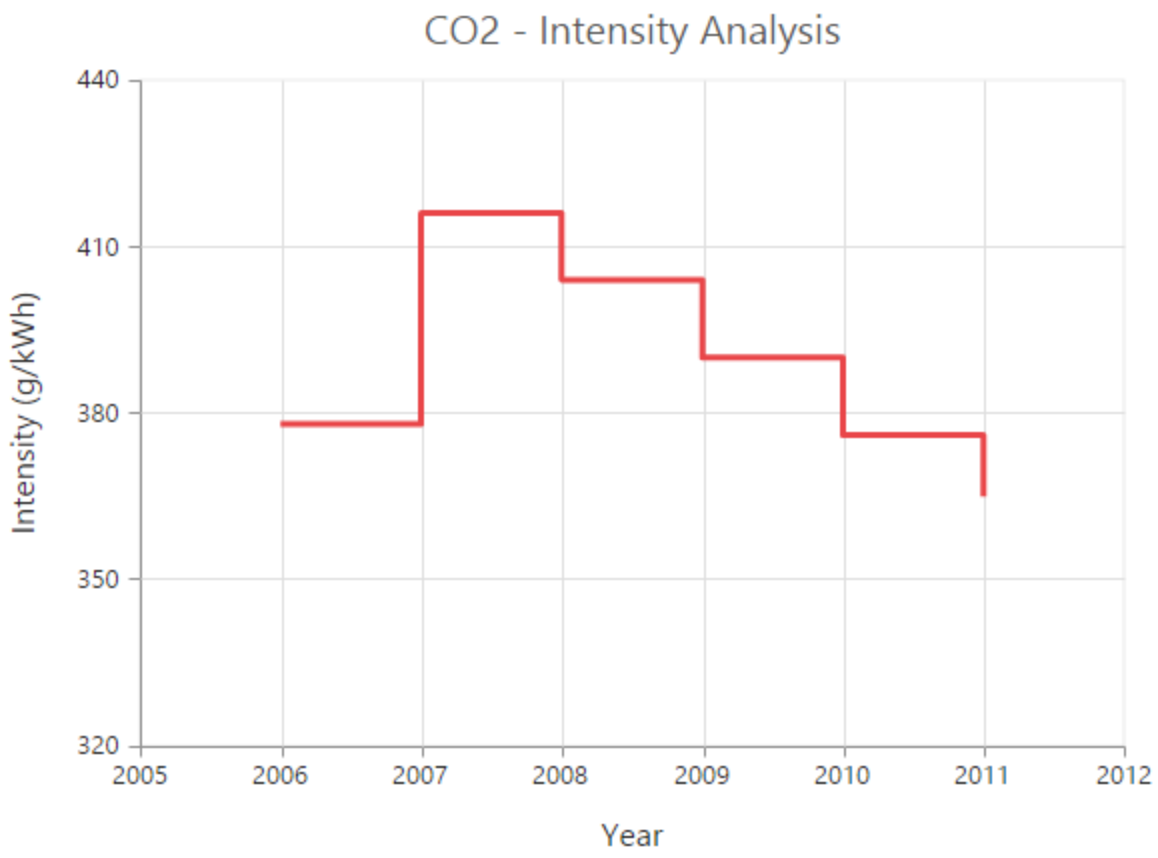
Click here to view the Step Line Chart online demo sample.

Changing the line width

To change the line width, you can use the **Width** property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the width of step line series
sr.Width(3).Add();
})
//...
)
```

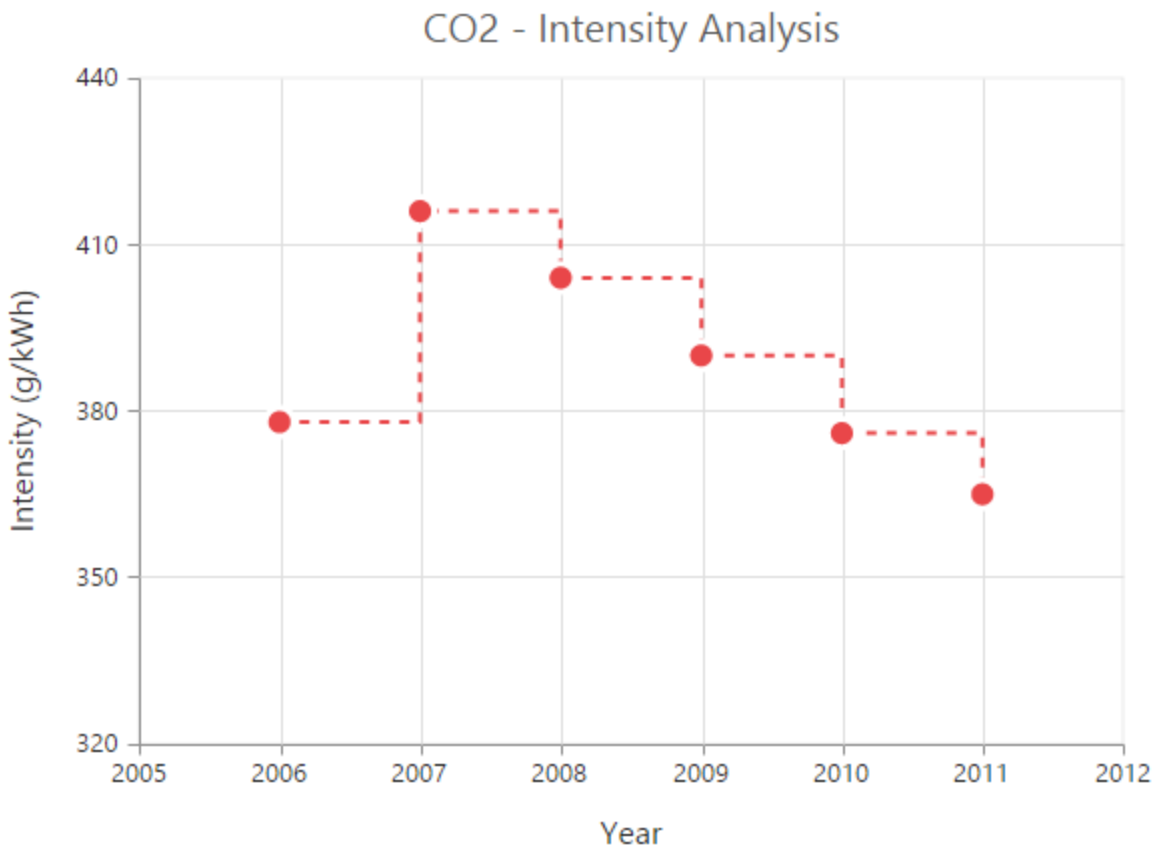


Dashed lines

To render the step line series with dotted lines, you can use the DashArray option of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change dash array to display dotted or dashed lines
sr.DashArray("5,5").Add();
})
//...
)
```



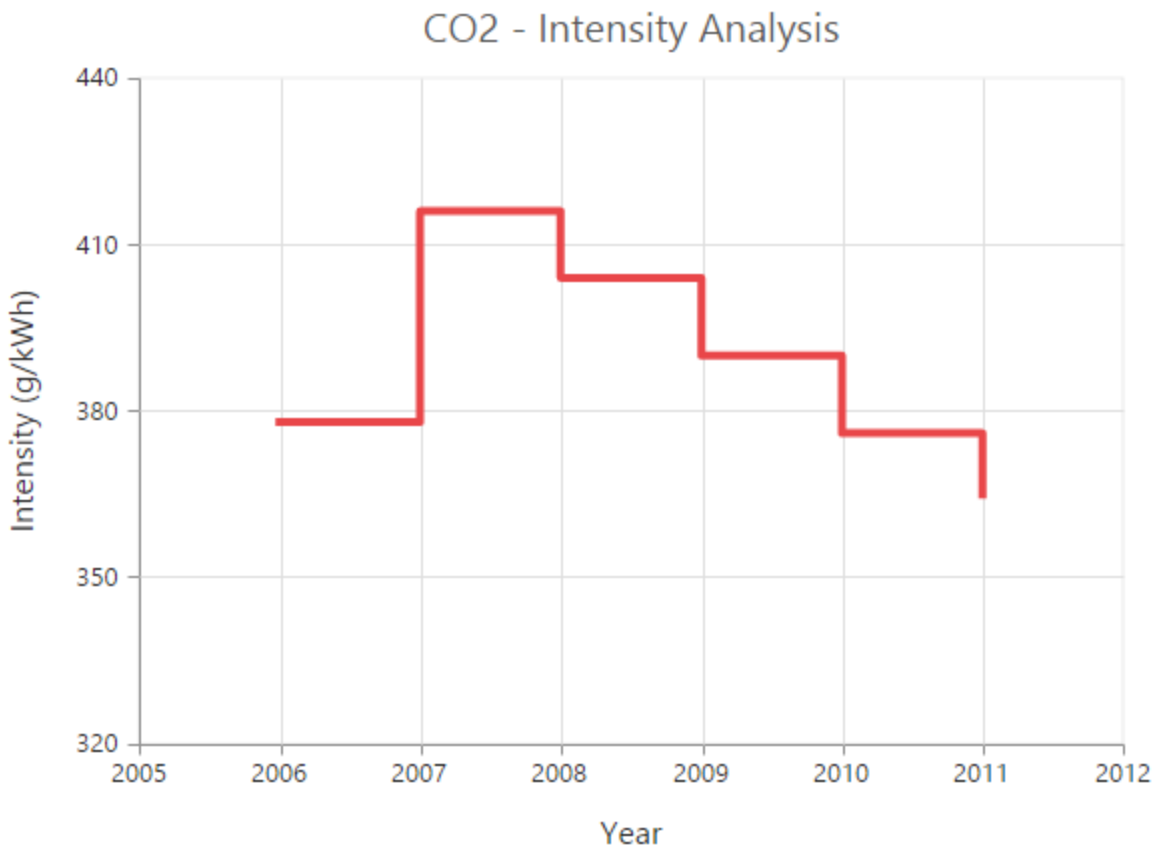
Changing the line cap

For customizing the start and end caps of the line segment, you can use the LineCap property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change line cap
sr.LineCap(ChartLineCap.Square).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



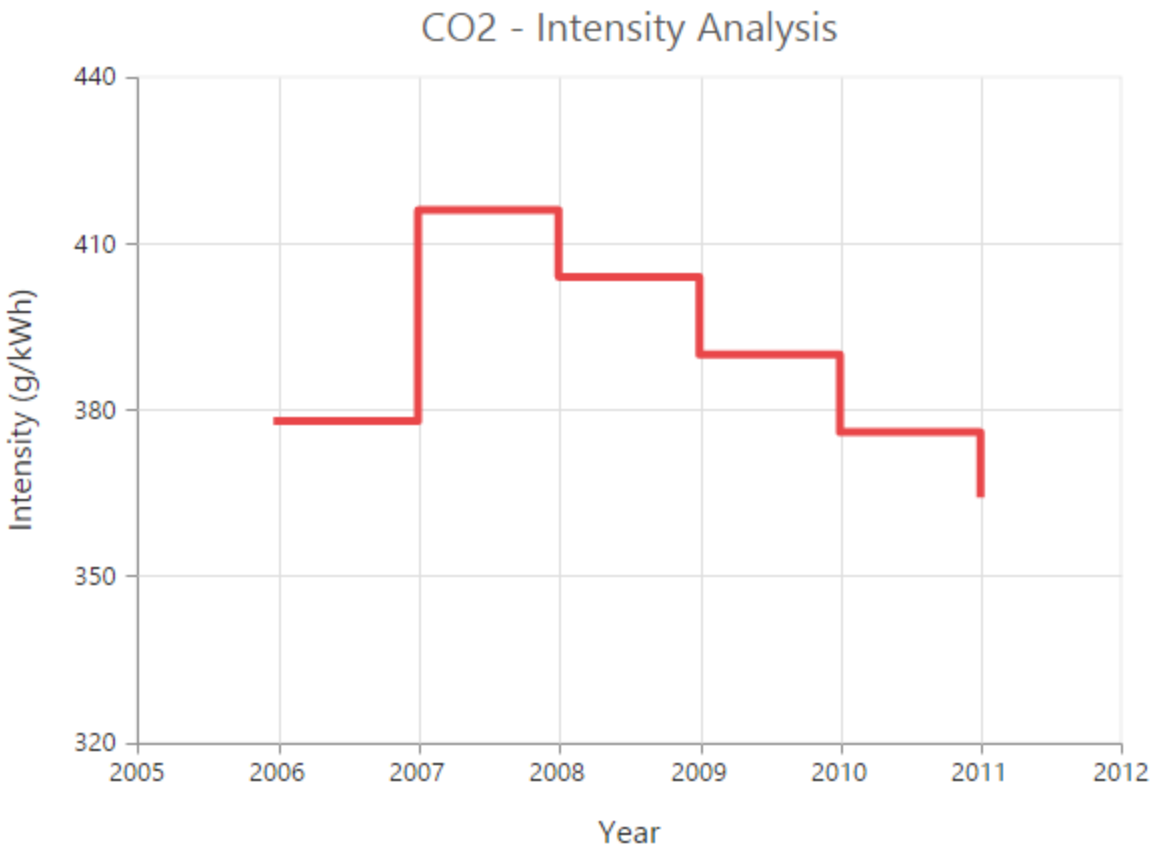
Changing the line join

You can use the LineJoin property to specify how two intersecting line segments should be joined.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change line join
sr.LineJoin(ChartLineJoin.Round).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



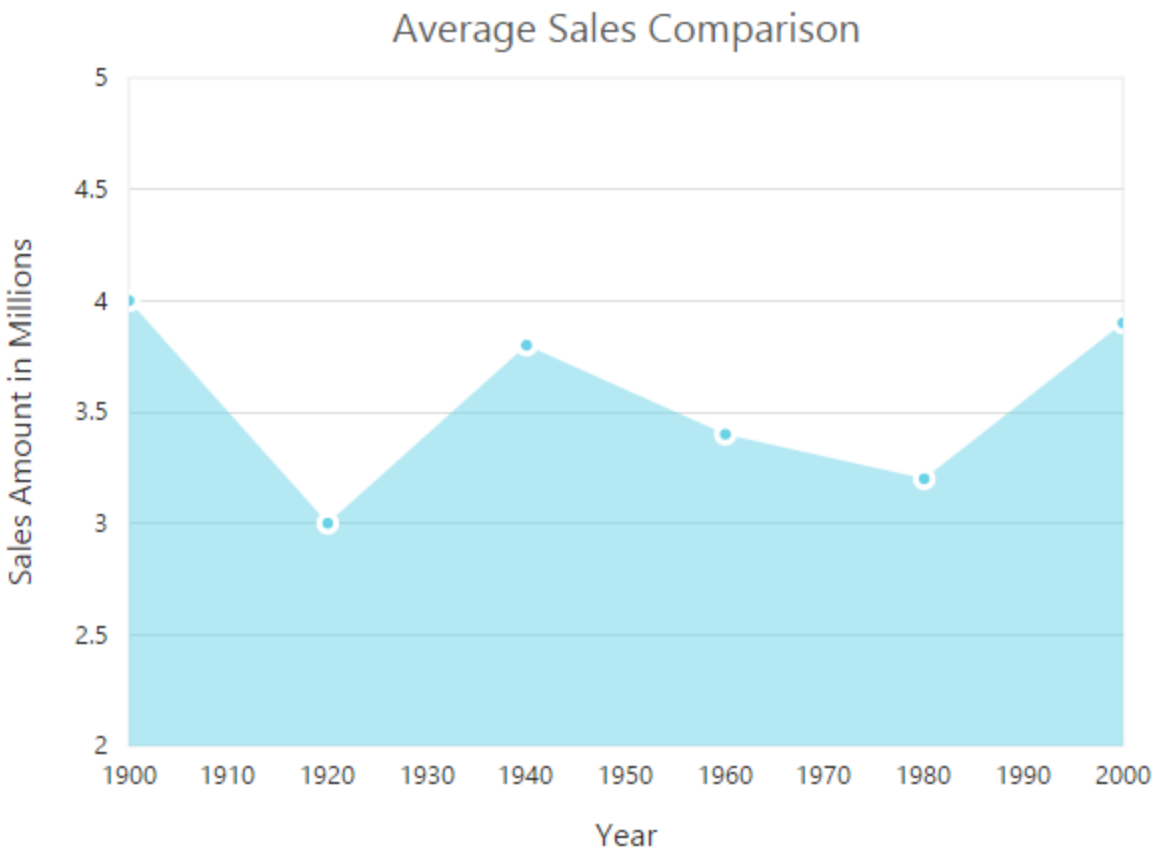
Area Chart

To render an Area chart, you can specify the series Type as **Area** in the chart series. To change the Area color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series.
sr.Type(SeriesType.Area).Fill("#69D2E7").Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Area Chart online demo.

Range Area Chart

To render a Range Area Chart, set the Type as **RangeArea** in the chart series. To change the RangeArea color, you can use the Fill property of the series.

Since the RangeArea series requires two y values for a point, you have to add the **High** and **Low** value. High and Low value specifies the maximum and minimum range of the points.

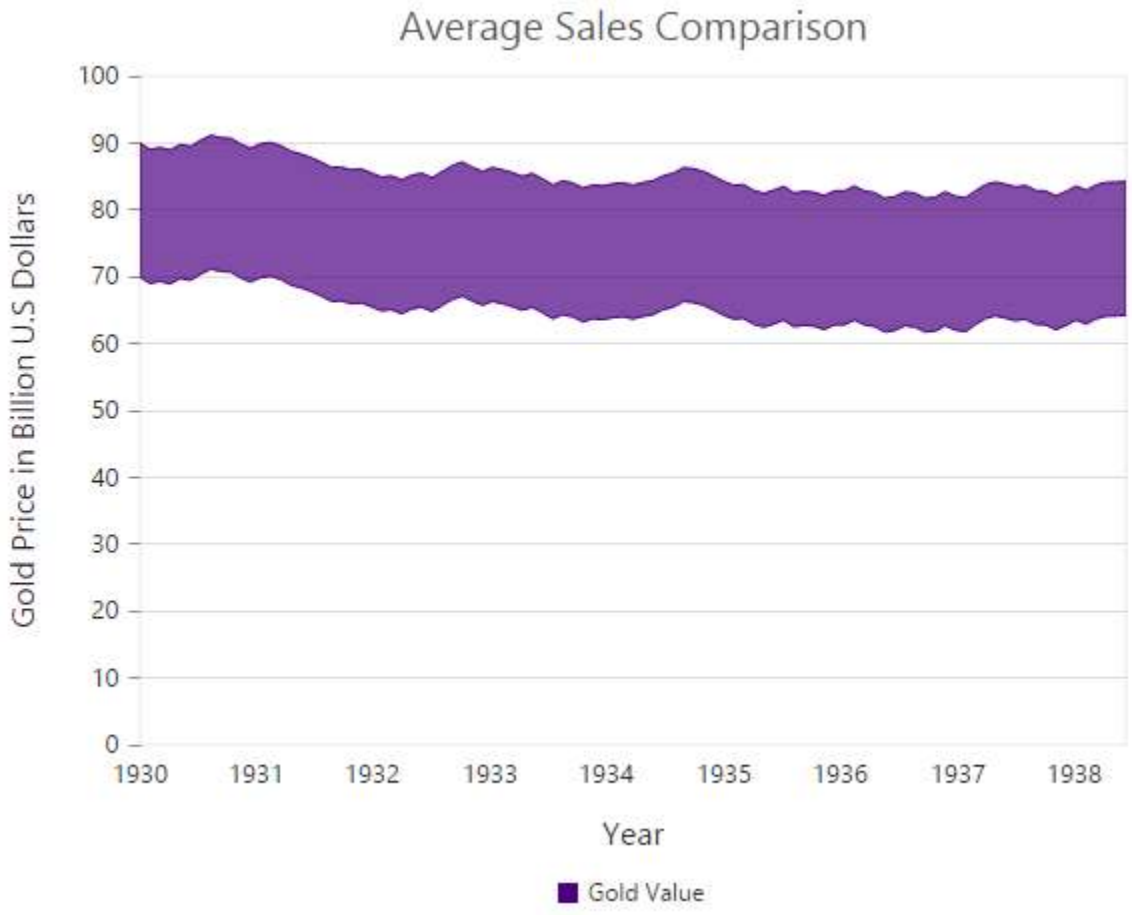
- When you are using the Points option, specify the high and low values by using the High and Low option of the point.
- When you are using the DataSource option to assign the data, map the fields from the dataSource that contain High and Low values by using the Series.High and Series.Low options.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the series type and fill color
sr.Type(SeriesType.RangeArea).Fill("Indigo")
//Use High and Low values instead of Y
.Points(pt=>{pt.X("1935").High(80).Low(70).Add();
//..
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Add();
})
//...
)
```



Click here to view Range Area Chart online demo.

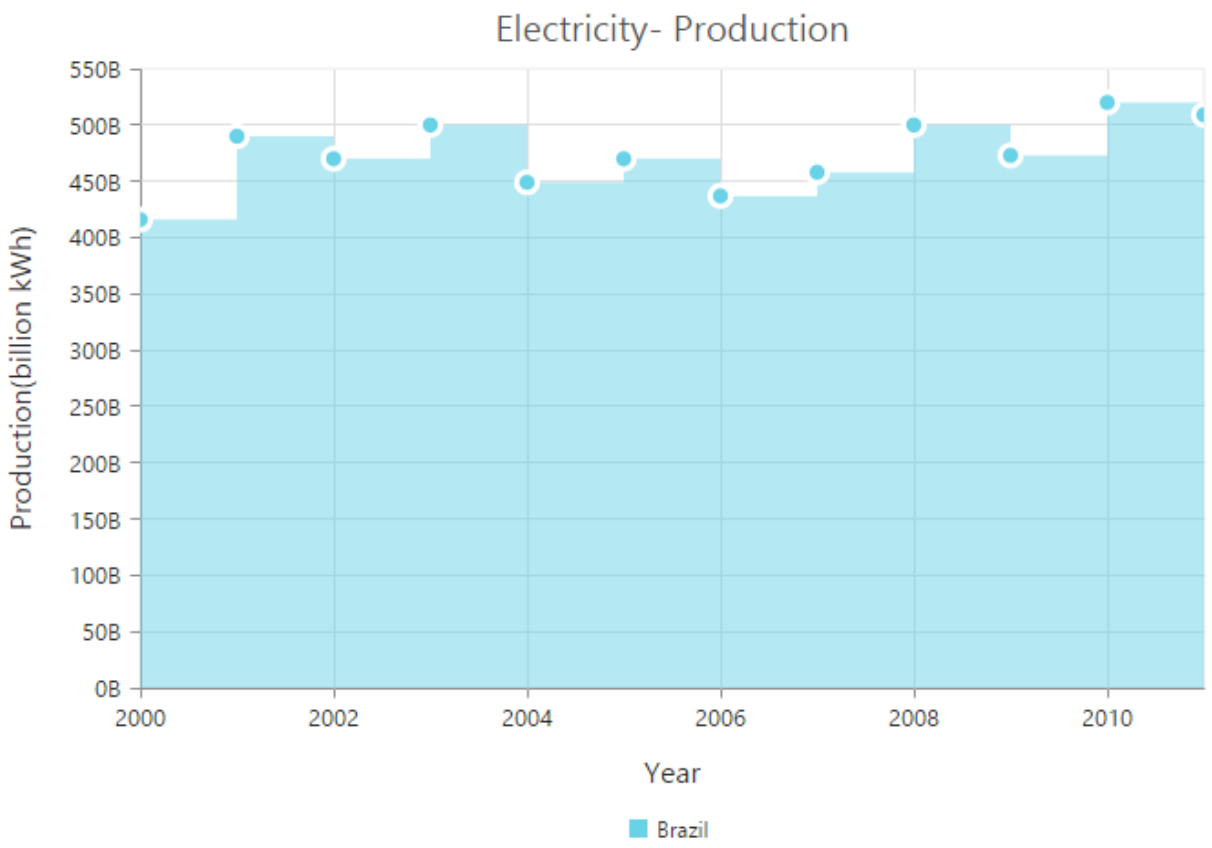
Step Area Chart

To render a Step Area Chart, set the Type as **StepArea** in the chart series. To change the StepArea color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the series type and fill color
sr.Type(SeriesType.StepArea).Fill("#69D2E7")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



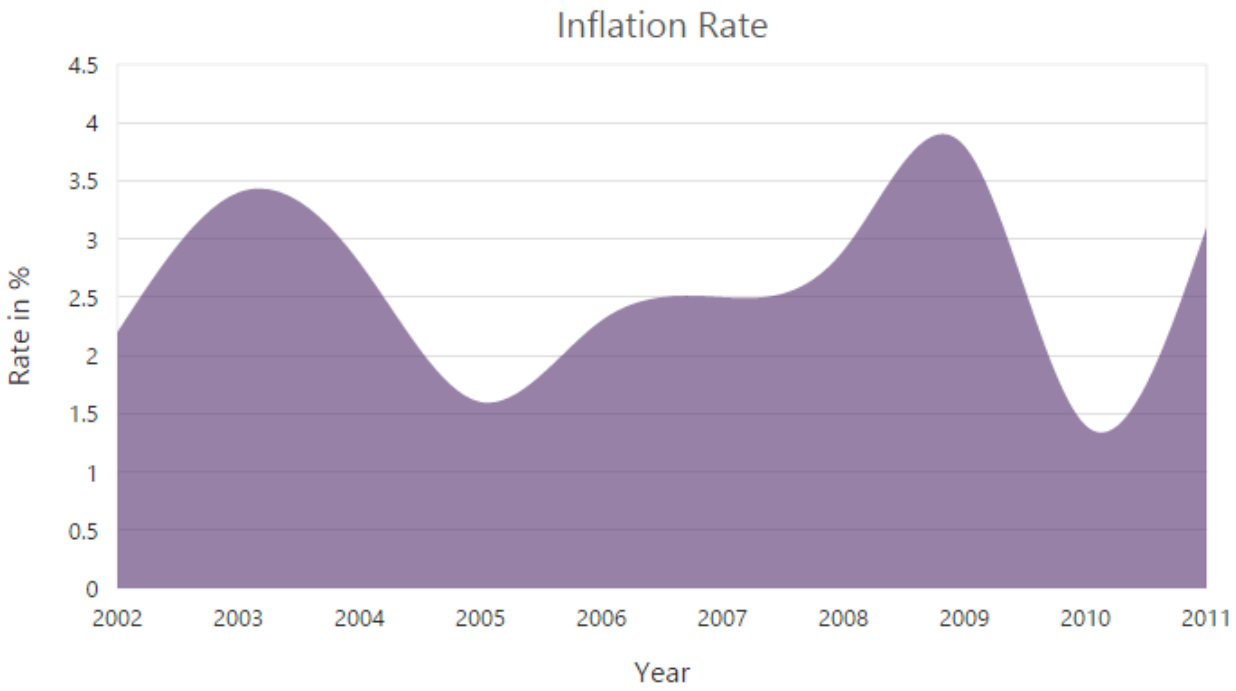
Click here to view Step Area Chart online demo.

Spline Area Chart

To render a Spline Area Chart, set the Type as **SplineArea** in the chart series. To change the SplineArea color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the series type and fill color
sr.Type(SeriesType.SplineArea).Fill("#C4C24A")
//...
.Add();
})
//...
)
```



Click here to view Spline Area Chart online demo.

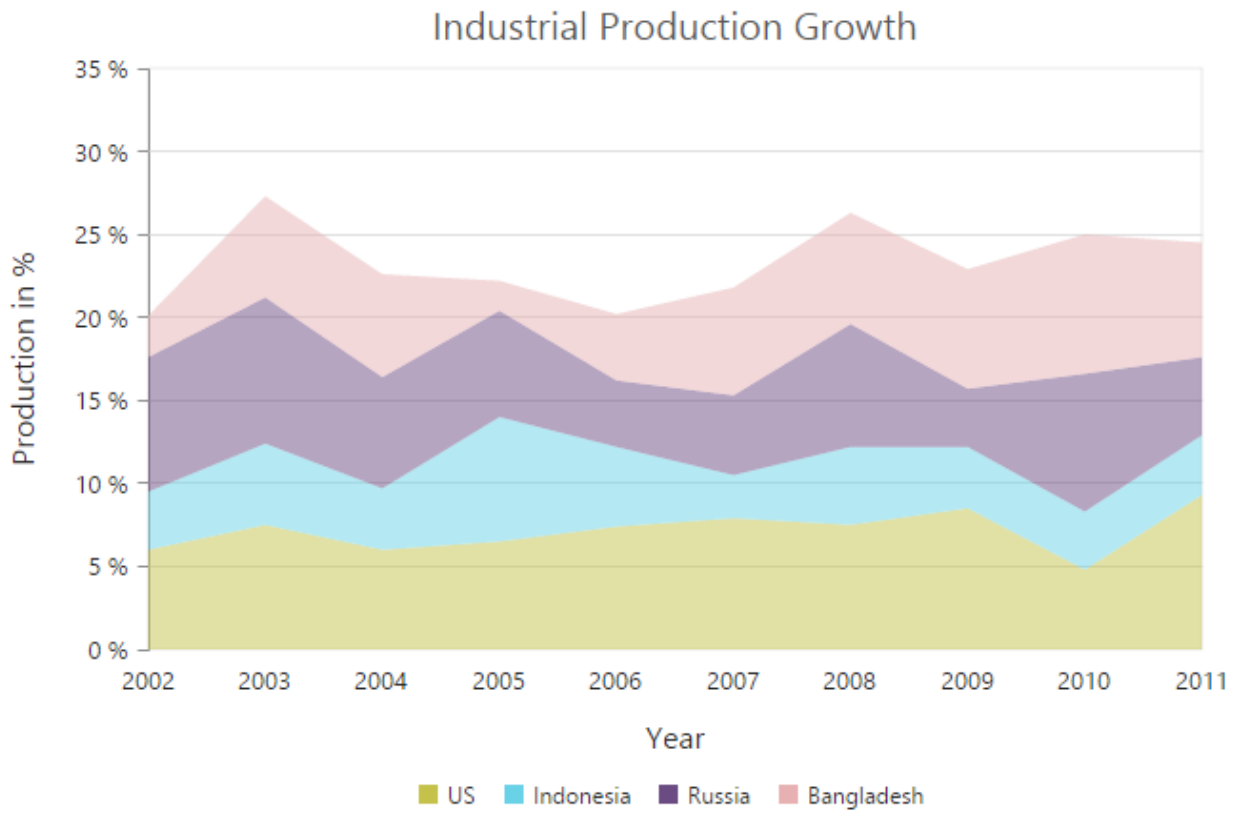
Stacked Area Chart

To render a Stacked Area Chart, set the Type as **StackingArea** in the chart series. To change the StackingArea color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the series type and fill color
sr.Type(SeriesType.StackingArea).Fill("#69D2E7")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



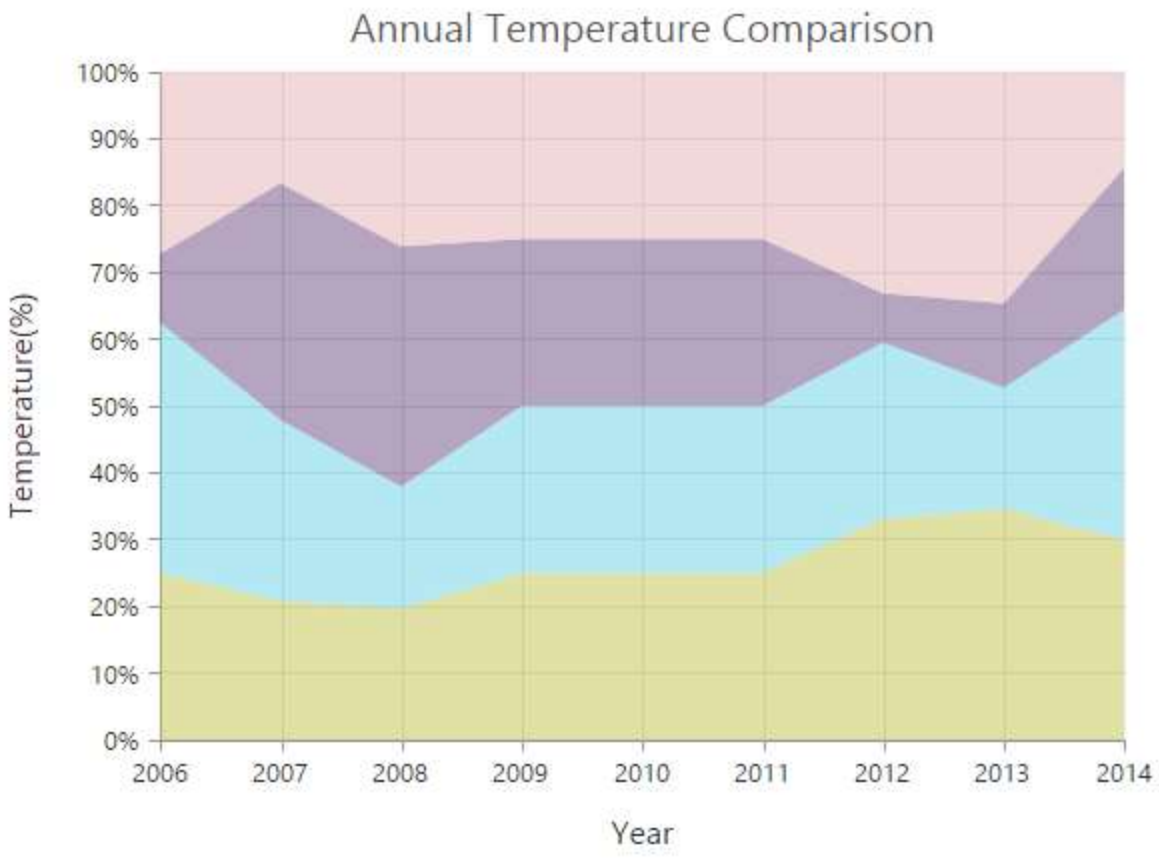
Click here to view Stacked Area Chart online demo.

100% Stacked Area Chart

To render a 100% Stacked Area Chart, set the Type as **StackingArea100** in the chart series. To change the StackingArea100 color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the series type and fill color
sr.Type(SeriesType.StackingArea100).Fill("#C4C24A")
//...
.Add();
})
//...
)
```



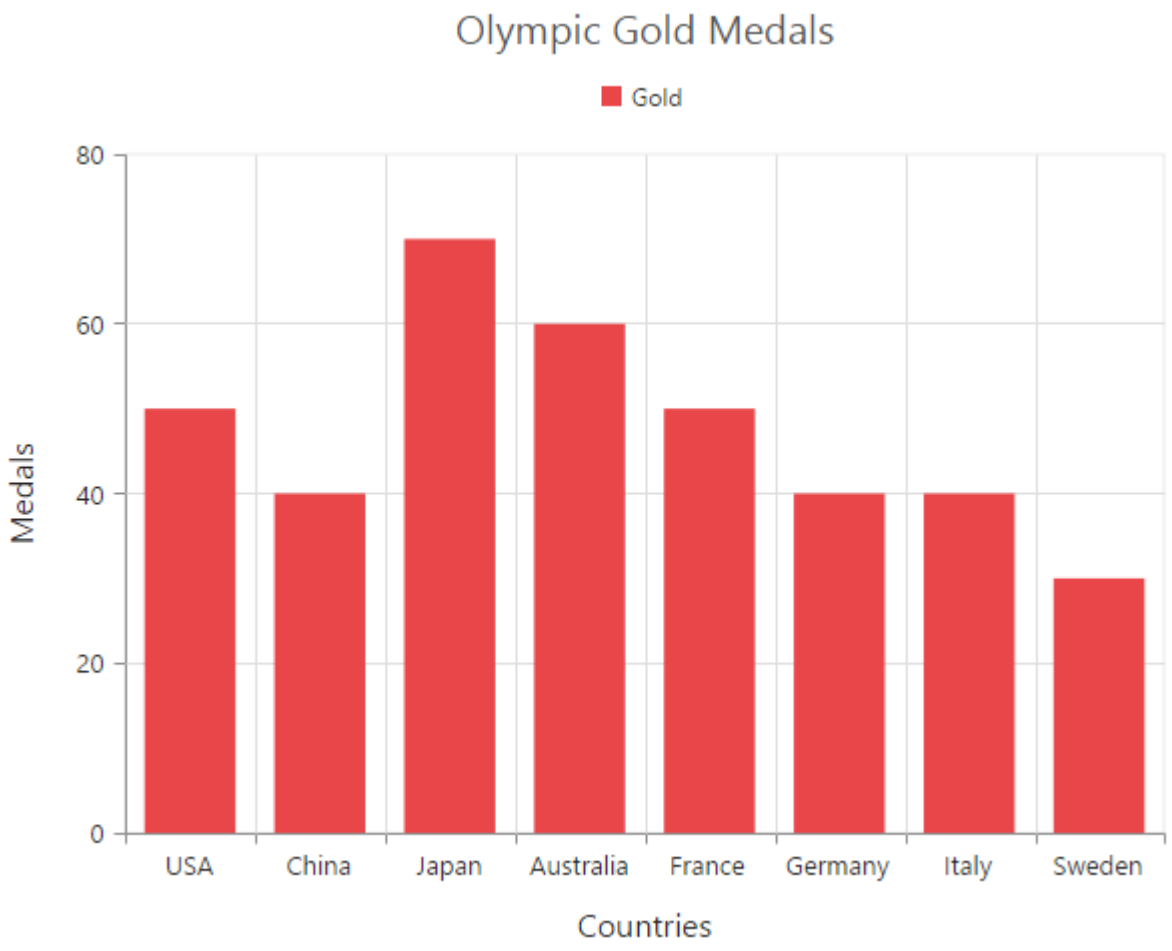
Click here to view 100% Stacked Area Chart online demo.

Column Chart

To render a Column Chart, set the Type as **Column** in the chart series. To change the color of the column series, you can use the Fill property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the series type and fill color
sr.Type(SeriesType.Column).Fill("#E94649")
//...
.Add();
})
//...
)
```



Click here to view Column Chart demo.

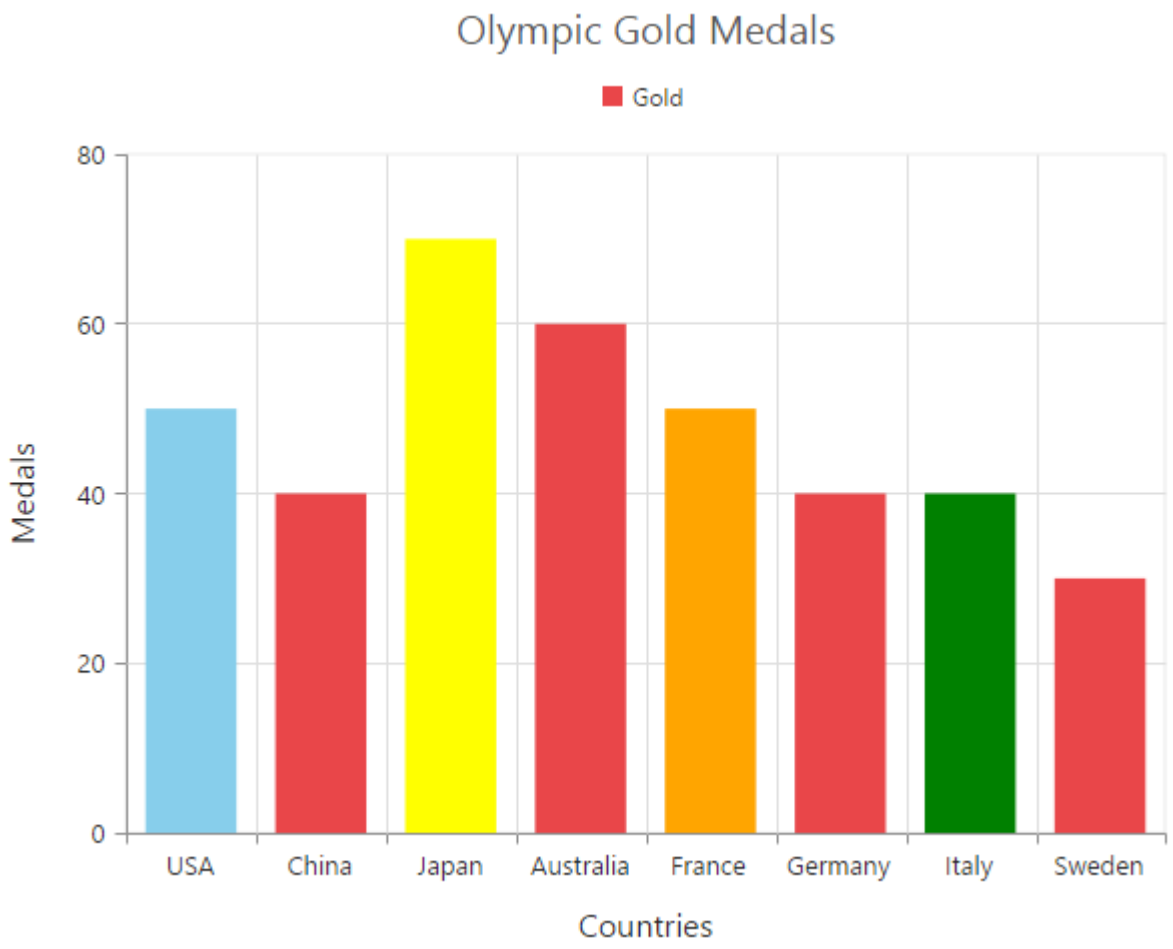
Change a point color

You can change the color of a column by using the Fill property of the point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Column width customization

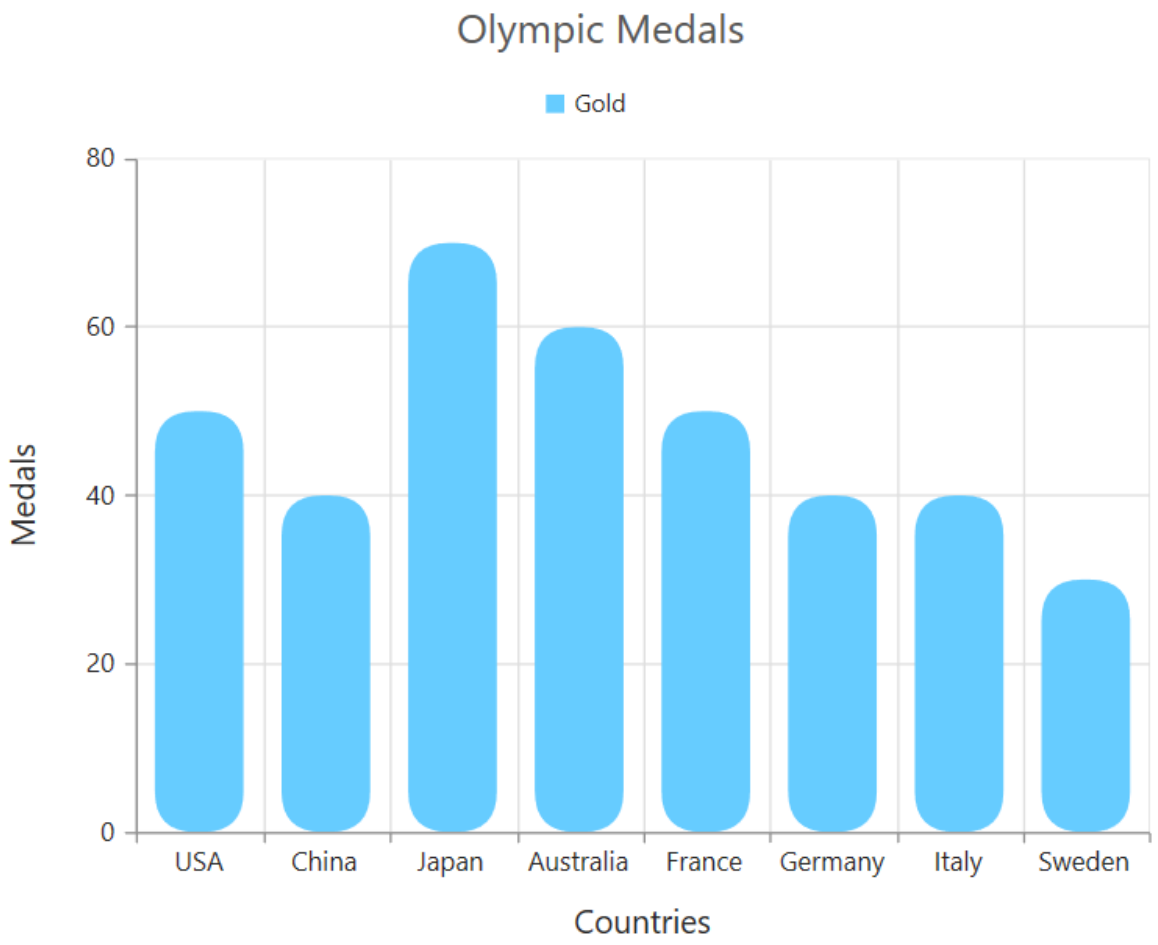
Width of any column type series can be customized by using the **ColumnWidth** property. Default value of *ColumnWidth* is 0.7. Value ranges from 0 to 1. Here 1 corresponds to 100% of available width and 0 corresponds to 0% of available width.

Note: Width of a column also depends upon the *ColumnSpacing* property, because *ColumnSpacing* will reduce the space available for drawing a column. This is also applicable for StackingColumn, StackingColumn100, Bar, StackingBar, StackingBar100, RangeColumn, HiLo, HiLoOpenClose, Candle and Waterfall charts.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//Customizing column width for a specific series
sr.ColumnWidth(0.8).Add();
//...
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Column with rounded corners

Corners of the column chart can be customized by setting value to the **CornerRadius** property.

ASPX-CS

```
@(Html.EJ().Chart("container")
.CommonSeriesOptions(options => options.CornerRadius(20)
))
```



Spacing between column series

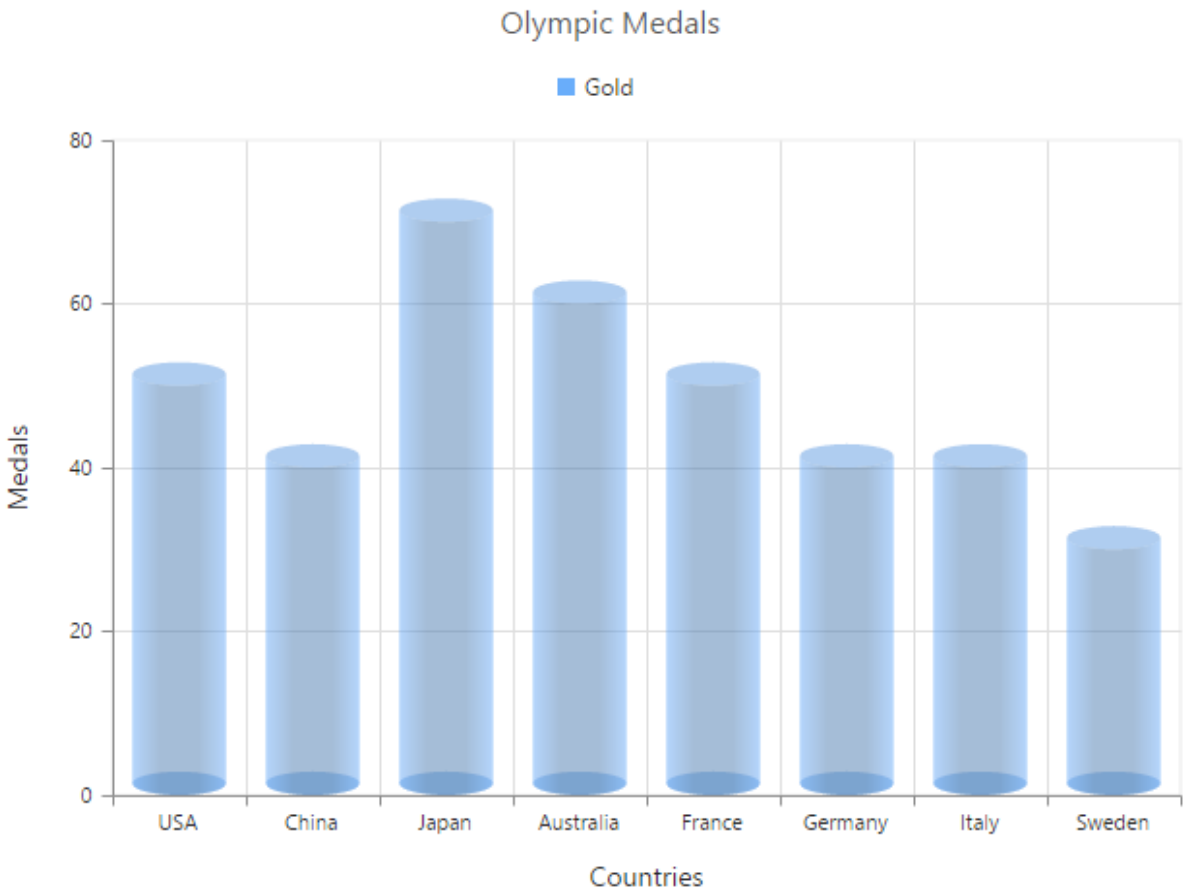
Spacing between column type series can be customized using the **ColumnSpacing** property. Default value of *ColumnSpacing* is 0. Value ranges from 0 to 1. Here 1 corresponds to 100% available space and 0 corresponds to 0% available space.

Note: Space between columns will also affect the width of the column. For example, setting 20% spacing and 100% width will render columns with 80% of total width. This is also applicable for StackingColumn, StackingColumn100, Bar, StackingBar, StackingBar100, RangeColumn, HiLo, HiLoOpenClose, Candle and Waterfall charts.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
//Customizing column spacing for a specific series
sr.ColumnSpacing(0.2).Add();
//...
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Cylindrical Chart

To render a cylindrical chart, set the **ColumnFacet** property as "Cylinder" in the chart series along with the series type.

The following chart types can be rendered as cylinder in both 2D and in 3D view.

- Column Chart, Bar Chart, Stacked Column Chart, Stacked Bar Chart, 100% Stacked Column Chart, 100% Stacked Bar Chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
// To change the shape of the series.
sr.Type(SeriesType.Line).ColumnFacet(ColumnFacet.Cylinder).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



RangeColumn Chart

To render a Range Column Chart, set the Type as **RangeColumn** in the chart series. To change the RangeColumn color, use the Fill property of the series.

Since, the RangeColumn series requires two y values for a point, add the High and Low value. High and Low value specifies the maximum and minimum range of the points.

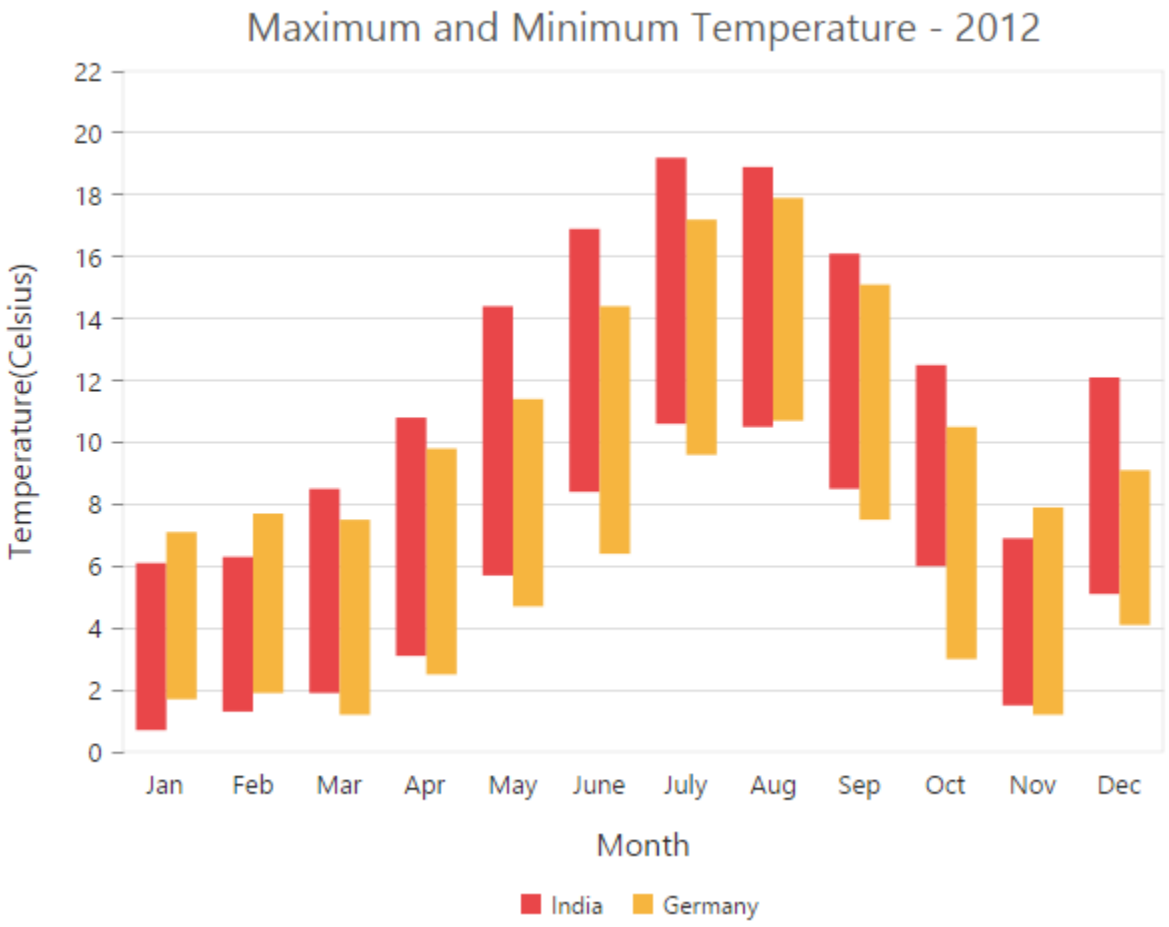
- When you are using the Points option, specify the high and low values by using the High and Low option of the point.
- When you are using the DataSource option to assign the data, you have to map the fields from the dataSource that contains high and low values by using the Series.High and Series.Low) options.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.RangeColumn).Fill("#E94649")
//Use high and low values instead of y
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Points(pt=>{pt.High("6.1").Low(0.7).Add();})
//...
.Add();
})
//...
)
```



Click here to view Range Column Chart online demo.

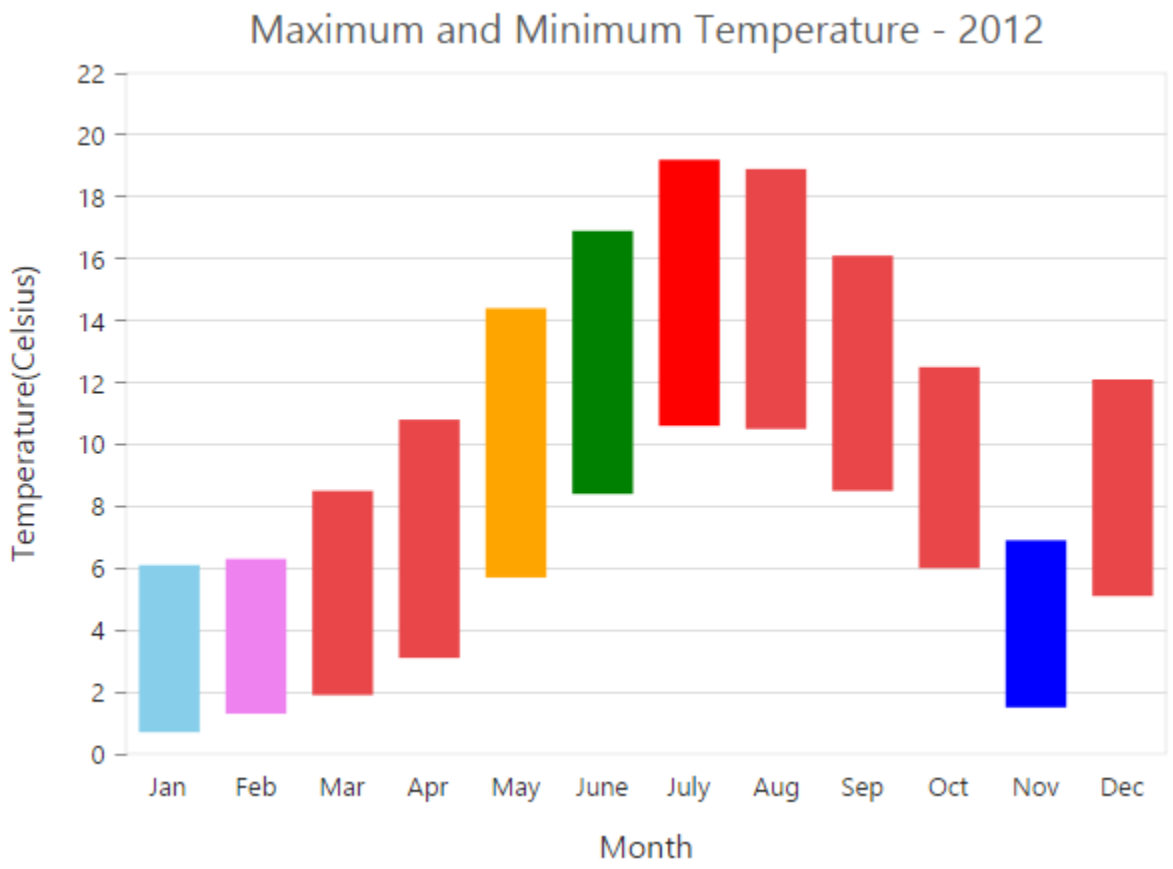
Change a point color

To change the color of a range column, you can use the Fill property of point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//change a point color of Range column
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
```

```
)
```

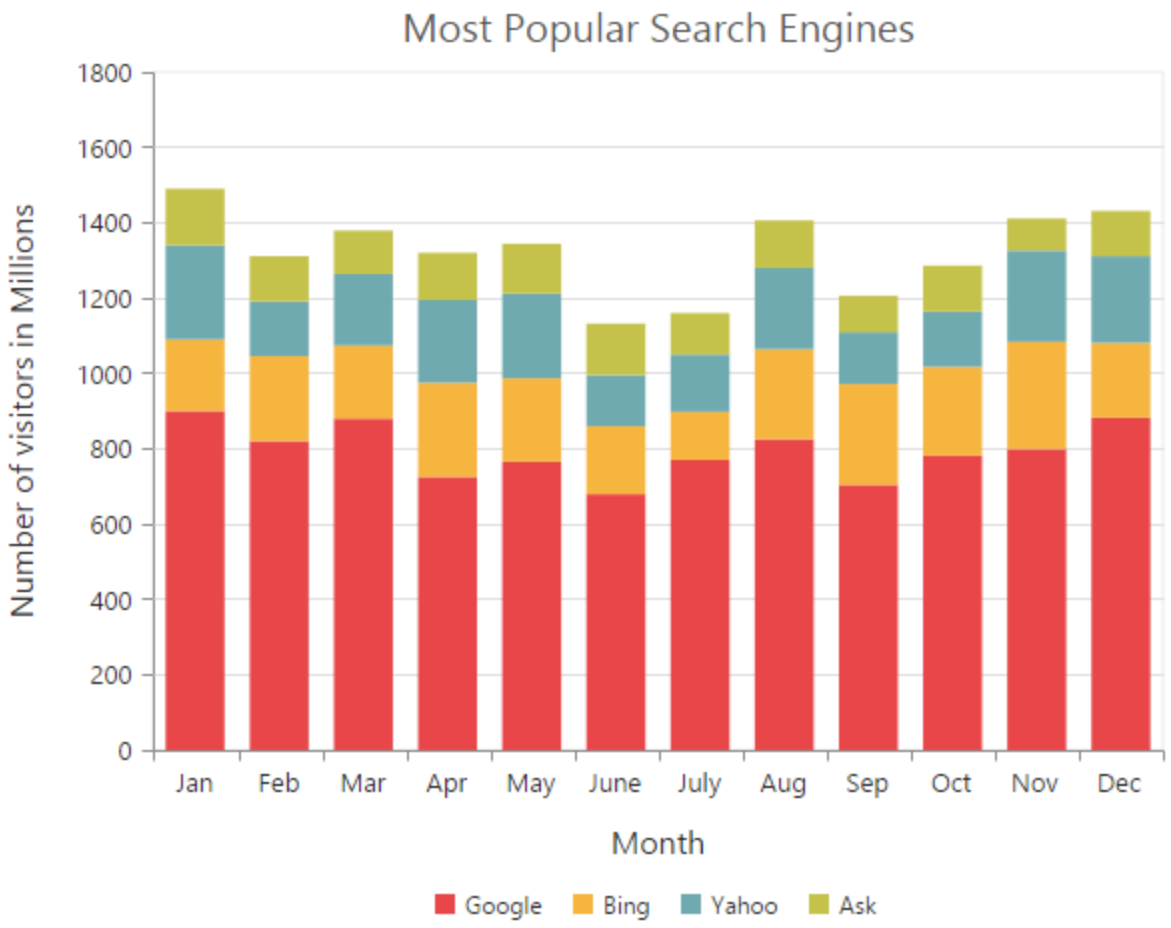


Stacked Column Chart

To render a Stacked Column Chart, set the Type as **StackingColumn** in the chart series. To change the StackingColumn color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.StackingColumn).Fill("#E94649")
//...
.Add();
})
//...
)
```



Click here to view Stacked Column Chart online demo.

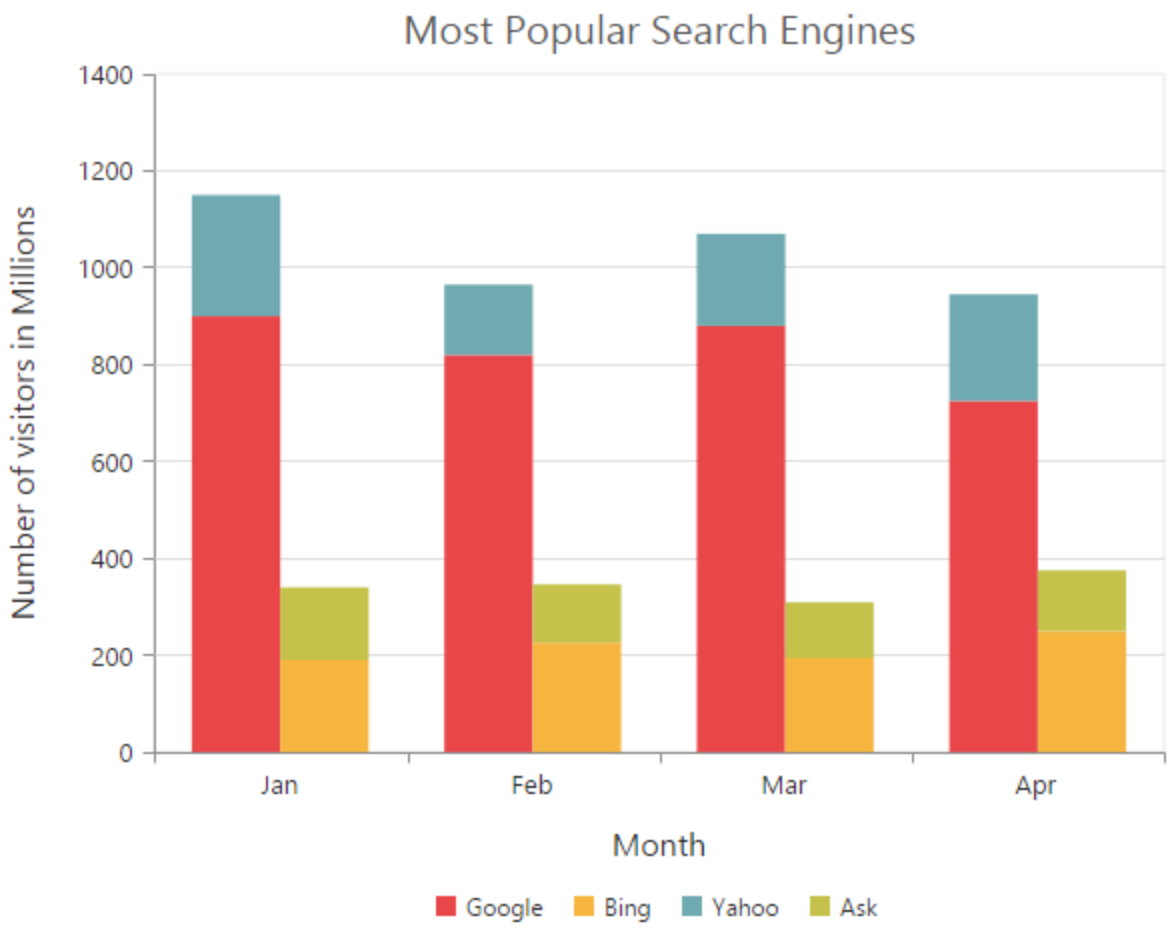
Cluster / Group stacked columns

You can use the **StackingGroup** property to group the stacked columns. Columns with same group name are stacked on top of each other.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
// For grouping stacked columns
sr.StackingGroup("GroupOne")
//...
.Add();
// For grouping stacked columns
sr.StackingGroup("GroupOne")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



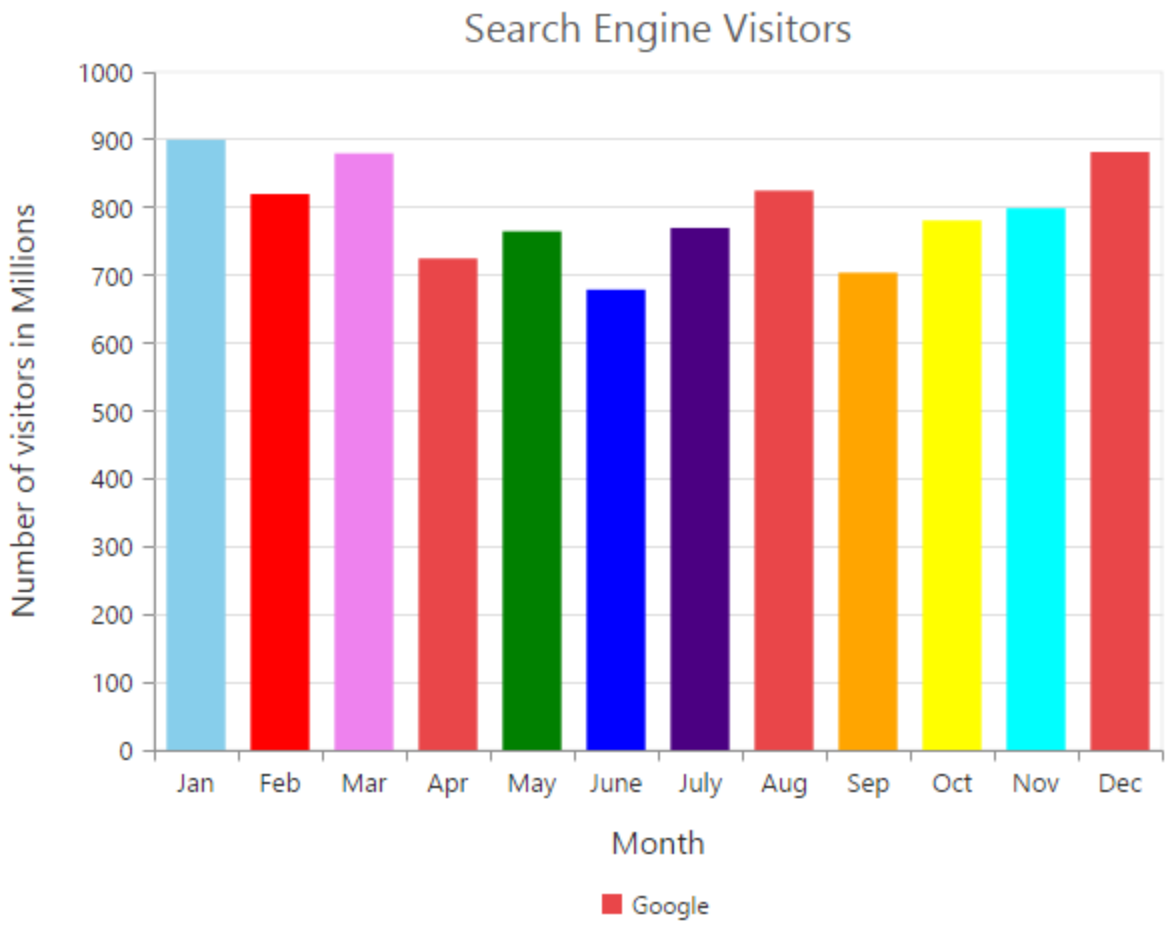
Change a point color

To change the color of a stacking column, you can use the Fill property of the point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//change a point color of Stacking column
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



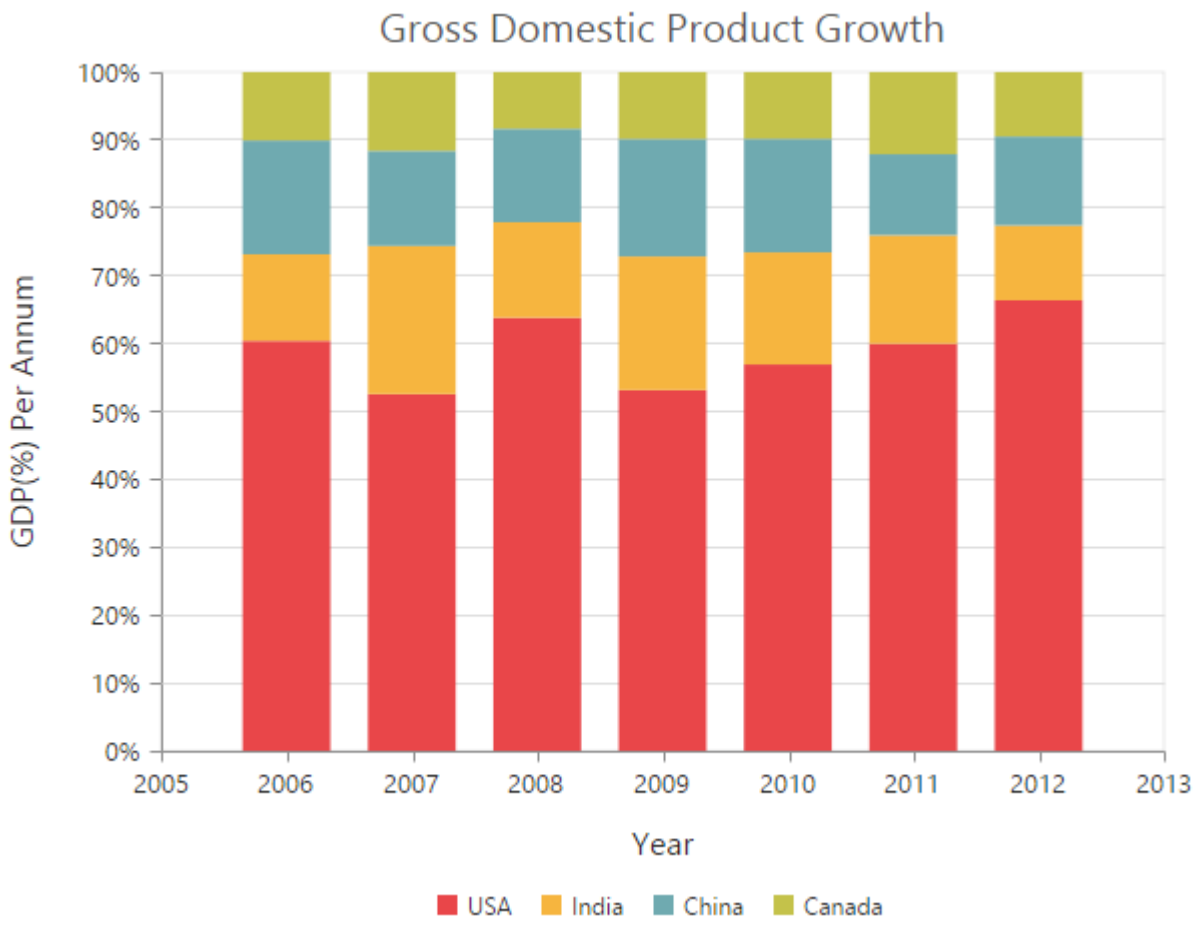
100% Stacked Column Chart

To render a 100% Stacked Column Chart, set the Type as **StackingColumn100** in the chart series. To change the StackingColumn100 color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series
sr.Type(SeriesType.StackingColumn100).Fill("#E94649")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view 100% Stacked Column Chart online demo.

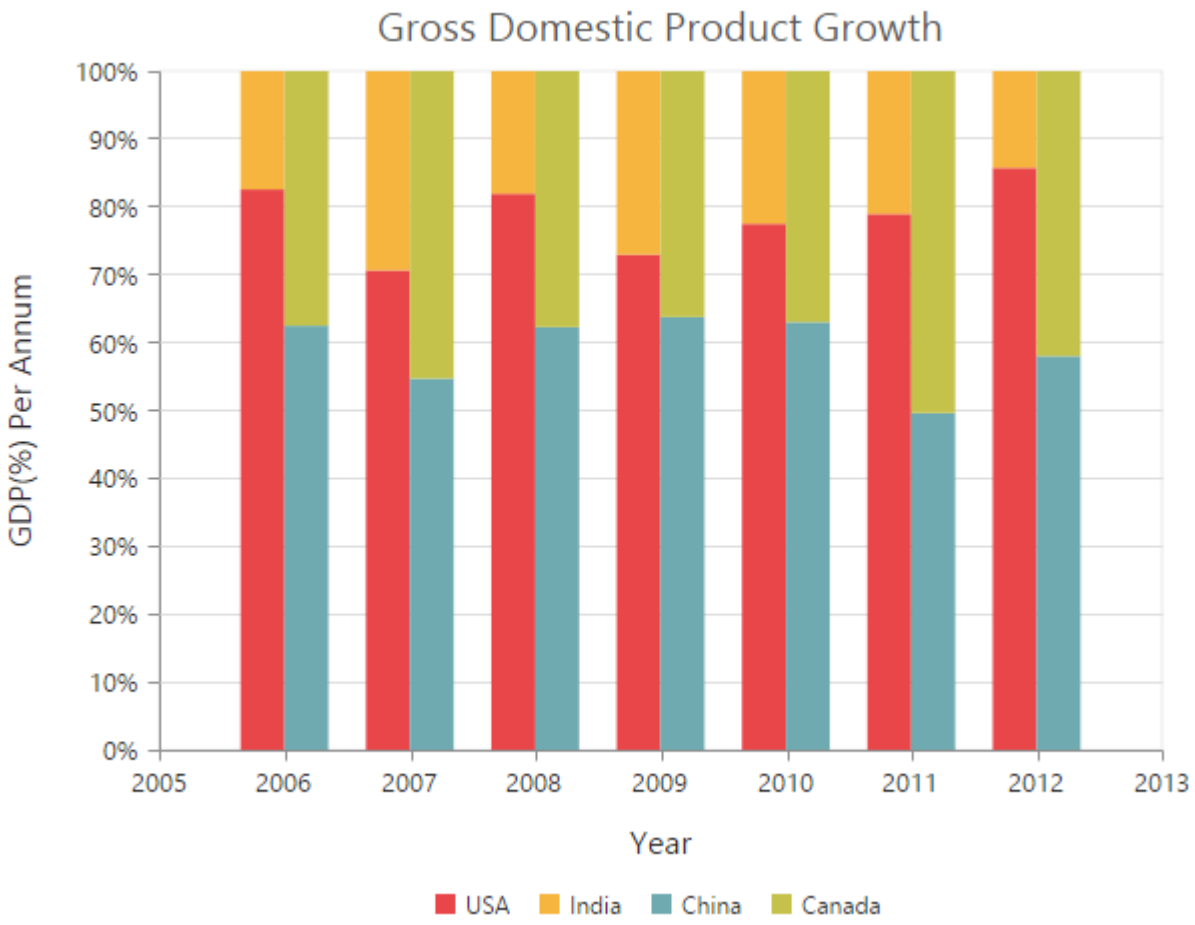
Cluster / Group 100% stacked columns

By using the **StackingGroup** property, you can group the 100% stacking columns. Columns with same group name are stacked on top of each other.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
// For grouping 100% stacked columns
sr.StackingGroup("GroupOne")
//...
.Add();
// For grouping 100% stacked columns
sr.StackingGroup("GroupOne")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

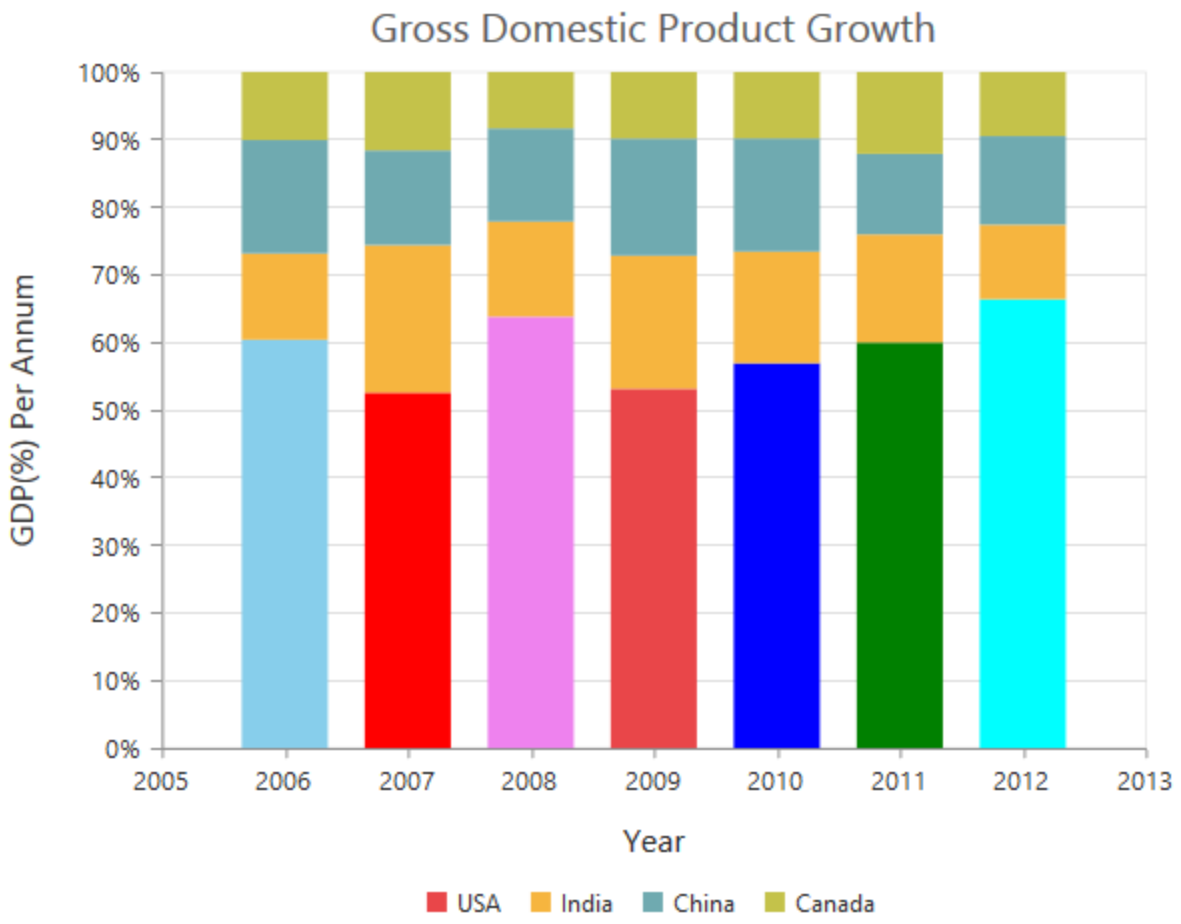


Change a point color

To change the color of a 100% stacking column, you can use the Fill property of the point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//change a point color of 100% Stacking column
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
)
```

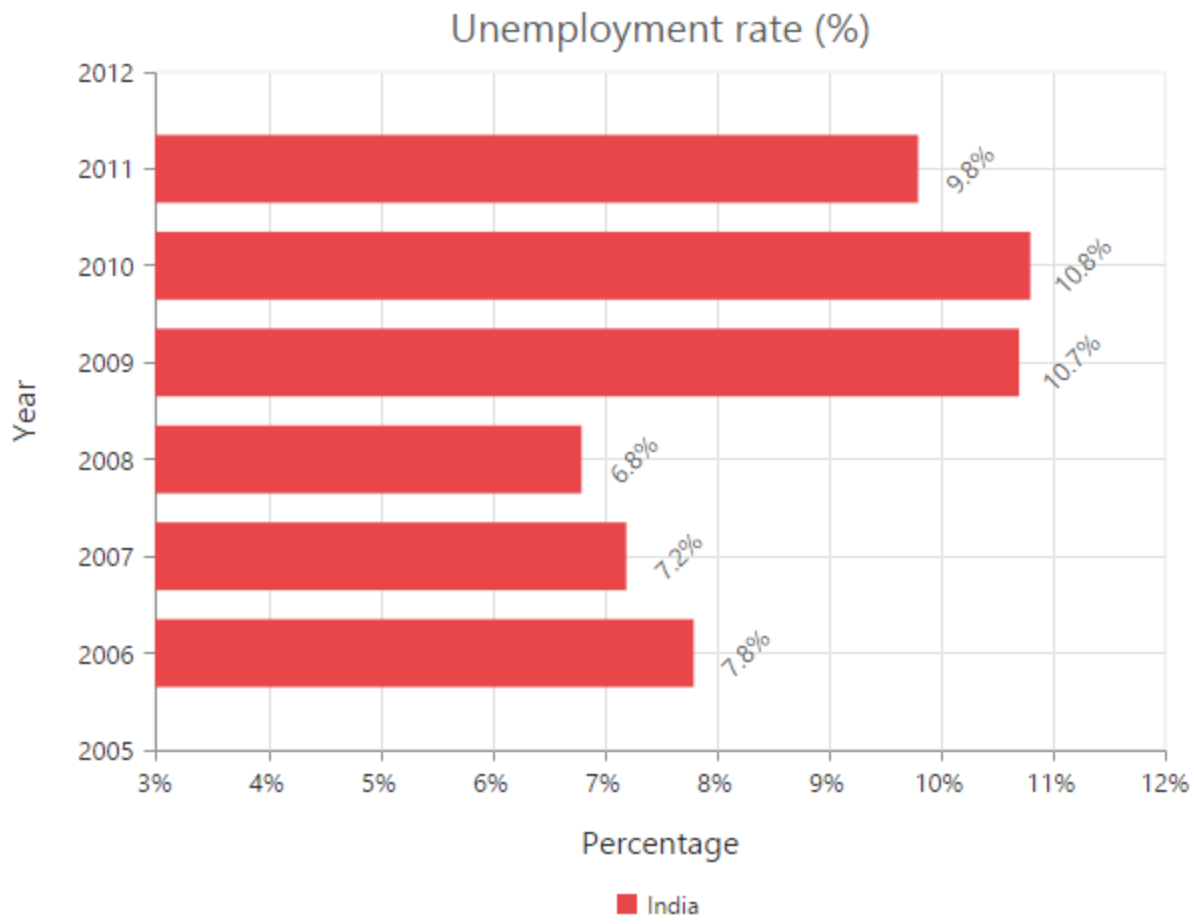


Bar Chart

To render a bar Chart, set the Type as **Bar** in the chart series. To change the bar color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series
sr.Type(SeriesType.Bar).Fill("#E94649")
//...
.Add();
})
//...
)
```



Click here to view Bar Chart demo.

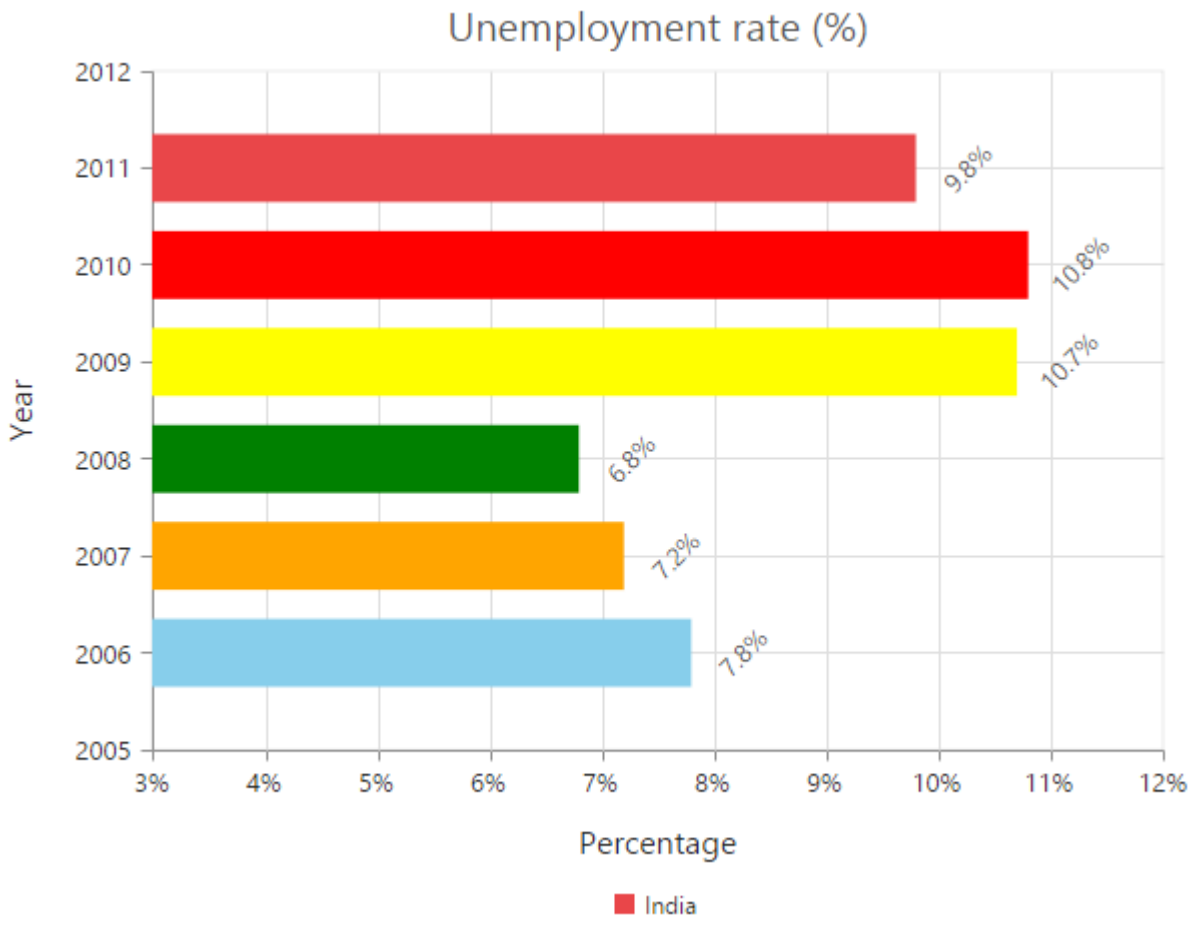
Change the color of a bar

By using the Fill property of the point, you can change the specific point of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//change a point color of bar
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

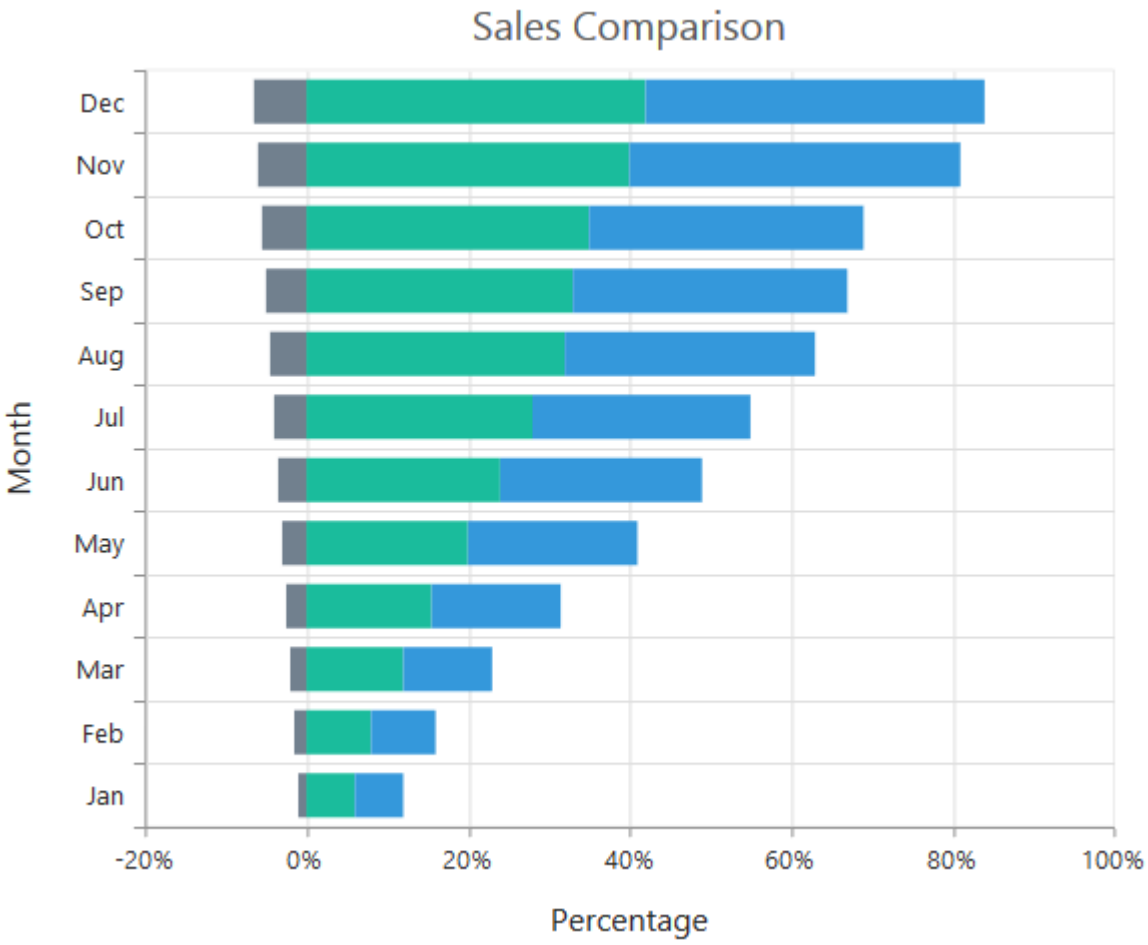


Stacked Bar Chart

To render a Stacked Bar Chart, set the Type as **StackingBar** in the chart series. To change the StackingBar color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series
sr.Type(SeriesType.StackingBar).Fill("#E94649")
//...
.Add();
})
//...
)
```



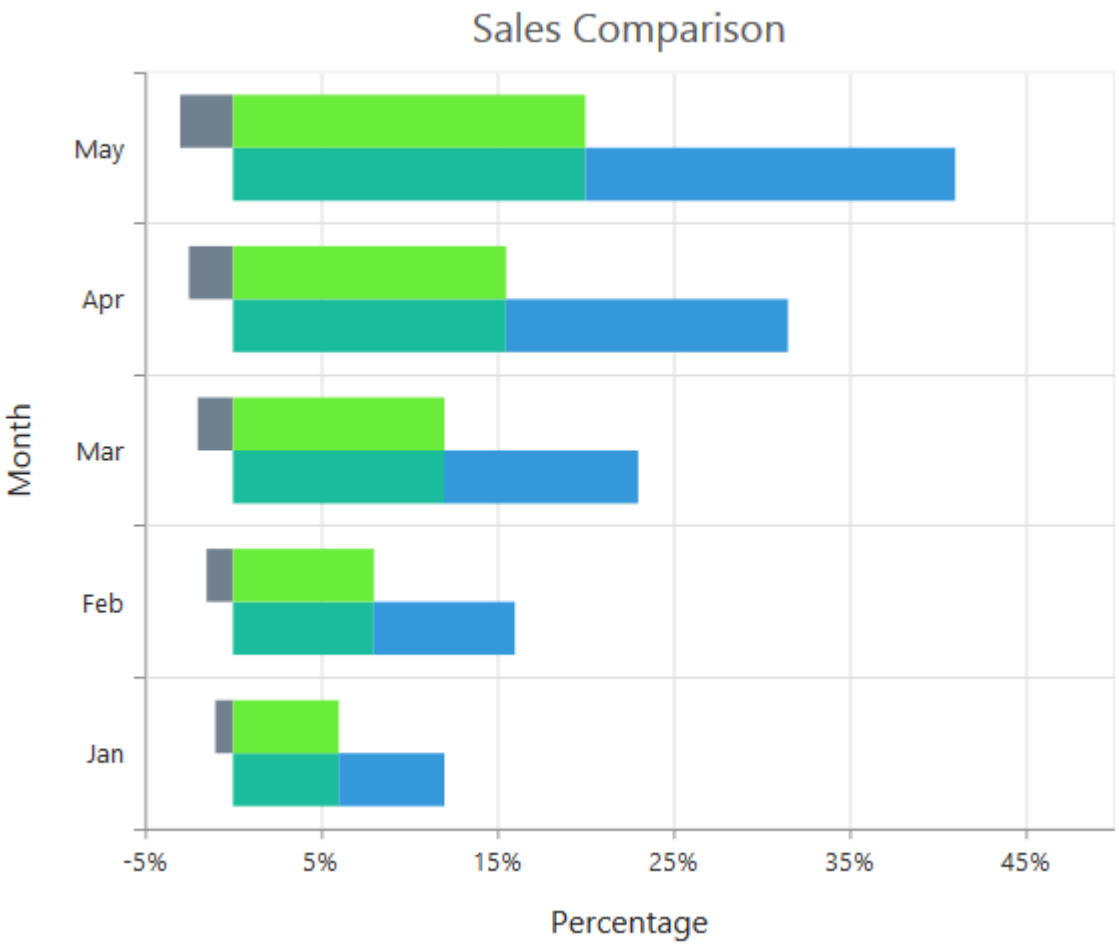
Click here to view Stacked Bar Chart online demo.

Cluster / Group stacked bars

You can use the **StackingGroup** property to group the stacking bars with the same group name.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
// For grouping stacking bar
sr.StackingGroup("GroupOne")
//...
.Add();
// For grouping stacking bar
sr.StackingGroup("GroupOne")
//...
.Add();
})
//...
)
```

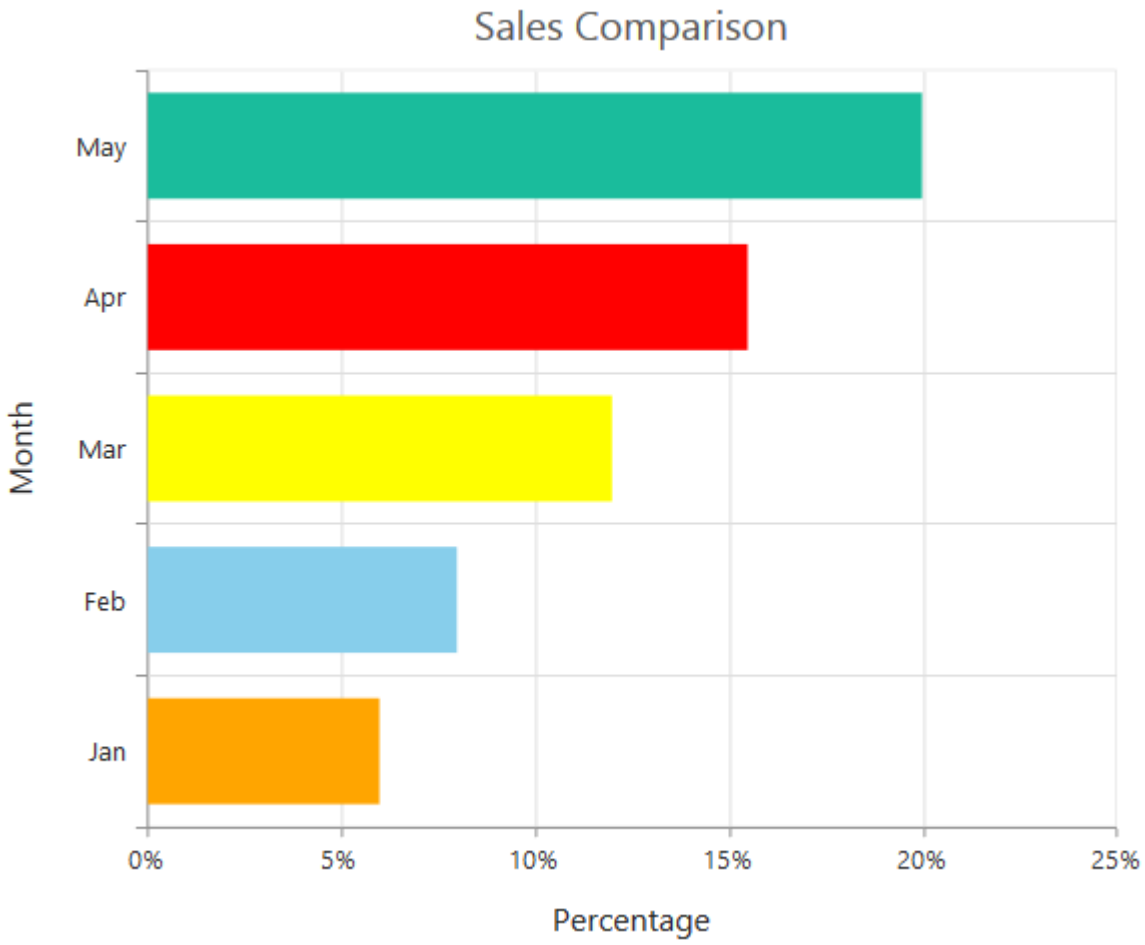


Change a point color

You can change the color of a stacking bar by using the Fill property of the point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//change a point color of stacking bar
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
)
```



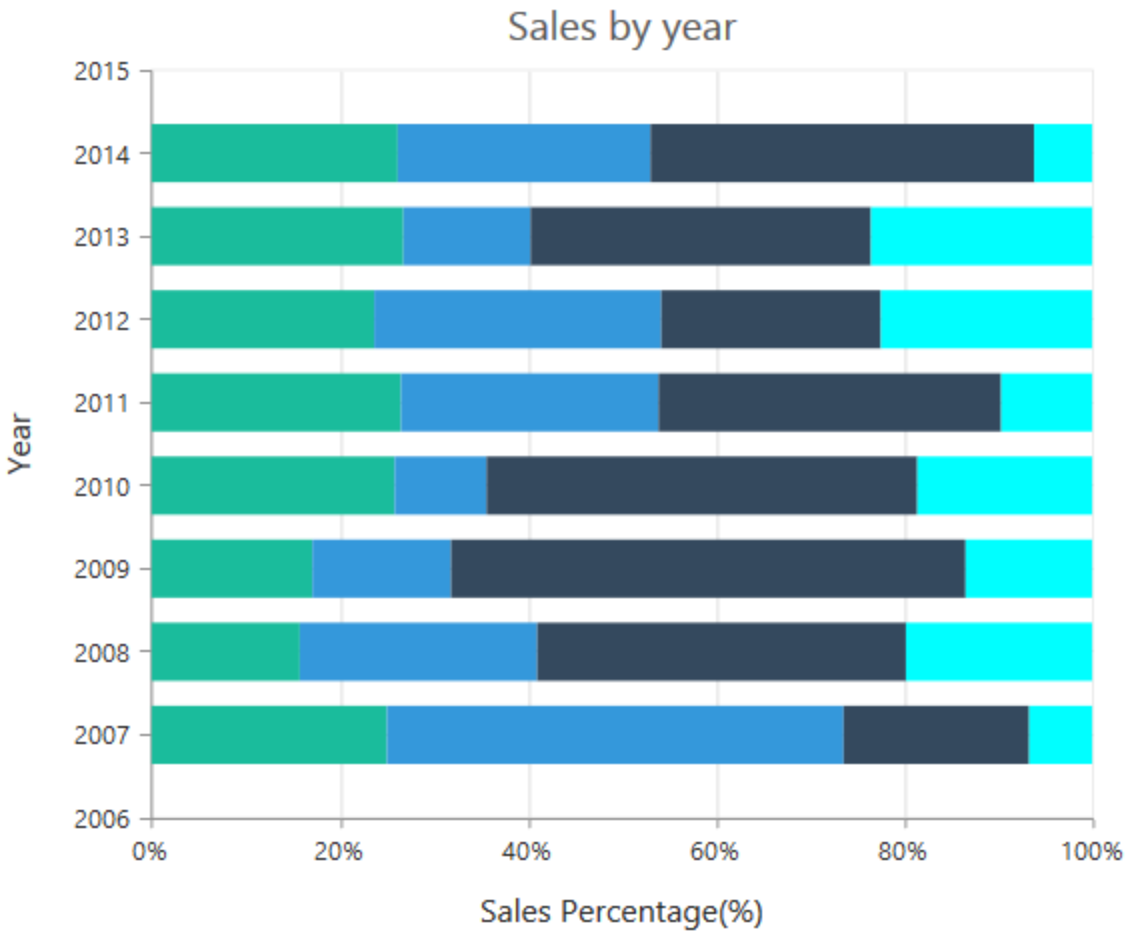
100% Stacked Bar Chart

To render a 100% Stacked Bar Chart, set the Type as **StackingBar100** in the chart series. To change the StackingBar100 color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series
sr.Type(SeriesType.StackingBar100).Fill("#E94649")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



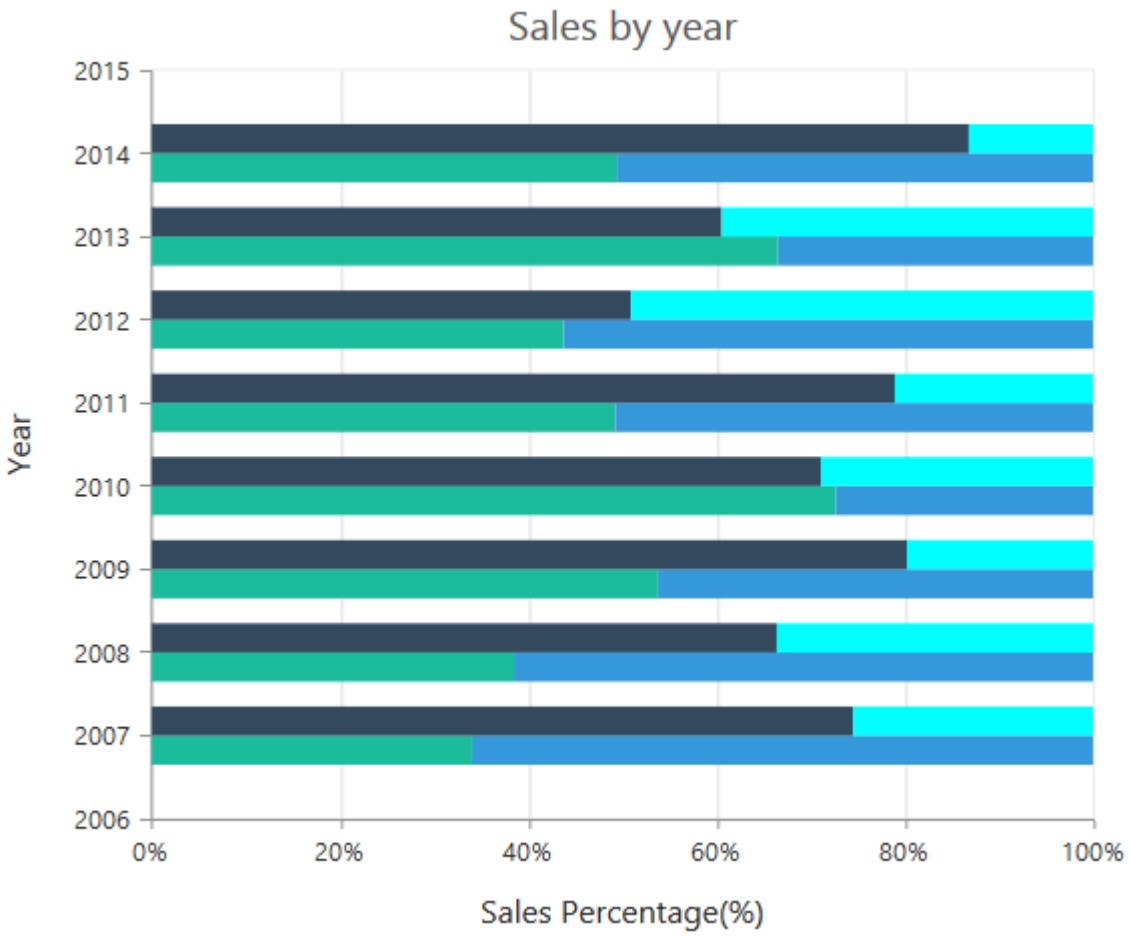
Click here to view 100% Stacked Bar Chart online demo.

By using the StackingGroup property, you can group the 100% stacking bars with the same group name.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
// For grouping 100% stacking bar
sr.StackingGroup("GroupOne")
//...
.Add();
// For grouping 100% stacking bar
sr.StackingGroup("GroupOne")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

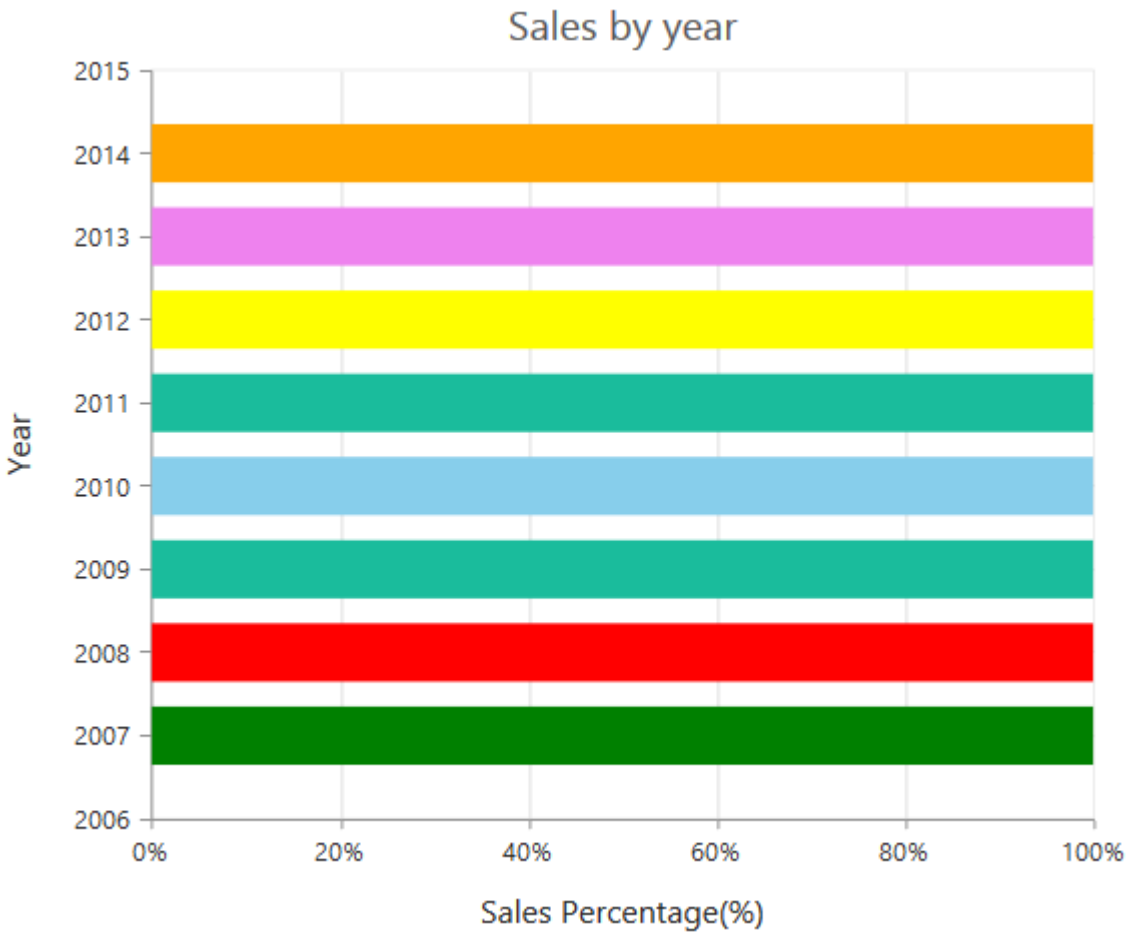


Change a point color

To change the color of a 100% stacking bar, you can use the Fill property of the point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//change a point color of 100% stacking bar
sr.Points(pt=>{pt.Fill("skyblue").Add();})
//...
.Add();
})
//...
)
```

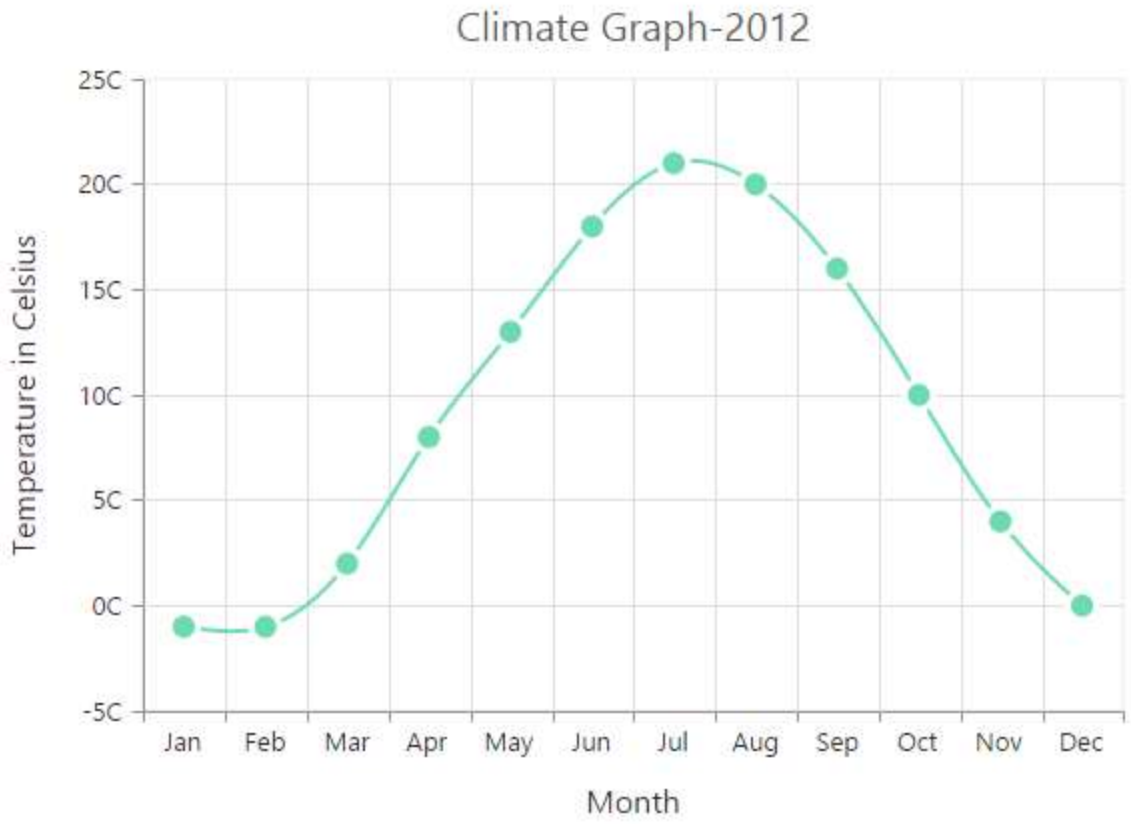


Spline Chart

To render a Spline Chart, set the Type as **Spline** in the chart series. To change the Spline segment color, you can use the Fill property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change type and color of the series
sr.Type(SeriesType.Spline).Fill("#6ADCB0")
//...
.Add();
})
//...
)
```



Click here to view the Spline Chart online demo sample.

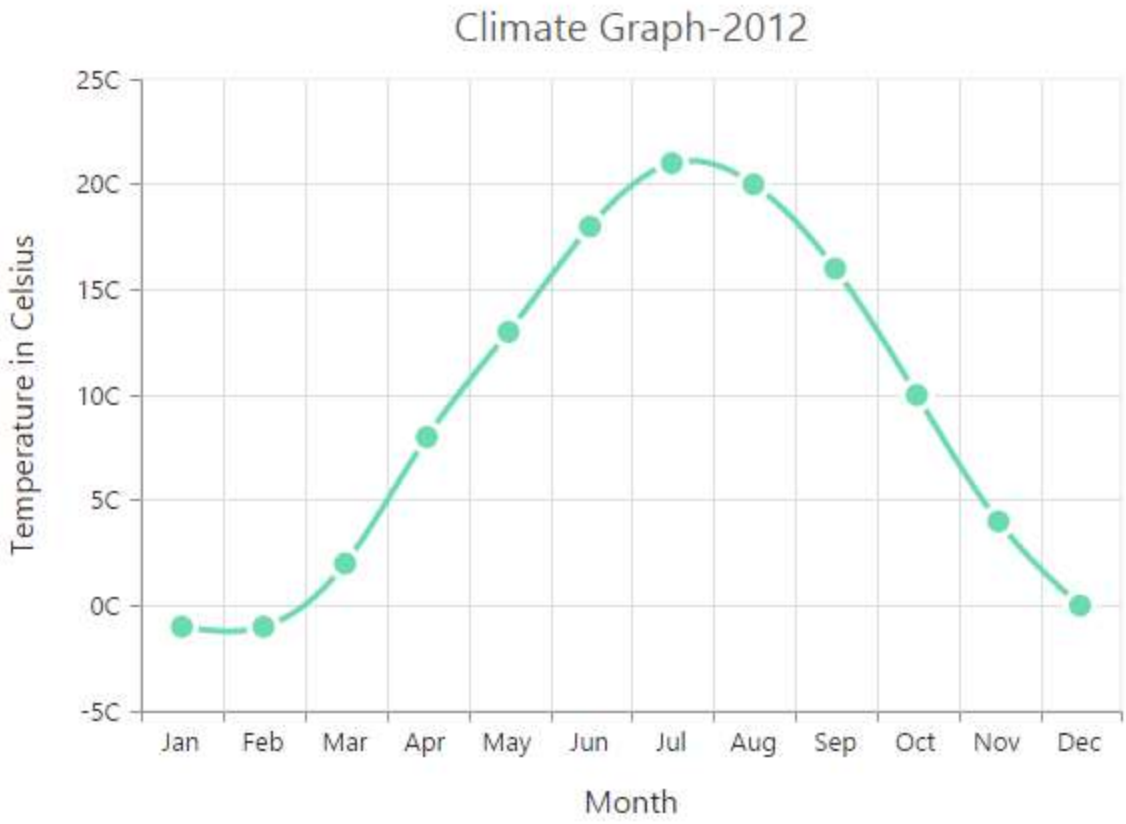
Change the spline width

To change the spline segment width, you can use the Width property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the width of spline series
sr.Width(3)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

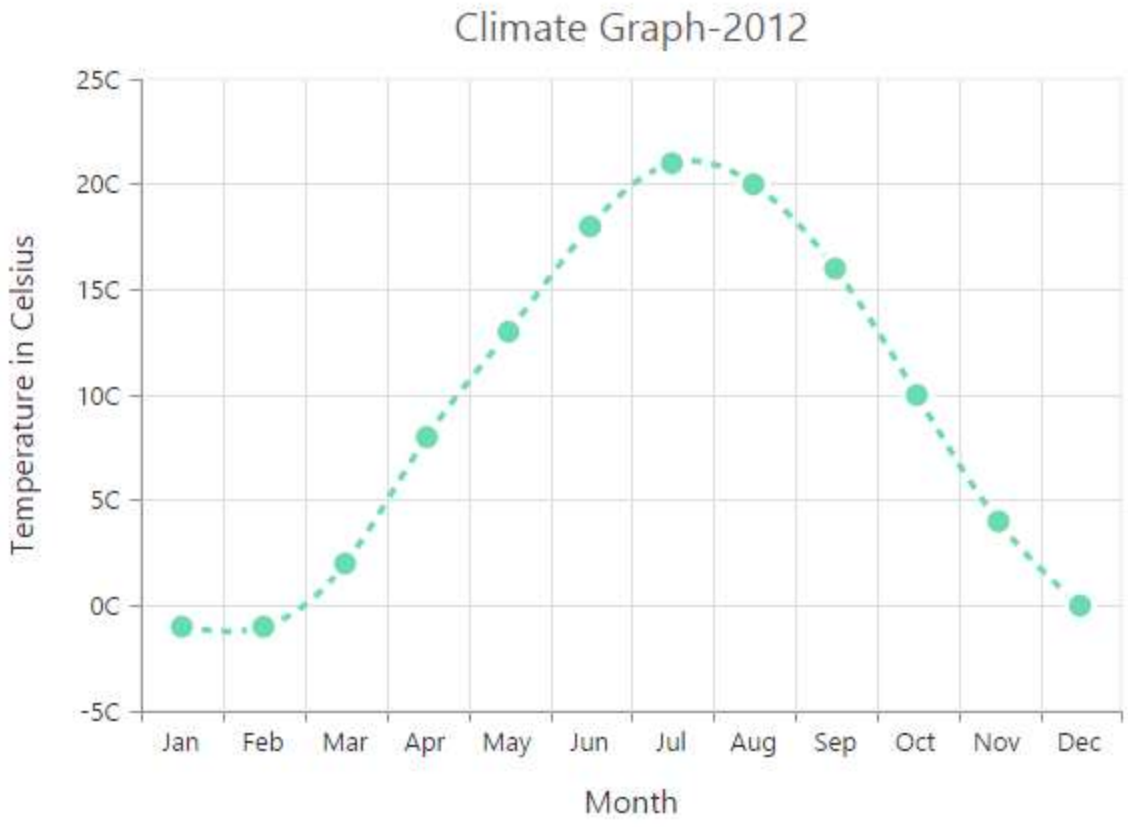


Dashed lines

To render the spline series with dotted lines, you can use the DashArray option of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change dash array to display dotted or dashed splines
sr.DashArray("5,5")
//...
.Add();
})
//...
)
```



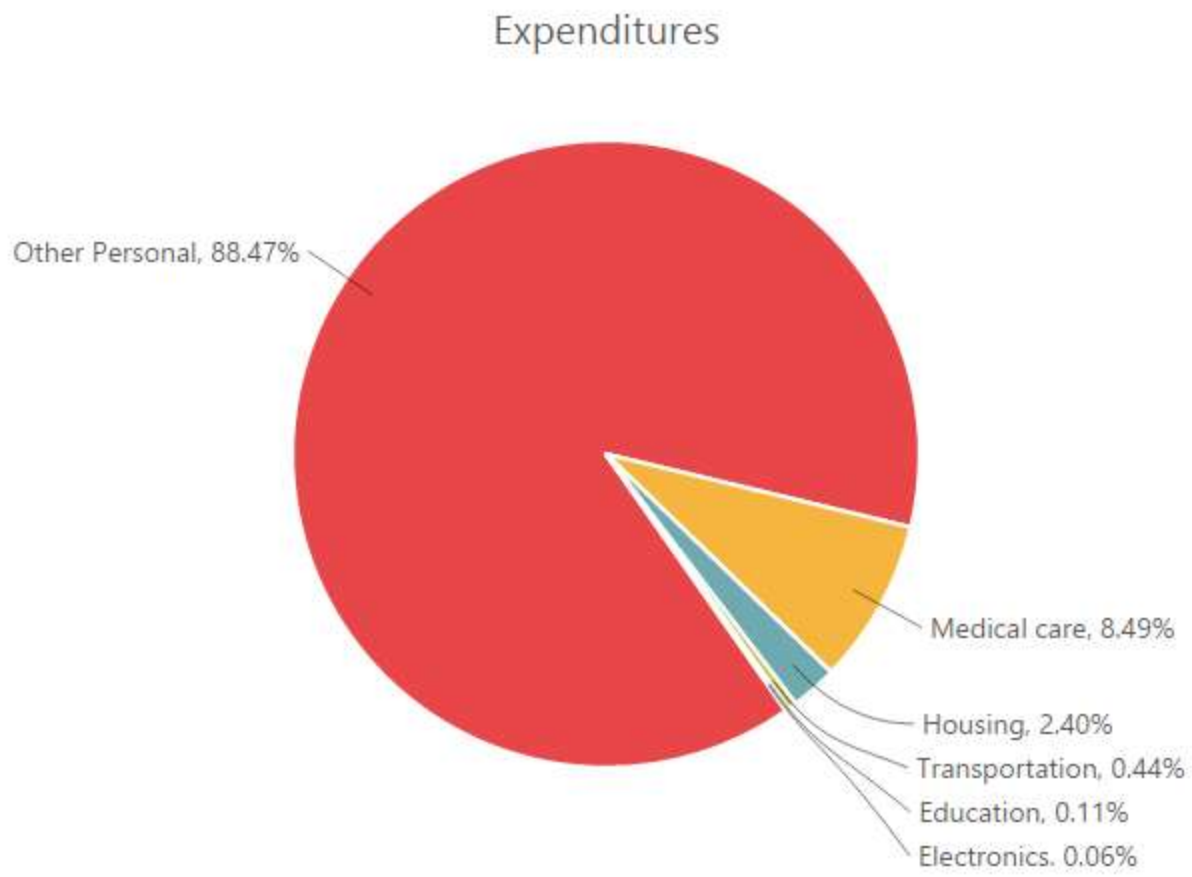
Pie Chart

You can create a pie chart by setting the series Type as **Pie** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Pie)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Pie chart online demo sample.

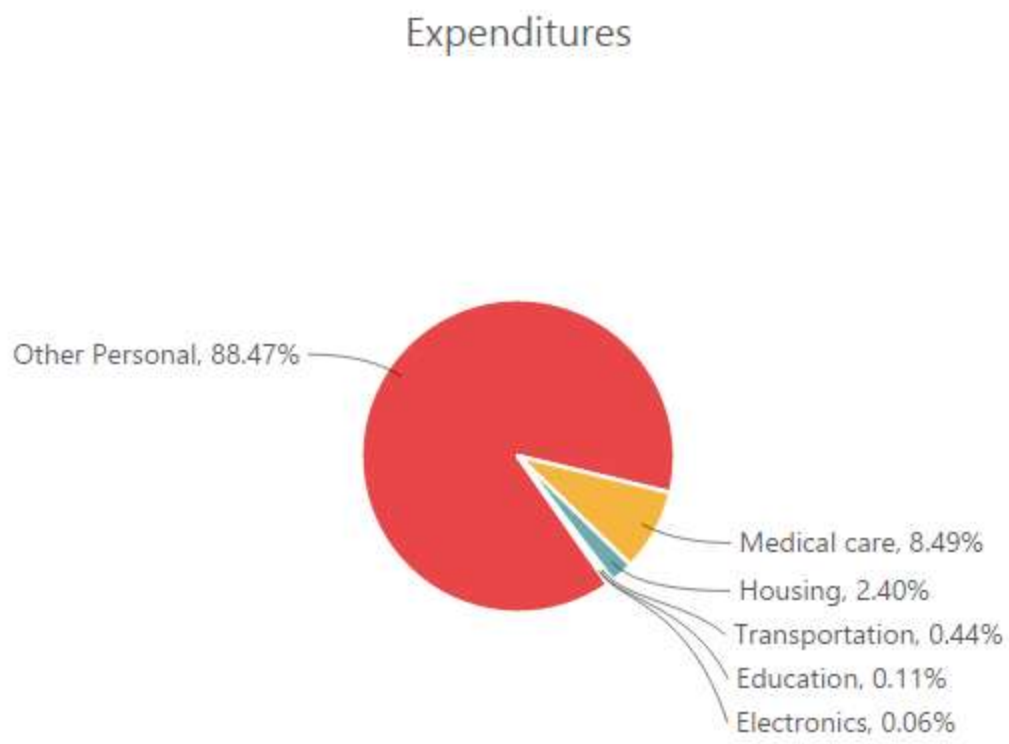
Change the pie size

You can use the **PieCoefficient** property to change the diameter of the Pie chart with respect to the plot area. It ranges from 0 to 1 and the default value is **0.8**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change pie chart coefficient value
sr.PieCoefficient(0.4f)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



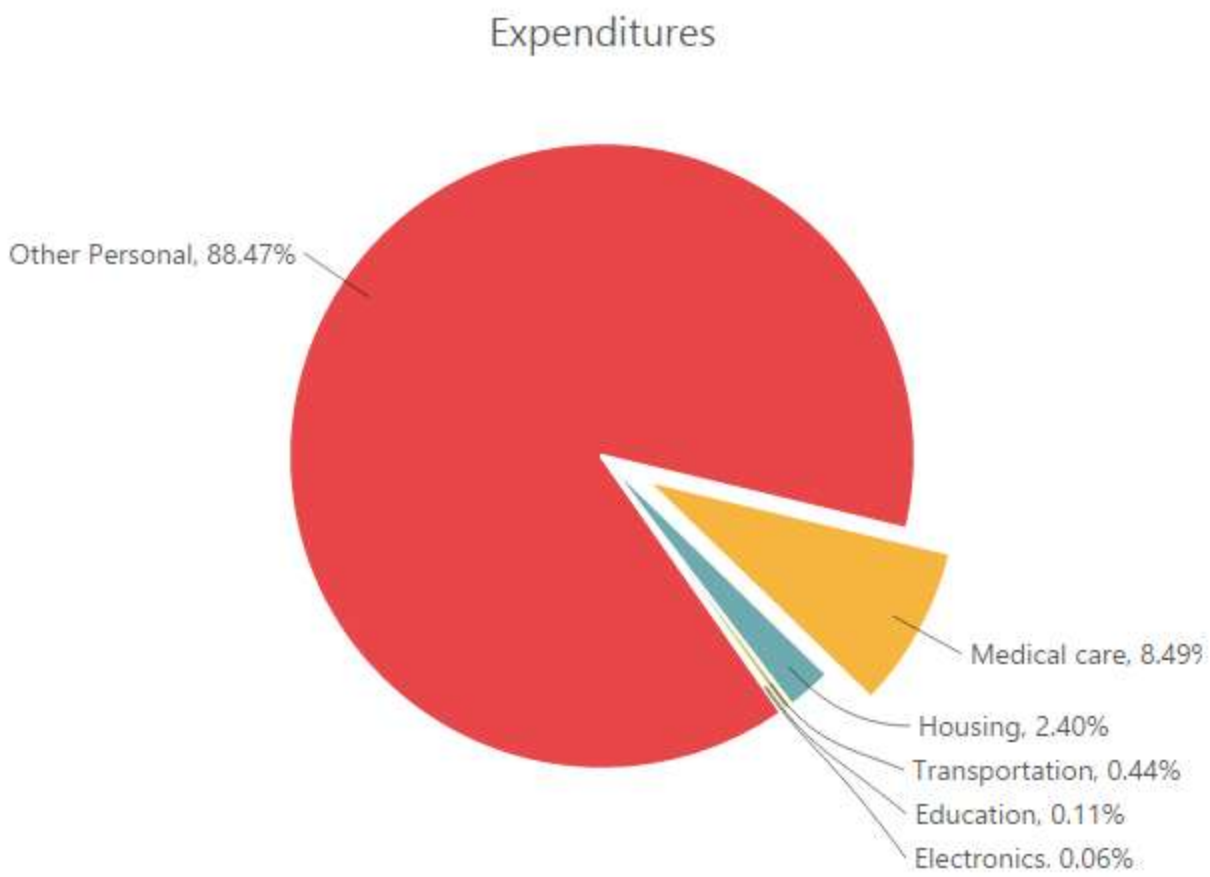
Explode a pie segment

You can explode a pie segment on the chart load by using the **ExplodeIndex** of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set point index value to explode the pie segment.
sr.ExplodeIndex(1)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



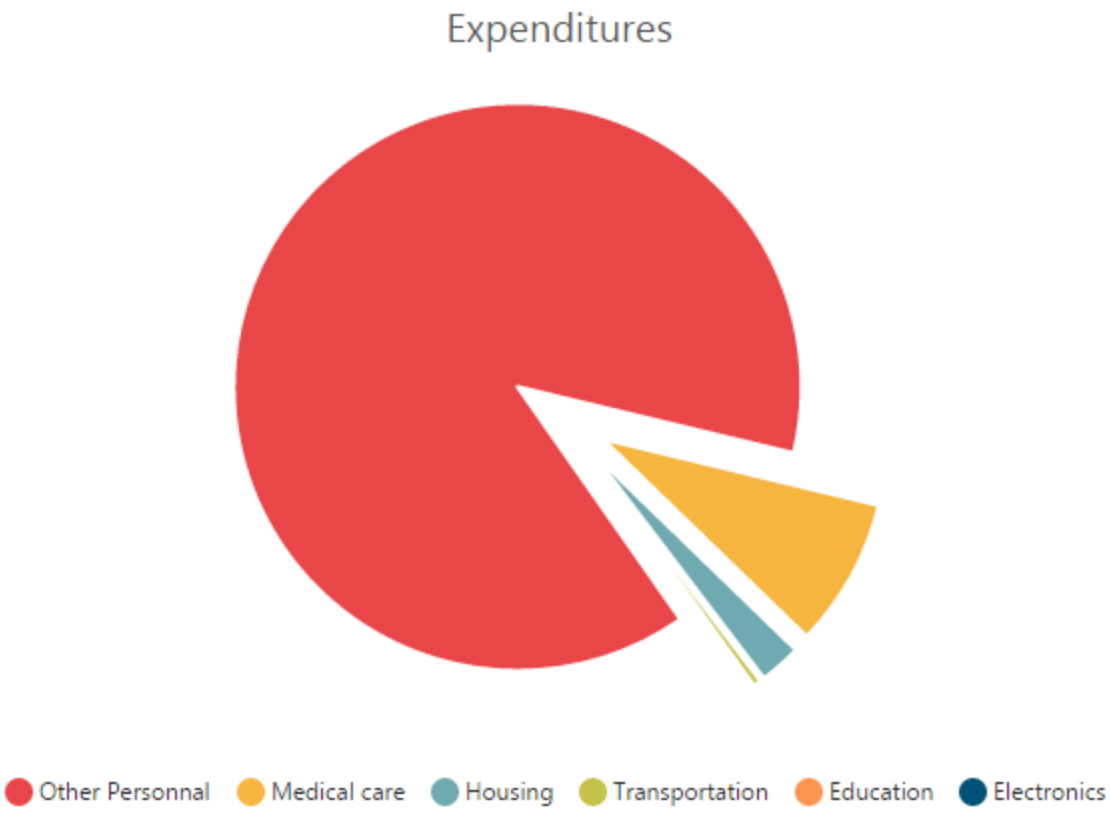
Explode all the segments

To explode all the segments of the Pie chart, you can enable the **ExplodeAll** property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Enable explodeAll property for pie chart.
sr.ExplodeAll(true)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



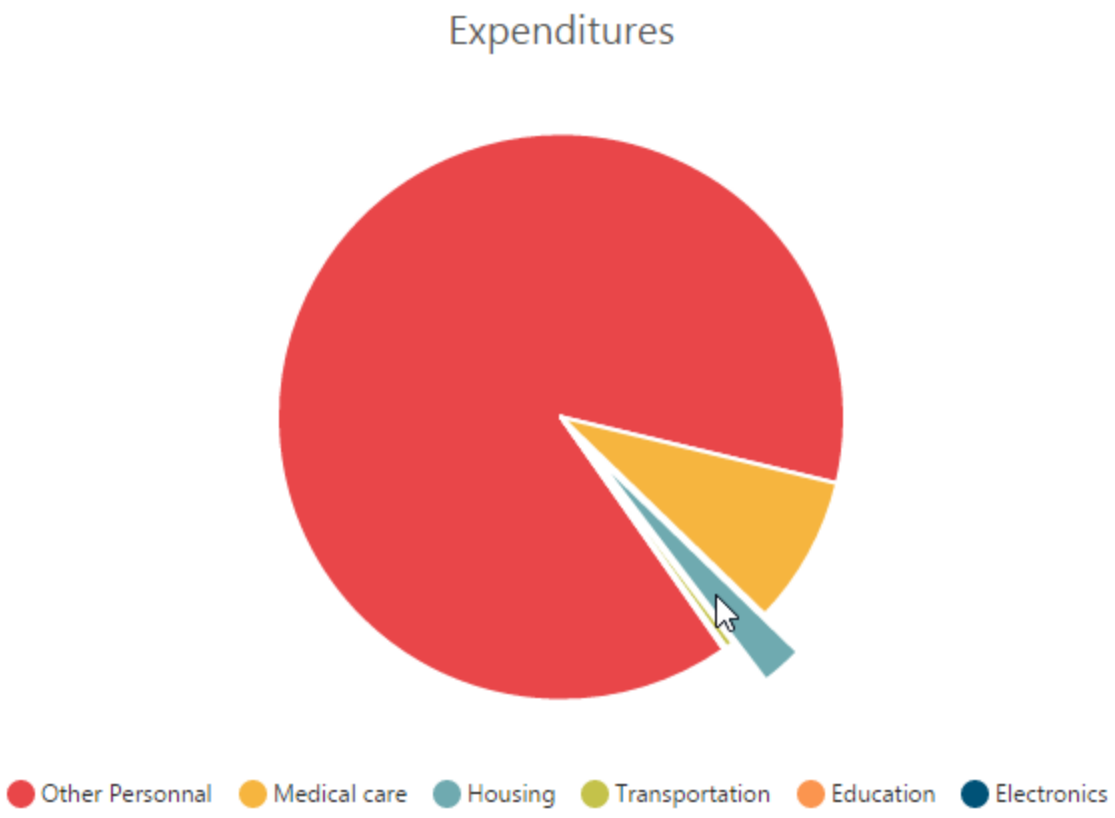
Explode a pie segment on mouse over

To explode a pie segment on a mouse over, you can enable the **Explode** property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Enable pie explode option on mouse over the chart
sr.Explode(true)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



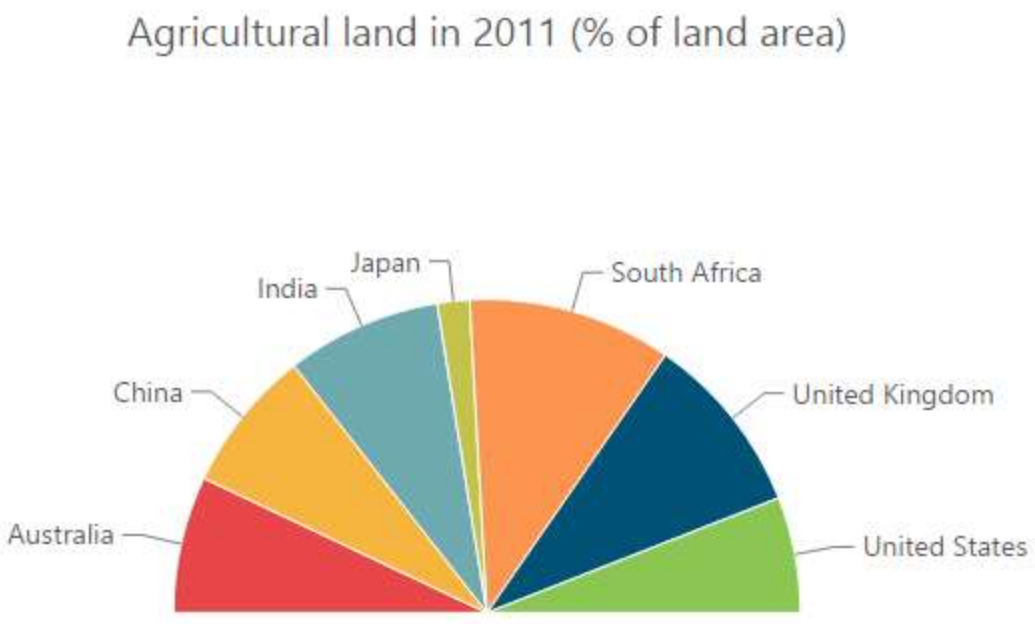
Sector of Pie

EjChart allows you to render all the data points/segments in the semi-pie, quarter-pie or in any sector by using the **StartAngle** and **EndAngle** options.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
sr.Type(SeriesType.Pie)
//Set startAngle and endAngle to draw the semi pie chart
.StartAngle(-90).EndAngle(90)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



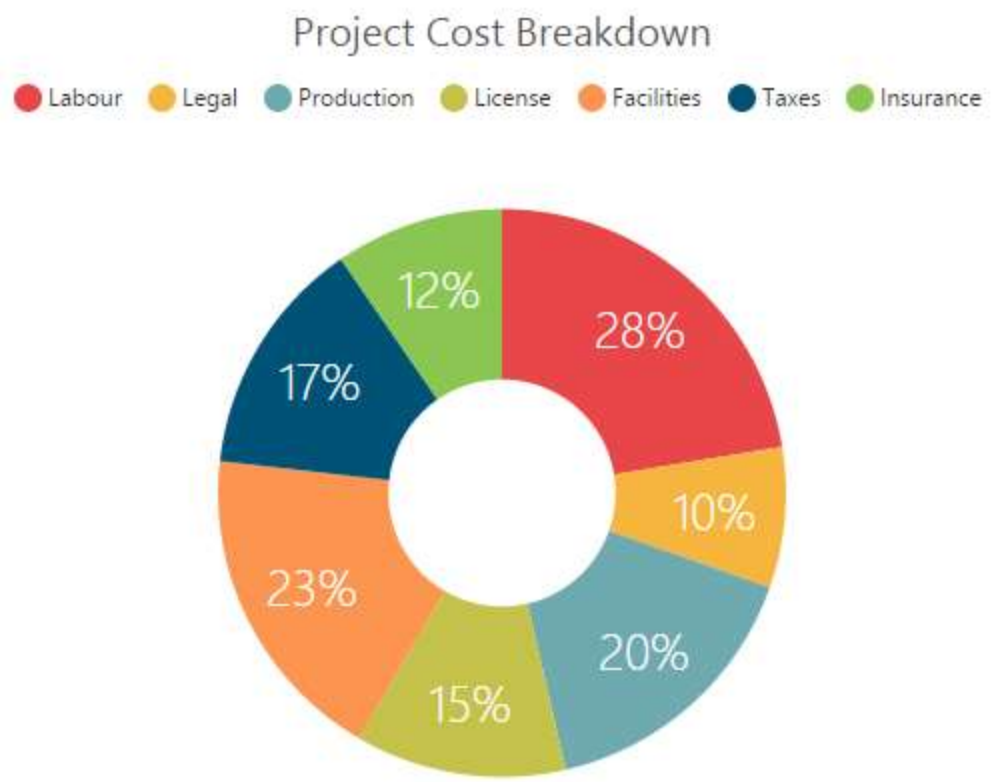
Click here to view the Semi Pie Chart online demo sample.

Doughnut Chart

To create a Doughnut chart, you can specify the series Type as **Doughnut** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Doughnut)
//...
.Add();
})
//...
)
```



Click here to view the Doughnut Chart online demo sample.

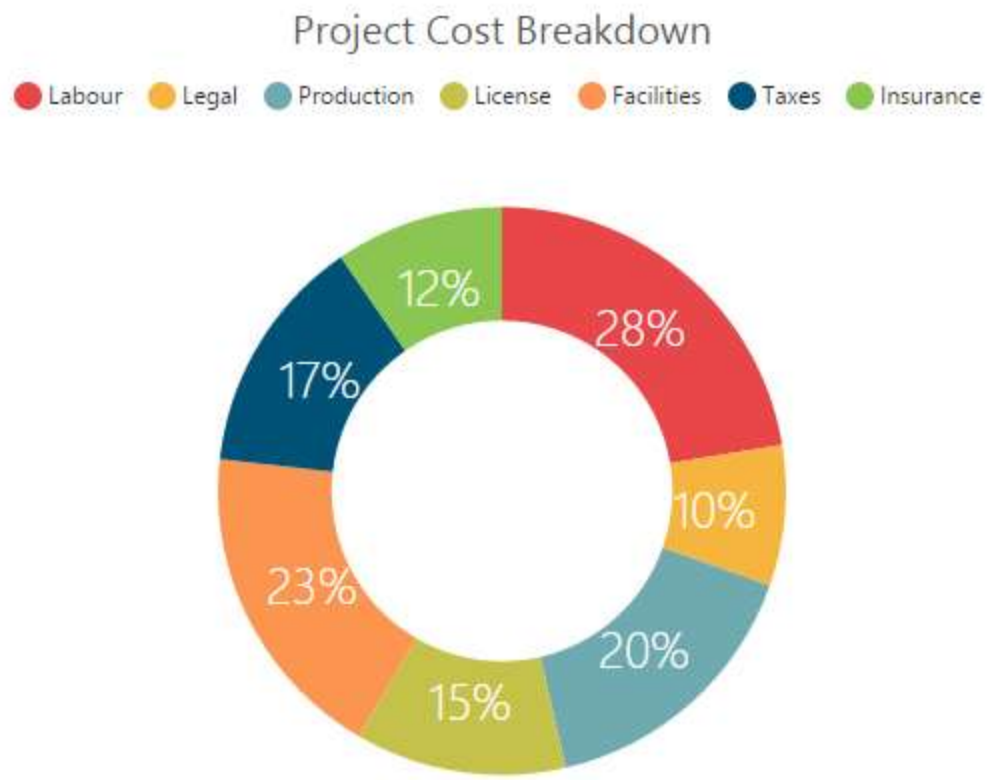
Change Doughnut inner radius

You can change the doughnut chart inner radius by using the **DoughnutCoefficient** with respect to the plot area. It ranges from 0 to 1 and the default value is **0.4**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change doughnut chart coefficient value
sr.DoughnutCoefficient(0.6f)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Change the doughnut size

You can use the **DoughnutSize** property to change the diameter of the Doughnut chart with respect to the plot area. It ranges from 0 to 1 and the default value is **0.8**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change doughnut chart coefficient value
sr.DoughnutSize(0.4f)
//...
.Add();
})
//...
)
```



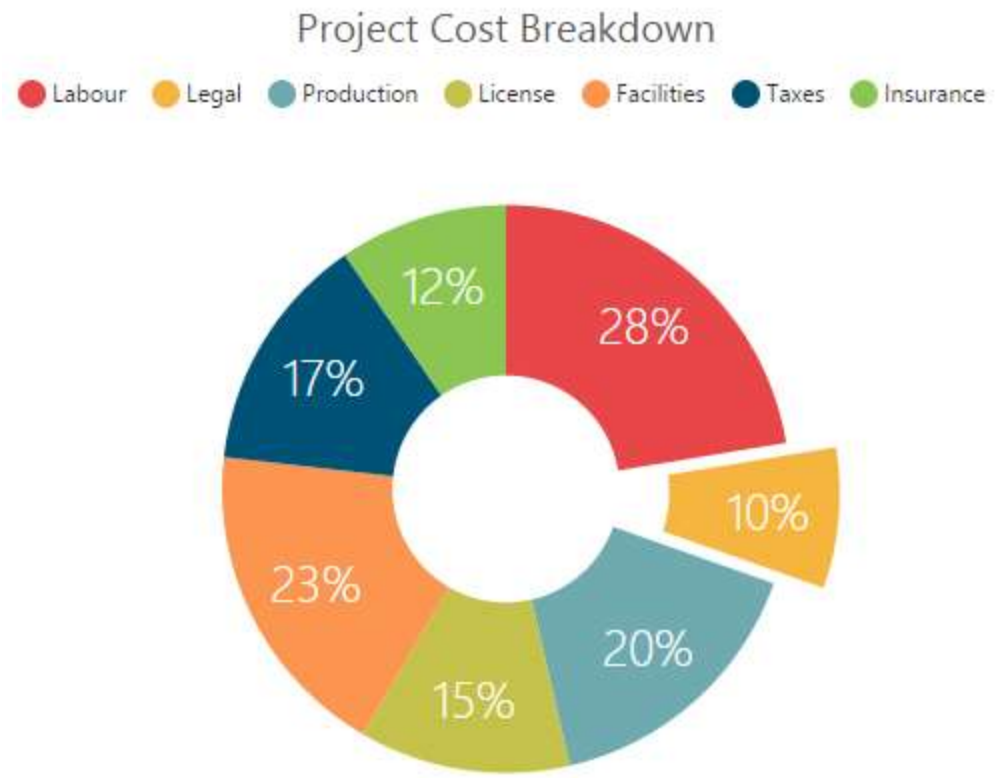
Explode a doughnut segment

To explode a specific doughnut segment, set the index to be exploded by using the **ExplodeIndex** option of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set point index value to explode the doughnut segment.
sr.ExplodeIndex(1)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



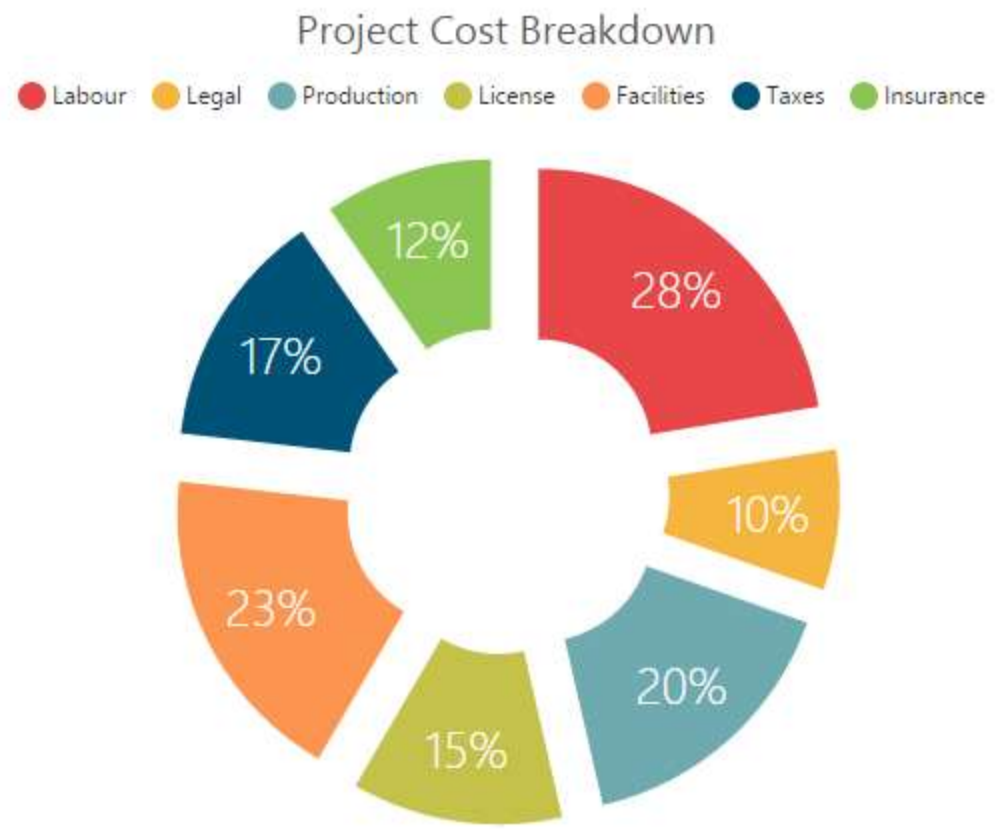
Explode all the segments

To explode all the segments, you can enable the **ExplodeAll** property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Enable explodeAll property of doughnut chart
sr.ExplodeAll(true)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



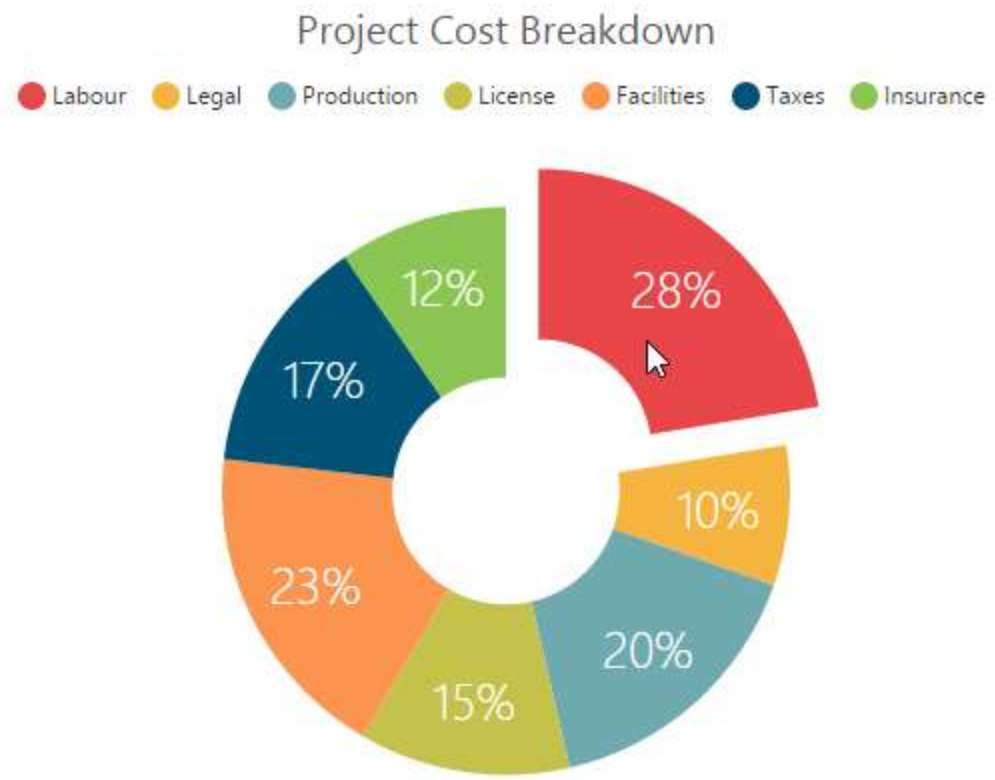
Explode a doughnut segment on mouse over

To explode a doughnut segment on a mouse over, you can enable the **Explode** property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Enable doughnut explode option on mouse over the chart
sr.Explode(true)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



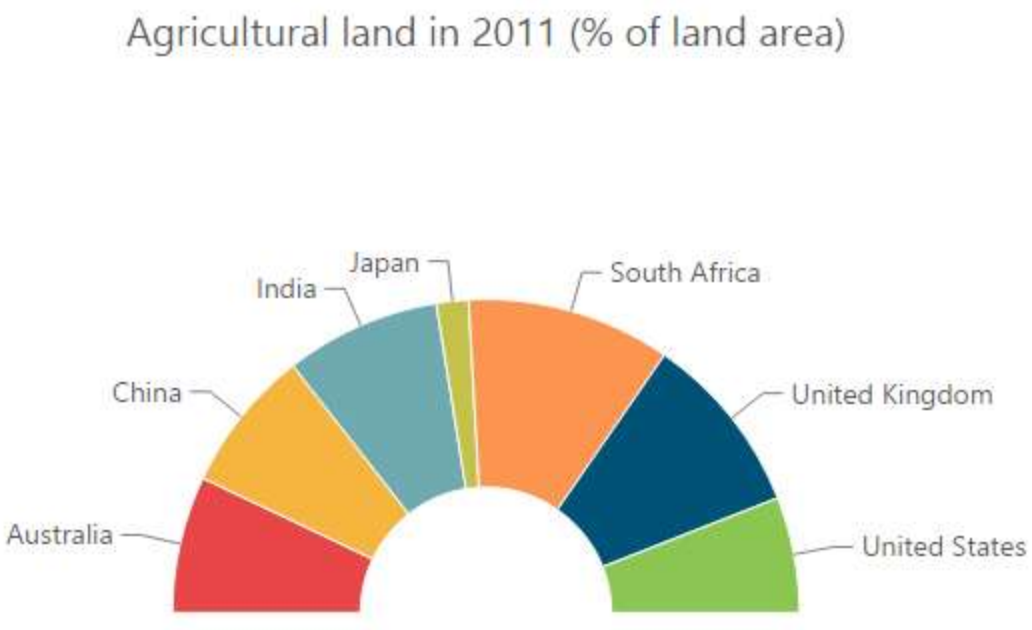
Sector of Doughnut

EjChart allows you to render all the data points/segments in the semi-doughnut, quarter- doughnut or in any sector by using the **StartAngle** and **EndAngle** options.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
sr.Type(SeriesType.Doughnut)
//Set startAngle and endAngle to draw the semi doughnut chart
.StartAngle(-90).EndAngle(90)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Semi Doughnut Chart online demo sample.

Multiple Pie Chart

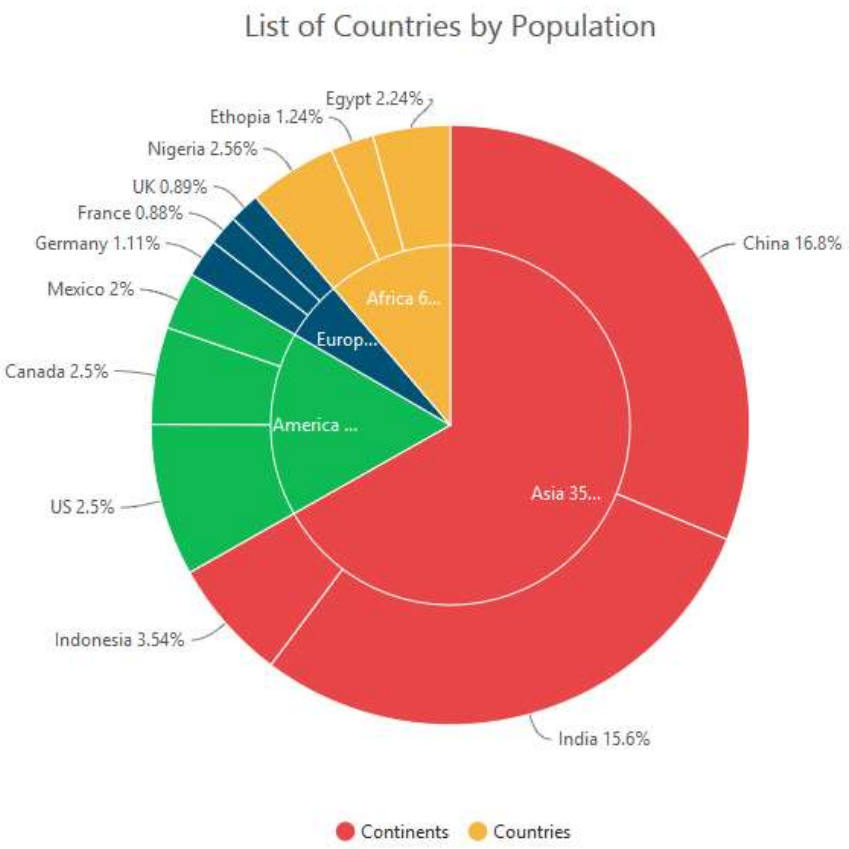
EjChart provides support to render more than one series in pie and in doughnut chart. Radius of each series is calculated based on the radius of the previous series. And in addition legend is displayed according to the list of chart series.

ASPX-CS

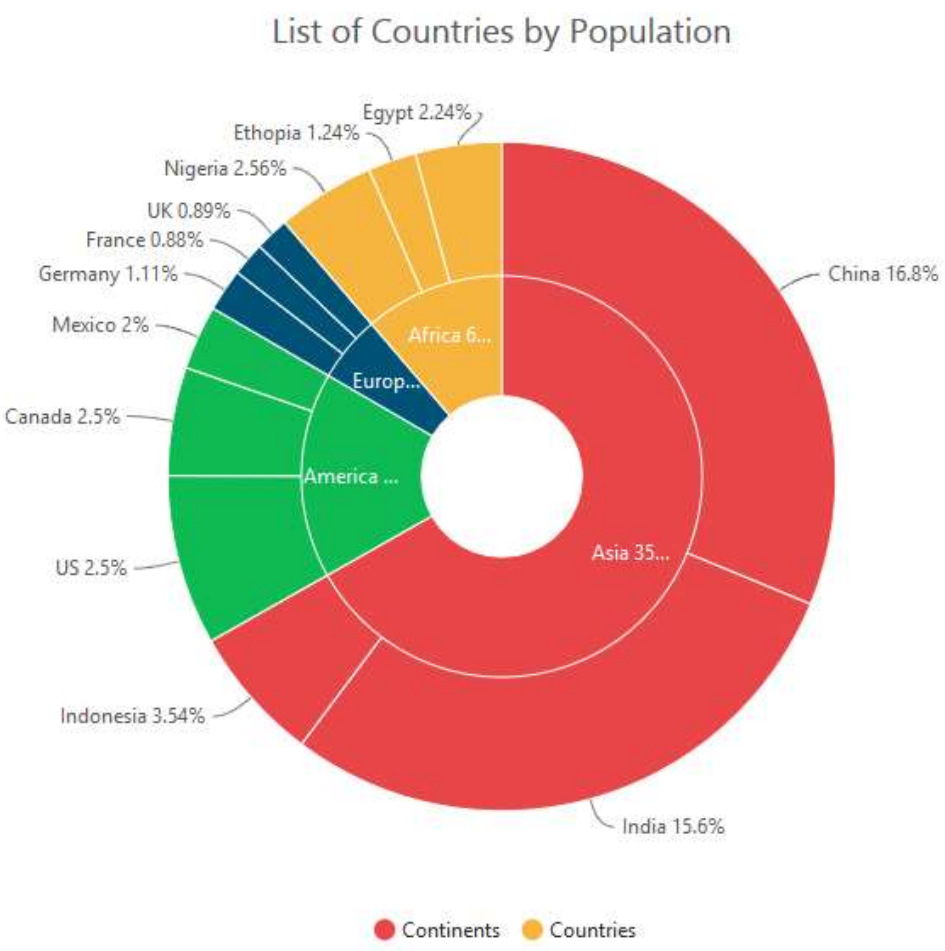
```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Adding multiple pie series
sr.Type(SeriesType.Pie)
//...
.Add();
//Adding multiple pie series
sr.Type(SeriesType.Pie).Add();
})
//...
)
```

Multiple Pie

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Doughnut



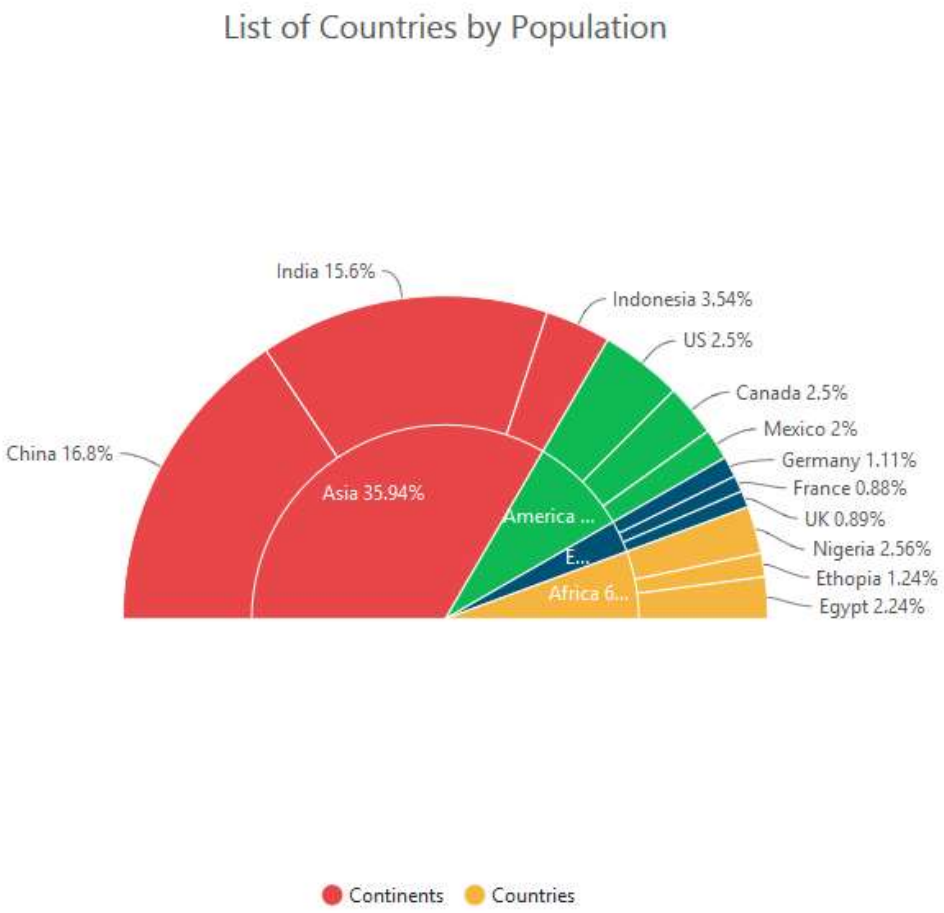
Click here to view the Multiple Pie chart online demo sample.

Start and End Angle Support

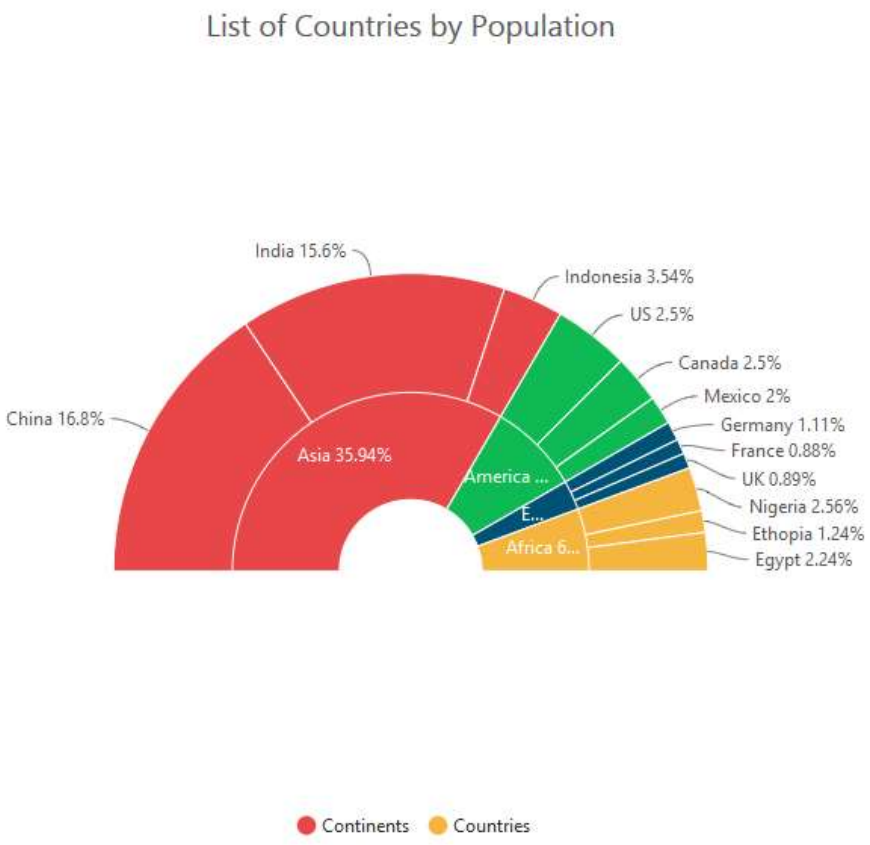
In the Multiple Pie chart, the start and end angle property is also supported.

Sector of Multiple Pie

Copyright © 2001 - 2020 Syncfusion Inc.



Sector of Multiple Doughnut



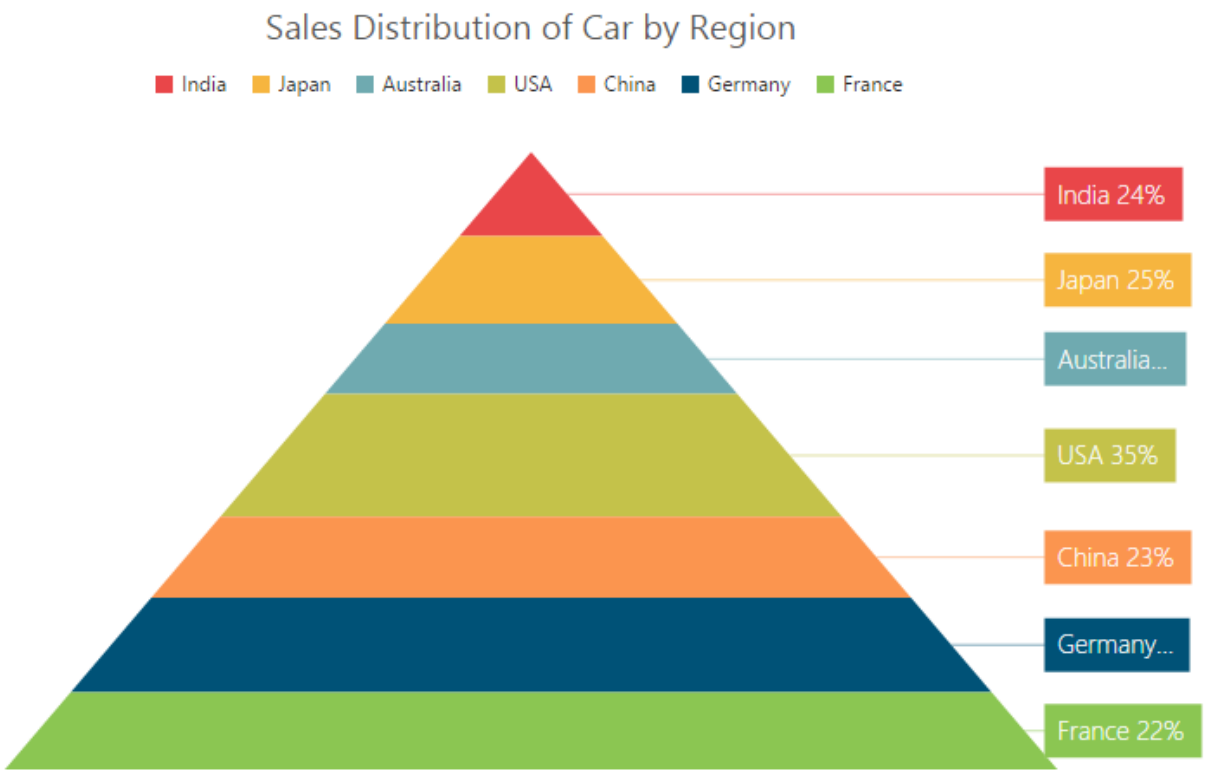
Pyramid Chart

To create a Pyramid chart, you can specify the series Type as **Pyramid** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Pyramid)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Pyramid Chart online demo sample.

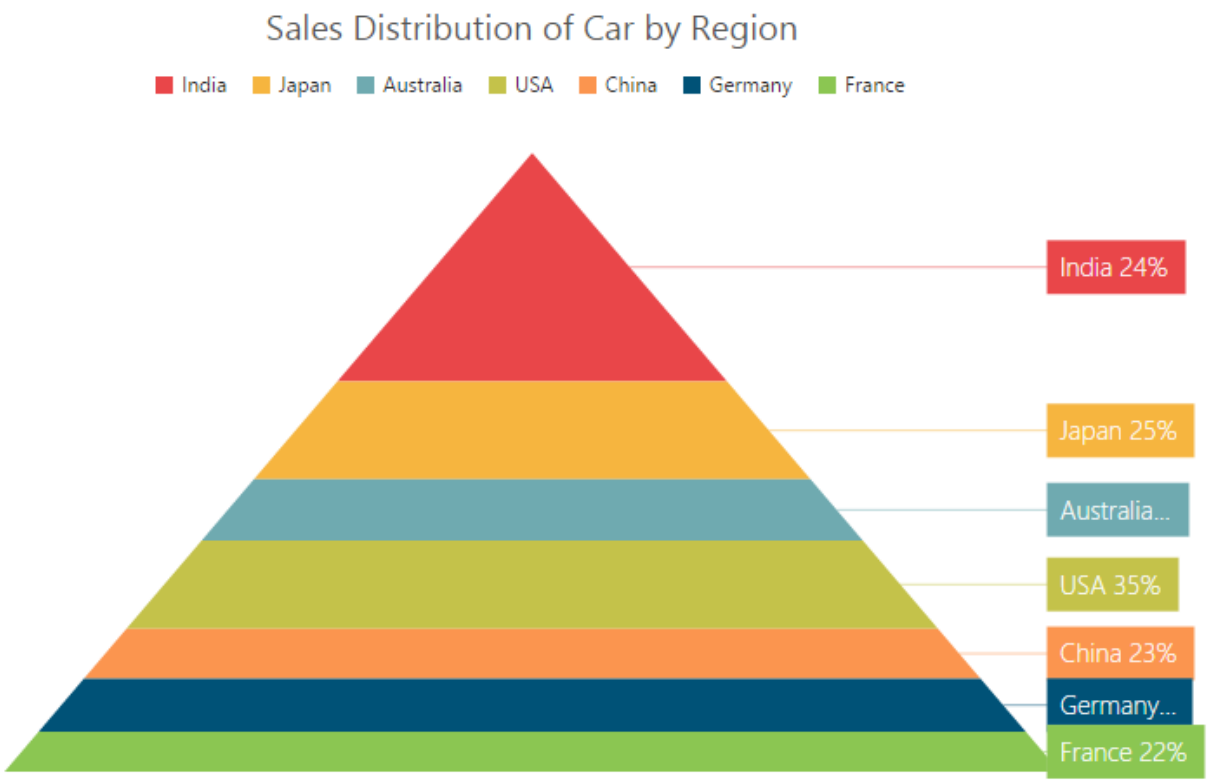
Pyramid Mode

Pyramid mode has two types, *Linear* and *Surface* respectively. The default **PyramidMode** type is "Linear".

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change pyramid mode
sr.PyramidMode(PyramidMode.Surface)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



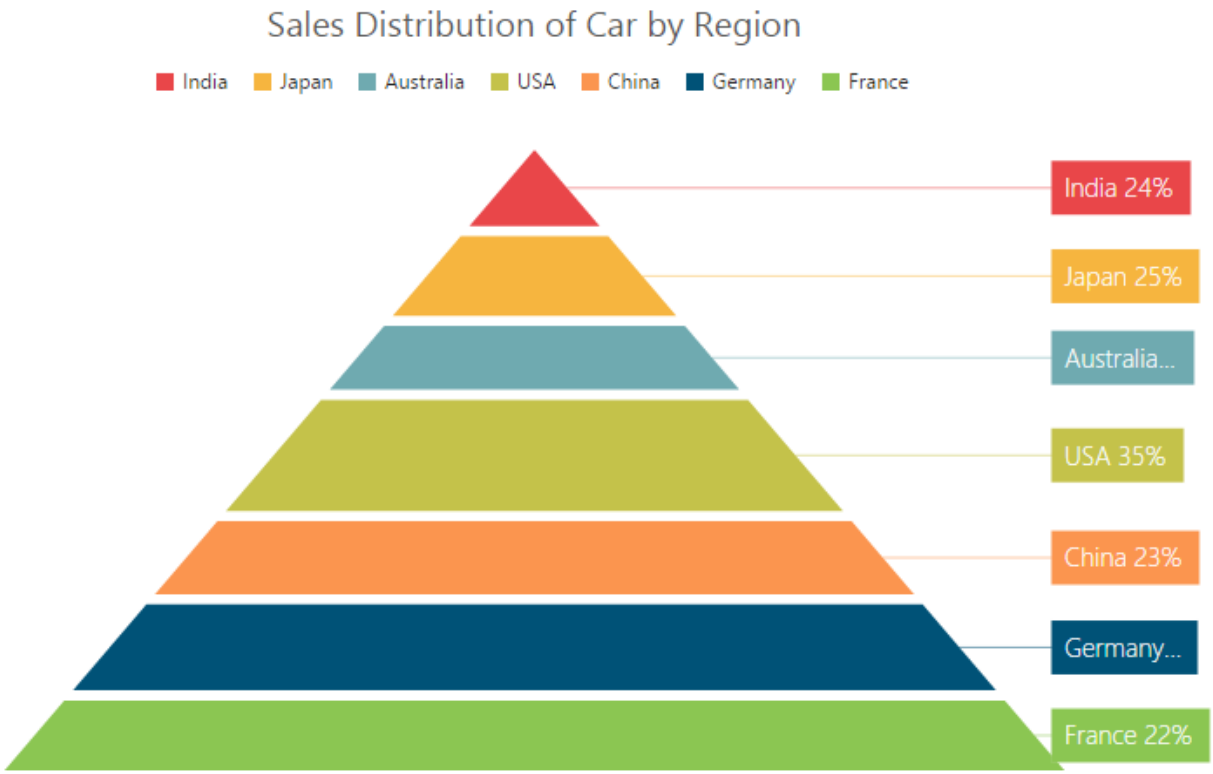
Gap between the segments

You can control the gap between the segments by using the **GapRatio** option of the series. Its ranges from 0 to 1.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set gapRatio value to pyramid chart
sr.GapRatio(0.1f)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



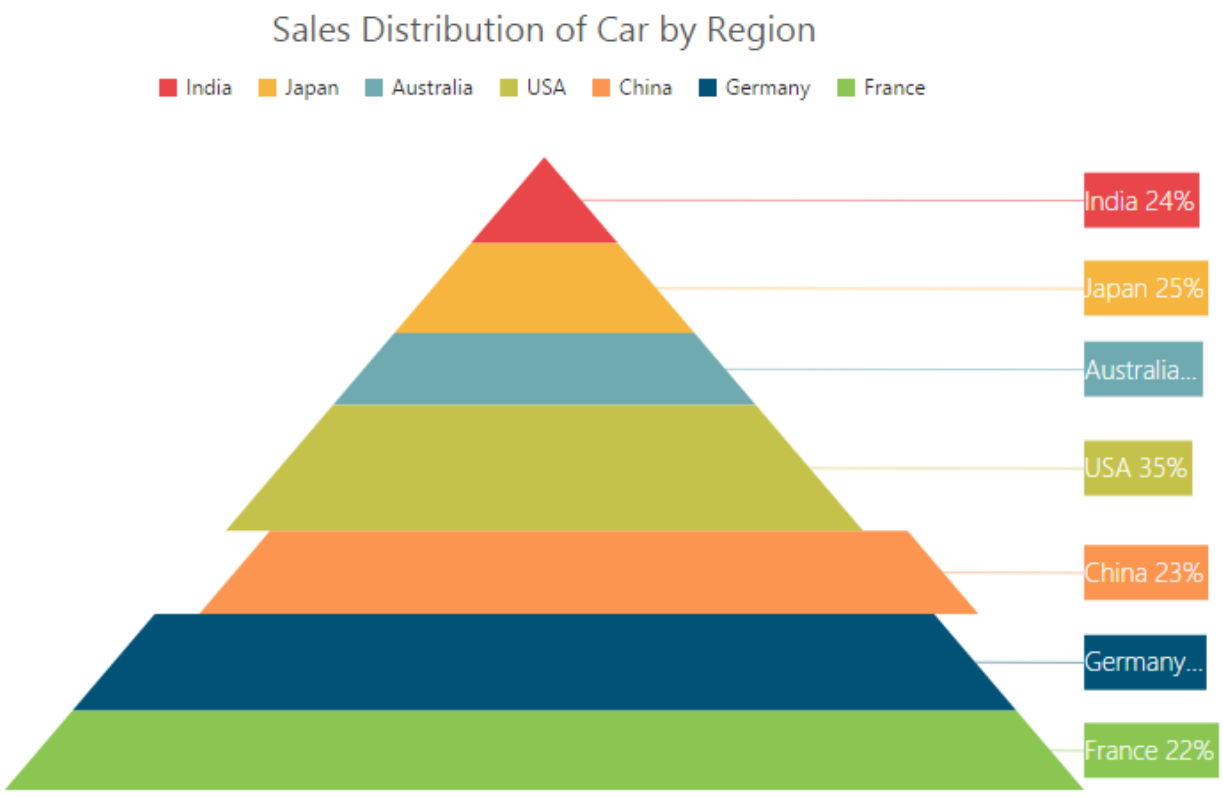
Explode a pyramid segment

You can explode a pyramid segment on the chart load by using the **ExplodeIndex** of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set point index value to explode the pyramid segment.
sr.ExplodeIndex(4)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



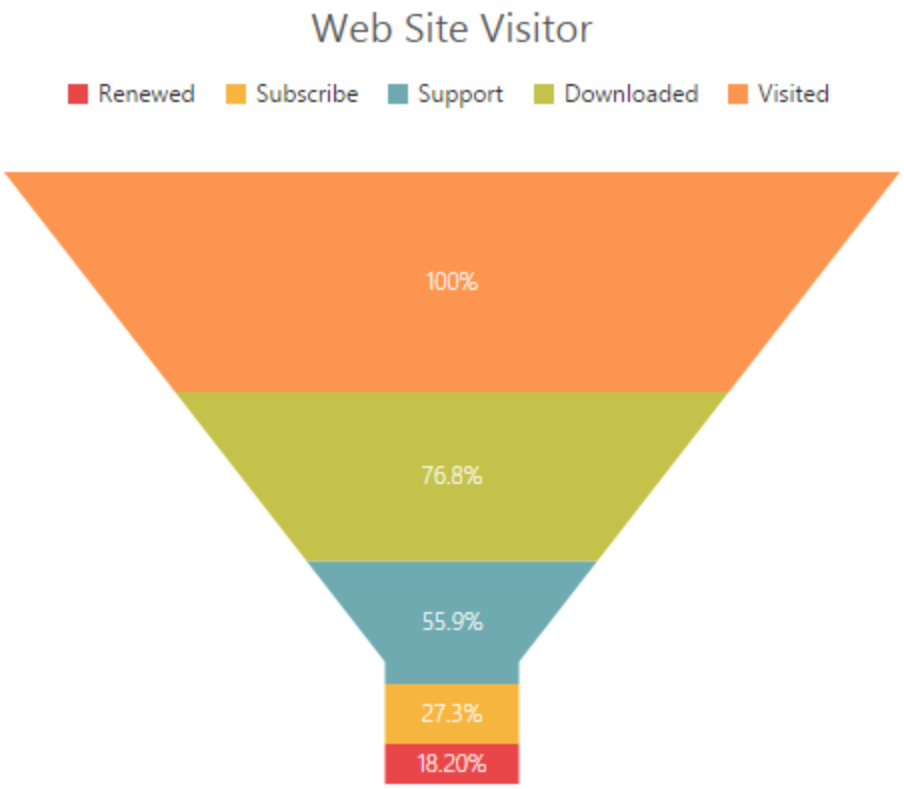
Funnel Chart

You can create a funnel chart by setting the series Type as **Funnel** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Funnel)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Funnel Chart online demo sample.

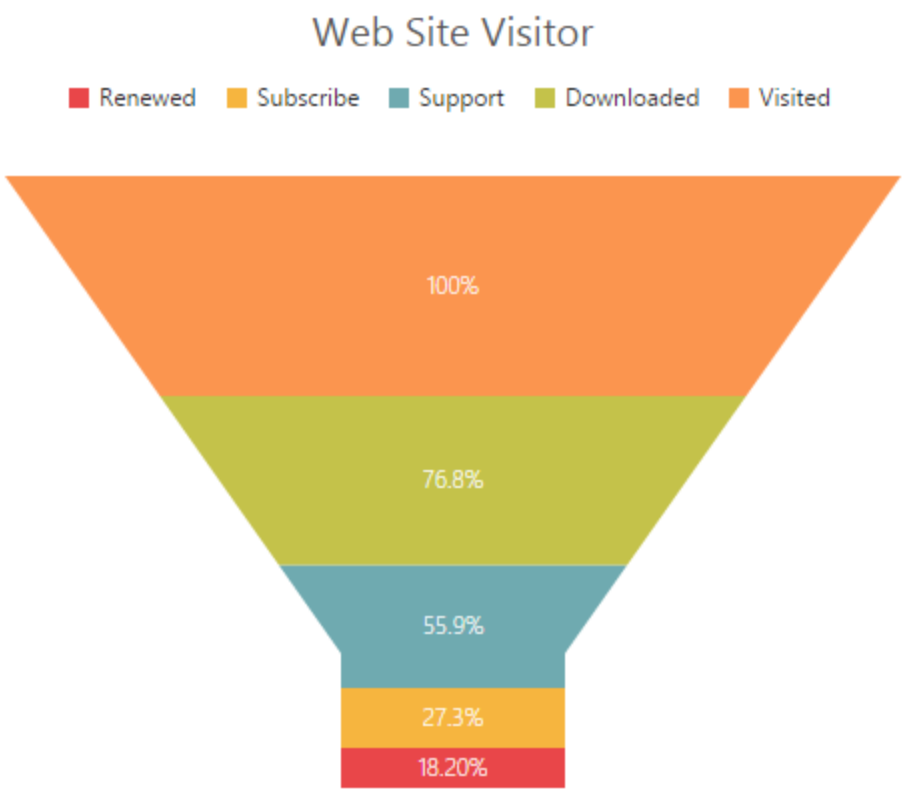
Change the funnel width and height

Funnel segments height and width is calculated from the chart size, by default. You can change this height and width directly without changing the chart size by using the **FunnelHeight** and **FunnelWidth** property of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change funnel height and width
sr.FunnelHeight("22%")
.FunnelWidth("25%")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



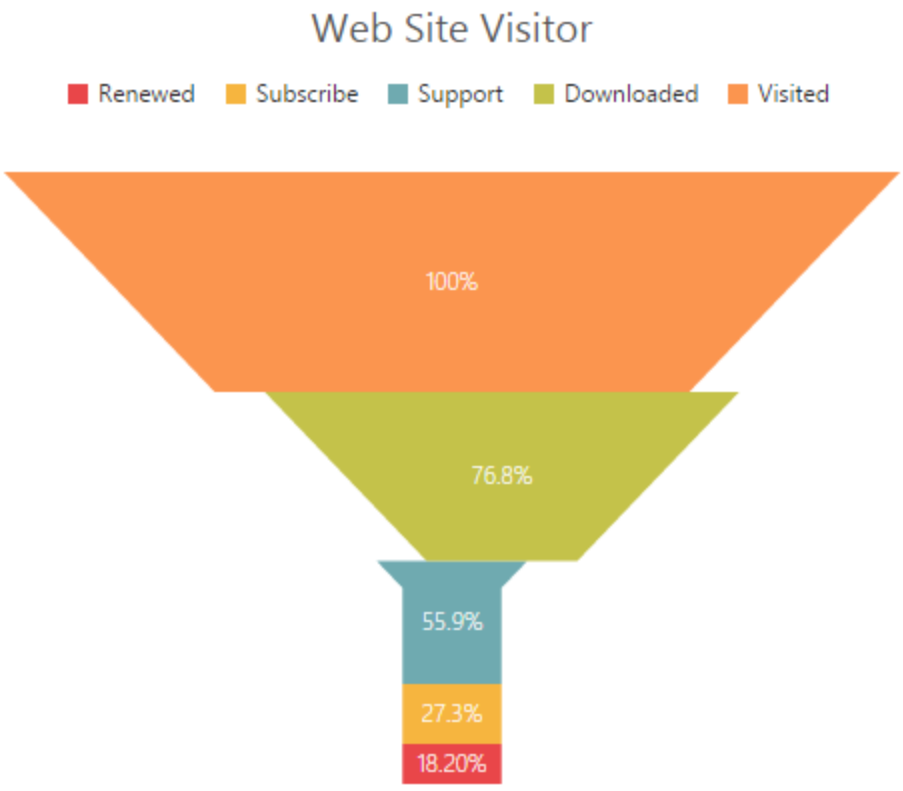
Explode a funnel segment

You can explode a funnel segment on the chart load by using the **ExplodeIndex** of the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set point index value to explode the funnel segment.
sr.ExplodeIndex(3)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Bubble Chart

To create a Bubble chart, you can set the series Type as **Bubble** in the chart series. Bubble chart requires 3 fields (*X, Y and Size*) to plot a point. Here, **Size** is used to specify the size of each bubble segment.

CSHARP

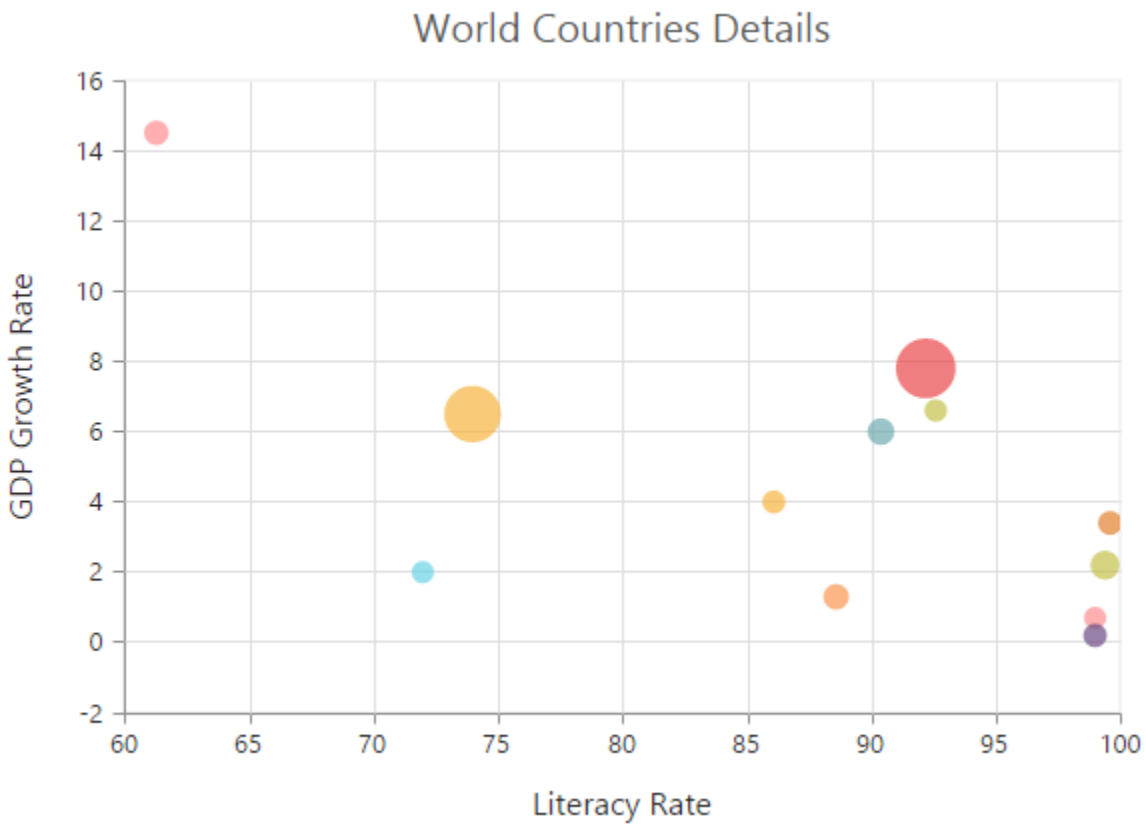
```
// Create dataSource to bubble chart
List<ChartData> data =new List<ChartData>();
data.Add(new ChartData("Jan", 35, 1.34));
data.Add(new ChartData("Feb", 28, 1.05));
data.Add(new ChartData("Mar", 34, 0.45));
data.Add(new ChartData("Apr", 32, 1.10));
ViewBag.ChartData = data;
```

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Bubble)
//Add datasource and set xName, yName and size to bubble chart
.XName("Month")
.XName("Sales")
.Size("Profit")
.DataSource(ViewBag.ChartData)
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Add();
})
//...
)
```



Click here to view the Bubble Chart online demo sample.

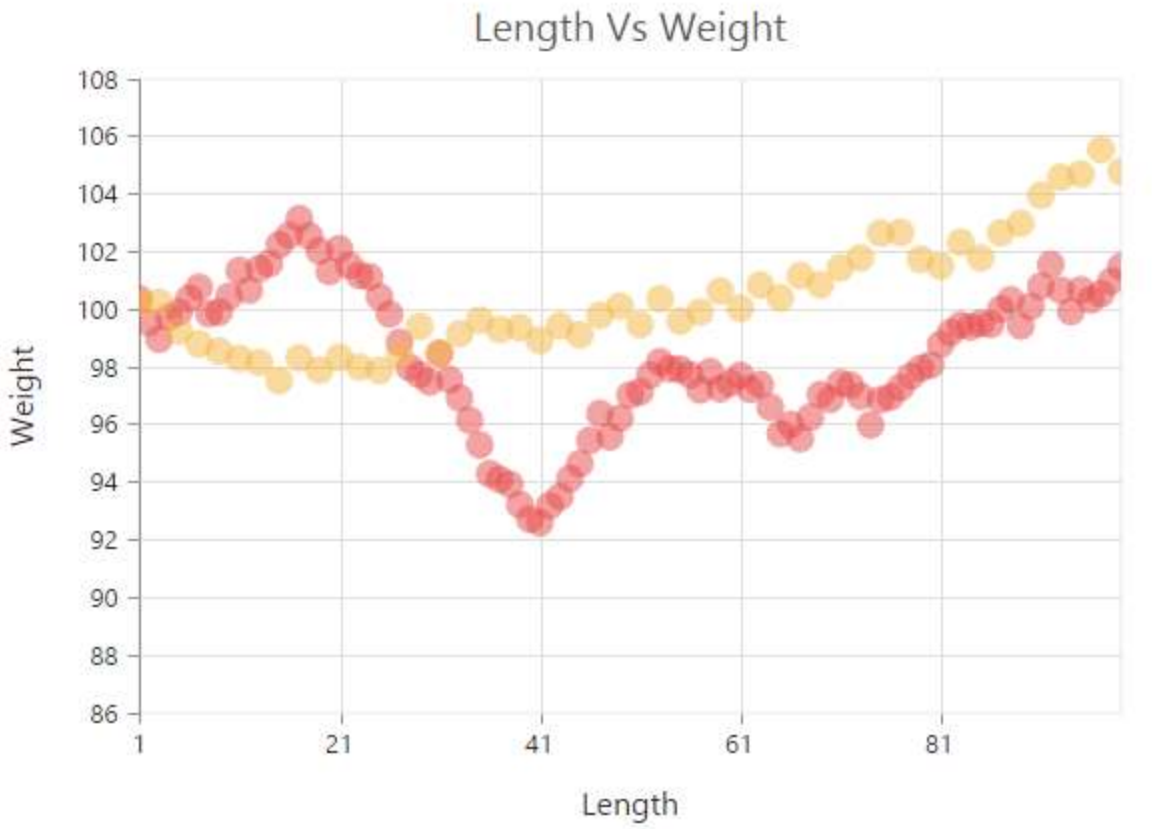
Scatter

To create a Scatter chart, you can set the series Type as **Scatter** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Scatter)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



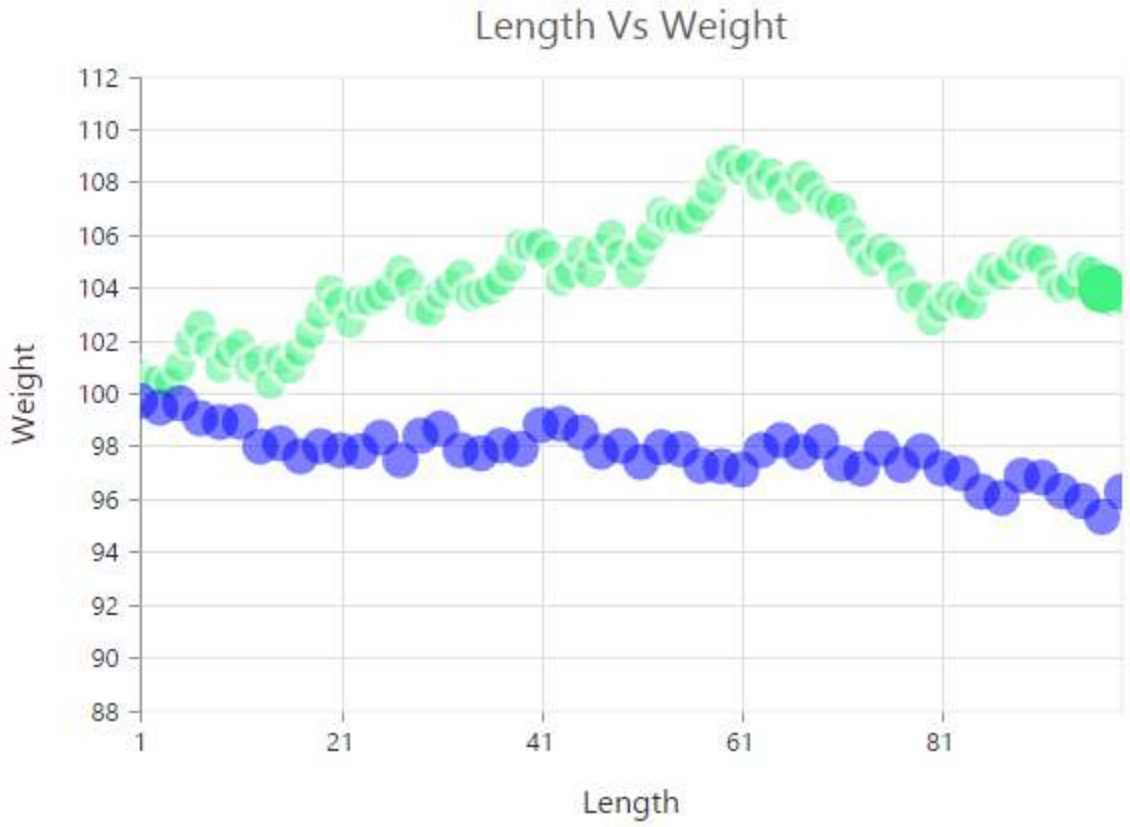
Click here to view the Scatter Chart online demo sample.

Customize the scatter chart

You can change the scatter size by using the **Size** property of the series marker. And you can change the scatter color by using the series Fill property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
sr.Marker(marker=>marker.Size(size=>size.Height(15).Width(15)))
//Set fill color to scatter series
.Fill("#41F282")
//...
.Add();
})
//...
)
```



HiLoOpenClose Chart

To create a HiLoOpenClose chart, you can set the series Type as **HiloOpenClose** in the chart series. HiLoOpenClose chart requires 5 fields *(X, High, Low, Open and Close)* to plot a segment.

CSHARP

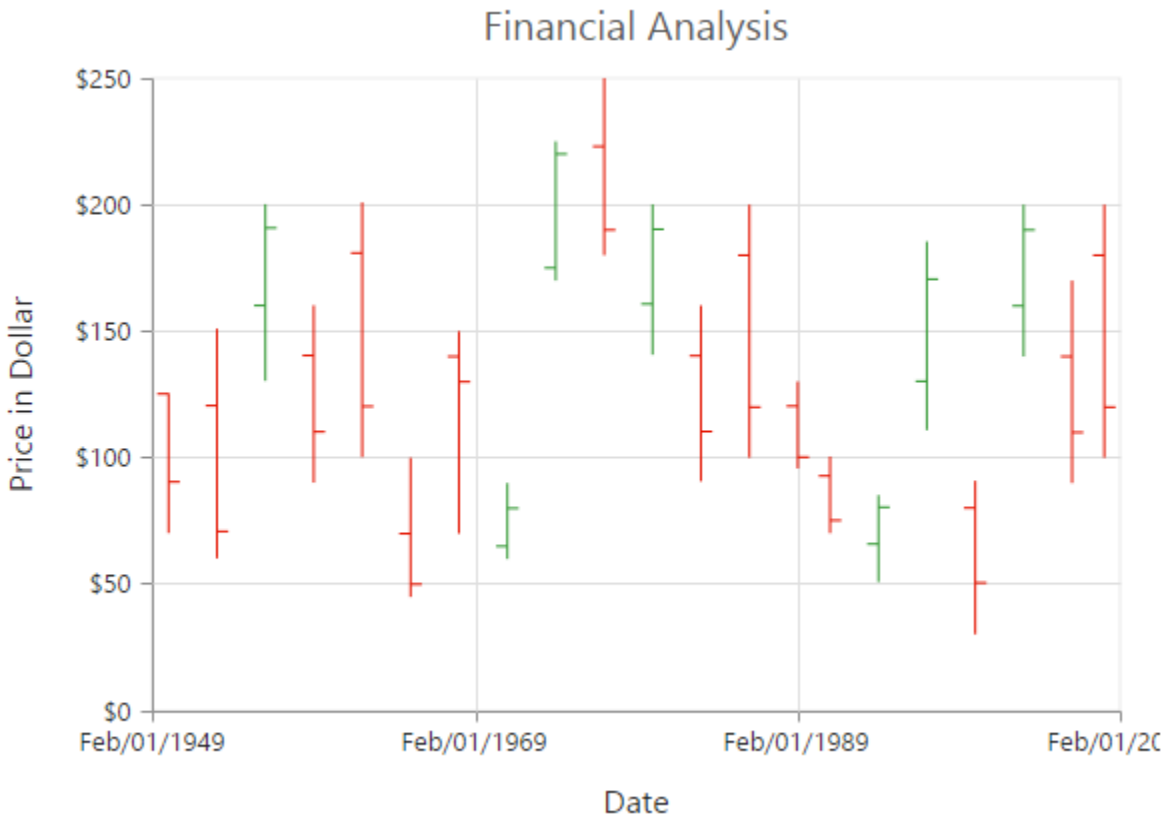
```
// Create dataSource to chart
List<ChartData> data =new List<ChartData>();
data.Add(new ChartData("Jan", 38, 10,38, 29));
data.Add(new ChartData("Feb", 28, 15, 18, 27));
data.Add(new ChartData("Mar", 54, 35, 38, 49));
data.Add(new ChartData("Apr", 52, 21, 35, 29));
ViewBag.ChartData = data;
```

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.HiloOpenClose)
//Add datasource and set xName, high and low to HiLo open close chart
.XName("Month")
.High("High")
.Low("Low")
.Open("Open")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Close("Close")
.DataSource(ViewBag.ChartData)
//...
.Add();
})
//...
)
```



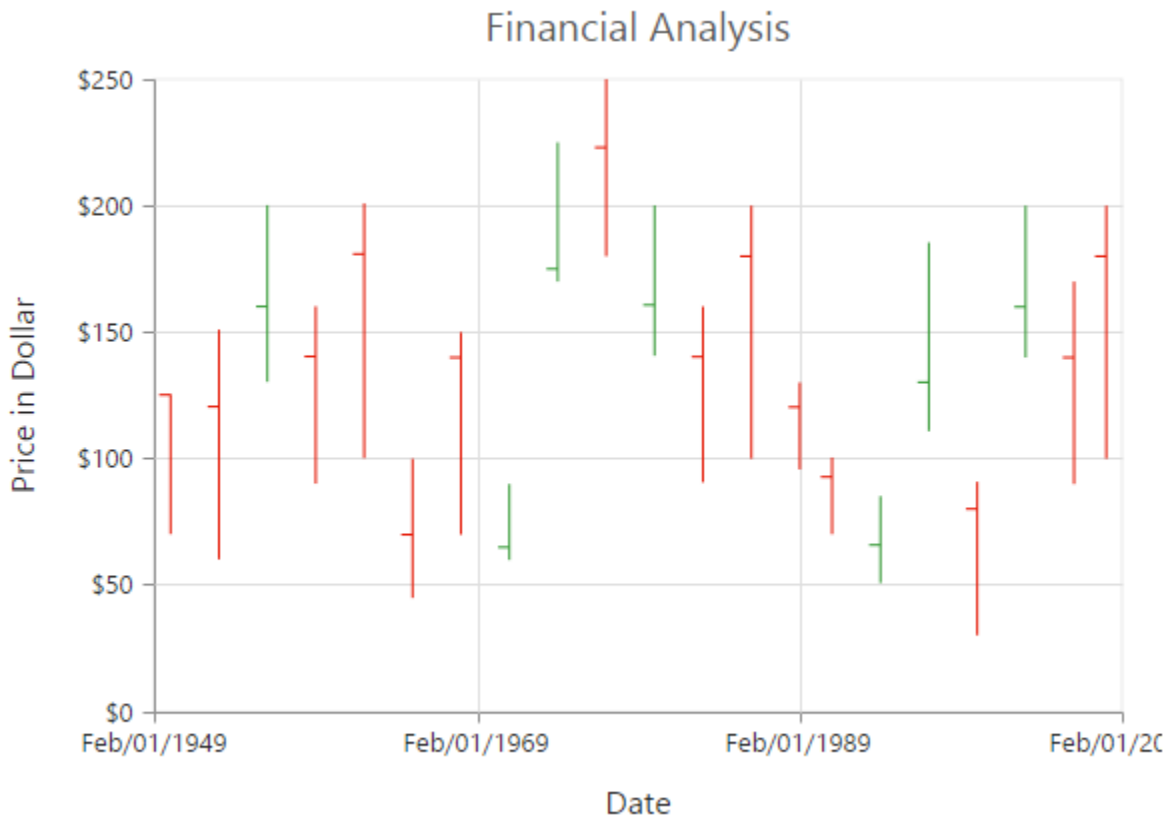
Click here to view the HiLoOpenClose Chart online demo sample.

DrawMode

You can change the HiLoOpenClose chart **DrawMode** to *Open, Close* or *Both*. The default value of DrawMode is **Both**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change the OHLC drawMode type
sr.DrawMode(SeriesDrawMode.Open)
//...
.Add();
})
//...
)
```



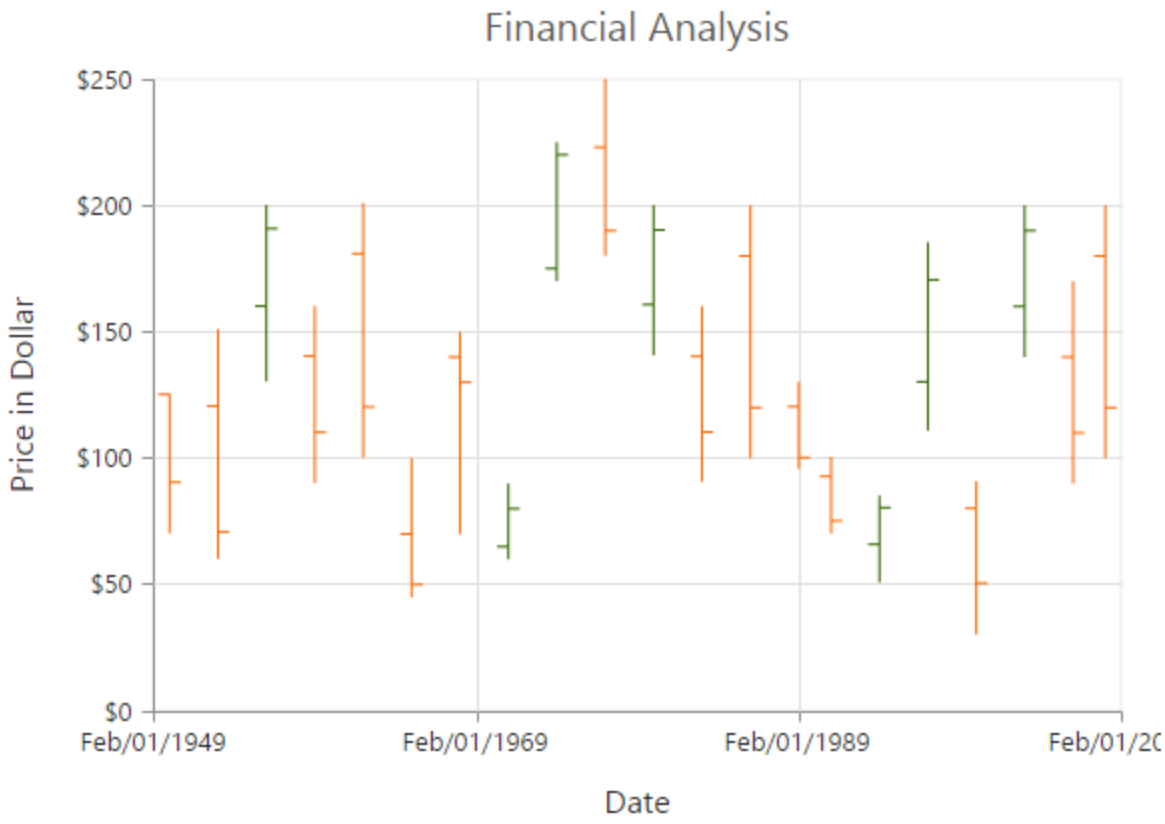
Bull and Bear Color

HiLoOpenClose chart **BullFillColor** is used to specify a fill color for the segments that indicates an increase in stock price in the measured time interval and **BearFillColor** is used to specify a fill color for the segments that indicates a decrease in the stock price in the measured time interval.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change bullFill and bearFill color of HiLo open close chart
sr.BearFillColor("#FF6600")
.BullFillColor("#336600")
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Candle

You can create a Candle chart by specifying the series Type as **Candle** in the chart series. Candle chart requires 5 fields *(X, High, Low, Open and Close)* to plot a segment.

CSHARP

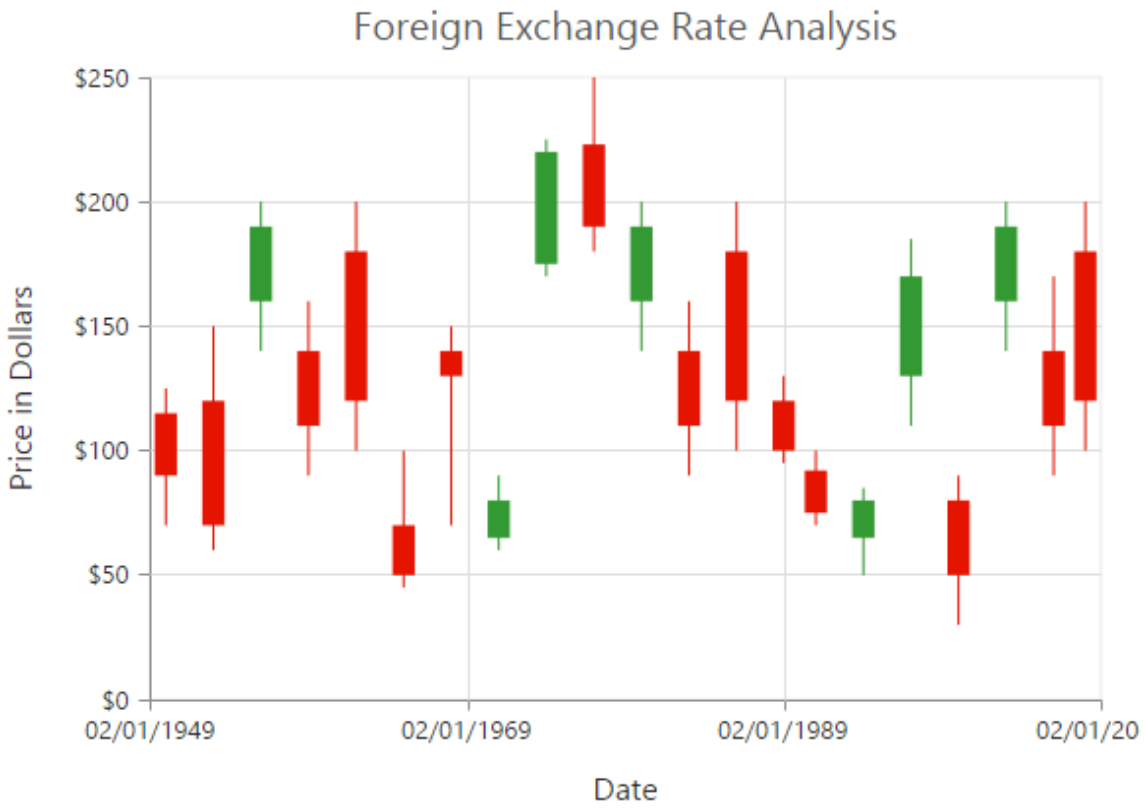
```
// Create dataSource to chart
List<ChartData> data =new List<ChartData>();
data.Add(new ChartData("Jan", 38, 10,38, 29));
data.Add(new ChartData("Feb", 28, 15, 18, 27));
data.Add(new ChartData("Mar", 54, 35, 38, 49));
data.Add(new ChartData("Apr", 52, 21, 35, 29));
ViewBag.ChartData = data;
```

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Candle)
//Add datasource and set xName, high and low to candle chart
.XName("Month")
.High("High")
.Low("Low")
.Open("Open")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Close("Close")
.DataSource(ViewBag.ChartData)
//...
.Add();
})
//...
)
```



Click here to view the Candle Chart online demo sample.

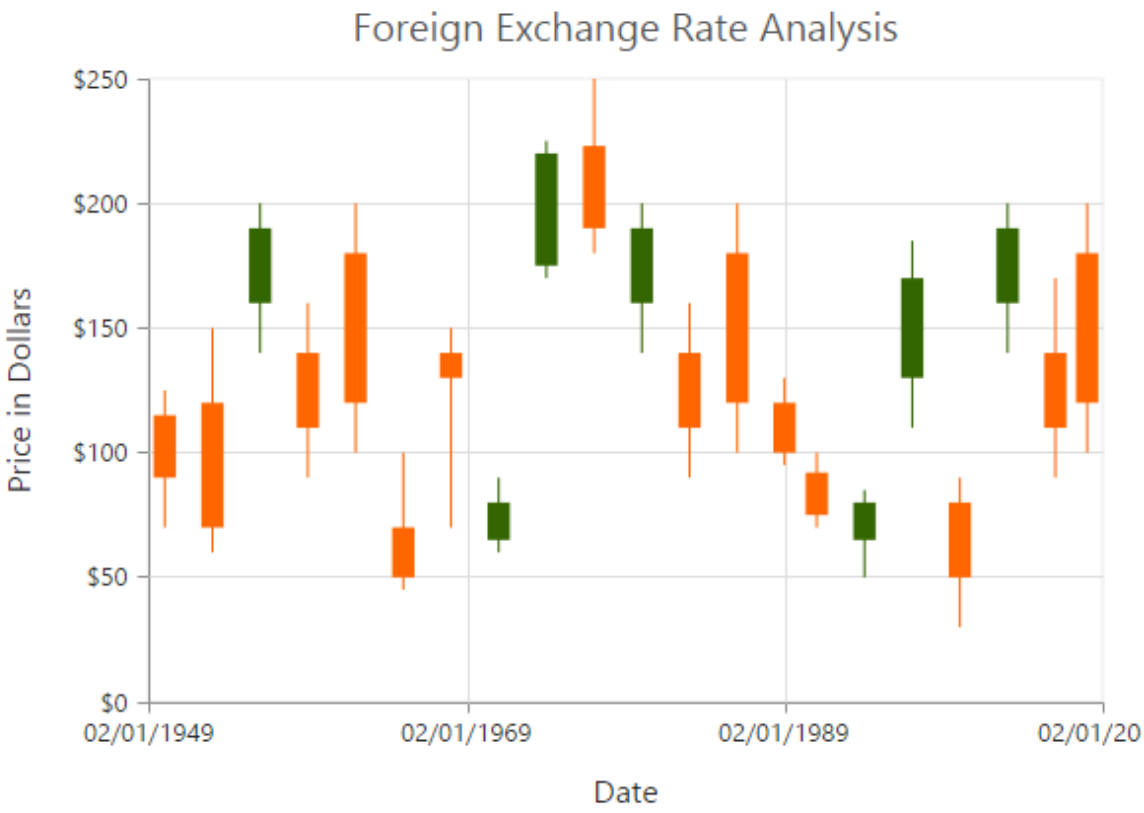
Bull and Bear Color

Candle chart **BullFillColor** is used to specify a fill color for the segments that indicates an increase in the stock price in the measured time interval and **BearFillColor** is used to specify a fill color for the segments that indicates a decrease in the stock price in the measured time interval.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change bullFill and bearFill color of candle chart
sr.BearFillColor("#FF6600")
.BullFillColor("#336600")
//...
.Add();
```

```
})
//...
)
```



HiLo

HiLo chart is created by setting the series Type as **Hilo** in the chart series. HiLo chart requires 3 fields *(X, High and Low)* to plot a segment.

CSHARP

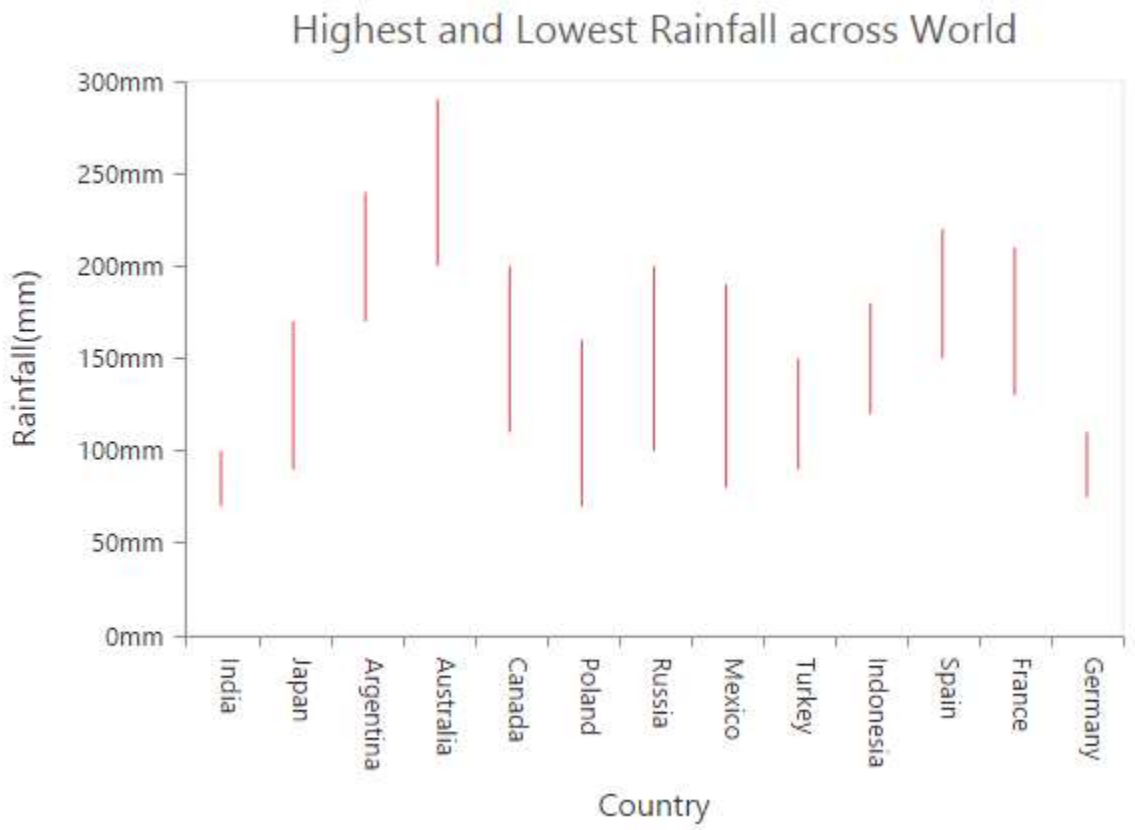
```
// Create dataSource to chart
List<ChartData> data =new List<ChartData>();
data.Add(new ChartData("Jan", 38, 10));
data.Add(new ChartData("Feb", 28, 15));
data.Add(new ChartData("Mar", 54, 35));
data.Add(new ChartData("Apr", 52, 21));
ViewBag.ChartData = data;
```

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Hilo)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Add datasource and set xName, high and low to HiLo chart
.XName("Month")
.High("High")
.Low("Low")
//...
.Add();
})
//...
)
```



Click here to view the HiLo Chart online demo sample.

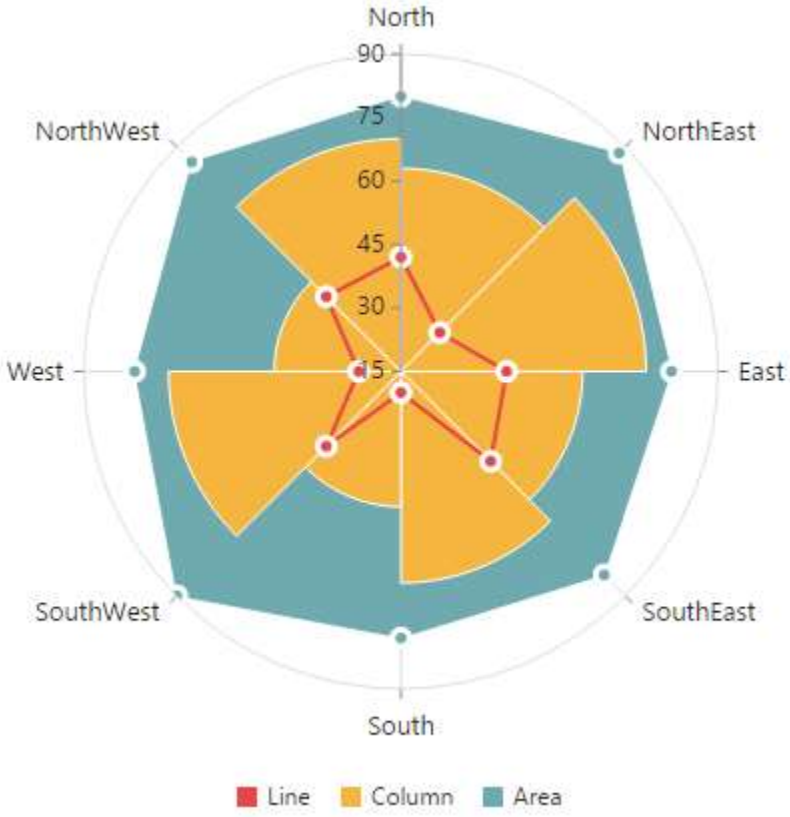
Polar

Polar chart is created by setting the series Type as **Polar** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Polar)
//...
.Add();
})
```

```
//...
)
```



Click here to view the Polar Chart online demo sample.

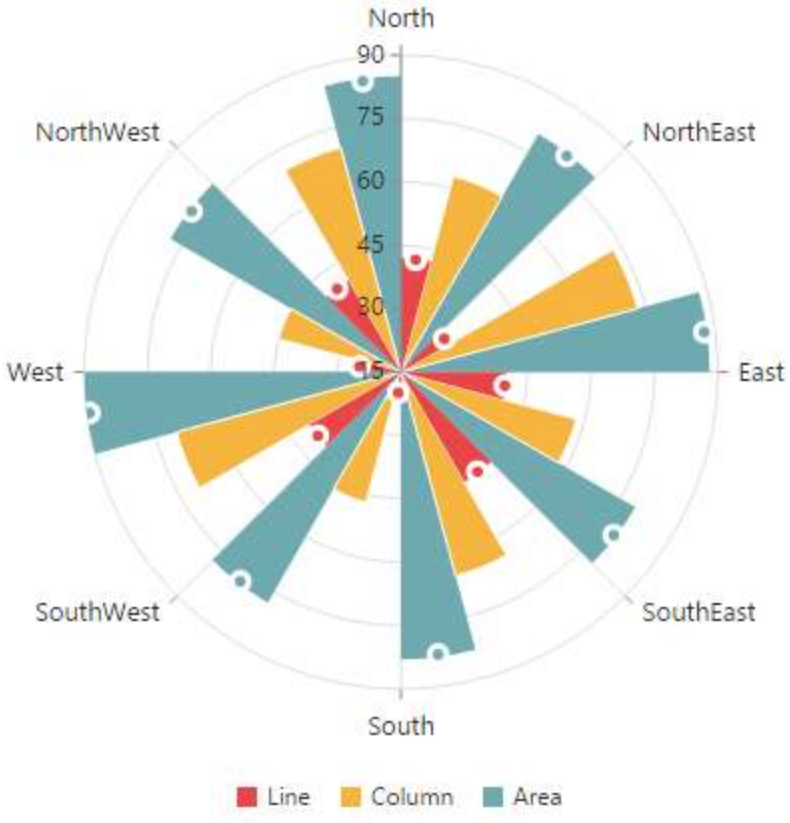
DrawType

Polar **DrawType** property is used to change the series plotting type to *Line*, *Column* or *Area*. The default value of DrawType is **Line**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change polar series drawType
sr.DrawType(DrawType.Column)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

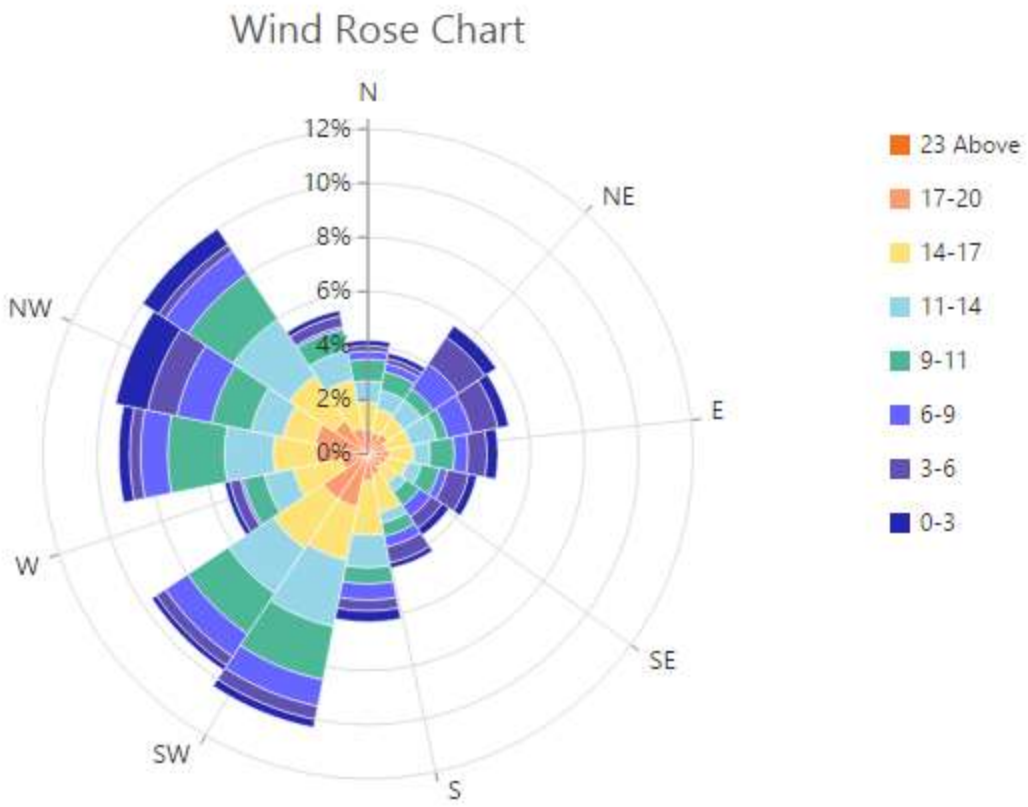


Stack columns in Polar chart

By using the **IsStacking** property, you can specify whether the column has to be stacked when the DrawType is column. Its default value is **false**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Enable isStacking property for stacked column polar chart
sr.IsStacking(true)
//...
.Add();
})
//...
)
```



Click here to view the Polar Wind Rose Chart online demo sample.

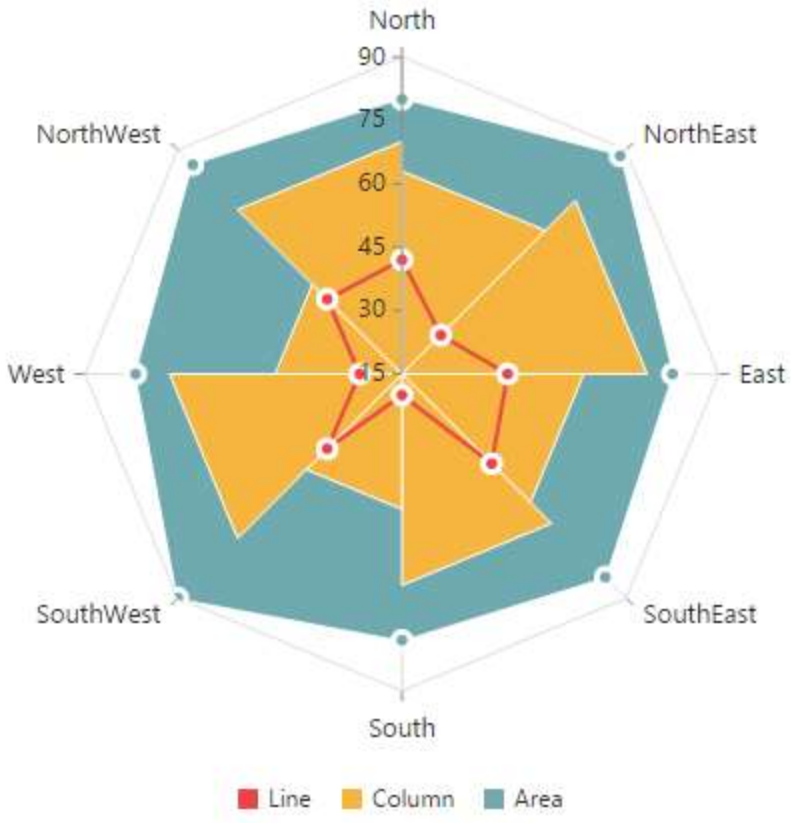
Radar Chart

To create a Radar chart, you can specify the series Type as **Radar** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Set chart type to series
sr.Type(SeriesType.Radar)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Radar Chart online demo sample.

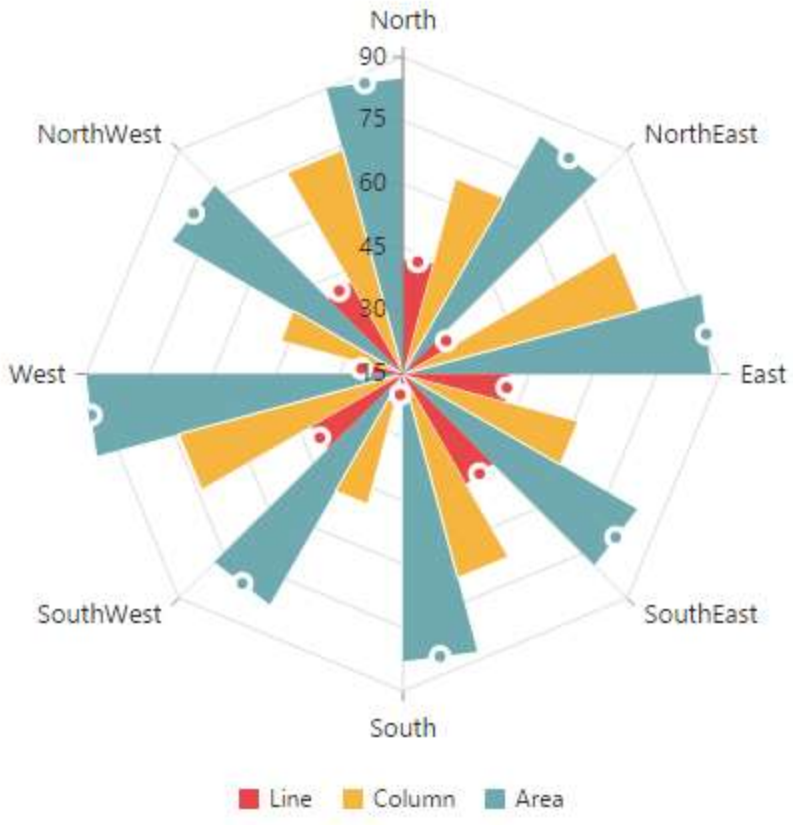
DrawType

Radar **DrawType** property is used to change the series plotting type to *Line*, *Column* or *Area*. The default value of DrawType is **Line**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Change radar series drawType
sr.DrawType(DrawType.Column)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



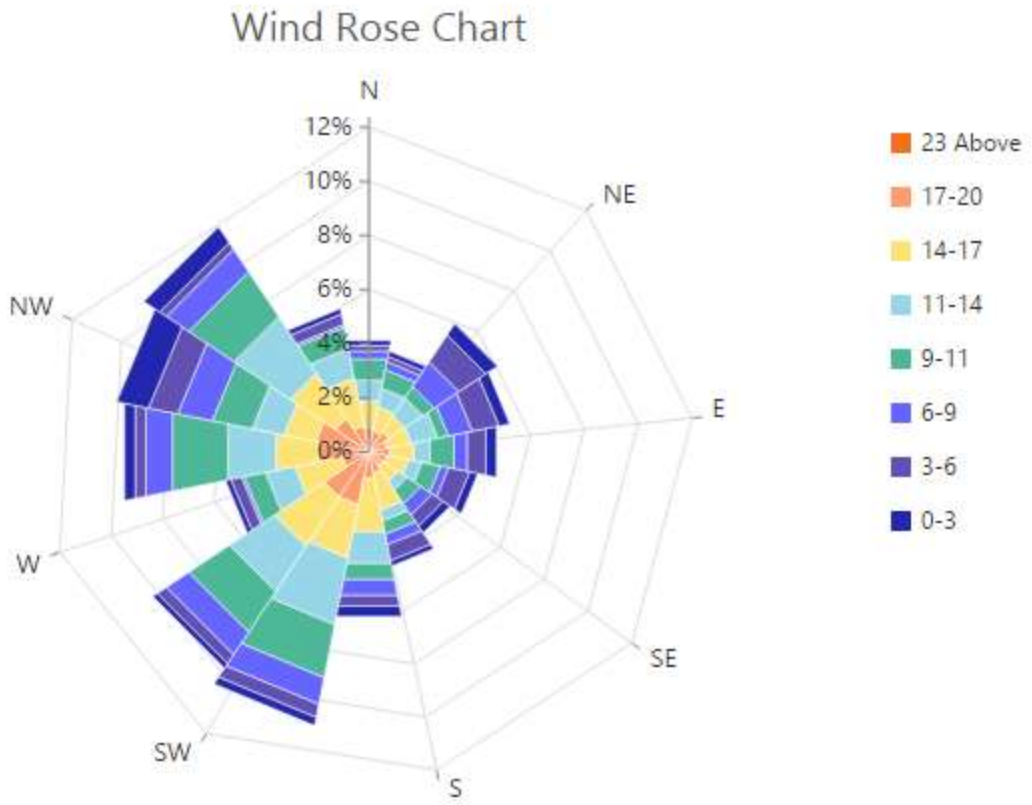
Stack columns in Radar chart

By using the **IsStacking** property, you can specify whether the column has to be stacked when the DrawType is set as Column. Its default value is set to *false*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
//Enable isStacking property for stacked column radar chart
sr.IsStacking(true)
//...
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Waterfall Chart

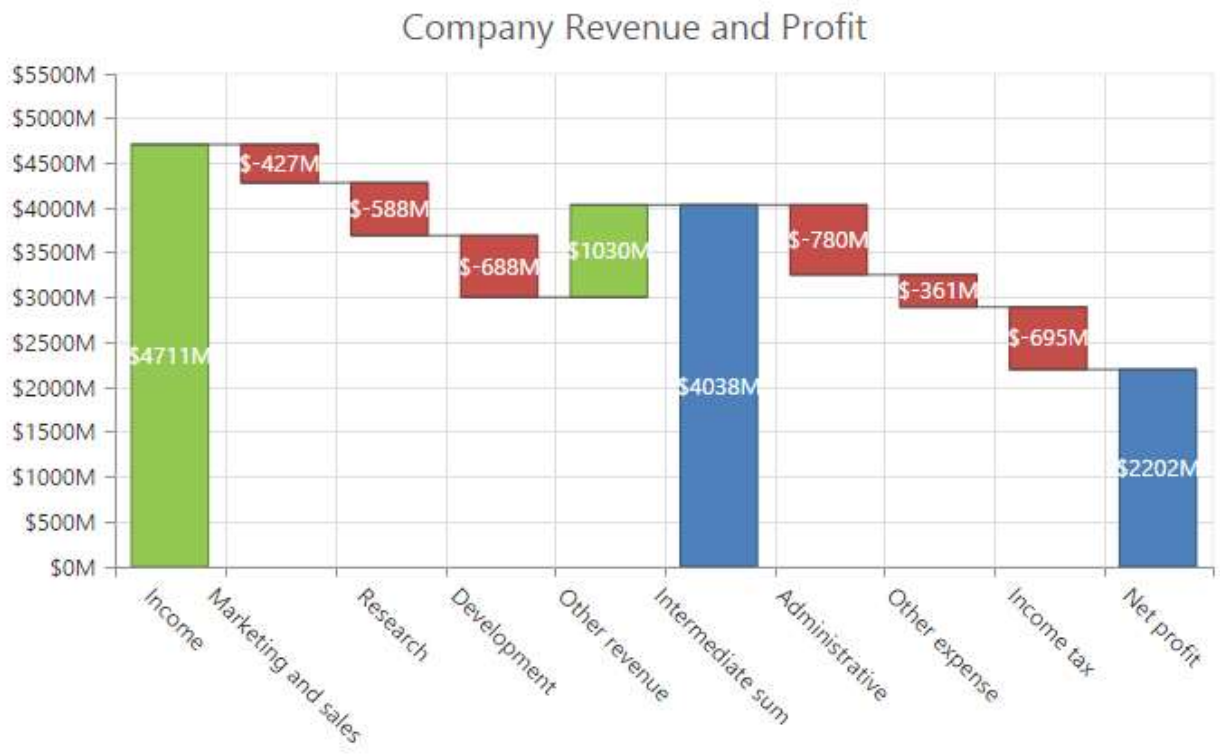
For rendering a Waterfall chart, set series *Type* as **Waterfall** in the chart series. To change the waterfall series segment color use *Fill* option of series and use *PositiveFill* property to differentiate the positive segments.

Note: The inline property of the **Series.PositiveFill** has the first priority and override the **Series.Fill**.

ASPX-CS

```
@(Html.EJ().Chart("container")
.Series(sr =>
{
//...
//Change series type
sr.Type(SeriesType.Waterfall)
.Fill("#C64E4A").PositiveFill("#93C952").Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Waterfall Chart online demo sample.

ShowIntermediateSum

To display the summary of values since the last intermediate point of the waterfall series, set **ShowIntermediateSum** property as true in the specific point.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
sr.Points(pt =>
{
//Enable showIntermediateSum in to a point.
pt.X("Intermediate sum").ShowIntermediateSum(true).Add();
//...
}).Add();
})
//...
)
```

ShowTotalSum

The sum of all previous point in the waterfall series is displayed on enabling the **ShowTotalSum** property for a specific point.

ASPX-CS

```
@(Html.EJ().Chart("container")
```

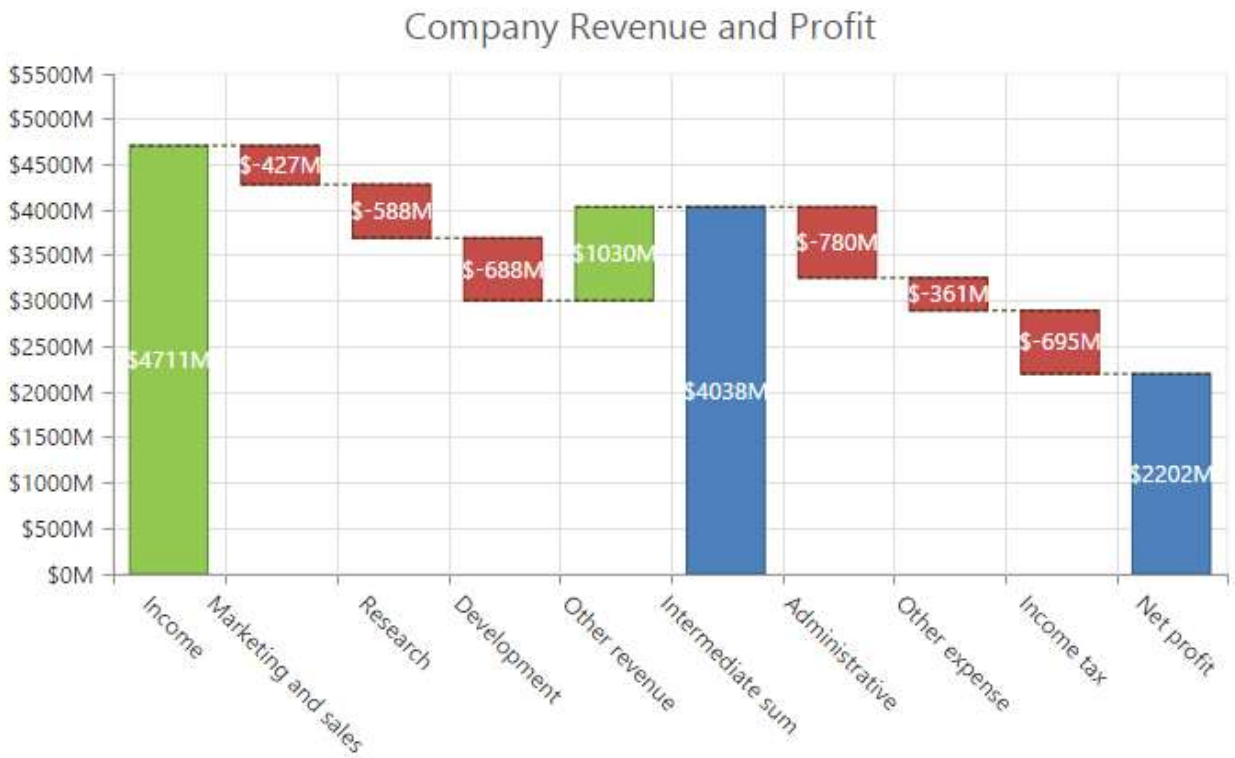
```
//...
.Series(sr =>
{
sr.Points(pt =>
{
//Enable showTotalSum in to a point.
pt.X("Total sum").ShowTotalSum(true).Add();
//...
}).Add();
})
//...
)
```

ConnectorLine

To customize the connector line color, width, opacity and dashArray of the waterfall series, you can use **ConnectorLine** option of series.

ASPX-CS

```
@(Html.EJ().Chart("container")
.Series(sr =>
{
sr .ConnectorLine(cl=>cl.Color("#333000")
.Width(1).Opacity(1).DashArray("2,3"))
.Add();
})
//...
)
```



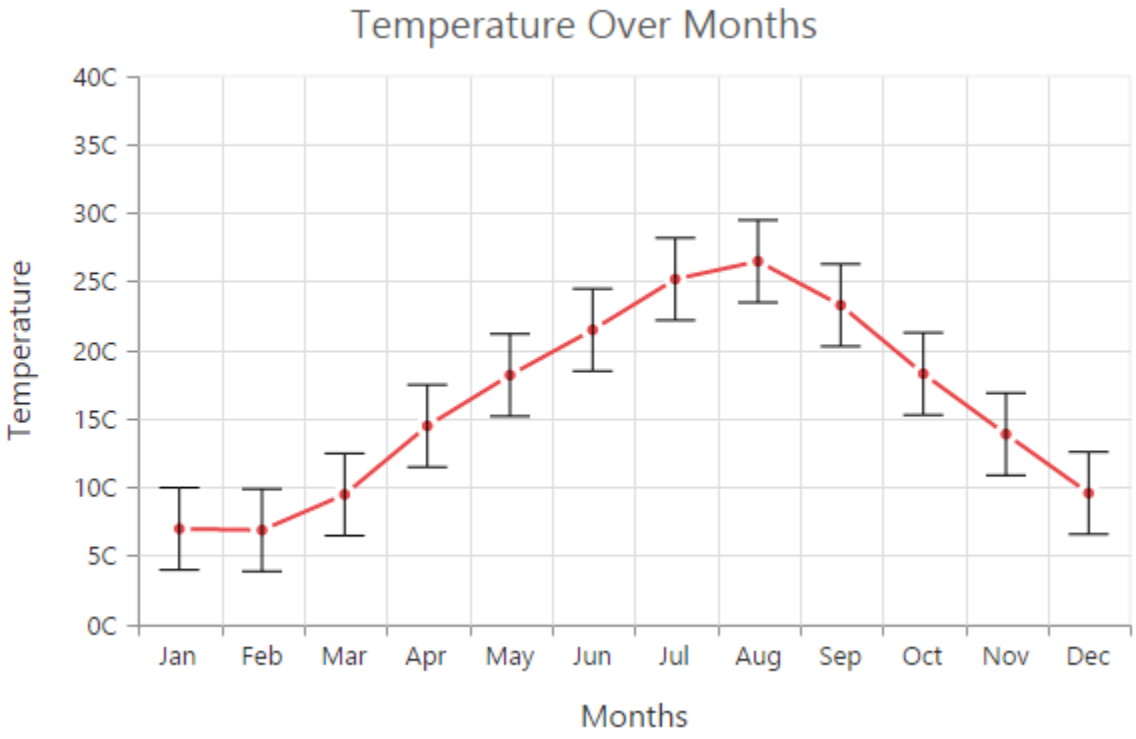
Copyright © 2001 - 2020 Syncfusion Inc.

ErrorBar Chart

EjChart can generate Error bar for Cartesian type series *(Line, Column, Bar, Scatter, Area, Candle, HiLo, etc.)*. To render the Error bar for the series, set *Visibility* as *"visible"* to **ErrorBar** in the series.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
//To toggle the error bar visibility
sr.ErrorBar(era =>
era.Visible("visible")).Add();
})
//...
)
```



Click here to view the ErrorBar Chart online demo sample.

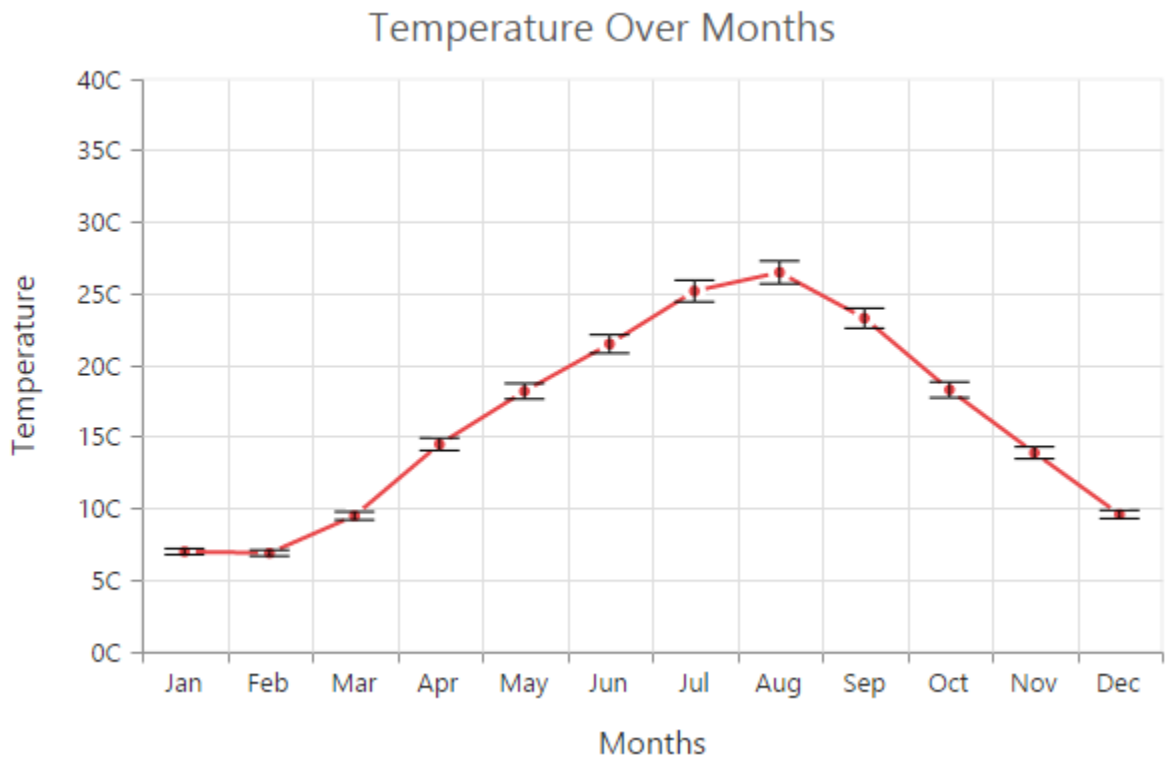
Changing Error Bar Type

You can change the error bar rendering type using **Type** *(like FixedValue, Percentage, StandardDeviation, StandardError and Custom)* option of errorBar. To change the error bar line length you can use **VerticalErrorValue** property.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
//To change the error bar type
```

```
sr.ErrorBar(era =>
era.Type(ErrorBarType.FixedValue)
.VerticalErrorValue(3)).Add();
})
//...
)
```



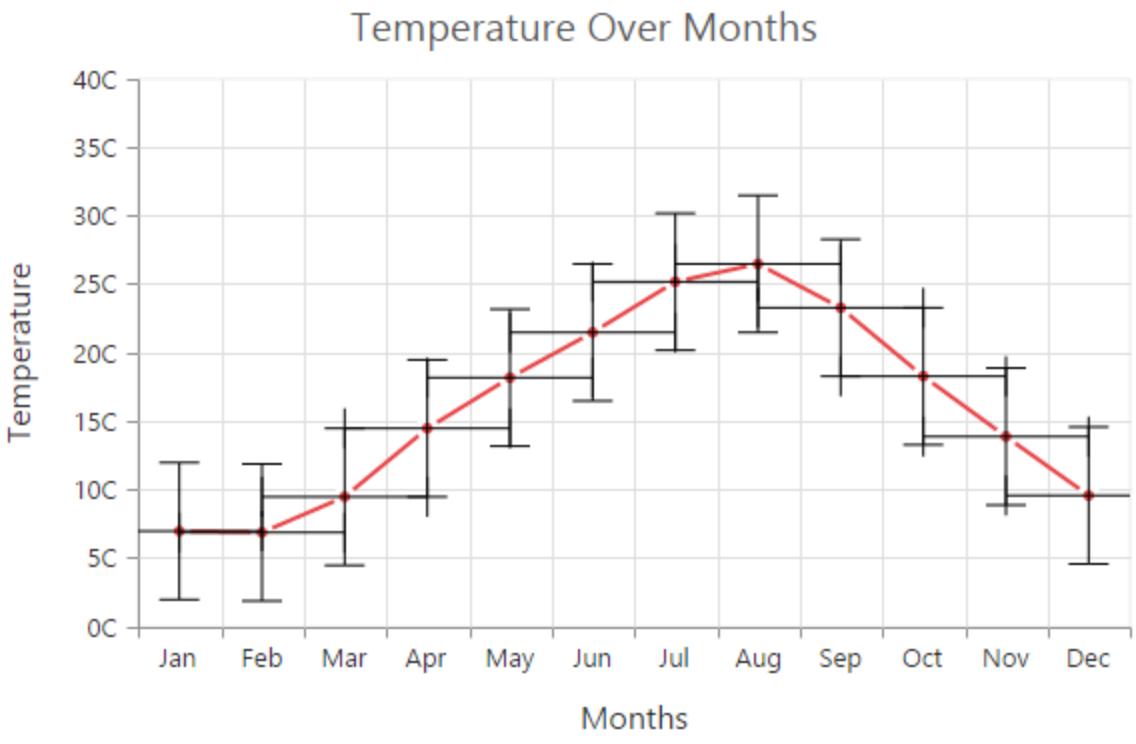
Customizing error bar type

To customize the error bar type, set error bar Type as **Custom** and then change the horizontal/vertical positive and negative value of error bar.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
//To change the error bar type.
sr.ErrorBar(era =>
era.Type(ErrorBarType.Custom)
.VerticalPositiveErrorValue(5)
.HorizontalPositiveErrorValue(1)
.VerticalNegativeErrorValue(5)
.HorizontalNegativeErrorValue(1)).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



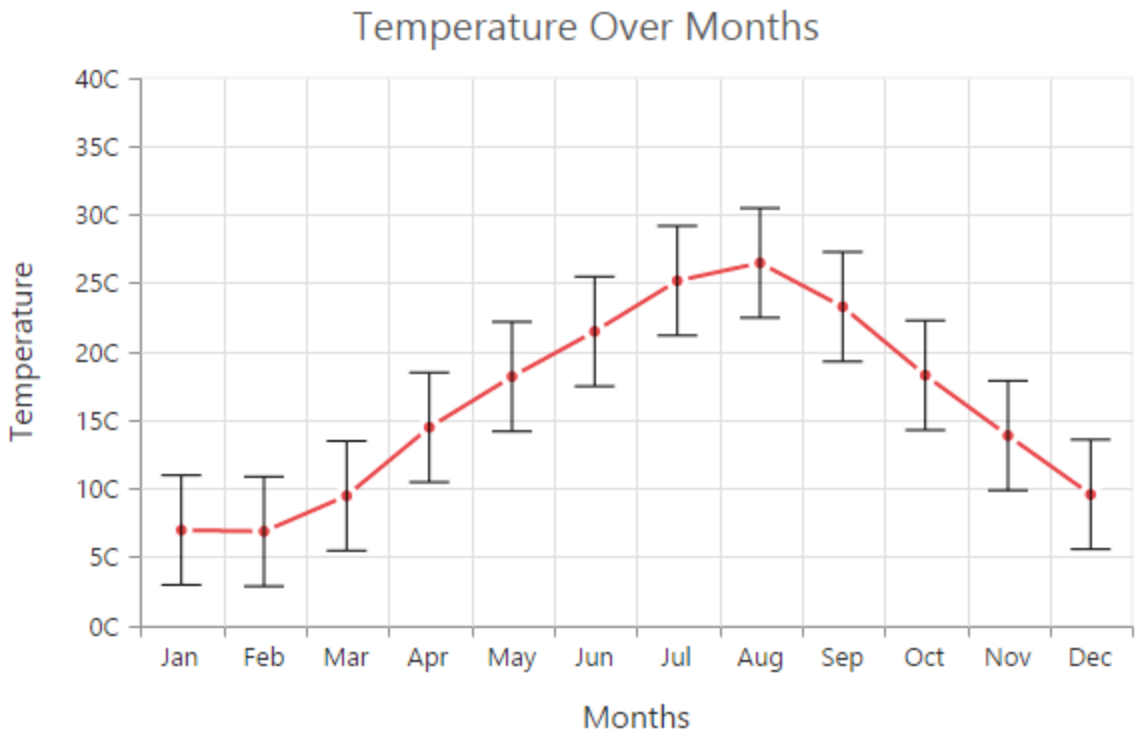
Changing Error Bar Mode

Error bar mode is used to define whether the error bar line has to be drawn *Horizontally, Vertically* or in *Both* side. To change the error bar mode use ErrorBar.Mode option.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
//To change the error bar mode.
sr.ErrorBar(era =>
era.Type(ErrorBarType.FixedValue)
.Mode(ErrorBarMode.Vertical)).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



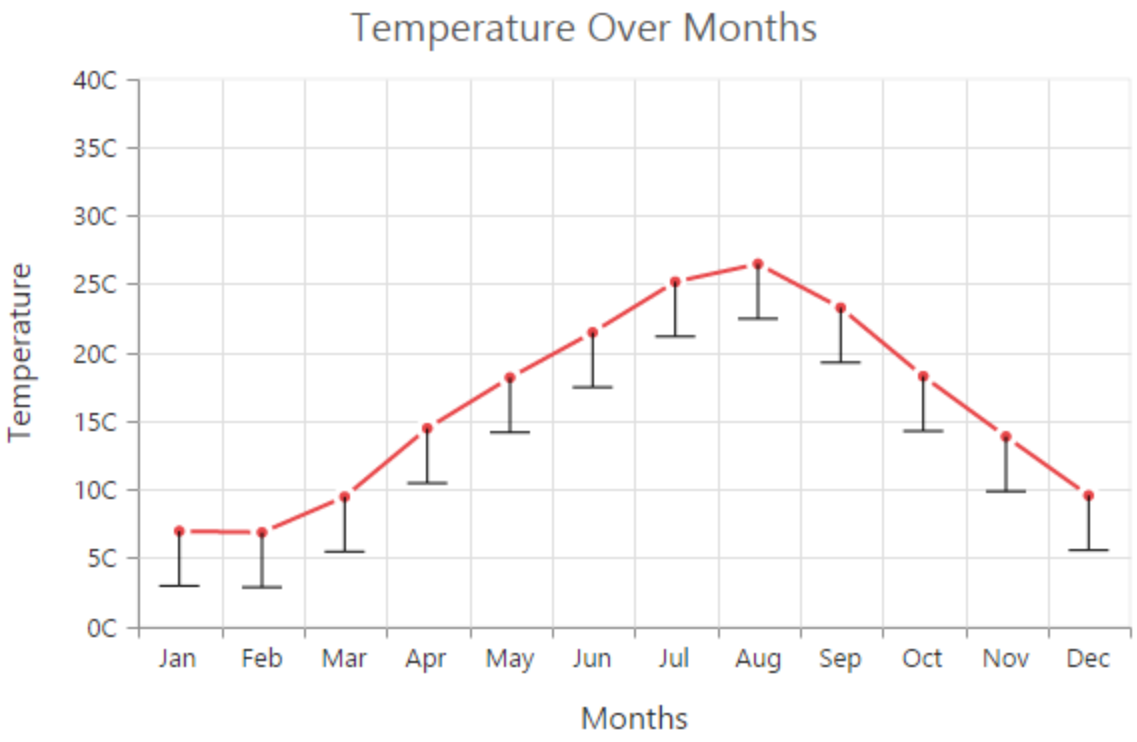
Changing Error Bar Direction

You can change the error bar direction to plus, minus or both side using ErrorBar.Directions option.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
//To change the error bar direction
sr.ErrorBar(era =>
era.Type(ErrorBarType.FixedValue)
.Mode(ErrorBarMode.Vertical)
.Direction(ErrorBarDirection.Minus)).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

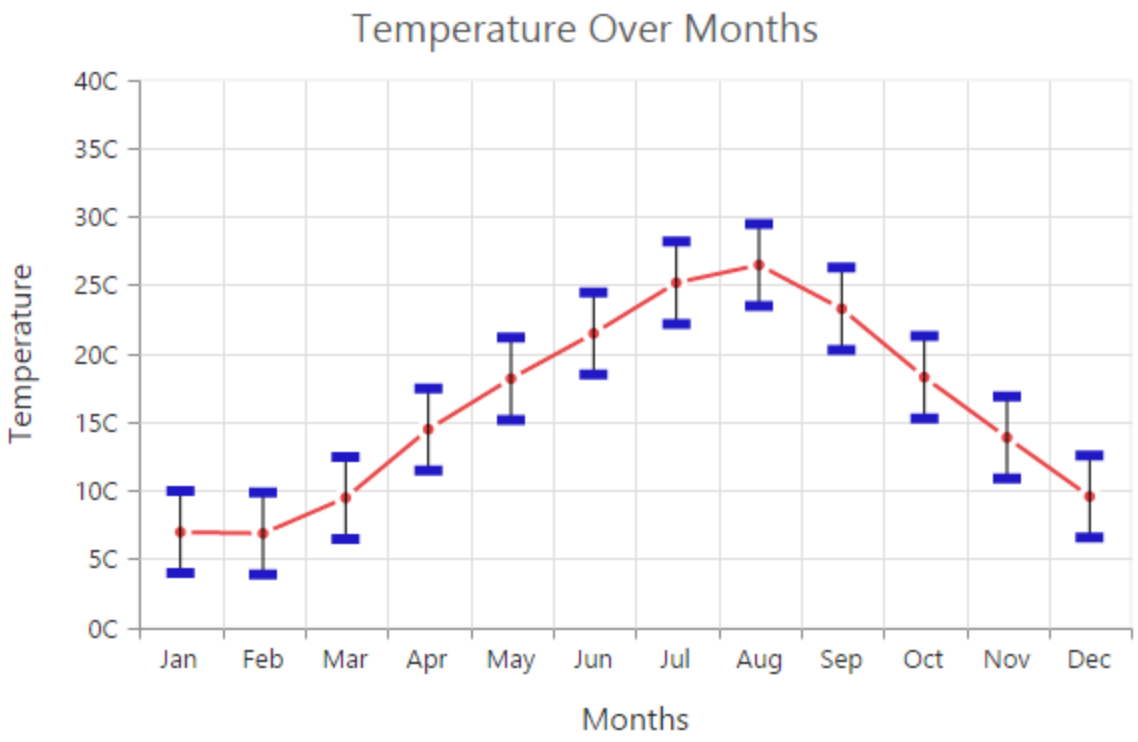


Customizing Error bar cap

To customize the errorBar cap visibility, length, width and fill color, you can use **Cap** option in the **Series.ErrorBar**.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
//To customize the error bar cap
sr.ErrorBar(era =>
era.Cap(cap=>
cap.Visible(true)
.Length((20),
.Width(1),
.Fill("#000000"))).Add();
})
//...
)
```



Box and Whisker Chart

To render a Box and Whisker Chart, set the series *type* as **BoxAndWhisker** .Box and Whisker chart requires

2 fields (x and y) to plot a segment. The field y requires n number of data or it should contain minimum of five values to plot a segment.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
sr.Points(pt =>
{
pt.X("Development").Y(22,22,23,25,25,25,26,27,27,28,28,29,30,32,34,32,34,36,
35,38).Add();
pt.X("Testing").Y(22,33,23,25,26,28,29,30,34,33,32,31,50).Add();
pt.X("HR").Y(22,24,25,30,32,34,36,38,39,41,35,36,40,56).Add();
pt.X("Finance").Y(26,27,28,30,32,34,35,37,35,37,45).Add();
pt.X("R&D").Y(26,27,29,32,34,35,36,37,38,39,41,43,58).Add();
})
.Type(SeriesType.BoxAndWhisker)
})
//...
)
```



BoxPlotMode

You can change the rendering mode of the Box and Whisker series using the *BoxPlotMode* property .The default value of *BoxPlotMode* is **exclusive**.

The other boxPlotModes available are inclusive and normal.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
.BoxPlotMode(BoxPlotMode.Inclusive)
})
//...
)
```

ShowMedian

Box and Whisker *showMedian* property is used to show the box and whisker average value. The default value of *showMedian* is **false**.

ASPX-CS

```
@(Html.EJ().Chart("container")
```

```
//...
.Series(sr =>
{
.ShowMedian(true)
})
//...
)
```



Customize the Outlier

Outlier symbol, width and height can be customized using outlierSettings through *outlierSettings* property. By default Outlier symbol is displayed as circle with a height and width of 6 pixels.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
.OutlierSettings(outlier=>outlier.Shape(ChartShape.Triangle)
.Size(size=>size.Height(10).Width(10))).Add();
})
//...
)
```



Click here to view the Box and Whisker Chart online demo sample.

Pie Of Pie Chart

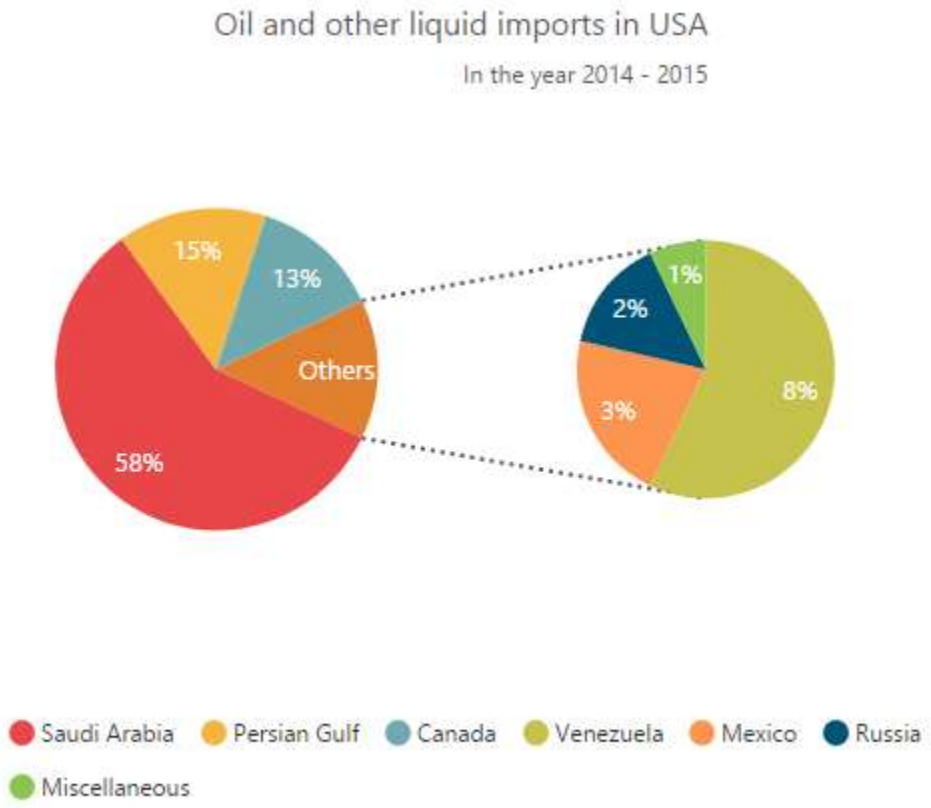
To render the pie of pie chart, set the series Type as **PieOfPie**. Pie of pie chart is used for displaying the data of a pie slice as another pie chart. The values in the second pie is displayed based on the **splitMode** property.

ASPX-CS

```
@(Html.EJ().Chart("container")
.Series(sr =>
{
sr.Points(pt =>
{
pt.X("Saudi Arabia").Y(58).Text("58%").Add();
pt.X("Persian Gulf").Y(15).Text("15%").Add();
pt.X("Canada").Y(13).Text("13%").Add();
pt.X("Venezuela").Y(8).Text("8%").Add();
pt.X("Mexico").Y(3).Text("3%").Add();
pt.X("Russia").Y(2).Text("2%").Add();
pt.X("Miscellaneous").Y(1).Text("1%").Add();
})
.SplitValue("10")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Type(SeriesType.PieOfPie).Add();
})
)
```



Click here to view the Pie Of Pie Chart online demo sample.

Split Mode and Split Value

The points to be displayed in the second pie is decided based on the splitMode property.**SplitMode** property takes the following values.

- Position – Have to split the data points based on its position
- Value – Have to split the data points based on its Y value
- Percentage – Have to split the points based on the percentage value
- Indexes – The data points with the specified indexes are split separately

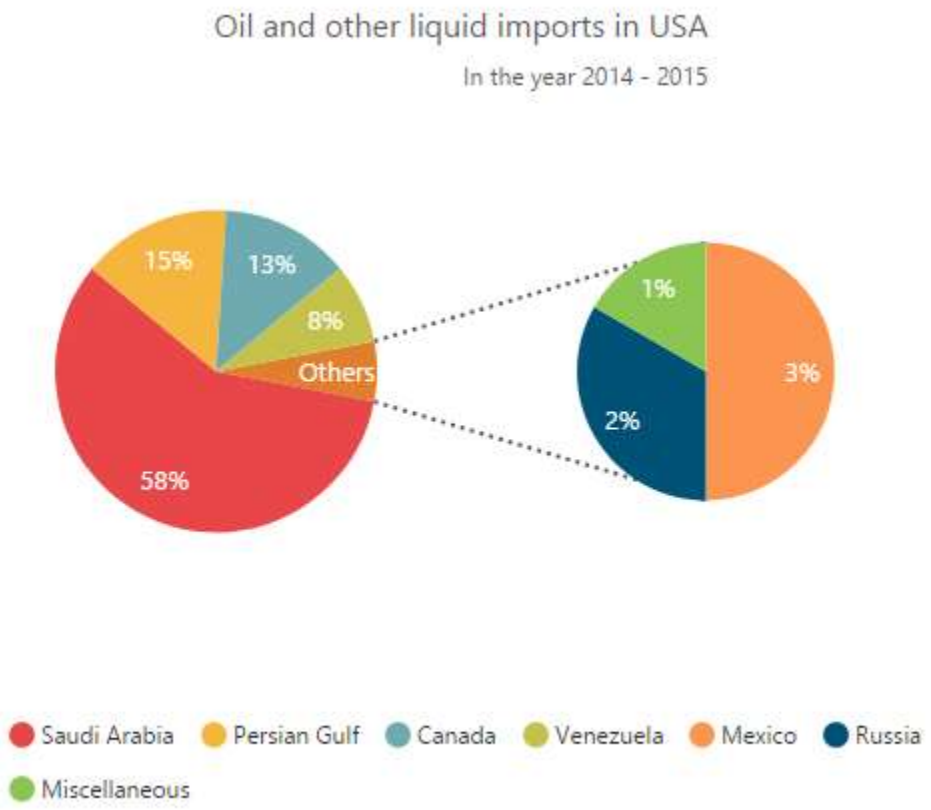
By default, the splitMode is set to **Value**.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
.SplitMode(SplitMode.Position)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.SplitValue("3").Add();
})
//...
)
```



Changing Pie Of Pie Size

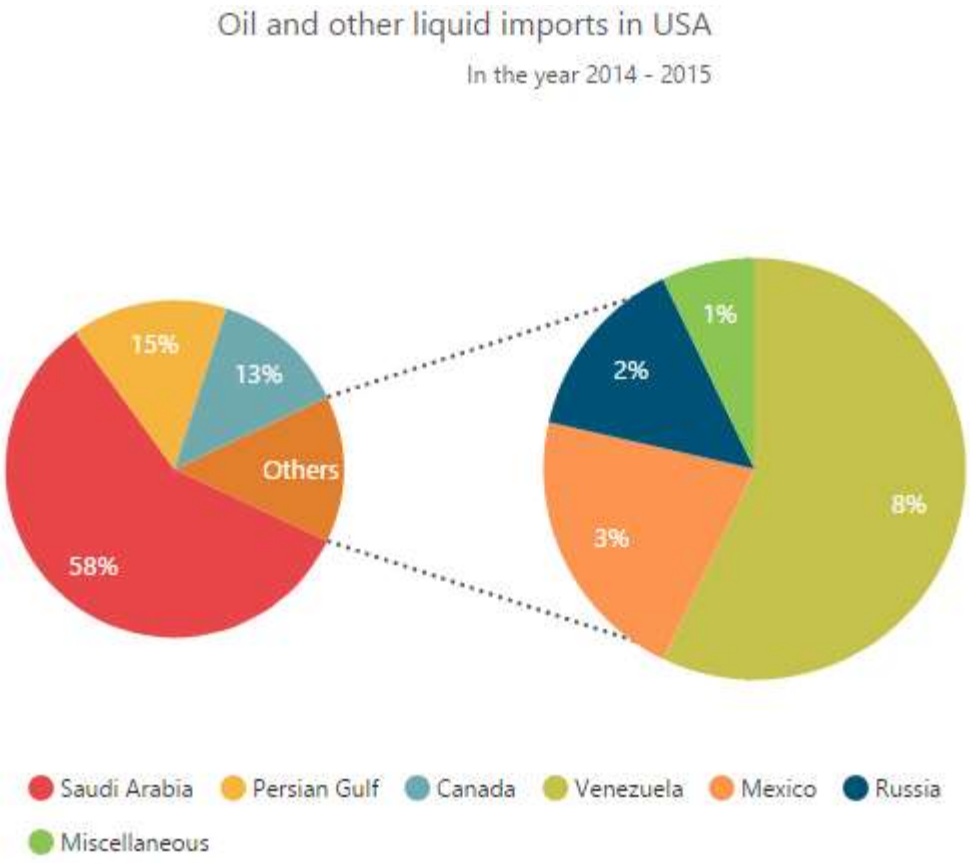
The size of the second Pie can be customized by using the PieOfPieCoefficient property. The default value of pieOfPieCoefficient is **0.6**.Its value ranges from 0 to 1.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
.PieOfPieCoefficient("1").Add();
})
//...
)
```

The following screenshot represents the pie of pie series with pieOfPieCoefficient as 1

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing the Gap

The distance between the two pies in the pie of pie chart can be controlled by using the GapWidth property. The default value is **50**.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.Series(sr =>
{
.GapWidth("150").Add();
})
//...
)
```

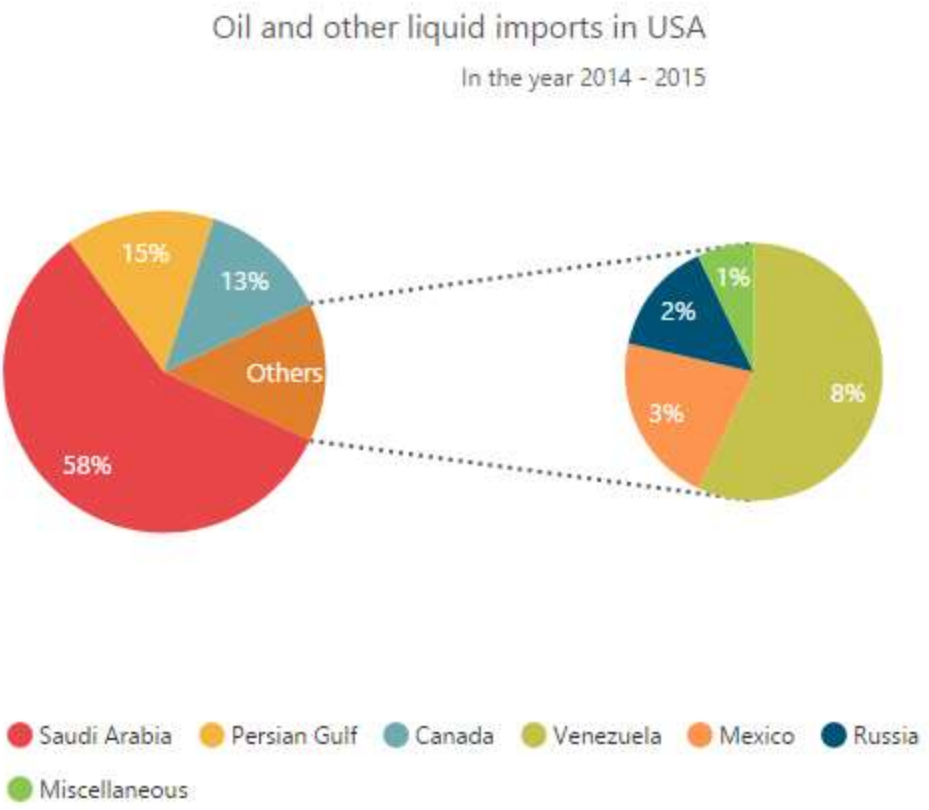


Chart Series

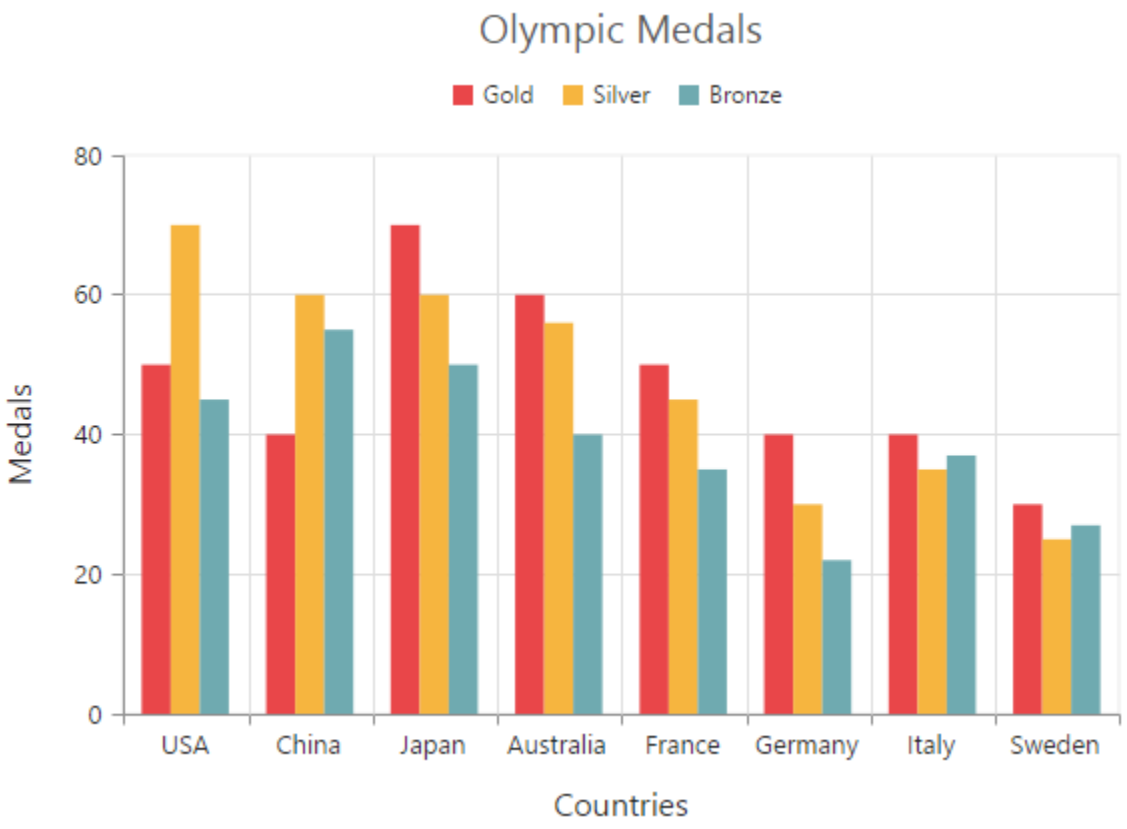
Multiple Series

In EjChart, you can add multiple series object in the **Series** options. The series are rendered in the order it is added to the Series option, by default. You can change this order by using the **ZOrder** option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Adding Multiple Series
.Series(sr =>
{
// Add first series
sr.Type(SeriesType.Column).Points(pt=>{
pt.X("USA").Y(50).Add();
//...
}).Add();
// Add second series
sr.Type(SeriesType.Column).Points(pt => {
pt.X("USA").Y(70).Add();
//...
}).Add();
// Add third series
sr.Type(SeriesType.Column).Points(pt => {
pt.X("USA").Y(45).Add();
//...
```

```
}).Add();
})
)
```



Click here to view the multiple series online demo sample.

Customizing all series together

By using the **CommonSeriesOptions**, you can customize the series options for all the series commonly, instead of setting the options directly on each series object.

Note: The inline properties of the series has the first priority and override the commonSeriesOptions.

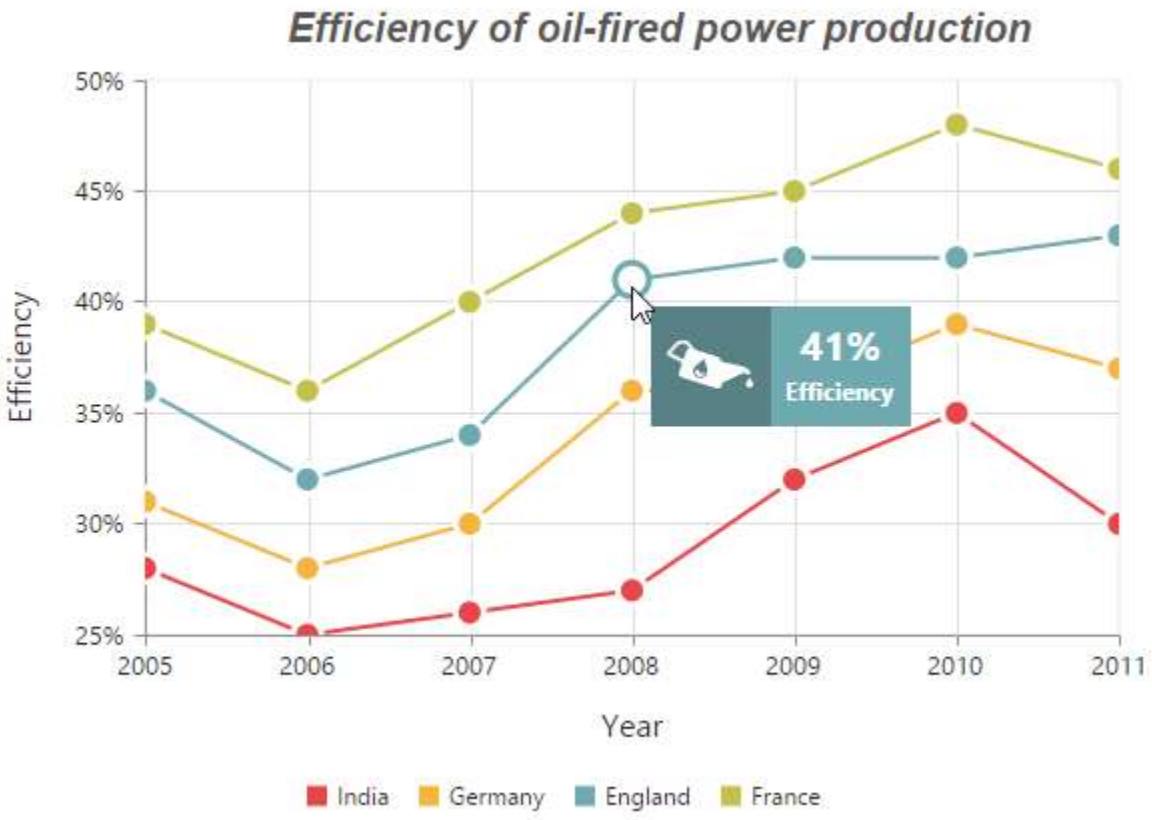
The following code example explains on how to enable marker, tooltip and animation for the chart series by using the commonSeriesOptions.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Common Properties for all the series
.CommonSeriesOptions(options=>options.Type(SeriesType.Line).EnableAnimation(
true)
.Border(border=>border.Width(2))
.Marker(marker=>marker.Visible(true).Shape(ChartShape.Circle).Size(size=>siz
e.Height(10).Width(10)))
.Tooltip(tooltip=>tooltip.Visible(true).Template("Tooltip")))
//Adding Multiple Series
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Series(sr =>
{
// Add first series
sr.Points(pt=>{
pt.X("USA").Y(50).Add();
//...
}).Add();
// Add second series
sr.Points(pt => {
pt.X("USA").Y(70).Add();
//...
}).Add();
// Add third series
sr.Points(pt => {
pt.X("USA").Y(45).Add();
//...
}).Add();
})
)
```



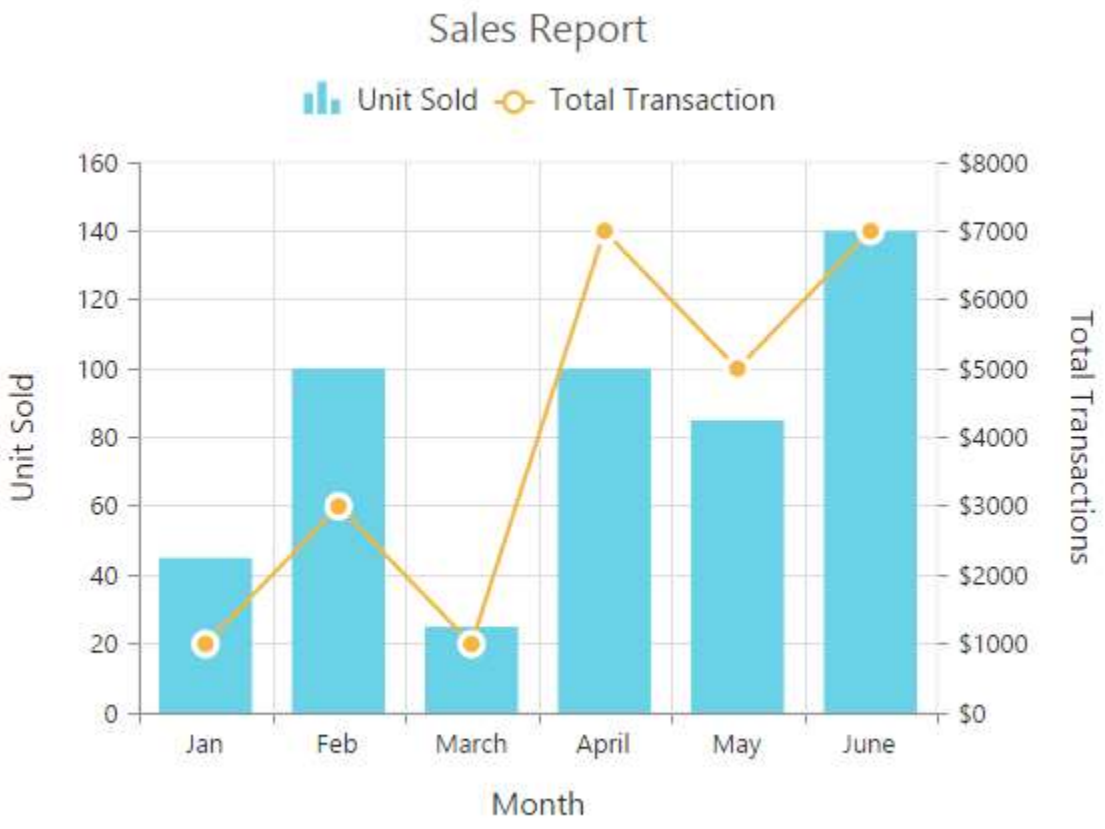
Combination Series

EjChart allows you to render the combination of different series in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
```

```
// ...
//Adding Multiple Series
.Series(sr =>
{
// Add first series
sr.Type(SeriesType.Column).Points(pt=>{
pt.X("USA").Y(50).Add();
//...
}).Add();
// Add second series
sr.Type(SeriesType.Line).Points(pt => {
pt.X("USA").Y(70).Add();
//...
}).Add();
})
)
```



Click here to view the combination series online demo sample.

Limitation of combination chart

- Bar, StackingBar, and StackingBar100 cannot be combined with the other Cartesian type series.
- Cartesian type series cannot be combined with the accumulation series Pie, Doughnut, Funnel, and Pyramid.
- Polar and Radar series cannot be combined with the accumulation and Cartesian type series.

Copyright © 2001 - 2020 Syncfusion Inc.

When the combination of Cartesian and accumulation series types are added to the series option, the series that are similar to the first series are rendered and other series are ignored. The following code example illustrates this,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(sr =>
{
// Add line series
sr.Type(SeriesType.Line).Points(pt=>{
pt.X("Jan").Y(50).Add();
//...
}).Add();
// Add pie series
sr.Type(SeriesType.Pie).Points(pt => {
pt.X("Jan").Y(70).Add();
//...
}).Add();
})
)
```



Data Markers

Data markers are used to provide information about the data point to the user. You can add a shape and label to adorn each data point.

Copyright © 2001 - 2020 Syncfusion Inc.

Add Shapes

You can add shapes to any chart types but they are often used with line, area and spline series to indicate each data point. It is highlighted when you hover the mouse on the shape.

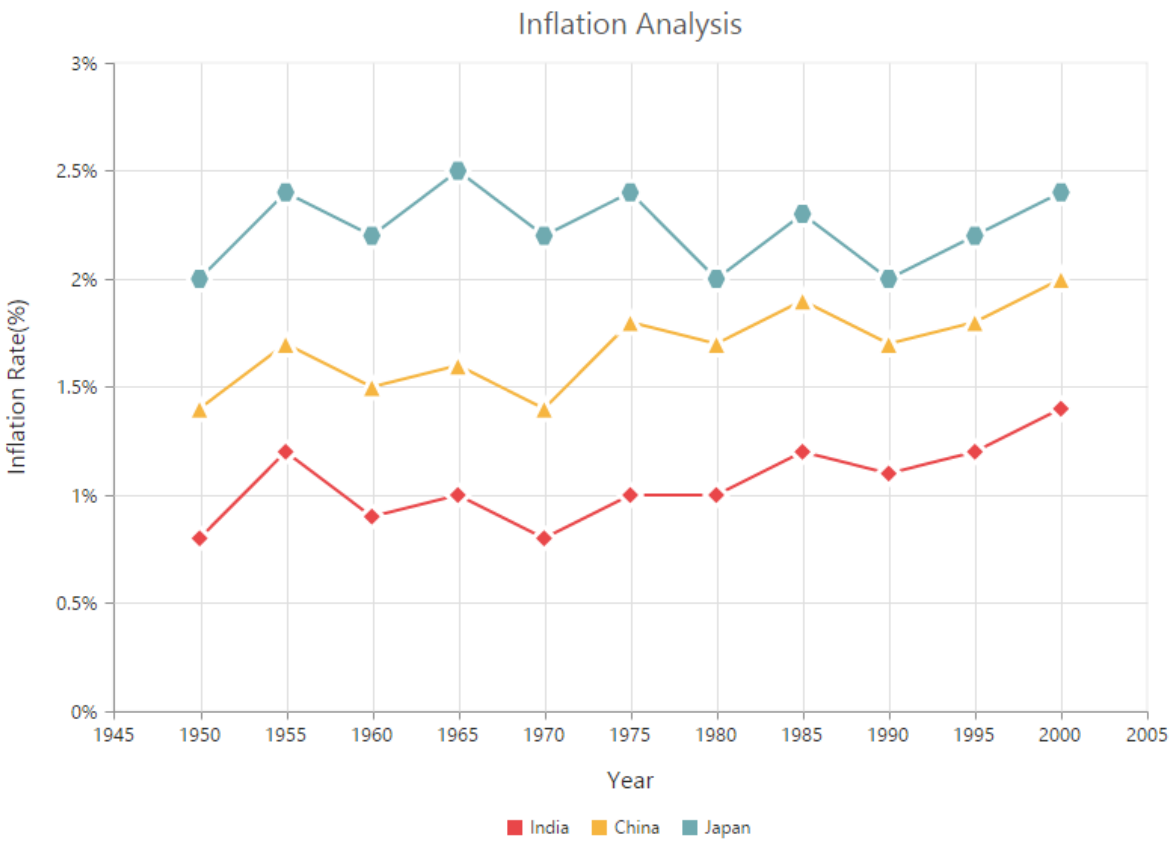
Shapes can be added to the chart by enabling the Visible option of the **Marker** property. There are different shapes you can add to the chart by using the **Shape** option such as Rectangle, Circle, Diamond etc.

The following code example explains on how to enable series marker and add shapes,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
//...
.Marker(mr=>mr.Visible(true).Shape(ChartShape.Diamond)).Add();
//Adding shapes to series2
sr
//...
.Marker(mr => mr.Visible(true).Shape(ChartShape.Triangle)).Add();
//Adding shapes to series3
sr
//...
.Marker(mr => mr.Visible(true).Shape(ChartShape.Hexagon)).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Add image as marker

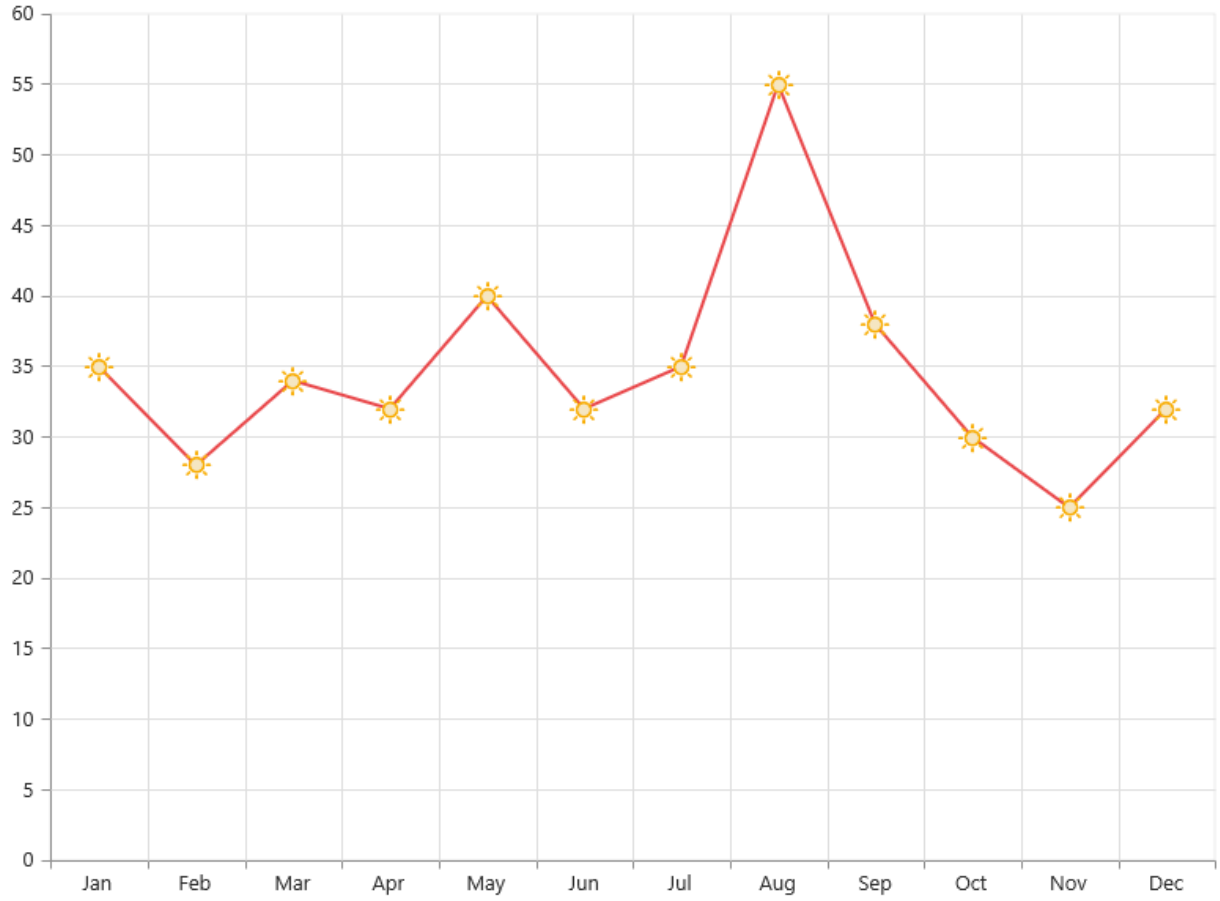
Apart from the shapes, you can also add images to mark the data point by using the **ImageUrl** option.

The following code example illustrates this,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
//..
.Marker(mr=>mr.Visible(true).Shape(ChartShape.Image)
.Size(size => size.Height(20).Width(20)).ImageUrl("sun_annotation.png")
).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Add labels

Data label can be added to a chart series by enabling the *Visible* property in the **DataLabel** option. The labels appear at the top of the data point, by default.

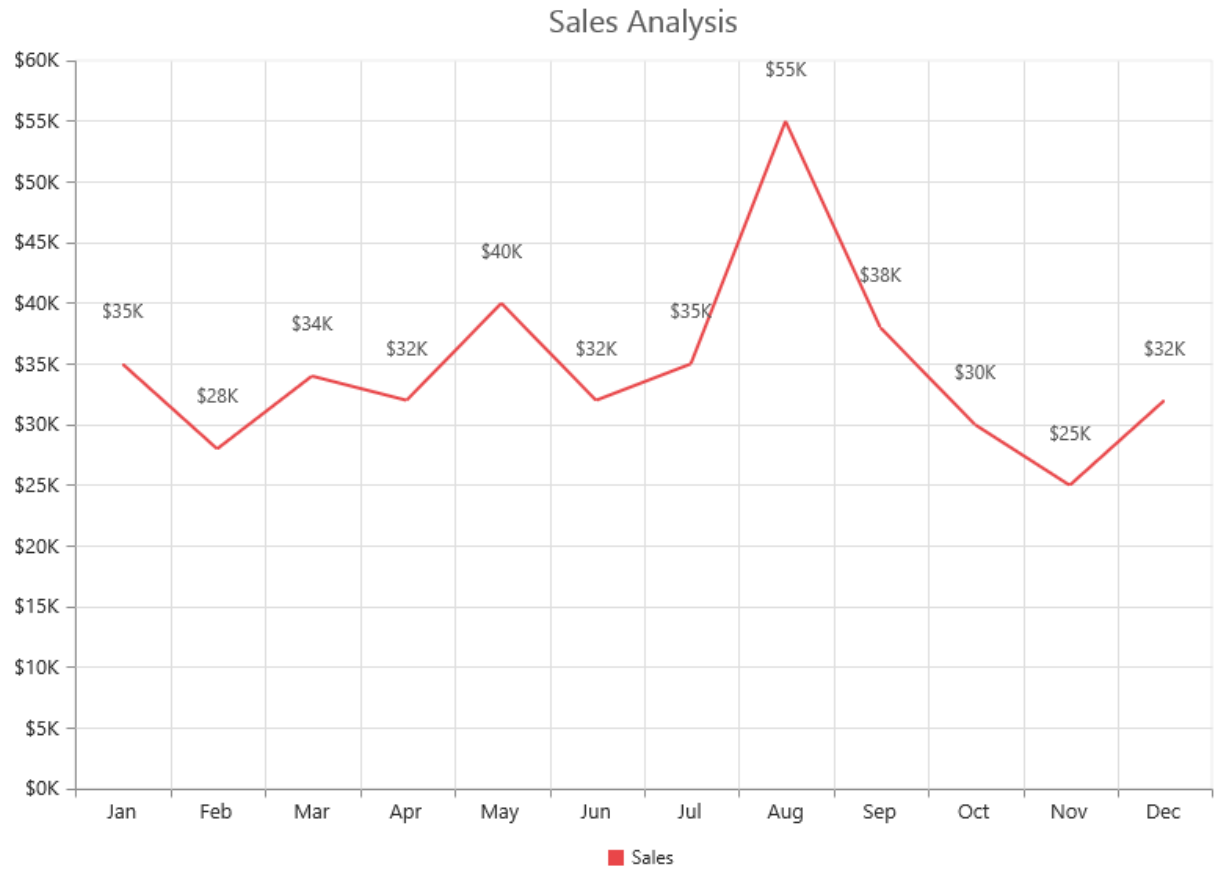
The following code example shows how to enable data label and set its horizontal and vertical text alignment.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
//..
.Marker(mr=>mr
.DataLabel(label=>label
//Set text alignment to data label text
.Visible(true)
.HorizontalTextAlignment(Syncfusion.JavaScript.DataVisualization.TextAlignme
nt.Center)
.VerticalTextAlignment(Syncfusion.JavaScript.DataVisualization.TextAlignment
.Far))
).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
//...
)
```



Label content can be formatted by using the template option. Inside the template, you can add the placeholder text *"point.x"* and *"point.y"* to display corresponding data points x & y value.

You can adorn the labels with background shapes by setting *Shape* option.

The following code example shows how to add background shapes and set template to data label.

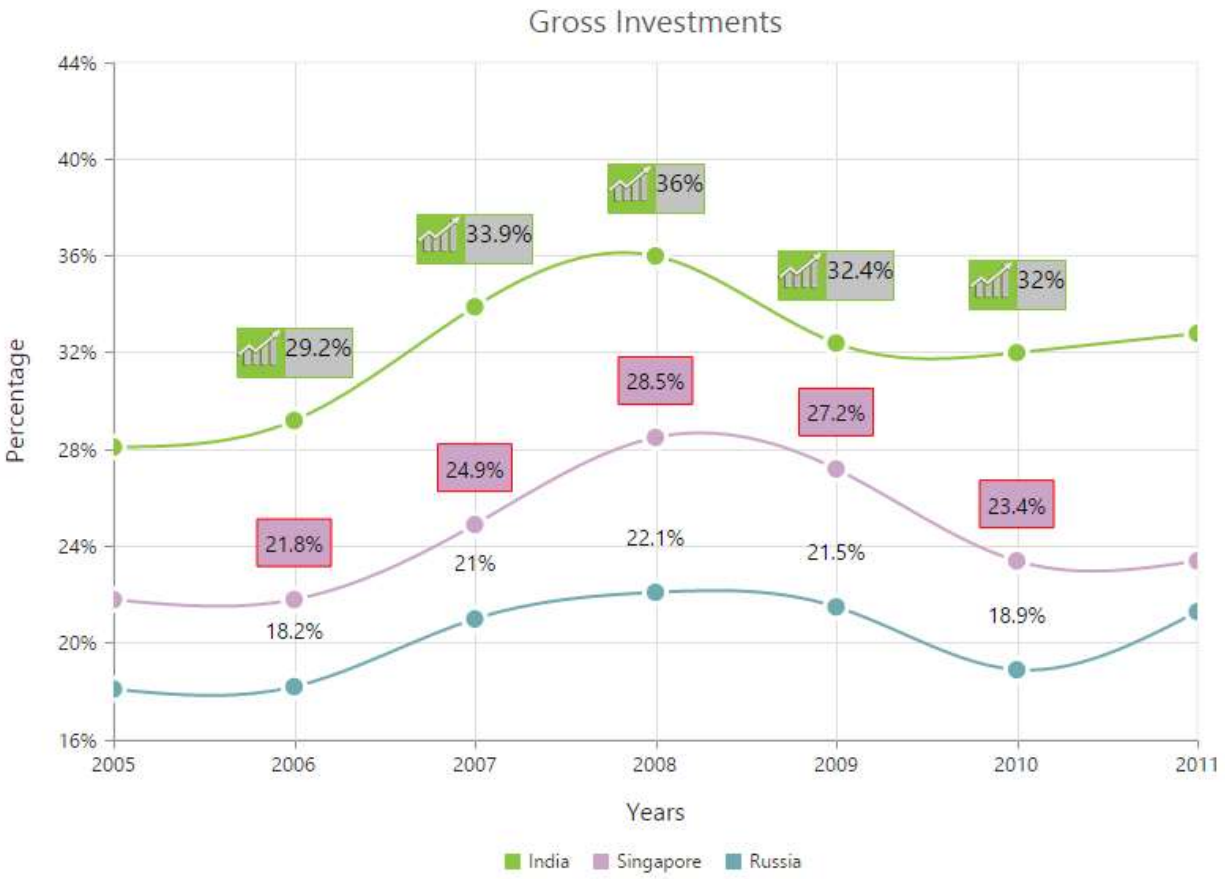
ASPX-CS

```
<div id="template">
<div id="left">
<img src="../images/chart/icon_investments.png"/>
</div>
<div id="right">
<div id="point">#point.y#%</div>
</div>
</div>
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//..
.Marker(mr=>mr
.DataLabel(label=>label
.Visible(true)
//Set template to data label
.Template("template"))
).Add();
//Adding shapes to series2
sr
//..
.Marker(mr=>mr
.DataLabel(label=>label
.Visible(true)
//Add background shape to the data label
.Shape(ChartShape.Rectangle).Border(br=>br.Width(1).Color("red"))
)
).Add();
//Adding shapes to series3
sr
//..
.Marker(mr=>mr
.DataLabel(label=>label
.Visible(true))
).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

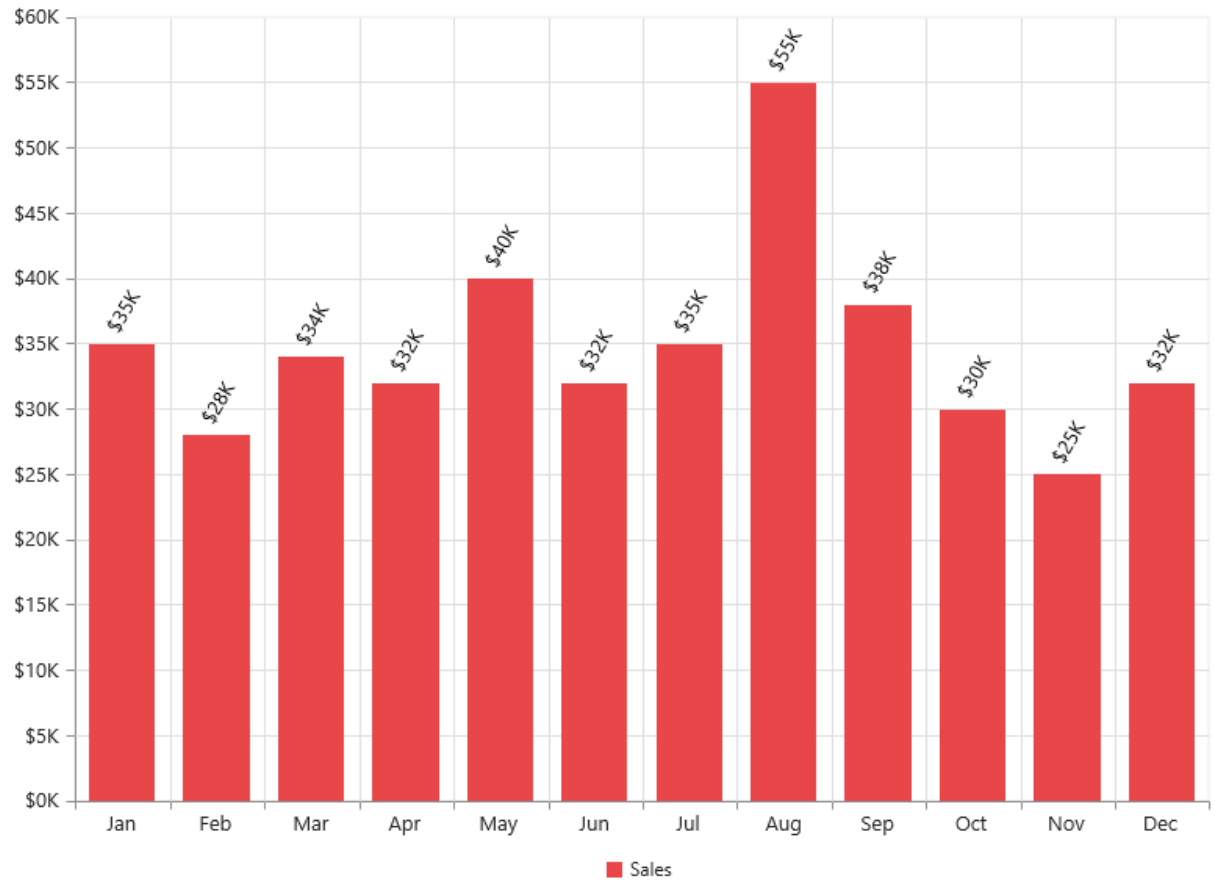


The appearance of the labels can be customized by using the *Font* and *Offset* options. The Offset option is used to move the labels vertically. Also, labels can be rotated by using the *Rotate* option.

The following code example shows how to rotate data label text and customize the font.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
//..
.Marker(mr=>mr
.DataLabel(label=>label
.Visible(true)
//Rotate data label and customize the font
.Angle(300)
.Offset(15)
.Font(font=>font.Color("black").Size("13px"))
)
).Add();
})
//...
)
```



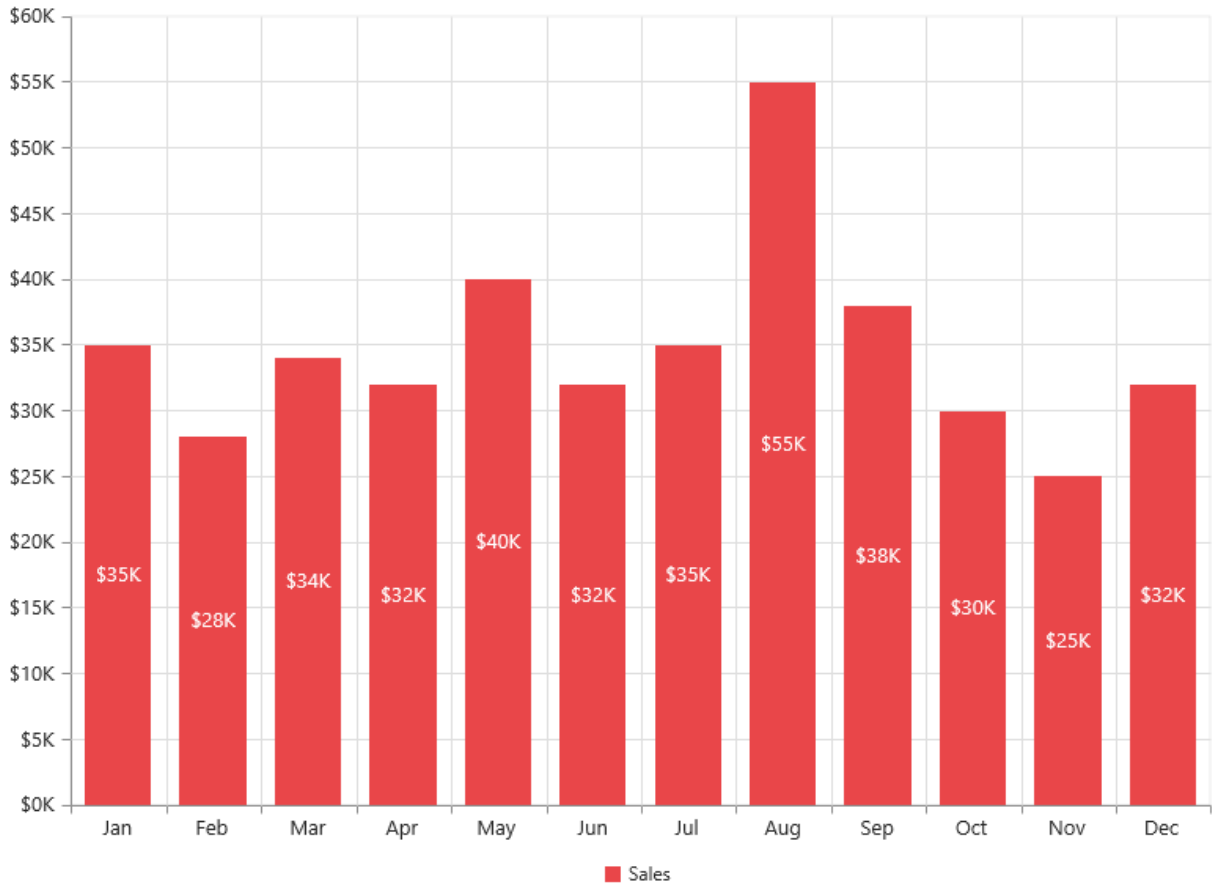
You can position the label to the top, center or bottom position of the segment by using the **TextPosition** option for the chart types such as Column, Bar, Stacked bar, Stacked column, 100% Stacked bar, 100% Stacked column, Candle and OHLC.

The following code example shows how to set textPosition to display data label in the middle of the column rectangle.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
//..
.Marker(mr=>mr
.DataLabel(label=>label
.Visible(true)
// Place the data label text position in the center of the rectangle
.TextPosition(TextPosition.Middle)
)
).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



The label can be positioned inside or outside the perimeter of the series by using the **LabelPosition** option for the chart types such as Pie and Doughnut, .

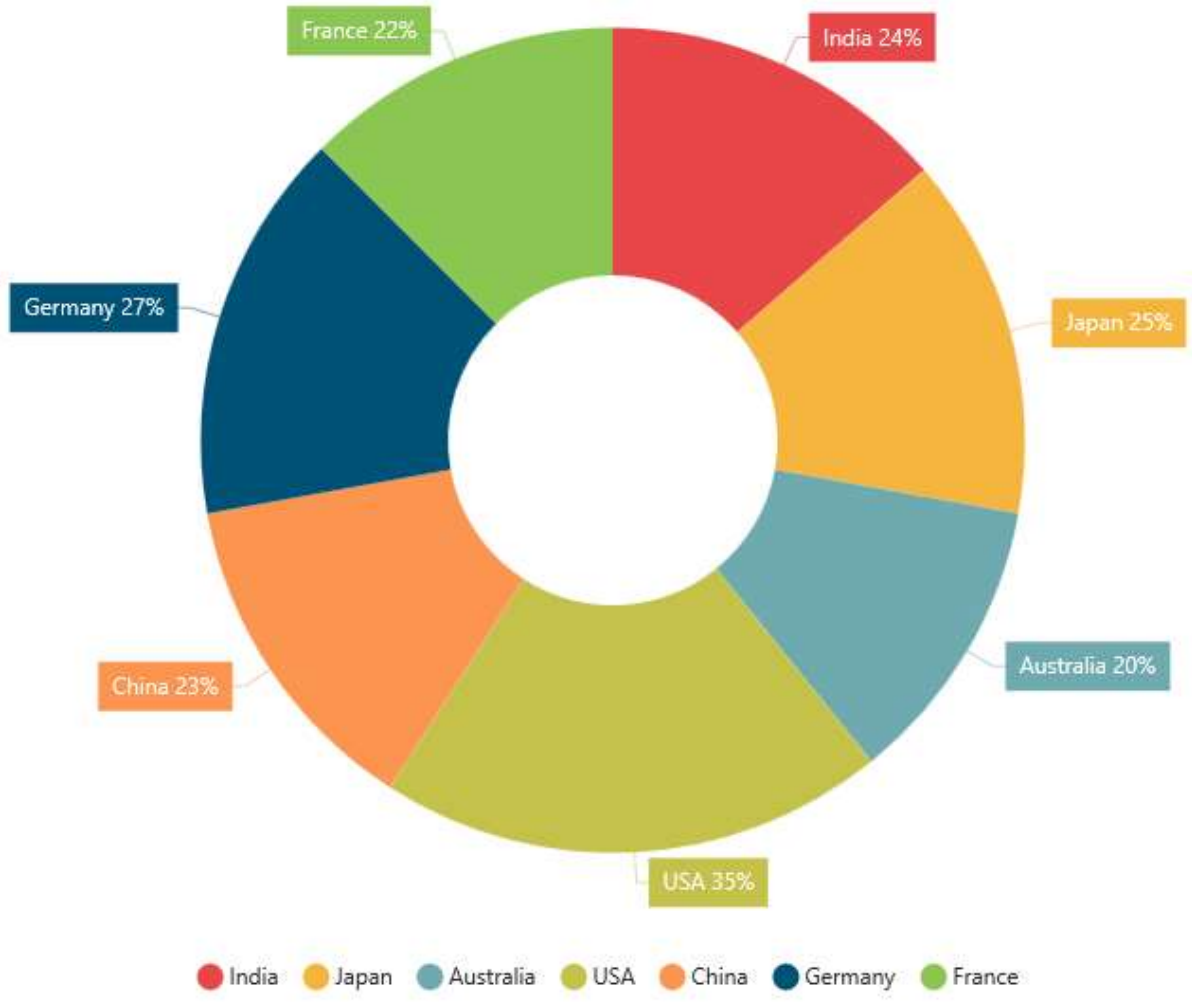
The following code example shows how to set the LabelPosition,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
.Points(pt=>{
pt.X("India").Y(24).Text("India 24%").Add();
pt.X("Japan").Y(25).Text("Japan 25%").Add();
pt.X("Australia").Y(20).Text("Australia 20%").Add();
pt.X("USA").Y(35).Text("USA 35%").Add();
pt.X("China").Y(23).Text("China 23%").Add();
pt.X("Germany").Y(27).Text("Germany 27%").Add();
pt.X("France").Y(22).Text("France 22%").Add();
})
.Marker(mr=>mr
.DataLabel(label=>label
.Visible(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

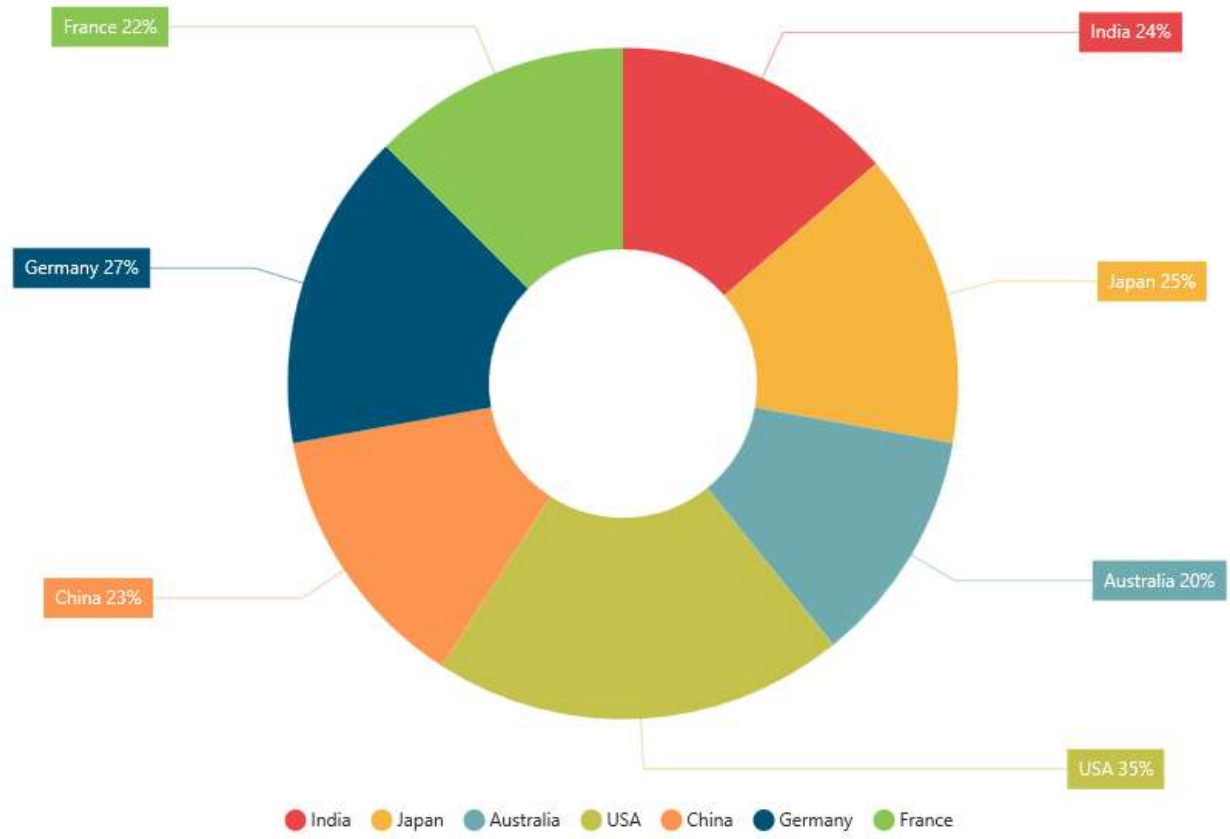
```
.Shape(ChartShape.Rectangle)
.Font(font=>font.Color("white"))
)
)
//Display data label outside position
.LabelPosition(ChartLabelPosition.Outside)
.Type(SeriesType.Doughnut).Add();
})
//...
)
```



The following screenshot displays the labels when the LabelPosition is set as *Inside* position.

The following screenshot displays the labels when the LabelPosition is set as *OutsideExtended* position.

Copyright © 2001 - 2020 Syncfusion Inc.

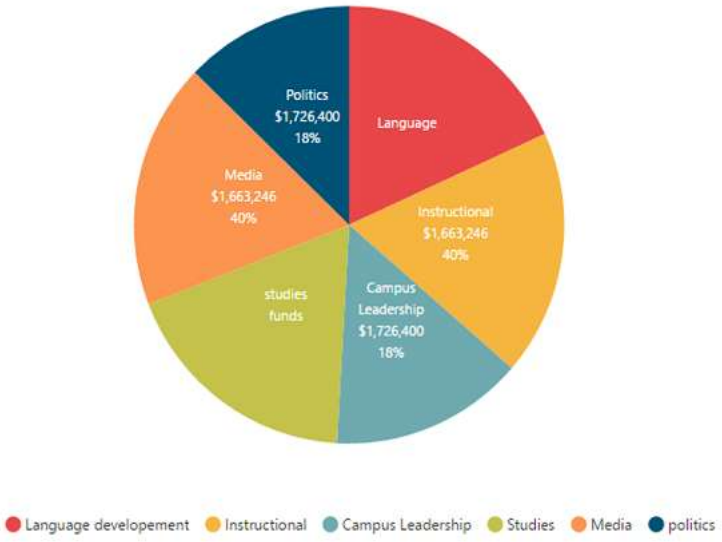


The label can be wrapped for pie, doughnut, funnel, and pyramid series by setting the enableWrap property.

JAVASCRIPT

```
@(Html.EJ().Chart("chartContainer")
//.... . .
.Series(sr => {
//Adding series
sr.Marker(mr => mr.DataLabel(label=>
label.Visible("true")
.EnableWrap("true")
.MaximumLabelWidth(32)
))
//.... . .
}).Add();
//.... . .
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Binding label from the datasource

You can bind the text value to the data label from the datasource and then you need to map the text value field with the **TextMappingName** property respectively.

ASPX-CS

```
//data source for chart with label
var chartData = [
{ month: 'Jan', sales: 35 , text: 'January'},
{ month: 'Feb', sales: 28 , text: 'February'}
//..
];
function onChartLoad(sender) {
var data = GetData();
sender.model.series[0].dataSource = chartData;
sender.model.series[0].xName = "month";
sender.model.series[0].yName = "sales";
}
@(Html.EJ().Chart("chartContainer")
//.... . .
.Series(sr => {
sr.Marker(mr => mr.DataLabel(label=>
label.Visible("true")
.TextMappingName("text")
))
.Add();
//.... . .
})
.Load("onChartLoad")
//....
})
)
```

Binding fill color to the points from the datasource

You can bind the color value to the points from the datasource and then you need to map the color value field to the **PointColorMappingName** property respectively.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
//data source for chart with fill color
var chartData = [
{ month: 'Jan', sales: 35 , color: 'red'},
{ month: 'Feb', sales: 28 , color: 'blue'}
//..
];
function onChartLoad(sender) {
var data = GetData();
sender.model.series[0].dataSource = chartData;
sender.model.series[0].xName = "month";
sender.model.series[0].yName = "sales";
}
@(Html.EJ().Chart("chartContainer")
//.... . .
.Series(sr => {
//Mapping the color values to the points
sr.PointColorMappingName("color") .Add();
//.... . .
})
.Load("onChartLoad")
//....
})
)
```

Customize specific points

By using the ejChart, you can also customize the individual/specific markers with different colors, shapes and also with different images.

There are two ways to achieve this based on how the data is fed to the series.

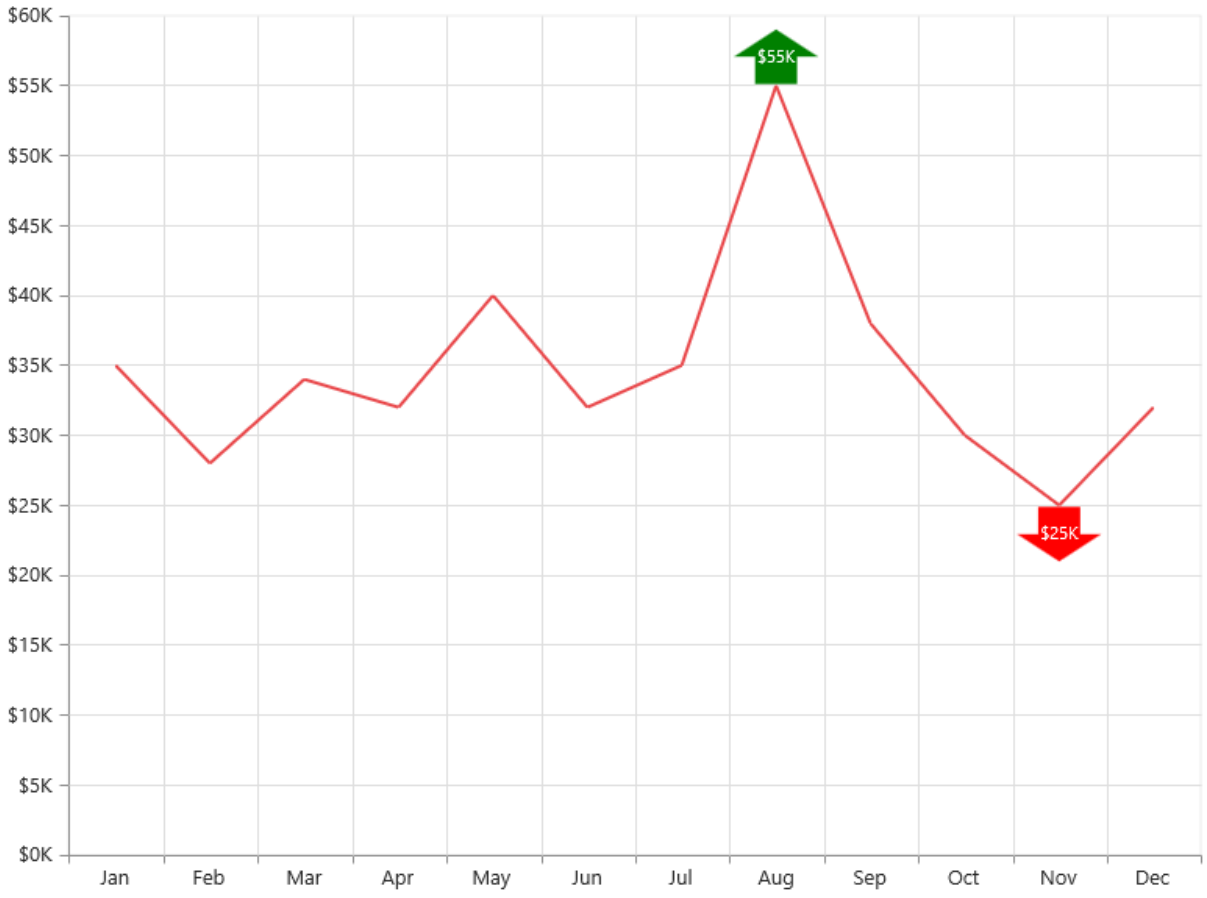
When the data is provided by using the Points option, you can add marker for each data point or specific point by using the Marker option as illustrated in the following code example.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
.Points(pt=>{
pt.X("Jan").Y(35).Add();
pt.X("Feb").Y(28).Add();
pt.X("Mar").Y(34).Add();
pt.X("Apr").Y(32).Add();
pt.X("May").Y(40).Add();
pt.X("Jun").Y(32).Add();
pt.X("Jul").Y(35).Add();
pt.X("Aug").Y(55).Marker(mr=>mr
//Enable and customize the data label for a point
.DataLabel(label=>label.Visible(true).Offset(-10).Shape(ChartShape.UpArrow)
.Font(font=>font.Color("white").Size("11px"))
.Margin(margin=>margin.Right(15).Left(15).Top(10).Bottom(10))
.Fill("green"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
).Add();
pt.X("Sep").Y(38).Add();
pt.X("Oct").Y(30).Add();
pt.X("Nov").Y(25).Marker(mr => mr
//Enable and customize the data label for a point
.DataLabel(label => label.Visible(true).Offset(-
22).Shape(ChartShape.DownArrow)
.VerticalTextAlignment(Syncfusion.JavaScript.DataVisualization.TextAlignment
.Near)
.Font(font => font.Color("white").Size("11px"))
.Margin(margin => margin.Right(15).Left(15).Top(10).Bottom(10))
.Fill("red"))
).Add();
pt.X("Dec").Y(32).Add();
}).Add();
})
//...
)
```



When the data is bound to the series by using the DataSource option, you can customize the points in the **SeriesRendering** event as illustrated in the following code example,

ASPX-CS

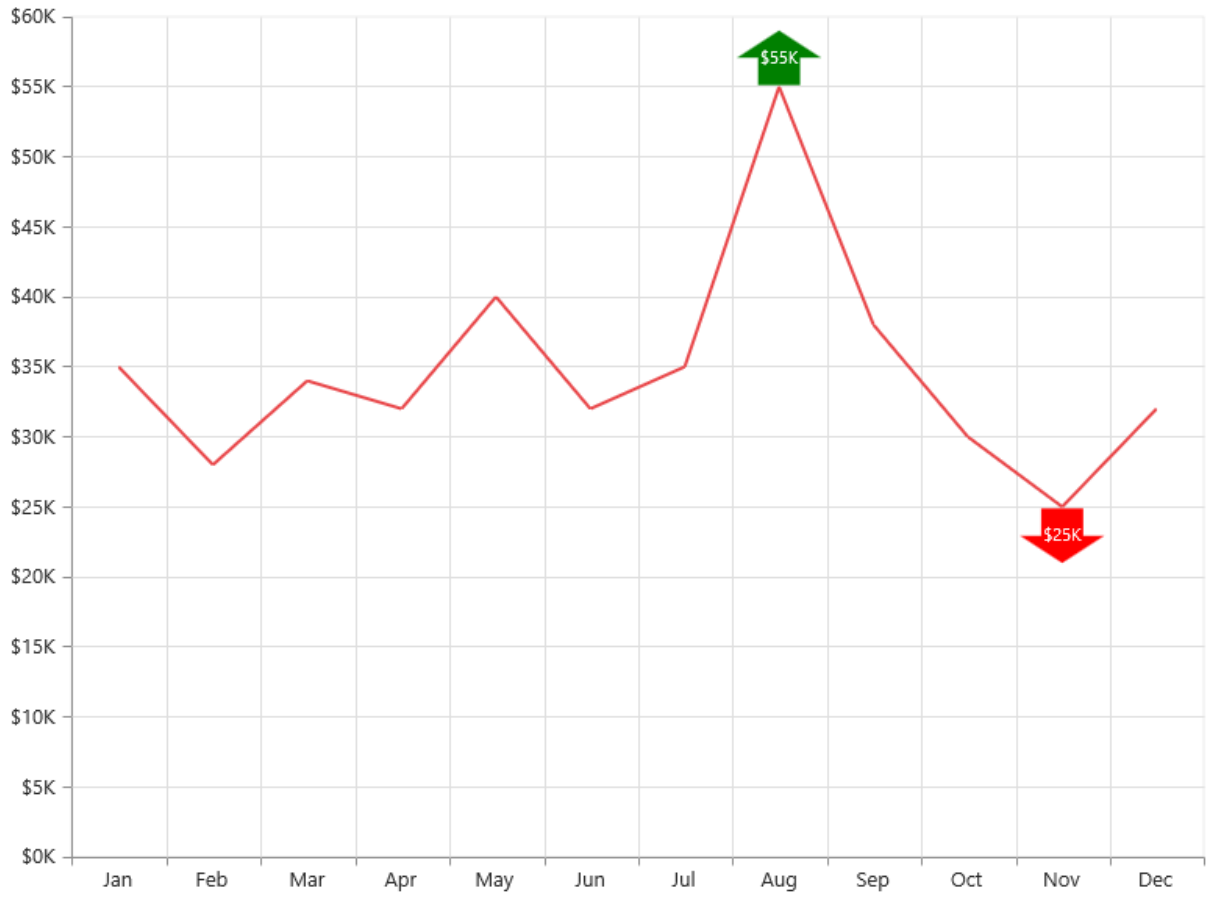
```
@(Html.EJ().Chart("chartContainer")
```

```
//...
.Series(sr =>
{
//Adding shapes to series1
sr
//Add datasource and set xName and yName
.XName("Month").YName("Sales")
.DataSource(ViewBag.ChartData)
.Add();
})
//Fires on rendering the series
.SeriesRendering("onSeriesRender")
//...
)
```

JS

```
//Define the seriesRendering client side event
function onSeriesRender(sender) {
//Enable and customize the dataLabel for a point using event
sender.data.series.points[7].marker = {
dataLabel: {
visible: true,
offset: -10,
shape: "upArrow", font: { color: "white", size: '11px' },
margin: { left: 15, right: 15, top: 10, bottom: 10 },
fill: "green"
}};
sender.data.series.points[10].marker = {
//Enable and customize the dataLabel for a point using event
dataLabel: {
visible: true,
offset: -22,
verticalTextAlignment: 'near',
shape: "downArrow", font: { color: "white", size: '11px' },
margin: { left: 15, right: 15, top: 10, bottom: 10 },
fill: "red"
}};
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Connect Line

This feature is used to connect label and data point by using a line. It can be enabled only for Pie, Doughnut, Pyramid and Funnel chart types. **ConnectorLine** types can be set as *Bezier* or *Line* by using the **Type** option.

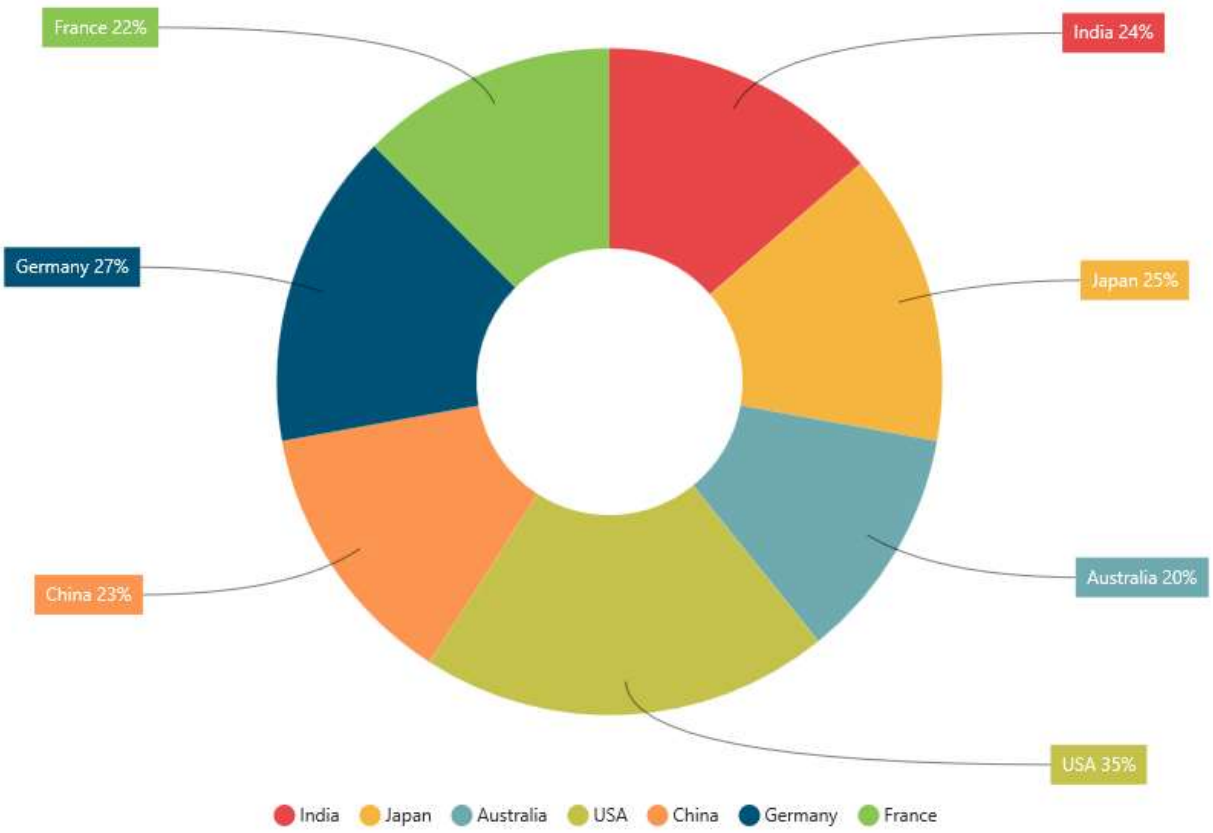
The following code example illustrates this,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
//Adding shapes to series1
sr
.Marker(mr=>mr.DataLabel(label=>label
// Set connector line type and customize the color
.ConnectorLine(cl=>cl.Color("black").Type(ConnectorType.Bezier))
))
.LabelPosition(ChartLabelPosition.OutsideExtended)
.Type(SeriesType.Doughnut)
.Add();
})
//Fires on rendering the series
.SeriesRendering("onSeriesRender")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//...
)
```



Smart labels

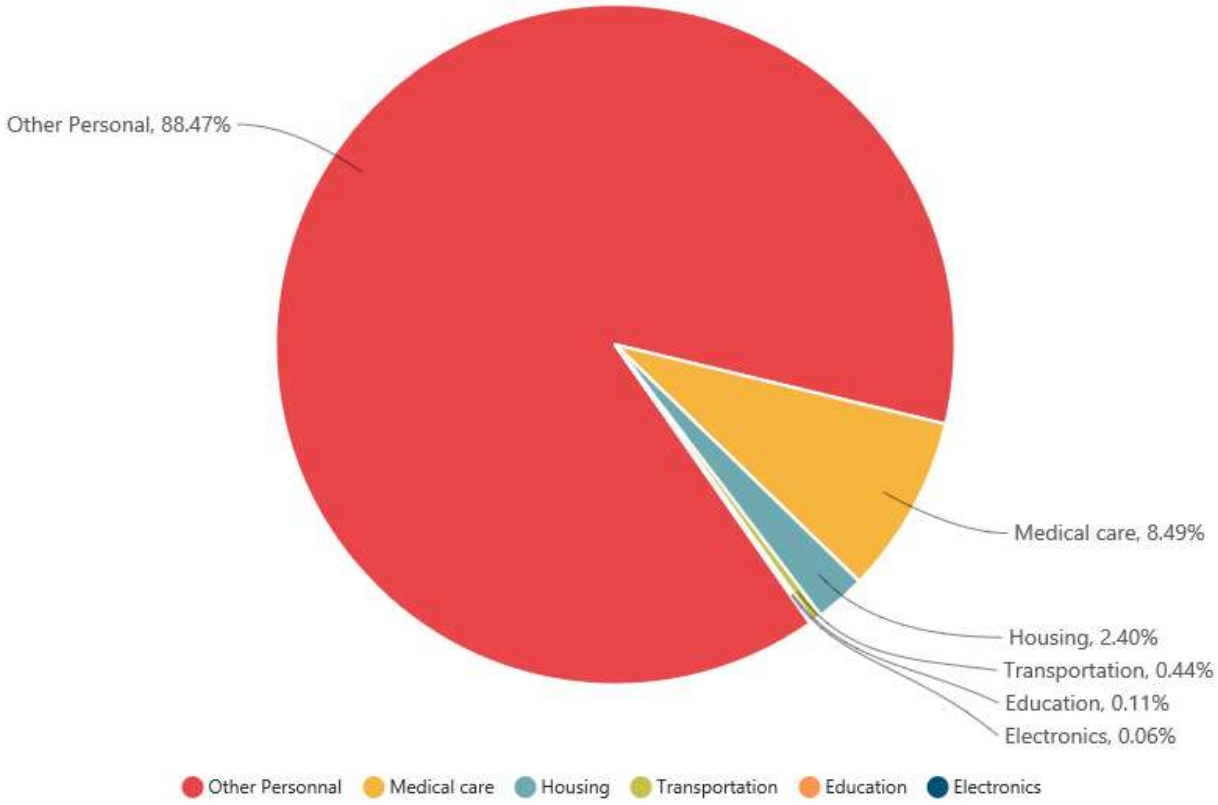
Overlapping of the labels can be avoided by enabling the **EnableSmartLabels** property. The default value is *true* for *Accumulation type series* and *false* for *other series types*.

The following code example shows how to enable smart labels,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
sr
//...
.EnableSmartLabels(true).StartAngle(145)
.Marker(mr =>
{
mr.DataLabel(label =>
{
label.Visible(true).ConnectorLine(cl => cl.Color("black")
.Height(60).Type(ConnectorType.Bezier)).Shape(ChartShape.None)
.Font(font => { font.Size("14px"); });
});
}).Name("Expenses").Type(SeriesType.Pie).Add();
```

```
})
//...
)
```



Click here to view the SmartLabel online demo sample.

Legend

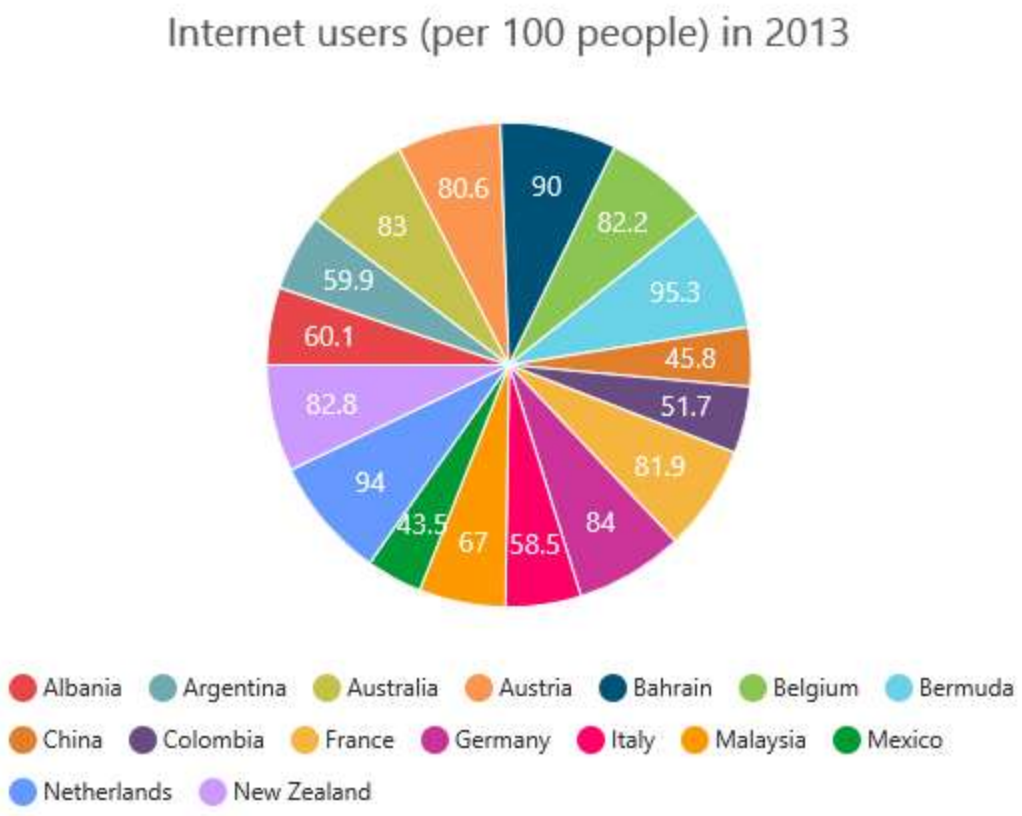
The legend contains the list of Chart Series and Trendlines that appear in a chart.

Legend Visibility

By default, the legend is enabled in the chart. You can enable or disable it by using the *Visible* option of the **Legend**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Visible chart legend
.Legend(lg=>lg.Visible(true))
//...
)
```



Click here to view the online demo sample for legend customization.

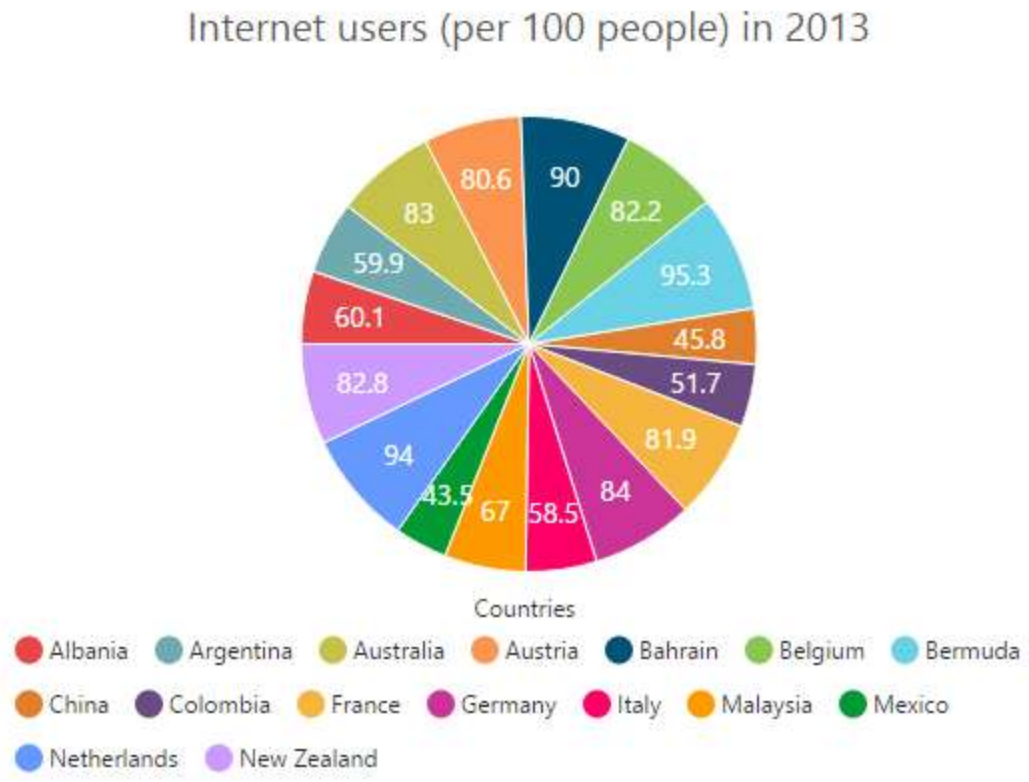
Legend title

To add the title to the legend, you have to specify the **Legend.Title.Text** option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Visible chart legend
.Legend(legend=>legend.Title(title=>
//Add title to the chart legend
title.Text("Countries"))
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



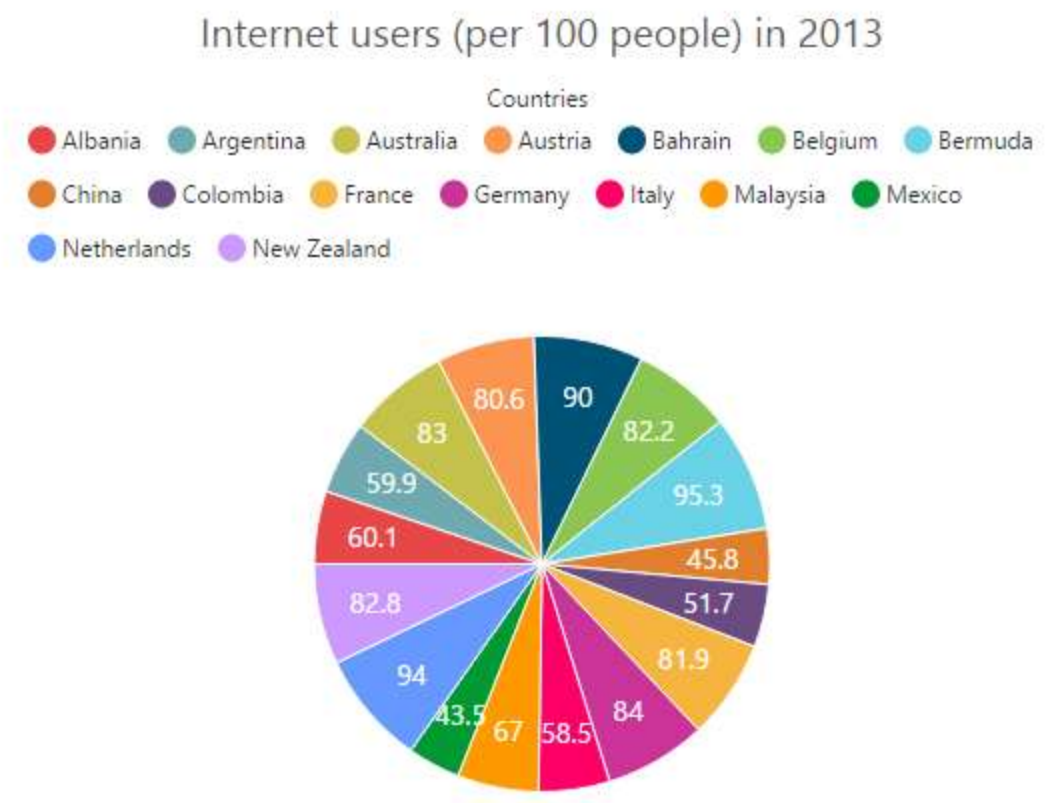
Position and Align the Legend

By using the **Position** option, you can position the legend at *Left*, *Right*, *Top* or *Bottom* of the chart. The legend is positioned at the **Bottom** of the chart, by default.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg.
//...
//Place the legend at top of the chart
Position(LegendPosition.Top)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



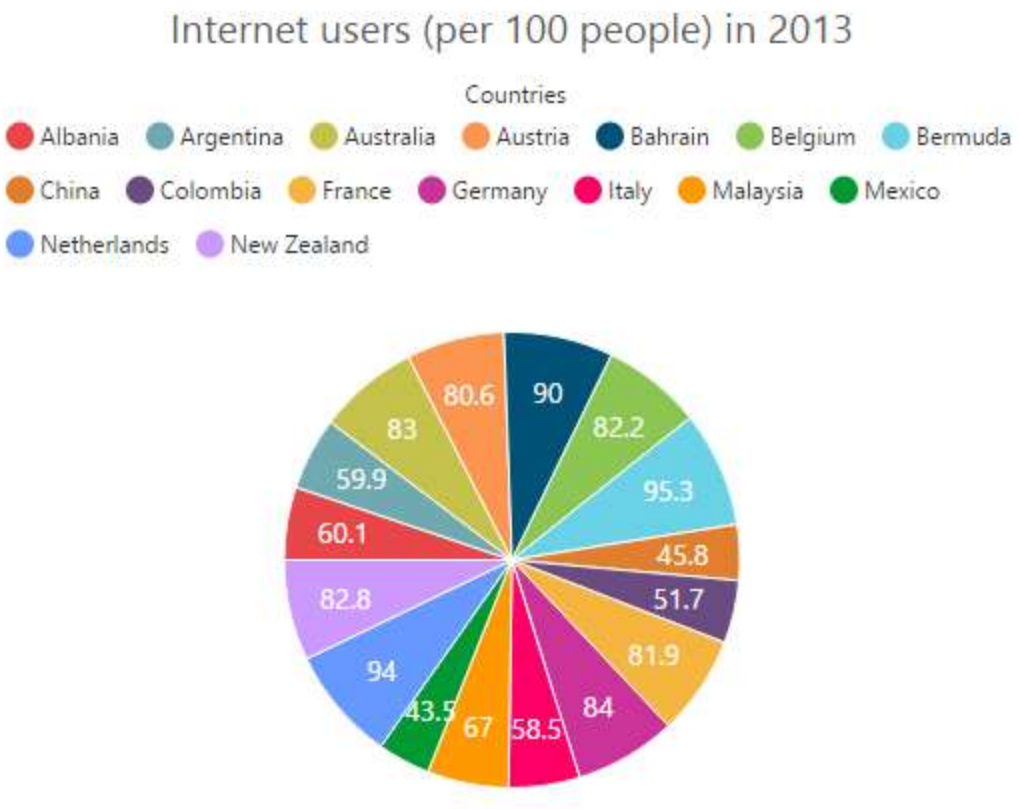
Legend Alignment

You can align the legend to the *Center*, *Far* or *Near* based on its position by using the **Alignment** option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg.
//...
//The below two settings will place the legend at the top-right corner of
the chart.
Position(LegendPosition.Top)
.Alignment(Syncfusion.JavaScript.DataVisualization.TextAlignment.Far)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Arrange legend items in the rows and columns

You can arrange the legend items horizontally and vertically by using the **RowCount** and **ColumnCount** options of the legend.

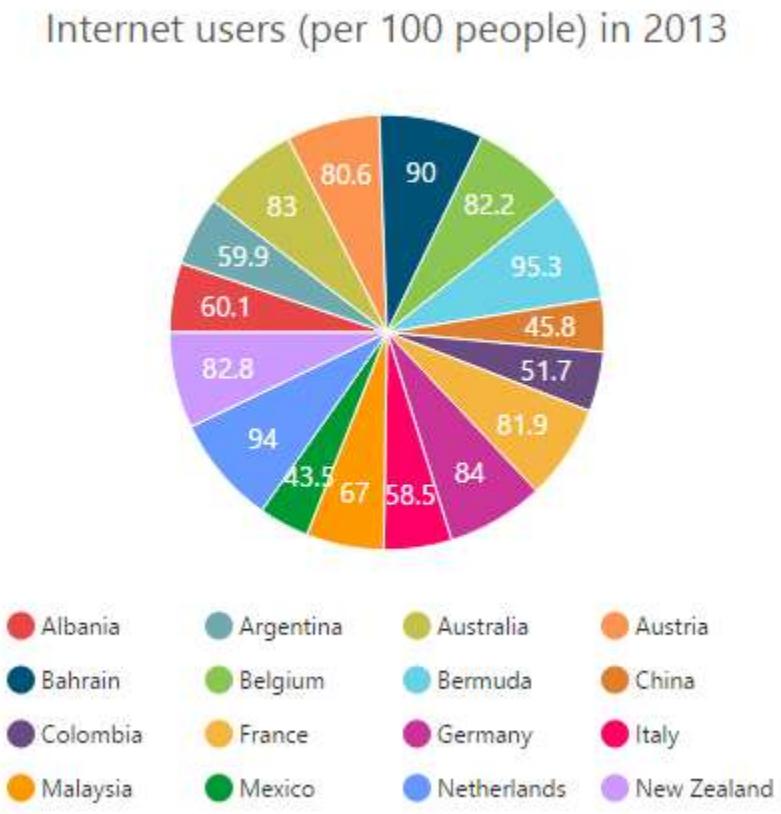
- When only the RowCount is specified, the legend items are arranged according to the RowCount and number of columns may vary based on the number of legend items.
- When only the ColumnCount is specified, the legend items are arranged according to the ColumnCount and number of rows may vary based on the number of legend items.
- When both the options are specified, then the one which has higher value is given preference. For example, when the RowCount is 4 and ColumnCount is 3, legend items are arranged in 4 rows.
- When both the options are specified and have the same value, the preference is given to the ColumnCount when it is positioned at the top/bottom position. The preference is given to the RowCount when it is positioned at the left/right position.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Arrange legend items in 4 rows and approximately 4 columns. Column
couldn't may vary based on number of items.
.RowCount(4)
.ColumnCount(4)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
//...
)
```



Customization

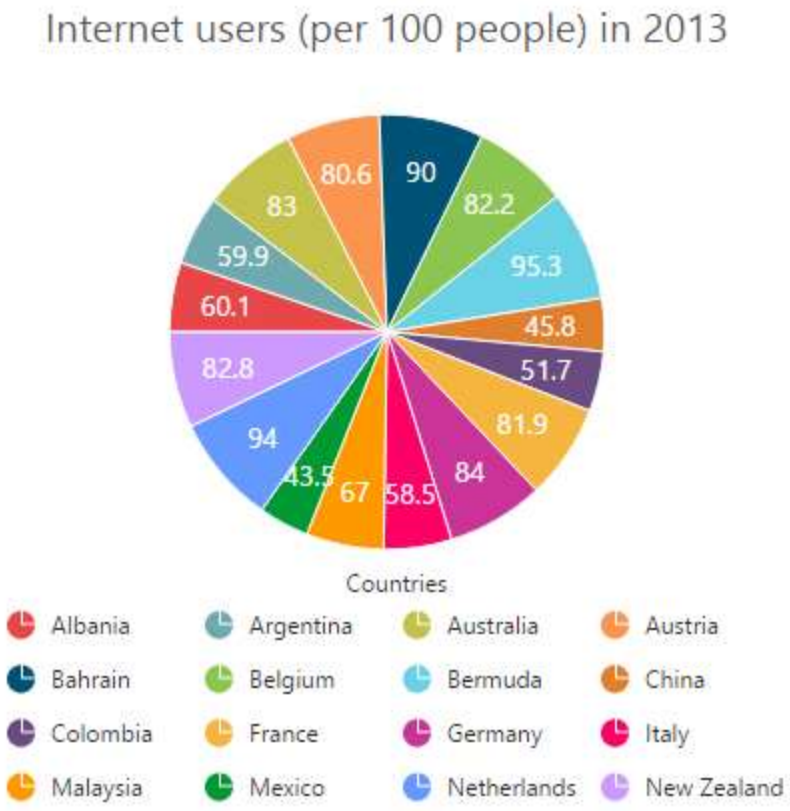
Legend shape

To change the legend icon shape, you have to specify the shape in the **Shape** property of the legend. When you want the legend icon to display the prototype of the series, you have to set the **SeriesType** as shape.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Change legend shape
.Shape(ChartShape.SeriesType)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



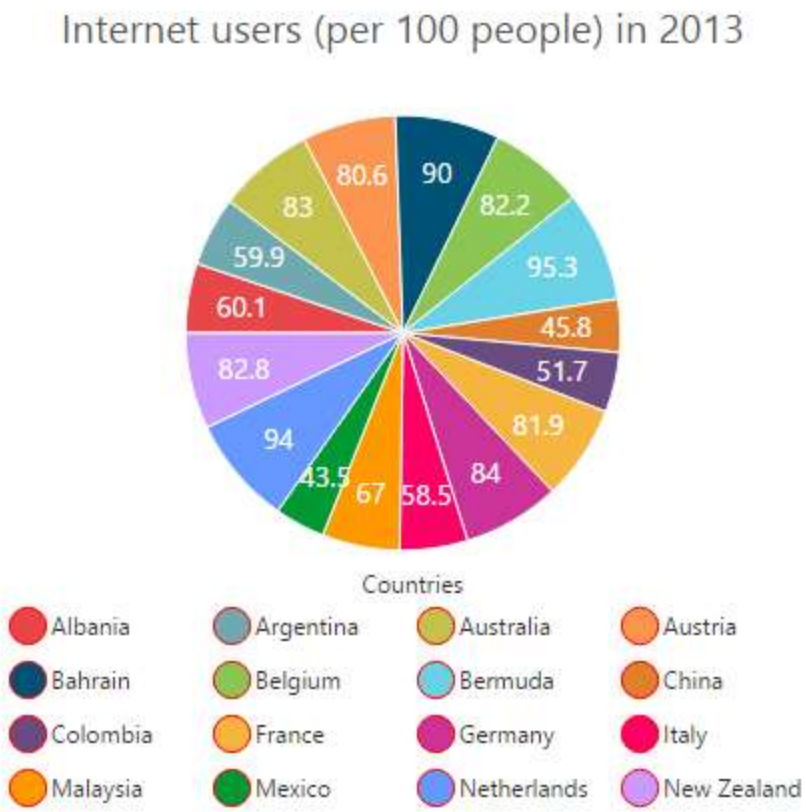
Legend items size and border

You can change the size of the legend items by using the *ItemStyle.Width* and *ItemStyle.Height* options. To change the legend item border, use **Border** option of the legend ItemStyle.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Change legend items border, height and width
.ItemStyle(item=>item.Width(13).Height(13).Border(br=>br.Width(1).Color("#FF
0000")))
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



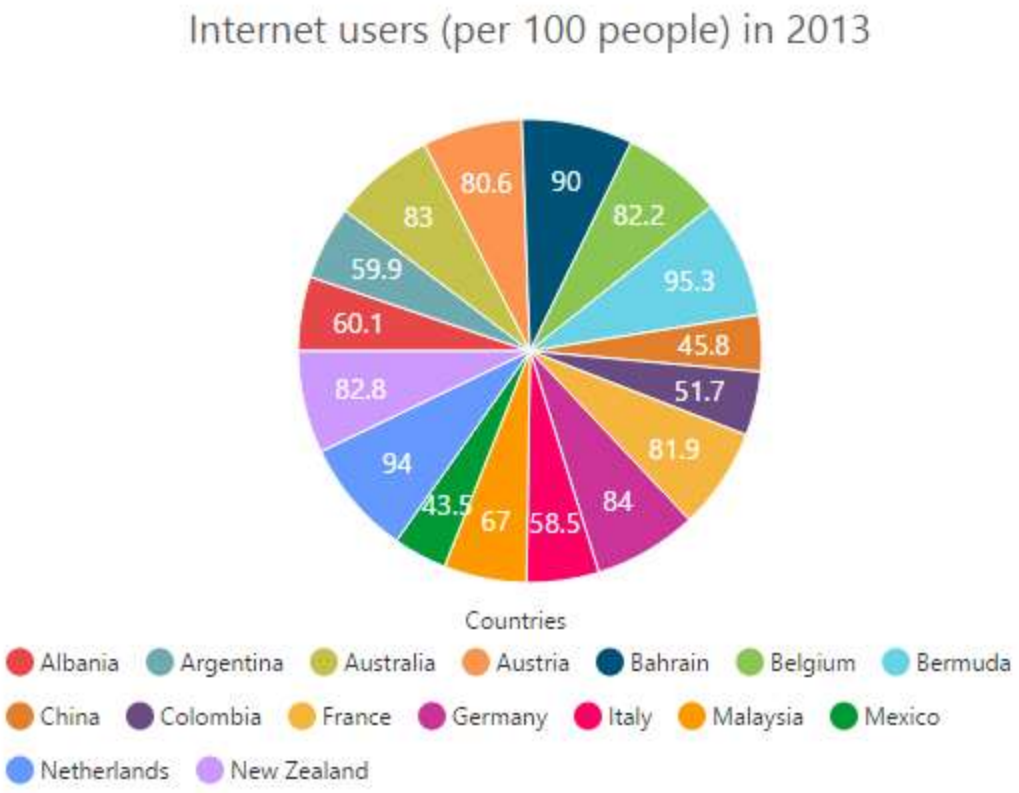
Legend size

By default, legend takes 20% of the **Height** horizontally when it was placed on the top or bottom position and 20% of the **Width** vertically while placing on the left or right position of the chart. You can change this default legend size by using the **Size** option of the legend.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Change legend size
.Size(size=>size.Height("100").Width("550"))
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



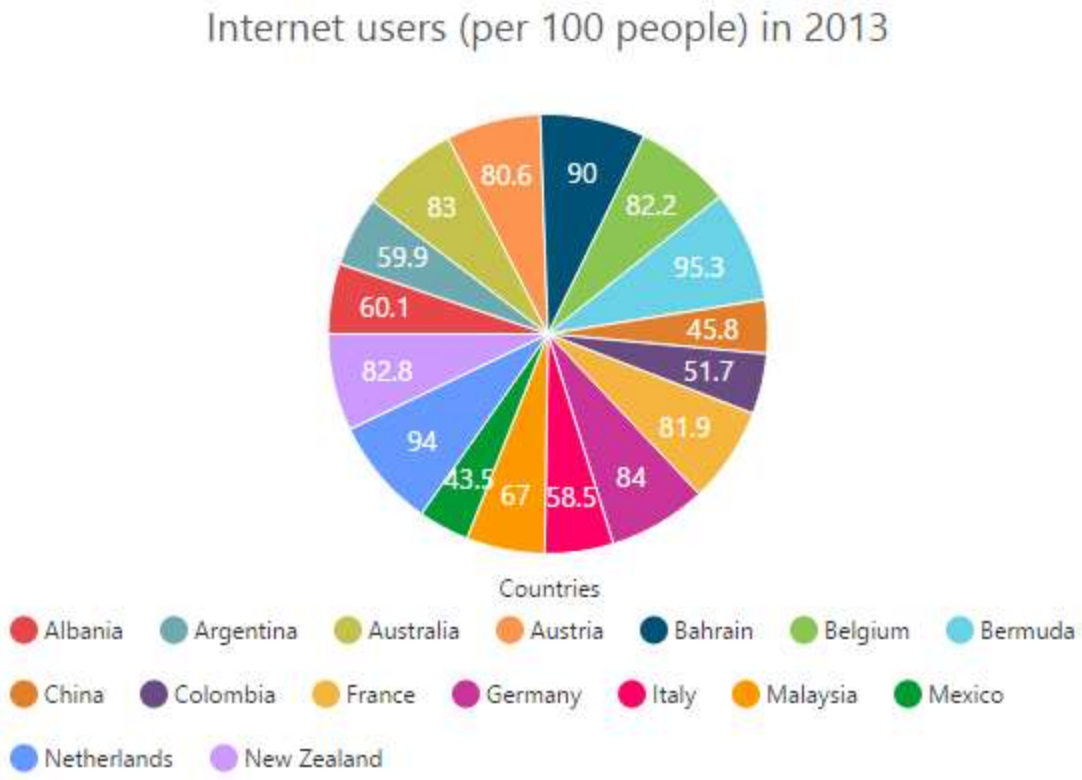
Legend Item Padding

You can control the spacing between the legend items by using the **ItemPadding** option of the legend. The default value is 10.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Add space between each legend item
.ItemPadding(15)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



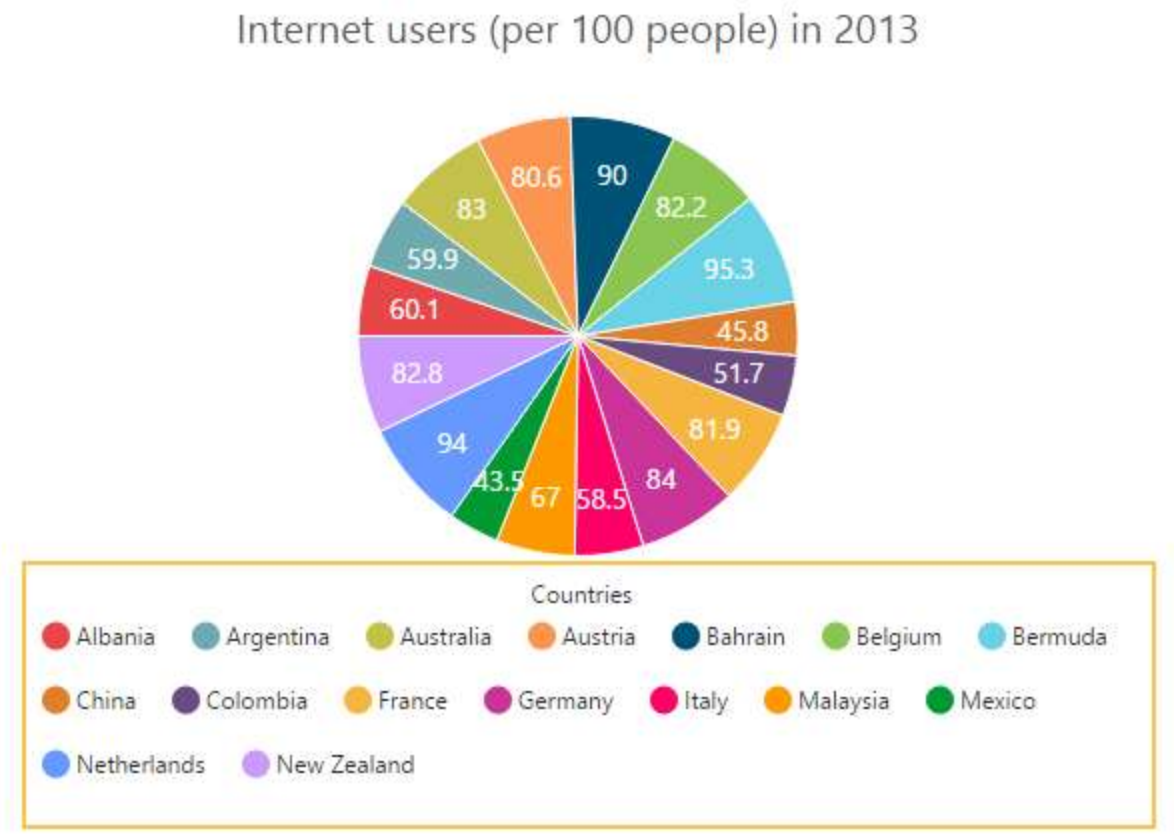
Legend border

You can customize the legend border by using the **Border** option in the legend.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Set border color and width to legend
.Border(br=>br.Color("#FFC342").Width(2))
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



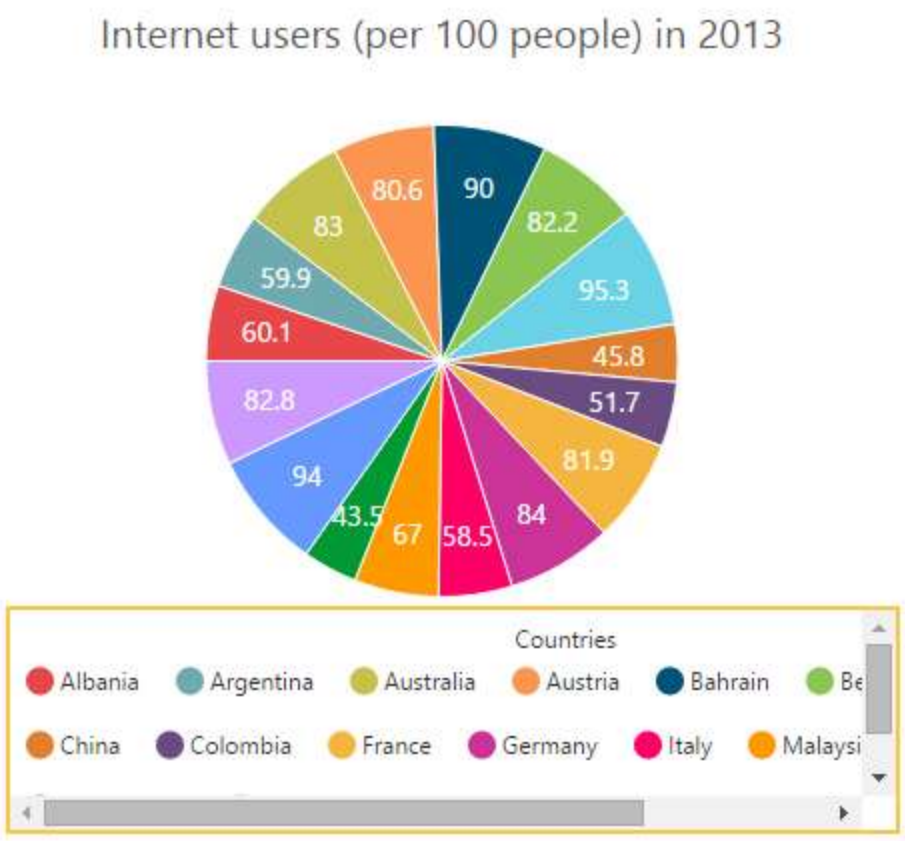
Scrollbar for legend

You can enable or disable the legend scrollbar by using the **EnableScrollbar** option of the legend. When you disable the scrollbar option, the legend does not consider the default size and chart draws in the remaining space. If you have specified the **size** to the legend with the scrollbar disabled, then the legends beyond this limit will get clipped.The default value of EnableScrollbar option is **true**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Enable scrollbar option in for legend
.EnableScrollbar(true)
.Size(size=>size.Width("430").Height("80"))
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



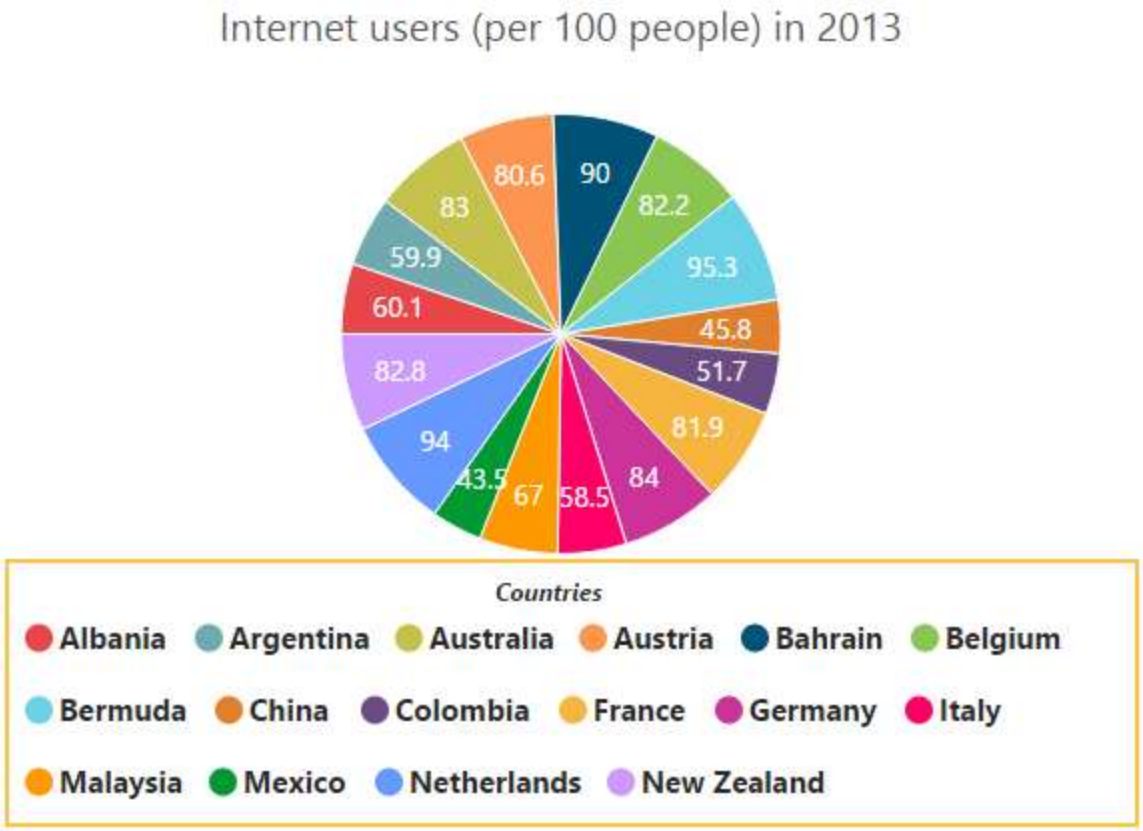
Customize the legend text

To customize the legend item text and title you can use the **Legend.Font** and **Legend.Title** options. You can change the legend title alignment by using the TextAlignment option of the legend title.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//...
//Customize the legend item text
.Font(font=>font.FontFamily("Segoe UI").FontStyle(ChartFontStyle.Normal)
.FontWeight(ChartFontWeight.Bold).Size("15px"))
.Title(title=>title
.TextAlignment(Syncfusion.JavaScript.DataVisualization.TextAlignment.Center)
//Customize the legend title text
.Font(font=>font.FontFamily("Segoe UI").FontStyle(ChartFontStyle.Italic)
.FontWeight(ChartFontWeight.Bold).Size("12px"))
)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



LegendItems Text Overflow

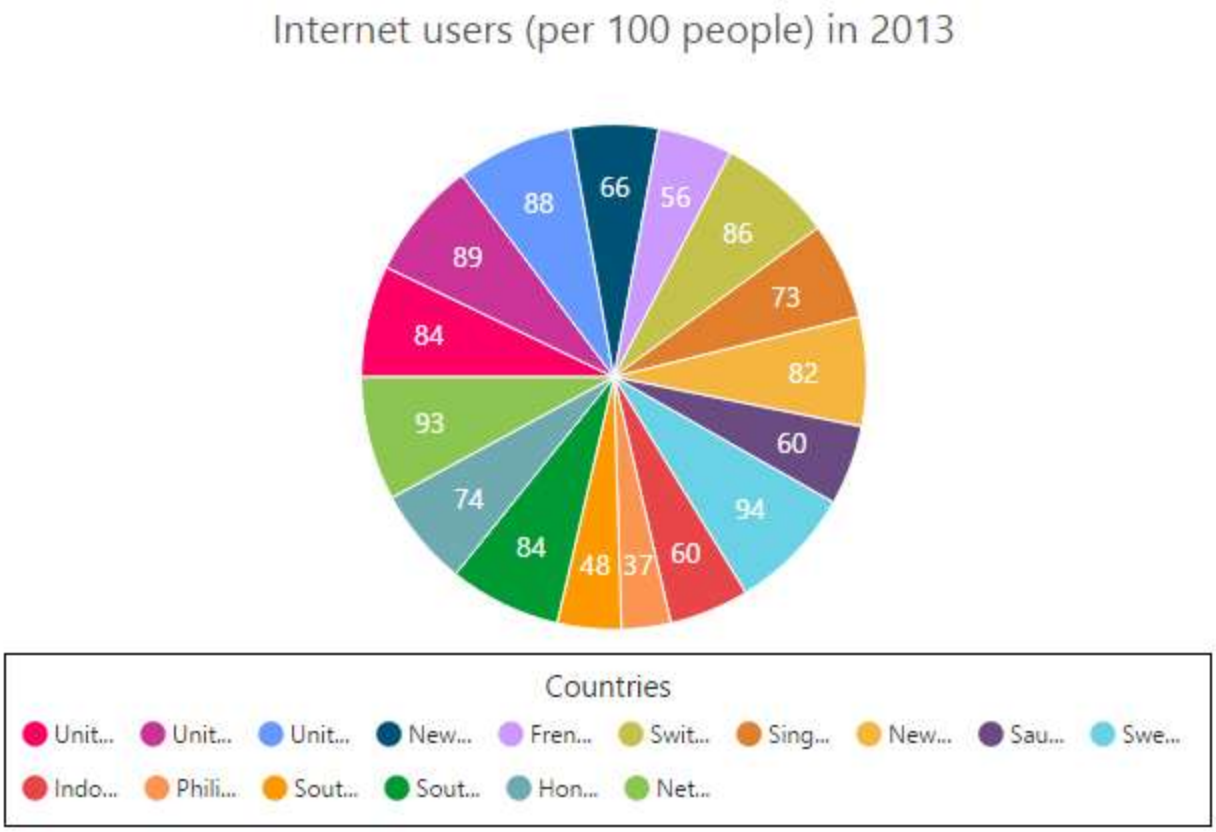
Trim

You can trim the legend item text when its width exceeds the **Legend.TextWidth**, by specifying *TextOverflow* as **Trim**. The original text will be displayed on mouse hover.

ASPX-CS

```
@(Html.EJ().Chart("container")
// ...
//trim the legend text
.Legend(lg=>lg.Visible(true).TextOverflow(TextOverflow.Trim)
.TextWidth(34))
// ...
)
```

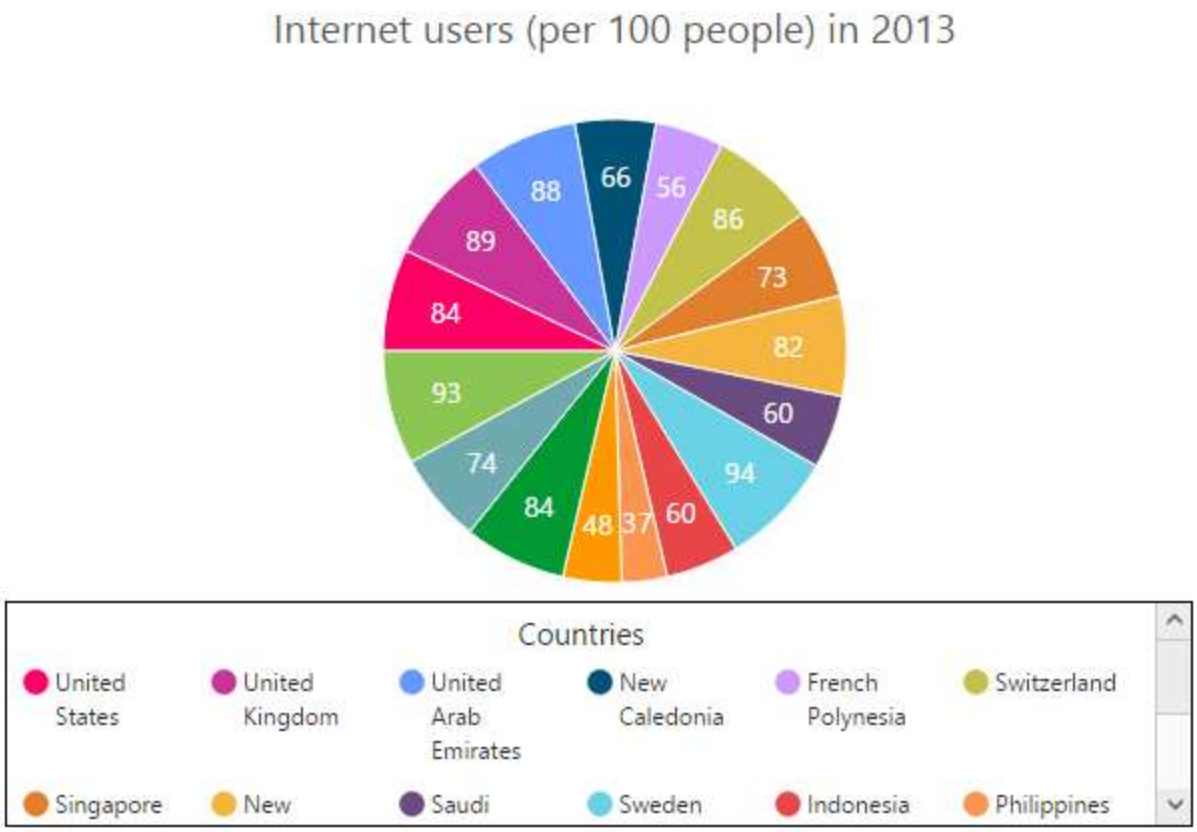
Copyright © 2001 - 2020 Syncfusion Inc.



Wrap

By specifying *TextOverflow* as **Wrap**, you can wrap the legend text by word.

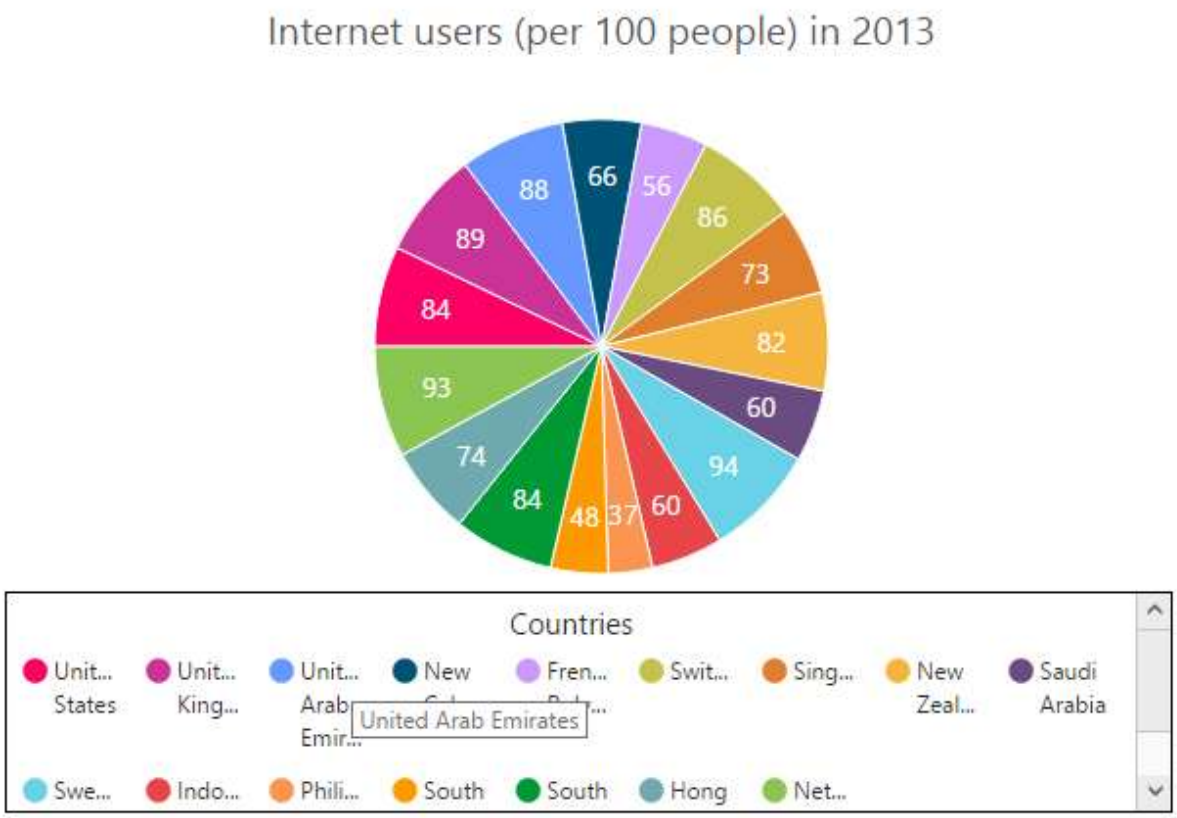
Copyright © 2001 - 2020 Syncfusion Inc.



WrapAndTrim

You can wrap and trim the legend text by specifying *TextOverflow* as **WrapAndTrim**. The original text will be displayed on mouse hover.

Copyright © 2001 - 2020 Syncfusion Inc.



Handle the legend item clicked

You can get the legend item details such as *Index*, *Bounds*, *Shape* and *Series* by subscribing the **LegendItemClick** event on the chart. When the legend item is clicked, it triggers the event and returns the Legend information.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
.Visible(true)
//...
)
//Subscribe the legend item click event
.LegendItemClick("onLegendClicked")
)
function onLegendClicked(sender) {
//Get legend item details on legend item click.
var legendItem = sender.data;
}
```

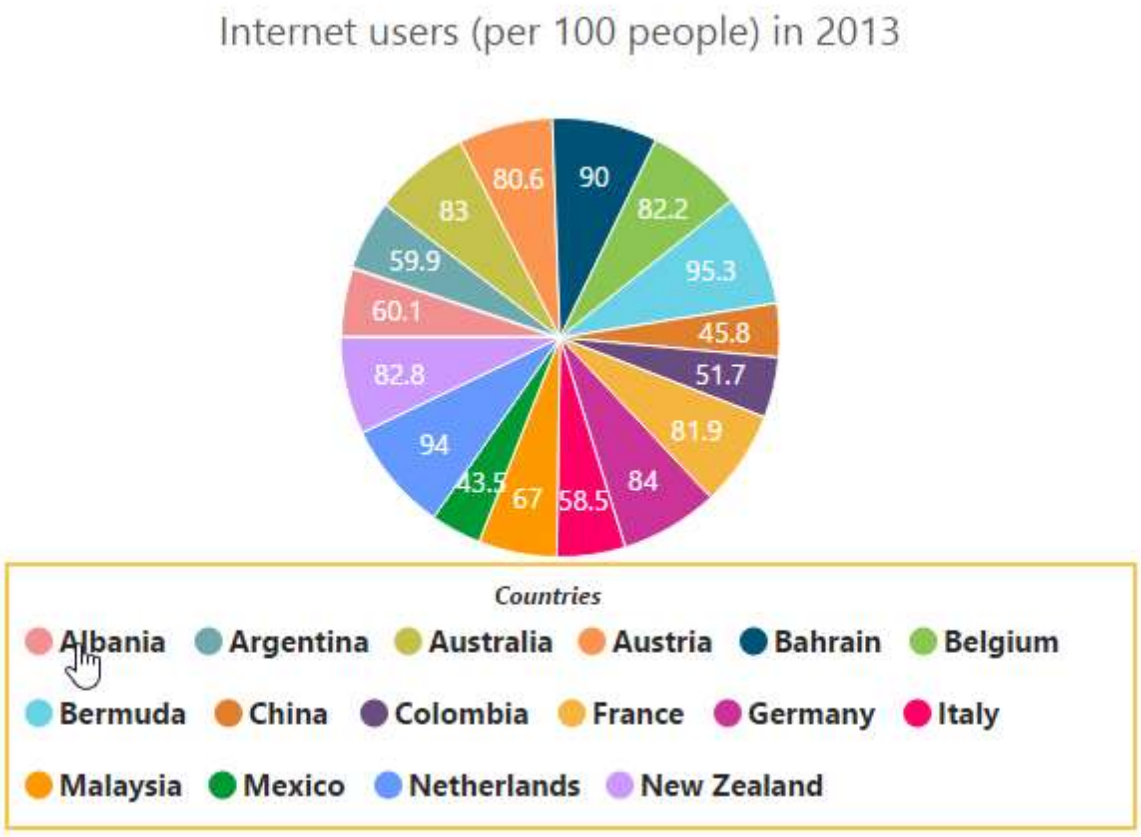
Series selection on legend item click

You can select a specific series or point while clicking on the corresponding legend item through disabling the **ToggleSeriesVisibility** option of the legend. The default value of toggleSeriesVisibility option is **true**. To customize the series selection refer to the series Selection.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Legend(lg=>lg
//Disable series collapsing on legend item clicked
.ToggleSeriesVisibility(false)
//...
)
//...
)
```



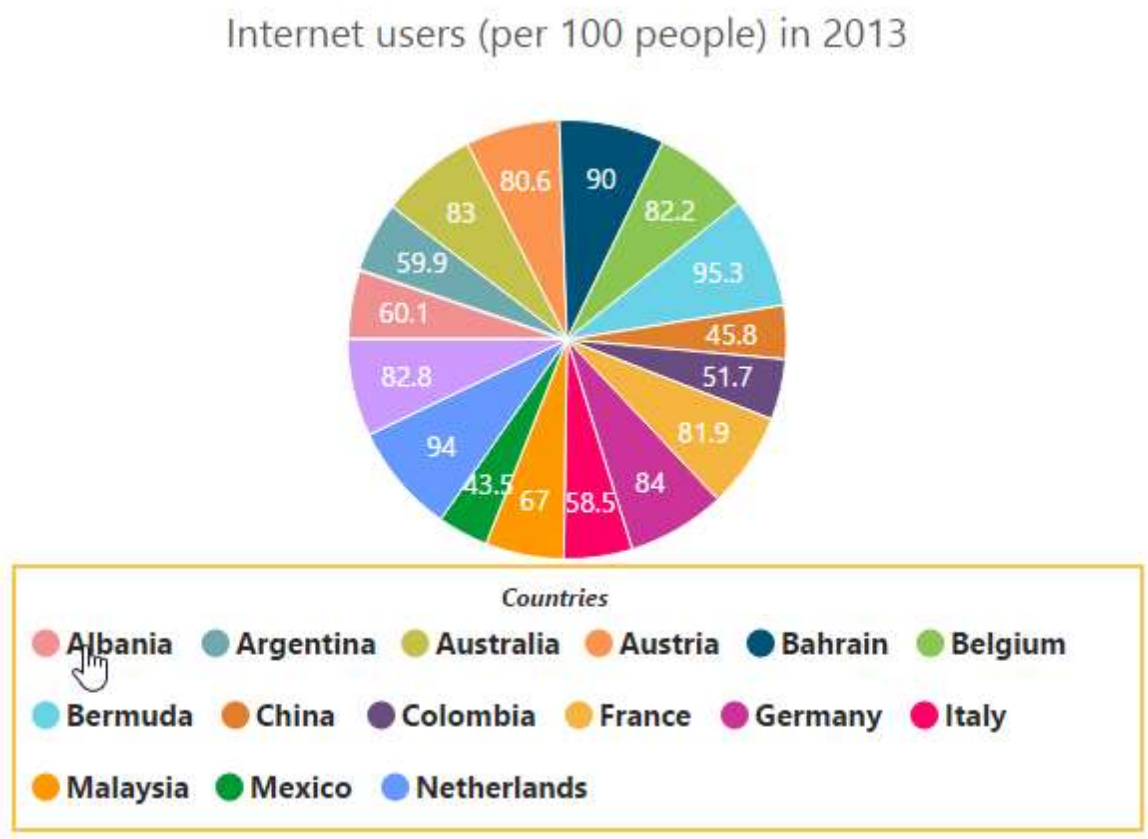
Collapsing legend item

You can collapse the specific series/point legend item displaying in the chart, by setting the **VisibleOnLegend** as *Hidden* in the point or series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
. Series(sr =>{
//Initializing Series
sr.Points(pt => {
pt.X("Albania").Y(60.1).Add();
//...
//Collapse the point's legend item in the legend collection
```

```
pt.X("New Zealand").Y(82.8).VisibleOnLegend(VisibleOnLegend.Hidden).Add();
})// visible the EmptyPointSettings
.EmptyPointSettings(e => e.Visible(true)).Add();
// ...
})
.Legend(lg=>lg.Visible(true))
// ...
)
```



Empty Points

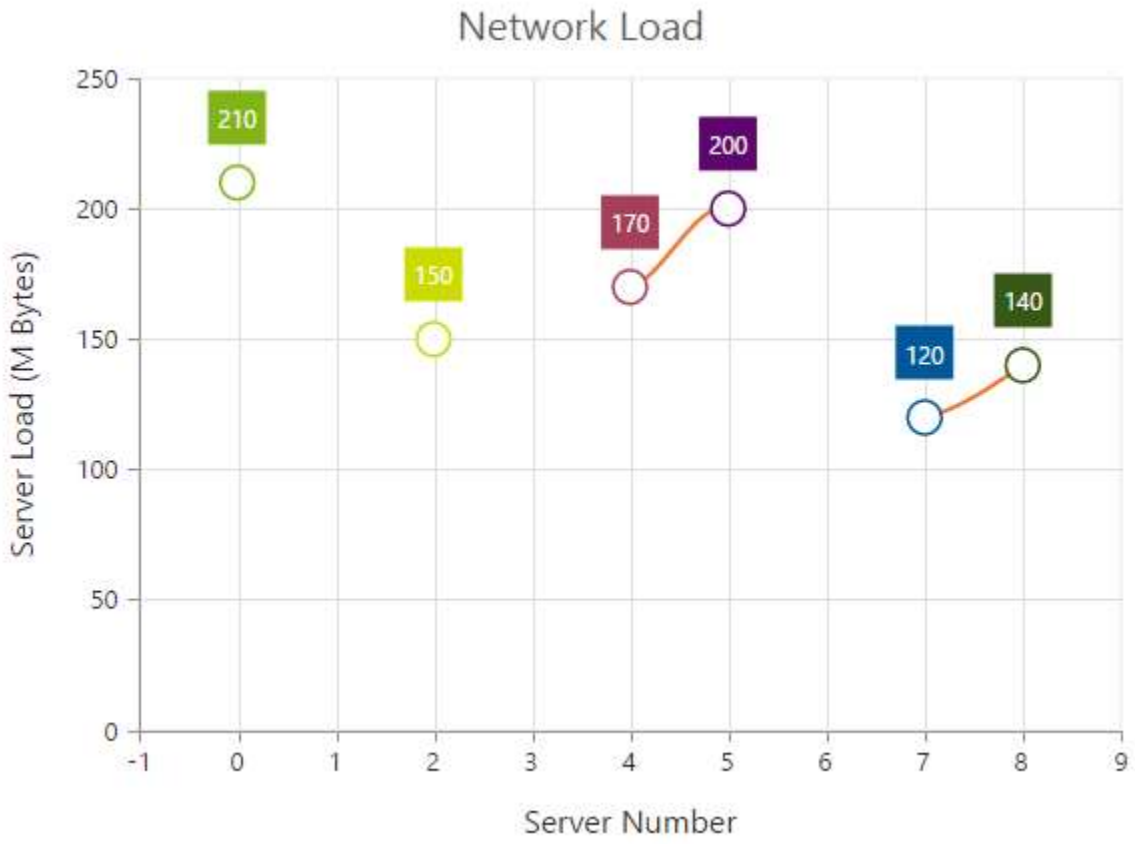
The Data points that uses the **null** or **undefined** as value are considered as empty points. Empty data points are ignored and not plotted in the Chart. When the data is provided by using the Points property, you can set the **IsEmpty** to true to specify that the particular point is an empty point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
. Series(sr =>{
//Using empty points
sr.Points(pt => {
pt.X("0").Y(210).Add(); pt.X("1").Y(0).Add();
pt.X("2").Y(150).Add(); pt.X("3").Y(180).IsEmpty(true).Add();
pt.X("4").Y(170).Add(); pt.X("5").Y(200).Add();
pt.X("6").Y(140).IsEmpty(true).Add();
pt.X("7").Y(120).Add(); pt.X("8").Y(140).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
// ...
})
// ...
)
```



Click here to view the online demo sample for empty points.

EmptyPointSettings

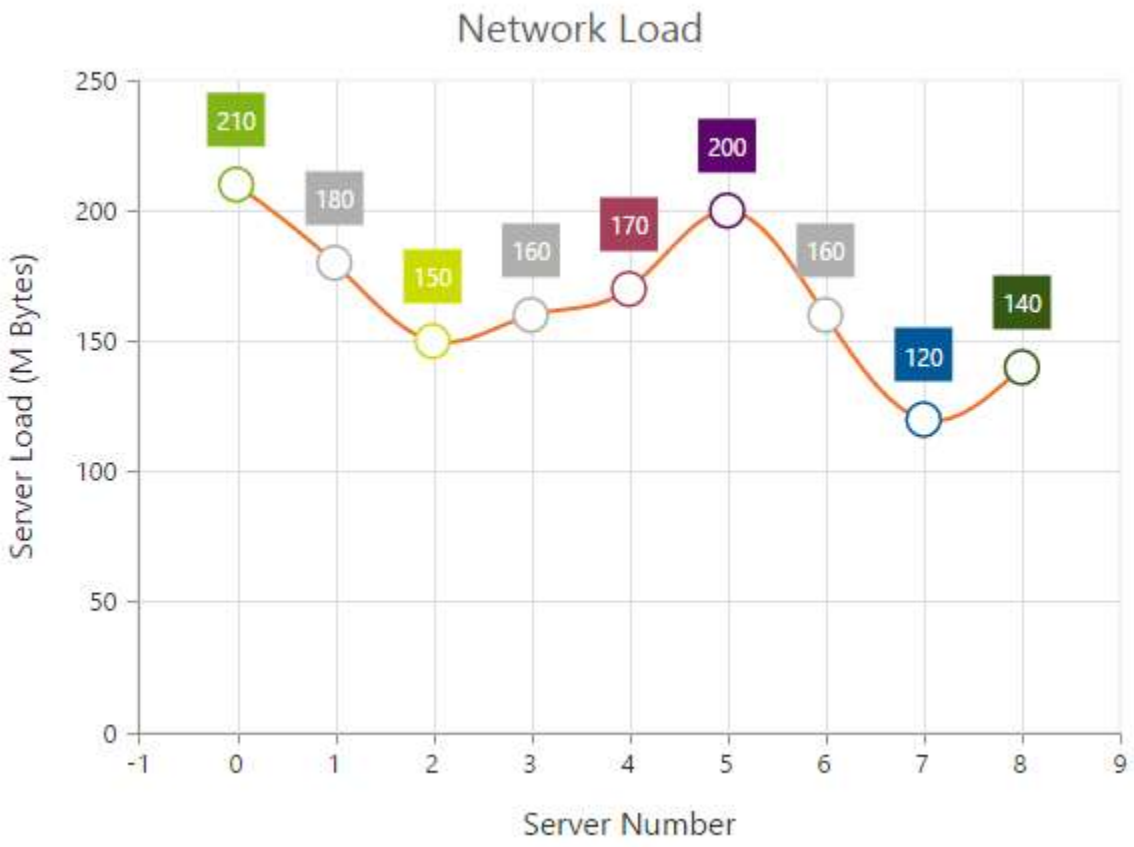
You can customize the empty points visibility and change its **DisplayMode** *(Gap, Zero and Average)* using *EmptyPointSettings* option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
. Series(sr =>{
//Using empty points
sr.Points(pt => {
pt.X("0").Y(210).Add(); pt.X("1").Y(0).Add();
pt.X("2").Y(150).Add(); pt.X("3").Y(180).IsEmpty(true).Add();
pt.X("4").Y(170).Add(); pt.X("5").Y(200).Add();
pt.X("6").Y(140).IsEmpty(true).Add();
pt.X("7").Y(120).Add(); pt.X("8").Y(140).Add();
})// visible the EmptyPointSettings
.EmptyPointSettings(e => e.Visible(true)
```

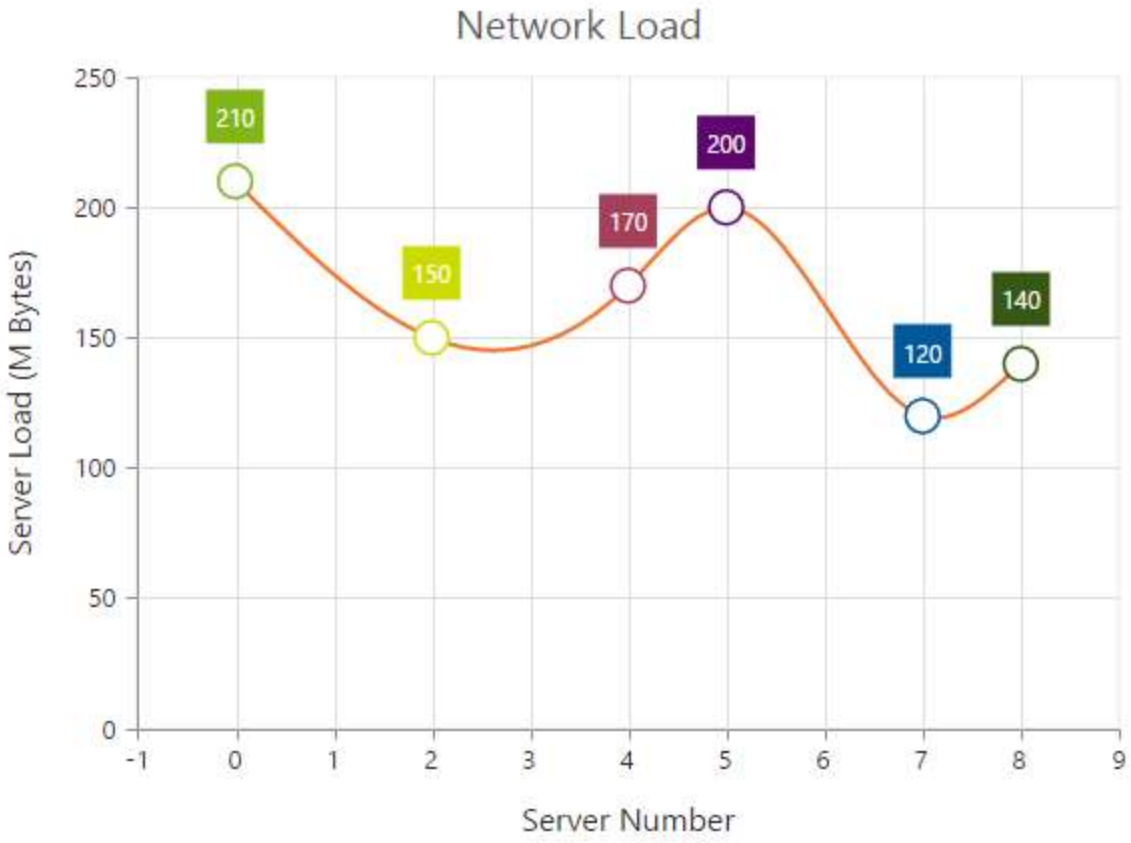
Copyright © 2001 - 2020 Syncfusion Inc.

```
.DisplayMode(ChartEmptyPointMode.Average)
).Add();
// ...
})
// ...
)
```



If the *Visible* property of EmptyPointSettings is *false*, then the empty points has been dropped and chart will be rendered without empty points.

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing Styles

Empty points color and border can be customized using **Style** property of EmptyPointSettings.

ASPX-CS

```
@(Html.EJ().Chart("container")
// ...
. Series(sr =>{
.EmptyPointSettings(e => e.Visible(true)
.Style(s => sl.Color("Pink")
.Border(b => b.Color("gray").Width(2)))
.Add();
// ...
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

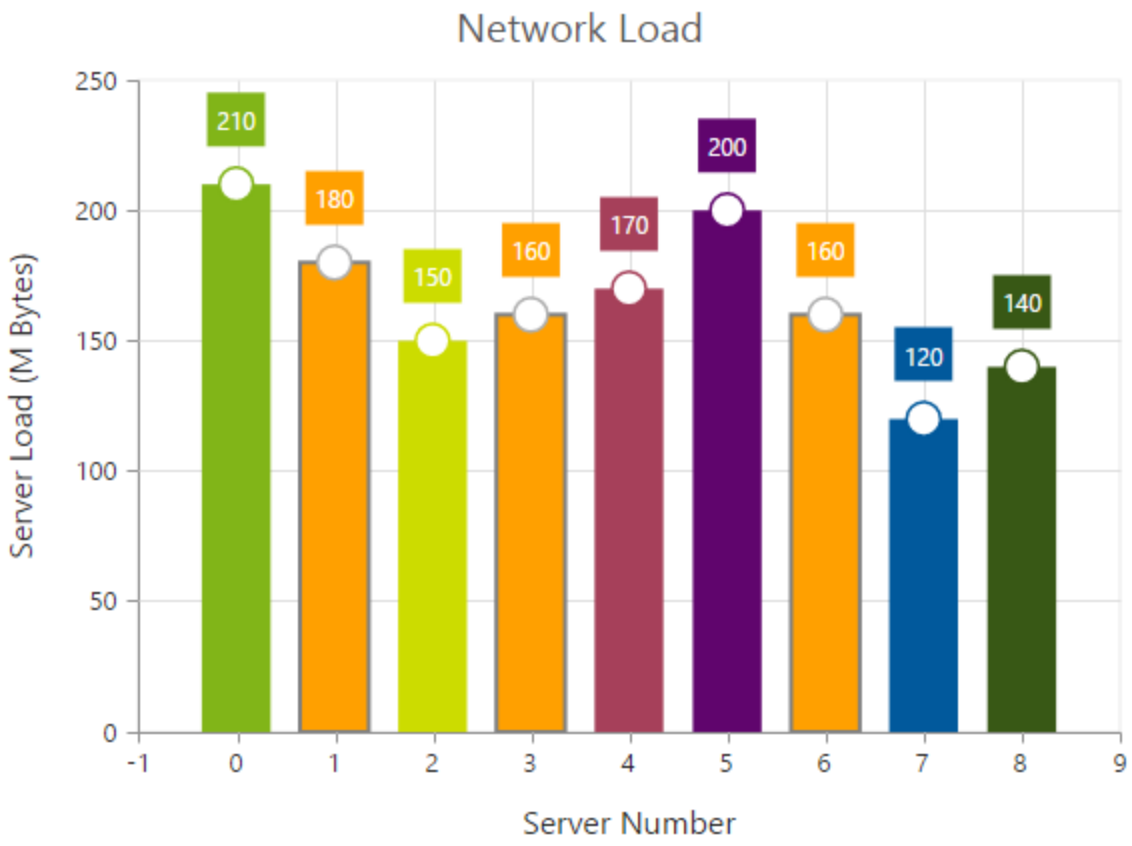


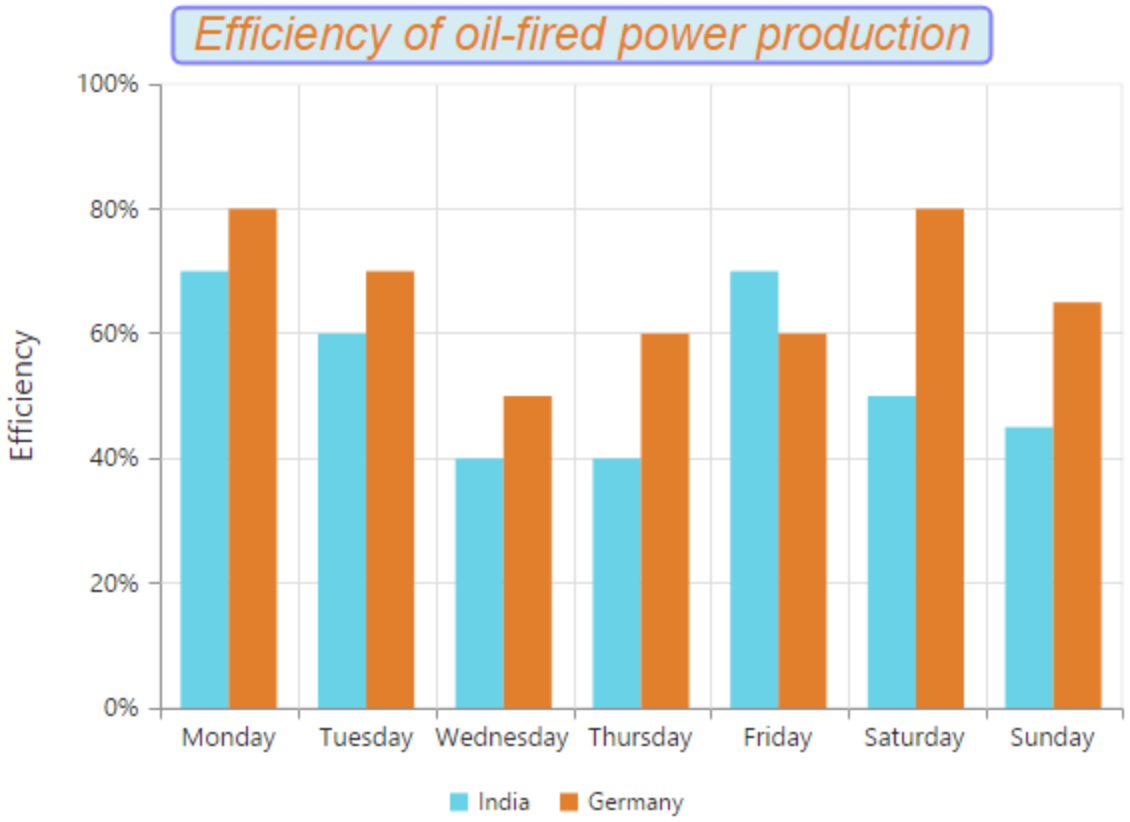
Chart Title & Subtitle

Title

By using the **Title** option, you can add the *Text* as well as customize its *Border, Background and Font*.

ASPX-CS

```
@(Html.EJ().Chart("container")
// ...
//Adding text to chart title
.Title(title => title.Text("Efficiency of oil-fired power production")
//Change the title text background color
.Background("lightblue")
//Customizing Chart title border
.Border(b => b.Color("blue").Width(2).CornerRadius(4.0).Opacity(0.5))
//Customizing Chart title font
.Font(f => f.Opacity(1).FontFamily("Arial").FontStyle("italic
").FontWeight("regular")
.Color("#E27F2D").Size("23px"))
,
// ...
)
```

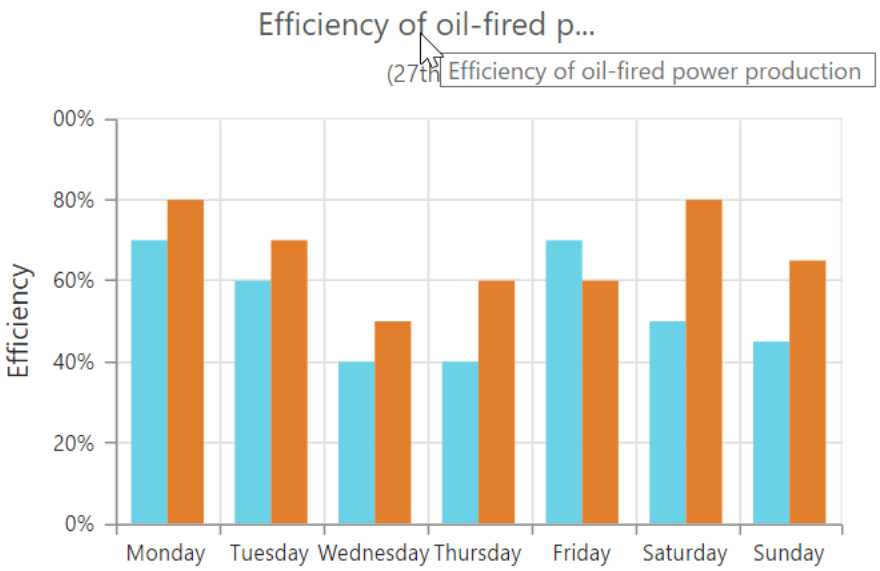


Click here to view the Chart Title online demo sample.

We can trim, wrap and wrapAndTrim to the chart title using textOverflow property. The original text will be displayed as tooltip on mouse hover.

JAVASCRIPT

```
@(Html.EJ().Chart("chartContainer")
//  ..
// To enable title trim and wrap
.Title(title => title.Text("Efficiency of oil-fired power production ")
.EnableTrim("true")
.MaximumWidth(150)
.TextOverflow("trim"))
//  ..
)
```



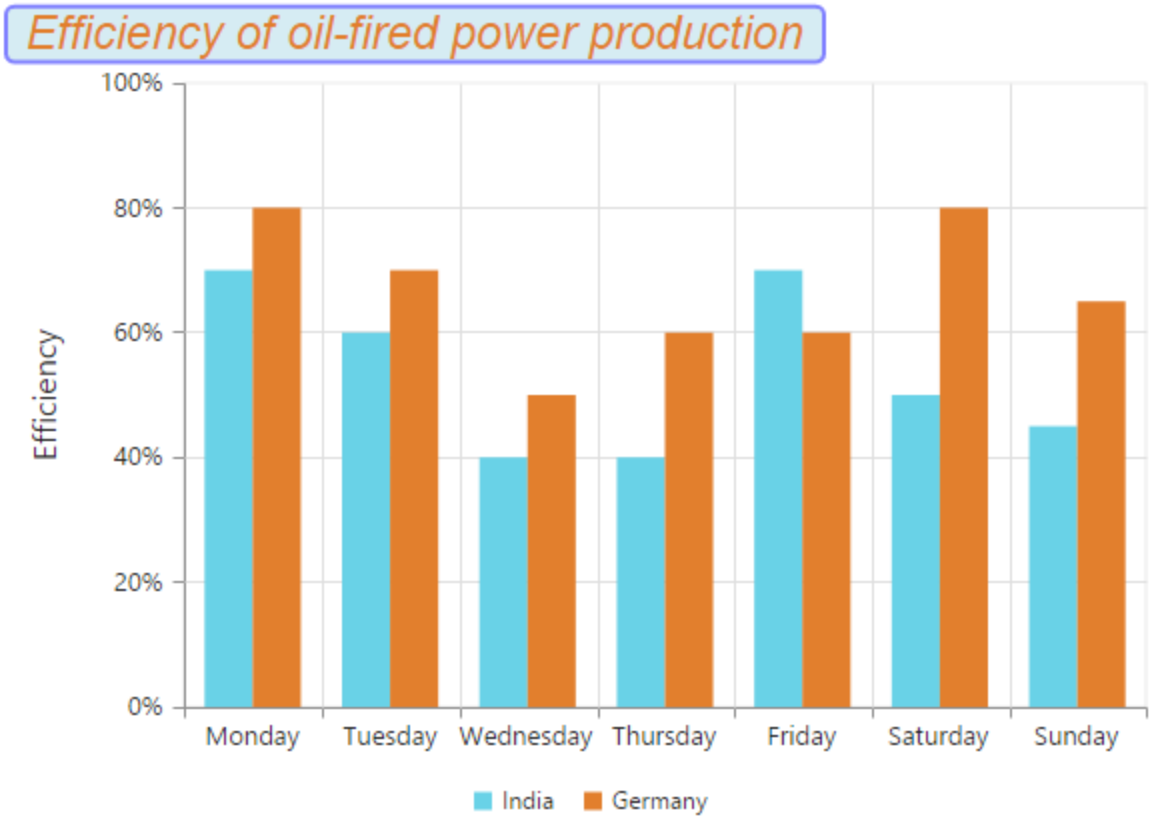
Title Alignment

You can change the title alignment to *Center*, *Far* and *Near* by using the **TextAlignment** property of the chart title.

ASPX-CS

```
@(Html.EJ().Chart("container")
// ...
.Title(title => title.Text("Efficiency of oil-fired power production")
//Change title text alignment
.TextAlignment(Syncfusion.JavaScript.DataVisualization.TextAlignment.Far)
//...
}
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



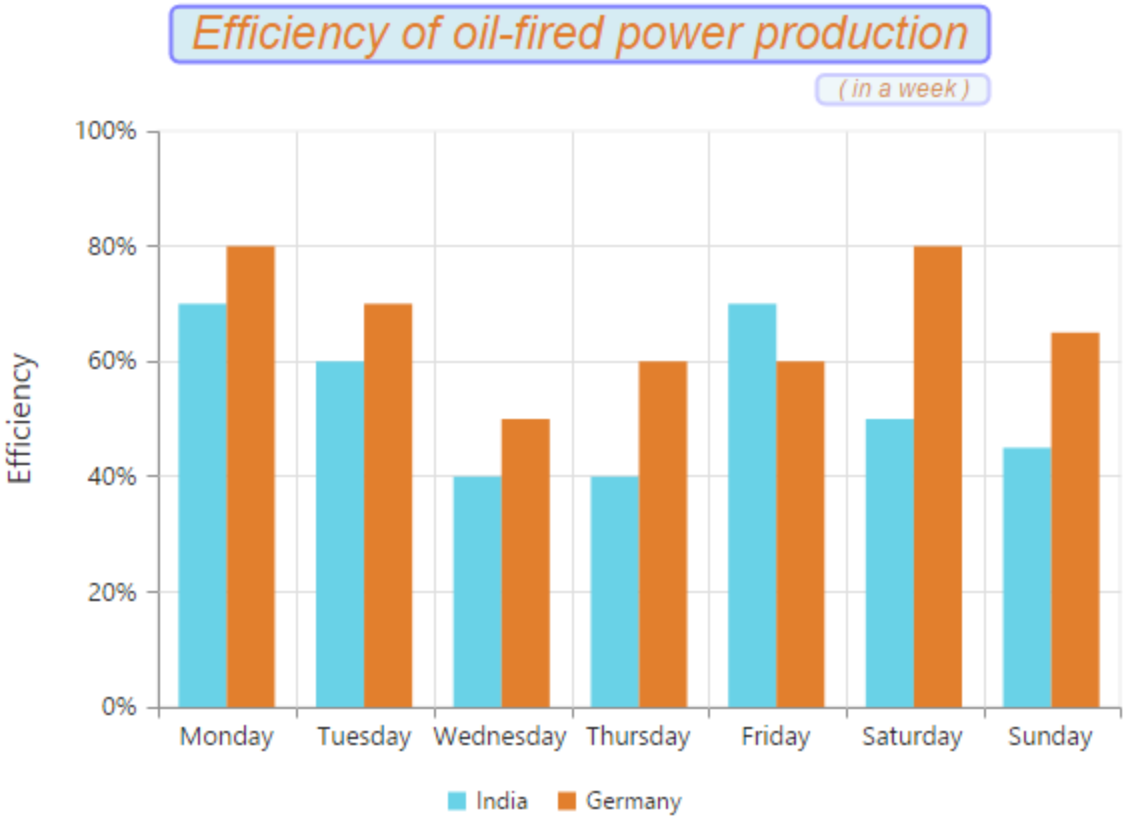
Add Subtitle to the chart

By using the **SubTitle** option, you can add the SubTitle to the chart title and customize its *Border, Background and Font*.

ASPX-CS

```
@(Html.EJ().Chart("container")
// ...
.Title(title => title.Text("Efficiency of oil-fired power production")
//Add subtitle to chart title
.SubTitle(t2 => t2.Text("(in a week)")
//Change the subtitle text background color
.Background("lightblue")
//Customizing Chart subtitle border
.Border(b => b.Color("blue").Width(2).CornerRadius(4.0).Opacity(0.2))
//Customizing Chart subtitle font
.Font(f =>
f.Opacity(1).FontFamily("Arial").FontStyle("italic").FontWeight("regular")
.Color("#E27F2D").Size("12px"))
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

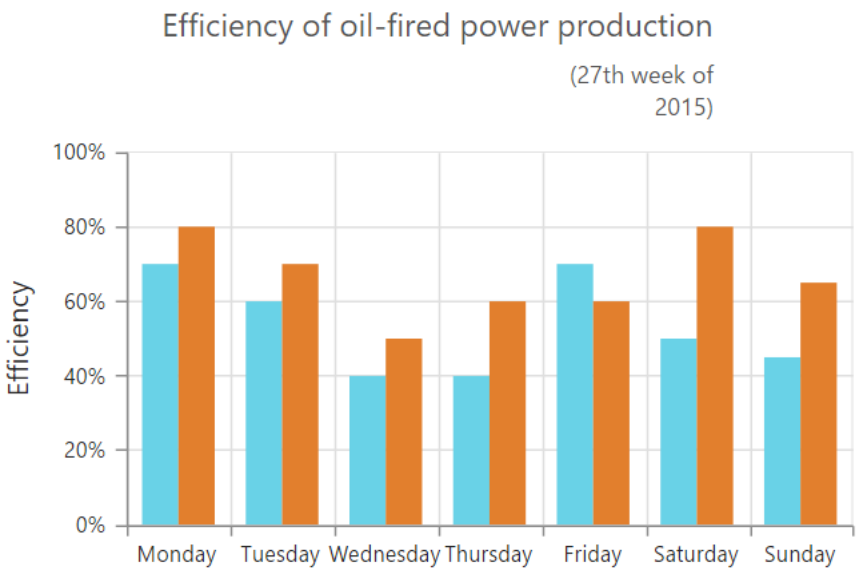


We can trim, wrap and wrapAndTrim to the chart subtitle using textOverflow property. The original text will be displayed as tooltip on mouse hover.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//  ..
// To enable sub-title trim and wrap
.Title(title => title.SubTitle(sTitle => sTitle.Text("(In a Week) ")
.EnableTrim("true")
.MaximumWidth(150)
.TextOverflow("wrap")))
//  ..
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



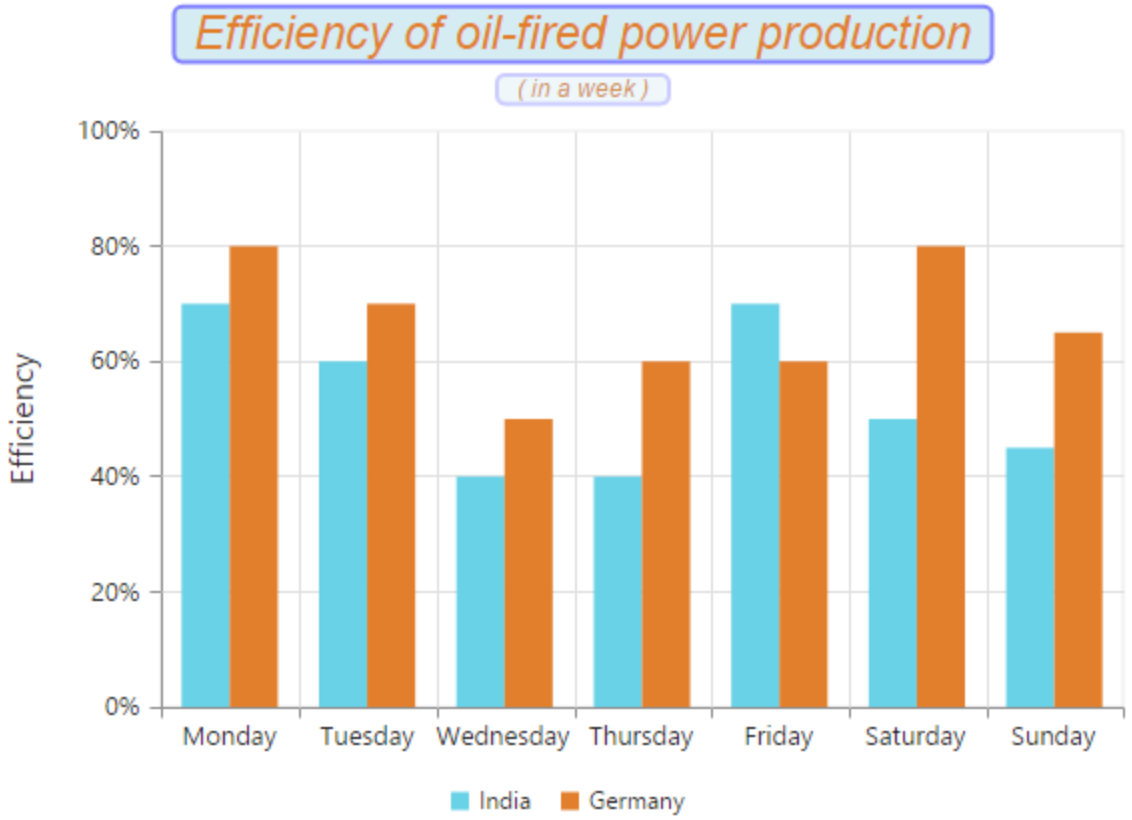
Subtitle Alignment

You can change the SubTitle alignment to *Center*, *Far* and *Near* by using the **TextAlignment** property of the SubTitle.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Title(title = >title
//...
.SubTitle(st=>st
//Change subtitle text alignment
.TextAlignment(Syncfusion.JavaScript.DataVisualization.TextAlignment.Center)
//...
)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Striplines

EjChart supports horizontal and vertical striplines.

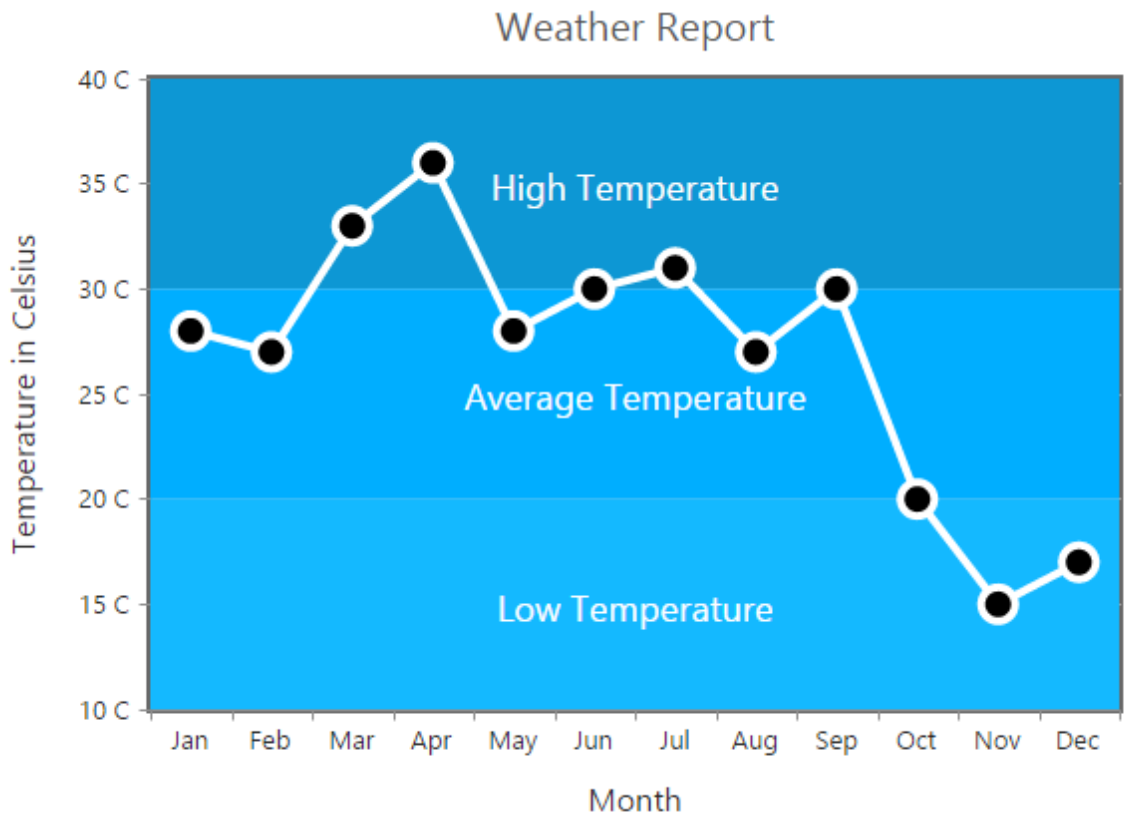
Horizontal Stripline

You can create horizontal stripline by adding the **Stripline** in the *vertical axis* and set **Visible** option to *true*. Striplines are rendered in the specified **Start** to **End** range and you can add more than one stripline for an axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Primary Y Axis
.PrimaryYAxis(axis => axis.StripLine(line =>
{
//Enable Stripline
line.Visible(true)
.Start(30).End(40).Add();
}))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



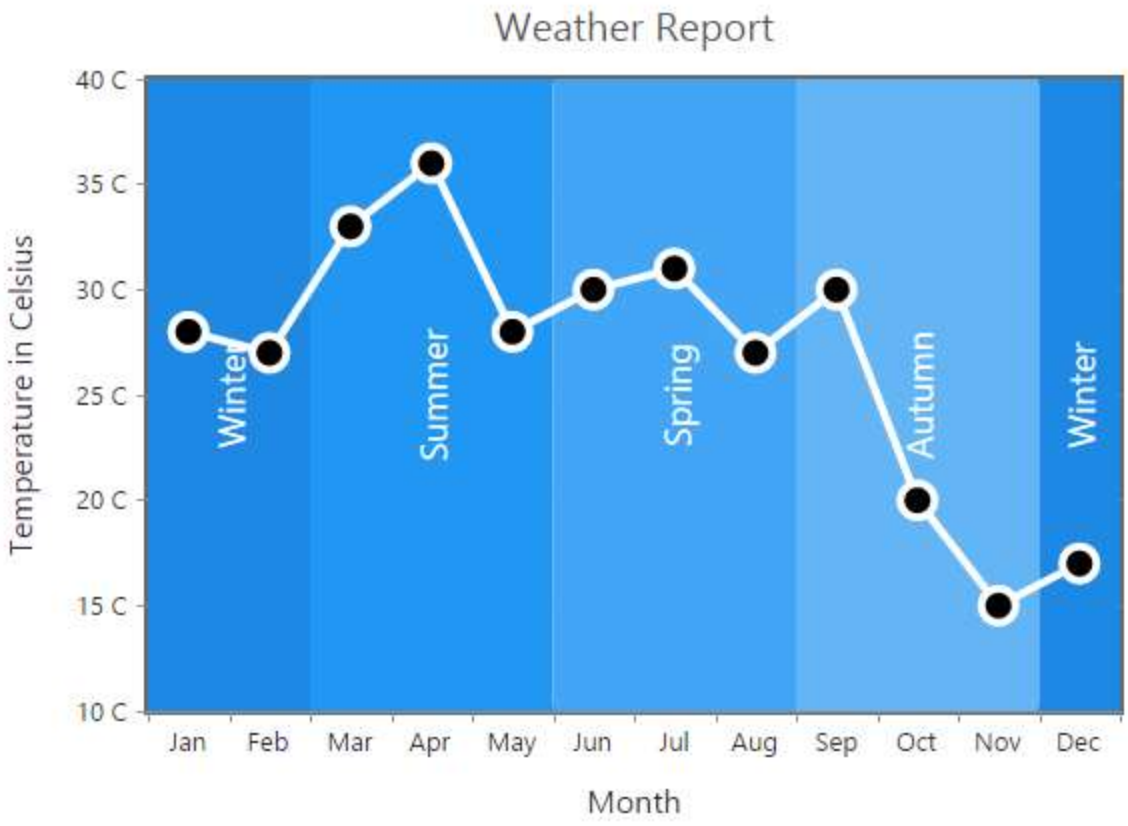
Click here to view the Striplines online demo sample.

Vertical Stripline

You can create vertical stripline by adding the Stripline in the *horizontal axis* and set Visible option to *true*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Primary X Axis
.PrimaryXAxis(axis => axis.StripLine(line =>
{
//Create vertical Stripline using horizontal Axis
line.Visible(true)
.Start(3).End(7).Add();
}))
//...
)
```

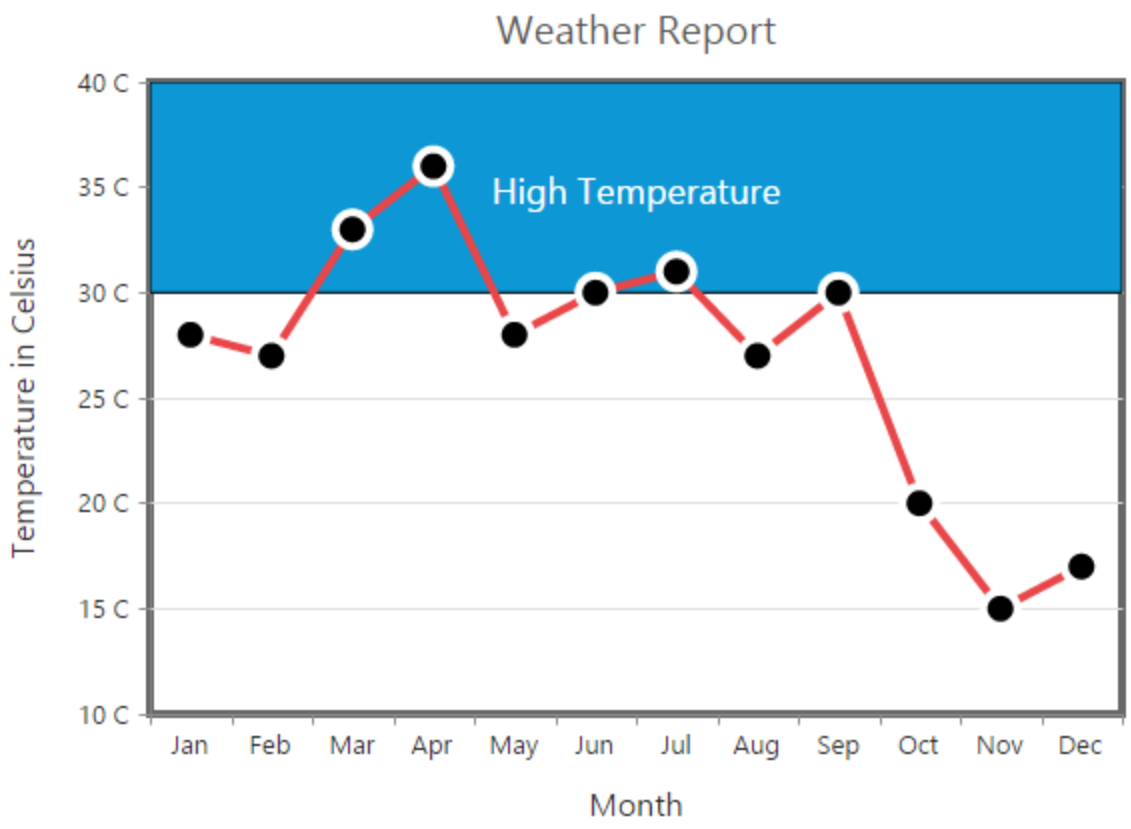


Customize the Text

To customize the stripLine text, use the **Text** and **Font** options.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Primary Y Axis
.PrimaryYAxis(axis => axis.StripLine(line =>
{
//Customize the stripLine text and font styles
line.Text("High Temperature")
.Font(font=>font.Size("18px").Color("white")).Add();
}))
//...
)
```



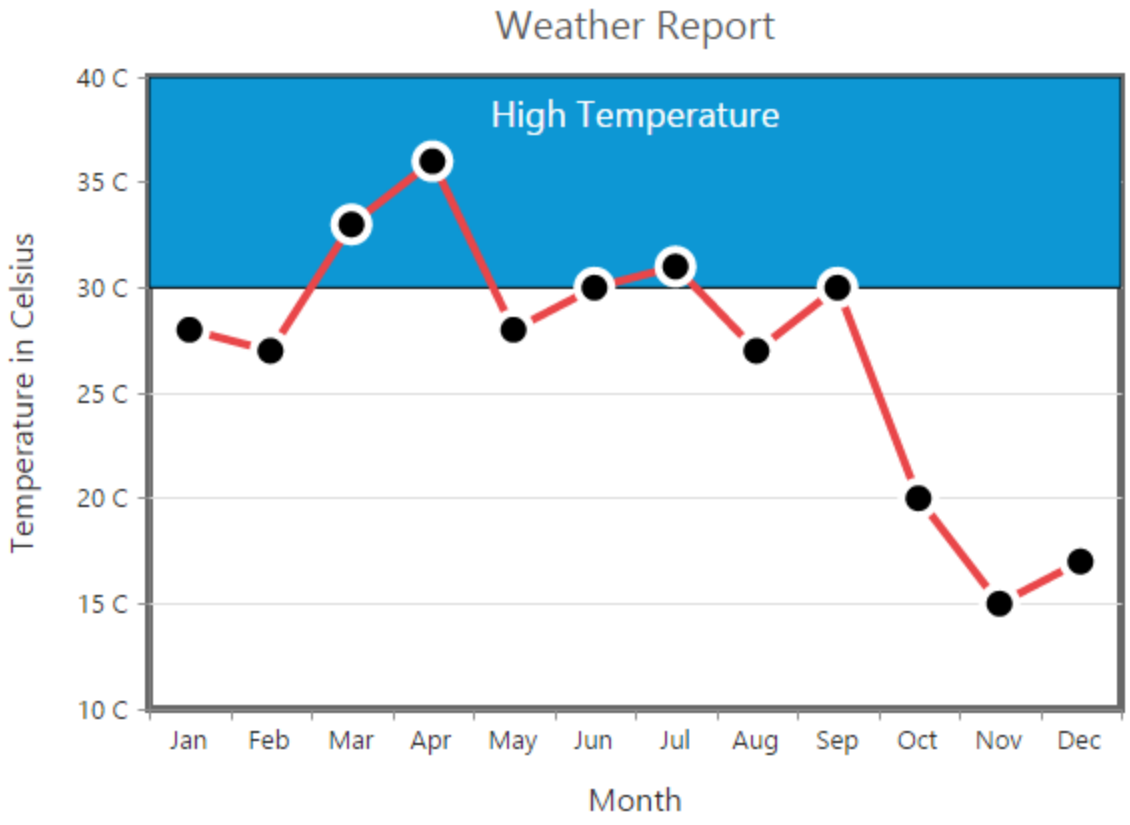
Text Alignment

Stripline text can be aligned by using the **TextAlignment** property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Primary Y Axis
.PrimaryYAxis(axis => axis.StripLine(line =>
{
//Set stripLine text alignment to top position
line.TextAlignment(StriplineTextAlignment.MiddleTop)
.Add();
}))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



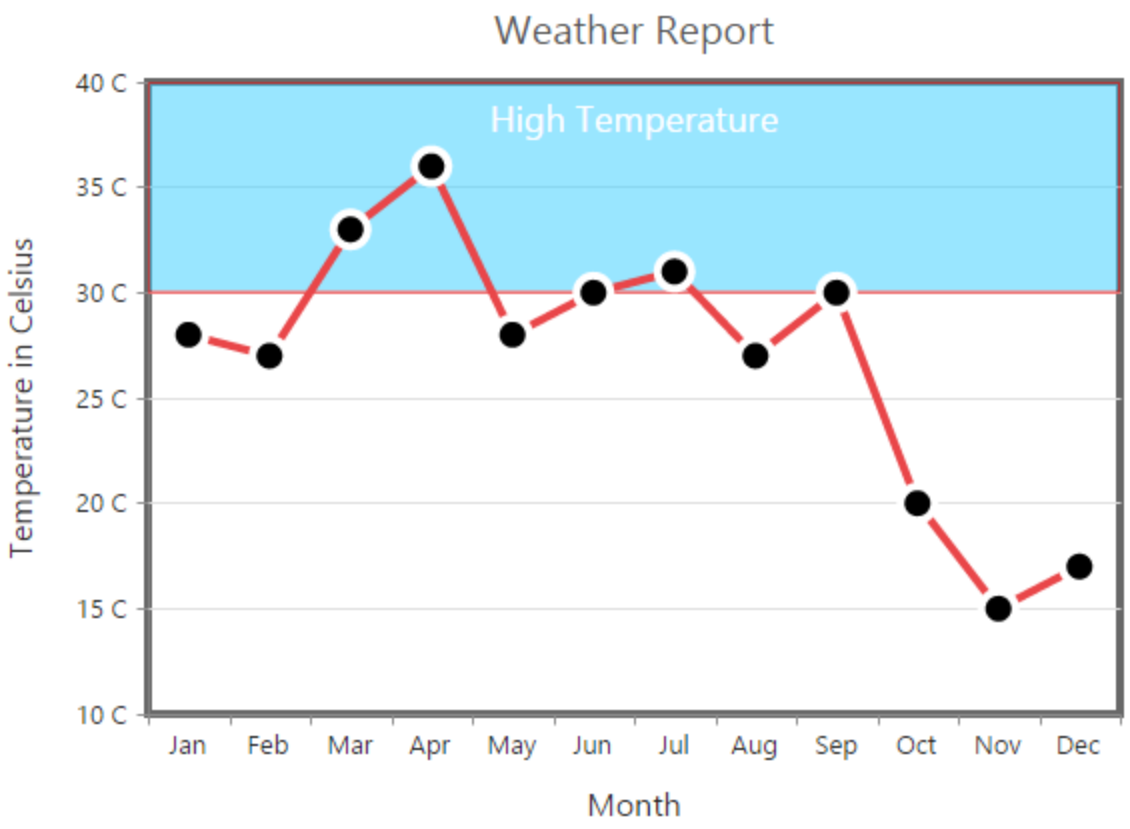
Customize the Stripline

To customize the stripLine styles, use the *Color, Opacity, BorderWidth and BorderColor* properties.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Primary Y Axis
.PrimaryYAxis(axis => axis.StripLine(line =>
{
//Customize the StripLine rectangle
line.Color("#33CCFF").BorderWidth(2).Opacity(0.5).BorderColor("red")
.Add();
}))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



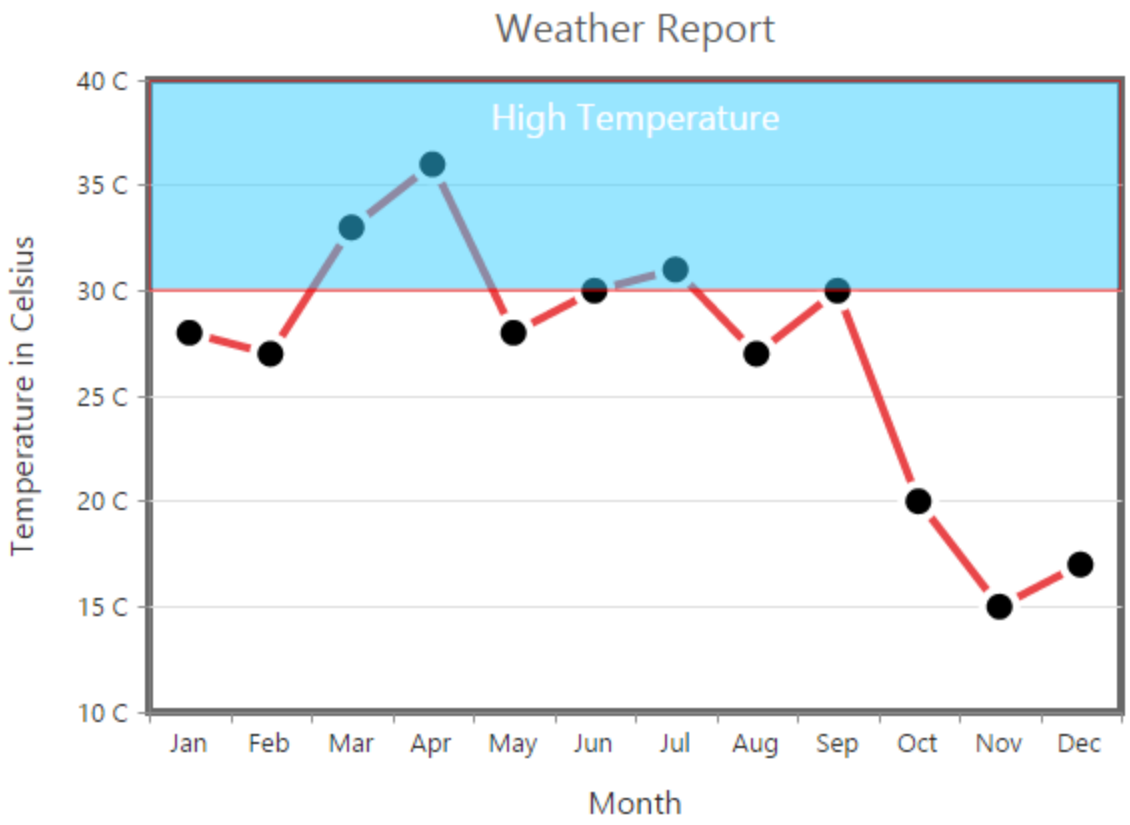
Change the Z-order of the stripline

Stripline **ZIndex** property is used to display the stripLine either behind or over the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Primary Y Axis
.PrimaryYAxis(axis => axis.StripLine(line =>
{
//Change stripLine ZIndex
line.ZIndex(ChartZIndex.Over)
.Add();
}))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



User Interactions

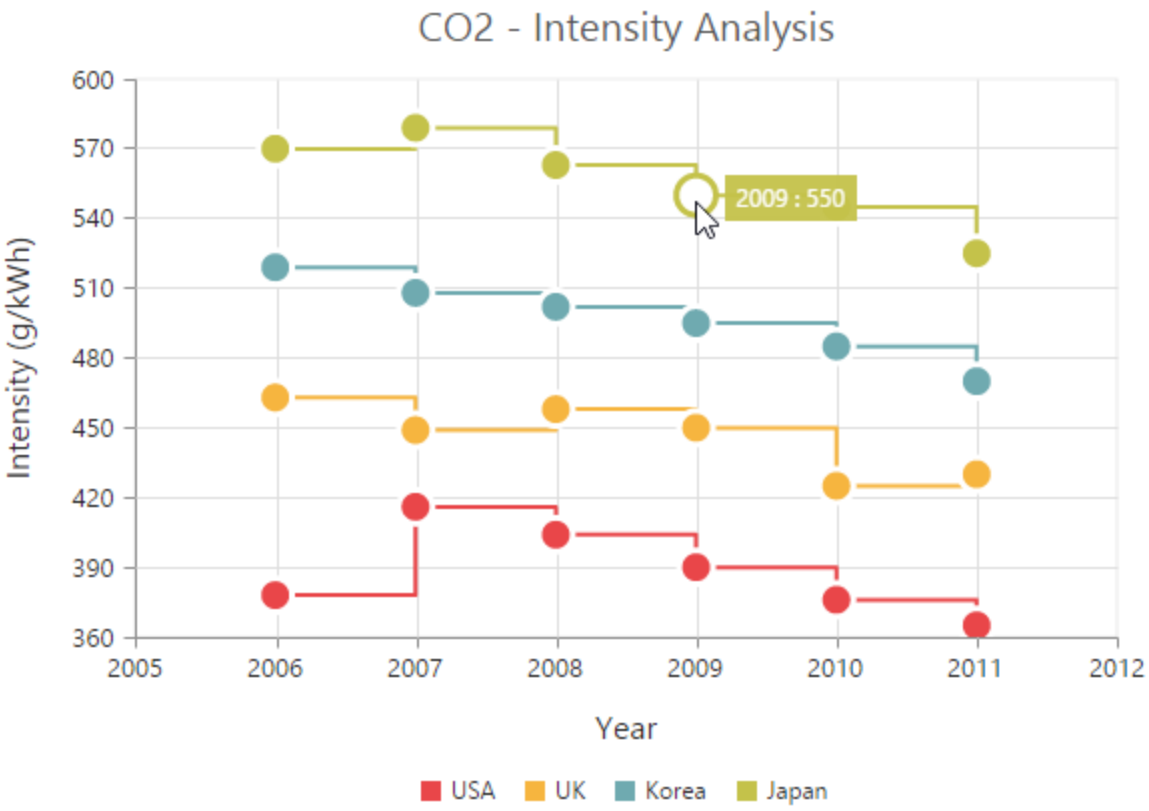
Tooltip

Enable tooltip for data point

Tooltip for the data points can be enabled by using the *Visible* option of the **Tooltip** in the series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(ser =>
{
ser.Tooltip(tooltip=>tooltip
//Enable tooltip for the series
.Visible(true)
).Add();
})
//...
)
```



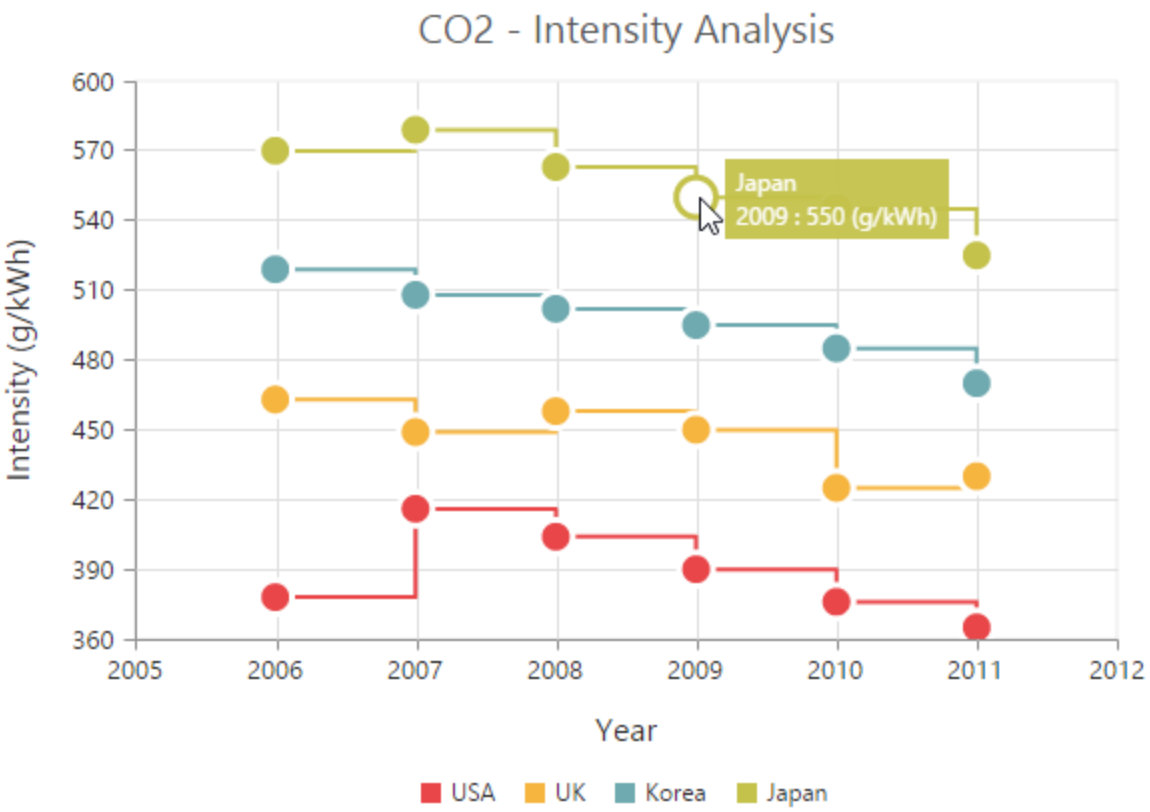
Format the tooltip

Tooltip displays data specified by the **Format** option of the tooltip. The default value of the format option is *#point.x# : #point.y#* . Here, *#point.x#* is the placeholder for x value of the point and *#point.y#* is the placeholder for y value of the point.

You can also use *#series.<optionname>#* as placeholder to display the value of an option in corresponding series and use *#point.<optionname>#* as place holder to display the value of an option in the corresponding point.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(ser =>
{
ser.Tooltip(tooltip=>tooltip
//Displaying tooltip in a format
.Format("#series.name# <br/> #point.x# : #point.y#  (g/kWh)")
).Add();
})
//...
)
```



Tooltip Template

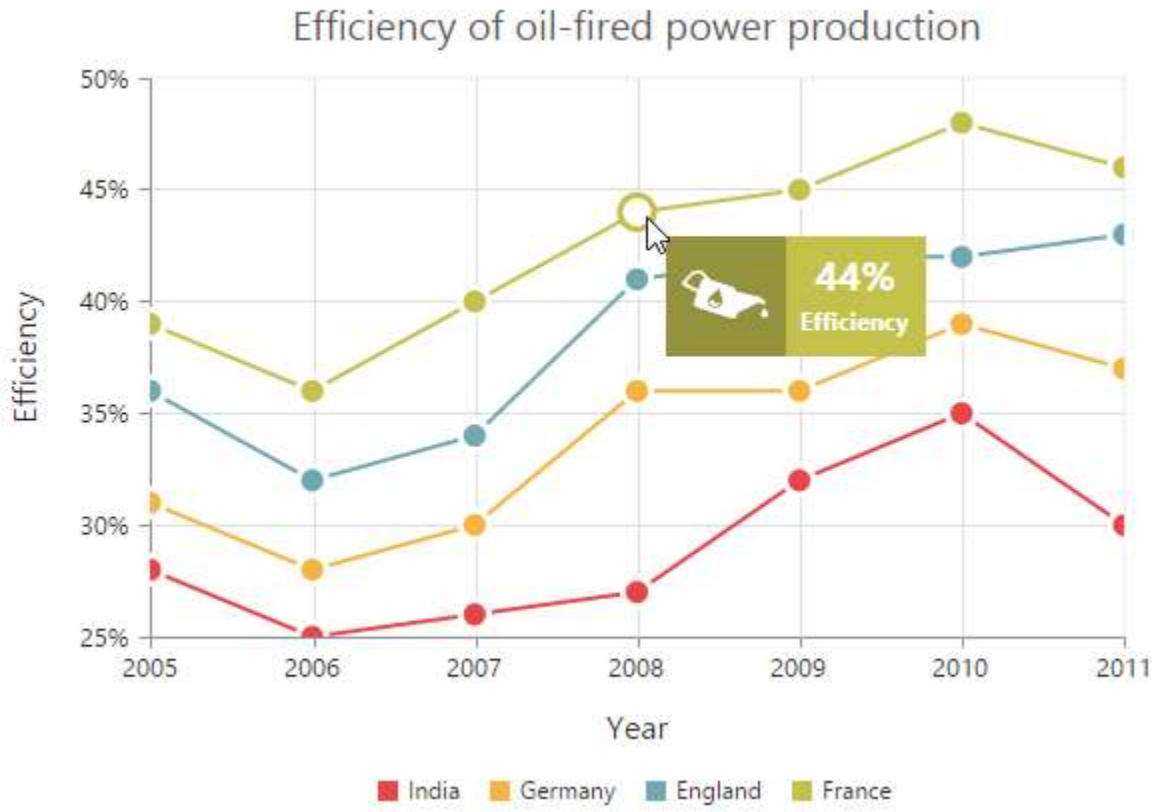
HTML elements can be displayed in the tooltip by using the **Template** option of the tooltip. The template option takes the value of the id attribute of the HTML element. You can use the *#point.x#* and *#point.y#* as place holders in the HTML element to display the x and y values of the corresponding data point.

You can also use *#series.<optionname>#* as place holder to display the value of an option in corresponding series of the tooltip and use *#point.<optionname>#* as place holder to display the value of an option in the corresponding point for which the tooltip is displayed.

ASPX-CS

```
<!-- Create Tooltip template here -->
<div id="Tooltip" style="display: none;">
<div id="icon"> <div id="eficon"> </div>  </div>
<div id="value">
<div>
<label id="efpercentage"> #point.y#% </label>
<label id="ef">Efficiency </label>
</div>
</div>
</div>
@(Html.EJ().Chart("chartContainer")
//...
.Series(ser =>
{
ser.Tooltip(tooltip=>tooltip
```

```
//Set template id to tooltip template
.Template("Tooltip")
).Add();
})
//...
)
```



Click here to view the Tooltip template online demo sample.

Tooltip template animation

You can enable animation by setting the *EnableAnimation* to true. Tooltip animates when the mouse moves from one data point to another point. The Duration property in tooltip specifies the time taken to animate the tooltip. the duration is set to **"500ms"**, by default.

Note: Tooltip is animated only if the template is specified for tooltip.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(ser =>
{
ser.Tooltip(tooltip=>tooltip
//Enable tooltip template animation and set duration time
.EnableAnimation(true).Duration("1000ms")
).Add();
})
```

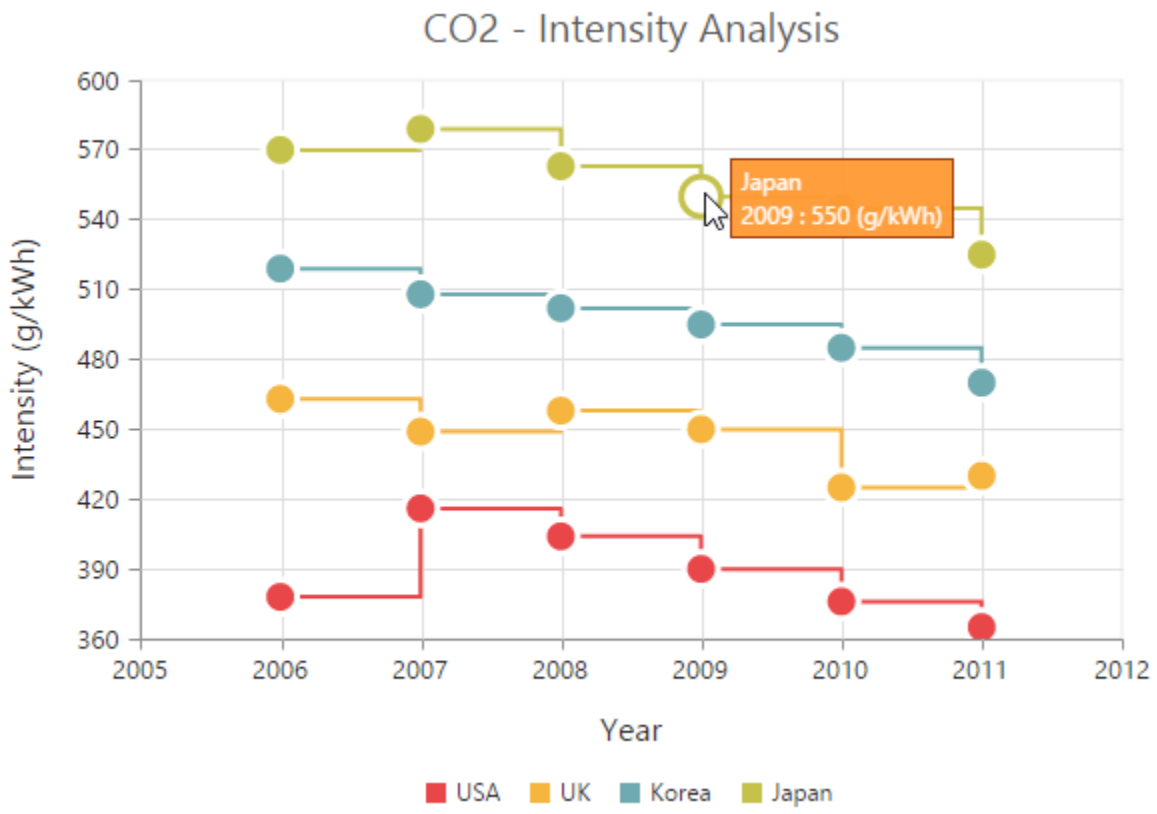
```
//...
)
```

Customize the appearance of tooltip

The *Fill* and *Border* options are used to customize the background color and border of the tooltip respectively. The *Font* option in the tooltip is used to customize the font of the tooltip text.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(ser =>
{
ser.Tooltip(tooltip=>tooltip
//Change tooltip color and border
.Fill("#FF9933")
.Border(br => br.Color("#993300").Width(1))
// ...
).Add();
})
//...
)
```

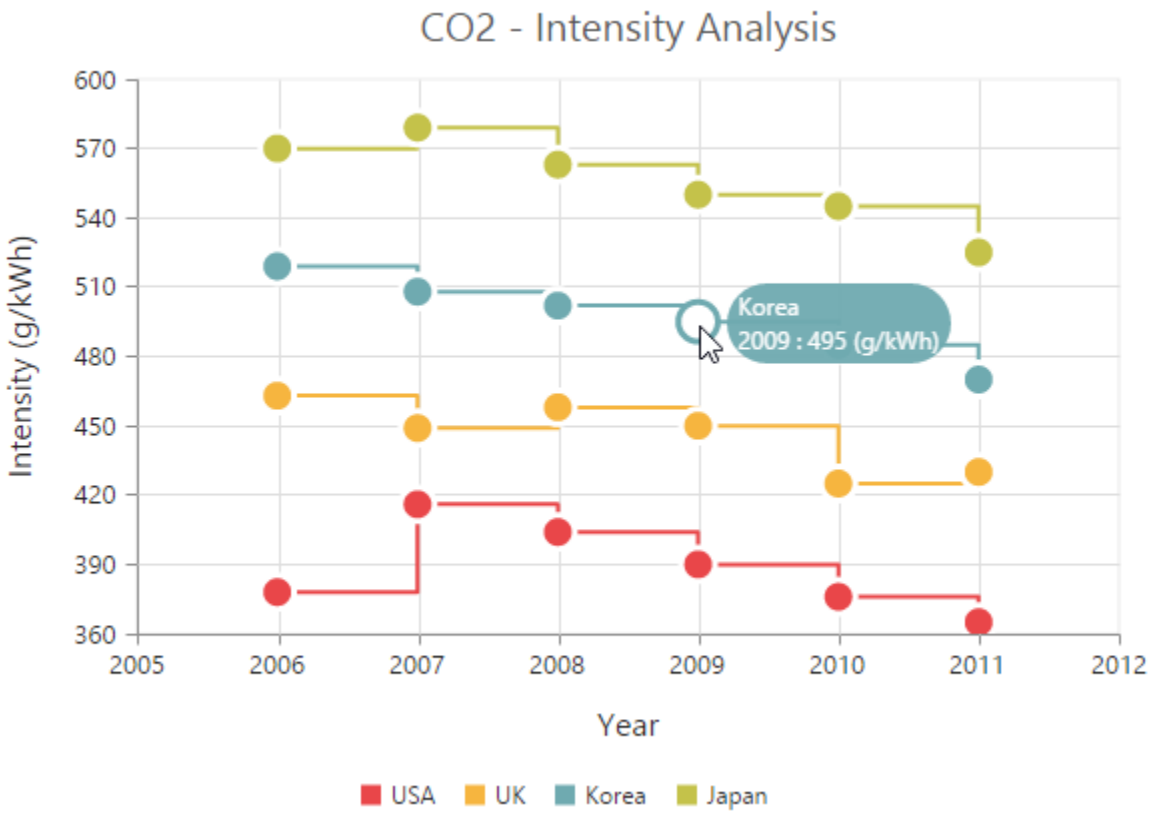


Tooltip with rounded corners

The options *RX* and *RY* are used to customize the corner radius of the tooltip rectangle.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(ser =>
{
ser.Tooltip(tooltip=>tooltip
//Customize the corner radius of the tooltip rectangle.
.RX(50).RY(50)
// ...
).Add();
})
//...
)
```



Zooming and Panning

Enable Zooming

There are two ways you can zoom the chart,

- When the **Zooming.Enable** option is set to true, you can zoom the chart by using the rubber band selection.
- When the **Zooming.EnableMouseWheel** option is set to true, you can zoom the chart on mouse wheel scrolling.

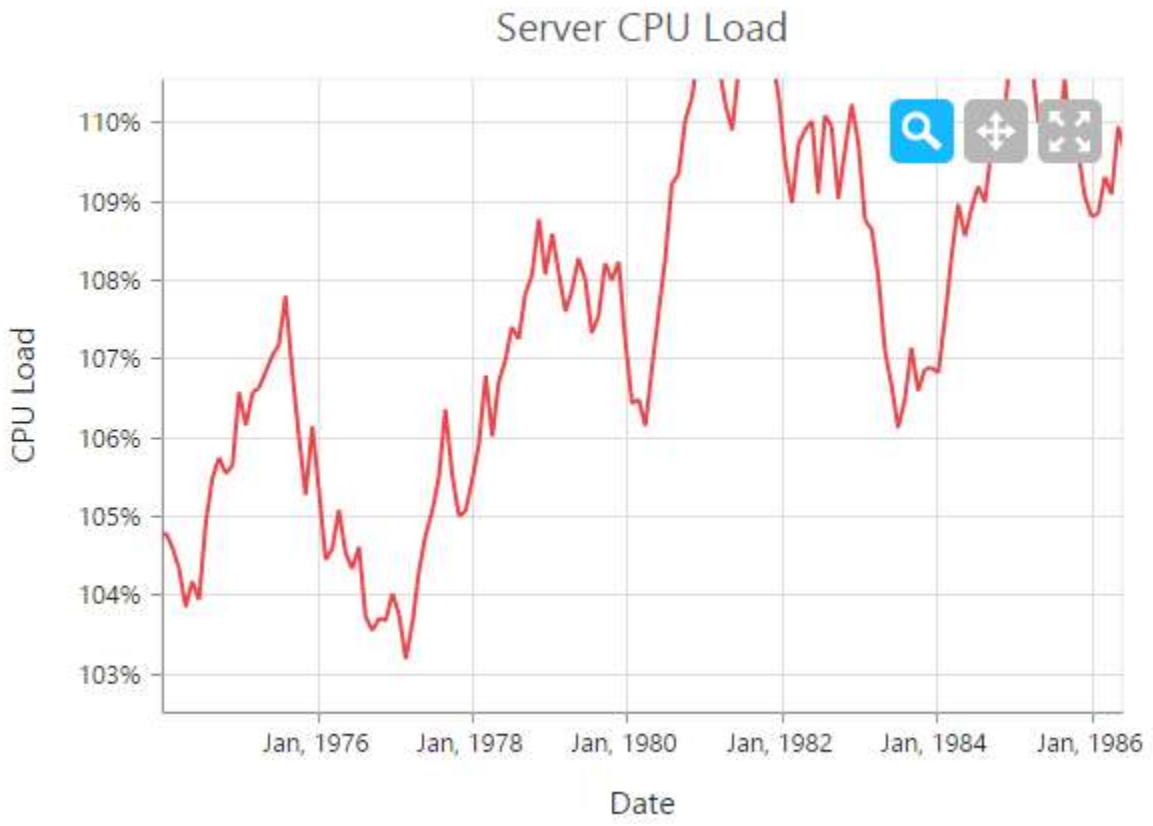
Copyright © 2001 - 2020 Syncfusion Inc.

- When **Zooming.EnablePinching** option is set to *true*, you can zoom the chart through pinch gesture.

Note: Pinch zooming is supported only in browsers that support multi-touch gestures. Currently IE10, IE11, Chrome and Opera browsers support multi-touch in desktop devices.

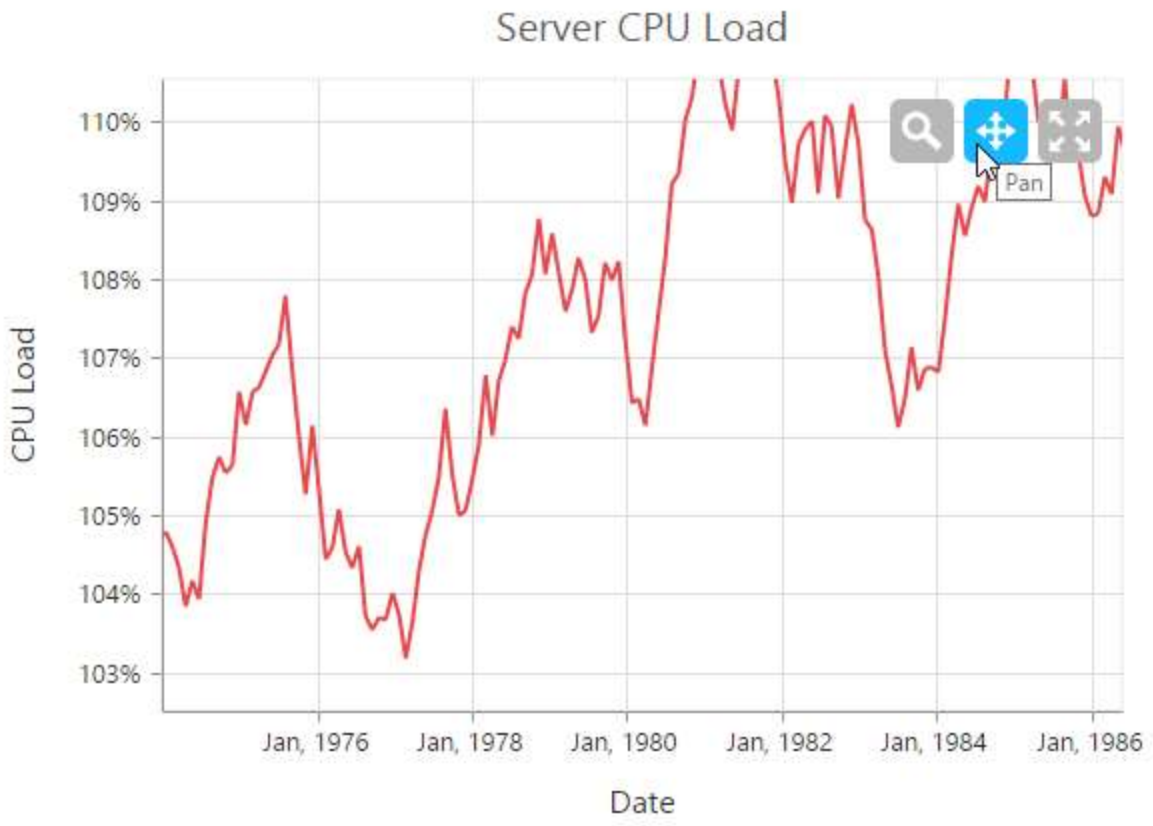
ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Zooming(zoom=>zoom
//Enable zooming in chart
.Enable(true)
)
//...
)
```



After zooming the chart, a zooming toolbar will appear with options to *zoom*, *pan* and *reset*. Selecting the Pan option will allow to pan the chart and selecting the Reset option will reset the zoomed chart.

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Zooming and Panning online demo sample.

Types of zooming

The *Type* option in zooming specifies whether the chart is allowed to scale along the horizontal axis(X) or vertical axis(Y) or along both axis(X,Y). The default value of the Type is **"X,Y"** (both axis).

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Zooming(zoom=>zoom
.Enable(true)
//Enable horizontal zooming
.Type("x")
)
//...
)
```

Customizing zooming toolbar

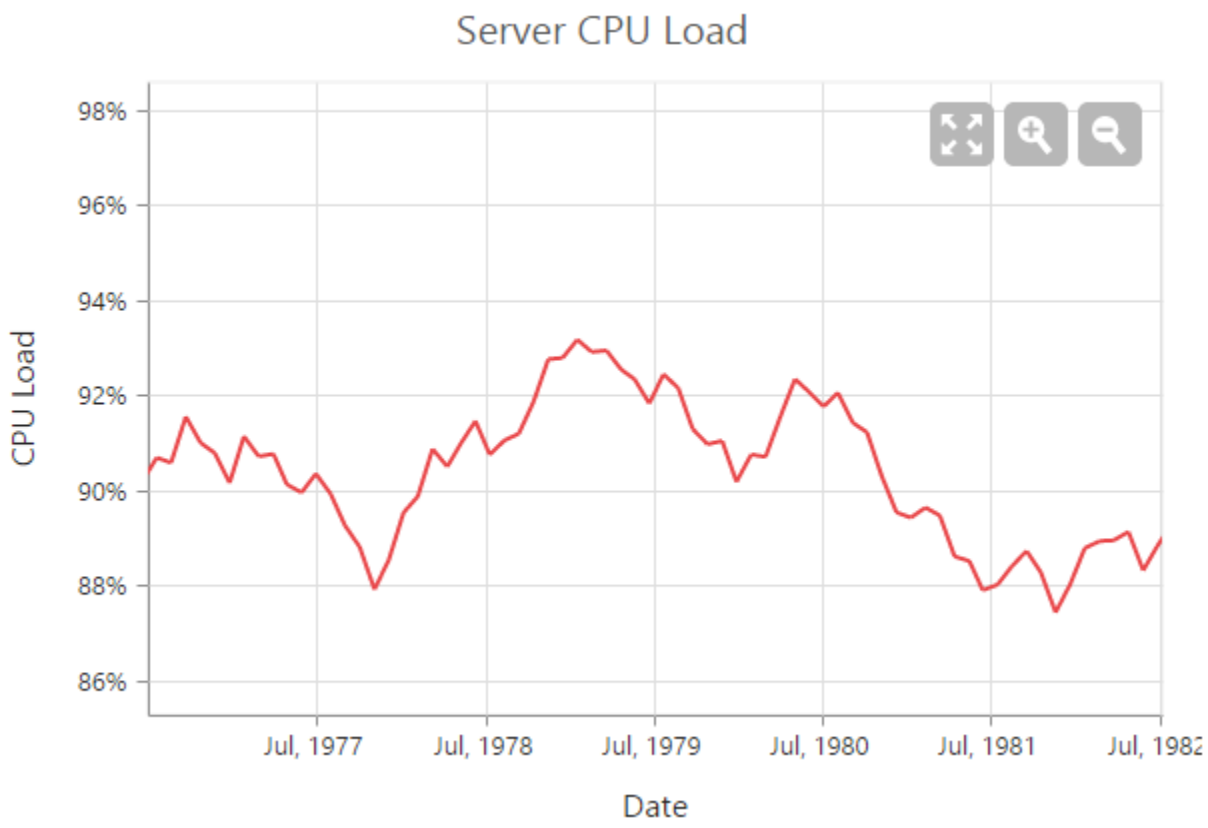
The user can choose the items displayed in the zooming toolbar by specifying the property **ToolBarItems**.

ASPX-CS

```
@(Html.EJ().Chart("container")
//Customizing zooming toolbar
.Zooming(
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
zoom => zoom.Enable(true).ToolbarItems (new List<string>()
{"reset", "zoomIn", "zoomOut"})
)
//...
)
```



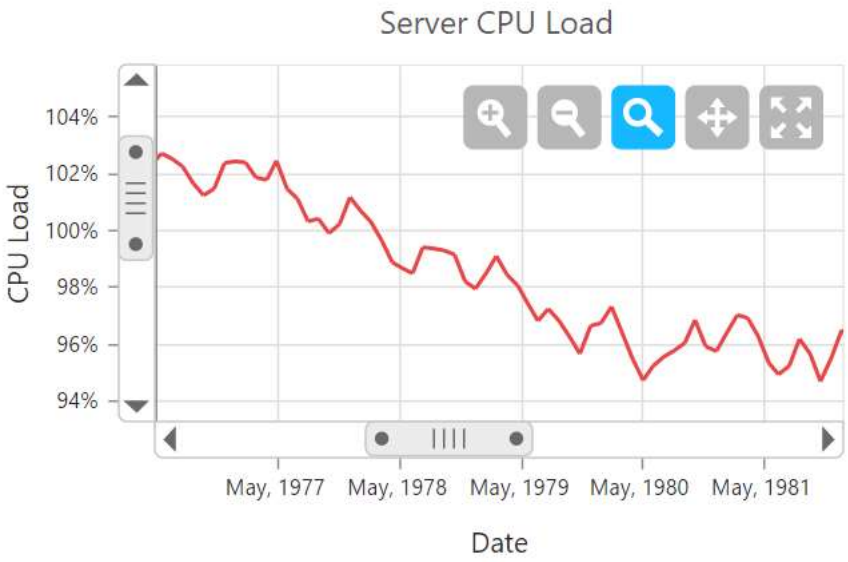
Enable ScrollBar

EjChart provides scrollbar support to view the other portions of chart area which is not shown in the view port when zoomed, by setting true to **EnableScrollbar** option in **Zooming**.

ASPX-CS

```
@(Html.EJ().Chart("container")
//Customizing zooming toolbar
.Zooming(
zoom => zoom.Enable(true)
.EnableScrollbar(true)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Crosshair

Crosshair is used to view the value of an axis at mouse position or touch contact point.

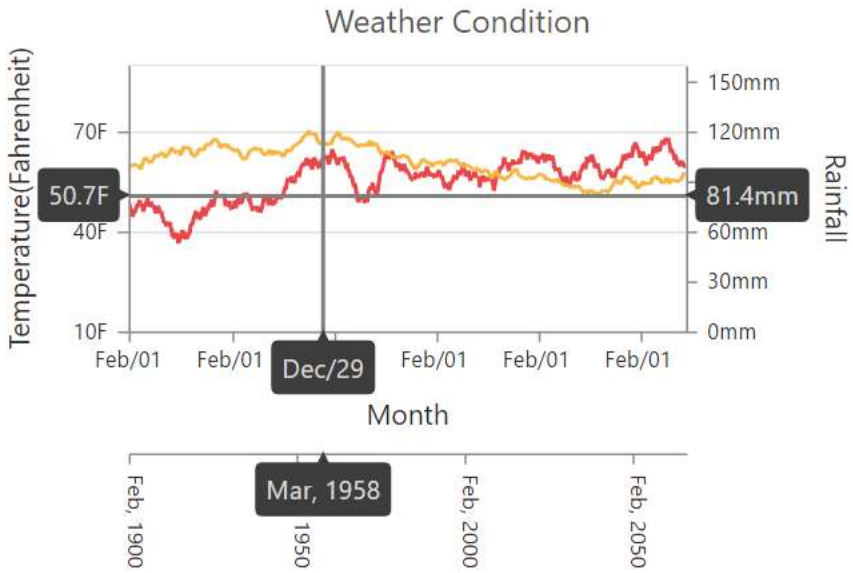
Enable Crosshair and Crosshair label

Crosshair can be enabled by using the **Visible** option in the Crosshair. Crosshair label for an axis can be enabled by using the Visible option of CrosshairLabel in the corresponding axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.PrimaryXAxis(axis=>axis.CrosshairLabel(label=>label
//Enable crosshairLabel to X-Axis
.Visible(true))
)
.PrimaryYAxis(axis=>axis.CrosshairLabel(label=>label
//Enable crosshairLabel to Y-Axis
.Visible(true))
)
//Initializing Crosshair
.Crosshair(crosshair=>crosshair.Visible(true))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



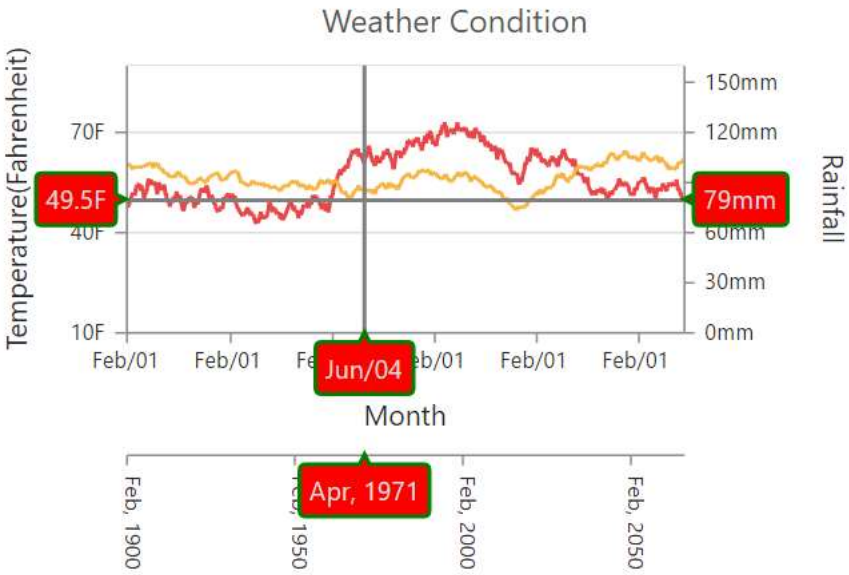
Click here to view the Crosshair online demo sample.

Customize the Crosshair line and Crosshair label

The *Fill* and *Border* options of the **CrosshairLabel** is used to customize the background color and border of the crosshair label respectively. Color and width of the crosshair line can be customized by using the **Line** option in the Crosshair.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.PrimaryYAxis(axis=>axis.CrosshairLabel(label=>label
//Customizing the crosshair label background color and border
.Fill("red").Border(br=>br.Color("green").Width(2))
))
//Initializing Crosshair
.Crosshair(crosshair=>crosshair
.Visible(true)
//Customizing the crosshair line
.Line(line=>line.Color("grey").Width(2))
)
//...
)
```



Trackball

Trackball is used to track a data point close to the mouse position or touch contact point. Trackball marker indicates the closest point and trackball tooltip displays the information about the point.

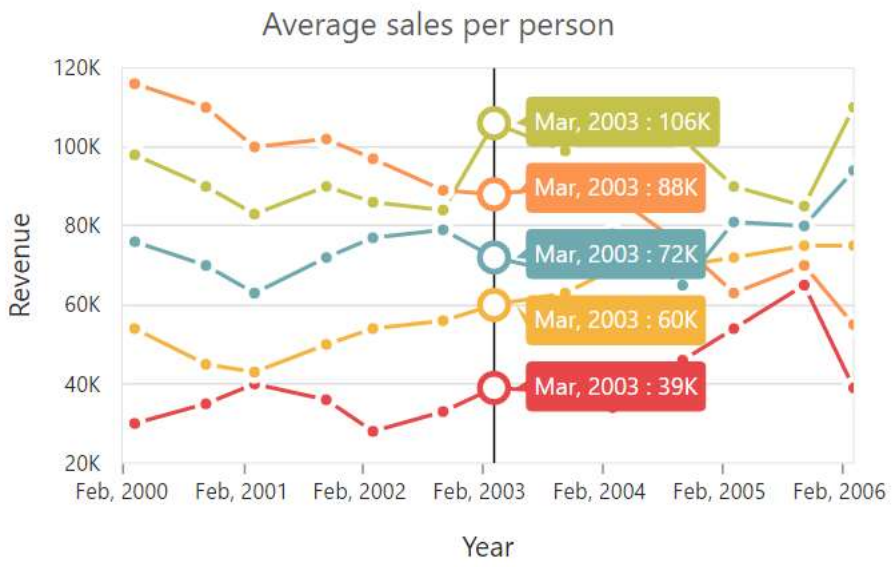
Enable Trackball

Trackball can be enabled by setting the *Visible* option of the crosshair to *true* and then set the **Type** as *Trackball*. The default value of type is *Crosshair*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
//Initializing Crosshair
.Crosshair(crosshair=>crosshair
.Visible(true)
//Change crosshair type to trackball
.Type(CrosshairType.Trackball)
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Trackball online demo sample.

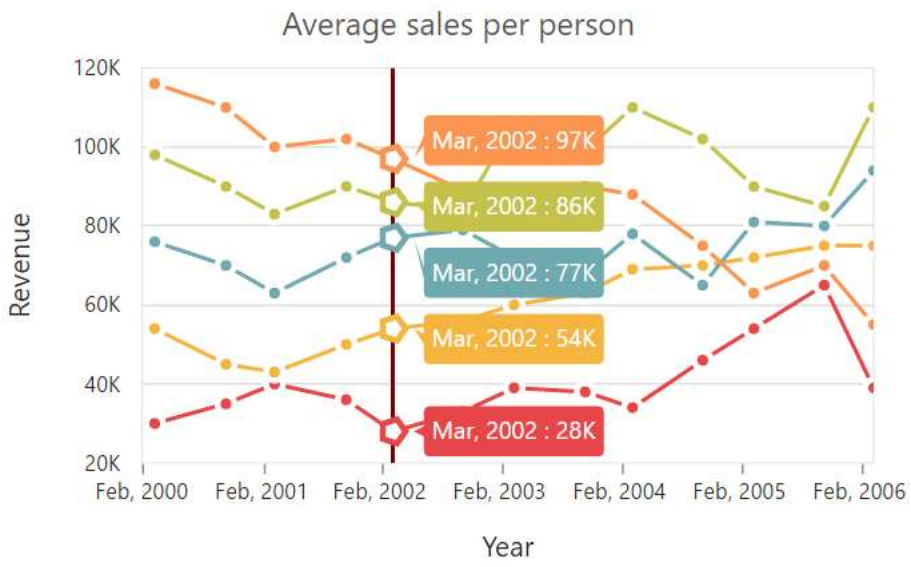
Customize trackball marker and trackball line

Shape and size of the trackball marker can be customized by using the *Shape* and *Size* options of the Crosshair marker. Color and width of the trackball line can be customized by using the *Line* option in the crosshair.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
//Initializing Crosshair
.Crosshair(crosshair=>crosshair
.Visible(true)
//Customize the trackball line color and width
.Line(line=>line.Color("#800000").Width(2))
//Customize the trackball marker shape size and visibility
.Marker(mr=>mr
//Enable/disable trackball marker
.Visible(true).Shape(ChartShape.Pentagon).Size(size=>size.Width(9).Height(9)
)
)
//...
)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



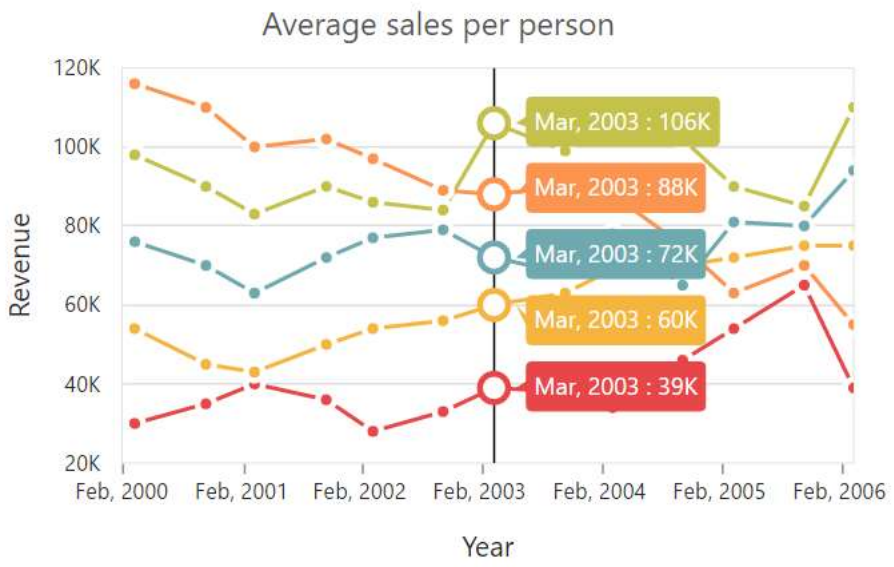
Format Trackball tooltip

X and Y values displayed in the trackball tooltip are formatted based on its axis *LabelFormat*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
//Add format to crosshair label
.PrimaryXAxis(axis=>axis.LabelFormat("MMM, yyyy"))
.PrimaryYAxis(axis=>axis.LabelFormat("{value}K"))
//Initializing Crosshair
.Crosshair(crosshair=>crosshair
.Visible(true)
//...
)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

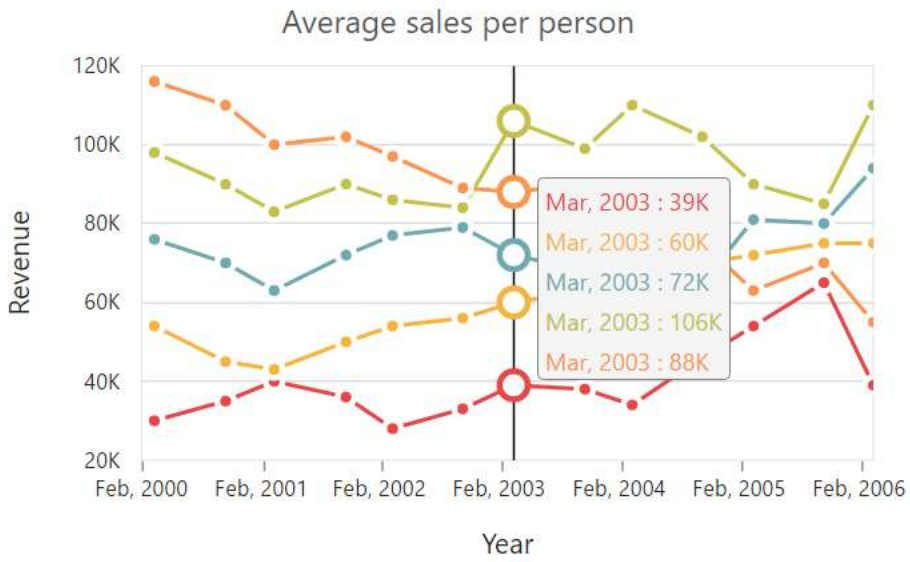


You can able to show the trackball tooltip in two modes, using trackballTooltipSettings.

1. Grouping. 2. Float.

JAVASCRIPT

```
@(Html.EJ().Chart("chartContainer")
//  ..
.Crosshair(crosshair =>
crosshair.Visible(true).type(CrosshairType.Trackball)
.TrackballTooltipSettings(trackball => trackball.Rx("3")
.Ry("3").Fill("whitesmoke").Mode(TrackballDisplayMode.Grouping)
.Border(br => br.Width("1").Color("grey")))
//  ..
)
```



Highlight

EjChart provides highlighting support for the series and data points on mouse hover. To enable the highlighting option, set the *Enable* property to *true* in the **HighlightSettings** of the series.

Note: When hovering mouse on the data points, the corresponding series legend also will be highlighted.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
// enable the highlight settings
.Enable(true)).Add();
})
)
```

Click here to view the highlight and selections online demo sample.

Highlight Mode

You can set three different highlight mode for the highlighting data point and series by using the **Mode** property of the *HighlightSettings*.

- Series
- Points
- Cluster

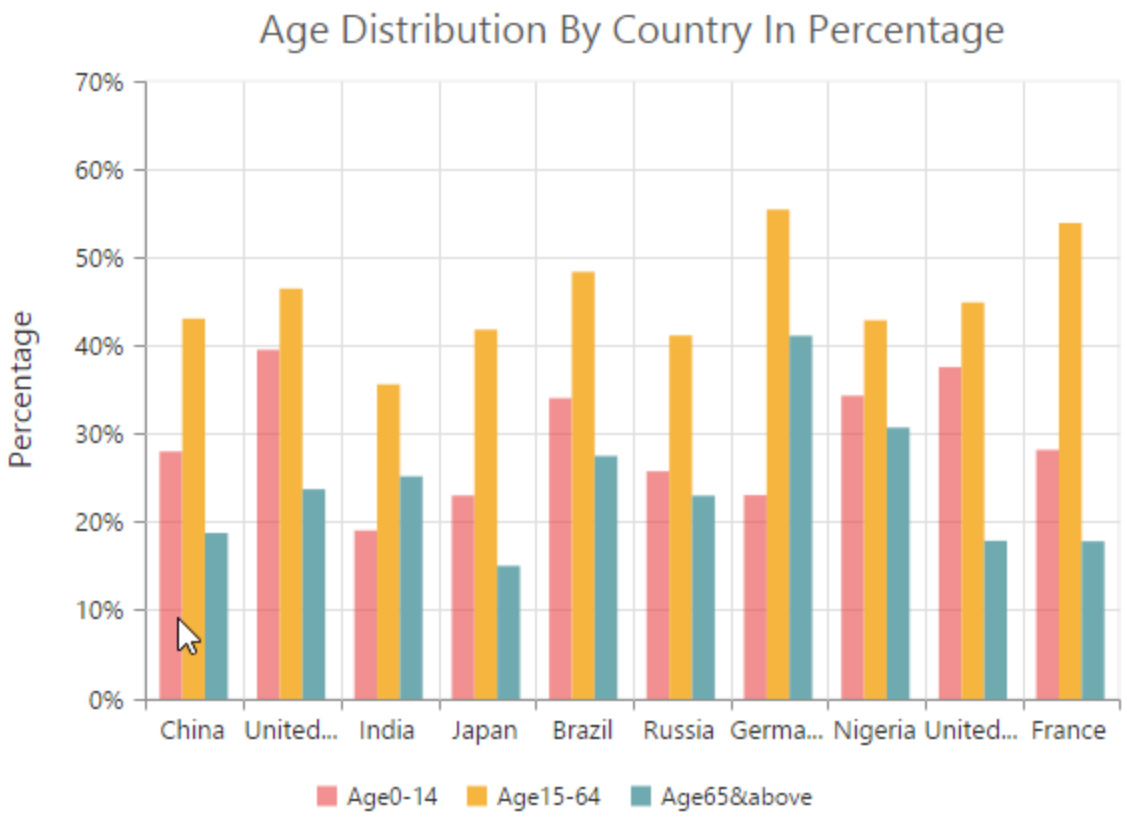
Series mode

To highlight all the data points of the specified series, you can set the **Series** value to the *Mode* option in the HighlightSettings.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
//Change highlight mode
.Mode(ChartMode.Series)
).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



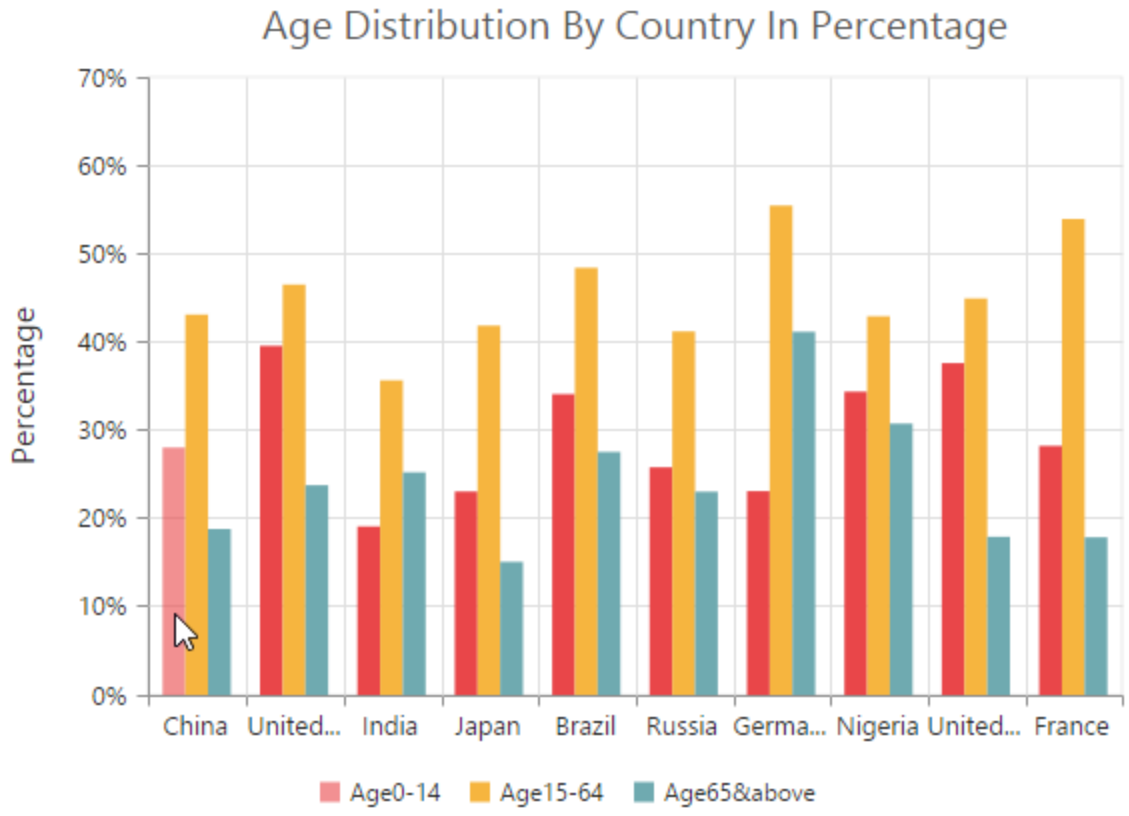
Point mode

To highlight a single point, you can set the **Point** value to the *Mode* option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
//Change highlight mode
.Mode(ChartMode.Point)
).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



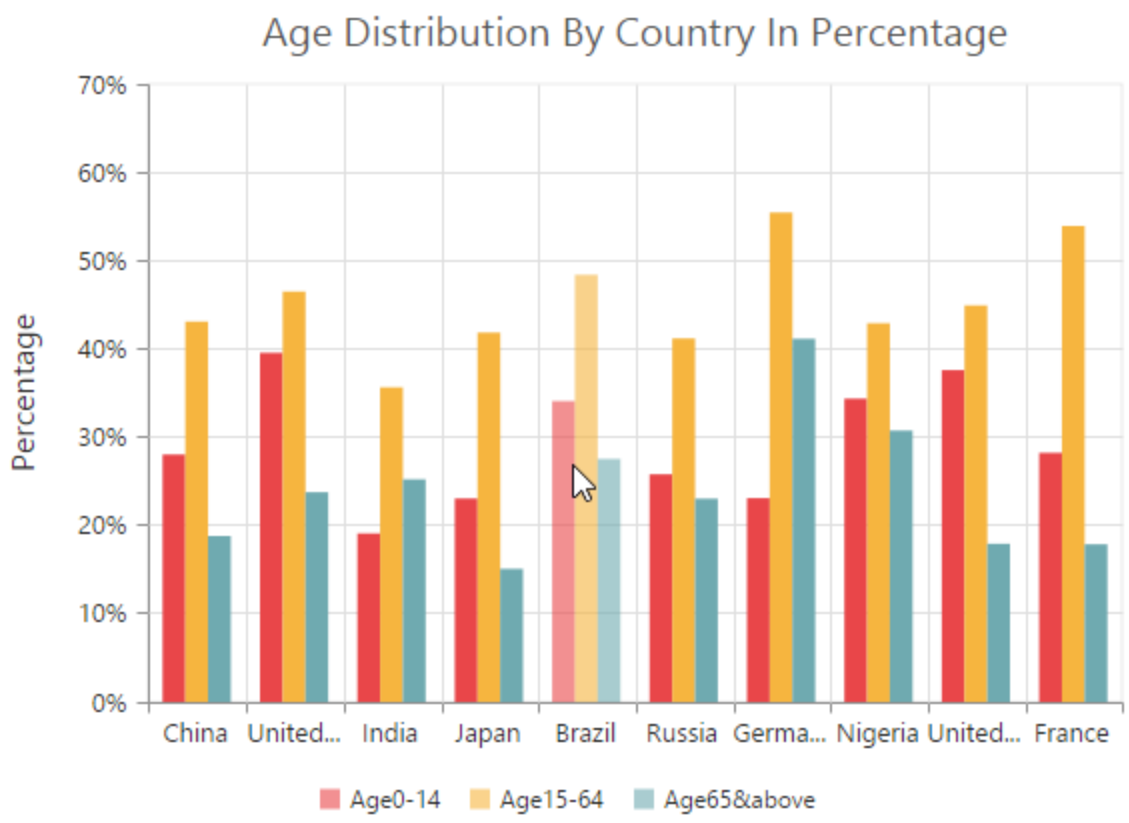
Cluster mode

To highlight the points that corresponds to the same index in all the series, set the **Cluster** value to the *Mode* option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
//Change highlight mode
.Mode(ChartMode.Cluster)
).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



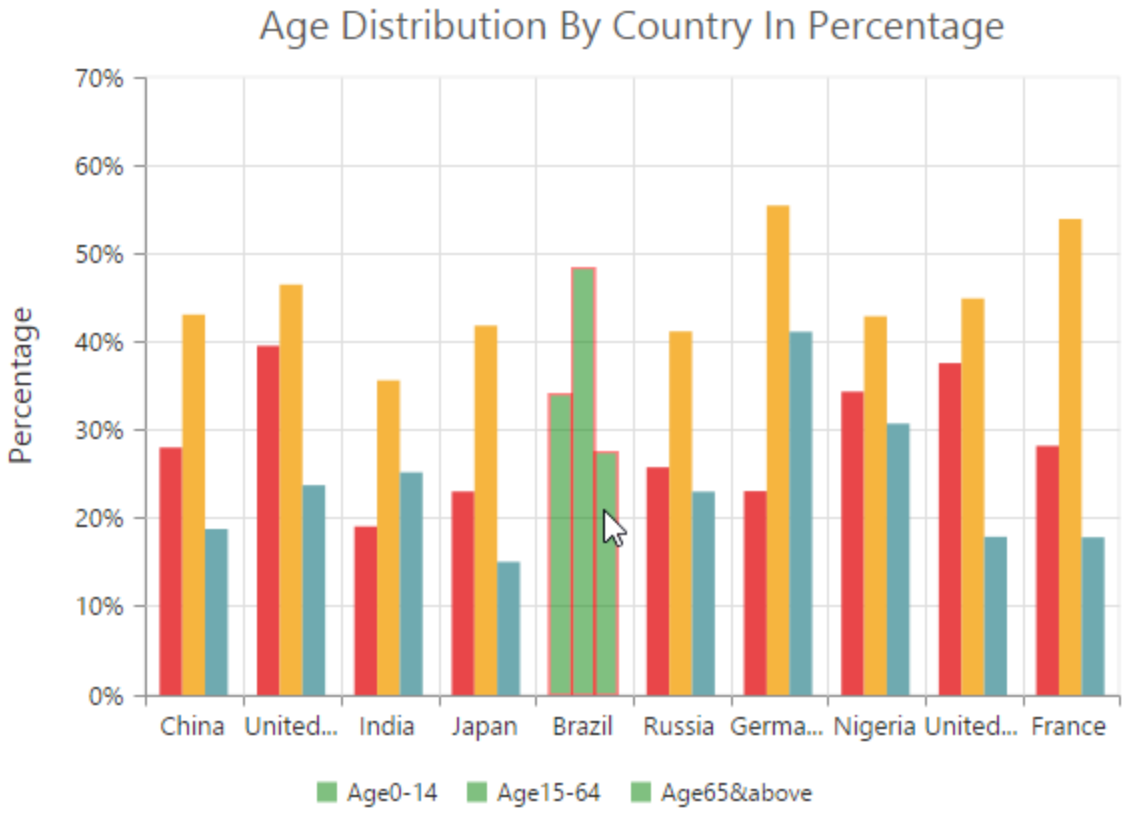
Customize the highlight styles

To customize the highlighted series, use the *Color*, *Border* and *Opacity* options in the HighlightSettings.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
//Customizing styles
.Border(br=>br.Color("red").Width(1.5))
.Opacity(0.5).Color("green")
).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



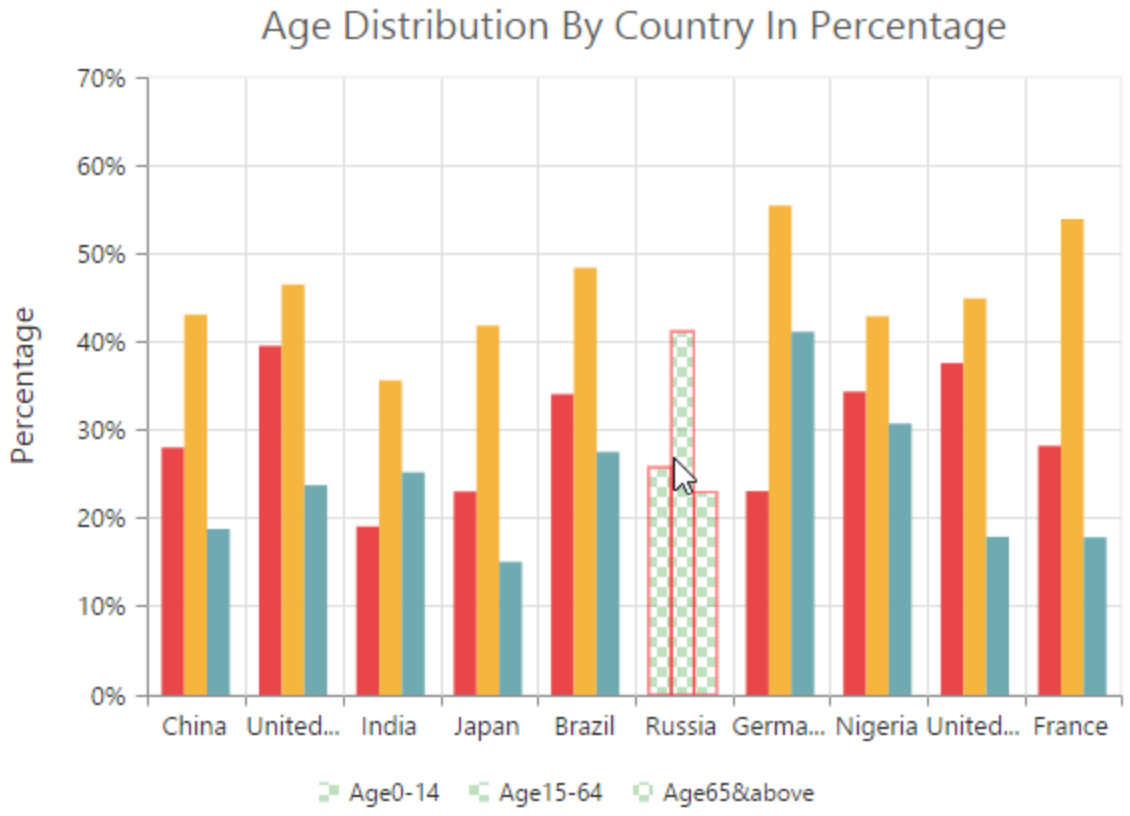
Patterns to highlight

EjChart provides pattern support for highlighting the data by setting the value to the **Pattern** property of the HighlightSettings. The different types of highlight patterns are as follows.

1. Chessboard 2. Crosshatch 3. Dots 4. Pacman 5. Grid 6. Turquoise 7. Star 8. Triangle 9. Circle 10. Tile 11. HorizontalDash 12. VerticalDash 13. Rectangle 14. Box 15. VerticalStripe 16. HorizontalStripe 17. Bubble 18. DiagonalBackward 19. DiagonalForward

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
.Pattern(ChartPattern.Chessboard)
).Add();
})
)
```



Custom pattern

To create a custom pattern for the highlighting data points, set the pattern type as **Custom** and add the custom pattern *Id* in the **CustomPattern** option of the HighlightSettings.

ASPX-CS

```
<svg>
<pattern id="dots_a" patternUnits="userSpaceOnUse" width="6" height="6">
<rect x="0" y="0" width="6" height="6" transform="translate(0,0)"
fill="black" opacity="1"></rect>
<path d='M 3 -3 L -3 3 M 0 6 L 6 0 M 9 3 L 3 9'stroke-width="1"
stroke="white"></path>
</pattern>
</svg>
@(Html.EJ().Chart("chartContainer")
//...
// enable the highlight settings
.Series(sr =>
{
sr
.HighlightSettings(highlight => highlight
//Add custom pattern for highlighting data
.Pattern(ChartPattern.Custom)
.CustomPattern("dots_a")
//...
.Enable(true)
).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```

Selection

EjChart provides selection support for the series and data points on mouse click. To enable the selection option, set the *Enable* property to *true* in the SelectionSettings* of the series.

Note: When mouse is clicked on the data points, the corresponding series legend also will be selected.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
// enable the selection settings
.Enable(true)
).Add();
})
)
```

Click here to view the highlight and selections online demo sample.

Selection Mode

You can set four different selection mode for highlighting the data point and series by using the **Mode** property of the SelectionSettings.

- Series
- Points
- Cluster
- Range

Series mode

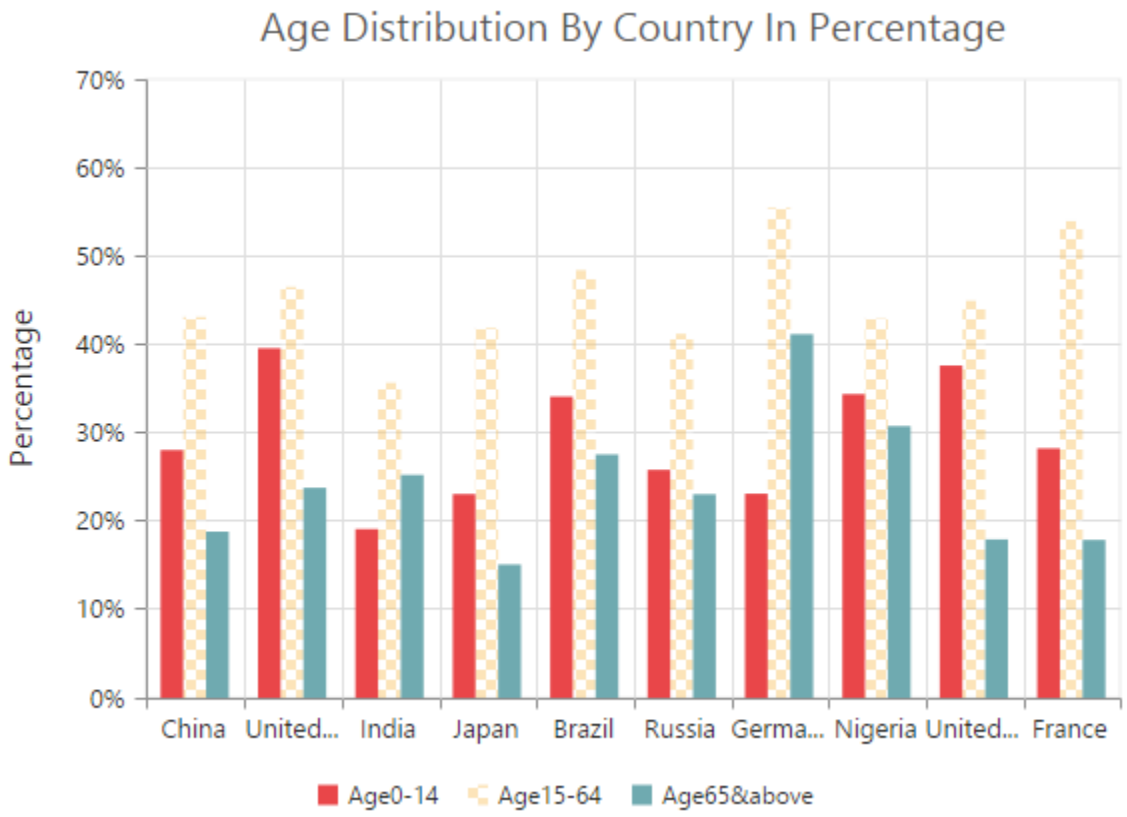
To select all the data points of the specified series, you can set the **Series** value to the *Mode* option in the SelectionSettings.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Change selection mode
.Mode(ChartMode.Series)
.Pattern(ChartPattern.Chessboard)
.Enable(true)
).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```

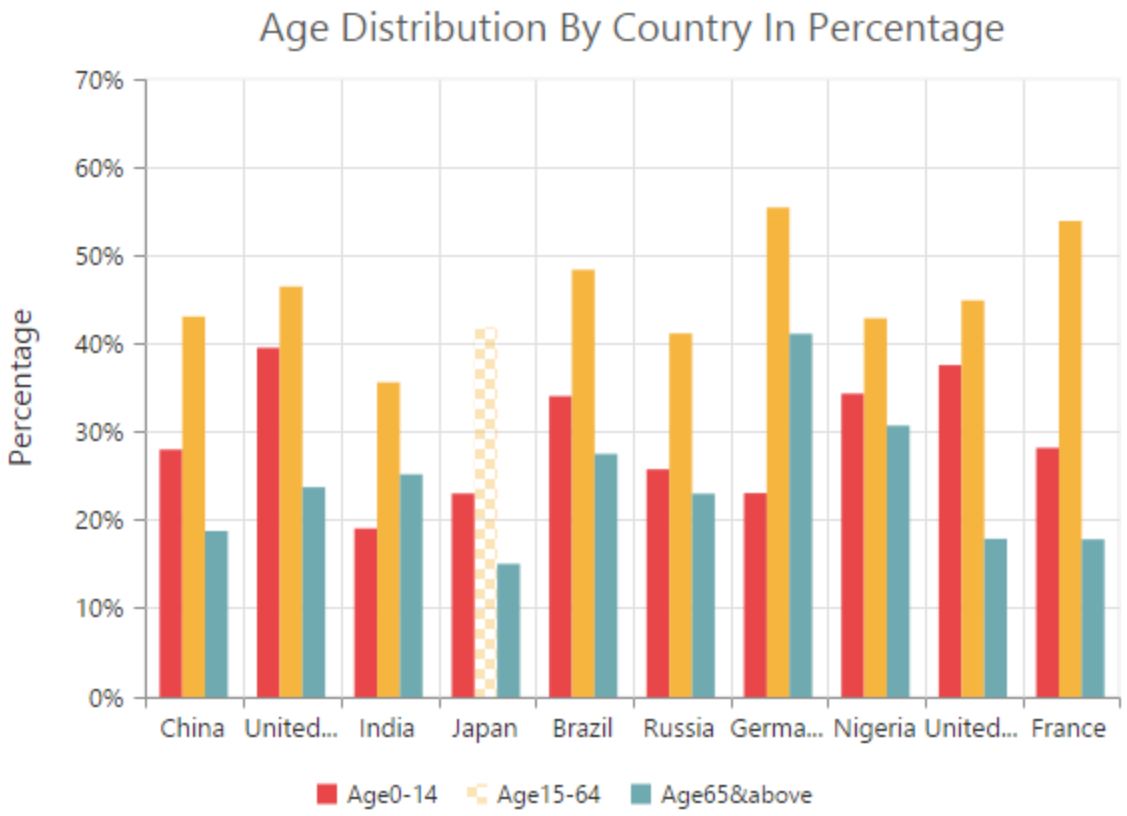


Point mode

To highlight a single point, you can set the **Point** value to the *Mode* option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Change selection mode
.Mode(ChartMode.Point)
//...
.Enable(true)
).Add();
})
)
```



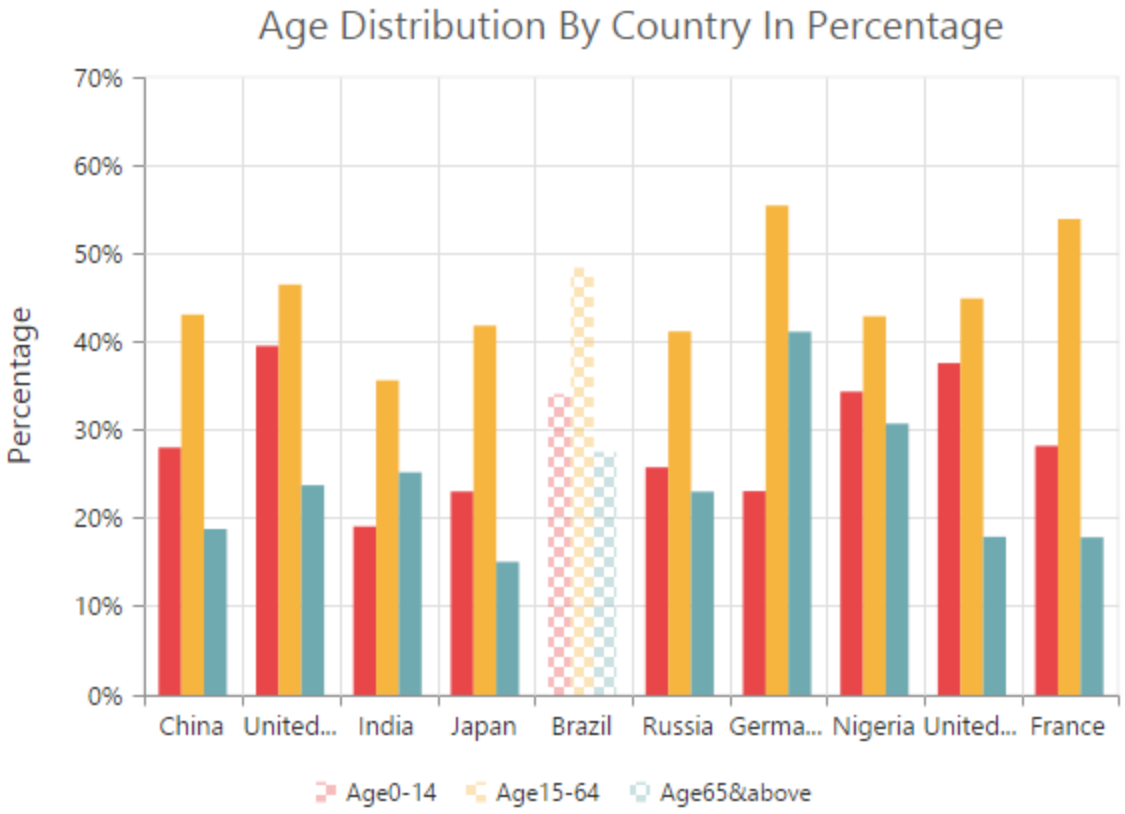
Cluster mode

To select the points that corresponds to the same index in all the series, set the **Cluster** value to the *Mode* option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Change selection mode
.Mode(ChartMode.Cluster)
//...
.Enable(true)
).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Range mode

To fetch the selected area data points value, you can set the selectionSettings Mode as **Range** in the chart series. The selection rectangle can be drawn as horizontally, vertically or in both direction by using **RangeType** property and the selected data's are returned as an array collection in the **RangeSelected** event.

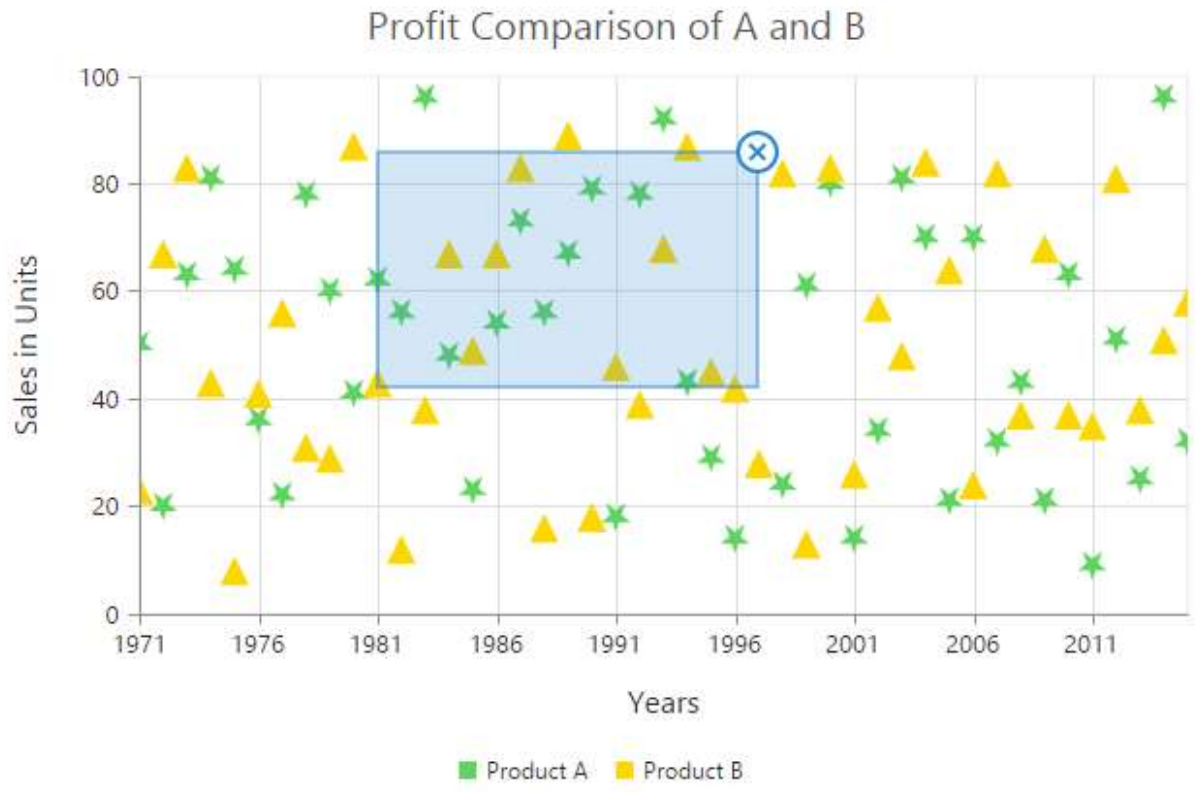
ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Change selection mode
.Mode(ChartMode.Range)
.RangeType(MultiSelectType.XY)
//...
.Enable(true)
).Add();
})
//Subscribe the rangeSelected event.
.RangeSelected("rangeSelection")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
//event to fetch the selected data point values
rangeSelection:function (sender){
var selectedData = sender.data.selectedDataCollection;
//...
}
```



Click here to view the Multiple data selection online demo sample.

Selection Type

You can set two different selection type for selecting the data point and series on mouse click by using the **Type** property of the SelectionSettings.

- Single
- Multiple

Single Type

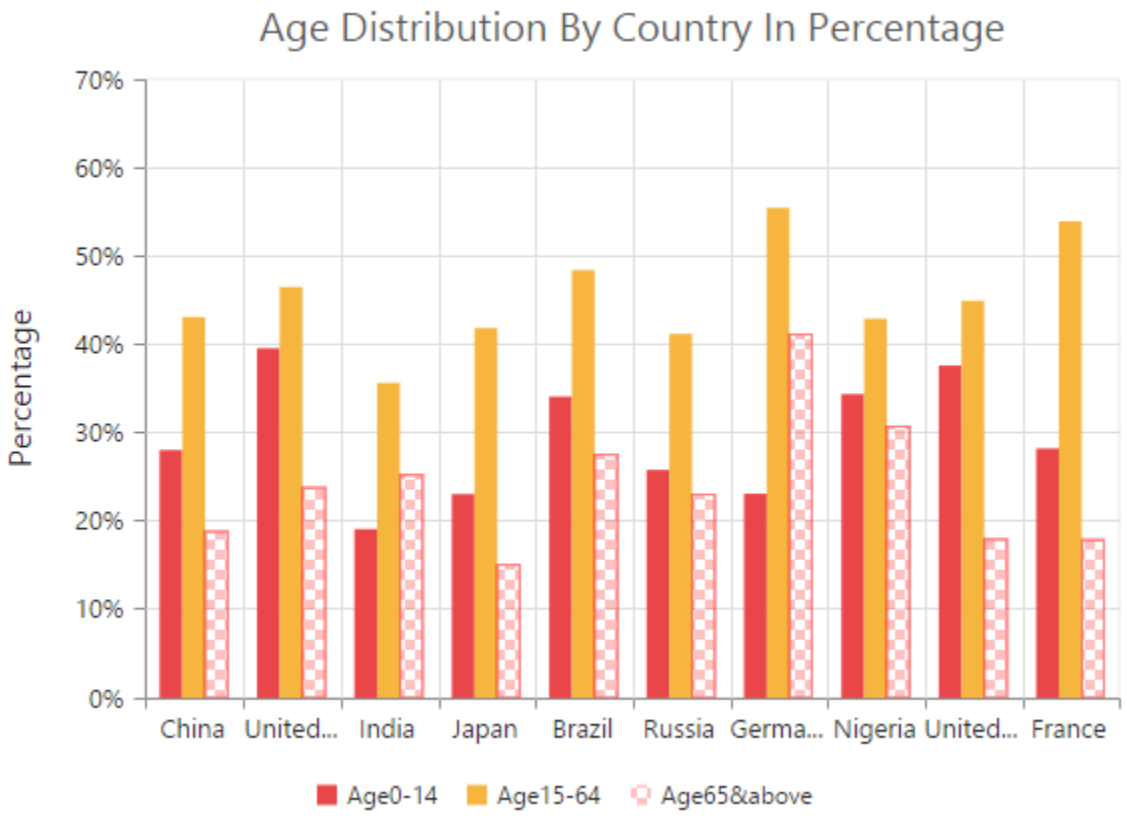
To select a data point or a series on mouse click based on the **SelectionSettings.Mode**, set **SelectionSettings.Type** as **Single** in the series.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.CommonSeriesOptions(
cm => cm.SelectionSettings(
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
selection => selection.Enable(true)
//Selection mode is series or point ,cluster
.Mode(ChartMode.Series)
//Selection type is single
.Type(ChartSelectionType.Single))
)
//...
)
```



Multiple Type

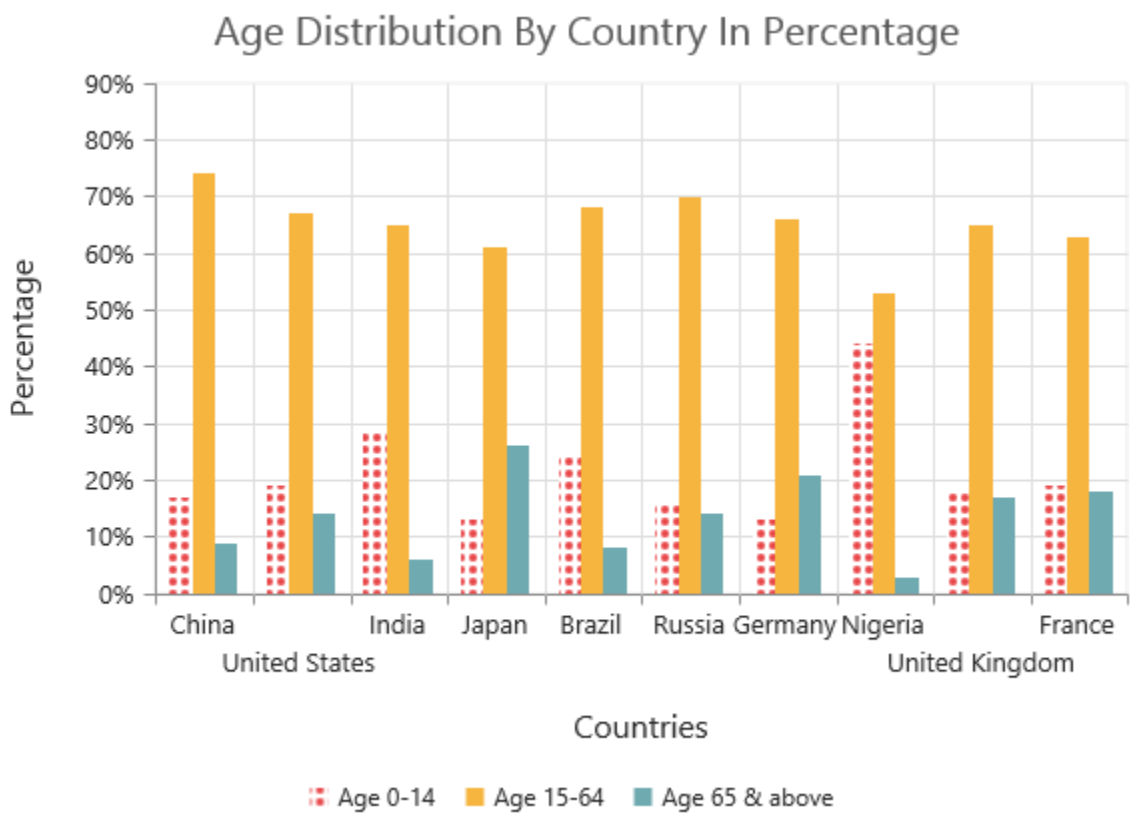
For selecting multiple data points or series on mouse click, set **SelectionSettings.Type** as **Multiple** in the series.

ASPX-CS

```
@(Html.EJ().Chart("container")
//...
.CommonSeriesOptions(
cm => cm.SelectionSettings(
selection => selection.Enable(true)
//Selection mode is series or point, cluster
.Mode(ChartMode.Series)
//Selection type is multiple
.Type(ChartSelectionType.Multiple))
)
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

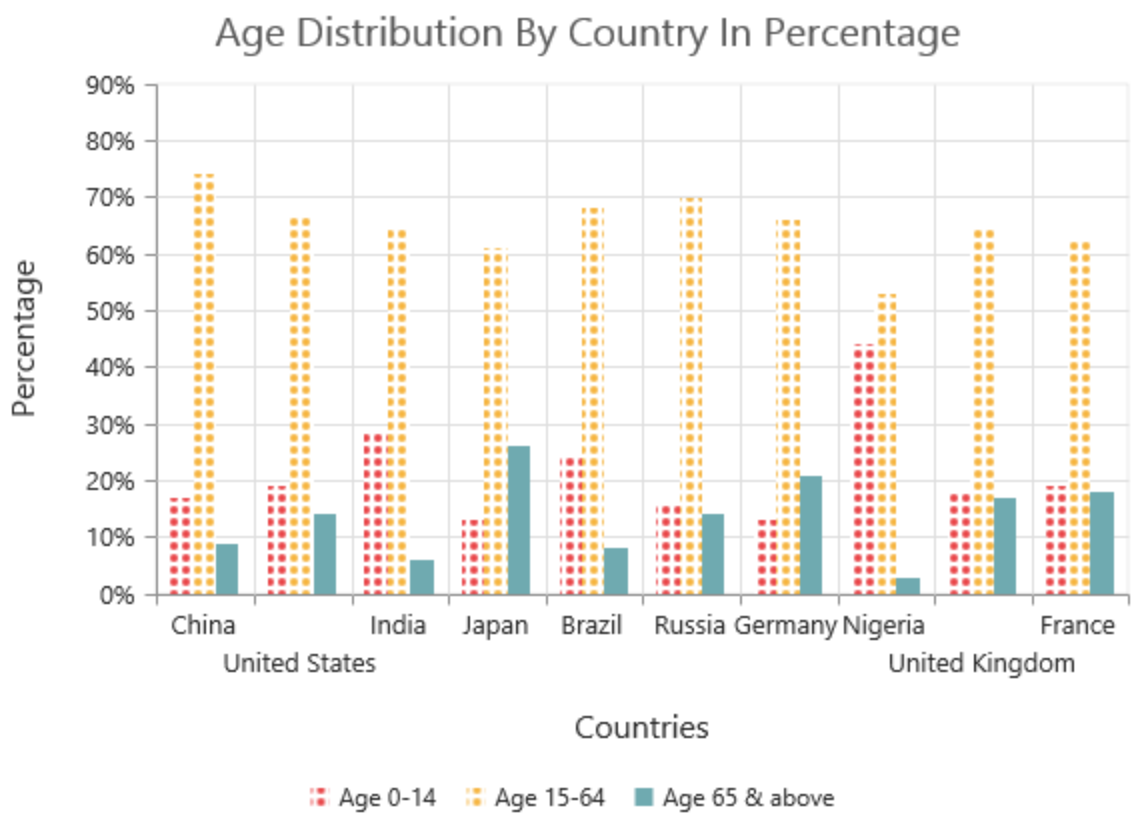


Customizing selection styles

To customize the selection styles, use the *Color*, *Border* and *Opacity* options in the SelectionSettings.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Customizing selection rectangle styles
.Border(br=>br.Width(1.5).Color("red"))
.Opacity(0.5).Color("red")
.Enable(true)
).Add();
})
)
```



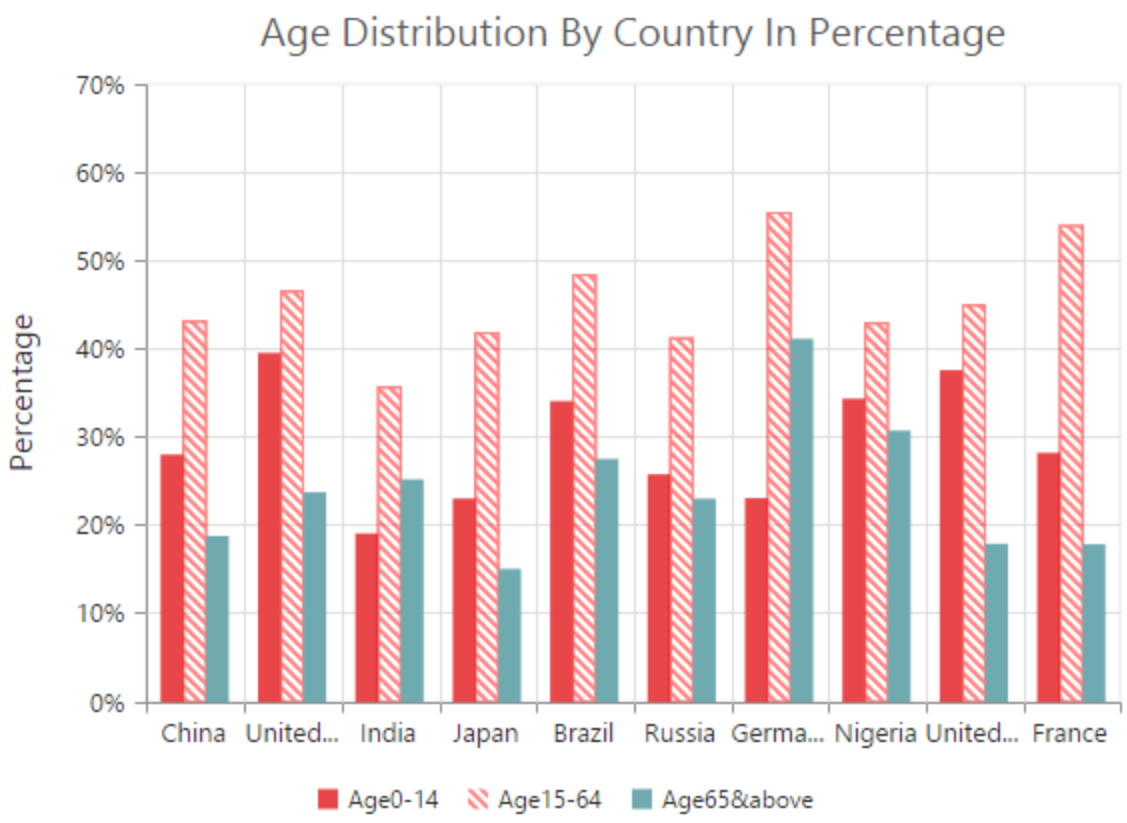
Patterns for selection

EjChart provides pattern support for the data selection by setting the value to the **Pattern** property of the SelectionSettings. The different types of selection patterns are as follows.

1. Chessboard 2. Crosshatch 3. Dots 4. Pacman 5. Grid 6. Turquoise 7. Star 8. Triangle 9. Circle 10. Tile 11. HorizontalDash 12. VerticalDash 13. Rectangle 14. Box 15. VerticalStripe 16. HorizontalStripe 17. Bubble 18. DiagonalBackward 19. DiagonalForward

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Change selection pattern
.Pattern(ChartPattern.DiagonalForward)
.Enable(true)
).Add();
})
)
```



Custom pattern

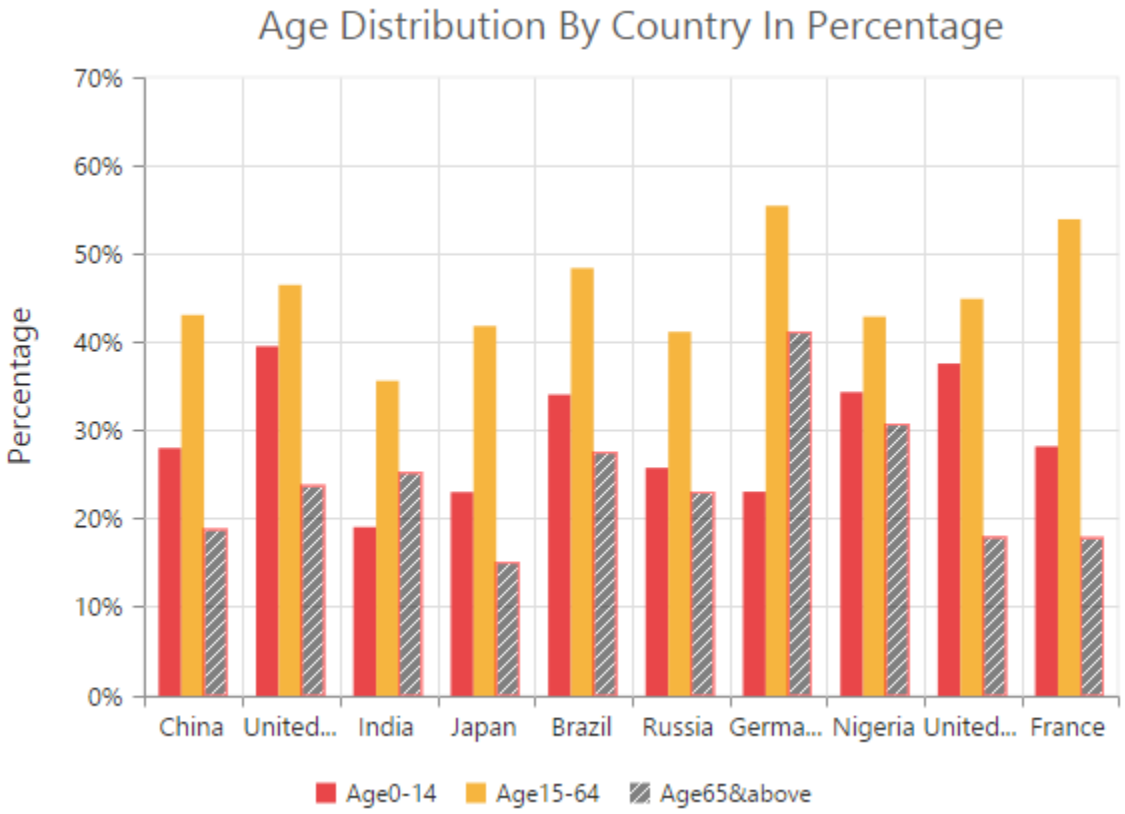
To create a custom pattern for selecting the data points, set the *Pattern* type as **Custom** and add the custom pattern *id* in the **CustomPattern** option of the SelectionSettings.

ASPX-CS

```
<div id="container"></div>
<svg>
<pattern id="dots_a" patternUnits="userSpaceOnUse" width="6" height="6">
<rect x="0" y="0" width="6" height="6" transform="translate(0,0)"
fill="black" opacity="1"></rect>
<path d='M 3 -3 L -3 3 M 0 6 L 6 0 M 9 3 L 3 9'stroke-width="1"
stroke="white"></path>
</pattern>
</svg>
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Add custom pattern for selection data
.Pattern(ChartPattern.Custom)
.CustomPattern("dots_a")
.Enable(true)
).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```



Handling Series Selection

To get the series information when selecting the specific series, subscribe to the **SeriesRegionClick** event and set the **SelectionSettings.Mode** as *Series*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
// enable the selection settings
.Series(sr =>
{
sr
.SelectionSettings(ss => ss
//Change selection mode
.Mode(ChartMode.Series)
//...
.Enable(true)
).Add();
})
//Subscribe series selection event
.SeriesRegionClick("seriesSelection")
)
```

JS

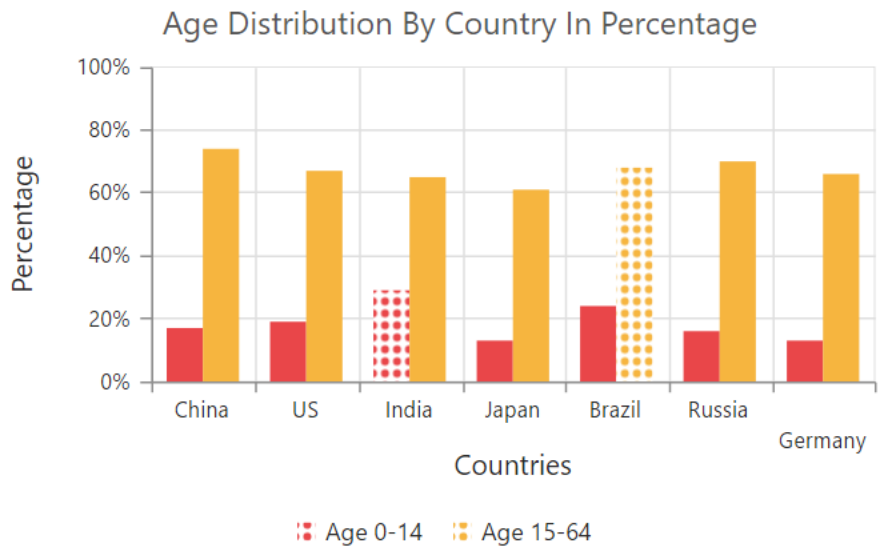
```
function seriesSelection(sender) {
//Get Series information on series selection
var seriesData = sender.series;
}
```

Selection on Load

We can able to select the point/series programmatically on chart load, by setting series and point index in the selectedDataPointIndexes property.

JAVASCRIPT

```
@(Html.EJ().Chart("chartContainer")
//.... . .
.SelectedDataPointIndexes(sr => {
//Adding Selected Data point indexes
sr.SeriesIndex("0").PointIndex("2")
sr.SeriesIndex("1").PointIndex("4")
//.... . .
}).Add();
//.... . .
})
)
```



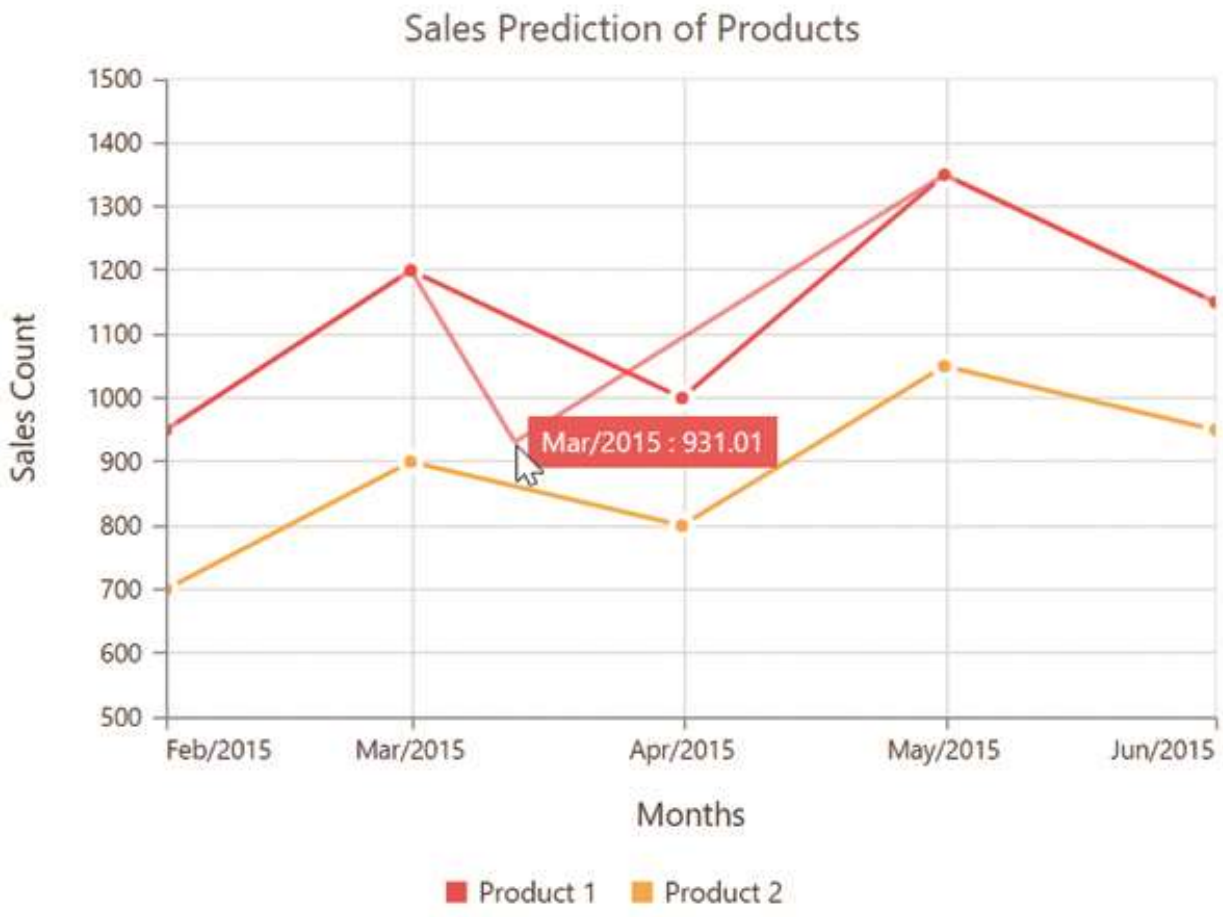
Data Editing

EjChart provides support to change the location of the rendered points. This can be done by dragging the point and dropping it on another location in chart. To enable the data editing, set the **enable** property to true in the *dragSettings* of the series.

ASPX-CS

```
@(Html.EJ().Chart("container")
.CommonSeriesOptions(options => options.DragSettings(drag =>
drag.Enable(true)))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the data editing online demo sample.

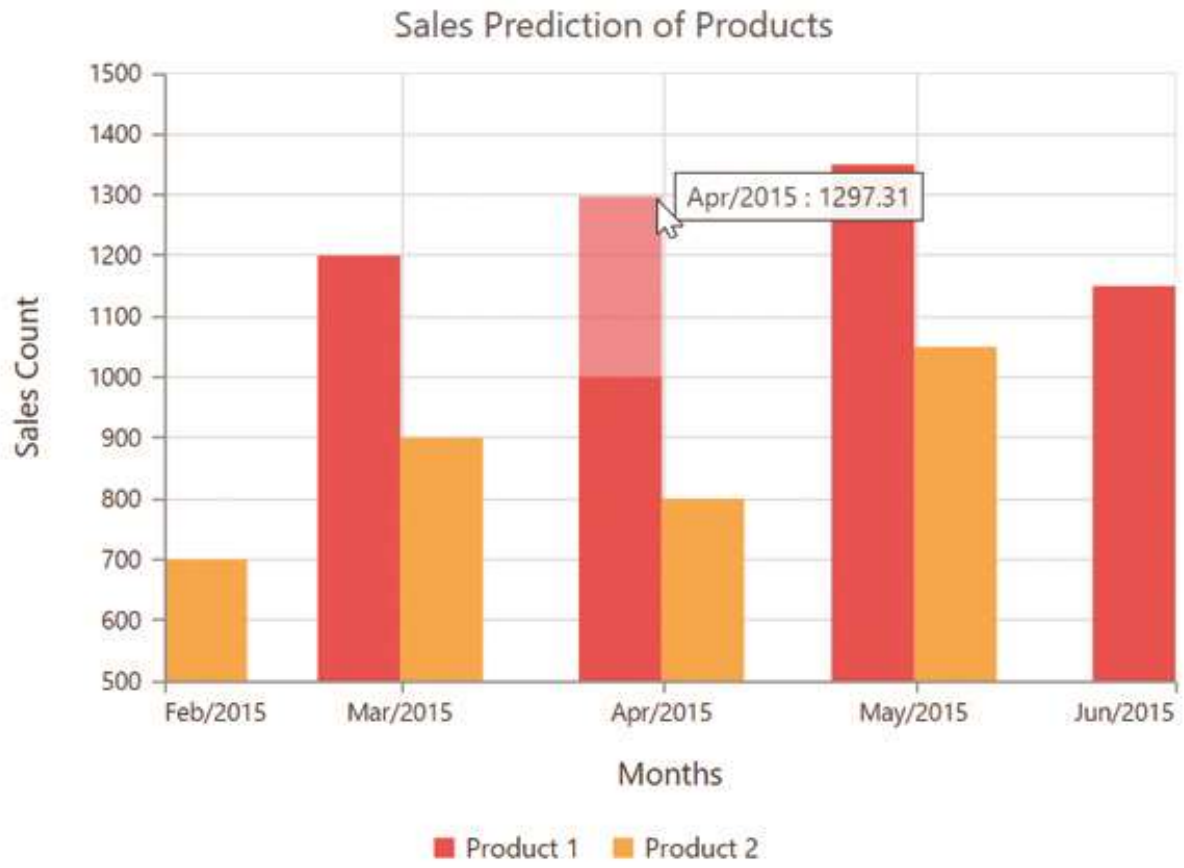
Customize Dragging direction

To drag the point along x and y axes, you can specify **type** as xy in *dragSettings*. And to drag along x axis alone, specify the type as x and to drag along y axis, specify type as y.

ASPX-CS

```
@(Html.EJ().Chart("container")
.CommonSeriesOptions(options => options.DragSettings(drag =>
drag.Type(DragType.Y)))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Performance

- When there are large number of points to load, you can enable canvas rendering mode in chart. Canvas rendering is faster than SVG because it does not involve manipulating DOM elements as much as SVG rendering.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Enable Canvas rendering mode
.EnableCanvasRendering(true)
//...
)
```

Click here to view the online demo sample that shows Chart performance with 100,000 data points.

- Instead of enabling data markers and labels when there are large number of data points, you can use **Trackball** and **Tooltip** to view point information.

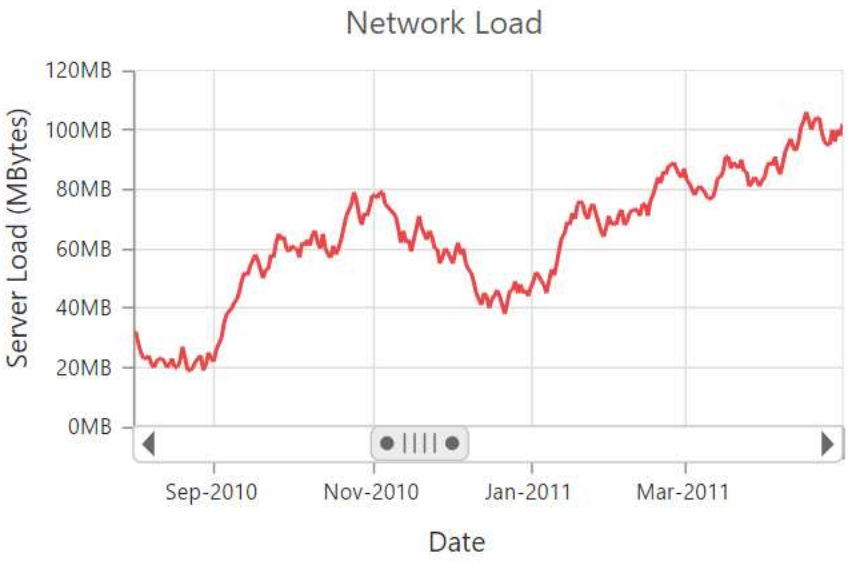
Copyright © 2001 - 2020 Syncfusion Inc.

Lazy Loading

Lazy loading feature provides an effective way for loading data on demand by scrolling and viewing a smaller range of data from a larger collection.

JAVASCRIPT

```
@(Html.EJ().Chart("chartContainer")
// Enable to zoom and scrollbar option
.PrimaryXAxis(axis => axis.ScrollbarSettings(scroll =>
scroll.Visible("true")
.CanResize("true")
.Range(range => range.Min("2009/1/1").Max("2014/1/1"))))
//  ..
)
```



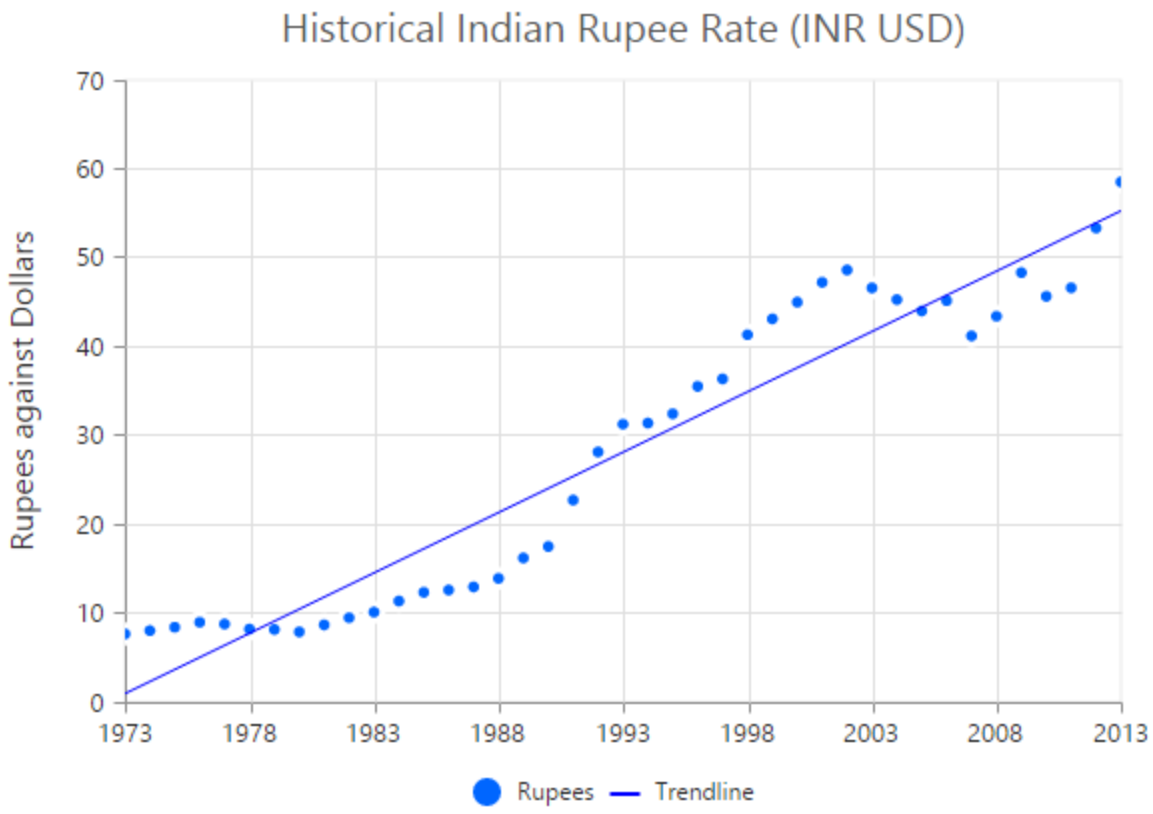
Trendlines

EjChart can generate Trendlines for Cartesian type series *(Line, Column, Scatter, Area, Candle, HiLo etc.)* except Bar type series. You can add more than one trendline object to the **Trendlines** option.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Enable Trendline to chart series
.Visibility("visible").Add();
}).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



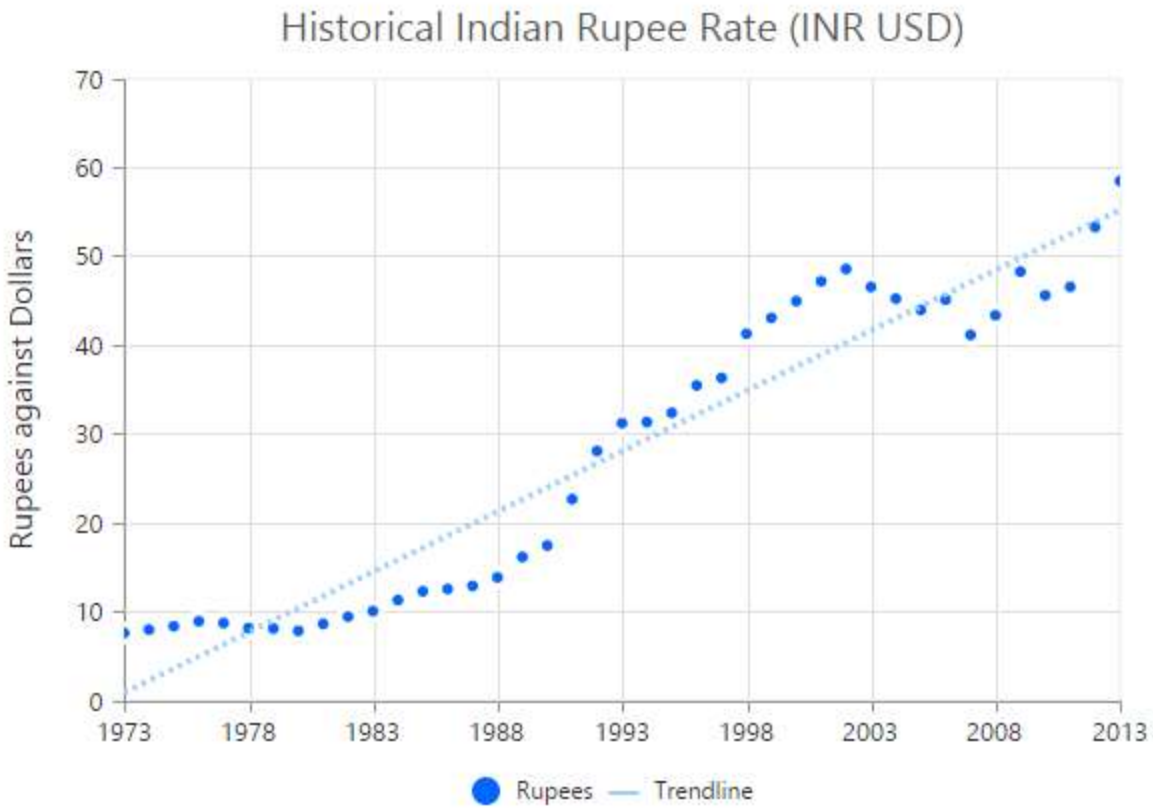
Click here to view the Trendlines online demo sample.

Customize the trendline styles

A trendline can be customized by using the properties such as *Fill*, *Width*, *DashArray* and *Opacity*. The default type of Trendline is **Linear**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Customize the Trendline styles
.Fill("#99CCFF").Width(3).Opacity(1).DashArray("2,3").Add();
}).Add();
})
//...
)
```



Types of Trendline

EjChart supports the following type of Trendlines.

- Linear
- Exponential
- Logarithmic
- Power
- Polynomial
- MovingAverage

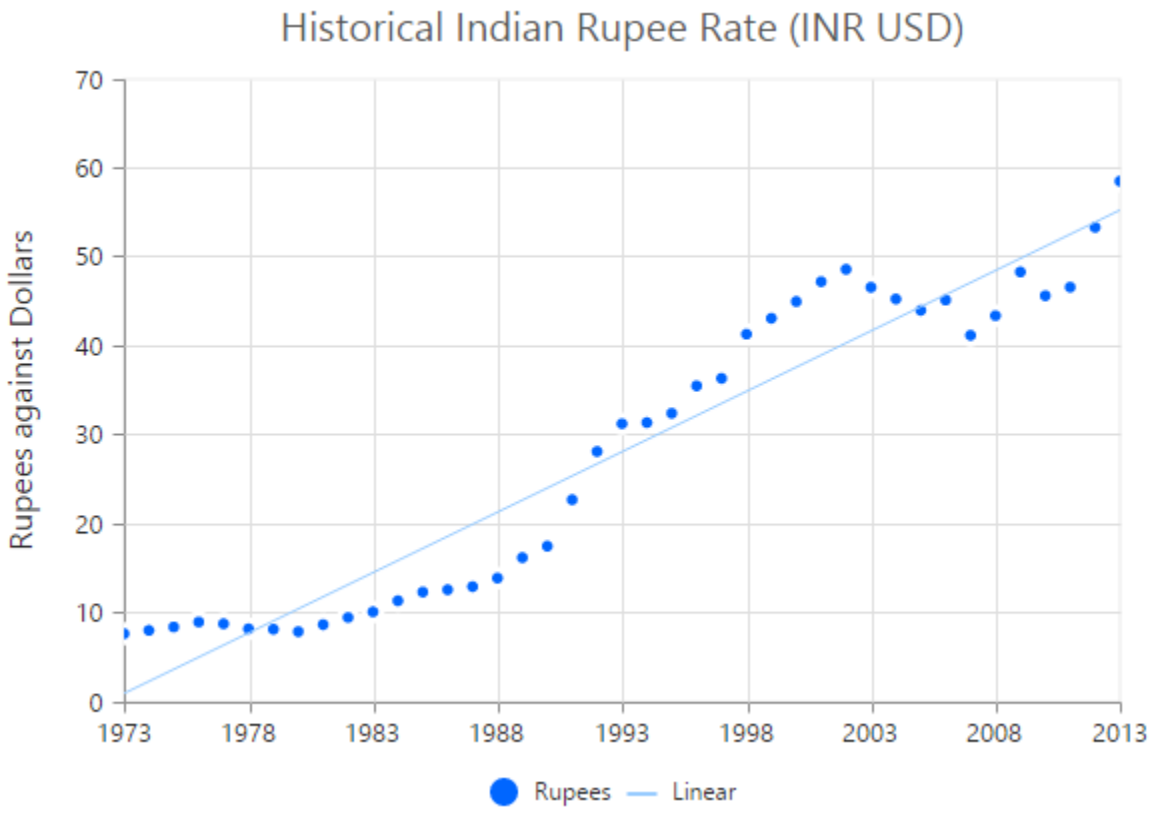
Linear

To render Linear Trendline, you have to set the **Type** as *Linear*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Change Trendline type
.Type(TrendlineType.Linear).Add();
```

```
}).Add();
})
//...
)
```

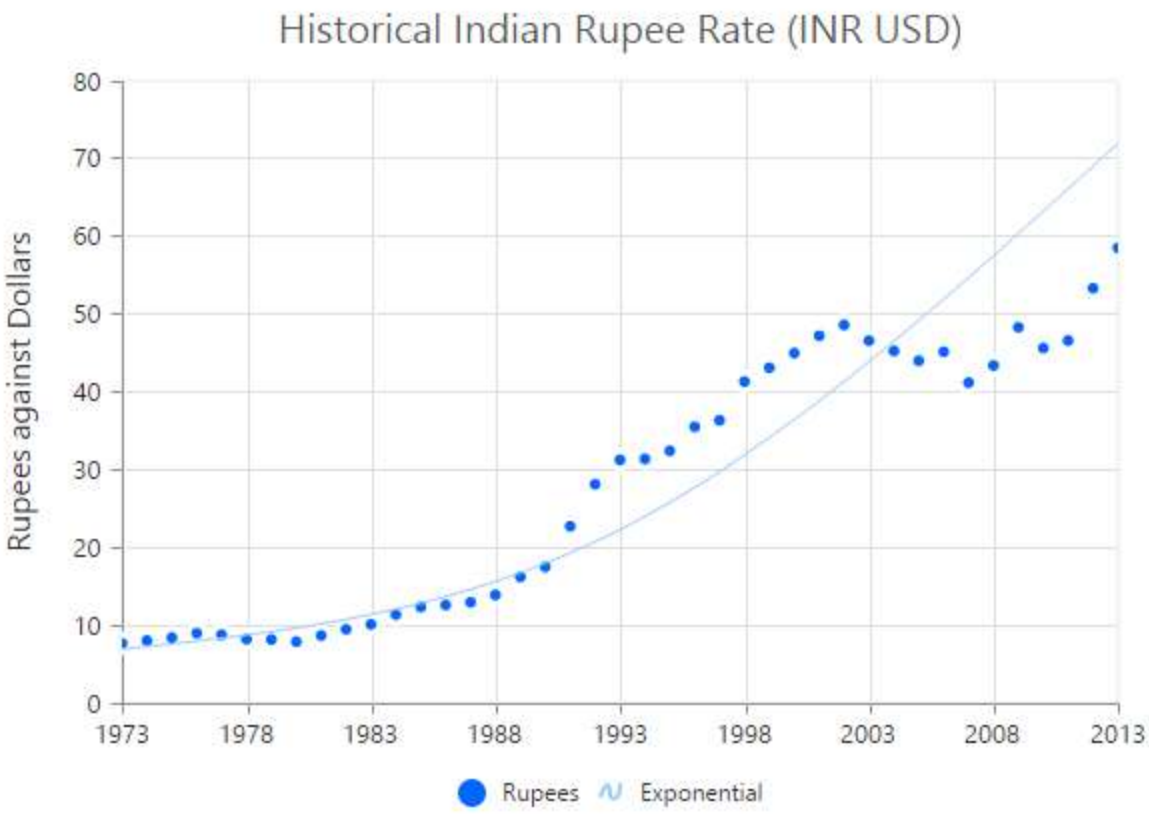


Exponential

Exponential Trendline can be rendered by setting the *Type* as Exponential.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Change Trendline type
.Type(TrendlineType.Exponential).Add();
}).Add();
})
//...
)
```



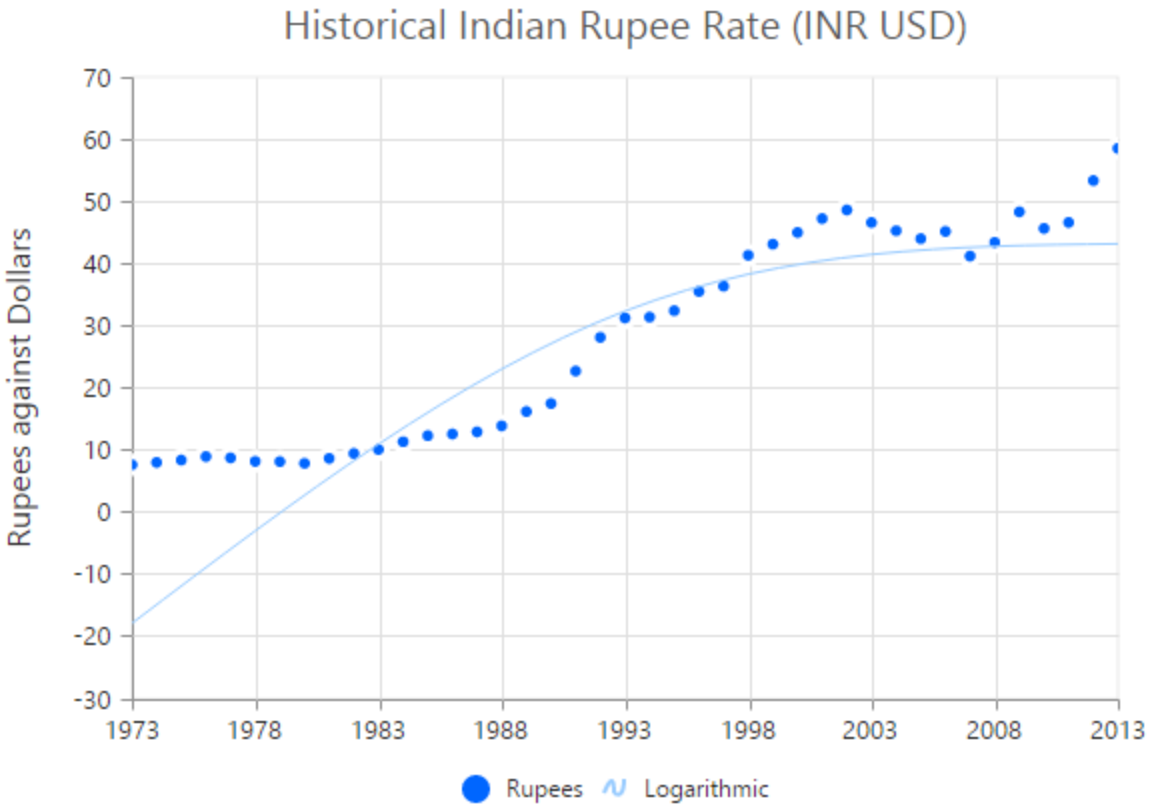
Logarithmic

Logarithmic Trendline can be rendered by setting the *Type* as Logarithmic.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Change Trendline type
.Type(TrendlineType.Logarithmic).Add();
}).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



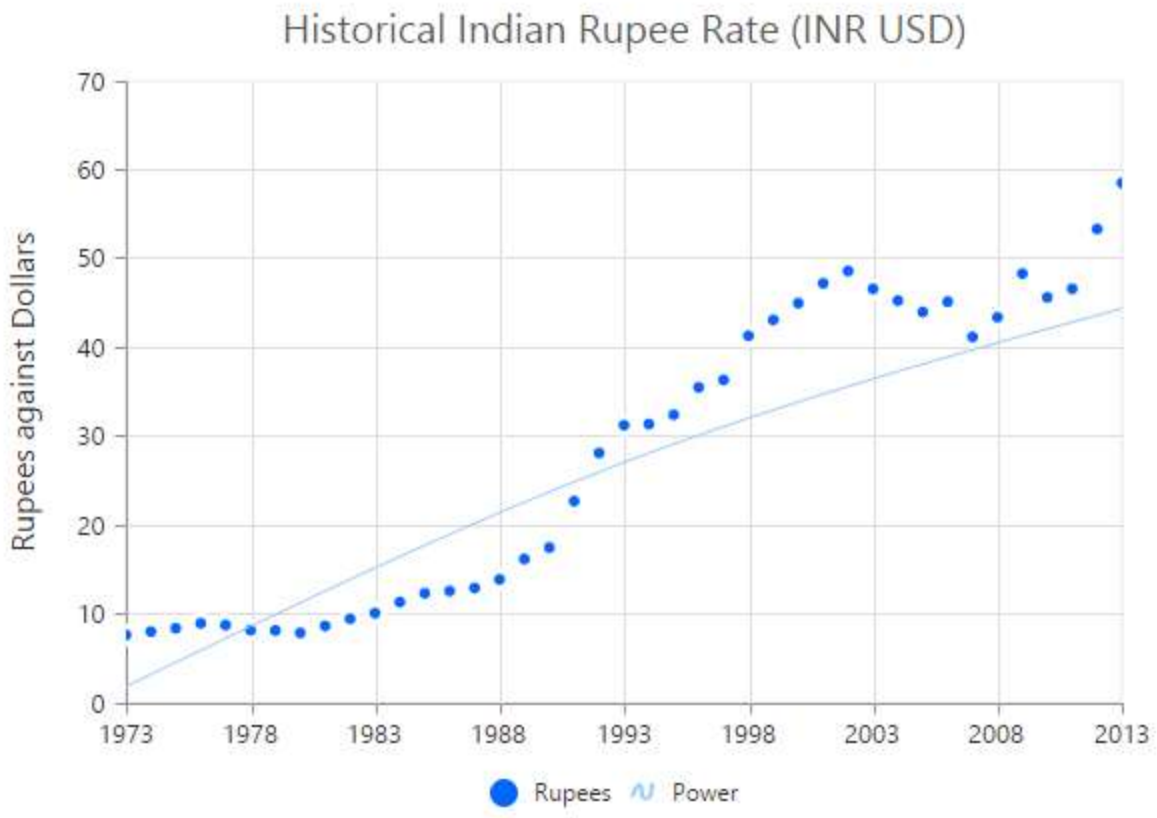
Power

Power Trendline can be rendered by setting the *Type* of the trendline as Power.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Change Trendline type
.Type(TrendlineType.Power).Add();
}).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

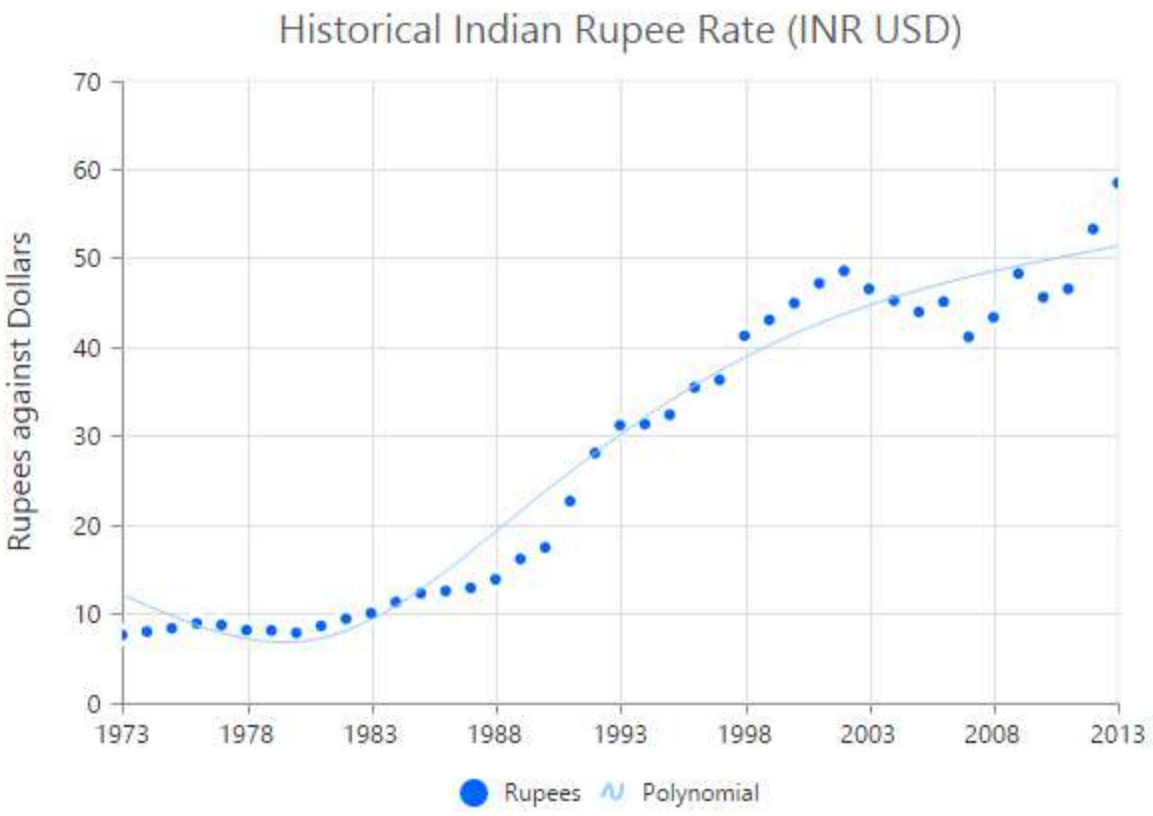


Polynomial

Polynomial Trendline can be rendered by setting the trendline *Type* as Polynomial. You can change the polynomial order by using the **PolynomialOrder** of the trendlines. It ranges from 2 to 6.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Change Trendline type
.Type(TrendlineType.Polynomial).Add();
}).Add();
})
//...
)
```

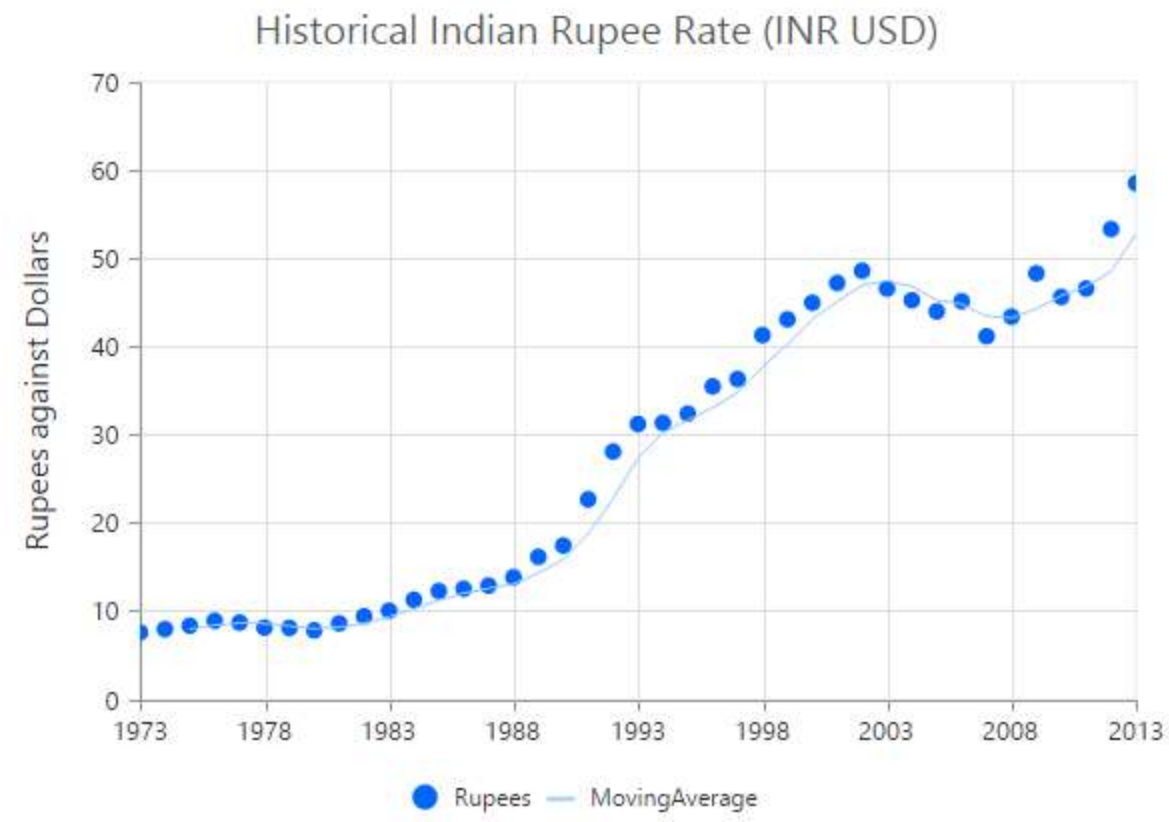


MovingAverage

MovingAverage Trendline can be rendered by setting the Type of the trendline as **MovingAverage**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Change Trendline type
.Type(TrendlineType.MovingAverage).Add();
}).Add();
})
//...
)
```



Forecasting

Trendline forecasting is the prediction of future/past situations. **Forecasting** can be classified into two types as follows.

- Forward Forecasting
- Backward Forecasting

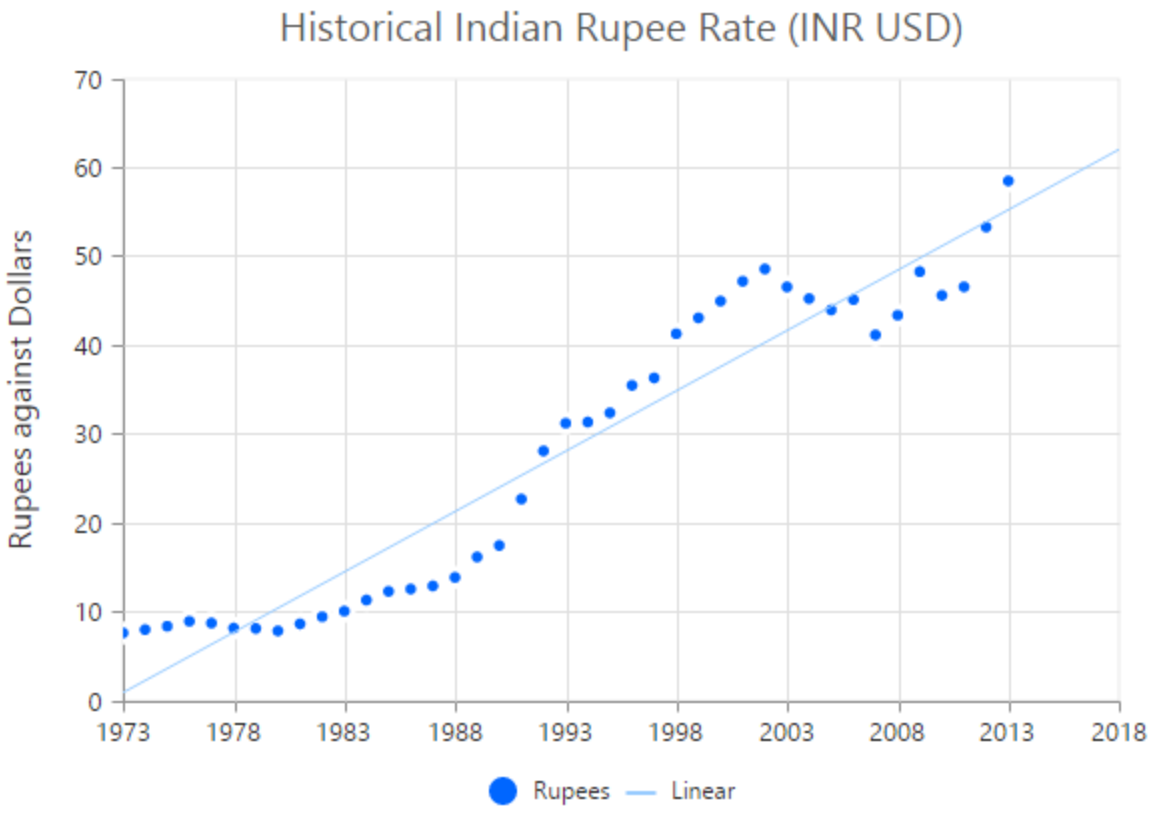
Forward Forecasting

The value set for **ForwardForecast** is used to determine the distance moving towards the future trend.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Set forward forecasting value
.ForwardForecast(5).Add();
}).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



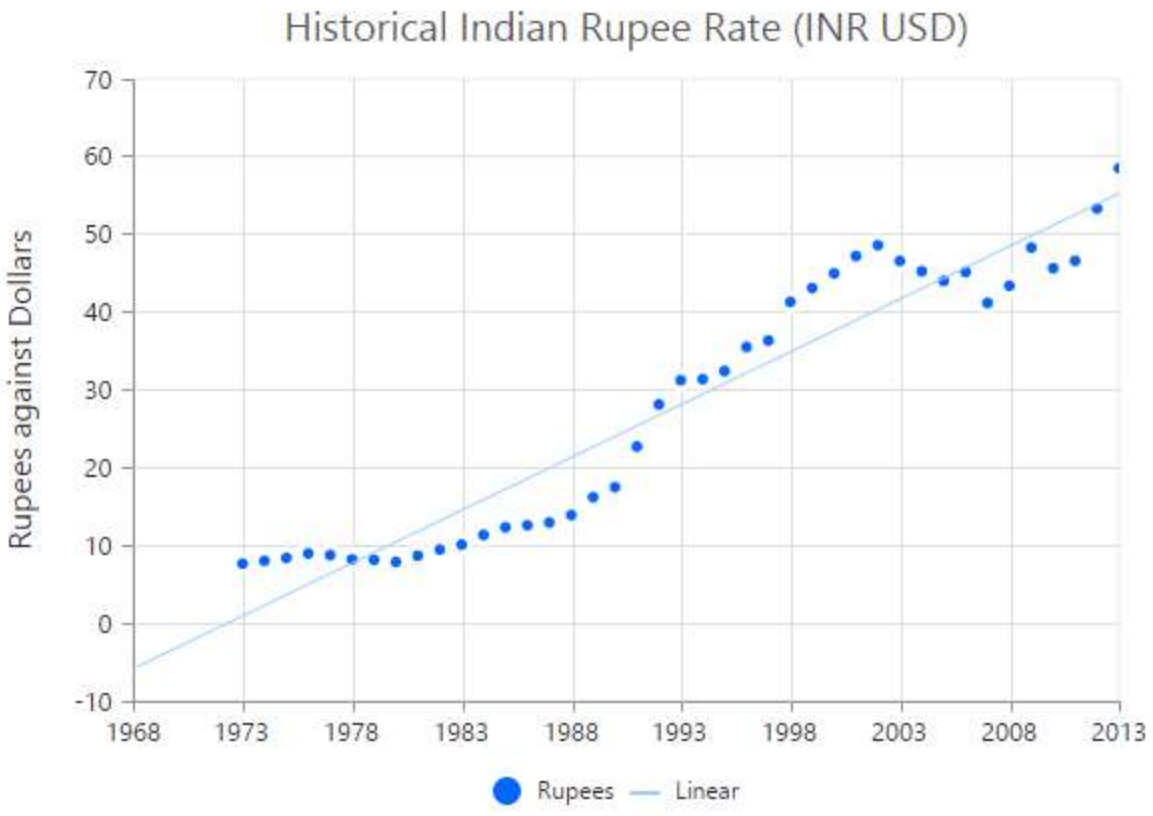
Backward Forecasting

The value set for the **BackwardForecast** is used to determine the past trends.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Set backward forecasting value
.BackwardForecast(5).Add();
}).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



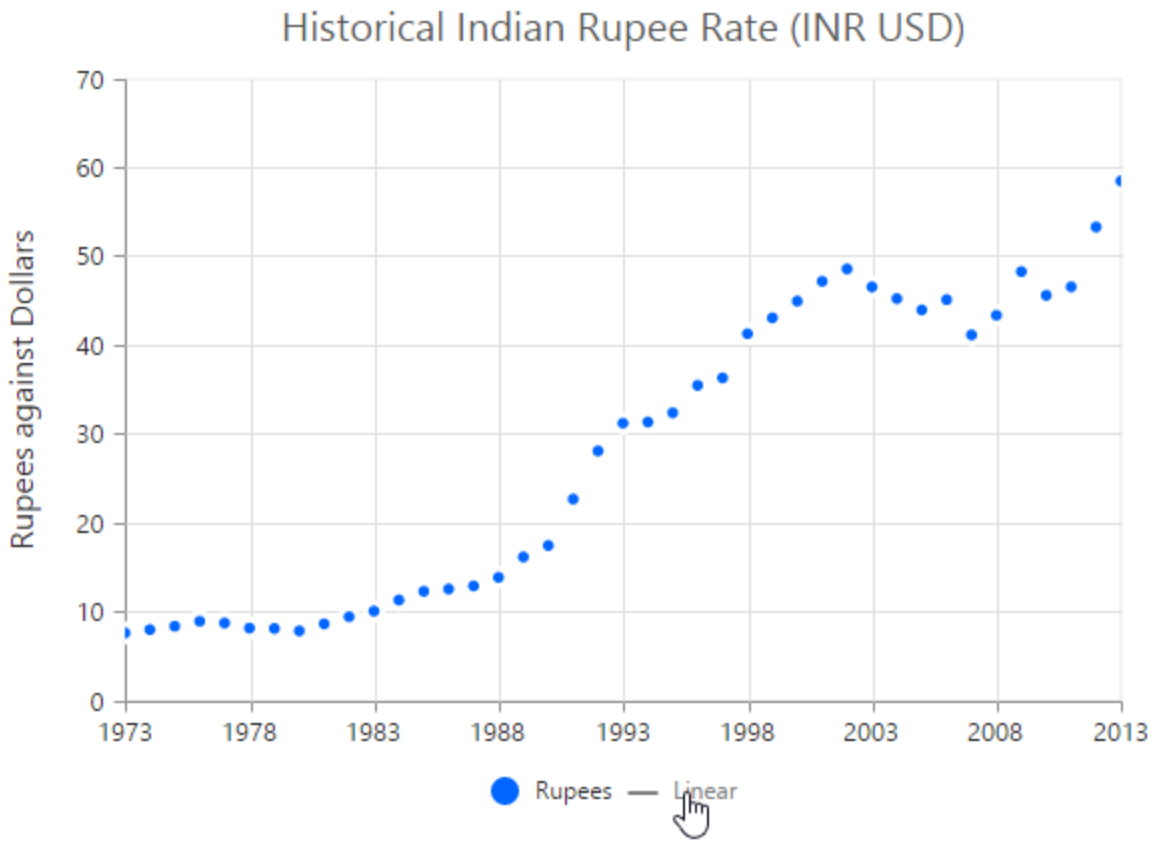
Trendlines Legend

To display the legend item for trendline, use the **Name** property. You can interact with the trendline legends similar to the series legends *(show/hide trendlines on legend click)*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//...
.Series(sr =>
{
sr
.Trendlines(trendline =>
{
trendline
//Set Trendline name to display in the legend
.Name("Linear").Add();
}).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Technical Indicators

EjChart control supports 10 types of technical indicators.

Bind data to render the indicator

You can bind the series **DataSource** to the indicator by setting the specific series name to the indicator by using the *Indicators.SeriesName* property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
sr
.XName("XDate").High("High").Low("Low").Open("Open").Close("Close")
//Set name to series
.Name("Hilo")
.DataSource(ViewBag.ChartData).Add();
})
//Initializing Indicators
.Indicators(ind => { ind
//Set HiLo series dataSource to indicator using seriesName
.SeriesName("Hilo").Add();
})
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

CSHARP

```
//// Create dataSource to chart
List<ChartData> data = new List<ChartData>();
data.Add(new ChartData("2010/7/1", 38, 10, 58, 27));
data.Add(new ChartData("2010/7/8", 28, 15, 48, 58));
data.Add(new ChartData("2010/7/9", 54, 35, 85, 98));
data.Add(new ChartData("2010/7/10", 52, 21, 38, 85));
///...
ViewBag.ChartData = data;
```

Also, you can add data to the indicator directly by using the DataSource option of the indicator.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Indicators
.Indicators(ind => { ind
//Add dataSource to indicator directly
.XName("XDate").High("High").Low("Low").Open("Open").Close("Close")
.DataSource(ViewBag.ChartData).Add();
})
//...
)
```

Indicator Types

Accumulation Distribution

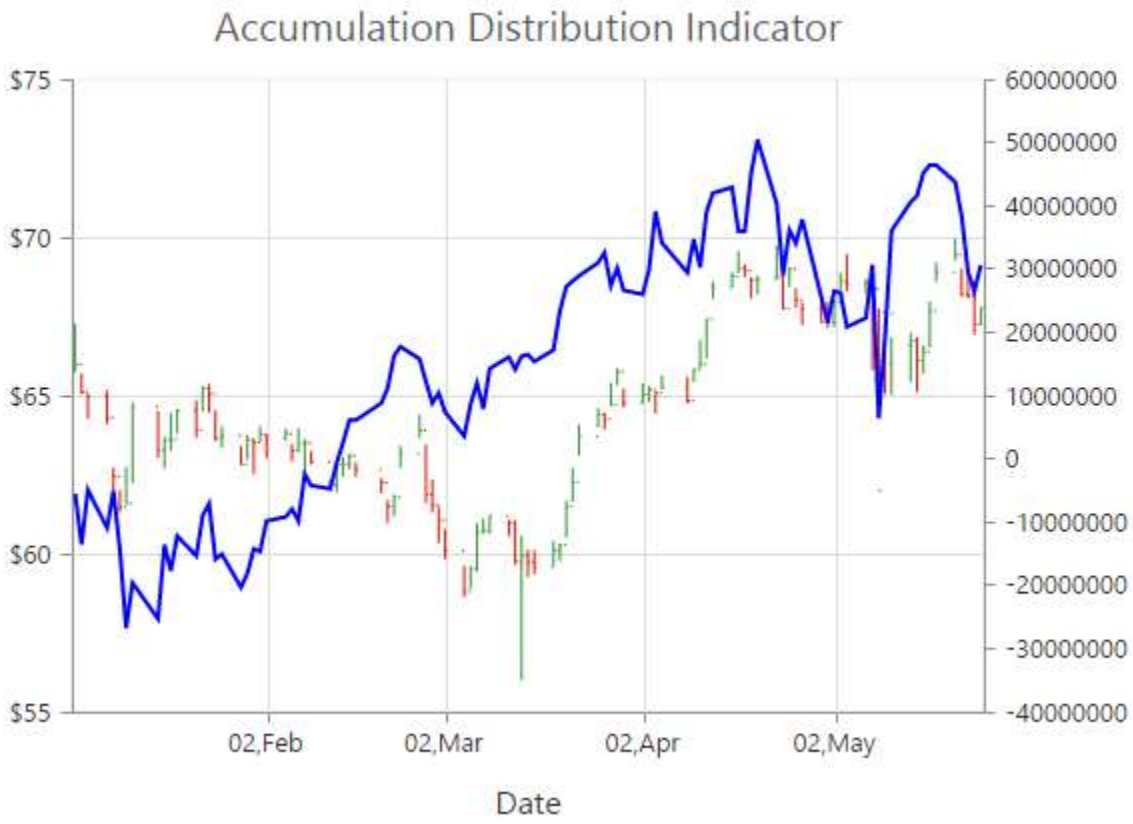
To create an Accumulation Distribution indicator, set the *Indicators.Type* as **AccumulationDistribution**. Accumulation Distribution require **Volume** field additionally with the DataSource to calculate the signal line.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
sr
.XName("XDate").High("High").Low("Low").Open("Open").Close("Close").Volume("
Volume")
//Set name to series
.Name("Hilo")
.DataSource(ViewBag.ChartData).Add();
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.AccumulationDistribution)
.SeriesName("Hilo").Add();
})
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```



Click here to view the Accumulation Distribution indicator online demo sample.

Average True Range (ATR)

You can create an ATR indicator by setting the Indicators.Type as **ATR** in the Indicators.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.ATR).Add();
})
//...
)
```



Click here to view the ATR indicator online demo sample.

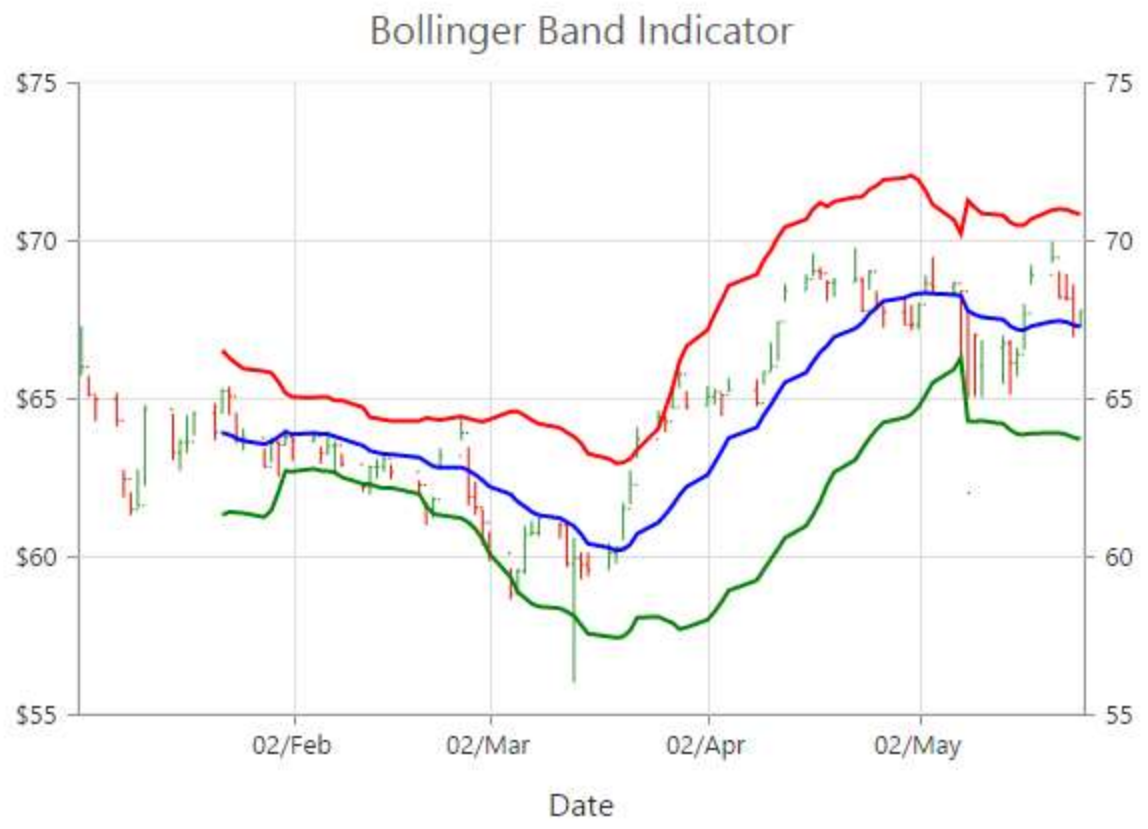
Bollinger Band

Bollinger Band indicator is created by setting the Indicators.Type as **BollingerBand**. It contains three lines, namely upper band, lower band and signal line. Bollinger Band default value of the *Period* is 14 and *StandardDeviations* is 2.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.BollingerBand).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Bollinger Band indicator online demo sample.

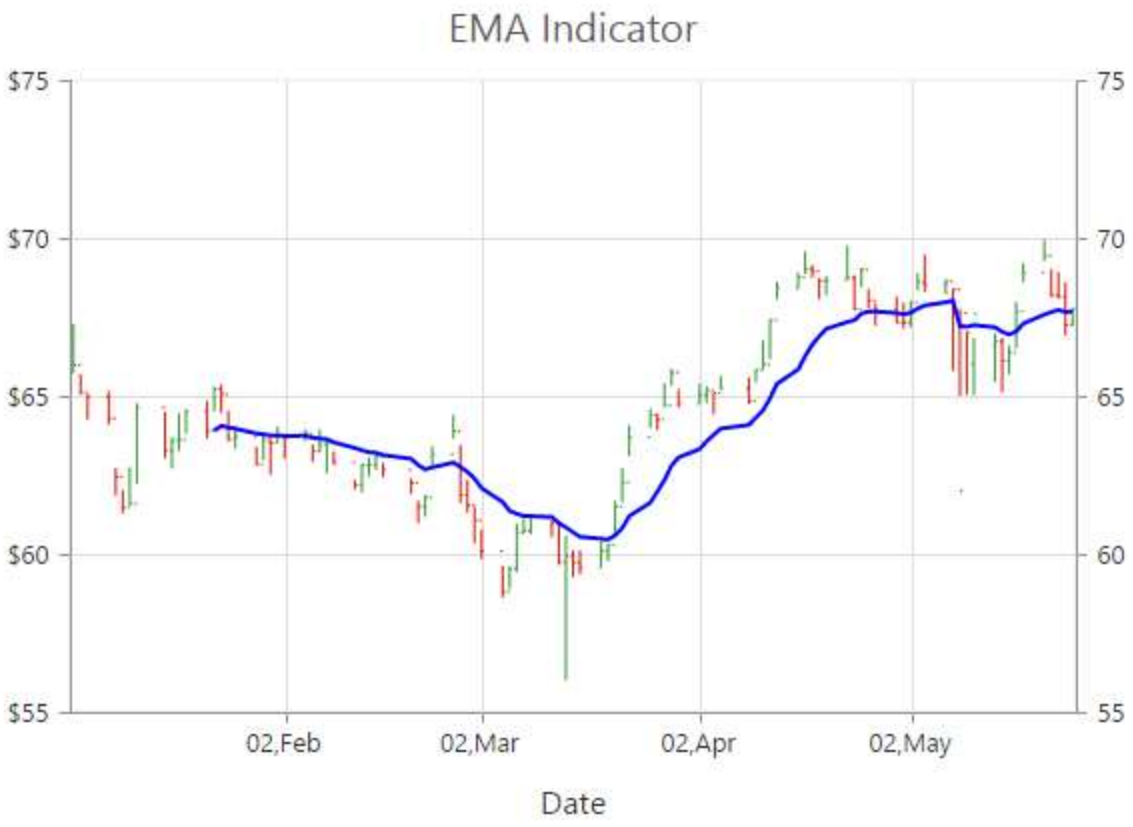
Exponential Moving Average (EMA)

To render an EMA indicator, you have to set the Indicators.Type as **EMA**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.EMA).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the EMA indicator online demo sample.

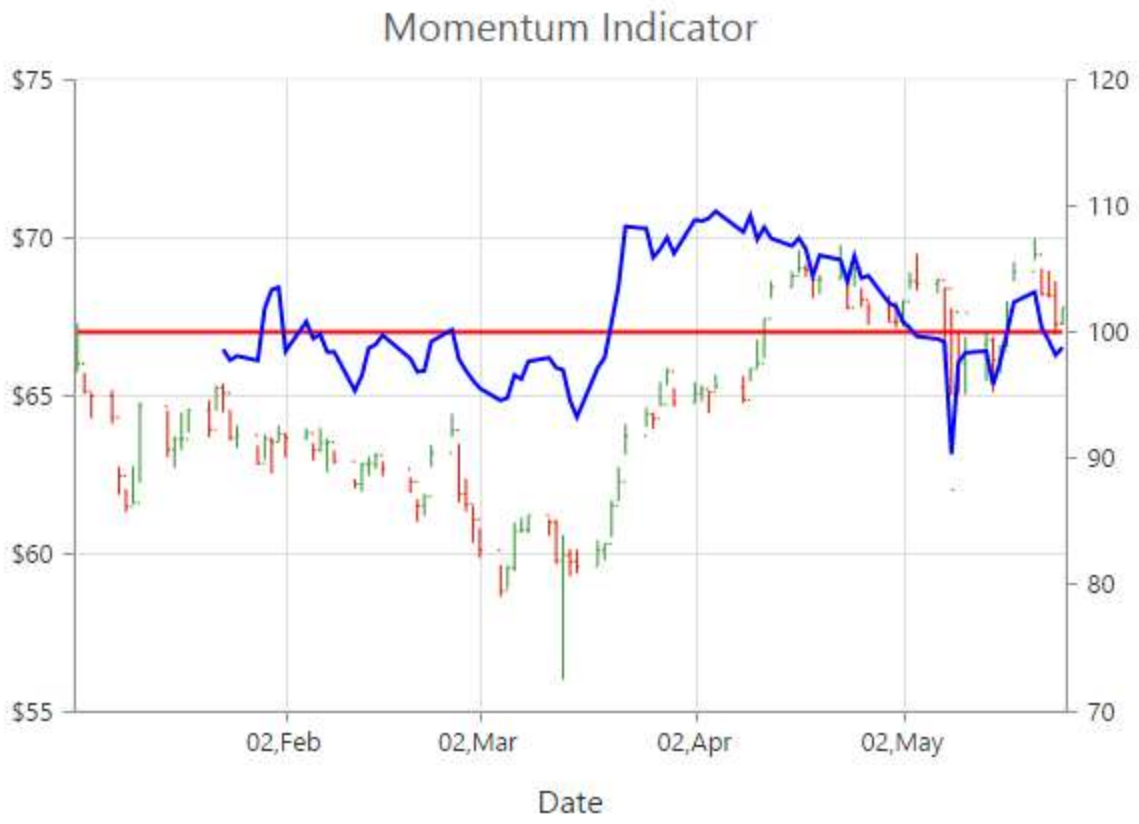
Momentum

Momentum Technical indicator is created by setting the Indicators.Type as **Momentum**. The momentum indicator renders two lines, namely upper band and signal line. Upper band always rendered at the value 100 and the signal line is calculated based on the momentum of the data.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.Momentum).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Momentum indicator online demo sample.

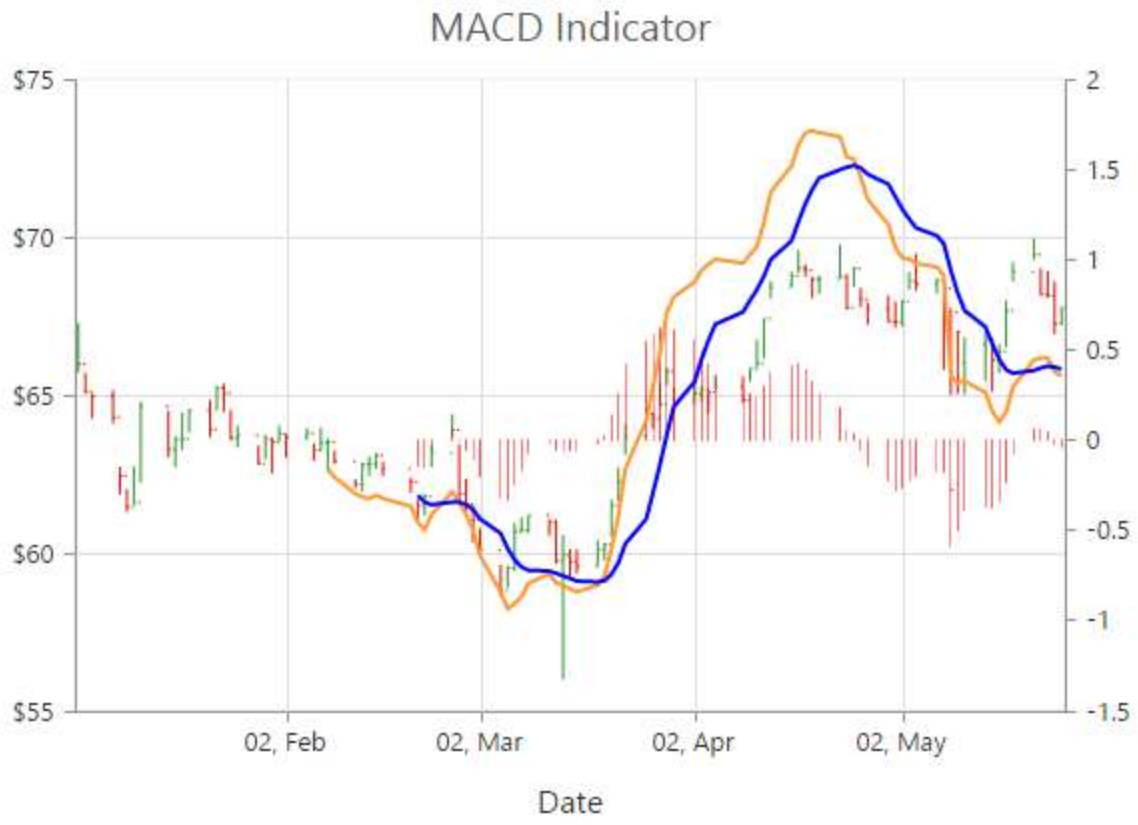
Moving Average Convergence Divergence (MACD)

To render an MACD indicator, you have to set the Indicators.Type as **MACD**. MACD indicator contains MACD line, Signal line and Histogram column. Histogram is used to differentiate MACD and signal line.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.MACD).Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



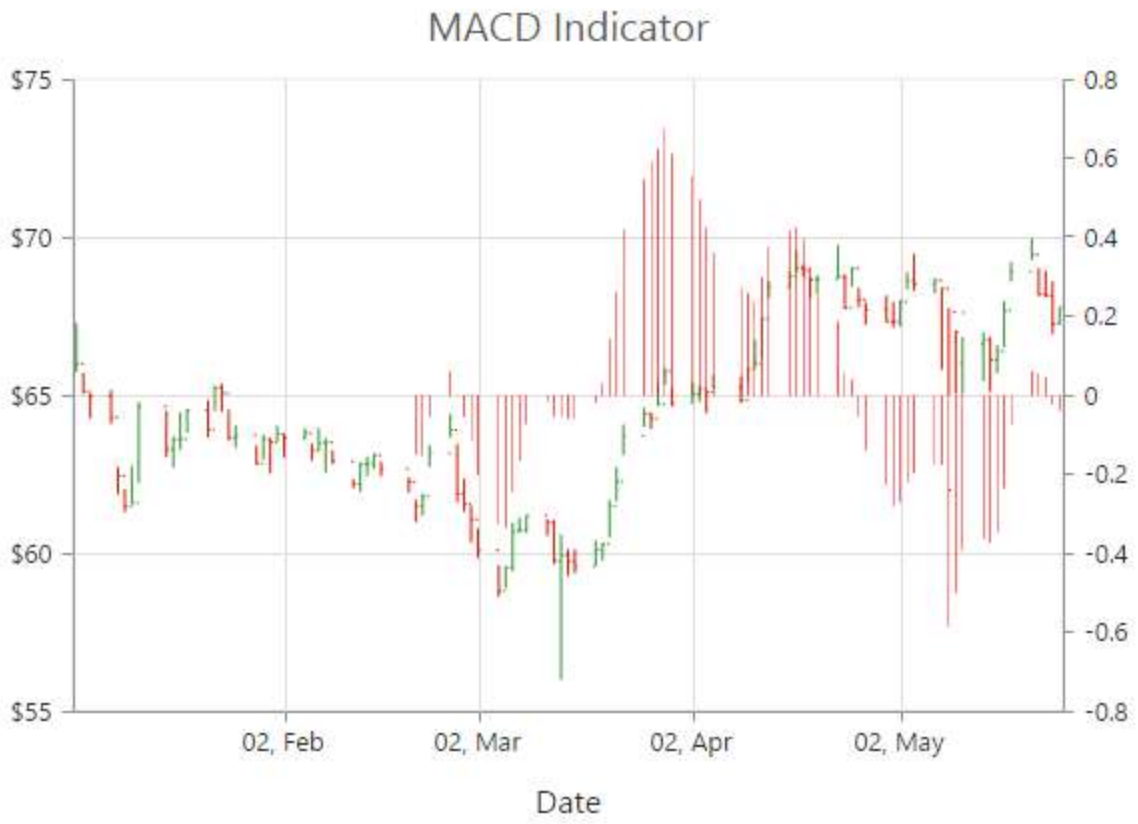
Click here to view the MACD indicator online demo sample.

MacdType

By using the **MacdType** enumeration property, you can change the MACD rendering as *Line*, *Histogram* or *Both*.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
.Type(ChartIndicatorType.MACD)
//Set MACD draw type
.MacdType(MacdType.Histogram)
.Add();
})
//...
)
```



Relative Strength Index (RSI)

To render the RSI indicator, set the Indicators.Type as **RSI**. It contains three lines, namely upper band, lower band and signal line. Upper and lower band always render at value 70 and 30 respectively and signal line is calculated based on the *RSI* formula.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.RSI)
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the RSI indicator online demo sample.

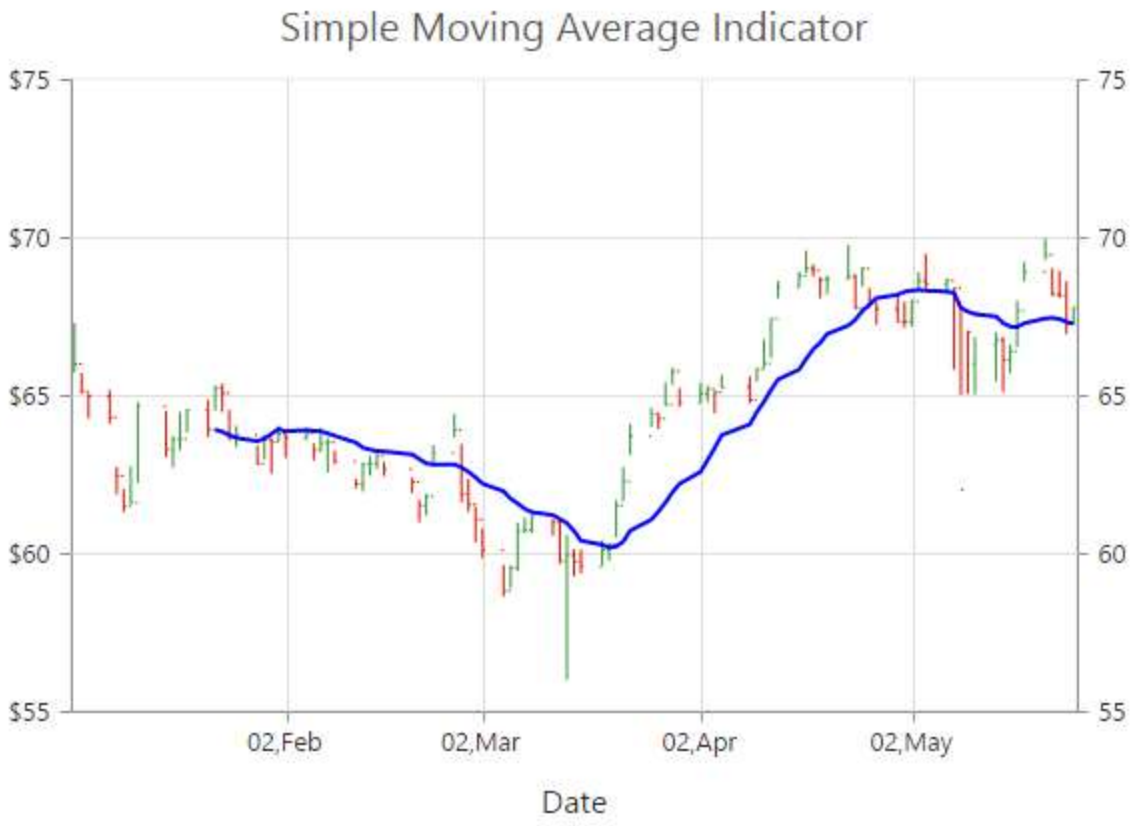
Simple Moving Average (SMA)

To render the SMA indicator, you should specify the Indicators.Type as **SMA**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.SMA)
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the SMA indicator online demo sample.

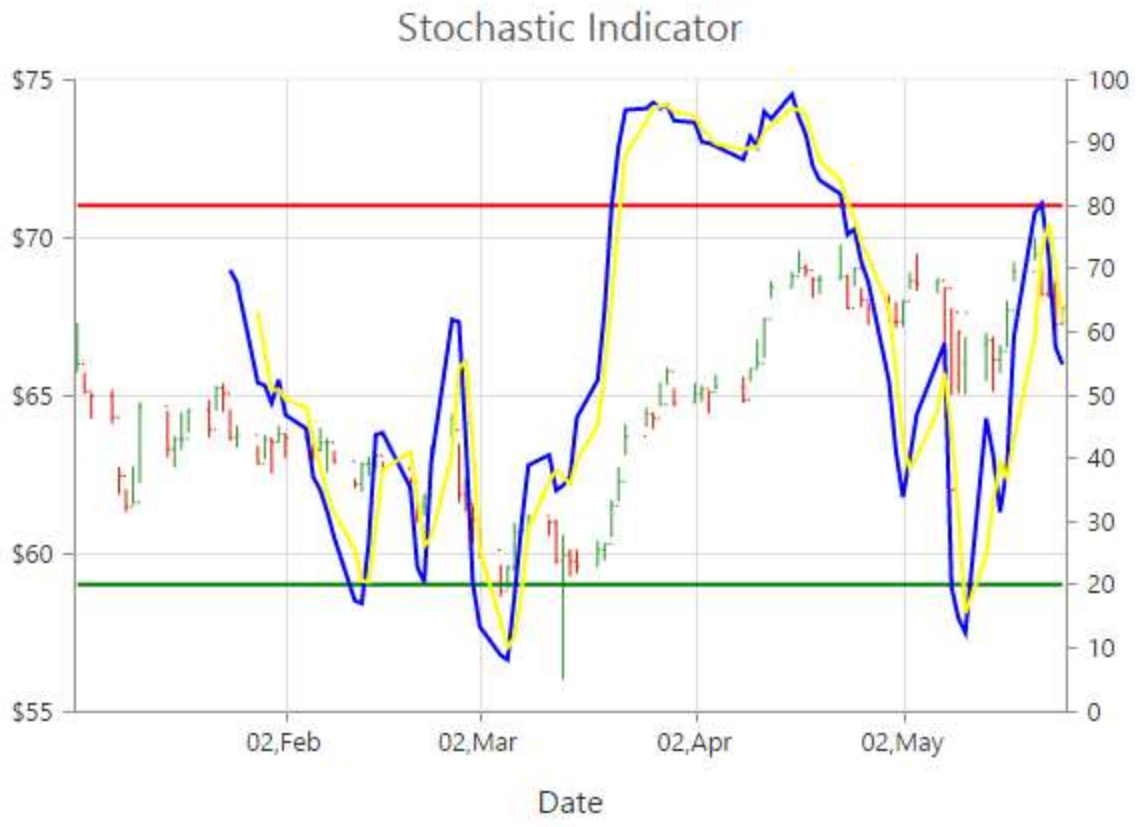
Stochastic

For the Stochastic indicator, you need to set the Indicators.Type as **Stochastic**. The Stochastic indicator renders four lines namely, upper line, lower line, stochastic line and the signal line. Upper line always rendered at value 80 and the lower line is rendered at value 20. Stochastic and Signal Lines are calculated based on the stochastic formula.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.Stochastic)
.Add();
})
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the stochastic indicator online demo sample.

Triangular Moving Average (TMA)

To render the TMA indicator, you should specify the Indicators.Type as **TMA**.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//Set indicator type
.Type(ChartIndicatorType.TMA)
.Add();
})
//...
)
```



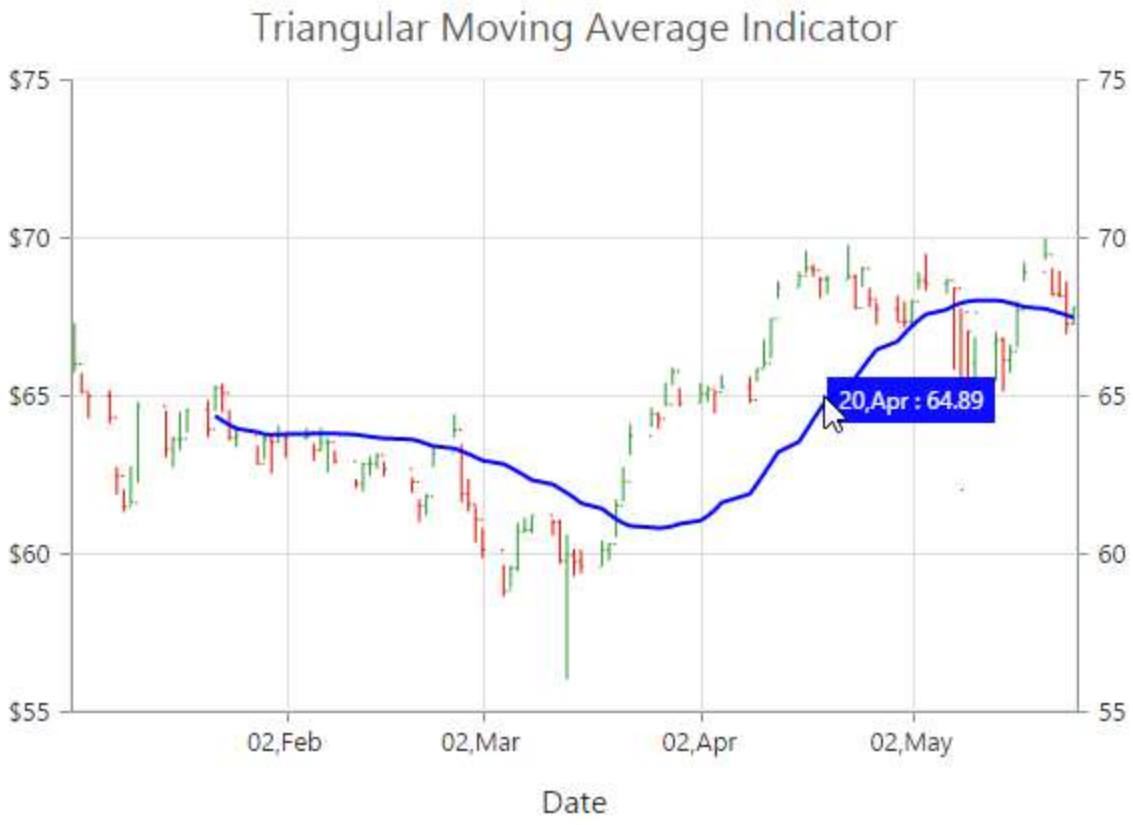
Click here to view the TMA indicator online demo sample.

Enable Tooltip

To display the indicator tooltip, use *Visible* option of the **Indicators.Tooltip**. Also, you can change and customize the tooltip color, border, format and font properties similar to the series tooltip.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Initializing Series
.Series(sr =>
{
//...
})
//Initializing Indicators
.Indicators(ind => { ind
//...
.Tooltip(tooltip=>tooltip.
//Enable tooltip for indicator
Visible(true))
.Add();
})
//...
)
```



Annotations

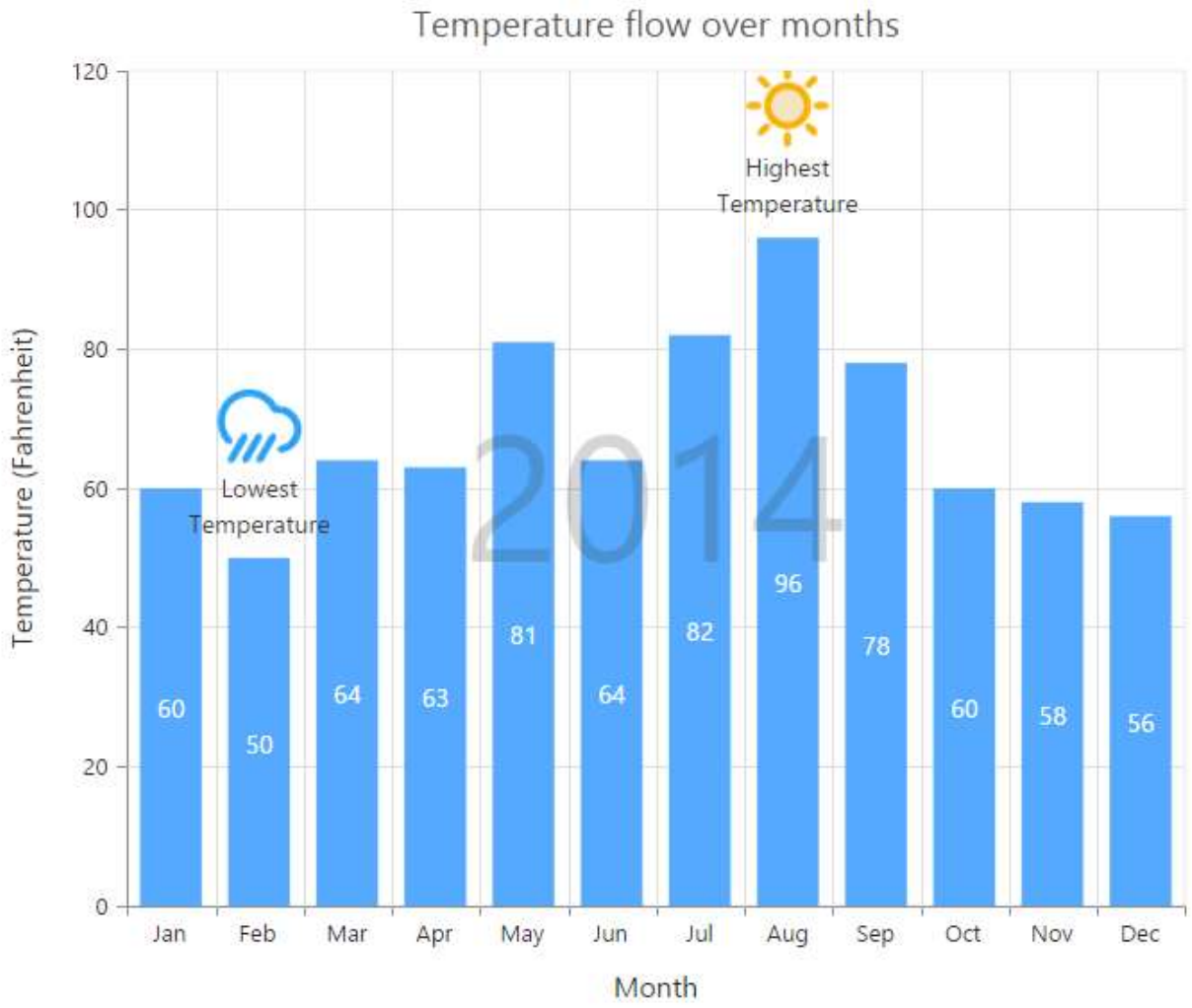
annotations are used to mark the specific area of interest in the chart area with texts, shapes or images.

You can add annotations to the chart by using the **Annotations** option. By using the **Content** option of annotation object, you can specify the id of the element that needs to be displayed in the chart area.

ASPX-CS

```
<div id= "watermark" style="font-size:100px; display:none">2014</div>
<div>
@(Html.EJ().Chart("chartContainer")
.Annotations(an => {
//Add Annotation content here
an.Visible(true).Content("watermark").Opacity(0.2).Region(Region.Series).Add
();
})
// ...
)
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Annotations online demo sample.

Note: Annotations are not supported in 3D chart types.

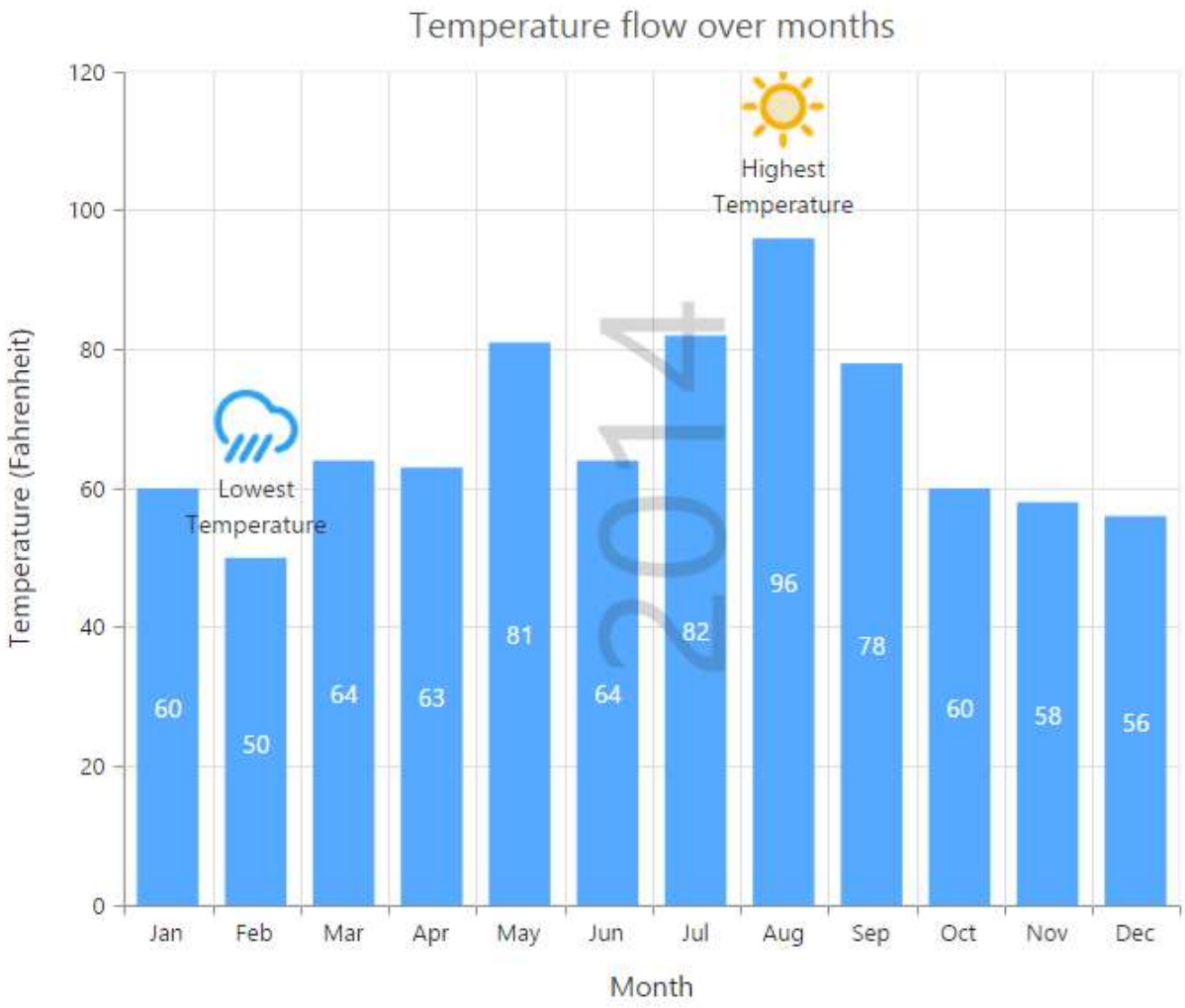
Rotate the annotation template

To rotate the annotation template, you can use the **Angle** property of the annotations.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Annotations(an => {
an.Visible(true)
// ...
//Rotate the Annotation template
.Angle(270).Add();
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Positioning Annotation

You can position annotations either by using the coordinates (**X** and **Y**) options or by using the alignment options **HorizontalAlignment** and **VerticalAlignment**.

By using the **CoordinateUnit** option, you can specify whether the value provided in the X and Y options are relative to the chart or axis.

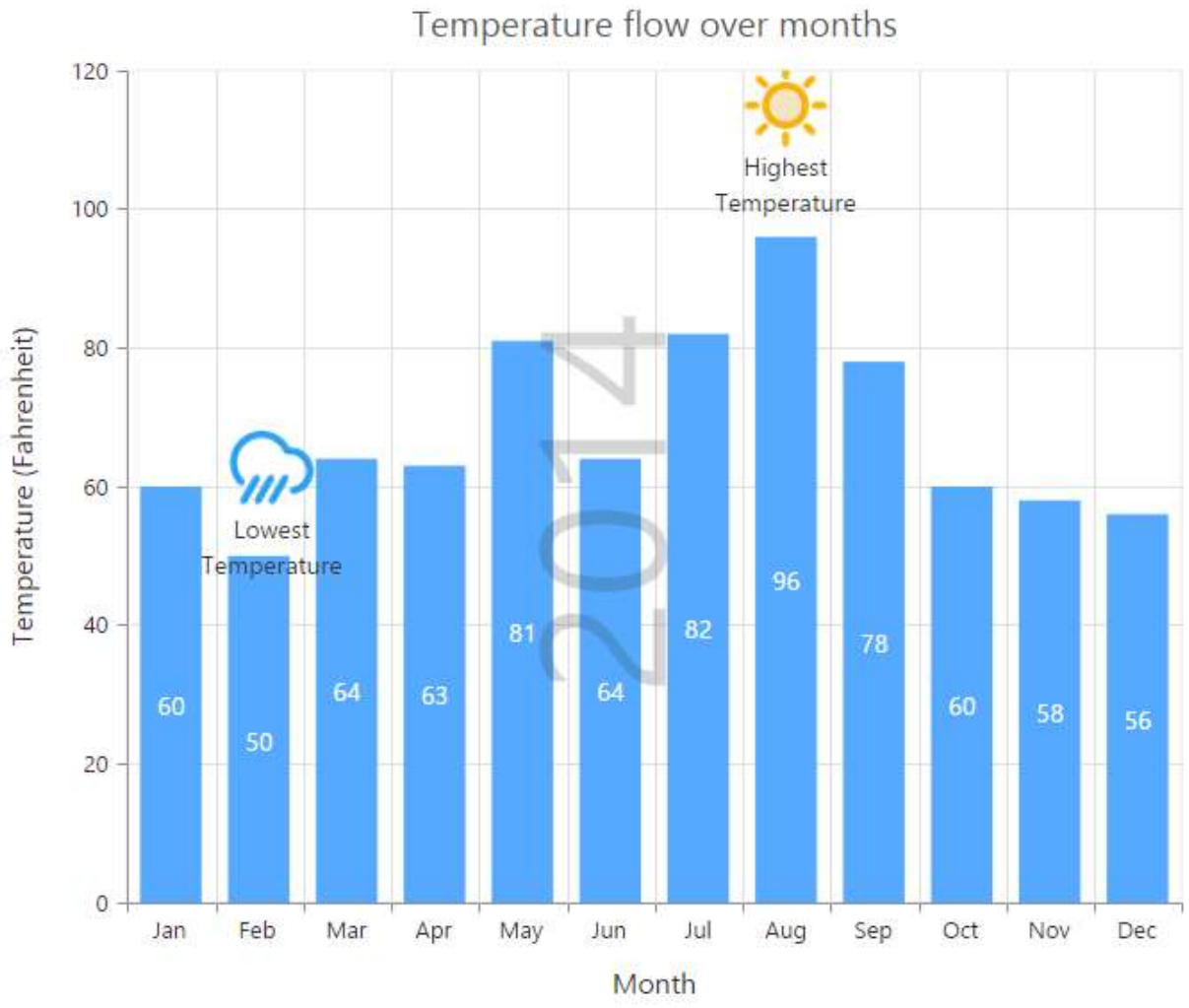
- If the CoordinateUnit is set to None, the annotations are placed relative to the chart/plot area by using the HorizontalAlignment and VerticalAlignment options.
- If the coordinateUnit is set to points, the x and y values of the annotation are the coordinates relative to the axis and annotation is positioned relative to the axis. By default, the x and y values are associated with the **PrimaryXAxis** and **PrimaryYAxis**. In case, when the chart contains multiple axis and you want to associate the annotation with a particular axis, you can specify the **XAxisName** and **YAxisName** options of the annotation object.
- If the coordinateUnit is set to Pixels, the X and Y values are coordinates relative to the top-left corner of the chart/plot area.

Note: By using the **Region** option, you can specify whether the annotation is placed relative to the entire chart or plot area.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Annotations(an => {
an.Visible(true)
// ...
//Change coordinateUnit type to pixels
.CoordinateUnit(CoordinateUnit.Pixels).X(170).Y(350).Add();
})
// ...
)
```



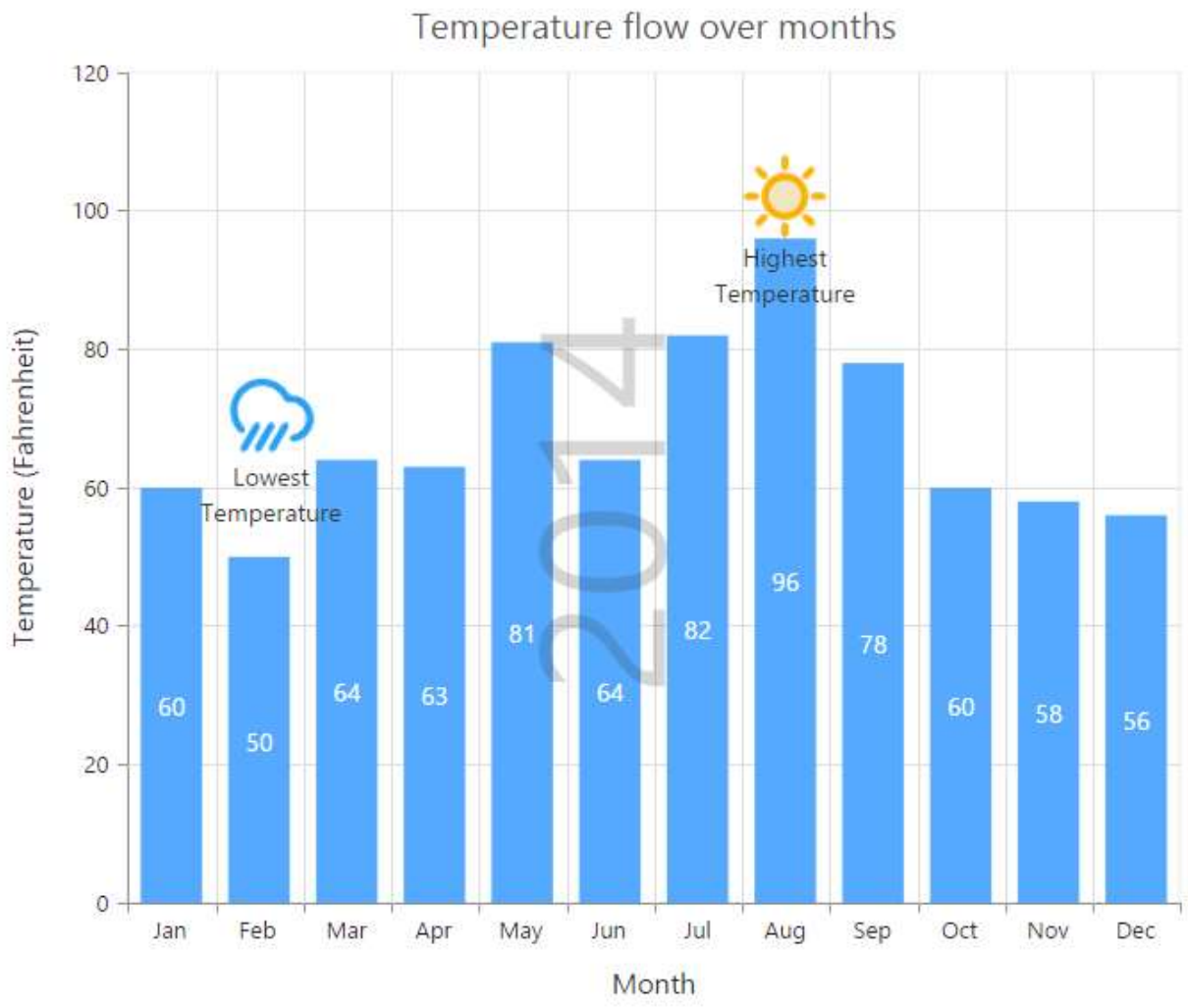
Annotation alignments

When the coordinateUnit is set to pixels or points, you can align the annotation relative to the coordinates by using the **HorizontalAlignment** and **VerticalAlignment** options.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Annotations(an => {
an.Visible(true)
```

```
// ...
//Change alignment of annotation template
.VerticalAlignment(VerticalAlignment.Middle).HorizontalAlignment(HorizontalA
lignment.Left)
.Margin(mr=>mr.Right(40)).Add();
})
// ...
)
```



Appearance

Custom Color Palette

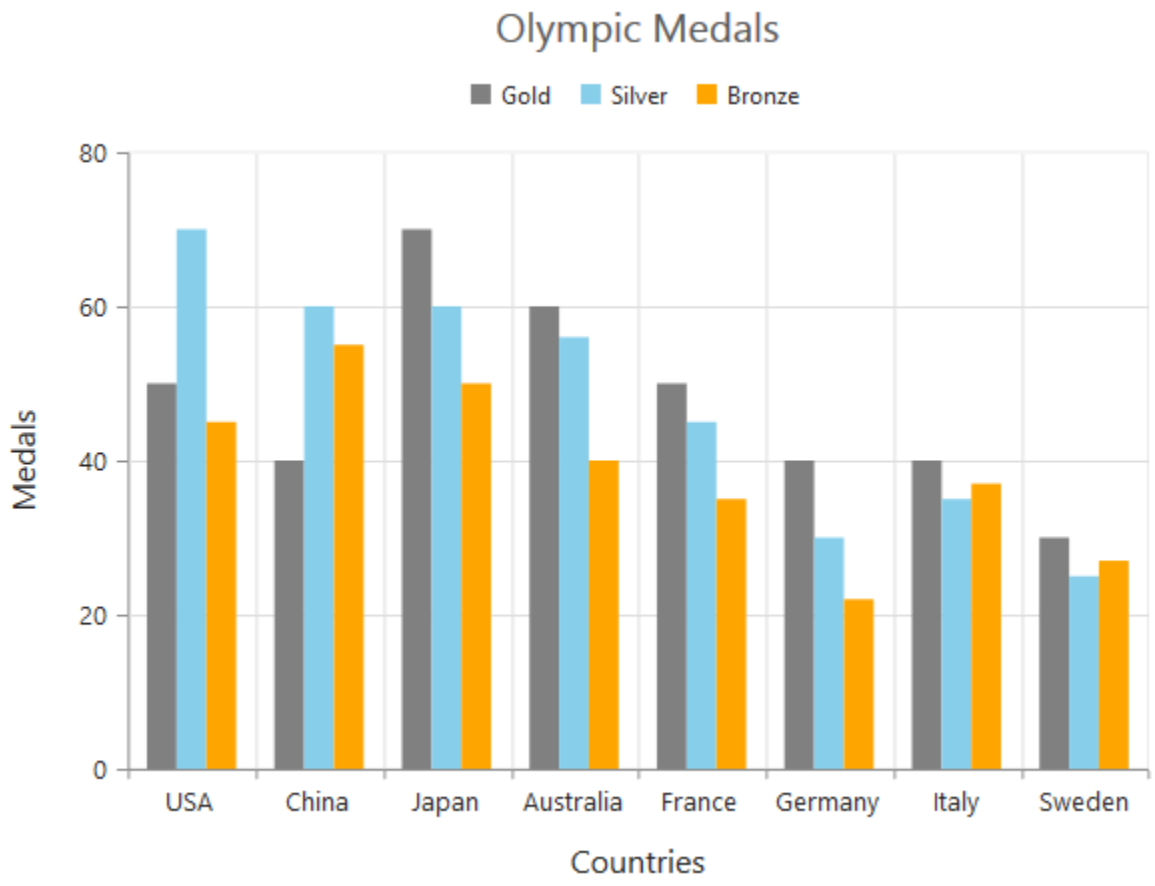
The Chart displays different series in different colors by default. You can customize the color of each series by providing a custom color palette of your choice by using the **Palette** property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//Providing a custom palette
.Palette(pl => { pl.Add("grey"); pl.Add("skyblue"); pl.Add("orange"); })
// ...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```



Note: The Color palette is applied to the points in accumulation type series

Built-in Themes

Following are the built-in themes available in the Chart

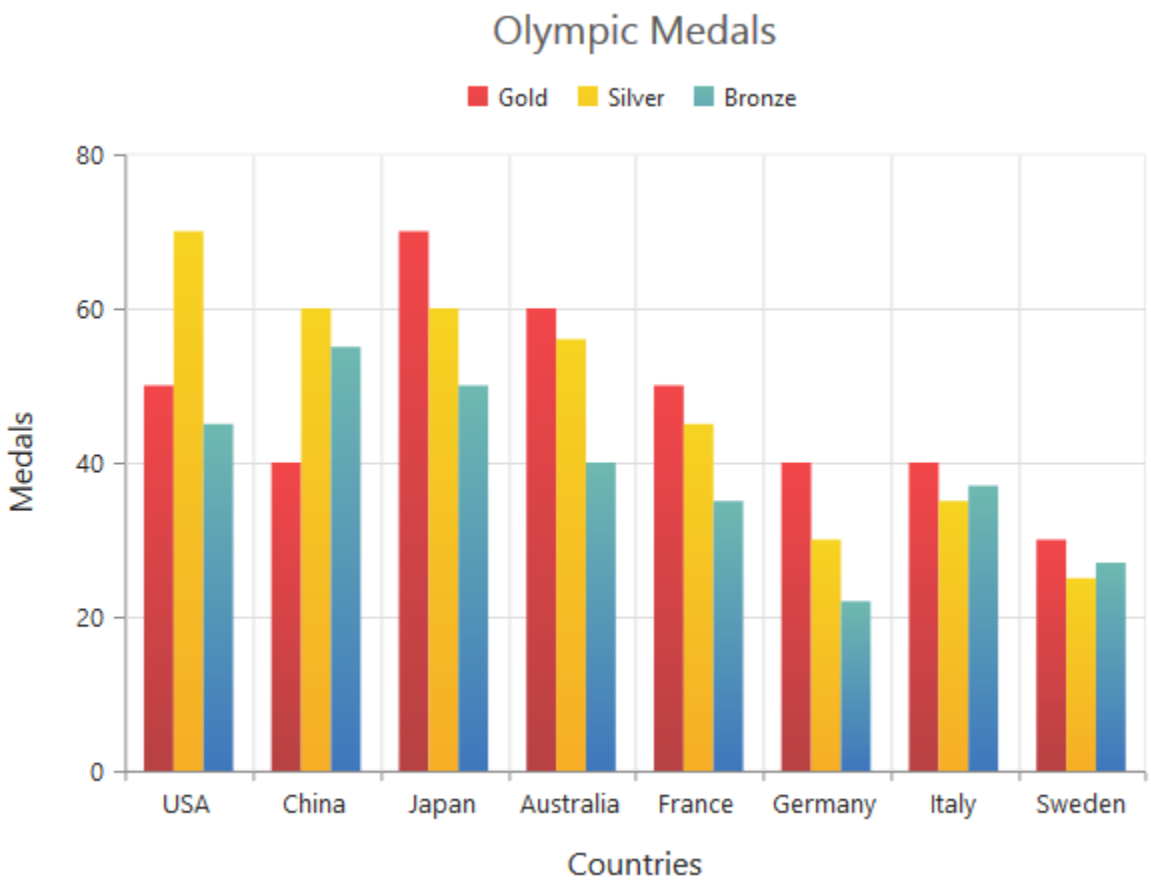
- Azure
- Azuredark
- Flatdark
- Flatlight
- GradientDark
- GradientLight
- Lime
- LimeDark
- Saffron
- SaffronDark

You can set your desired theme by using the **Theme** property. FlatLight is the default theme used in the Chart.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Chart("chartContainer")
//Using gradient theme
.Theme(Syncfusion.JavaScript.DataVisualization.ChartTheme.GradientLight)
// ...
)
```



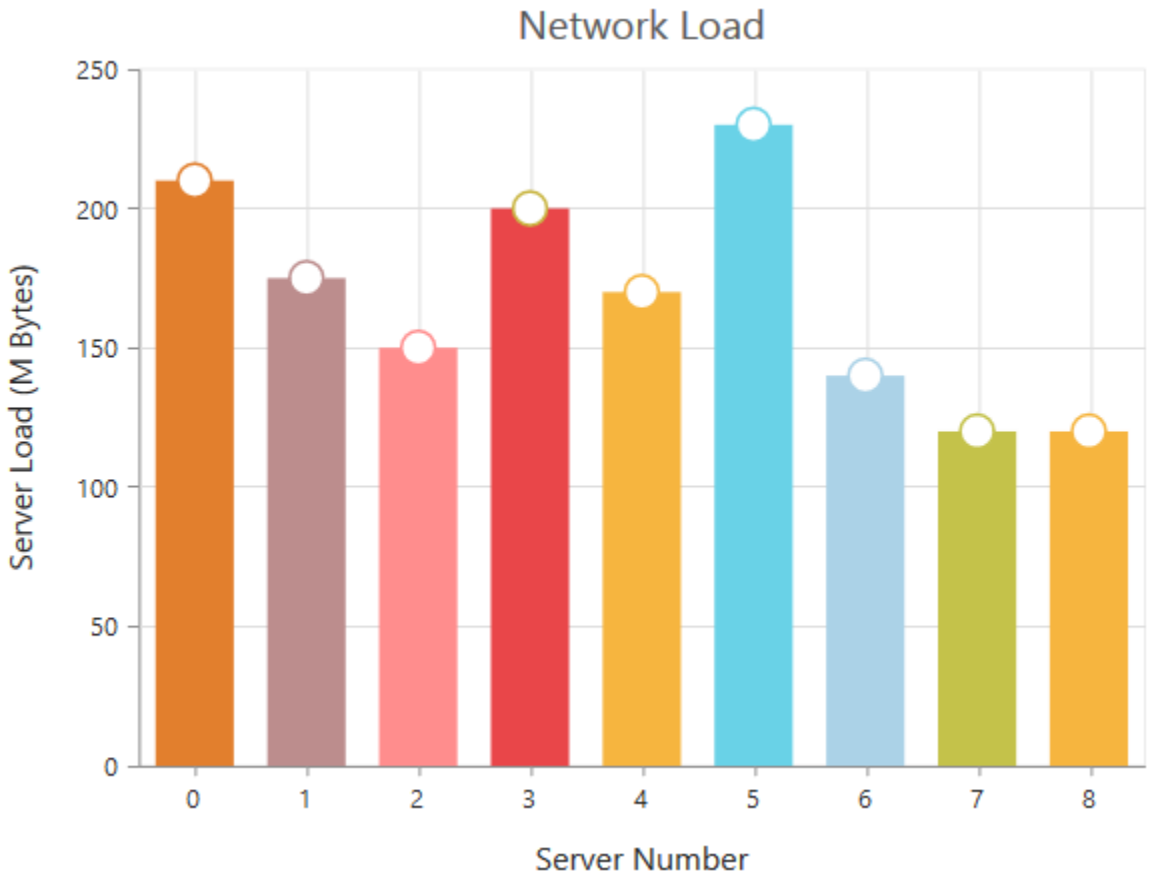
Point level customization

Marker, DataLabel and Fill color of each point in a series can be customized individually by using the **Points** collection.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
sr.Points(pt => {
//Customizing marker and fill color of a point
pt.X("0").Y(210).Fill("#E27F2D").Marker(mr => mr.Visible(true)).Add();
//...
})
.Add();
//...
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Series border customization

To customize the series border color, width and dashArray, you can use **Series.Border** option. The series border styles can be applied only to the column, bar and area type of series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr =>
{
sr.Border(border=>border.Color("blue").Width(2).DashArray("5,3")).Add();
//...
})
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

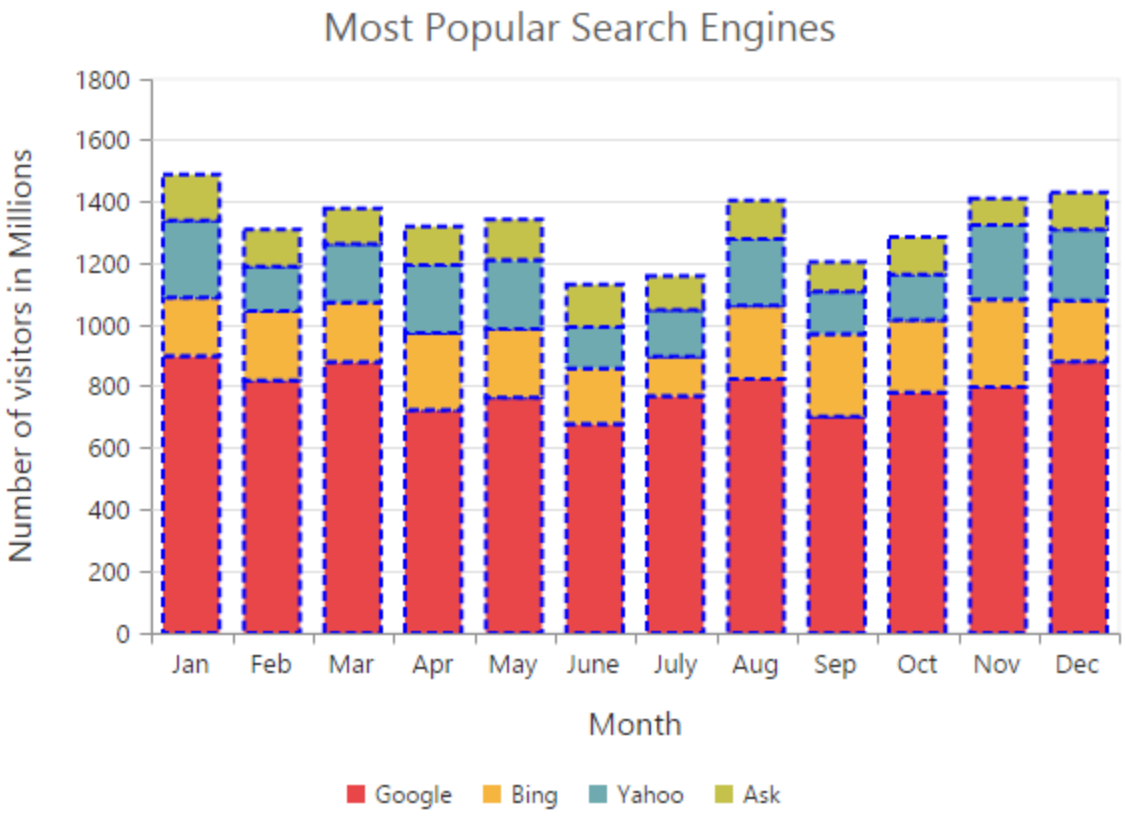


Chart area customization

Customize chart background

The Chart background can be customized by using the **Background** property of the Chart. To customize the chart border, use **Border** option of the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//Customizing Chart background
.Background("skyblue")
//Customize the chart border and opacity
.Border(br=>br.Color("#FF0000").Width(2).Opacity(0.35))
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

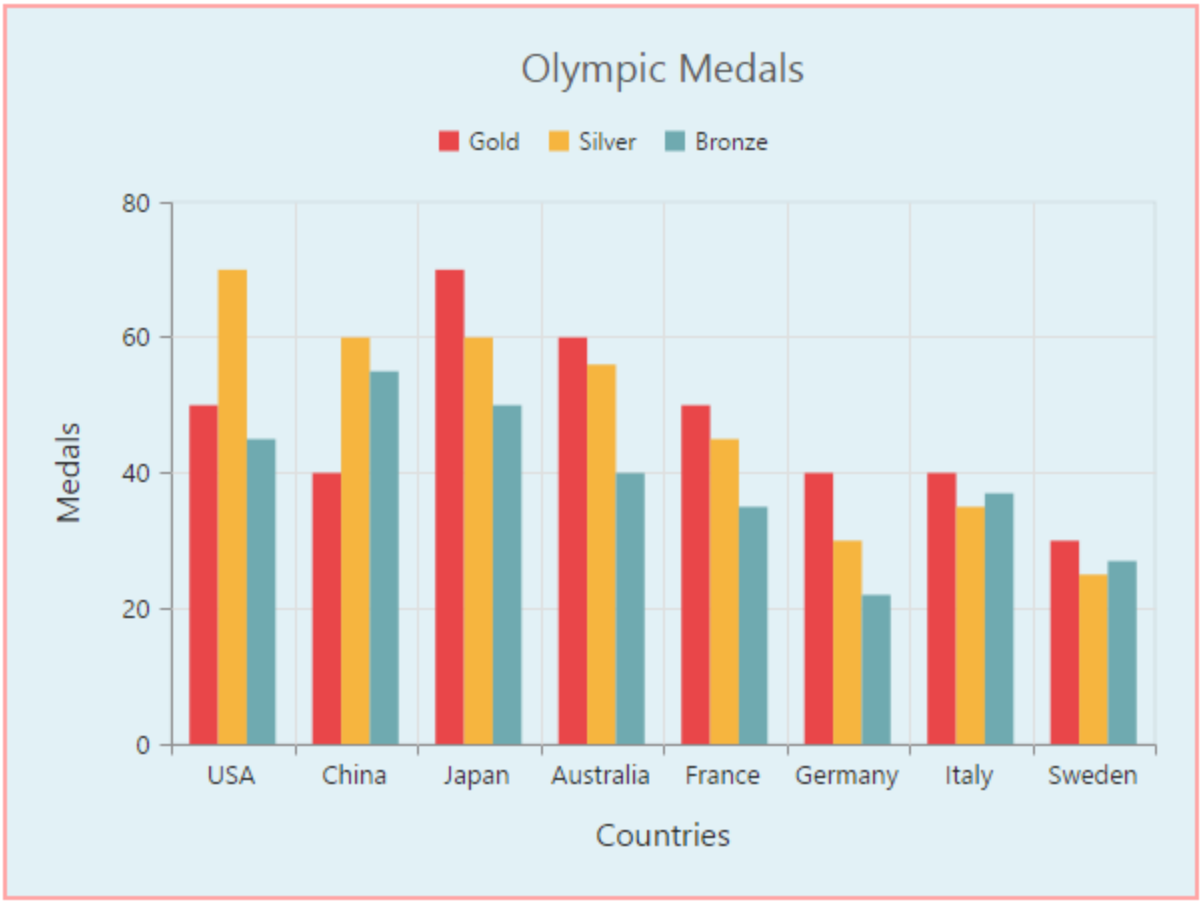


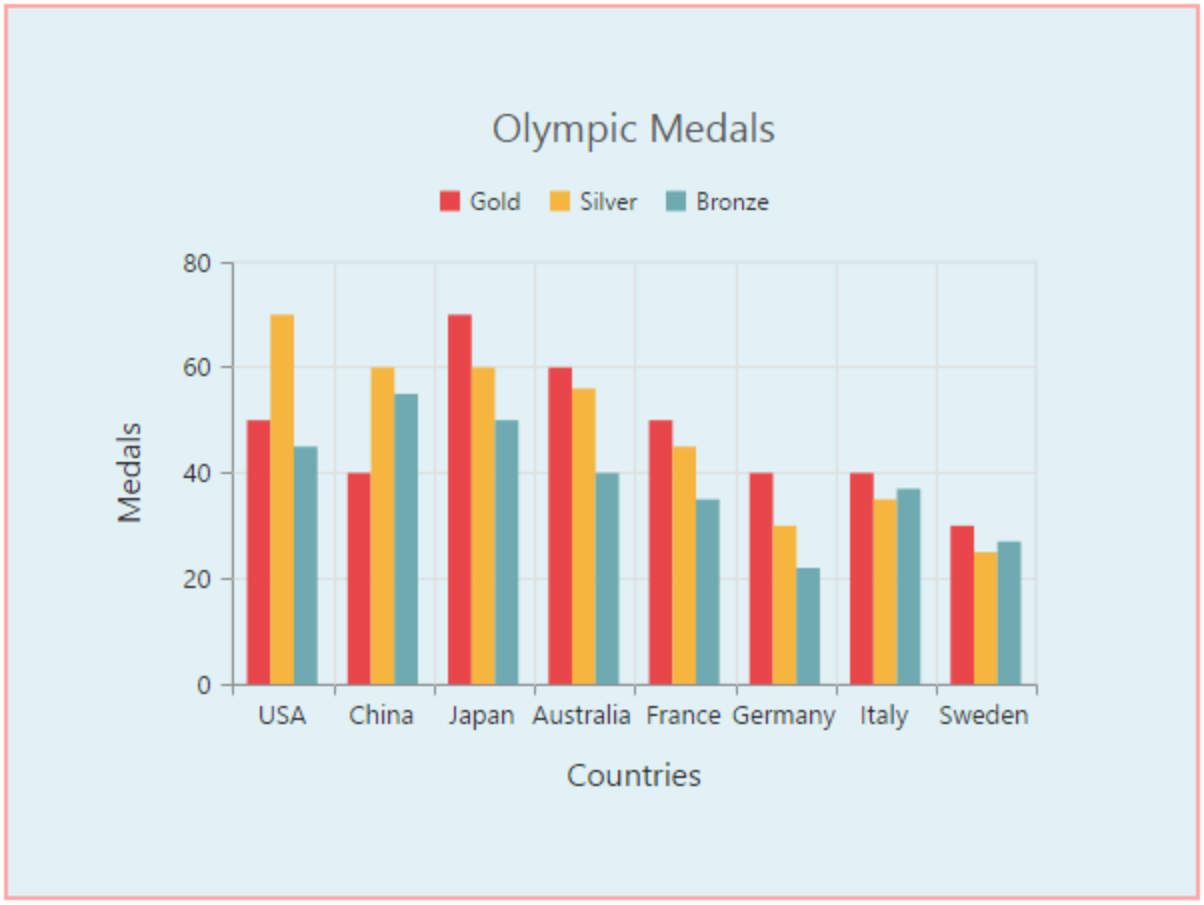
Chart Margin

The Chart **Margin** property is used to add the margin to the chart area at the Left, Right, Top and Bottom position.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
//Change chart margin to left, right, top and bottom
.Margin(mr=>mr.Left(40).Right(40).Bottom(40).Top(40))
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



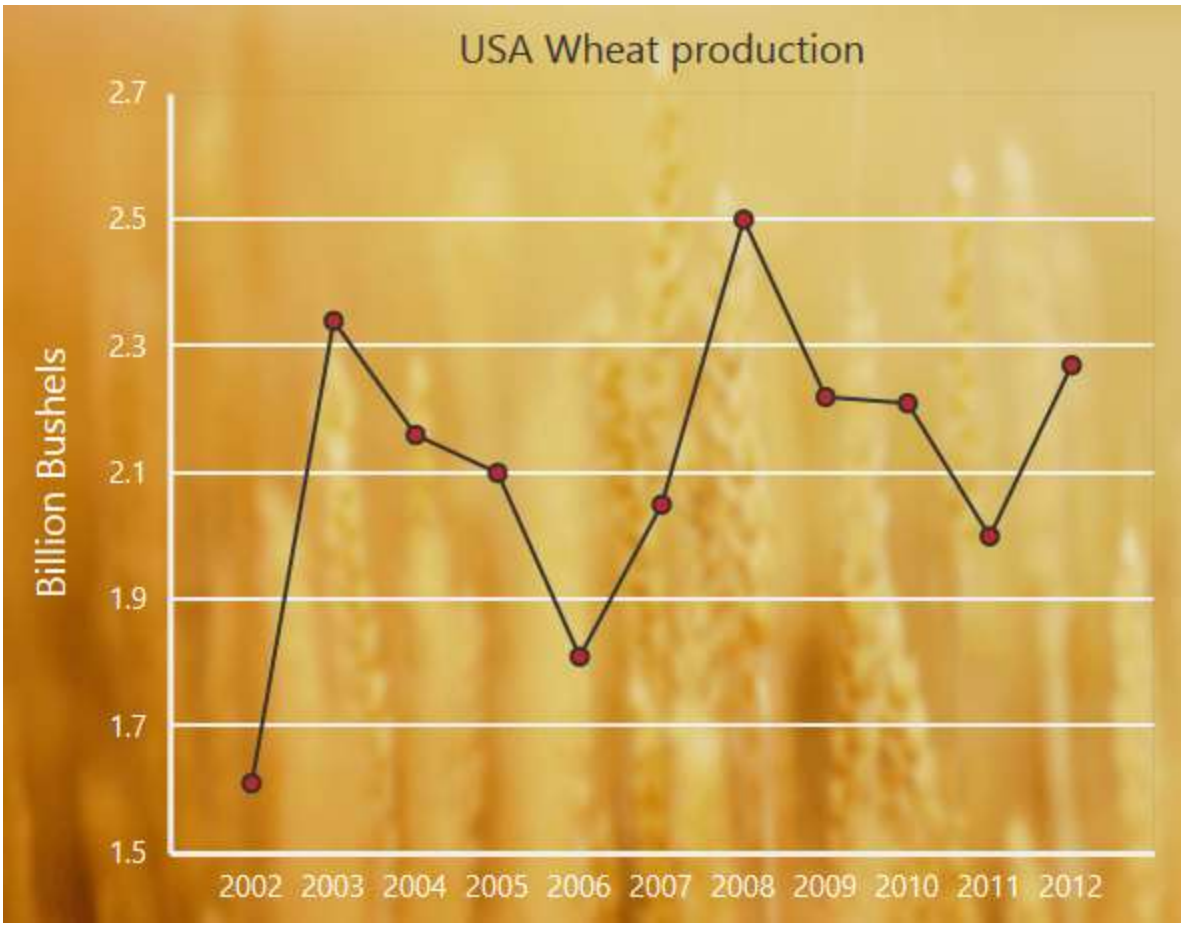
Setting background image

Background image can be added to the chart by using the **BackGroundImageUrl** property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Setting an image as Chart background
.BackGroundImageUrl("images/chart/wheat.png")
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view our online demo sample for setting Chart background image.

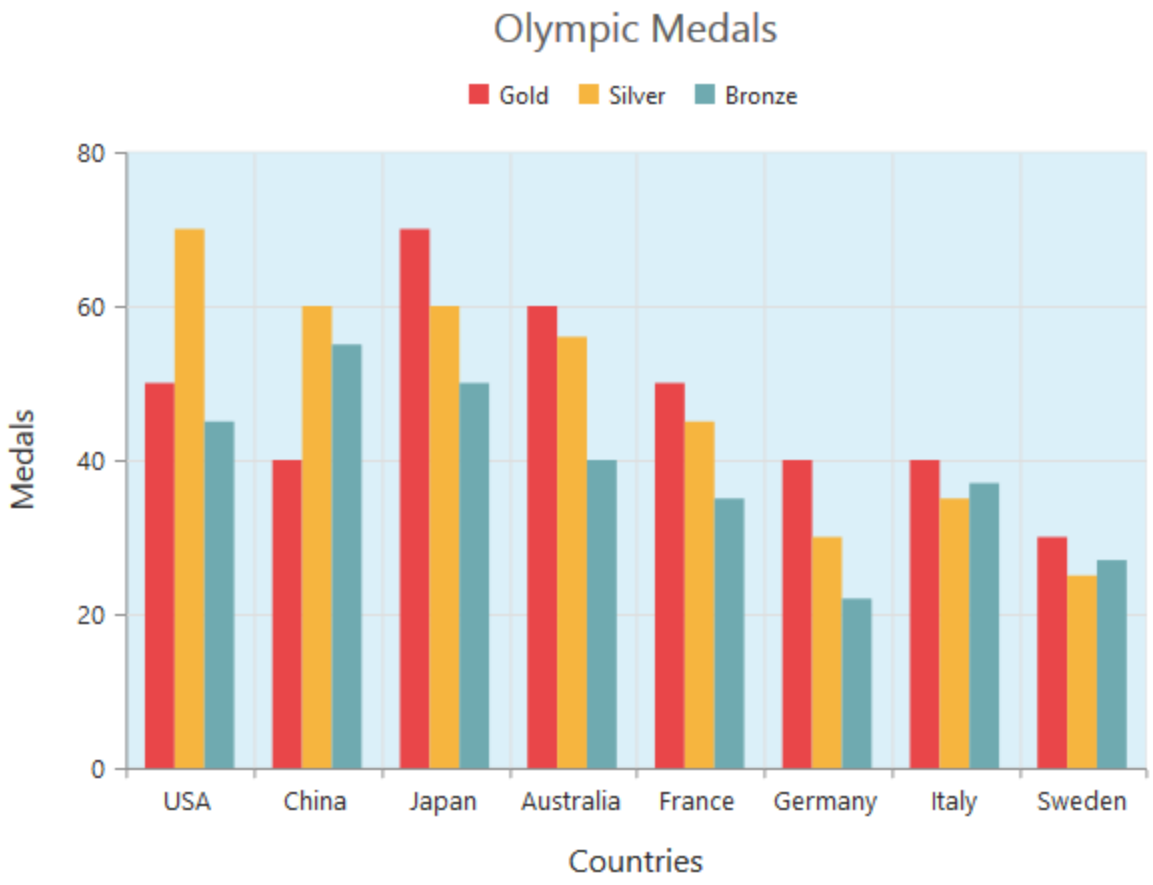
Chart area background

The Chart area background can be customized by using the **Background** property in the chart area.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.ChartArea(ca=>
//Setting background for Chart area
ca.Background("skyblue")
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



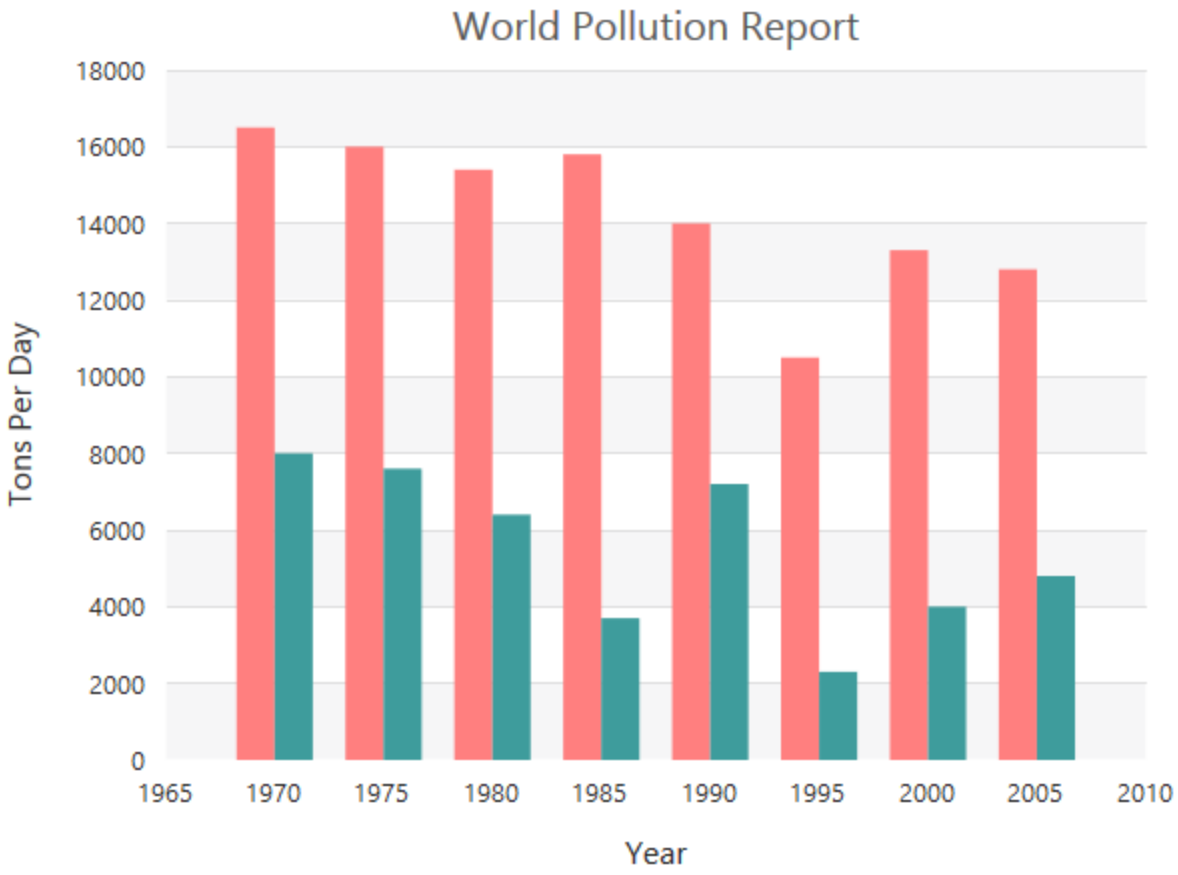
Customize chart area grid bands

You can provide different color for alternate grid rows and columns formed by the grid lines in the chart area by using the **AlternateGridBand** property of the axis. The properties **Odd** and **Even** are used to customize the grid bands at odd and even positions respectively.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.PrimaryYAxis(axis=>
//Customizing horizontal grid bands at even position
axis.AlternateGridBand(band=>band.Even(en=>en.Fill("#A7A9AB").Opacity(0.1)))
)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the AlternateGridBand online demo sample.

Animation

You can enable animation by using the **EnableAnimation** property of the series. This animates the chart series on two occasions – when the chart is loaded for the first time or whenever you change the series type by using the type property.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(
//Enabling animation of series
sr => { sr.EnableAnimation(true).Add();
})
// ...
)
```

However, you can force the chart to animate series by calling the animate method as illustrated in the following code example,

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
.Series(
//Enabling animation of series
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
sr => { sr.EnableAnimation(true).Add();
})
// ...
)
```

JS

```
//Dynamically animating Chart
function animateChart(){
//Calling the animate method for dynamic animation
$("#chartContainer").ejChart("animate");
}
```

Rendering Modes

Chart uses following three rendering technologies

- VML
- SVG
- HTML5 Canvas

VML

VML is used to render chart in IE lower versions such as IE7 and IE8. VML is used by default in these two browsers, as SVG and Canvas are not supported in these browsers.

Limitations:

- Label rotation is not supported.
- Animation is not supported.
- Patterns are not supported.
- Zooming and panning features are not supported.

SVG

SVG is used to render Chart by default for all browsers other than IE7 and IE8.

Canvas

You can switch between SVG and Canvas rendering by using the **EnableCanvasRendering** option. The canvas mode rendering is used in the following scenarios,

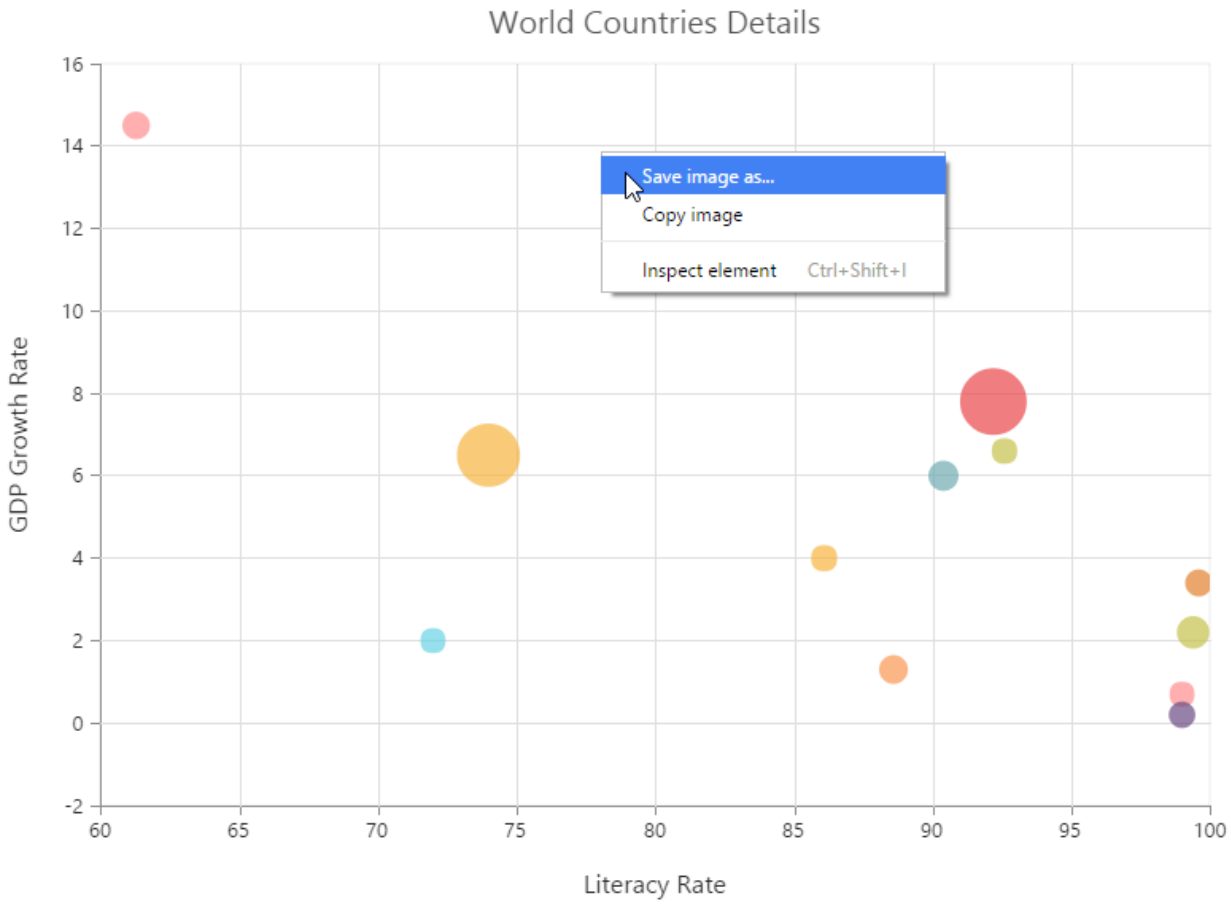
- Plotting large number of data points.
- Performing high frequency live updates.

The following code example shows how to enable HTML5 Canvas rendering in chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Rendering chart in canvas mode
.EnableCanvasRendering(true)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Limitations:

- Animation is not supported.
- 3D charts are not supported.

Real-Time Chart

Chart can be updated dynamically with the real time data. Whenever you add a point or remove a point from the dataSource, you can call the **redraw** method to request the chart to redraw its content.

Note: You can get the chart **instance** using instance method.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Rendering empty Chart without data
.Series(sr =>
{
sr.Points(pt =>
{
}).Add();
})
//...
```

```
)
```

JS

```
//Using set interval to update chart dynamically
window.setInterval(updateChart, 200);
//Function that updates chart dynamically
function updateChart(){
//Creating chart instance
var chart =  $("#chartContainer").ejChart("instance");
if (chart.model.series[0].points.length > 10)
chart.model.series[0].points.splice(0, 1);
var point = chart.model.series[0].points;
var xValue = point.length > 0 ? point[point.length - 1].x + 1 : 1;
point[point.length] = { x:  xValue, y: getRandomNumber( 1000 ) }
//Update Chart dynamically using redraw option
//chart.redraw() can also be used here instead of redraw option
$("#chartContainer").ejChart("redraw");
}
```

Click here to view the online demo sample for real-time Chart.

Exporting Chart

Exporting a chart can be done in both client-side and in server-side. This can be modified by setting values to the property “mode” in exporting. Default value for mode is client.

Client-side Exporting

In client-side rendered chart can be exported as PNG image or as SVG file.

- Export as PNG - The chart can be exported to image when it is rendered in HTML5 Canvas. To render a chart in canvas, set the *EnableCanvasRendering* option to true. To export the chart, you can use the export method of the chart. Refer to the online KB link of exporting chart to know more about chart exporting.
- Export as SVG - Chart can be exported as SVG if it is rendered as a scalable vector graphics element. By default chart will be rendered as SVG.

ASPX-CS

```
<!--Chart download link-->
<a id="download" style="cursor: pointer; position: absolute;right:
150px;">ExportChart</a>
@(Html.EJ().Chart("container").ChartArea(area => area.Border(ab =>
ab.Color("transparent")))
.EnableCanvasRendering(true)
.ExportSettings(ex
=>ex.FileName("ChartSnapshot").Type(ChartExportingType.PNG)))
<script>
function download() {
var canvas = $("#container").ejChart("export");
this.href = canvas;
}
if (document.getElementById('download').addEventListener)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
document.getElementById('download').addEventListener('click', download,
false);
else
document.getElementById('download').attachEvent('onclick', download, false);
</script>
```

Click here to view the Export chart online demo sample.

Server-side Exporting

Server-side operations can be done at the controller page.

Server side implementation

To convert the chart data from client to server-side, refer to the following steps.

1. Create an MVC application and add a controller.
2. Add Syncfusion.EJ, Syncfusion.EJ.MVC, Syncfusion.EJ.Export, Syncfusion.XlsIO, Syncfusion.DocIO and Syncfusion.PDF.Base dll's as references to the application.
3. Specify the name of the method defined in the controller in the property "action" in exporting
4. Currently, the chart data can be exported at server-side only through the helper functions in the ".Net". So to use exporting in your projects, it is required to create a server with any of the following.

i). ASP.NET MVC Controller

ii). ASP.NET Web Form

iii). WebAPI

iv). WCF Service

ASPX-CS

```
<!--Export Chart-->
<button onclick="download()" value="Export">Export</button>
@(Html.EJ().Chart("container").ChartArea(area => area.Border(ab =>
ab.Color("transparent")))
.EnableCanvasRendering(true)
.ExportSettings(ex =>
ex.Type(ChartExportingType.JPG).Action("ExportToImage").Mode(ChartExportingM
ode.Server))
)
<script>
//Export chart to excel
function download() {
$("#container").ejChart("export");
}
</script>
```

At the server side the chart can be exported as JPG, PNG, SVG, PDF, word document and as excel documents. For this the following code have to place in the controller

CSHARP

```
public void ExportChart(string Data, string ChartModel)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// declaration
ChartProperties obj = ConvertChartObject(ChartModel);
string type = obj.ExportSettings.Type.ToString().ToLower();
string fileName = obj.ExportSettings.FileName;
string orientation = obj.ExportSettings.Orientation.ToString();
if (type == "svg")       // for SVG export
{
StringWriter oStringWriter = new StringWriter();
string data = HttpUtility.HtmlDecode(Data);
data = HttpUtility.UrlDecode(Data);
data = System.Uri.UnescapeDataString(Data);
oStringWriter.WriteLine(System.Uri.UnescapeDataString(Data));
Response.ContentType = "text/plain";
Response.AddHeader("Content-Disposition",
String.Format("attachment;filename={0}", (obj.ExportSettings.FileName +
".svg")));
Response.Clear();
using (StreamWriter writer = new StreamWriter(Response.OutputStream))
{
data = oStringWriter.ToString();
writer.Write(oStringWriter.ToString());
}
Response.End();
}
else if (type == "xlsx")        // to export chart as excel
{
List<ExportChartData> chartData = new List<ExportChartData>();
chartData.Add(new ExportChartData("John", 10));
chartData.Add(new ExportChartData("Jake", 12));
chartData.Add(new ExportChartData("Peter", 18));
chartData.Add(new ExportChartData("James", 11));
chartData.Add(new ExportChartData("Mary", 9.7));
ExcelExport exp = new ExcelExport();
exp.Export(obj, (IEnumerable)chartData, fileName + ".xlsx",
ExcelVersion.Excel2010, null, null);
}
else
{
Data = Data.Remove(0, Data.IndexOf(',') + 1);
MemoryStream stream = new MemoryStream(Convert.FromBase64String(Data));
if (type == "docx")         // to export as word document
{
WordDocument document = new WordDocument();
IWSection section = document.AddSection();
IWParagraph paragraph = section.AddParagraph();
if (obj.ExportSettings.Orientation.ToString() == "Landscape")
section.PageSetup.Orientation = PageOrientation.Landscape;
else
section.PageSetup.Orientation = PageOrientation.Portrait;
paragraph.AppendPicture(Image.FromStream(stream));
document.Save(fileName + ".doc", Syncfusion.DocIO.FormatType.Doc,
HttpContext.ApplicationInstance.Response,
Syncfusion.DocIO.HttpContentDisposition.Attachment);
}
else if (type == "pdf")       // to export as PDF
{
PdfDocument pdfDoc = new PdfDocument();
```

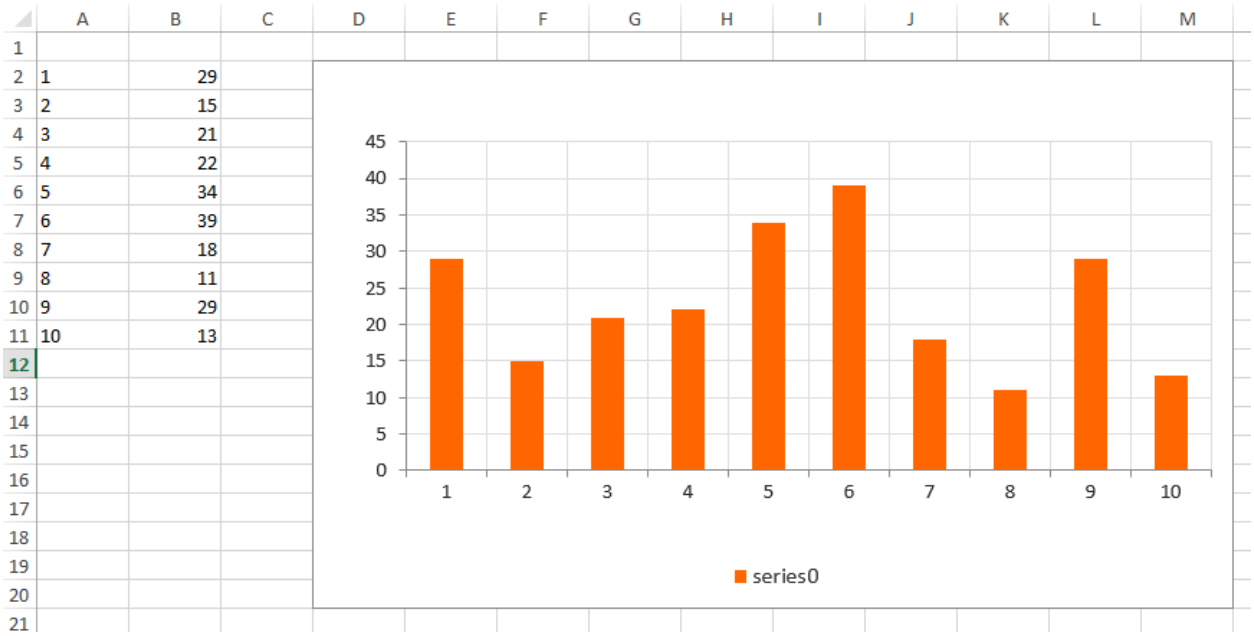
Copyright © 2001 - 2020 Syncfusion Inc.

```
pdfDoc.Pages.Add();
if (obj.ExportSettings.Orientation.ToString() == "Landscape")
pdfDoc.Pages[0].Section.PageSettings.Orientation =
PdfPageOrientation.Landscape;
else
pdfDoc.Pages[0].Section.PageSettings.Orientation =
PdfPageOrientation.Portrait;
pdfDoc.Pages[0].Graphics.DrawImage(PdfImage.FromStream(stream), new
PointF(10, 30));
pdfDoc.Save(obj.ExportSettings.FileName + ".pdf",
HttpContext.ApplicationInstance.Response, HttpReadType.Save);
pdfDoc.Close();
}
else                          // to export as image
{
stream.WriteTo(Response.OutputStream);
Response.ContentType = "application/octet-stream";
Response.AddHeader("Content-Disposition",
String.Format("attachment;filename={0}", fileName + "." + type));
Response.Flush();
stream.Close();
stream.Dispose();
}
}
}
private ChartProperties ConvertChartObject(string ChartModel)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(ChartModel,
typeof(IEnumerable));
ChartProperties chartProp = new ChartProperties();
foreach (KeyValuePair<string, object> d in div)
{
var property = chartProp.GetType().GetProperty(d.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(d.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(chartProp, value, null);
}
}
return chartProp;
}
```

Excel Exporting

Excel exporting is a server-side operation. In addition have to refer Syncfusion.XlsIO assembly to export as excel.

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Chart Excel Exporting

EjChart supports exporting more than one charts in a page, with the *third* argument for the **export** method.

Note: Refer to the MultipleExportType.AppendToSheet, MultipleExportType.NewSheet.

ASPX-CS

```
//Render Chart1
$("#container1").ejChart();
//Render Chart2
$("#container2").ejChart();
//Export multiple chart to excel
function downloadExcel() {
var chart = $("#container1").ejChart("instance");
chart.export('Excel',
'http://js.syncfusion.com/ExportingServices/api/JSChartExport/ExcelExport',
true);
}
```

Export multiple chart to excel at server-side

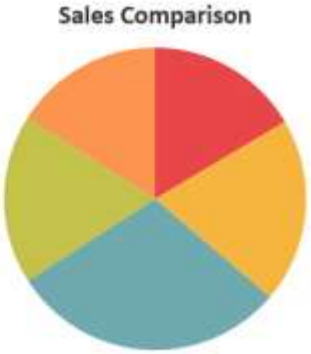
CSHARP

```
public class JSChartExportController : ApiController
{
[System.Web.Http.ActionName("ExcelExport")]
[AcceptVerbs("POST")]
public void ExcelExport()
{
string chartModel = HttpContext.Current.Request.Params["ChartModel"];
IWorkbook book = null;
foreach (string chartProperty in ChartModel)
{
if (chartProperty != null)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
ExcelExport exp = new ExcelExport();
ChartProperties obj = ConvertChartObject(chartProperty);
if (initial)
{
book = exp.Export((obj as ChartProperties), (IEnumerable)data,
"Export1.xlsx",
ExcelVersion.Excel2010, true, null, null);
initial = false;
}
else
{
exp.Export((obj as ChartProperties), (IEnumerable)data, "Export.xlsx",
ExcelVersion.Excel2010, false, book, MultipleExportType.NewSheet, null,
null);
}
}
}
}
}
```





Naming the exported file

EjChart provides options to customize the name of the file to be exported. This can be done by setting the name of the file to the property “fileName” in exporting.

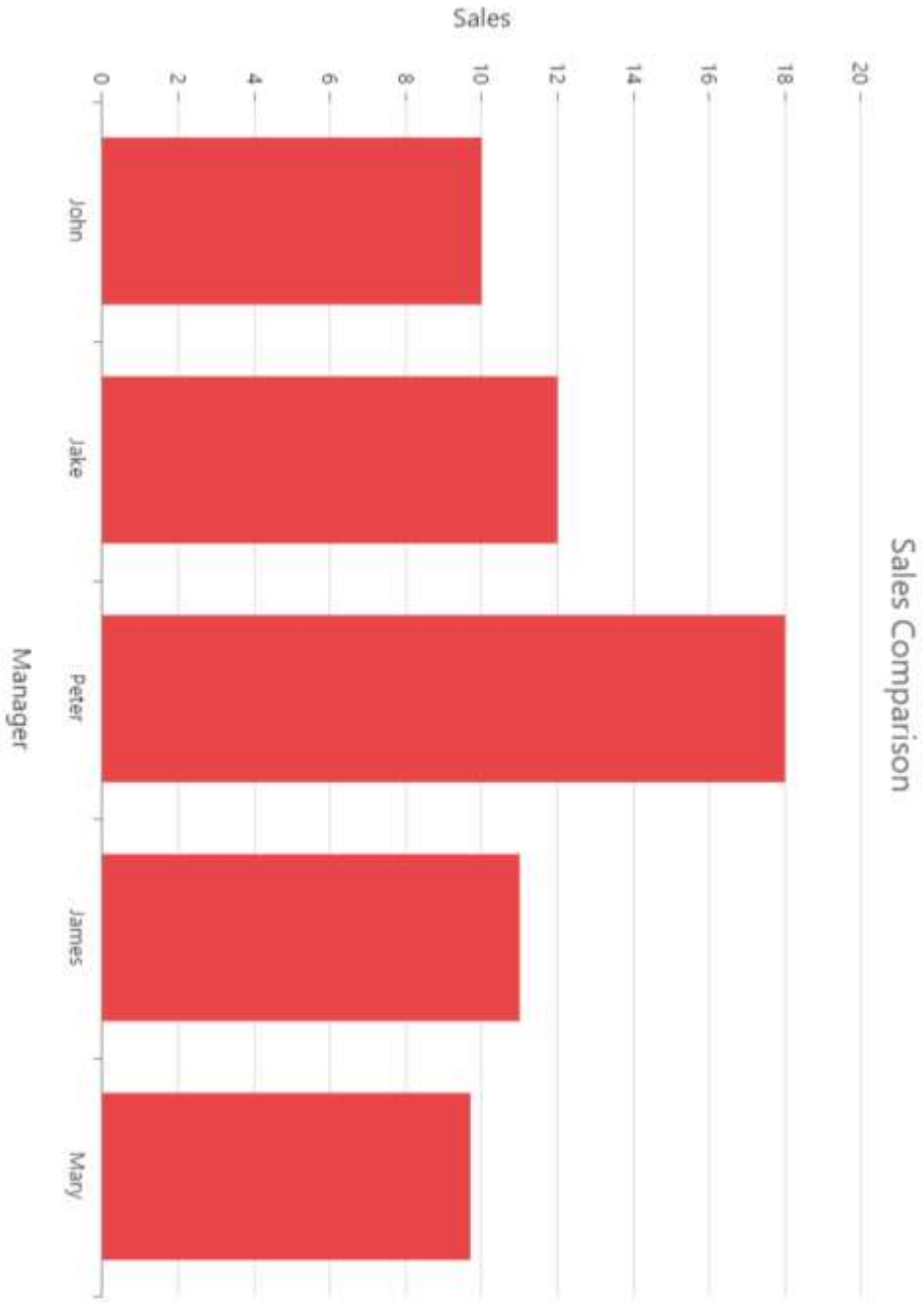
Rotating the chart

We can also rotate the chart and can export it. Possible angles of rotation are 0, 90, -90 and 180 degree. This can be achieved by setting values to the “angle” property in exporting.

ASPX-CS

```
@(Html.EJ().Chart("container")
.ExportSettings(ex => ex.Angle(180).Action("ExportToImage"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Setting orientation for the document

This is applicable for PDF, excel and word documents. By setting values to the “orientation” property in exporting, we can change the orientation of those documents. By default it will export with portrait orientation.

Printing Chart

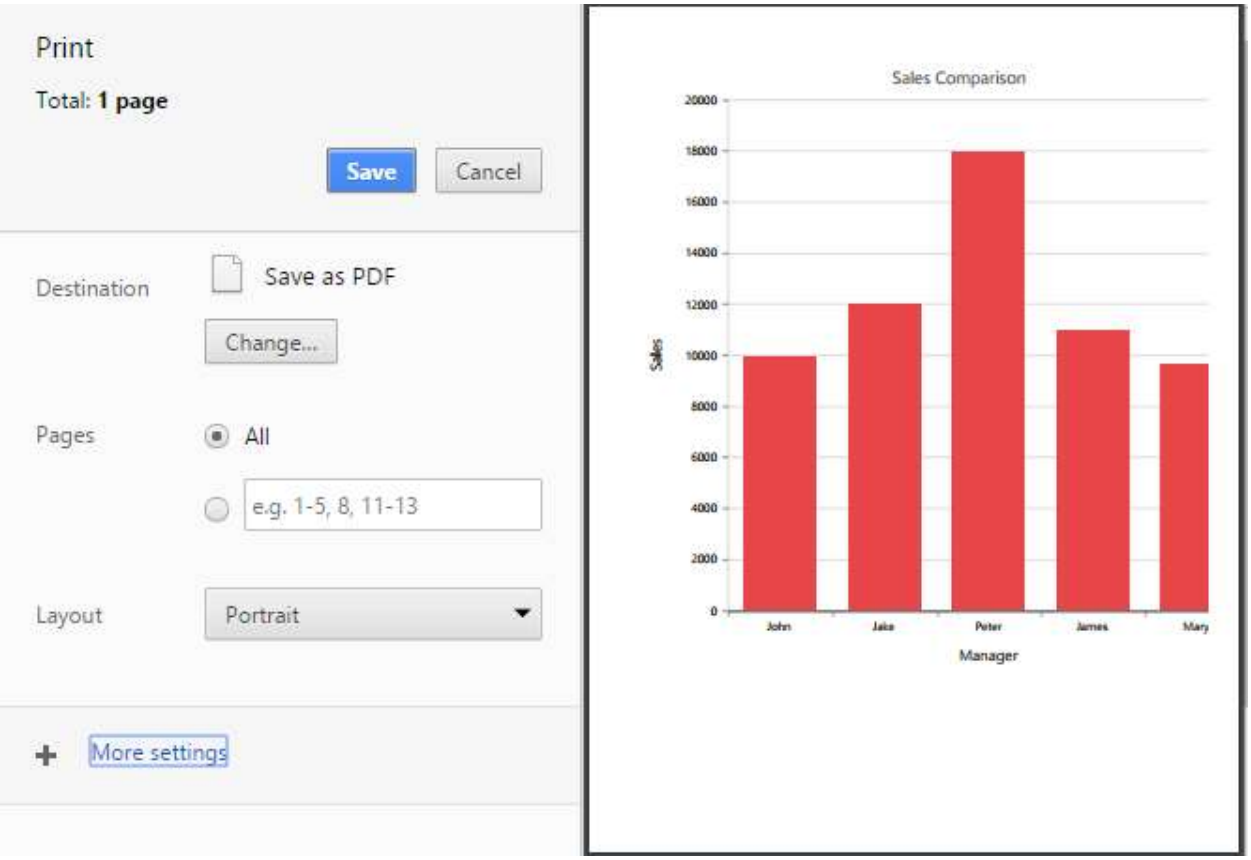
The rendered chart can be printed directly from the browser by calling the public method **print**. ID of the chart div element must be passed as argument to that method.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<button type="button" onclick="print()" ></button>
@(Html.EJ().Chart("container"))
<script>
function print() {
var chartObj = $("#container").ejChart("instance");
chartObj.print("container");
}
</script>
```

This print method can be called by performing any action on the web page. For example, by clicking a button. While calling the print method in chart, print preview will be displayed in the browser.



Click here to view the Printing chart online demo sample

Printing Multiple chart

Multiple charts in a web page can be printed together. For printing multiple charts, ID of the chart div elements have to be passed as shown in the below code

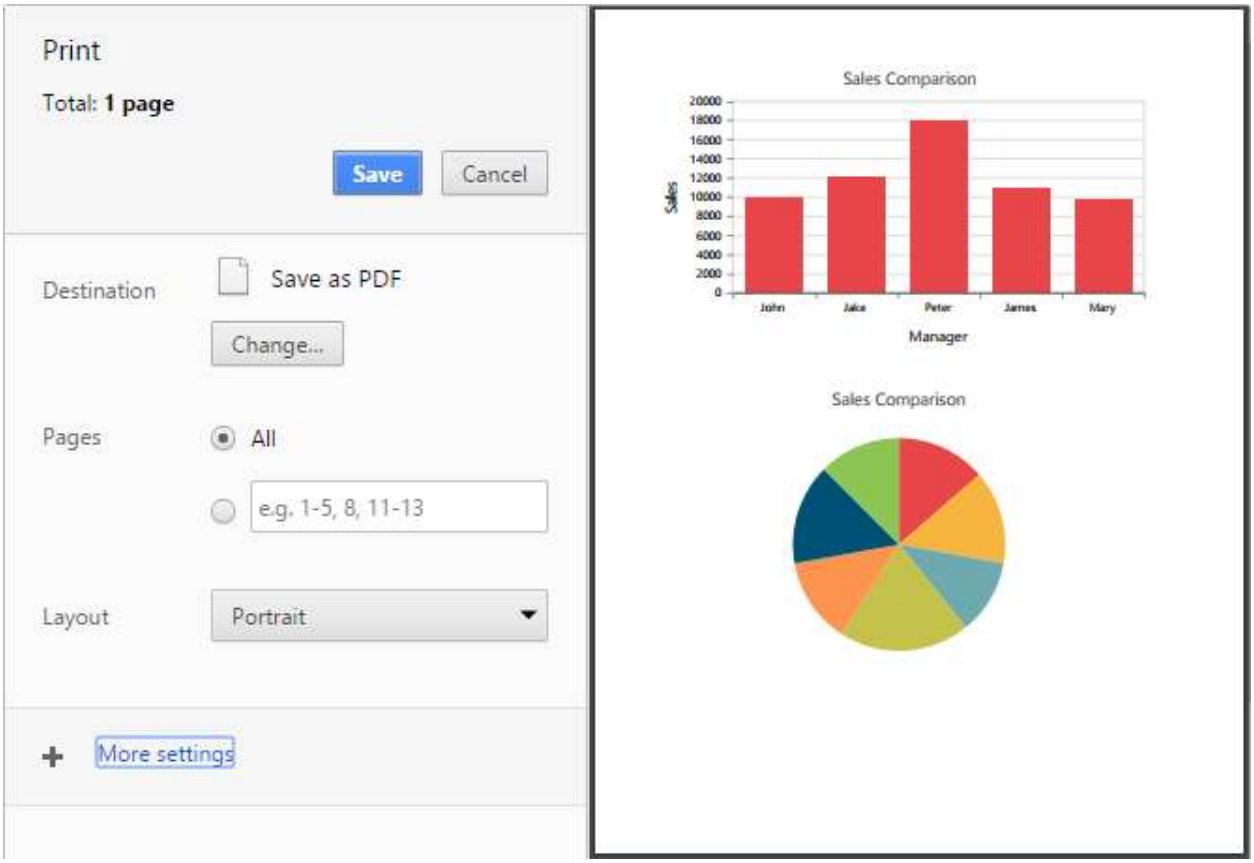
ASPX-CS

```
<button type="button" onclick="print()" ></button>
@(Html.EJ().Chart("container1"))
@(Html.EJ().Chart("container2"))
<script>
function print() {
var chartObj = $("#container1").ejChart("instance");
chartObj.print("container1","container2");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

The Print preview of multiple Charts is shown below



Page Setup

Some of print options are not configurable through JavaScript code. You need to customize layout, paper size, margins options through browser's page setup dialog. Please find the following guidelines link to browser page setup.

- Chrome
- Firefox
- Safari
- IE

3D Chart

Essential 3D Chart for ASP.NET MVC allows you to view 8 chart types in 3D view such as Bar, StackingBar, StackingBar100, Column, Stacked Column, 100% Stacked Column, Pie, Doughnut.

3D Column Chart

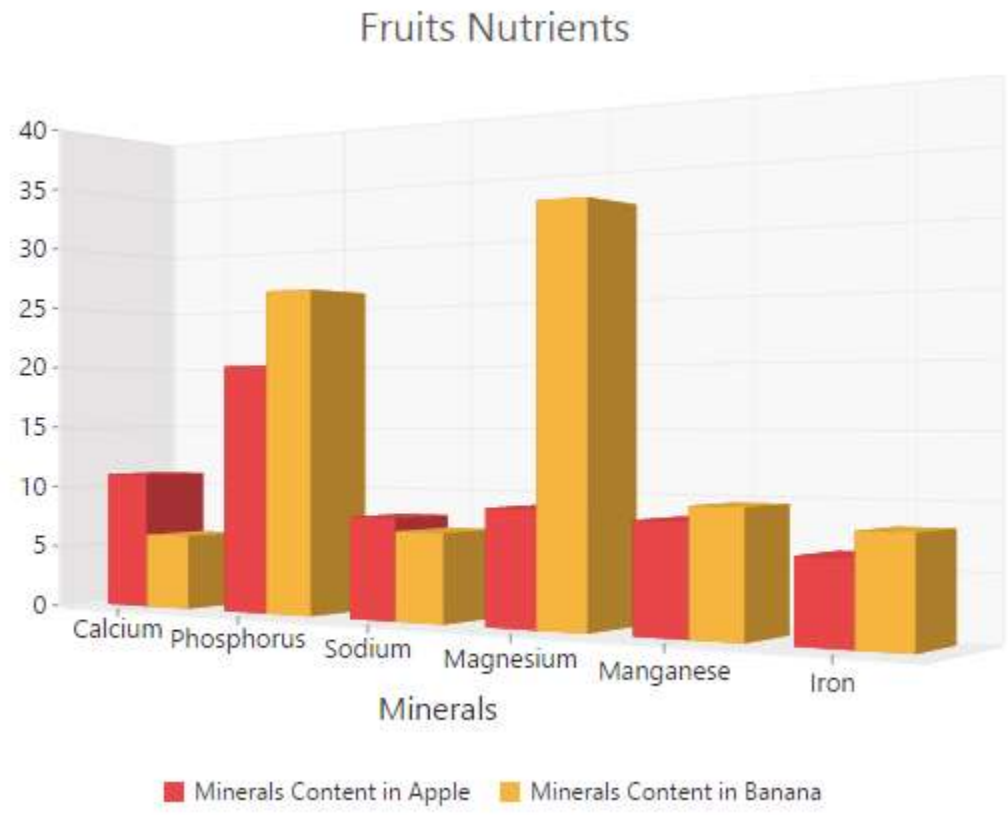
For rendering a 3D Column Chart, specify the series **Type** as **Column** in the chart series and set **Enable3D** option as **true** in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Series(sr => { sr.Type(SeriesType.Column).Add(); })
// Enable 3D Chart
.Enable3D(true)
// ...
)
```



Click here to view the Column 3DChart online demo sample.

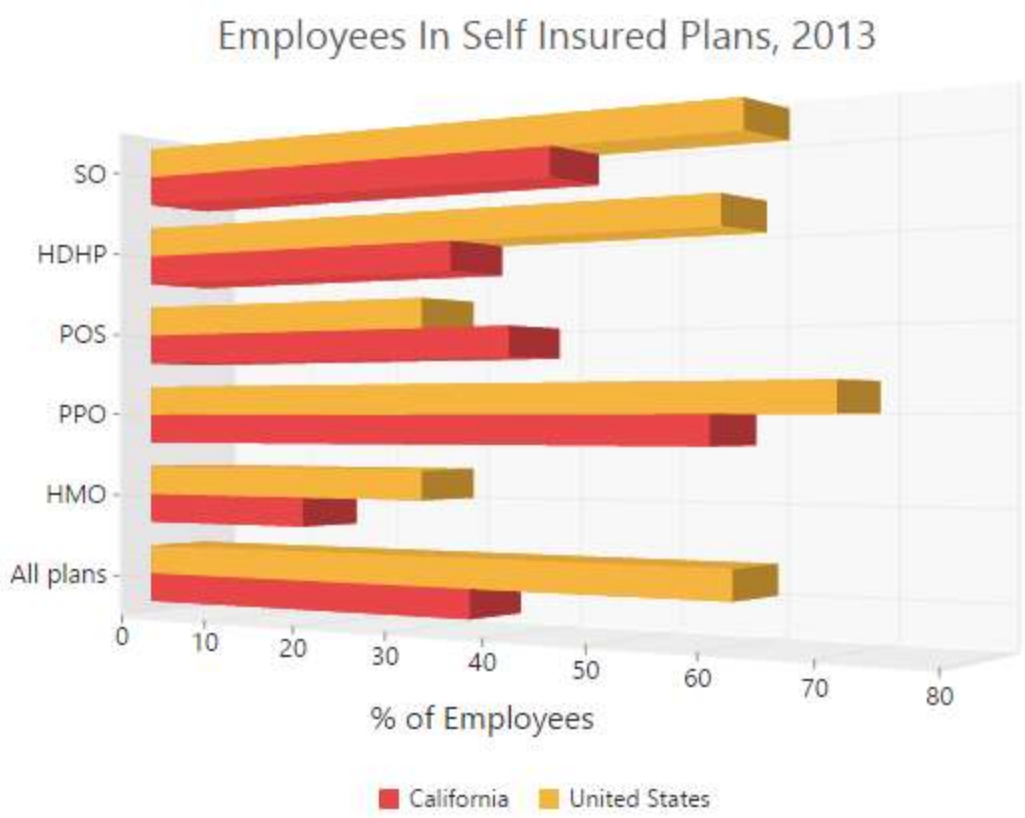
3D Bar Chart

You can create a 3D Bar Chart by setting the series **Type** as **Bar** in the chart series and enable Enable3D option in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => { sr.Type(SeriesType.Bar).Add(); })
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Bar 3DChart online demo sample.

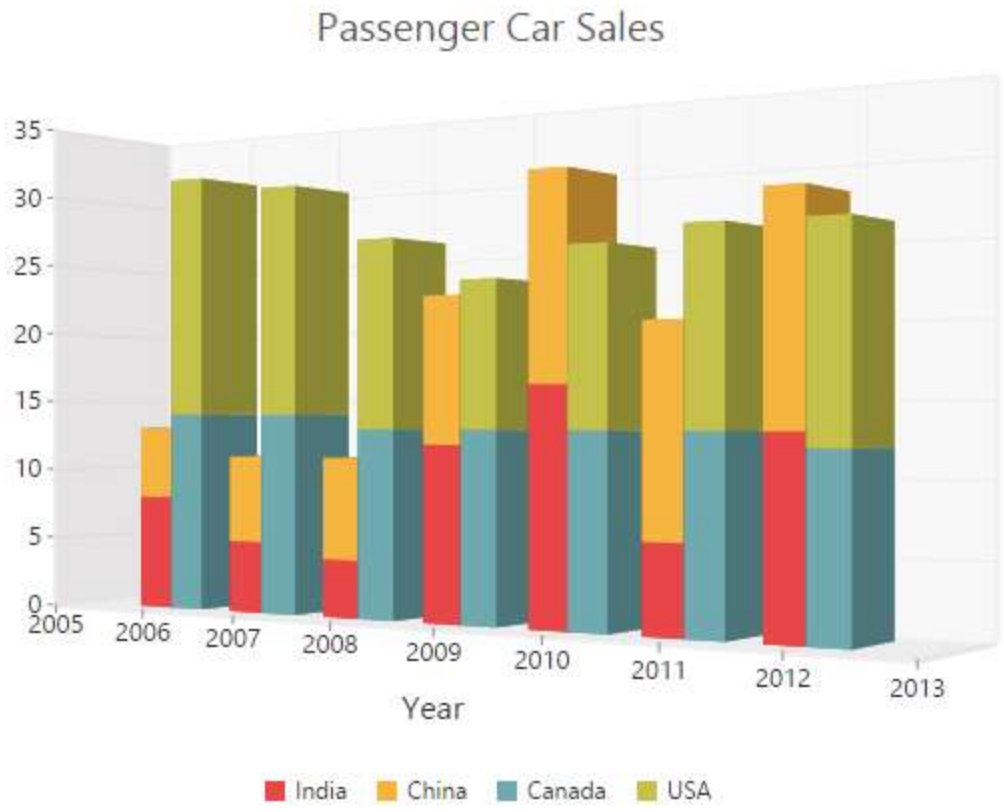
3D Stacked Column Chart

You can create a 3D Stacked Column Chart by setting the series **Type** as **StackingColumn** in the chart series and enable Enable3D option in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => {
//Add First Series
sr.Type(SeriesType.StackingColumn).Add();
//Add Second Series
sr.Type(SeriesType.StackingColumn).Add();})
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Stacked Column 3DChart online demo sample.

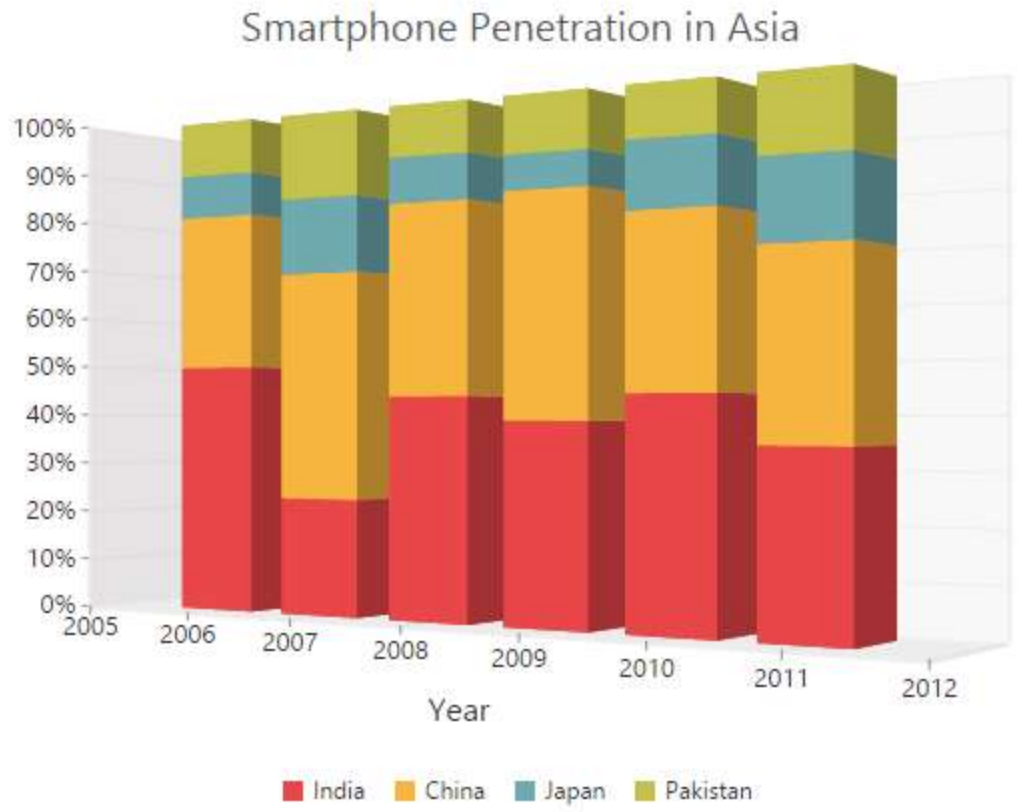
3D 100% Stacked Column Chart

100% Stacking Column 3DChart is rendered by specifying the series **Type** as **StackingColumn100** in the chart series and enable Enable3D option in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => {
//Add First Series
sr.Type(SeriesType.StackingColumn100).Add();
//Add Second Series
sr.Type(SeriesType.StackingColumn100).Add();})
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the 100% Stacked Column 3DChart online demo sample.

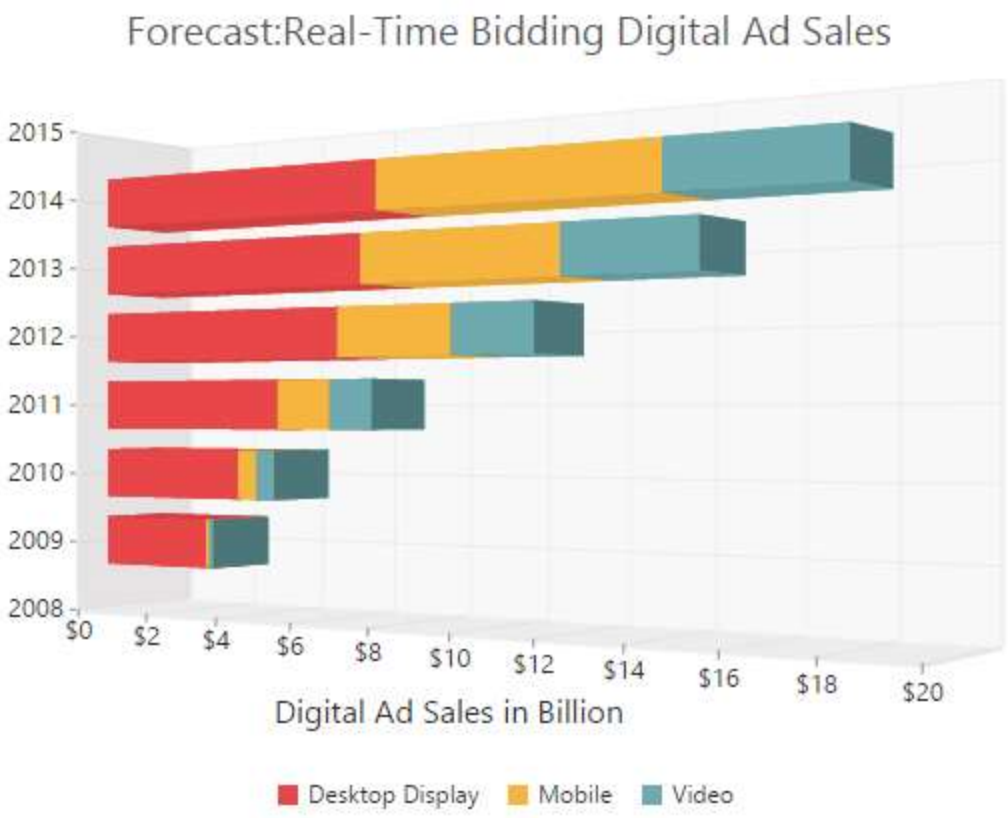
3D Stacked Bar Chart

To create Stacking Bar 3DChart, set the series **Type** as **StackingBar** in the chart series and enable Eenable3D option in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => {
//Add First Series
sr.Type(SeriesType.StackingBar).Add();
//Add Second Series
sr.Type(SeriesType.StackingBar).Add();})
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Stacked Bar 3DChart online demo sample.

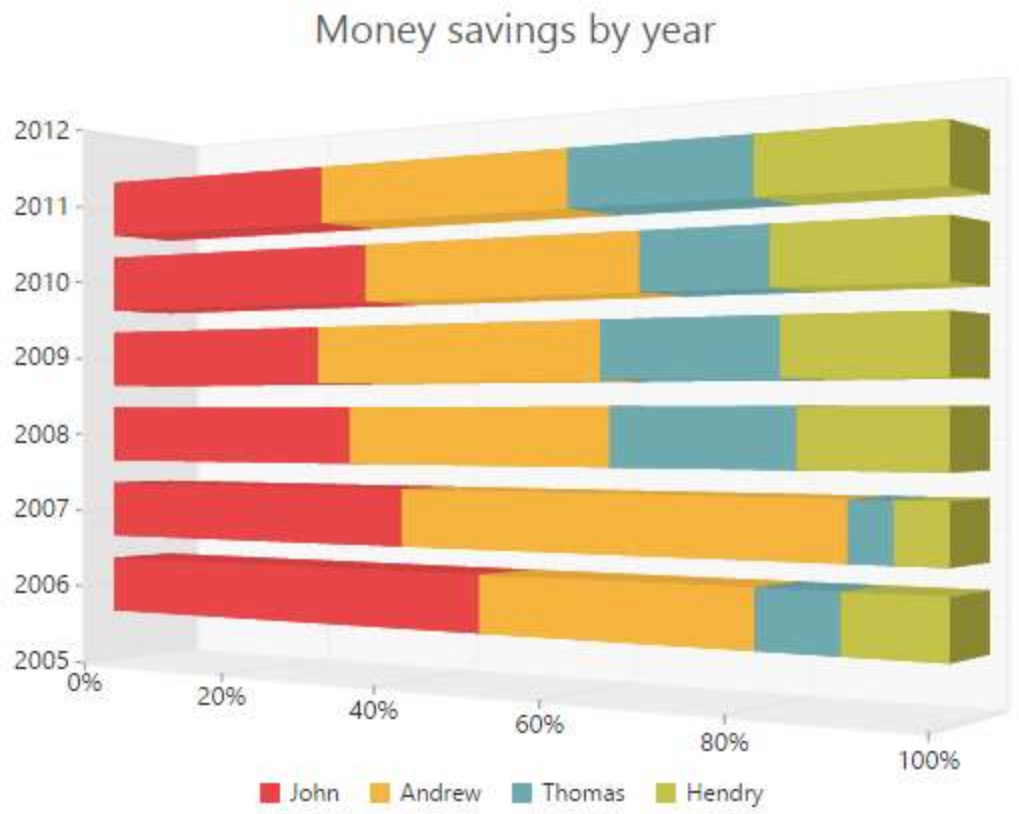
3D 100% Stacked Bar Chart

You can create 100% Stacking Bar 3DChart by setting the series **Type** as **StackingBar100** in the chart series and enable Eenable3D option in chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => {
//Add First Series
sr.Type(SeriesType.StackingBar100).Add();
//Add Second Series
sr.Type(SeriesType.StackingBar100).Add();})
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the 100% Stacked Bar 3DChart online demo sample.

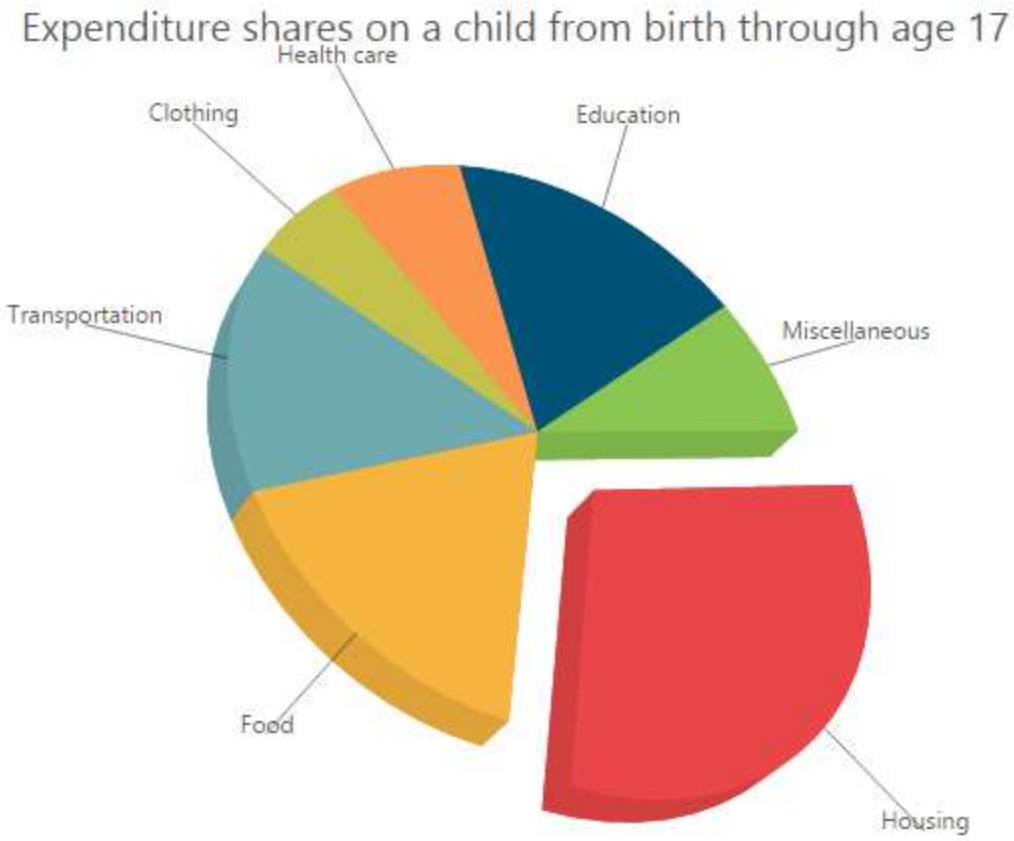
3D Pie Chart

To render the Pie Chart in 3D view, enable the **Enbel3D** option in the chart and set the series **Type** as **Pie** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => {
//Set chart type to series
sr.Type(SeriesType.Pie).Add();})
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Pie 3DChart online demo sample.

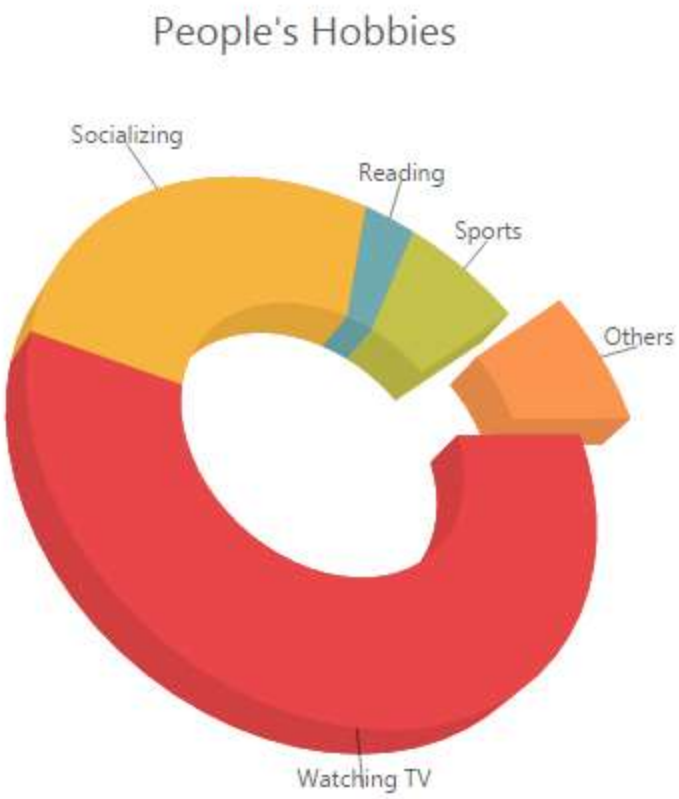
3D Doughnut Chart

To render the Doughnut Chart in 3D view, enable the **Enbel3D** option in the chart and set the series **Type** as **Doughnut** in the chart series.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
.Series(sr => {
//Set chart type to series
sr.Type(SeriesType.Doughnut).Add();})
// Enable 3D Chart
.Enable3D(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the Doughnut 3DChart online demo sample.

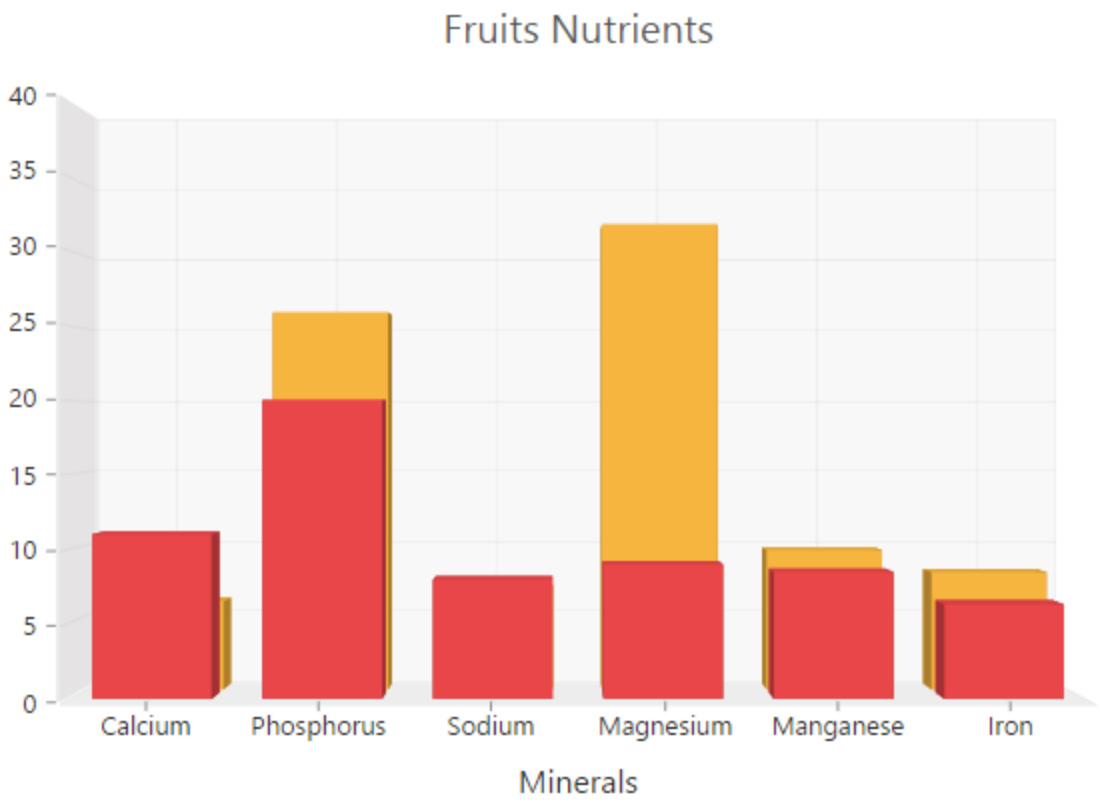
Configure 3D Chart

3D View

To render the EjChart in 3D view, set the **Enable3D** option as **true** in the chart.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// Enable 3D Chart
.Enable3D(true)
// ...
)
```



Click here to view the 3DChart online demo sample.

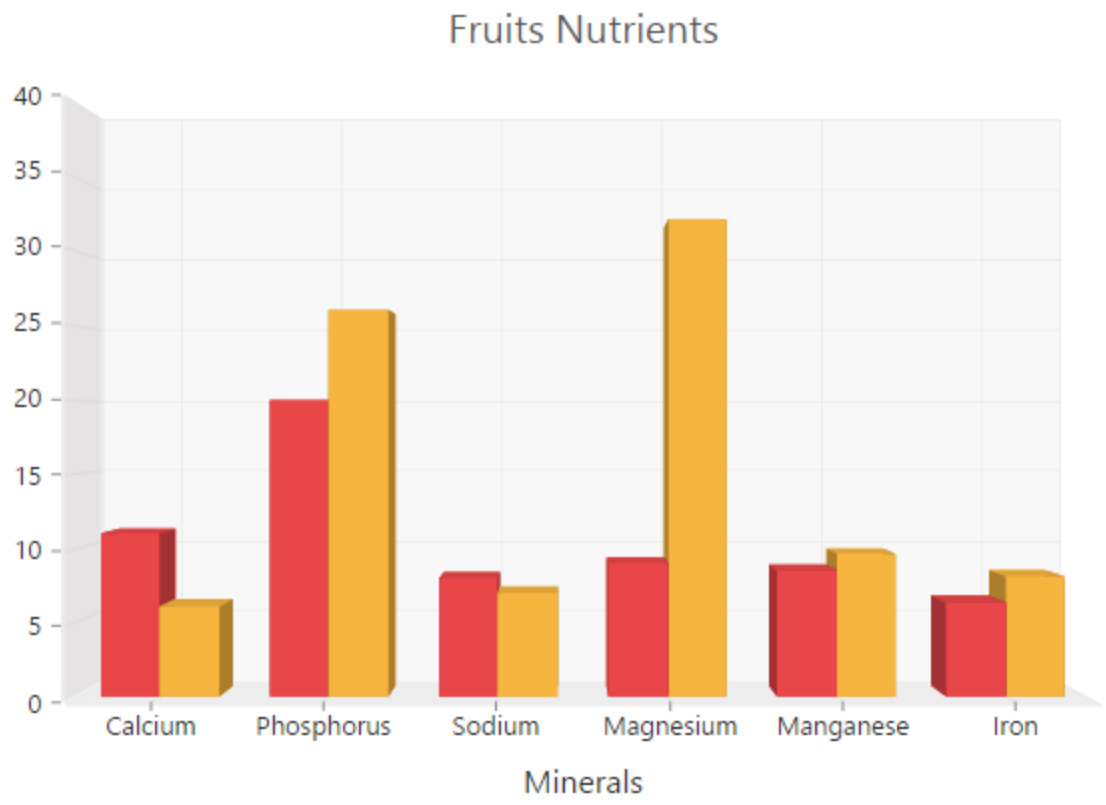
Placing Bar / Column kind of series side-by-side

The **SideBySideSeriesPlacement** defines the appearance of the different sets of data on the 3D Chart. When this property is enabled, the data is displayed side by side, otherwise it is displayed along the depth of the axis.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Enable SideBySideSeriesPlacement for 3DChart
.SideBySideSeriesPlacement(true)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



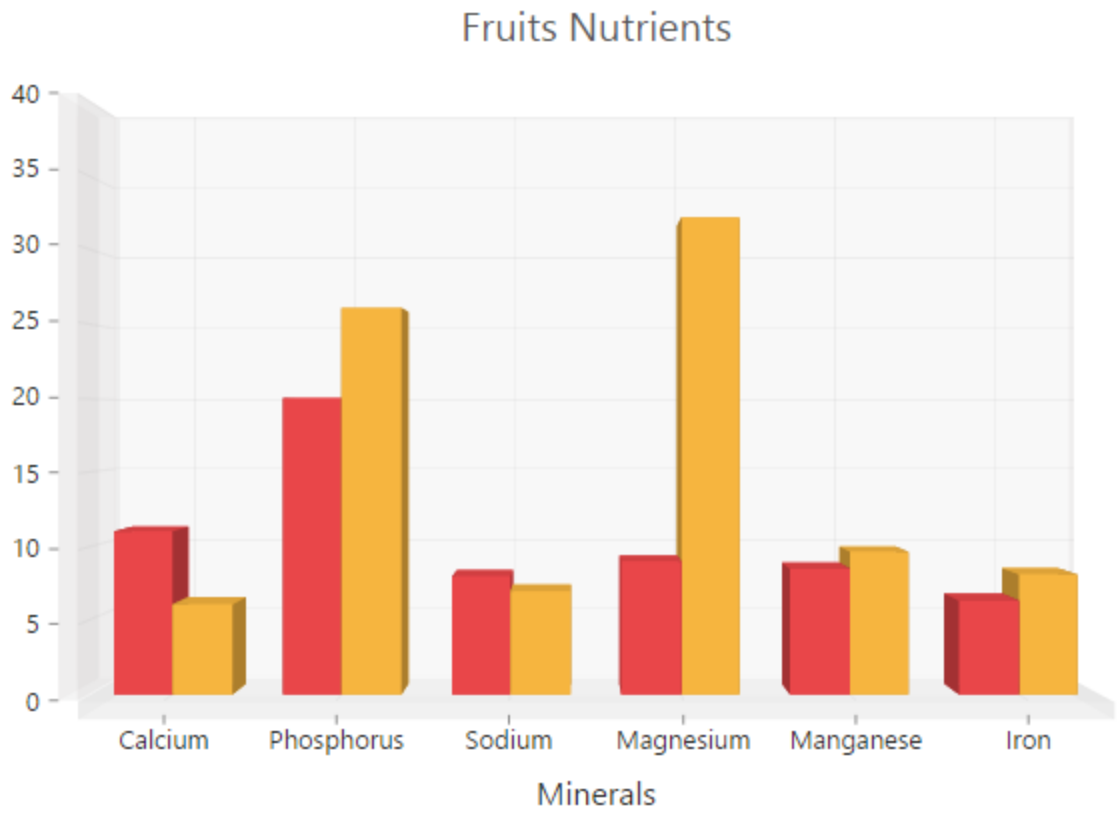
Setting Axis Wall Size

In 3DChart, Cartesian axes lines are represented as walls and it defines the width of the 3D wall. 3D Pie and Doughnut Chart does not support **WallSize** because they don't have axes.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Change 3DChart axis wall size
.WallSize(10)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

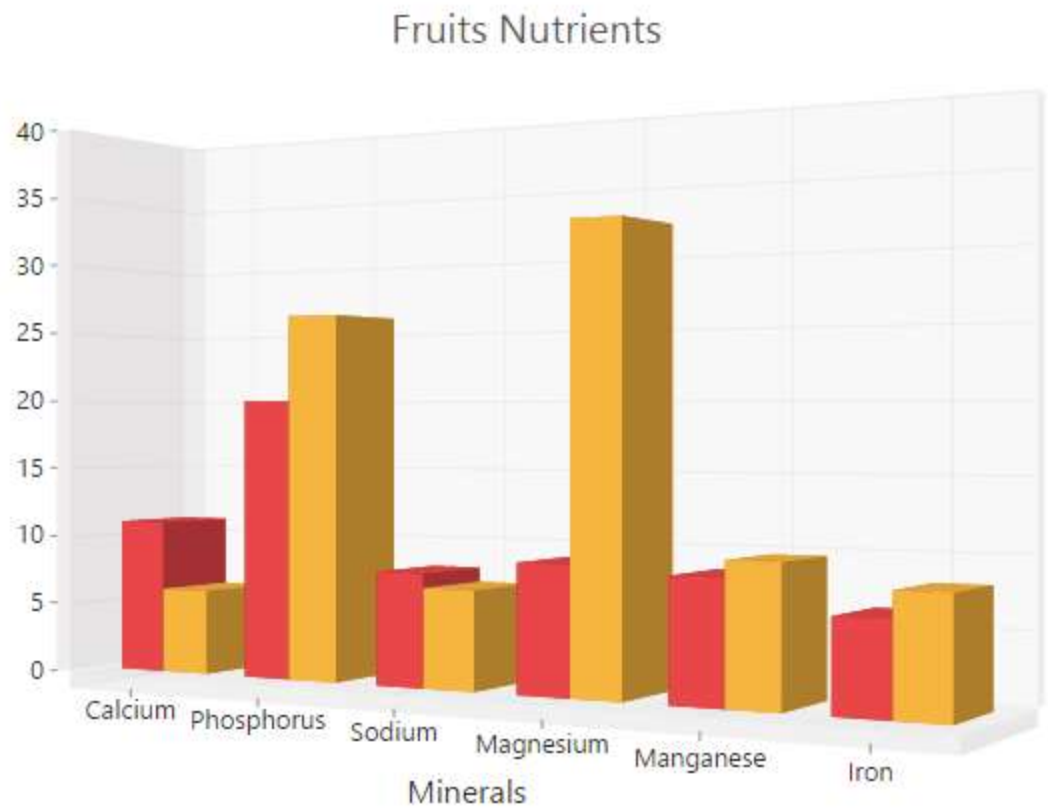


3D Depth

By using the **Depth** property, you can view the 3D Chart from the front view of the series to the background wall.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Change 3DChart depth value
.Depth(120)
// ...
)
```



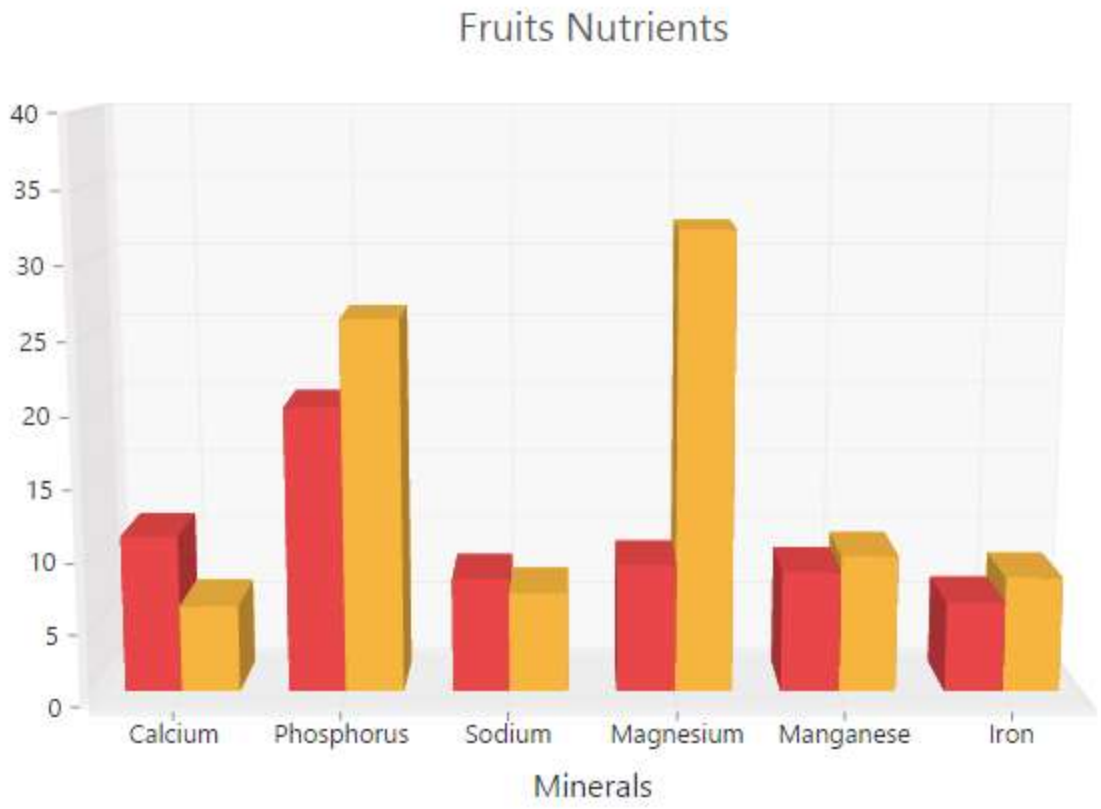
Rotating and Tilting 3D Chart

To spin the 3D Chart on mouse dragging, enable **EnableRotation** option in the chart. The **Tilt** property specifies the angle of the slope of the 3D Chart. The positive and negative values are declared to the side where the slope is present. The **Rotation** option is used to rotate the 3D chart towards left or right side of the chart. The direction of the chart depends upon the positive and negative values of the angle.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Change 3DChart tilt value
.Tilt(10)
//Enable 3DChart  rotation  on mouse dragging
.EnableRotation(true)
//Change 3DChart rotation on chart load
.Rotation(40)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



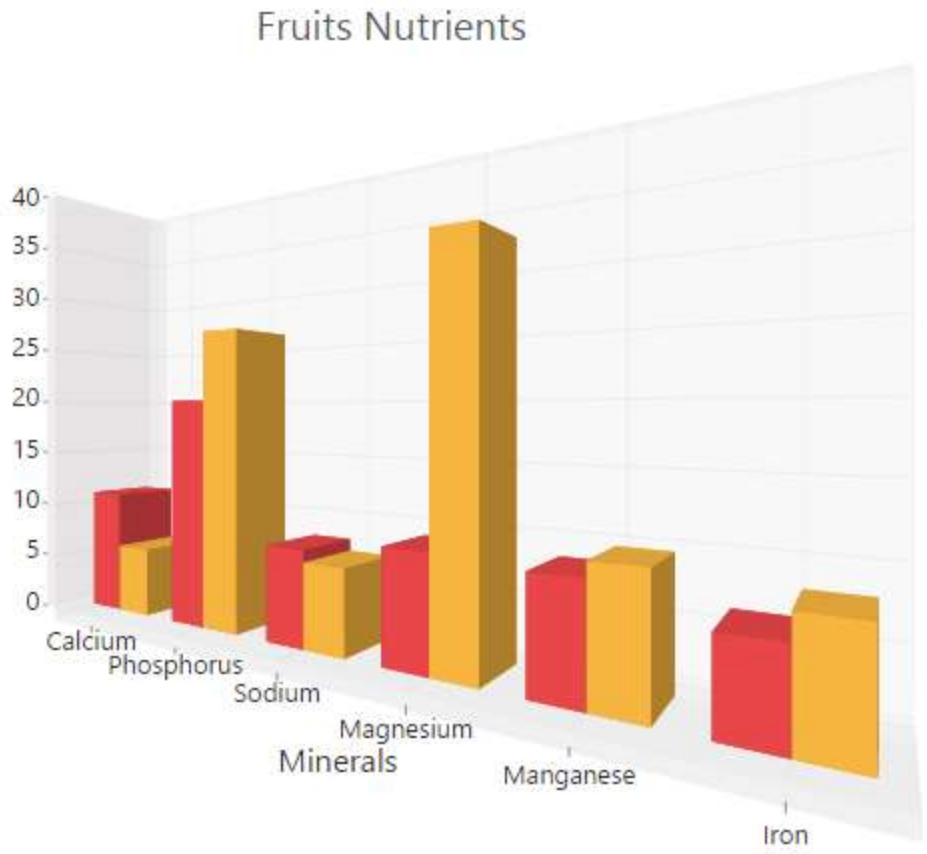
PerspectiveAngle

The **PerspectiveAngle** specifies the appearance of the height, width, depth and wall of the 3D Chart. When the PerspectiveAngle is decreased, the 3D object appears very close to the viewer. But when it is increased, the 3D object appears far away from the viewer.

ASPX-CS

```
@(Html.EJ().Chart("chartContainer")
// ...
//Change 3DChart perspective angle
.PerspectiveAngle(150)
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



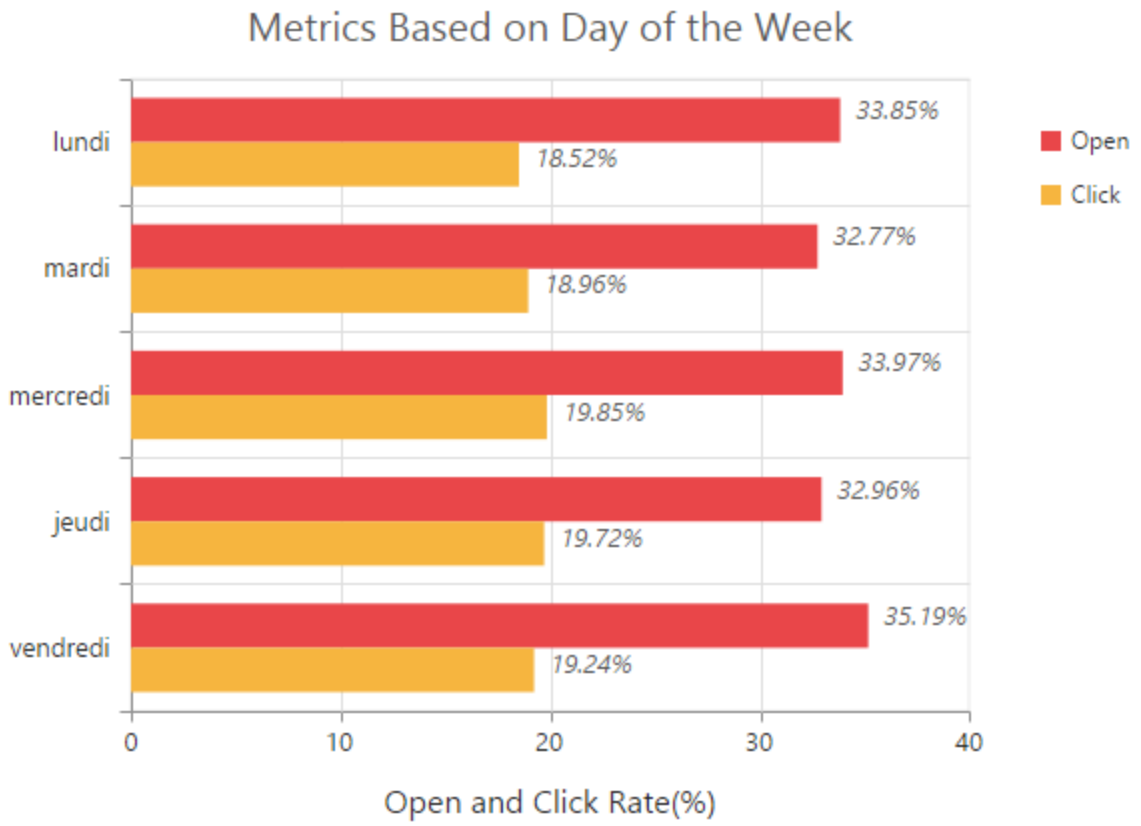
Localization

EjChart supports localization for its axis labels and tooltip. To render the chart with specific culture you have to refer the corresponding **globalize** culture script and need to specify the culture name in **Locale** property of chart.

ASPX-CS

```
<!--Refer french globalize culture script-->
<script src="../scripts/cultures/globalize.culture.fr-FR.min.js"></script>
@(Html.EJ().Chart("chartContainer")
// ...
//Render chart in French locale
.Locale("fr-FR")
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the localization chart online demo sample.

Checkbox

Checkbox

The Essential ASP.NET MVC Checkbox control allows you to check an option to perform an action. This control allows you to select true, false or an intermediate option. These checkboxes are supported with themes. The HTML check box control is rendered as Essential ASP.NET MVC Checkbox control.

Key Features

- Trendy Look : Rich Appearance with Theme Support
- RTL : Supports for Right to Left alignment
- Tri-State: Supports Checkbox with three states.
- Easy Customization: The customization of Checkbox is made simple

Getting Started

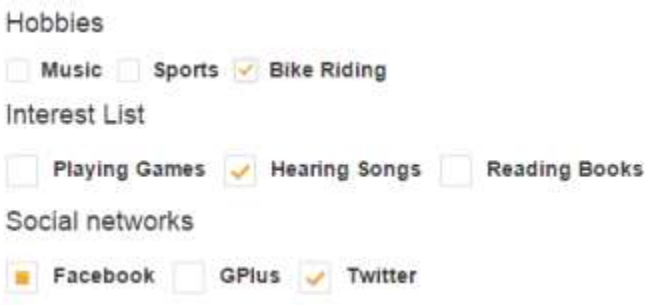
This section explains briefly about how to create a Checkbox in your application with ASP.NET MVC.

Create your first Checkbox in MVC

Essential ASP.NET MVC Checkbox provides support to multiple selections within your web page, and allows you to select options from the list. Refer the following guidelines to create multiple or single

Copyright © 2001 - 2020 Syncfusion Inc.

selection List Receiving App using Checkbox that helps you to get the value of checked Checkbox using the button. The following screenshot illustrates the functionality of Checkbox with button action.



Checkboxes Control

In the above screenshot, you can select hobbies, interest list and social networks receiving app using Checkbox. The Checkbox performs the action to render the checked values when button clicked.

Create a Checkbox

ASP.NET MVC Checkbox widget has built-in features like multiple selections. You can easily create the Checkbox widget using simple HTML helper "@Html.EJ().CheckBox()" as follows.

1. You can create an MVC Project and add necessary assemblies, styles and scripts with the help of the given MVC-Getting Started Documentation.
2. Add the following code to the corresponding view page to render Checkbox.

HTML

```
<div class="frame">
<div class="control">
<div class="chkalign">
<h4>Hobbies</h4>
<table>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("check1").Value("Music")
<label for="check1" class="clslab">
Music
</label>
</td>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox3").Value("Sports")
<label for="Checkbox3" class="clslab">
Sports
</label>
</td>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox4").Value("Bike Riding")
<label for="Checkbox4" class="clslab">
Bike Riding
</label>
</td>
</tr>
</table>
```

```
<h4>Interest List</h4>
<table>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox1").Value("Playing Games").Size(Size.Medium)
<label for="Checkbox1" class="clslab">Playing Games</label>
</td>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox5").Value("Hearing Songs").Size(Size.Medium)
<label for="Checkbox5" class="clslab">Hearing Songs</label>
</td>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox6").Value("Reading Books").Size(Size.Medium)
<label for="Checkbox6" class="clslab">Reading Books</label>
</td>
</tr>
</table>
</div>
</div>
</div>
```

3. Add the following styles to the corresponding view page to show the Checkbox in horizontal order.

CSS

```
<style>
td {
float: left;
}
label
{
float: right; margin:5px 10px;
}
</style>
```

4. Run the above code to render the following output.



Checkbox Creation

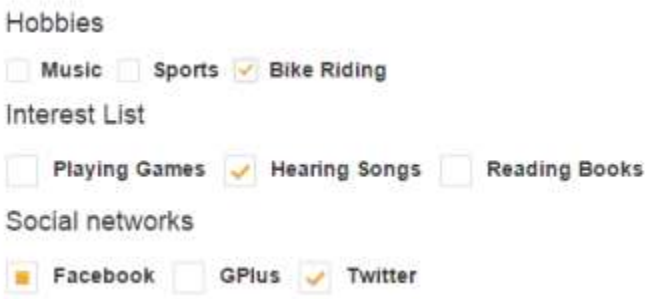
Create a Tri-State Checkbox

ASP.NET MVC Tri-State Checkbox widget renders by setting EnableTriState property to true. You can add the following code to create Tri-state Checkbox in the element of the corresponding view page.

JS

```
<h4>Social networks</h4>
<table>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox2").Value("Facebook").Size(Size.Medium).EnableT
riState(true).CheckState(CheckState.Indeterminate) <label for="Checkbox2"
class="clslab">Facebook</label>
</td>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox7").Value("GPlus").Size(Size.Medium).EnableTriS
tate(true).CheckState(CheckState.Uncheck) <label for="Checkbox7"
class="clslab">GPlus</label>
</td>
<td class="chkrad">
@Html.EJ().CheckBox("Checkbox8").Value("Twitter").Size(Size.Medium).EnableTr
iState(true).CheckState(CheckState.Check) <label for="Checkbox8"
class="clslab">Twitter</label>
</td>
</tr>
</table>
```

Run the above code to render the following output.



Tri-state Checkbox Creation

Receive Hobbies and Interest

You can receive the Hobbies and Interest values using Checkbox. You can create a button in your corresponding view page element using @Html.EJ().Button() and add the script section to the view page. The following steps illustrate how to create and set action to the button.

Add the following code for Button creation.

ASPX-CS

```
@Html.EJ().Button("buttonNormal").Text("Submit").Size(ButtonSize.Normal)
```

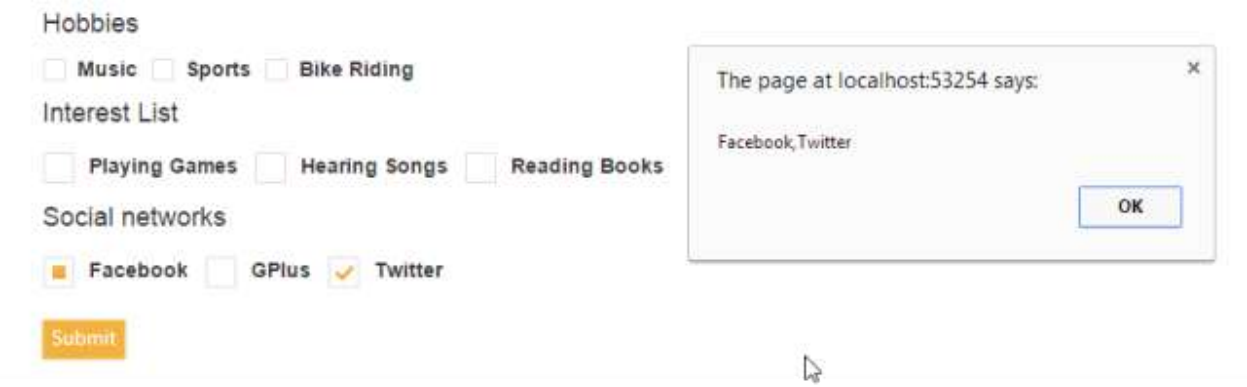
Add the script into your view page.

JS

```
<script>
$(document).ready(function () {
$("button").click(function () {
var checkedItem = [];
$(".e-checkbox").each(function () {
```

```
if ($("#" + $(this)[0].id).ejCheckBox("option", "checked"))
checkedItem.push($("#" + $(this)[0].id).ejCheckBox("option","value"));
});
alert(checkedItem);         });
});
</script>
```

Execute the above code example to render the following output.



Receive hobbies and interest

Receive Media Player

You can get the Media Player file type application like video, audio and picture using Checkbox. You can refer the following steps to render Media Player file types.

1. Add the following code in

   element of the corresponding view page.

HTML

```
<table>
<label for="check2" class="clslab">
Audio
</label>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("Checkboxs3").Value(".mp3").Size(Size.Medium)
<label for="Checkboxs3" class="clslab">
.mp3
</label>
</td>
</tr>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("Checkboxs4").Value(".AVI").Size(Size.Medium)
<label for="Checkboxs4" class="clslab">
.AVI
</label>
</td>
</tr>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</table>
<table>
<label for="Checkboxs1" class="clslab">Video</label>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("Checkboxs5").Value(".mp4").Size(Size.Medium)
<label for="Checkboxs5" class="clslab">.mp4</label>
</td>
</tr>
<tr>
<td class="chkrad">
@Html.EJ().CheckBox("Checkboxs6").Value(".wave").Size(Size.Medium)
<label for="Checkboxs6" class="clslab">.wave</label>
</td>
</tr>
</table>
```

Execute the above code to render the following output.



Receive Media player

Easy customization

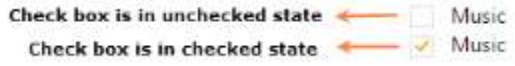
Checked state

Using Checked property, you can set the state of Checkbox. When checked property is true then tick mark is displayed and Checkbox is in checked state. When it is false, the tick mark is not displayed is in non tri-state and Checkbox is in unchecked state. When you want to use this Checked property, then checkbox should be in non Tri-state and EnableTriState property should be false.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* set the state for non Tri-state checkbox using Checked property*@
@Html.EJ().CheckBox("checkbox_unchecked").Checked(false)
<label for="checkbox_unchecked" class="clslab">Music</label>
<br />
@Html.EJ().CheckBox("checkbox_checked").Checked(true)
<label for="checkbox_checked" class="clslab">Music</label>
```

Execute the above code to render the following output.



Checkbox in binary states

Enable Tri-State

Sometimes, it is essential for you to represent the answer in partially true state. To represent the partially true types, an indeterminate state option is present. The state between checked and unchecked state is called indeterminate state. For example, a Checkbox presented to select files to send via FTP can use a tree view so that files can be selected one at a time, or by folder. When only some of the files in a folder are selected, then the checkbox for that folder could be in indeterminate state.

When you enable Tri-state, then the Checkbox includes the indeterminate state. The Checkbox has three states. EnableTriState property specifies to enable or disable the Tri-State option for Checkbox.

The following steps explains you the details about rendering the Checkbox with Tri-state options.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* Enable or disable the Tri-state using EnableTriState property *@
@Html.EJ().CheckBox("checkbox_nonTriState").EnableTriState(false)
<label for="checkbox_nonTriState" class="clslab">Music</label>
<br />
@Html.EJ().CheckBox("checkbox_triState").EnableTriState(true)
<label for="checkbox_triState" class="clslab">Music</label>
```

Execute the above code to render the following output.



Checkbox with Non-Tri state and Tri-state

Check State

You require an option to set indeterminate state for Checkbox. By using CheckState property, you can set any state that is illustrated in following table. Before using this property, enable the Tri-state for Checkbox. EnableTriState property is set true.

List of check states

Check	Check box will be in checked state
Uncheck	Check box will be in un-checked state
Indeterminate	Check box will be in indeterminate state

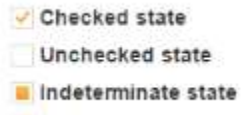
The following steps explains you the details about rendering the Checkbox with specified checked state, when the checkbox is in tri-state.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@*Add the code in CSHTML page to configure and initialize the control*@
@*set the state of Tri-state checkbox using CheckState property*@
@Html.EJ().CheckBox("checkbox_check").EnableTriState(true).CheckState(CheckS
tate.Check)
<label for="checkbox_check" class="clslab">Checked state</label>
<br />
@Html.EJ().CheckBox("checkbox_uncheck").EnableTriState(true).CheckState(Chec
kState.Uncheck)
<label for="checkbox_uncheck" class="clslab">Unchecked state</label>
<br />
@Html.EJ().CheckBox("checkbox_indeterminate").EnableTriState(true).CheckStat
e(CheckState.Indeterminate)
<label for="checkbox_indeterminate" class="clslab">Indeterminate
state</label>
```

Execute the above code to render the following output.



Checkbox in three different states

Checkbox Size

You can render Checkbox in different sizes. The following table contains some predefined size option for rendering a Checkbox in easiest way. Each size option has different height and width. Mainly it avoids the complexity in rendering Checkbox with complex CSS class.

List of checkbox size

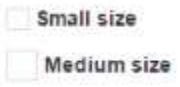
Small	Creates checkbox with Built-in small size height, width specified.
Medium	Creates checkbox with Built-in medium size height, width specified.

The following steps explains you the details about rendering the Checkbox with different size.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@* set the size of checkbox using Size property *@
@Html.EJ().CheckBox("checkbox_small").Size(Size.Small)
<label for="checkbox_small" class="clslab">Small size</label>
<br />
@Html.EJ().CheckBox("checkbox_medium").Size(Size.Medium)
<label for="checkbox_medium" class="clslab">Medium size</label>
```

Execute the above code to render the following output.



Checkbox in different sizes

Text

It specifies the text content for Checkbox. In previous programs, separate label for each Checkbox is created. You can also set the text for checkbox using Text property. Therefore, it is not essential to add label tag for each checkbox in HTML code.

The following steps explains you the details about rendering the Checkbox with text content and without writing label tag

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@*set text for checkbox using Text property *@
@Html.EJ().CheckBox("checkbox_text").Text("Music")
```

Execute the above code to render the following output.

Checkbox with text content

Rounded corner for checkbox

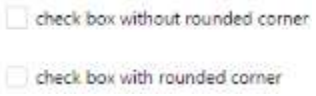
Specifies the corner of Checkbox in rounded shape. Checkbox doesn't have rounded corner by default. To set rounded corner, you can enable ShowRoundedCorner property.

The following steps explains you the details about rendering the Checkbox with rounded corner.

ASPX-CS

```
@*Add the code in CSHTML page to configure and initialize the control*@
@*set the rounded corner for checkbox *@
@Html.EJ().CheckBox("checkbox_normalCorner").ShowRoundedCorner(false).Text("
checkbox without rounded corner")
<br />
@Html.EJ().CheckBox("checkbox_roundedCorner").ShowRoundedCorner(true).Text("
checkbox with rounded corner")
```

Execute the above code to render the following output.



Checkbox with non-rounded & rounded corner

Styles Customization

CheckBox allows you to customize its appearance by using user-defined CSS and custom skin options such as colors and backgrounds. To apply custom themes, use **CssClass** property. CssClass property sets the root class for CheckBox theme.

By using this CssClass, you can override the existing styles under the theme style sheet. The theme style sheet applies theme-specific styles like colors and backgrounds. From the root class, you can customize the CheckBox control theme.

In the following example, the border color and border width of the active CheckBox is customized through the custom classes to create the success, and danger indication with CheckBox.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<span style="margin-left: 15px">
@Html.EJ().CheckBox("check1").Text("Primary").Size(Syncfusion.JavaScript.Size.Medium).Checked(true).CssClass("success")
@Html.EJ().CheckBox("check2").Text("Danger").Size(Syncfusion.JavaScript.Size.Medium).Checked(true).CssClass("danger")
</span>
```

CSS

```
<style>
.success .e-chkbox-medium .e-chk-inact .e-chk-image,
.success .e-chkbox-medium .e-chk-act .e-chk-image {
background: #17ad37;
color: white;
}
.success .e-chkbox-medium .e-chk-inact,
.success .e-chkbox-medium .e-chk-act {
border-color: transparent;
}
.danger .e-chkbox-medium .e-chk-inact .e-chk-image,
.danger .e-chkbox-medium .e-chk-act .e-chk-image {
background: #d84315;
color: white;
}
.danger .e-chkbox-medium .e-chk-inact,
.danger .e-chkbox-medium .e-chk-act {
border-color: transparent;
}
</style>
```



Miscellaneous

Checkbox Id

Checkbox id is not displayed in user interface. Here, Id mentions the id attribute of the root element of Checkbox control. When you assign a value for Id property, then this older id is replaced by new id. This id value should be unique.

Set id for Checkbox control as follows.

ASPX-CS

```
@*set id for checkbox *@
@Html.EJ().CheckBox("checkbox_id").Id("sync")
```

Checkbox Id Prefix

Id prefix value is the one that is appended to id value. It is used to mention the prefix for the wrapper's id attribute. When you assign a value for IdPrefix property, the older prefix id is replaced by new prefix id.

Copyright © 2001 - 2020 Syncfusion Inc.

Set prefix id for Checkbox control as follows.

ASPX-CS

```
@*set prefix id for checkbox *@
@Html.EJ().CheckBox("checkbox_idPrefix").IdPrefix("JS")
```

Checkbox Name

The name attribute is used to identify from data after it is submitted to the server. Checkbox also contains name attribute. This name should be a unique one. It is used to receive the Checkbox value. Without using name, you can't get the particular checkbox values at the time of submitting form.

Checkbox Value

For Checkbox, the contents of the value property do not appear in the user interface. The value property contains only a meaning when submitting a form. When a Checkbox is in checked state and when the form is submitted, the name of the Checkbox is sent along with the value of the value property (When the checkbox is not checked, no information is sent).

You can set name and value for Checkbox control as follows.

ASPX-CS

```
@*set name and value for checkbox *@
@Html.EJ().CheckBox("checkbox").Name("Conformation").Value("received")
```

How to?

Render Checkbox from Code behind

Checkbox can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Checkbox properties in the controller.

C#

```
//Namespace to get the JavaScript (Checkbox) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class CheckboxController : Controller
{
// GET: Checkbox
public ActionResult CheckboxFeatures()
{
//Initializing the Checkbox model
CheckBoxProperties BtnObj = new CheckBoxProperties();
//Initializing the datasource and other properties
BtnObj.Checked = true;
BtnObj.Text = "Checkbox";
BtnObj.ShowRoundedCorner = true;
//Passing Checkbox properties using the ViewData
ViewData["BtnModel"] = BtnObj;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Binding the Checkbox properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

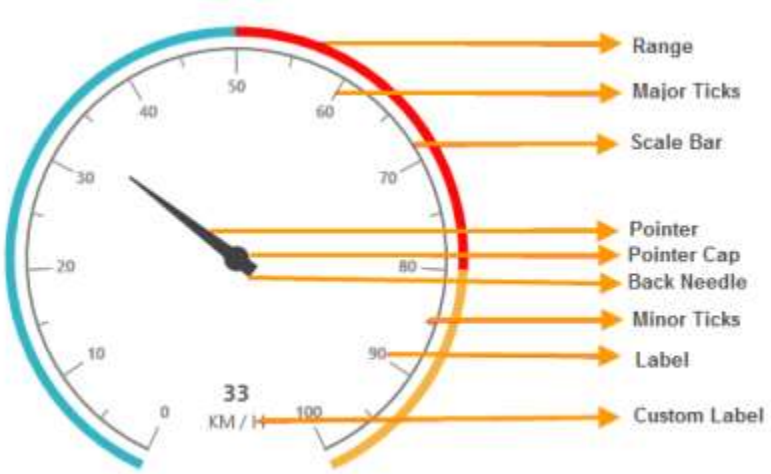
```
@{
Html.EJ().CheckBox("searchCustomer",(Syncfusion.JavaScript.Models.CheckBoxProperties)ViewData["BtnModel"]).Render();
}
```

CircularGauge

CircularGauge

The **CircularGauge** control visualizes the numerical values of scales in a circular manner. It is also a feature-rich control that provides extensive appearance customization options with support for the animation of a pointer element. The **CircularGauge** control comprises the following basic elements:

- Scales
- Pointers
- Ticks
- Labels
- Ranges
- Indicators



Circular Gauge Control Structure

The gauge display can also be customized either as a full circle or a half circle based on individual requirements. Among other helpful features, the **CircularGauge** control also includes advanced user interactivity.

Key Features

- **Sub-Gauge**: Supports for rendering of multiple sub-gauges within the main Circular Gauge.
- **User Interaction**: Allows you to directly interact with the pointers of the gauge.

Copyright © 2001 - 2020 Syncfusion Inc.

- **Indicators**: Supports for the indicator feature that shows the active or inactive state of the gauge.
- **Ranges**: Supports for highlighting the range of values in the gauge scale.
- **Pointers**: Supports for adding multiple pointers to the gauge.
- **Frame types**: Supports two types of gauge displays, full circle and half circle.
- **Animation**: Supports the animation of a pointer.
- **CustomLabel**: Supports the addition of custom label text in the required location of the gauge.

Getting Started

- The ASP.NET MVC Circular Gauge provides support to display the **Circular Gauge** within your web page and allows you to customize it. This section encompasses the details on how to configure Circular Gauge.
- You will learn how to provide data for a Circular Gauge and display that data in a suitable way. In addition, you will learn how to customize the default Circular Gauge appearance, according to your requirements.
- As a result, you will get a Circular Gauge that shows you how the Automobile speedometer works with rpm (Rotation per Minute), kph (Kilometer per hour) and speed level indication (Safe, Caution and Danger).



Analog Speedometer

Create a Circular Gauge

ASP.NET MVC Circular Gauge widget basically renders with animation and flexible APIs. You can easily create the Circular Gauge widget by using the following steps.

1. First create an MVC Project and add necessary Dll's and script with the help of the given MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page to render CircularGauge.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().CircularGauge("circulargauge"))
```

3. Add the following code example in the controller page.

C#

```
public ActionResult Default()
{
return View();
}
```

Run the above code example to get a default CircularGauge with default values.



Circular Gauge

Set Height and Width

Pointers have different height and width so you can set the Height and Width of the gauge according to your requirements.Set the basic values of the gauge such as height and width of the canvas element values that are to be rendered.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
)
```

Run the above code example and you will see the following output.



Circular Gauge with Height and Width

Set Background Color

The speedometer must have some dark BackgroundColor so that its value is clearly visible and you can vary the speed of the pointer by setting ReadOnly as false for user interaction.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
)
```

Run the above code example and you will see the following output.



Circular Gauge with Dark Background

Copyright © 2001 - 2020 Syncfusion Inc.

Provide Scale Values

- The Scales values specifies the pointers, ticks, labels, indicators and ranges of **Circular Gauge**.
- The PointerCap can be customized with the following options. Cap Radius, cap BorderColor, cap BackgroundColor, pointer cap BorderWidth are some of the properties that are customizable.
- The speed limit in the gauge has maximum value of 200 kph. So you can set Maximum value for the gauge as 200.
- Major Ticks have the MajorIntervalValue of 20 and minor ticks have the MinorIntervalValue of 5. ShowRanges and ShowIndicators are used to display the ranges and indicators in their respective positions.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {
scale.ShowRanges(true)
.ShowIndicators(true)
.PointerCap(cap=>
cap.Radius(15)
.BorderWidth(0)
.BackgroundColor("#797C79")
.BorderColor("#797C79"))        .Maximum(200)
.MajorIntervalValue(20)
.MinorIntervalValue(5).Add();
//Add the labels customization code here
//Add the pointers customization code here
//Add the ticks customization code here
//Add the ranges customization code here
//Add the indicators customization code here
//Add the Custom labels customization code here
}))
```

Run the above code example and you will see the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with Scale Values

Add Label Customization

To display the value around the scales, Labels are used. By customizing the label Color it displays as specified.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {scale
//Add the labels customization code here
.Labels(label => label.Color("#ffffff").Add())
.Add();
//Add the pointers customization code here
//Add the ticks customization code here
//Add the ranges customization code here
//Add the indicators customization code here
//Add the Custom labels customization code here
}))
```

Run the above code example and you will see the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with Label

Add Pointers

Here, you have three Pointers that denote the kilometer value, rotation per minute value and torque value.The torque value pointer needs not be similar to the other two pointers. You can set torque pointer as marker pointer. And you can set other attributes for pointer such as Value, ShowBackNeedle, Type, MarkerType, NeedleType, BackgroundColor, Border Color, Length, Width, Radius and DistanceFromScale.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {scale
//Add the labels customization code here
//Add the pointers customization code here
.Pointers(pointer=>{
pointer.Value(140)
.DistanceFromScale(60)
.ShowBackNeedle(false)
.Length(20)
.Type(PointerType.Marker)
.MarkerType(MarkerType.Triangle)
.Width(10)
.BackgroundColor("#FF940A")
.Border(border => border.Color("#FF940A")).Add();
pointer.Value(110)
.ShowBackNeedle(false)
.Length(150)
.NeedleType(NeedleType.Rectangle)
.Width(2)
.BackgroundColor("#05AFFF")
.Border(border =>border.Color("#05AFFF")).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
pointer.Value(67)
.ShowBackNeedle(false)
.Length(100)
.Width(15)
.BackgroundColor("#FC5D07")
.Border(border => border.Color("#FC5D07")).Add();
})
.Add();
//Add the ticks customization code here
//Add the ranges customization code here
//Add the indicators customization code here
//Add the Custom labels customization code here
}))
```

Run the above code example and you will see the following output.



Circular Gauge with Pointers

Add Tick Details

- You can set Major Ticks with their Width and Height equal to Minor Ticks.
- You can set Color according to your preference for better visibility in dark backgrounds.
- You can specify the tick Type either major or minor type tick and DistanceFromScale values.
- To display and customize the tick value add the following code example.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {
scale
//Add the labels customization code here
```

```
//Add the pointers customization code here
//Add the ticks customization code here
.Ticks(tic => {
tic.Type(CircularTickTypes.Major)
.DistanceFromScale(70)
.Height(20)
.Width(3)
.Color("#ffffff").Add();
tic.Type (CircularTickTypes.Minor)
.DistanceFromScale(70)
.Height(12)
.Width(1)
.Color("#ffffff").Add();
})
.Add();
//Add the ranges customization code here
//Add the indicators customization code here
//Add the Custom labels customization code here
}))
```

Run the above code example and you will see the following output.



Circular Gauge with Ticks

Add Range Values

- Ranges denote the property of the scale value in the speedometer. The color values of the Ranges denote speed variation. Set ShowRanges as **true** for showing the ranges in the **Circular Gauge**.
- For Low speed, you can mention it as safe zone; for moderate speed, you can call it as caution zone and for high speed, you can mark it as high speed.
- You can customize the range with properties such as StartValue, EndValue, StartWidth, EndWidth, BackgroundColor , Border Color, DistanceFromScale,etc.,

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {
scale
//Add the labels customization code here
//Add the pointers customization code here
//Add the ticks customization code here
//Add the ranges customization code here
.Ranges(range => {
range.DistanceFromScale(30)
.StartValue(0)
.EndValue(70)
.BackgroundColor("#5DF243")
.Border(border => border.Color("#ffffff")).Add();
range.DistanceFromScale(30)
.StartValue(70)
.EndValue(140)
.BackgroundColor("#F6FF0A")
.Border(border => border.Color("#ffffff")).Add();
range.DistanceFromScale(30)
.StartValue(140)
.EndValue(200)
.BackgroundColor("#FF1807")
.Border(border => border.Color("#ffffff")).Add();
})
.Add();
//Add the indicators customization code here
//Add the Custom labels customization code here
}))
```

Run the above code example and you will see the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Circular Gauge with Ranges

Add Indicator Details

- Indicators denote whether the pointers values are in their respective zones or not. Positioning the Indicators on the respective range value gives you the required changes.
- By using Position property, you can set the location of the indicator. You can also specify Height, Width and Type for indicators. StateRanges defines how the indicator should behave when the pointer is in certain values.
- You can customize state ranges with properties like EndValue, StartValue, BackgroundColor, BorderColor, Text, TextColor, etc.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {
//Add the labels customization code here
//Add the pointers customization code here
//Add the ticks customization code here
//Add the ranges customization code here
//Add the indicators customization code here
.Indicators(ind =>
{
ind.Height(10)
.Width(10)
. Type(IndicatorType.Circle)
.Position(loc => loc.X(210).Y(300))
.StateRanges(state =>
{
state.EndValue(70)
.StartValue(0)
.BackgroundColor("#5DF243")
.BorderColor("#5DF243").Text("").TextColor("#ffffff")
.Add();
state.EndValue(200)
.StartValue(70)
.BackgroundColor("#145608")
.BorderColor("#145608").Text("").TextColor("#ffffff")
.Add();
}).Add();
ind.Height(10)
.Width(10)
. Type(IndicatorType.Circle)
.Position(loc => loc.X(255).Y(200))
.StateRanges(state =>
{
state.EndValue(140)
.StartValue(70)
.BackgroundColor("#F6FF0A")
.BorderColor("#F6FF0A").Text("").TextColor("#ffffff")
.Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
state.EndValue(70)
.StartValue(0)
.BackgroundColor("#969B0C")
.BorderColor("#969B0C").Text("").TextColor("#ffffff")
.Add();
state.EndValue(200)
.StartValue(140)
.BackgroundColor("#969B0C")
.BorderColor("#969B0C").Text("").TextColor("#ffffff")
.Add();
}).Add();
ind.Height(10)
.Width(10)
. Type(IndicatorType.Circle)
.Position(loc => loc.X(300).Y(300))
.StateRanges(state =>
{
state.EndValue(140)
.StartValue(200)
.BackgroundColor("#890F06")
.BorderColor("#890F06").Text("").TextColor("#ffffff")
.Add();
state.EndValue(140)
.StartValue(0)
.BackgroundColor("#FF1807")
.BorderColor("#FF1807").Text("").TextColor("#ffffff")
.Add();
}).Add();
}).Add();
//Add the Custom labels customization code here }))
```

Run the above code example and you will see the following output.



Circular Gauge with Indicators

Copyright © 2001 - 2020 Syncfusion Inc.

Add Custom Label Details

CustomLabels are used to specify the texts that need to be displayed in the gauge. You can customize it through various properties such as Value, Position, Color, Font Size, FontFamily and FontStyle. To display the three range description, custom texts are used here.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.BackgroundColor("#3D3F3D")
.ReadOnly(false)
.Scales(scale => {
//Add the labels customization code here
//Add the pointers customization code here
//Add the ticks customization code here
//Add the ranges customization code here
//Add the indicators customization code here
//Add the Custom labels customization code here
.CustomLabels(label=>{
label.Value("Safe")
.Color("#5DF243")
.Position(loc=>loc.X(200).Y(280))
.Font(font => { font.FontFamily("Arial").Size("12px").FontStyle("Bold"); })
.Add();
label.Value("Caution")
.Color("#F6FF0A")
.Position(loc => loc.X(253).Y(212))
.Font(font => { font.FontFamily("Arial").Size("12px").FontStyle("Bold"); })
.Add();
label.Value("Danger")
.Color("#FF1807")
.Position(loc => loc.X(290).Y(280))
.Font(font => { font.FontFamily("Arial").Size("12px").FontStyle("Bold"); })
.Add();
}).Add();
}))
```

Run the above code example and you will see the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with Custom label

Interaction and Animation

Interaction

- **Circular Gauge** control contains **Interaction** feature. You can use this interaction feature to change the pointer values manually. You can achieve this by clicking and dragging the pointer over the Gauge and you can see the value of pointer changes dynamically while dragging.
- To Enable/Disable the user interaction you can use the Boolean property called ReadOnly. The user interaction option is enabled when you set the value as false for the property readOnly.By default it holds the true value.That is by default it does not support interaction.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
// For User interaction
.ReadOnly(false))
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with basic interaction property

Animations

- **Circular Gauge** contains an attractive concept called **Animation**. The animation option enables the pointer to rotate from the minimum value to the current value with animation effects. By using this animation you can change the pointer value dynamically.You can apply the animation on pointer either by clockwise or counterclockwise based on the scale direction.
- You can enable / disable it using the property EnableAnimation. Animation is enabled when you set enableAnimation as 'true'. By default it holds the true value. You can control the speed of the pointer during animating by using the property AnimationSpeed. It is a numerical value that holds the time in milliseconds. That is when the value is given as 1000, it is considered as 1 second.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
// For enabling animation
.EnableAnimation(true)
// For setting animation speed
.AnimationSpeed(1000)
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with basic Animation property

Gradient

You can change the interior gradient of **Circular Gauge** by using InteriorGradient property. The IsRadialGradient property is used to check whether the gradient is circular or not.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.InteriorGradient(gradient=>gradient
.ColorInfo(
//..
).Add();
)
.IsRadialGradient(true)
)
```

Distance From Corner

You can display the circular gauge from distance apart from the corner by specifying value for DistanceFromCorner property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.DistanceFromCorner(25)
)
```

Resize

Circular gauge can be responsive while resizing by specifying EnableResize property as true.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.EnableResize(true)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Localization

The circular gauge can be localized based on name of culture specified in Locale property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Locale("en-US")
)
```

Themes

CircularGauge Theme is a set of pre-defined options that are applied to the control before **CircularGauge** is instantiated. Following predefined themes are available in JavaScript **CircularGauge**.

1. flatlight
2. flatdark
3. gradientlight
4. gradientdark
5. azure
6. azuredark
7. lime
8. limedark
9. saffron
10. saffrondark
11. gradientazure
12. gradientazuredark
13. gradientlime
14. gradientlimedark
15. gradientsaffron
16. gradientsaffrondark

The theme for circular gauge can be specified using Theme property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Theme("flatlight")
)
```

Circular Gauge Values

The Minimum, Maximum, Radius and Value attributes of circular gauge are used to render the circular gauge with specified location.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Maximum(120)
.Minimum(10)
.Value(30)
.Radius(100)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Scales

Scales are the basic functional block of the **Circular Gauge**. By customizing the Scales, the appearance of the **Gauge** can be improved. The functional blocks of Circular Gauge are

- Pointers
- Labels
- CustomLabels
- Indicators
- Ticks
- Ranges
- Sub gauges.

Adding Scale Collection

Scale collection is directly added to the Gauge object. Refer the following code example to add scale collection in **Gauge** control.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(SC =>
{
SC.Radius(130).Add();
}))
```

Execute the above code to render the following output.



Circular Gauge with scale radius.

Scale Customization

Colors and Border

- The Scale border is modified with the object called Border. It has two border property namely Color and Width which are used to customize the border color of the scale and border width of the scale.

Copyright © 2001 - 2020 Syncfusion Inc.

- Setting the background color improves the look and feel of the **Circular Gauge**. You can customize the background color of the scale using BackgroundColor.
- The scale bar of circular gauge can be enabled by setting ShowScaleBar property as true.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(SC =>
{
SC.Radius(130)
.BackgroundColor("Red")
.ShowScaleBar(true)
.Border(b =>{
//For scale border width
b.Width(3)
//For scale border color
.Color("Blue");
})
.Pointers(pointer =>
{
pointer.Length(80).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized scale border

Pointer Cap

- **Pointer cap** is a circular shape element that is located at the center of the **Circular Gauge**. The pointer cap is one of the cynosure of the Circular Gauge. By customizing the pointer cap, Gauge style is improved. The pointer cap is modified with the object PointerCap.
- It contains Radius, BorderColor, BorderWidth, InteriorGradient and BackgroundColor properties. The property radius is used to set the radius for the pointer cap. InteriorGradient is used to provide the gradient effects to the pointer cap.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(SC =>
{
SC.PointerCap(cap =>
{
//For setting pointer cap radius
cap.Radius(10)
// For setting pointer cap border width
.BorderWidth(4)
// For setting pointer cap border color
.BorderColor("Blue")
// For setting pointer cap background
.BackgroundColor("Red");
}).Add();
}))
```

Execute the above code to render the following output.



Circular Gauge with customized pointer cap

Appearance

- Circular Gauge contains two types of scale direction such as clockwise and counter clockwise. You can set them by enumerable property called Direction. And you can set the minimum and maximum values for the scale with the properties Minimum and Maximum. The two properties MinorIntervalValue and MajorIntervalValue are the values used to set interval value for the ticks and labels.
- The Radius property is used to set the radius value for the circular scale and the Size property is used to set the scale bar width. You can also adjust the Opacity of the scale with the property Opacity. The value for opacity lies between 0 and 1. You can also give some shadow effects for the scale by using the property ShadowOffset. The property StartAngle is used to set starting

Copyright © 2001 - 2020 Syncfusion Inc.

position of the scale at certain angle and SweepAngle is used to shrink or expand the scale to certain angle.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
// For setting scale bar size
scale.Size(30)
// For setting scale minimum value
.Minimum(20)
// For setting scale maximum value
.Maximum(120)
// For setting scale major interval value
.MajorIntervalValue(20)
// For setting scale minor interval
.MinorIntervalValue(5)
// For setting scale background color
.BackgroundColor("Red")
// For setting scale bar opacity
.Opacity(0.5)
// For setting scale bar shadow offset
.ShadowOffset(20)
// For setting scale direction
.Direction(Directions.CounterClockwise).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized scale values

Enable/Disable properties

You can enable / disable properties in Circular Gauge using some properties in scale collection. The ShowIndicators property is used to enable/disable the indicators. ShowLabels, ShowTicks,

ShowRanges, ShowPointers ans ShowScaleBar are used to enable/ disable labels, ticks, ranges, pointers and scale bar respectively.

Multiple Scales

You can set **Multiple scales** for a single **Circular Gauge** control by using an array of scale objects. Each scale object is independent of each other. The following code example refers to two scale objects in a Gauge.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
// For setting  first scale
scale.ShowScaleBar(true)
.Size(10)
.Radius(150)
.Minimum(20)
.Maximum(120)
.MajorIntervalValue(20)
.MinorIntervalValue(5)
.Pointers(p => { p.Length(120).Value(50).Add(); })
.Direction(Directions.Clockwise)
.ShadowOffset(20).Add();
// For setting second scale
scale.Size(10)
.ShowScaleBar(false)
.Radius(80)
.MajorIntervalValue(10)
.Labels(lb =>lb.DistanceFromScale(-40).Color("Red").Add())
.Ticks(t =>t.Color("Red").Add())
.Pointers(p => { p.Length(50).Value(40).DistanceFromScale(-30).Add();  })
.Direction(Directions.CounterClockwise)
.Opacity(0.5)
.ShadowOffset(5).Add();
})
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with multiples scales

Pointers

Pointer value points out the actual value set in the Circular Gauge. You can customize the Pointers to improve the appearance of Gauge.

Adding Pointer Collection

Pointer collection is directly added to the scale object. To add pointer collection in a Gauge control refer the following code example.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
scale.Pointers(PO =>
{
// For setting pointer value
PO.Value(30).Add();
}).Add();
})
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with pointer collection

Adding Pointer Value

Pointer value is the important element in the **Circular Gauge** that indicates the Gauge value. Real purpose of the Circular Gauge is based on the pointer value. You can set the pointer Value either directly during rendering the control or it can be achieved by public method too.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
scale.ShowScaleBar(true)
.MajorIntervalValue(10)
.MinorIntervalValue(2)
.Radius(150)
.Size(2)
.ShowRanges(true)
.Ranges(ran =>
{
ran.StartValue(20)
.EndValue(80)
.BackgroundColor(ConsoleColor.Green.ToString())
.Add();
})
.Pointers(PO =>
{
// For setting pointer value
PO.Value(30).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized pointer value

Pointer Styles

Colors and Border

- The Pointers border is modified with the object called Border as in scales. It has two border property called Color and Width which are used to customize the border color of the pointer and border width of the pointer.
- You can set the background color to improve the look of the **Circular Gauge** and you can customize the background color of the scale using BackgroundColor.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowScaleBar(true)
.Radius(110)
.Pointers(PO =>
{
// For setting pointer Value
PO.Value(45)
// For setting pointer length
.Length(80)
// For setting pointer width
.Width(16)
// For setting pointer background
.BackgroundColor(ConsoleColor.Yellow.ToString())
// For setting pointer border
.Border(border =>border.Color(ConsoleColor.Green.ToString()).Width(2))
// For setting pointer opacity
.Opacity(0.6).Add();
}).Add();
})
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with pointer collection

Appearance

- Based on the value, the **pointer** point out the label value. You can set the pointer length and width using Length and Width property respectively.
- And you can also adjust the opacity of the pointer using the property Opacity which holds the value between 0 and 1. You can add the gradient effects to the pointer using Gradients object.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowScaleBar(true)
.Radius(110)
.BackgroundColor(ConsoleColor.Yellow.ToString())
.Border(border =>border.Color(ConsoleColor.Red.ToString()).Width(2))
.Pointers(PO =>
{
PO.Value(45)
// For setting pointer background
.BackgroundColor(ConsoleColor.Yellow.ToString())
// For setting pointer border
.Border(border =>border.Color(ConsoleColor.Red.ToString()).Width(2))
// For setting pointer length
.Length(80)
// For setting pointer width
.Width(16)
// For setting pointer opacity
.Opacity(0.6).Add();
}).Add();
})
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with customized pointer styles

Position the pointer

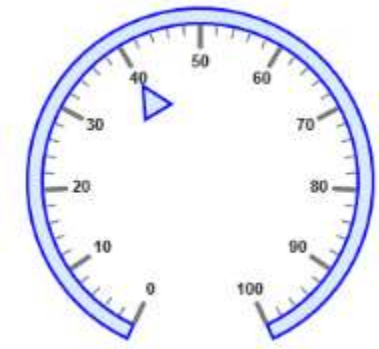
- Pointer can be positioned with the help of two properties such as **DistanceFromScale** and **Placement**. DistanceFromScale property defines the distance between the scale and pointer. Placement property is used to locate the pointer with respect to scale either inside the scale or outside the scale or along the scale.
- It is an enumerable data type. Both the property is applied only if pointer Type is marker. For NeedleType marker, it renders with default position that is unchangeable.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowScaleBar(true)
.Size(10)
.Radius(110)
.BackgroundColor("#DCEBF9")
.Border(border=>border.Color(ConsoleColor.Blue.ToString()).Width(2))
.Pointers(PO =>
{
PO.Value(40)
.Length(20)
.Width(20)
.BackgroundColor("#DCEBF9")
.Border(border=>border.Color(ConsoleColor.Blue.ToString()).Width(2))
// For setting distance between scale and pointer
.DistanceFromScale(10)
// For setting pointer placement
.Placement(PointerPlacement.Near)
// For setting pointer type
.Type(PointerType.Marker)
// For setting marker type
.MarkerType(MarkerType.Triangle)
.Add();
}).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code to render the following output.



Circular Gauge with customized pointer type as marker

Types

- Circular gauge pointer has two types such as,
    1. Needle
    2. Marker
- Needle type pointers are the default pointers that cannot be positioned and that is located at the center of the gauge. There are four different shapes of needle pointers such as
    1. Rectangle
    2. Triangle
    3. Trapezoid
    4. Arrow
- For marker pointer, the available dimensions are
    1. Rectangle
    2. Triangle
    3. Ellipse
    4. Diamond
    5. Pentagon
    6. Circle
    7. Slider
    8. Pointer
    9. Wedge
    10. Trapezoid
    11. Rounded Rectangle

Pointer Image

In ASP.NET MVC Circular Gauge, it is possible to replace the pointer with some images. You can fix an image instead of rendering the pointer.

ImageUrl

To implement the pointer image we need to give the API called `ImageUrl`. It is a string data type. Image type pointer is applicable for both marker and needle type pointers and it is possible to combine the normal marker pointer type with an image type. The three possibilities are

Copyright © 2001 - 2020 Syncfusion Inc.

1. Needle Image
2. Marker Image
3. Marker pointer with Image

Needle Image

In NeedleType, needle pointer is completely replaced by image. You can implement it with the help of the following example.

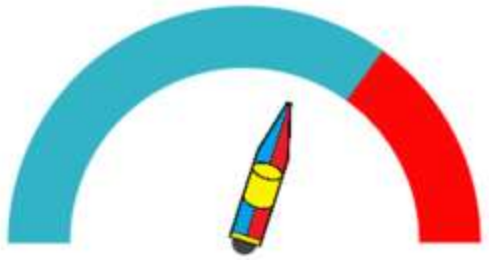
ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// To set frame type as half circle .Frame(fr =>
fr.FrameType(Frame.HalfCircle))
// To set scale options.Scales(scale =>      {
// set basic appearance
scale.ShowRanges(true).ShowLabels(false).StartAngle(180)
.SweepAngle(180).Radius(130).ShowScaleBar(false)
// To set pointer option
.Pointers(pointer => {
// To set pointer type as needle
pointer.Type(PointerType.Needle)
// To set needle type as image
.NeedleType(NeedleType.Image)
// To set image url for pointer image
.ImageUrl("nib.png")
// To set pointer value
.Value(60)
// To set pointer dimension
.Length(30).Width(100).Add(); })
// To set tick options
.Ticks(tic => { tic.Height(0).Width(0).Add(); })
// To set range options
.Ranges(range => {
range.DistanceFromScale(-30).StartValue(0)
.EndValue(70).Size(40).Add();
range.DistanceFromScale(30).StartValue(70)
.EndValue(100).BackgroundColor("#fc0606")
.Border(border => border.Color("#fc0606")).Size(40).Add();
}).Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
//        // GET: /ToolTip/
public ActionResult Semicircular()
{
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Semi-circular Gauge with needle pointer as image

Marker Image

In MarkerType, the marker pointer is completely replaced by the image. You can implement it with the help of the following example.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// To set frame type as half circle .Frame(fr =>
fr.FrameType(Frame.HalfCircle))
// To set scale options.Scales(scale =>
{
// set basic appearance
scale.ShowRanges(true).ShowLabels(false).StartAngle(180)
.SweepAngle(180).Radius(130).ShowScaleBar(false)
// To set pointer option
.Pointers(pointer => {
// To set pointer type as marker
pointer.Type(PointerType.Marker)
// To set needle type as image
.MarkerType(MarkerType.Image)
// To set image url for pointer image
.ImageUrl("ball.png")
// To set pointer value
.Value(60)
// To set pointer dimension
.Length(30).Width(100).Add(); })
// To set tick options
.Ticks(tic => { tic.Height(0).Width(0).Add(); })
// To set range options
.Ranges(range => {
range.DistanceFromScale(-30).StartValue(0)
.EndValue(70).Size(40).Add();
range.DistanceFromScale(30).StartValue(70)
.EndValue(100).BackgroundColor("#fc0606")
.Border(border => border.Color("#fc0606")).Size(40).Add();
}).Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
//        // GET: /ToolTip/
```

```
public ActionResult Semicircular()
{
return View();
}
}
```



Semi-circular Gauge with marker pointer as image

Marker pointer with Image

In this type, marker pointer is drawn first and then the image is loaded. You can implement it with the help of following example.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// To set frame type as half circle
.Frame(fr => fr.FrameType(Frame.HalfCircle))
// To set scale options
.Scales(scale =>      {
// set basic appearance
scale.ShowRanges(true).ShowLabels(false).StartAngle(180)
.SweepAngle(180).Radius(130).ShowScaleBar(false)
// To set pointer option
.Pointers(pointer => {
// To set pointer type as marker
pointer.Type(PointerType.Marker)
// To set needle type as rectangle
.MarkerType(MarkerType.Rectangle)
// To set image url for pointer image
.ImageUrl("ball.png")
// To set pointer value
.Value(50)
// To set pointer dimension
.Length(30).Width(100)
.Border(border => border.Color("Black").Width(3))
.Add(); })
// To set tick options
.Ticks(tic => { tic.Height(0).Width(0).Add(); })
// To set range options
.Ranges(range => {
range.DistanceFromScale(-30).StartValue(0)
.EndValue(70).Size(40).Add();
range.DistanceFromScale(30).StartValue(70)
.EndValue(100).BackgroundColor("#fc0606")
.Border(border => border.Color("#fc0606")).Size(40).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
//         // GET: /ToolTip/
public ActionResult Semicircular()
{
return View();
}
}
```



Semi-circular Gauge with marker pointer with image

Multiple Pointers

Circular Gauge can have multiple pointers on it. You can use any combination and any number of pointers in a Gauge. That is, a Gauge can contain any number of marker pointer and any number of needle pointers. Refer the following code example containing two pointers.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowScaleBar(true)
.Size(10)
.Radius(110)
.BackgroundColor("#DCEBF9")
.Border(border =>border.Color(ConsoleColor.Green.ToString()).Width(2))
.Pointers(PO =>
{
// For setting pointer1
PO.Value(40)
.Length(80)
.Width(16)
.Opacity(0.6)
.BackgroundColor("#DCEBF9")
.Border(border
=>border.Color(ConsoleColor.Green.ToString()).Width(2)).Add();
// For setting pointer2
PO.Placement(PointerPlacement.Near)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Type(PointerType.Marker)
.MarkerType(MarkerType.Triangle)
.Length(20)
.Width(20)
.Value(60)
.BackgroundColor("#DCEBF9")
.Border(border
=>border.Color(ConsoleColor.Green.ToString()).Width(2)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with multiple pointers

Pointer Value

- Gauge PointerValueText is used to display the current value of the pointer in the **Circular Gauge** control.
- You can position the Circular Gauge pointer value with the gauge as center by using the API called Distance. You can Disable/ Enable these pointers value by using the API ShowValue.

ASPX-CS

```
@(Html.EJ().CircularGauge("CoreCircularGauge")
// Setting basic properties
.Radius(100).Value(55).BackgroundColor("transparent")
// Setting scale properties
.Scales(SC =>{
// enable rangesSC
.ShowRanges(true)
// Setting ticks option
.Ticks(tic => { tic.Height(0).Width(0).Add(); })
// Setting range properties
.Ranges(range =>
{
range.Size(40).StartValue(0).EndValue(50).BackgroundColor("#1B4279").Border(
border =>
border.Color("#1B4279")).Add();
range.Size(40).StartValue(50).EndValue(100).BackgroundColor("#91B8F3").Borde
r(border =>
```

```
border.Color("#91B8F3")).Add();
})
// Setting pointer option
.Pointers(pointer =>{pointer.PointerValueText(value=>
// enable showValue property
value.ShowValue(true)
// setting distance property
.Distance(10)
.Color("#8c8c8c"))
.Add();})
.Add();}))
```

C#

```
public partial class CircularGaugeController : Controller
{
//// GET: /ToolTip/
public ActionResult Semicircular()
{
return View();
}
}
```

Run the above code to render the output as follows.



Circular Gauge with pointer value text.

Appearance

Appearance of the Circular Gauge PointerValueText is adjusted by using four properties. Such as **color**, **angle**, **autoAngle** and **opacity**.

- Color property is used to set the color of the pointer value text.
- Angle property is used to set the angle in which the text is displayed.
- AutoAngle is used to display the text in certain angle based on pointer position angle.
- Opacity is used to customize the brightness of the text.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge")
// Setting basic properties
.Radius(100).Value(55).BackgroundColor("transparent")
// Setting scale properties.
Scales(SC =>{
// enable ranges
SC.ShowRanges(true)
// Setting ticks option
.Ticks(tic => { tic.Height(0).Width(0).Add(); })
// Setting range properties
.Ranges(range =>
{
range.Size(40).StartValue(0).EndValue(50).BackgroundColor("#1B4279").Border(
border =>
border.Color("#1B4279")).Add();
range.Size(40).StartValue(50).EndValue(100).BackgroundColor("#91B8F3").Borde
r(border =>
border.Color("#91B8F3")).Add();
})
// Setting pointer option
.Pointers(pointer
=>{pointer.PointerValueText(value=>value.ShowValue(true).Distance(10)
// Setting opacity property
.Opacity(1)
// Setting color property
.Color("Red")
// Setting auto angle property
.autoAngle(false)
// Setting angle property.Angle(0))
.Add();
}).Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
//        // GET: /ToolTip/
public ActionResult Semicircular()
{
return View();
}
}
```

Run the above code to render the output as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with customized pointer value text.

Font Options

Similar to other collection, Font option is also available in this pointer value text such as Size, FontFamily and FontStyle.

ASPX-CS

```
@(Html.EJ().CircularGauge("CoreCircularGauge")
// Setting basic properties
.Radius(100).Value(55).BackgroundColor("transparent")
// Setting scale properties
.Scales(SC =>{
// enable rangesSC
.ShowRanges(true)
// Setting ticks option
.Ticks(tic => {
tic.Height(0).Width(0)
.Add(); })
// Setting range properties
.Ranges(range =>{
range.Size(40).StartValue(0).EndValue(50).BackgroundColor("#1B4279").Border(
border =>
border.Color("#1B4279")).Add();
range.Size(40).StartValue(50).EndValue(100).BackgroundColor("#91B8F3").Borde
r(border =>
border.Color("#91B8F3")).Add();})
// Setting pointer option
.Pointers(pointer
=>{pointer.PointerValueText(value=>value.ShowValue(true).Distance(10).Opacit
y(1).Color("Red").autoAngle(false).Angle(0)
// Setting font
option.Font(fo=>fo.Size("15px").FontFamily("Arial").FontStyle("Normal"))).Ad
d();}).Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
//// GET: /ToolTip/
public ActionResult Semicircular()
{
return View();
}
}
```

Run the above code to render the output as follows.



Circular Gauge with customized font option in pointer value text.

Labels

Labels are units that are used to display the values in the scales. You can customize Labels with the properties like Angle, Color, Font, Opacity, etc.

Adding Label Collection

Label collection is directly added to the scale object. Refer the following code example to add label collection in a **Gauge**.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.Labels(lb =>
{
// For setting label angle
lb.Angle(30).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with label collection

Label Customization

Appearance

- The attribute Angle is used to display the labels in the specified angles and Color attribute is used to display the labels in specified color. You can adjust the opacity of the label with the property Opacity and the values of it lies between 0 and 1.
- You can adjust the labels based on the tick's direction by setting AutoAngle as true. IncludeFirstValue is an special property especially used in some special scenarios such as in clock, where the value 0 needs to be replaced with that of 12. By enabling this property the first value of the label is not rendered.
- Font option is also available on the labels. The basic three properties of fonts such as Size, Family and Style can be achieved by size, fontStyle and fontFamily. Labels are two types such as major and minor.Major types labels are for major interval values and minor types labels are for minor interval values.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowScaleBar(true)
.BackgroundColor("#FAF4B5")
.Border(bo =>bo.Color("Yellow").Width(2))
.Radius(110)
.Size(10)
.Pointers(pointer =>
{
pointer.Border(bo =>bo.Width(2).Color("Yellow"))
.BackgroundColor("#FAF4B5")
.Value(40).Length(80)
.Width(16).Opacity(0.6).Add();
})
.Labels(lb =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
// For setting label angle
lb.Angle(30)
// For setting label opacity
.Opacity(0.8)
// For disable the include first value property
.IncludeFirstValue(false)
// For setting label color
.Color("Yellow")
// For setting label font
.Font(f =>f.Size("15px").FontFamily("Arial").FontStyle("bold")).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized label

Unit text and Position

- UnitText is used to add some text along with the labels. For example, in speedometer, you need to mention the units in kph. You can also add the unit text in front of the labels. You can achieve this by using an enumerable property UnitTextPosition. With this you can position the unit text in front or back.
- Labels can be positioned with the help of two properties such as DistanceFromScale and Placement. **DistanceFromScale** property defines the distance between the scale and labels. **Placement** property is used to locate the labels with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150).Size(2)
.Pointers(pointer =>
{
pointer.Value(40).Length(100)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ShowBackNeedle(true).Add();
})
.Labels(lb =>{
// For setting unit text
lb.UnitText("kmph")
// For setting unit text position
.UnitTextPosition(UnitTextPlacement.Back).Add();
})
.Ranges(ran =>
{
ran.DistanceFromScale(-
30).StartValue(0).EndValue(50).BackgroundColor("Green").Placement(RangePlace
ment.Far).Add();
ran.DistanceFromScale(-
30).StartValue(50).EndValue(80).BackgroundColor("Yellow").Placement(RangePla
cement.Far).Add();
ran.DistanceFromScale(-
30).StartValue(80).EndValue(100).BackgroundColor("Red").Placement(RangePlace
ment.Far).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with unit text

Multiple Labels

You can achieve multiple labels such as minor and major Type in a **Gauge** sample scale. Refer the following code example for multiple labels variation.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.ShowScaleBar(true)
```

```
.Radius(150)
.Size(5)
.MinorIntervalValue(5)
.BackgroundColor("Yellow")
.Border(border =>border.Width(1.5).Color("Red"))
.PointerCap(cap =>
cap.BackgroundColor("Yellow").BorderColor("Red").BorderWidth(0.5).Radius(10)
)
.Labels(lb =>
{
// For setting label1
lb.Type(GaugeTypes.Minor).Color("Yellow").Add();
// For setting label2
lb.Type(GaugeTypes.Major).Color("Red").Add();
})
.Pointers(pointer =>
{
pointer.Length(110)
.BackgroundColor("Yellow")
.Border(bo =>bo.Width(1.5).Color("Red")).Add();
})
.Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with multiple labels

Ticks

Ticks are used to mark some values on the scale. Based on the tick's Value you can set the labels on the required position.

Adding Tick Collection

Tick collection is directly added to the scale object. Refer the following code example to add tick collection in a **Gauge** control.

ASPX-CS

```
//For circular gauge rendering
```

```
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.Ticks(tick =>
{
// For setting tick
tick.Type(CircularTickTypes.Major).Color("Red").Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with tick collection

Tick Customization

- Height and width of the ticks can be applied by using the properties Height and Width. You can customize ticks with the properties such as angle, color, etc. Angle attribute is used to display the labels in the specified angles and Color attribute is used to display the labels in specified color. Ticks are two types such as major and minor.
- Major Type ticks are for major interval values and minor type ticks are for minor interval values.You can position ticks with the help of two properties such as DistanceFromScale and Placement. **DistanceFromScale** property defines the distance between the scale and ticks. **Placement** property is used to locate the ticks with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.Ticks(tick =>
{
```

```
// For setting tick1
tick.Type(CircularTickTypes.Major).Color("Red").Add();
// For setting tick2
tick.Type(CircularTickTypes.Minor)
// For setting tick color
.Color("Yellow")
// For setting tick height
.Height(8)
// For setting tick placement
.Placement(TickPlacement.Near)
// For setting tick distance from scale
.DistanceFromScale(5).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with tick customization

Indicators

Indicators simply indicates the current status of the pointer. Indicators are in several formats such as in shape format, textual format and image format.

Adding Indicator Collection

Indicators collection is directly added to the scale object. Refer the following code to add indicator collection in a **Gauge** control.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowIndicators(true)
.Indicators(ind =>
{
```

```
ind.Height(10)
.Width(10)
.Type(IndicatorType.Circle)
.Position(position =>position.X(185).Y(300)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with indicator collection.

Basic Customization

- You can enable indicators by setting ShowIndicators to 'true'. The Height and Width property for the indicators are used to specify the area allocated to the indicator for the width and height respectively. You can use the Position collection to position the indicators along X and Y axis.
- Indicators are of several types such as, circle, rectangle, rounded rectangle, text and image. By using the Type property you can avail those shapes. For image type ImageUrl property is used.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowIndicators(true)
.MinorIntervalValue(5)
.BackgroundColor("#5DF243")
.Border(bo =>bo.Width(1.5).Color("Black"))
.ShowScaleBar(true).Radius(120).Size(5)
.Pointers(pointer =>
{
pointer.BackgroundColor("#5DF243")
.Border(bo =>bo.Width(1.5).Color("Black"))
.Length(110).Add();
})
```

```
.Indicators(ind =>
{
// For setting indicator height
ind.Height(10)
// For setting indicator width
.Width(10)
// For setting indicator type
.Type(IndicatorType.Circle)
// For setting indicator position
.Position(position =>position.X(185).Y(300)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized indicator with basic properties

State Ranges

- State ranges are used to specify the indicator behavior in the specified region. Use StartValue and EndValue to set the range bound for the pointer. Whenever the pointer cross the specified region, the indicator attributes are applied for ranges.
- The BackgroundColor and BorderColor sets the appearance behavior for the indicators. For text type indicators you can give value for text. And Text can be changed whenever the pointer crosses its state range area. There are many basic Font options available for the text in the state range such as Size, FontStyle and FontFamily.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowIndicators(true)
.MinorIntervalValue(5)
.BackgroundColor("#5DF243")
.Border(bo =>bo.Width(1.5).Color("Black"))
.ShowScaleBar(true).Radius(150).Size(5)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Pointers(pointer =>
{
pointer.BackgroundColor("#5DF243")
.Border(bo =>bo.Width(1.5).Color("Black"))
.Length(110).Add();
})
.Indicators(ind =>
{
// For setting indicator height
ind.Height(10)
// For setting indicator width
.Width(10)
// For setting indicator type
.Type(IndicatorType.Circle)
// For setting indicator position
.Position(position =>position.X(185).Y(300))
// For setting indicator state range collection
.StateRanges(st =>
{
// For setting state range End Value
st.EndValue(100)
// For setting state range start value
.StartValue(0)
// For setting indicator background color
.BackgroundColor("#5DF243")
// For setting indicator border color
.BorderColor("Black")
// For setting indicator text
.Text("")
// For setting indicator text color
.TextColor("#870505").Add();
}).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized indicator

Copyright © 2001 - 2020 Syncfusion Inc.

Multiple Indicators

You can use multiple indicators for a single Gauge. Each indicator have a list of StateRanges. Refer the following code example for multiple Indicators.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowIndicators(true)
.ShowRanges(true)
.MinorIntervalValue(5)
.ShowScaleBar(true).Radius(150).Size(5)
.Pointers(pointer =>
{
pointer.Length(110).Value(70).Add();
})
.Ranges(ran =>
{
ran.DistanceFromScale(-
30).StartValue(0).EndValue(50).BackgroundColor("Green").Placement(RangePlace
ment.Far).Add();
ran.DistanceFromScale(-
30).StartValue(50).EndValue(100).BackgroundColor("Red").Placement(RangePlace
ment.Far).Add();
})
.Indicators(ind =>
{
//Indicator1
ind.Height(10).Width(10)
.Type(IndicatorType.Circle)
.Position(position =>position.X(165).Y(300))
.StateRanges(st =>
{
st.EndValue(50)
.StartValue(0)
.BackgroundColor("#24F92F")
.BorderColor("Black").Add();
st.EndValue(50)
.StartValue(100)
.BackgroundColor("#322C04")
.BorderColor("Black").Add();
}).Add();
//Indicator2
ind.Height(10).Width(10)
.Type(IndicatorType.Circle)
.Position(position =>position.X(215).Y(300))
.StateRanges(st =>
{
st.EndValue(50)
.StartValue(0)
.BackgroundColor("#600000")
.BorderColor("Black").Add();
st.EndValue(100)
.StartValue(50)
.BackgroundColor("#FF4F2A")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.BorderColor("Black").Add();
}).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with multiple indicators

Ranges and Frames

Ranges are used to specify or group the scale values. By using ranges, you can describe the values in the pointers.

Add Range Collection

Range collection is directly added to the scale object. Refer the following code example to add range collection in a **Gauge** control.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.Ranges(ran =>
{
ran.StartValue(20)
.EndValue(80)
.BackgroundColor("Green")
.Placement(RangePlacement.Far).Add();
}).Add();
})
)
```

Range Customization

Appearance

- The API Size is used to specify the width of the ranges. The major attributes for ranges are StartValue and EndValue. **StartValue** defines the start position of the ranges and **EndValue** defines the end position of the ranges.
- StartWidth and EndWidth are used to specify the range width at the starting and ending position of the ranges. You can add the gradient effects and opacity to the ranges by using Gradient object and Opacity property.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150).Size(5)
.Ranges(ran =>
{
//For setting range start value
ran.StartValue(20)
//For setting range end value
.EndValue(80)
//For setting range background color
.BackgroundColor("Green")
//For setting range placement
.Placement(RangePlacement.Far).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized ranges with startValues and endValues

Colors and Border

- By customizing the ranges, the appearance of the Gauge can be improved. The range border is modified with the object called Border. It has two border property such as Color and Width. These are used to customize the border color of the ranges and border width of the ranges.
- You can set the background color to improve the look and feel of the Circular Gauge. For customizing the background color of the ranges, BackgroundColor is used.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150).Size(5)
.Ranges(ran =>
{
//For setting range start value
ran.StartValue(20)
//For setting range end value
.EndValue(80)
//For setting range background color
.BackgroundColor("Yellow")
//For setting range placement
.Placement(RangePlacement.Far)
//For setting range border
.Border(bo =>bo.Width(2).Color("Green")).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized range colors and borders

Copyright © 2001 - 2020 Syncfusion Inc.

Position the ranges

- You can position ranges using two properties such as DistanceFromScale and Placement.
- **DistanceFromScale** property defines the distance between the scale and range.
- **Placement property** is used to locate the pointer with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150).Size(2)
.Ranges(ran =>
{
//For setting range start value
ran.StartValue(0)
//For setting range end value
.EndValue(100)
//For setting range background color
.BackgroundColor("Green")
//For setting range placement
.Placement(RangePlacement.Far)
//For setting distance between scale and ranges
.DistanceFromScale(-30)
//For setting range border
.Border(bo =>bo.Width(2).Color("Black")).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized ranges

Copyright © 2001 - 2020 Syncfusion Inc.

Multiple Ranges

You can set multiple ranges by adding an array of ranges objects. Refer the following code example for multiple ranges functionality.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(sc1 =>
{
sc1.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150).Size(2)
.Pointers(p =>
{
p.Value(40)
.ShowBackNeedle(true)
.Length(100).Add();
})
.Ranges(ran =>
{
//For setting range1
ran.StartValue(0)
.EndValue(50)
.BackgroundColor("Green")
.Placement(RangePlacement.Far)
.DistanceFromScale(-30).Add();
//For setting range2
ran.StartValue(50)
.EndValue(80)
.BackgroundColor("Yellow")
.Placement(RangePlacement.Far)
.DistanceFromScale(-30).Add();
//For setting range3
ran.StartValue(80)
.EndValue(100)
.BackgroundColor("Red")
.Placement(RangePlacement.Far)
.DistanceFromScale(-30).Add();
}).Add();
})
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with multiple ranges

Frames

- Frame is the element that decides the appearance of the Circular Gauge. You can customize it using the object called Frame. It has the properties such as frameType, backGroundUrl, halfCircleFrameStartAngle and halfCircleFrameEndAngle.
- FrameType is used to specify whether frame is a half circle frame or full circle frame. HalfCircleFrameStartAngle and HalfCircleFrameEndAngle are used to specify the angle for Gauge with frame type as half circle. BackgroundUrl is used to set the background image for the frame.

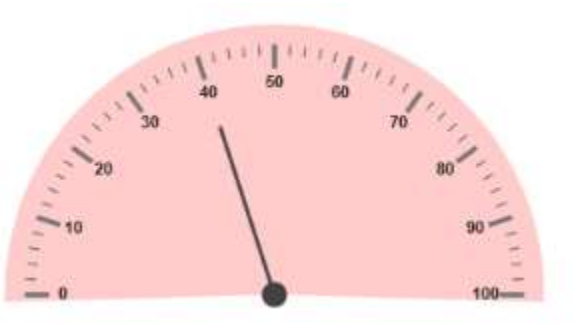
ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Frame(f =>
{
//For setting type
f.FrameType(Frame.HalfCircle)
//For setting half circle frame start angle
.HalfCircleFrameStartAngle(205)
//For setting half circle frame end angle
.HalfCircleFrameEndAngle(335).ToString();
})
.BackgroundColor("#FFCCCC")
.Scales(scale =>
{
scale.StartAngle(180)
.SweepAngle(180)
.PointerCap(cap =>cap.Radius(8))
.Pointers(pointer =>
{
pointer.NeedleType(NeedleType.Rectangle)
.Width(1).Length(120)
.Value(40).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

Execute the above code to render the following output.



Circular Gauge with multiple ranges

Legend

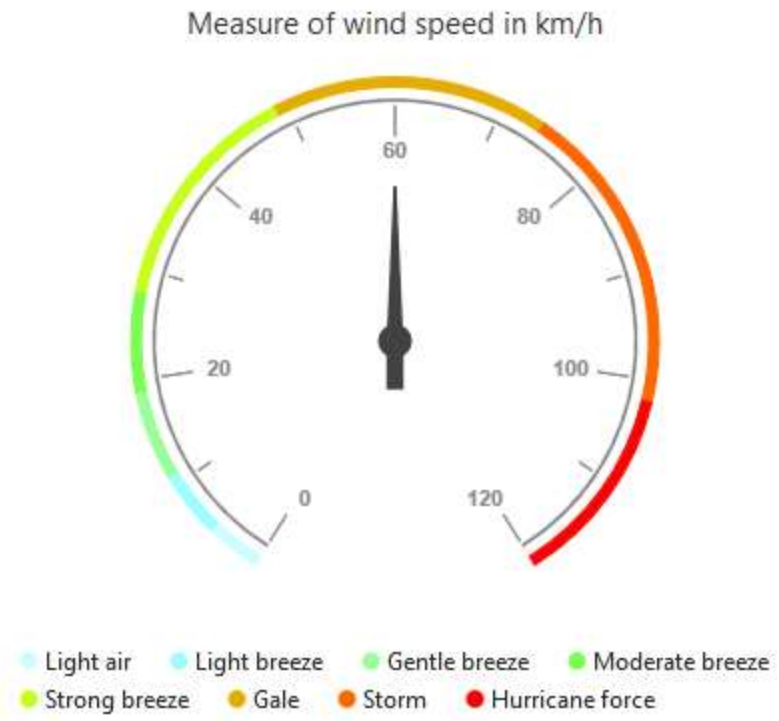
The Legend contains the list of the ranges that appear in the circular gauge

Legend Visibility

By default, the legend will not be displayed in the circular gauge. You can enable or disable it by using the Visible property of the legend.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
//Visible CircularGauge legend
.Legend(lg=>lg.Visible(true))
//...
)
```



Click here to view the online demo sample for legend in the circular Gauge.

Legend Text

The text displayed in the legend can be customized by using the LegendText property present in the ranges of the circular gauge. When the legendText is not specified in the ranges, then the legend item for that particular range will not displayed. By default the legendText value is **null** .

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
.Scales(scale =>
{
scale.Ranges(range =>
{
range.LegendText("Light air").Add();
})
})
)
```

Position and Align the Legend

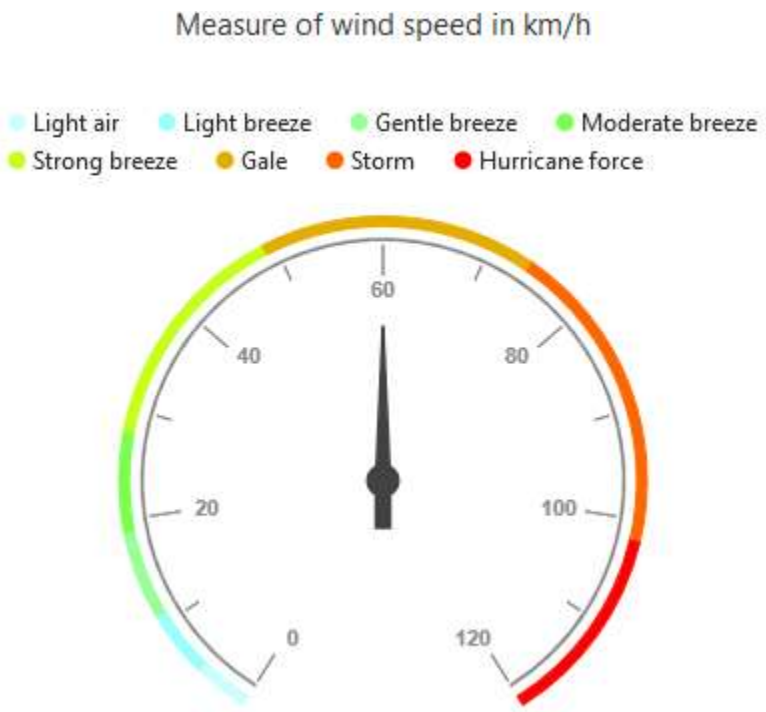
By using the Position property, you can position the legend at *left*, *right*, *top* or *bottom* of the CircularGauge. The legend is positioned at the **bottom** of the circular gauge, by default.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg.
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Place the legend at top of the CircularGauge
Position(CircularLegendPosition.Top)
)
//...
)
```



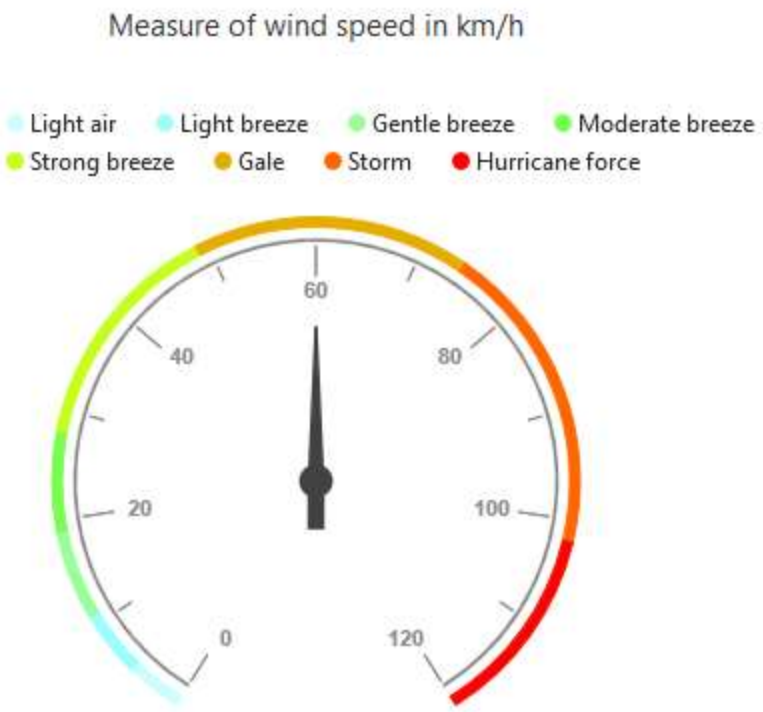
Legend Alignment

You can align the legend to the *center*, *far* or *near* based on its position by using the Alignment property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg.
//...
//The below two settings will place the legend at the top-right corner of
the gauge.
Position(CircularLegendPosition.Top)
.Alignment(CircularLegendAlignment.Far)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customization

Legend Fill and Opacity

You can change the opacity and fill color of legend text using Opacity and Fill property of legend.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
//fill color of legend
.Fill('red')
//opacity of legend
.Opacity(0.8)
)
//...
)
```

Legend shape

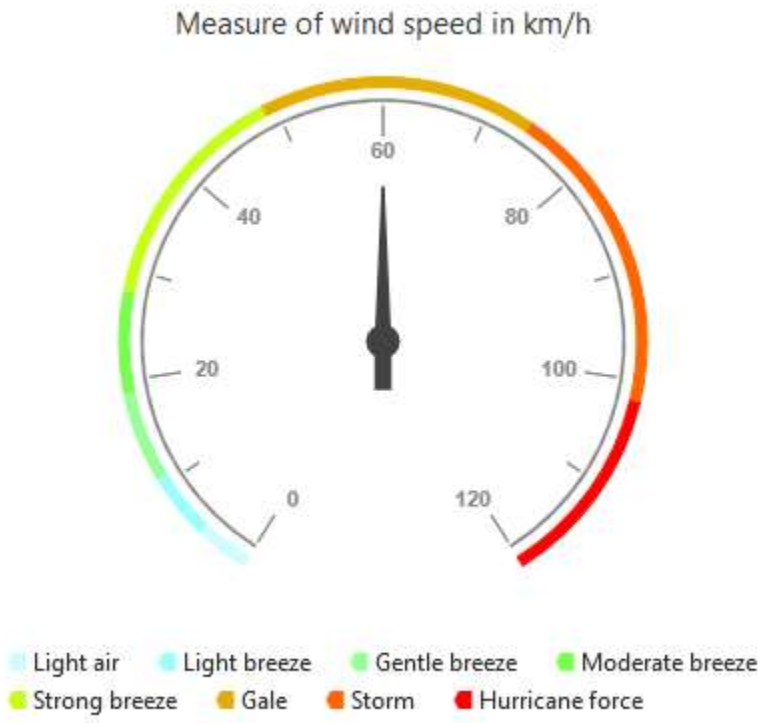
To change the legend item shape, you have to specify the desired shape in the Shape property of the legend. By default, the shape of the legend is **circle**.It also supports rectangle,diamond,triangle,slider,line,pentagon,trapezoid and wedge shapes.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Change legend shape
.Shape(CircularLegendShape.Slider)
)
//...
)
```

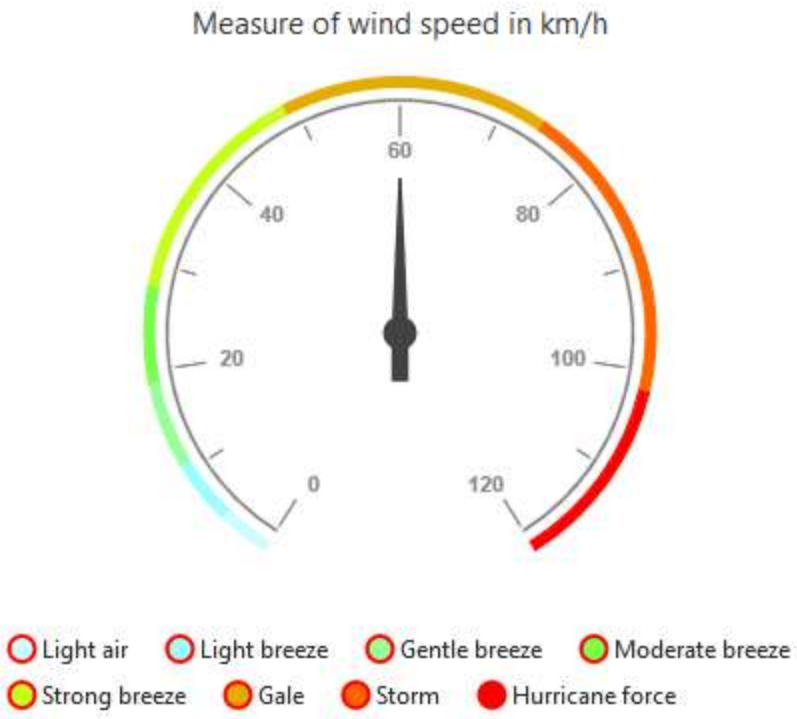


Legend Item Size and Border

You can change the size of the legend items by using the Width and Height properties in the ItemStyle. To change the legend item border, use Border property of the legend itemStyle. The color and width of legend item border can be customized using border Color and Width property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
//Change legend items border, height and width
.ItemStyle(item=>item.Width(13).Height(13).Border(br=>br.Width(2).Color("#FF
0000")))
)
//...
)
```

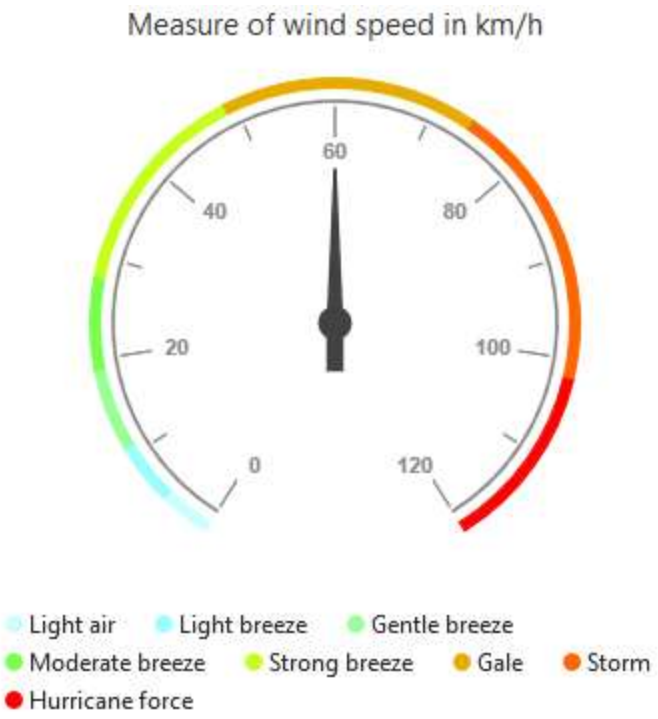


Legend size

You can change the default legend size by using the Size property of the legend. The height and width of legend size can customized using Height and Width property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
//Change legend size
.Size(size=>size.Height("100").Width("350"))
)
//...
)
```



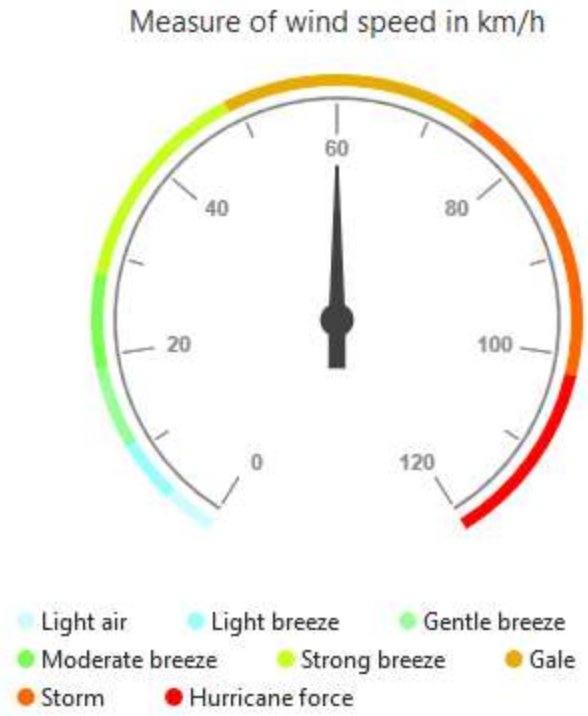
Legend Item Padding

You can control the spacing between the legend items by using the ItemPadding option of the legend. The default value is 20.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
//Add space between each legend item
.ItemPadding(30)
)
//...
)
>
```

Copyright © 2001 - 2020 Syncfusion Inc.

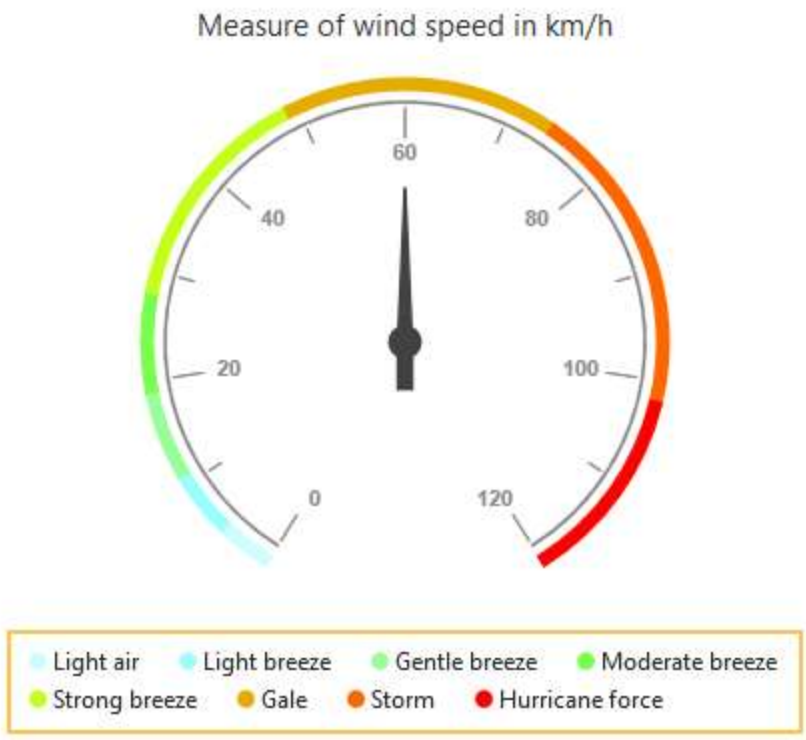


Legend border

You can customize the legend border by using the Border option in the legend. The legend border can be customized using border Color and Width property.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
//Set border color and width to legend
.Border(br=>br.Color("#FFC342").Width(2))
)
//...
)
```



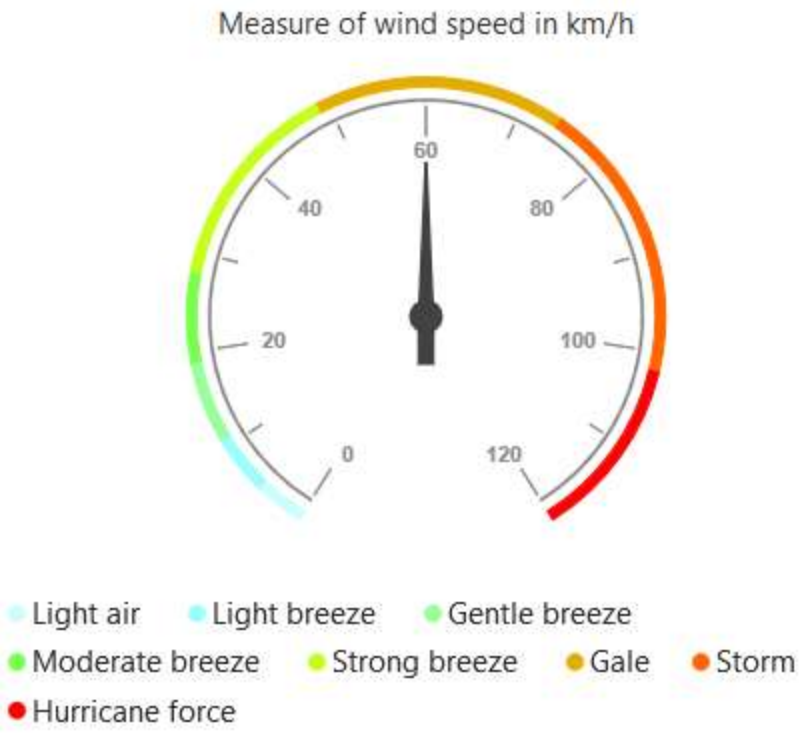
Font of the legend text

The Font of the legend item text can be customized by using properties such as FontFamily, FontStyle, FontWeight and Size of legend font.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
//...
//Customize the legend item text
.Font(font=>font.FontFamily("Segoe
UI").FontStyle(CircularGaugeFontStyle.Normal)
.FontWeight(CircularGaugeFontWeight.Bold).Size("15px"))
)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Events

Legend Item Render

LegendItemRender event triggers before rendering the legend items. This event is triggered for each legend item in Circular gauge. You can use this event to customize legend item shape or add custom text to legend item dynamically

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
.Visible(true)
//...
)
//Subscribe the legend item render event
.LegendItemRender("onLegendRender")
)
function onLegendRender(sender) {
//Get legend item details before rendering
var legendItem = sender.data;
}
```

Legend Item Click

You can get the legend item details such as *RangeIndex, bounds and shape* by subscribing the LegendItemClick event of the circular gauge. When the legend item is clicked, it triggers the event and returns the legend information

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().CircularGauge("circulargauge1")
// ...
.Legend(lg=>lg
.Visible(true)
//...
)
//Subscribe the legend item click event
.LegendItemClick("onLegendClicked")
)
function onLegendClicked(sender) {
//Get legend item details on legend item click.
var legendItem = sender.data;
}
```

Custom labels

Custom labels are the texts that you can use them in any location of the **Gauge**.

Adding Custom Label Collection

Custom labels collection is directly added to the scale object. Refer the following code to add CustomLabels collection in a Gauge control.

ASPX-CS

```
//For circular gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
scale.CustomLabels(label =>
{
//custom label
label.TextAngle(10)
.Color("Red")
.Value("CustomLabel1")
.Font(f =>f.Size("20px").FontFamily("Arial").FontStyle("bold"))
.Position(p =>p.X(180).Y(100)).Add();
}).Add();
})
)
```

Basic Customization

- You can customize custom labels using the properties such as TextAngle, Color and Font. **TextAngle** attribute is used to display the custom labels in the specified angles and **Color** attribute is used to display the custom labels in specified color.
- You can use Value attribute to set the text value in the custom labels. To display the custom labels, set ShowCustomLabels as ‘true’. To set the location of the custom label in Circular Gauge, Position property is used. By using X and Y axis you can adjust the position of the custom labels.
- Font option is also available on custom labels. The basic three properties of fonts such as size, family and style can be achieved by Size, FontStyle and FontFamily attributes.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
scale.Size(2)
.ShadowOffset(10)
.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150)
.ShowLabels(true)
.CustomLabels(cl =>
{
//For setting custom label text angle
cl.TextAngle(10)
//For setting custom label color
.Color("Red")
//For setting custom label color
.Value("CustomLabel1")
//For setting custom label font option
.Font(f =>f.Size("20px").FontFamily("Arial").FontStyle("bold"))
//For setting custom label position
.Position(p =>p.X(180).Y(100)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with customized custom label

Multiple Custom Labels

You can set multiple custom labels in a single **Circular Gauge** by adding an array of custom label objects. Refer the following code example for multiple custom label functionality.

ASPX-CS

```
// For Circular Gauge rendering
@(Html.EJ().CircularGauge("circulargauge")
.Scales(scale =>
{
```

```
scale.Size(2)
.ShadowOffset(10)
.ShowRanges(true)
.ShowScaleBar(true)
.Radius(150)
.ShowLabels(true)
.CustomLabels(cl =>
{
//custom label1
cl.TextAngle(10)
.Color("Red")
.Value("CustomLabel1")
.Font(f =>f.Size("20px").FontFamily("Arial").FontStyle("bold"))
.Position(p =>p.X(180).Y(100)).Add();
//custom label2
cl.TextAngle(10)
.Color("Green")
.Value("CustomLabel2")
.Font(f =>f.Size("20px").FontFamily("Arial").FontStyle("bold"))
.Position(p =>p.X(180).Y(250)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with multiple custom labels

Outer Custom Label

- **Outer Custom Label** is used to show custom labels outside the gauge control. The Outer Custom Label can be positioned with API called `OuterCustomLabelPosition`. The value for this API is enumerable type and its possible values are,
    1. Right
    2. Left
    3. Top
    4. Bottom

Copyright © 2001 - 2020 Syncfusion Inc.

- When a custom label is to be displayed as an Outer Custom Label, set the API PositionType as Outer. Refer to the following code example to get the Outer Custom Label.

ASPX-CS

```
@(Html.EJ().CircularGauge("circularGaugeTooltip")
//Defines the outer label position.
.OuterCustomLabelPosition(OuterCustomLabelPosition.Right)
//Defines the tooltip object.
.Tooltip(tooltip=>tooltip
// Enables the label tooltip.
.ShowLabelTooltip(true)
// Enables the custom label tooltip.
.ShowCustomLabelTooltip(true))
// Customizes the scale options.
.Scales(SC =>{SC.Radius(130).ShowLabels(true)
// Customizes the custom label options.
.CustomLabels(cl => {cl.Value("AverageSpeed").Font(f
=>f.Size("20px").FontFamily("Arial").FontStyle("bold")).Position(p
=>p.X(360).Y(30)).Add();})
// Customizes the pointers options.
.Pointers(PO =>{PO.Value(60).Length(90).Add();}).Add();}))
```

C#

```
public ActionResult Print()
{
var DataSource = new ScheduleDataDataContext().DefaultSchedules.ToList();
ViewBag.dataSource = DataSource;return View();
}
```

Execute the above code to render the following output.



Circular gauge with outer custom label.

Tooltip

- **Tooltip** feature has been added to the Circular Gauge. **Circular Gauge** has several elements such as pointers, label, customLabel, scales, etc.

Copyright © 2001 - 2020 Syncfusion Inc.

- There is a need for Tooltip feature in the Circular Gauge control because whenever the text hides or overrides with other gauge elements, it may not be fully visible. For resolving those problems Tooltip feature has been implemented in the Circular Gauge control.

Default Tooltip

- Tooltip has three attributes in it. The first two attributes such as ShowLabelTooltip and ShowCustomLabelTooltip are for enabling the Tooltip for label as well as custom label in default appearance.
- ShowLabelTooltip is to enable the Tooltip for labels and ShowCustomLabelTooltip is for enabling the Tooltip option for customLabels.

ASPX-CS

```
@(Html.EJ().CircularGauge("circularGaugeTooltip")
//Defines the tooltip object.
.Tooltip(tooltip=>tooltip
// Enables the label tooltip.
.ShowLabelTooltip(true)
// Enables the custom label tooltip.
.ShowCustomLabelTooltip(true))
//Customizes the scale options.
.Scales(SC =>{SC.Radius(130).ShowLabels(true)
//Customizes the custom label options.
.CustomLabels(cl => {cl.Value("0 9 5 3 4 5").Font(f
=>f.Size("20px").FontFamily("Arial").FontStyle("bold")).Position(position =>
position.X(180).Y(200))
.Add();})
//Customizes the pointers options.
.Pointers(PO =>{PO.Value(60).Length(90)
.Add();})
.Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
//// GET: /ToolTip/
public ActionResult Tooltip()
{
return View();
}
}
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Gauge with Default ToolTip

Tooltip Template

In **Tooltip** option, you can customize the Tooltip window by adding the tooltip template on that page with the help of API TemplateID. Refer to the following code example to know more about Tooltip template.

ASPX-CS

```
@(Html.EJ().CircularGauge("circularGaugeTooltip")
//Defines the tooltip object.
.Tooltip(tooltip=>tooltip
// Enables the label tooltip.
.ShowLabelTooltip(true)
// Enables the custom label tooltip.
.ShowCustomLabelTooltip(true)
// Enables the tooltip template ID.
.TemplateID("Tooltip"))
// Customizes the scale options.
.Scales(SC =>{SC.Radius(130).ShowLabels(true)
// Customizes the custom label options.
.CustomLabels(cl => {cl.Value("0 9 5 3 4 5").Font(f
=>f.Size("20px").FontFamily("Arial").FontStyle("bold")).Position(position =>
position.X(180).Y(200)).Add();})
// Customizes the pointers options..pointers(PO
=>{PO.Value(60).Length(90).Add();})
.Add();
})
)
```

C#

```
public partial class CircularGaugeController : Controller
{
// GET: /ToolTip/
public ActionResult Tooltip()
{
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code to render the following output.



Circular gauge with tooltip template

Sub Gauges

A **Circular Gauge** containing another circular gauge is said to be SubGauges. In order to make a sample like watch that has second gauge, minute gauge and hour gauge, sub gauges are used.

Adding SubGauges

Sub gauge collection is directly added to the scale object. Refer the following code example to add custom sub gauge collection in a **Gauge** control

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge")
.Value(50)
.Scales(scale =>
{
scale.Radius(190)
//For setting sub gauge
.SubGauges(gauge =>
{
gauge.ControlID("Subgauge1")
.Height(200)
.Width(200)
.Position(p =>p.X(100).Y(120)).Add();
}).Add();
})
)
@(Html.EJ().CircularGauge("Subgauge1")
.BackgroundColor("Blue")
.Value(50)
.Scales(scale=>{scale.Radius(150).Add();})
)
```

Basic Customization

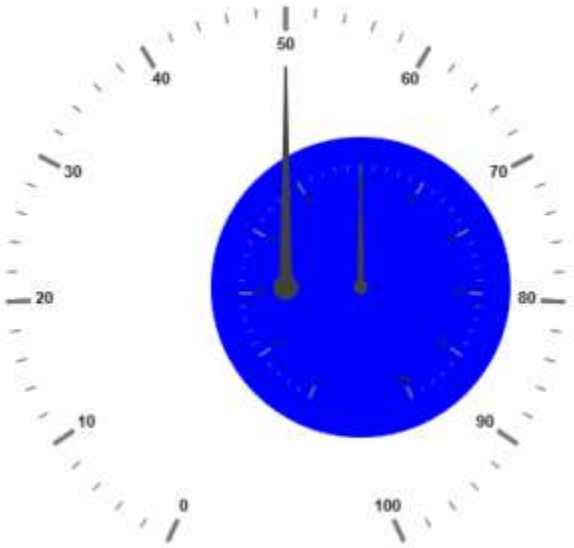
Basic attributes such as Height and Width property are used to set height and width of the sub gauge. You can easily position the gauge in another gauge using the Position object and by giving the X and Y Coordinates value. **ControlID** attribute is used to specify the sub gauge ID.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().CircularGauge("Subgauge1")
.BackgroundColor("Blue")
.Value(50)
.Radius(110)
.Scales(scale=>{scale.Radius(110).Add();})
)
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.Value(50)
.Scales(scale =>
{
scale.Radius(190)
.SubGauges(gauge =>
{
//For setting sub gauge control ID
gauge.ControlID("Subgauge1")
//For setting sub gauge Height
.Height(250)
//For setting sub gauge width
.Width(250)
//For setting sub gauge position
.Position(p =>p.X(150).Y(100)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.



Circular Gauge with sub gauge

Multiple SubGauges

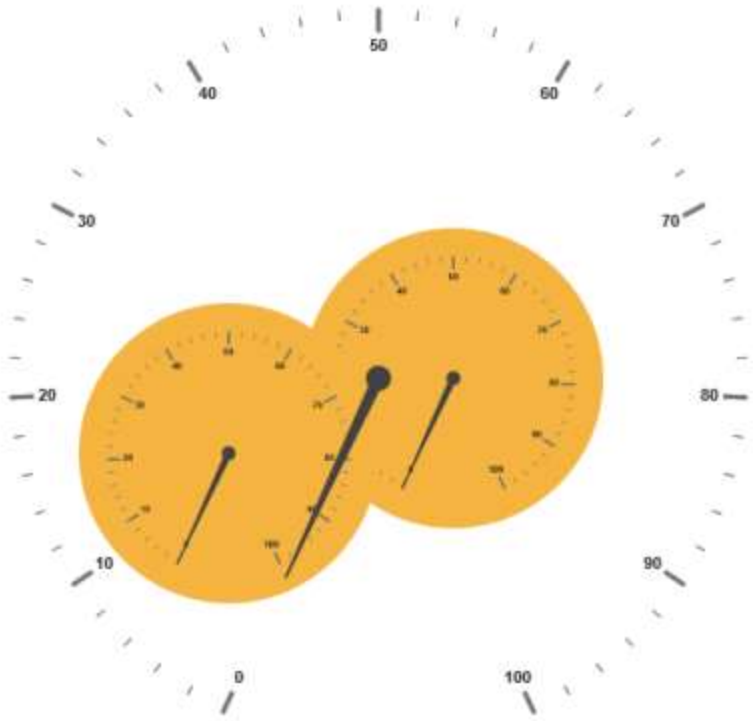
You can set multiple sub gauges in a single **Circular Gauge** by adding an array of sub gauge objects. Refer the following code example for multiple sub gauges functionality.

ASPX-CS

```
@(Html.EJ().CircularGauge("Subgauge1")
.BackgroundColor("#f5b43f")
.Scales(scale => { scale.Radius(150).Add(); })
)
@(Html.EJ().CircularGauge("Subgauge2")
.BackgroundColor("#f5b43f")
.Scales(scale => { scale.Radius(150).Add(); })
)
@(Html.EJ().CircularGauge("circulargauge")
.Height(500)
.Width(500)
.Scales(scale =>
{
scale.Radius(250)
.SubGauges(gauge =>
{
//Sub gauge1
gauge.ControlID("Subgauge1")
.Height(200)
.Width(200)
.Position(p =>p.X(200).Y(150)).Add();
//Sub gauge2
gauge.ControlID("Subgauge2")
.Height(200)
.Width(200)
.Position(p =>p.X(50).Y(200)).Add();
}).Add();
})
)
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with multiple sub gauges

Gauge Position

Semi-circular Gauge can be positioned within the canvas element which provides better appearance for the gauge in the canvas.

Positioning

- Semi-circular Gauge can be positioned with the help of the attribute called GaugePosition. It is an enumerable value. You can position the gauge away from the corner with the help of the DistanceFromCorner attribute.
- The possible enum values for the gaugePosition are as follows:
    1. TopLeft
    2. TopCenter
    3. TopRight
    4. MiddleLeft
    5. Center
    6. MiddleRight
    7. BottomLeft
    8. BottomCenter
    9. BottomRight

ASPX-CS

```
@(Html.EJ().CircularGauge("SemiCircularGauge").BackgroundColor("transparent"
)
```

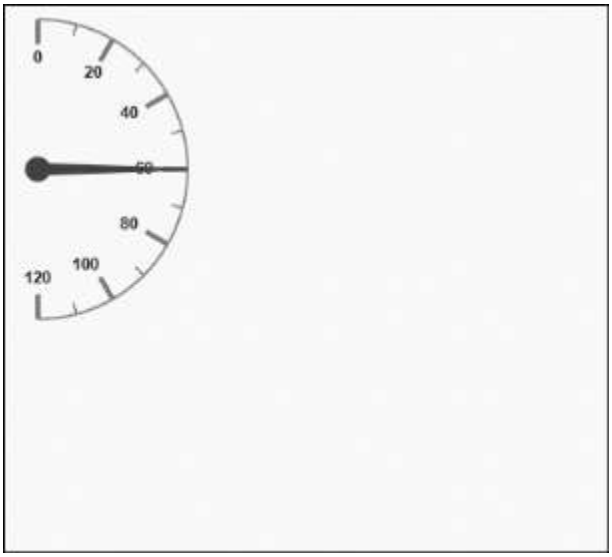
Copyright © 2001 - 2020 Syncfusion Inc.

```
// To set dimension of the canvas.
.Width(500).Height(500).Radius(100).Value(60)
// To set gauge position.
.GaugePosition(GaugePosition.TopLeft)
// To set gauge distance from corner.
.DistanceFromCorner(25)
// To set frame settings.
.Frame(fr=>fr.FrameType(Frame.HalfCircle)
// To set frame start angle.
.HalfCircleFrameStartAngle(270)
// To set frame end angle.
.HalfCircleFrameEndAngle(90))
// To set basic scale settings.
.Scales(SC =>{SC.Radius(100)
.Maximum(120)
.MajorIntervalValue(20)
.MinorIntervalValue(10)
.ShowScaleBar(true)
.Size(1)
.Border(border => border.Width(0.5))
.StartAngle(270)
.SweepAngle(180).Add();}))
```

C#

```
public partial class CircularGaugeController : Controller
{
//        // GET: /ToolTip/
public ActionResult Semicircular()
{
return View();
}
}
```

Execute the above code to render the following output.



Semi-circular Gauge with default top left position

Exporting

- Circular Gauge has an exporting feature that converts Gauge control into image format and then export in client side. The method API `exportImage` is used to export the Circular Gauge.
- It has two arguments such as **file name** and **file format** to specify the file name and file formats. For exporting refer the following code example.

JS

```
@(Html.EJ().CircularGauge("circulargauge"))
<input type="submit" value="Export Image" id="btnExportImage" />
<script type="text/javascript">
$(function () {
$("#btnExportImage").ejButton({ width: "100px", click: "buttonClickEvent",
});
});
function buttonClickEvent() {
var FileName = $("#txtFileName").val();
var FileFormat = $("#ddlFileType").val();
$("#circulargauge").ejCircularGauge("exportImage", FileName, FileFormat);
}
</script>
```

Execute the above code to render the following output.



Circular Gauge control Export Functionality

Methods and Events

Public Methods

Destroying the Circular Gauge

The `destroy` method is used to destroy the **CircularGauge** widget. All events bound using this._on will be unbind automatically and bring the control to pre-init state.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script type="text/javascript">
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.destroy();
</script>
```

Getting Back Needle Length

The getBackNeedleLength method is used to get the needle length of **CircularGauge**.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({ scales: [{ pointers: [{
showBackNeedle: true }] }] });
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getBackNeedleLength(0, 0);
</script>
```

Getting Custom Label Angle

The getCustomLabelAngle method is used to get the angle of custom label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales:
[{customLabels:[{textAngle:30,value:"MyLabel",position:{x:250,y:300},color:"
#fc0606",font:{size: "20px", fontFamily: "Arial", fontStyle: "Bold" }}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getCustomLabelAngle(0, 0);
</script>
```

Getting Custom Label value

The getCustomLabelValue method is used to get the value of custom label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales:
[{customLabels:[{textAngle:30,value:"MyLabel",position:{x:250,y:300},color:"
#fc0606",font:{size: "20px", fontFamily: "Arial", fontStyle: "Bold" }}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getCustomLabelValue(0, 0);
</script>
```

Getting Label Angle

The getLabelAngle method is used to get angle of label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getLabelAngle(0, 0);
</script>
```

Getting Label Distance From Scale

The getLabelDistanceFromScale method is used to get the distance value from scale for label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getLabelDistanceFromScale(0, 0);
</script>
```

Getting Label Placement

The getLabelPlacement method is used to get placement of label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getLabelPlacement(0, 0);
</script>
```

Getting Label Style

The getLabelStyle method is used to get style of label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getLabelStyle(0, 0);
</script>
```

Getting Major Interval Value

The getMajorIntervalValue method is used to get major interval value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getMajorIntervalValue(0);
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Maker Distance From Scale

The getMarkerDistanceFromScale method is used to get distance from scale value of marker.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getMarkerDistanceFromScale(0);
</script>
```

Getting Maker Style

The getMarkerStyle method is used to get distance from scale value of marker.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getMarkerStyle(0, 0);
</script>
```

Getting Maximum Value

The getMaximumValue method is used to get maximum value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getMaximumValue(0);
</script>
```

Getting Minimum Value

The getMinimumValue method is used to get minimum value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getMinimumValue(0);
</script>
```

Getting Minor Interval Value

The getMinorIntervalValue method is used to get minor interval value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getMinorIntervalValue(0);
</script>
```

Getting Needle Style

The getNeedleStyle method is used to get style of needle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getNeedleStyle(0, 0);
</script>
```

Getting Pointer Cap Border Width

The getPointerCapBorderWidth method is used to get border width of pointer cap.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerCapBorderWidth(0);
</script>
```

Getting Pointer Cap Radius

The getPointerCapRadius method is used to get radius of pointer cap.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerCapRadius(0);
</script>
```

Getting Pointer Length

The getPointerLength method is used to get pointer length.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerLength(0, 0);
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Pointer Needle Type

The getPointerNeedleType method is used to get needle type of pointer.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerNeedleType(0, 0);
</script>
```

Getting Pointer Placement

The getPointerPlacement method is used to get placement of pointer.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerPlacement(0, 0);
</script>
```

Getting Pointer Value

The getPointerValue method is used to get pointer value.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerValue(0, 0);
</script>
```

Getting Pointer Value

The getPointerWidth method is used to get pointer width.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getPointerWidth(0, 0);
</script>
```

Getting Range Border Width

The getRangeBorderWidth method is used to get border width of range.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getRangeBorderWidth(0, 0);
</script>
```

Getting Range Distance From Scale

The getRangeDistanceFromScale method is used to get range distance from scale.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getRangeDistanceFromScale(0, 0);
</script>
```

Getting Range End Value

The getRangeEndValue method is used to get end value of range.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getRangeEndValue(0, 0);
</script>
```

Getting Range Position

The getRangePosition method is used to get range position.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getRangePosition(0, 0);
</script>
```

Getting Range Size

The getRangeSize method is used to get range size.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
circulargaugeObj.getRangeSize(0, 0);
</script>
```

Getting Range Start Value

The getRangeStartValue method is used to get range start value.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getRangeStartValue(0, 0);
</script>
```

Getting Scale Bar Size

The getScaleBarSize method is used to get scale bar size.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getScaleBarSize(0);
</script>
```

Getting Scale Border Width

The getScaleBorderWidth method is used to get border width of scale.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({ scales: [{showScaleBar: true,
size: 6, border:{Width: 1.5} }] });
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getScaleBorderWidth(0);
</script>
```

Getting Scale Direction

The getScaleDirection method is used to get scale direction.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getScaleDirection(0);
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Scale Radius

The getScaleRadius method is used to get scale radius.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getScaleRadius(0);
</script>
```

Getting Start Angle

The getStartAngle method is used to get start angle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getStartAngle(0);
</script>
```

Getting Sub Gauge Location

The getSubGaugeLocation method is used to get location of subGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<div id="subGauge1"></div>
<script>
$("#subGauge1").ejCircularGauge({backgroundColor: "#f5b43f",scales:
[{radius: 150}]});
$("#CoreCircularGauge").ejCircularGauge({height: 500,width: 500,scales:
[{radius: 250,subGauges: [{controlID: "subGauge1",height: 400,width:
200,position: { x: 200, y: 150 }}]}]});
circulargaugeObj.getSubGaugeLocation(0, 0);
</script>
```

Getting Sweep Angle

The getSweepAngle method is used to get sweep angle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getSweepAngle(0);
</script>
```

Getting Tick Angle

The getTickAngle method is used to get tick angle.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getTickAngle(0, 0);
</script>
```

Getting Tick Distance From Scale

The getTickDistanceFromScale method is used to get tick distance from scale value.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getTickDistanceFromScale(0, 0);
</script>
```

Getting Tick Height

The getTickHeight method is used to get tick height.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getTickHeight(0, 0);
</script>
```

Getting Tick Placement

The getTickPlacement method is used to get tick placement.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getTickPlacement(0, 0);
</script>
```

Getting Tick Style

The getTickStyle method is used to get tick style.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getTickStyle(0, 0);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

Getting Tick Width

The getTickWidth method is used to get tick width.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.getTickWidth(0, 0);
</script>
```

Setting IncludeFirstValue

The includeFirstValue method is used to set includeFirstValue.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.includeFirstValue(0, 0, false);
</script>
```

Redrawing Circular Gauge

The redraw method is used to redraw the Circular Gauge widget.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.redraw("scale");
</script>
```

Setting Back Needle Length

The setBackNeedleLength method is used to set the needle length of **CircularGauge**.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({ scales: [{ pointers: [{
showBackNeedle: true }] }] });
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setBackNeedleLength(0, 0, 10);
</script>
```

Setting Custom Label Angle

The setCustomLabelAngle method is used to set the angle of custom label.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales:
[{customLabels:[{value:"MyLabel",position:{x:250,y:300},color:"#fc0606",font
: { size: "20px", fontFamily: "Arial", fontStyle: "Bold" }}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setCustomLabelAngle(0, 0, 10);
</script>
```

Setting Custom Label value

The setCustomLabelValue method is used to set the value of custom label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales:
[{customLabels:[{value:"MyLabel",position:{x:180,y:300},color:"#fc0606",font
:{size: "20px", fontFamily: "Arial", fontStyle: "Bold" }}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setCustomLabelValue(0, 0, "CircularGauge");
</script>
```

Setting Label Angle

The setLabelAngle method is used to set angle of label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setLabelAngle(0, 0, 10);
</script>
```

Setting Label Distance From Scale

The setLabelDistanceFromScale method is used to set the distance value from scale for label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setLabelDistanceFromScale(0, 0, 10);
</script>
```

Setting Label Placement

The setLabelPlacement method is used to set placement of label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setLabelPlacement(0, 0, 'far');
</script>
```

Setting Label Style

The setLabelStyle method is used to set style of label.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setLabelStyle(0, 0, 'major');
</script>
```

Setting Major Interval Value

The setMajorIntervalValue method is used to set major interval value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setMajorIntervalValue(0, 10);
</script>
```

Setting Maker Distance From Scale

The setMarkerDistanceFromScale method is used to set distance from scale value of marker.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setMarkerDistanceFromScale(0, 10);
</script>
```

Setting Maker Style

The setMarkerStyle method is used to set distance from scale value of marker.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setMarkerStyle(0, 0, 'rectangle');
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Setting Maximum Value

The setMaximumValue method is used to set maximum value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setMaximumValue(0, 130);
</script>
```

Setting Minimum Value

The setMinimumValue method is used to set minimum value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setMinimumValue(0, 10);
</script>
```

Setting Minor Interval Value

The setMinorIntervalValue method is used to set minor interval value of CircularGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setMinorIntervalValue(0, 2);
</script>
```

Setting Needle Style

The setNeedleStyle method is used to set style of needle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setNeedleStyle(0, 0, 'arrow');
</script>
```

Setting Pointer Cap Border Width

The setPointerCapBorderWidth method is used to set border width of pointer cap.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerCapBorderWidth(0, 5);
</script>
```

Setting Pointer Cap Radius

The setPointerCapRadius method is used to set radius of pointer cap.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerCapRadius(0, 10);
</script>
```

Setting Pointer Length

The setPointerLength method is used to set pointer length.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerLength(0, 0, 90);
</script>
```

Setting Pointer Needle Type

The setPointerNeedleType method is used to set needle type of pointer.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerNeedleType(0, 0, 'triangle');
</script>
```

Setting Pointer Placement

The setPointerPlacement method is used to set placement of pointer.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerPlacement(0, 0,'near');
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Setting Pointer Value

The setPointerValue method is used to set pointer value.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerValue(0, 0, 10);
</script>
```

Setting Pointer Value

The setPointerWidth method is used to set pointer width.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setPointerWidth(0, 0, 10);
</script>
```

Setting Range Border Width

The setRangeBorderWidth method is used to set border width of range.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setRangeBorderWidth(0, 0, 5);
</script>
```

Setting Range Distance From Scale

The setRangeDistanceFromScale method is used to set range distance from scale.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setRangeDistanceFromScale(0, 0, 10);
</script>
```

Setting Range End Value

The setRangeEndValue method is used to set end value of range.

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setRangeEndValue(0, 0, 70);
</script>
```

Setting Range Position

The setRangePosition method is used to set range position.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setRangePosition(0, 0, 'far');
</script>
```

Setting Range Size

The setRangeSize method is used to set range size.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setRangeSize(0, 0, 10);
</script>
```

Setting Range Start Value

The setRangeStartValue method is used to set range start value.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({scales: [{showRanges: true,ranges:
[{distanceFromScale: -30,startValue: 0,endValue: 70}]}]});
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setRangeStartValue(0, 0, 10);
</script>
```

Setting Scale Bar Size

The setScaleBarSize method is used to set scale bar size.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setScaleBarSize(0, 160);
</script>
```

Setting Scale Border Width

The setScaleBorderWidth method is used to set border width of scale.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge({ scales: [{showScaleBar: true,
size: 6, border:{Width: 1.5} }] });
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setScaleBorderWidth(0, 3);
</script>
```

Setting Scale Direction

The setScaleDirection method is used to set scale direction.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setScaleDirection(0, 'clockwise');
</script>
```

Setting Scale Radius

The setScaleRadius method is used to set scale radius.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setScaleRadius(0, 140);
</script>
```

Setting Start Angle

The setStartAngle method is used to set start angle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setStartAngle(0, 10);
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Setting Sub Gauge Location

The setSubGaugeLocation method is used to set location of subGauge.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<div id="subGauge1"></div>
<script>
$("#subGauge1").ejCircularGauge({backgroundColor: "#f5b43f",scales:
[{radius: 150}]});
$("#CoreCircularGauge").ejCircularGauge({height: 500,width: 500,scales:
[{radius: 250,subGauges: [{controlID: "subGauge1",height: 400,width:
200,position: { x: 200, y: 150 }}]}]});
circulargaugeObj.setSubGaugeLocation(0, 0, {x:50,y:100});
</script>
```

Setting Sweep Angle

The setSweepAngle method is used to set sweep angle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setSweepAngle(0, 220);
</script>
```

Setting Tick Angle

The setTickAngle method is used to set tick angle.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setTickAngle(0, 0, 10);
</script>
```

Setting Tick Distance From Scale

The setTickDistanceFromScale method is used to set tick distance from scale value.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setTickDistanceFromScale(0, 0, 15);
</script>
```

Setting Tick Height

The setTickHeight method is used to set tick height.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setTickHeight(0, 0, 10);
</script>
```

Setting Tick Placement

The setTickPlacement method is used to set tick placement.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setTickPlacement(0, 0, 'near');
</script>
```

Setting Tick Style

The setTickStyle method is used to set tick style.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setTickStyle(0, 0, 'minor');
</script>
```

Setting Tick Width

The setTickWidth method is used to set tick width.

JS

```
@(Html.EJ().CircularGauge("CoreCircularGauge"))
<script>
$("#CoreCircularGauge").ejCircularGauge();
var circulargaugeObj = $("#CoreCircularGauge").data("ejCircularGauge");
circulargaugeObj.setTickWidth(0, 0, 5);
</script>
```

Events

Draw Custom Label

The DrawCustomLabel event is triggered while custom labels are drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw custom label event
.DrawCustomLabel("onDrawCustomLabel")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//...
)
function onDrawCustomLabel(sender) {
// do something
}
```

Draw Indicators

The DrawIndicators event is triggered while indicators are being drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw indicators event
.DrawIndicators("onDrawIndicators")
//...
)
function onDrawIndicators(sender) {
// do something
}
```

Draw Labels

The DrawLabels event is triggered while labels are being drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw labels event
.DrawLabels("onDrawLabels")
//...
)
function onDrawLabels(sender) {
// do something
}
```

Draw Pointer Cap

The DrawPointerCap event is triggered while pointer cap is being drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw pointer cap event
.DrawPointerCap("onDrawPointerCap")
//...
)
function onDrawPointerCap(sender) {
// do something
}
```

Draw Pointers

The DrawPointers event is triggered while pointer cap is being drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw pointers event
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.DrawPointers("onDrawPointers")
//...
)
function onDrawPointers(sender) {
// do something
}
```

Draw Range

The DrawRange event is triggered when ranges starts to be drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw range event
.DrawRange("onDrawRange")
//...
)
function onDrawRange(sender) {
// do something
}
```

Draw Ticks

The DrawTicks event is triggered when ticks are being drawn on the gauge.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//draw ticks event
.DrawTicks("onDrawTicks")
//...
)
function onDrawTicks(sender) {
// do something
}
```

Load

The Load event is triggered when gauge starts to load.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//load event
.Load("onLoad")
//...
)
function onLoad(sender) {
// do something
}
```

Mouse Click

The MouseClick event is triggered when left mouse button is clicked.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//mouse click event
.MouseClick("onMouseClick")
//...
)
function onMouseClick(sender) {
// do something
}
```

Mouse Click Move

The MouseClickMove event is triggered when clicking and dragging the mouse pointer over the gauge pointer.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//mouse click move event
.MouseClickMove("onMouseClickMove")
//...
)
function onMouseClickMove(sender) {
// do something
}
```

Mouse Click Up

The MouseClickUp event is triggered when clicking and dragging the mouse pointer over the gauge pointer.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//mouse click up event
.MouseClickUp("onMouseClickUp")
//...
)
function onMouseClickUp(sender) {
// do something
}
```

Render Complete

The RenderComplete event is triggered when rendering of the gauge is completed.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//render complete event
.RenderComplete("onRenderComplete")
//...
)
function onRenderComplete(sender) {
// do something
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Range Mouse Move

The RangeMouseMove event is triggered when moving mouse on ranges.

ASPX-CS

```
@(Html.EJ().CircularGauge("circulargauge1")
//range mouse move event
.RangeMouseMove("onRangeMouseMove")
//...
)
function onRangeMouseMove(sender) {
// do something
}
```

DoubleClick

The DoubleClick event is triggered when double clicking the gauges.

ASPX-CS

```
@(Html.EJ().CircularGauge("gauge")
//double click event
.DoubleClick("onDoubleClick")
//...
)
function onDoubleClick(sender) {
// do something
}
```

RightClick

The RightClick event is triggered when right clicking the gauges.

ASPX-CS

```
@(Html.EJ().CircularGauge("gauge")
//right click event
.RightClick("onRightClick")
//...
)
function onRightClick(sender) {
// do something
}
```

MVVM

AngularJS

Circular Gauge contains AngularJS support. You can add object as well as array object in the Circular Gauge. The two way binding support is given to the pointer value, minimum scale value and maximum scale value.

Rendering the Circular Gauge

ej-CircularGauge is the control tag in which ej is tag prefix and CircularGauge is the control name.The following code example helps you to render Circular Gauge.

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div ng-app="syncApp">
<div ng-controller="CircularGauge">
<ej-CircularGauge id="CircularGauge1" e-backgroundColor="transparent" e-
value="50"
e-width="500" e-readOnly="false" e-load= "loadGaugeTheme"
e-enableAnimation="false">
</ej-CircularGauge>
</div>
</div>
```

JS

```
<script type="text/javascript">
<!—binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('CircularGauge', function ($scope) {
$scope.value = 50;
$scope.minimum = 0;
$scope.maximum = 120;
});
</script>
```

Execute the above code to render the output as follows.



Default Circular gauge in AngularJS

Adding Scale Collection

Scale is an array object and you can use the inner tag for it. Object in the array collection (i.e. border) is extended with hyphen in the same tag.

Example: e-border-width and e-border-color.

HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
<!--Adding Scale collection to the Circular gauge-->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<e-scales>
<e-scale e-showRanges="true" e-startAngle="122" e-sweepAngle="296"
e-radius="130" e-showScaleBar="true" e-size="1" e-maximum="120"
e-majorIntervalValue="20" e-minorIntervalValue="10"
e-border-width="0.5">
</e-scale>
</e-scales>
</ej-CircularGauge>
```

Execute the above code to render the following output.





Circular gauge with scale customization in AngularJS

Adding Pointer Collection

Pointer is an array object and you can use the inner tag for it. Object in the array collection (i.e. pointer cap) is extended with hyphen in the same tag.

Example: e-pointerCap-radius.

HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
<!--Adding Scale collection to the Circular gauge-->
<e-scales>
<e-scale>
<!--Adding pointer collection to the scale collection-->
<e-pointers>
<e-pointer e-showBackNeedle="true" e-backNeedleLength="20"
e-length="95" e-width="7" e-value="80"
e-pointerCap-radius="12">
</e-pointer>
</e-pointers>
</e-scale>
</e-scales>
</ej-CircularGauge>
```

Execute the above code to render the output as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



50

Circular gauge with pointer customization in AngularJS

Adding Label Collection

Label is also an array object. You can use the inner tag for it.

HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
<!--Adding Scale collection to the Circular gauge-->
<e-scales>
<e-scale>
<!--Adding pointer collection to the scale collection-->
<e-pointers>…</e-pointers>
<!--Adding labels collection to the scale collection-->
<e-labels>
<e-label e-color="#8c8c8c">
</e-label>
</e-labels>
</e-scale>
</e-scales>
</ej-CircularGauge>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



50

Circular gauge with label customization in AngularJS

Adding Tick Collection

Tick is an array object. You can use the inner tag for it.

HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
<!--Adding Scale collection to the Circular gauge-->
<e-scales>
<e-scale>
<!--Adding pointer collection to the scale collection-->
<e-pointers>…</e-pointers>
<!--Adding labels collection to the scale collection-->
<e-labels>…</e-labels>
<!--Adding ticks collection to the scale collection-->
<e-ticks>
<e-tick e-type="major" e-distanceFromScale="2" e-height="16"
e-width="1" e-color="#8c8c8c">
</e-tick>
<e-tick e-type="minor" e-distanceFromScale="2" e-height="8"
e-width="1" e-color="#8c8c8c">
</e-tick>
</e-ticks>
</e-scale>
</e-scales>
</ej-CircularGauge>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



50

Circular gauge with ticks customization in AngularJS

Adding Range Collection

Range is an array object. You can use the inner tag for it. Object in the array collection (i.e. border) is extended with hyphen in the same tag.

Example: e-border-color.

HTML

```
<!--To Render the Circular gauge-->
<ej-CircularGauge id="CircularGauge1">
<!--Adding Scale collection to the Circular gauge-->
<e-scales>
<e-scale>
<!--Adding pointer collection to the scale collection-->
<e-pointers>…</e-pointers>
<!--Adding labels collection to the scale collection-->
<e-labels>…</e-labels>
<!--Adding ticks collection to the scale collection-->
<e-ticks>…</e-ticks>
<!--Adding ranges collection to the scale collection-->
<e-ranges>
<e-range e-distanceFromScale="-30" e-startValue="0" e-endValue="70">
</e-range>
<e-range e-distanceFromScale="-30" e-startValue="70"
e-endValue="110" e-backgroundColor="#fc0606"
e-border-color="#fc0606">
</e-range>
<e-range e-distanceFromScale="-30" e-startValue="110"
e-endValue="120" e-backgroundColor="#f5b43f"
e-border-color="#f5b43f">
</e-range>
</e-ranges>
</e-scale>
</e-scales>
</ej-CircularGauge>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code to render the following output.



Circular gauge Rendered in AngularJS

Two Way Binding

Circular Gauge support the two way binding for the property value, minimum and maximum as mentioned earlier. The following code example explains how to achieve the two way binding in Circular Gauge.

HTML

```
<div ng-app="syncApp">
<div ng-controller="CircularGauge">
<div>
<ej-CircularGauge id="CircularGauge1" e-backgroundColor="transparent" e-
value="nvalue" e-width="500" e-readOnly="false" e-load= "loadGaugeTheme" e-
enableAnimation="false">
<e-scales>
<e-scale e-showRanges="true" e-startAngle="122" e-sweepAngle="296"
e-radius="130" e-showScaleBar="true" e-size="1"
<!—binding maximum value using angular JS -->
e-maximum="maximum"
<!—binding minimum value using angular JS -->
e-minimum="minimum"
e-majorIntervalValue="20"
e-minorIntervalValue="10" e-border-width="0.5">
<e-pointers>
<e-pointer e-showBackNeedle="true" e-backNeedleLength="20"
e-length="95" e-width="7"
<!—binding pointer value using angular JS -->
e-value="value"
e-pointerCap-radius="12">
</e-pointer>
</e-pointers>
<e-labels>
<e-label e-color="#8c8c8c"></e-label>
</e-labels>
<e-ticks>
<e-tick e-type="major" e-distanceFromScale="2" e-height="16"
e-width="1" e-color="#8c8c8c">
</e-tick>
```

```
<e-tick e-type="minor" e-distanceFromScale="2" e-height="8"
e-width="1" e-color="#8c8c8c">
</e-tick>
</e-ticks>
<e-ranges>
<e-range e-distanceFromScale="-30" e-startValue="0" e-endValue="70">
</e-range>
<e-range e-distanceFromScale="-30" e-startValue="70"
e-endValue="110" e-backgroundColor="#fc0606"
e-border-color="#fc0606">
</e-range>
<e-range e-distanceFromScale="-30" e-startValue="110"
e-endValue="120" e-backgroundColor="#f5b43f"
e-border-color="#f5b43f">
</e-range>
</e-ranges>
</e-scale>
</e-scales>
</ej-CircularGauge>
</div>
<input type="text" id="txtMax" e-value="nvalue" ej-numerictextbox ng-
model="nvalue"  e-decimalplaces="2" e-showspinbutton="false"
Style="width:110px"/>
</div>
</div>
<script type="text/javascript">
<!—binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('CircularGauge', function ($scope) {
$scope.value = 50;
$scope.minimum = 0;
$scope.maximum = 120;
});
</script>
```

Execute the above code to render the following output.



50

Circular gauge with two way binding in AngularJS

KnockoutJS

- KnockoutJS support allows you to bind the HTML elements against any of the available data models.Two types of KnockoutJS binding is supported as of AngularJS,
    1. one-way binding
    2. two-way binding
- One way binding refers to the process of applying observable values to all the available properties of the Circular Gauge control, but the changes made in gauge control does not reflect and trigger in turn to the observable collection. This kind of binding applies to all the properties of the circular gauge control.
- Two-way binding supports both the processes – it applies the observable values to the Circular Gauge properties as well as the changes made in the Circular Gauge control also reflects back and triggers within the observable collections. Only few of the schedule properties support two-way binding and they are as follows
    1. value
    2. maximum
    3. minimum

ASPX-CS

```
<div id="CircularGauge1"
data-bind="ejCircularGauge: {
value:sampleValue,
minimum:minimumValue,
maximum:maximumValue
}">
</div>
```

JS

```
<script type="text/javascript">
$(function () {
window.viewModel = {
sampleValue: ko.observable(50),
minimumValue: ko.observable(10),
maximumValue: ko.observable(150)
};
$(function () {
ko.applyBindings(viewModel);
});
});
</script>
```

Execute the above code to render the following output. The following screenshot is the output of ASP.

Copyright © 2001 - 2020 Syncfusion Inc.



Circular Gauge with KnockoutJS binding

ColorPicker

Overview

The ASP.NET MVCColorPicker control provides you a rich visual interface for color selection. You can select the color from the professionally designed palettes or custom color. By clicking a point on the color, you can change the active color to the color that is located under the pointer.

You can also choose colors in different specifications; red-green-blue-alpha (opacity) (RGBA), hue-saturation-value (HSV) and hexadecimal (HEX). The ColorPicker provides a selection of basic colors, standard presets, custom colors and color swatches.

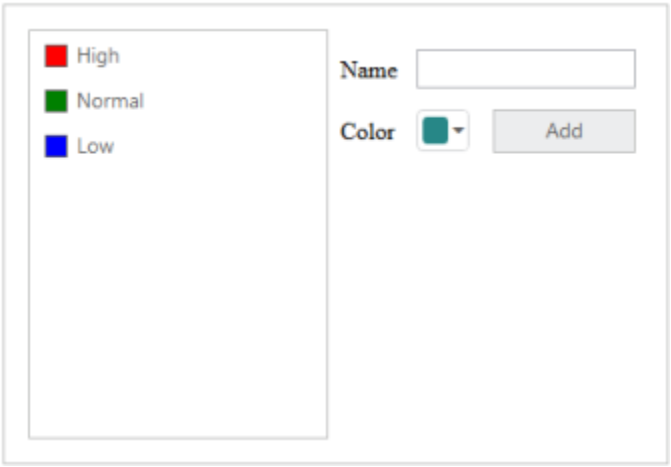
Getting Started

This section explains briefly about how to create a ColorPicker in your application with, ASP.NET MVC.

Create your first ColorPicker in ASP.NET MVC

The Essential ASP.NET MVCColorPicker control provides support for selecting the colors in different sources such as palettes, picker or custom palettes. You can also render the color value from control in three formats such as RGB, HSV and HEXCODE.

In this example, you can learn how to customize ColorPicker control in a category Application.



Copyright © 2001 - 2020 Syncfusion Inc.

ColorPicker Control

In the following sections you can learn, How to:

- Create ColorPicker control
- Initialize the other widgets
- Add Value to ListBox Control

Create ColorPicker Control

Use the following steps to create the ColorPicker control.

1. You can create a MVC Project and add necessary Dll's and Scripts with the help of the given MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page for rendering the ColorPicker window.

ASPX-CS

```
@Html.EJ().ColorPicker("CategoryColor").Value("#278787")
```

3. Run this code to render the resultant output of the above steps.



Initialize the other widgets

To add the priority value to the ListBox, the text value is obtained from the input element and color for each priority is received by ColorPicker control. To add each new priority value to ListBox control, click the Add button.

You can refer to the following link for more information on creation on ListBox

http://help.syncfusion.com

1. The following code example is used to create the Priority form using ListBox control and ColorPicker control.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="sample-area">
<div class="frame">
<div id="control">
<ul id="ColorValues">
<li><span class="color high"></span>High</li>
<li><span class="color normal"></span>Normal</li>
<li><span class="color low"></span>Low</li>
</ul>
@Html.EJ().ListBox("selectPriority").TargetID("ColorValues")
</div>
</div>
</div>
<div id="Properties">
<table class="prop-grid">
<tr class="row">
<td class="column">Name
</td>
<td class="column">
<input type="text" id="categoryName" />
</td>
</tr>
<tr class="row">
<td class="column">Color
</td>
<td class="column">
<!--Colorpicker element-->
@Html.EJ().ColorPicker("CategoryColor").Value("#278787")
</td>
<td class="column">
<!--Add button for add the new category-->
@Html.EJ().Button("AddCategory").Text("Add").Width("82px").Height("28px").Ty
pe(ButtonType.Button).ClientSideEvents(events =>
events.Click("addCategoryValue"))
</td>
</tr>
<tr class="row">
</tr>
</table>
</div>
</div>
</div>
```

2. Add the following style section to align form fields.

CSS

```
<style>
.content-container-fluid > .row {
width: 410px;
border: 1px solid #bbbcbb;
padding: 16px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.color.high {
background-color: red;
}
.color.normal {
background-color: green;
}
.color.low {
background-color: blue;
}
.sample-area {
width: 205px;
}
.sample-area, #Properties {
display: inline-block;
float: left;
}
#Properties #categoryName {
width: 140px;
height: 20px;
}
#Properties .column {
display: inline-block;
width: 45px;
margin: 10px 0 0;
}
#Properties .row {
padding: 10px 0px 5px 0px;
}
.color {
width: 13px;
height: 13px;
border: 1px solid;
display: inline-block;
margin-right: 6px;
margin-bottom: -3px;
}
</style>
```

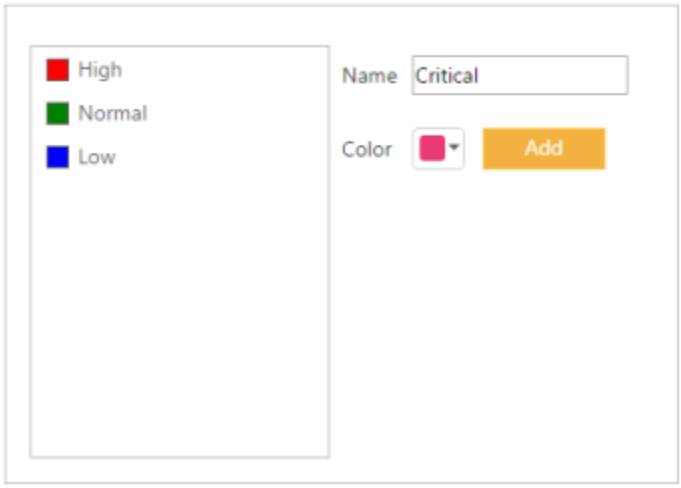
3. Initialize the element in

JS

```
<script>
var listBoxObj, colorObj;
jQuery(function ($) {
//initialize the listbox with object creation
listBoxObj = $("#selectPriority").data('ejListBox');
//initialize the colorpicker with object creation
colorObj = $("#CategoryColor").data('ejColorPicker');
});
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

4. Run the above code to render the following output.



Color Picker control

Add value to ListBox Control

You can add the value to ListBox with selected color by performing the button click event. The following script section define the click event for the button element.

1. Initialize the click event to button element in

JS

```
<script>
jQuery(function ($) {
//reuse the previous section script block
//The following function used to add the new value to the listbox control
function addCategoryValue() {
if ($("#categoryName").val() !== "") {
//To get the selected color from the colorpicker by using getValue()
listBoxObj.addItem("<span class='color' style='background-color: " +
colorObj.getValue() + "' ></span>" + $("#categoryName").val());
$("#categoryName").val("");
}
}
</script>
```

2. The following screenshot illustrates the resultant output after you click Add button.



Value from Color Picker control

Behavior Settings

ShowPreview

The ColorPicker control provides live preview support for current cursor selection color and selected color. ShowPreview property allows you to preview the selected color in the picker or from the palette.

The ShowPreview property is Boolean type and its default value is true.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").ShowPreview(true)
```

The following screenshot displays the output of the above code example.



ColorPicker with Show Preview option

Copyright © 2001 - 2020 Syncfusion Inc.

showRecentColors

The ColorPicker control allows you to store the color values in custom list by using ShowRecentColors property. The ColorPicker keeps up to 11 colors in a custom list. By clicking the add button, the selected color from picker or palette gets added in the recent color list.

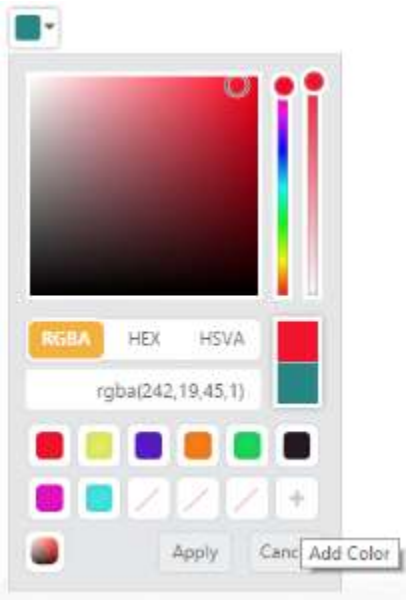
The ShowRecentColors property is Boolean type and its default value is false.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").ShowRecentColors(true
)
```

The following screenshot displays the output of the above code example.



ColorPicker with Recent Color Swatches

EnableOpacity

The ColorPicker control allows you to enable or disable the opacity slider. You can achieve this by using the EnableOpacity property.

The EnableOpacity property is Boolean type and its default value is true.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").EnableOpacity(false)
```

The following screenshot displays the output of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



ColorPicker with Opacity Slider as disabled state

Columns

The palette model consists of color values in the rows and columns order. Palette only consists of predefined colors and allows you to select anyone color from it. The Columns property allows you to modify the number of columns in palette model.

The Columns property is Number type and its default value is 10.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").Columns(9)
```

The following screenshot displays the output of the above code example.



ColorPicker with Columns

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Values

OpacityValue

The ColorPicker control allows you to change the opacity value by using the OpacityValue property. The selected color opacity is adjusted by using the OpacityValue.

The OpacityValue property is Number type and its default value is 100.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").OpacityValue(40)
```

The following screenshot displays the output of the above code example.



ColorPicker with Opacity Value

Button and TooltipText

ButtonText

The ColorPicker control allows you to define the text to be displayed in button elements. You can specify the text by using ButtonText property. In ColorPicker control, popup contains two button elements “Apply” and “Cancel”.

To configure the ButtonText property for the button elements, use the corresponding default values listed in the following table.

List of Button elements

Element	Default value
Apply	Apply
Cancel	Cancel

Copyright © 2001 - 2020 Syncfusion Inc.

TooltipText

The ColorPicker control consists of more number of sub controls and elements. To provide some information about each element and sub control, you can use the tooltip concept and you can achieve this by using TooltipText property.

To configure the TooltipText, use the following listed elements and its corresponding default value.

List of Tooltip elements

Element	Default value
Switcher	Switcher
AddButton	Add Color
Basic	Basic
MonoChrome	Mono Chrome
FlatColors	Flat Colors
SeaWolf	Sea Wolf
WebColors	Web Colors
Sandy	Sandy
PinkShades	Pink Shades
Misty	Misty
Vintage	Vintage
MoonLight	Moon Light
CandyCrush	Candy Crush
CurrentColor	Current Color
SelectedColor	Selected Color
Citrus	Citrus

When it is necessary to set the button text and TooltipText values in Spanish culture, the ColorPicker allow you to define the culture values to ButtonText and TooltipText property. The following section explains on how to define the Spanish culture values to ColorPicker control.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").ColorPickerButtonText
(color=>
color.Apply("aplicar").Cancel("cancelar")).ColorPickerTooltipText(tooltip=>
tooltip.Sandy("arenoso"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays the output of the above code example.



ColorPicker with Culture Support

Appearance and Styling

modelType

The ColorPicker allows you to define the model type to be displayed in control at initial time by using the property *ModelType*.

The ModelType property is Enum type and its default value is default.

List of modelType

ModelType	Syntax	Description
Default	ModelType(ModelType.Default)	Control rendered with both model. You can switch to palette or picker model.
Picker	ModelType(ModelType.Picker)	Control rendered with picker model only.
Palette	ModelType(ModelType.Palette)	Control rendered with palette model only.

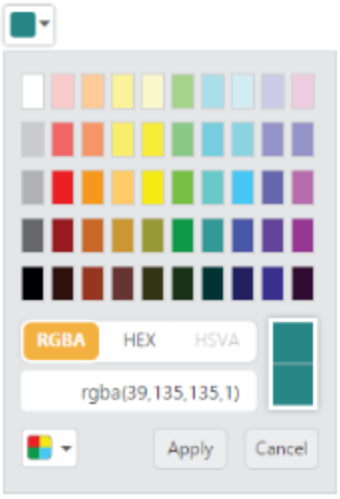
In the following code example, the ColorPicker popup model type is set as Palette when you drop down the ColorPicker popup.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").ModelType(ModelType.Palette)
```

The following screenshot displays the output of the above code example.



ColorPicker rendered with Palette Model

Palette

The ColorPicker allow you to define the palette type to be displayed in control at initial time by using the Palette property. The Palette property is Enum type and its default value is BasicPalette.

List of palette

Palette	Syntax	Description	Dependent Property
BasicPalette	Palette(PaletteType.BasicPalette)	The palette model rendered with predefined color values.	ModelType(ModelType.Palette)
CustomPalette	Palette(PaletteType.CustomPalette)	The palette model renders with the specified custom color values.	

BasicPalette

The BasicPalette type renders with predefined color values. The BasicPalette model has 12 different preset patterns. Each pattern consists of 50 colors and over 600 colors are available by default.

PresetType

The ColorPicker control allows you to define the preset model to be rendered initially in palette type. This can be achieved by using the “PresetType” property. Totally 12 types of presets are available.

The PresetType property is Enum type and its default value is “Basic”.

Property Table

PresetType	Syntax	Dependent Property
Basic	PresetType(PresetsType.Basic)	Palette(PaletteType.BasicPalette)
MonoChrome	PresetType(PresetsType.MonoChrome)	

Copyright © 2001 - 2020 Syncfusion Inc.

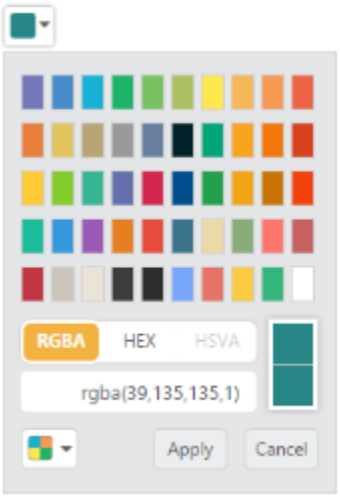
FlatColors	PresetType(PresetsType.FlatColors)	
SeaWolf	PresetType(PresetsType.SeaWolf)	
WebColors	PresetType(PresetsType.WebColors)	
Sandy	PresetType(PresetsType.Sandy)	
PinkShades	PresetType(PresetsType.PinkShades)	
Misty	PresetType(PresetsType.Misty)	
Citrus	PresetType(PresetsType.Citrus)	
Vintage	PresetType(PresetsType.Vintage)	
MoonLight	PresetType(PresetsType.MoonLight)	
CandyCrush	PresetType(PresetsType.CandyCrush)	

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").ModelType(ModelType.Palette).PresetType(PresetsType.FlatColors)
```

The following screenshot displays the output of the above code example.



ColorPicker with Presets

CustomPalette

The ColorPicker control allows you to define the custom colors in the palette model by using Palette property. Custom palettes are created by passing a comma delimited string of HEX values or an array of colors in Custom property. The CustomPalette model is only applicable when you set ModelType as Palette.

Copyright © 2001 - 2020 Syncfusion Inc.

The CustomPalette property is a dependent property of Palette and ModelType property.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@{List<String> colors = new List<string>() { "ffffff", "ffccff", "ff99ff",
"ff66ff", "ff33ff", "ff00ff", "ccffff", "ccccff","cc99ff", "cc66ff",
"cc33ff", "cc00ff", "99ffff", "99ccff", "9999ff", "9966ff","9933ff",
"9900ff", "ffffcc", "ffcccc" };}
@Html.EJ().ColorPicker("colorPicker").Value("#278787").ModelType(ModelType.P
alette).Palette(PaletteType.CustomPalette).Custom(colors)
```

The following screenshot displays the output of the above code example.



ColorPicker with Customized Colors

DisplayInline

The ColorPicker control allows you to embed the popup in the order of DOM element flow by using the DisplayInline property. Using DisplayInline property to make ColorPicker popup always in visible state. Also associate ColorPicker with element instead of input.

The DisplayInline property is Boolean type and its default value is false.

The following steps explain you how to get the ColorPicker popup in DisplayInline state.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787").DisplayInline(true).T
agName("div")
```

The following screenshot displays the output of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



ColorPicker in Display Inline

Theme Support

The ColorPicker control supports rich appearance. It supports 12 different themes of Essential ASP.NET MVC and bootstrap themes. To use these twelve themes, refer the themes files in HTML page.

You require two style sheets to apply styles to ColorPicker control; one ej.widgets.core.min.css and one ej.theme.min.css. When you use ej.widgets.all.min.css then, it is not necessary to use ej.widgets.core.min.css and ej.theme.min.css because ej.widgets.all.min.css is a combination of these two.

The core style sheet applies styles related to positioning and size, but are not related to the color scheme and always require the control to look correct and function properly. The theme style sheet applies theme-specific styles like colors and backgrounds.

The following list is the twelve themes supported by ColorPicker:

- default-theme
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

1. In the CSHTML page, configure the ColorPicker widget with built in theme support.

HTML

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
<!doctype html>
<html>
<head>
<title>Essential Studio for JavaScript : ColorPicker - Built-in
ThemeSupport</title>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"
charset="utf-8"  />
<link href="http://cdn.syncfusion.com/js/web/flat-saffron-dark/ej.web.all-
latest.min.css" rel="stylesheet" />
<script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/js/web/ej.web.all-
latest.min.js"></script>
</head>
<body>
<div id="control">
@Html.EJ().ColorPicker("colorPicker").Value("#278787")
</div>
</body>
</html>
```

Note: jQuery.easing external dependency has been removed from version 14.3.0.49 onwards. Kindly include this jQuery.easing dependency for versions lesser than 14.3.0.49 in order to support animation effects.

The following screenshot displays the output of the above code example.



ColorPicker with Theme Support

CustomCss

The ColorPicker control also allows you to customize its appearance using user-defined CSS and custom skin options such as colors and backgrounds. To apply custom themes use the CssClass property. CssClass property sets the root class for ColorPicker theme.

Using this property you can override the existing styles under the theme style sheet. The theme style sheet applies theme-specific styles like colors and backgrounds. In the following example, the value of CssClass property is set as Light-Blue. Light-Blue is added as root class to ColorPicker control at the runtime. From this root class you can customize the ColorPicker control theme.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().ColorPicker("colorPicker").Value("#278787").CssClass("Light-
Blue")
```

2. Custom CSS Styles.

CSS

```
<style>
.Light-Blue.e-colorwidget.e-widget, .Light-Blue.e-colorpicker.e-
popup,.Light-Blue.e-colorwidget .e-in-wrap.e-box .e-select, .Light-Blue.e-
colorwidget .e-in-wrap.e-box, .Light-Blue.e-colorwidget .e-down-arrow
{
background: none repeat scroll 0 0 lightblue;
}
.Light-Blue.e-colorpicker .e-footer .e-cancelButton.e-btn, .Light-Blue.e-
colorpicker .e-footer .e-applyButton.e-btn
{
background: none repeat scroll 0 0 white;
}
</style>
```

The following screenshot displays the output of above steps.



ColorPicker with Custom Theme

KnockoutJS Binding

KnockoutJS support allows you to bind the HTML elements against any of the available data models.

Two types of KnockoutJS binding is supported,

- One-way binding
- Two-way binding

One-way binding refers to the process of applying observable values to all the available properties of the ColorPicker widget. The changes made in ColorPicker widget are not reflected and triggered in turn to the observable collection. This kind of binding is applied to all the properties of the ColorPicker widget.

Copyright © 2001 - 2020 Syncfusion Inc.

Two-way binding supports both the processes. It applies the observable values to the ColorPicker widget properties and also the changes made in the ColorPicker widget are reflected back and triggered within the observable collections.

For more information about KnockoutJS binding, you can refer to the online documentation in the following link location,

http://help.syncfusion.com/aspnetmvc/colorpicker/knockout-binding

1. The following example depicts how you can bind data to the ColorPicker widget through KnockoutJS support that enables and populates data to a ColorPicker widget based on the value set to the other ColorPicker widget.

ASPX-CS

```
@*Add the following script in view page for KnockoutJS support*@
<script src="http://cdn.syncfusion.com/js/assets/external/knockout.min.js">
</script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.unobtrusive.min.js">
</script>
<script src="http://cdn.syncfusion.com/13.1.0.21/js/ej.widget.ko.min.js">
</script>
<div class="content-container-fluid">
<div class="row" style="width: 100%">
<div class="cols-sample-area" style="width: 100%">
<div class="frame" style="width: 420px">
<div id="control" style="float: left; width: 70%; margin-left: 10px">
<input id="colorpick" data-bind="ejColorPicker: { value: value, modelType:
palette }" />
<h6><span style="font-style: italic; font-weight: normal; position:
absolute; margin-top: 5px;">Note:Two Way Knockout Support</span></h6>
</div>
<div id="binding" style="float: left; width: 23%">
<input id="colorpick1" data-bind="ejColorPicker: { value: value, modelType:
picker }" />
</div>
</div>
</div>
</div>
</div>
```

ASPX-CS

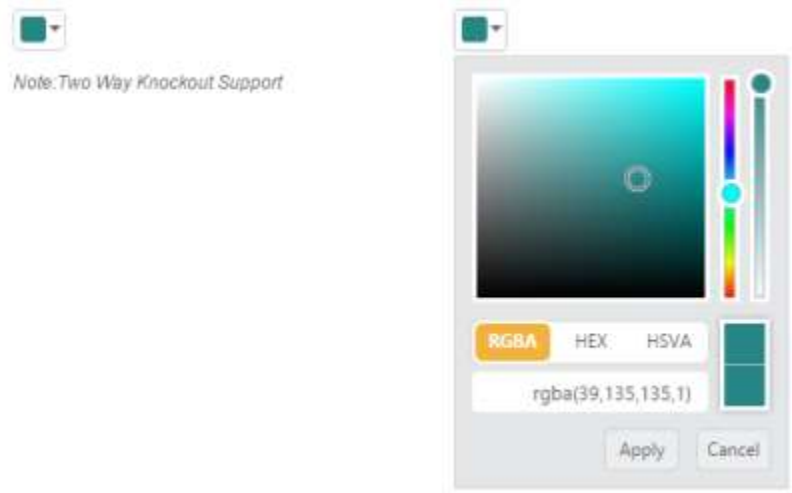
```
<script>
window.viewModel = {
value: ko.observable("#278787"),
palette: ko.observable("palette"),
picker: ko.observable("picker")
};
$(function () {
ko.applyBindings(viewModel);
});
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

CSS

```
<style>
.element {
display: inline-block;
}
.frame {
width: 600px;
border: 0px;
}
#control {
width: 600px;
}
</style>
```

The following screenshot displays the output of the above code example.



ColorPicker with KnockoutJS Support

AngularJS Binding

The ColorPicker widget is availed with two types of AngularJS support namely,

- One-way binding
- Two-way binding

One-way binding refers to the process of applying scope values to all the available properties of the ColorPicker widget. The changes made in ColorPicker widget are not reflected or triggered in turn to the scope collection. This kind of binding is applied to all the properties of the ColorPicker widget.

Two-way binding supports both the processes. It applies the scope values to the ColorPicker properties, as well as the changes made in the ColorPicker widget are reflected back and triggered within the AngularJS scope change function.

Apply the plugin and property assigning to the ColorPicker widget element through the directive that starts with “e-“.

To know more about the AngularJS binding, you can refer to the online documentation in the following link location,

Copyright © 2001 - 2020 Syncfusion Inc.

http://help.syncfusion.com/js/angularjs

1. The following code example depicts the way to bind data to the ColorPicker widget through AngularJS support.

ASPX-CS

```
@*Add the following scripts in your view page for AngularJS support*@
<script
src="http://cdn.syncfusion.com/js/assets/external/angular.min.js"></script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.unobtrusive.min.js">
</script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/ej.widget.angular.min.js">
</script>
<div class="content-container-fluid" ng-app="PickerCtrl">
<div class="row" style="width: 100%" ng-controller="ColorPickerCtrl">
<div class="cols-sample-area" style="width: 100%">
<div class="frame">
<div id="control">
<div class="element" style="margin-left: 45px;">
<input id="picker" ej-colorpicker e-value="colorValue" e-modeltype="palette"
/>
</div>
<div class="element" style="margin-left: 234px">
<input id="custom" ej-colorpicker e-value="colorValue" e-modeltype="picker"
/>
</div>
<h6><span style="font-style: italic; font-weight: normal; position:
absolute; margin-top: 5px; margin-left: 45px;">Note:Two Way AngularJS
Support</span></h6>
</div>
</div>
</div>
<script>
angular.module('PickerCtrl', ['ejangular'])
.controller('ColorPickerCtrl', function ($scope) {
$scope.colorValue = "#278787";
});
</script>
</div>
</div>
```

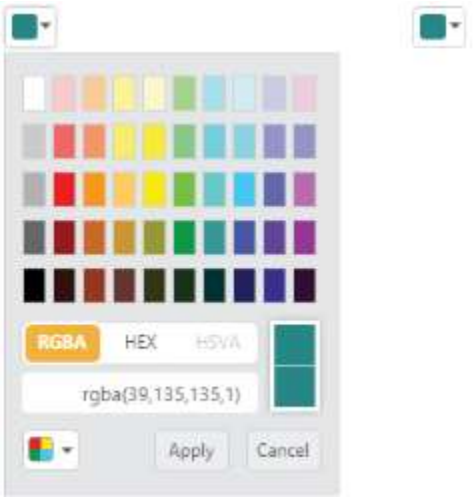
CSS

```
<style>
.element {
display: inline-block;
}
.frame {
width: 457px;
border: 0px;
}
#control {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
width: 600px;
}
</style>
```

The following screenshot displays the output of the above code example.



ColorPicker with AngularJS Support

Keyboard Interaction

You can use keyboard shortcut keys as an alternative to the mouse while using the ColorPicker widget. The ColorPicker widget allows you to perform all kinds of actions using keyboard shortcuts.

Keyboard shortcut keys

Shortcut Key	Description
Alt + j	Focuses into the ColorPicker control
Alt + Down	Open / Close the Popup
Up	Increase the brightness value
Down	Decrease the brightness value
Right	Increase the saturation value
Left	Decrease the saturation value
Enter	Choose the current color
Esc	Closes the popup
Tab	Choose the next element
Home	Downwards to value 0
End	Upwards to value 100

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Keyboard Interaction

The following steps explain how you can enable keyboard interaction for ColorPicker textbox.

1. In the CSHTML page, configure the ColorPicker widget and enable keyboard interaction by the access key property.

ASPX-CS

```
@*In the CSHTML page, add the HTMl helpers to render ColorPicker widget and
enable keyboard interaction by the access key property*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787")
```

JS

```
<script>
jQuery(function ($) {
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#colorPickerWrapper").focus();
}
});
});
</script>
```

The following screenshot displays the output of the above code example.



ColorPicker with Keyboard Navigation

Miscellaneous

getValue

The getValue() method in ColorPicker returns the hexadecimal value.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787")
```

JS

```
<script>
var ColorObj;
jQuery(function ($) {
ColorObj = $("#colorPicker").ejColorPicker({ value: "#278787"
}).data('ejColorPicker');
ColorObj.getValue();
});
</script>
```

setValue

The setValue() method in ColorPicker is used to set the color value. The given value is in hexadecimal form.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker")
```

JS

```
<script>
var ColorObj;
jQuery(function ($) {
ColorObj = $("#colorPicker").ejColorPicker().data('ejColorPicker');
ColorObj.setValue("#278787");
});
</script>
```

getColor

The getColor() method in ColorPicker control returns the color value in r,g,b,a form.

1. In the CSHTML page, configure the ColorPicker widget as follows.

ASPX-CS

```
@*In the CSHTML page, add the HTML helpers to render ColorPicker widget*@
@Html.EJ().ColorPicker("colorPicker").Value("#278787")
```

JS

```
<script>
var ColorObj;
jQuery(function ($) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ColorObj = $("#colorPicker").ejColorPicker({value: "#278787"
}).data('ejColorPicker');
ColorObj.getColor();
});
</script>
```

How to?

Render ColorPicker from Code behind

ColorPicker can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of ColorPicker properties in the controller.

C#

```
//Namespace to get the JavaScript (ColorPicker) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class ColorPickerController : Controller
{
// GET: ColorPicker
public ActionResult ColorPickerFeatures()
{
//Initializing the ColorPicker model
ColorPickerProperties ColorObj = new ColorPickerProperties();
//Initializing the datasource and other properties
ColorObj.ShowPreview= true;
ColorObj.Value = "#278787";
//Passing ColorPicker properties using the ViewData
ViewData["ColorModel"] = ColorObj;
return View();
}
}
}
```

Binding the ColorPicker properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().ColorPicker("searchCustomer",(Syncfusion.JavaScript.Models.ColorPi
ckerProperties)ViewData["ColorModel"]).Render();
}
```

ComboBox

Overview

The ComboBox component allows the user to type a value or choose an option from the list of predefined options.

When an arrow icon accompanied with this component is pressed, the dropdown displays a list of values, from which the user can select one.

Copyright © 2001 - 2020 Syncfusion Inc.

Key features

- **Data binding - Allows binding and accessing the list of items from local or server-side data source.**
- **[Custom values](https://help.syncfusion.com/aspnetmvc/combobox/getting-started#custom values)** - Allows setting user-defined values that is not in the popup list.
- **Grouping - Supports grouping of logically related items under a single or specific category.**
- **Sorting - Supports sorting of list items in an alphabetical order (either ascending or descending).**
- **Filtering - Allows filtering of list items based on a character typed in the component.**
- **Templates - Allows customizing the list items, selected value, header, footer, category group header, and no records content.**
- **Accessibility - Provided with built-in accessibility support that helps to access all the ComboBox component features through the keyboard, screen readers, or other assistive technology devices.**

Getting Started

This section explains how to create a simple **ComboBox** component and configure its available functionalities.

1.Create an MVC Project and add necessary assemblies, scripts, and CSS files given in MVC-Getting Started Documentation. 2.Add ComboBox control using the helper from EJ namespace.

Initialize the ComboBox

The ComboBox can be initialized as:

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("select")
.Width("100%")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.ComboBoxFields(f=>f.Text("text"))
.Placeholder("Select")
.Render();
}
</div>
</div>
```

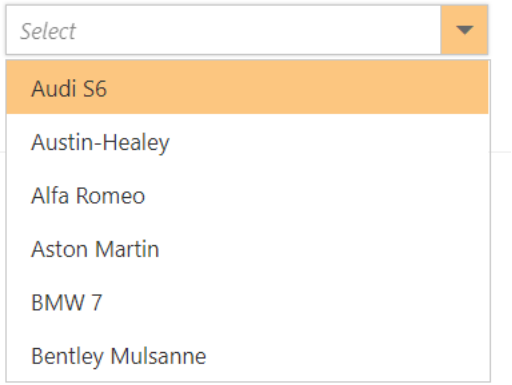
C#

```
public class CarsList
{
public string text { get; set; }
public static List<CarsList> GetCarsList()
{
List<CarsList> car = new List<CarsList>();
car.Add(new CarsList { text = "Audi S6" });
car.Add(new CarsList { text = "Austin-Healey" });
car.Add(new CarsList { text = "Alfa Romeo" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
car.Add(new CarsList { text = "Aston Martin" });
car.Add(new CarsList { text = "BMW 7" });
car.Add(new CarsList { text = "Bentley Mulsanne" });
return car;
}
public ActionResult Default()
{
ViewBag.datasource = GetCarsList();
return View();
}
}
```

Output for ComboBox control is as follows.



Binding data source

After initializing, populate the ComboBox with data using the **DataSource** property. Here, an array of string values is passed to the ComboBox component.

HTML

```
<div class="frame">
<div class="control">
@{
string[] datasource = new string[] { "Badminton", "Cricket", "Football",
"Golf", "Tennis" };
Html.EJ()
.ComboBox("select")
.Width("100%")
.Datasource(datasource)
.Placeholder("Select a game")
.Render();
}
</div>
</div>
```

Output for ComboBox control is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Custom values

The ComboBox allows the user to give input as custom value which is not required to be present in the predefined set of values. By default, this support is enabled by the **AllowCustom** property. In this case, both text field and value field are considered as same. The custom value will be sent to post back handler when a form is about to be submitted.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("select")
.Width("100%")
.Datasource((IEnumerable<Flowers>)ViewBag.datasource)
.AllowCustom(true)
.ComboBoxFields(f=>f.Text("text").Value("text"))
.Placeholder("Select")
.Render();
}
</div>
</div>
```

C#

```
public class CarsList
{
public string text { get; set; }
public static List<CarsList> GetCarsList()
{
List<CarsList> car = new List<CarsList>();
car.Add(new CarsList { text = "Audi S6" });
car.Add(new CarsList { text = "Austin-Healey" });
car.Add(new CarsList { text = "Alfa Romeo" });
car.Add(new CarsList { text = "Aston Martin" });
car.Add(new CarsList { text = "BMW 7" });
car.Add(new CarsList { text = "Bentley Mulsanne" });
return car;
}
public ActionResult Default()
{
ViewBag.datasource = GetCarsList();
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
```

Output for allowCustom ComboBox control is as follows.



Configure the popup list

By default, the width of the popup list automatically adjusts according to the ComboBox input element's width, and the height of the popup list is '300px'.

The height and width of the popup list can also be customized by using the **popupHeight** and **popupWidth** properties respectively.

In the following sample, popup list's width and height are configured.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("select")
.Width("100%")
.Datasource((IEnumerable<CarsList>)ViewBag.datasource)
.PopupHeight("200px")
.PopupWidth("200px")
.Placeholder("Select")
.Render();
}
</div>
</div>
```

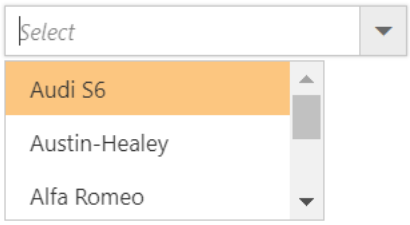
C#

```
public class CarsList
{
public string text { get; set; }
public static List<CarsList> GetCarsList()
{
List<CarsList> car = new List<CarsList>();
car.Add(new CarsList { text = "Audi S6" });
car.Add(new CarsList { text = "Austin-Healey" });
car.Add(new CarsList { text = "Alfa Romeo" });
car.Add(new CarsList { text = "Aston Martin" });
car.Add(new CarsList { text = "BMW 7" });
car.Add(new CarsList { text = "Bentley Mulsanne" });
return car;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public ActionResult Default()
{
ViewBag.datasource = GetCarsList();
return View();
}
}
```

Output for ComboBox control is as follows.



ComboBox Dependencies

The external script dependencies of the ComboBox control are as follows:

- jQuery 1.7.1 and later versions.

The internal script dependency ej.web.all.js is a bundle of all Syncfusion ASP.NET MVC controls. If you use ej.web.all.js in your application, you can leave this section, or else try to render ComboBox in your application using the ej.combobox.min.js file. Refer to the following frameworks and controls in your project.

FileÂ	Description / Usage
ej.core.min.js	Must be referred always before using all the JS controls.
ej.data.min.js	Used to handle data operation and should be used while binding data to JS controls.
ej.ComboBox.min.js	The ComboBoxâ€™s main file
ej.globalize.min.js	Should be referred when localization used in ComboBox.

Data Binding

The ComboBox loads the data either from local data sources or remote data services using the `DataSource` property. It supports the data type of `array` or **DataManager**.

The ComboBox also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

| Fields | Type | Description |

|------|------|-------------|

| text | `string` | Specifies the display text of each list item. |

| value | string | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| groupBy | string | Specifies the category under which the list item has to be grouped. |

| iconCss | string | Specifies the icon class of each list item. |

When binding complex data to the ComboBox, the fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of simple data

The ComboBox has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

HTML

```
<div class="frame">
<div class="control">
@{
string[] datasource = new string[] { "Badminton", "Cricket", "Football",
"Golf", "Tennis" };
Html.EJ()
.ComboBox("ComboBox")
.Width("100%")
.Datasource(datasource)
.Placeholder("Select a game")
.Render();
}
</div>
</div>
```

Output for array-data ComboBox control is as follows.



2. Array of JSON data

The ComboBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the **ComboBoxFields** property.

In the following example, id column and text column from complex data have been mapped to the value field and text field, respectively.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("ComboBoxId")
.Width("100%")
.Datasource((IEnumerable<Flowers>)ViewBag.datasource)
.AllowCustom(true)
.ComboBoxFields(f=>f.Text("text").Value("id"))
.Placeholder("Select a flower")
.Render();
}
</div>
</div>
```

C#

```
public class Flowers
{
public string text { get; set; }
public int id { get; set; }
public static List<Flowers> GetFlowers()
{
List<Flowers> flower = new List<Flowers>();
flower.Add(new Flowers { text = "Anemone Galilee", id=0 });
flower.Add(new Flowers { text = "Allium drumstick", id=1 });
flower.Add(new Flowers { text = "Artichoke thistle" id=2 });
flower.Add(new Flowers { text = "Boronia" id=3 });
flower.Add(new Flowers { text = "Bouvardia" id=4 });
flower.Add(new Flowers { text = "Blue lace flower" id=5 });
flower.Add(new Flowers { text = "Bird of paradise" id=6 });
flower.Add(new Flowers { text = "Cone flower" id=7 });
flower.Add(new Flowers { text = "Cosmos" id=8 });
flower.Add(new Flowers { text = "Calla lily white" id=9 });
flower.Add(new Flowers { text = "Common Yarrow" id=10 });
return flower;
}
public ActionResult Default()
{
ViewBag.datasource = GetFlowers();
return View();
}
}
```

Output for remote-data ComboBox control is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.

Binding remote data

The ComboBox supports retrieval of data from remote data services with the help of **DataManager** component. The **Query** property is used to fetch data from the database and bind it to the ComboBox.

In the following sample, displayed first 6 contacts from the customer table of Northwind Data Service.

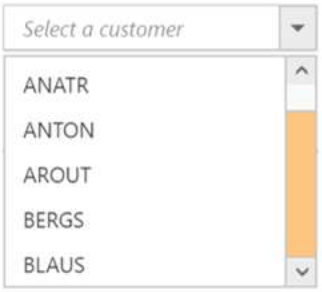
HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("searchCustomer")
.Datasource(d =>
d.URL("//js.syncfusion.com/ejServices/wcf/NorthWind.svc/").Offline(false).Cr
ossDomain(true))
.Query("ej.Query().from('Suppliers').select('SupplierID', 'ContactName')")
.ComboBoxFields(f => f.Text("ContactName").Value("SupplierID"))
.Width("100%")
.Placeholder("Search a customer")
.Render();
}
</div>
</div>
```

C#

```
public ActionResult Databindingremote()
{
return View();
}
```

Output for remote-data ComboBox control is as follows.



SQL Data Binding

Fetch the Data from SQL data source and initialize the control with DataSource property. Specify the column names in the Fields property.

HTML

```
@Html.EJ()ComboBox("selectEmployee").Datasource((IEnumerable<Object>)ViewBag
.Data)ComboBoxFields(df =>
df.ID("Id").Text("Name").Value("Designation")).Width("150px")
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public ActionResult DropdownlistFeatures()
{
List<Employee> data = new List<Employee>();
SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\SampleDB.mdf;
Integrated Security=True");
using (con)
{
con.Open();
SqlDataReader myReader = null;
SqlCommand myCommand = new SqlCommand("select * From [Table]",
con);
myReader = myCommand.ExecuteReader();
if (myReader.Read())
{
data.Add(new Employee(myReader["Name"].ToString(),
myReader["Designation"].ToString()));
}
con.Close();
ViewBag.Data = data;
}
return View();
}
public class Employee
{
public string Name { get; set; }
public string Designation { get; set; }
public Employee(string name, string deg)
{
this.Name = name;
this.Designation = deg;
}
}
```

LINQ Data Binding

Language-Integrated Query (LINQ) is not only about retrieving data. It is also a powerful tool for transforming data. By using a LINQ query, you can use a source sequence as input, and modify it in many ways to create a new output sequence.

Defines model class

C#

```
public class Employee
{
public string name { set; get; }
public string employeeID { set; get; }
public string designation { set; get; }
public Employee() { }
public Employee(string name, string id, string designation)
{
name = name;
employeeID = id;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
designation = designation;
}
}
```

In code behind, fetch the data from Employee class and assigned to the ComboBox are as follows:

C#

```
public ActionResult ComboBoxFeatures()
{
List<Employee> data = new List<Employee>() { };
data.Add(new Employee("Nancy", "E11011", "Technical Writer"));
data.Add(new Employee("Angel", "E11012", "Professor"));
data.Add(new Employee("Daniah", "E11013", "Dancer"));
data.Add(new Employee("Jenifer", "E11014", "Beautician"));
data.Add(new Employee("Prince", "E11015", "Developer"));
var db = from result in data
select result;
ViewBag.dataSource = db.ToList();
return View();
}
```

HTML

```
@Html.EJ().ComboBox("select").Datasource((IEnumerable<Employee>)ViewBag.data
source)ComboBoxFields(df =>
df.ID("employeeID").Text("name").Value("designation")).Placeholder("Select a
Student").Width("100%")
```

XmlDataSource

The XmlDataSource is used to work with XML documents. The following steps explain the details about the data binding from XmlDataSource.

Step 1: Create an XML Data source as follows:

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<Students>
<Student>
<Name>Abhishek</Name>
<Location>Dhanbad</Location>
</Student>
<Student>
<Name>Aman</Name>
<Location>Samastipur</Location>
</Student>
<Student>
<Name>Vicky</Name>
<Location>Munger</Location>
</Student>
<Student>
<Name>Chandan</Name>
<Location>Bhagalpur</Location>
</Student>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<Student>
<Name>Ravi</Name>
<Location>Dhanbad</Location>
</Student>
</Students>
```

Step 2: Read data from the XML file.

C#

```
using System.Xml;
using DataBinding.Models;
public ActionResult ComboBoxFeatures()
{
List<Product> data = new List<Product>();
using (XmlReader reader =
XmlReader.Create("D:/Sample/MVC/DataBinding/DataBinding/App_Data/XMLData.xml
"))
{
while (reader.Read())
{
if (reader.IsStartElement())
{
switch (reader.Name.ToString())
{
case "Name":
data.Add(new Product { name = reader.ReadString() });
break;
}
}
}
}
ViewBag.data = data;
return View();
}
```

Step 3: Assign the data to the ComboBox's DataSource property. Specify the column names in the ComboBoxFields property.

RAZOR

```
@using DataBinding.Models
@Html.EJ().ComboBox("selectCar").Datasource((IEnumerable<Product>)ViewBag.da
ta).ComboBoxFields(df => df.Text("name").Value("name")).Placeholder("Select
a Student").Width("100%")
```

EntityDataSource

Bind data to the ComboBox through Entity Framework. Please follow the below steps.

1. Create an entity data model

Please refer to the link for more information.

2. Refer the entity data model to your project and bind the data to the ComboBox.

Copyright © 2001 - 2020 Syncfusion Inc.

In code behind, create an object for the Entity mode (NORTHWNDEntities), and fetch the data from the Employees class and store it in the List.

Assign the data to the ComboBox's DataSource property. Specify the column names in the fields property.

C#

```
public ActionResult ComboBoxFeatures()
{
NORTHWNDEntities sample = new NORTHWNDEntities();
var dataset = sample.Employees
.Select(x => new Employee
{
EmployeeID = x.EmployeeID,
FirstName = x.FirstName
}).ToList();
ViewBag.DataSource = dataset;
}
public class Employee
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
}
```

RAZOR

```
@Html.EJ().ComboBox("selectEmployee").Datasource((IEnumerable<Object>)ViewBa
g. DataSource).ComboBoxFields(df => df.Text("FirstName ").Value("EmployeeID
")).Width("150px")
```

AccessDataSource

The data can also be bound to the ComboBox using OLEDB database as depicted below.

RAZOR

```
Html.EJ().ComboBox("selectEmployee").Datasource((IEnumerable<Object>)ViewBag
. DataSource).ComboBoxFields(df =>
df.ID("Id").Text("Name").Value("Designation")).Width("150px")
```

The server-side code to retrieve and bind the data to ComboBox are as follows. Also, define a class with all the required fields as depicted in the below code example.

C#

```
public ActionResult ComboBoxFeatures()
{
OleDbConnection conn = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|\Database.mdb");
conn.Open();
OleDbCommand cmd = new OleDbCommand(" SELECT * from [Employees]", conn);
OleDbDataReader reader = null;
List<EmployeeDetails> data = new List<EmployeeDetails>();
reader = cmd.ExecuteReader();
while (reader.Read())
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
data.Add(new EmployeeDetails(reader["Name"].ToString(),
reader["Designation"].ToString()));
}
conn.Close();
ViewBag. DataSource = data;
}
public class EmployeeDetails
{
public string Name { get; set; }
public string Designation { get; set; }
public EmployeeDetails(string name, string deg)
{
this.Name = name;
this.Designation = deg;
}
}
```

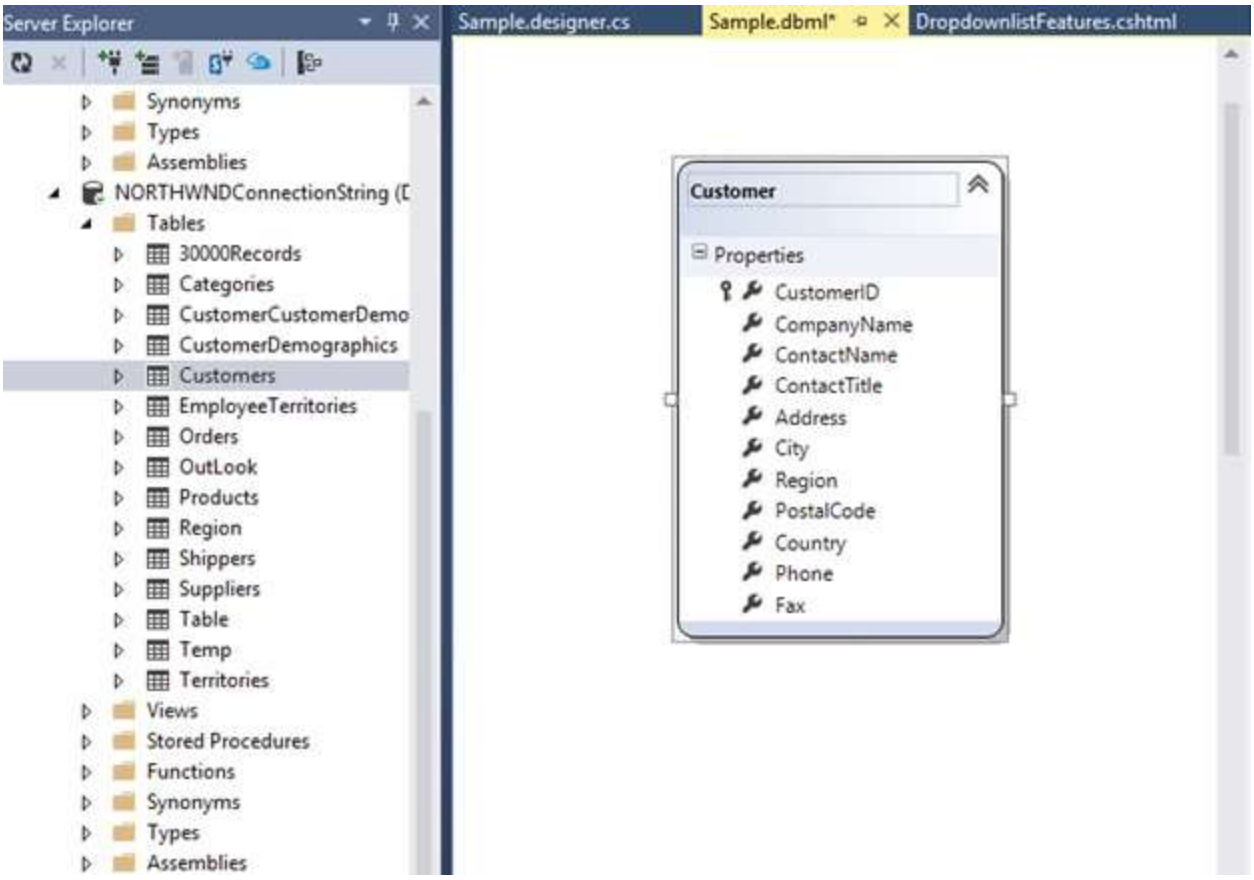
LINQ to SQL Data Binding

The LINQ to SQL can be used as the data source of the ComboBox in that the data model of a relational database is mapped to an object model and allow us to use the LINQ technology to access SQL database.

To create LINQ to SQL classes

- Right-click the App_Code folder and then click Add New Item.
- Under Installed templates, select LINQ to SQL Classes template, type a name for the .dbml file, and then click Add.
- The Object Relational Designer window is displayed.
- In Server Explorer, drag the database table into the Object Relational Designer window.
- The table and its columns are represented as an entity in the designer window.

Copyright © 2001 - 2020 Syncfusion Inc.



In code behind, fetch the data from classes and stored it to the ViewBag as follows

C#

```
ViewBag.dataSource = new SampleDataContext().Customers.ToList();
```

HTML

```
@Html.EJ().ComboBox("customers").Datasource((IEnumerable<Customer>)ViewBag.d
ataSource).ComboBoxFields(df =>
df.Text("CompanyName").Value("CustomerID")).Placeholder("Select a
Customer").Width("100%")
```



Copyright © 2001 - 2020 Syncfusion Inc.

Remote data

To bind remote data to the ComboBox, you can assign a service data as an instance of DataManager to the Datasource property.

OData

OData is a standardized protocol for creating and consuming data. You can provide the OData service URL directly to the Datasource URL property.

HTML

```
@Html.EJ().ComboBox("ComboBoxOData").Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders")).Co
mboBoxFields(df=>df.Text("ShipCountry").Value("OrderID"))
```

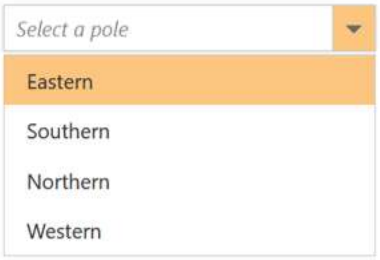
OData Version 4

For OData Version 4 support ODataV4Adaptor should be used. By using URL property of Datasource, you can bind OData Version 4 Service link and specify Adaptor value as enum AdaptorType.ODataV4Adaptor.

For further details about OData service refer to the link.

HTML

```
@Html.EJ().ComboBox("ComboBox1").Datasource(dataSource
=>dataSource.URL("http://services.odata.org/V4/Northwind/Northwind.svc/Regio
ns/").Adaptor(AdaptorType.ODataV4Adaptor)).ComboBoxFields(df =>
df.Text("RegionDescription").Value("RegionID"))
```



Note: Events associated with remote data bind is listed here.

WebAPI

Using WebApiAdaptor, you can bind WebApi service's data to ComboBox. The data from WebApi service must be returned as an object that has the property "Items" with its value as data source and another property "Count" with its value as dataSource's total records count.

HTML

```
@Html.EJ().ComboBox("ComboBox1").Datasource(dataSource =>
dataSource.URL("/api/Orders").Adaptor(AdaptorType.WebApiAdaptor)).ComboBoxFi
elds(df => df.Text("Name").Value("EmployeeID"))
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class OrdersController : ApiController
{
NorthwindDataContext db = new NorthwindDataContext();
// GET API/<controller>
public PageResult<EmployeePhoto> Get(ODataQueryOptions opts)
{
List<EmployeePhoto> photos = db.EmployeePhotos.ToList();
return new PageResult<EmployeePhoto>(photos as IEnumerable<EmployeePhoto>,
null, photos.Count);
}
}
```



Templates

The ComboBox has been provided with several options to customize each list item, group title, selected value, header, and footer elements.

Item template

The content of each list item within the ComboBox can be customized with the help of **ItemTemplate** property.

In the following sample, each list item is split into two columns to display the relevant data.

HTML

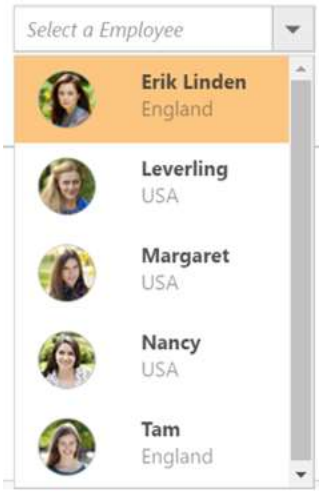
```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.Width("100%")
.Datasource((IEnumerable<empList>)ViewBag.datasource)
.ComboBoxFields(h=>h.Text("text"))
.HeaderTemplate("<div class=\"head\">  Photo  <span style=\"padding-
left:42px\"> Contact Info </span></div>")
.ItemTemplate("<div><img class=\"eimg\"
src=\"../Images/combobox/${eimg}.png\" alt=\"employee\"/><div
class=\"ename\"> ${text} </div><div class=\"temp\"> ${country}
</div></div>")
.FooterTemplate("<div class=\"Foot\"> Total Items Count: 5 </div>")
.Placeholder("Select a employee")
.Render();
}
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

CSS

```
<style>
.ename {
font-weight: bold;
display: block !important;
opacity: .87;
}
.tempName {
padding: 5px 42px;
opacity: .87;
}
.temp {
margin-top: -15px;
opacity: .54;
}
.eimg {
border-radius: 50%;
padding: 10px 16px;
width: 40px;
height: 40px;
float: left;
}
.tempImg {
padding-bottom: 3px;
border-radius: 50%;
float: left;
}
.e-dropdownbase .e-list-item * {
display: block;
}
</style>
```

Output for item template ComboBox control is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.

Group template

The group header title with appropriate sub-items are categorized that is customized with the help of **GroupTemplate** property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

{% highlight html %}

@{

Html.EJ()

.ComboBox("selectCountry")

.Width("100%")

.Datasource((IEnumerable<empList>)ViewBag.datasource)

.ComboBoxFields(h=>h.Text("text").GroupBy("country"))

.ItemTemplate(" <div class=\"ename\"> ${text} <div class=\"temp\"> ${country} ")

.GroupTemplate("${country}")

.Placeholder("Select a country")

.Render();

}

{% endhighlight %}

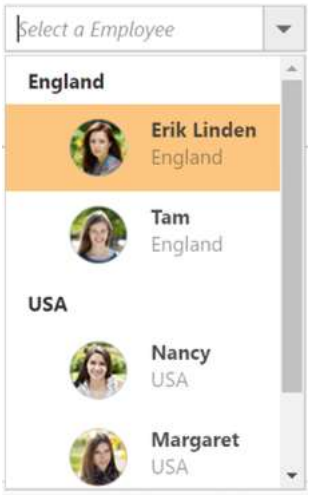
CSS

```
<style>
.ename {
font-weight: bold;
display: block !important;
opacity: .87;
}
.tempName {
padding: 5px 42px;
opacity: .87;
}
.temp {
margin-top: -15px;
opacity: .54;
}
.eimg {
border-radius: 50%;
padding: 10px 16px;
width: 40px;
height: 40px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
float: left;
}
.tempImg {
padding-bottom: 3px;
border-radius: 50%;
float: left;
}
.e-dropdownbase .e-list-item * {
display: block;
}
</style>
```

Output for group template ComboBox control is as follows.



Header template

The header element is shown statically at the top of the popup list items within the ComboBox, and any custom element can be placed as a header element using the **HeaderTemplate** property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.Width("100%")
.Datasource((IEnumerable<empList>)ViewBag.datasource)
.ComboBoxFields(h=>h.Text("text"))
.ItemTemplate("<div><img class=\"eimg\"
src=\"../Images/combobox/${eimg}.png\" alt=\"employee\"/><div
class=\"ename\"> ${text} </div><div class=\"temp\"> ${country}
</div></div>")
.HeaderTemplate("<div class=\"head\">  Photo  <span style=\"padding-
left:42px\"> Contact Info </span></div>")
.FooterTemplate("<div class=\"Foot\"> Total Items Count: 5 </div>")
.Placeholder("Select a country")
```

Copyright © 2001 - 2020 Syncfusion Inc.

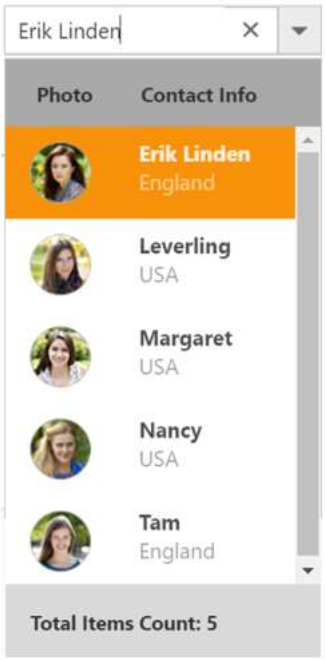
```
.Render();
}
</div>
</div>
```

CSS

```
<style>
.head {
background-color: #a9a9a9;
height: 30px;
font-weight: bold;
padding: 14px 0 0 20px;
}
.ename {
font-weight: bold;
display: block !important;
opacity: .87;
}
.tempName {
padding: 5px 42px;
opacity: .87;
}
.temp {
margin-top: -15px;
opacity: .54;
}
.eimg {
border-radius: 50%;
padding: 10px 16px;
width: 40px;
height: 40px;
float: left;
}
.tempImg {
padding-bottom: 3px;
border-radius: 50%;
float: left;
}
.e-dropdownbase .e-list-item * {
display: block;
}
</style>
```

Output for item template ComboBox control is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Footer template

The ComboBox has an option to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the **FooterTemplate** property.

In the following sample, footer element displays the total number of list items present in the ComboBox.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.Width("100%")
.Datasource((IEnumerable<empList>)ViewBag.datasource)
.ComboBoxFields(h=>h.Text("text"))
.ItemTemplate("<div><img class=\"eimg\"
src=\"../Images/combobox/${eimg}.png\" alt=\"employee\"/><div
class=\"ename\"> ${text} </div><div class=\"temp\"> ${country}
</div></div>")
.FooterTemplate("<div class=\"Foot\"> Total Items Count: 5 </div>")
.Placeholder("Select a employee")
.Render();
}
</div>
</div>
```

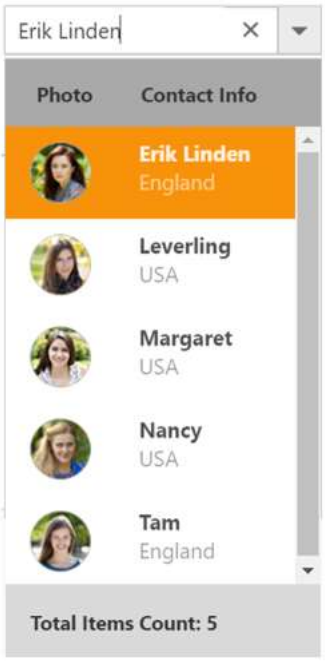
CSS

```
<style>
.Foot {
background-color: #dadada;
vertical-align: middle;
padding: 16px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
font-weight: bold;
}
.ename {
font-weight: bold;
display: block !important;
opacity: .87;
}
.tempName {
padding: 5px 42px;
opacity: .87;
}
.temp {
margin-top: -15px;
opacity: .54;
}
.eimg {
border-radius: 50%;
padding: 10px 16px;
width: 40px;
height: 40px;
float: left;
}
.tempImg {
padding-bottom: 3px;
border-radius: 50%;
float: left;
}
.e-dropdownbase .e-list-item * {
display: block;
}
</style>
```

Output for footer template ComboBox control is as follows.



No records template

The ComboBox is provided with the support to custom design the popup list content when no data is found and no matches found on search with the help of **NoRecordsTemplate** property.

In the following sample, popup list content displays the notification of no data available.

{% highlight html %}

@{

Html.EJ()

.ComboBox("searchCustomer")

.Datasource(d => d.URL("//js.syncfusion.com/ej/ejServices/wcf/NorthWind.svc/").Offline(false).CrossDomain(true))

.Query("ej.Query().from('Suppliers').select('SupplierID', 'ContactName').take(0)")

.ComboBoxFields(f => f.Text("ContactName").Value("SupplierID"))

.Width("100%")

.NoRecordsTemplate(" NO DATA AVAILABLE")

.Placeholder("Search a customer")

.Render();

}

{% endhighlight %}

C#

```
public ActionResult Databinding()
{
return View();
}
```

Output for no records template ComboBox control is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved by using the **ActionFailureTemplate** property.

In the following sample, when the data fetch request fails, the ComboBox displays the notification.

{% highlight html %}

@{

Html.EJ()

.ComboBox("searchCustomer")

.Datasource(d => d.URL("//js.syncfusion.com/ej/ejServices/wcf/NorthWind.svc/").Offline(false).CrossDomain(true))

.Query("ej.Query().from('Suppliers').select('SupplierID', 'ContactName')")

.ComboBoxFields(f => f.Text("ContactName").Value("SupplierID"))

.Width("100%")

.ActionFailureTemplate("Data fetch get fails")

.Placeholder("Search a customer")

.Render();

}

{% endhighlight %}

C#

```
public ActionResult Databinding()
{
return View();
}
```

Grouping

The ComboBox supports the wrapping of nested elements into a group based on different categories. The category of each list item can be mapped through the **GroupBy** field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using the GroupBy field.

HTML

```
<div class="frame">
<div class="control">
@{
```

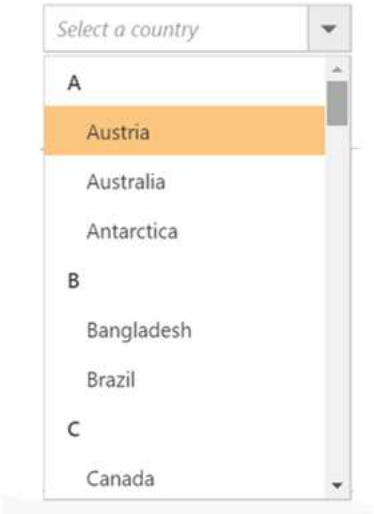
Copyright © 2001 - 2020 Syncfusion Inc.

```
Html.EJ()
.ComboBox("groupingCountry")
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Width("100%")
.ComboBoxFields(fields => fields.GroupBy("category").Text("text"))
.Placeholder("Select a country")
.Render();
}
</div>
</div>
```

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
return country;
}
}
}
```

Output for grouping ComboBox control is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.

Customization

The grouping header is also provided with the customization option. This allows custom designing using the groupTemplate property for both the inline and fixed headers as referred here:

Group Template support to ComboBox

Sorting

The ComboBox has built-in support to sort data items when SortOrder is specified. The available sort orders are ascending, descending and none.

To sort the data in ascending order, set the SortOrder as ascending.

The following sample illustrates how to sort the data.

HTML

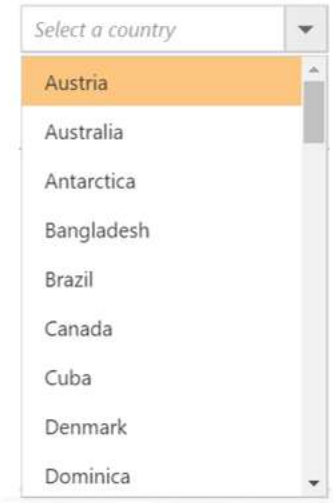
```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("Sorting")
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Width("100%")
.ComboBoxFields(fields => fields.Text("text"))
.Placeholder("Select a country")
.SortOrder(SortOrder.Ascending)
.Render();
}
</div>
</div>
```

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
return country;
}
}
}
```

Output for Sorting is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Filtering

The ComboBox has built-in support to filter data items when AllowFiltering is enabled. The filter operation starts as soon as you start typing characters in the component.

To display filtered items in the popup, filter the required data and return it to the ComboBox via **updateData** method by using the **filtering** event.

The following sample illustrates how to query the data source, and pass the data to the ComboBox through the updateData method in Filtering event.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.AllowFiltering(true)
.ComboBoxFields(h => h.Text("text").Value("category"))
.Query("new ej.Query().select(['text','category']).take(25)")
.ClientSideEvents(h => h.Filtering("filtering"))
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Placeholder("Select a country")
.Render();
}
</div>
</div>
<script type="text/javascript">
function filtering(e) {
var query = new ej.Query().select(['text','category']);
query = (e.text !== '') ? query.where('text', 'startswith', e.text, true) :
query;
e.updateData(e.model.dataSource, query);
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
...
return country;
}
}
}
```

Output for filtering ComboBox control is as follows.



Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the ComboBox. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.AllowFiltering(true)
.ComboBoxFields(h => h.Text("text").Value("category"))
.Query("new ej.Query().select(['text','category']).take(25)")
.ClientSideEvents(h => h.Filtering("filtering"))
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Placeholder("Select a country")
.Render();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
</div>
<script type="text/javascript">
function filtering(e) {
if(e.text == '') e.updateData(e.model.dataSource);
else{
// restrict the remote request until search key contains 3 characters.
if (e.text.length < 3) { return; }
var query = new Query().select(['text', 'category']);
query = (e.text !== '') ? query.where('text', 'startswith', e.text, true) :
query;
e.updateData(e.model.dataSource, query);
}
}
</script>
```

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
...
return country;
}
}
}
```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with the `endsWith` type.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.AllowFiltering(true)
.ComboBoxFields(h => h.Text("text").Value("category"))
.Query("new ej.Query().select(['text','category']).take(25)")
.ClientSideEvents(h => h.Filtering("filtering"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

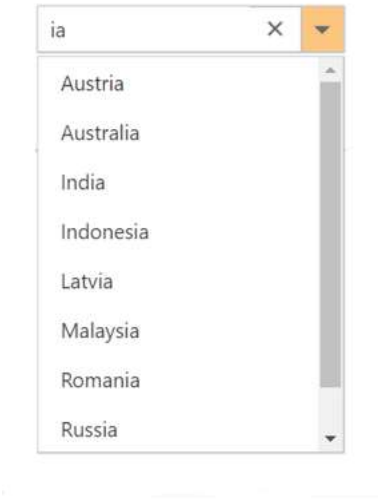
```
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Placeholder("Select a country")
.Render();
}
</div>
</div>
<script type="text/javascript">
function filtering(e) {
var query = new ej.Query().select(['text','category']);
query = (e.text !== '') ? query.where('text', 'endswith', e.text, true) :
query;
e.updateData(e.model.dataSource, query);
}
</script>
```

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
...
return country;
}
}
}
```

Output for filtering ComboBox control is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the **where** clause.

The following example shows how to perform case-sensitive filter.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("selectCountry")
.AllowFiltering(true)
.ComboBoxFields(h => h.Text("text").Value("category"))
.Query("new ej.Query().select(['text','category']).take(25)")
.ClientSideEvents(h => h.Filtering("filtering"))
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Placeholder("Select a country")
.Render();
}
</div>
</div>
<script type="text/javascript">
function filtering(e) {
var query = new ej.Query().select(['text','category']);
query = (e.text !== '') ? query.where('text', 'startswith', e.text, false) :
query;
e.updateData(e.model.dataSource, query);
}
</script>
```

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
...
return country;
}
}
}
```

Output for filtering ComboBox control is as follows.



Localization

The Localization library allows you to localize the static text content of the **noRecordsTemplate** and **actionFailureTemplate** properties according to the culture currently assigned to the ComboBox.

| Locale key | en-US (default) |

|------|------|-------------|

| noRecordsTemplate | No records found |

| actionFailureTemplate | The request failed |

Loading translations

To load translation object to your application, use the `load` function of **L10n** class.

In the following sample, French culture is set to the ComboBox and no data is loaded. Hence, the `noRecordsTemplate` property displays its text in French culture initially, and if the sample is run offline, the `actionFailureTemplate` property displays its text appropriately.

HTML

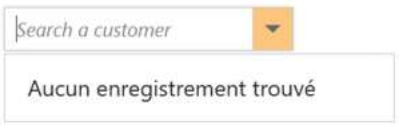
```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("searchCustomer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Datasource(d =>
d.URL("//js.syncfusion.com/ejServices/wcf/NorthWind.svc/").Offline(false).Cr
ossDomain(true))
.Query("ej.Query().from('Suppliers').select('SupplierID',
'ContactName').take(0)")
.ComboBoxFields(f => f.Text("ContactName").Value("SupplierID"))
.Width("100%")
.Locale("fr-BE")
.Placeholder("Sélectionnez un customer")
.Render();
}
</div>
</div>
<script>
ej.ComboBox.locale["fr-BE"]={
'noRecordsTemplate': "Aucun enregistrement trouvé",
'actionFailureTemplate': "Modèle d'échec d'action"
}
</script>
```

C#

```
public ActionResult Databindingremote()
{
return View();
}
```



Accessibility

The ComboBox component has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties along with keyboard support. This component is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who uses assistive technologies (AT) or those who completely rely on keyboard navigation.

ARIA attributes

The ComboBox component uses the combobox role, and each list item has an option role. The following

ARIA attributes denote the ComboBox state.

| **Properties** | **Functionalities** |

| --- | --- |

| aria-haspopup | Indicates whether the ComboBox input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list is expanded or not. |

Copyright © 2001 - 2020 Syncfusion Inc.

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the ComboBox element. |

| aria-disabled | Indicates whether the ComboBox component is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

| aria-autocomplete | This attribute contains the 'both' to a list of options shows and the currently selected suggestion also shows inline. |

Keyboard interaction

You can use the following key shortcuts to access the ComboBox without interruptions.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Selects the first item in the ComboBox when no item selected. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Page Down | Scrolls down to the next page and selects the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and selects the first item when popup list opens. |

| Enter | Selects the focused item and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the component. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list.|

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | Selects the first item when popup open state. If it is in closed state, cursor moves to before of first character in input |

| End | Selects the last item when popup open state. If it is in closed state, cursor moves to next of last character in input |

In the following sample, focus the ComboBox component using the alt+t keys.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
```

Copyright © 2001 - 2020 Syncfusion Inc.

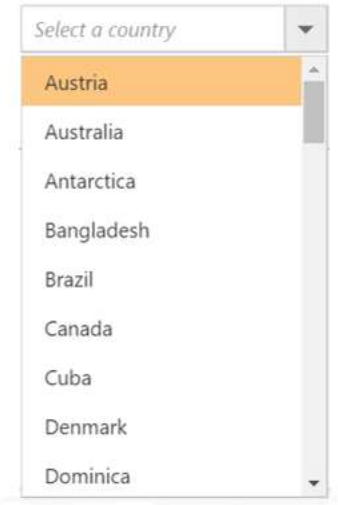
```
.ComboBox("groupingCountry")
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Width("100%")
.ComboBoxFields(fields => fields.Text("text"))
.Placeholder("Select a country")
.Render();
}
</div>
</div>
<script type="text/javascript">
$(function () {
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 84 /* t */) {
// press alt+t to focus the control.
$("#groupingCountry").focus();
}
});
});
</script>
```

C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
return country;
}
}
}
```

Output for accessibility ComboBox control is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



How To

Configure the Cascading ComboBox

The cascading ComboBox is a series of ComboBox, where the value of one ComboBox depends upon the another value. This can be configured by using the change event of the parent ComboBox. Within that change event handler, data has to be loaded to the child ComboBox based on the selected value of the parent ComboBox.

The following example shows the cascade behavior of the country, state, and city

ComboBox. Here, the dataBind method is used to reflect the property changes immediately

to the ComboBox.

HTML

```
<div class="frame">
<div class="row">
<div class="col-xs-8 col-sm-4">
<span class="txt">Select Group</span>
@Html.EJ().ComboBox("groupsList").Datasource((IEnumerable<groups>)ViewBag.da
tasource).ComboBoxFields(f =>
f.Value("parentId").Text("text")).ClientSideEvents(e =>
e.Change("onChange")).Placeholder("Select")
</div>
<div class="col-xs-8 col-sm-4" >
<span class="txt">Select Country</span>
@Html.EJ().ComboBox("countryList").Datasource((IEnumerable<Countries>)ViewBa
g.datasource1).ComboBoxFields(f=>f.Text("text")).Enabled(false)
</div>
</div>
</div>
<script type="text/javascript">
function onChange(e) {
var country = $('#countryList').data("ejComboBox");
if (e.model.value == null) country.option({ enabled: false });
```

```
else country.option({ enabled: true, query: new ej.Query().where('parentId',
'equal', e.model.value), value: null });
}
</script>
```

C#

```
List<groups> group = new List<groups>();
List<Countries> country = new List<Countries>();
public ActionResult Cascading()
{
group.Add(new groups { parentId = "a", text = "Group A" });
group.Add(new groups { parentId = "b", text = "Group B" });
group.Add(new groups { parentId = "c", text = "Group C" });
group.Add(new groups { parentId = "d", text = "Group D" });
group.Add(new groups { parentId = "e", text = "Group E" });
ViewBag.datasource = group;
country.Add(new Countries { value = 11, parentId = "a", text = "Algeria" });
country.Add(new Countries { value = 12, parentId = "a", text = "Armenia"});
country.Add(new Countries { value = 13, parentId = "a", text = "Bangladesh"
});
country.Add(new Countries { value = 14, parentId = "a", text = "Cuba"});
country.Add(new Countries { value = 15, parentId = "b", text = "Denmark"});
country.Add(new Countries { value = 16, parentId = "b", text = "Egypt" });
country.Add(new Countries { value = 17, parentId = "c", text = "Finland" });
country.Add(new Countries { value = 18, parentId = "c", text = "India" });
country.Add(new Countries { value = 19, parentId = "c", text = "Malaysia"
});
country.Add(new Countries { value = 20, parentId = "d", text = "New Zealand"
});
country.Add(new Countries { value = 21, parentId = "d", text = "Norway" });
country.Add(new Countries { value = 22, parentId = "d", text = "Romania" });
country.Add(new Countries { value = 23, parentId = "e", text = "Singapore"
});
country.Add(new Countries { value = 24, parentId = "e", text = "Thailand"
});
country.Add(new Countries { value = 25, parentId = "e", text = "Ukraine"});
ViewBag.datasource1 = country;
return View();
}
}
}
```

Output for ComboBox control is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Show the list items with icons

You can render **icons** to the list items by mapping the

IconCss field. This IconCss field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with IconCss field.

HTML

```
<div class="frame">
<div class="control">
@Html.EJ().ComboBox("maillots").Datasource((IEnumerable<IconCss>)ViewBag.dat
asource).ComboBoxFields(f =>
f.Text("Name").IconCss("IconClass")).Placeholder("Select a
icon").Width("100%")
</div>
</div>
<style>
.mail {
display: block;
background-image: url('../../Images/dropdownlist/iconsapps.png');
height: 25px;
width: 25px;
background-position: center center;
background-repeat: no-repeat;
}
.mail.done {
background-position: 0 0;
}
.mail.moveto {
background-position: 0 -22px;
}
.mail.categorize {
background-position: 0 -46px;
}
.mail.flag {
background-position: 0 -70px;
}
.mail.forward {
background-position: 0 -94px;
}
.mail.new {
background-position: 0 -116px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

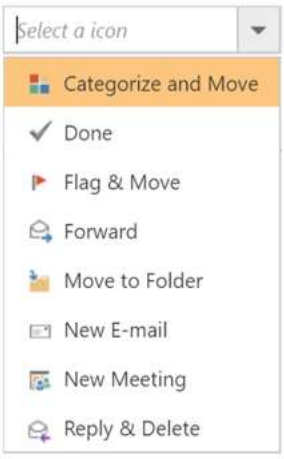
```
}
.mail.reply {
background-position: 0 -140px;
}
.mail.meeting {
background-position: 0 -164px;
}
.control {
margin-left: 20px;
}
.ctrllabel {
padding-bottom: 3px;
}
</style>
```

C#

```
public string Name { get; set; }
public string IconClass { get; set; }
public static List<IconCss> GetIconList()
{
List<IconCss> icon = new List<IconCss>();
icon.Add(new IconCss { Name = "Categorize and Move", IconClass = "mail
categorize" });
icon.Add(new IconCss { IconClass = "mail done", Name = "Done" });
icon.Add(new IconCss { IconClass = "mail flag", Name = "Flag & Move" });
icon.Add(new IconCss { IconClass = "mail forward", Name = "Forward" });
icon.Add(new IconCss { IconClass = "mail moveto", Name = "Move to Folder"
});
icon.Add(new IconCss { IconClass = "mail new", Name = "New E-mail" });
icon.Add(new IconCss { IconClass = "mail meeting", Name = "New Meeting" });
icon.Add(new IconCss { IconClass = "mail reply", Name = "Reply & Delete" });
return icon;
}
public ActionResult Icons()
{
ViewBag.datasource = IconCss.GetIconList();
return View();
}
}
}
```

Output for ComboBox control is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



AutoFill supported with ComboBox

The ComboBox supports the AutoFill behavior with the help of AutoFill property. Whenever you change the input value,

the ComboBox will autocomplete your data by matching the typed character. If no matches found, the ComboBox will not suggest any item.

In the following sample, showcase that how to work autofill with ComboBox.

HTML

```
<div class="frame">
<div class="control">
@{
Html.EJ()
.ComboBox("groupingCountry")
.Datasource((IEnumerable<Countries>)ViewBag.datasource)
.Width("100%")
.ComboBoxFields(fields => fields.Text("text"))
.Placeholder("Select a country")
.AutoFill(true)
.Render();
}
</div>
</div>
```

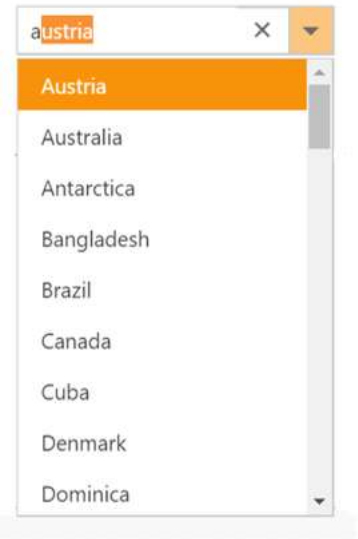
C#

```
public string text { get; set; }
public string category { get; set; }
public static List<Countries> GetCountries()
{
List<Countries> country = new List<Countries>();
country.Add(new Countries { text = "Austria", category = "A" });
country.Add(new Countries { text = "Australia", category = "A" });
country.Add(new Countries { text = "Antarctica", category = "A" });
country.Add(new Countries { text = "Bangladesh", category = "B" });
country.Add(new Countries { text = "Brazil", category = "B" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
country.Add(new Countries { text = "Canada", category = "C" });
country.Add(new Countries { text = "Cuba", category = "C" });
country.Add(new Countries { text = "Denmark", category = "D" });
country.Add(new Countries { text = "Dominica", category = "D" });
return country;
}
}
}
```

Output for grouping ComboBox control is as follows.



Validation of ComboBox using jQuery Validator

Validation of ComboBox can be done on form submission using jQuery Validations by adding name attribute for ComboBox through htmlAttributes property. Also, you can remove this error message during item selection through select or change event of ComboBox

Note: jquery.validate.min script file should be referred for validation, for more details, refer here.

HTML

```
<form id="form1">
<div class="frame">
<div class="row">
<div class="col-xs-8 col-sm-4">
<span class="txt">Select Country</span>
@Html.EJ().ComboBox("countryList").Datasource((IEnumerable<CountryList>)View
Bag.datasource1).ComboBoxFields(h =>
h.Text("text")).AutoFill(true).Placeholder("Select").HtmlAttributes(new
Dictionary<string, object> { { "name", "select" }
}).ClientSideEvents(e=>e.Select("select"))
</div>
<label class="message"></label>
</div>
<button type="submit" id="valid" onclick="validate()"> Validate</button>
</div>
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function validate()
{
var rules = {};
$("form[id$=form1] input[name$=select]").each(function () {
rules[this.name] = "required";
});
$('form[id$="form1"]').validate({
rules: rules,
errorPlacement: function (error, element) {
$(error).insertAfter($(".message"));
}
});
}
function select(args)
{
if(args.value!="")
{
$("label.error").css("display", "none")  //hide error message when value is
selected.
}
}
</script>
</form>
```

Render ComboBox from code behind

ComboBox can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of ComboBox using ComboBox properties from code behind.

RAZOR

```
@{Html.EJ().ComboBox("Customer", Model).Render(); }
```

C#

```
public ActionResult Index()
{
ComboBoxProperties combobox = new ComboBoxProperties();
combobox.DataSource =
"http://js.syncfusion.com/ejServices/wcf/NorthWind.svc/";
combobox.Query = "ej.Query().from('Suppliers').select('SupplierID',
'ContactName').take(10)";
combobox.ComboBoxFields.Text = "ContactName";
combobox.ComboBoxFields.Value = "SupplierID";
combobox.Width = "300";
return View(combobox);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

CurrencyTextBox

Overview

Essential ASP.NET MVC CurrencyTextBox is used to display only currency values. It has Spin buttons to increase or decrease the values in the Text Box.

Key Features

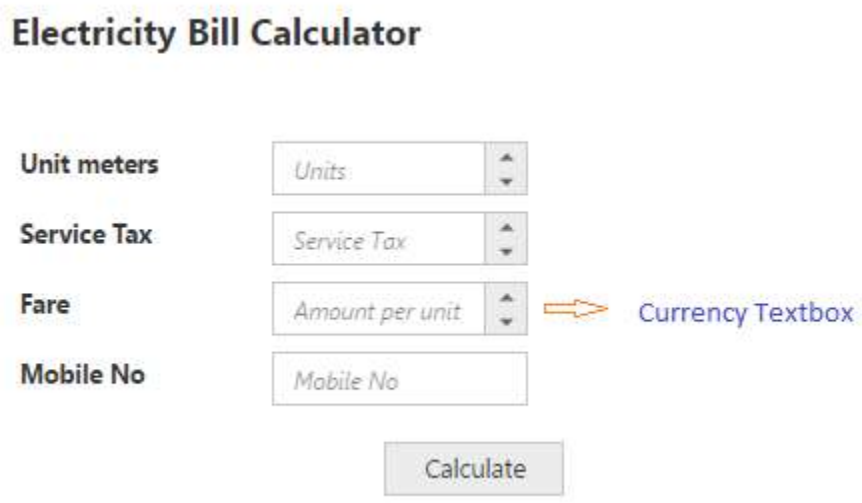
- Min and Max Values—Specifies value range for the textbox.
- Spin Buttons—Enable you to increase or decrease the current value in the textbox.
- Step Value—Enables you to increment or decrement the current value by step value.
- Globalization— EJ MVC CurrencyTextBox has globalization support. It use ej.globalize.js file to globalize the number format, and parse numbers according to the culture.
- Keyboard Navigation—You can interact with CurrencyTextBox by using keyboard.
- RTL Support—Support for right to left alignment of textbox input.
- Decimal Values—You can configure the textbox to accept decimal values.
- Themes—CurrencyTextBox consist of 17 built-in themes , and also support custom skins for creating user-defined themes.

Getting Started

This section explains briefly about how to create a EJMVC CurrencyTextBox in ASP.NET MVC platform.

Create your first CurrencyTextBox in MVC

From the following steps you can learn how to create and use CurrencyTextBox in your application. Here we have showcased, a small Electric bill calculator application using EJMVC Editors widgets.The Essential ASP.NET MVC Editors control includes Numeric, Percentage, Currency and MaskEdit textbox controls. This will guide you to use the wide range of Editors functionalities to complete this application.



Electricity Bill Calculator

Create Editors

ASP.NET MVC Editors renders built-in features like keyboard navigation, min and max range and flexible API's.

Copyright © 2001 - 2020 Syncfusion Inc.

1. Create a MVC Project and add necessary Dll's and Scripts. Refer MVC-Getting Started.
2. Add necessary helper elements to render the Editor components.

HTML

```
<div class="editors">
<div class="element-text" style="">Electricity Bill Calculator</div>
<br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>    @* NumericTextBox creation with watermark text *@
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units")
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>     @* PercentageTextBox creation with watermark text *@
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
<td>    @* CurrencyTextBox creation with watermark text *@
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per unit")
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>    @* MaskEditTextBox creation with watermark text *@
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone number")
</td>
</tr>
</tbody></table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

3. The following styles are added to arrange the Editors.

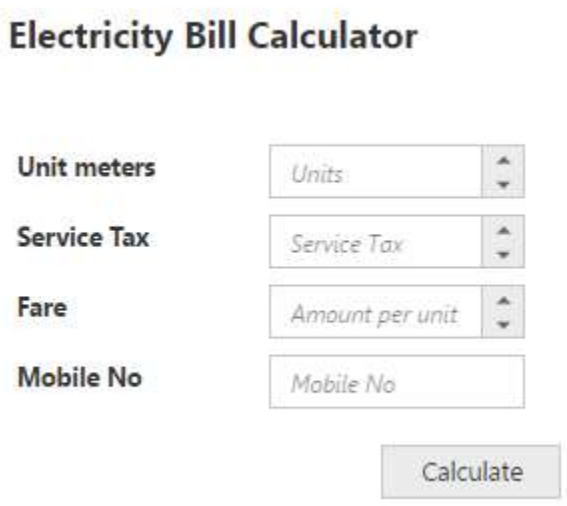
CSS

```
<style type="text/css" class="cssStyles">
.element-text
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
display: inline-block;
font-size: 20px;
font-weight: bolder;
height: 50px;
position: relative;
text-align: center;
top: -20px;
}
.editors
{
max-width: 400px;
border: 2px solid #DDDDDD;
}
.editors table
{
border:0px;
padding-left:50px;
}
.pay-bill
{
margin:15px 0px 10px 208px;
}
.editors label
{
display: block;
width: 130px;
}
</style>
```

4. Execute the code to render Editors as follows



CurrencyTextBox with watermark text

Set MinValue, MaxValue and value in CurrencyTextBox

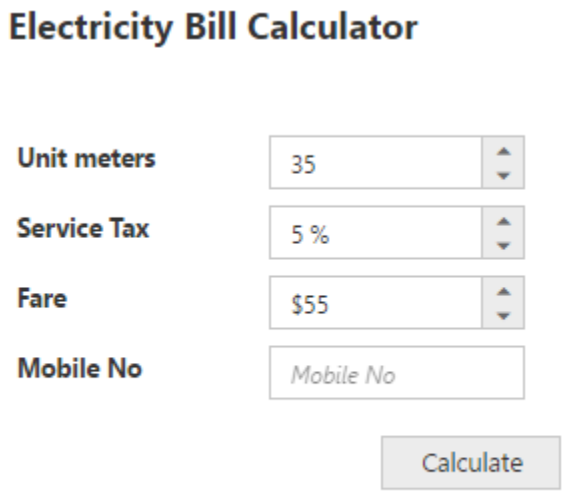
You can set the “MinValue”,“MaxValue” and “Value” in Currency, Percentage and Currency text boxes for maintaining the range in Editors widgets. In this scenario, you have to enter the values between the default ranges and enter the phone number in the MaskEdit widget by using the ”MaskFormat” property. By using DecimalPlaces property for CurrencyTextBox you can get decimal values. The following code example illustrates how to achieve this.

HTML

```
<div class="editors">
<div class="element-text" style="">Electricity Bill Calculator</div>
<br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units").MinValue(1).MaxV
alue(10000).Value("70")
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax").MinValue(5).Max
Value(100).Value("15")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
<td>
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per
unit").MinValue(0.00).MaxValue(100000.00).Value("350.00").DecimalPlaces(2)
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone
number").MaskFormat("99-999-99999").Value("9090909090")
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot illustrates the output of the above code examples.



CurrencyTextBox with Min/Max Value ranges

Setting the Strict Mode Option

You can set the “StrictMode” option to restrict entering values defined outside the range. The following code example illustrates how to set strict mode option.

HTML

```
<div class="editors">
<div class="element-text" style="">ElectricityBill Calculator</div><br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units").MinValue(1).MaxV
alue(10000).Value("70").EnableStrictMode(true)
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax").MinValue(5).Max
Value(100).Value("15")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
```

```
</td>
<td>
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per
unit").MinValue(0.00).MaxValue(100000.00).Value("350.00").DecimalPlaces(2)
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone
number").MaskFormat("99-999-99999").Value("9090909090")
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate").
</div>
</div>
```

Run the above code example and you can see that it restricts entering a value exceeding the MinValue and MaxValue range mentioned in the Numeric textbox.

Set Calculation process with CurrencyTextBox Widgets

You can use events to calculate the total and displays the value. You can achieve this with the help of Click event in the button widget. The calculation steps are written in the call back function of Click event button.

To customize the button, you can set the ContentType as TextAndImage to include the icon before the text and add the ClientSideEvents for click event.

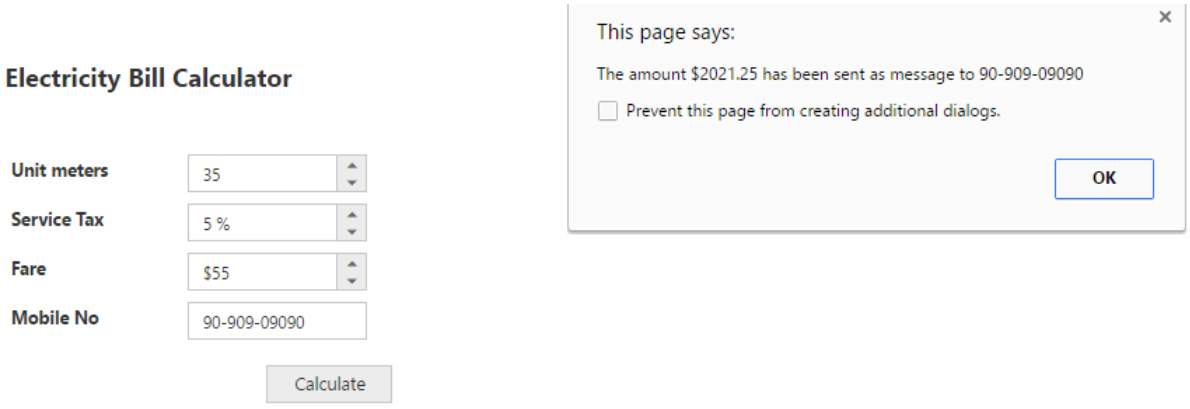
JS

```
<div class="editors">
@* Please refer the table format for textboxes customization *@
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate").ClientSide
Events(c=>c.Click("calculateBill"))
</div>
</div>
<script type="text/javascript">
function calculateBill() {
// Declares Necessary variable creation
var unitMeter, serviceTax, amountPerKm;
unitMeter = $("#numeric").data("ejNumericTextbox");// Object of Numeric
serviceTax = $("#percent").data("ejPercentageTextbox");// Object of
Percentage
amountPerKm = $("#currency").data("ejCurrencyTextbox"); // Object of
Currency
mobileNumber = $("#maskedit").data("ejMaskEdit"); // Object of MaskEdit
// This is used to calculate the Net amount
var netAmount = unitMeter.model.value * amountPerKm.model.value;
// This is used to calculate the service tax amount
var sTax = (netAmount * serviceTax.model.value) / 100;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// This shows the calculated amount for the units
alert("The amount $" + (netAmount + sTax) + " has been sent as message to "
+ mobileNumber.model.value + ".");
}
</script>
```

Run the above code sample, fill the required Textbox fields and click the Calculate button. The values are displayed and an alert message is shown. The following screen shot illustrates the final output of the Electricity bill calculator.



Electricity bill calculator with alert

Behavior Settings

Decimal Places

DecimalPlaces property specifies number of values allowed after the decimal point.The default value of DecimalPlaces property is 0 i.e., by default you cannot specify decimal value in CurrencyTextBox. We need to add this property to allow decimal values. To set the DecimalPlaces to “-1”, that allows the decimals without any limit in CurrencyTextBox control.

Configure Decimal Places

1. In the View page add CurrencyTextBox helper, and configure the DecimalPlaces property as follows.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").DecimalPlaces(4).Value("555")
```

The output is as follows.



CurrencyTextBox with decimalPlaces

Persistence Support

CurrencyTextBox provides state maintenance support. You can maintain the previous changes made in the control after a page loads.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Persistence Support

In the View page add CurrencyTextBox helper, and configure the EnablePersistence property as follows.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("33").EnablePersistence(true)
```

Output of CurrencyTextBox with EnablePersistence is as follows.



CurrencyTextBox at initial load



CurrencyTextBox after changing the value and after page refresh

Strict Mode Support

CurrencyTextBox allows you to use the strict mode option by setting the EnableStrictMode property. You can set the MinValue and MaxValue to the controls to enable strict mode functionality. Default value of this property is false. When the textbox value exceeds the MaxValue, it restricts the exceeded value and returns the MaxValue. Likewise when the textbox value goes below MinValue, it restricts the new value and returns the MinValue. When this property is enabled, it will not restrict the specified value and an error class will be added to indicate wrong value is provided to the textbox.

Configure Strict Mode Support

In the View page add CurrencyTextBox helper, and configure the EnableStrictMode property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").MinValue(-
1).MaxValue(8).Value(10).EnableStrictMode(true)
```

Output when EnableStrictMode is “true” is as follows.



Currency with EnableStrictMode

Enabled or Disabled

CurrencyTextBox has an option to enable or disable its element. You can set the Enabled property as “false” to enable the Textbox controls.

Also you can enable/disable the **CurrencyTextBox** by using enable and disable methods.

Configure Enabled or Disabled

In the View page add CurrencyTextBox helper, and configure the Enabled property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("3").Enabled(false)
```

Output when Enabled is “true” and when Enabled is “false”.

Copyright © 2001 - 2020 Syncfusion Inc.

CurrencyTextBox with enabled as false



CurrencyTextBox with enabled as true

Adjusting Textbox Size

CurrencyTextBox size can be modified by using the Height and Width properties.

Configure Height and Width

In the View page add CurrencyTextBox helper, and configure the Height and Width property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Width("100").Height("50").Value("1")
```

Output of CurrencyTextBox after setting “Height” and “Width” is as follows.



CurrencyTextBox with height and width

Increment Step

The IncrementStep property is used to increase or decrease the amount of value in the Currency textbox.

Configure Increment Step

In the View page add CurrencyTextBox helper, and configure the IncrementStep property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").IncrementStep(4).Value("1")
```

Output of Currency textbox with IncrementStep is as follows.



CurrencyTextBox at initial load



CurrencyTextBox after increasing two step

Define Name

When you have placed the CurrencyTextBox in a form, the Name property is used to send the field value at form submission. The default value of the Name property is null.

Configure Name

In the View page add CurrencyTextBox helper, and configure the Name property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Name("Currency")
```

Define Value

The value of CurrencyTextBox can be assigned by using the Value property. The default value for Value property is null.

You can get the value of **CurrencyTextBox** by using getValue method.

Configure Value

In the View page add CurrencyTextBox helper, and configure the Value property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("33")
```

Output of CurrencyTextBox with the value property is as follows.



CurrencyTextBox with value

Define MaxValue and MinValue

MaxValue

The maximum limit value can be assigned to the CurrencyTextBox by using the MaxValue property. The default value of MaxValue property is 1.7976931348623157e+308.

MinValue

The minimum limit value can be assigned to the CurrencyTextBox by using the MinValue property. The default value of MinValue property is -1.7976931348623157e+308.

Configure MaxValue and MinValue

In the View page add CurrencyTextBox helper, and configure the MinValue and MaxValue property. .

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("4").MinValue(-3).MaxValue(4)
```

Output of CurrencyTextBox with MaxValue and MinValue is as follows.



CurrencyTextBox with maxValue



CurrencyTextBox with minValue

Read Only Support

CurrencyTextBox supports read only option. When you enable the ReadOnly property to the control, the value cannot be changed in the CurrencyTextBox. You can set the ReadOnly property as “true” to enable this option.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Read Only

In the View page add CurrencyTextBox helper, and configure the ReadOnly property.

@Html.EJ().CurrencyTextbox("currency").Value("3").ReadOnly(true)

Output of CurrencyTextBox when ReadOnly is “true” is as follows. CurrencyTextBox values cannot be edited or changed.



CurrencyTextBox with readOnly

Appearance

Theme

CurrencyTextBox control’s style and appearance can be controlled based on CSS classes. In order to apply styles you need to refer 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. If the file ej.web.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.web.all.min.css is the combination of these two.

By default, there are 17 themes support available namely:

- bootstrap
- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- high-contrast-01
- high-contrast-02
- material
- office-365

CSS Class

The CSS can be customized by using the CssClass in the CurrencyTextBox. You can customize the CurrencyTextBox with CssClass property to appear like your desired control.

Configure CSS Class

In the View page add CurrencyTextBox helper, and configure the CssClass property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("3").CssClass("customCss")
```

Copyright © 2001 - 2020 Syncfusion Inc.

1. Customize the CSS properties in custom CSS class.

CSS

```
<style>
.customCss .e-box {
border-color: #9d241b;
}
.customCss .e-input {
background-color: #f6db8d;
}
.customCss .e-select {
background-color: #ecf6ac;
border-color: #3c36e7;
}
</style>
```

Output of CurrencyTextBox after applying CssClass is as follows.



CurrencyTextBox with cssClass

Rounded Corner Support

CurrencyTextBox provides you with rounded corner support whose appearance is different from normal textbox controls.

Configure Rounded Corner Support

In the View page add CurrencyTextBox helper, and configure the ShowRoundedCorner property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("3").ShowRoundedCorner(true)
```

Output of CurrencyTextBox when ShowRoundedCorner is “true”.



CurrencyTextBox with ShowRoundedCorner

Spin Button Support

CurrencyTextBox provides you the option as to whether to display the spin button in the widget or remove it from the control by using showSpinButton property.

Configure Spin Button

In the View page add CurrencyTextBox helper, and configure the ShowSpinButton property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("3").ShowSpinButton(false)
```

Output of CurrencyTextBox when ShowSpinButton is “false”.

Copyright © 2001 - 2020 Syncfusion Inc.



CurrencyTextBox with ShowSpinButton is false

Water Mark Text Support

The CurrencyTextBox provide water mark text support. You can display the initial value in the control by water mark.

Configure Water Mark Text

In the View page add CurrencyTextBox helper, and configure the ShowSpinButton property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Currency")
```

Output of CurrencyTextBox after applying WaterMarkText is as follows.



CurrencyTextBox with watermark text

Globalization Support

Globalization is language support based on the culture in CurrencyTextBox. You can achieve the **Globalization** using "**locale"** property in CurrencyTextBox.

The CurrencyTextBox widget provides multi-language support using globalization. You can customize the CurrencyTextBox with your own language style by using this feature. You can change the globalization by using the **locale** property. The default value for **locale** property is **en-US** in CurrencyTextBox controls. Also you can specify the currencySymbol property when the user wants to overwrite the currency symbol commonly instead of the current culture symbol.

More than 350 culture specific files are available to localize the value. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: All the culture-specific script files are available within the below specified location, once you have installed Essential Studio in your machine, therefore it is not necessary to download these files explicitly.

'(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n'
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

JAVASCRIPT

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/i18n/ej.culture.de-DE.min.js"></script>
```

You can dynamically change the language based on their culture.

Configure Localization

The following example describes the way to use localization in CurrencyTextBox.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("33").Locale("de-DE")
```

Output of CurrencyTextBox with localization.



CurrencyTextBox with de-DE locale



CurrencyTextBox with default (en-US) locale

RTL Support

CurrencyTextBox provides RTL (Right-To-Left) support. The alignment of CurrencyTextBox can be changed from Left-To-Right into Right-To-Left.

Enable RTL

In the View page add CurrencyTextBox helper, and configure the EnableRTL property.

ASPX-CS

```
@Html.EJ().CurrencyTextbox("currency").Value("33").EnableRTL(true)
```

Output of CurrencyTextBox when EnableRTL is “true” is as follows.



CurrencyTextBox with enableRTL

Keyboard Interaction

With the keyboard navigation enabled in the CurrencyTextBox control, it is possible to control the actions with the provided shortcut keys. Almost all the CurrencyTextBox functionality that are done through mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the CurrencyTextBox control are discussed in the following table.

Shortcut Key	Description
Access key + j	Focuses the control
Up	Increments the value
Down	Decrements the value
Tab	Focus the next element

Copyright © 2001 - 2020 Syncfusion Inc.

Configuring Keyboard Navigation

The following steps explain the implementation of keyboard interaction in CurrencyTextBox.

In the View page add CurrencyTextBox helper. Set the access key property to the CurrencyTextBox for focusing the control while key is pressed. We need to use 'HtmlAttributes' property to add the "access key" attribute to the Editor elements. 'HtmlAttributes' property is used to add HTML attributes like, id, class etc.. to the components. We need to use IDictionary<string,object> to specify the HTML attributes. Please check the below code.

ASPX-CS

```
@{IDictionary<string, object> currencyAttribute = new Dictionary<string,
object>();
currencyAttribute.Add("accesskey", "l");
}
@Html.EJ().CurrencyTextbox("currency").Name("currency").Value("33").HtmlAttr
ibutes(currencyAttribute)
```

Run the above example and press Access key + j key to focus the CurrencyTextBox widget. Perform provided functionality by using keyboard shortcuts.



CurrencyTextBox focused with keyboard shortcut

DataManager

DataManager

DataManager helps in managing relational data in the ASP.NET MVC. Its rich features make querying data sources easier. The ODataClient supports OData queries that are enabled in the Web API & WCF data services.

DataManager supports different types of data binding such as:

1. JSON
2. Web Services
3. OData

Basically, all services that have JSON acts as a transport.

Key Features

DataManager has many abilities and responsibilities, among which to:

1. **DataManager** - Communicates with data source and returns the desired result based on the Query provided.
2. **JavaScript Query** – **DataManager** have APIs for generating JavaScript data query with ease.
3. **CRUD in individual requests and Batch** – **CRUD** operations are fully supported. The options are enabled to commit the data as a single or multiple requests.
4. **Adaptors** – **Adaptors** are specific dataSource type aware interfaces that are used by **DataManager** to communicate with DataSource. Three adaptors are created. They are, ODataAdaptor, JsonAdaptor and UrlAdaptor

Copyright © 2001 - 2020 Syncfusion Inc.

5. **Model binding** – Simple model binding is made using **DataManager**. You can directly bind Query result to **HTML** element.
6. Calculate and maintain aggregates, sorting order and paging

In other words, the DataManager fully supports CRUD (Create, Read, Update, Destroy) data operations, and provides both client-side and server-side support for sorting, paging, filtering, grouping, and aggregates.

Getting Started

Create your DataManager in the ASP.NET MVC

DataManager is used to manage relational data. It supports CRUD (Create, Read, Update, and Destroy) in individual requests and Batch. DataManager uses DataManager control for processing and ej.Query for serving the data. DataManager control communicates with data source and ej.Query generates data queries that are read by DataManager.

Configure Demo Application

This section briefly describes how to make a connection to WCF “Northwind” OData service and generate a report with top five orders from customer HANAR with higher Freight charges. In this application scenario, you can learn how to bind the DataManager to the Grid control to do paging, filtering and sorting with Grid control in the DataManager by using ej.Query.

Create Connection

To define connection to data source, you can use the DataManager control. The data source can be local or remote. Local data source is the local data and remote data source is any web service.

The local data is set as data source by using the following code example.

ASPX-CS

```
<%--connection local reference--%>
@(Html.EJ().DataManager("FlatData").URL("Home/DataSource").Adaptor(AdaptorTy
pe.UrlAdaptor))
public class HomeController : Controller
{
public ActionResult Index()
{
return View();
}
public ActionResult DataSource(Syncfusion.JavaScript.DataManager
dataManagerObj)
{
var DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
result.result =
DataSource.Skip(dataManagerObj.Skip).Take(dataManagerObj.Take).ToList();
result.count = DataSource.Count();
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable<EditableOrder> result { get; set; }
public int count { get; set; }
}
}
```

In this application, as you have web service for Northwnd database, you can assign the web service URL link to the URL property of DataManager, and you can enable crossDomain to retrieve data from another domain.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").CrossDomain(true)
```

You can use the ej.Query to generate the report from web service.

Binding with Grid Control

You can bind the DataManager with Grid by defining the ID of DataManager in the DataManagerID property of the Grid control.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData") //DataManagerID(DataManager ID)
.Query("new ej.Query().take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
```

When you run, the following table is displayed.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

DataManager with Grid Control

Filter

You can generate the Filter query to filter the CustomerID column based on VINET value and it is ran by using the DataManager.

The where function is used to filter the records based on the specified filter condition.

Copyright © 2001 - 2020 Syncfusion Inc.

The select property of ejQuery is used to retrieve the specified columns from the data source.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID', 'ShipCity', 'Freight']).where('CustomerID', 'equal', 'VINET').take(5)")
//where(fieldName, operator, value, [ignoreCase])
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
```

When you run the filter query and bind the result to the Grid, the following table is displayed.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10274	VINET	6	Reims	6.0100
10295	VINET	2	Reims	1.1500
10737	VINET	2	Reims	7.7900
10739	VINET	3	Reims	11.0800

Data with Filtering

Sort

You can generate the Sort query to sort the Freight column in descending order and that is executed by using the DataManager.

The sortBy property of ejQuery is used to sort the records based on the field and direction specified.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID', 'ShipCity', 'Freight']).where('CustomerID', 'equal', 'VINET').sortBy('Freight desc').take(5)")
//sortBy(field direction)
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
```

When you run the sort query and bind the result to the table, the following table is displayed.

Order ID	Customer ID	Employee ID	Ship City	Freight
11078	VINET	5	Reims	32.3800
11908	VINET	5	Reims	32.3800
12738	VINET	5	Reims	32.3800
13568	VINET	5	Reims	32.3800
14398	VINET	5	Reims	32.3800

Data with Sorting

Page

You can generate the Paging query to get the top four orders and it is ran by using the DataManager.

The Page property of ejQuery is used to retrieve the records based on the given pageIndex and pageSize.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).where('CustomerID', 'equal',
'VINET').sortBy('Freight desc').page(3,5)")
//page(pageIndex,pageSize)
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Ship City	Freight
10739	VINET	3	Reims	11.0800
11569	VINET	3	Reims	11.0800
12399	VINET	3	Reims	11.0800
13229	VINET	3	Reims	11.0800
14059	VINET	3	Reims	11.0800

In this section, you can learn how to enable basic properties available in the DataManager and the usage of the various queries in the DataManager.

Data Binding

Local

A data source can be bound to a Grid through the DataManager. The DataManager supports JSON array binding. It is useful to bind records in a client-side by using JSON data that is very helpful in Single Page Application (SPA) and in feature rich web application. To achieve this, you can refer to the following code example.

ASPX-CS

```
//connection local reference
@(Html.EJ().DataManager("FlatData").URL("Home/DataSource").Adaptor(AdaptorTy
pe.UrlAdaptor))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).where('CustomerID', 'equal', 'VINET').take(5)")
//where(fieldName, operator, value, [ignoreCase])
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

C#

```
public class HomeController : Controller
{
public ActionResult Index()
{
return View();
}
public ActionResult DataSource(Syncfusion.JavaScript.DataManager
dataManagerObj)
{
```

```
var DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
result.result =
DataSource.Skip(dataManagerObj.Skip).Take(dataManagerObj.Take).ToList();
result.count = DataSource.Count();
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable<EditableOrder> result { get; set; }
public int count { get; set; }
}
}
```

The result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Binding local data to grid

REST Services

OData binding

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service by using DataManager. You can refer to the following code example of remote Data binding by using the OData service.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

OData binding

OData V4

The OData v4 is an improved version of OData protocols and the DataManager can also retrieve and consume OData v4 services. For more details on OData v4 Services, refer to the odata documentation.

You can refer to the following code example for consuming OData v4 services and bind the result to the Grid. In the the following code, crossDomain is enabled to make cross domain request.

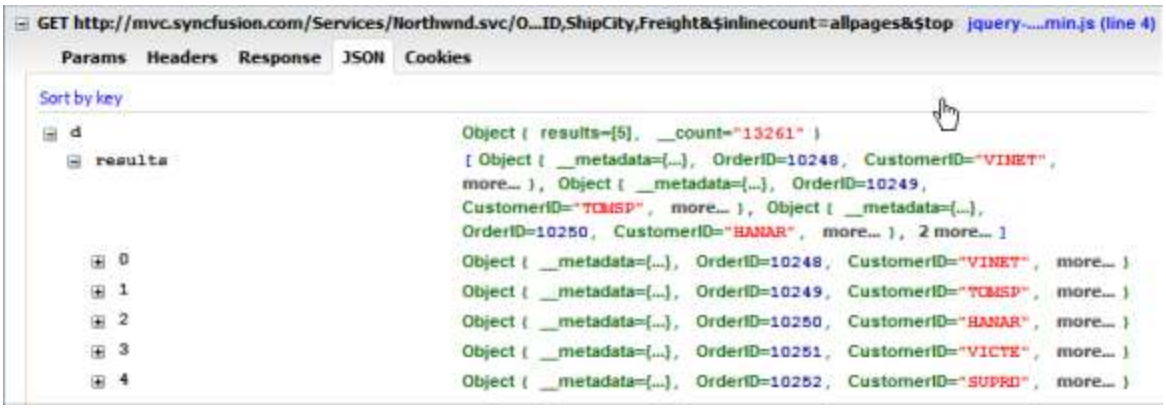
HTML

```
@Html.EJ().DataManager("FlatData").URL("http://services.odata.org/V4/Northwi
nd/Northwind.svc/Orders").Adaptor(AdaptorType.ODataV4Adaptor).CrossDomain(tr
ue)
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The request and response to the service from the DataManager are illustrated as follows.

The result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



OData v4 request and response

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

OData v4 binding

WebAPI binding

The Web API is a programmatic interface to define the request and response messages system that is mostly exposed in JSON or XML. The DataManager contains default adaptor to handle the Web API request and responses. The WebApiAdaptor is discussed briefly in the Adaptor section.

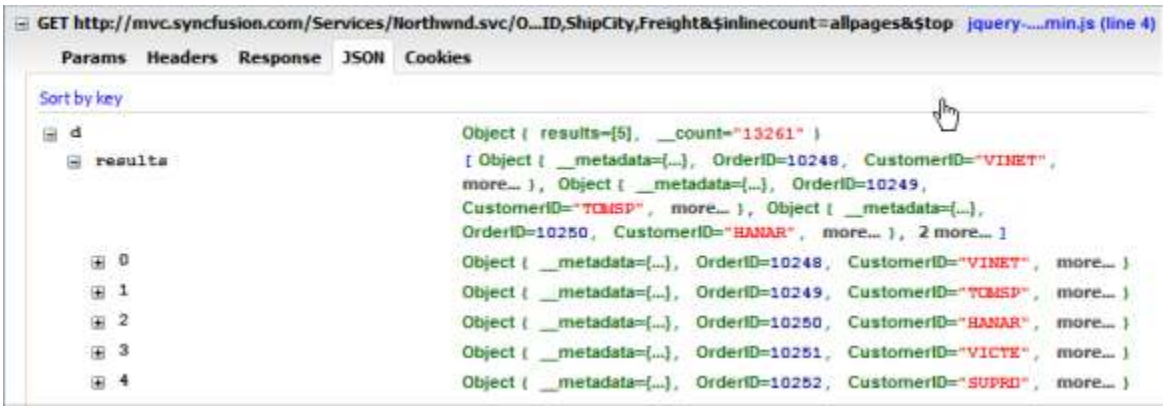
Refer to the following code example for consuming the Web API data by using the DataManager.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/UGService/
api/Orders").Adaptor(AdaptorType.WebApiAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).where('CustomerID', 'equal', 'VINET').take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The request to the Web API and response is illustrated as follows.



Web API adaptor request and response

The result for the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Web API data binding

Other Web Services binding

The DataManager can also retrieve data from the ASP.NET Web methods and ASP.NET MVC Controller`s action. You can achieve this by using the UrlAdaptor of the DataManager. The UrlAdaptor is discussed briefly in the adaptor section. By default, the URL Adaptor is used when accessing remote data.

Refer to the following code example to know how the DataManager can be used to consume data from the web method.

Create a sample using ASP.NET Web Services

HTML

```
public class WebService1 : System.Web.Services.WebService
{
static string cons =
ConfigurationManager.ConnectionStrings["SQLConnectionString"].ConnectionStri
ng;
static SqlConnection con = new SqlConnection(cons);
[WebMethod]
public DataTable Get()
{
SqlCommand getData = new SqlCommand();
getData.CommandText = "USP_DEV_ChangeLog_Select"; // Stored procedure for
retrieve data from suppliers table
getData.CommandType = CommandType.StoredProcedure;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
getData.Connection = con;
if (con.State != ConnectionState.Open)
con.Open();
DataTable sqlData = new DataTable();
SqlDataAdapter sqlAdapter = new SqlDataAdapter(getData);
sqlData.TableName = "Suppliers";
sqlAdapter.Fill(sqlData);
return sqlData;
}
}
```

In the above code snippet, we have created web services by using the ASP.NET web service and bound dataSource to Grid, in code behind GetDataSource method.

HTML

```
[WebMethod]
[ScriptMethod(ResponseFormat = ResponseFormat.Json)] // Return the JSON
formatted result
public static object GetDataSource()
{
CRUD_Service.WebService1 service = new CRUD_Service.WebService1();
var sqlData = service.Get();   // Get data from web services
DataResult result = new DataResult();
List<EditableCustomer> data = (from demand in sqlData.AsEnumerable() //
Perform skip take for on demand load paging
select new EditableCustomer
{
OrderID = demand.ItemArray[0].ToString(),
CustomerID = demand.ItemArray[1].ToString(),
EmployeeID = demand.ItemArray[5].ToString(),
Freight = demand.ItemArray[7].ToString(),
ShipCity = demand.ItemArray[8].ToString()
}).ToList();
con.Close();
return data;
}
```

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("WebService1.asmx/GetDataSource"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

ASP.NET Web service binding

Complex Data Sorting

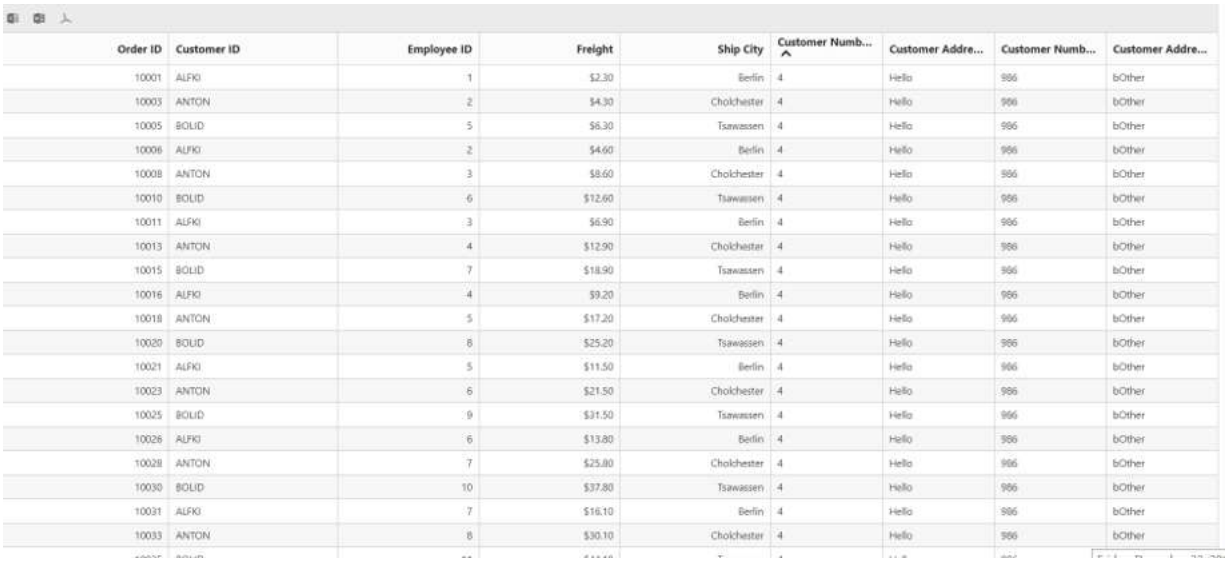
You can perform a data operation based on nested column data and bind to the data control. Here, if you want to perform the server side data operation for the nested dataSource like sorting, grouping, filtering, searching, take, or skip then we have to define the generic type of dataSource.

ASPX-CS

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(ds => ds.URL("GetData").Adaptor(AdaptorType.UrlAdaptor))
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowSorting()
.AllowPaging()
.IsResponsive()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(90).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship
City").TextAlign(TextAlign.Right).Width(80).Add();
for (int i = 0; i < 2; i++)
{
col.Field("customer." + i + ".CustomerNumber").HeaderText("Customer Number "
+ i).Width(60).Add();
col.Field("customer." + i + ".OtherAddress").HeaderText("Customer Address2 "
+ i).Width(60).Add();
}
})
)
```

C#

```
public ActionResult GetData(Syncfusion.JavaScript.DataManager dataManager)
{
BindDataSource();
IEnumerable<Orders> Data =order;
Syncfusion.JavaScript.DataSources.DataOperations operation = new
Syncfusion.JavaScript.DataSources.DataOperations();
if (dataManager.Sorted != null && dataManager.Sorted.Count > 0) //Sorting
{
Data = operation.PerformSorting(Data,dataManager.Sorted);
}
int count=Data.AsQueryable().Count();
return JSON(new { result = Data, count = count },
JsonRequestBehavior.AllowGet);
}
```



Order ID	Customer ID	Employee ID	Freight	Ship City	Customer Numb...	Customer Addre...	Customer Numb...	Customer Addre...
10001	ALFKI	1	$2.30	Berlin	4	Hello	986	bOther
10003	ANTON	2	$4.30	Cholchester	4	Hello	986	bOther
10005	BOLID	5	$6.30	Tsawassen	4	Hello	986	bOther
10006	ALFKI	2	$4.60	Berlin	4	Hello	986	bOther
10008	ANTON	3	$8.60	Cholchester	4	Hello	986	bOther
10010	BOLID	6	$12.60	Tsawassen	4	Hello	986	bOther
10011	ALFKI	3	$6.90	Berlin	4	Hello	986	bOther
10013	ANTON	4	$12.90	Cholchester	4	Hello	986	bOther
10015	BOLID	7	$18.90	Tsawassen	4	Hello	986	bOther
10016	ALFKI	4	$9.20	Berlin	4	Hello	986	bOther
10018	ANTON	5	$17.20	Cholchester	4	Hello	986	bOther
10020	BOLID	8	$25.20	Tsawassen	4	Hello	986	bOther
10021	ALFKI	5	$11.50	Berlin	4	Hello	986	bOther
10023	ANTON	6	$21.50	Cholchester	4	Hello	986	bOther
10025	BOLID	9	$31.50	Tsawassen	4	Hello	986	bOther
10026	ALFKI	6	$13.80	Berlin	4	Hello	986	bOther
10028	ANTON	7	$25.80	Cholchester	4	Hello	986	bOther
10030	BOLID	10	$37.80	Tsawassen	4	Hello	986	bOther
10031	ALFKI	7	$16.10	Berlin	4	Hello	986	bOther
10033	ANTON	8	$30.10	Cholchester	4	Hello	986	bOther

Data Adaptors

DataManger uses adaptors to process data. There are three types of adaptors in the DataManger. They are

- JSON Adaptor
- URL Adaptor
- OData Adaptor

Here, you can learn when and how each adaptor is used.

JSON Adaptor

JSONAdaptor is used to process JSON data. It contains methods to process the given JSON data based on the queries. The following code example illustrates on how to use the JSONAdaptor.

JSONAdaptor has the following unique in-built methods,

Properties	Parameters	Description

Copyright © 2001 - 2020 Syncfusion Inc.

<table>
<tr><td>processQuery(dataManagerObj, query)</td><td><table><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>query</td><td>ej.Query</td><td>Sets the default query for the data source</td></tr></table></td><td>Used to prepare query string for the request data</td></tr>
<tr><td>processResponse(data, dataObj, query, xhr)</td><td><table><tr><td>data</td><td>Object</td><td>JSON data or JSON array</td></tr><tr><td>dataObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>query</td><td>ej.Query</td><td>Sets the default query for the data source</td></tr><tr><td>xhr</td><td>Object</td><td>XMLHTTPRequest object</td></tr></table></td><td>Used to precess the response which is return from the Data Source</td></tr>
<tr><td>insert(dataManagerObj, data)</td><td><table><tr><td>data</td><td>Object</td><td>JSON data or JSON array</td></tr><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr></table></td><td>Inserts a data item in the data table.</td></tr>
<tr><td>remove(dataManagerObj, keyField, value, tableName)</td><td><table><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>keyField</td><td>String</td><td>KeyColumn to find the data</td></tr><tr><td>String</td><td>Object</td><td>Specified value for the keyField</td></tr><tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr></table></td><td>It is used to remove the data from the dataSource</td></tr>
<tr><td>update(dataManagerObj, keyField, value, tableName)</td><td><table><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>keyField</td><td>String</td><td>KeyColumn to find the data</td></tr><tr><td>String</td><td>Object</td><td>Specified value for the keyField</td></tr><tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr></table></td><td>Updates existing record and saves the changes to the table..</td></tr>
</table>

C#

```
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.dataSource = DataSource;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.dataSo
urce))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The result of above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10001	TOMSP	1	Münster	2.3
10002	HANAR	3	Rio de Janeiro	3.3
10003	VICTE	2	Lyon	4.3
10004	VINET	4	Reims	5.3
10005	SUPRD	5	Charleroi	6.3

JSON adaptor

URL Adaptor

URL Adaptor of the DataManager can be used when you are required to use remote service to retrieve data. It interacts with server-side for all DataManager Queries and CRUD operations. Now, in the following code example, the data is retrieved from the MVCController.

UrlAdaptor has the following unique in-built methods,

Properties	Parameters			Description
processQuery(dataManagerObj, query, hierarchyFilters)	dataManagerObj	Object	ej.DataManager object	Used to prepare query string for the request data
	query	ej.Query	Sets the default query for the data source	

<table>
<tr><td></td><td><table><tr><td>hierarchyFilters</td><td>ej.Query</td><td>The hierarchical query can be provided by using the hierarchical function.</td></tr></table></td><td></td></tr>
<tr><td>processResponse(data, dataObj, query, xhr, request, changes)</td><td><table><tr><td>data</td><td>Object</td><td>JSON data or JSON array of Result</td></tr><tr><td>dataObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>query</td><td>ej.Query</td><td>Sets the default query for the data source</td></tr><tr><td>xhr</td><td>Object</td><td>XMLHTTPRequest object</td></tr><tr><td>request</td><td>Object</td><td>request object to the Data Source</td></tr><tr><td>changes</td><td>Object</td><td>Specified changes to the Data Source</td></tr></table></td><td>Used to precess the response which is return from the Data Source</td></tr>
<tr><td>insert(dataManagerObj, data, tableName, query)</td><td><table><tr><td>data</td><td>Object</td><td>JSON data or JSON array</td></tr><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr><tr><td>query</td><td>ej.Query</td><td>Sets the default query for the data source</td></tr></table></td><td>Inserts a data item in the data table.</td></tr>
<tr><td>remove(dataManagerObj, keyField, value, tableName, query)</td><td><table><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr><tr><td>keyField</td><td>String</td><td>KeyColumn to find the data</td></tr><tr><td>String</td><td>Object</td><td>Specified value for the keyField</td></tr><tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr><tr><td>query</td><td>ej.Query</td><td>Sets the default query for the data source</td></tr></table></td><td>It is used to remove the data from the dataSource</td></tr>
<tr><td>update(dataManagerObj, keyField, value, tableName, query)</td><td><table><tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr></table></td><td>Updates existing record and saves the</td></tr>
</table>

Copyright © 2001 - 2020 Syncfusion Inc.

	keyField	String	KeyColumn to find the data	changes to the table..
	String	Object	Specified value for the keyField	
	tableName	String	Name of the source table	
	query	ej.Query	Sets the default query for the data source	

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("Home/DataSource").Adaptor(AdaptorTy
pe.UrlAdaptor))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

C#

```
public class HomeController : Controller
{
public ActionResult Index()
{
return View();
}
public ActionResult DataSource(Syncfusion.JavaScript.DataManager
dataManagerObj)
{
var DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
result.result =
DataSource.Skip(dataManagerObj.Skip).Take(dataManagerObj.Take).ToList();
result.count = DataSource.Count();
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable<EditableOrder> result { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public int count { get; set; }
}
}
```

The result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

URL Adaptor

OData Adaptor

OData Adaptor that is extended from URL Adaptor is used for consuming data through OData Service. You can use the following code example to use OData adaptor.

ODataAdaptor has the following unique in-built methods,

<table>
<tr><th>Properties</th><th colspan="3">Parameters</th><th>Description</th></tr>
<tr><td rowspan="5">processResponse(data, dataObj, query, xhr, request, changes)</td><td>data</td><td>Object</td><td>JSON data or JSON array</td><td rowspan="5">Used to precess the response which is return from the Data Source</td></tr>
<tr><td>dataObj</td><td>Object</td><td>ej.DataManager object</td></tr>
<tr><td>query</td><td>ej.Query</td><td>Sets the default query for the data source</td></tr>
<tr><td>xhr</td><td>Object</td><td>XMLHTTPRequest object</td></tr>
<tr><td>request</td><td>Object</td><td>request object to the Data Source</td></tr>
<tr><td>changes</td><td>Object</td><td>Specified changes to the Data Source</td></tr>
<tr><td rowspan="3">insert(dataManagerObj, data, tableName)</td><td>data</td><td>Object</td><td>JSON data or JSON array</td><td rowspan="3">Inserts a data item in the data table.</td></tr>
<tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr>
<tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr>
</table>

Copyright © 2001 - 2020 Syncfusion Inc.

<table>
<tr><td>remove(dataManagerObj, keyField, value, tableName)</td><td><table>
<tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr>
<tr><td>keyField</td><td>String</td><td>KeyColumn to find the data</td></tr>
<tr><td>String</td><td>Object</td><td>Specified value for the keyField</td></tr>
<tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr>
</table></td><td>It is used to remove the data from the dataSource</td></tr>
<tr><td>update(dataManagerObj, keyField, value, tableName)</td><td><table>
<tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr>
<tr><td>keyField</td><td>String</td><td>KeyColumn to find the data</td></tr>
<tr><td>String</td><td>Object</td><td>Specified value for the keyField</td></tr>
<tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr>
</table></td><td>Updates existing record and saves the changes to the table..</td></tr>
</table>

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders/").Offline(true).CrossDomain(true))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

OData adaptor

WebAPI Adaptor

WebAPIAdaptor extended from the UrlAdaptor of the DataManager is used for retrieving data from WebAPI service. Refer to the following code example.

WebApiAdaptor has the following unique in-built methods,

Properties	Parameters			Description
processResponse(data, dataObj, query, xhr, request, changes)	data	Object	JSON data or JSON array	Used to precess the response which is return from the Data Source
	dataObj	Object	ej.DataManager object	
	query	ej.Query	Sets the default query for the data source	
	xhr	Object	XMLHTTPRequest object	
	request	Object	request object to the Data Source	
	changes	Object	Specified changes to the Data Source	
insert(dataManagerObj, data, tableName)	data	Object	JSON data or JSON array	Inserts a data item in the data table.
	dataManagerObj	Object	ej.DataManager object	
	tableName	String	Name of the source table	
remove(dataManagerObj, keyField, value, tableName)	dataManagerObj	Object	ej.DataManager object	It is used to remove the data from the dataSource
	keyField	String	KeyColumn to find the data	

Copyright © 2001 - 2020 Syncfusion Inc.

	String	Object	Specified value for the keyField	
	tableName	String	Name of the source table	
update(dataManagerObj, keyField, value, tableName)	dataManagerObj	Object	ej.DataManager object	Updates existing record and saves the changes to the table..
	keyField	String	KeyColumn to find the data	
	String	Object	Specified value for the keyField	
	tableName	String	Name of the source table	

C#

```
public class EmployeeController : ApiController
{
static readonly IEmployeeRepository repository = new EmployeeRepository();
// GET API/<controller>
[HttpGet]
public object Get()
{
var data = repository.GetAll().ToList();
return new { Items = data, Count = data.Count() };
}
}
```

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("/api/Employee").Adaptor(AdaptorType
.WebApiAdaptor))
@(Html.EJ().Grid<Sample.Models.Employee>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['EmployeeID', 'FirstName', 'LastName'])")
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Width(75
).Add();
col.Field("FirstName").HeaderText("FirstName").Width(80).Add();
col.Field("LastName").HeaderText("LastName").Add();
})
)
```

Result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Employee	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	ANGEL	Reims

Insert

Web API Adaptor

RemoteSave Adaptor

RemoteSaveAdaptor extended from the JsonAdaptor of theDataManager is used for binding local data and performs all DataManager queries in client-side. It interacts with server-side only for CRUD operations to pass the modified records. Refer to the following code example.

RemoteSaveAdaptor has the following unique in-built methods,

Properties	Parameters			Description
insert(dataManagerObj, data, tableName, query)	data	Object	JSON data or JSON array	Inserts a data item in the data table.
	dataManagerObj	Object	ej.DataManager object	
	tableName	String	Name of the source table	
	query	ej.Query	Sets the default query for the data source	
remove(dataManagerObj, keyField, value, tableName, query)	dataManagerObj	Object	ej.DataManager object	It is used to remove the data from the dataSource
	keyField	String	KeyColumn to find the data	
	String	Object	Specified value for the keyField	
	tableName	String	Name of the source table	

Copyright © 2001 - 2020 Syncfusion Inc.

	query	ej.Query	Sets the default query for the data source	
update(dataManagerObj, keyField, value, tableName, query)	dataManagerObj	Object	ej.DataManager object	Updates existing record and saves the changes to the table..
	keyField	String	KeyColumn to find the data	
	String	Object	Specified value for the keyField	
	tableName	String	Name of the source table	
	query	ej.Query	Sets the default query for the data source	

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(dataSource =>
dataSource.Json((IEnumerable<object>)ViewBag.datasource).UpdateURL("/Home/Update")
.InsertURL("/Home/Insert").RemoveURL("/Home/Remove").Adaptor(AdaptorType.RemoteSaveAdaptor))
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

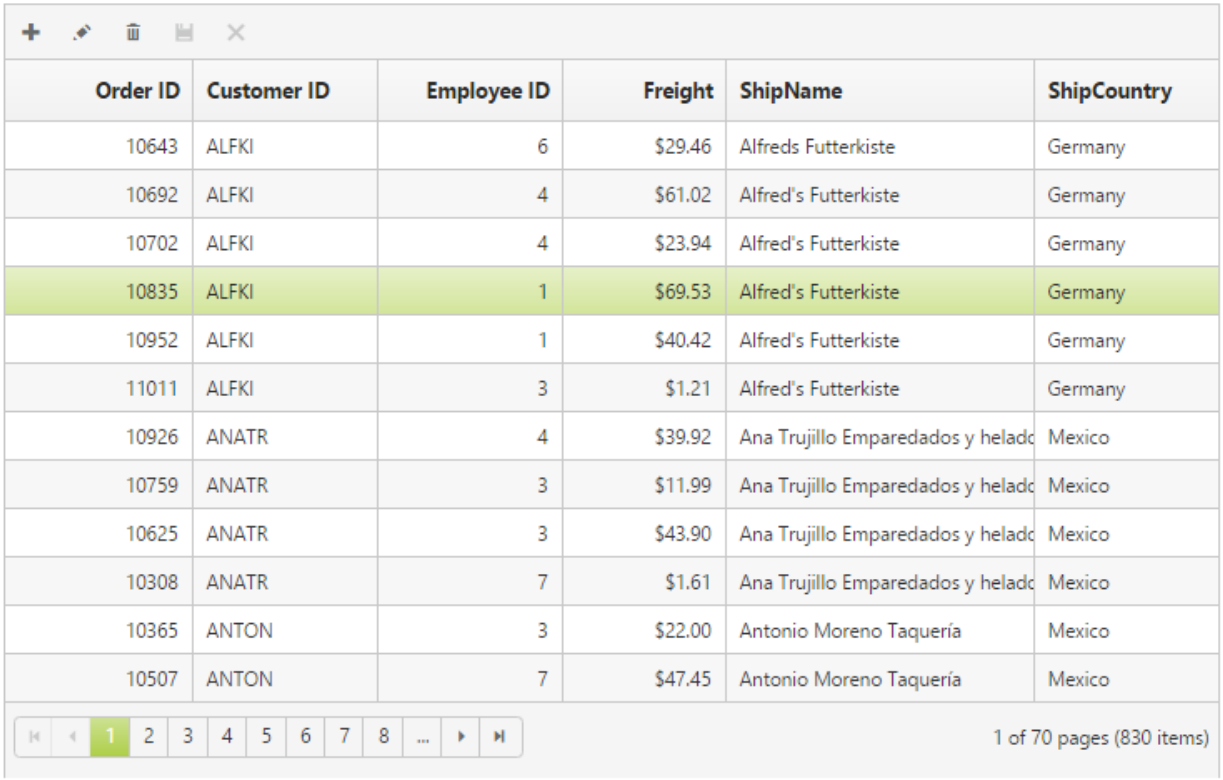
```
namespace EJGrid.Controllers
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public class HomeController : Controller
{
public ActionResult Index()
{
var data = OrderRepository.GetAllRecords();
ViewBag.dataSource = data;
return View();
}
public ActionResult Update(EditableOrder value)
{
OrderRepository.Update(value);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(EditableOrder value)
{
OrderRepository.Add(value);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
public ActionResult Remove(int key)
{
OrderRepository.Delete(key);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
}
}
```

Result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Order ID	Customer ID	Employee ID	Freight	ShipName	ShipCountry
10643	ALFKI	6	$29.46	Alfreds Futterkiste	Germany
10692	ALFKI	4	$61.02	Alfred's Futterkiste	Germany
10702	ALFKI	4	$23.94	Alfred's Futterkiste	Germany
10835	ALFKI	1	$69.53	Alfred's Futterkiste	Germany
10952	ALFKI	1	$40.42	Alfred's Futterkiste	Germany
11011	ALFKI	3	$1.21	Alfred's Futterkiste	Germany
10926	ANATR	4	$39.92	Ana Trujillo Emparedados y helado	Mexico
10759	ANATR	3	$11.99	Ana Trujillo Emparedados y helado	Mexico
10625	ANATR	3	$43.90	Ana Trujillo Emparedados y helado	Mexico
10308	ANATR	7	$1.61	Ana Trujillo Emparedados y helado	Mexico
10365	ANTON	3	$22.00	Antonio Moreno Taquería	Mexico
10507	ANTON	7	$47.45	Antonio Moreno Taquería	Mexico

1 2 3 4 5 6 7 8 ... 1 of 70 pages (830 items)

RemoteSave Adaptor

Custom Adaptor

Custom Adaptor is a key technique to customize adaptors in the DataManager. It is useful to write own adaptor. Normally ej.Adaptor is base class for all adaptors. Therefore, first inherit the ej.Adaptor to develop the customized one and then override functionality in Custom Adaptor with base class. The following code example illustrates you how to create the Custom Adaptor.

HTML

```
@(Html.EJ().DataManager("FlatData"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript">
$(function () {// Document is ready.
//new custom adaptor implementation
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//able to implement more option in custom adaptor other than insert
var customAdaptor = new ej.Adaptor().extend({
insert: function (dataManagerObj, data) {
return dataManagerObj.dataSource.json.push(data);
},
processQuery: ej.JsonAdaptor.prototype.processQuery // reused process query
from json adaptor
});
window.FlatData.Adaptor = new customAdaptor()
window.FlatData.insert({ OrderID: 10240, CustomerID: "HANAR", EmployeeID: 3,
ShipCity: "Reims", Freight: "23.4" });
window.FlatData.insert({ OrderID: 10241, CustomerID: "HANAR", EmployeeID: 2,
ShipCity: "Reims", Freight: "21.4" });
window.FlatData.insert({ OrderID: 10242, CustomerID: "VINET", EmployeeID: 5,
ShipCity: "Lyon", Freight: "13.4" });
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(window.FlatData.executeLocal(new ej.Query().take(7)));
})
</script>
```

Result of above code example is as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10240	HANAR	3	Reims	23.4
10241	HANAR	2	Reimse	21.4
10242	VINET	5	Lyon	13.4

Custom adaptor

Using Custom Adaptor, you can override the existing method of Extended Adaptor,

Properties	Description
beforeSend	Custom headers can be set using pre-request callback beforeSend, by using the setRequestHeader method can be used to modify the XMLHTTPRequest
processQuery	Used to prepare query string for the request data
processResponse	Used to precess the response which is return from the Data Source
insert	The insert method of the data manager is used to add a new record to the table
remove	The remove action submits the data items that should be deleted
update	The update method is used to update the modified changes made to a record in the data source of the DataManager.

Cache Adaptor

Cache Adaptor is used to cache the data of the visited pages. It prevents new requests for the previously visited pages. It can be enabled by using the EnableCaching property. You can configure cache page

Copyright © 2001 - 2020 Syncfusion Inc.

size and duration of caching by using CachingPageSize and TimeTillExpiration properties of the "DataManager".

CacheAdaptor has the following unique in-built methods,

Properties	Parameters			Description
processQuery(dataManagerObj, query, hierarchyFilters)	dataManagerObj	Object	ej.DataManager object	Used to prepare query string for the request data
	query	ej.Query	Sets the default query for the data source	
	hierarchyFilters	ej.Query	The hierarchical query can be provided by using the hierarchical function.	
processResponse(data, dataObj, query, xhr, request, changes)	data	Object	JSON data or JSON array	Used to precess the response which is return from the Data Source
	dataObj	Object	ej.DataManager object	
	query	ej.Query	Sets the default query for the data source	
	xhr	Object	XMLHTTPRequest object	
	request	Object	Sets the default query for the data source	
	changes	Object	XMLHTTPRequest object	
insert(dataManagerObj, data, tableName)	data	Object	JSON data or JSON array	Inserts a data item in the data table.
	dataManagerObj	Object	ej.DataManager object	
	tableName	String	Name of the source table	
remove(dataManagerObj, keyField, value, tableName)	dataManagerObj	Object	ej.DataManager object	It is used to remove the data from the dataSource
	keyField	String	KeyColumn to find the data	
	String	Object	Specified value for the keyField	
	tableName	String	Name of the source table	

Copyright © 2001 - 2020 Syncfusion Inc.

<table>
<tr>
<td>update(dataManagerObj, keyField, value, tableName)</td>
<td>
<table>
<tr><td>dataManagerObj</td><td>Object</td><td>ej.DataManager object</td></tr>
<tr><td>keyField</td><td>String</td><td>KeyColumn to find the data</td></tr>
<tr><td>String</td><td>Object</td><td>Specified value for the keyField</td></tr>
<tr><td>tableName</td><td>String</td><td>Name of the source table</td></tr>
</table>
</td>
<td>Updates existing record and saves the changes to the table..</td>
</tr>
</table>

ASPX-CS

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(dataSource => dataSource.URL(@Url.Action("DataSource"))
.EnableCaching()
.CachingPageSize(4)
.TimeTillExpiration(120000)
.Adaptor(AdaptorType.UrlAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order ID").Add();
col.Field(p => p.CustomerID).HeaderText("Customer
ID").TextAlign(TextAlign.Left).Add();
col.Field(p => p.EmployeeID).HeaderText("Employee ID").Add();
col.Field(p => p.Freight).HeaderText("Freight").Format("{0:C2}").Add();
})
)
```

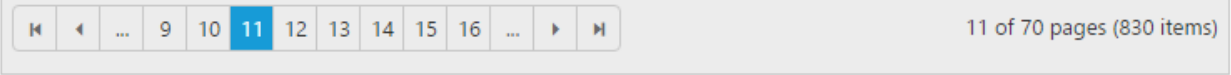
C#

```
//Refer the Syncfusion DataSource assembly
using Syncfusion.JavaScript;
using Syncfusion.JavaScript.DataSources;
public IEnumerable OrderData = new
NorthwindDataContext().OrdersViews.ToList();
public ActionResult GridFeatures()
{
return View();
}
public ActionResult DataSource(DataManager dataManagerObj)
{
IEnumerable data = OrderData;
DataOperations operation = new DataOperations();
if (dataManagerObj.Sorted != null && dataManagerObj.Sorted.Count > 0)
//Sorting
{
data= operation.PerformSorting(data, dataManagerObj.Sorted);
}
if (dataManagerObj.Where != null && dataManagerObj.Where.Count > 0)
//Filtering
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
data= operation.PerformWhereFilter(data, dataManagerObj.Where,
dataManagerObj.Where[0].Operator);
}
int count = data.Cast<OrdersView>().Count();
if (dataManagerObj.Skip != 0)
{
data= operation.PerformSkip(data, dataManagerObj.Skip);
}
if (dataManagerObj.Take != 0)
{
data= operation.PerformTake(data, dataManagerObj.Take);
}
return Json(new { result = data, count = count });
}
}
```

Order ID	Customer ID	Employee ID	Freight
10746	CHOPS	1	$31.43
10731	CHOPS	7	$96.65
10966	CHOPS	4	$27.19
11029	CHOPS	4	$47.84
11041	CHOPS	3	$48.22
11042	COMMI	2	$29.99
10969	COMMI	1	$0.21
10466	COMMI	4	$11.93
10494	COMMI	4	$65.99
10290	COMMI	8	$79.70
10435	CONSH	8	$9.21
10462	CONSH	2	$6.17



Cache Adaptor

Cache Adaptor has the following unique properties,

Properties	Description
TimeTillExpiration	Specifies the duration of cached pages in milliseconds
CachingPageSize	A number of pages to be cached

Query

DataManager provides support for multiple queries in order to perform various operations like filtering, sorting, cloning, expanding, searching, grouping etc., in the data source. Here, you can learn the query options in detail.

Select

The select query of the DataManager is used to select only some particular fields or columns from the data source. The following code example illustrates on how to select only particular fields by using the select query.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID', 'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Result of using “select”

From

The from query of the DataManager is used to select the table from where the data is retrieved and bound to the table. The following code example illustrates how to use the from query.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Query("new ej.Query().from('Orders').select(['OrderID', 'CustomerID',
'EmployeeID', 'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Result of using “from”

Clone

The clone query of the DataManager is used to duplicate the query. The following code example illustrates how to clone a query.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().from('Orders').select(['OrderID', 'CustomerID',
'EmployeeID', 'ShipCity', 'Freight']).take(15)")
.ClientSideEvents(e => e.ActionComplete("onComplete"))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var flag = true;
function OnComplete() {
if (flag) {
flag = false;
var query = ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)
var query1 = query.clone();
var proxy = $("#FlatGrid").ejGrid("instance");
var dataManagerObj = window.FlatData.executeQuery(query1).done(function (e1)
{
proxy.dataSource(e1.result);
})
}
}
</script>
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Result of using “clone”

Expand

The expand query of the DataManager is used to perform complex data binding.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().from('Orders').select(['OrderID', 'CustomerID',
'EmployeeID', 'ShipCity', 'Freight']).take(5).expand('Employee')")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("Employee.FirstName").HeaderText("Employee
Name").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Emp Name	Ship City	Freight
10248	VINET	Nancy	STOCKHOLM	6.0000
10249	TOMSP	Michael	MOSCOW	18.0000
10250	HANAR	Anne	OTTAWA	8.0000
10251	VICTE	Janet	KIEV	43.0000
10252	SUPRD	Margaret	COPENHAGEN	28.0000

Result of using “expand”

Sorting

Default

Sorting is basic query in the DataManager. It enables you to view the items or records in ascending or descending order based on a particular field and sorting direction. The query parameter of the DataManager enables you to retrieve the data in the sorted fashion and thus utilizing the resultant data obtained.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().sortBy('CustomerID', ej.sortOrder.Ascending, false).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10643	ALFKI	6	Berlin	29.4600
10692	ALFKI	4	Berlin	61.0200
10702	ALFKI	4	Berlin	23.9400
10835	ALFKI	1	Berlin	69.5300
10952	ALFKI	1	Berlin	40.4200

Sorting

Copyright © 2001 - 2020 Syncfusion Inc.

SortByDesc

The sortByDesc query of the DataManager is used to sort the specified field in descending order, by default. You can use the following code example for sortByDesc query.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().sortByDesc('EmployeeID').take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10250	HANAR	9	OTTAWA	8.0000
10255	RICSU	9	MOSCOW	148.0000
10263	ERNSH	9	Graz	146.0600
10324	SAVEA	9	Boise	214.2700
10331	BONAP	9	Marseille	10.1900

Sort in descending order

Dynamic sorting

The table can be dynamically sorted by using an external button click event. The value of the column to be sorted can be obtained by the sortBy query and thus the sorted data is retrieved and bounded to the table. The following code example illustrates you to dynamically sort the data source.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().sortByDesc('EmployeeID').take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<input id="field" type="text" placeholder="fieldName" />
@Html.EJ().Button("submit").Text("Sort").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
function onClick(e) {
var field = $("#field").val();
var obj = $("#FlatGrid").ejGrid("instance")
var query = ej.Query().sortByDesc(field).take(5)
var dataManagerObj = window.FlatData.executeQuery(query).done(function (e1)
{
obj.dataSource(e1.result);
})
}
</script>
```

Result of above code example is illustrated as follows.



Dynamic sorting

Multi sorting

Multi sorting is a special technique, where you can sort multiple fields by adding multiple sorting queries to the DataManager.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().sortByDesc('CustomerID').sortBy('EmployeeID',
'ascending').take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
```

```
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10258	ERNSH	1	Graz	140.5100
10270	WARTH	1	Oulu	136.5400
10275	MAGAA	1	Bergamo	26.9300
10285	QUICK	1	Cunewalde	76.8[illegible]0

Multi sorting

Searching

Searching is a basic query technique in the Data Manager. It is used to filter the records from the entire data source based on the search key parameter.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().search(4, 'EmployeeID').take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10004	VINET	4	Reims	5.3
10007	HANAR	4	Rio de Janeiro	6.6
10013	VICTE	4	Lyon	12.899999999999
10016	TOMSP	4	Münster	9.2

Searching

Filtering

Filtering is a basic technique in the DataManager query. The where query is used to filter some particular or related records from the data source to review details of records.

Filter Operators

Filter operators are generally used to specify the filter type. The various filter operators corresponding to the type of the column are listed in the following table.

Filter Operators

Column type	Filter operators
Number	ej.FilterOperators.greaterThan ej.FilterOperators.greaterThanOrEqual ej.FilterOperators.lessThan ej.FilterOperators.lessThanOrEqual ej.FilterOperators.equal
String	ej.FilterOperators.startsWith ej.FilterOperators.endsWith ej.FilterOperators.contains ej.FilterOperators.equal ej.FilterOperators.notEqual
Boolean	ej.FilterOperators.equal ej.FilterOperators.notEqual
Date	ej.FilterOperators.greaterThan ej.FilterOperators.greaterThanOrEqual ej.FilterOperators.lessThan ej.FilterOperators.lessThanOrEqual ej.FilterOperators.equal

lessThan

This operator is used to get the records with values lesser than the filter value.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Query("new ej.Query().where('OrderID', 'lessThan', 10252, false)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000

Filter using “lessThan” operator

greaterThan

This operator is used to get the records with values greater than that of the filter value.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('OrderID', 'greaterThan', 10252,
false).take(5)")
.Columns(col =>        {
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10253	HANAR	3	Juneau	35.0000
10254	CHOPS	5	STOCKHOLM	28.0000
10255	RICSU	9	MOSCOW	148.0000
10256	WELLI	3	Resende	13.9700
10257	HILAA	4	San Cristóbal	81.9100

Filter using “greaterThan” operator

lessThanOrEqual

This operator is used to get the records with values lesser than or equal to the filter value.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('OrderID', 'lessThanOrEqual', 10252, false)")
.Columns(col =>        {
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Filter using “lessThanOrEqual” operator

greaterThanOrEqual

This operator is used to get the records with values greater than or equal to the filter value.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('OrderID', 'greaterThanOrEqual', 10252,
false).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10252	SUPRD	4	COPENHAGEN	28.0000
10253	HANAR	3	Juneau	35.0000
10254	CHOPS	5	STOCKHOLM	28.0000
10255	RICSU	9	MOSCOW	148.0000
10256	WELLI	3	Resende	13.9700

Filter using “greaterThanOrEqual” operator

equal

This operator is used to get the records with values equal to that of the filter value.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('EmployeeID', 'equal', 4).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10004	VINET	4	Reims	5.3
10007	HANAR	4	Rio de Janeiro	6.6
10013	VICTE	4	Lyon	12.899999999999
10016	TOMSP	4	Münster	9.2

Filter using “equal” operator

notEqual

This operator is used to get the records with values not equal to that of the filter value specified.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('EmployeeID', 'notEqual', 4).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10253	HANAR	3	Juneau	35.0000

Filter using “notEqual” operator

contains

This operator is used to get the records that contains the filter value.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.DataManagerID("FlatData")
.Query("new ej.Query().where('CustomerID', 'contains', 'A', false
).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10250	HANAR	9	OTTAWA	8.0000
10253	HANAR	3	Juneau	35.0000
10257	HILAA	4	San Cristóbal	81.9100
10262	RATTC	8	Albuquerque	48.2900
10266	WARTH	3	Oulu	25.7300

Filter using “contains” operator

startswith

This operator is used to get the records that starts with the filter value specified.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('CustomerID', 'startswith', 'A', false
).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10308	ANATR	7	México D.F.	1.6100
10355	AROUT	6	Colchester	41.9500
10365	ANTON	3	México D.F.	22.0000
10383	AROUT	8	Colchester	34.2400
10453	AROUT	1	Colchester	25.3600

Filter using “startswith” operator

endswith

This operator is used to get the records that ends with the filter value specified.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where('CustomerID', 'endswith', 'A', false
).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10257	HILAA	4	San Cristóbal	81.9100
10275	MAGAA	1	Bergamo	26.9300
10300	MAGAA	2	Bergamo	17.6800
10324	SAVEA	9	Boise	214.2700
10347	FAMIA	4	Sao Paulo	3.1000

Filter using “endswith” operator

and predicate

The and predicate is used to add n-number of predicates with and condition and filter the data.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where(ej.Predicate('OrderID',
ej.FilterOperators.greaterThan, 10399, true)
.and('CustomerID', ej.FilterOperators.startsWith, 'V', true)).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10450	VICTE	8	Lyon	7.2300
10459	VICTE	4	Lyon	25.0900
10465	VAFFE	1	Århus	145.0400
10478	VICTE	2	Lyon	4.8100
10546	VICTE	1	Lyon	194.7200

Result of using “and” condition

or predicate

By using this method, you can add n-number of predicates with or condition and filter the data.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().where(ej.Predicate('OrderID',
ej.FilterOperators.greaterThan, 10399, true)
.or('CustomerID', ej.FilterOperators.startsWith, 'V', true)).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10251	VICTE	3	KIEV	43.0000
10274	VINET	6	Reims	6.0100
10295	VINET	2	Reims	1.1500
10334	VICTE	8	Lyon	8.5600

Result of using “or” condition

Containment Operators

notcontains

This operator is used to get the records that not contains the filter value.

HTML

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("ej.Query()
.from('Orders').select('OrderID', 'CustomerID', 'EmployeeID', 'Freight',
'ShipCountry')
.page(1,5).where('ShipCountry', 'notcontains', 'CAN', false)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship City").Width(110).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Freight	Ship Country
10248	VINET	1	6.0000	SWEDEN
10249	TOMSP	6	18.0000	INDIA
10251	VICTE	3	43.0000	UKRAINE
10252	SUPRD	4	28.0000	DENMARK
10253	HANAR	3	35.0000	Venesuela

contains

This operator is used to get the records that contains the filter value.

HTML

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("ej.Query()
.from('Orders').select('OrderID', 'CustomerID', 'EmployeeID', 'Freight', 'ShipCountry')
.page(1,5).where('ShipCountry', 'contains', 'CAN', false)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship City").Width(110).Add();
})
)
```

Order ID	Customer ID	Employee ID	Freight	Ship Country
10250	HANAR	9	8.0000	CANADA
10332	MEREP	3	52.8400	Canada
10339	MEREP	2	15.6600	Canada
10376	MEREP	1	20.3900	Canada
10389	BOTTM	4	47.4200	Canada

in

This operator used to fetch the records with value match with the given filter value.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("ej.Query()
.from('Orders').select('OrderID', 'CustomerID', 'EmployeeID', 'Freight', 'ShipCountry')
.page(1,5).where('ShipCountry', 'IN', ['INDIA'], false)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(110).Add();
})
)
```

Order ID	Customer ID	Employee ID	Freight	Ship Country
10249	TOMSP	6	18.0000	INDIA

notin

This operator used to fetch the records with value not match with the given filter value.

HTML

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("ej.Query()
.from('Orders').select('OrderID', 'CustomerID', 'EmployeeID', 'Freight', 'ShipCountry')
.page(1,5).where('ShipCountry', 'notin', ['INDIA'], false)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(110).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Freight	Ship Country
10248	VINET	1	6.0000	SWEDEN
10250	HANAR	9	8.0000	CANADA
10251	VICTE	3	43.0000	UKRAINE
10252	SUPRD	4	28.0000	DENMARK
10253	HANAR	3	35.0000	Venesuela

▼ **Query String Parameters** view source view URL encoded

$select: OrderID,CustomerID, EmployeeID,Freight,ShipCountry
$filter: ShipCountry ne 'INDIA'
$skip: 0
$top: 5

Lambda Operators

all

The ALL operator returns true if all the sub query values meet the condition. This operator used to fetch the records of the fields match with the given value's fields.

HTML

```
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(110).Add();
})
)
```

JAVASCRIPT

```
setTimeout(function () {
var gridData = [{ OrderID: 10248, CustomerID: "VINET", EmployeeID: 5,
OrderDate: new Date(8364186e5), ShipName: "Vins et alcools Chevalier",
ShipCity: "Reims", ShipAddress: "59 rue de l'Abbaye", ShipRegion: null,
ShipPostalCode: "51100", ShipCountry: "France", Freight: 32.38, Verified: !0
}];
var data = [{ OrderID: 10248, CustomerID: "VINET", EmployeeID: 5, OrderDate:
new Date(8364186e5), ShipName: "Vins et alcools Chevalier", ShipCity:
"Reims", ShipAddress: "59 rue de l'Abbaye", ShipRegion: null,
ShipPostalCode: "51100", ShipCountry: "France", Freight: 32.38, Verified: !0
}];
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var dataManager = ej.DataManager(gridData);
var dataSource = ej.DataManager(data);
var result = ej.Query().select("ShipCountry")
var ShipCountry = dataSource.executeLocal(result);
var query = ej.Query()
.where("ShipCountry", "equal all", ShipCountry, false).select("OrderID",
"CustomerID", "EmployeeID", "Freight", "ShipCountry");
var execute = dataManager.executeLocal(query); // executing query
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(execute);
}, 1000);
```

Order ID	Customer ID	Employee ID	Freight	Ship Country
10248	VINET	5	32.38	France

any

The ANY operator returns true if any of the sub query values meet the condition. The returned data match with any one of the field in existing table.

HTML

```
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(110).Add();
})
)
```

JAVASCRIPT

```
setTimeout(function () {
var gridData = window.gridData;
var data = [{ OrderID: 10343, CustomerID: "LEHMS", EmployeeID: 4, OrderDate:
new Date(8467002e5), ShipName: "Lehmanns Marktstand", ShipCity: "Frankfurt
a.M.", ShipAddress: "Magazinweg 7", ShipRegion: null, ShipPostalCode:
"60528", ShipCountry: "Germany", Freight: 110.37, Verified: !0 }, { OrderID:
10344, CustomerID: "WHITC", EmployeeID: 4, OrderDate: new Date(8467866e5),
ShipName: "White Clover Markets", ShipCity: "Seattle", ShipAddress: "1029 -
12th Ave. S.", ShipRegion: "WA", ShipPostalCode: "98124", ShipCountry:
"USA", Freight: 23.29, Verified: !1 }];
var dataManager = ej.DataManager(gridData);
var dataSource = ej.DataManager(data);
var result = ej.Query().select("ShipCountry")
var ShipCountry = dataSource.executeLocal(result);
var query = ej.Query()
.where("ShipCountry", "equal any", ShipCountry, false).select("OrderID",
"CustomerID", "EmployeeID", "Freight", "ShipCountry");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var execute = dataManager.executeLocal(query); // executing query
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(execute);
}, 1000);
```

Order ID	Customer ID	Employee ID	Freight	Ship Country
10249	TOMSP	6	11.61	Germany
10260	OTTIK	4	55.09	Germany
10262	RATTC	8	48.29	USA
10267	FRANK	4	208.58	Germany
10269	WHITC	5	4.56	USA
10271	SPLIR	6	4.54	USA
10272	RATTC	6	98.03	USA
10273	QUICK	3	76.07	Germany
10277	MORGK	2	125.77	Germany
10279	LEHMS	8	25.83	Germany
10284	LEHMS	4	76.56	Germany
10285	QUICK	1	76.83	Germany

Grouping

Grouping technique is also supported in the DataManager. When you want to analyze any particular record based on its category, simply you can group that column and analyze the records based on category. The following code example illustrates the grouping behavior in table.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().page(1,5).group('CustomerID')")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Result of the above code example is illustrated as follows.



Grouping

CRUD Data Operations

The DataManager fully supports the CRUD (Create, Read, Update, Destroy) data operations. However, it must be combined with some user interface or another Syncfusion UI widget such as the Grid, ListView, etc.

Local CRUD Operations

The information in this section is applicable to scenarios, in which the data is already available on the client, or when it is you that is going to take care of its retrieval and submission. In other words, DataManager will not make any HTTP requests on its own.

Insert

The insert method of the data manager is used to add a new record to the table. The JSON data passed as a parameter to the insert method that is inserted to the data source of the data manager.

C#

```
public static List<Employee> employee = new List<Employee>();
public ActionResult Insert()
{
BindDataSource();
ViewBag.data = order;
return View();
}
public void BindDataSource()
{
int code = 10000;
order.Add(new OrderDetails(code + 1, "ALFKI", 1 + 0, 2.3 * 1, "Berlin"));
order.Add(new OrderDetails(code + 2, "ANATR", 1 + 2, 3.3 * 1, "Madrid"));
order.Add(new OrderDetails(code + 3, "ANTON", 1 + 1, 4.3 * 1,
"Cholchester"));
order.Add(new OrderDetails(code + 4, "BLONP", 1 + 3, 5.3 * 1, "Marseille"));
order.Add(new OrderDetails(code + 5, "BOLID", 1 + 4, 6.3 * 1, "Tsawassen"));
}
public class OrderDetails
{
public OrderDetails()
{
```

```
}
public OrderDetails(int OrderID, string CustomerId, int EmployeeId, double
Freight, string ShipCity)
{
this.OrderID = OrderID;
this.CustomerID = CustomerId;
this.EmployeeID = EmployeeId;
this.Freight = Freight;
this.ShipCity = ShipCity;
}
public int? OrderID { get; set; }
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
}
```

C#

```
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.data).
Adaptor(AdaptorType.JsonAdaptor))
@Html.EJ().Button("submit").Text("Insert").ClientSideEvents(e => {
e.Click("onClick"); })
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight'])")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
function onClick(e) {
var record = { OrderID: 10253, CustomerID: "ANGEL", EmployeeID: 4, ShipCity:
"Reims", Freight:23.1 };
window.FlatData.insert(record)
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(window.FlatData.executeLocal(ej.Query().sortByDesc("OrderID")
));
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Freight	Ship City
10253	ANGEL	4	$23.10	Reims
10005	BOLID	5	$6.30	Tsawassen
10004	BLONP	4	$5.30	Marseille
10003	ANTON	2	$4.30	Cholchester
10002	ANATR	3	$3.30	Madrid
10001	ALFKI	1	$2.30	Berlin

Insert

Update

The update method is used to update the modified changes made to a record in the data source of the DataManager.

C#

```
public static List<Employee> employee = new List<Employee>();
public ActionResult Update()
{
BindDataSource();
ViewBag.data = order;
return View();
}
public void BindDataSource()
{
int code = 10000;
order.Add(new OrderDetails(code + 1, "ALFKI", 1 + 0, 2.3 * 1, "Berlin"));
order.Add(new OrderDetails(code + 2, "ANATR", 1 + 2, 3.3 * 1, "Madrid"));
order.Add(new OrderDetails(code + 3, "ANTON", 1 + 1, 4.3 * 1,
"Cholchester"));
order.Add(new OrderDetails(code + 4, "BLONP", 1 + 3, 5.3 * 1, "Marseille"));
order.Add(new OrderDetails(code + 5, "BOLID", 1 + 4, 6.3 * 1, "Tsawassen"));
}
public class OrderDetails
{
public OrderDetails()
{
}
public OrderDetails(int OrderID, string CustomerId, int EmployeeId, double
Freight, string ShipCity)
{
this.OrderID = OrderID;
this.CustomerID = CustomerId;
this.EmployeeID = EmployeeId;
this.Freight = Freight;
this.ShipCity = ShipCity;
}
public int? OrderID { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
}
```

C#

```
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.data).
Adaptor(AdaptorType.JsonAdaptor))
@Html.EJ().Button("submit").Text("Update").ClientSideEvents(e => {
e.Click("onClick"); })
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight'])")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
function onClick(e) {
var record = { OrderID: 10005, CustomerID: "ANGEL", EmployeeID: 5, ShipCity:
"Reims", Freight:23.1 };
window.FlatData.update("OrderID", record, window.FlatData.dataSource.json);
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(window.FlatData.executeLocal(ej.Query().sortByDesc("OrderID")
));
}
</script>
```

Order ID	Customer ID	Employee ID	Freight	Ship City
10005	ANGEL	5	$23.10	Reims
10004	BLONP	4	$5.30	Marseille
10003	ANTON	2	$4.30	Cholchester
10002	ANATR	3	$3.30	Madrid
10001	ALFKI	1	$2.30	Berlin

Update

Copyright © 2001 - 2020 Syncfusion Inc.

Remove

The remove function receives the items to be deleted in the Data Table. The function should remove the provided items from the data source of the DataManager.

C#

```
public static List<Employee> employee = new List<Employee>();
public ActionResult Remove()
{
BindDataSource();
ViewBag.data = order;
return View();
}
public void BindDataSource()
{
int code = 10000;
order.Add(new OrderDetails(code + 1, "ALFKI", 1 + 0, 2.3 * 1, "Berlin"));
order.Add(new OrderDetails(code + 2, "ANATR", 1 + 2, 3.3 * 1, "Madrid"));
order.Add(new OrderDetails(code + 3, "ANTON", 1 + 1, 4.3 * 1,
"Cholchester"));
order.Add(new OrderDetails(code + 4, "BLONP", 1 + 3, 5.3 * 1, "Marseille"));
order.Add(new OrderDetails(code + 5, "BOLID", 1 + 4, 6.3 * 1, "Tsawassen"));
}
public class OrderDetails
{
public OrderDetails()
{
}
public OrderDetails(int OrderID, string CustomerId, int EmployeeId, double
Freight, string ShipCity)
{
this.OrderID = OrderID;
this.CustomerID = CustomerId;
this.EmployeeID = EmployeeId;
this.Freight = Freight;
this.ShipCity = ShipCity;
}
public int? OrderID { get; set; }
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
}
```

C#

```
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.data).
Adaptor(AdaptorType.JsonAdaptor))
@Html.EJ().Button("submit").Text("Remove").ClientSideEvents(e => {
e.Click("onClick"); })
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight'])")
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
function onClick(e) {
var record = { OrderID: 10005, CustomerID: "BOLID", EmployeeID: 4, ShipCity:
"Tsawassen", Freight: 6.3 };
window.FlatData.remove("OrderID", record, window.FlatData.dataSource.json);
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(window.FlatData.executeLocal(ej.Query().sortByDesc("OrderID")
));
}
</script>
```

Order ID	Customer ID	Employee ID	Freight	Ship City
10004	BLONP	4	$5.30	Marseille
10003	ANTON	2	$4.30	Cholchester
10002	ANATR	3	$3.30	Madrid
10001	ALFKI	1	$2.30	Berlin

Remove

Remote CRUD Operations

The information in this section is applicable to scenarios, in which the data should be retrieved from and submitted to a remote data service via HTTP requests made by the DataManager.

CRUD operations with remote data rely on server code to perform the read, update, create and destroy actions. Instead of configuring client functions, the DataManager defines remote service URLs and the expected format in which data should be sent and received. Theoretically, it is possible to use remote CRUD operations with transport functions, similar to the above examples that use local data, but this is rarely required.

Each of the CRUD operation settings—read, update, create, destroy—provides some common settings that need to be set accordingly:

- The client request type can be "get" or "post".
- Additional optional headers parameters can be sent to the server if needed.
- The client request and expected server response dataType can be "json", "jsonp", "odata", etc.

Copyright © 2001 - 2020 Syncfusion Inc.

Insert

The insert method of the data manager is used to add a new record to the table. The JSON data passed as a parameter to the insert method that is inserted to the data source of the data manager.

C#

```
//Employee Model class
public class Employee
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public Employee()
{
}
public Employee(int id, string LN, string FN)
{
this.EmployeeID = id;
this.LastName = LN;
this.FirstName = FN;
}
}
//Employee Repository interface
interface IEmployeeRepository
{
IEnumerable<Employee> GetAll();
Employee Get(int EmployeeID);
Employee Add(Employee employee);
void Remove(int EmployeeID);
bool Update(Employee employee);
}
//EmployeeRepository class
public class EmployeeRepository : IEmployeeRepository
{
private List<Employee> employee = new List<Employee>();
public EmployeeRepository()
{
employee.Add(new Employee(1, "Davolio", "Nancy"));
employee.Add(new Employee(2, "Fuller", "Andrew"));
employee.Add(new Employee(3, "Leverling", "Janet"));
employee.Add(new Employee(4, "Peacock", "Margaret"));
employee.Add(new Employee(5, "Buchanan", "Steven"));
}
public IEnumerable<Employee> GetAll()
{
return employee;
}
public Employee Get(int id)
{
return employee.Find(p => p.EmployeeID == id);
}
public Employee Add(Employee eObj)
{
if (eObj == null)
{
throw new ArgumentNullException("eObj");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
employee.Add(eObj);
return eObj;
}
public void Remove(int id)
{
employee.RemoveAll(p => p.EmployeeID == id);
}
public bool Update(Employee eObj)
{
if (eObj == null)
{
throw new ArgumentNullException("eObj");
}
int index = employee.FindIndex(p => p.EmployeeID == eObj.EmployeeID);
if (index == -1)
{
return false;
}
employee.RemoveAt(index);
employee.Add(eObj);
return true;
}
}
// Employee WebAPI controller
public class EmployeeController : ApiController
{
static readonly IEmployeeRepository repository = new EmployeeRepository();
// GET API/<controller>
[HttpGet]
public object Get()
{
var queryString = HttpContext.Current.Request.QueryString;
var data = repository.GetAll().ToList();
return new { Items = data, Count = data.Count() };
}
public Employee GetEmployee(int id)
{
Employee employee = repository.Get(id);
if (employee == null)
{
throw new HttpResponseException(HttpStatusCode.NotFound);
}
return employee;
}
// POST API/<controller>
public HttpResponseMessage PostEmployee(Employee employee)
{
employee = repository.Add(employee);
var response = Request.CreateResponse<Employee>(HttpStatusCode.Created,
employee);
string addEmployee = Url.Link("Employee", new { id = employee.EmployeeID });
response.Headers.Location = new Uri(addEmployee);
return response;
}
[HttpPut]
// PUT API/<controller>
public void PutEmployee(Employee employee)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
if (!repository.Update(employee))
{
throw new HttpResponseException(HttpStatusCode.NotFound);
}
}
[HttpDelete]
public void Delete(int id)
{
Employee employee = repository.Get(id);
if (employee == null)
{
throw new HttpResponseException(HttpStatusCode.NotFound);
}
repository.Remove(id);
}
}
```

HTML

```
@(Html.EJ().DataManager("FlatData").URL("/api/Employee").Adaptor(AdaptorType
.WebApiAdaptor))
@(Html.EJ().Grid<Sample.Models.Employee>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['EmployeeID', 'FirstName', 'LastName'])")
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Width(75
).Add();
col.Field("FirstName").HeaderText("FirstName").Width(80).Add();
col.Field("LastName").HeaderText("LastName").Add();
})
)
@Html.EJ().Button("submit").Text("Insert").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
function onClick(e) {
var record = { EmployeeID: 6, FirstName: "ANGEL", LastName: "Reims" };
window.FlatData.insert(record);
setTimeout(function () {
var query = ej.Query();
var dataSource = window.FlatData.executeQuery(query).done(function (args) {
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(args.result);
// renderTable(args.result);
}).fail(function (args) {
//error handling
});
}, 2000);
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

EmployeeI	FirstName	LastName
1	Nancy	Davolio
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	ANGEL	Reims

Insert

Update

The update method is used to update the modified changes made to a record in the data source of the DataManager.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("/api/Employee").Adaptor(AdaptorType
.WebApiAdaptor))
@(Html.EJ().Grid<Sample.Models.Employee>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['EmployeeID', 'FirstName', 'LastName'])")
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Width(75
).Add();
col.Field("FirstName").HeaderText("FirstName").Width(80).Add();
col.Field("LastName").HeaderText("LastName").Add();
})
)
@Html.EJ().Button("submit").Text("Update").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
function onClick(e) {
var record = { EmployeeID: 1, FirstName: "Nancy", LastName: "Kamal" };
window.FlatData.update("ItemID", record);
setTimeout(function () {
var query = ej.Query();
var dataSource = window.FlatData.executeQuery(query).done(function (args) {
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(args.result);
// renderTable(args.result);
}).fail(function (args) {
//error handling
});
}, 2000);
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Employee	FirstName	LastName
2	Andrew	Fuller
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	ANGEL	Reims
1	Nancy	Kamal

Update

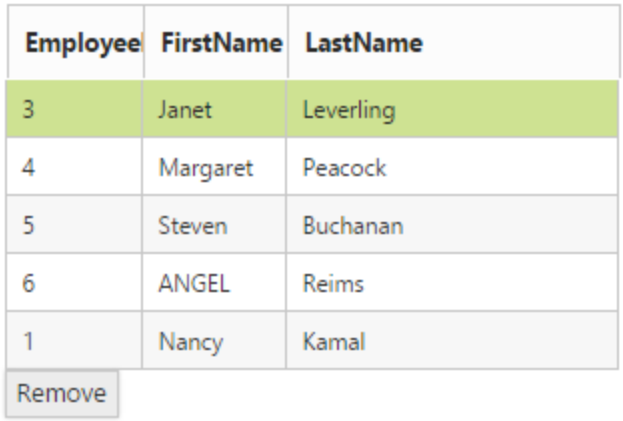
Remove

The remove action submits the data items that should be deleted, or just its IDs.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("/api/Employee").Adaptor(AdaptorType
.WebApiAdaptor))
@(Html.EJ().Grid<Sample.Models.Employee>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['EmployeeID', 'FirstName', 'LastName'])")
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Width(75
).Add();
col.Field("FirstName").HeaderText("FirstName").Width(80).Add();
col.Field("LastName").HeaderText("LastName").Add();
})
)
@Html.EJ().Button("submit").Text("Remove").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
function onClick(e) {
window.FlatData.remove("EmployeeID", 2);
setTimeout(function () {
var query = ej.Query();
var dataSource = window.FlatData.executeQuery(query).done(function (args) {
var obj = $("#FlatGrid").ejGrid("instance");
obj.dataSource(args.result);
// renderTable(args.result);
}).fail(function (args) {
//error handling
});
}, 2000);
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Employee	FirstName	LastName
3	Janet	Leverling
4	Margaret	Peacock
5	Steven	Buchanan
6	ANGEL	Reims
1	Nancy	Kamal

Remove

Batch Edit

Batch Editing is a unique feature, where requests to add, remove and change are handled altogether at a time rather than passing the request separately for each operation.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").Adaptor(AdaptorType.JsonAdaptor))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("ej.Query().take(5)")
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).TextAlig
n(TextAlign.Right).Width(75).Add();
col.Field("FirstName").HeaderText("FirstName").Width(80).Add();
col.Field("LastName").HeaderText("LastName").TextAlign(TextAlign.Right).Widt
h(75).Add();
})
)
```

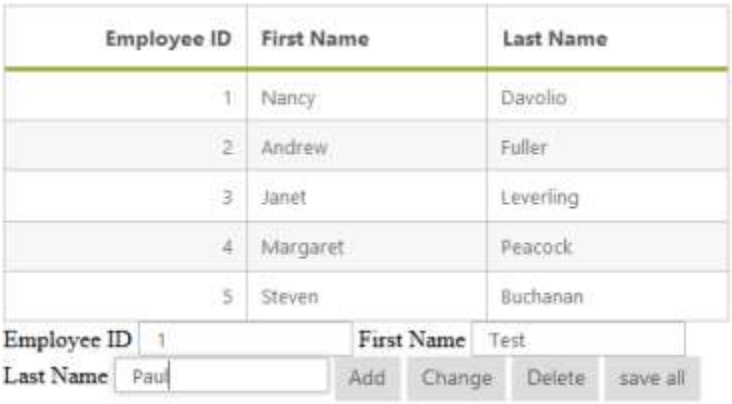
ASPX-CS

```
Employee ID
<input id="EmployeeID" class="e-ejinputtext" type="text" value="" />
First Name
<input id="FirstName" class="e-ejinputtext" type="text" value="" />
Last Name
<input id="LastName" class="e-ejinputtext" type="text" value="" />
@Html.EJ().Button("submit").Text("Add").ClientSideEvents(e => {
e.Click("onClick"); })
@Html.EJ().Button("submit").Text("Change").ClientSideEvents(e => {
e.Click("onClick"); })
@Html.EJ().Button("submit").Text("Delete").ClientSideEvents(e => {
e.Click("onClick"); })
@Html.EJ().Button("submit").Text("Save All").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
window.changes = { changed: [], added: [], deleted: [] };
function Click(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (document.activeElement.value == "Change") {
data = window.FlatData.executeLocal(ej.Query().where("EmployeeID",
ej.FilterOperators.equal, parseInt($("#EmployeeID").val(), 10)));
if (data.length) {
data[0].FirstName = $("#FirstName").val();
window.changes.changed.push(data);
}
}
else if (document.activeElement.value == "Add") {
window.changes.added.push({
EmployeeID: parseInt($("#EmployeeID").val(), 10),
FirstName: $("#FirstName").val(),
LastName: $("#LastName").val(),
});
}
else if (document.activeElement.value == "Delete") {
data = window.FlatData.executeLocal(ej.Query().where("EmployeeID",
ej.FilterOperators.equal, parseInt($("#EmployeeID").val(), 10)));
if (data.length)
window.changes.deleted.push(data[0]);
}
else {
var obj = $("#MainContent_OrdersGrid").ejGrid("instance");
window.FlatData.saveChanges(window.changes, "EmployeeID");
(window.FlatData.dataSource.json);
}
}
</script>
</asp:Content>
```

Result of the above code example is illustrated as follows.



Batch Editing

Paging

Paging is a very important query in the DataManager used to display only some records from the large data source. Here, you can learn the paging query in detail.

Default

The paging index and the paging size parameters of the paging query determine the number of records to be retrieved from the data source of the DataManager.

Copyright © 2001 - 2020 Syncfusion Inc.

Refer to the following code example for the paging options.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().page(5,5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result for the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10268	GROSR	8	Caracas	66.2900
10269	WHITC	5	Seattle	4.5600
10270	WARTH	1	Oulu	136.5400
10271	SPLIR	6	Lander	4.5400
10272	RATTC	6	Albuquerque	98.0300

Paging

Dynamic Paging

The paging operation can be dynamically performed by using the DataManager. With the help of an external button click event, the required page records can be obtained and processed accordingly. The following code example illustrates the dynamic paging.

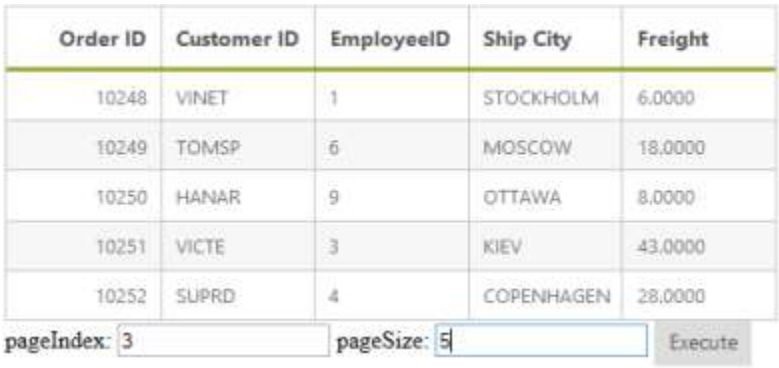
ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").CrossDomain(true))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).page(1,3)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
pageIndex: <input id="pageIndex" type="text" placeholder="pageIndex" />
pageSize:  <input id="pageSize" type="text" placeholder="pageSize" />
@Html.EJ().Button("submit").Text("Execute").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
function onClick(e) {
var from = parseInt($("#pageIndex").val());
var to = parseInt($("#pageSize").val());
var obj = $("#FlatGrid").ejGrid("instance")
tempQuery = new ej.Query();
tempQuery.page(from, to);
var dataManagerObj = window.FlatData.executeQuery(tempQuery).done(function
(e1) {
obj.dataSource(e1.result);
})
}
</script>
```

Result of above code example is illustrated as follows.



Dynamic Paging

Custom paging

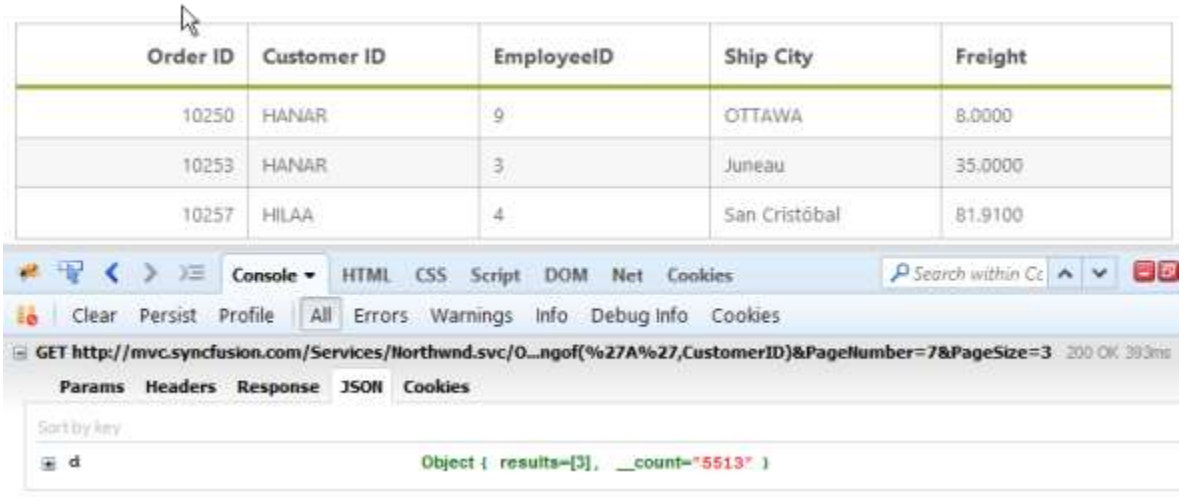
In this section, you can learn how to use the customized paging. The following code example illustrates the custom paging.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight',
'ShipCity').addParams('PageNumber',7).addParams('PageSize',3).where('Custome
rID', 'contains', 'A', false)")
.Columns(col =>
{
```

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of above code example is illustrated as follows.



Custom Paging

Skip

The skip query is used to skip some number of records.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').skip(100).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Result of the above code example is illustrated as follows.



Order ID	Customer ID	EmployeeID	Ship City	Freight
10348	WANDK	4	Stuttgart	0.7800
10349	SPLIR	7	Lander	8.6300
10350	LAMAI	6	Toulouse	64.1900

By using Skip

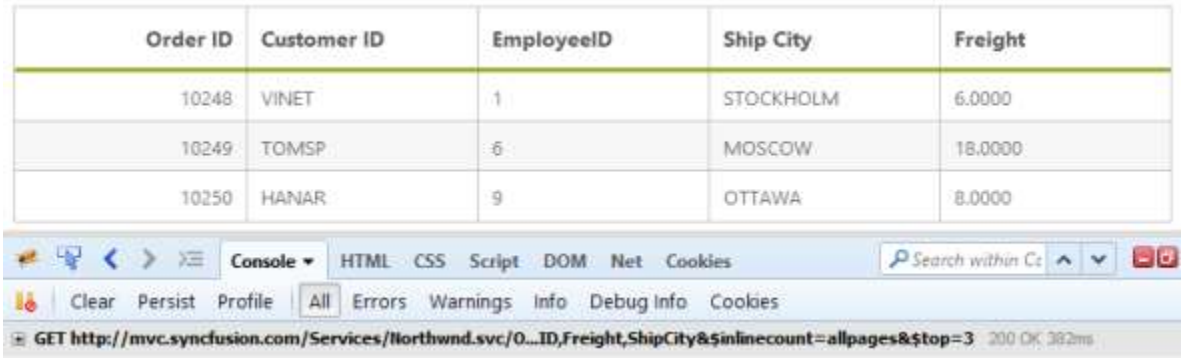
Take

The take query is used to get some number of records from the data source of the DataManager.

HTML

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').take(3)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.



Order ID	Customer ID	EmployeeID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000

By using take

Copyright © 2001 - 2020 Syncfusion Inc.

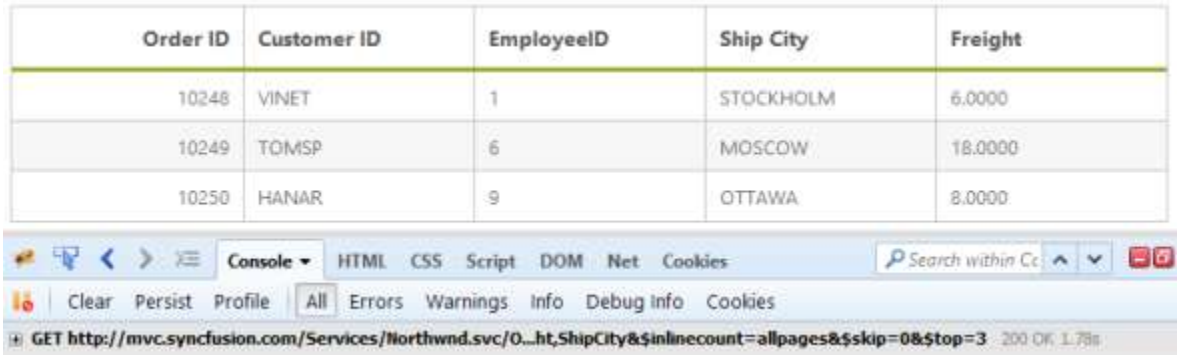
RequiresCount

The requiresCount query is used to get the count of the total number of records in the data source of the DataManager.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').page(1,3).requiresCount()")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.



By using requireCount

Range

The range query is used to get some particular range of records from the data source of the DataManager.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').range(25,30)")
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10273	QUICK	3	Cunewalde	76.0700
10274	VINET	6	Reims	6.0100
10275	MAGAA	1	Bergamo	26.9300
10276	TORTU	8	México D.F.	13.8400
10277	MORGK	2	Leipzig	125.7700



By using range

Summary

Summary is a key feature in the DataManager that helps to aggregate any data. DataManager provides several summary type by default, they are as follows.

- Sum
- Average
- Minimum
- Maximum
- Distinct

The ej provides several data utilization methods to achieve summary.

Sum

The Sum summary type provides the sum of the data. The Sum data utilization method accepts two parameters, they are JSON data and the fieldname where the sum is calculated. The following code example illustrates the Default Summary Types.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
```

```
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').range(25,30)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
setTimeout(function () {
var proxy = $("#FlatGrid").ejGrid("instance");
var data = proxy._currentJsonData;
var sum = ej.sum(data, "EmployeeID");//Calculates the sum Freight
$("body").append("<span>Sum:" + sum + "</spn>");
}, 3000);
</script>
```

The result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10273	QUICK	3	Cunewalde	76.0700
10274	VINET	6	Reims	6.0100
10275	MAGAA	1	Bergamo	26.9300
10276	TORTU	8	México D.F.	13.8400
10277	MORGK	2	Leipzig	125.7700

Sum:20

Summary - Sum

Min

The Minimum of a particular field can be calculated by using the ej.min data utilization method and this method accepts the arguments such as JSON data/array, field name and the comparer used for the comparison. When the data to the min method is a JSON array, then the whole record is returned.

The minimum of particular field can be calculated as follows.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').range(25,30)")
.Columns(col =>
{
```

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
setTimeout(function () {
var proxy = $("#FlatGrid").ejGrid("instance");
var data = proxy._currentJsonData;
var min = ej.min(data, "EmployeeID");//Calculates the min Freight
$("body").append("<span>Min:" + min.EmployeeID + "</spn>");
}, 3000);
</script>
```

The result of the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10273	QUICK	3	Cunewalde	76.0700
10274	VINET	6	Reims	6.0100
10275	MAGAA	1	Bergamo	26.9300
10276	TORTU	8	México D.F.	13.8400
10277	MORGK	2	Leipzig	125.7700

Min:1

Summary- Minimum

Max

The Maximum of a particular field can be calculated by using the ej.max data utilization method and this method accepts the arguments such as JSON data/array, field name and the comparer used for the comparison. When the data to the max method is a JSON array, then the whole record is returned.

The maximum of particular field can be calculated as follows.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').range(25,30)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<script type="text/javascript" class="jsScript">
setTimeout(function () {
var proxy = $("#FlatGrid").ejGrid("instance");
var data = proxy._currentJsonData;
var max = ej.max(data, "EmployeeID"); //Calculates the max Freight
$("body").append("<span>Max:" + max.EmployeeID + "</spn>");
}, 3000);
</script>
```

The result for the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10273	QUICK	3	Cunewalde	76.0700
10274	VINET	6	Reims	6.0100
10275	MAGAA	1	Bergamo	26.9300
10276	TORTU	8	México D.F.	13.8400
10277	MORGK	2	Leipzig	125.7700

Max:8

Summary - Maximum

Avg

The Average summary type provides the average of the given data. The Average data utilization method accepts two parameters, they are JSON/Array data and the fieldname where the sum is calculated. Use the following code example for calculating the average of the given JSON data.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').range(25,30)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
<script type="text/javascript" class="jsScript">
setTimeout(function () {
var proxy = $("#FlatGrid").ejGrid("instance");
var data = proxy._currentJsonData;
var avg = ej.avg(data, "EmployeeID");//Calculates the avg Freight
$("body").append("<span>Avg:" + avg + "</spn>");
}, 3000);
</script>
```

The result of the above code example is illustrated as follows.



Order ID	Customer ID	EmployeeID	Ship City	Freight
10273	QUICK	3	Cunewalde	76.0700
10274	VINET	6	Reims	6.0100
10275	MAGAA	1	Bergamo	26.9300
10276	TORTU	8	México D.F.	13.8400
10277	MORGK	2	Leipzig	125.7700

Avg:4

Summary - Average

Distinct

In a data, a field may contain many duplicate values; and sometimes you are required only to list the different (distinct) values. This can be achieved by using the ej.distinct method. This method accepts three parameters such as JSON/Array data, fieldname that you want to fetch as distinct and the third Boolean parameter set as true, returns the whole record when the data is a JSON array.

The following code example illustrates how to use the ej.distinct method. In the following code, the third parameter of distinct method is set as true and hence it fetches the whole record from the provided data.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select('OrderID', 'CustomerID', 'EmployeeID',
'Freight', 'ShipCity').range(25,35)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
<script type="text/javascript" class="jsScript">
setTimeout(function () {
var proxy = $("#FlatGrid").ejGrid("instance");
var data = proxy._currentJsonData;
var distinct = ej.distinct(data, "EmployeeID", true);//Calculates the
distinct
proxy.dataSource(distinct);
}, 3000);
</script>
```

The result for the above code example is illustrated as follows.

Order ID	Customer ID	EmployeeID	Ship City	Freight
10273	QUICK	3	Cunewalde	76.0700
10274	VINET	6	Reims	6.0100
10275	MAGAA	1	Bergamo	26.9300
10276	TORTU	8	México D.F.	13.8400
10277	MORGK	2	Leipzig	125.7700
10281	ROMEY	4	Madrid	2.9400

Summary - Distinct

Hierarchical Query

The DataManager contains support to manage the hierarchical query. The hierarchical queries are commonly required when you use foreign key binding. The hierarchical query can be provided by using the hierarchical function. This method accepts two parameters such as the query and the selector function.

foreignKey

The foreignkey method of ej.Query can be used to refer to another table fields. The foreignkey method accepts one parameter, the foreign key value.

The following code example illustrates the hierarchical query and foreignkey method.

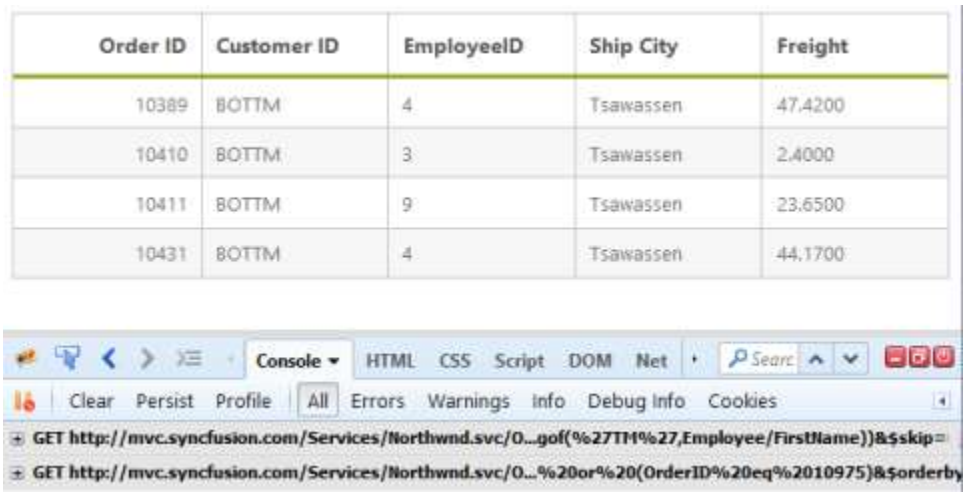
ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc"))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("ej.Query().from('Orders').search('TM', ['CustomerID',
'Employee.FirstName']).page(1,
4).hierarchy(ej.Query().foreignKey('OrderID').from('Order_Details').sortBy('
Quantity'),function () {return [10410, 10492, 10949, 10742, 10975]})")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The result for the above code example is illustrated as follows.



Hierarchical Query and foreignkey method

Helper Functions

ej.parseJSON

This method is used to parse the string to **JSON**.

ej.parseTable

This method is used to parse the **HTML** element into the **JSON** Array and is used commonly in element binding using **DataManager**.

ej.getGuid

This method returns the globally unique identifier and it generates unique identifier with the prefix provided as parameter.

ej.merge

The ej.merge is used to merge two arrays and the result is merged in the first array.

ej.support

The ej.support property contains a collection of properties representing different browser features or bugs. The property are as follows.

1. cors – represents the cross browser support.
2. enableLocalizedSort – enables the localized sort operation.
3. stableSort – enables stable sort operation.
4. outerHTML – represents the outerHTML support.

Copyright © 2001 - 2020 Syncfusion Inc.

ej.swap

This method is used to swap the position of element in an array. It accepts three arguments such as input array and two swap positions.

ej.serverTimezoneOffset

This property is used to set the time-zone offset from UTC, in hours

Note: Suppose, application has been hosted in EST time zone and client may be in IST, GMT etc. in that case, date will differ from database. The below given solution will fix the conflict.

Property Name: ej.serverTimezoneOffset Data type: number Default Value: 0

Note: By default, The server time zone will be in UTC, if its other than the UTC, set the proper format time zone offset value to the ej.serverTimezoneOffset property.

Please refer the online (link)[https://en.wikipedia.org/wiki/Time*zone#List*of*UTC*offsets] for time zone offset values.

Time-zone offset calculation from UTC: ej.serverTimezoneOffset = serverTimeZoneDifference(in hours) + ClientSideTimeZoneDiff(in hours);

HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div id="Grid"></div>
</div>
</div>
</div>
<script type="text/javascript">
var serverTimeZoneDifference = -5.0   // if your server is in EST time zone
(UTC -5.0) (in hours)
var serverTimeZoneDifference = new Date().getTimezoneOffset() / 60; // get
client time zone differences and convert it to hours;
ej.serverTimezoneOffset = serverTimeZoneDifference +
serverTimeZoneDifference;
$(function () {
var dataManagerObj = ej.DataManager({ url:
"http://mvc.syncfusion.com/services/Northwnd.svc/Orders" });
$("#Grid").ejGrid({
dataSource: dataManagerObj,
allowPaging: true,
columns: ["OrderID", "EmployeeID", "CustomerID", "OrderDate", "Freight"]
});
});
</script>
```

Note: When the Daylight saving time mode is enabled in the sever, take care about this time adjustment with the server time-zone offset calculation.

Copyright © 2001 - 2020 Syncfusion Inc.

The countries that are using Eastern Standard Time (EST) when observing standard time (autumn/winter) are 5 hours behind Coordinated Universal Time (UTC−05:00).

Eastern Daylight Time (EDT), when observing daylight saving time DST (spring/summer) is 4 hours behind Coordinated Universal Time (UTC−04:00).

Let see an example, when they observe the Daylight saving time.

HTML

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<div id="Grid"></div>
</div>
</div>
</div>
<script type="text/javascript">
var serverTimeZoneDifference = -4.0  // if your server is in EDT time zone
(UTC -4.0) (in hours)
var serverTimeZoneDifference = new Date().getTimezoneOffset() / 60; // get
client time zone differences and convert it to hours;
ej.serverTimezoneOffset = serverTimeZoneDifference +
serverTimeZoneDifference;
$(function () {
var dataManagerObj = ej.DataManager({ url:
"http://mvc.syncfusion.com/services/Northwnd.svc/Orders" });
$("#Grid").ejGrid({
dataSource: dataManagerObj,
allowPaging: true,
columns: ["OrderID", "EmployeeID", "CustomerID", "OrderDate", "Freight"]
});
});
</script>
```

KO ResultSet

The DataManager contains a default method to subscribe the view model properties as KO observable. This is done at the success of the executeQuery by using the getKnockoutModel. You can also provide computed properties to the view model by using the getKnockoutModel.

The following code example illustrates how the model is made observable and updated.

ASPX-CS

```
<div class="datatable" style="padding:10px">
<div class="row">
<div class="col-md-7">
<table id="table1" class="table table-striped table-bordered"
style="width:700px">
<thead>
<tr>
<th>EmployeeID</th>
<th>Full Name</th>
</tr>
</thead>
<tbody data-bind="foreach: employees">
<tr>
```

```
<td data-bind="text: EmployeeID"></td>
<td data-bind="text: FullName"></td>
</tr>
</tbody>
</table>
</div>
<div class="col-md-5">
<form class="form-horizontal" role="form">
<div class="form-group">
<label class="col-sm-4 control-label">EmployeeID</label>
<div class="col-sm-8">
<input type="text" class="form-control" id="employeeId">
</div>
</div>
<div class="form-group">
<label class="col-sm-4 control-label">FirstName</label>
<div class="col-sm-8">
<input type="text" class="form-control" id="first">
</div>
</div>
<div class="form-group">
<label class="col-sm-4 control-label">LastName</label>
<div class="col-sm-8">
<input type="text" class="form-control" id="last">
</div>
</div>
<div class="form-group">
<div class="col-sm-offset-4 col-sm-4">
<button type="button" id="formSubmit" class="btn btn-
default">Change</button>
</div>
</div>
</form>
</div>
</div>
</div>
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.dataso
urce).Adaptor(AdaptorType.JsonAdaptor))
<script src="~/Scripts/knockout.min.js"></script>
<script src="~/Scripts/ej/ej.widget.ko.min.js"></script>
<script type="text/javascript">
setTimeout( function (){
window.pageModel = {
employees: []
};
var query = ej.Query();
var promise = window.FlatData.executeQuery(query);
promise.done(function (e) {
pageModel.employees = e.getKnockoutModel({
FullName: function () {
return this.FirstName() + " " + this.LastName();
}
});
ko.applyBindings(pageModel);
});
}, 1000);
$("#formSubmit").click(function (e) {
```

```
var employeeId = parseInt($("#employeeId").val(), 10);
var fName = $("#first").val();
var lName = $("#last").val();
employee = window.pageModel.employees()[employeeId - 1];
employee.FirstName(fName);
employee.LastName(lName);
});
</script>
```

The result of the above code example is illustrated as follows.

Before changing the model, EmployeeID 1 has FullName value as Nancy Davolio. After changing, the result is as follows.



EmployeeID	Full Name
1	Kate Beckett
2	Andrew Fuller
3	Janet Leverling
4	Margaret Peacock
5	Steven Buchanan
6	Michael Suyama

Knockout ResultSet

Table Model

The **DataManager** contains a default support to bind a TableModel to the element. You can make the data observable using the **getTableModel** method. The **getTableModel** method also accept extra properties or properties with computed value that can be added to the TableModel. In the view, you can create a simple view by using the bindings **getTableModel**

Then the model is bound with the element using the **bindTo.**

ASPX-CS

```
<div class="datatable" style="padding:10px">
<div class="row">
<div class="col-md-7">
<table id="table1" class="table table-striped table-bordered"
style="width:700px">
<thead>
<tr>
<th>EmployeeID</th>
<th>Full Name</th>
</tr>
</thead>
<tbody id="tbody1">
<tr>
<td ej-observe="EmployeeID"></td>
<td ej-computed="FullName"></td>
</tr>
</tbody>
</table>
</div>
```

```
<div class="col-md-5">
<form class="form-horizontal" role="form">
<div class="form-group">
<label class="col-sm-4 control-label">EmployeeID</label>
<div class="col-sm-8">
<input type="text" class="form-control" id="employeeId">
</div>
</div>
<div class="form-group">
<label class="col-sm-4 control-label">FirstName</label>
<div class="col-sm-8">
<input type="text" class="form-control" id="first">
</div>
</div>
<div class="form-group">
<label class="col-sm-4 control-label">LastName</label>
<div class="col-sm-8">
<input type="text" class="form-control" id="last">
</div>
</div>
<div class="form-group">
<div class="col-sm-offset-4 col-sm-4">
<button type="button" id="formSubmit" class="btn btn-
default">Change</button>
</div>
</div>
</form>
</div>
</div>
</div>
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.dataso
urce).Adaptor(AdaptorType.JsonAdaptor))
<script type="text/javascript">
setTimeout(function () {
window.employees = [];
var query = ej.Query();
var promise = window.FlatData.executeQuery(query);
promise.done(function (e) {
employees = e.getTableModel({
FullName: {
value: function () {
return this.FirstName + ' ' + this.LastName;
},
deps: ["FirstName", "LastName"]
}
});
employees.bindTo($("#tbody1"));
});
}, 1000);
$("#formSubmit").click(function (e) {
var employeeId = parseInt($("#employeeId").val(), 10);
var fName = $("#first").val();
var lName = $("#last").val();
employee = employees.get(employeeId - 1);
employee.set("FirstName",fName);
employee.set("LastName",lName);
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

The result for the above code example is illustrated as follows.



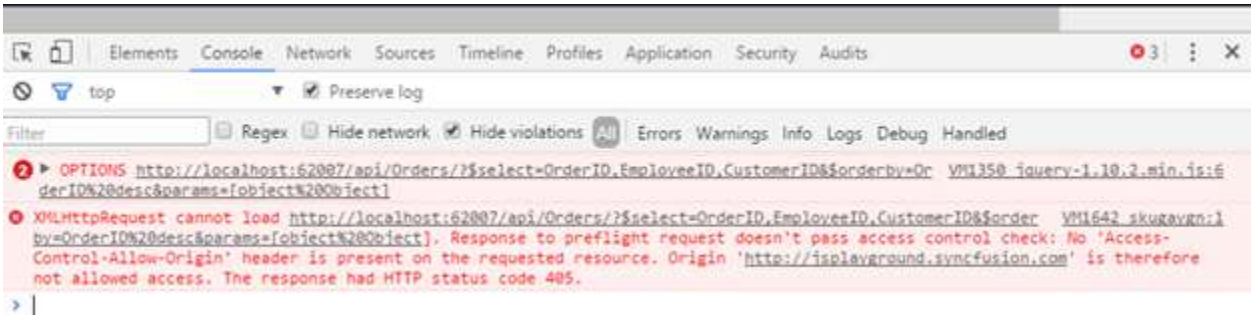
EmployeeID	Full Name
1	Kate Beckett
2	Andrew Fuller
3	Janet Leverling
4	Margaret Peacock
5	Steven Buchanan
6	Michael Suyama

CORS Data Fetching

The Cross-Origin Resource Sharing (CORS) mechanism enable secure cross-domain data transfers.

SAME-ORIGIN POLICY

This policy defined the rule for, “how a web page can be accessed by an external resource”. Browsers wont permits a request to access the resources who origin is differ than the current page. To overcome the restriction of same-origin policy, we can use a technique called Cross-origin resource sharing.



Access-Control-Allow-Origin

Cross-domain requests require common procedure between the Web page and the server. Initiate a cross-domain request in a web page and opening a connection to the mentioned domain. The browser will request the service (i.e data) from the domain’s server by sending an Origin header with the value of the origin. It will only complete the connection if the server responds with an Access-Control-Allow-Origin header of either * or the exact URL of the requesting page.

When you set, Access-Control-Allow-Origin value as “Testing1.com”.

With this rule, only scripts that originate from http://“Testing1.com are allowed to load resources. Any other domain trying to use AJAX to load resources will be given the security error message.

By using this rule, the domain owners can restrict which domains are allowed to use the resource.

Alternatively, the owners can grant wide-open access with the always ready to party asterisk: **Access-Control-Allow-Origin: ***.

HTML

```
<system.webServer>
<httpProtocol>
```

```
<customHeaders>
<add name="Access-Control-Allow-Origin" value="*" />
</customHeaders>
</httpProtocol>
</system.webServer>
```

Now, any website that wants to load a resource using AJAX can do so without getting the security error.

Access-Control-Request-Method

The Access-Control-Request-Method is used when issuing a preflight request to let the server know what HTTP method will be used when the request is made.

HTML

```
<httpProtocol>
<customHeaders>
<add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE,
OPTIONS" />
</customHeaders>
</httpProtocol>
```

Note:

Preflightedrequests:

Setting custom headers to XHR triggers a preflight request. With simple words this mean that preflight request first sends an HTTP request by the OPTIONS method to the resource on the remote domain, to make sure that the request is safe to send. According W3C for non-same origin requests using the HTTP GET method a preflight request is made when headers other than Accept and Accept-Language are set.

Access-Control-Request-Headers

The Access-Control-Request-Headers header is used when issuing a preflight request to let the server know what HTTP headers will be used when the request is made.

HTML

```
<httpProtocol>
<customHeaders>
<add name="Access-Control-Allow-Headers" value="*" />
</customHeaders>
</httpProtocol>
```

For more details, please refer to this link:

Advanced Functionalities

Offline Support

Offline support allows data-bound Syncfusion UI widgets to function without active server connection. Users can continue working with the data.

With offline as true, the DataManager requests the server only once and further data manipulation operation can be done at client side itself.

In the following code example, the offline property of the DataManager is set as true.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Offline(true))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The result of the above code example is illustrated as follows.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Figure offline mode

Load on demand

Load on demand is powerful technique to reduce band width size of consuming data. It allow you to retrieve the required range of data alone from the server and this feature helps you when the server contains large amount of data.

You can use the following code example for implementing load on demand using DataManager.

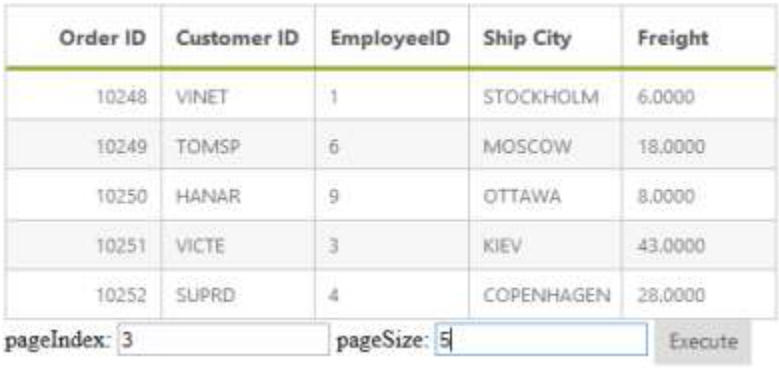
ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").CrossDomain(true))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).page(1,3)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
pageIndex: <input id="pageIndex" type="text" placeholder="pageindex" />
pageSize:  <input id="pageSize" type="text" placeholder="pageSize" />
@Html.EJ().Button("submit").Text("LoadOnDemand").ClientSideEvents(e => {
e.Click("onClick"); })
<script type="text/javascript" class="jsScript">
function onClick(e) {
var from = parseInt($("#pageIndex").val());
var to = parseInt($("#pageSize").val());
var obj = $("#FlatGrid").ejGrid("instance")
tempQuery = new ej.Query();
tempQuery.page(from, to);
var dataObj = window.FlatData.executeQuery(tempQuery).done(function (e1) {
obj.dataSource(e1.result);
})
}
</script>
```

The result of the above code example is illustrated as follows.



Load on demand

The request and the response for the above code are sent as follows.



Demanded data

Custom Request Headers

You can add custom request headers using **DataManager** and the headers can be added to the request headers in three ways that is illustrated in the following code example.

Adding Custom Request Headers to every Request using headers

You can add custom request headers to every request made by the **DataManager** using the headers property. Refer to the following code example for setting the custom request headers using the headers property.

C#

```
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.data = DataSource;
return View();
}
```

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").Json((IEnumerable<object>)ViewBag.data).
Headers(new { myData = 3232323 }))
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The above method generates the request header with custom header as follows.



*Adding Custom Request Headers to every Request using pre-request callback **beforeSend***

You can set the custom headers using pre-request callback **beforeSend** as follows. The **setRequestHeader** method can be used to modify the **XMLHTTPRequest**.

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public static List<OrderDetails> order = new List<OrderDetails>();
public ActionResult DataSource()
{
int code = 10000;
for (int i = 1; i < 10; i++)
{
order.Add(new OrderDetails(code + 1, "ALFKI", i + 0, 2.3 * i, "Berlin"));
order.Add(new OrderDetails(code + 2, "ANATR", i + 2, 3.3 * i, "Madrid"));
order.Add(new OrderDetails(code + 3, "ANTON", i + 1, 4.3 * i,
"Cholchester"));
order.Add(new OrderDetails(code + 4, "BLONP", i + 3, 5.3 * i, "Marseille"));
order.Add(new OrderDetails(code + 5, "BOLID", i + 4, 6.3 * i, "Tsawassen"));
code += 5;
}
var list = order.ToList();
DataResult result = new DataResult();
result.result = list;
result.count = list.Count;
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable<OrderDetails> result { get; set; }
public int count { get; set; }
}
public class OrderDetails
{
public OrderDetails()
{
}
public OrderDetails(int OrderID, string CustomerId, int EmployeeId, double
Freight, string ShipCity)
{
this.OrderID = OrderID;
this.CustomerID = CustomerId;
this.EmployeeID = EmployeeId;
this.Freight = Freight;
this.ShipCity = ShipCity;
}
public int? OrderID { get; set; }
public string CustomerID { get; set; }
public int? EmployeeID { get; set; }
public double? Freight { get; set; }
public string ShipCity { get; set; }
}
```

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("DataSource"))
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

JAVASCRIPT

```
<script type="text/javascript">
setTimeout(function () {
var customAdaptor = new ej.UrlAdaptor().extend({
beforeSend: function (request, settings) {
settings.setRequestHeader("myData1", "Syncfusion");
settings.setRequestHeader("myData2", 23243);
}
});
window.FlatData.adaptor = new customAdaptor();
var gridObj = $("#FlatGrid").ejGrid("instance");
gridObj.dataSource(window.FlatData.executeQuery(new ej.Query().take(7)));
}, 3000);
</script>
```

The above method generates the request header with custom header as follows.



*Adding Custom Request Headers using **addParams** method*

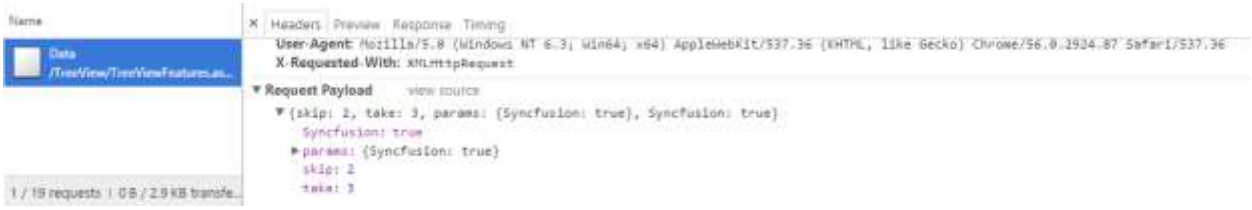
You can use the addParams method of ej.Query class, to add custom parameter to the data request.

ASPX-CS

```
@Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/N
orthwnd.svc/Orders/").Adaptor(AdaptorType.ODataAdaptor).CrossDomain(true)
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5).addParams('Syncfusion', true)")
.Columns(col =>
{
```

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

The custom parameter will be passed along with the data request of the grid as follows.



Cross domain & JSONP

The **DataManager** contains support for creating cross domain request, you can achieve this by using crossDomain and jsonp property of the **DataManager**. The following code example illustrate on how to create cross domain request.

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Services/
Northwnd.svc/Orders/").CrossDomain(true))
@(Html.EJ().Grid<MVCdoc.OrdersView>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(5)")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

Result of above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	1	STOCKHOLM	6.0000
10249	TOMSP	6	MOSCOW	18.0000
10250	HANAR	9	OTTAWA	8.0000
10251	VICTE	3	KIEV	43.0000
10252	SUPRD	4	COPENHAGEN	28.0000

Cross domain

HTML Table

Other than **JSON** and **Remote** datasource, the **DataManager** can also fetch and use data from **HTML** element. You can achieve this by using the **table** property of the **DataManager**. The **DataManager** can fetch data from the **HTML** table element.

Refer to the following code example for the **HTML** element binding using **DataManager**.

ASPX-CS

```
<script id="_table1" type="text/template">
<table id="datasource" style="display:none">
<thead>
<tr>
<th>OrderID</th>
<th>EmployeeID</th>
<th>CustomerID</th>
</tr>
</thead>
<tbody>
<tr><td>10248</td><td>VINET</td><td>5</td></tr>
<tr><td>10249</td><td>TOMSP</td><td>6</td></tr>
<tr><td>10250</td><td>HANAR</td><td>4</td></tr>
<tr><td>10251</td><td>VICTE</td><td>3</td></tr>
<tr><td>10252</td><td>SUPRD</td><td>4</td></tr>
</tbody>
</table>
</script>
@(Html.EJ().DataManager("FlatData").Table("#_table1"))
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
})
)
```

The result of the above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID
10248	VINET	5
10249	TOMSP	6
10250	HANAR	4
10251	VICTE	3
10252	SUPRD	4

HTML Table element binding

WebAPI Batch Edit

WebAPI Batch Editing is a unique feature, where requests to add, remove and change are handled altogether at a time rather than passing the request separately for each operation.Use the Save changes button to process all changes at once and cancel button to discard the changes.

Configuration

Below are listed the steps for you to follow when configuring the DataManager for ASP.NET MVC to do batch updates.

In order to using batch in Web API, you should register a route with a batch handler or create an OWIN server. The below code creates the server and configures a web API route and a batch route.

C#

```
using Microsoft.Owin;
using Owin;
using System.Web.Http;
using System.Web.Http.Batch;
[assembly: OwinStartupAttribute(typeof(Sample.Startup))]
namespace Sample
{
public partial class Startup
{
public void Configuration(IAppBuilder app)
{
ConfigureAuth(app);
HttpConfiguration config = new HttpConfiguration();
config.MapHttpAttributeRoutes();
HttpServer server = new HttpServer(config);
config.Routes.MapHttpBatchRoute(
routeName: "batch",
routeTemplate: "api/batch",
batchHandler: new DefaultHttpBatchHandler(server));
app.UseWebApi(config);
app.UseStaticFiles();
}
}
}
```

Web API uses route templates to determine which controller and action method to execute. At least one route template must be added into route table in order to handle various HTTP requests.

Copyright © 2001 - 2020 Syncfusion Inc.

Route (WebApiConfig.cs):

C#

```
config.Routes.MapHttpRoute(
name: "api",
routeTemplate: "api/{controller}/{id}",
defaults: new { id = RouteParameter.Optional }
);
```

Add a new class to the ~/Models folder. Name it Employee.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace Sample.Models
{
interface IEmployeeRepository
{
IEnumerable<Employee> GetAll();
Employee Get(int EmployeeID);
Employee Add(Employee employee);
void Remove(int EmployeeID);
bool Update(Employee employee);
}
public class Employee
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public Employee()
{
}
public Employee(int id, string LN, string FN)
{
this.EmployeeID = id;
this.LastName = LN;
this.FirstName = FN;
}
}
}
```

Add a new Repository Data Model class. Name it EmployeeRepository.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace Sample.Models
{
public class EmployeeRepository : IEmployeeRepository
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
private List<Employee> employee = new List<Employee>();
public EmployeeRepository()
{
employee.Add(new Employee(1, "Davolio", "Nancy"));
employee.Add(new Employee(2, "Fuller", "Andrew"));
employee.Add(new Employee(3, "Leverling", "Janet"));
employee.Add(new Employee(4, "Peacock", "Margaret"));
employee.Add(new Employee(5, "Buchanan", "Steven"));
}
public IEnumerable<Employee> GetAll()
{
return employee;
}
public Employee Get(int id)
{
return employee.Find(p => p.EmployeeID == id);
}
public Employee Add(Employee eObj)
{
if (eObj == null)
{
throw new ArgumentNullException("eObj");
}
employee.Add(eObj);
return eObj;
}
public void Remove(int id)
{
employee.RemoveAll(p => p.EmployeeID == id);
}
public bool Update(Employee eObj)
{
if (eObj == null)
{
throw new ArgumentNullException("eObj");
}
int index = employee.FindIndex(p => p.EmployeeID == eObj.EmployeeID);
if (index == -1)
{
return false;
}
employee.RemoveAt(index);
employee.Add(eObj);
return true;
}
}
}
```

Add a new ApiController name it as EmployeeController.cs and add a new Post/Put/Delete method which will return the Employee as JSON. The DataManager will make requests to this action based on route name.

C#

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using Sample.Models;
using System.Web;
using System.Threading.Tasks;
namespace Sample.Controllers
{
public class EmployeeController : ApiController
{
static readonly IEmployeeRepository repository = new EmployeeRepository();
[Route("get")]
public async Task<object> Get()
{
await Task.Delay(1000);
var queryString = HttpContext.Current.Request.QueryString;
var data = repository.GetAll().ToList();
return new { Items = data, Count = data.Count() };
}
public Employee GetEmployee(int id)
{
Employee employee = repository.Get(id);
if (employee == null)
{
throw new HttpResponseException(HttpStatusCode.NotFound);
}
return employee;
}
[Route("post")]
public async Task<IHttpActionResult> Post([FromBody] Employee employee)
{
if (employee == null)
{
return BadRequest(ModelState);
}
employee = repository.Add(employee);
return Content(HttpStatusCode.OK, employee);
}
[Route("put")]
public async Task<IHttpActionResult> Put([FromBody] Employee employee)
{
await Task.Delay(1000);
if (employee == null)
{
return BadRequest(ModelState);
}
else
{
if (!repository.Update(employee))
{
throw new HttpResponseException(HttpStatusCode.NotFound);
}
return Content(HttpStatusCode.OK, employee);
}
}
[Route("delete/{id}")]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public async Task<IHttpActionResult> Delete(int id)
{
Employee employee = repository.Get(id);
await Task.Delay(1000);
if (employee == null)
{
return NotFound();
}
else
{
repository.Remove(id);
return StatusCode(HttpStatusCode.OK);
}
}
}
}
```

In the view, configure the Data manager to use the Post/Put/Delete methods created in the previous steps.

C#

```
@(Html.EJ().DataManager("FlatData").BatchURL("/api/batch").UpdateURL("/put")
.RemoveURL("/delete").InsertURL("/post").Adaptor(AdaptorType.WebApiAdaptor))
@(Html.EJ().Grid<Sample.Models.Employee>("FlatGrid")
.DataManagerID("FlatData")
.Query("new ej.Query().select(['EmployeeID', 'FirstName', 'LastName'])")
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.Batch);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("EmployeeID").IsPrimaryKey(true).Width(75
).Add();
col.Field("FirstName").HeaderText("FirstName").Width(80).Add();
col.Field("LastName").HeaderText("LastName").Add();
})
)
```

Run the sample and perform batch operation. You can notice that the changes are reflected in the Database Table too.

Copyright © 2001 - 2020 Syncfusion Inc.

IEnumerable

IEnumerable will execute **select** query on server side, load data in-memory on client side and then filter data.IEnumerable can be used for querying data from in-memory collections like List, Array etc.

Data Binding

The **DataManager** contains a default support to bind data from a IEnumerable list to a control.

Refer the below given code to bind data from IEnumerable list through datamanager to Grid

C#

```
public ActionResult DataSource(Syncfusion.JavaScript.DataManager
dataManagerObj)
{
IEnumerable data = OrderRepository.GetAllRecords();
var count = data.AsQueryable().Count();
DataOperations ds = new DataOperations();
data = ds.Execute(data, dataManagerObj);
return Json(new { result=data , count = count },
JsonRequestBehavior.AllowGet);
}
```

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("Home/DataSource").Adaptor(AdaptorTy
pe.UrlAdaptor))
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData") // pass the datamanager Id
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight']).take(10)") // query to be executed
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
)
```

The result of above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Freight	Ship City
10248	VINET	5	$32.38	Reims
10249	TOMSP	6	$11.61	Münster
10250	HANAR	4	$65.83	Rio de Janeiro
10251	VICTE	3	$41.34	Lyon
10252	SUPRD	4	$51.30	Charleroi
10253	HANAR	3	$58.17	Rio de Janeiro
10254	CHOPS	5	$22.98	Bern
10255	RICSU	9	$148.33	Genève
10256	WELLI	3	$13.97	Resende
10257	HILAA	4	$81.91	San Cristóbal

Filtering Sorting and Searching with IEnumerable

We can perform filtering ,sorting and searching by passing the corresponding query through datamanager.

Refer the below given code to perform these actions

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("/Home/DataSource").Adaptor(AdaptorT
ype.UrlAdaptor))
@(Html.EJ().Grid<object>("FlatGrid")
.DataManagerID("FlatData") // pass the datamanager Id
.Query("new ej.Query().sortByDesc('OrderID').sortBy('EmployeeID',
'ascending').search(4, 'EmployeeID').where('OrderID', 'lessThan', 10435,
false).take(10)") // query to be executed
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
)
```

The result of above code example is illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Freight	Ship City
10431	BOTTM	4	$44.17	Tsawassen
10430	ERNSH	4	$458.78	Graz
10427	PICCO	4	$31.29	Salzburg
10426	GALED	4	$18.69	Barcelona
10419	RICSU	4	$137.35	Genève
10418	QUICK	4	$17.55	Cunewalde
10417	SIMOB	4	$70.29	Kobenhavn
10403	ERNSH	4	$73.79	Graz
10389	BOTTM	4	$47.42	Tsawassen
10382	ERNSH	4	$94.77	Graz

IQueryable

IQueryable is derived from IEnumerable and this executes select query on server side with all filters.IQueryable can be used for querying data from out-memory like remote database, service collections.

IQueryable support in DataManager

The DataManager now allows to execute a query against a specific data source by using the QueryableDataOperations. It can perform the IQueryable server side operation such as filtering, paging and sorting.

Refer the below code for performing these actions using IQueryable in Grid

C#

```
public ActionResult DataSource(Syncfusion.JavaScript.DataManager data)
{
IQueryable<OrdersView> datasource = new NorthwindDataContext().OrdersViews;
if (data.Where != null) // for filtering
datasource = QueryableDataOperations.PerformWhereFilter(datasource,
data.Where, data.Where[0].Condition);
if (data.Sorted != null)//for sorting
datasource = QueryableDataOperations.PerformSorting(datasource,
data.Sorted);
if (data.Search != null)
datasource = QueryableDataOperations.PerformSearching(datasource,
data.Search);
if (data.Skip >= 0)//for paging
datasource = QueryableDataOperations.PerformSkip(datasource, data.Skip);
if (data.Take > 0)//for paging
datasource = QueryableDataOperations.PerformTake(datasource, data.Take);
int count = datasource.Count();
return Json(new { result = datasource.ToList(), count = count },
JsonRequestBehavior.AllowGet);
}
```

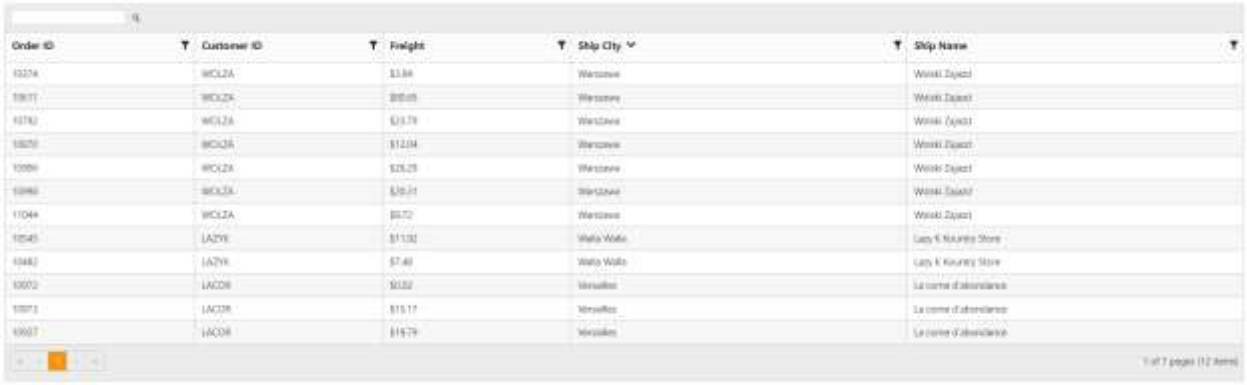
ASPX-CS

```
@(Html.EJ().Grid<OrdersView>("QueryableGrid")
.Datasource(ds =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
ds.URL("/Home/DataSource").Adaptor(AdaptorType.UrlAdaptor);
})
.AllowScrolling()
.AllowSorting()
.AllowMultiSorting()
.AllowFiltering()
.AllowSearching()
.ToolbarSettings(set1 => set1.ShowToolbar(true).ToolbarItems(new
List<ToolBarItems>() { ToolBarItems.Search }))
.AllowPaging()
.SortSettings(sort => sort.SortedColumns(col =>
col.Field("ShipCity").Direction(SortOrder.Descending).Add()))
.FilterSettings(filter => { filter.FilterType(FilterType.Menu); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").AllowSorting().Add();
col.Field("Freight").HeaderText("Freight").Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("ShipName").HeaderText("Ship Name").Add();
})
)
```

The result of above code example is illustrated as follows.



Anti-forgery Token

Anti-forgery tokens prevents anyone from submitting requests to your site while postback the data that are generated by a malicious script not generated by the actual user.

For this purpose, the input element with hidden value field and name attribute is created. The value from the input element stored in cookies. To enable the anti-forgery, set enableAntiForgery property as true.

HTML

```
<input type="hidden" name="_ejRequestVerifyToken" value="f2cd20a3-5ae1-4e19-
be61-d409191be3b1">
```

HTML

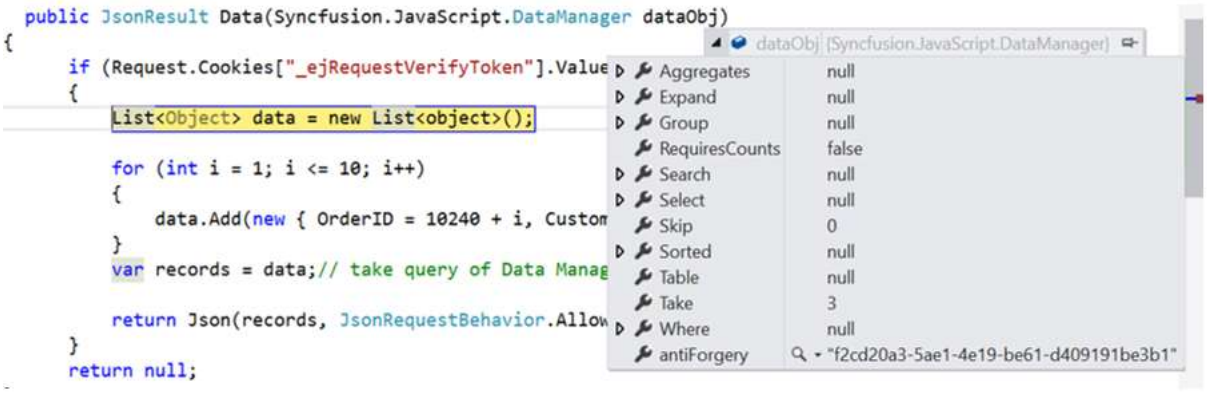
```
<div class="datatable">
```

```
<table id="table1" class="table table-striped table-bordered"
style="width:700px">
<thead>
<tr>
<th>Order ID</th>
<th>Customer ID</th>
<th>Employee ID</th>
</tr>
</thead>
<tbody></tbody>
</table>
</div>
@Html.EJ().DataManager("data").URL("Data").EnableAntiForgery(true).Adaptor(A
daptorType.UrlAdaptor).Render();
<script type="text/javascript">
$(function () {
var query = ej.Query();
var execute = window.data.executeQuery(query) // executing query
.done(function (e) {
$("#table1 tbody").html($("#tableTemplate").render(e.result));
});
});
</script>
<script id="tableTemplate" type="text/x-jsrender">
<tr>
<td>{{>OrderID}}</td>
<td>{{>CustomerID}}</td>
<td>{{>EmployeeID}}</td>
</tr>
</script>
```

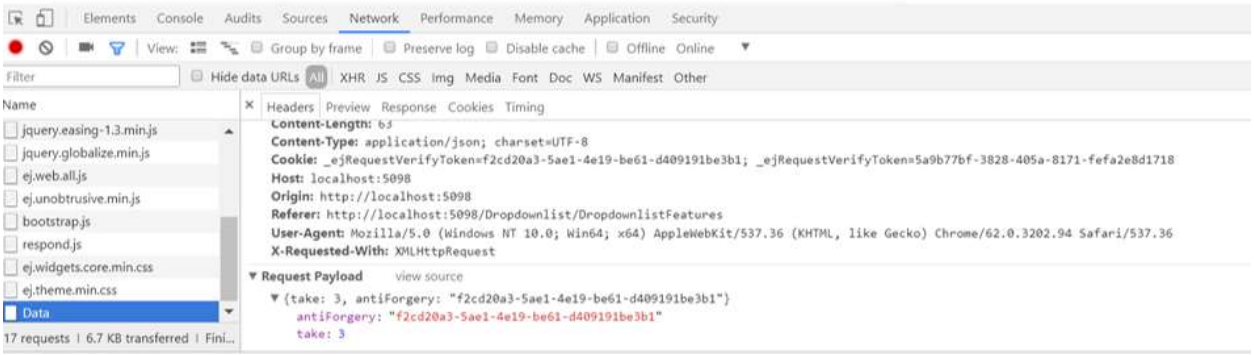
C#

```
public JsonResult Data(Syncfusion.JavaScript.DataManager dataObj)
{
if (Request.Cookies["_ejRequestVerifyToken"].Value == dataObj.antiForgery)
{
List<Object> data = new List<object>();
for (int i = 1; i <= 10; i++)
{
data.Add(new { OrderID = 10240 + i, CustomerID = "Customer" + i, EmployeeID
= i });
}
var records = data;// take query of Data Manager
return Json(records, JsonRequestBehavior.AllowGet);
}
return null;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



In the header, You can find the anti-forgery token value



DatePicker

Overview

The Essential ASP.NET MVC (EJMVC) DatePicker component provides the support for displaying calendar in web page and allows the user to pick or directly enter the date value as like EJ DatePicker, since EJMVC DatePicker is a server side wrapper of EJ DatePicker.

Key features

- Formatting the date value.
- Globalization.
- Range the date value to pick.
- Quick picking date by drill down or up.
- State persistence.
- Strongly typed HTML editor
- Responsive dimension.
- Flexible customization.
- Custom styling.
- Built-in jQuery validation.
- Accessibility (keyboard and ARIA).

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Started

A new MVC application and required assemblies with dependent files

To get start with Essential ASP.NET MVC DatePicker, create a new MVC application and add the required assemblies in references and then refer the below specified dependent CSS file as well as scripts

To create a MVC project and add necessary assemblies you can use the help of the given MVC-Getting Started Documentation.

CSS file

- [ej.web.all.min.css](http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-azure/ej.web.all.min.css) includes all widgets styles (To know more about theme refer Theming in Essential JavaScript Component)

External script files

- jQuery (from the version 1.7.1 to 3.1.0)

jQuery.easing external dependency has been removed from version 14.3.0.49 onwards. Kindly include this jQuery.easing dependency for versions lesser than 14.3.0.49 in order to support animation effects.

Internal script files

File	Description / Usage
ej.core.min.js	Includes only the widget basic functions and Framework features. Must be referred always before using all the JS controls
ej.globalize.min.js	To support the globalization.
ej.datepicker.min.js	DatePicker plugin.

Note: From V13.4.0.53 onwards, jQuery.globalize.min.js file has been replaced with our script file ej.globalize.min.js to support the globalization for our widgets. For version lower than 13.4.0.53, refer jQuery.globalize.min.js.

You can make use of **ej.web.all.min.js** file which encapsulates all EJMVC components and frameworks in single file.

- [ej.web.all.min.js](http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/ej.web.all.min.js) - includes all web widgets.

If you are using NuGet packages, then required files to render the EJMVC components, will get included in corresponding location automatically in your application.

Note: In production, we highly recommend you to use our custom script generator to create custom script file with required controls and its dependencies only. Also to reduce the file size further please use GZip compression in your server.

Copyright © 2001 - 2020 Syncfusion Inc.

Below is a simple Layout CSHTML file with required CSS and script reference added to create the Essential ASP.NET MVC DatePicker

ASPX-CS

```
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>@ViewBag.Title - My ASP.NET MVC Application</title>
<link rel="stylesheet" href="http://cdn.syncfusion.com/{{
site.releaseversion }}/js/web/flat-azure/ej.web.all.min.css" />
<script src="https://code.jquery.com/jquery-1.10.2.min.js"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"> </script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/common/ej.unobtrusive.min.js"></script>
</head>
```

DatePicker Initialization

The Essential ASP.NET MVC DatePicker can be created using HTML helper like below code example (Razor code)

ASPX-CS

```
@Html.EJ().DatePicker("datePicker")
```

Note: DatePicker is a form control, so on form submitting its value can be received by its name, and normally id has been treated as name. If you want to set the name manually you can use our HtmlAttributes property. You can get this property related information from here

Get / Set value

The Essential ASP.NET MVC DatePicker provides an options to configure all its properties and to get its value. DatePicker value can be assigned during initialization or at run time. Below code shows how to assign values at initialization.

ASPX-CS

```
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).DateFormat("y
yyy/MM/dd")
```

You can assign values to existing EJMVC DatePicker using its instance. Consider that you going to set date value at button click, refer the below code example.

JS

```
<script>
//bind below onClick action to button
function onClick() {
//create instance for datePicker.
// only after control creation we can get dateObj otherwise it throws
exception.
var dateObj = $("#datePicker").ejDatePicker('instance');
//set value using date object
dateObj.option('value', new Date());
//get value using date object and displays in alert box
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
alert(dateObj.option('value'));
}
</script>
```

Note: Existing DatePicker instance can be created by jQuery.data() and you can control the API's of DatePicker behavior.

Client Side Events

You can handle the all available client side events in Essential JavaScript DatePicker. Refer the below code example to use the client side event in EJMVC DatePicker

ASPX-CS

```
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).ClientSideEve
nts(e => e.Change("onChange"))
<script>
function onChange(args) {
//args contains entire model of DatePicker to get the value of all
properties.
//alert popup shows the previous date and selected date.
alert(" previous date is : " + args.prevDate + " \n selected date is : " +
args.value);
}
</script>
```

Strongly Typed HTML Helper

EJMVC DatePicker can be created using DatePickerFor extension, which behaves as strongly typed HTML helper. This reduce the run time issues while using model values in DatePickerFor , since it will find the issues if any in compilation time itself. Also, to know about the setting model properties to DatePickerFor control, please refer the link:

https://www.syncfusion.com/kb/4670/how-to-define-properties-for-for-controls

Please refer the below code to create the DatePickerFor control in your application

ASPX-CS

```
@Html.EJ().DatePickerFor(model => model.datepicker,
(Syncfusion.JavaScript.Models.DatePickerProperties)ViewData["date"])
@Html.ValidationMessageFor(model => model.datepicker1)
```

Behavior Settings

The Essential ASP.NET MVC DatePicker has some default behavior settings which helps to perform more operation by built-in.

Selected Date

The Essential ASP.NET MVC DatePicker value can be selected through picking date from DatePicker calendar or you can set it by using "Value" property.

EJMVC DatePicker allows only the valid date to be entered and it should be within the specified range. By default, strict mode is enabled in DatePicker, so it will restrict invalid date and resets to previous date if it's not valid. Also the valid date should be defined in specified range or else it resets to min or

Copyright © 2001 - 2020 Syncfusion Inc.

maximum date when the entered date is out of range. To know more about strict mode refer Strict Mode

Date Range

EJMVC DatePicker provides an option to restrict the user to pick the date from specified range of value. You can utilize this option by make use of 'MinDate' and 'MaxDate' property.

MinDate - specifies the minimum date to be picked in DatePicker calendar by disabling below date of minDate.

MaxDate - specifies the maximum date to be picked in DatePicker calendar by disabling above date of maxDate.

Refer below example to set min and max date range in DatePicker at initialization (You can change the **'MinDate'** and **'MaxDate'** value dynamical as like 'Value' property)

ASPX-CS

```
@*sets min and max date to be pick able in datepicker*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).MinDate("2014
/06/03").MaxDate("2014/06/19")
```

Note: You can change the **MinDate** and **MaxDate** value dynamically like **Value** property

Start and Depth Level

Start and depth represents the view of Essential ASP.NET MVC DatePicker calendar. DatePicker calendar has four different level of views, which are month, year, decade and century. It allows you to quick pick date from different months and years by navigating through different views.

By default DatePicker starts with month view and can be drill down into year, decade and century. You can also change the start and depth level view by using StartLevel and DepthLevel property. So that you can initiate DatePicker calendar to view at any level and control the navigation.

- "MonthÂ•" - shows the days in month to pick.
- "Year" - shows the months in year to pick.
- "Decade" - shows the years in decade to pick.
- "Century" - shows the decades in century to pick.

ASPX-CS

```
@*sets the start view and defines when the DatePicker calendar to return
date*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).StartLevel(Pe
riod.Year).DepthLevel(Period.Year).DateFormat("MMMM yyyy")
```

Display Inline Mode

The Essential ASP.NET MVC DatePicker provides an option to act as calendar by setting the DisplayInline property as true. In this mode the DatePicker calendar has been placed open state constantly to pick the date.

Refer below code example to represent DatePicker as calendar.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@*sets inline to represent datePicker as DatePicker calendar*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).DisplayInline
(true)
```

Strict Mode

Strict mode in Essential ASP.NET MVC DatePicker allows you to enter valid or invalid date in input textbox, but an error class will get added to exhibit if it is an invalid date. When you set EnableStrictMode to false, DatePicker allows you to enter only the valid date or else it will resets with previous value.

Also the valid date should be defined in specified range or else it resets to min or maximum date when the entered date is out of range

By default **EnableStrictMode•** is true, so you can enter any value other than date too, but an **error** class ("**e - error**") will get added to wrapper to highlight the invalid date.

ASPX-CS

```
@*sets active strict mode*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).EnableStrictM
ode(false)
```

You can also override the **error** class to highlight when invalid date is entered.

ASPX-CS

```
@*sets inactive strict mode*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).EnableStrictM
ode(true)
```

CSS

```
<style>
/* error class to highlight invalid date */
.e-error .e-input {
color: Red; /* highlights invalid date font color in input */
}
</style>
```

Formatting

Formatting is the way of displaying the date or number as string in some standard format which is based on culture specific or user need.

Below table shows the patterns to format date value.

Format Pattern	Description	Result
d	The day of the month between 1 and 31	"1" to "31"
dd	The day of the month with leading zero if required.	"01" to "31"
ddd	Abbreviated day name.	"Mon" to "Sun"

Copyright © 2001 - 2020 Syncfusion Inc.

dddd	The full day name	"Monday" to "Sunday"
M	The month of the year between 1 - 12	"1" to "12"
MM	The month of the year with leading zero if required	"01" to "12"
MMM	Abbreviated month name	"Jan" to "Dec"
MMMM	The full month name	"January" to "December"
yy	The year as a two-digit number	"99" or "08"
yyyy	The full four digit year	"1999" or "2008"

Date Format

Each culture has some specific date format. Date format defines a format or structure of the displayed date in the textbox of EJMVC DatePicker component. You can change the date format by using DateFormat property

The standard formats are listed as follows

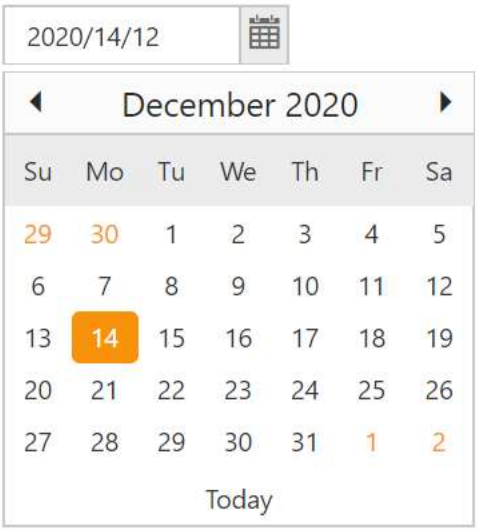
Format Name	Formats	Input Date	Formatted Date
Default	M/d/yyyy	"12/01/2020"	"12/1/2020"
Short	"d M, y"	"12/14/2020"	"14 12,20"
Medium	"d MM, y"	"12/14/2020"	"14 12, 20"
Full	dddd, d MMMM, yy	"12/14/2020"	"Monday, 14 December, 20"
UTC	yyyy-MM-dd	"12/14/2020"	"2020-12-14"

By default **en-US** culture Date format is "M/d/yyyy".

ASPX-CS

```
@*sets en-US culture with specified date format*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).Locale("en-
US").DateFormat("yyyy/dd/MM")
```

Copyright © 2001 - 2020 Syncfusion Inc.



To get the culture and date format of DatePicker, refer the below code example

JS

```
// create instance for datePicker
// only after control creation we can get dateObj otherwise it throws
exception
var dateObj = $("#datePicker").ejDatePicker('instance');
dateObj.option('locale'); //returns the culture in string
dateObj.option('dateFormat');// returns the date Format in string
```

Note: By default date format is based on culture specific. You have to refer the required culture specific files in layout page of your web application in order to localize DatePicker and customize different format for that culture. Refer this link to know more about this functionality support in DatePicker.

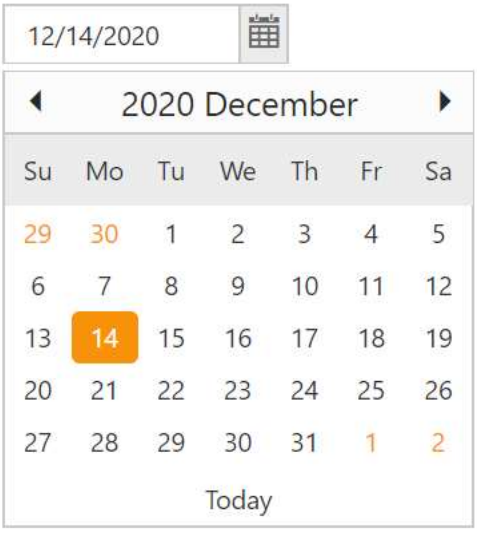
Header Format

EJMVC DatePicker calendar consists of header, day header, days and footer section. In which header section shows the current view of DatePicker calendar by displaying the selected day or month or year. It can be formatted as like date format by using HeaderFormat property.

ASPX-CS

```
@*sets the selected header format to display in header*@
@Html.EJ().DatePicker("datePicker").HeaderFormat("yyyy
MMMM").Value("12/14/2020")
```

Copyright © 2001 - 2020 Syncfusion Inc.

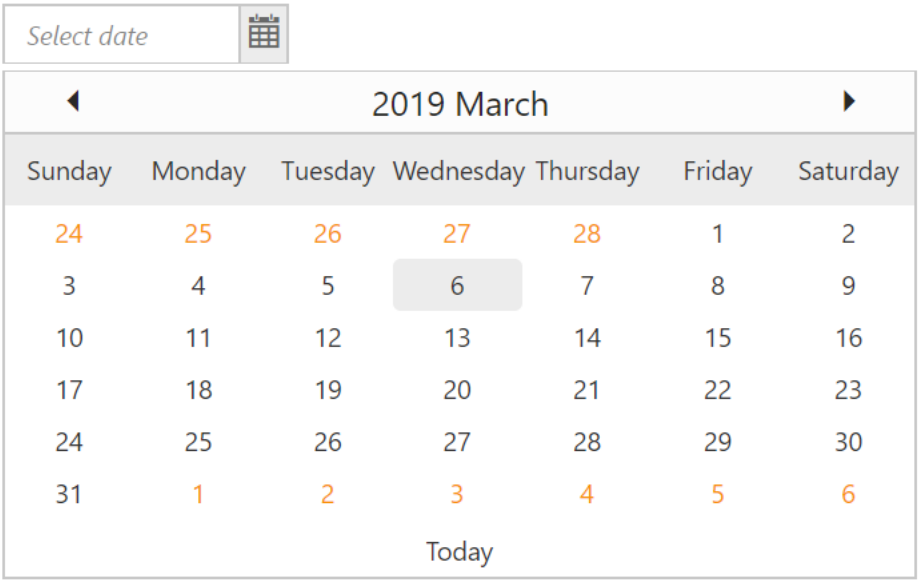


Day Header

Day header determines the days name to be displayed in terms of short, medium and long in EJMVC DatePicker calendar by using DayHeaderFormat property. Also the DatePicker calendar size varies with this specified values.

ASPX-CS

```
@*sets the day header as long*@
@Html.EJ().DatePicker("datePicker").DayHeaderFormat(Header.Long)
```



Tooltip with Formatting

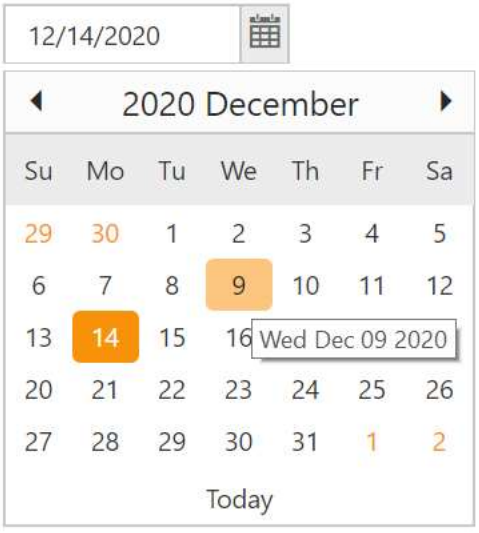
The EJMVC DatePicker calendar shows tooltip on hovering the date by specifying the formatted date of hovered date. Its helps you to get clear view about the date going to select. You can show or hide this tooltip option by using ShowTooltip property.

Copyright © 2001 - 2020 Syncfusion Inc.

You can also change the format of tooltip by using TooltipFormat property. Below codes example allows to show tooltip and format its value.

ASPX-CS

```
@*sets and shows tooltip on hovering date on calendar*@
@Html.EJ().DatePicker("datePicker").ShowTooltip(true).TooltipFormat("dd/MM/yy").Value("12/14/2020")
```



Globalization

Essential ASP.NET MVC DatePicker has been provided with built-in localization support, so that it will be able adapt based on culture specific locale defined for it.

More than 350 culture specific files are available to localize the date. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: Seven culture-specific script files are available in the below specified location. For all other culture files, please download from the GitHub location.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

To translate our control content from default English to any of the culture, say For example - German language, then you need to refer the ej.culture.de-DE.min.js file in your application,

The **en-US** locale is currently being used as default culture in DatePicker. You can set any other culture to DatePicker by using **Locale** property. Below code example shows German cultured DatePicker.

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script src="https://cdn.syncfusion.com/js/assets/i18n/ej.culture.de-DE.min.js"></script>
```

Set German culture to DatePicker at initialization.

ASPX-CS

```
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).Locale("de-DE")
```

Watermark and Today Button Text

By default the EJMVC DatePicker input has “select date” as watermark text, you can change this value by using WatermarkText property. Also there’s a today button in DatePicker calendar which allows you to quick select the current date, you can change this value by using **ButtonText** property.

ASPX-CS

```
@*sets watermark text and button text*@
@Html.EJ().DatePicker("datePicker").Value(System.DateTime.Now).WatermarkText("enter the date value").ButtonText("current date")
```

Based on culture specific, only date gets localized but by changing the watermark and button text, you can completely localize the DatePicker UI too.

Refer below code example to update those value based some culture say for example English and German.

JS

```
//create instance for datePicker.
// only after control creation we can get dateObj otherwise it throws exception.
var dateObj = $("#datePicker").ejDatePicker('instance');
//condition to check English culture and change watermark and button text using dateObj.
if (ej.cultureObject.name == "en-US") {
dateObj.option({ buttonText: "Today", watermarkText: "select date" })
}
//condition to check German culture and change watermark and button text using dateObj.
if (ej.cultureObject.name == "de-DE") {
dateObj.option({ buttonText: "heute", watermarkText: "Datum auswählen" })
}
```

Customization

EJMVC DatePicker appearance can be customized using below listed properties.

Set Dimension

By default DatePicker has standard height and width (height: "30" and width: "143"). You can change this height and width by using Height and Width property respectively.

ASPX-CS

```
@*sets height and width of the datepicker control*@
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().DatePicker("datePicker").Height("50px").Width("300px")
```

Since EJMVC DatePicker is a form control and you can make it as responsive by specifying its width as **100%**.

ASPX-CS

```
@*sets width as 100 percentage*@
@Html.EJ().DatePicker("datePicker").Width("100%")
```

Show Footer

You can show or hide the footer of EJMVC DatePicker calendar by using ShowFooter property.

ASPX-CS

```
@*hides the footer in popup calendar*@
@Html.EJ().DatePicker("datePicker").ShowFooter(false)
```

Note: Footer contains the today button to quickly navigate to the current date. By turning off it, you have to select current date by manually.

Show Popup Button

EJMVC DatePicker input textbox contains a button to open the DatePicker calendar. You can hide this DatePicker calendar button using ShowPopupButton value as false.

By hiding this button, you can open the DatePicker by focusing the input textbox element itself.

ASPX-CS

```
@*hides the popup calendar button*@
@Html.EJ().DatePicker("datePicker").ShowPopupButton(false)
```

Show Other Months

You can show or hide the other month days from EJMVC DatePicker calendar by using ShowOtherMonths property.

ASPX-CS

```
@*hides the days of other months in calendar*@
@Html.EJ().DatePicker("datePicker").ShowOtherMonths(false)
```

Highlight Special Date

The Essential ASP.NET MVC DatePicker allows you to highlight the special dates in EJMVC DatePicker calendar by using SpecialDates property. It receives the list of "dataSpecial" date fields such as, in which the data should contain the date value, icon CSS class and tooltip as field values with any names.

Special Date property has following properties,

Members	Description
Date	It specifies the mapping field of special date

Copyright © 2001 - 2020 Syncfusion Inc.

IconClass	It specifies the mapping field of icon CSS class.
Tooltip	It specifies the mapping field of tool tip text.

ASPX-CS

```
@*sets the special date in datepicker calendar*@
@Html.EJ().DatePicker("datePicker").SpecialDates(SpecialDate => {
SpecialDate.Add().Date("28/06/2015").IconClass("birthday").Tooltip("John
birthday"); })
```

Validation

EJMVC DatePicker provides very cool way to validate the value by custom rules and messages before its get processing. Enhanced APIs and built-in validation support in EJWEB DatePicker provides the quick validation in client side itself which provides the easy customization and better user experience than the validating the values in server side. Most common client side validation plugin is jQuery Validation.

DatePicker supports this validation by built-in, so you can do client validation with simple steps to validate and respond validation message. The jQuery validate plugin rules and custom methods can be used in the DatePicker control with the help of two properties,ValidationRules and ValidationMessage.

Before using those properties, you need to add the jQuery validate plugin to your HTML page.

Refer the below jQuery validation plugin script after jQuery script reference.

ASPX-CS

```
<script
src="http://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.m
in.js"></script>
```

jQuery validation library contains some default settings for validating the form. By default, it ignores hidden elements in from validation. But some of our EJ components (“Checkbox”, “MaskEdit”, ”NumericTextbox”, “CurrencyTextbox”, “PercentageTextbox”, “RTE”, “Dropdownlist”) contains hidden fields with values, these

values need to be validated at here. So to perform the validation properly, you have to set “[]” in “ignore” API of “$.validator.setDefaults”.

If validation gets fail, in-built “error” class will be added to corresponding element. Here you can specify a custom class with your own style using “errorClass” API and you can place the error message in necessary position using “errorPlacement” API. Refer following code block to customize the default jQuery validation settings.

JAVASCRIPT

```
$.validator.setDefaults({
//if we don’t set custom class, default “error” class will be added
errorClass: 'e-validation-error',
//it specifies the error message display position
errorPlacement: function (error, element) {
$(error).insertAfter(element.closest(".e-widget"));
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
```

ASPX-CS

```
<!--form to submit-->
@using (Html.BeginForm())
{
//Sets the field to be required with validation message
@Html.EJ().DatePicker("datePicker").ValidationRules(rule =>
rule.AddRule("required", true)).ValidationMessage(msg =>
msg.AddMessage("required", "Required Date value"));
<input type="submit" value="click here!" />
}
```

Note: jQuery validation works only within the form element.

State Persistence

You can sustain the entire widget model of EJMVC DatePicker even after form post or browser refresh by using EnablePersistence property. So the entire model values such as

- Selected date
- Highlighted date
- Start and depth level

are stored in local storage / cookies of browser before page refreshes and reinitialized with the restored model after refresh.

ASPX-CS

```
@*persists the DatePicker model*@
@Html.EJ().DatePicker("datePicker").EnablePersistence(true)
```

Miscellaneous

Start Day

By default EJMVC DatePicker calendar starts with "Sunday"• and ends with "Monday". You can redefine this start day by using StartDay property.

Refer below code to start Wednesday as start day.

ASPX-CS

```
@*sets start day as Wednesday in calendar*@
@Html.EJ().DatePicker("datePicker").StartDay(3)
```

Step Months

The EJMVC DatePicker calendar allows you to quick navigate back and forth from one month to previous or next month by clicking the arrow button. By default it is navigate one by one month. You can also navigate by skipping months in odd or even or any count by using StepMonths property.

ASPX-CS

```
@*skips the one months from current month(July to Sept to Nov)*@
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().DatePicker("datePicker").StepMonths(2)
```

Read Only

You can make EJMVC DatePicker as read only by setting ReadOnly property as true. It allows only to read the value and it cannot be changed by interaction.

ASPX-CS

```
@*sets DatePicker as read only*@
@Html.EJ().DatePicker("datePicker").ReadOnly(true)
```

Enable or Disable

You can enable or disable the EJMVC DatePicker textbox by using Enabled property. In inline mode DatePicker calendar also gets enabled or disabled.

ASPX-CS

```
@*disables the DatePicker textbox*@
@Html.EJ().DatePicker("datePicker").Enabled(false)
```

How to?

Customizing template with range selection between two DatePicker.

You can customize the date field to emphasize the particular dates in EJMVC DatePicker calendar with help of SpecialDates and set the date range using MinDate and MaxDate property.

Validating the date value in server side

The EJMVC DatePicker control, is a form control which allows us to validate the date value in client side and server side actions. Please refer from the validation to know more about this with EJMVC DatePicker control.

Localize DatePicker with browser specific culture

You can get the browser culture name at page load or document ready state. Based on the culture name, EJMVC DatePicker can be initiated with that specific culture value by assigning to Locale property.

Disable specific dates to restrict user

EJMVC DatePicker allows you to restrict date selection in specific range by using date range option. But you can also restrict selective date in DatePicker calendar by utilizing BeforeDateCreate event. This event will get triggered at each date creation. So you can disable the selective date in this event to restrict the user.

ASPX-CS

```
@*handler to listen each date create*@
@Html.EJ().DatePicker("datePicker").ClientSideEvents(clientSideEvent =>
clientSideEvent.BeforeDateCreate("restrictDate"))
<script>
//event get triggered for each date create.
function restrictDate(args) {
//date to disable in calendar to restrict selection.
var disableDate = new Date("09/22/2015");
//compares two and adds the disable class.
if (+args.date === +disableDate)
args.element.addClass('e-disable');
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// args contains current date and its element.
}
</script>
```

How to add clear button with DatePicker?

Clear button can be included in the DatePicker control. In the create event of DatePicker, clear button element should be appended in the input element and event for clearing the value should bind with the clear button element. Refer the sample from the link Clear button to know how to add the clear button with the DatePicker component. Based, on theme loaded you can adjust the styles of the clear button.

ASPX-CS

```
@Html.EJ().DatePicker("date").ClientSideEvents(e =>
e.Create("Created")).Value(System.DateTime.Now)
<script>
function Created() {
if (this.innerWrapper.find('.e-clear-date').length == 0) {
this.innerWrapper.append("<span class='e-clear-date e-icon'></span>"); //
create and append the 'div' element to the calendar
this._on($('.e-clear-date', this.innerWrapper), "click", function () {
this.option('value', null); if (!this.model.displayInline) this.hide(); });
// bind the 'Click' event to that 'div' element
}
}
</script>
<style>
/*styles for clear button*/
.e-clear-date {
text-align: center;
position: absolute;
right: 24px;
top: 0;
background: #ececec;
width: 21px !important;
height: 100% !important;
margin-top: -16px !important;
}
.e-clear-date:hover {
background: #86cbea;
cursor: pointer;
}
.e-clear-date:before {
content: "\e605";
font-size: 16px;
line-height: 1.8;
}
/*end of styles*/
</style>
```

How to integrate with bootstrap grid system?

EJMVC DatePicker is responsive control, you have to just set the input element width as 100%. In Bootstrap grid layout use the below code example to get responsive textbox

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@*sets the height and width to 100%*@
@Html.EJ().DatePicker("datePicker").Height("100%").Width("100%")
```

Render Datepicker from Code behind

DatePicker can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of DatePicker properties in the controller.

C#

```
//Namespace to get the JavaScript (DatePicker) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class DatePickerController : Controller
{
// GET: DatePicker
public ActionResult DatePickerFeatures()
{
//Initializing the DatePicker model
DatePickerProperties DateObj = new DatePickerProperties();
//Initializing the datasource and other properties
DateObj.Width = "300px";
DateObj.Value = "12/12/2018";
//Passing DatePicker properties using the ViewData
ViewData["DateModel"] = DateObj;
return View();
}
}
}
```

Binding the DatePicker properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().DatePicker("searchCustomer",(Syncfusion.JavaScript.Models.DatePick
erProperties)ViewData["DateModel"]).Render();
}
```

DateRangePicker

DateRangePicker

The Essential ASP.NET MVC DateRangePicker control allows the user to select date range from two calendars in the popup. Also, DateRangePicker has the flexible options to navigate back and forth from month and years and pick the start and end date in any month.DateRangePicker provides the easiest way to get the details within the date ranges in application.

Key Features

- Values formatting.
- Globalization.

Copyright © 2001 - 2020 Syncfusion Inc.

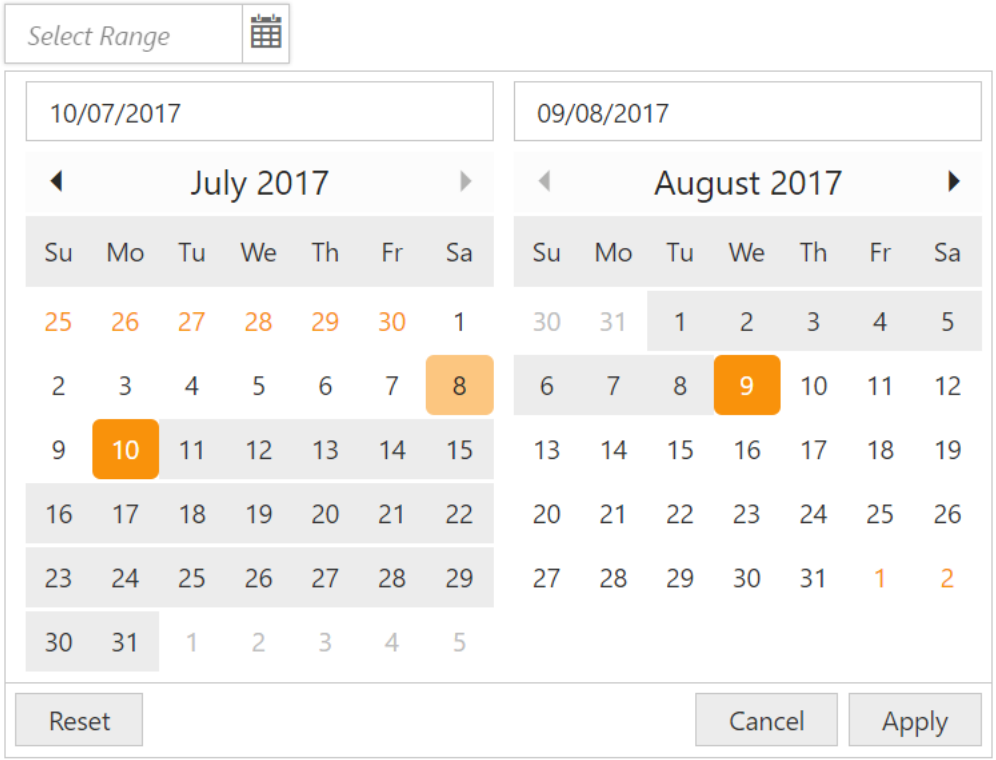
- Preset Ranges.
- Time Picker option.
- Drilldown navigation support.
- State persistence.
- Responsive dimension.
- Flexible customization.
- Custom Styling.

Getting Started

This section explains briefly about how to render a DateRangePicker in your application with ASP.NET MVC.

Create your first DateRangePicker in MVC

The Essential ASP.NET MVC DateRangePicker supports to display two calendars in the webpage and allows you pick date range from the calendars. Refer the following guidelines to customize the DateRangePicker. The following screenshot illustrates the functionality of DateRangePicker.



DateRangePicker

In the above screenshot, you can see the start date and end date selection of a date range in DateRangePicker. This control offers easy way to navigate and select desired date range from one month as start date to other month as end date or we can pick both the start date and end date from the same month.

Create a DateRangePicker

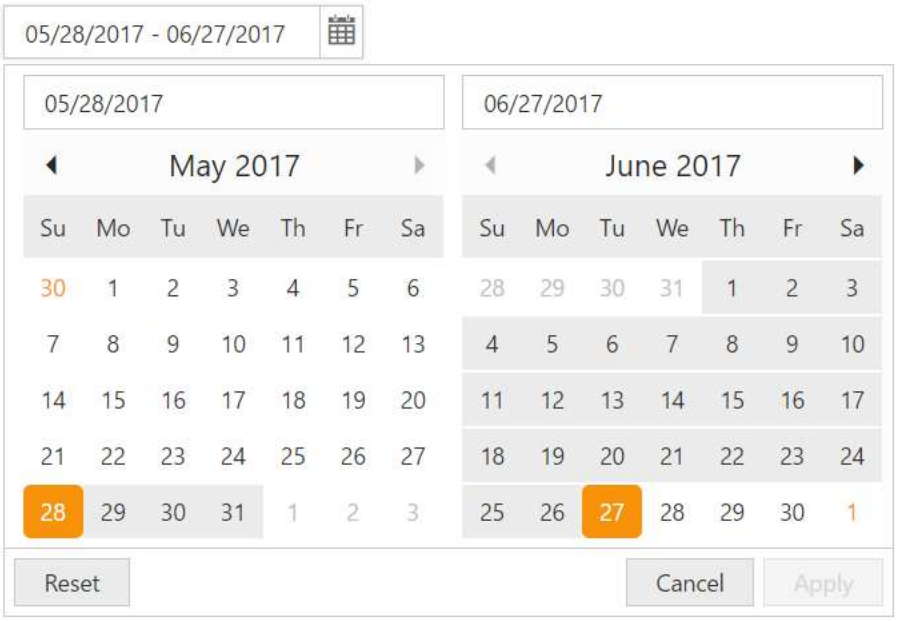
ASP.NET MVC DateRangePicker widget basically renders with built-in features like navigation between two calendars in popup with animations and flexible APIs. You can easily create the DateRangePicker widget using simple DateRangePicker elements as follows.

1. Create an MVC Project, add necessary assemblies and scripts.
2. Refer [MVC-Getting Started] (https://help.syncfusion.com/aspnetmvc/getting-started).
3. Add the following code to the corresponding view page to render DateRangePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateRangePicker widget*@
@Html.EJ().DateRangePicker("DateRangePick").Value("05/28/2017-
06/27/2017").DateFormat("dd/MM/yyyy").Width("200px")
```

Execute the above code to render the following output.



DateRangePicker

Start Date

Start Date can be selected from any calendar in the popup of DateRangePicker. The date which is selected in the calendar in any following cases will be considered as StartDate of a date range,

1. Click on any date in an calendar of dateRange popup where there is no selection in calendars.
2. Click on date which is lesser than existing start date
3. Click on calendar when already there is start date and end date is updated.In this case exiting selection will be cleared fully and clicked date will be set as StartDate to form new range.

Start Date of range can be set using API called **StartDate**, please refer the below code example:

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateRangePicker widget with start date *@
@Html.EJ().DateRangePicker("DateRangePick").StartDate("5/1/2017")
```

Copyright © 2001 - 2020 Syncfusion Inc.

And also, StartDate can be set to popup, by entering the date value into first input box in popup.

End Date

End Date of the date range can be selected from popup directly. Else this can be also updated by using the API called EndDate.

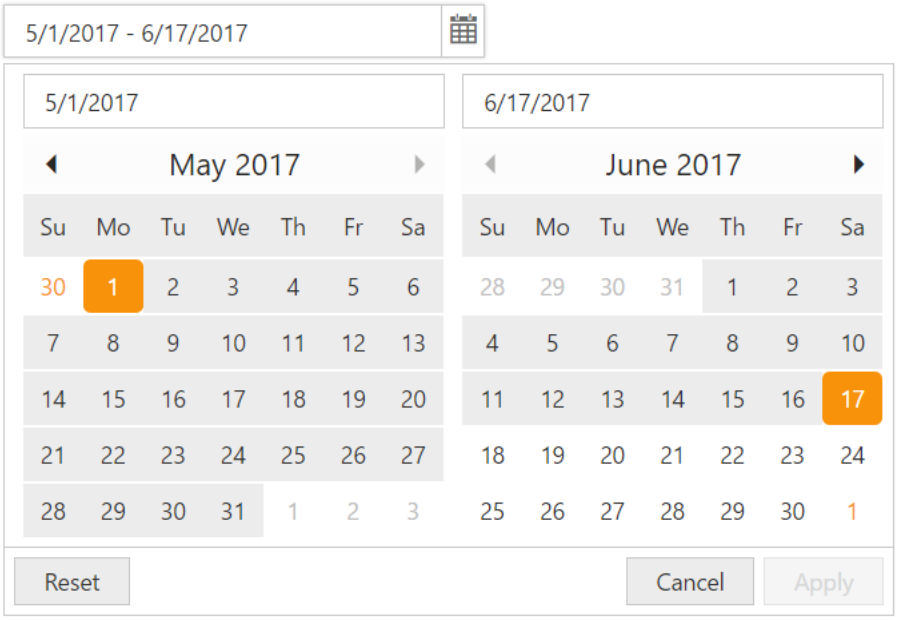
The Selection next to the **StartDate** will be considered as end date. This selected date should be higher or equal date than the existing start date. Else this selection will be considered **StartDate** as we have mentioned in StartDate section above.

Below code will explain to use the **EndDate** API to set the end Date in popup.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateRangePicker widget with customized Start Date and  End Date*@
@Html.EJ().DateRangePicker("DateRangePick").Width("25%").StartDate("5/1/2017
").EndDate("6/17/2017")
```

Execute the above code to render the following output.



DateRangePicker with StartDate and EndDate

End Date can be set to popup by entering the date into the second input box in popup.

Behavior-Settings

DateRangePicker has some default behavior settings which helps you to perform more operations by built-in.

Value

DateRangePicker value can be selected through picking two date values from available two calendars on popup or you can set it by using **Value** property.

Add the following code in your CSHTML page to render DateRangePicker with value.

ASPX-CS

```
@Html.EJ().DateRangePicker("DateRangePick").Value("5/12/2017 -
12/5/2017").ClientSideEvents(p => p.Change("onChange"))
```

ASPX-CS

```
function onChange(args) {
//args contains entire model of DateRangePicker to get the value of all
properties.
//alert DateRangePicker shows the start date and end date.
alert(" start date is : " + args.startDate + " \n end date is : " +
args.endDate);
}
```

Separator

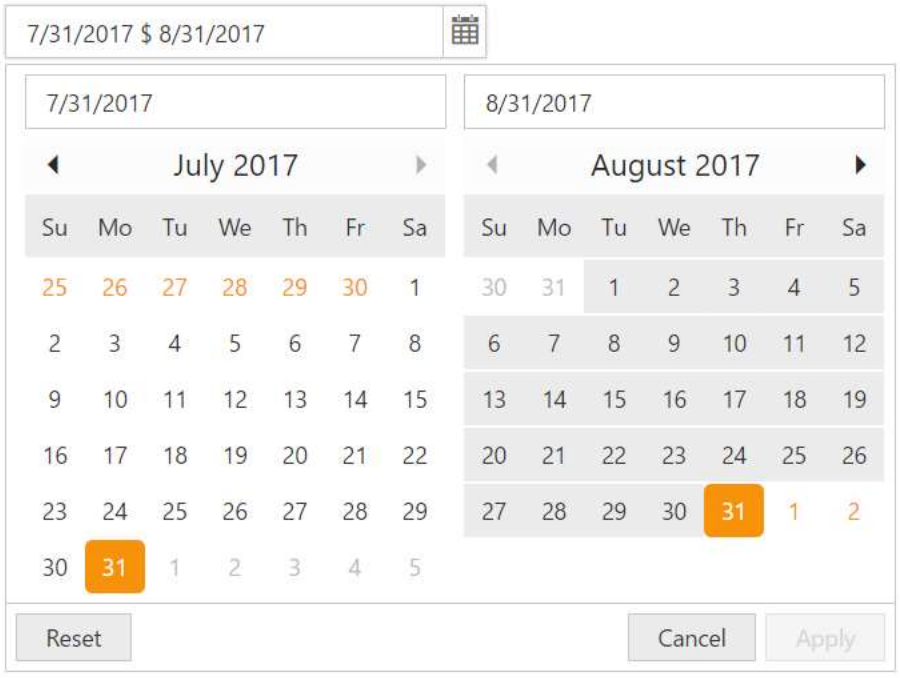
The value of the DateRangePicker popup will be presented with two date strings which is separated by **Separator** (e.g “11/1/2016 - 11/2/2017”).By default , the symbol "-" separates the two date strings of the value and this can be changed using API called **Separator**. Please check with below code example to set the separator using **Separator** API.

Add the following code in your CSHTML page to render DateRangePicker widget with value separated by separator.

ASPX-CS

```
@Html.EJ().DateRangePicker("DateRangePick").Value("7/31/2017 $
8/31/2017").Width("25%").Separator("$")
```

Execute the above code to render the following output.



DateRangePicker

Localization

DateRangePicker, has built in support with Localization. so, it allows users to use culture specified DateRangePicker component with their application using locale API.

Essential ASP.NET MVC DateRangePicker has been provided with built-in localization support, so that it can adapt based on culture specific locale defined for it.

More than 350 culture specific files are available to localize the DateRangePicker. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: Seven culture-specific script files are available in the below specified location. For all other culture files, please download from the GitHub location.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

To translate our control content from default English to any of the culture, say For example - French language, then you need to refer the ej.culture.fr-FR.min.js file in your application,

The **en-US** locale is currently being used as default culture in **DateRangePicker**. You can set any other culture to DateRangePicker using **Locale** property. Below code example shows French cultured DateRangePicker.

Refer the below French culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

JAVASCRIPT

```
<script src="https://cdn.syncfusion.com/js/assets/i18n/ej.culture.fr-FR.min.js"></script>
```

If you want to change month names to your culture month just replace month names with your culture month names or your customized format.

The following example is used to set DateRangePicker in French language.

ASPX-CS

```
calendars: {
standard: {
'/': '/',
':': ':',
firstDay: 1,
days: {
names: ["dimanche","lundi","mardi","mercredi","jeudi","vendredi","samedi"],
namesAbbr: ["dim.","lun.","mar.","mer.","jeu.","ven.","sam."],
namesShort: ["di","lu","ma","me","je","ve","sa"]
},
months: {
```

Copyright © 2001 - 2020 Syncfusion Inc.

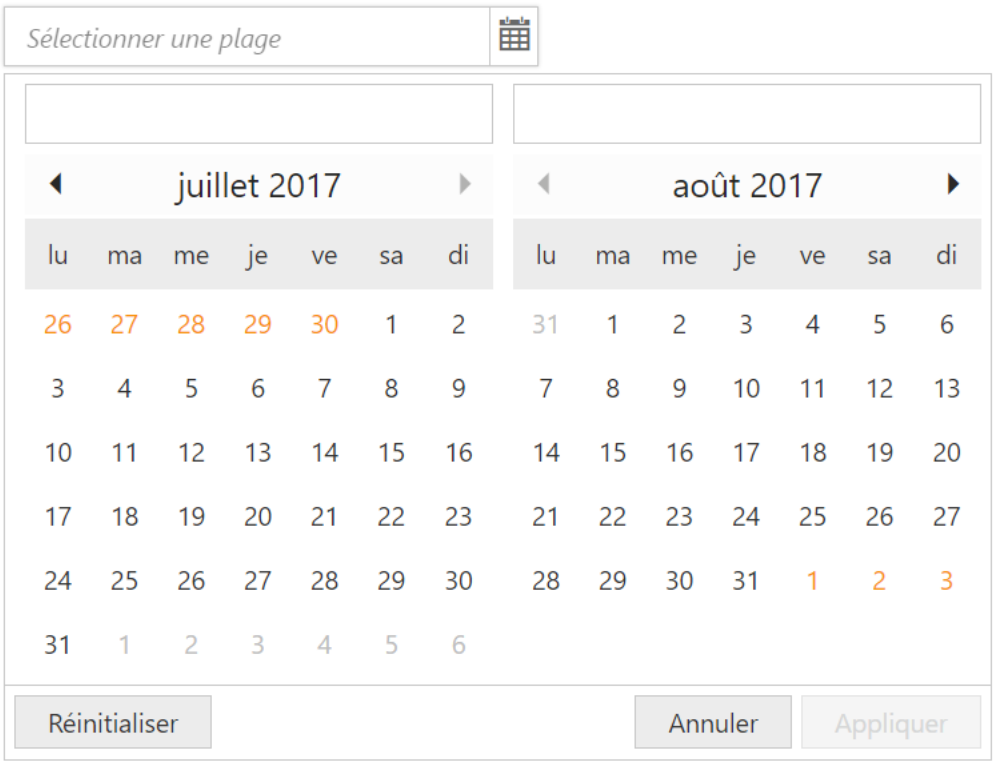
```
names:
["janvier","février","mars","avril","mai","juin","juillet","août","septembre
","octobre","novembre","décembre",""],
namesAbbr:
["janv.","févr.","mars","avr.","mai","juin","juil.","août","sept.","oct.","n
ov.","déc.",""]
},
AM: null,
PM: null,
eras: [{"name":"ap. J.-C.","start":null,"offset":0}],
patterns: {
d: "dd/MM/yyyy",
D: "dddd d MMMM yyyy",
t: "HH:mm",
T: "HH:mm:ss",
f: "dddd d MMMM yyyy HH:mm",
F: "dddd d MMMM yyyy HH:mm:ss",
M: "d MMMM"
}
}
}
```

Add the following code in your CSHTML page to render DateRangePicker widget.

ASPX-CS

```
@Html.EJ().DateRangePicker("DateRange").Locale("fr-FR").Width("25%")
```

The following screenshot displays the output for the above code.



Showcase for DateRangePicker with French culture

Customization

Customization of DateRangePicker will be easier with its flexible customization options.

Setting Dimension

Height and width of the DateRangePicker can be changed using corresponding API (**Height**,**Width**) like below code examples.

Add the following code in your CSHTML page to render DateRangePicker with customized **Height** and **Width**.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateRangePicker widget with customized height and width*@
@Html.EJ().DateRangePicker("DateRange").Height("50").Width("300")
```

Rounded Corners

ShowRoundedCorner property is used to add rounded borders to the input and popup elements. By default, ShowRoundedCorners property will be disabled state. we can enable the property by setting **ShowRoundedCorner** as **true**.

Add the following code in your CSHTML page to render DateRangePicker widget with rounded corners.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateRangePicker widget with rounded corners*@
@Html.EJ().DateRangePicker("DateRange").ShowRoundedCorner(true)
```

Persistence

The value of DateRangePicker can be sustained even after form post back and page refreshing by enabling the **EnablePersistence** API like below code example.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to enable
state persistence in rendered DateRangePicker widget*@
@Html.EJ().DateRangePicker("DateRange").EnablePersistence(true)
```

The DateRangePicker Model value will be stored in local storage / cookies of browser before page refreshes and reinitialized with the restored model after refresh.

Preset Ranges

We can make use of **Ranges** API for easy selection of preset ranges from the popup. Each preset range will have a label which will be displayed on the right side of the popup with user-defined name. By clicking the labels the associated date ranges will get updated in the popup, automatically.

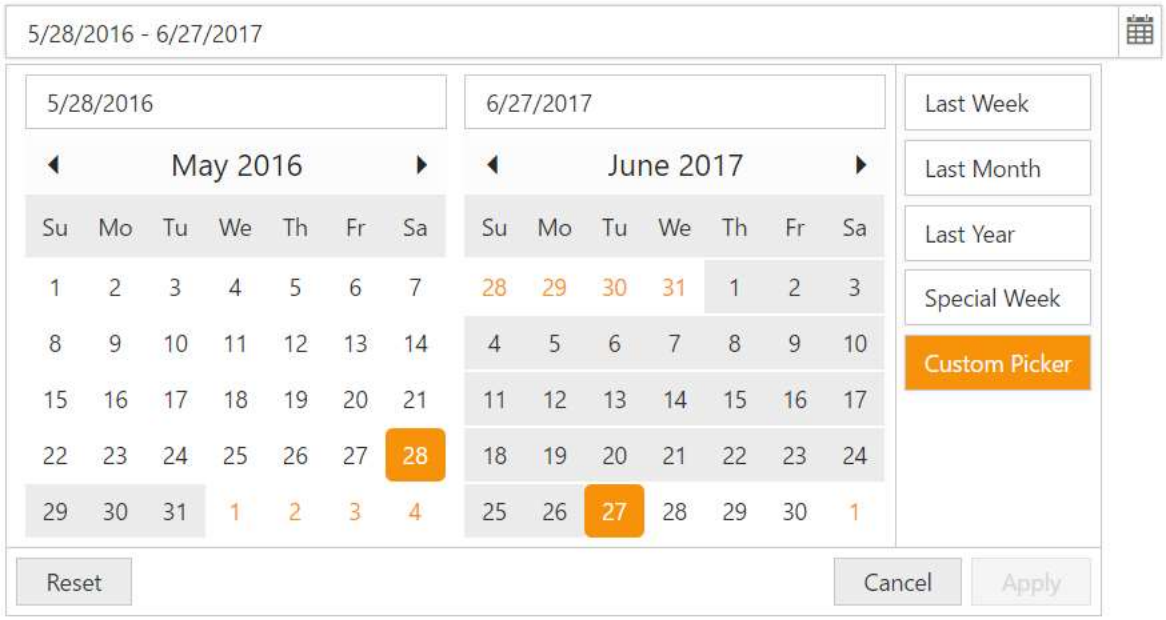
ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to enable
state persistence in DateRangePicker widget*@
@{List<Object> lastWeek = new List<Object>() { DateTime.Today.AddDays(-
7),DateTime.Today.AddDays(-1)};}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{List<Object> lastMonth = new List<Object>() { DateTime.Today.AddMonths(-
1), DateTime.Today.AddDays(-1) };}
@{List<Object> lastYear = new List<Object>() { DateTime.Today.AddYears(-1),
DateTime.Today.AddDays(-1) };}
@{List<Object> special = new List<Object>() { "12/25/2017", "1/1/2018" };}
@Html.EJ().DateRangePicker("DateRange").Value("05/28/2016-
06/27/2017").Ranges(e => e.Add().Label("Last
Week").Range(lastWeek)).Ranges(p => p.Add().Label("Last
Month").Range(lastMonth)).Ranges(m=>m.Add().Label("Last
Year").Range(lastYear)).Ranges(n => n.Add().Label("Special
Week").Range(special)).Width("60%")
```

Execute the above code to render the following output.



DateRangePicker with preset ranges

TimePicker Option

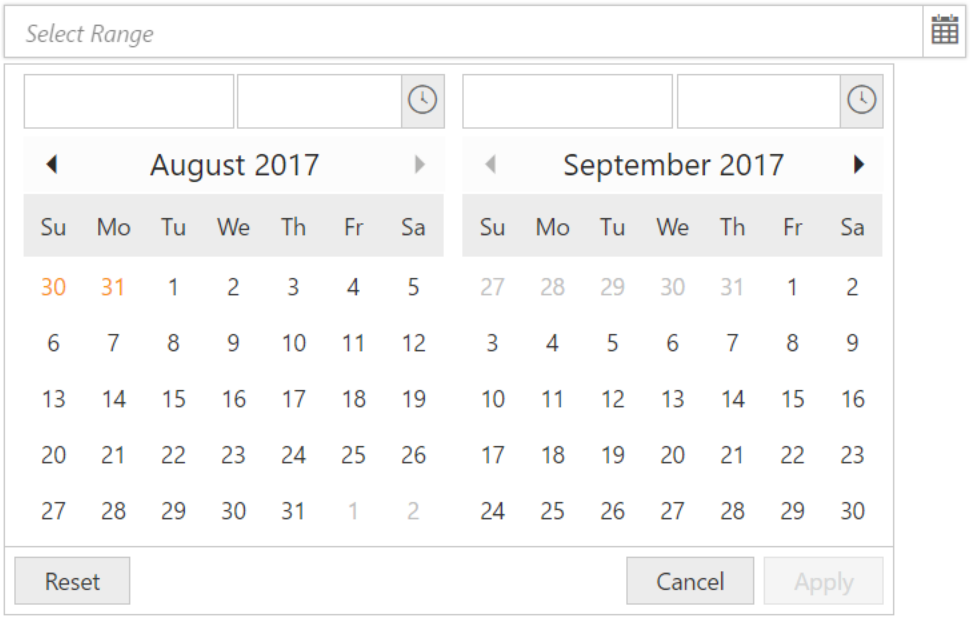
The date ranges can also be selected with start time and end time by enable the TimePicker in popup using **EnableTimePicker** API. Each calendar will have separate Time Pickers in which we can select start time along with start date and end time along with end date. Please check with the below code example to enable the time picker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateRangePicker with TimePicker widget*@
@Html.EJ().DateRangePicker("DateRange").EnableTimePicker(true)
```

Execute the above to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



DateRangePicker with TimePicker option

Allow Editing

Editing in input box can be disabled by setting the false value to **AllowEdit** API. By default this property will be true and we can edit the value in input box.

ASPX-CS

```
@Html.EJ().DateRangePicker("DateRange").AllowEdit(false)
```

Enable/Disable

The control can be enabled / disabled using public methods, **enable** or **disable**.

ASPX-CS

```
@Html.EJ().DateRangePicker("DateRange").Value("05/28/2016-
06/27/2016").ClientSideEvents(p => p.Create("Create"))
<script>
//Create event triggers after DateRangePicker created successfully.
function Create() {
$("#DateRange").ejDateRangePicker('disable');
}
</script>
```

Clear Ranges

To clear the all selection and ranges in a popup we can make use of **clearRanges** method.

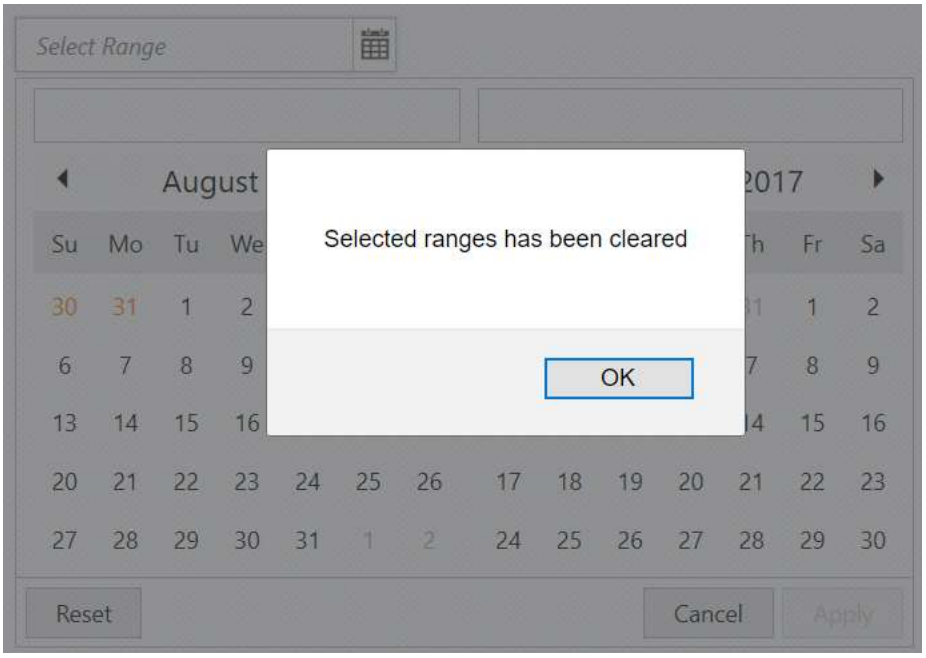
ASPX-CS

```
@Html.EJ().DateRangePicker("DateRange").ClientSideEvents(c=>c.BeforeClose("C
lear"))
<script>
//BeforeClose event triggers before closing DateRangePicker popup
function Clear() {
$("#DateRange").ejDateRangePicker('clearRanges');
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
```

The following screenshot displays the output for the above code.



BeforeClose Event of DateRangePicker

How to?

Render DateRangePicker from Code behind

DateRangePicker can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of DateRangePicker properties in the controller.

C#

```
//Namespace to get the JavaScript (DateRangePicker) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class DateRangePickerController : Controller
{
// GET: DateRangePicker
public ActionResult DateRangePickerFeatures()
{
//Initializing the DateRangePicker model
DateRangePickerProperties DateObj = new DateRangePickerProperties();
//Initializing the datasource and other properties
DateObj.Width = "300px";
DateObj.StartDate = "2/2/2018";
DateObj.EndDate = "2/25/2018";
//Passing DateRangePicker properties using the ViewData
ViewData["DateModel"] = DateObj;
return View();
```

```
}
}
}
```

Binding the DateRangePicker properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().DateRangePicker("searchCustomer",(Syncfusion.JavaScript.Models.Dat
eRangePickerProperties)ViewData["DateModel"]).Render();
}
```

DateTimePicker

DateTimePicker

The Essential ASP.NET MVC DateTimePicker control is used to input the date and time with a specific format. The DateTimePicker control combines the DatePicker and TimePicker controls so that users can select the date and time with their desired format.

Key Features

- DateTime format: Supports all valid date and time formats.
- Localization: Supports localization to different cultures.
- Persist: Supports state maintenance during page refresh.
- RTL: Support for Right to Left alignment of content in DateTimePicker control.
- Themes: Essential ASP.NET MVC controls consist of 12 built-in themes (6 – flat and 6 – gradient effects), and also support custom skin options to set user-defined themes.

Note: As the DateTimePicker inherits the functionalities of the DatePicker and TimePicker controls, it supports the basic functionalities of both controls.

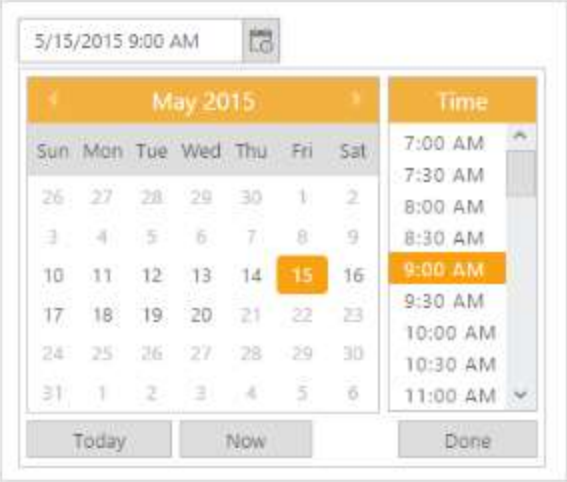
Getting Started

This section explains briefly about how to create a DateTimePicker in your application with ASP.NET MVC.

Create your first DateTimePicker in MVC

The ASP.NET MVC DateTimePicker provides support to display the calendar within your web page and allows you to pick date and time from the calendar. Refer the following guidelines to customize date and time pickers for a real time appointment booking scenario that helps you to book an appointment, within a limited number of days. The following screenshot illustrates the functionality of DatePicker with date range.

Copyright © 2001 - 2020 Syncfusion Inc.



DateTimePicker

In the above screenshot, you can select a date and time, within a week ranging from the current date. This helps you to avoid booking an appointment beyond the current date and mentioned date range.

Create a DateTimePicker

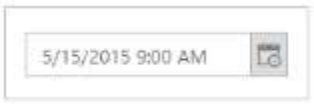
ASP.NET MVC DateTimePicker widget basically renders built-in features like Month and Time Navigation, with animations and flexible APIs. You can easily create the DateTimePicker widget using simple DateTimePicker element as follows.

1. Create a MVC Project and add necessary assemblies and scripts.
2. Refer MVC-Getting Started.
3. Add the following code to the corresponding view page to render DateTimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render DateTimePicker widget*@
@Html.EJ().DateTimePicker("Datetime").Width("175px").Value("05/15/2015 09:00 AM")
```

Execute the above code to render the following output.



DateTimePicker

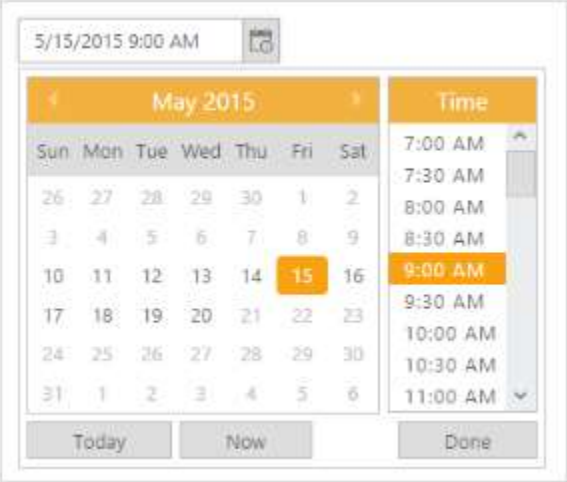
Set the Min/Max Date and Time Intervals

In a real-time appointment booking scenario, the booking is open only for a limited number of days. You can select the date from the given range. This is achieved using the properties MinDateTime and MaxDateTime. Only the dates ranging between MinDateTime and MaxDateTime are enabled in the date picker. Using the Interval property, you can set the interval.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with customized Min time and Max time*@
@Html.EJ().DateTimePicker("DateTime").MinDateTime("5/10/2015 09:00
AM").MaxDateTime("5/20/2015 06:00
PM").Interval(30).Width("175px").Value("5/15/2015 09:00 AM")
```



DateTimePicker with Min/Max Date and Time Interval

Display format

DateTime format

DateTimePicker allows you to define the text representation of a date and time value to be displayed in the DateTimePicker control. The format specified is achieved by the DateTimeFormat property. Default value of this property is “M/d/yyyy h: mm tt”. To change the “Time Popup” display format, “TimeDisplayFormat” is used here. The default value of this property is “h:mm t”.

If your company’s website is going to be used all over the world, following the UTC time is better. Main benefit of UTC Time is that the time is always guaranteed to be *consistent*. In other words, whenever the time zone of customer is changed you don't have to go back or forth in time from the logging time of the customer to your time zone.

DateTime format

Format	Display in DateTimePicker
Short Date and Timeâ€“ d/M/yy h:mm tt	9/12/2014 2:04 PM
Medium Date-d MMM yy h:mm tt	12 Sep 14 2:04: PM
Full Date and short time - dddd, MMMM dd, yyyy HH:mm tt	Friday, September 12,2014 2:04 PM
Full Date and Long Time - dddd, MMMM dd, yyyy HH:mm:ss tt	Friday, September 12,2014 2:04:00 PM
UTC - yyyy-MM-dThh:mm:ssz	2014-09-12T2:04:00+5

You can also customize the format. Refer the following list to create your custom format for DateTimePicker.

Copyright © 2001 - 2020 Syncfusion Inc.

- d - Day of the month.
- ddd - Short name of day of the week.
- dddd - Full name of day of the week.
- M – The month, from 1 through 12.
- MMM- Short name of Month.
- MMMM- Long name of the Month.
- yy - Last two digit if year.
- yyyy - Full Year.
- hh – Hour.
- mm – Minutes.
- ss – Seconds.
- tt - The AM/PM designator.

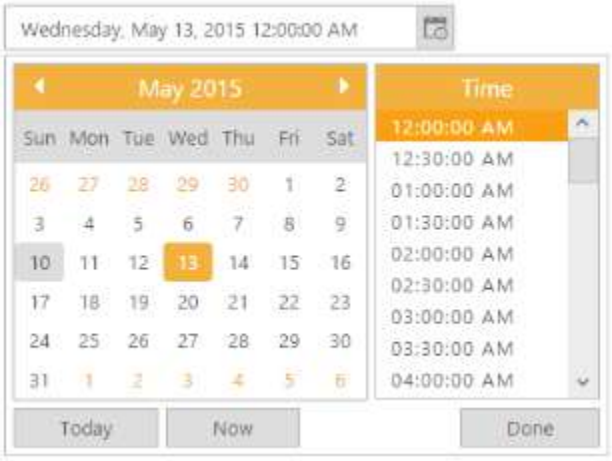
In the following example, set DateTimeFormat to full datetime format.

1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render DateTimePicker widget with customized date time format*@
@Html.EJ().DateTimePicker("DateTime").TimePopupWidth(150).TimeDisplayFormat("hh:mm:ss tt").Width("300px").DateTimeFormat("dddd, MMMM dd, yyyy hh:mm:ss tt")
```

2. The following screenshot displays the output for the above code.



Showcase of DateTimePicker with UTC time format

Day Header Format

You can change the format for the days of the week names using DayHeaderFormat property. By default in our DateTimePicker day of the week format in ShowHeaderMin format. For example, Sun for Sunday. To know the different types of day format refer the following table.

DayHeaderFormat

Header Format types	Description
ShowHeaderNone	Removes the day header
ShowHeaderShort	Shows the day header format in min like Su, Mo, Tu â€¦
ShowHeaderMin	Shows the day header format in short like Sun, Mon, Tue â€¦
ShowHeaderLong	Shows the day header format in long like Sunday, Monday, Tuesday â€¦

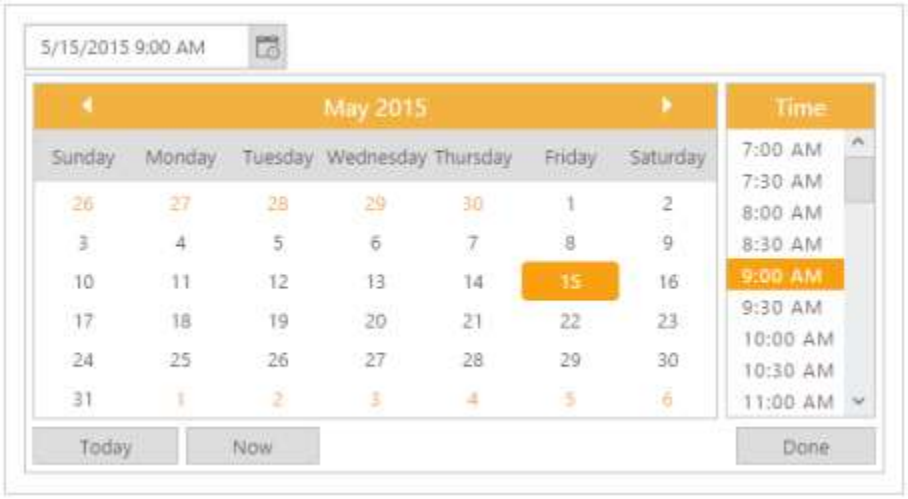
You can also customize the format according to your needs. This is achieved by changing the day names information in the culture script file. This is explained later under the Localization section of this document. In the following sample is displayed, the short name of the day of the week, by setting day header format as ShowHeaderLong.

1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with customized day header format*@
@Html.EJ().DateTimePicker("DateTime").DayHeaderFormat(Header.ShowHeaderLong)
.Width("175px").Value("5/15/2015 09:00 AM")
```

2. The following screenshot displays the output for the above code.



Showcase of DateTimePicker with short name of day of the week

Date Range

Date Range between two dates is achieved by MinDateTime, MaxDateTime property of DateTimePicker.

In DateTimePicker, a property called EnableStrictMode is available.When this property is set to false, it does not allow wrong values and corrects the entered value automatically. If it is true, then DateTimePicker allows values with error class to indicate the selected date is wrong.

Copyright © 2001 - 2020 Syncfusion Inc.

MinDateTime - Sets the minimum value to the DateTimePicker. Based on the EnableStrictMode value behind the minimum value an error class is added to the wrapper element or it is corrected automatically to the nearest correct date value.

MaxDateTime - Sets the maximum value to the DateTimePicker. Based on the EnableStrictMode value beyond the maximum value an error class is added to the wrapper element or it is corrected automatically to the nearest correct date value.

Sometimes, you can give restrictions on selecting the date before or after the particular date. Consider you are going to make a project for hotel reservation system. The “In DateTime” has to be lesser than the “Out DateTime” and vice versa. So you have to set “In DateTime” as minimum DateTime and “Out DateTime” as maximum DateTime for selection in DateTimePicker control.

The dates before min date and after the max date are considered as invalid dates and it is disabled for selection.

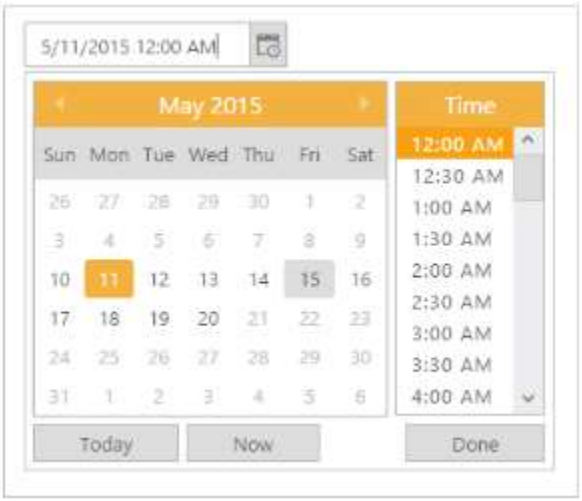
In the following example May 10, 2015 09:00 AM is set as MinDateTime and May 20, 2015 09:00 PM set as MaxDateTime.

1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with customized Min time and Max time*@
@Html.EJ().DateTimePicker("DateTime").MinDateTime("05/10/2015 09:00
AM").MaxDateTime("05/20/2015 09:00 PM").Interval(30).Width("175px")
```

2. The following screenshot displays the output for the above code.



Showcase of min and max dates in DateTimePicker

Copyright © 2001 - 2020 Syncfusion Inc.

Start and Depth level

Start Level

DateTimePicker allows you to change the starting level view of Calendar inside the DateTimePicker popup. Consider you are creating a login form for your blog. If the “Birth date” field in the form starts from year, it is easy to select date from year, month and date. By default, the Start Level of DateTimePicker is “Month” level view. The “StartLevel” property receives “enum” value. The “Period” class contains four types of levels. Refer the following table to know the different types of start level.

Start Level

StartLevel	Syntax	Description
Month	StartLevel(Period.Month)	Starts from month level view.
Year	StartLevel(Period.Year)	Starts from year level view.
Decade	StartLevel(Period.Decade)	Starts from year decade level view.
Century	StartLevel(Period.Century)	Starts from century level view.

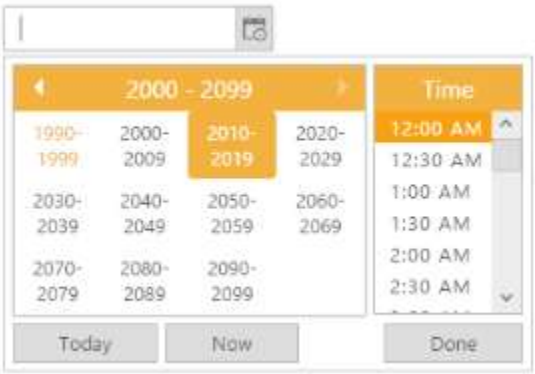
In the following example DateTimePicker popup Start level view is set to “century level” when you drop down the DateTimePicker popup.

1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render DateTimePicker widget with customized Start level*@
@Html.EJ().DateTimePicker("DateTime").StartLevel(Period.Century).Width("175px")
```

2. The following screenshot displays the output for the above code.



Showcase of DateTimePicker with short name of day of the week

Copyright © 2001 - 2020 Syncfusion Inc.

Depth Level

DateTimePicker enables you to set the drill down level of DateTimePicker. Consider, you are going to take the monthly report of sales in your company. In that case, there is no need for selecting date from month view. You can restrict the drill-down of DateTimePicker popup to month view.

In the following example, start level is set from century and drill down is set to year view. So drill-down to month is restricted. So you can avoid selecting the day of the month. By default, the current day of the month is maintained in the day number field. Refer the following code to set drill down value to year view.

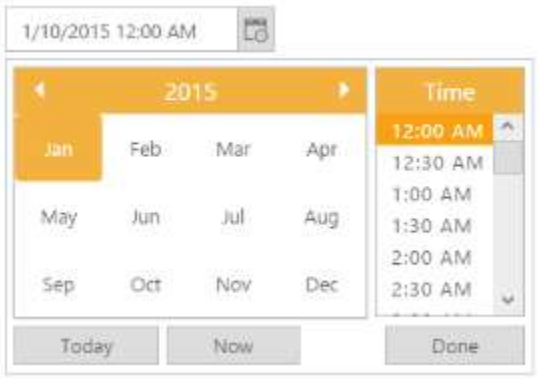
The following code snippet is set to DepthLevel in DateTimePicker.

1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with customized Depth level*@
@Html.EJ().DateTimePicker("DateTime").StartLevel(Period.Century).DepthLevel(
Period.Year) .Width("175px")
```

2. The following screenshot displays the output for the above code.



Showcase of DateTimePicker with Depth level

Date in other months

DateTimePicker calendar can display the dates in other months at the start or end of the current month. To enable or disable the display of other month dates in the current month, you can use the property called ShowOtherMonths. By setting this property value as “true” you can display the dates in other months at the start or end of the current month. By default the value of this property is “true”.

Consider you are going to calculate the monthly report of your company’s employee attendance. To avoid the mistake of selecting other month dates while calculating current month report, you can disable showing other month dates in the current month. You can achieve this requirement by setting ShowOtherMonths value as “false”.

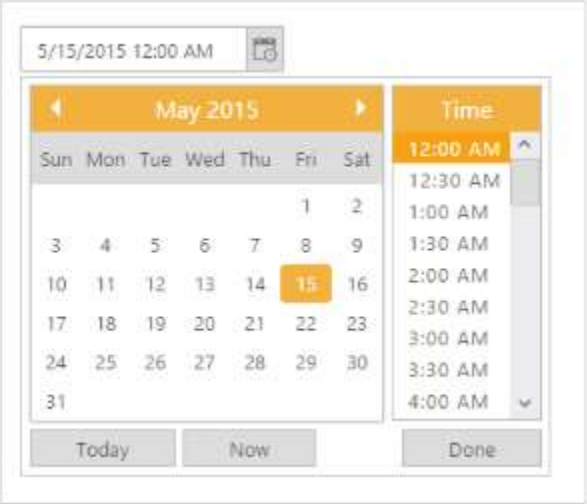
1. Add the following code in your CSHTML page to render DateTimePicker widget.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with customized other months*@
@Html.EJ().DateTimePicker("DateTime").ShowOtherMonths(false).Width("175px")
```

2. The following screenshot displays the output for the above code.



Showcase for DateTimePicker without other month dates

Time Interval

You can set time interval between two adjacent time values in the time popup manually using the “Interval” property. By default the value of the “Interval” property is 30 minutes. Setting this value as 60 is considered as 1 hour. Sometimes you need to update for every hour work log reports. In that case to select time from time popup window with 1 hour interval to update the every 1 hour report you can use Interval option by setting time interval value as “60 minutes”.

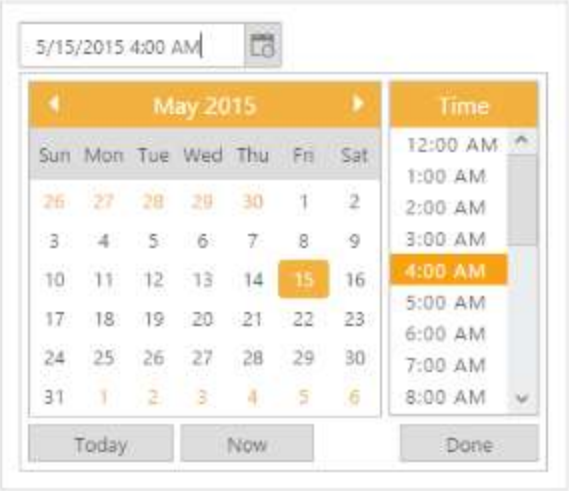
1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with customized interval*@
@Html.EJ().DateTimePicker("DateTime").Interval(60).Width("175px")
```

2. The following screenshot displays the output for the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Showcase for DateTimePicker with I hour interval in time picker popup

Localization

You can globalize your DateTimePicker control. People of different culture can make use of it. All culture files are supported by Syncfusion components

Essential ASP.NET MVC DateTimePicker has been provided with built-in localization support, so that it can adapt based on culture specific locale defined for it.

More than 350 culture specific files are available to localize the datetime. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: Seven culture-specific script files are available in the below specified location. For all other culture files, please download from the GitHub location.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

To translate our control content from default English to any of the culture, say For example - Spanish language, then you need to refer the ej.culture.es-ES.min.js file in your application,

The **en-US** locale is currently being used as default culture in DateTimePicker. You can set any other culture to DateTimePicker using **Locale** property. Below code example shows Spanish cultured DateTimePicker.

Refer the below Spanish culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

JAVASCRIPT

```
<script src="https://cdn.syncfusion.com/js/assets/i18n/ej.culture.es-ES.min.js"></script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

If you want to change month names to your culture month just replace month names with your culture month names or your customized format.

The following code example is used to set DateTimePicker in Spanish language.

ASPX-CS

```
calendars: {
standard: {
firstDay: 1,
days: {
names: ["domingo","lunes","martes","miércoles","jueves","viernes","sábado"],
namesAbbr: ["do.","lu.","ma.","mi.","ju.","vi.","sá."],
namesShort: ["D","L","M","X","J","V","S"]
},
months: {
names:
["enero","febrero","marzo","abril","mayo","junio","julio","agosto","septiembre","octubre","noviembre","diciembre",""],
namesAbbr:
["ene.","feb.","mar.","abr.","may.","jun.","jul.","ago.","sep.","oct.","nov.","dic.",""]
},
AM: null,
PM: null,
eras: [{"name":"d. C.","start":null,"offset":0}],
patterns: {
d: "dd/MM/yyyy",
D: "dddd, d' de 'MMMM' de 'yyyy",
t: "H:mm",
T: "H:mm:ss",
f: "dddd, d' de 'MMMM' de 'yyyy H:mm",
F: "dddd, d' de 'MMMM' de 'yyyy H:mm:ss",
M: "d' de 'MMMM",
Y: "MMMM' de 'yyyy"
}
}
}
```

The following code example can be used to get Spanish culture in DateTimePicker.

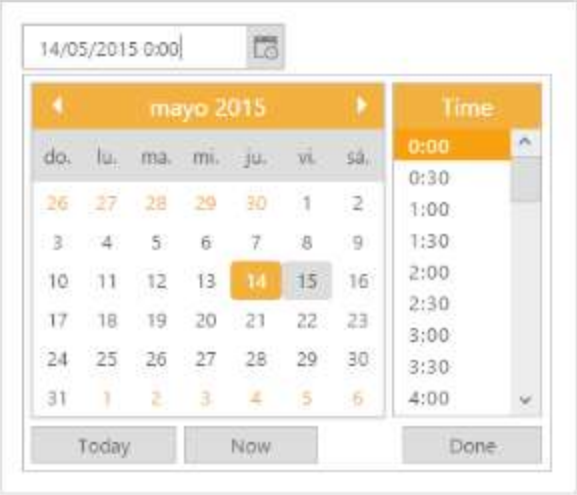
3. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render DateTimePicker widget with customized localization*@
@Html.EJ().DateTimePicker("DateTime").Locale("es-ES").Width("175px")
```

4. The following screenshot displays the output for the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Showcase for DateTimePicker with Spanish culture

Right-to-Left

RTL control supports right-to-left functionality and features for languages that work right-to-left for selecting and editing date and time. You can change your popup and textbox display to read right-to-left. Arabic and Hebrew are written from right to left. The customer who has a right to left writing style can use this feature. You can achieve this in your DateTimePicker by using EnableRTL property. Setting this property to “true” allows you to write in the right to left format. Position of the toolbars are also changed to right to left.

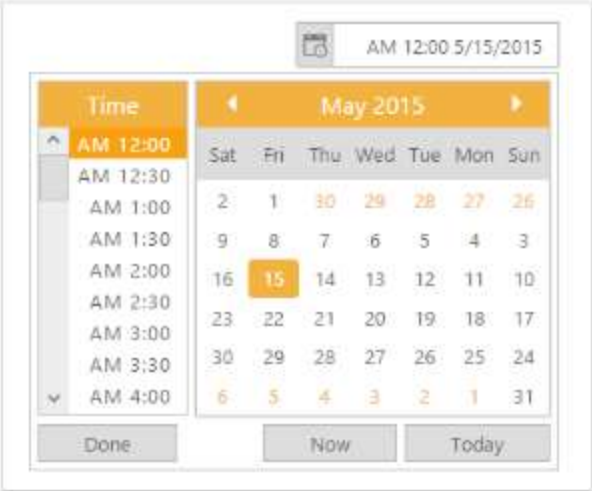
1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with right to left appearance*@
@Html.EJ().DateTimePicker("DateTime").EnableRTL(true).Width("175px")
```

2. The following screenshot displays the output for the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Showcase of DateTimePicker with Right to Left appearance

Appearance and Styling

Theme

DateTimePicker control support rich appearance. This control consist of six flat themes and six gradient themes. To use these twelve themes, refer the themes files in HTML file.

You need two style sheets to apply styles to DateTimePicker control; one ej.widgets.core.min.css and one ej.theme.min.css. If you use ej.widgets.all.min.css then you don't need to use ej.widgets.core.min.css and ej.theme.min.css because ej.widgets.all.min.css is a combination of these two.

The core style sheet applies styles related to positioning and size, but are not related to the color scheme and always require the control to look correct and function properly. The theme style sheet applies theme-specific styles like colors and backgrounds.

The following list is of the twelve themes supported by DateTimePicker:

- default-theme
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

1. Add the following code in your CSHTML page to render DateTimePicker widget.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@*Add the following code example to the corresponding CSHTML page to render
DateTimePicker widget with right to left appearance*@
@Html.EJ().DateTimePicker("DateTime").Width("175px")
```

2. The following screenshot displays the output for the above code.



Showcase of DateTimePicker with gradient lime dark theme

CSS Class

DateTimePicker control also allows you to customize its appearance using user-defined CSS and custom skin options such as colors and backgrounds. To apply custom themes you have a property called CssClass. CssClass property sets the root class for DateTimePicker theme.

Using this CssClass you can override the existing styles under the theme style sheet. The theme style sheet applies theme-specific styles like colors and backgrounds. In the following example, the value of CssClass property is set as “Purple-dark”. Purple-dark is added as root class to DateTimePicker control at the runtime. From this root class you can customize the DateTimePicker control theme.

1. Add the following code in your CSHTML page to render the DateTimePicker.

ASPX-CS

```
@*Add the following code in your view page to render the DateTimePicker.*@
@Html.EJ().DateTimePicker("dt").CssClass("Purple-dark").Width("175px")
```

In the following style sheet the exiting theme style sheet file has been over-ridden using root class “Purple-dark”.

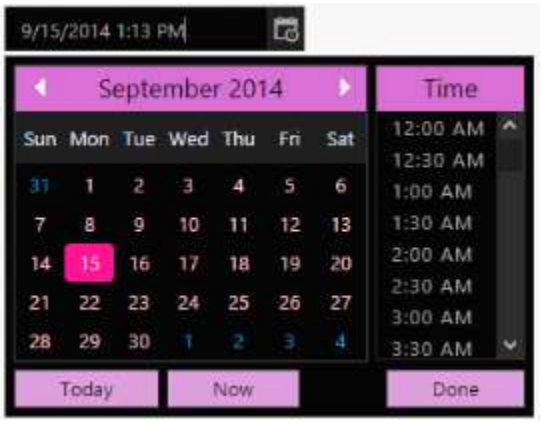
2. Add the following code in your style section.

CSS

```
.Purple-dark .e-week-header {
color: #EBADD6;
}
.Purple-dark .e-text {
color: black;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.Purple-dark .e-state-default {
color: pink;
}
.Purple-dark .e-active {
background-color: #FF1975;
}
.Purple-dark .e-state-default:hover {
color: #EBADD6;
}
.Purple-dark .e-dt-button {
color: black;
background-color: #E085C2;
}
.Purple-dark .e-header {
background-color: #E085C2;
color: black;
}
.Purple-dark .e-timecontainer .e-header {
background-color: #E085C2;
color: black;
}
```



Showcase for DateTimePicker with cssClass “Purple-dark”

How to?

Render DateTimePicker from Code behind

DateTimePicker can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of DateTimePicker properties in the controller.

C#

```
//Namespace to get the JavaScript (DateTimePicker) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class DateTimePickerController : Controller
{
// GET: DateTimePicker
public ActionResult DateTimePickerFeatures()
```

```
{
//Initializing the DateTimePicker model
DateTimePickerProperties DateObj = new DateTimePickerProperties();
//Initializing the datasource and other properties
DateObj.Width = "300px";
DateObj.Value = "05/15/2015 09:00 AM";
//Passing DateTimePicker properties using the ViewData
ViewData["DateModel"] = DateObj;
return View();
}
}
}
```

Binding the DateTimePicker properties passed via **ViewData** from the controller in the client side as below.

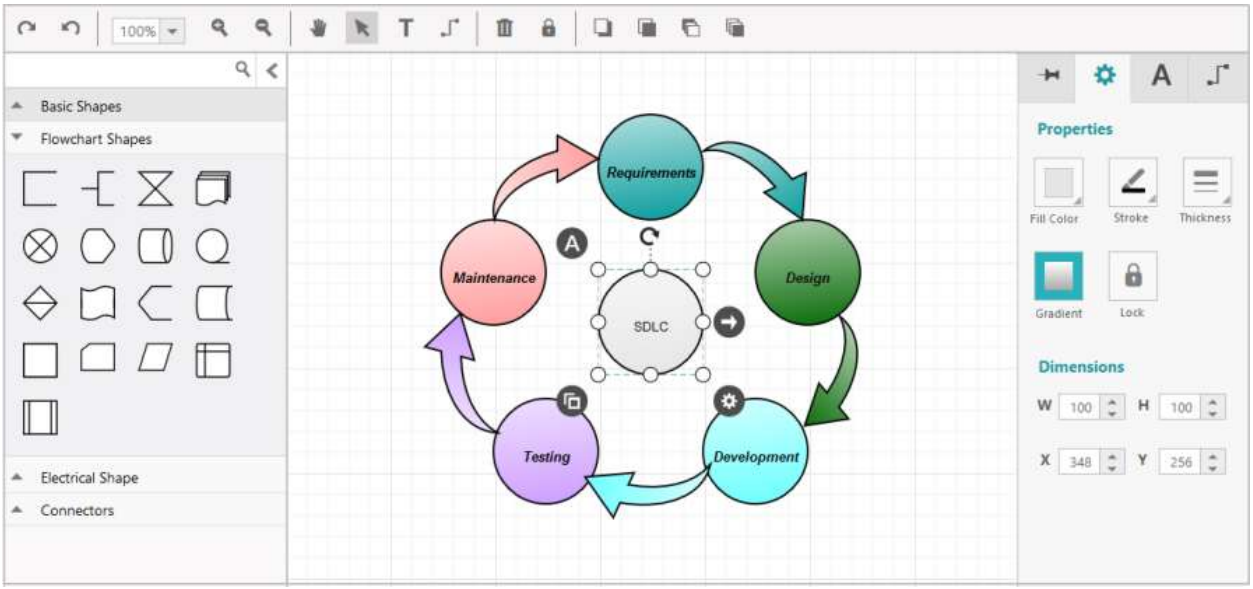
ASPX-CS

```
@{
Html.EJ().DateTimePicker("searchCustomer",(Syncfusion.JavaScript.Models.Date
TimePickerProperties)ViewData["DateModel"]).Render();
}
```

Diagram

Diagram

EssentialDiagram ASP.NET MVC creates rich Visio-like applications. Its Framework comprises of many Elements that helps you to create an application easily. The rich feature set of the Diagram control includes Snapping, Guidelines, Gridlines, Serialization and Zooming.



Diagram

The list of features of the Diagram control in ASP.NET MVC are as follows.

- Node, Connector, Group, Port, and Label: Elements used to compose the Diagram.
- Symbol Palette: It holds a list of symbols dropped over the Diagram.
- Clipboard Commands: Performs Cut, Copy and Paste operations.
- Undo/Redo: Performs correction in recent change.
- Serialization: Saves current state of the Diagram, and loads them back when required.
- Snapping: Snaps the Diagram elements towards the nearest elements.
- Gridlines: Visual horizontal/vertical lines that help align elements on Diagram.
- Interaction: Zoom, pan, multiple selections, keyboard shortcuts, and snapping.
- Layout: Arranges nodes in a tree-like structure based on a relationship between the Nodes.

Getting Started

Getting started with your MVC Diagram is easy. You can start by creating a simple flow chart.

Flow Diagram

Initialize Diagram

1\. Create an ASP.NET MVC application. You can refer MVC Getting Started documentation to create a new project and to add necessary assemblies and script files. 2\. Add a placeholder div element to initialize diagram control

RAZOR

```
<div>
@Html.EJ().Diagram("diagram").Width("100%").Height("600px")
</div>
```

3\. This creates an empty diagram

Create and add Node

Let us create and add a node with specific position, size, label and shape.

RAZOR

```
@*Initializes diagram control*@
<div>
@Html.EJ().Diagram("diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
</div>
```

C#

```
DiagramProperties model = new DiagramProperties();
FlowShape start = CreateNode("Start", 140, 50, 300, 50,
FlowShapes.Terminator, "Start");
model.Nodes.Add(start);
ViewData["diagramModel"] = model;
FlowShape CreateNode(string name, int width, int height, int offsetX, int
offsetY, FlowShapes shape, string text)
{
FlowShape node = new FlowShape()
{
Shape = shape,
Name = name,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Width = width,
Height = height,
OffsetX = offsetX,
OffsetY = offsetY,
Labels = new Collection() { new Label() { Text = text } }
};
return node;
}
```

Note: Labels property is an array, which indicates that more than one label can be added to a node.

Added node will be displayed in diagram as shown below.

Connect nodes

- Create another node with another set of data.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(CreateNode("Start", 140, 50, 300, 50, FlowShapes.Terminator,
"Start"));
model.Nodes.Add(CreateNode("Init", 140, 50, 300, 140, FlowShapes.Process,
"var i = 0;"));
ViewData["diagramModel"] = model;
FlowShape CreateNode(string name, int width, int height, int offsetX, int
offsetY, FlowShapes shape, string text)
{
FlowShape node = new FlowShape()
{
Shape = shape,
Name = name,
Width = width,
Height = height,
OffsetX = offsetX,
OffsetY = offsetY,
Labels = new Collection() { new Label() { Text = text } }
};
return node;
}
```

Connect these two nodes by adding a connector into Connectors collection with reference to source and target end.

C#

```
//Add a connector to `Connectors` collection of diagram model
model.Connectors.Add(ConnectNodes("connector1", "Start", "Init"));
Connector ConnectNodes(string name, string source, string target)
{
return new Connector()
{
Name = name,
```

```
SourceNode = source,
TargetNode = target,
Segments = new Collection() {
new Segment(Segments.Orthogonal)
}
};
}
```

- Connector connects the two nodes as shown below.

- Default values for all nodes and connectors can be set using default settings. For example if all nodes have same Width and Height, we can move such properties into DefaultSettings. Above code can be rewritten as shown below.

C#

```
DiagramProperties model = new DiagramProperties();
//Default Settings
model.DefaultSettings.Node = new Node() { Type = Shapes.Flow, Width = 140,
Height = 50, OffsetX = 300 };
model.DefaultSettings.Connector = new Connector()
{
Labels = new Collection() { new Label() { FillColor = "white" } },
Segments = new Collection() { new Segment(Segments.Orthogonal) }
};
//Create Nodes
model.Nodes.Add(CreateNode("Start", 50, FlowShapes.Terminator, "Start"));
model.Nodes.Add(CreateNode("Init", 140, FlowShapes.Process, "var i = 0;"));
//Connect Nodes
model.Connectors.Add(ConnectNodes("connector1", "Start", "Init"));
ViewData["diagramModel"] = model;
Connector ConnectNodes(string name, string source, string target)
{
return new Connector()
{
Name = name,
SourceNode = source,
TargetNode = target
};
}
FlowShape CreateNode(string name, int width, int height, int offsetX, int
offsetY, FlowShapes shape, string text)
{
FlowShape node = new FlowShape()
{
Shape = shape,
Name = name,
OffsetY = offsetY,
Labels = new Collection() { new Label() { Text = text } }
};
return node;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Complete flow diagram

Similarly we can add required nodes and connectors to form a complete flow diagram.

C#

```
//Initialize model properties
DiagramProperties model = new DiagramProperties();
//Default Settings
model.DefaultSettings.Node = new Node() { Type = Shapes.Flow, Width = 140,
Height = 50, OffsetX = 300 };
model.DefaultSettings.Connector = new Connector()
{
Labels = new Collection() { new Label() { FillColor = "white" } },
Segments = new Collection() { new Segment(Segments.Orthogonal) }
};
//Create Nodes
model.Nodes.Add(CreateNode("Start", 50, FlowShapes.Terminator, "Start"));
model.Nodes.Add(CreateNode("Init", 140, FlowShapes.Process, "var i = 0;"));
model.Nodes.Add(CreateNode("Condition", 230, FlowShapes.Decision, "i <
10?"));
model.Nodes.Add(CreateNode("Print", 320, FlowShapes.PreDefinedProcess, "i <
10?"));
model.Nodes.Add(CreateNode("Increment", 410, FlowShapes.Process, "i <
10?"));
model.Nodes.Add(CreateNode("End", 500, FlowShapes.Terminator, "i < 10?"));
//Connect Nodes
model.Connectors.Add(ConnectNodes("connector1", "Start", "Init"));
model.Connectors.Add(ConnectNodes("connector2", "Init", "Condition"));
model.Connectors.Add(ConnectNodes("connector3", "Condition", "Print",
"Yes"));
model.Connectors.Add(ConnectNodes("connector4", "Condition", "End", "No",
new Segment() { Type = Segments.Orthogonal, Direction = "right", Length = 30
}));
model.Connectors.Add(ConnectNodes("connector5", "Print", "Increment"));
model.Connectors.Add(ConnectNodes("connector5", "Increment", "Condition",
"",
new Segment() { Type = Segments.Orthogonal, Direction = "left", Length = 30
}));
ViewData["diagramModel"] = model;
//Helper methods
Connector ConnectNodes(string name, string source, string target, string
text = "", Segment segment = null)
{
Connector connector = new Connector()
{
Name = name,
SourceNode = source,
TargetNode = target,
Labels = new Collection() { new Label() { Text = text } }
};
if (segment != null) connector.Segments = new Collection() { segment };
return connector;
}
FlowShape CreateNode(string name, int offsetY, FlowShapes shape, string
text)
```

```
{
FlowShape node = new FlowShape()
{
Shape = shape,
Name = name,
OffsetY = offsetY,
Labels = new Collection() { new Label() { Text = text } }
};
return node;
}
```

Final flow chart will looks as shown below.

Automatic organization chart

In 'Flow Diagram' section we saw how to create a diagram manually, now let us see how to create and position diagram automatically.

Initialize diagram

Initializing diagram is already discussed in Flow Diagram > Initialize diagram section.

Business object (Employee information)

- Define Employee Information as JSON data. The following code example shows an employee array whose,
- `Name` is used as a unique identifier and
- `ReportingPerson` is used to identify the person to whom an employee report to, in the organization.

RAZOR

```
//Initialize data source - Saved in a JSON file...
{
data: [
{ Name: "Elizabeth", Role: "Director" },
{ Name: "Christina", ReportingPerson: "Elizabeth", Role: "Manager" },
{ Name: "Yoshi", ReportingPerson: "Christina", Role: "Lead" },
{ Name: "Philip", ReportingPerson: "Christina", Role: "Lead" },
{ Name: "Yang", ReportingPerson: "Elizabeth", Role: "Manager" },
{ Name: "Roland", ReportingPerson: "Yang", Role: "Lead" },
{ Name: "Yvonne", ReportingPerson: "Yang", Role: "Lead" }
]
}
```

Map data source

- You can configure this "Employee Information" with Diagram, so that the node and connector are automatically generated using mapping properties. The following code examples show how dataSourceSetting is used to map id and parent with property name identifiers for employee information.

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
DiagramProperties model = new DiagramProperties();
model.DataSourceSettings.DataSource = GetEmployeeData();
model.DataSourceSettings.Id = "Name";
model.DataSourceSettings.Parent = "ReportingPerson";
Array GetEmployeeData()
{
string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/employee.json"));
Dictionary<string, object> data = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(text);
return (Array)data["data"];
}
```

Visualize employee

Following code examples indicate how to define the default appearance of node and connector using defaultSettings. The NodeTemplate is used to update each node based on employee data.

C#

```
DiagramProperties model = new DiagramProperties();
//Configure data source
model.DataSourceSettings.DataSource = GetEmployeeData();
model.DataSourceSettings.Id = "Name";
model.DataSourceSettings.Parent = "ReportingPerson";
//Default Settings
model.DefaultSettings.Node = new Node()
{
Width = 70,
Height = 30,
Shape = { Type = Shapes.Rectangle, CornerRadius = 5 },
Labels = new Collection() { new Label() { FontSize = 11, Bold = true,
FontFamily = "Segoe UI", FontColor = "white" } }
};
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) }
};
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
```

RAZOR

```
//To represent the roles
var codes = {
Director: "rgb(0, 139,139)",
Manager: "rgb(30, 30,113)",
Lead: "rgb(0, 100,0)"
}
// Bind custom data with node
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Name;
node.fillColor = codes[node.Role];
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

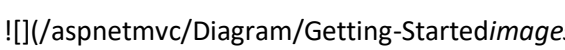
Apply org chart layout

- Next you need to arrange nodes in an organizational chart structure, and to do this you can apply layout as shown in following code example. You can see that spacing, margin and orientation are defined, that can also be customized based on the needs.

C#

```
DiagramProperties model = new DiagramProperties();
model.Layout.Type = LayoutTypes.OrganizationalChart;
model.Layout.MarginX = 10;
model.Layout.MarginY = 50;
model.Layout.HorizontalSpacing = 50;
model.Layout.VerticalSpacing = 50;
model.Layout.Orientation = LayoutOrientations.TopToBottom;
```

- The Employee details are displayed in the Diagram as follows.

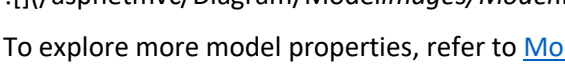


Model

The Diagram model represents the data to render the Diagram and to manipulate the Diagram elements. The following code illustrates how to define Diagram model.

RAZOR

```
//Creates diagram
<div>
@Html.EJ().Diagram("diagram").Width("100%").Height("100%").PageSettings(s=>s
.PageWidth(2000).PageHeight(2000))
</div>
```

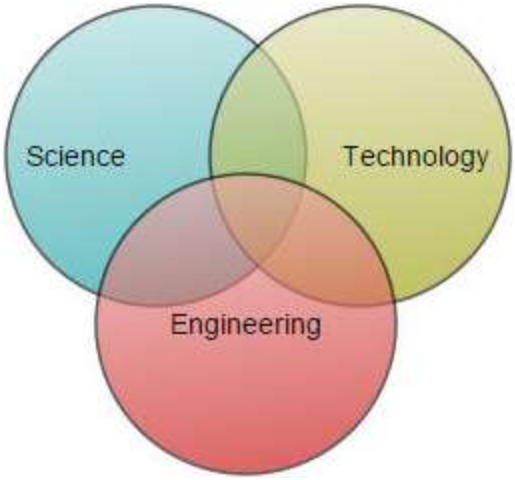


To explore more model properties, refer to Model Properties.

Node

Nodes are graphical object that represent visual data to be placed on the page.

Copyright © 2001 - 2020 Syncfusion Inc.



Create Node

A node can be created and added to the Diagram, either programmatically or interactively. Nodes are stacked on the Diagram area from bottom to top in the order they are added.

Add node through nodes collection

To create a node, You have to create the node object and add that to Nodes collection of the Diagram Model. The following code example illustrates how to add a node to the Diagram.

C#

```
//Initializes Diagram
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "900px";
BasicShape Node = new BasicShape();
//Name of the node
Node.Name = "node1";
//Sets the size
Node.Width = 100;
Node.Height = 100;
//Sets the position
Node.OffsetX = 250;
Node.OffsetY = 250;
//Customizes the appearance
Node.FillColor = "darkcyan";
Node.BorderWidth = 2;
//Adds node to nodes collection
Model.Nodes.Add(Node);
```



Copyright © 2001 - 2020 Syncfusion Inc.

Add node at runtime

Nodes can be added at runtime by using public method, add. The following code illustrates how to add a node.

JS

```
// Defines JSON to create a node
var node = {
name: "node1",
width: 100,
height: 100,
//Sets position
offsetX: 250,
offsetY: 250,
fillColor: "darkcyan",
borderWidth: 2
};
var diagram = $("#diagram").ejDiagram("instance");
// Adds node to the Diagram
diagram.add(node);
```



Add node from palette

Nodes can be predefined and added to palette and can be dropped into the Diagram when needed. For more information about adding nodes from symbol palette, refer to Symbol Palette.

Create node through data source

Nodes can be generated automatically with the information provided through data source. The default properties for these nodes are fetched from default settings. For more information about data source, refer to Data Binding.

Draw nodes

Nodes can be interactively drawn by clicking and dragging the Diagram surface by using **DrawingTool**. For more information about drawing nodes, refer to Draw Nodes.

Update Node at runtime

The client side method updateNode is used to update the nodes at run time. The following code example illustrates how to update a node at runtime.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
diagram.updateNode("nodeName", {
fillColor: "#1BA0E2",
borderWidth: 5,
borderColor: "#000000"
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

Position

Position of a node is controlled by using its OffsetX and OffsetY properties. By default, these offset properties represent the distance between origin of the Diagram's page and node's center point. You may expect this Offset values to represent the distance between page origin and node's top left corner instead of center. Pivot property helps solve this problem. Default value of node's Pivot point is (0.5, 0.5), that means center of Node.

The following table illustrates how pivot relates offset values with node boundaries.

| Pivot | Offset |

|---|---|

| (0.5,0.5) | OffsetX and OffsetY values are considered as the node's center point. |

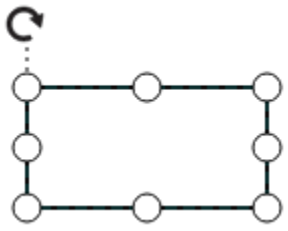
| (0,0) | OffsetX and OffsetY values are considered as the top left corner of node |

| (1,1) | OffsetX and OffsetY values are considered as the bottom right corner of the node. |

The following code illustrates how to change the Pivot value.

C#

```
//Initializes Diagram
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "900px";
BasicShape Node = new BasicShape();
//Name of the node
Node.Name = "node1";
//Sets the size
Node.Width = 100;
Node.Height = 50;
//Sets the position
Node.OffsetX = 100;
Node.OffsetY = 100;
//Sets pivot point
Node.Pivot = new DiagramPoint(0, 0);
//Adds node to nodes collection
Model.Nodes.Add(Node);
```



Types

Diagram allows to add different kind of nodes. To explore the types of nodes, refer to Types of Nodes.

Appearance

You can customize the appearance of a node by changing its font, fill colors, patterns, line weight and style, or shadow. The following code illustrates how to customize the appearance of the shape.

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
//Initializes Diagram
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "900px";
BasicShape Node = new BasicShape();
Node.Name = "node1";
Node.Width = 100;
Node.Height = 100;
Node.OffsetX = 100;
Node.OffsetY = 100;
//Sets styles to a node to customize the appearance
Node.FillColor = "darkcyan";
Node.BorderWidth = 2;
Node.BorderColor = "black";
Node.BorderDashArray = "5 5";
//Adds node to nodes collection
Model.Nodes.Add(Node);
```



Gradient

There are two types of gradients.

- **Linear gradient -** Defines a smooth transition between a set of colors (so-called "stops") on a line.
- **Radial gradient -** Defines a smooth transition between stops on a circle.

The Gradient property of Node allows you to define and applies the gradient effect to that node.

C#

```
BasicShape Node = new BasicShape();
Node.Name = "node1";
Node.Width = 100;
Node.Height = 100;
Node.OffsetX = 100;
Node.OffsetY = 100;
//Initialize the radial gradient
RadialGradient Radial = new RadialGradient();
Radial.CX = 50;
Radial.CY = 50;
Radial.FX = 50;
Radial.FY = 50;
//Initialize the stops
Collection Stops = new Collection();
Stop Stop = new Stop();
Stop.Color = "white";
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Stop.Offset = 0;
Stops.Add(Stop);
Stop = new Stop();
Stop.Color = "darkCyan";
Stop.Offset = 100;
Stops.Add(Stop);
Radial.Stops = Stops;
//Initialize the linear gradient
LinearGradient Linear = new LinearGradient();
Linear.X1 = 0;
Linear.X2 = 50;
Linear.Y1 = 0;
Linear.Y2 = 50;
//Initialize the stops
Stops = new Collection();
Stop = new Stop();
Stop.Color = "white";
Stop.Offset = 0;
Stops.Add(Stop);
Stop = new Stop();
Stop.Color = "darkCyan";
Stop.Offset = 100;
Stops.Add(Stop);
Linear.Stops = Stops;
//Sets the node gradient as linear gradient
Node.Gradient = Linear;
Model.Nodes.Add(Node);
```



Linear Gradient



Radial Gradient

Shadow

Diagram provides support to add **shadow** effect to a node that is disabled by default. It can be enabled with the Constraints property of node. The following code illustrates how to drop shadow.

C#

```
// Defines the node
BasicShape Node = new BasicShape();
Node.Name = "node1";
Node.Width = 100;
Node.Height = 100;
Node.OffsetX = 100;
Node.OffsetY = 100;
//Enables Shadow effect for a node.
Node.Constraints = NodeConstraints.Default | NodeConstraints.Shadow;
```

Copyright © 2001 - 2020 Syncfusion Inc.



The following code illustrates how to disable shadow effect at runtime.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var node = diagram.findNode("node");
var nodeConstraints = ej.datavisualization.Diagram.NodeConstraints;
//Disables Shadow effect for a node.
constraints = node.constraints & ~nodeConstraints.Shadow;
diagram.updateNode("node", { constraints: constraints });
```

Customizing Shadow

The angle, translation, and opacity of the Shadow can be customized with the Shadow property of node. The following code example illustrates how to customize shadow.

C#

```
// Defines the node
BasicShape Node = new BasicShape();
Node.Name = "node1";
Node.Width = 100;
Node.Height = 100;
Node.OffsetX = 100;
Node.OffsetY = 100;
//Enables Shadow effect for a node.
Node.Constraints = NodeConstraints.Default | NodeConstraints.Shadow;
//Customize the Node's Shadow effect
Node.Shadow.Opacity = .8f;
Node.Shadow.Angle = 50;
Node.Shadow.Distance = 9;
```



Interaction

Diagram provides support to drag, resize, or rotate the node interactively. For more information about editing a node at runtime, refer to Edit Nodes.

Copyright © 2001 - 2020 Syncfusion Inc.

Constraints

The constraints property of node allows you to enable/disable certain features. For more information about node constraints, refer to Node Constraints.

Connector

Connectors are objects used to create link between two points, nodes or ports to represent the relationships between them.



Create Connector

Connector can be created by defining the start and end points. The path to be drawn can be defined with a collection of segments.

To explore the properties of a Connector, refer to Connector Properties.

Add connectors through connectors collection

The SourcePoint and TargetPoint properties of connector allow you to define the end points of a Connector. The following code example illustrates how to add a connector through connector collection.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
//Name of the connector
Name = "Connector",
//Sets source and target points
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 200f),
};
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```



Add connector at run time

Connectors can be added at runtime with the client side method, add. The following code example illustrates how to add connector at runtime.

JS

```
// Defines JSON
var connector = {
name: "connector",
sourcePoint: {
x: 100,
y: 100
},
targetPoint: {
x: 200,
y: 200
}
};
var diagram = $("#DiagramContent").ejDiagram("instance");
// Adds to the Diagram
diagram.add(connector);
```



Connectors from palette

Connectors can be predefined and added to the symbol palette. You can drop those connectors into the Diagram, when required.

For more information about adding connectors from symbol palette, refer to Symbol Palette.

Connectors through data source

Connectors are automatically generated based on the relationships defined through the data source.

The default properties for these connectors are fetched from default settings.

For more information about data source, refer to Data Binding.

Draw connectors

Connectors can be interactively drawn by clicking and dragging on the Diagram surface by using **DrawingTool**. For more information about drawing connectors, refer to Draw Connectors.

Update Connector at runtime

The client side method, updateConnector is used to update the connectors at run time. The following code example illustrates how to update a connector at runtime.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
diagram.updateConnector("connectorName", {
lineColor: "#1BA0E2",
lineWidth: 5,
lineDashArray: "5,5"
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

Connect nodes

The SourceNode and TargetNode properties allow to define the nodes to be connected. The following code example illustrates how to connect two nodes.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
FlowShape task1 = new FlowShape(){
Name = "task1", OffsetX = 200, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } }
};
FlowShape task1 = new FlowShape(){
Name = "task2", OffsetX = 400, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 2" } }
};
//Sets nodes collection to the Diagram model
model.Nodes.Add(task1);
model.Nodes.Add(task2);
Connector connector = new Connector() {
Name = "flow1",
//Sets source and target node
SourceNode = "task1",
TargetNode = "task2"
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
//Defines the properties that carry the common values
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 50, FillColor = "darkCyan",
BorderColor = "black", Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { FontColor = "white" } }
};
ViewData["diagramModel"] = model;
return View();
}
```



Note: By default, connections are created at the intersecting point of segments and node bounds. The connection between any specific point of source and target nodes can be achieved with connection ports.

Connections with ports

The SourcePort and TargetPort properties allow to create connections between some specific points of source/target nodes. The following code example illustrates how to create port to port connections.

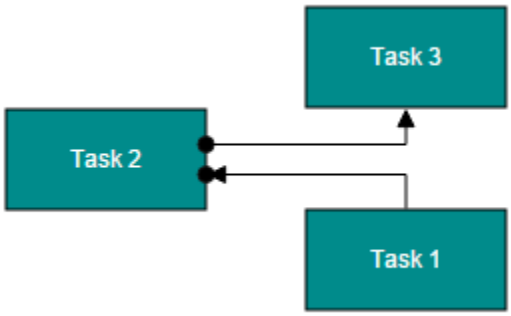
Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
FlowShape task1 = new FlowShape(){
Name = "task1", OffsetX = 350, OffsetY = 300, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } }
};
FlowShape task2 = new FlowShape(){
Name = "task2", OffsetX = 200, OffsetY = 250, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 2" } },
//Defines ports for task2
Ports = new Collection() {
new Port(){
Name="in", Offset= new DiagramPoint(1f, 0.65f), Shape = PortShapes.Circle,
Visibility = PortVisibility.Visible, FillColor = "black"
},
new Port(){
Name="out", Offset= new DiagramPoint(1f, 0.35f), Shape = PortShapes.Circle,
Visibility = PortVisibility.Visible, FillColor = "black"
}
}
};
FlowShape task3 = new FlowShape(){
Name = "task3", OffsetX = 350, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } }
};
//Sets nodes collection to the Diagram model
model.Nodes.Add(task1);
model.Nodes.Add(task2);
model.Nodes.Add(task3);
Connector connector = new Connector() {
Name = "flow1",
//Sets source and target node
SourceNode = "task1",
TargetNode = "task2",
//Name of the target port defined in the target node
TargetPort = "in"
};
Connector connector = new Connector() {
Name = "flow2",
//Sets source and target node
SourceNode = "task2",
TargetNode = "task3",
//Name of the target port defined in the target node
SourcePort = "out"
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
//Defines the properties that carry the common values
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 50, FillColor = "darkCyan",
BorderColor = "black", Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { FontColor = "white" } }
};
model.DefaultSettings.Connector = new Connector(){
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Segments = new Collection(){ new Segment(){ Type = Segments.Orthogonal }}
}
ViewData["diagramModel"] = model;
return View();
}
```



Segments

The path of the connector is defined with a collection of segments. There are three types of segments.

Straight

Straight segment allows to create a straight line.

To create a straight line, you should specify the Type of the segment as "straight" and add a straight segment to Segments collection. The following code example illustrates how to create a default straight segment.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 200f),
//Defines segment collection
//When there is no previous segment, line starts from source point
//When the end point is not specified, line ends at target point
//Defines the type of the segment
Segments = new Collection() { new Segment() { Type = Segments.Straight }}
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



The Point property of straight segment allows you to define the end point of it. The following code example illustrates how to define the end point of a straight segment.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 300f),
//Defines segment collection
//When there is no previous segment, line starts from source point
//When the end point is not specified, line ends at target point
//Defines the type of the segment
Segments = new Collection() { new Segment() {
Type = Segments.Straight,
// Defines the end point of the segment
// Additional straight line will be drawn from this end point to the target
point
Point = new DiagramPoint(100f, 200f)
}}
}
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

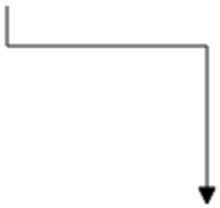
Orthogonal

Orthogonal segments are used to create segments that are perpendicular to each other.

Set the segment Type as "Orthogonal" to create a default orthogonal segment. The following code example illustrates how to create a default orthogonal segment.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 200f),
//Defines segment collection
//Defines the type of the segment
Segments = new Collection() { new Segment() { Type = Segments.Orthogonal }}
}
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```



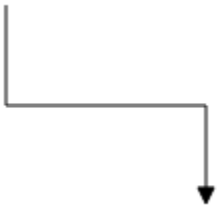
The Length and Direction properties allow to define the flow and length of segment. The following code example illustrates how to create customized orthogonal segments.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 200f),
//Defines segment collection
Segments = new Collection() { new Segment() {
//Defines the type of the segment
Type = Segments.Orthogonal,
// Orthogonal segment of 50px length to the bottom
Length = 50,
// Additional orthogonal segments will be added from the end of the last
segment to the target point
Direction = "right"
}}
```

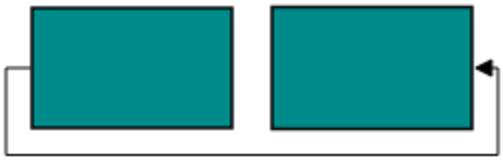
Copyright © 2001 - 2020 Syncfusion Inc.

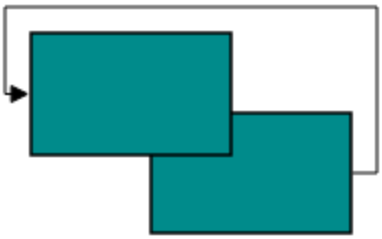
```
}
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```



Avoid overlapping

Orthogonal segments are automatically re-routed, in order to avoid overlapping with the source and target nodes. The following images illustrate how orthogonal segments are re-routed.





Note: Overlapping with source and target nodes are only avoided. Other nodes are not considered as obstacles.

Bezier

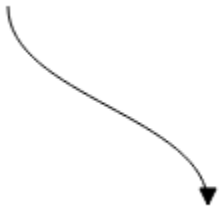
Bezier segments are used to create curve segments and the curves are configurable either with the control points or with vectors.

To create a bezier segment, the Segment.Type is set as Bezier. The following code example illustrates how to create a default Bezier segment.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 200f),
```

```
//Defines segment collection
Segments = new Collection() { new Segment() {
//Defines the type of the segment
Type = Segments.Bezier
}}
}
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```

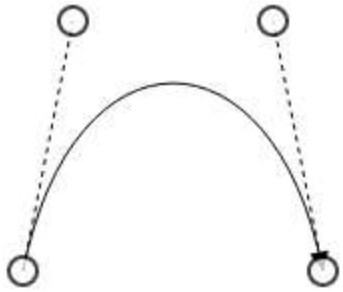


The Point1 and Point2 properties of bezier segment enable you to set the control points. The following code example illustrates how to configure the Bezier segments with control points.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 200f),
TargetPoint = new DiagramPoint(250f, 200f),
//Defines segment collection
Segments = new Collection() { new Segment() {
//Defines the type of the segment
Type = Segments.Bezier,
//First control point: an absolute position from the page origin
Point1 = new DiagramPoint(125f, 75f),
//Second control point: an absolute position from the page origin
Point2 = new DiagramPoint(225f, 75f)
}}
}
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



The Vector1 and Vector2 properties of bezier segment enable you to define the vectors. The following code illustrates how to configure a bezier curve with vectors.

JS

```
//Defines JSON
var connector = {
name: "connector",
sourcePoint: { x: 100, y: 200 },
targetPoint: { x: 250, y: 200 },
//Defines segment collection
segments: [
{
// Defines the type of the segment
type: "bezier",
// Length and angle between the source point and the first control point
vector1: { angle: 270, distance: 75 },
// Length and angle between the target point and the second control point
vector2: { angle: 270, distance: 75 }
}]
};
connectors.push(connector);
```



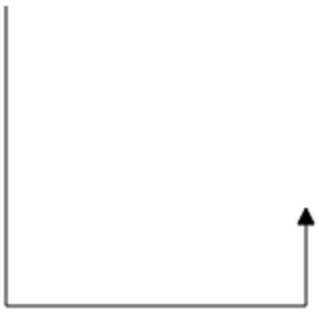
Complex segments

Multiple segments can be defined one after another. To create a connector with multiple segments, define and add the segments to Connector.Segments collection. The Following code example illustrates how to create a connector with multiple segments.

C#

```
public ActionResult Index()
{
```

```
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1",
SourcePoint = new DiagramPoint(100f, 200f), TargetPoint = new
DiagramPoint(250f, 300f),
//Defines segment collection
Segments = new Collection() {
// Segment of length 100px to the bottom
new Segment() { Type = Segments.Orthogonal, Length = 100, Direction =
"bottom" },
// Segment of length 150px to the bottom
//Additional orthogonal segments will be added from the end of the last
segment to the target point
new Segment() { Type = Segments.Orthogonal, Length = 150, Direction =
"right" }
}
}
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```



Decorator

Start and end points of a connector can be decorated with some customizable shapes like arrows, circles, diamond or path. You can decorate the connection end points with the SourceDecorator and TargetDecorator properties of connector.

To explore the properties of decorators, refer to Decorator Properties.

The Shape property of decorator allows to define the shape of the decorators. The following code example illustrates how to create decorators of various shapes.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector1 = new Connector() {
Name = "connector1",SourcePoint = new DiagramPoint(100f, 100f), TargetPoint
= new DiagramPoint(200f, 200f),
//Decorator shape - Circle
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
SourceDecorator = new Decorator() { Shape = DecoratorShapes.Circle, Width =
10, Height = 10 },
//Decorator shape - Arrow
TargetDecorator = new Decorator() { Shape = DecoratorShapes.Arrow, Width =
10, Height = 10 }
};
Connector connector2 = new Connector() {
Name = "connector2",SourcePoint = new DiagramPoint(300f, 100f), TargetPoint
= new DiagramPoint(400f, 200f),
//Decorator shape - Diamond
SourceDecorator = new Decorator() { Shape = DecoratorShapes.Diamond, Width =
10, Height = 10 },
//Decorator shape - Open Arrow
TargetDecorator = new Decorator() { Shape = DecoratorShapes.OpenArrow, Width
= 10, Height = 10 }
};
Connector connector3 = new Connector() {
Name = "connector3",SourcePoint = new DiagramPoint(500f, 100f), TargetPoint
= new DiagramPoint(600f, 200f),
//Decorator shape - Path
TargetDecorator = new Decorator() { Shape = DecoratorShapes.Path, PathData =
"M 376.892,225.284L 371.279,211.95L 376.892,198.617L 350.225,211.95L
376.892,225.284 Z" }
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector1);
model.Connectors.Add(connector2);
model.Connectors.Add(connector3);
ViewData["diagramModel"] = model;
return View();
}
```



Padding

Padding is used to leave space between the Connector's end point and the object to where it is connected.

The SourcePadding and TargetPadding properties of connector define the space to be left between the connection end points and the source and target nodes of connector. The following code example illustrates how to leave space between the connection end points and source, target nodes.

C#

```
public ActionResult Index()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DiagramProperties model = new DiagramProperties();
FlowShape task1 = new FlowShape(){
Name = "task1", OffsetX = 200, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } }
};
FlowShape task2 = new FlowShape(){
Name = "task2", OffsetX = 400, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 2" } }
};
model.Nodes.Add(task1);
model.Nodes.Add(task2);
model.Nodes.Add(new Connector() {
Name = "flow1", SourceNode = "task1", TargetNode = "task2",
//Space between source point and source object
SourcePadding = 5,
//Space between target point and target object
TargetPadding = 10
});
//Defines the properties that carry the common values
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 50, BorderColor = "black", FillColor = "darkCyan",
Labels = new Collection() { new Label() { FontColor = "white" } }
}
ViewData["diagramModel"] = model;
return View();
}
```



The ConnectorPadding property of node defines the space to be left between the node bounds and its edges. The following code example illustrates how to leave the space between a node and its connections.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
FlowShape task1 = new FlowShape() {
Name = "task1", OffsetX = 200, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } },
//Space between the node and its edges
ConnectorPadding = 5
};
FlowShape task2 = new FlowShape() {
Name = "task2", OffsetX = 400, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 2" } }
};
model.Nodes.Add(task1);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
model.Nodes.Add(task2);
model.Connectors.Add( new Connector() { Name = "flow1", SourceNode =
"task1", TargetNode = "task2" });
//Sets the default properties for nodes
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 50, BorderColor = "black", FillColor = "darkCyan",
Labels = new Collection() { new Label() { FontColor = "white" } }
};
ViewData["diagramModel"] = model;
return View();
}
```



The ConnectorPadding property of port defines the space between the ports and its in/out edges. The following code example illustrates how to leave the space between ports and its connections.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
FlowShape task1 = new FlowShape() {
Name = "task1", OffsetX = 200, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } }
};
FlowShape task2 = new FlowShape() {
Name = "task2", OffsetX = 400, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 2" } },
//Defines ports for task2
Ports = new Collection() {
new Port(){
Name = "port", Offset = new DiagramPoint(0f, 0.5f), Shape =
PortShapes.Circle,
Visibility = PortVisibility.Visible, FillColor = "black",
//Space between port and its edges
ConnectorPadding = 5
}
}
};
model.Nodes.Add(task1);
model.Nodes.Add(task2);
model.Connectors.Add(new Connector() { Name = "flow1", SourceNode = "task1",
TargetNode = "task2", TargetPort = "port"});
//Sets the default properties for nodes
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 50, BorderColor = "black", FillColor = "darkCyan",
Labels = new Collection() { new Label() { FontColor = "white" } }
}
ViewData["diagramModel"] = model;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
```



Bridging

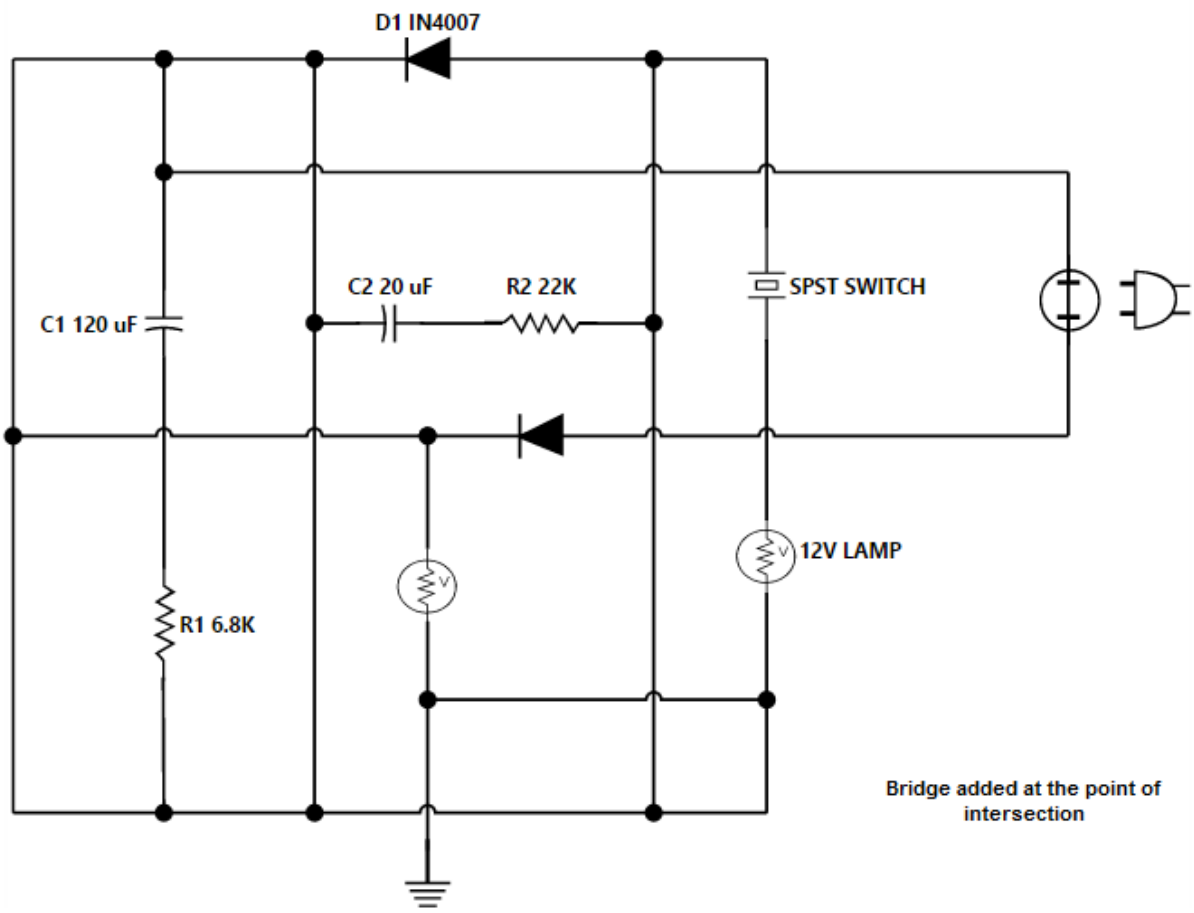
Line Bridging creates a bridge for lines to smartly cross over other lines, at points of intersection. When two line connectors meet each other, the line with the higher z-order (upper one) draws an arc over the underlying connector.

Bridging can be enabled/disabled either with the Connector.Constraints or Diagram.Constraints. The following code example illustrates how to enable line bridging.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "Connector1", SourcePoint = new DiagramPoint(100f, 100f), TargetPoint
= new DiagramPoint(200f, 200f),
//Removes inherit bridging or else bridging is enabled/disabled based on the
Diagram constraints
//And includes bridging
Constraints = ConnectorConstraints.Default &
~ConnectorConstraints.InheritBridging | ConnectorConstraints.Bridging
};
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



The direction of the bridge can be customized with the property BridgeDirection. BridgeDirection defines the intersecting segment where the bridge has to be inserted. By default, the bridge direction points to the top.

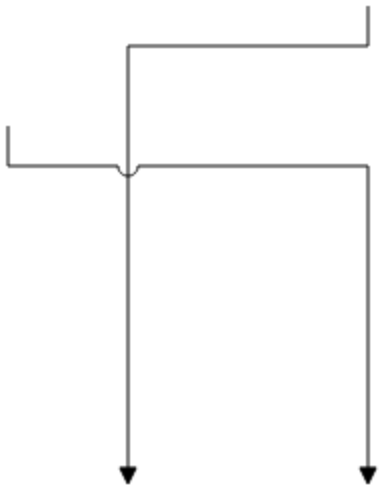
To explore the bridge directions, refer to Bridge Directions.

The following code example illustrates how to draw the bridge at the bottom direction.

RAZOR

```
@using Syncfusion.JavaScript.DataVisualization.DiagramEnums
@using Syncfusion.JavaScript.DataVisualization.Models.Diagram
@(Html.EJ().Diagram("Diagram").Width("100%").Height("600px")
@*Sets the bridge direction*@
.BridgeDirection(BridgeDirection.Bottom)
@*Enables bridging for every connector added in the model*@
.Constraints(DiagramConstraints.Default | DiagramConstraints.Bridging)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Limitation: Bezier segments do not support bridging.

Corner radius

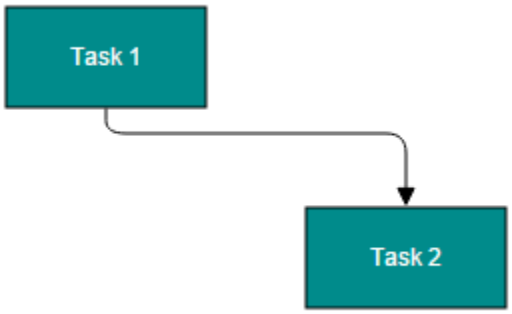
Corner radius allows to create connectors with rounded corners. The radius of the rounded corner is set with CornerRadius property.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
FlowShape task1 = new FlowShape() {
Name = "task1", OffsetX = 200, OffsetY = 200, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 1" } }
};
FlowShape task1 = new FlowShape() {
Name = "task2", OffsetX = 350, OffsetY = 300, Shape = FlowShapes.Process,
Labels = new Collection() { new Label() { Text = "Task 2" } },
};
Connector connector = new Connector() {
Name = "flow1",
SourceNode = "task1",
TargetNode = "task2",
//Sets the radius for the rounded corner
CornerRadius = 10
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
//Sets the default properties for nodes
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 50, BorderColor = "black", FillColor = "darkCyan",
Labels = new Collection() { new Label() { FontColor = "white" } }
};
model.DefaultSettings.Connector = new Connector(){
Segments = new Collection(){ new Segment(){ Type = Segments.Orthogonal }}
}
ViewData["diagramModel"] = model;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
```



Appearance

Stroke width, stroke color, and style of the lines and decorators can be customized with a set of defined properties.

Segment Appearance

The following code example illustrates how to customize the segment appearance.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
Name = "connector", SourcePoint = new DiagramPoint(100f, 100f), TargetPoint
= new DiagramPoint(200f, 200f),
//Stroke width of the line
LineWidth = 2,
//Stroke color
LineColor = "green",
//Line style
LineDashArray = "2,2",
//Opacity of the line
Opacity = 0.8,
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```

Decorator Appearance

The following code example illustrates how to customize the appearance of the decorator.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector = new Connector() {
//Customizes the appearance of decorator
```

```
TargetDecorator = = new Decorator(){
//Defines the shape
Shape = DecoratorShapes.Arrow,
//Fills color of the decorator
FillColor = "red",
//Stroke color
BorderColor = "green",
//Stroke width
BorderWidth = 2,
Width = 10,
Height = 10
}
};
//Sets connectors collection to the Diagram model
model.Connectors.Add(connector);
ViewData["diagramModel"] = model;
return View();
}
```



Interaction

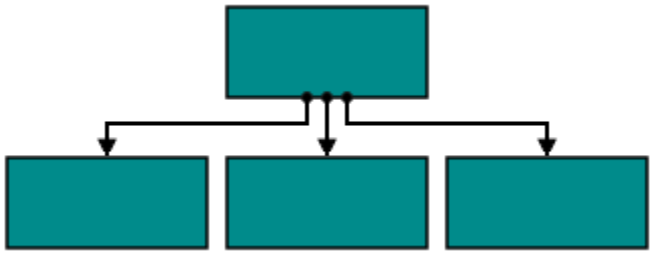
Diagram allows to edit the connectors at runtime. To edit the connector segments at runtime, refer to Connection Editing.

Constraints

The Constraints property of connector allows to enable/disable certain features of connectors. For more information about constraints, refer to Connector Constraints.

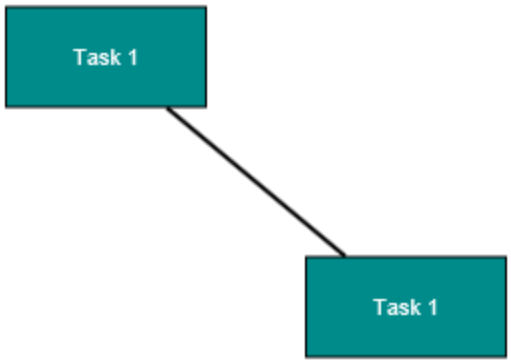
Port

Essential Diagram for JS provides support to define custom ports for making connections.

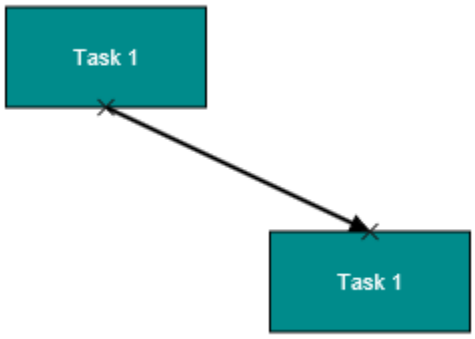


When a connector is connected between two nodes, its end points are automatically docked to node's nearest boundary as shown in the following image.

Copyright © 2001 - 2020 Syncfusion Inc.



Ports act as the connection points of node and allows to create connections with only those specific points as shown in the following image.



Create Port

Add ports when initializing nodes

To add a connection port, you need to define the port object and add it to node's Ports collection. The Offset property of port accepts an object of fractions and used to determine the position of ports. The following code illustrates how to add ports when initializing the node.

C#

```
BasicShape Node = new BasicShape();
Node.Name = "node1";
Node.Width = 100;
Node.Height = 100;
Collection Ports = new Collection();
Port port = new Port();
port.Name = "port1";
port.Offset = new DiagramPoint(0, 0.5f);
Node.Ports = Ports;
Model.Nodes.Add(Node);
```

Add ports at runtime

You can add ports at runtime by using the client side method addPorts. The following code illustrates how to add ports to node at runtime.

JS

```
// Defines a collection of ports that have to be added at runtime
var ports = [
{
name: "port1",
// Specifies the port offset - fraction value relative
to node bounds - determines the position of port on node
offset: {
x: 0,
y: 0.5
}
},
{ name: "port2",offset: {x: 1,y: 0.5 }},
{ name: "port3",offset: {x: 0.5,y: 0 }},
{ name: "port4",offset: {x: 0.5,y: 1 }}
];
// Gets the instance for the Diagram
var diagram = $("#diagram").ejDiagram("instance");
// Adds the ports to the node of name "node"
diagram.addPorts("node", ports)
```



To explore the set of properties for defining a port, refer to Port Properties

Update Port at runtime

The client side API updatePort is used to update the ports at run time. The following code example illustrates how to change the port properties.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var selectedObject = diagram.model.selectedItems.children[0];
var visibility = ej.datavisualization.Diagram.PortVisibility.Visible;
diagram.updatePort(selectedObject.name, selectedObject.ports[0], {
fillColor: "red", visibility: visibility });
```

Connect with ports

Connector's SourcePort and TargetPort properties allow to create connections between some specific points of source/target nodes.

For more information about creating connections with port, refer to Connections with ports

Appearance

You can change the shape of port by using its Shape property. To explore the different types of port shapes, refer to Port Shapes.

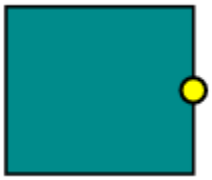
The appearance of ports can be customized with a set of style specific properties.

The following code illustrates how to change the appearance of port.

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
BasicShape Node = new BasicShape();
Collection Ports = new Collection();
Port Port = new Port();
//Defines the shape of port
Port.Shape = PortShapes.Circle;
//Specifies the port visibility
Port.Visibility = PortVisibility.Visible;
//Customizes the appearance
Port.FillColor = "yellow";
Port.Size = 12;
Port.BorderColor = "black";
Port.BorderWidth = 2;
// Specifies the port position
Port.Name = "port1";
Port.Offset = new DiagramPoint(1, 0.5f);
Node.Ports = Ports;
Model.Nodes.Add(Node);
```



Constraints

The Constraints property allows to enable/disable certain behaviors of ports. For more information about port constraints, refer to Port Constraints

Label

Label is a block of text that can be displayed over a node or connector. Label is used to textually represent an object with a string that can be edited at run time. You can add Multiple Labels to a node/connector.

Create Label

You can add a label to a node/connector by defining the label object and adding that to the Labels collection of node/connector. The Text property of label defines the text to be displayed. The following code illustrates how to create a Label.

RAZOR

```
@*Initializes diagram control*@
<div>
@Html.EJ().Diagram("diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
</div>
```

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
//Initializes labels collection
Labels = new Collection(){
new Label()
{
//Defines the text to be displayed
Text= "Label"
}
}
});
model.Connectors.Add(new Connector()
{
Name = "connector1",
SourcePoint = new DiagramPoint(200, 50),
TargetPoint = new DiagramPoint(300, 150),
Segments = new Collection() {
new Segment() { Type = Segments.Orthogonal, Direction = "bottom", Length =
50 } },
//Initializes labels collection
Labels = new Collection(){
new Label()
{
//Defines the text to be displayed
Text= "Label",
FillColor = "white"
}
}
});
ViewData["diagramModel"] = model;
```


To explore more label properties, refer to Label Properties.

Update Label at runtime

The client side API updateLabel is used to update the labels at run time.

The following code example illustrates how to change the label properties.

C#

```
var diagram = $("#diagram").ejDiagram("instance");
var selectedObject = diagram.model.selectedItems.children[0];
diagram.updateLabel(selectedObject.name, selectedObject.labels[0], { text:
"label", fillColor: "red" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

Alignment

Label can be aligned relative to the node boundaries. It has margin, offset, horizontal and vertical alignment settings. It is quite tricky when all four alignments are used together but gives you more control over alignment.

Offset

The Offset property of label is used to align the labels based on fractions. 0 represents top/left corner, 1 represents bottom/right corner, and 0.5 represents half of width/height.

The following image shows the relationship between the label position (black colored circle) and offset (fraction values).

Horizontal and vertical alignments

The HorizontalAlignment property of label is used to set how the label is horizontally aligned at the label position determined from the fraction values. The VerticalAlignment property is used to set how label is vertically aligned at the label position.

The following tables illustrates all the possible alignments visually with **offset (0, 0).**

| Horizontal Alignment | Vertical Alignment | Output with Offset(0,0) |

|---|---|---|

| Left | Top | |

| Center | | |

| Right | | |

| Left | Center | |

| Center | | |

| Right | | |

| Left | Bottom | |

| Center | | |

| Right | | |

The following codes illustrates how to align labels.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Labels = new Collection(){
new Label(){
Text= "Label",
```

```
// Sets offset to label
Offset= new DiagramPoint(0,0.5f),
// Sets to align label horizontally relative to given offset
HorizontalAlignment= HorizontalAlignment.Left,
// Sets to align label vertically relative to given offset
VerticalAlignment= VerticalAlignment.Center,
// Sets text alignment to label
TextAlign= TextAlign.Center
}
}
});
ViewData["diagramModel"] = model;
```


Margin

Margin is an absolute value used to add some blank space in any one of its four sides. You can displace the labels with the Margin property.

The following code example illustrates how to align a label based on its Offset, HorizontalAlignment, VerticalAlignment and Margin values.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Labels = new Collection(){
new Label(){
Text= "Label",
Offset=new DiagramPoint(0.5f,1),
HorizontalAlignment= HorizontalAlignment.Center,
VerticalAlignment= VerticalAlignment.Top,
//Sets margin to add label outside a node
Margin= new LabelMargin(){Top=10}
}
}
});
ViewData["diagramModel"] = model;
```


Text Alignment

The TextAlign property of label allows you to set how the text should be aligned (left, right, center, or justify) inside the text block. The following codes illustrate how to set textAlign for a label.

C#

```
DiagramProperties model = new DiagramProperties();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Labels = new Collection(){
new Label(){
Text= "Text Align is set as Left",
// Sets text alignment for a label
TextAlign = TextAlign.Left
}
}
});
ViewData["diagramModel"] = model;
```


| TextAlign | Image |

|---|---|

| Left | |

| Right | |

| Center | |

| Justify | |

Wrapping

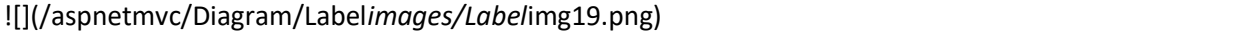
When text overflows node boundaries, you can control it by using text wrapping. So, it is wrapped into multiple lines. The Wrapping property of label defines how the text should be wrapped. The following code illustrates how to wrap a text in a node.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Labels = new Collection(){
new Label(){
Text= "Text-Wrapping",
//Enables Text-wrapping
Wrapping = TextWrapping.Wrap
}
}
});
```

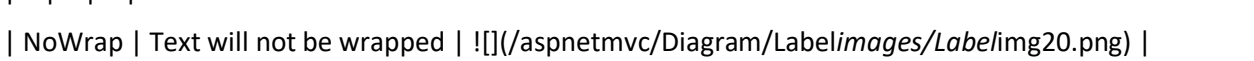
Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewData["diagramModel"] = model;
```

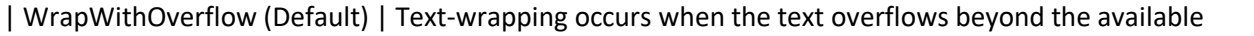


| Values | Description | Image |

|---|---|---|

| NoWrap | Text will not be wrapped |  |

| Wrap | Text-wrapping occurs when the text overflows beyond the available node width. |  |

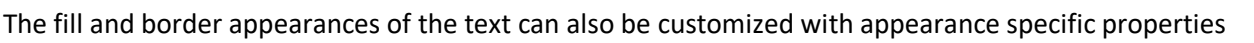
| WrapWithOverflow (Default) | Text-wrapping occurs when the text overflows beyond the available node width. However, the text may overflow beyond the node width in the case of a very long word. |  |

Appearance

You can change the font style of the labels with the font specific properties(`FontSize`,`FontFamily`,`FontColor`.,). The following code illustrates how to customize the appearance of a label.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Labels = new Collection(){
new Label(){
Text= "Label Text",
//Sets styles to a label
FontSize= 12,
FontFamily= "TimesNewRoman",
FontColor= "black",
Bold= true,
Italic= true,
TextDecoration= TextDecorations.Underline
}
}
});
ViewData["diagramModel"] = model;
```



The fill and border appearances of the text can also be customized with appearance specific properties of label.The following code illustrates how to customize background and border of a label.

C#

```
DiagramProperties model = new DiagramProperties();
```

```
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Labels = new Collection(){
new Label(){
Text= "Label Text",
//Customizes background and borders of a label
FillColor= "white",
BorderColor= "black",
BorderWidth= 1
}
}
});
ViewData["diagramModel"] = model;
```


Drag

A **Label** can be displaced from its original position to any preferred location interactively. Dragging is disabled by default. You can enable label dragging with the Constraints property of node/connector. The following code illustrates how to enable label **dragging**.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Constraints=NodeConstraints.Default | NodeConstraints.DragLabel,
//Initializes labels collection
Labels = new Collection(){
new Label()
{
//Defines the text to be displayed
Text= "Label"
}
}
});
model.Connectors.Add(new Connector()
{
Name = "connector1",
SourcePoint = new DiagramPoint(200, 50),
TargetPoint = new DiagramPoint(300, 150),
Segments = new Collection() {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
new Segment() { Type = Segments.Orthogonal, Direction = "bottom", Length =
50 } },
Constraints=ConnectorConstraints.Default | ConnectorConstraints.DragLabel,
//Initializes labels collection
Labels = new Collection(){
new Label()
{
//Defines the text to be displayed
Text= "Label",
FillColor = "white"
}
}
});
ViewData["diagramModel"] = model;
```


Rotate

You can rotate the labels to any desired angle. Labels are rotated to the angle that is defined by the RotateAngle property of label. The following code illustrates how to rotate a label.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Constraints = NodeConstraints.Default | NodeConstraints.DragLabel,
//Initializes labels collection
Labels = new Collection(){
new Label()
{
Text="Label",
//Sets label's rotation Angle
RotateAngle=45
}
}
});
ViewData["diagramModel"] = model;
```


Note: There is no built-in support to rotate labels interactively.

Edit

Diagram provides support to edit a Label at runtime, either programmatically or interactively. By default, label is in **View** mode. But it can be brought to edit mode in two ways;

1. By double-clicking the label.

2. By selecting the item and pressing the **F2** key.

Double-clicking any label will enables **editing** of that. Double-clicking the node enables first label editing. When the focus of editor is lost, the label for the node is updated.

You can programmatically edit the label by changing the Mode of the label. The following code illustrates how to edit the label programmatically.

RAZOR

```
var diagram = $("#diagram").ejDiagram("instance");
var node = diagram.model.selectedItems.children[0];
//Sets label mode as Edit
var options = {
mode: ej.datavisualization.Diagram.LabelEditMode.Edit
};
diagram.updateLabel(node.name, node.labels[0], options);
```


Read Only labels

Diagram allows to create read only labels. You have to set the ReadOnly property of label to enable/disable the read only mode. The following code illustrates how to enable **readOnly** mode.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
Constraints = NodeConstraints.Default | NodeConstraints.DragLabel,
//Initializes labels collection
Labels = new Collection(){
new Label()
{
Text="Label",
ReadOnly=true
}
}
});
ViewData["diagramModel"] = model;
```

Drag Limit

Diagram label dragging can be restricted by enabling/disabling DragLimit constraints of the connectors. Also we can define the drag limit in all direction using DragLimit property of the connector. Drag limit only applicable for connector labels.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
DiagramProperties model = new DiagramProperties();
Collection Labels = new Collection();
Labels.Add(new Label()
{
// Defines the drag limit property in all direction
DragLimit = new LabelMargin()
{
Left = 30,
Right = 30,
Bottom = 30,
Top = 30
}
});
model.Connectors.Add(new Connector()
{
Name = "node",
// Enable DragLimit constraints to the connector.
Constraints = ConnectorConstraints.Default | ConnectorConstraints.DragLabel
| ConnectorConstraints.DragLimit,
Labels = Labels
});
ViewData["diagramModel"] = model;
```

Multiple labels

You can add any number of labels to a node or connector. The following code illustrates how to add multiple labels to a node.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
BorderColor = "black",
FillColor = "#1BA0E2",
//Initializes labels collection
Labels = new Collection(){
new Label()
{
Text="Left",
Offset=new DiagramPoint(0.12f, 0.1f)
},
new Label(){
Text="Center",
Offset= new DiagramPoint(0.5f, 0.5f)
},
new Label(){
Text="Right",
Offset=new DiagramPoint(0.82f, 0.9f)
}
}
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewData["diagramModel"] = model;
```


Limitation

- To enable faster rendering, labels are rendered in a separate layer because of this, all the labels always stay on top. When two nodes are overlapped, text of underlying node is not hidden by the overlapped node.

| Expected behavior | Current behavior |

|---|---|

| | |

Group

Group is used to cluster multiple nodes and connectors into a single element. It acts like a container for its children (nodes, groups, and connectors). Every change made to the group also affects the children. Child elements can be edited individually.

Create Group

Add group when initializing diagram

You can add a group to the Diagram model through Nodes collection. You can create Group like node and you need to add the child objects to the Children collection of the group. The following code illustrates how to create a group node.

C#

```
BasicShape node1 = new BasicShape(){
Name = "rectangle1",
OffsetX = 100,
OffsetY = 100,
Width = 100,
Height = 100,
FillColor = "darkCyan",
BorderWidth = 2,
};
node1.Labels.Add(new Label() { Text = "Rectangle1" });
BasicShape node2 = new BasicShape()
{
Name = "rectangle2",
OffsetX = 200,
OffsetY = 200,
Width = 100,
Height = 100,
FillColor = "darkCyan",
BorderWidth = 2,
};
node2.Labels.Add(new Label() { Text = "Rectangle2" });
Group group = new Group() { Name = "group1" };
group.Children.Add(node1);
group.Children.Add(node2);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Diagram.Model.Nodes.Add(group);
```

Add group at run time

You can add a group node at runtime by using the client side method add.

The following code illustrates how a group node is added at run time.

JS

```
var group = {
name: "group1",
type: "group",
children: [{
name: "rectangle1",
offsetX: 100,
offsetY: 100,
width: 100,
height: 100,
type: "node",
fillColor: "darkCyan",
borderWidth: 2,
labels: [{
text: "rectangle1"
}]
}, {
name: "rectangle2",
offsetX: 200,
offsetY: 200,
width: 100,
height: 100,
type: "node",
fillColor: "darkCyan",
borderWidth: 2,
labels: [{
text: "rectangle2"
}]
}]
};
var diagram = $("#DiagramContent").ejDiagram("instance");
// Adds group to the Diagram.
diagram.add(group);
```

Group from palette

Group nodes can be predefined and added to symbol palette. You can drop those groups into Diagram, when required.

To explore how to add groups from symbol palette, refer to Symbol Palette

Container

Containers are used to automatically measure and arrange the size and position of the child elements in a predefined manner.

There are two types of containers available.

Copyright © 2001 - 2020 Syncfusion Inc.

Canvas

- The Canvas panel supports absolute positioning and provides the least layout functionality to its contained Diagram elements.
- Canvas allows you to position its contained elements by using margin and alignment properties.
- It allows elements to be either vertically or horizontally aligned.

The `Container` property of group should be defined and its `Type` should be set as `Canvas` to create a canvas panel. The following code illustrates how to add a canvas panel.

C#

```
BasicShape node1 = new BasicShape()
{
Name = "node1",
FillColor = "darkCyan",
Width = 100,
Height = 100,
MarginTop = 0,
MarginLeft = 0,
};
BasicShape node2 = new BasicShape()
{
Name = "node2",
FillColor = "white",
Width = 100,
Height = 100,
// Sets the margin to define the space around the child node.
MarginTop = 30,
MarginLeft = 30,
};
BasicShape node3 = new BasicShape()
{
Name = "node3",
FillColor = "darkCyan",
Width = 100,
Height = 100,
MarginTop = 60,
MarginLeft = 60,
};
BasicShape node4 = new BasicShape()
{
Name = "node4",
FillColor = "white",
Width = 100,
Height = 100,
MarginTop = 90,
MarginLeft = 90,
};
Group group = new Group()
{
Name = "canvas",
OffsetX = 400,
OffsetY = 400,
//Sets the container as canvas.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Container = new Container() { Type = ContainerType.Canvas },
FillColor = "#E7EBF4",
BorderColor = "black",
//Sets the padding to give space between the group border and group content.
PaddingLeft = 30,
PaddingTop = 30,
PaddingRight = 30,
PaddingBottom = 30,
};
group.Children.Add(node1);
group.Children.Add(node2);
group.Children.Add(node3);
group.Children.Add(node4);
Diagram.Model.Nodes.Add(group);
```



Stack

- Stack panel is used to arrange its children in a single line or stack order, either vertically or horizontally.
- It controls spacing by setting margin properties of child and padding properties of group. By default, a Stack Panel's Orientation is vertical.

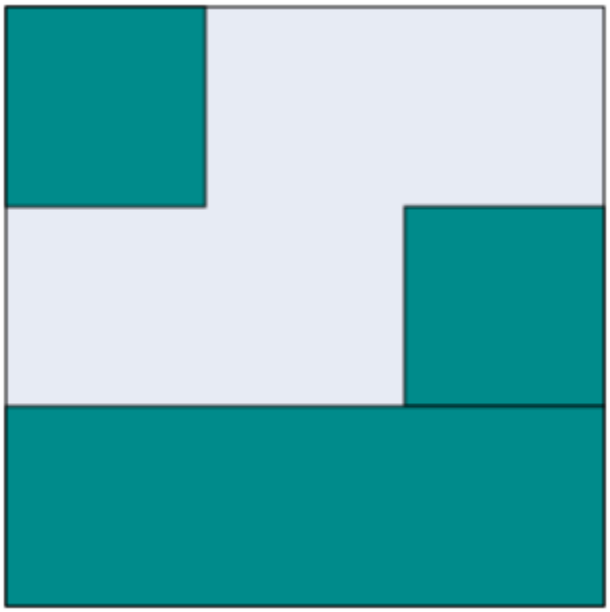
The Container property of group should be defined and its Type should be set as Stack to create a canvas panel The following code illustrates how to add a stack panel.

C#

```
Node node1 = new Node() {
Name = "node1", FillColor = "darkCyan", Width = 100, Height = 100,
//Sets the horizontal Alignment for child node.
HorizontalAlign = HorizontalAlignment.Left
};
Node node2 = new Node() {
Name = "node2", FillColor = "darkCyan", Width = 100, Height = 100,
HorizontalAlign = HorizontalAlignment.Right
};
Node node3 = new Node()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
Name = "node3", FillColor = "darkCyan", Width = 100, Height = 100,
HorizontalAlign = HorizontalAlignment.Stretch
};
Group group = new Group() {
Name = "stack",
OffsetX = 600,
OffsetY = 200,
//Sets the container as canvas.
Container = new Container() { Type = ContainerType.Stack },
FillColor = "#E7EBF4",
BorderColor = "black",
// Sets the minimum size for stack panel.
MinHeight = 300,
MinWidth = 300,
};
group.Children.Add(node1);
group.Children.Add(node2);
group.Children.Add(node3);
Diagram.Model.Nodes.Add(group);
```



Difference between a basic group and containers

| Group | Container |

|---|---|

| It arranges the child elements based on the child elementâ??s position and size properties. | Each container has a predefined behaviour to measure and arrange its child elements. Canvas and stack containers are supported in the Diagram. |

| Padding, Min and Max Size properties are not applicable for basic group. | It is applicable for container. |

| Children's margin and alignment properties are not applicable for basic group. | It is applicable for container. |

Copyright © 2001 - 2020 Syncfusion Inc.

Interaction

You can edit the group and its children at runtime. For more information about how to interact with a group, refer to Edit Groups.

Swim lane

Swim-lane Diagrams are typically used to visualize the relationship between a business process and the department responsible for it by focusing on the logical relationships between activities. Swimlanes may be arranged either horizontally or vertically.

Initialize the Diagram model

Create a Object for diagram properties model element that can be used to initialize the diagram widget as shown below.

C#

```
//Initialize the Diagram Model
DiagramProperties Model = new DiagramProperties();
```

Create a swimlane

To create a swimlane, you need to use the Swimlane class. By default, the swimlanes are arranged vertically. You can change that with the Orientation property of swimlane.

The following code example illustrates how to define a swimlane object.

C#

```
//Disables tooltip for any node/connector
DiagramProperties Model = new DiagramProperties();
SwimLane Swimlane = new SwimLane();
Swimlane.Name = "swimlaneNode";
//Changes the orientation
Swimlane.Orientation = "horizontal";
//Sets the position and size
Swimlane.OffsetX = 400;
Swimlane.OffsetY = 200;
Swimlane.Height = 100;
Swimlane.Width = 700;
//Sets the type of object as swimlane
Swimlane.IsSwimlane = true;
```

Add swimlane into diagram

Adding a swimlane to the Diagram is same as adding a node. You can add either through the Nodes collection or through the client side method add. You can also drag and drop a swimlane from symbol palette.

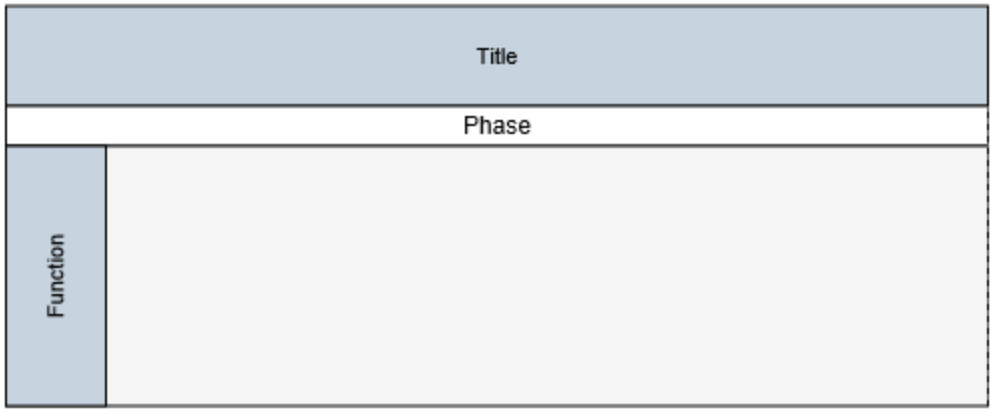
For more information about adding a node/swimlane to the Diagram, refer to Add Nodes.

The following code example illustrates how to add a swimlane to the Diagram through Nodes collection.

C#

```
//Add the swimlane in to the Nodes collection
Model.Nodes.Add(Swimlane);
```

Copyright © 2001 - 2020 Syncfusion Inc.



Headers

Swimlane allows to define a header to textually describe it. The Header property of swimlane allows you to define its textual description(Text) and to customize its appearance. The following code example illustrates how to define swimlane header.

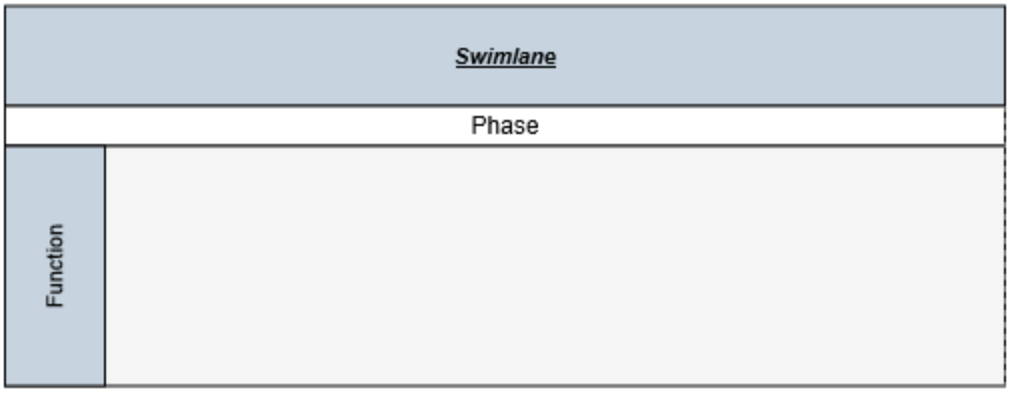
C#

```
//Defines the header and format its text
Header Header = new Header();
Header.Text = "Swimlane";
Header.Height = 50;
Header.FillColor = "#C7D4DF";
Header.FontColor = "black";
Header.FontSize = 11;
Header.FontFamily = "Arial";
Header.Italic = true;
Header.Bold = true;
Header.TextDecoration = "underline";
SwimLane Swimlane = new SwimLane();
Swimlane.Name = "swimlaneNode";
//Changes the orientation
Swimlane.Orientation = "horizontal";
//Sets the position and size
Swimlane.OffsetX = 400;
Swimlane.OffsetY = 200;
Swimlane.Height = 100;
Swimlane.Width = 700;
//Sets the type of object as swimlane
Swimlane.IsSwimlane = true;
//Sets the swimlane header
Swimlane.Header = Header;
//Add the swimlane in to the Nodes collection
Model.Nodes.Add(Swimlane);
```

JS

```
//Updates the swimlane header at runtime
var diagram = $("#diagram").ejDiagram("instance");
diagram.updateNode("swimlaneName", { header: {fontColor:"white"} })
```

Copyright © 2001 - 2020 Syncfusion Inc.



Update Header

Swimlane headers can be updated at runtime with the client side method updateNode. The following code example illustrates how to update a lane's header at runtime.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Defines the header and format its text
var header = {
text: "swimlane",
bold: true,
italic: true
};
diagram.updateNode("swimlane", { header: header });
```

Disable headers

You can hide the swimlane headers. The following code example illustrates how to hide headers.

C#

```
//Defines the header
Header Header = new Header();
//Sets "0" to hide header
Header.Height = 0;
```

Lane

Lane is a functional unit or a responsible department of a business process that helps to map a process within the functional unit or in between other functional units.

You can add any number of lanes to a swimlane and the lanes are automatically stacked inside a swimlane based in the order they are added.

Create an empty lane

To create an empty lane, you need to define an object with IsLane property that helps identify the object as a lane. The following example illustrates how to define a swimlane with a lane.

C#

```
SwimLane Swimlane = new SwimLane();
Swimlane.Name = "swimlaneNode";
//Changes the orientation
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Swimlane.Orientation = "horizontal";
//Sets the position and size
Swimlane.OffsetX = 400;
Swimlane.OffsetY = 200;
Swimlane.Height = 100;
Swimlane.Width = 700;
Swimlane.Header = Header;
//Sets the type of object as swimlane
Swimlane.Type = "swimlane";
//Defines an empty lanes Collection
Collection Lanes = new Collection();
Lane Lane = new Lane();
Lane.Name = "lane1";
Lane.FillColor = "#f5f5f5";
Lane.Height = 120;
//Sets the object as Lane
Lane.IsLane = true;
Lanes.Add(Lane);
Swimlane.Lanes = Lanes;
//Add the swimlane in to the Nodes collection
Model.Nodes.Add(Swimlane);
```

Create a lane with header

The Header property of the lane allows you to textually describe the lane(text) and to customize the appearance of the description. The following code example illustrates how to define a lane header.

You can limit the size of a lane with its MinWidth, MinHeight, MaxWidth, and MaxHeight properties.

C#

```
//Defines the header and format its text
Header Header = new Header();
Header.Text = "Swimlane";
Header.Height = 50;
Header.FillColor = "#C7D4DF";
Header.FontColor = "black";
Header.FontSize = 11;
Header.FontFamily = "Arial";
Header LaneHeader = new Header();
LaneHeader.Text = "Lane";
LaneHeader.Width = 50;
LaneHeader.FillColor = "#C7D4DF";
LaneHeader.FontColor = "black";
LaneHeader.FontSize = 11;
LaneHeader.FontFamily = "Arial";
//Defines an empty lanes Collection
Collection Lanes = new Collection();
Lane Lane = new Lane();
Lane.Name = "lane1";
Lane.FillColor = "#f5f5f5";
Lane.Height = 120;
//Sets the object as Lane
Lane.IsLane = true;
//Sets the object as Lane
Lane.Header = LaneHeader;
//Specifies the minimum and maximum size of the lane
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Lane.MinWidth = 500;
Lane.MaxWidth = 700;
Lane.MinHeight = 120;
Lane.MaxHeight = 200;
Lanes.Add(Lane);
SwimLane Swimlane = new SwimLane();
Swimlane.Name = "swimlaneNode";
//Changes the orientation
Swimlane.Orientation = "horizontal";
//Sets the position and size
Swimlane.OffsetX = 400;
Swimlane.OffsetY = 200;
Swimlane.Height = 100;
Swimlane.Width = 700;
Swimlane.Header = Header;
//Sets the type of object as swimlane
Swimlane.Type = "swimlane";
Swimlane.Lanes = Lanes;
//Add the swimlane in to the Nodes collection
Model.Nodes.Add(Swimlane);
```

Disable/Update header

You can disable/update the lane header at runtime with the client side method, updateNode. The following code example illustrates how to disable the lane header at run time.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Sets "0" to hide header
diagram.updateNode("laneName", { header: {height: 0} })
```

Add nodes to a lane

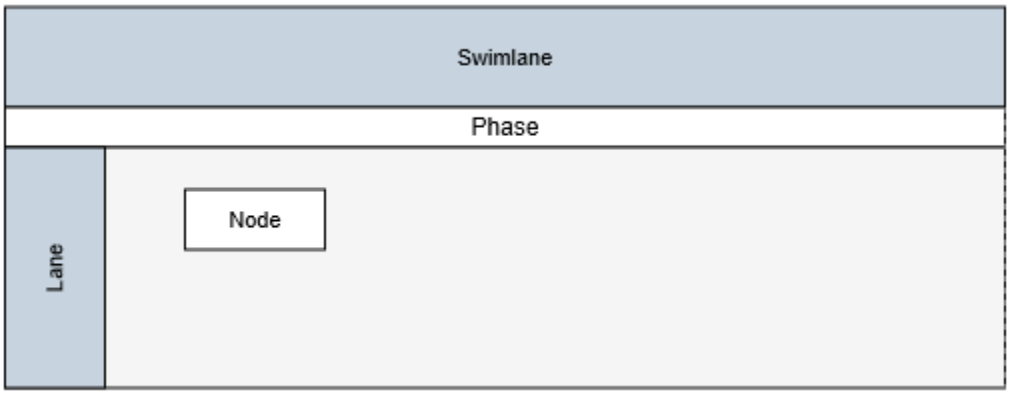
To add nodes to a lane, You need to add them to the Children collection of lane. The following code example illustrates how to add nodes to a lane.

C#

```
Label Label = new Label() { Text = "Node" };
Node Child = new Node()
{
Name = "node",
Width = 70,
Height = 30,
MarginLeft = 70,
MarginTop = 1
};
//Defines an empty lanes Collection
Collection Lanes = new Collection();
//Defines an empty Children Collection
Collection Children = new Collection();
Lane Lane = new Lane();
Lane.Name = "lane1";
Lane.FillColor = "#f5f5f5";
Lane.Height = 120;
//Sets the object as Lane
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Lane.IsLane = true;
//Sets the object as Lane
Lane.Header = LaneHeader;
//Add the Child to the children of the lane
Children.Add(Child);
Lane.Children = Children;
Lanes.Add(Lane);
SwimLane Swimlane = new SwimLane();
Swimlane.Name = "swimlaneNode";
//Changes the orientation
Swimlane.Orientation = "horizontal";
//Sets the position and size
Swimlane.OffsetX = 400;
Swimlane.OffsetY = 200;
Swimlane.Height = 100;
Swimlane.Width = 700;
Swimlane.Header = Header;
Swimlane.Lanes = Lanes;
//Add the swimlane in to the Nodes collection
Model.Nodes.Add(Swimlane);
```



Phase

Phases are the sub-processes that are used to break the swimlane into multiple smaller regions.

Add phase

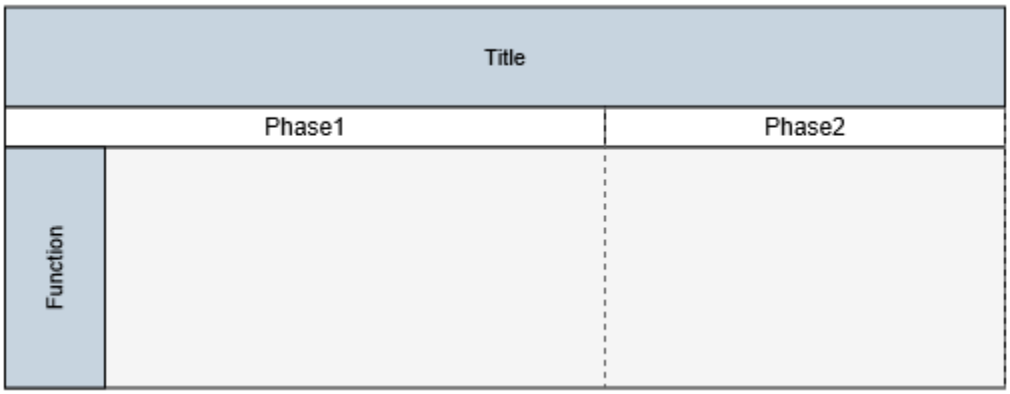
To define a phase, you have to set the length of the region to the Offset property of phase. Every region can be textually described with the Label property of phase.

The following code example illustrates how to add a phase on initializing swimlane.

C#

```
Label PhaseLabel = new Label() { Text = "Phase1" };
//Defines an empty Children Collection
Collection Phases = new Collection();
//Creates a phase
Phase Phase1 = new Phase()
{
Name = "Phase1",
//Length of the first region
Offset = 300,
//Initializes labels for phases
```

```
Label = PhaseLabel,
//Specifies the appearance of separator
LineWidth = 1,
LineDashArray = "3,3",
LineColor = "#606060"
};
PhaseLabel = new Label() { Text = "Phase2" };
Phase Phase2 = new Phase()
{
Name = "Phase2",
Label = PhaseLabel,
};
Phases.Add(Phase1);
Phases.Add(Phase2);
SwimLane Swimlane = new SwimLane();
Swimlane.Name = "swimlaneNode";
//Changes the orientation
Swimlane.Orientation = "horizontal";
//Sets the position and size
Swimlane.OffsetX = 400;
Swimlane.OffsetY = 200;
Swimlane.Height = 100;
Swimlane.Width = 700;
Swimlane.Header = Header;
//Sets the type of object as swimlane
Swimlane.Type = "swimlane";
//Sets the Lanes Collection
Swimlane.Lanes = Lanes;
//Sets the Phase Collection
Swimlane.Phases = Phases;
//Add the swimlane in to the Nodes collection
Model.Nodes.Add(Swimlane);
```



Add phase at runtime

You can add a region at runtime with the client side method, addPhase. The following code example illustrates how to add a phase at runtime.

JS

```
var phase = {
name: "Phase3",
label: { text: "Phase3" }
```

```
};
var diagram = $("#diagram").ejDiagram("instance");
diagram.addPhase("swimlaneName", phase);
```

A phase can be updated at runtime with the client side API updateNode. The following code example illustrates how to a update phase at runtime.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var options = {
//Specifies the style of the phase to be updated
lineDashArray: "3,3",
lineColor: "#C7D4DF",
lineWidth: 2
}
diagram.updateNode("phaseName", options);
```

Note: A default phase is added, when the phase collection of the swimlane is empty. When the phase collection is initialized, a default phase is appended at the end of swimlane.

Limitations

- You cannot add connectors as the children of lanes.

Gridlines

Gridlines are the pattern of lines drawn behind the Diagram elements. It provides a visual guidance while dragging or arranging the objects on the Diagram surface.

Customize the gridlines visibility

The SnapSettings.SnapConstraints enables you to show/hide the gridlines. The following code example illustrates how to show or hide gridlines.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.SnapSettings = new SnapSettings()
{
//Shows both horizontal and vertical gridlines
SnapConstraints = SnapConstraints.ShowLines
}
ViewData["diagramModel"] = model;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

To show only horizontal/vertical gridlines or to hide gridlines, refer to Constraints

Appearance

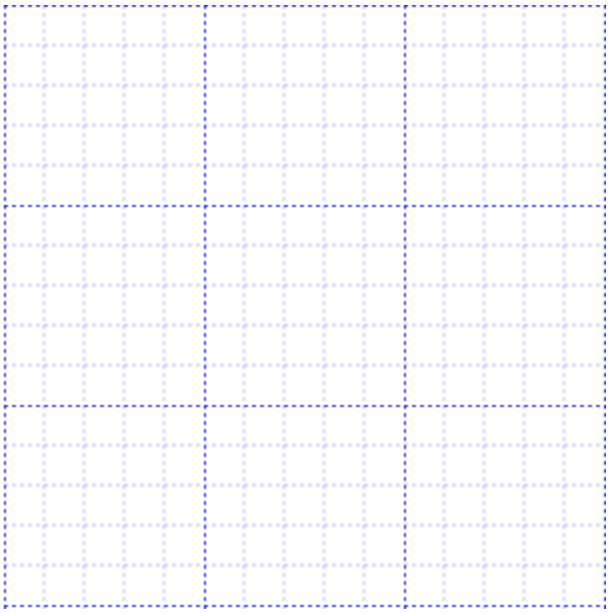
You can customize the appearance of the gridlines by using a set of predefined properties. To explore those properties, refer to Gridlines

The HorizontalGridLines and VerticalGridLines properties allow to customize the appearance of the gridlines. The following code example illustrates how to customize the appearance of gridlines.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.SnapSettings = new SnapSettings()
{
//Shows both horizontal and vertical gridlines
SnapConstraints = SnapConstraints.ShowLines,
// Customizes the line color and line style to the gridlines.
HorizontalGridlines = new GridLines() { LineColor = "blue", LineDashArray =
"2 2" },
VerticalGridlines = new GridLines() { LineColor = "blue", LineDashArray = "2
2" }
};
ViewData["diagramModel"] = model;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



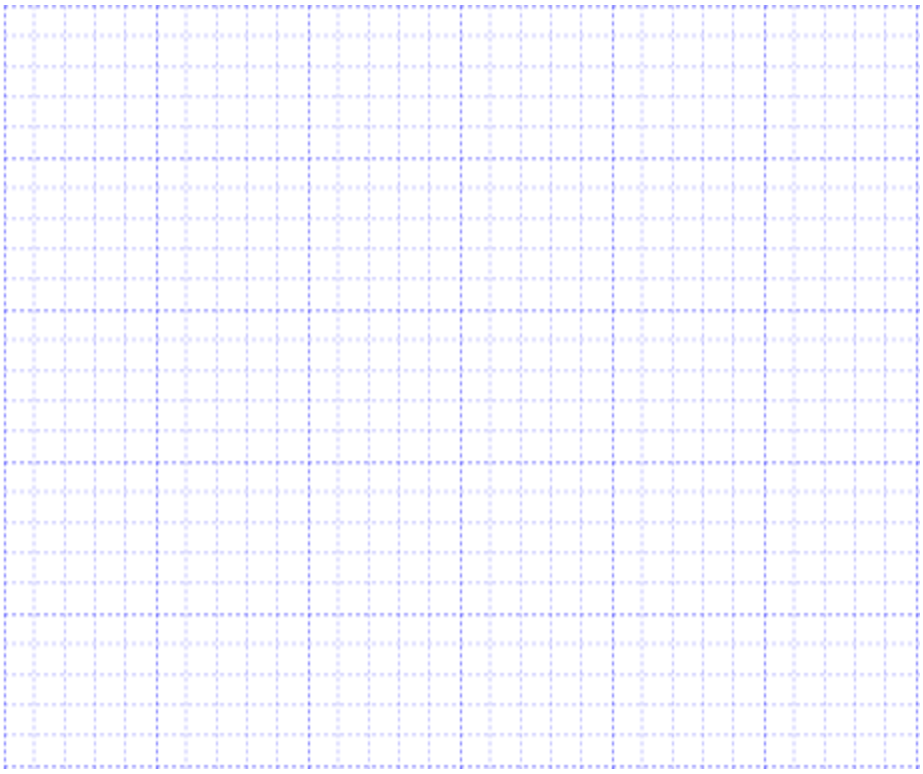
Line Intervals

Thickness and the space between gridlines can be customized by using LinesInterval property. In the linesInterval collections, values at the odd places are referred as the thickness of lines and the values at the even places are referred as the space between gridlines.

The following code example illustrates how to customize the thickness of lines and the line intervals.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.SnapSettings = new SnapSettings()
{
//Shows both horizontal and vertical gridlines
SnapConstraints = SnapConstraints.ShowLines,
// Customizes the line color and line style to the gridlines.
// Defines the thickness and intervals for a pattern of lines
HorizontalGridlines = new GridLines() {
LinesInterval = new List<decimal>(){1.25m, 14, 0.25m, 15, 0.25m, 15, 0.25m,
15, 0.25m, 15},
LineColor = "blue",
LineDashArray = "2 2"
},
VerticalGridlines = new GridLines() {
LinesInterval = new List<decimal>(){1.25m, 14, 0.25m, 15, 0.25m, 15, 0.25m,
15, 0.25m, 15},
LineColor = "blue",
LineDashArray = "2 2"
}
};
ViewData["diagramModel"] = model;
return View();
}
```



Snapping

Snap To Lines

This feature allows the Diagram objects to snap to the nearest intersection of gridlines while being dragged or resized. This feature enables easier alignment during layout or design.

Snapping to gridlines can be enabled/disabled with the SnapSettings.SnapConstraints. The following code example illustrates how to enable/disable the snapping to gridlines.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.SnapSettings = new SnapSettings()
{
//Enables snapping to both the horizontal and vertical lines.
SnapConstraints = SnapConstraints.SnapToLines,
};
ViewData["diagramModel"] = model;
return View();
}
```

To enable/disable snapping to horizontal/vertical lines, refer to Constraints

Customization of Snap Intervals

By default, the objects are snapped towards the nearest gridline. The gridline or position towards where the diagram object snaps can be customized with the property, snapInterval. The following code example illustrates how to customize the snap intervals.

RAZOR

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.SnapSettings = new SnapSettings()
{
HorizontalGridlines = new GridLines() { LinesInterval = new
List<decimal>(){10} },
VerticalGridlines = new GridLines() { LinesInterval = new
List<decimal>(){10} },
SnapConstraints = SnapConstraints.All
};
ViewData["diagramModel"] = model;
return View();
}
```

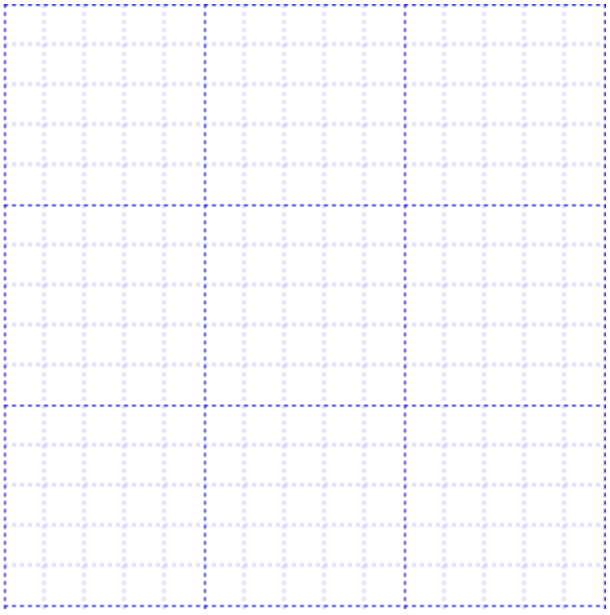
Snap To Objects

The snap-to-object provides visual cues to assist with aligning and spacing Diagram elements. A node can be snapped with its neighboring objects based on certain alignments. Such alignments are visually represented as smart guides.

The EnableSnapToObject property allows you to enable/disable smart guides. The following code example illustrates how to enable/disable the smart guides.

RAZOR

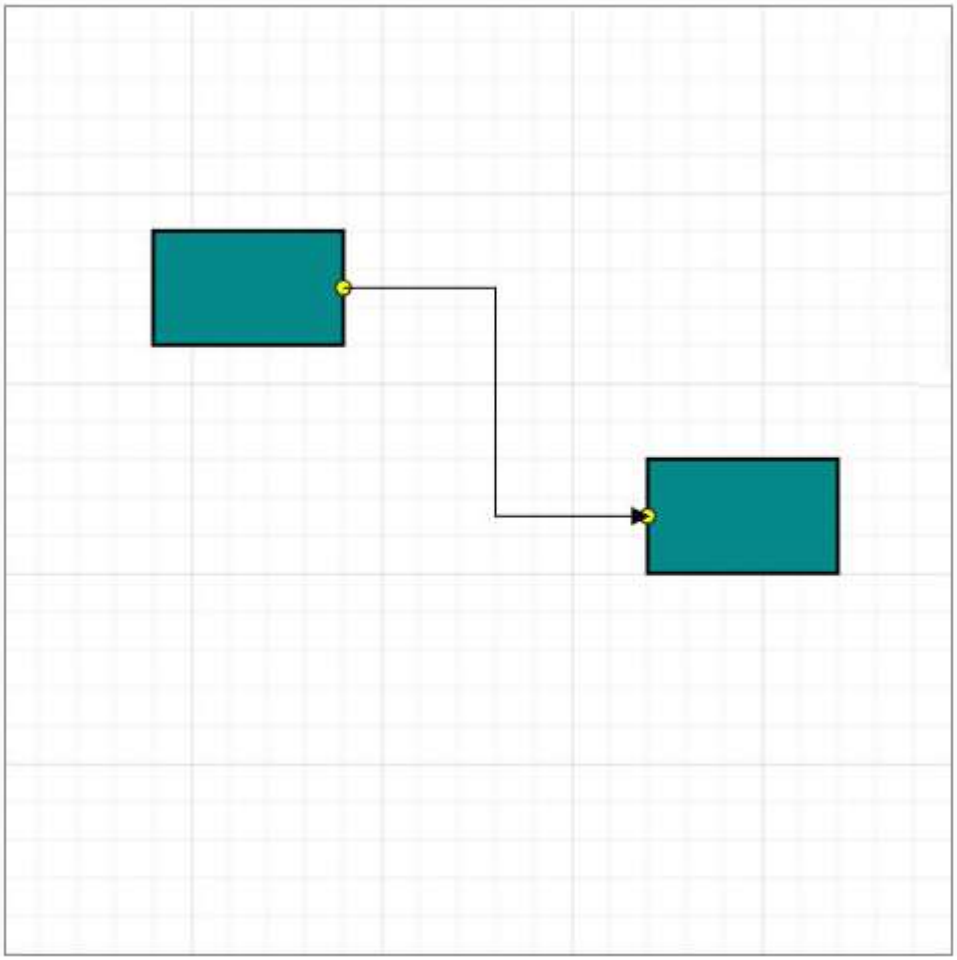
```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.EnableSnapToObject = true;
ViewData["diagramModel"] = model;
return View();
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

Page Settings

Page settings enable to customize the appearance, width, and height of the Diagram page.



Page size and appearance

The size and appearance of the Diagram pages can be customized with the PageSettings property.

The PageWidth and PageHeight properties of page settings define the size of the page. In addition to that, you can customize the appearance of the page with a set of appearance specific properties.

To explore those properties, refer Page Settings.

You can also customize the appearance of off-page regions with the property BackgroundColor.

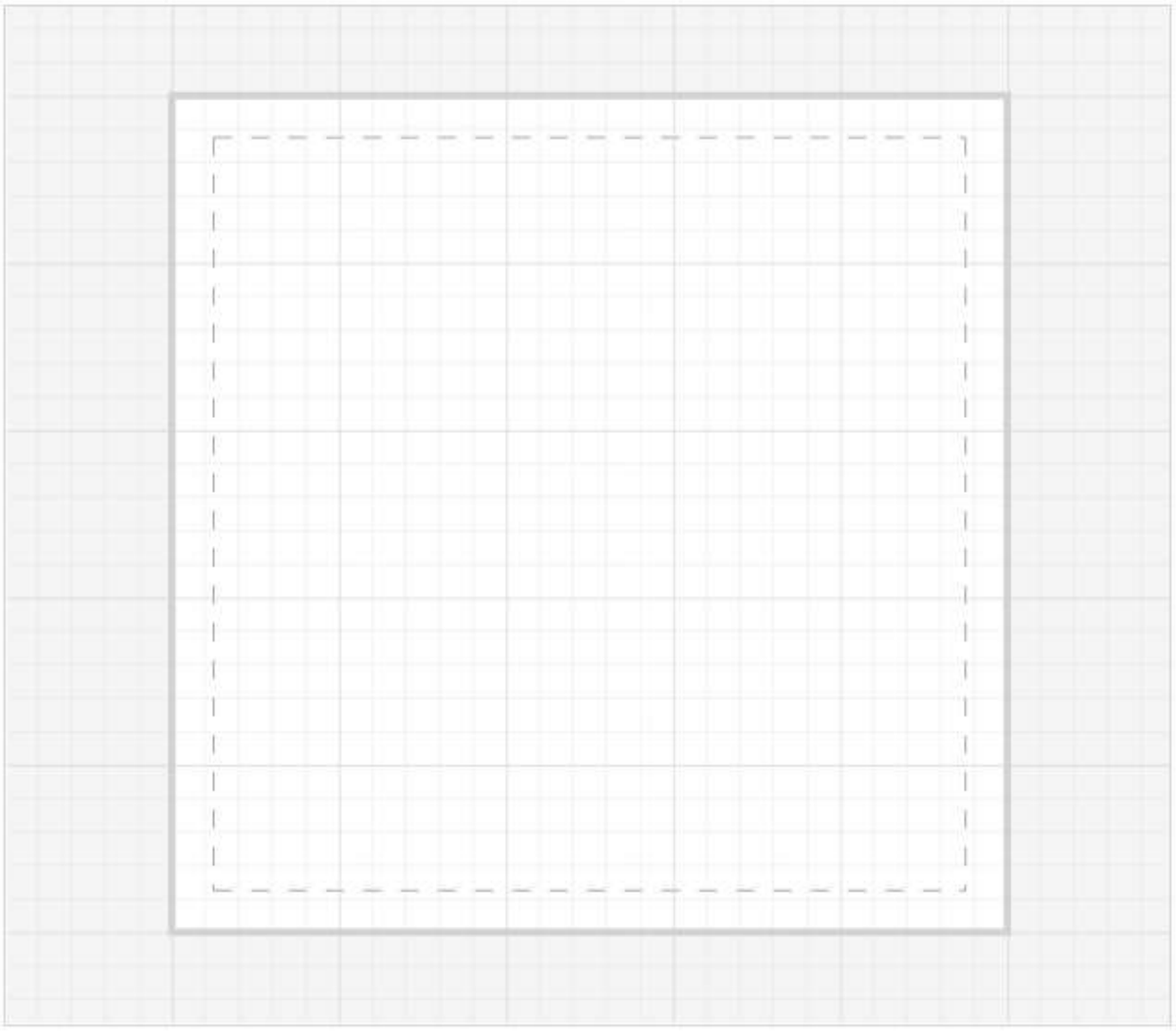
The following code illustrates how to customize the page size and the appearance of page and off-page.

C#

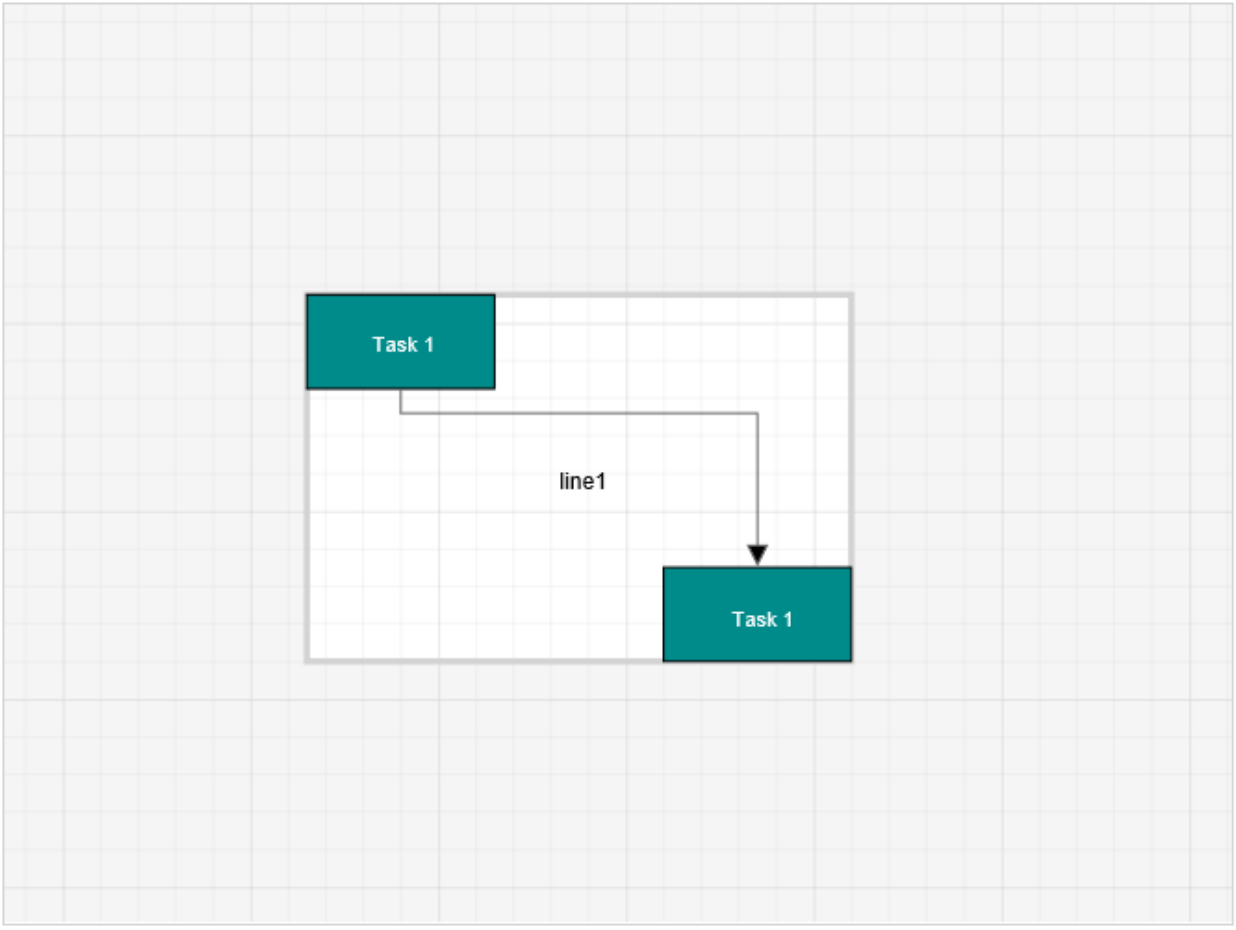
```
//Initializes Diagram
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "900px";
//Sets page size
Model.PageSettings.PageHeight = 500;
Model.PageSettings.PageWidth = 500;
//Customizes the appearance of page
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Model.PageSettings.PageBorderWidth = 4;
Model.PageSettings.PageBackgroundColor = "white";
Model.PageSettings.PageBorderColor = "lightgray";
Model.PageSettings.PageMargin = 25;
Model.PageSettings.ShowPageBreak = true;
Model.PageSettings.MultiplePage = true;
Model.PageSettings.PageOrientation = PageOrientation.Portrait;
```



Copyright © 2001 - 2020 Syncfusion Inc.

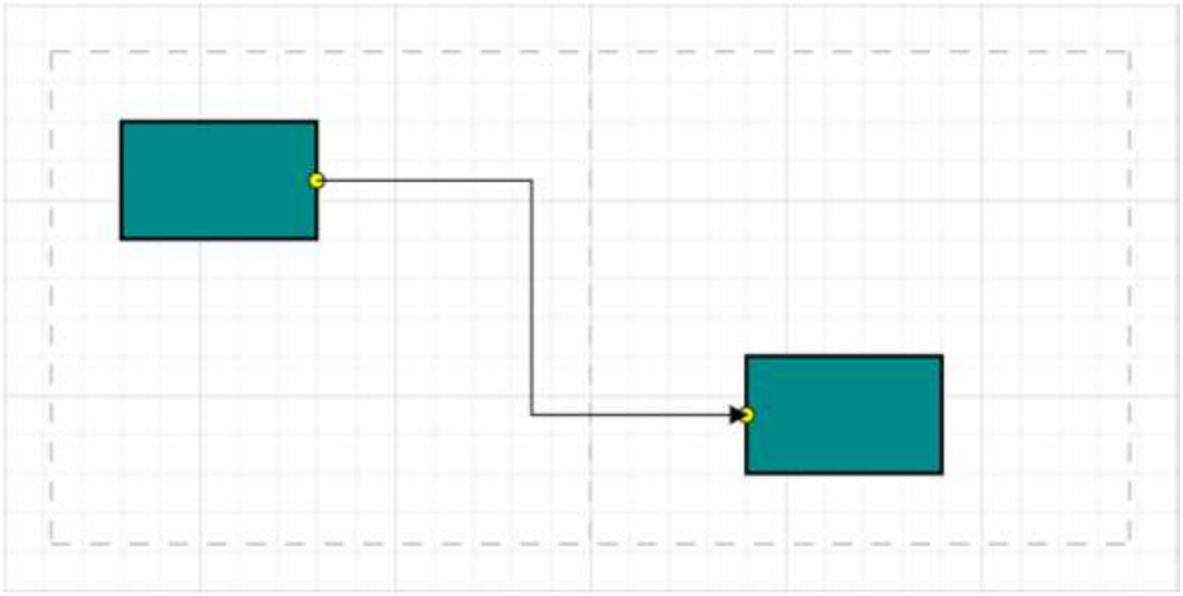


Note: When the PageWidth and PageHeight are not specified, the rectangular region that completely fits all nodes and connectors are considered as page size.

MultiplePage and PageBreaks

When MultiplePage is enabled, size of the page dynamically increases or decreases in multiples of page width and height and completely fits diagram within the page boundaries. Page Breaks is used as a visual guide to see how pages are split into multiple pages.

Copyright © 2001 - 2020 Syncfusion Inc.



MultiplePage and ShowPageBreak properties of page settings allow you to enable/disable multiple pages and page breaks respectively.

The following code illustrates how to enable multiple page and page break lines.

C#

```
//Initializes Diagram
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "900px";
//Enables the multiple page
Model.PageSettings.MultiplePage = true;
//Enables the page break lines
Model.PageSettings.ShowPageBreak = true;
```

Boundary Constraints

Diagram provides support to restrict/customize the interactive region, out of which the elements cannot be dragged, resized or rotated.

BoundaryConstraints property of page settings allows you to customize the interactive region. To explore the boundary constraints, refer refer Boundary Constraints.

The following code example illustrates how to define boundaryConstraints.

C#

```
//Initializes Diagram
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "900px";
//Allows to drag within the diagram content
Model.PageSettings.BoundaryConstraints = BoundaryConstraints.Diagram;
```

Copyright © 2001 - 2020 Syncfusion Inc.

Rulers

The Ruler provides a Horizontal and Vertical guide for measuring in the Diagram control. The Ruler can be used to measure the Diagram objects, indicate positions, and align Diagram elements. This is especially useful in creating scale models.

Adding Rulers to the Diagram

Rulers can be enabled by setting the ruler Setting's ShowRulers property for the diagram control.

Use the following code example to enable/disable the ruler to the Diagram.

C#

```
DiagramBuilderModel diagramModel = new DiagramBuilderModel();
diagramModel.Model.RulerSettings = new RulerSettings();
diagramModel.Model.RulerSettings.ShowRulers = true;
```


Customizing the Ruler

The rulerSetting's Interval property is used to define the number of tick's between two major stroke lines of the ruler.

The ruler Setting's SegmentWidth is used to defines the space between the two major tick strokes.

The ArrangeTick event will be triggered when each tick have been drawn.By using this event you can customize the length of the tick.

Use the following code example to customize the ruler's segment.

C#

```
DiagramBuilderModel diagramModel = new DiagramBuilderModel();
diagramModel.Model.RulerSettings = new RulerSettings();
diagramModel.Model.RulerSettings.ShowRulers = true;
//Customizing the horizontal ruler.
DiagramRuler HorizontalRuler = new DiagramRuler();
//Creating a custom segment with 6 intervals.
HorizontalRuler.Interval = 6;
// Customizing the ruler segment width.
HorizontalRuler.SegmentWidth = 100;
// Customizing the Ruler ticks.
HorizontalRuler.ArrangeTick = "arrangeTick";
DiagramRuler VerticalRuler = new DiagramRuler();
VerticalRuler.Interval = 6;
VerticalRuler.SegmentWidth = 100;
VerticalRuler.ArrangeTick = "arrangeTick";
diagramModel.Model.RulerSettings.HorizontalRuler = HorizontalRuler;
diagramModel.Model.RulerSettings.VerticalRuler = VerticalRuler;
```

JS

```
function arrangeTick(args) {
// Customizing the Ruler ticks.
if (args.tickInterval % 100 == 0) {
}
else if (args.tickInterval % 50 == 0) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
args.tickLength = 12.5
}
}
```

The width of the ruler can be set through Thickness property. If it's an horizontal ruler, thickness will be taken as height of the ruler.

Tick alignment can be modified using TickAlignment as either LeftOrTop or RightOrBottom.

The MarkerColor property of the ruler is used to indicate the position of the mouse cursor.

Use the following code example to customize the ruler's tick.

C#

```
DiagramBuilderModel diagramModel = new DiagramBuilderModel();
diagramModel.Model.RulerSettings = new RulerSettings();
diagramModel.Model.RulerSettings.ShowRulers = true;
//Customizing the horizontal ruler.
DiagramRuler HorizontalRuler = new DiagramRuler();
// Customizing the Ruler ticks alignment.
HorizontalRuler.TickAlignment = TickAlignment.LeftOrTop;
// Customizing the Ruler marker color
HorizontalRuler.MarkerColor = "blue";
// Customizing the thickness of the ruler bar.
HorizontalRuler.Thickness = 25;
DiagramRuler VerticalRuler = new DiagramRuler();
VerticalRuler.TickAlignment = TickAlignment.LeftOrTop;
VerticalRuler.MarkerColor = "blue";
VerticalRuler.Thickness = 25;
diagramModel.Model.RulerSettings.HorizontalRuler = HorizontalRuler;
diagramModel.Model.RulerSettings.VerticalRuler = VerticalRuler;
```


Interaction

Selection

Selector provides a visual representation of selected elements. It behaves like a container and enables you to update the size, position, and rotation angle of the selected elements through interaction and programmatically. Single or multiple elements can be selected at a time.

Single selection

An element can be selected by clicking that element. During single click, all previously selected items are cleared. The following image shows how the selected elements are visually represented.

Selecting a group

When a child element of any group is clicked, its contained group is selected instead of the child element. With consecutive clicks on the selected element, selection is changed from top to bottom in the hierarchy of parent group to its children.

Multiple selection

Multiple elements can be selected with the following ways.

Copyright © 2001 - 2020 Syncfusion Inc.

1. Ctrl+Click

During single click, any existing item in the selection list be cleared, and only the item clicked recently is there in the selection list. To avoid cleaning the old selected item, Ctrl key must be on hold when clicking.

2. Selection rectangle / Rubber band selection

Clicking and dragging the Diagram area allows to create a rectangular region. The elements that are covered under the rectangular region are selected at the end.

Multiple selected elements are visually represented as shown.

Select/Unselect elements programmatically

The client side methods addSelection, clearSelection and removeSelection help select/unselect elements at runtime. The following code example illustrates how to select/unselect an item through programmatically.

RAZOR

```
var diagram = $("#DiagramContent").ejDiagram("instance");
var node = diagram.findNode("node1");
//Selects an element
diagram.addSelection(node);
//Clears the current selection
diagram.clearSelection();
//Unselects a particular object
diagram.removeSelection(node);
```

You can get the current selected items from the children collection of selectedItems property of the Diagram model.

The client side method updateSelector helps you to change the size, position, and rotation angle of the selected elements at runtime. The following code example illustrates how to read the selected items and how to update the position, size, and rotation angle.

RAZOR

```
var diagram = $("#DiagramContent").ejDiagram("instance");
//Updates the size, position and rotateAngle of selected elements.
diagram.updateSelector({
offsetX: 200,
offsetY: 200,
width: 200,
height: 200,
rotateAngle: 200
});
//Iterates the selected elements
for (var i = 0; i < diagram.model.selectedItems.children.length; i++) {
var child = diagram.model.selectedItems.children[i];
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: SelectedItems's children is a read-only property. You cannot change the children collection at run time.

Drag

- An object can be dragged by clicking and dragging it. When multiple elements are selected, dragging any one of the selected elements move every selected element.
- While dragging, the objects are snapped towards the nearest objects to make better alignments. For better alignments, refer to Snapping.

Resize

- Selector is surrounded by eight thumbs. When dragging these thumbs, selected items can be resized smaller or larger.
- When one corner of the selector is dragged, opposite corner is in a static position.
- While resizing, the objects are snapped towards the nearest objects to make better alignments. For better alignments, refer to Snapping.

Rotate

- A rotate handler is placed above the selector. Clicking and dragging the handler in a circular direction lead to rotate the node.
- The node is rotated with reference to the static pivot point.
- Pivot thumb (thumb at the middle of the node) appears while rotating the node to represent the static point.
- For more information about pivot, refer to Position.

Connection editing

- Each segment of a selected connector is editable with some specific handles/thumbs.

End point handles

Source and target points of the selected connectors are represented with two handles. Clicking and dragging those handles help you to adjust the source and target points.

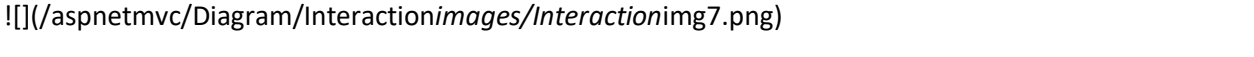
Straight segment editing

- End point of each straight segment is represented by a thumb that enables to edit the segment.
- Any number of new segments can be inserted into a straight line by clicking that when shift and ctrl keys are pressed. (Ctrl+Shift+Click).
- Straight segments can be removed by clicking the segment end point, when ctrl and shift keys are pressed. (Ctrl+Shift+Click).

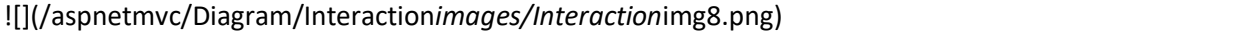
Copyright © 2001 - 2020 Syncfusion Inc.

Orthogonal thumbs

- Orthogonal thumbs allow to adjust the length of adjacent segments by clicking and dragging it.



- When necessary, some segments are added or removed automatically, when dragging the segment. This is to maintain proper routing of orthogonality between segments.



Bezier thumbs

- Bezier segments are annotated with two thumbs to represent the control points. Control points of the curve can be configured by clicking and dragging the control thumbs.



Drag and drop nodes over other elements

Diagram provides support to drop a node/connector over another node/connector. Drop event is raised to notify that an element is dropped over another one and it is disabled by default. It can enabled with the Constraints property. The following code illustrates how to enable **dropping**.

C#

```
DiagramProperties model = new DiagramProperties();
model.Nodes.Add(new BasicShape()
{
Name = "node",
Width = 100,
Height = 100,
OffsetX = 100,
OffsetY = 100,
FillColor = "#1BA0E2",
//Allows to drop an object over the node
Constraints = NodeConstraints.Default | NodeConstraints.AllowDrop
});
model.Connectors.Add(new Connector()
{
Name = "connector1",
SourcePoint = new DiagramPoint(200, 50),
TargetPoint = new DiagramPoint(300, 150),
Segments = new Collection() { new Segment(Segments.Orthogonal) },
//Allows to drop an object over the connector
Constraints = ConnectorConstraints.Default | ConnectorConstraints.AllowDrop
});
ViewData["diagramModel"] = model;
```

The following code examples illustrates how to insert a child to a group interactively.

C#

```
DiagramProperties model = new DiagramProperties();
//Sets the drop event
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
model.Drop = "drop";
```

RAZOR

```
function drop(args) {
if (args.target && args.element) {
//Element that is being dropped
var node = args.element;
//Element over which another element is dropped
var target = args.target;
if (diagram.getObjectType(target) == "group" && diagram.getObjectType(node)
== "node") {
//Inserts the node into group
node.parent = target.name;
target.children.push(node.name);
}
}
}
```

User handles

User handles are used to add some frequently used commands around the selector. To create user handles, define and add them to the userHandles collection of selectedItems property.

The tool property of user handle should be set with any appropriate tool to enable interaction.

The following code example illustrates how to draw a user handle.

RAZOR

```
// Defines a tool to interact with the user handle
var CloneTool = (function (base) {
ej.datavisualization.Diagram.extend(CloneTool, base);
// Initializes the tool
function CloneTool(name) {
base.call(this, name);
}
// Defines the required events
CloneTool.prototype.mouseup = function (event) {
//clones the selected elements when the custom handle is clicked.
this.diagram.copy();
//Pastes the cloned elements.
this.diagram.paste();
base.prototype.mouseup.call(this, event);
};
return CloneTool;
})(ej.datavisualization.Diagram.ToolBase);
var userHandles = [];
//Initializes the handles
var cloneHandle = ej.datavisualization.Diagram.UserHandle();
// Name of the user handle.
cloneHandle.name = "Clone";
// Sets the tool that you have defined to interact with handle.
cloneHandle.tool = new CloneTool(cloneHandle.name);
userHandles.push(cloneHandle);
//Add user handles to diagram
$("#DiagramContent").ejDiagram({
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Sets user handles
selectedItems: {
userHandles: userHandles
}
});
```


Appearance

Position, size, and style of the user handle can be customized with a set of predefined properties.

The following code example illustrates how to customize the appearance of the user handle.

RAZOR

```
var UserHandlePositions = ej.datavisualization.Diagram.UserHandlePositions;
var userHandles = [];
var cloneHandle = ej.datavisualization.Diagram.UserHandle();
cloneHandle.name = "Clone";
// Sets the position of the user handle.
cloneHandle.position = UserHandlePositions.BottomLeft;
// Sets the user handle visibility.
cloneHandle.visible = true;
// Defines whether user handle should be enabled for multiple selection or
not.
cloneHandle.enableMultiSelection = true;
// Sets the size of the user handle.
cloneHandle.size = 30;
// Customizes the appearance of user handle
cloneHandle.backgroundColor = "#4D4D4D";
cloneHandle.pathColor = "white";
cloneHandle.borderColor = "red";
cloneHandle.borderWidth = "2";
cloneHandle.pathData = "M4.6350084,4.8909971 L4.6350084,9.3649971
9.5480137,9.3649971 9.5480137,4.8909971 z M3.0000062,2.8189973
L11.184016,2.8189973 11.184016,10.999997 3.0000062,10.999997 z M0,0
L7.3649998,0 7.3649998,1.4020001 1.4029988,1.4020001 1.4029988,8.0660002
0,8.0660002 0,1.4020001 0,0.70300276 z";
userHandles.push(cloneHandle);
$("#DiagramContent").ejDiagram({
selectedItems: {
userHandles: userHandles
}
});
```


Zoom pan

- When a large Diagram is loaded, only certain portion of the Diagram is visible. The remaining portions are clipped. Clipped portions can be explored by scrolling the scrollbars or panning the Diagram.
- Diagram can be zoomed in or out by using Ctrl + mouse wheel.

Copyright © 2001 - 2020 Syncfusion Inc.

Keyboard

Diagram provides support to interact with the elements with key gestures. By default, some built-in commands are bound with a relevant set of key combinations.

The following table illustrates those commands with the associated key values.

| Shortcut Key | Command | Description |

|---|---|---|

| Ctrl + A | selectAll | Select all nodes/connectors in diagram |

| Ctrl + C | copy | Copy the diagram selected elements |

| Ctrl + V | paste | Paste the copied elements |

| Ctrl + X | cut | Cut the selected elements |

| Ctrl + Z | undo | Undo(Reverse the last editing action performed on diagram) |

| Ctrl + Y | redo | Redo(Restores the last editing action when no other actions have occurred since the last undo on diagram) |

| Delete | delete | Delete the selected elements |

| Ctrl /Shift+ Click on object | | Multiple selection(Selector binds all selected nodes/connectors) |

| Up Arrow | nudge("up") | nudgeUp(move the selected elements towards up by one pixel) |

| Down Arrow | nudge("down") | nudgeDown(move the selected elements towards down by one pixel) |

| Left Arrow | nudge("left") | nudgeLeft(move the selected elements towards left by one pixel) |

| Right Arrow | nudge("right") | nudgeRight(move the selected elements towards right by one pixel) |

| Ctrl+MouseScroll | zoom | Zoom(Zoom in/Zoom out the diagram) |

| F2 | startLabelEditing | Starts to edit the label of selected element |

| Esc | endLabelEditing | Sets the label mode as View and stops editing. |

To add custom commands, configure or modify key/mouse gesture through Command Manager.

History Manager

Diagram tracks the history of actions that are performed after loading the time and provides support to reverse and restore those changes.

Undo and Redo

Diagram provides built-in support to track the changes that are made through interaction and through public APIs. The changes can be reverted or restored either through short cut keys or through commands.

Undo redo through short cut keys

Undo redo commands can be executed through short cut keys. Short cut key for undo is ctrl+z and short cut key for redo is ctrl+y.

For more information, refer to Keyboard Interactions.

Copyright © 2001 - 2020 Syncfusion Inc.

Undo redo through public APIs

The client side methods undo and redo help you to revert/restore the changes. The following code example illustrates how to undo/redo the changes through script.

JS

```
var diagram = $("#Diagram").ejDiagram("instance");
// Reverts the last action performed
diagram.undo();
// Restores the last undone action
diagram.redo();
```

Track custom changes

Diagram provides options to track the changes that are made to custom properties. For example, in case of an employee relationship Diagram, you may need to track the changes in the employee information. The HistoryManager of the Diagram model enables you to track such changes.

The following example illustrates how to track such custom property changes.

- Before changing the employee information, save the existing information to history manager by using the client side method push of historyManager.

The following code example illustrates how to save the existing property values.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Creates a custom entry and adds that to history manager
var entry = { object: node, prevState: node.employeeInfo };
diagram.model.historyManager.push(entry);
//Updates the new information
var newValue = { role: "New role" };
node.employeeInfo = newValue;
```

- Change the employee information

JS

```
//Updates the new information
var newValue = { role: "New role" };
node.employeeInfo = newValue;
```

- Define the methods to handle the custom changes. These methods are called when you try to revert/restore the custom changes.

You need to define the methods to handle the custom changes and you need to assign that to Undo and Redo properties of HistoryManager.

The following code example illustrates how to define methods to handle the custom changes.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
DiagramProperties model = new DiagramProperties();
//Called to revert a custom action
model.HistoryManager.Undo = "customUndoRedo";
//Called to restore the reverted custom action
model.HistoryManager.Redo = "customUndoRedo";
ViewData["DiagramModel"] = diagram;
```

JS

```
//Method to handle the custom action
function customUndoRedo(args) {
var diagram = $("#diagram").ejDiagram("instance");
var node = args.object;
var currentState = node.employeeInfo;
//Resets the state
node.employeeInfo = args.prevState;
//Saves the previous state
args.prevState = currentState;
}
```

Group multiple changes

History manager allows to revert or restore multiple changes through a single undo/redo command. For example, you may need to revert/restore the fill color change of multiple elements at a time.

The client side method startGroupAction is used to notify the Diagram to start grouping the changes. The client side method closeGroupAction is used to notify to stop grouping the changes. The following code illustrates how to undo/redo fillColor change of multiple elements at a time.

JS

```
var group = diagram.model.selectedItems
// Start to group the changes
diagram.model.historyManager.startGroupAction();
//Makes the changes
for (var i = 0; i < group.children.length; i++) {
var option = {};
var item = group.children[i];
// Updates the fillColor for all the child elements.
option.fillColor = args.style.backgroundColor;
diagram.updateNode(item.name, option);
}
//Ends grouping the changes
diagram.model.historyManager.closeGroupAction();
```

Serialization

Serialization is the process of saving and loading for state persistence of the Diagram.

Save

The Diagram is serialized as JSON data while saving. The client side method, save helps to serialize the Diagram as JSON. The following code illustrates how to save the Diagram.

JS

```
var diagram = $("#Diagram").ejDiagram("instance");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//save() returns serialized JSON data of the Diagram
var saveData = diagram.save();
```

This JSON data can be converted to string and stored for future use. The following snippet illustrates how to save the serialized JSON into local storage.

JS

```
//Saves the JSON object in to local storage
localStorage.setItem("diagram", JSON.stringify(json));
```

Diagram can also be saved as raster or vector image files. For more information about saving the Diagram as images, refer to Exporting.

Load

Diagram is loaded from the Serialized JSON data. The client side method, load helps you to load the Diagram from the serialized JSON. The following code illustrates how to load the Diagram from serialized JSON data.

JS

```
//Retrieves the JSON object from local storage
saveData = JSON.parse(localStorage.getItem("diagram"));
//Loads the Diagram from saved JSON data
diagram.load(json);
```

Note: Before loading a new Diagram, existing Diagram is cleared.

Limitation

There are some limitations in saving/loading the Diagrams and they are listed as follows.

- When define the events as a string, functions need to be maintained in the application level while loading the diagram.

C#

```
DiagramProperties model = new DiagramProperties();
//define the name of the function.
model.NodeCollectionChange = "nodeCollectionChange";
```

JS

```
function nodeCollectionChange(args) {
}
```

- HTML / Native template needs to be retained in the application level, while loading the native and HTML node.

HTML

```
<!-- Template content needs to be retained while loading the diagram.-->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script id="htmlTemplate" type="text/x-jsrender">
<div>
<input type="button" value="button" style="color: #ffffff; background-color:
#fbb139; border-color: #f89b1c" />
</div>
</script>
```

JS

```
diagram.load(saveData);
```

- CSS classes have to be retained in the application level, while loading the Diagram.

HTML

```
<style>
<!-- CSS class needs to be retained while loading the Diagram.-->
.nodeCss {
fill: black;
stroke: cyan;
}
</style>
```

JS

```
diagram.load(json);
```

Commands

There are several commands available in the Diagram as follows.

- Alignment commands
- Spacing commands
- Sizing commands
- Clipboard commands
- Grouping commands
- Z-order commands
- Zoom commands
- Nudge commands
- FitToPage commands
- Undo/Redo commands

Align

Alignment commands enable you to align the selected objects such as nodes and connectors with respect to the selection boundary.

| Command | Parameter | Description |

|---|---|---|

Copyright © 2001 - 2020 Syncfusion Inc.

| align | direction (string) | Align all the nodes/connectors in the selection list with respect to specified direction of the selection boundary. |

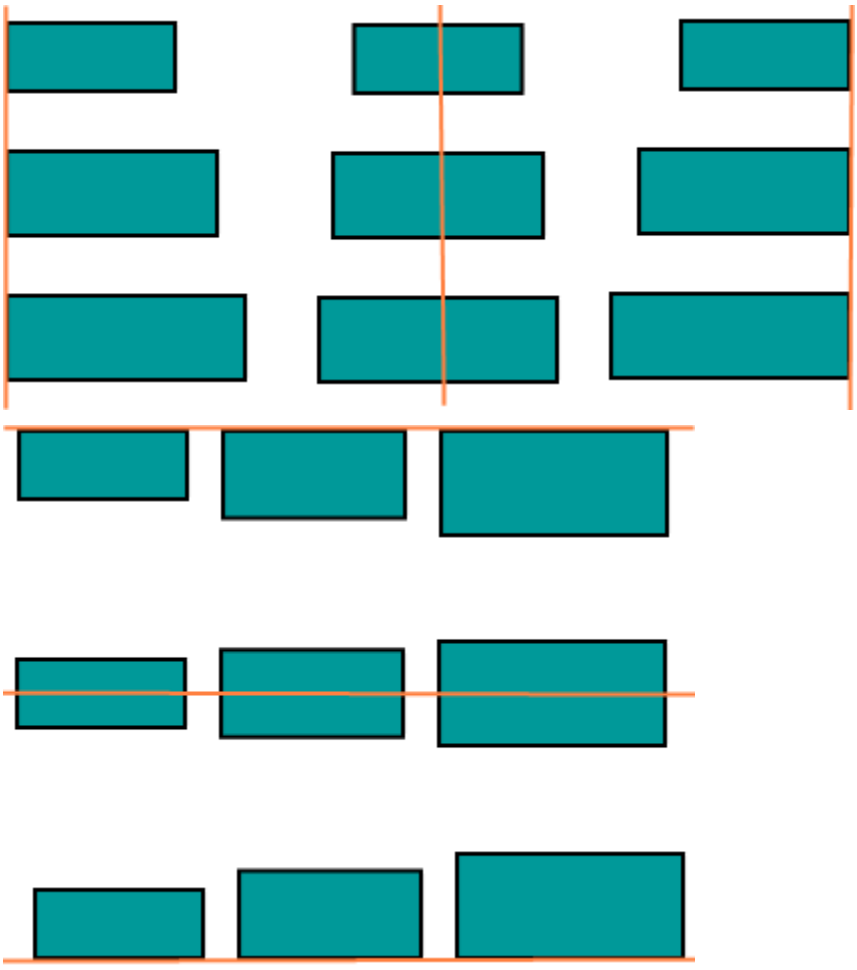
The accepted values of the argument "direction" are as follows.

- "left" - Align all the selected objects at the left of the selection boundary.
- "right" - Align all the selected objects at the right of the selection boundary.
- "center" - Align all the selected objects at the center of the selection boundary.
- "top" - Align all the selected objects at the top of the selection boundary.
- "bottom" - Align all the selected objects at the bottom of the selection boundary.
- "middle" - Align all the selected objects at the middle of the selection boundary.

The following code example illustrates how to align all the selected objects at the left side of selection boundary.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
//Sets direction as left
diagram.align("left");
```



Copyright © 2001 - 2020 Syncfusion Inc.

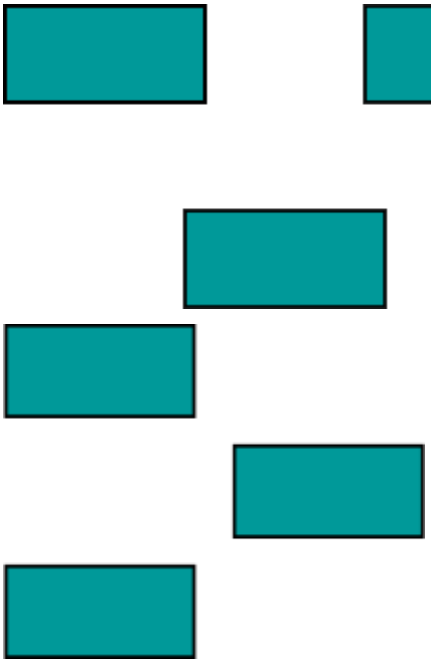
Space

Spacing commands enable you to place the selected objects on the page at equal intervals from each other. The selected objects are equally spaced within the selection boundary.

The following code example illustrates how to execute the space commands.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
//Equally spaces the selected nodes horizontally
diagram.spaceAcross();
//Equally spaces the selected nodes vertically
diagram.spaceDown();
```



Sizing

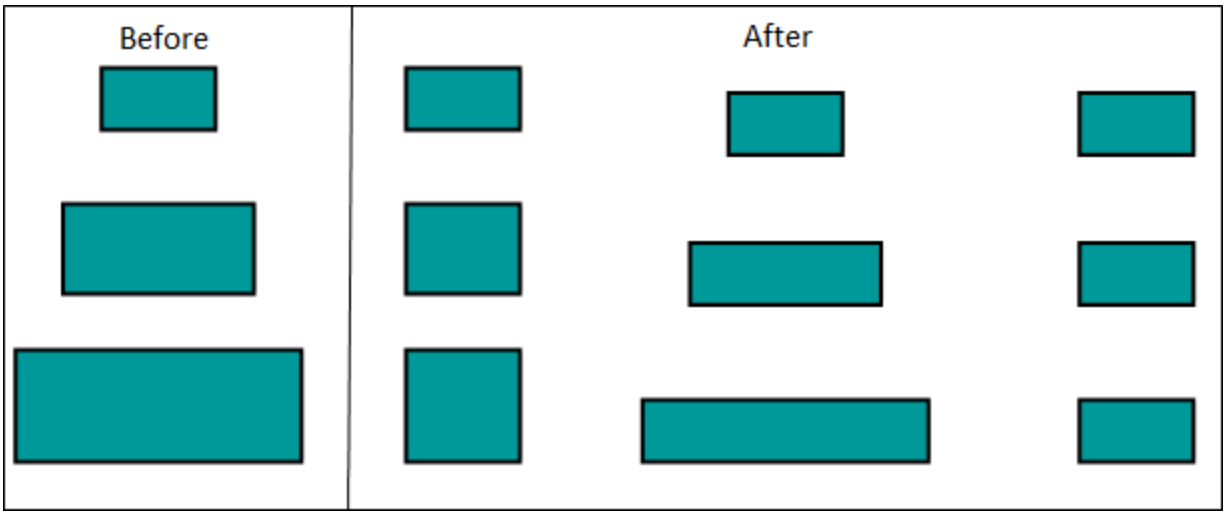
Sizing commands enable to equally size the selected nodes with respect to the first selected object.

The following code example illustrates how to execute the size commands.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
//Scales the selected items to the size of first selected object
diagram.sameSize();
//Vertically scales the selected items to the height of first selected
object
diagram.sameHeight();
//Horizontally scales the selected items to the width of first selected
object
diagram.sameWidth();
```

Copyright © 2001 - 2020 Syncfusion Inc.



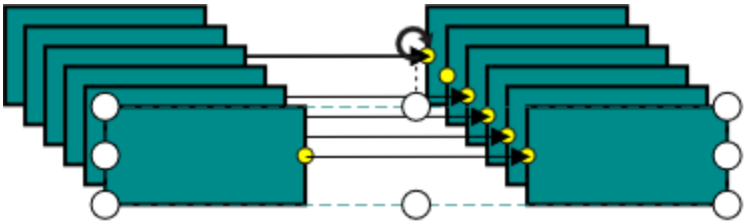
Clipboard

Clipboard commands are used to cut, copy or paste the selected elements.

The following code illustrates how to execute the clipboard commands.

JS

```
//Cuts the selected elements from the Diagram to the Diagram's clipboard
diagram.cut();
//Copies the selected elements from the Diagram to the Diagram's clipboard.
diagram.copy();
//Pastes the Diagram's clipboard data (nodes/connectors) into the Diagram.
diagram.paste();
```



Grouping

Grouping commands are used to group/ungroup the selected elements on the Diagram.

The following code illustrates how to execute the Grouping commands.

JS

```
//Groups the selected elements.
diagram.group();
//Ungroups the selected group.
diagram.ungroup();
```

Z-Order Command

Z-Order commands enable you to visually arrange the selected objects such as nodes and connectors on the page.

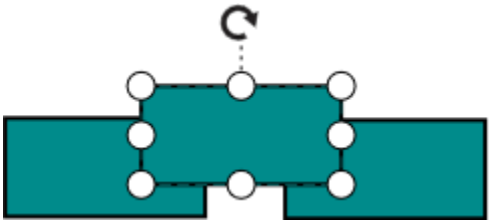
Copyright © 2001 - 2020 Syncfusion Inc.

bringToFront Command

The bringToFront command visually brings the selected element to front over all the other overlapped elements. The following code illustrates how to execute the bringToFront command.

JS

```
//Brings to front
diagram.bringToFront();
```

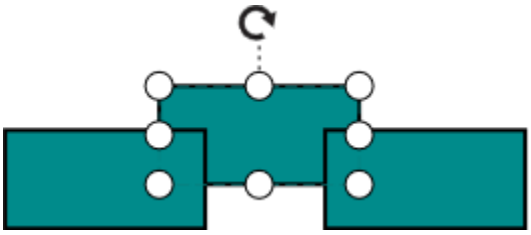


sendToBack Command

The sendToBack command visually moves the selected element behind all the other overlapped elements. The following code illustrates how to execute the sendToBack command.

JS

```
//Sends to back
diagram.sendToBack();
```

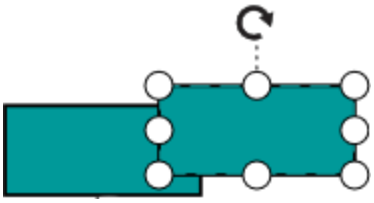


moveForward Command

The moveForward command visually moves the selected element over the nearest overlapping element. The following code illustrates how to execute the moveForward Command.

JS

```
//Moves forward
diagram.moveForward();
```



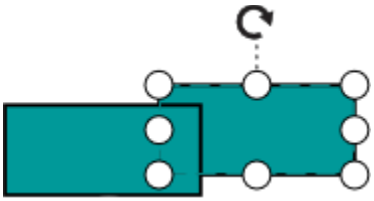
sendBackward Command

The sendBackward command visually moves the selected element behind the underlying element. The following code illustrates how to execute the sendBackward command.

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Sends backward
diagram.sendBackward();
```



Zoom

zoomTo command is used to zoom-in and zoom-out the Diagram view.

The following code illustrates how to zoom-in/zoom out the Diagram.

JS

```
function Zoom() {
var diagram = $("#diagram").ejDiagram("instance");
var zoom = {
// Sets the zoomFactor
zoomFactor: 0.2,
// Sets the zoomCommand as zoomIn to zoom-in the Diagram
zoomCommand: ej.datavisualization.Diagram.ZoomCommand.ZoomIn,
// for zoomOut
//zoomCommand: ej.datavisualization.Diagram.ZoomCommand.ZoomOut
//Defines the focusPoint to zoom the Diagram with respect to any point
//When you do not set focus point, zooming is performed with reference to
the center of current Diagram view.
focusPoint: {
x: 100,
y: 100
}
};
diagram.zoomTo(zoom);
}
```

Nudge Command

nudge commands move the selected elements towards up, down, left or right by 1 pixel.

| Command | Parameter | Description |

|---|---|---|

| nudge | direction (string),

delta(integer) | Nudge command moves the selected elements towards the specified direction by the number of pixels specified by the parameter delta. When delta is not specified, it is considered as 1 pixel, by default. |

The accepted values of the argument "direction" are as follows.

- "up" - moves the selected elements towards up by the the specified delta value.
- "down" - moves the selected elements towards down by the specified delta value.
- "left" - moves the selected elements towards left by the specified delta value.

Copyright © 2001 - 2020 Syncfusion Inc.

- "right" - moves the selected elements towards right by the specified delta value.

The following code illustrates how to execute Nudge command.

JS

```
var diagram = $("#diagramContent").ejDiagram("instance");
//Nudges up
diagram.nudge("up", 5);
```

Nudge by using Arrow Keys

The corresponding arrow keys are used to move the selected elements towards up, down, left, or right direction by 1 pixel.



Nudge commands are particularly useful for accurate placement of elements.

For more information, refer to Keyboard Interaction.

BringIntoView

bringIntoView command brings the specified rectangular region into the view port of the Diagram

The following code illustrates how to execute the bringIntoView command.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
//Brings the specified rectangular region of the Diagram content to the
viewport of the page.
diagram.bringIntoView({
x: 700,
y: 500,
width: 100,
height: 100
});
```

BringToCenter

bringToCenter command brings the specified rectangular region of the Diagram content to the center of the view port.

The following code illustrates how to execute the bringToCenter command.

JS

```
var diagram = $("#DiagramContent").ejDiagram("instance");
//Brings the specified rectangular region of the Diagram content to the
center of the viewport.
diagram.bringToCenter({
x: 700,
y: 500,
width: 100,
height: 100
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

FitToPage command

fitToPage command helps to fit the Diagram content into the view with respect to either width, height, or at the whole.

Command	Parameter	Description
fitToPage	mode (string) Value accepted: ej.datavisualization.Diagram.FitMode region (string) Value accepted- ej.datavisualization.Diagram.Region margin (object)	FitToPage command fits the Diagram into the view. The area/bounds to be fit into view is specified through the parameters.mode – FitToMode.region – Region.margin – Space that is to be left in between the content and viewport.

The following code illustrates how to execute FitToPage command.

JS

```
//Fits to page - fit Diagram based on elements
diagram.fitToPage("page", "content", {
left: 25,
top: 25,
right: 25,
bottom: 25
});
```

FitToMode

Mode specifies whether the Diagram content has to be fit into view with respect to width, height, or entire bounds of the Diagram. To explore the mode, refer to FitToMode.

Region

Region specifies the region/bounds of the Diagram content that is to be fit into the view. For more information about Region, refer to Region.

Command Manager

Diagram provides support to map/bind command execution with desired combination of key gestures. Diagram provides some built-in commands. For more information about built-in commands, refer to Keyboard Interaction.

Command Manager provides support to define custom commands. The custom commands are executed, when the specified key gesture is recognized.

Custom command

To define a custom command, you need to specify following properties.

- execute: a method to be execute.
- canExecute: a method to define whether the command can be executed at the moment.

Copyright © 2001 - 2020 Syncfusion Inc.

- gesture: the combination of key gestures.

To explore the properties of custom commands, refer to Commands

The following code example illustrates how to define a custom command.

RAZOR

```
@Html.EJ().Diagram("OrganizationChart", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Method to define whether the command can be executed at the current moment
function canExecuteClone(args) {
//Defines that the clone command can be executed, if and only if the
selection list is not empty.
if (args.model.selectedItems.children.length) {
return true;
}
}
//Command handler
function executeClone(args) {
var diagram = $("#diagram").ejDiagram("instance");
//Logic to clone the selected element
diagram.copy();
diagram.paste();
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Commands to clone the selected item
Command clone = new Command() {
Execute = "executeClone", CanExecute = "canExecuteClone",
//Defines that the clone command has to be executed on the recognition of
Shift+C key press.
Gesture = new Gesture() { Key = Keys.C, KeyModifiers = KeyModifiers.Shift }
};
model.CommandManager.Commands.Add("clone", clone);
ViewData["diagramModel"] = model;
return View();
}
```

Modify the existing command

When any one of the default commands is not desired, they can be disabled. To change the functionality of a specific command, the command can be completely modified.

The following code example illustrates how to disable a command and how to modify the built-in commands.

RAZOR

```
@Html.EJ().Diagram("OrganizationChart", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
//Method to define whether the command can be executed at the current moment
function canExecuteNudgeLeft(args) {
if (args.model.selectedItems.children.length) {
return true;
}
}
//Command handler
function executeNudgeLeft(args) {
var diagram = $("#diagram").ejDiagram("instance");
diagram.nudge("left");
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Command nudgeLeft = new Command() { Execute = "executeNudgeLeft", CanExecute
= "canExecuteNudgeLeft", Gesture = new Gesture() { Key = Keys.Left } };
//Assigns null value to an existing command and disables its execution
model.CommandManager.Commands.Add("nudgeUp", new Command());
model.CommandManager.Commands.Add("nudgeDown", new Command());
model.CommandManager.Commands.Add("nudgeRight", new Command());
//Modifies the existing command - nudgeLeft
model.CommandManager.Commands.Add("nudgeLeft", nudgeLeft);
ViewData["diagramModel"] = model;
return View();
}
```

Data Binding

- Diagram can be populated with the nodes and connectors based on the information provided from an external data source.
- Diagram exposes its specific data-related properties allowing you to specify the data source fields from where the node information has to be retrieved from.

To explore those properties, see Data source settings

- Diagram supports three different kinds of Data binding.
- Local Data
- Remote Data
- HTML Table Data

Local Data

Diagram can be populated based on the user defined JSON data (**Local Data**) by mapping the relevant data source fields.

To map the user defined JSON data with Diagram, you have to configure the fields of DataSourceSettings. The following code example illustrates how to bind local data with the Diagram.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Binds custom JSON with node
function nodeTemplate(diagram, node) {
// Sets the Name field of JSON data as label.
node.labels[0].text = node.Name;
}
</script>
```

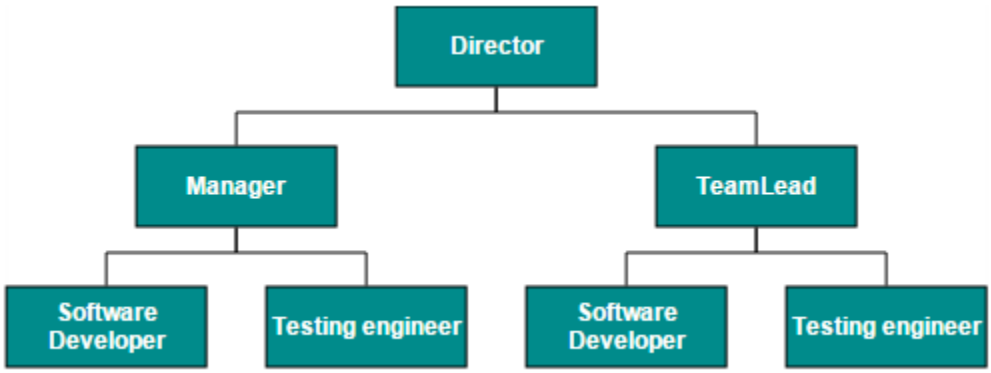
C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.HierarchicalTree;
model.Layout.HorizontalSpacing=30;
model.Layout.VerticalSpacing=30;
//Sets the default properties for nodes and connectors
Label label = new Label() { Name = "label1", FontColor = "white", Bold =
true };
model.DefaultSettings.Node = new Node() { Width = 100, Height = 40,
FillColor = "darkcyan" };
model.DefaultSettings.Node.Labels.Add(label);
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
//Defines the unique field of each JSON data
model.DataSourceSettings.Id = "Name";
//Defines the parent field which builds the relationship
model.DataSourceSettings.Parent = "ReportingPerson";
//Sets the local data source to the diagram.
model.DataSourceSettings.DataSource = GetDataSource();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
{root: {data: [
{ "Name": "Director" },
{ "Name": "Manager", "ReportingPerson": "Director" },
{ "Name": "TeamLead", "ReportingPerson": "Director" },
{ "Name": "Software Developer", "ReportingPerson": "TeamLead" },
{ "Name": "Testing engineer", "ReportingPerson": "TeamLead" },
{ "Name": "Software Developer", "ReportingPerson": "Manager" },
{ "Name": "Testing engineer", "ReportingPerson": "Manager" }
]}}
```



Remote Data

You can bind the Diagram with Remote Data by using dataManager.

- DataManager supports the following types of data-binding: JSON, Web Services, oData.
- It uses two different classes: ej.DataManager for processing and ej.Query for serving data. ej.DataManager communicates with data source and ej.Query generates data queries that are read by the dataManager.
- To learn more, refer to Data Manager.

To bind remote data to the Diagram, you have to configure the fields of DataSourceSettings. The following code illustrates how to bind remote data to the Diagram.

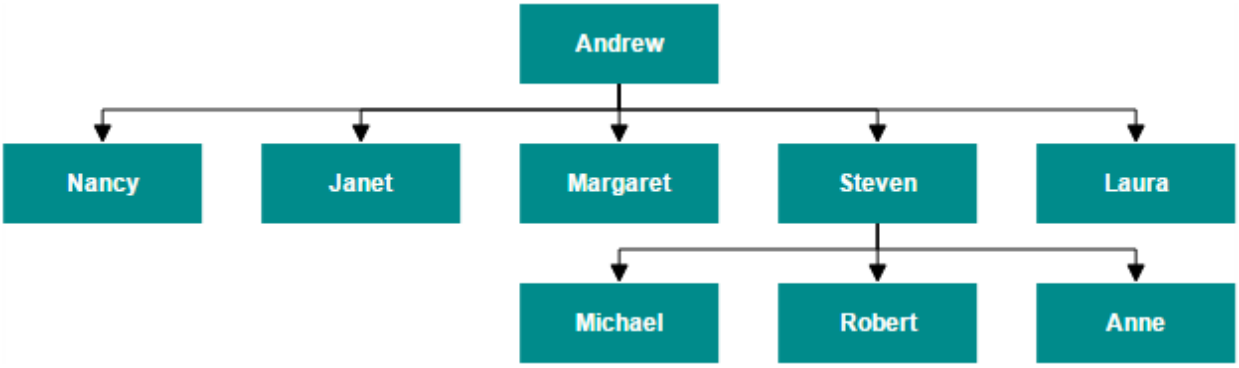
RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Binds custom JSON with node
function nodeTemplate(diagram, node) {
// Sets the Name field of JSON data as label.
node.labels[0].text = node.FirstName;
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
```

```
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.HierarchicalTree;
model.Layout.HorizontalSpacing = 30;
model.Layout.VerticalSpacing = 30;
//Sets the default properties for nodes and connectors
Label label = new Label() { Name = "label1", FontColor = "white", Bold =
true };
model.DefaultSettings.Node = new Node() { Width = 100, Height = 40,
FillColor = "darkcyan" };
model.DefaultSettings.Node.Labels.Add(label);
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
model.DataSourceSettings.DataSource =
"http://mvc.syncfusion.com/Services/Northwnd.svc/";
model.DataSourceSettings.Query =
"ej.Query().from('Employees').select('EmployeeID,ReportsTo,FirstName')";
model.DataSourceSettings.Id = "EmployeeID";
model.DataSourceSettings.Parent = "ReportsTo";
model.DataSourceSettings.TableName = "Employees";
}
```



HTML Table Data

The Diagram provides support to populate the Diagram from the **HTML table**. It is flexible to convert HTML table to Diagram by using **Data Manager**.

The following code illustrates how to convert HTML table to the Diagram.

C#

```
DiagramProperties Model = new DiagramProperties();
Model.Height = "600px";
Model.Width = "600px";
//Initializes the node template.
Model.NodeTemplate = "nodeTemplate";
Model.Layout.Type = LayoutTypes.HierarchicalTree;
Model.Layout.VerticalSpacing = 30;
Model.Layout.HorizontalSpacing = 30;
Model.DefaultSettings.Node = new Node()
```

Copyright © 2001 - 2020 Syncfusion Inc.

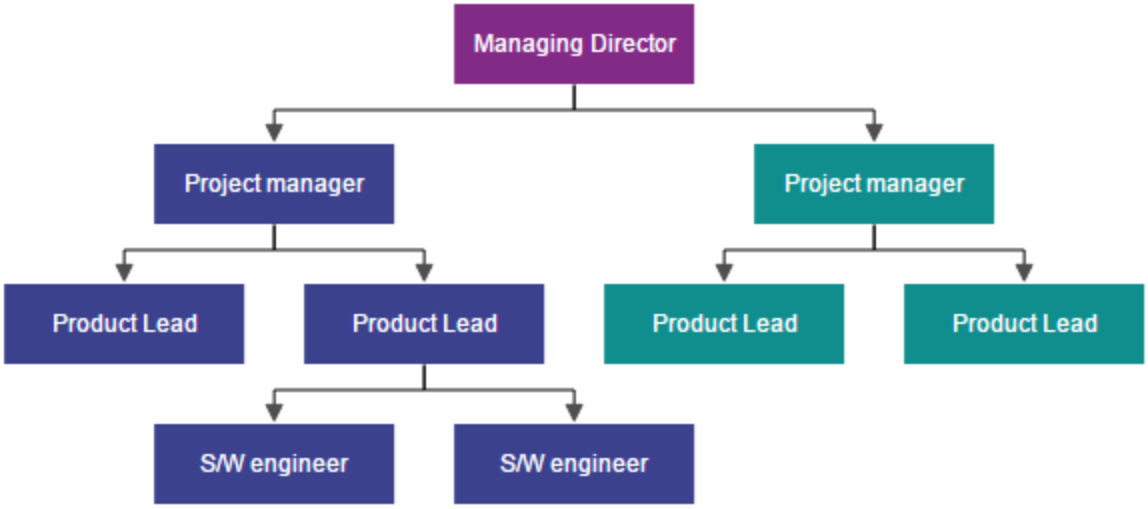
```
{
Width = 120,
Height = 40,
BorderColor = "transparent",
Labels = new Collection() { new Label() { Name = "label1", FontColor =
"#ffffff" } }
};
Model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { FillColor = "#4F4F4F", BorderColor =
"#4F4F4F" }
};
ViewData["Diagram"] = Model;
```

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
@using Syncfusion.JavaScript.DataVisualization.DiagramEnums
@using Syncfusion.JavaScript.DataVisualization.Models.Diagram
@using Syncfusion.JavaScript.DataVisualization.Models.Collections
@(Html.EJ().Diagram("Diagram").Width("100%").Height("600px")
@*Configures data source for Diagram*@
.DataSourceSettings(e =>
{
e.DataSource(dataSource => dataSource.Table("#binding"))
.Id("Id")
.Parent("ReportingPerson");
}))
<!-- HTML Table -->
<script id="binding" type="text/template" >
<table style="display: none">
<thead>
<tr>
<th>Id</th>
<th>Designation</th>
<th>Color</th>
<th>ReportingPerson</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>Managing Director</td>
<td>#822b86</td>
<td>null</td>
</tr>
<tr>
<td>1</td>
<td>Project manager</td>
<td>#3c418d</td>
<td>parent</td>
</tr>
<tr>
<td>2</td>
```

```
<td>Project manager</td>
<td>#108d8d</td>
<td>parent</td>
</tr>
<tr>
<td>3</td>
<td>Product Lead</td>
<td>#3c418d</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Product Lead</td>
<td>#3c418d</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Product Lead</td>
<td>#108d8d</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Product Lead</td>
<td>#108d8d</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>S/W engineer</td>
<td>#3c418d</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>S/W engineer</td>
<td>#3c418d</td>
<td>4</td>
</tr>
</tbody>
</table>
</script>
<script type="text/javascript">
//Binds custom JSON with node
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Designation;
node.fillColor = node.Color;
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



CRUD

- This feature allows the user to read the DataSource and perform add/edit/delete the data in datasource at runtime.

Read DataSource

- This feature allows you to define nodes and connectors collection in the DataSource and ConnectionDataSource respectively.
- We can set the data collection in the model's DataSourceSettings DataSource property. The nodes will be generated based on the data specified in the DataSource.
- We can set the connector collection in the model's DataSourceSettings ConnectionDataSource property.
- If we have a data (data will be set in the DataSource property) with parent relationship in the database and also defined the connector in the ConnectionDataSource simultaneously, then the connectors set in the ConnectionDataSource will be considered as a priority to render the connector.

C#

```
model.DataSourceSettings = new DataSourceSettings()
{
//set the data collection
DataSource = DiagramContext.HierarchicalData.ToList(),
//bind the column name to the Id property which generates the unique name
for the object
Id = "Name",
//Specify the custom property/columns which is available in the DataSource.
CustomFields = new List<string>()
{
"Description",
"Color"
},
ConnectionDataSource = new ConnectionDataSourceSettings()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
//set the connector collection
DataSource = DiagramContext.HierarchicalDetails.ToList(),
//bind the column name to the Id property which generates the unique name
for the object
Id = "Name",
SourceNode = "SourceNode",
TargetNode = "TargetNode",
//Specify the custom property/columns which is available in the
ConnectionDataSource.
CustomFields = new List<string>()
{
"Description",
"Color"
},
}
};
```

How to perform Editing at runtime

We need to specify the URL for create/update/destroy of nodes and connectors.

InsertData

- The following code example illustrates how to send the newly added/inserted data from client to server side.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Sends the newly added nodes/connectors from client side to the server side
through the URL which is specified in server side.
diagram.insertData();
```

C#

```
model.DataSourceSettings = new DataSourceSettings()
{
CrudAction = new CRUDAction()
{
//Specify the method name which is used to get the newly added data from
client side to the server side
Create = "/Diagram/InsertShape",
},
ConnectionDataSource = new ConnectionDataSourceSettings()
{
CrudAction = new CRUDAction()
{
//Specify the method name which is used to get the newly added data from
client side to the server side
Create = "/Diagram/InsertConnector",
}
}
};
[HttpPost]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public void InsertShape(List<HierarchicalData> data)
{
foreach (HierarchicalData hierarchicalData in data)
{
DiagramContext.HierarchicalData.InsertOnSubmit(hierarchicalData);
DiagramContext.SubmitChanges();
}
}
[HttpPost]
public void InsertConnector(List<HierarchicalDetail> data)
{
foreach (HierarchicalDetail hierarchicalData in data)
{
DiagramContext.HierarchicalDetails.InsertOnSubmit(hierarchicalData);
DiagramContext.SubmitChanges();
}
}
```

UpdateData

- The following code example illustrates how to send the updated data from client to the server side.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Sends the updated nodes/connectors from client side to the server side
through the URL which is specified in server side.
diagram.updateData();
```

C#

```
model.DataSourceSettings = new DataSourceSettings()
{
CrudAction = new CRUDAction()
{
//Specify the method name which is used to get the updated data from client
side to the server side
Update = "/Diagram/UpdateShape",
},
ConnectionDataSource = new ConnectionDataSourceSettings()
{
CrudAction = new CRUDAction()
{
//Specify the method name which is used to get the updated data from client
side to the server side
Update = "/Diagram/UpdateConnector",
}
}
};
[HttpPost]
public void UpdateShape(List<HierarchicalData> data)
{
foreach (HierarchicalData hierarchicalData in data)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
HierarchicalData originalData = DiagramContext.HierarchicalData.Single(h =>
h.Name == hierarchicalData.Name);
originalData.Description = hierarchicalData.Description;
originalData.Color = hierarchicalData.Color;
DiagramContext.SubmitChanges();
}
}
[HttpPost]
public void UpdateConnector(List<HierarchicalDetail> data)
{
foreach (HierarchicalDetail hierarchicalData in data)
{
HierarchicalDetail originalData =
DiagramContext.HierarchicalDetails.Single(h => h.Name ==
hierarchicalData.Name);
originalData.SourceNode = hierarchicalData.SourceNode;
originalData.TargetNode = hierarchicalData.TargetNode;
DiagramContext.SubmitChanges();
}
}
```

DeleteData

The following code example illustrates how to send the deleted data from client to the server side.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Sends the deleted nodes/connectors from client side to the server side
through the URL which is specified in server side.
diagram.removeData();
```

C#

```
model.DataSourceSettings = new DataSourceSettings()
{
CrudAction = new CRUDAction()
{
//Specify the method name which is used to get the deleted data from client
side to the server side
Destroy = "/Diagram/DeleteShape",
},
ConnectionDataSource = new ConnectionDataSourceSettings()
{
CrudAction = new CRUDAction()
{
//Specify the method name which is used to get the updated data from client
side to the server side
Destroy = "/Diagram/DeleteConnector",
}
}
};
[HttpPost]
public void DeleteShape(List<HierarchicalData> data)
{
foreach (HierarchicalData hierarchicalData in data)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
HierarchicalData originalData = DiagramContext.HierarchicalData.Single(h =>
h.Name == hierarchicalData.Name);
DiagramContext.HierarchicalData.DeleteOnSubmit(originalData);
DiagramContext.SubmitChanges();
}
}
[HttpPost]
public void DeleteConnector(List<HierarchicalDetail> data)
{
foreach (HierarchicalDetail hierarchicalData in data)
{
HierarchicalDetail originalData =
DiagramContext.HierarchicalDetails.Single(h => h.Name ==
hierarchicalData.Name);
DiagramContext.HierarchicalDetails.DeleteOnSubmit(originalData);
DiagramContext.SubmitChanges();
}
}
```

Automatic Layout

Diagram provides support to auto-arrange the nodes in the Diagram area that is referred as **Layout**. It includes the following layout modes.

- Hierarchical Layout
- Organization Chart
- Radial Tree

Hierarchical Layout

The Hierarchical Tree Layout arranges nodes in a tree-like structure, where the nodes in the hierarchical layout may have multiple parents. There is no need to specify the layout root.

To arrange the nodes in hierarchical structure, you need to specify the layout Type as hierarchical tree. The following example shows how to arrange the nodes in a hierarchical structure.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Binds custom JSON with node
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Name;
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.HierarchicalTree;
model.Layout.HorizontalSpacing=30;
```

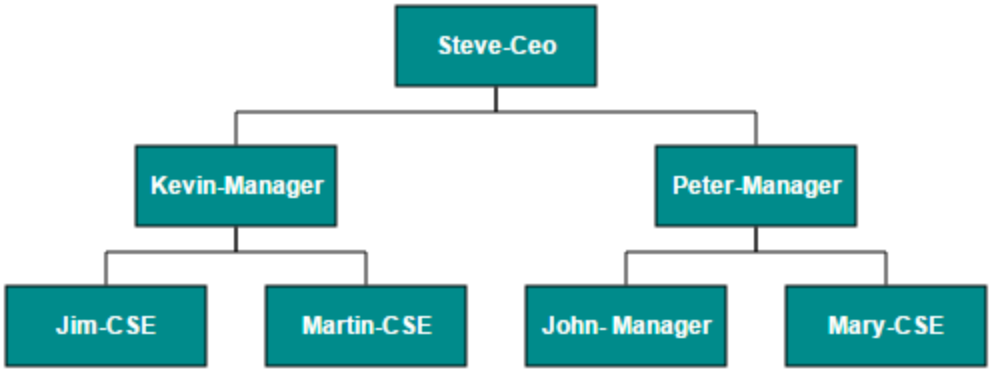
Copyright © 2001 - 2020 Syncfusion Inc.

```
model.Layout.VerticalSpacing=30;
//Sets the default properties for nodes and connectors
Label label = new Label() { Name = "label1", FontColor = "white" };
model.DefaultSettings.Node = new Node() { Width = 100, Height = 40,
FillColor = "darkcyan" };
model.DefaultSettings.Node.Labels.Add(label);
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
//Defines the unique field
model.DataSourceSettings.Id = "Name";
//Defines the relationship
model.DataSourceSettings.Parent = "ReportingPerson";
//Specifies the dataSource
model.DataSourceSettings.DataSource = GetDataSource();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

```
{root: {data: [
{Name: "Steve-CEO"},
{Name: "Kevin-Manager", ReportingPerson: "Steve-CEO"},
{Name: "Peter-Manager", ReportingPerson: "Steve-CEO"},
{Name: "John- Manager", ReportingPerson: "Peter-Manager"},
{Name: "Mary-CSE ", ReportingPerson: "Peter-Manager"},
{Name: "Jim-CSE ", ReportingPerson: "Kevin-Manager"},
{Name: "Martin-CSE", ReportingPerson: "Kevin-Manager"}
]}}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Note: You can ignore a particular Node from layout arrangement by setting its **ExcludeFromLayout** property as true.

Radial Tree Layout

The Radial Tree layout arranges nodes on a virtual concentric circles around a root node. Sub-trees formed by the branching of child nodes are located radially around the child nodes. This arrangement results in an ever-expanding concentric arrangement with radial proximity to the root node indicating the node level in the hierarchy. When no root node is set, the algorithm automatically considers one of the Diagram nodes as the root node.

To arrange nodes in a radial tree structure, you need to set the `Layout.Type` as `RadialTree`. The following code illustrates how to arrange the nodes in a radial tree structure.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Binds custom JSON with node
function nodeTemplate(diagram, node) {
node.name = node.Id;
node.source = node.ImageUrl;
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.RadialTree;
model.Layout.HorizontalSpacing=30;
model.Layout.VerticalSpacing=30;
//Sets the default properties for nodes
model.DefaultSettings.Node = new ImageNode() { Width = 40, Height = 40,
BorderColor = "transparent" };
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "ReportingPerson";
model.DataSourceSettings.DataSource = GetOrgChartData();
//Initializes the node template.
```

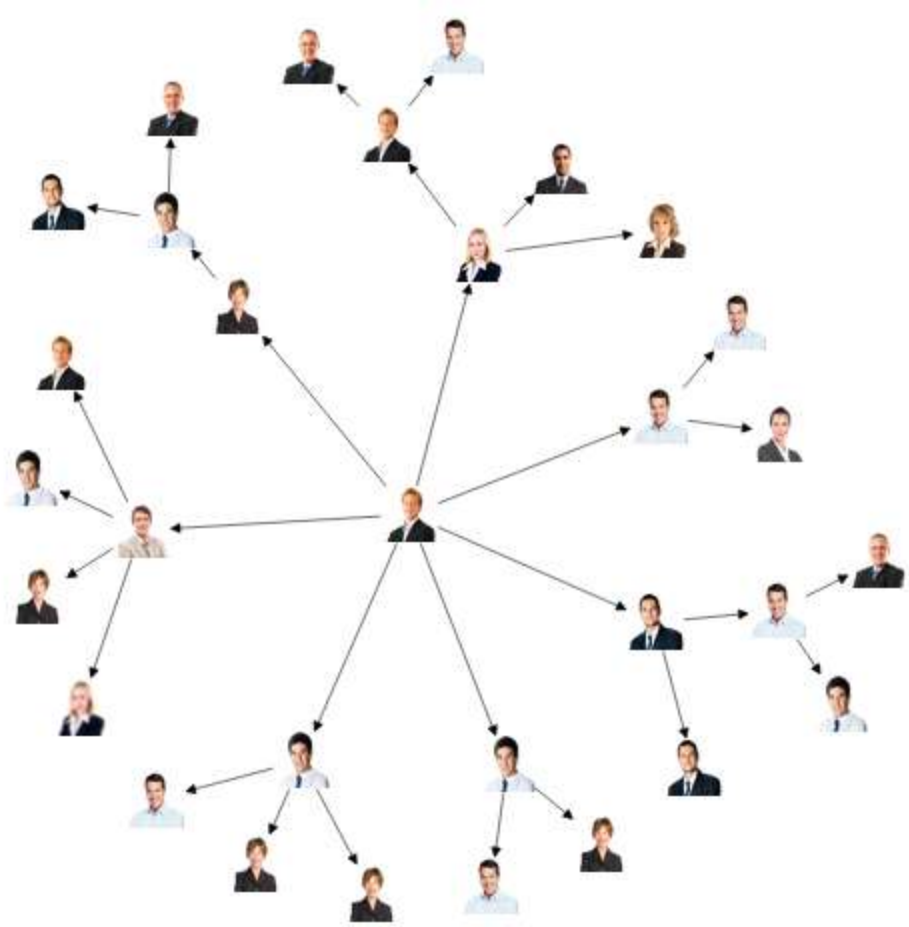
Copyright © 2001 - 2020 Syncfusion Inc.

```
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

```
{root: {data: [
{Id: "parent", ImageUrl: "images/Clayton.png"},
{Id: 1, ImageUrl: "images/image55.png", ReportingPerson: "parent"},
{Id: 2, ImageUrl: "images/Robin.PNG", ReportingPerson: "parent"},
{Id: 3, ImageUrl: "images/Robin.PNG", ReportingPerson: "parent"},
{Id: 4, ImageUrl: "images/Paul.png", ReportingPerson: "parent"},
{Id: 5, ImageUrl: "images/image53.png", ReportingPerson: "parent"},
{Id: 6, ImageUrl: "images/Maria.png", ReportingPerson: "parent"},
{Id: 7, ImageUrl: "images/Jenny.png", ReportingPerson: 3},
{Id: 8, ImageUrl: "images/Thomas.PNG", ReportingPerson: "parent"},
{Id: 9, ImageUrl: "images/Jenny.png", ReportingPerson: 2},
{Id: 10, ImageUrl: "images/Thomas.png", ReportingPerson: 2},
{Id: 11, ImageUrl: "images/Maria.PNG", ReportingPerson: 4},
{Id: 12, ImageUrl: "images/Thomas.PNG", ReportingPerson: 1},
{Id: 13, ImageUrl: "images/Clayton.png", ReportingPerson: 6},
{Id: 14, ImageUrl: "images/Jenny.png", ReportingPerson: 3},
{Id: 15, ImageUrl: "images/Thomas.png", ReportingPerson: 3},
{Id: 16, ImageUrl: "images/John.png", ReportingPerson: 6},
{Id: 17, ImageUrl: "images/Jenny.png", ReportingPerson: 4},
{Id: 18, ImageUrl: "images/Robin.png", ReportingPerson: 4},
{Id: 19, ImageUrl: "images/Clayton.png", ReportingPerson: 4},
{Id: 20, ImageUrl: "images/image57.png", ReportingPerson: 12},
{Id: 21, ImageUrl: "images/Robin.png", ReportingPerson: 5},
{Id: 22, ImageUrl: "images/image51.png", ReportingPerson: 6},
{Id: 23, ImageUrl: "images/image55.png", ReportingPerson: 19},
{Id: 24, ImageUrl: "images/Thomas.png", ReportingPerson: 8},
{Id: 25, ImageUrl: "images/image56.png", ReportingPerson: 8},
{Id: 26, ImageUrl: "images/image55.png", ReportingPerson: 1},
{Id: 27, ImageUrl: "images/image57.png", ReportingPerson: 13},
{Id: 28, ImageUrl: "images/Robin.PNG", ReportingPerson: 12},
{Id: 29, ImageUrl: "images/Thomas.PNG", ReportingPerson: 13},
{Id: 30, ImageUrl: "images/image57.png",ReportingPerson: 19}
]}}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Organizational Chart

An **Organizational Chart** is a Diagram that displays the structure of an organization and relationships. To create an organizational chart, `Layout.Type` should be set as `OrganizationalChart`.

The following code example illustrates how to create an organizational chart.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Binds custom JSON with node
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Role;
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.OrganizationalChart;
```

Copyright © 2001 - 2020 Syncfusion Inc.

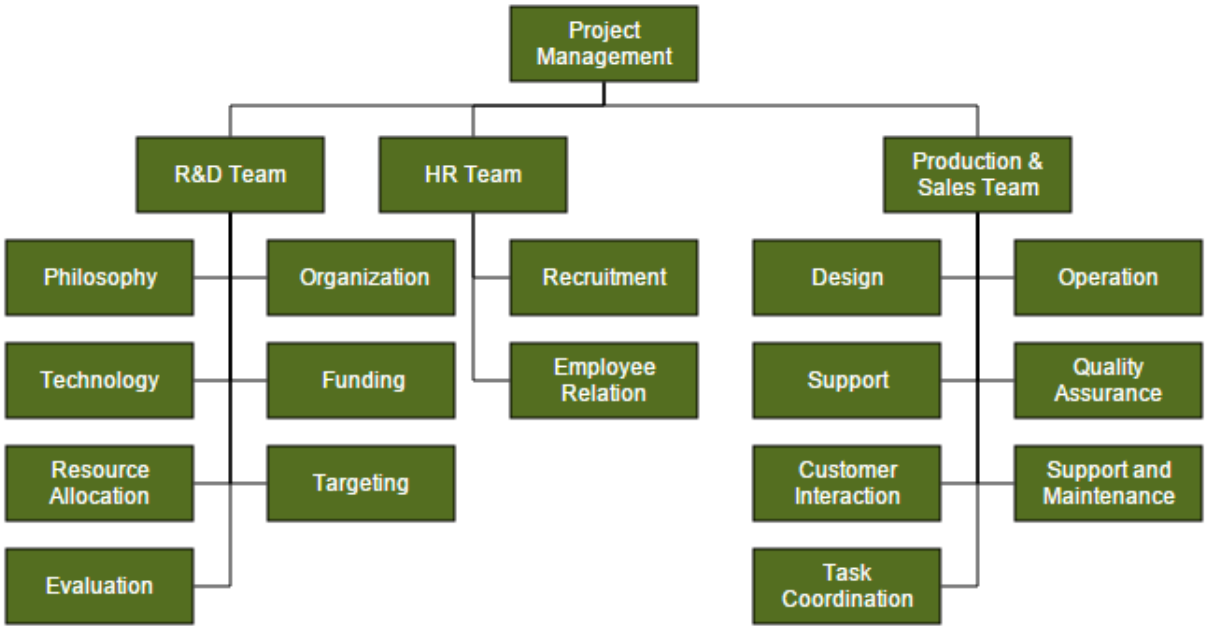
```
model.Layout.HorizontalSpacing=30;
model.Layout.VerticalSpacing=30;
//Sets the default properties for nodes
Label label = new Label() { Name = "label1", FontColor = "white" };
model.DefaultSettings.Node = new Node() { Width = 100, Height = 40,
FillColor = "#546e20" };
model.DefaultSettings.Node.Labels.Add(label);
//Sets the default properties for connectors
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "Team";
model.DataSourceSettings.DataSource = GetOrgChartData();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

```
{root: {data: [
{Id: "parent", Role: "Project Management"},
{Id: 1, Role: "R&D Team", Team: "parent"},
{Id: 3, Role: "Philosophy", Team: "1"},
{Id: 4, Role: "Organization", Team: "1"},
{Id: 5, Role: "Technology", Team: "1"},
{Id: 7, Role: "Funding", Team: "1"},
{Id: 8, Role: "Resource Allocation", Team: "1"},
{Id: 9, Role: "Targeting", Team: "1"},
{Id: 11, Role: "Evaluation", Team: "1"},
{Id: 156, Role: "HR Team", Team: "parent"},
{Id: 13, Role: "Recruitment", Team: "156"},
{Id: 113, Role: "Training", Team: "12"},
{Id: 112, Role: "Employee Relation", Team: "156"},
{Id: 14, Role: "Record Keeping", Team: "12"},
{Id: 15, Role: "Compensations & Benefits", Team: "12"},
{Id: 16, Role: "Compliances", Team: "12"},
{Id: 17, Role: "Production & Sales Team", Team: "parent"},
{Id: 119, Role: "Design", Team: "17"},
{Id: 19, Role: "Operation", Team: "17"},
{Id: 20, Role: "Support", Team: "17"},
{Id: 21, Role: "Quality Assurance", Team: "17"},
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{Id: 23, Role: "Customer Interaction", Team: "17"},
{Id: 24, Role: "Support and Maintenance", Team: "17"},
{Id: 25, Role: "Task Coordination", Team: "17"}
]}}
```



Organizational chart layout starts parsing from root and iterate through all its child elements. 'getLayoutInfo' method provides necessary information of a node's children and the way to arrange (direction, orientation, offsets, etc.) them. You can customize the arrangements by overriding this function as explained.

GetLayoutInfo

You can set Chart orientations, chart types, and offset to be left between parent and child nodes by overriding the method, model.layout.getLayoutInfo. The getLayoutInfo method is called to configure every subtree of the organizational chart. It takes the following arguments.

- diagram: Reference of diagram
- node: Parent node to that options are to be customized
- options: object to set the customizable properties

The following code example illustrates how to define the method getLayoutInfo.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
var chartOrientations = ej.datavisualization.Diagram.ChartOrientations;
var chartTypes = ej.datavisualization.Diagram.ChartTypes;
//Defines getLayoutInfo
function getLayoutInfo(diagram, node, options) {
options.orientation = chartOrientations.Vertical;
//Configures the sub tree of the node vertically at right-side.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
options.type = chartTypes.Right;
options.offset = 10;
}
</script>
```

The following table illustrates the properties that "options" argument takes.

Property|Description|Default Value

options.children|Contains the list of child nodes. Children collection can be modified.|Array of child nodes

options.assistants|By default, the collection is empty. When any of the child nodes have to be set as "Assistant", you can remove from children collection and have to insert into assistants collection. |Empty array

options.orientation|Gets or sets the organizational chart orientation. |ChartOrientation.Vertical

options.type|Gets or sets the chart organizational chart type |For horizontal chart orientation:ChartType.Center For Vertical chart orientation:ChartType.Alternate

options.offset|Offset is the horizontal space to be left between parent and child nodes.|20 pixels.Applicable only for vertical chart orientations.

options.hasSubTree|Gets whether the node contains sub trees.|Boolean

options.level|Gets the depth of the node from layout root|Number

options.enableRouting|By default, Connections are routed based on the chart type and orientations.This property gets or sets whether default routing is to be enabled or disabled.|true

The following table illustrates the different chart orientations and chart types.

| Orientation | Type | Description | Example |

|---|---|---|---|

| Horizontal | Left | Arranges the child nodes horizontally at the left side of parent. |



|

| | Right | Arranges the child nodes horizontally at the right side of parent. |



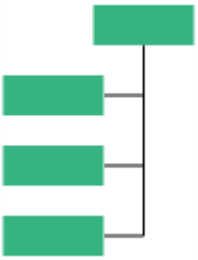
|

Copyright © 2001 - 2020 Syncfusion Inc.

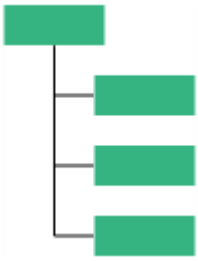
| | Center | Arranges the children like standard tree layout orientation. |



|



| Vertical | Left | Vertically arranges the children at the left side of parent | |



| | Right | Vertically arranges the children at the right side of parent | |

| | Alternate | Vertically arranges the children at both left and right sides of parent |



|

The following code example illustrates how to set the vertical right arrangement to the leaf level trees.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
var chartOrientations = ej.datavisualization.Diagram.ChartOrientations;
var chartTypes = ej.datavisualization.Diagram.ChartTypes;
//Creates the node template
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Role;
}
function getLayoutInfo(diagram, node, options) {
if (!options.hasSubTree) {
options.type = chartTypes.Right;
options.orientation = chartOrientations.Vertical;
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

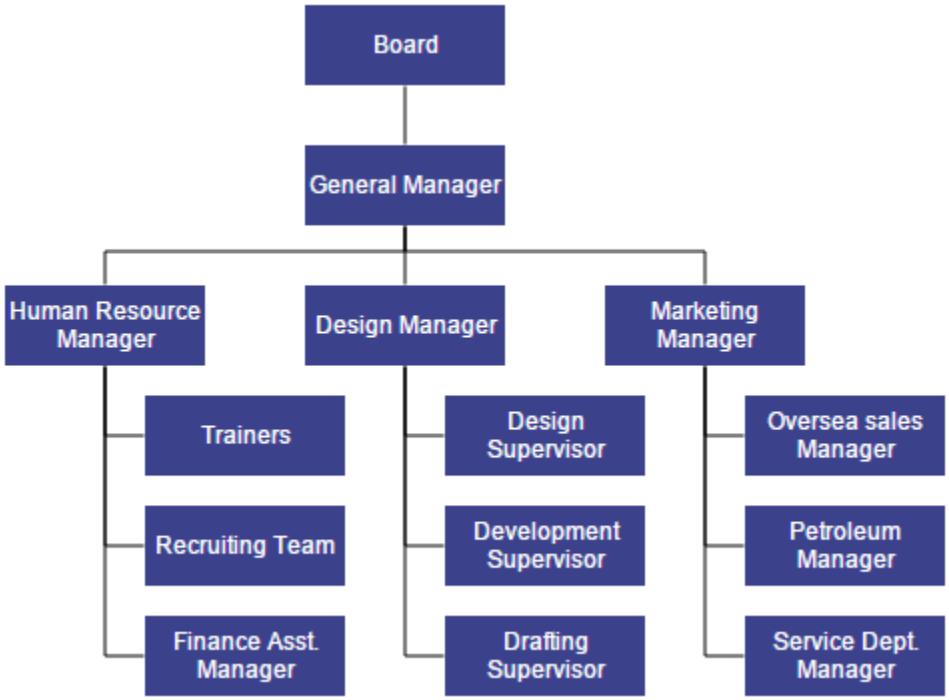
```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.OrganizationalChart;
model.Layout.HorizontalSpacing=30;
model.Layout.VerticalSpacing=30;
model.Layout.GetLayoutInfo = "getLayoutInfo";
//Sets the default properties for nodes
Label label = new Label() { Name = "label1", FontColor = "white" };
model.DefaultSettings.Node = new Node() { Width = 100, Height = 40,
FillColor = "#546e20" };
model.DefaultSettings.Node.Labels.Add(label);
//Sets the default properties for connectors
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "Manager";
model.DataSourceSettings.DataSource = GetOrgChartData();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

```
{root: {data: [
{ Id: "parent", Role: "Board" },
{ Id: "1", Role: "General Manager", Manager: "parent" },
{ Id: "2", Role: "Human Resource Manager", Manager: "1" },
{ Id: "3", Role: "Trainers", Manager: "2" },
{ Id: "4", Role: "Recruiting Team", Manager: "2" },
{ Id: "5", Role: "Finance Asst. Manager", Manager: "2" },
{ Id: "6", Role: "Design Manager", Manager: "1" },
{ Id: "7", Role: "Design Supervisor", Manager: "6" },
{ Id: "8", Role: "Development Supervisor", Manager: "6" },
{ Id: "9", Role: "Drafting Supervisor", Manager: "6" },
{ Id: "10", Role: "Marketing Manager", Manager: "1" },
{ Id: "11", Role: "Oversea sales Manager", Manager: "10" },
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ Id: "12", Role: "Petroleum Manager", Manager: "10" },
{ Id: "13", Role: "Service Dept. Manager", Manager: "10" }
]}}
```



Assistant

Assistants are child item that have a different relationship with the parent node. They are laid out in a dedicated part of the tree. You can specify a node as an assistant of its parent by adding it to assistants property of the argument "options".

The following code example illustrates how to add assistants to layout.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Creates the node template
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Role;
}
//Defines getLayoutInfo
function getLayoutInfo(diagram, node, options) {
if (node.labels[0].text == "General Manager") {
options.assistants.push(options.children[0]);
options.children.splice(0, 1);
}
options.orientation = "horizontal";
options.type = "center"
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

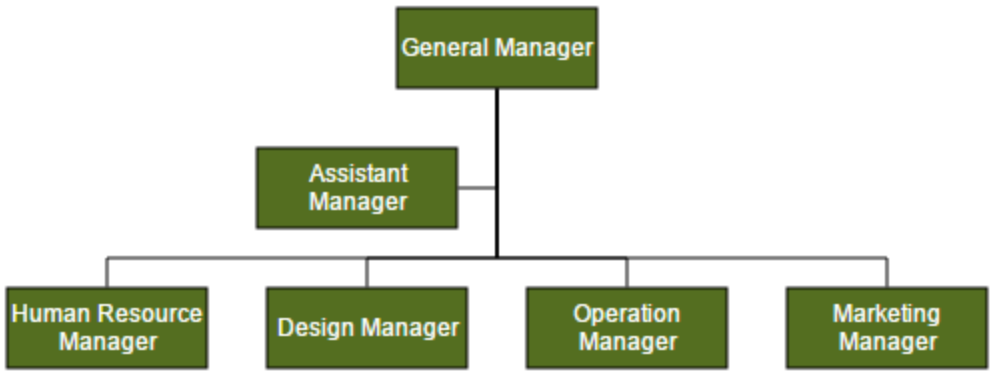
C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Uses layout to auto-arrange nodes on the Diagram page
model.Layout.Type = LayoutTypes.OrganizationalChart;
model.Layout.HorizontalSpacing=30;
model.Layout.VerticalSpacing=30;
model.Layout.GetLayoutInfo = "getLayoutInfo";
//Sets the default properties for nodes
Label label = new Label() { Name = "label1", FontColor = "white" };
model.DefaultSettings.Node = new Node() { Width = 100, Height = 40,
FillColor = "#546e20" };
model.DefaultSettings.Node.Labels.Add(label);
//Sets the default properties for connectors
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "Team";
model.DataSourceSettings.DataSource = GetOrgChartData();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

```
{root: {data: [
{ Id: 1, Role: "General Manager" },
{ Id: 2, Role: "Assistant Manager", Team: 1 },
{ Id: 3, Role: "Human Resource Manager", Team: 1 },
{ Id: 4, Role: "Design Manager", Team: 1 },
{ Id: 5, Role: "Operation Manager", Team: 1 },
{ Id: 6, Role: "Marketing Manager", Team: 1 }
]}}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customize Layout

Orientation, spacings, and position of layout can be customized with a set of properties.

To explore layout properties, refer to Layout Properties.

Layout Orientation

Diagram provides support to customize the orientation of layout. You can set the desired orientation using Layout.Orientation. For more information about orientation, refer to Layout Orientations

The following code illustrates how to arrange the nodes in a "BottomToTop" orientation.

RAZOR

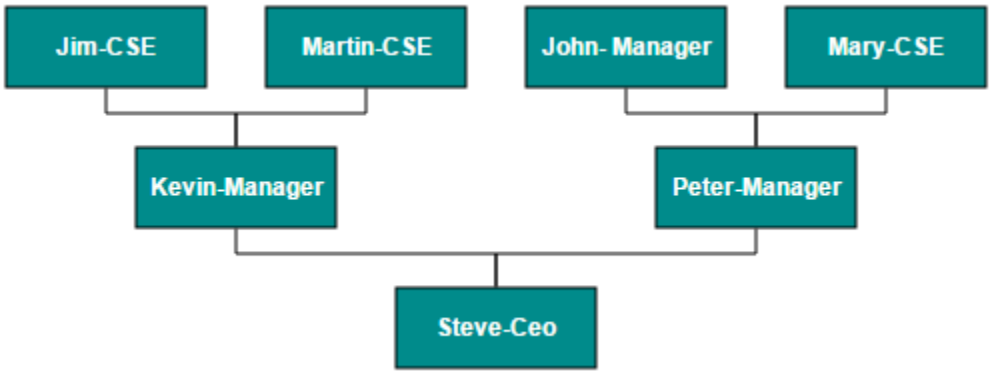
```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Creates the node template
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Role;
}
//Defines getLayoutInfo
function getLayoutInfo(diagram, node, options) {
if (node.labels[0].text == "General Manager") {
options.assistants.push(options.children[0]);
options.children.splice(0, 1);
}
options.orientation = "horizontal";
options.type = "center";
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Sets the type of the layout
model.Layout.Type = LayoutTypes.HierarchicalTree;
//Sets the orientation
model.Layout.Orientation = LayoutOrientations.BottomToTop;
//Sets the space to be horizontally left between nodes
model.Layout.HorizontalSpacing = 20;
//Sets the space to be vertically left between nodes
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
model.Layout.VerticalSpacing = 20;
model.Layout.GetLayoutInfo = "getLayoutInfo";
//Sets the default properties for nodes and connectors
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 40, FillColor = "#546e20",
Labels = new Collection() { new Label() { Name = "label1", FontColor =
"white" } }
};
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "Manager";
model.DataSourceSettings.DataSource = GetOrgChartData();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```



Fixed Node

Layout provides support to arrange the nodes with reference to the position of a fixed node and the fixed node has to be set to the Layout.FixedNode.

This is helpful when you try to expand/collapse a node. It might be expected that the position of the double-clicked node should not be changed.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
//Creates the node template
function nodeTemplate(diagram, node) {
node.labels[0].text = node.Role;
}
//Defines getLayoutInfo
function getLayoutInfo(diagram, node, options) {
options.orientation = "horizontal";
options.type = "center"
}
</script>
```

C#

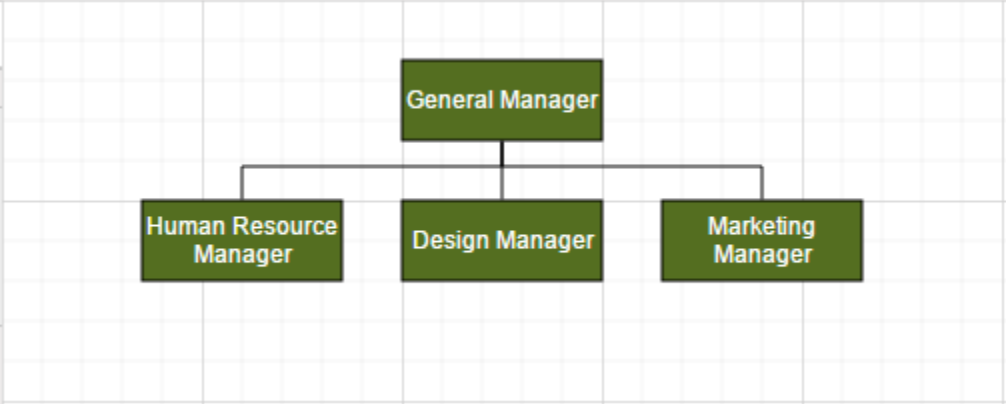
```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Defines default layout as organizational chart
model.Layout.Type = LayoutTypes.HierarchicalTree;
model.Layout.FixedNode = "parent";
model.Layout.GetLayoutInfo = "getLayoutInfo";
//Sets the default properties for nodes and connectors
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 40, FillColor = "#546e20",
Labels = new Collection() { new Label() { Name = "label1", FontColor =
"white" } }
};
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "Manager";
model.DataSourceSettings.DataSource = GetOrgChartData();
//Initializes the node template.
model.NodeTemplate = "nodeTemplate";
ViewData["diagramModel"] = model;
return View();
}
public Array GetDataSource()
{
String allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/data.json"));
Dictionary<string, object> requestArgs = (Dictionary<string, object>)new
JavaScriptSerializer().DeserializeObject(allText);
requestArgs = (Dictionary<string, object>)requestArgs["root"];
return (Array)requestArgs.Values.ElementAt(0);
}
```

JS

```
{root: {data: [
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ name: "parent", Role: "General Manager", offsetX: 250, offsetY: 50 },
{ name: "1", Role: "Human Resource Manager", Manager: "parent" },
{ name: "2", Role: "Design Manager", Manager: "parent"},
{ name: "3", Role: "Marketing Manager", Manager: "parent"}
]}}
```



Expand and collapse

Diagram allows to expand/collapse the sub trees of a layout. Node.IsExpanded allows you to expand/collapse its children. The following code example shows how to expand/collapse the children of a node.

RAZOR

```
@Html.EJ().Diagram("Diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
<script type="text/javascript">
function onDoubleClick(args) {
var diagram = $("#diagram").ejDiagram("instance");
var node = args.element;
// Sets the double clicked node as fixed node
$("#diagram").ejDiagram({ layout: { fixedNode: node.name } });
//Expands/collapses the children of node
diagram.updateNode(node.name, { isExpanded: !node.isExpanded });
}
</script>
```

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Defines default layout as organizational chart
model.Layout.Type = LayoutTypes.HierarchicalTree;
model.Layout.HorizontalSpacing=30;
model.Layout.VerticalSpacing=30;
model.Layout.GetLayoutInfo = "getLayoutInfo";
//Defines double click event
model.DoubleClick = "onDoubleClick";
//Sets the default properties for nodes and connectors
model.DefaultSettings.Node = new Node() {
Width = 100, Height = 40, FillColor = "#546e20",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Labels = new Collection() { new Label() { Name = "label1", FontColor =
"white" } }
};
model.DefaultSettings.Connector = new Connector()
{
Segments = new Collection() { new Segment(Segments.Orthogonal) },
TargetDecorator = new Decorator() { Shape = DecoratorShapes.None },
};
//Configures data source for Diagram
model.DataSourceSettings.Id = "Id";
model.DataSourceSettings.Parent = "Team";
model.DataSourceSettings.DataSource = GetOrgChartData();
ViewData["diagramModel"] = model;
return View();
}
```

In above example, while expanding/collapsing a node, it is set as fixed node in order to prevent it from repositioning.

Refresh layout

Diagram allows to refresh the layout at runtime. To refresh the layout, refer to Refresh layout.

Overview Control

Overview control allows you to see a preview or an overall view of the entire content of a Diagram. This helps you to look at the overall picture of a large Diagram and also to navigate, pan, or zoom, on a particular position of the page.

When you work on a very large Diagram, you may not know the part you are actually working on, or navigation from one part to another might be difficult. One solution for navigation is to zoom out the entire Diagram and find where you are. Then, you can zoom in a particular area you want to. This solution is not suitable when you need some frequent navigation.

Overview control solves these problems by showing you a preview, that is, an overall view of the entire Diagram. A rectangle indicates viewport of the Diagram. Navigation becomes easy by dragging this rectangle.

Create overview

The SourceID property of overview should be set with the corresponding Diagram ID for you need the overall view. The following code illustrates how to create overview.

HTML

```
//Initializes overview
<div>
@Html.EJ().Overview("Overview", ViewData["overview"] as
Syncfusion.JavaScript.DataVisualization.Models.OverviewProperties)
</div>
//Initializes Diagram
<div>
@Html.EJ().Diagram("Diagram1", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
DiagramProperties model = new DiagramProperties();
model.Height = "600px";
model.Width = "1000px";
ViewData["diagramModel"] = model;
OverviewProperties overview = new OverviewProperties();
overview.Height = "300px";
overview.Width = "233px";
overview.SourceID = "Diagram1";
ViewData["overview"] = overview;
```

Zoom and Pan

In overview, the view port of the Diagram is highlighted with a red colored rectangle. Diagram can be zoomed/panned by interacting with that. You can interact with overview as follows.

- Resize the rectangle - Zooms in/out the Diagram
- Drag the rectangle - Pans the Diagram
- Click at a position - Navigates to the clicked region
- Choose a particular region by clicking and dragging - Navigates to the specified region

The following image shows how the Diagram is zoomed/panned with overview.

Symbol Palette

The **SymbolPalette** displays a collection of palettes. The Palette shows a set of nodes and connectors. It allows you to drag and drop the nodes and connectors into the Diagram.

Create symbol palette

The DiagramId property of symbolpalette should be set with the corresponding Diagram ID to drag and drop the nodes and connectors into the Diagram. The following code illustrates how to create symbolpalette.

HTML

```
//Initializes overview
<div>
@Html.EJ().SymbolPalette("SymbolPalette", ViewData["SymbolPaletteModel"] as
Syncfusion.JavaScript.DataVisualization.Models.SymbolPaletteProperties)
@Html.EJ().Diagram("Diagram", ViewData["DiagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
</div>
```

C#

```
//Initializes the symbol palette
SymbolPaletteProperties symbolPalette = new SymbolPaletteProperties();
//Defines the palette collection
symbolPalette.Palettes = new Collection()
{
palette
};
//Specifies the symbol palette size
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
symbolPalette.Width = "100%";
symbolPalette.Height = "100%";
ViewData["SymbolPaletteModel"] = symbolpalette;
DiagramProperties model = new DiagramProperties();
model.Height = "550px";
model.Width = "800px";
ViewData["DiagramModel"] = model;
```

Add palettes to SymbolPalette

A palette allows to display a group of related symbols and it textually annotates the group with its header.

To initialize a palette, define a JSON object with the property `Name` that is displayed as the header text of palette. The `Expanded` property of palette allows to expand/collapse its palette items.

The following code example illustrates how to define a palette and how its added to symbol palette.

JAVASCRIPT

```
//Creates a node
BasicShape ellipse = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
//Specifies node offset
OffsetX = 50,
OffsetY = 50,
//Specifies node shape
Shape = BasicShapes.Ellipse
};
Palette palette = new Palette("Basic Shapes");
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(ellipse);
```

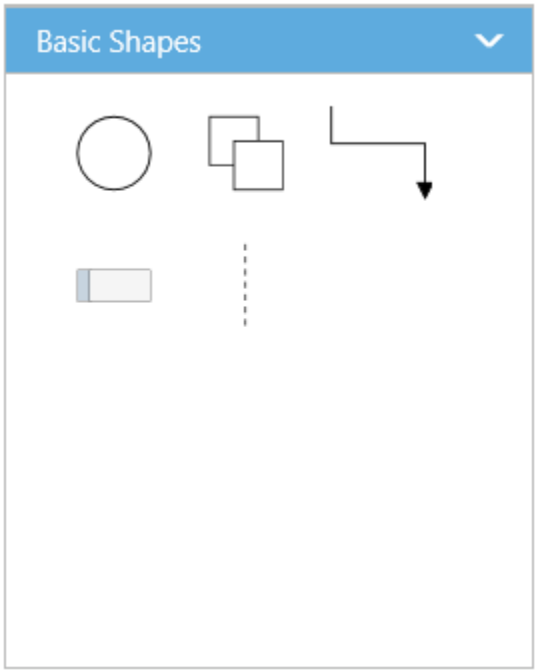
You can add any number of palettes to the `Palettes` collection of the symbol palette. The following example illustrates how to define symbol palette with a palette object that is defined in the previous step.

JAVASCRIPT

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Palettes = new Collection()
{
palette
};
ViewData["SymbolPaletteModel"] = symbolpalette;
DiagramProperties model = new DiagramProperties();
model.Height = "550px";
model.Width = "800px";
ViewData["DiagramModel"] = model;
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following image shows the symbol palette with multiple palette Items.



Customize the Palette Header

Palettes can be annotated with its header texts and you can change the height of palette header by using HeaderHeight property of symbol palette.

Also, you can embed any HTML element into a palette header by defining the ScriptTemplate id to palette's templateId property. Following code example illustrates how to customize palette headers.

HTML

```
<!--dependency scripts-->
<script id="svgTemplate" type="text/x-jsrender">
<!--define html element-->
<svg version="1.0" xmlns="http://www.w3.org/2000/svg" width="225px"
height="28px">
<g visibility="visible">
<image width="26px" height="26px" opacity="1"
xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="image.png"></image>
<text x="40" y="18" font-size="14">
<tspan>Basic Shapes</tspan>
</text>
</g>
</svg>
</script>
```

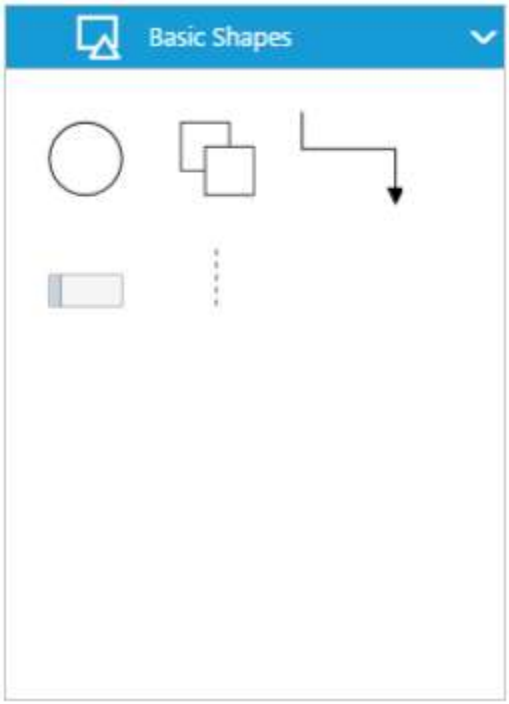
JAVASCRIPT

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
Palette palette = new Palette("Basic Shapes");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Sets the id of the script template
palette.TemplateId = "svgTemplate";
palette.Expanded = true;
symbolpalette.HeaderHeight = 50;
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```

The following image shows the customized palette header



Add symbols to palette

The symbol need to be defined and added to the Items collection of the palette. You can create a symbol as a node, group, connector, lane, or phase except swimlane.

The following code example illustrates how to define a palette with symbols that are defined in the previous section.

JS

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
//Creates a node
BasicShape ellipse = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
//Specifies node offset
OffsetX = 50,
OffsetY = 50,
//Specifies node shape
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Shape = BasicShapes.Ellipse
};
Palette palette = new Palette("Basic Shapes");
//Sets the id of the script template
palette.TemplateId = "svgTemplate";
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(ellipse);
symbolpalette.HeaderHeight = 50;
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```

Customize the size of symbols

You can customize the size of the individual symbol. The PaletteItem property of node enables you to define the size of the symbols. The following code example illustrates how to change the size of a symbol.

JS

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
//Creates a node
BasicShape ellipse = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
//Specifies node shape
Shape = BasicShapes.Ellipse,
PaletteItem = new PaletteItem() { Width = 50, Height = 50, Margin = new
Margin() { Left = 20, Top = 20, Right = 20, Bottom = 20 } }
};
Palette palette = new Palette("Basic Shapes");
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(ellipse);
//Specifies the default size to render symbols
symbolpalette.PaletteItemWidth = 50;
symbolpalette.PaletteItemHeight = 50;
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```

Symbol size will be set based on the following precedence.

Precedence

| Palette Item | Rendering Size |

|---|---|---|

| Width | PaletteItem.Width > model.PaletteItemWidth > Node.Width |

| Height | PaletteItem.height > model.PaletteItem.Height > Node.Height |

Copyright © 2001 - 2020 Syncfusion Inc.

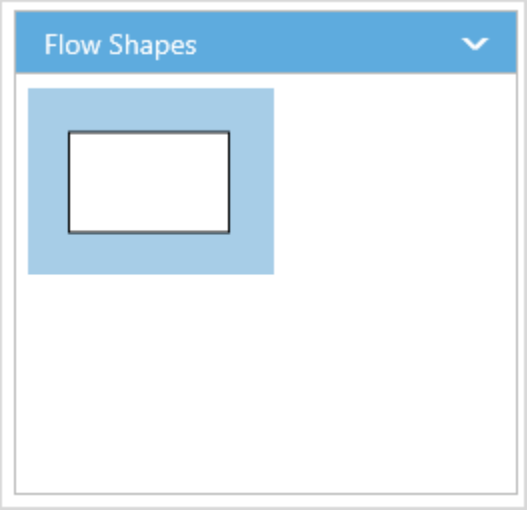
- Symbol size will be rendered in the palette based on Node.PaletteItem's Width and Height property.
- If PaletteItem's Width and Height property is not specified, symbol size will be rendered in the palette based on model's PaletteItemWidth and PaletteItemHeight property.
- If you don't specify above two, then symbol size will be rendered in the palette based on node's Width and Height property.

Stretch the symbols into the palette

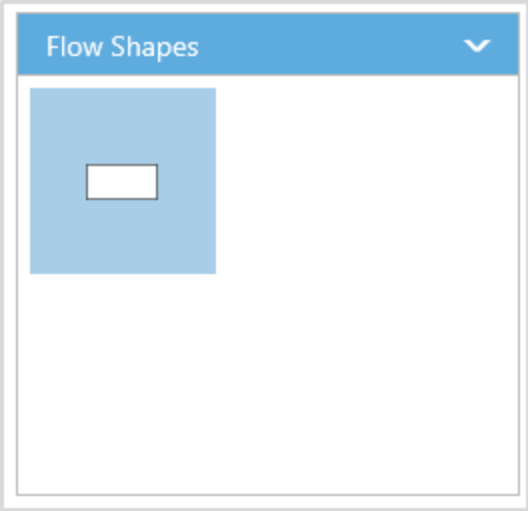
The EnableScale property of the paletteItem enables you to customize the size of the symbol regardless of the precedence. The following code example illustrates how to customize the symbol size.

JAVASCRIPT

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
//Creates a node
BasicShape rectangle = new BasicShape()
{
//Specifies node size
Width = 80,
Height = 40,
// Enables to fit the content into the specified palette item size
// When it is set as false, the element is rendered with actual node size
PaletteItem = new PaletteItem() { EnableScale = true }
};
Palette palette = new Palette("Basic Shapes");
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(rectangle);
//Specifies the default size to render symbols
symbolpalette.PaletteItemWidth = 50;
symbolpalette.PaletteItemHeight = 50;
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```



Copyright © 2001 - 2020 Syncfusion Inc.



Symbol Previews

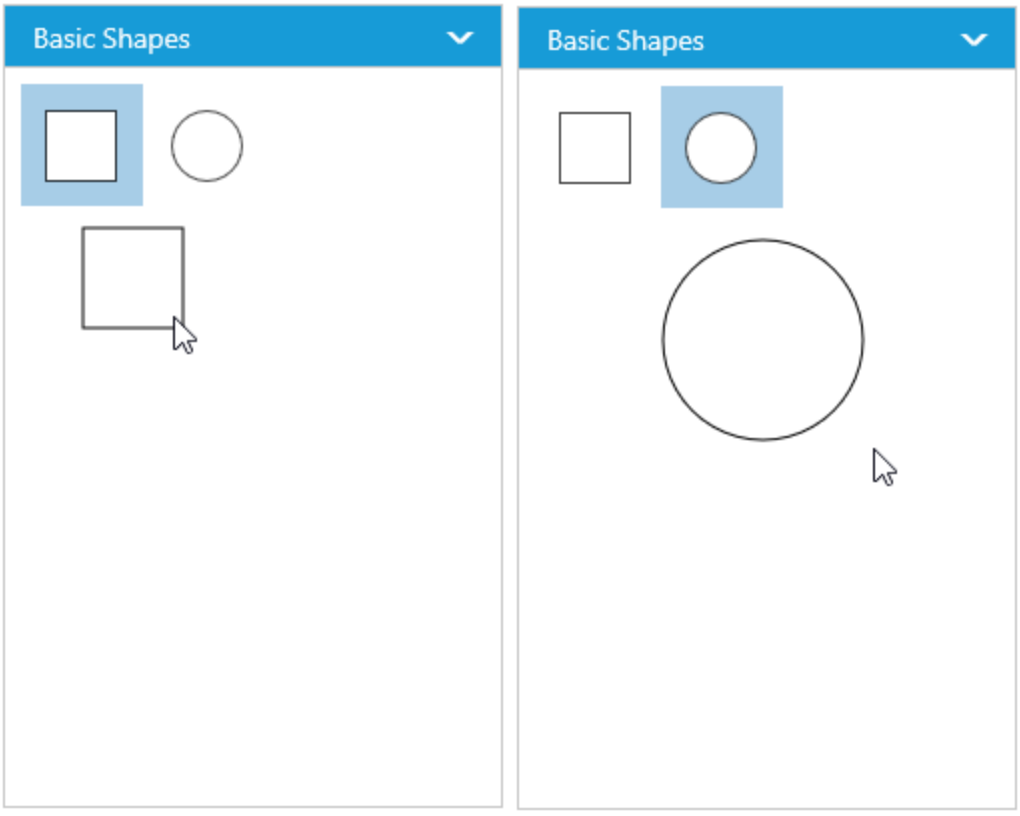
Image, simple snippet to customize the preview size

You can customize the preview size of the individual palette items. The PaletteItem property of node enables you to define the preview size of the symbol items. The following code example illustrates how to change the preview size of a palette item.

JAVASCRIPT

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
//Creates a node
BasicShape rectangle = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
//Specifies the individual palette item preview size
PaletteItem = new PaletteItem() { PreviewHeight = 50, PreviewWidth = 50 }
};
BasicShape ellipse = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
Shape = BasicShapes.Ellipse,
};
Palette palette = new Palette("Basic Shapes");
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(rectangle);
palette.Items.Add(ellipse);
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```

Copyright © 2001 - 2020 Syncfusion Inc.

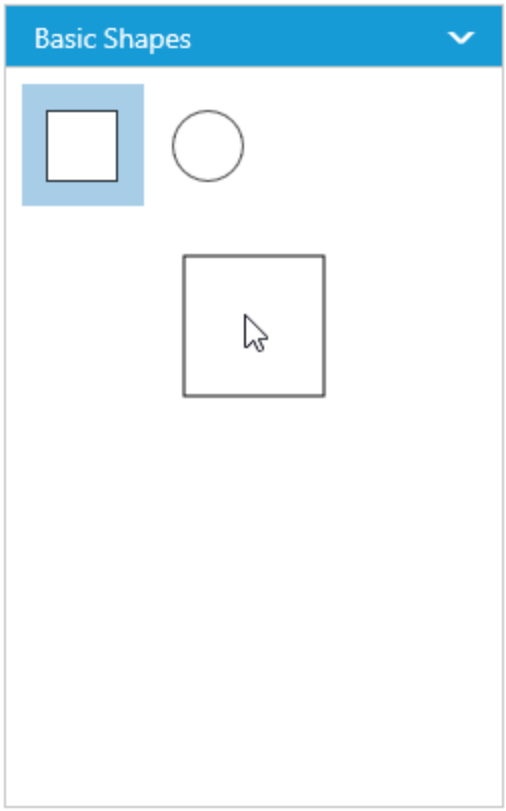


You can also customize the preview size of the all palette items. The PreviewWidth and PreviewHeight property of SymbolPalette enables you to define the preview size to all the symbol palette items. The following code example illustrates how to change the preview size of a symbol palette items.

JS

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
//Creates a node
BasicShape rectangle = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
};
Palette palette = new Palette("Basic Shapes");
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(rectangle);
symbolpalette.PreviewHeight = 100;
symbolpalette.PreviewWidth = 100;
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```

Copyright © 2001 - 2020 Syncfusion Inc.

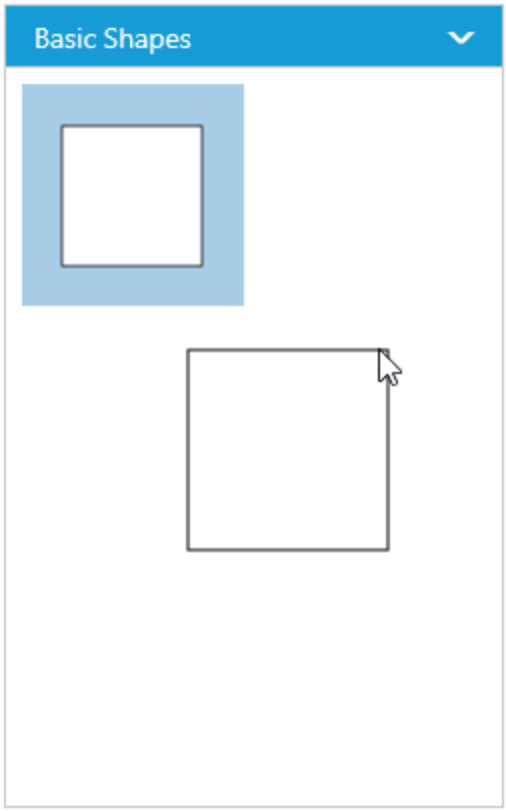


Symbol palette allows to sets the offset of the dragging helper relative to the mouse cursor.

JAVASCRIPT

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
//Creates a node
BasicShape rectangle = new BasicShape()
{
//Specifies node size
Width = 50,
Height = 50,
};
Palette palette = new Palette("Basic Shapes");
palette.Expanded = true;
//Adds the palette items to palette
palette.Items.Add(rectangle);
symbolpalette.PreviewOffset = new DiagramPoint() { X = 50, Y = 50 };
symbolpalette.Palettes = new Collection() { palette };
ViewData["SymbolPaletteModel"] = symbolpalette;
```

Copyright © 2001 - 2020 Syncfusion Inc.



Symbol preview size will be set based on the following precedence.

Precedence

| Palette Item | Preview Size |

|---|---|

| Width | PaletteItem.PreviewWidth > Model.PreviewWidth > Node.Width |

| Height | PaletteItem.PreviewHeight > Model.PreviewHeight > Node.Height |

- Symbol preview size will be set based on node.paletteItem's PreviewWidth and PreviewHeight property.
- If paletteItem's width and height property is not specified, symbol size will be set based on model's PreviewWidth and PreviewHeight property.
- If you don't specify above two, then symbol size will be rendered in the palette based on node's Width and Height property.

Show/hide the symbol Text

You can show/hide the symbol text by using the ShowPaletteItemText property of symbol palette.

JS

```
//Initializes the symbol palette
SymbolPaletteProperties symbolpalette = new SymbolPaletteProperties();
symbolpalette.Height = "550px";
symbolpalette.Width = "300px";
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Specifies whether palette item text should be visible or not
symbolpalette.ShowPaletteItemText = true;
ViewData["SymbolPaletteModel"] = symbolpalette;
```

To explore the properties of symbol palette, refer to Symbol Palette Properties.

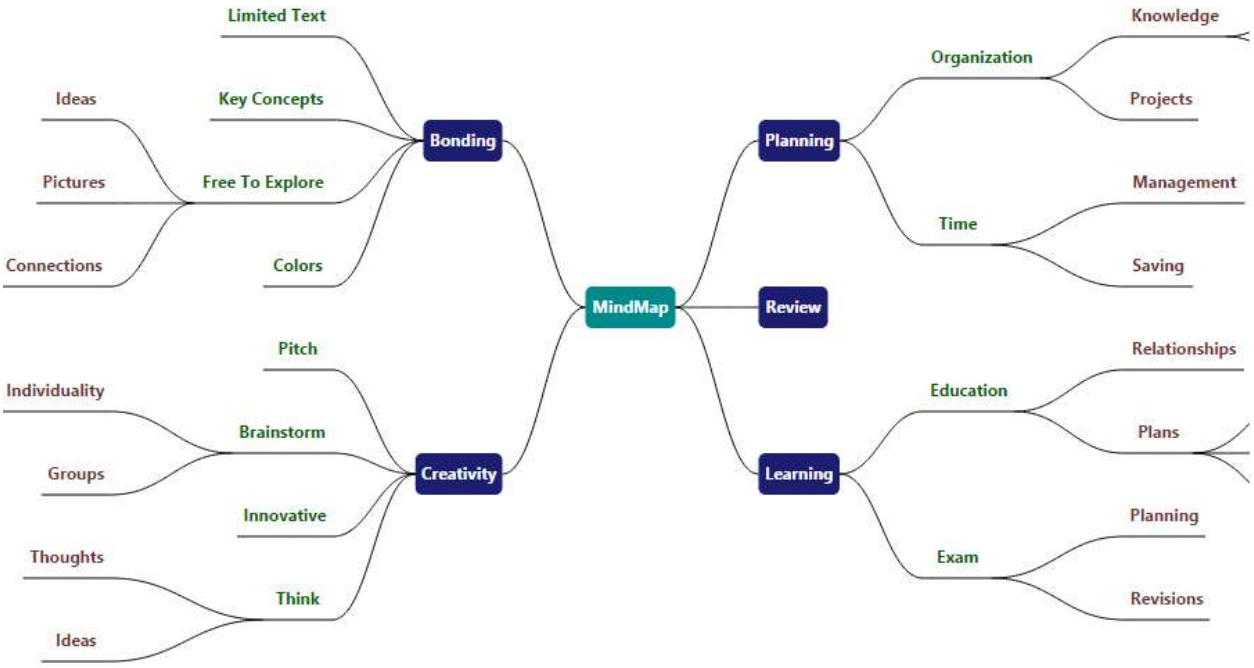
Exporting

Diagram provides support to export its content as image/svg files.

The client side method exportDiagram helps to export the Diagram. The following code illustrates how to export the Diagram as image.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//Exports the Diagram as an image of JPEG format
diagram.exportDiagram();
```



Exporting options

Diagram provides support to export the desired region of the Diagram to desired formats.

FileName

FileName is the name of the file to be downloaded. By default, the file name is set as "Diagram".

Format

Format is to specify the type/format of the exported file. By default, Diagram is exported as .jpg format. You can export Diagram to the following formats.

- JPG
- PNG
- BMP
- SVG

Copyright © 2001 - 2020 Syncfusion Inc.

For more information about the exportable formats, refer to File Formats.

Margin

Margin specifies the amount of space that has to be left around the Diagram.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var options = {
//Name of the file to be downloaded
fileName: "diagram",
//Margin to the exported file/data
margin: {
left: 30,
right: 30,
top: 30,
bottom: 30
}
};
diagram.exportDiagram(options);
```

Mode

Mode specifies whether the Diagram is to be exported as files or as data(ImageURL/SVG). The exporting options are as follows.

- Exports and downloads Diagram as image
- Exports Diagram as data of formats ImageURL/SVG

For more information about the exporting modes, refer to Exporting Modes.

The following code example illustrates how to export the Diagram as raw data.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var options = {
fileName: "diagram",
//Specifies whether to export as files/data
mode: "data"
};
diagram.exportDiagram(options);
```

Note: In IE-9, Diagrams cannot be exported as downloadable files. Instead, Diagram provides the exported data (DataURL/SVG) that enables to export it from application.

Region

You can export any particular region of the Diagram and the region is categorized as follows.

- Region that fits all nodes and connectors that are added to model
- Region that fits all pages (single or multiple pages based on page settings)

For more information about region, refer to Regions.

The following code example illustrates how to export the region occupied by the Diagram elements.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var options = {
//Specifies the exporting region
mode: "content"
};
diagram.exportDiagram(options);
```

The following code example illustrates how to export any specific region of the Diagram.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var options = {
fileName: "diagram",
//Specifies whether to export as files/data
mode: "download",
//Format of the exported file
format: "jpg",
// Defines the custom bounds that has to be exported
bounds: {
x: 1000,
y: 1000,
width: 500,
height: 500
},
};
diagram.exportDiagram(options);
```

Localization

- Localization is the process of providing controls in different cultures to help you set your own culture easily. Diagram provides localization support for Context Menu items.
- The Diagram model's Locale property is used to define the culture code.

The following code illustrates how to provide localization support for Context Menu items.

RAZOR

```
@*Initialize diagram control*@
<div>
@Html.EJ().Diagram("diagram", ViewData["diagramModel"] as
Syncfusion.JavaScript.DataVisualization.Models.DiagramProperties)
</div>
// Defines the context menu items with Spanish language
ej.datavisualization.Diagram.Locale["es-ES"] = {
cut: "Corte",
copy: "Copia",
paste: "Pasta",
undo: "Deshacer",
redo: "Rehacer",
selectAll: "Seleccionar todo",
grouping: "Agrupación",
group: "Grupo",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ungroup: "Desagrupar",
order: "Fin",
bringToFront: "Traer a delante",
moveForward: "Movimiento adelante",
sendToBack: "Enviar a espalda",
sendBackward: "Enviar hacia atrás"
};
```

C#

```
DiagramProperties model = new DiagramProperties();
//Adds nodes and connectors to model
model.Nodes.Add(new Node()
{
Name = "rectangle1",
OffsetY = 100,
Labels = new Collection() {
new Label(){Text="Rectangle1"}
}
});
model.Nodes.Add(new Node()
{
Name = "rectangle2",
OffsetY = 300,
Labels = new Collection() {
new Label(){ Text="Rectangle2" }
}
});
model.Connectors.Add(new Connector()
{
Name = "connector1",
SourceNode = "rectangle1",
TargetNode = "rectangle2"
});
//Defines the default properties
model.DefaultSettings.Node = new Node()
{
Width = 100,
Height = 100,
OffsetX = 100,
BorderColor = "#1BA0E2",
FillColor = "darkcyan",
Labels = new Collection() { new Label() { FontColor = "white" } }
};
//Sets the culture
model.Locale = "es-ES";
ViewData["diagramModel"] = model;
```


Note: You have to define the textual descriptions of the context menu items for your custom cultures.

Copyright © 2001 - 2020 Syncfusion Inc.

Tools

Drawing Tools

Drawing tool allows you to draw any kind of node/connector during runtime by clicking and dragging on the Diagram page.

Shapes

To draw a shape, you need to set the JSON of that shape to the drawType property of the Diagram model and you have to activate the drawing tool by using the tool property. The following code example illustrates how to draw a rectangle at run time.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//JSON to create a rectangle
diagram.model.drawType = {
type: ej.datavisualization.Diagram.Shapes.Basic,
shape: "rectangle",
//customize the appearance of the shape
fillColor: "#fcbc7c",
borderColor: "#f89b4c",
labels: [{
"text": "Rectangle"
}]
};
//To draw an object once, activate draw once
diagram.update({
tool: ej.datavisualization.Diagram.Tool.DrawOnce,
});
//To draw an object multiple times, activate continuous draw tool
diagram.update({
tool: ej.datavisualization.Diagram.Tool.ContinuesDraw,
});
```



The following code example illustrates how to draw a path.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
diagram.model.drawType = {
name: "Path",
fillColor: "#fbe172",
labels: [{
text: "Path"
}],
shape: "path",
```

```
pathData: "M13.560 67.524 L 21.941 41.731 L 0.000 25.790 L 27.120 25.790 L
35.501 0.000 L 43.882 25.790 L 71.000 25.790 L 49.061 41.731 L 57.441 67.524
L 35.501 51.583 z"
};
//To draw an object once, activate draw once
diagram.update({
tool: ej.datavisualization.Diagram.Tool.DrawOnce,
});
```



Connectors

To draw connectors, you have to set the JSON of the connector to drawType property. The drawing tool can be activated by using the tool property as shown. The following code example illustrates how to draw a straight line connector.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//JSON to create a straight line connector
diagram.model.drawType = {
type: "connector",
segments: [{
type: "straight"
}]
};
//To draw an object once, activate draw once
diagram.update({
tool: ej.datavisualization.Diagram.Tool.DrawOnce
});
```



Text

Diagram allows you to create a textNode as soon as you click on the Diagram page. The following code illustrates how to draw a text.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
//JSON to draw a text
diagram.model.drawType = { type: "text" };
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
diagram.update({
tool: ej.datavisualization.Diagram.Tool.DrawOnce
});
```

Once you activate the TextTool, you can also able to perform label editing of a node/connector.

Polyline Tool

Diagram allows to create the polyline segments with straight lines and angled vertices at the control points by clicking and moving the mouse at run time on the diagram page.

JAVASCRIPT

```
var diagram = $("#diagram").ejDiagram("instance");
// set the default segment type as polyline
diagram.model.drawType = {
type: "connector",
segments: [{ type: "polyline" }]
};
var tool = diagram.tool();
//To draw an object once, activate draw once
diagram.update({ tool: tool | ej.datavisualization.Diagram.Tool.DrawOnce })
```

Once you activate the polyline tool, you can also able to draw the polyline connector on diagram page.

Tool Selection

There are some functionalities that can be achieved by clicking and dragging on the Diagram surface. They are as follows.

- Draw selection rectangle - MultipleSelect tool
- Pan the Diagram - Zoom pan
- Draw nodes/connectors - ContinuousDraw / DrawOnce

As all the three behaviors are completely different, You can achieve only one behavior at a time based on the tool that you choose.

When more than one of those tools are applied, a tool is activated based on the precedence given in the following table.

| Precedence | Tools | Description |

|---|---|---|

| 1st | ContinuesDraw | Allows you to draw the nodes or connectors continuously. Once it is activated, you cannot perform any other interaction in the Diagram. |

| 2nd | DrawOnce | Allows you to draw single node or connector. Once you complete the drawOnce action, SingleSelect and MultipleSelect tools are automatically enabled. |

| 3rd | ZoomPan | Allows you to pan the Diagram. When you enable both the SingleSelect and ZoomPan tools, you can perform the basic interaction as the cursor hovers node/connector. Panning is enabled when cursor hovers the Diagram. |

Copyright © 2001 - 2020 Syncfusion Inc.

| 4th | MultipleSelect | Allows you to select multiple nodes and connectors. When you enable both the MultipleSelect and ZoomPan tools, cursor hovers the Diagram. When panning is enabled, you cannot select multiple nodes. |

| 5th | SingleSelect | Allows you to select individual nodes or connectors. |

| 6th | None | Disables all tools |

You can set the desired tool to the tool property of the Diagram model. The following code illustrates how to enable single/multiple tools.

C#

```
// To Enable Single Tool
DiagramProperties model = new DiagramProperties();
model.Tool = Tool.SingleSelect;
ViewData["Diagram"] = model;
// Enables multiple tools
DiagramProperties model = new DiagramProperties();
model.Tool = Tool.SingleSelect | Tool.ZoomPan;
ViewData["Diagram"] = model;
```

Tooltip

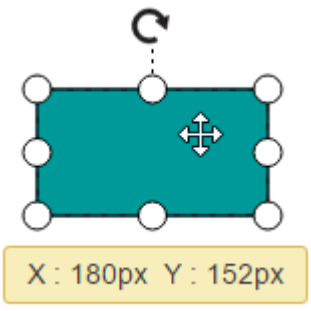
In Graphical User Interface (GUI), tooltip is a message that is displayed when mouse hovers over an element. Diagram provides tooltip support while dragging, resizing, rotating a node, and when mouse hovers any Diagram element.

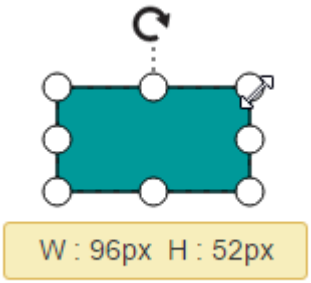
Default tooltip

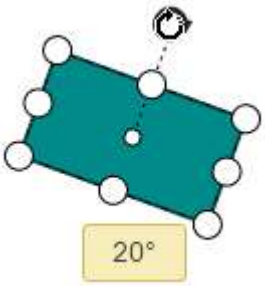
By default, Diagram displays a tooltip to provide the size, position, and angle related information while dragging, resizing, and rotation. The following images illustrate how the Diagram displays the node information during interaction.

| Drag | Resize | Rotate |

|---|---|---|







| | | |

Disable default tooltip

To disable the default tooltip, You need to set SelectedItems.Tooltip as null. The following code example illustrates how to disable default tooltip.

C#

```
// To Enables multiple tools
DiagramProperties Model = new DiagramProperties();
```

```
//Disable tooltip during interaction
Model.SelectedItems.Tooltip = null;
```

Common tooltip for all nodes and connectors

Diagram provides support to show tooltip when mouse hovers over any node/connector.

To show tooltip on mouse over, the Tooltip property of Diagram model needs to be set with the tooltip TemplateId and Alignment as shown in the following example.

HTML

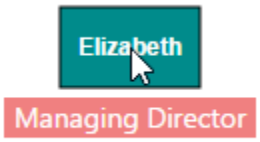
```
<!--Define tooltip template-->
<script type="text/x-jsrender" id="mouseovertooltip">
<div style="background-color: #F08080; color: white; white-space: nowrap;
height: 20px">
<span style="padding: 5px;"> {{:addInfo.Designation}} </span>
</div>
</script>
```

C#

```
public partial class DiagramController : Controller
{
public ActionResult DiagramFeatures()
{
//Initialize the Diagram Model
DiagramProperties Model = new DiagramProperties();
Model.Width = "100%";
Model.Height = "700px";
//Defines mouse over tooltip
Model.Tooltip = new Tooltip();
Model.Tooltip.TemplateId = "mouseovertooltip";
Model.Tooltip.Alignment.Horizontal = HorizontalAlignment.Center;
Model.Tooltip.Alignment.Vertical = VerticalAlignment.Center;
Label Label = new Label() { Text = "Elizabeth", FontColor = "white", Bold =
true };
Collection Labels = new Collection();
Labels.Add(Label);
//Defines nodes
Node Node = new Node()
{
Name = "elizabeth",
Width = 70,
Height = 40,
OffsetX = 100,
OffsetY = 100,
FillColor = "darkCyan",
Labels = Labels
};
Node.AddInfo = new Dictionary<string, object>();
Dictionary<string, object> AddInfo = new Dictionary<string, object>();
AddInfo.Add("Designation", "Managing Director");
Node.AddInfo = AddInfo;
Model.Nodes.Add(Node);
ViewData["diagramModel"] = Model;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
```



Disable tooltip at runtime

Tooltips on mouse over can be disabled by assigning tooltip property as null. The following code example illustrates how to disable the mouse over tooltip at runtime.

JS

```
$("#diagram").ejDiagram({
//Disables mouse over tooltip at runtime
tooltip: null
});
```

Tooltip for a specific node/connector

Tooltips can be customized for every node. Tooltip can be defined for individual node/connector by using the Tooltip property of that node/connector. In addition to that, you have to remove the **InheritTooltip** option from the Constraints of that node/connector. The following code example illustrates how to customize tooltips for individual elements.

C#

```
//Enables tooltip for any node/connector
DiagramProperties Model = new DiagramProperties();
Node Node = new Node();
Node.Name = "elizabeth";
//Remove InheritTooltip not to inherit the tooltip defined in model
Node.Constraints = NodeConstraints.Default &
~NodeConstraints.InheritTooltip;
//Defines mouse over tooltip for a node
Node.Tooltip.TemplateId = "nodetooltiptemplate";
Model.Nodes.Add(Node);
//Disables tooltip for any node/connector
Node.Tooltip = null;
Model.Nodes.Add(Node);
```

Tooltip alignments

Tooltip relative to object

Diagram provides support to show tooltip around the node/connector that is hovered by mouse. You can align the tooltip as you wish by using the Alignment and Margin properties of tooltip. The RelativeMode property of tooltip defines whether the tooltip has to be displayed around the object or at the mouse position. The following code example illustrates how to position the tooltip around object.

HTML

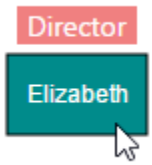
```
<!--Define tooltip template-->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/x-jsrender" id="mouseovertooltip">
<div style="background-color: #F08080; color: white; padding: 5px;">
<span style="padding: 5px;"> {{:addInfo.Designation}} </span>
</div>
</script>
```

C#

```
//Initialize the Diagram Model
DiagramProperties Model = new DiagramProperties();
Model.Width = "100%";
Model.Height = "700px";
//Defines mouse over tooltip
Model.Tooltip = new Tooltip();
Model.Tooltip.TemplateId = "mouseovertooltip";
Model.Tooltip.Alignment.Horizontal = HorizontalAlignment.Center;
Model.Tooltip.Alignment.Vertical = VerticalAlignment.Center;
Label Label = new Label() { Text = "Elizabeth", FontColor = "white", Bold =
true };
Collection Labels = new Collection();
Labels.Add(Label);
BasicShape Node = new BasicShape();
//Defines nodes
Node.Name = "elizabeth";
Node.Width = 70; Node.Height = 40;
Node.OffsetX = 100; Node.OffsetY = 100;
//Removes inherit tooltip from constraints
Node.Constraints = NodeConstraints.Default &
~NodeConstraints.InheritTooltip;
//Defines tooltip for a node
Node.Tooltip = new Tooltip();
//Defines template id
Node.Tooltip.TemplateId = "mouseovertooltip";
//Sets to show tooltip around the element
Node.Tooltip.RelativeMode = RelativeMode.Object;
//Sets the alignment properties
//Defines horizontal alignment around node
Node.Tooltip.Alignment.Horizontal = HorizontalAlignment.Center;
//Defines vertical alignment around node
Node.Tooltip.Alignment.Vertical = VerticalAlignment.Top;
Node.AddInfo = new Dictionary<string, object>();
Dictionary<string, object> AddInfo = new Dictionary<string, object>();
AddInfo.Add("Designation", "Managing Director");
Node.AddInfo = AddInfo;
Model.Nodes.Add(Node);
```



Copyright © 2001 - 2020 Syncfusion Inc.

Tooltip relative to mouse position

To display the tooltip at mouse position, you need to set "Mouse" option to the RelativeMode property of tooltip. The following code example illustrates how to show tooltip at mouse position.

C#

```
//Initialize the Diagram Model
DiagramProperties Model = new DiagramProperties();
Model.Width = "100%";
Model.Height = "700px";
//Defines mouse over tooltip
Model.Tooltip = new Tooltip();
Model.Tooltip.TemplateId = "mouseovertooltip";
Model.Tooltip.Alignment.Horizontal = HorizontalAlignment.Center;
Model.Tooltip.Alignment.Vertical = VerticalAlignment.Center;
Label Label = new Label() { Text = "Elizabeth", FontColor = "white", Bold =
true };
Collection Labels = new Collection();
Labels.Add(Label);
BasicShape Node = new BasicShape();
//Defines nodes
Node.Name = "elizabeth";
Node.Width = 70; Node.Height = 40;
Node.OffsetX = 100; Node.OffsetY = 100;
//Removes inherit tooltip from constraints
Node.Constraints = NodeConstraints.Default &
~NodeConstraints.InheritTooltip;
//Disables tooltip for a node
//Defines template id
Node.Tooltip.TemplateId = "mouseovertooltip";
//Sets to show tooltip around the element
Node.Tooltip.RelativeMode = RelativeMode.Mouse;
//Sets margin - absolute distance between mouse position and tooltip
Node.Tooltip.Margin.Top = 10;
Node.Tooltip.Margin.Left = 10;
Node.AddInfo = new Dictionary<string, object>();
Dictionary<string, object> AddInfo = new Dictionary<string, object>();
AddInfo.Add("Designation", "Managing Director");
Node.AddInfo = AddInfo;
Model.Nodes.Add(Node);
```



Constraints

Constraints are used to enable/disable certain behaviors of the diagram, node and connector. Constraints are provided as flagged enumerations, so that multiple behaviors can be enabled/disabled with bitwise operators (&, |, ~, <<, etc.).

To know more about bitwise operators, refer to Bitwise Operations

Copyright © 2001 - 2020 Syncfusion Inc.

DiagramConstraints

Diagram constraints allow to enable or disable the following behaviors.

- Page Editing
- Line Bridging
- Zoom and Pan
- Undo Redo

For more information about Diagram constraints, refer to Diagram Constraints.

Example

The following example illustrates how to disable page editing.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Set Diagram in read-only mode
model.Constraints = DiagramConstraints.Default &~
DiagramConstraints.PageEditable;
ViewData["diagramModel"] = model;
return View();
}
```

NodeConstraints

NodeConstraints allows to enable or disable the following behaviors of node.

- Selection
- Deletion
- Drag
- Resize
- Rotate
- Connect
- Drop shadow
- Drag label
- Define tooltip

For more information about node constraints, refer to Node Constraints

Example

The following code illustrates how to disable rotation.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Disables rotation
BasicShape task1 = new BasicShape() {
Constraints = NodeConstraints.Default &~ NodeConstraints.Rotate
};
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
model.Nodes.Add(task1);
ViewData["diagramModel"] = model;
return View();
}
```

ConnectorConstraints

ConnectorConstraints allow to enable or disable certain behaviors of Connectors. They are as follows.

- Selection
- Deletion
- Drag
- Segment editing
- Define tooltip
- Bridging
- Label dragging

For more information about connector constraints, refer to Connector Constraints

Example

The following code illustrates how to disable selection.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Disables Selection
Connector connector1 = new Connector() {
Constraints = ConnectorConstraints.Default &~ ConnectorConstraints.Select
};
model.Connectors.Add(connector1);
ViewData["diagramModel"] = model;
return View();
}
```

PortConstraints

You can enable or disable certain behaviors of Port. They are as follows.

- Connect

For more information about port constraints, refer to Port Constraints

Example

The following code illustrates how to disable creating connections with a port.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
BasicShape node1 = new BasicShape(){
Name = "Node1",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Defines ports for task2
Ports = new Collection() {
new Port(){ Constraints = PortConstraints.None },
}
};
model.Nodes.Add(node1);
ViewData["diagramModel"] = model;
return View();
}
```

SelectorConstraints

Selector visually represents the selected elements with certain editable thumbs. The visibility of the thumbs can be controlled with selector constraints. The part of selector is categorized as follows.

- Resizer
- Rotator
- User handles

For more information about Selector constraints, refer to Selector Constraints

Example

The following code illustrates how to hide rotator.

C#

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
//Hides rotator
model.SelectedItems = new SelectedItems(){
Constraints = SelectorConstraints.All &~ SelectorConstraints.Rotator
};
ViewData["diagramModel"] = model;
return View();
}
```

SnapConstraints

Snap Constraints control the visibility of gridlines and enable/disable snapping. Snap constraints allow to set the following behaviors.

- Show only horizontal or vertical gridlines
- Show both horizontal and vertical gridlines
- Snap to either horizontal or vertical gridlines
- Snap to both horizontal and vertical gridlines

For more information about snap constraints, refer to Snap Constraints

Example

The following code illustrates how to show only horizontal gridlines.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
model.SnapSettings = new SnapSettings()
{
SnapConstraints = SnapConstraints.ShowHorizontalLines
};
ViewData["diagramModel"] = model;
return View();
}
```

Inherit behaviors

Some of the behaviors can be defined through both the specific object(node/connector) and Diagram. When the behaviors are contradictorily defined through both, the actual behavior is set through inherit options.

The following code example illustrates how to inherit the line bridging behavior from the Diagram model.

RAZOR

```
public ActionResult Index()
{
DiagramProperties model = new DiagramProperties();
Connector connector1 = new Connector() {
Name = "Connector",
SourcePoint = new DiagramPoint(100f, 100f),
TargetPoint = new DiagramPoint(200f, 200f),
//Sets to inherit bridging from model
Constraints = ConnectorConstraints.Default |
ConnectorConstraints.InheritBridging
};
model.Connectors.Add(connector1);
//Enables line bridging for all connectors
model.Constraints = DiagramConstraints.Default |
DiagramConstraints.Bridging;
ViewData["diagramModel"] = model;
return View();
}
```

Bitwise Operations

Bitwise Operations are used to manipulate the flagged enumerations [enum]. In this section, Bitwise Operations are illustrated by using node Constraints. The same is applicable while working with Node Constraints, Connector Constraints, or Port Constraints.

Add Operation

You can **add** or **enable** multiple values at a time by using **Bitwise** '\|' (OR) **operator**.

C#

```
node.Constraints = NodeConstraints.Select | NodeConstraints.Rotate;
```

In the above example, you can do both the selection and rotation.

Copyright © 2001 - 2020 Syncfusion Inc.

Remove Operation

You can **remove** or **disable** values by using **Bitwise** '&~' (XOR) **operator**.

C#

```
node.Constraints = node.Constraints & ~(NodeConstraints.Rotate);
```

In the above example, **Rotation** is disabled but other constraints are enabled.

Check Operation

You can check any value by using **Bitwise** '&' (AND) **operator**.

C#

```
if ((node.Constraints & (NodeConstraints.Rotate)) ==
(NodeConstraints.Rotate))
```

In the above example, you can check whether the rotate constraints are enabled in a Node. When Node constraints have rotate constraints, the expression returns a rotate constraint.

Scroll Settings

The Diagram can be scrolled by using the vertical and horizontal scrollbars. In addition to the scrollbars, you can use mouse wheel to scroll the Diagram.

Diagram's ScrollSettings enables you to read the current scroll status, view port size, current zoom, and zoom factor. It also allows you to scroll the Diagram programmatically.

Get current scroll status

Scroll settings allows you to read the scroll status, view port size, and current zoom factor with a set of properties. To explore those properties, see Scroll Settings

Define scroll status

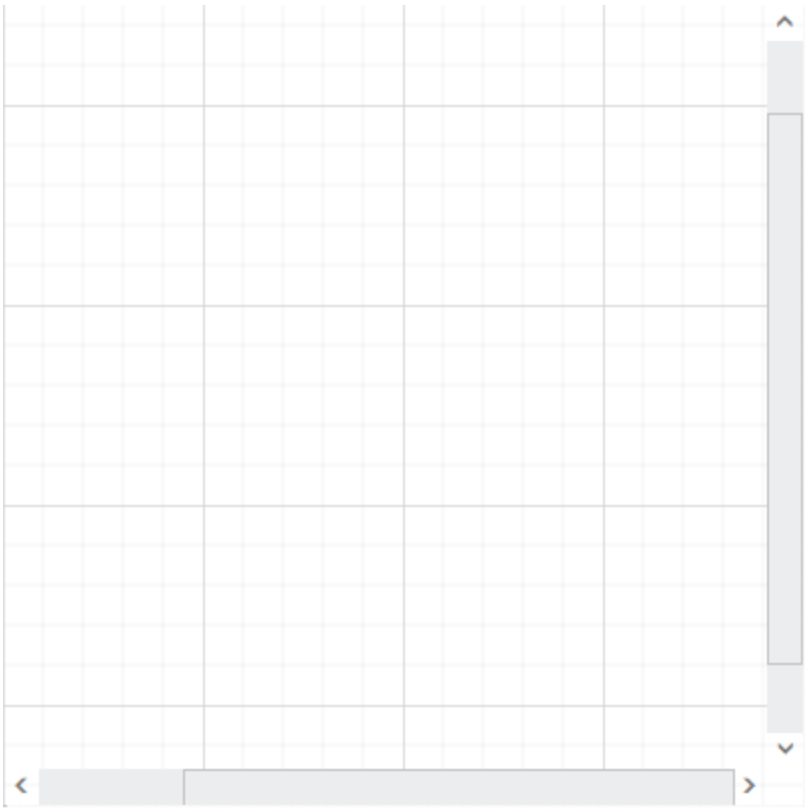
Diagram allows you to pan the Diagram before loading, so that any desired region of a large Diagram is made to view. You can programmatically pan the Diagram with the HorizontalOffset and VerticalOffset properties of scroll settings. The following code illustrates how to set pan the Diagram programmatically.

C#

```
// To Enables multiple tools
DiagramProperties Model = new DiagramProperties();
Model.Height = "400px";
Model.Width = "400px";
//Sets horizontal and vertical scroll offsets
Model.ScrollSettings.HorizontalOffset = 100;
Model.ScrollSettings.VerticalOffset = 50;
Model.ScrollSettings.ZoomFactor = 0.2;
//Sets page settings
Model.PageSettings.PageWidth = 500;
Model.PageSettings.PageHeight = 500;
```

In the example given below, the vertical scroll bar is scrolled down by 50px and horizontal scroll bar is scrolled to right by 100px.

Copyright © 2001 - 2020 Syncfusion Inc.



Update scroll status

You can programmatically change the scroll offsets at runtime by using the client side method update. The following code illustrates how to change the scroll offsets and zoom factor at runtime.

JS

```
var diagram = $("#diagram").ejDiagram("instance");
var scrollSettings = {
//Sets scroll status
horizontalOffset: 200,
verticalOffset: 200,
//Sets zoomFactor
zoomFactor: 0.5
}
//Updates scroll settings
diagram.update({ scrollSettings: scrollSettings });
```

AutoScroll

Autoscroll feature automatically scrolls the Diagram whenever the node or connector is moved beyond the boundary of the Diagram. So that, it is always visible during dragging, resizing, and multiple selection operations. Autoscroll is automatically triggered when any one of the following is done towards the edges of the Diagram.

- Node dragging, resizing
- Connection editing
- Rubber band selection
- Label dragging

Copyright © 2001 - 2020 Syncfusion Inc.

Autoscroll border

The Autoscroll border is used to specify the maximum distance between the object and Diagram edge to trigger Autoscroll. The default value is set as 15 for all sides (left, right, top, and bottom) and it can be changed by using the AutoScrollBorder property of page settings. The following code example illustrates how to set Autoscroll border.

C#

```
// To Enables multiple tools
DiagramProperties Model = new DiagramProperties();
Model.Height = "400px";
Model.Width = "400px";
// Specifies autoscroll border
Model.PageSettings.AutoScrollBorder.Left = 150;
Model.PageSettings.AutoScrollBorder.Top = 15;
Model.PageSettings.AutoScrollBorder.Right = 15;
Model.PageSettings.AutoScrollBorder.Bottom = 15;
```

Scroll limit

The scroll limit allows you to define the scrollable region of the Diagram. It includes the following options.

- Allows to scroll in all directions without any restriction.
- Allows to scroll within the Diagram content.
- Allows to scroll within the specified scrollable area.

ScrollLimit property of scroll settings helps to limit the scrolling. For the accepted values of the scrollLimit, refer to

Scroll Limit.

The following code example illustrates how to specify the scroll limit.

C#

```
// To Enables multiple tools
DiagramProperties Model = new DiagramProperties();
Model.Height = "400px";
Model.Width = "400px";
//Sets the scroll limit
Model.PageSettings.ScrollLimit = ScrollLimit.Infinity;
```

Scrollable Area

You can restrict scrolling beyond any particular rectangular area by using the ScrollableArea property of scroll settings. To restrict scrolling beyond any custom region, you have to set the ScrollLimit as "Limited". The following code example illustrates how to customize scrollable area.

C#

```
// To Enables multiple tools
DiagramProperties Model = new DiagramProperties();
Model.Height = "400px";
Model.Width = "400px";
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Sets scroll limit as limited
Model.PageSettings.ScrollLimit = ScrollLimit.Limited;
//Sets the limited scrollable area
Model.PageSettings.ScrollableArea.X = 0;
Model.PageSettings.ScrollableArea.Y = 0;
Model.PageSettings.ScrollableArea.Width = 500;
Model.PageSettings.ScrollableArea.Height = 500;
```

Context Menu

In graphical user interface (GUI), a context menu is a type of menu that appears when you perform right-click operation. Nested level of context menu items can be created.

Diagram provides some built-in context menu items and allows to define custom menu items.

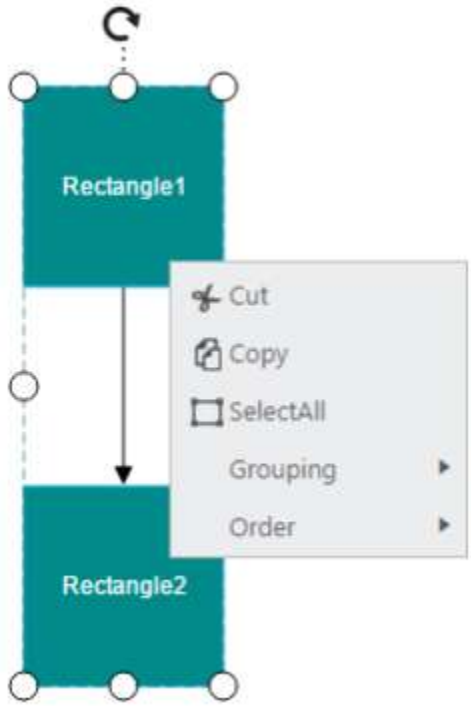
Default Context Menu

The EnableContextMenu property helps you to enable/disable the context menu. Diagram provides some default context menu items to ease the execution of some frequently used commands.

The following code illustrates how to enable the default context menu items.

C#

```
DiagramProperties model = new DiagramProperties();
//Disables the context menu
model.EnableContextMenu = false;
ViewData["Diagram"] = model;
```



Copyright © 2001 - 2020 Syncfusion Inc.

Customize Context Menu

Apart from the default context menu items, you can define some additional context menu items. Those additional items have to be defined and added to ContextMenu.Items. Sub menu items for context menu can set using ContextMenu.Items.SubItems

The following code example illustrate how to add custom context menu items.

C#

```
DiagramProperties model = new DiagramProperties();
model.Height = "600px";
model.Width = "600px";
//enable the context menu
model.EnableContextMenu = true;
model.ContextMenu = new DiagramContextMenu();
List<ContextMenuItem> SubItems = new List<ContextMenuItem>();
SubItems.Add(new ContextMenuItem() { Name = "zoomIn", Text = "ZoomIn", });
SubItems.Add(new ContextMenuItem() { Name = "zoomOut", Text = "ZoomOut", });
model.ContextMenu.Items.Add(new ContextMenuItem() { Name = "zoom", Text =
"Zoom", SubItems = SubItems });
ViewData["Diagram"] = model;
```

When you want to display only your custom context menu items, you can set true to the showCustomMenuItemsOnly property.

Icons of context menu items can be customized by overriding the default context menu item style.

The following code example illustrates how to customize the icon of context menu items.

CSS

```
#Zoom_image {
background-image: url("zoom.png");
}
#ZoomIn_image {
background-image: url("zoom-in.png");
}
#ZoomOut_image {
background-image: url("zoom-out.png");
}
```



Context Menu Events

You would be notified with events when you try to open the context menu items(contextMenuBeforeOpen) and when you click the menu items(contextMenuClick). The following code example illustrates how to define those events.

C#

```
DiagramProperties model = new DiagramProperties();
model.Height = "600px";
model.Width = "600px";
//enable the context menu
model.EnableContextMenu = true;
model.ContextMenu = new DiagramContextMenu();
List<ContextMenuItem> SubItems = new List<ContextMenuItem>();
SubItems.Add(new ContextMenuItem() { Name = "zoomIn", Text = "ZoomIn", });
SubItems.Add(new ContextMenuItem() { Name = "zoomOut", Text = "ZoomOut", });
model.ContextMenu.Items.Add(new ContextMenuItem() { Name = "zoom", Text =
"Zoom", SubItems = SubItems });
model.ContextMenuBeforeOpen = "contextMenuBeforeOpen";
model.ContextMenuClick = "contextMenuClick";
```

JS

```
function contextMenuBeforeOpen(args) {
//do your custom action here.
}
function contextMenuClick(args) {
switch (args.text) {
case "zoom":
//do your custom action here.
break;
}
}
```

How To

Bitwise Operations

Bitwise Operation

Bitwise Operations are used to manipulate the flagged enumerations [enum]. In this section, Bitwise Operations are illustrated by using Graph Constraints. The same is applicable while working with Node Constraints, Connector Constraints, or Port Constraints.

Add Operation

You can add or enable multiple values at a time by using Bitwise '|' (OR) operator.

C#

```
node.Constraints = NodeConstraints.Select | NodeConstraints.Rotate;
```

In the above example, you can do both selection and rotation.

Remove Operation

You can remove or disable values by using Bitwise '&~' (XOR) operator.

C#

```
node.Constraints = node.Constraints &~ NodeConstraints.Rotate;
```

In the above example, Rotation is disabled, but other constraints are enabled.

Copyright © 2001 - 2020 Syncfusion Inc.

Check Operation

You can check any value by using Bitwise '&' (AND) operator.

C#

```
if ((node.constraints & (NodeConstraints.Rotate)) ==
(NodeConstraints.Rotate));
```

In the above example, you can check whether the rotate constraints are enabled in a Node. When Node constraints have rotate constraints, the expression returns a rotate constraint.

Dialog

Overview

The Dialog control displays a Dialog window within a web page. The Dialog enables a message to be displayed, such as supplementary content like images and text, and an interactive content like forms.

Key Features

- Model dialog support: Displays the content in a modal dialog, disabling interaction with other items on the page
- AJAX Load: Load AJAX content in dialog content panel.
- Drag support: Drag the Dialog within the page.
- Customized dialog position: By default, the dialog is shown in the center of the container. If given a position, the dialog is displayed at the particular position.
- Keyboard navigation: Users can interact with the dialog by using the keyboard.

Getting Started

This section helps to understand the getting started of the Dialog widget with the step-by-step instructions.

Create a Dialog

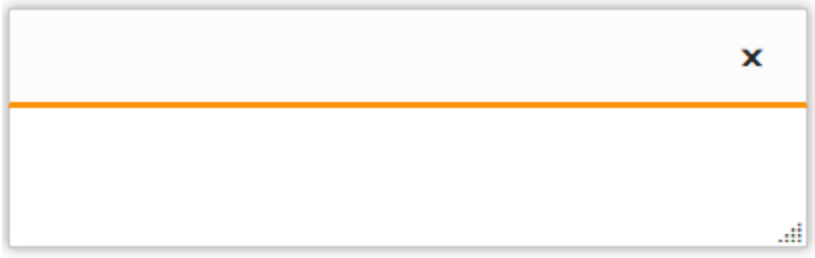
Create a MVC Project and add the necessary DLL and scripts with the help of the given MVC Getting Started documentation.

Add the following code snippet to the corresponding view page to render the Dialog.

RAZOR

```
@{
Html.EJ()
.Dialog("dialog")
.Render();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

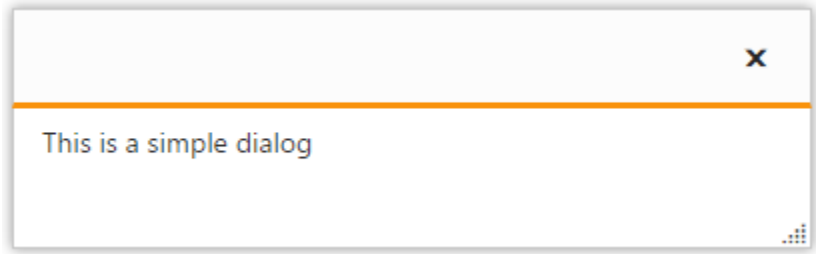


Set content

Add the contents for the dialog using ContentTemplate property as below.

RAZOR

```
@{
Html.EJ()
.Dialog("dialog")
.ContentTemplate(@<p>This is a simple dialog</p>)
.Render();
}
```

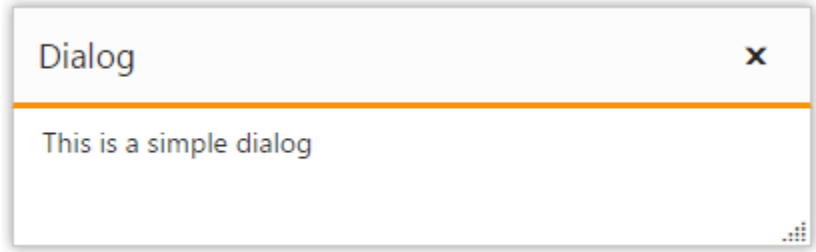


Set Title

The Dialog widget's title can be set as follows.

RAZOR

```
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ContentTemplate(@<p>This is a simple dialog</p>)
.Render();
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

Open Dialog dynamically

In most cases, the Dialog widgets are needed only in dynamic actions like showing some messages on clicking a button, to provide alert, etc. So the Dialog widget provides “open” and “close” methods to open/close the dialogs dynamically.

The Dialog widget can be hidden on initialize using ShowOnInit property which should be set to false.

Button.

Refer the below example. The dialog will be opened on clicking the Button widget.

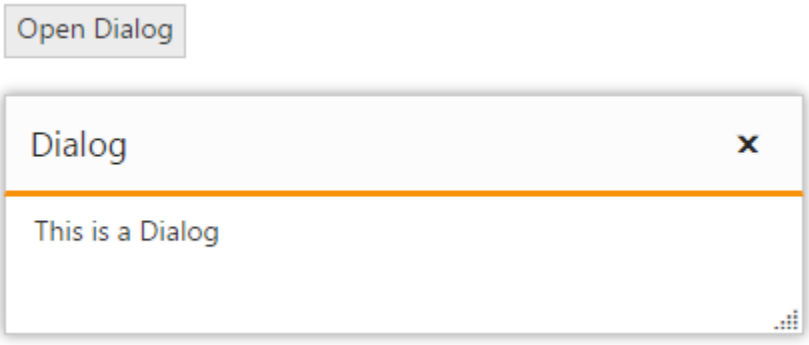
RAZOR

```
@*button widget*@
@{
Html.EJ()
.Button("button")
.Text("Open Dialog")
.Type(ButtonType.Button)
.ClientSideEvents(evt => evt.Click("openDialog"))
.Render();
}
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ShowOnInit(false)
.ContentTemplate(@<p>This is a Dialog</p>)
.Render();
}
```

Add the below script to the view page

JS

```
<script>
function openDialog() {
$("#dialog").ejDialog("open");
}
</script>
```



Copyright © 2001 - 2020 Syncfusion Inc.

Load content

By default, the content inside the ContentTemplate property element is considered as the content for the Dialog widget.

Also, we can render the Dialog widget content through the following ways.

1. Request through AJAX
2. Request iframe content
3. Request image content

This settings can be specified through ContentType property.

N> Create a view page (AjaxContent.cshtml) which contains the content of the dialog.

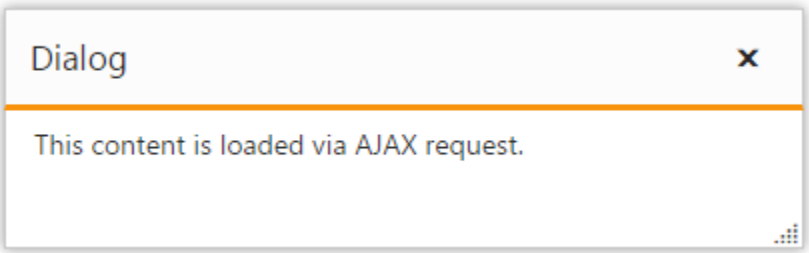
RAZOR

```
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ContentType("ajax")
.ContentUrl("AjaxContent")
.Render();
}
```

AjaxContent.cshtml

RAZOR

```
<div id="content"> This content is loaded via AJAX request. </div>
```



We can handle the AJAX request's success and failures through the events "ClientSideOnAjaxSuccess" and "ClientSideOnAjaxError" events respectively.

You can modify the previous example as below to handle the success and failure events.

RAZOR

```
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ContentType("ajax")
.ContentUrl("AjaxContent")
.ClientSideEvents(evt => evt.AjaxSuccess("onSuccess").AjaxError("onError"))
.Render();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

Add the below script to the view page

JS

```
<script>
function onSuccess(args) {
//handle success event
}
function onError(args) {
//handle success event
}
</script>
```

Note: The same way we can render the iframe and image content for the Dialog widget by specifying the ContentType as “iframe” and “image” respectively and also by specifying the proper location in the ContentUrl property. If you wish to dynamically change the dialog content, you can set the ContentUrl and the ContentType property through setModel on any action.

Action Buttons

The Dialog widget provides the following action buttons.

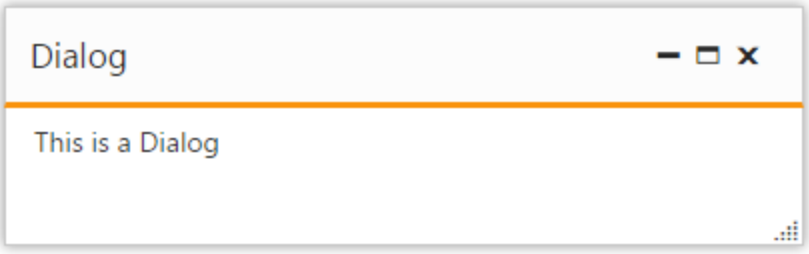
1. Close
2. Maximize
3. Minimize
4. Pin/Unpin
5. Collapse/Expand

You can display only the necessary buttons in the Dialog widget by configuring the ActionButtons property.

RAZOR

```
@{
List<string> actionButtons = new List<string>();
actionButtons.Add("close");
actionButtons.Add("maximize");
actionButtons.Add("minimize");
}
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ActionButtons(actionButtons)
.ContentTemplate(@<p>This is a Dialog</p>)
.Render();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing Action Buttons

We can customize the action buttons in dialog widget.

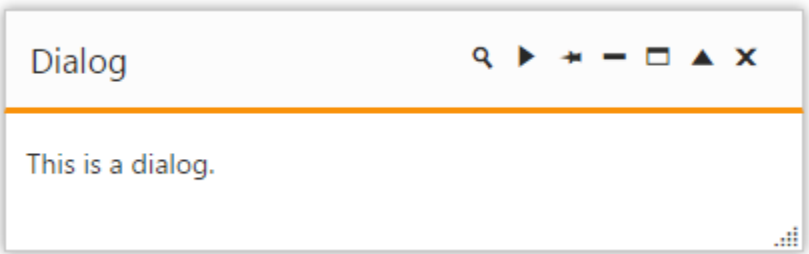
You can add new action button in the dialog widget by configuring the actionButtonClick event.

RAZOR

```
@{
List<string> actionButtons = new List<string>();
actionButtons.Add("close");
actionButtons.Add("maximize");
actionButtons.Add("minimize");
actionButtons.Add("collapsible");
actionButtons.Add("pin");
actionButtons.Add("mediaplay");
actionButtons.Add("search");
}
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ActionButtons(actionButtons)
.ClientSideEvents(event =>event.ActionButtonClick("playMedia"))
.ContentTemplate(@<p>This is a Dialog</p>)
.Render();
}
```

JAVASCRIPT

```
function playMedia(args)
{
console.log(args.buttonID);
}
```



Animation

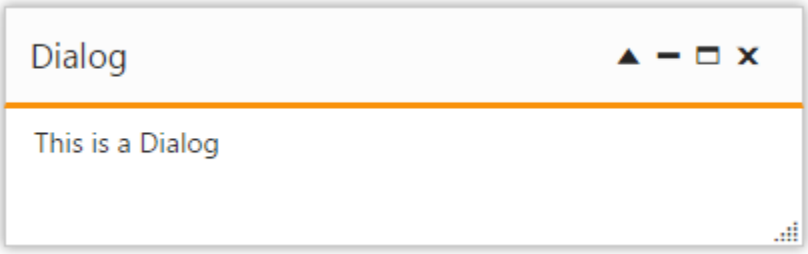
The Dialog widget can be animated while opening and closing the dialog.

Copyright © 2001 - 2020 Syncfusion Inc.

We can specify the effect and the duration for the animation. There are two types of effects (slide and fade). We can configure these settings separately for open and close dialog actions.

RAZOR

```
@{
List<string> actionButtons = new List<string>();
actionButtons.Add("close");
actionButtons.Add("maximize");
actionButtons.Add("minimize");
actionButtons.Add("collapsible");
}
@{
Html.EJ()
.Dialog("dialog")
.Title("Dialog")
.ActionButtons(actionButtons)
.Animation(animate => animate.Show(show =>
show.Duration(500).Effect("slide")).Hide(hide =>
hide.Duration(500).Effect("fade")))
.ContentTemplate(@<p>This is a Dialog</p>)
.Render();
}
```



Keyboard interaction

You can use Keyboard shortcut keys as an alternative for mouse actions to interact with the Dialog widget. The keyboard interaction will be enabled by default in the Dialog widget. It can be disabled by setting AllowKeyboardNavigation to false. Please refer the below table for details about short cut keys and its corresponding usage.

Shortcut Key	Usage
Up	Moves the dialog at upward direction.
Down	Moves the dialog at downward direction.
Left	Moves the dialog at left direction.
Right	Moves the dialog at right direction.
Ctrl Up	Reduces the dialog height.
Ctrl Down	Increases the dialog height.
Ctrl Left	Reduces the dialog width.

Copyright © 2001 - 2020 Syncfusion Inc.

Ctrl Right	Increases the dialog width.
Esc	Closes the dialog N> It will work only if closeOnEscape is set to true.

How To?

Create Multiple Dialogs

Essential Studio for ASP.NET library supports multiple Dialog widgets in the same web page with different contents and different functionalities.

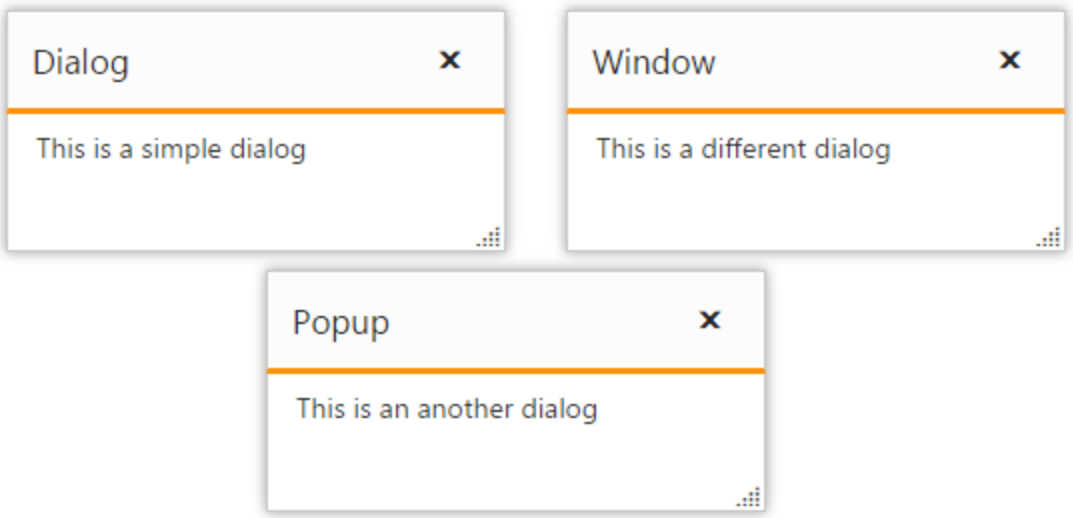
Initialize the Dialog widgets by configuring as below.

RAZOR

```
@{
Html.EJ()
.Dialog("firstDialog")
.Title("Dialog")
.Width(250)
.Position(position => position.XValue("20px").YValue("20px"))
.ContentTemplate(@<p>This is a simple dialog</p>)
.Render();
}
@{
Html.EJ()
.Dialog("secondDialog")
.Title("Window")
.Width(250)
.Position(position => position.XValue("300px").YValue("20px"))
.ContentTemplate(@<p>This is a different dialog</p>)
.Render();
}
@{
Html.EJ()
.Dialog("thirdDialog")
.Title("Popup")
.Width(250)
.Position(position => position.XValue("150px").YValue("150px"))
.ContentTemplate(@<p>This is an another dialog</p>)
.Render();
}
```

Note: If the position of the dialog is not set as above, all the three dialogs will be overlapped with each other.

Copyright © 2001 - 2020 Syncfusion Inc.



Create Nested Dialog

A Dialog widget can be nested within another Dialog widget.

Create a div element to render the child Dialog widget and use it as a content of parent Dialog widget.

RAZOR

```
@{
Html.EJ()
.Button("outerButton")
.Text("Open Dialog")
.Type(ButtonType.Button)
.ClientSideEvents(clientEvent => clientEvent.Click("openDialog"))
.Render();
}
@{
Html.EJ()
.Dialog("outerDialog")
.Title("Dialog")
.Width(500)
.Height(400)
.ShowOnInit(false)
.ContentTemplate(@<div>
@Html.EJ().Button("innerButton").Text("Open Nested
Dialog").ClientSideEvents(clientEvent =>
clientEvent.Click("openNestedDialog"))
</div>)
.Render();
}
@{
Html.EJ()
.Dialog("secondDialog")
.Title("Window")
.Width(400)
.Height(300)
.ShowOnInit(false)
.ContentTemplate(@<p>This is a nested dialog</p>)
.Render();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Add the below script to the view page

JS

```
<script>
function openDialog() {
$("#outerDialog").ejDialog("open");
}
function openNestedDialog() {
$("#secondDialog").ejDialog("open");
}
</script>
```

Create a Confirmation Dialog with Footer section

Essential JS library supports Alert Dialog widgets.

Using ShowFooter property to render Alert Dialog with Footers in Dialog widget.

Create a Dialog Widget with enabling Footer property.

RAZOR

```
<div class="control">
@Html.EJ().Button("btnOpen").Size(ButtonSize.Medium).Type(ButtonType.Button)
.Height("30").Width("150").ClientSideEvents(evt=>evt.Click("onOpen")).Text("
Click to open dialog")
@{Html.EJ().Dialog("basicDialog").Containment(".control").Title("Software
Installation Agreement").ContentTemplate(
@<div class="cnt">
Do you really leave the session?
</div>).Target(".control").ShowFooter(true).FooterTemplateId("sample").Clien
tSideEvents(evt => evt.Close("onDialogClose")).Render();
}
</div>
```

Add the following script to close and open the Dialog widget.

JAVASCRIPT

```
$("#btnOpen").hide();
function onDialogClose(args) {
$("#btnOpen").show();
}
function onOpen() {
$("#btnOpen").hide();
$("#basicDialog").ejDialog("open");
});
```

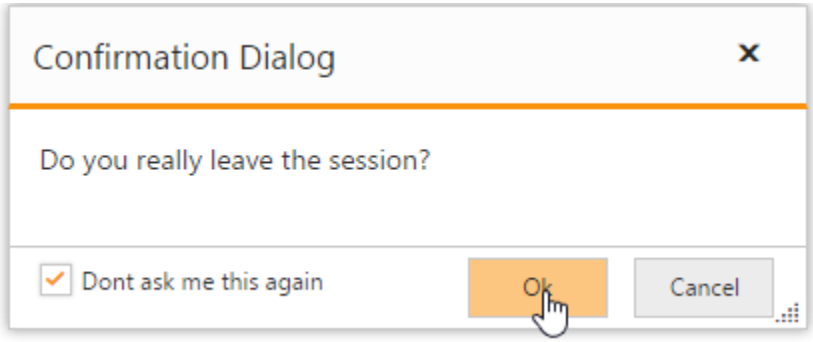
Initialize Footer in Dialog widgets by adding the script section as below.

JAVASCRIPT

```
<script id="sample" type="text/x-jsrender">
<div class="footerspan" style="float:right">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Button("btn1").Size(ButtonSize.Mini).Height("30").Width("70").Tex
t("Ok")
@Html.EJ().Button("btn2").Size(ButtonSize.Mini).Height("30").Width("70").Tex
t("Cancel")
</div>
<div class="condition" style="float:left; margin-left:15px">
@Html.EJ().CheckBox("check1").Text("Don't ask me this again")
</div>
</script>
```



Render Dialog from Code behind

Dialog can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Dialog properties in the controller.

C#

```
//Namespace to get the JavaScript (Dialog) component properties
using Syncfusion.JavaScript;
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class DialogController : Controller
{
public ActionResult DialogFeatures()
{
//Initializing the Dialog model
DialogProperties DialogObj = new DialogProperties();
//Initializing the content and other properties
DialogObj.ShowOnInit = true;
DialogObj.Title = "My Dialog";
DialogObj.ContentTemplate = new MvcTemplate<object>
{
RazorViewTemplate = (data) =>
{
return "This is Dialog content";
}
};
DialogObj.Width = "550";
DialogObj.IsResponsive = true;
//Passing Dialog properties using the ViewData
ViewData["DialogModel"] = DialogObj;
return View();
```

```
}
}
}
```

Binding the Dialog properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Dialog("basicDialog",
(Syncfusion.JavaScript.Models.DialogProperties)ViewData["DialogModel"]).Rend
er();   // render the Dialog in view
}
```

DigitalGauge

DigitalGauge

Digital Gauge for Essential Studio encompasses the visualization of the segmented display of both the 7-segment numeric values and the 14-segment alphanumeric characters. It can display the following five kinds of character types:

- Seven Segment

- Fourteen Segment

- Sixteen Segment

- EightCrossEightDotMatrix

- EightCrossEightSquareMatrix

Copyright © 2001 - 2020 Syncfusion Inc.



Digital Gauge

There are several other properties available in Digital Gauge which enable you to perform various customizations such as changing the space between the characters, segment width, segment length, spacing between the segments, and other customizations.

Key Features

- Character Types – Provide support for five different character types.
- Background Image - Provide support to add images as the background for the Digital Gauge.
- Custom Font - Provide support to enable the custom font for text within the Digital Gauge.

Getting Started

- The ASP.NET MVC **Digital Gauge** provides support to display the **DigitalGauge** within your web page and allows you to customize it. This section encompasses the details on how to configure DigitalGauge. Here you will learn how to provide data for a **DigitalGauge** and display the data in the required way.
- In addition, you will learn how to customize the default **DigitalGauge** appearance according to your requirements. As a result, you will get a DigitalGauge that shows it as Digital thermometer.
- You can use this **DigitalGauge** in advertisements, decorative purposes, displaying share details in share market, game score boards, token systems, etc.



Digital Thermometer

Create a Digital Gauge

ASP.NET MVC Digital Gauge widget basically renders with flexible APIs. You can easily create the Digital Gauge widget by using the following steps.

1. First create an MVC Project and add necessary Dll's and scripts with the help of the given MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page to render Digital Gauge.

ASPX-CS

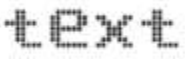
```
@(Html.EJ().DigitalGauge("digitalGauge"))
```

3. Add the following code example in the controller page.

CSHARP

```
public ActionResult Default()
{
return View();
}
```

Run the above code example and you will get a default Digital Gauge as follows.



Digital Gauge

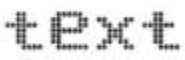
Set Height and Width values

Basic attributes of each canvas elements are height and width. You can set the height and width of the gauge.

ASPX-CS

```
@(Html.EJ().DigitalGauge("digitalGauge")
.Height(145)
.Width(260)
)
```

Run the above code example and you will see a default gauge with the specified height and width values.



Digital Gauge with Height and Width

Set Items Property

Items have different properties to customize the Digital Gauge.

Copyright © 2001 - 2020 Syncfusion Inc.

Add Segment and Character Properties

- In the Welcome Board, the text color must be attentive in nature. You can give some segment properties such as segment spacing, segment width, segment color, segment length and segment opacity.
- Character type is to define the Digital representation of the character. The five types of character representation available are,
    1. EightCrossEightDotMatrix
    2. SevenSegment
    3. FourteenSegment
    4. SixteenSegment
    5. EightCrossEightSquareMatrix.

ASPX-CS

```
@(Html.EJ().DigitalGauge("digitalGauge")
.Height(145)
.Width(260)
.Items(item=>
{
item.SegmentSettings(seg=>seg.Length(20).Width(2))
.Value("102")
. CharacterSettings(character=>{character.Spacing(12)
.Type(CharacterType.SevenSegment);}).Add();
}))
```

Run the above code example and you will see the following output.



Digital Gauge Segment Properties

Add Background Image

- Add a

  element to set the background for the **Digital Gauge**.

- Add a style tag in the View page to add the background image for the Digital Gauge.
- Add the required properties to show the background image such as position, margin, display, etc.,

ASPX-CS

```
<div id="frameDiv">
@(Html.EJ().DigitalGauge("digitalGauge")
.Height(145)
.Width(260)
.Items(item=>
{
```

```
item.SegmentSettings(seg=>seg.Length(20).Width(2))                .Value("102")
. CharacterSettings(character=>{character.Spacing(12)
.Type(CharacterType.SevenSegment);}).Add();
}))
</div>
<style>
#frameDiv {
align : center;
position : relative;
margin : 0px auto;
display :table;
background-image :url("script/frame.png");
background-repeat :no-repeat;
}
</style>
```

Run the above code example and you will see the following output.



Digital Gauge Background Image

Add Location

The Location property is used to position the digital letters inside the canvas element.

ASPX-CS

```
@(Html.EJ().DigitalGauge("DigitalGauge1").Height(145).Width(260).Items(item=
>
{
item.SegmentSettings(seg=>seg.Length(20).Width(2))
.Value("102").CharacterSettings(character=>{character.Spacing(12).Type(Chara
cterType.SevenSegment);})
.Position(loc=>loc.X(15).Y(40)).Add();
}))
```

Run the above code example and you will see the following output.



Digital Gauge with Segment Location

Add Items Collection

You can further add the Items Collection to display the temperature value like Digital Thermometer.

ASPX-CS

```
@(Html.EJ().DigitalGauge("digitalGauge").Height(145).Width(260).Items(item=>
{
item.SegmentSettings(seg=>seg.Length(20).Width(2))
.Value("102").CharacterSettings(character=>{character.Spacing(12).Type(Chara
cterType.SevenSegment);})
.Position(loc=>loc.X(15).Y(40)).Add();
item.SegmentSettings(seg=>seg.Length(5).Width(2))
.Value("0").CharacterSettings(character=>{character.Spacing(12).Type(Charact
erType.SevenSegment);})
.Position(loc => loc.X(85).Y(28)).Add();
item.SegmentSettings(seg=>seg.Length(20).Width(2))
.Value("F").CharacterSettings(character=>{character.Spacing(12).Type(Charact
erType.SevenSegment);})
.Position(loc => loc.X(170).Y(40)).Add();
item.SegmentSettings(seg=>seg.Length(9).Width(1).Color("#F5b43f"))
.Value("38").CharacterSettings(character=>{character.Spacing(12).Type(Charac
terType.SevenSegment);})
.Position(loc => loc.X(70).Y(90)).Add();
item.SegmentSettings(seg=>seg.Length(3).Width(1).Color("#F5b43f"))
.Value("0").CharacterSettings(character=>{character.Spacing(12).Type(Charact
erType.SevenSegment);})
.Position(loc => loc.X(90).Y(80)).Add();
item.SegmentSettings(seg=>seg.Length(9).Width(1).Color("#F5b43f"))
.Value("c").CharacterSettings(character=>{character.Spacing(12).Type(Charact
erType.SevenSegment);})
.Position(loc => loc.X(120).Y(90)).Add();
}))
```

Run the above code example and you will see the following output.



Digital Gauge with Item Collection

Basic Settings

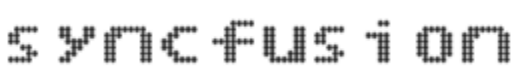
Height and Width Customization

The basic customization for any control is to set the dimension. Here dimension refers to two major attributes such as **height** and **width**. The height and width assigned in the control will render the canvas element in the given size. The code example to set height and width is as follow.

ASPX-CS

```
@(Html.EJ().DigitalGauge("DigitalGauge1")
@*For setting height of the canvas element. *@
.Height(200)
// For setting width of the canvas element.
.Width(500)
// For setting text
.Value("syncfusion"))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with height and width values

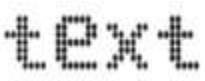
Responsive Layout

- For any display devices, the control will be rendered based on the space available in that device. For this purpose, **resizing** property is given to the **Digital Gauge** control. The **Digital Gauge** renders with a given value.
- When the browser resize the canvas element checks the dimension with its parent element. If there are any changes in parent dimension, **Gauge** control will changes the dimension based on its parent element change. This feature is enabled by using the property IsResponsive.

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For enabling responsive.
.IsResponsive(true))
```

Execute the above code examples to render the DigitalGauge as follows.



Digital Gauge control with Responsible layout

Themes

Themes give the good appearance to the control. There are two types of Themes available for **DigitalGauge** as follows

- flatlight
- flatdark

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting theme for digital gauge.
.Themes(Themes.FlatDark)
// For setting text
.Value("LOS ANGELS 40 KM"))
```

Execute the above code examples to render the DigitalGauge as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Digital Gauge control with flatdark theme

Setting Text for Digital Gauge

Digital Gauge is mainly used to display the text in the digital format. We can define the value for the text with the help of value property. It is string data type. The Code example for setting the text is as follows.

ASPX-CS

```
@* For Digital Gauge rendering  *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting text
.Value("GO SLOW"))
```

Execute the above code examples to render the DigitalGauge as follows.



Digital Gauge control with text as "GO SLOW"

Frames

Inner and Outer Customization

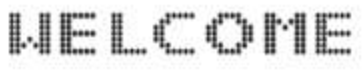
Frames are space that enclose the **Digital Gauge**. The inner width of the Frame is the distance between the canvas element and the frame. The outer width is the distance from the frame. The code example to set frame's innerWidth and outerWidth is as follow.

ASPX-CS

```
@*For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting text
.Value("WELCOME").Frame(fr=>fr
// For setting frame inner width
.InnerWidth(6)
// For setting outer width
.OuterWidth(10)))
```

Execute the above code examples to render the **DigitalGauge** as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Digital Gauge control with frame inner and outer width

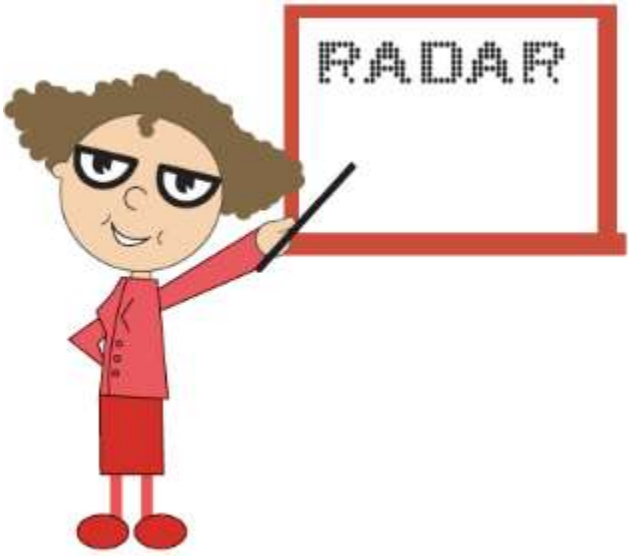
Setting Background Image

For a better appearance, you can set the background image for the **Digital Gauge** using the property backgroundImageUrl.

ASPX-CS

```
@*For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting text
.Frame(fr=>fr
// For setting frame background image
.BackgroundImageUrl("../Content/images/gauge/board3.jpg"))
.Items(it=> { it.Value("RADAR")
// For setting segment color
.SegmentSettings(cs=> cs.Color("Black"))
.Position(position=>position.X(80).Y(10)).Add();}))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with frame background image

Digital Elements

Text Customization

- The attribute value refers the text displayed in the **Digital Gauge**. This text is applicable only for that item instead of all items. Text color is changed by using the property textColor.
- It is possible to align the text inside the **Digital Gauge** control by using the property textAlign. Two possible values for text align are as follows

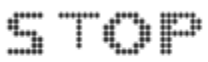
Copyright © 2001 - 2020 Syncfusion Inc.

1. left
2. right

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1").Items(it=>{it
// For Setting the text value
.Value("STOP")
// For aligning the text
.TextAlign(TextAlign.Right).Add();
}))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with text customization

Segment Settings

Appearance

- **Digital Gauge** consists of several digital segments. Segment is customized with some properties. Color of the segment is set by using color property. Color is either given as string or hexadecimal value.
- You can add gradient effects to the segments with the help of gradient attribute. The opacity of the segment is also adjustable. The space between two segments are adjusted with spacing property.

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
.Width(800)
.Value("GO AHEAD")
.Items(it=>
{ it.SegmentSettings(ss =>
// For setting segment opacity
ss.Opacity(0.1)
// For setting segment spacing
.Spacing(4)
// For setting segment color
.Color("Green")).Add();   }))
```

Execute the above code examples to render the **DigitalGauge** as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Digital Gauge control with segment settings

Dimension Modification

- **Digital Gauge** consists of several digital segments. Segment is customized with some properties. Color of the segment is set by using color property. Color is either given as string or hexadecimal value.
- You can add gradient effects to the segments with the help of gradient attribute. The opacity of the segment is also adjustable. The space between two segments are adjusted with spacing property.

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
.Width(800)
.Value("WELCOME")
.Items(it=>
{ it.SegmentSettings(ss =>
// For setting segment opacity
ss.Length(3)
// For setting segment color
.Width(3)).Add();
}))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with segment dimension

Character Settings

Appearance

The opacity of the character is adjustable with the help of opacity property. The space between two characters are adjusted with spacing property as like in the segment settings.

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
.Width(800)
.Value("Syncfusion")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Items(it=>
{ it.CharacterSettings(cs => cs
// For setting character opacity
.Opacity(0.3)
// For setting character spacing
.Spacing(3)).Add();
}))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with character setting

Count and Type

- The number of text to be displayed can be limited by the attribute called count. In **Digital Gauge** five different types of characters are supported. They are as follows,
    1. EightCrossEightDotMatrix
    2. SevenSegment
    3. FourteenSegment
    4. SixteenSegment
    5. EightCrossEightSquareMatrix.

ASPX-CS

```
@(Html.EJ().DigitalGauge("DigitalGauge1")
.Width(800)
.Value("123456789")
.Items(it=>
{ it.CharacterSettings(cs => cs
// For setting character count
.Count(10)
// For setting character spacing
.Spacing(10)
// For setting character type
.Type(CharacterType.SevenSegment))
.SegmentSettings(ss=>
// For setting segment length
ss.Length(8)
// For setting segment width
.Width(1)).Add();
}))
```

Execute the above code examples to render the **DigitalGauge** as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Digital Gauge control with character type as seven segment

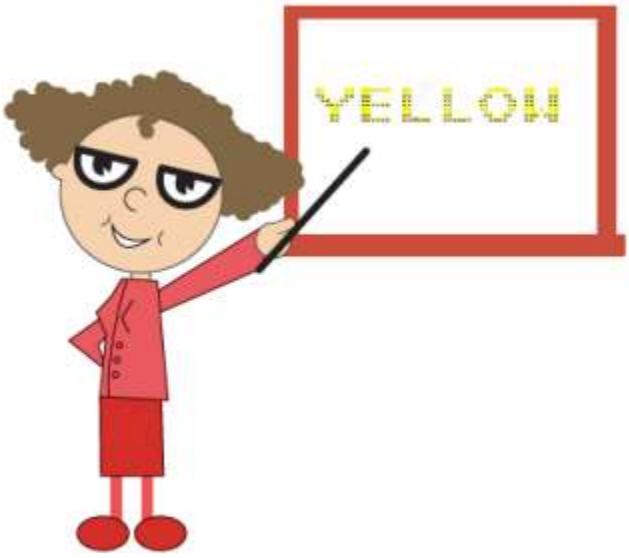
Text Positioning

The text in the **DigitalGauge** is positioned with position object. This object contains two attributes such as x and y. The x variable positions the text in the horizontal axis and the y variable positions the text in the vertical axis.

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting Width and Height
.Width(800).Height(300)
.Frame(fr=>fr
// For setting frame background image
.BackgroundImageUrl("../Content/images/gauge/Board1.jpg"))
.Items(it=> { it.Value("YELLOW")
// For setting segment color
.SegmentSettings(cs=>cs.Color("Yellow"))
.Position(position=>position
// For setting horizontal position
.X(80)
// For setting Vertical position
.Y(10)).Add();}))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with position text based on the background image

Copyright © 2001 - 2020 Syncfusion Inc.

Shadow Effects

- You can add the shadow effects for text using following properties.
- You can enable/disable the blurring effect for the shadows of the text using shadow blur property.
- You can specify the color of the text shadow using shadow color property.
- You can set the x-offset value for the shadow of the text, indicating the location where it needs to be displayed.
- You can set the y-offset value for the shadow of the text, indicating the location where it needs to be displayed.

ASPX-CS

```
@* For Digital Gauge rendering*@
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting Width and Height
.Width(800)
.Items(it=> { it.Value("WELCOME")
// For setting Shadow blur
.ShadowBlur(20)
// For setting Shadow color  .ShadowColor("Yellow")
// For setting Shadow horizontal offset     .ShadowOffsetX(15)
// For setting Shadow vertical offset .ShadowOffsetY(15)
// For setting segment length and width
.SegmentSettings(cs=>
cs.Length(3) .Width(3)).Add();}))
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with shadow option

Font Customization

You can customize the **font** of the text as per your requirement. To customize the font, you have to set enableCustomFont. Following font customization options are available.

Font-family- used to set the font-family of the text.

Font-style- used to set the font-style of the text.

Font-size- used to set the font-size of the text.

ASPX-CS

```
@* For Digital Gauge rendering *@
@(Html.EJ().DigitalGauge("DigitalGauge1")
.Width(800)
.Value("Syncfusion")
.Items(it=>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ it.CharacterSettings(cs => cs
// For setting character opacity
.EnableCustomFont(true)
// For setting character spacing
.Font(ft=>ft.FontFamily("Arial").FontStyle(FontStyle.Bold).Size("18px"))
).Add();
}))
```

Multiple Items

The text in the **Digital Gauge** is positioned with position object. This object contains two attributes such as x and y. The x variable positions the text in the horizontal axis and y variable positions the text in the vertical axis.

ASPX-CS

```
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting Width and Height
.Width(1300).Height(300)
.Frame(fr=>fr.BackgroundImageUrl("../Content/images/gauge/Board1.jpg"))
.Items(it =>
{
// For setting Item1
it.Value("YELLOW").SegmentSettings(cs =>
cs.Color("Yellow")).Position(position=>position.X(80).Y(0)).Add();
// For setting Item2
it.Value("RED").SegmentSettings(cs =>
cs.Color("Red")).Position(position => position.X(80).Y(20)).Add();
// For setting Item3
it.Value("GREEN").SegmentSettings(cs =>
cs.Color("Green")).Position(pos => pos.X(80).Y(40)).Add();
}))
```

Execute the above code example to render the **DigitalGauge** as follows.



Digital Gauge control with multiple items

Copyright © 2001 - 2020 Syncfusion Inc.

Exporting the Digital Gauge

Digital Gauge has an exporting feature where **Gauge** control is converted into image format and then exported to client-side. The method API exportImage exports the Digital Gauge. It has two arguments such as **filename** and **file format**. For exporting, you can refer the following code example.

ASPX-CS

```
// For Digital Gauge rendering
@(Html.EJ().DigitalGauge("DigitalGauge1")
// For setting text
.Value("Syncfusion"))
<script>
$("#DigitalGauge1").ejDigitalGauge("exportImage", "Digital", "PNG");
<script/>
```

Execute the above code examples to render the **DigitalGauge** as follows.



Digital Gauge control with Export functionality

MVVM

AngularJS

Digital Gauge contains AngularJS support. It is possible to add object as well as array object in the **Digital Gauge**. The two way binding support is given to the value for displaying the text.

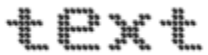
Rendering the Digital Gauge

HTML

```
**ej-DigitalGauge** is the control tag, where ej is tag prefix and
**DigitalGauge** is the control name.**Digital Gauge** is rendered with the
following code example.
<div ng-app="syncApp">
<div ng-controller="DigitalGauge">
<ej-DigitalGauge id="digitalCore" e-height="500" e-load="loadGaugeTheme">
</ej-DigitalGauge>
</div>
</div>
<script type="text/javascript">
<!—binding the value to the scope variables in application controller-->
```

```
angular.module('syncApp', ['ejangular'])
.controller('DigitalGauge', function ($scope) {
$scope.value = "text";
});
</script>
```

Execute the above code to render the following output.



Default Digital Gauge in AngularJS

Adding the Digital Gauge Items

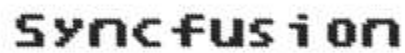
Digital Gauge is rendered with the following code example. You can extend the Object in the array collection such as, position, characterSetting, segmentSetting, etc. with hyphen in the same tag.

Example: e-position-x.

HTML

```
<!--To Render the Digital gauge-->
<ej-DigitalGauge id="digitalCore">
<!--Adding Item collection to the digital gauge-->
<e-items>
<e-item e-segmentSettings-width="1" e-segmentSettings-spacing="0"
e-value="Syncfusion" e-characterSetting-opacity="0.8"
e-position-x="52" e-position-y="52">
</e-item>
</e-items>
</ej-DigitalGauge>
```

Finally while running the above codes, the following output will be rendered.



Digital Gauge Rendered in AngularJS

Two Way Binding

Digital Gauge supports the two way biding for the property value as mentioned earlier. Following code example explains how to achieve the two way binding to the **Digital Gauge**.

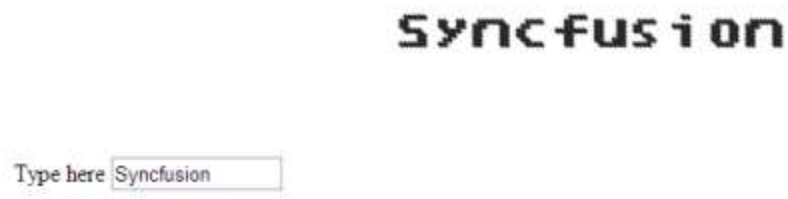
HTML

```
<div ng-app="syncApp">
<div ng-controller="DigitalGauge">
<ej-DigitalGauge id="digitalCore" e-height="200" e-load="loadGaugeTheme">
<e-items>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<e-item e-segmentSettings-width="1" e-segmentSettings-spacing="0"
e-characterSetting-opacity="0.8" e-position-x="52"
e-value="value" e-position-y="52">
</e-item>
</e-items>
</ej-DigitalGauge>
Type here <input type="text" id="txtValue" ng-model="nvalue"
Style="width:110px"/>
</div>
</div>
<script type="text/javascript">
<!—binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('DigitalGauge', function ($scope) {
$scope.value = "Syncfusion";
});
</script>
```

Execute the above code to render the following output.



Digital Gauge with two way binding in AngularJS

KnockoutJS Binding

- KnockoutJS support allows you to bind the HTML elements against any of the available data models.Two types of KnockoutJS binding is supported as of AngularJS,
    1. one-way binding
    2. two-way binding
- **One way binding** refers to the process of applying observable values to all the available properties of the **Digital Gauge** control, but the changes made in it does not reflect and trigger in turn to the observable collection. This kind of binding applies to all the properties of the Digital Gauge control.
- **Two-way binding** supports both the processes – it applies the observable values to the **Digital Gauge** properties as well as the changes made in it is also reflected back and triggered within the observable collections. Only **Value** of the schedule properties support two-way binding.

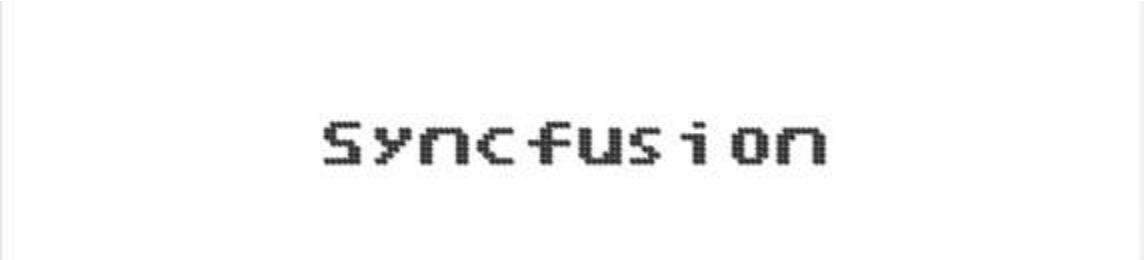
ASPX-CS

```
<div id="digitalCore"
data-bind="ejDigitalGauge:
{
value: sampleValue,
}">
</div>
<script type="text/javascript">
$(function ()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
window.viewModel =
{
sampleValue: ko.observable("syncfusion"),
};
$(function ()
{
ko.applyBindings(viewModel);
});
});
</script>
```

Execute the above code to render the following output.



Digital Gauge with KnockoutJS binding

Methods

destroy()

To destroy the digital gauge

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.destroy();
```

exportImage(fileName, fileType)

To export Digital Gauge as Image

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.exportImage();
```

getPosition(itemIndex)

Gets the location of an item that is displayed on the gauge.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.getPosition();
```

getValue(itemIndex)

ClientSideMethod getValue Gets the value of an item that is displayed on the gauge

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.getValue();
```

refresh()

Refresh the digital gauge widget

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.refresh();
```

setPosition(itemIndex, value)

ClientSideMethod Set Position Sets the location of an item to be displayed in the gauge

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.setPosition();
```

setValue(itemIndex, value)

ClientSideMethod SetValue Sets the value of an item to be displayed in the gauge.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1"))
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
var gauge = $("#Gauge1").data("ejDigitalGauge");
gauge.setValue();
```

Events

Init

Triggers when the gauge is initialized.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.Init("init")
)
```

JS

```
function init(){
// Do Something
}
```

ItemRendering

Triggers when the gauge item rendering.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.ItemRendering("itemRendering")
)
```

JS

```
function itemRendering(){
// Do Something
}
```

Load

Triggers when the gauge is start to load.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.Load("load")
)
```

JS

```
function load(){
// Do Something
}
```

RenderComplete

Triggers when the gauge render is completed.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.RenderComplete("renderComplete")
)
```

JS

```
function renderComplete(){
// Do Something
}
```

Click

Click event triggers on clicking the gauges.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.Click("click")
)
```

JS

```
function click(){
// Do Something
}
```

DoubleClick

The DoubleClick event is triggered when double clicking the gauges.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.DoubleClick("doubleClick")
)
```

JS

```
function doubleClick(){
// Do Something
}
```

RightClick

The RightClick event is triggered when right clicking the gauges.

ASPX-CS

```
@(Html.EJ().DigitalGauge("Gauge1")
.RightClick("rightClick")
)
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
function rightClick(){
// Do Something
}
```

DropDownList

DropDownList

The DropDownList control displays a single column list of items which enables you to select single or multiple items from the list. By default, there will be no selection made in the control and the user have to navigate through the items using mouse or keyboard actions to select an item.

Key Features

- **DataSources** - Binds the DropDownList control with a list, database table or remote data service. Supports various client-side and remote data sources such as JSON, RESTful services, OData services, WCF services and much more.
- **Sorting** - Sorts in both ascending and descending orders.
- **Grouping** - Categorizes the list Items.
- **Searching** - Provides both incremental and Filter Search.
- **Virtual scrolling** - Fetches the data from remote on demand.
- **PopupList Resize** - Resizes the popup list in the needed dimensions dynamically.
- **Template** - Designs own layout of list items.
- **Cascading** - Cascades the data sources with multiple DropDownList.
- **State persistence** - Persistent state of properties on page refresh.
- **Responsive** - Suits responsive layouts.
- **Validation** - Built-in jQuery validation.
- **Localization** - Supports localization to different cultures.
- **Selection** - Supports single or multi selection with the DropDownList and display the text in two modes, delimiter and visual mode.
- **Accessibility** - Supports keyboard and ARIA accessibility.

Getting Started

Creating your first DropDownList in MVC application

1. Create an MVC Project and add necessary assemblies, scripts and CSS files given in MVC-Getting Started Documentation.
2. Add DropDownList control using the helper from EJ namespace.

HTML

```
@Html.EJ().DropDownList("DropDownList1")
```

3. Execute the code and get a empty DropDownList control as below



Copyright © 2001 - 2020 Syncfusion Inc.

Populating data

The DropDownList can be bounded to any local list data and remote data services. You can use DataManager component to serve data from the data services based on the query provided.You can bind data locally from controller also. To render the DropDownList items, map the DropDownListFields with corresponding Fields

HTML

```
@Html.EJ().DropDownList("customersList").Datasource((IEnumerable<Customers>)
ViewBag.datasource).DropDownListFields(df =>
df.ID("id").Text("text").Value("text"))
```

C#

```
List<Customers> customer = new List<Customers>();
public ActionResult Index()
{
customer.Add(new Customers { id = "1", text = "ALFKI" });
customer.Add(new Customers { id = "2", text = "ANATR" });
customer.Add(new Customers { id = "3", text = "ANTON" });
customer.Add(new Customers { id = "4", text = "AROUT" });
customer.Add(new Customers { id = "5", text = "BERGS" });
customer.Add(new Customers { id = "6", text = "BLAUS" });
ViewBag.datasource = customer;
return View();
}
public class Customers
{
public string id { get; set; }
public string text { get; set; }
}
```

Execute the code and to get a DropDownList control with data bound from controller



Setting Dimensions

DropDownList dimensions can be set using Width and Height Properties.

HTML

```
@Html.EJ().DropDownList("customersList").Datasource((IEnumerable<Customers>)
ViewBag.datasource).DropDownListFields(df =>
df.ID("id").Text("text").Value("text")).Width("300px").Height("50px")
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
List<Customers> customer = new List<Customers>();
public ActionResult Index()
{
customer.Add(new Customers { id = "1", text = "ALFKI" });
customer.Add(new Customers { id = "2", text = "ANATR" });
customer.Add(new Customers { id = "3", text = "ANTON" });
customer.Add(new Customers { id = "4", text = "AROUT" });
customer.Add(new Customers { id = "5", text = "BERGS" });
customer.Add(new Customers { id = "6", text = "BLAUS" });
ViewBag.datasource = customer;
return View();
}
public class Customers
{
public string id { get; set; }
public string text { get; set; }
}
```

Setting dimensions to Popup list

PopupWidth and PopupHeight can be used to create a fixed size popup list.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).Height("50px").Width("500px").PopupHeight("2
00px").PopupWidth("300px")
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "List Item 1" });
DropdownData.Add(new Data { Value = "item2", Text = "List Item 2" });
DropdownData.Add(new Data { Value = "item3", Text = "List Item 3" });
DropdownData.Add(new Data { Value = "item4", Text = "List Item 4" });
DropdownData.Add(new Data { Value = "item5", Text = "List Item 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Setting and Getting Value

You can select single or multiple values from DropDownList control. To assign a value initially to the DropDownList, you can use Value property.

HTML

```
@model MVCApplication.Controllers.HomeController
@using (Html.BeginForm("DropdownlistFeatures", "Dropdownlist",
FormMethod.Post, null))
{
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).Height("50px").Width("500px").PopupHeight("2
00px").PopupWidth("300px").Value("item3")
<input type="submit" value="Get Value" />
}
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "List Item 1" });
DropdownData.Add(new Data { Value = "item2", Text = "List Item 2" });
DropdownData.Add(new Data { Value = "item3", Text = "List Item 3" });
DropdownData.Add(new Data { Value = "item4", Text = "List Item 4" });
DropdownData.Add(new Data { Value = "item5", Text = "List Item 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
[HttpPost]
public ActionResult DropdownlistFeatures(string DropDownList1)
{
//DropDownList1 is ID of DropDownList used in this example. You can get the
selected items value in controller using the ID
string DropDownValue = DropDownList1;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

DropDownList Dependencies

The external script dependencies of the DropDownList control are,

- jQuery 1.7.1 and later versions.
- jQuery.easing - to support the animation effects.

The internal script dependency ej.web.all.js is a bundle of all Syncfusion ASP.NET MVC controls. If you use ej.web.all.js in your application, you can leave this section or else you can try to render

Copyright © 2001 - 2020 Syncfusion Inc.

DropDownList in your application using ej.dropdownlist.min.js file. You can refer the following frameworks and controls in your project.

FileÂ	Description / Usage
ej.core.min.js	Must be referred always before using all the JS controls.
ej.data.min.js	Used to handle data operation and should be used while binding data to JS controls.
ej.dropdownlist.min.js	The dropdownlistâ€™s main file
ej.scroller.min.js	Should be referred when using scrolling in DropDownList.
ej.draggable.min.js	Should be referred when using popup resize functionality in DropDownList.

Rendering Mode

DropDownList control can be created using UL-LI element.

By default DropDownList control is rendered using DropDownList helper to which you can pass popup items using Datasource and DropDownListFields property.

Target Rendering

You can bind the predefined set of UL-LI elements to generate the list of popup items. These items can be customized by adding any images, div elements, radio buttons, text boxes etc.

Create a div with UL-LI elements and assign that div id into TargetID property for rendering the DropDownList control.

HTML

```
@Html.EJ().DropDownList("DropDownList1").TargetID("mail")
<div id="mail">
<ul>
<li>
<div class="mailtools categorize"></div>
Categorize and Move</li>
<li>
<div class="mailtools done"></div>
Done</li>
<li>
<div class="mailtools flag"></div>
Flag & Move</li>
<li>
<div class="mailtools forward"></div>
Forward</li>
<li>
<div class="mailtools movetofolder"></div>
Move to Folder</li>
<li>
<div class="mailtools newmail"></div>
New E-mail</li>
<li>
<div class="mailtools meeting"></div>
New Meeting</li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<li>
<div class="mailtools reply"></div>
Reply & Delete</li>
</ul>
</div>
```

CSS

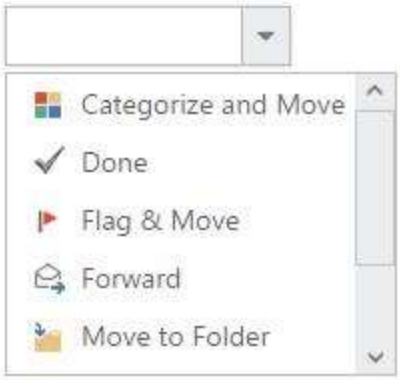
```
.mailtools {
display: block;
background-image: url('iconsapps.png');
height: 25px;
width: 25px;
background-position: center center;
background-repeat: no-repeat;
}
.mailtools.done {
background-position: 0 0;
}
.mailtools.movetofolder {
background-position: 0 -22px;
}
.mailtools.categorize {
background-position: 0 -46px;
}
.mailtools.flag {
background-position: 0 -70px;
}
.mailtools.forward {
background-position: 0 -94px;
}
.mailtools.newmail {
background-position: 0 -116px;
}
.mailtools.reply {
background-position: 0 -140px;
}
.mailtools.meeting {
background-position: 0 -164px;
}
```

JS

```
$(function() {
$('#dropdown1').ejDropDownList({
targetID: "mail"
});
});
```

Note: Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\themes\images

Copyright © 2001 - 2020 Syncfusion Inc.



Note: Any EJ control can be embedded in the target element but the default action of that control should be prevented in order to create custom actions. To show a sample, integration with TreeView control is demonstrated here

Data Binding

To populate data in the DropDownList control, define DataSource property with associated fields. You can bind any list data and other remote services in the DropDownList.

Fields

The following properties provides you a way to bind either local or remote data to the DropDownList control.

Properties	Description
DataSource	The data source contains the list of data for generating the popup listÂ items.
Query	It specifies the query to retrieve the data from the online server.
Fields	It specifies the mapping fields for the data items of the DropDownList control.
ID	It specifies the ID of the tag.
Text	It specifies the text content of the tag.
Value	It specifies the value of the tag.
Category	It is used to categorize the items based on a specific field. The value mapped to this field must be able to group the popup items.
ImageUrl	It defines the image location.
ImageAttributes	It defines the image attributes such as height, width, styles, etc.
SpriteCssClass	It defines the sprite CSS for the image tag to add a prefix icon to all popup items.
HtmlAttributes	It defines the HTML attributes such as class and styles for an item.
Selected	This field defines the tag value to be selected initially. Corresponding field mapped has Boolean values to select the list items on control creation. The data with valueÂ trueÂ in this field is selected automatically when the control is initialized with checkbox.

Copyright © 2001 - 2020 Syncfusion Inc.

TableName	It defines the table name for the tag value or displays text while rendering remote data.

Local Data

Define a List data and initialize the control with DataSource property. Specify the column names in the Fields property.

Note: The columns are bounded automatically when the fields are specified with the default names like id, text, etc...

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Employee>)V
iewData["LocalDataSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Country").Category("Role").ImageUrl("Image").ImageAtt
ributes("ImageAttr"))
```

CSS

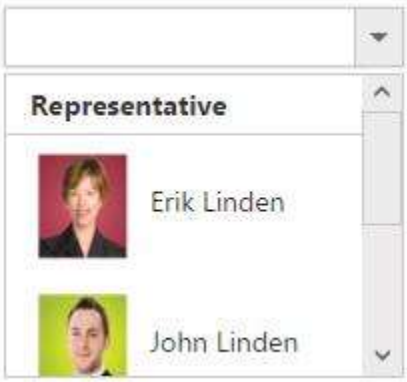
```
.ImageId {
margin: 0;
padding: 3px 10px 3px 3px;
border: 0 none;
width: 60px;
height: 60px;
float: none;
}
```

C#

```
public ActionResult Index()
{
List<Employee> Data = new List<Employee>();
Data.Add(new Employee
{
Text = "Erik Linden",
Role = "Executive",
Country = "England",
Image = "../Content/Employees/3.png",
ImageAttr = "class='ImageId'"
});
Data.Add(new Employee
{
Text = "John Linden",
Role = "Representative",
Country = "Norway",
Image = "../Content/Employees/6.png",
ImageAttr = "class='ImageId'"
});
Data.Add(new Employee
{
Text = "Louis",
Role = "Representative",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Country = "Australia",
Image = "../Content/Employees/7.png",
ImageAttr = "class='ImageId'"
});
Data.Add(new Employee
{
Text = "Lawrence",
Role = "Executive",
Country = "India",
Image = "../Content/Employees/8.png",
ImageAttr = "class='ImageId'"
});
ViewData["LocalDataSource"] = Data;
return View();
}
public class Employee
{
public string Text { get; set; }
public string Role { get; set; }
public string Country { get; set; }
public string Image { get; set; }
public string ImageAttr { get; set; }
}
```



Note: Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\themes\images

SQL Data Binding

Fetch the Data from SQL data source and initialize the control with DataSource property. Specify the column names in the Fields property.

HTML

```
@Html.EJ().DropDownList("selectEmployee").Datasource((IEnumerable<Object>)Vi
ewBag.Data).DropDownListFields(df =>
df.ID("Id").Text("Name").Value("Designation")).Width("150px")
```

C#

```
public ActionResult DropdownlistFeatures()
{
List<Employee> data = new List<Employee>();
```

```
SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\SampleDB.mdf;
Integrated Security=True");
using (con)
{
con.Open();
SqlDataReader myReader = null;
SqlCommand myCommand = new SqlCommand("select * From [Table]",
con);
myReader = myCommand.ExecuteReader();
if (myReader.Read())
{
data.Add(new Employee(myReader["Name"].ToString(),
myReader["Designation"].ToString()));
}
con.Close();
ViewBag.Data = data;
}
return View();
}
public class Employee
{
public string Name { get; set; }
public string Designation { get; set; }
public Employee(string name, string deg)
{
this.Name = name;
this.Designation = deg;
}
}
```

LINQ Data Binding

Language-Integrated Query (LINQ) is not only about retrieving data. It is also a powerful tool for transforming data. By using a LINQ query, you can use a source sequence as input and modify it in many ways to create a new output sequence.

Defines model class

C#

```
public class Employee
{
public string name { set; get; }
public string employeeID { set; get; }
public string designation { set; get; }
public Employee() { }
public Employee(string name, string id, string designation)
{
name = name;
employeeID = id;
designation = designation;
}
}
```

In code behind, fetch the data from Employee class and assigned to the DropDownList as follows

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public ActionResult DropdownlistFeatures()
{
List<Employee> data = new List<Employee>() { };
data.Add(new Employee("Nancy", "E11011", "Technical Writer"));
data.Add(new Employee("Angel", "E11012", "Professor"));
data.Add(new Employee("Daniah", "E11013", "Dancer"));
data.Add(new Employee("Jenifer", "E11014", "Beautician"));
data.Add(new Employee("Prince", "E11015", "Developer"));
var db = from result in data
select result;
ViewBag.dataSource = db.ToList();
return View();
}
```

HTML

```
@Html.EJ().DropDownList("selectCar").Datasource((IEnumerable<Employee>)ViewB
ag.datasource).DropDownListFields(df =>
df.ID("employeeID").Text("name").Value("designation")).WatermarkText("Select
a Student").Width("100%")
```

XmlDataSource

XmlDataSource is used to work with XML documents. The following steps explain the details about the data binding from XmlDataSource

Step 1: Create an XML Data source as follows

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<Students>
<Student>
<Name>Abhishek</Name>
<Location>Dhanbad</Location>
</Student>
<Student>
<Name>Aman</Name>
<Location>Samastipur</Location>
</Student>
<Student>
<Name>Vicky</Name>
<Location>Munger</Location>
</Student>
<Student>
<Name>Chandan</Name>
<Location>Bhagalpur</Location>
</Student>
<Student>
<Name>Ravi</Name>
<Location>Dhanbad</Location>
</Student>
</Students>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Step 2: Read data from the XML file

C#

```
using System.Xml;
using DataBinding.Models;
public ActionResult DropdownlistFeatures()
{
List<Product> data = new List<Product>();
using (XmlReader reader =
XmlReader.Create("D:/Sample/MVC/DataBinding/DataBinding/App_Data/XMLData.xml
"))
{
while (reader.Read())
{
if (reader.IsStartElement())
{
switch (reader.Name.ToString())
{
case "Name":
data.Add(new Product { name = reader.ReadString() });
break;
}
}
}
}
ViewBag.data = data;
return View();
}
```

Step 3: Assign the data to the DropDownList's DataSource property. Specify the column names in the fields property.

RAZOR

```
@using DataBinding.Models
@Html.EJ().DropDownList("selectCar").Datasource((IEnumerable<Product>)ViewBa
g.data).DropDownListFields(df =>
df.Text("name").Value("name")).WatermarkText("Select a
Student").Width("100%")
```

EntityDataSource

Bind data to the DropDownList through Entity Framework. Please follow the below steps.

1. Create an entity data model

Please refer the link for more information

2. Refer the entity data model to your project and bind the data to the DropDownList

In code behind, create an object for the Entity mode (NORTHWNDEntities) and fetch the data from the Employees class and stored it to the List.

Assign the data to the DropDownList's DataSource property. Specify the column names in the fields property.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult DropdownlistFeatures()
{
NORTHWNDEntities sample = new NORTHWNDEntities();
var dataset = sample.Employees
.Select(x => new Employee
{
EmployeeID = x.EmployeeID,
FirstName = x.FirstName
}).ToList();
ViewBag.DataSource = dataset;
}
public class Employee
{
public int EmployeeID { get; set; }
public string FirstName { get; set; }
}
```

RAZOR

```
@Html.EJ().DropDownList("selectEmployee").Datasource((IEnumerable<Object>)Vi
ewBag. DataSource).DropDownListFields(df => df.Text("FirstName
").Value("EmployeeID ")).Width("150px")
```

AccessDataSource

The data can also be bound to the DropDownList using OLEDB database as depicted below.

RAZOR

```
Html.EJ().DropDownList("selectEmployee").Datasource((IEnumerable<Object>)Vie
wBag. DataSource).DropDownListFields(df =>
df.ID("Id").Text("Name").Value("Designation")).Width("150px")
```

The server-side code to retrieve and bind the data to DropDownList are as follows. Also, define a class with all the required fields as depicted in the below code example.

C#

```
public ActionResult DropdownlistFeatures()
{
OleDbConnection conn = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|\Database.mdb");
conn.Open();
OleDbCommand cmd = new OleDbCommand(" SELECT * from [Employees]", conn);
OleDbDataReader reader = null;
List<EmployeeDetails> data = new List<EmployeeDetails>();
reader = cmd.ExecuteReader();
while (reader.Read())
{
data.Add(new EmployeeDetails(reader["Name"].ToString(),
reader["Designation"].ToString()));
}
conn.Close();
ViewBag. DataSource = data;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class EmployeeDetails
{
public string Name { get; set; }
public string Designation { get; set; }
public EmployeeDetails(string name, string deg)
{
this.Name = name;
this.Designation = deg;
}
}
```

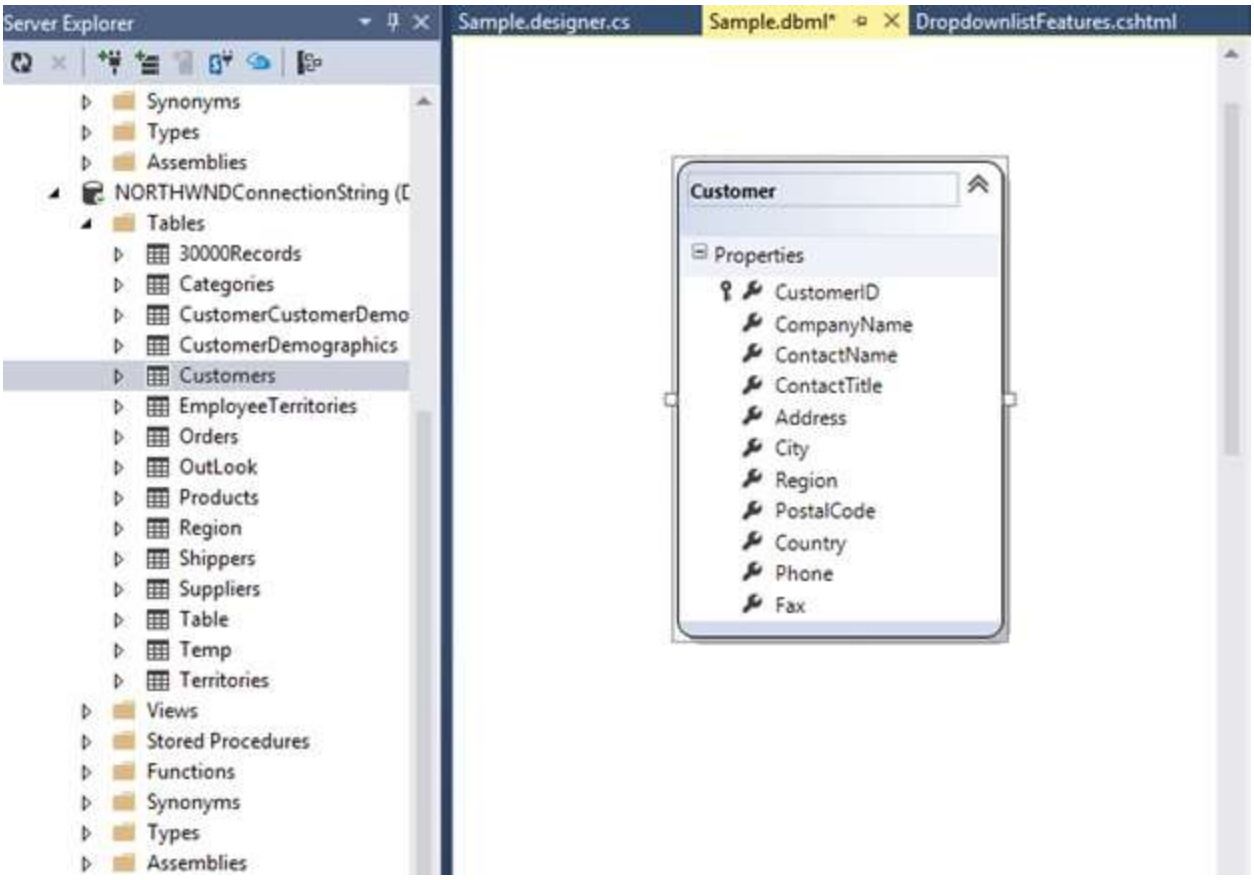
LINQ to SQL Data Binding

The LINQ to SQL can be used as the data source of the DropDownList in which the data model of a relational database is mapped to an object model and allow us to use the LINQ technology to access SQL database.

To create LINQ to SQL classes

- Right-click the App_Code folder and then click Add New Item.
- Under Installed templates, select LINQ to SQL Classes template, type a name for the .dbml file, and then click Add.
- The Object Relational Designer window is displayed.
- In Server Explorer, drag the database table into the Object Relational Designer window.
- The table and its columns are represented as an entity in the designer window.

Copyright © 2001 - 2020 Syncfusion Inc.



In code behind, fetch the data from classes and stored it to the ViewBag as follows

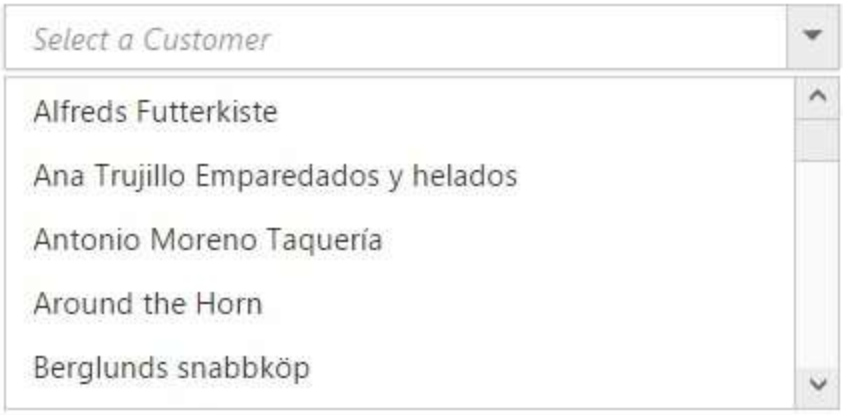
C#

```
ViewBag.dataSource = new SampleDataContext().Customers.ToList();
```

HTML

```
@Html.EJ().DropDownList("customers").Datasource((IEnumerable<Customer>)ViewB
ag.dataSource).DropDownListFields(df =>
df.Text("CompanyName").Value("CustomerID")).WatermarkText("Select a
Customer").Width("100%")
```

Copyright © 2001 - 2020 Syncfusion Inc.



Remote data

To bind remote data to the DropDownList, you can assign a service data as an instance of DataManager to the Datasource property.

OData

OData is a standardized protocol for creating and consuming data. You can provide the OData service URL directly to the Datasource URL property.

HTML

```
@Html.EJ().DropDownList("DropDownList1").Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders")).Dr
opDownListFields(df=>df.Text("ShipCountry").Value("OrderID"))
```

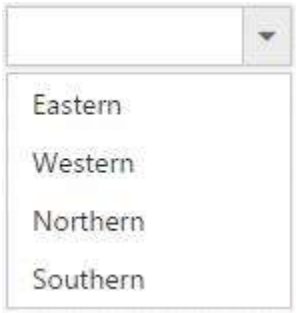
OData Version 4

For OData Version 4 support ODataV4Adaptor should be used. By using URL property of Datasource, you can bind OData Version 4 Service link and specify Adaptor value as enum AdaptorType.ODataV4Adaptor.

For further details about OData service please refer the link.

HTML

```
@Html.EJ().DropDownList("DropDownList1").Datasource(dataSource
=>dataSource.URL("http://services.odata.org/V4/Northwind/Northwind.svc/Regio
ns/").Adaptor(AdaptorType.ODataV4Adaptor)).DropDownListFields(df =>
df.Text("RegionDescription").Value("RegionID"))
```



Note: Events associated with remote data bind is listed here.

Copyright © 2001 - 2020 Syncfusion Inc.

WebAPI

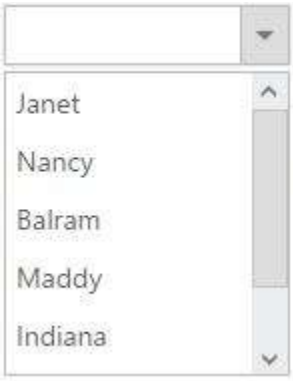
Using WebApiAdaptor, you can bind WebApi service's data to DropDownList. The data from WebApi service must be returned as an object that has property "Items" with its value as data source and another property "Count" with its value as dataSource's total records count.

HTML

```
@Html.EJ().DropDownList("DropDownList1").Datasource(dataSource =>
dataSource.URL("/api/Orders").Adaptor(AdaptorType.WebApiAdaptor)).DropDownLi
stFields(df => df.Text("Name").Value("EmployeeID"))
```

C#

```
public class OrdersController : ApiController
{
NorthwindDataContext db = new NorthwindDataContext();
// GET API/<controller>
public PageResult<EmployeePhoto> Get(ODataQueryOptions opts)
{
List<EmployeePhoto> photos = db.EmployeePhotos.ToList();
return new PageResult<EmployeePhoto>(photos as IEnumerable<EmployeePhoto>,
null, photos.Count);
}
}
```



Virtual Scrolling

To improve the performance when displaying large data set, you can use "AllowVirtualScrolling" and VirtualScrollMode property. This retrieves only a fixed amount of list items and loads remaining data on scrolling. The items will be fetched via AJAX request.

This supports two modes of virtualization. They are,

- Normal Mode
- Continuous Mode

Information: 1. Sorting and Grouping is not supported with Virtual Scrolling

Information: 2. "VirtualScrollMode" property accepts Syncfusion.JavaScript.VirtualScrollMode enum value.

Copyright © 2001 - 2020 Syncfusion Inc.

Normal Mode

It loads the data on scrolling the list of items. This can be achieved by setting Syncfusion.JavaScript.VirtualScrollMode.Normal value to the VirtualScrollMode property.

HTML

```
@Html.EJ().DropDownList("DropDownList1").Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/services/Northwnd.svc/Orders")).Dr
opDownListFields(df =>
df.Text("ShipName").Value("ShipCountry")).AllowVirtualScrolling(true).Virtua
lScrollMode(VirtualScrollMode.Normal).ItemsCount(7)
```

Continuous Mode

It loads the set of items when the scroller reaches at the end. This behaves like infinity scrolling. So when scroll reaches the end, it will fetch the remaining set of items and bind with your DropDownList. This can be achieved by setting Syncfusion.JavaScript.Continuous value to the "VirtualScrollMode" property.

Note: In both modes, set of items will be fetched based on the count specified in the ItemsCount property and next set of items will be loaded on scrolling.

HTML

```
@Html.EJ().DropDownList("DropDownList1").Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/services/Northwnd.svc/Orders")).Dr
opDownListFields(df =>
df.Text("ShipName").Value("ShipCountry")).AllowVirtualScrolling(true).Virtua
lScrollMode(VirtualScrollMode.Continuous).ItemsCount(7)
```

Checkbox

DropDownList displays checkboxes to the left of each item when you set ShowCheckbox property to true. It allows you to select more than one item at a time from DropDownList. Popup list stays open until the user finishes selection. When you click on an item’s text or checkbox then the checkbox checked status get change.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowCheckbox(true)
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropDownData;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



if you want to showcase the DropDownList with default checked items on data binding, specify Selected field with Boolean values.

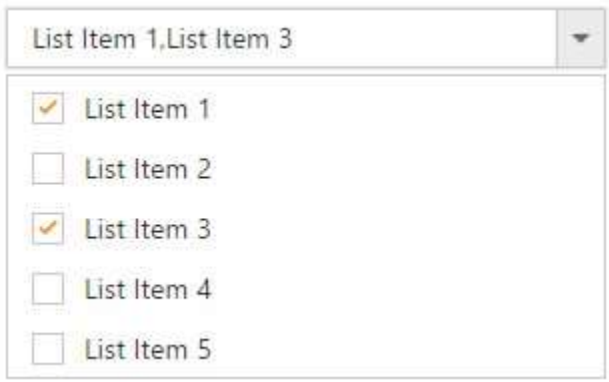
HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value").Selected("Selected")).ShowCheckbox(true).Widt
h("300px")
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1",
Selected=true });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2",
Selected=false });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3",
Selected=true });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4",
Selected=false });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5",
Selected=false });
ViewData["DropDownSource"] = DropDownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
public bool Selected { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Selection Modes

The MultiSelectMode property enables you to make multiple selections in the following two ways:

- Delimiter
- Visual Mode

Information: "MultiSelectMode" property accepts **Syncfusion.JavaScript.MultiSelectModeTypes** enum value.

Delimiter

Each checked item's text is appended to the textbox with delimiter "," by default. This is enabled by assigning **Syncfusion.JavaScript.MultiSelectModeTypes.Delimiter** (enum) value to MultiSelectMode property. You can customize the delimiter option by using DelimiterChar property.

HTML

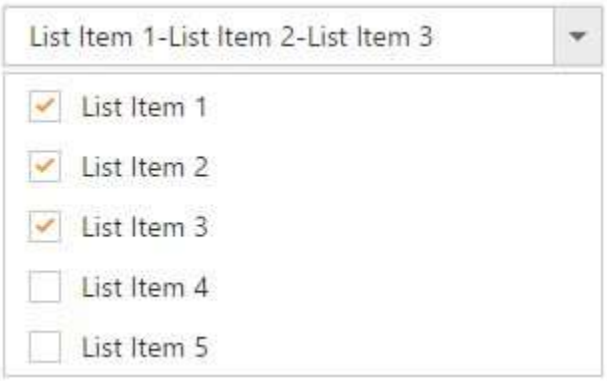
```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowCheckbox(true).MultiSelectMode(MultiSele
ctModeTypes.Delimiter).DelimiterChar("-").Width("300px")
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropDownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```



Visual Mode

When you enable this option in DropDownList control, each checked item's text is appended to the text box in a box model layout. This is enabled by assigning **Syncfusion.JavaScript.MultiSelectModeTypes.VisualMode** (enum) value to MultiSelectMode property.

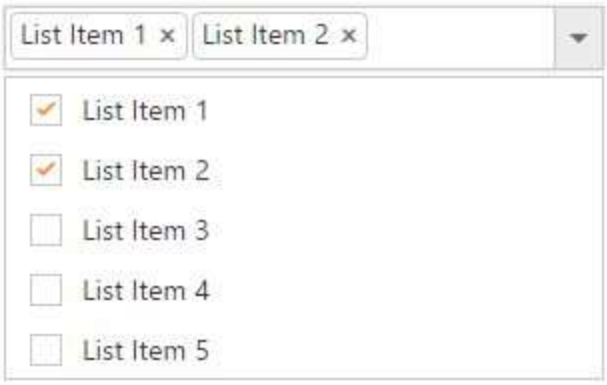
HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowCheckbox(true).Width("300px").MultiSelec
tMode(MultiSelectModeTypes.VisualMode)
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropDownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Select multiple values from controller

To pass selected values from controller, pass selected items using ViewBag, and assign this to the Value property of DropDownList. Refer to the following code.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).Value(ViewBag.Selected).ShowCheckbox(true)
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropDownData;
ViewBag.Selected = "item1,item2";
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

To pass multiple selected items from View to Controller, use ajax post to send selected items through change event of DropDownList. Refer to the following code

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ClientSideEvents(e=>e.Change("Changed")).Sho
wCheckbox(true).Value(ViewBag.Selected)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
function Changed(args) {
$.ajax({
type: "POST",
url: '@Url.Action("Update")',
data: '{value: ' + JSON.stringify(args.model.value) + '}',
dataType: "json",
contentType: "application/json; charset=utf-8",
success: function (e) {
console.log(e.result);
},
error: function () {
alert("Error while sending data");
}
});
}
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropDownData;
ViewBag.Selected = "item1,item2";
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
[HttpPost]
public ActionResult Update(string Value)
{
var selected = Value;
return Json(new { result = Value, JsonRequestBehavior.AllowGet });
}
```

Refer HowTo section to add check all option in popup list

Functionalities

Selection

By default only one item can be selected from the popup list. For multiple selection, you have to enable checkboxes. The selected item consist of active class (“e-active”) to differentiate it from other items.

The following API’s, select the items in the DropDownList via text or value or indices.

Properties	Description

value	To select an item initially, you can pass the itemâ€™s value via value property. Multiple items can select via value property, the given values should be separated by delimiter character.
text	To select an item initially, you can pass the itemâ€™s text via text property. Multiple items can select via text property, the given values should be separated by delimiter character.
selectedIndex	Select a single item by passing an index value to the selectedIndex property.
selectedIndices	Select more than one items by passing index values to the selectedIndices property when multi selection enabled.

Note: Index starts from 0 here.

To use “selectedIndices” property, you should enable wither ShowCheckbox or MultiSelectMode property.

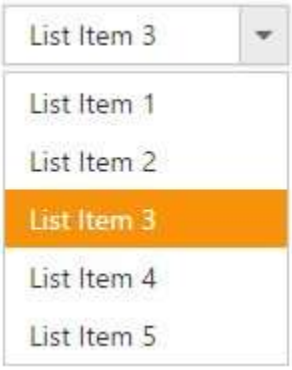
The following methods, select the items in the DropDownList.

Methods	Description
selectItemByIndices	This method is used to select the list of items in the DropDownList through the Index of the items.
selectItemByText	This method is used to select an item in the DropDownList by using the given text value.
selectItemByValue	This method is used to select an item in the DropDownList by using the given value.

The following methods, used to retrieve the items from the DropDownList.

Methods	Description
getListData	This method is used to retrieve the items that are bound with the DropDownList.
getSelectedItem	This method is used to get the selected items in the DropDownList.
getSelectedValue	This method is used to retrieve the items value that are selected in the DropDownList.

Copyright © 2001 - 2020 Syncfusion Inc.



Information: When multiSelectMode is enabled in a DropDownList and selected items having same text but its value is different means, the items can be selected. Please refer the online link

Using value or text

To select an item initially you can pass the item's value via Value property or SelectItemByValue API. To achieve this DropDownList control must be initiated with the associate value.

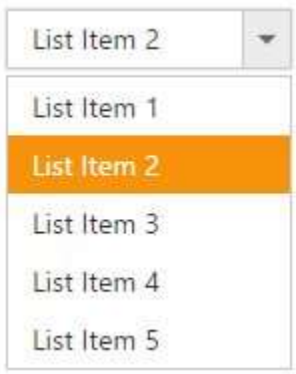
HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).Value("item3")
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```


Copyright © 2001 - 2020 Syncfusion Inc.



Note: To retrieve the selected item's value in controller page, you can use DropDownList control name in the controller post action.

HTML

```
@model MVCApplication.Controllers.HomeController
@using (Html.BeginForm("DropdownlistFeatures", "Dropdownlist",
FormMethod.Post, null))
{
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).Height("50px").Width("500px").PopupHeight("2
00px").PopupWidth("300px")
<input type="submit" value="Get Value" />
}
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
[HttpPost]
public ActionResult DropdownlistFeatures(string DropDownList1)
{
//DropDownList1 is ID of DropDownList used in this example. You can get the
selected items value in controller using the ID
string DropDownValue = DropDownList1;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Using indices

You can select a single or more than one item by passing index values to the properties SelectedIndex or SelectedIndices respectively. Index starts from 0 here.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<Data>)ViewDa
ta["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).SelectedIndex(1)
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```


Information: To use "SelectedIndices" property, you should enable either ShowCheckbox or MultiSelectMode property First.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<Data>)ViewDa
ta["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowCheckbox(true).SelectedIndices(new
List<int> { 1,2 })
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Unselect items

Similarly, you can unselect a single or multiple items by using unselectItemByValue or unselectItemByIndices or unselectItemByText methods. This will remove the selection state of the corresponding data item from the popup list and textbox.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<Data>)ViewDa
ta["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowCheckbox(true).SelectedIndices(new
List<int> { 1, 2, 3 })
<input type="button" value="Unselect" onclick="unselect()" />
```

JS

```
var obj;
$(function() {
obj = $('#dropdown1').data("ejDropDownList");
console.log("Selected Item's Text - " + obj.option("text"));
console.log("selected Item's Value - " + obj.option("value"));
});
function unselect() {
obj.unselectItemByValue("item2");
obj.unselectItemByIndices(2);
obj.unselectItemByText("ListItem 4");
}
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Grouping

The DropDownList items can be categorized by using a specific field in the popup list. This is enabled by using Category field on data source binding. By default grouping is disabled in DropDownList.

The below given example explains the behavior of grouping with List data binding.

HTML

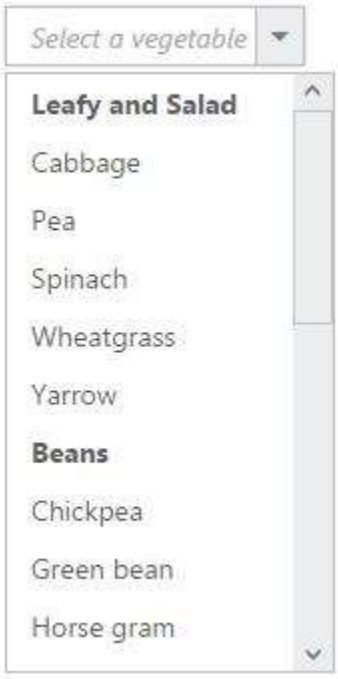
```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<VegetableLis
t>)ViewData["VegData"]).DropDownListFields(Df =>
Df.Text("Name").Category("Category")).Width("150px").PopupHeight("300px").Wa
termarkText("Select a vegetable")
```

C#

```
public ActionResult Index()
{
List<VegetableList> Vegetables = new List<VegetableList>();
Vegetables.Add(new VegetableList { Name = "Cabbage", Category = "Leafy and
Salad" });
Vegetables.Add(new VegetableList { Name = "Pea", Category = "Leafy and
Salad" });
Vegetables.Add(new VegetableList { Name = "Spinach", Category = "Leafy and
Salad" });
Vegetables.Add(new VegetableList { Name = "Wheat grass", Category = "Leafy
and Salad" });
Vegetables.Add(new VegetableList { Name = "Yarrow", Category = "Leafy and
Salad" });
Vegetables.Add(new VegetableList { Name = "Chickpea", Category = "Beans" });
Vegetables.Add(new VegetableList { Name = "Green bean", Category = "Beans"
});
Vegetables.Add(new VegetableList { Name = "Horse gram", Category = "Beans"
});
Vegetables.Add(new VegetableList { Name = "Peanut", Category = "Beans" });
Vegetables.Add(new VegetableList { Name = "Pigeon pea", Category = "Beans"
});
Vegetables.Add(new VegetableList { Name = "Garlic", Category = "Bulb and
Stem" });
Vegetables.Add(new VegetableList { Name = "Garlic Chives", Category = "Bulb
and Stem" });
Vegetables.Add(new VegetableList { Name = "Lotus root", Category = "Bulb and
Stem" });
Vegetables.Add(new VegetableList { Name = "Nopal", Category = "Bulb and
Stem" });
Vegetables.Add(new VegetableList { Name = "Onion", Category = "Bulb and
Stem" });
Vegetables.Add(new VegetableList { Name = "Shallot", Category = "Bulb and
Stem" });
Vegetables.Add(new VegetableList { Name = "Beetroot", Category = "Root and
Tuberous" });
Vegetables.Add(new VegetableList { Name = "Carrot", Category = "Root and
Tuberous" });
Vegetables.Add(new VegetableList { Name = "Ginger", Category = "Root and
Tuberous" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Vegetables.Add(new VegetableList { Name = "Potato", Category = "Root and
Tuberous" });
Vegetables.Add(new VegetableList { Name = "Radish", Category = "Root and
Tuberous"});
Vegetables.Add(new VegetableList { Name = "Turmeric", Category = "Root and
Tuberous" });
ViewData["VegData"] = Vegetables;
return View();
}
public class VegetableList
{
public string Name { get; set; }
public string Category { get; set; }
}
```



Note: Grouping has restrictions in the following scenarios,

1. It is not supported on using HTML "select" element with predefined set of options
2. When using UL-LI elements you need to use “e-category” class in LI element to specify it as the grouping header. The following code will explain this behavior,
3. The sorting behavior varies when grouping is enabled in the DropDownList, based on browser as we have used browser based stable sorting method when there is multiple level of sorting.
4. To overcome this behavior on sorting order with browser, we suggest you to set ej.support.stableSort as false from the script when the page is loaded or in document ready function.

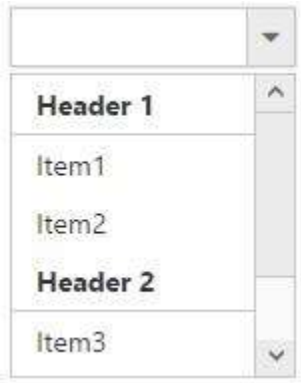
JAVASCRIPT

```
<script type="text/javascript">
$(document).ready(function () {
ej.support.stableSort = false;
```

```
});
</script>
```

HTML

```
@Html.EJ().DropDownList("DropDownList").TargetID("grouping").Width("150px").
PopupHeight("300px").WatermarkText("Select a vegetable")
<div id="grouping">
<ul>
<span class="e-category">Header 1</span>
<li>Item 1</li>
<li>Item 2</li>
<span class="e-category">Header 2</span>
<li>Item 3</li>
<li>Item 4</li>
<li>Item 5</li>
</ul>
</div>
```



Information: Virtual scrolling is not supported with Grouping.

Sorting

Sorting is enabled in order to display the items alphabetically in either ascending or descending order. By default the items is displayed in the initialized order, use EnableSorting property to automatically sort strings based on text field value. You can assign either SortOrder.Ascending or SortOrder.Descending enum values to the SortOrder property to sort out the list items. By default ascending order is followed when SortOrder property is not specified.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<Data>)ViewDa
ta["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).EnableSorting(true).SortOrder(SortOrder.Desc
ending)
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
```

```
DropDownData.Add(new Data { Value = "item1", Text = "List Item 1" });
DropDownData.Add(new Data { Value = "item5", Text = "List Item 5" });
DropDownData.Add(new Data { Value = "item4", Text = "List Item 4" });
DropDownData.Add(new Data { Value = "item2", Text = "List Item 2" });
DropDownData.Add(new Data { Value = "item3", Text = "List Item 3" });
ViewData["DropDownSource"] = DropDownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Information: Virtual scrolling is not supported with Sorting.

Cascading

This works for series of DropDownList in which items are filtered based on the previous DropDownList's selection. Cascading is performed based on the value field and this field should be bounded with a foreign key. To perform cascading, specify the child DropDownList's id in CascadeTo property and use delimiter (",") to specify more than one child DropDownList.

Configuring the data items for cascading to the series of DropDownList is demonstrated below

HTML

```
@model MVCApplication.Controllers.HomeController
<div style="width: 400px;">
<div style="float: left;">
<span>Select Group</span>
@Html.EJ().DropDownList("GroupDropDown").Datasource((IEnumerable<Groups>)Vie
wData["Groups"]).DropDownListFields(Df =>
Df.Text("Text").ID("ParentId").Value("ParentId")).CascadeTo("CountryDropDown
")
</div>
<div style="float: right;">
<span>Select Country</span>
@Html.EJ().DropDownList("CountryDropDown").Datasource((IEnumerable<Country>)
ViewData["CountriesData"]).Enabled(false).DropDownListFields(df=>df.Text("Te
xt"))
</div>
</div>
```

C#

```
public ActionResult Index()
{
List<Groups> GroupList = new List<Groups>();
GroupList.Add(new Groups{ ParentId = "a", Text = "Group A" });
GroupList.Add(new Groups{ ParentId = "b", Text = "Group B" });
GroupList.Add(new Groups{ ParentId = "c", Text = "Group C" });
GroupList.Add(new Groups{ ParentId = "d", Text = "Group D" });
GroupList.Add(new Groups { ParentId = "e", Text = "Group E" });
List<Country> CountryList = new List<Country>();
CountryList.Add(new Country{ ParentId = "a", Text = "Algeria" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
CountryList.Add(new Country{ ParentId = "a", Text = "Armenia" });
CountryList.Add(new Country{ ParentId = "a", Text = "Bangladesh" });
CountryList.Add(new Country{ ParentId = "a", Text = "Cuba" });
CountryList.Add(new Country{ ParentId = "b", Text = "Denmark" });
CountryList.Add(new Country{ ParentId = "b", Text = "Egypt" });
CountryList.Add(new Country{ ParentId = "c", Text = "Finland" });
CountryList.Add(new Country{ ParentId = "c", Text = "India" });
CountryList.Add(new Country{ ParentId = "c", Text = "Malaysia" });
CountryList.Add(new Country{ ParentId = "d", Text = "New Zealand" });
CountryList.Add(new Country{ ParentId = "d", Text = "Norway" });
CountryList.Add(new Country{ ParentId = "d", Text = "Poland" });
CountryList.Add(new Country{ ParentId = "e", Text = "Romania" });
CountryList.Add(new Country{ ParentId = "e", Text = "Singapore" });
CountryList.Add(new Country{ ParentId = "e", Text = "Thailand" });
CountryList.Add(new Country{ ParentId = "e", Text = "Ukraine" });
ViewData["Groups"] = GroupList;
ViewData["CountriesData"] = CountryList;
return View();
}
public class Groups
{
public string ParentId { get; set; }
public string Text { get; set; }
}
public class Country
{
public string ParentId { get; set; }
public string Text { get; set; }
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

Binding the data source to the cascading DropDownList using cascade event

Bind the data source to the cascading DropDownList dynamically using ClientSideEvent Cascade as demonstrated below,

HTML

```
@model MVCApplication.Controllers.HomeController
<div style="width: 530px;">
<div style="float: left;">
<span>Select Group</span>
@Html.EJ().DropDownList("GroupDropDown").Datasource((IEnumerable<Groups>)Vie
wData["Groups"]).DropDownListFields(Df =>
Df.Text("Text").ID("ParentId").Value("ParentId")).CascadeTo("CountryDropDown
, PlayersDropDown")
</div>
<div style="float: left; padding-left: 50px;">
<span>Select Country</span>
@Html.EJ().DropDownList("CountryDropDown").Datasource((IEnumerable<Country>)
ViewData["CountriesData"]).Enabled(false).DropDownListFields(df =>
df.Text("Text"))
</div>
<div style="float: right;">
<span>Select Players</span>
@Html.EJ().DropDownList("PlayersDropDown").Datasource((IEnumerable<Players>)
ViewData["PlayersData"]).Enabled(false).DropDownListFields(df =>
df.Text("Text"))
</div>
</div>
```

C#

```
public ActionResult Index()
{
List<Groups> GroupList = new List<Groups>();
GroupList.Add(new Groups{ ParentId = "a", Text = "Group A" });
GroupList.Add(new Groups{ ParentId = "b", Text = "Group B" });
List<Country> CountryList = new List<Country>();
CountryList.Add(new Country{ ParentId = "a", Text = "Algeria" });
CountryList.Add(new Country{ ParentId = "a", Text = "Armenia" });
CountryList.Add(new Country{ ParentId = "b", Text = "Denmark" });
CountryList.Add(new Country{ ParentId = "b", Text = "Egypt" });
List<Players> PlayerList = new List<Players>();
PlayerList.Add(new Players { ParentId = "a", Text = "Adams" });
PlayerList.Add(new Players { ParentId = "a", Text = "Clarke" });
PlayerList.Add(new Players { ParentId = "b", Text = "Brett" });
PlayerList.Add(new Players { ParentId = "b", Text = "James" });
ViewData["Groups"] = GroupList;
ViewData["CountriesData"] = CountryList;
ViewData["PlayersData"] = PlayerList;
return View();
}
public class Groups
{
public string ParentId { get; set; }
public string Text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class Country
{
public string ParentId { get; set; }
public string Text { get; set; }
}
public class Players
{
public string ParentId { get; set; }
public string Text { get; set; }
}
```



Multi-Level Cascading

The below scenario can be explained with three DropDownList for the multi-level cascading.

C#

```
public partial class DropdownlistController: Controller
{
List<groups> group = new List<groups>();
List<Countries> country = new List<Countries>();
List<Capital> capital = new List<Capital>();
public ActionResult DropdownlistFeatures()
{
group.Add(new groups { parentId = "a", text = "Group A" });
group.Add(new groups { parentId = "b", text = "Group B" });
group.Add(new groups { parentId = "c", text = "Group C" });
group.Add(new groups { parentId = "d", text = "Group D" });
group.Add(new groups { parentId = "e", text = "Group E" });
ViewBag.datasource = group;
country.Add(new Countries { value = 11, parentId = "a", text = "Algeria" });
country.Add(new Countries { value = 12, parentId = "a", text = "Armenia" });
country.Add(new Countries { value = 13, parentId = "a", text = "Bangladesh"
});
country.Add(new Countries { value = 14, parentId = "a", text = "Cuba" });
country.Add(new Countries { value = 15, parentId = "b", text = "Denmark" });
country.Add(new Countries { value = 16, parentId = "b", text = "Egypt"});
country.Add(new Countries { value = 17, parentId = "c", text = "Finland"});
country.Add(new Countries { value = 18, parentId = "c", text = "India" });
country.Add(new Countries { value = 19, parentId = "c", text = "Malaysia"});
country.Add(new Countries { value = 20, parentId = "d", text = "New
Zealand"});
country.Add(new Countries { value = 21, parentId = "d", text = "Norway" });
country.Add(new Countries { value = 22, parentId = "d", text = "Romania" });
country.Add(new Countries { value = 23, parentId = "e", text =
"Singapore"});
country.Add(new Countries { value = 24, parentId = "e", text = "Thailand"
});
```

```
country.Add(new Countries { value = 25, parentId = "e", text = "Ukraine"});
ViewBag.datasource1 = country;
capital.Add(new Capital { value = 11, text = "Algiers" });
capital.Add(new Capital { value = 12, text = "Cairo" });
capital.Add(new Capital { value = 13, text = "Copenhagen" });
capital.Add(new Capital { value = 14, text = "Washington" });
capital.Add(new Capital { value = 15, text = "Denmark" });
capital.Add(new Capital { value = 16, text = "Yerevan" });
capital.Add(new Capital { value = 17, text = "Copenhagen" });
capital.Add(new Capital { value = 18, text = "New Delhi" });
capital.Add(new Capital { value = 19, text = "Havana" });
capital.Add(new Capital { value = 20, text = "Brasilia" });
capital.Add(new Capital { value = 21, text = "Lima" });
capital.Add(new Capital { value = 22, text = "Canberra" });
capital.Add(new Capital { value = 23, text = "Wellington" });
capital.Add(new Capital { value = 24, text = "Alfred Faure" });
capital.Add(new Capital { value = 25, text = "King Edward Point" });
ViewBag.capital = capital;
return View();
}
}
public class groups
{
public string text { get; set; }
public string parentId { get; set; }
}
public class Countries
{
public string text { get; set; }
public int value { get; set; }
public string parentId { get; set; }
}
public class Capital
{
public string text { get; set; }
public int value { get; set; }
}
```

HTML

```
<div class="control" style="float: left;">
<span class="txt">Select Group</span>
@Html.EJ().DropDownList("groupsList").Datasource((IEnumerable<groups>)ViewBa
g.datasource).DropDownListFields(f =>
f.Value("parentId").Text("text")).CascadeTo("countryList")
</div>
<div class="control" style="float: left;">
<span class="txt">Select Country</span>
@Html.EJ().DropDownList("countryList").Datasource((IEnumerable<Countries>)Vi
ewBag.datasource1).DropDownListFields(f =>
f.Value("value").Text("text")).CascadeTo("capitalList").Enabled(false)
</div>
<div class="control" style="float: left;">
<span class="txt">Select Country</span>
@Html.EJ().DropDownList("capitalList").Datasource((IEnumerable<Capital>)View
Bag.capital).Enabled(false)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
```

First two DropDownList cascaded based on the parentId, and then from second to third, cascading performed based on the value field.

Validations

DropDownList value can be validated in following ways,

1. Using jQuery Validator
2. Client side Validation
3. Server side validation

Using jQuery Validator

You can validate the DropDownList value on form submission using jQuery Validations, by applying “ValidationRules” and “ValidationMessage” to the DropDownList.

Note: jquery.validate.min script file should be referred for validation, for more details, refer here.

Validation Rules

The validation rules help you to verify the selected text by adding validation attributes to the input element. This can be set by using ValidationRules property.

Validation Messages

You can set your own custom error message by using ValidationMessage property. To display the error message, specify the corresponding annotation attribute followed by the message to display.

Note: jQuery predefined error messages to that annotation attribute will be shown when this property is not defined. The below given example explain this behavior of ‘required’ attribute,

When the DropDownList control is rendered, it creates an input hidden element which is used to store the selected items value. Hence, the validation is performed based on the value stored in this hidden element.

Required field and min value validation is demonstrated in the below given example.

HTML

```
@model MVCApplication.Controllers.HomeController
<form id="form1">
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ValidationMessage(message =>
message.AddMessage("required", "* Required").AddMessage("min","Select >
30")).ValidationRules(vr => vr.AddRule("required", true).AddRule("min",30))
<br /><input type="submit" value="Submit" />
</form>
```

JAVASCRIPT

```
$.validator.setDefaults({
ignore: [],
errorClass: 'e-validation-error', // to get the error message on jQuery
validation
errorPlacement: function (error, element) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$(error).insertAfter(element.closest(".e-widget"));
}
// any other default options and/or rules
});
//If necessary, we can create custom rules as below. here method defined for
min
$.validator.addMethod("min",
function (value, element, params) {
if (!/Invalid|NaN/.test(value)) {
return parseInt(value) > params;
}
}, 'Must be greater than 30.');
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "10", Text = "10" });
DropdownData.Add(new Data { Value = "20", Text = "20" });
DropdownData.Add(new Data { Value = "30", Text = "30" });
DropdownData.Add(new Data { Value = "40", Text = "40" });
DropdownData.Add(new Data { Value = "50", Text = "50" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Client Side Validation

When you are developing the MVC application in Visual Studio then the client-side becomes enabled by default, but you can easily enable or disable the writing of the following app setting code snippet in the web.config file.

XML

```
<appSettings>
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

After setting the value as **true,** refer to the jQuery validation script file in the **_Layout page** as shown in the following code example.

HTML

```
<script src="@Url.Content("~/Scripts/jquery.validate.js")"></script>
```

The jQuery validation plug-in takes advantage of the Data Annotation attributes defined in the model. Let's create a Sample model that has a single property with Data annotation attributes.

C#

```
using System.ComponentModel.DataAnnotations;
public class DropDownListModel
{
[Required(ErrorMessage = "DropDownList value is Required")]
public List<DropDownValue> DropData { get; set; }
}
public class DropDownValue
{
public string Text { get; set; }
public string Value { get; set; }
}
```

After that you need to create the controller's action methods. These render views on the UI and bind a model with the view. So let's create a controller as follows.

The view is created as in the following code snippet:

RAZOR

```
@model MvcApplication.Models.DropDownListModel
@using (Html.BeginForm())
{
@Html.ValidationSummary(true)
@Html.EJ().DropDownListFor(Model => Model.
DropData,(Syncfusion.JavaScript.Models.DropDownListProperties)ViewData["prop
erties"])
<br />
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Post").Type(ButtonType
.Submit)
}
```

C#

```
using MvcApplication.Models;
using Syncfusion.JavaScript.Models;
public ActionResult DropdownlistFeatures()
{
BindingData();
return View();
}
public void BindingData()
{
List<DropDownValue> data = new List<DropDownValue>() { };
data.Add(new DropDownValue() { Value = "item1", Text = "List Item 1" });
data.Add(new DropDownValue() { Value = "item2", Text = "List Item 2" });
data.Add(new DropDownValue() { Value = "item3", Text = "List Item 3" });
data.Add(new DropDownValue() { Value = "item4", Text = "List Item 4" });
data.Add(new DropDownValue() { Value = "item5", Text = "List Item 5" });
DropDownListProperties obj = new DropDownListProperties();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
obj.DataSource = data;
DropDownListFields fields = new DropDownListFields();
fields.Text = "Text";
fields.Value = "Value";
obj.DropDownListFields = fields;
ViewData["properties"] = obj;
}
[HttpPost]
public ActionResult DropdownlistFeatures(DropDownListModel model)
{
BindingData();
return View(model);
}
```

Server side Validation

The ASP.NET MVC Framework validates any data passed to the controller action that is executing.

For DropDownList, we have to create a sample using Data Annotation API to validate the model data for it.

1. Create a model named DropDownListModel (DropDownListModel.*cs*) under the *Models* folder and applies Data Annotation attributes on the properties of the DropDownListModel class that is the Text property of the DropDownList.
2. Now, create an action method in the controller that returns a view with a model after the post request.

C#

```
using System.ComponentModel.DataAnnotations;
public class DropDownListModel
{
public List<DropDownValue> DropData { get; set; }
}
public class DropDownValue
{
public string Text { get; set; }
public string Value { get; set; }
}
```

C#

```
using MvcApplication.Models;
using Syncfusion.JavaScript.Models;
public ActionResult DropdownlistFeatures()
{
BindingData();
return View();
}
public void BindingData()
{
List<DropDownValue> data = new List<DropDownValue>() { };
data.Add(new DropDownValue() { Value = "item1", Text = "List Item 1" });
data.Add(new DropDownValue() { Value = "item2", Text = "List Item 2" });
data.Add(new DropDownValue() { Value = "item3", Text = "List Item 3" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
data.Add(new DropDownValue() { Value = "item4", Text = "List Item 4" });
data.Add(new DropDownValue() { Value = "item5", Text = "List Item 5" });
DropDownListProperties obj = new DropDownListProperties();
obj.DataSource = data;
DropDownListFields fields = new DropDownListFields();
fields.Text = "Text";
fields.Value = "Value";
obj.DropDownListFields = fields;
ViewData["properties"] = obj;
}
[HttpPost]
public ActionResult DropdownlistFeatures(DropDownListModel model)
{
if(ModelState.IsValid)
BindingData();
return View(model);
}
```

3. After that, created a view to get value and show an error message if the value of the DropDownList is not given.

RAZOR

```
@using (Html.BeginForm())
{
@Html.ValidationSummary(true)
@Html.EJ().DropDownListFor(Model =>
Model.DropData,(Syncfusion.JavaScript.Models.DropDownListProperties)ViewData
["properties"])
<br />
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Post").Type(ButtonType
.Submit)
}
```

Template Support

By default you can add any text or image to the DropDownList list item. To customize the items layout or to create your own visualized elements you can use this template support.

Header Template

You can create the popup header by using HeaderTemplate property. You can add any HTML content in header template.

Note: Please refer the How-To section for the check all option in Header

Template Field

Create a set of div containers with common syntax or elements and assign it to the Template property. You can add any HTML mark-up element inside the DropDownList list using this property.

In the demo, a List data is created with Text, Image, Role and Country which is initialized with DataSource property. Content template is created by using the corresponding fields and assigned in template property. The content template is customized with images and custom CSS styles to visualize the items in popup.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Employee1>)
ViewData["LocalDataSource"]).HeaderTemplate("<div
class='eheader'><span>PHOTO</span>
<span>DETAILS</span></div>").Template("<div><Image class='ImageId'
src='../Content/Employees/${Image}.png' alt='employee'/> <div class='ename'>
${Text} </div><div class='role'> ${Role} </div><div class='cont'> ${Country}
</div></div>")
```

CSS

```
.ImageId {
margin: 0;
padding: 3px 10px 3px 3px;
border: 0 none;
width: 60px;
height: 60px;
float: none;
}
```

C#

```
public ActionResult Index()
{
List<Employee1> Data = new List<Employee1>();
Data.Add(new Employee1
{
Text = "Erik Linden",
Role = "Representative",
Country = "England",
Image = "3",
ImageAttr = "class='ImageId'"
});
Data.Add(new Employee1
{
Text = "John Linden",
Role = "Representative",
Country = "Norway",
Image = "6",
ImageAttr = "class='ImageId'"
});
Data.Add(new Employee1
{
Text = "Louis",
Role = "Representative",
Country = "Australia",
Image = "7",
ImageAttr = "class='ImageId'"
});
Data.Add(new Employee1
{
Text = "Lawrence",
Role = "Representative",
Country = "India",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Image = "8",
ImageAttr = "class='ImageId'"
});
ViewData["LocalDataSource"] = Data;
return View();
}
public class Employee
{
public string Text { get; set; }
public string Role { get; set; }
public string Country { get; set; }
public string Image { get; set; }
public string ImageAttr { get; set; }
}
```

Note: Images for this sample are available in (installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\samples\web\themes\images



Customization

Adding watermark text

It provides the short description of the expected value in dropdown and will display the text until any item is selected. You can set this text using WatermarkText property.

HTML

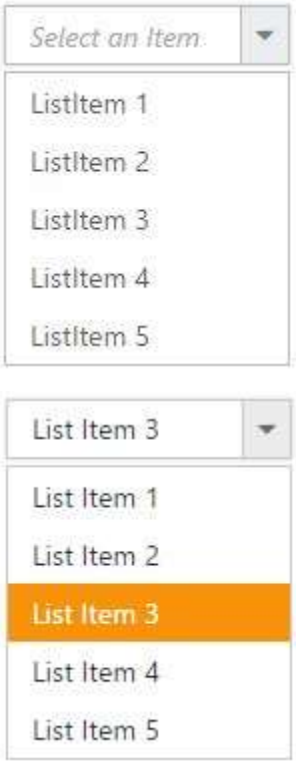
```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).WatermarkText("Select an Item")
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Applying Rounded Corner

You can use ShowRoundedCorner property to add rounded borders to the input and popup elements. By default, rounded corner property is disabled in DropDownList.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowRoundedCorner(true)
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Information: The browser support details for rounded corner is given here.

Enable/Disable the control

The Enabled property is used to indicate whether the control can respond to the user interaction or not. You can disable it by assigning false to this property. When the control is disabled state, you cannot interact with the control.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).Enabled(false)
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Applying HTML Attributes

Additional HTML attributes can be applied to the control by using HtmlAttributes property. The attributes such as name, required, read-only and disabled are directly applied to the input element of DropDownList, and other attributes such as style, class will be applied to the outer wrapper element of DropDownList.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).HtmlAttributes((IDictionary<string,object>)V
iewData["HtmlAttrData"])
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
object StyleObj = new object();
StyleObj = "border:1px solid red;";
Dictionary<string, object> AttrVal = new Dictionary<string, object>
{
{"style" , StyleObj}
};
ViewData["HtmlAttrData"] = AttrVal;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Setting dimensions

Control Sizing

Fixed Size DropDownList control

You can customize the control dimensions using Width and Height properties. Fixed size values can be specified in pixel or percentage values. By default the DropDownList wrapper will be assigned with "143px" Width and "30px" Height.

Copyright © 2001 - 2020 Syncfusion Inc.

Fixed size popup list

You can customize the popup list dimensions using PopupWidth and PopupHeight properties. Fixed size values can be specified in pixel or percentage values. By default popup width is auto and popup height is "152px".

Auto Sizing

DropDownList is adaptive to mobile and web layout such that it is adjustable with screen resolution. The textbox will be rendered based on its parent containers dimensions on assigning 100% values to the width property. Default value for PopupWidth is auto, so when you assign 100% to PopupWidth then it will be rendered based on specified range.

Limit the number of items

You can use ItemsCount property to fetch only the specific number of items from the data source. To fetch the remaining items you can enable virtual scrolling support which loads the data on scrolling the data items in popup list.

Note: By default popup list is shown on DropDownList button click but you can display the list initially by enabling the ShowPopupOnLoad property.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowPopupOnLoad(true).ItemsCount(3)
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

Popup resizing

To show a resize handle in the popup list, use EnablePopupResize property. You can customize the resize functionality by setting dimensions to the following properties.

MinPopupWidth	Default value is 0, once set you cannot resize below to the specified width
MaxPopupWidth	Default value is null, once set you cannot extend beyond to the specified width
MinPopupHeight	Default value is 0, once set you cannot resize below to the specified height
MaxPopupHeight	Default value is null, once set you cannot extend beyond to the specified height

HTML

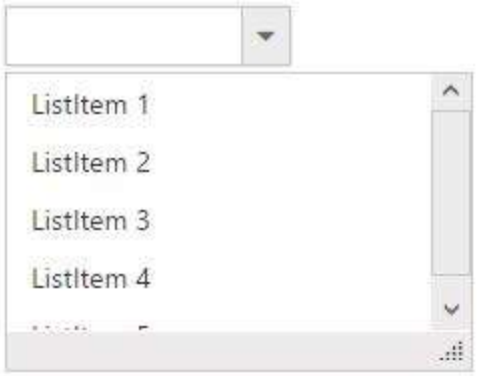
```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).EnablePopupResize(true).MaxPopupHeight("550p
x").MaxPopupWidth("550px").MinPopupHeight("150px").MinPopupWidth("150px")
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropdownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropdownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropdownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropdownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Copyright © 2001 - 2020 Syncfusion Inc.



Search

Items are searched based on the keyed in values to the textbox. There are two types of searches,

- Incremental Search
- Filter Search

Incremental Search

Selects the item in the popup list based on the keyed in value. If the time taken to type exceeds 1000 milliseconds then filtered items will be reset based on the current input value. By default this mode of search is enabled. Incremental search can be case sensitive or case insensitive. To make case sensitive, you can use CaseSensitiveSearch property.

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<Data>)ViewDa
ta["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).EnableIncrementalSearch(true).CaseSensitiveS
earch(true)
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "emp1", Text = "Adams", Selected = true
});
DropDownData.Add(new Data { Value = "emp2", Text = "James", Selected = false
});
DropDownData.Add(new Data { Value = "emp3", Text = "Maria", Selected = true
});
DropDownData.Add(new Data { Value = "emp4", Text = "Jessica", Selected =
false });
DropDownData.Add(new Data { Value = "emp5", Text = "Jenneth", Selected =
false });
ViewData["DropDownSource"] = DropDownData;
return View();
}
public class Data
{
public string Value { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public string Text { get; set; }
}
```



Filter search

You can quickly locate specific item within a large data source by filtering matches with a search box. A text box appears in the popup list for searching when EnableFilterSearch property is enabled. By default, filtering returns the matched items list based on text in search textbox.

You can configure the search filter by using FilterType property that takes SearchFilterType enum values. There is two types of filter options,

- Starts With
- Contains

Note: Items are filtered based on “SearchFilterType.Contains” filter type by default.

HTML

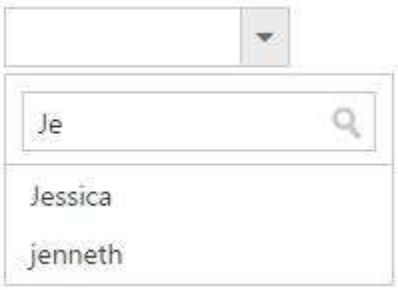
```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList").Datasource((IEnumerable<Data>)ViewDa
ta["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).EnableFilterSearch(true).FilterType(SearchFi
lterType.StartsWith)
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "emp1", Text = "Adams", Selected = true
});
DropDownData.Add(new Data { Value = "emp2", Text = "James", Selected = false
});
DropDownData.Add(new Data { Value = "emp3", Text = "Maria", Selected = true
});
DropDownData.Add(new Data { Value = "emp4", Text = "Jessica", Selected =
false });
DropDownData.Add(new Data { Value = "emp5", Text = "Jenneth", Selected =
false });
ViewData["DropDownSource"] = DropDownData;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```



Information: When VirtualScrolling enabled with searching, then filter will be applied only on the DropDownList items available at the moment.

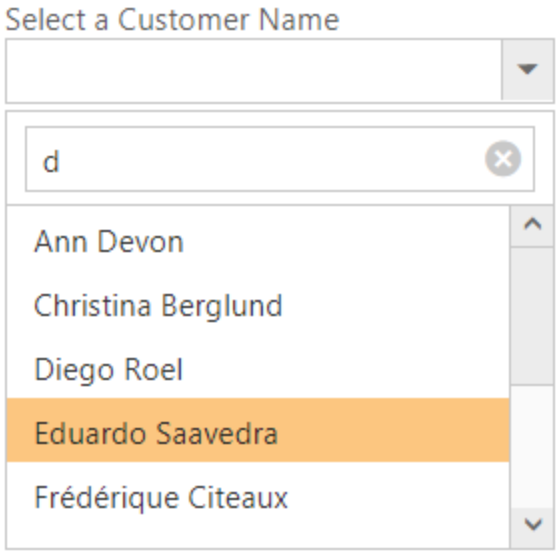
Server Filtering

Server filtering for displaying only a fixed amount of dataset from the whole dataset. In general, DropDownList displays just the data returned from the server. This feature is convenient for you to apply when the user does not want to see the whole dataset in the popup wrapper.

EnableServerFiltering If set to true, the filtering operations performed in the remote service and returns the result.

HTML

```
<div class="ctrllabel">Select a Customer Name</div>
@(Html.EJ().DropDownList("ContactName")
.Datasource(ds =>
ds.URL("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Customer
s").CrossDomain(true))
.DropDownListFields(f => f.Text("ContactName").Value("ContactName"))
.Width("100%")
.EnableFilterSearch(true)
.EnableServerFiltering(true)
.WatermarkText("Select a Customer Name")
)
```



This sample raises the query on Customer service. Returns ContactName records for customers with ContactName containing the string “d”.



State Persistence

DropDownList stores its model is local storage by defining the property EnablePersistence).

You can sustain the below given functionalities in DropDownList by enabling persistence property,

- Selected items value in the textbox
- Item’s selection state in the popup list

Control model will be stored in local storage / cookies of browser before page refreshes and reinitialized with the restored model after refresh.

The following properties are not included while maintaining DropDownList’s state in local storage to keep localStorage compact.

- Fields
- Data source
- Query

HTML

```
@model MVCApplication.Controllers.HomeController
<form id="form1">
```

```
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ShowCheckbox(true).Width("300px").EnablePers
istence(true)
<input type="submit" value="Get Value" />
</form>
```

C#

```
public ActionResult Index()
{
List<Data> DropDownData = new List<Data>();
DropDownData.Add(new Data { Value = "item1", Text = "ListItem 1" });
DropDownData.Add(new Data { Value = "item2", Text = "ListItem 2" });
DropDownData.Add(new Data { Value = "item3", Text = "ListItem 3" });
DropDownData.Add(new Data { Value = "item4", Text = "ListItem 4" });
DropDownData.Add(new Data { Value = "item5", Text = "ListItem 5" });
ViewData["DropDownSource"] = DropDownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

Accessing currently stored state

Persisted state can be accessed through local storage using corresponding key name. Key name formation would be in below order. It is combination of plugin name and control id.

JS

```
// DropDownList state as string
var dropdownlistStateString = window.localStorage.ejDropDownList_dropDown;
//DropDownList state as object
var dropdownlistStateObject =
JSON.parse(window.localStorage.ejDropDownList_dropDown);
```

Note: In the above example, ‘ejDropDownList’ is plugin name and _dropDown is control id.

Strongly Typed HTML Helper

The DropDownList control supports strongly typed HTML helpers which uses lambda expression to refer models or view models passed to a view template. These helpers allow you to define the value of the DropDownListFor from the model. The following steps explain how to use the strongly typed helpers to create DropDownListFor.

Add a class named ”DropDownListModel" in the Models folder and replace the code with the following code:

C#

```
public class DropDownListModel
{
[Display(Name = "Drop Down")]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public List<DropDownValue> DropData { get; set; }
}
public class DropDownValue
{
public string Text { get; set; }
public string Value { get; set; }
}
```

In the controller, pass the model to the View.

C#

```
using MvcApplication.Models;
using Syncfusion.JavaScript.Models;
public ActionResult DropdownlistFeatures()
{
DropDownListModel model = new DropDownListModel();
BindingData();
return View(model);
}
public void BindingData()
{
List<DropDownValue> data = new List<DropDownValue>() { };
data.Add(new DropDownValue() { Value = "item1", Text = "List Item 1" });
data.Add(new DropDownValue() { Value = "item2", Text = "List Item 2" });
data.Add(new DropDownValue() { Value = "item3", Text = "List Item 3" });
data.Add(new DropDownValue() { Value = "item4", Text = "List Item 4" });
data.Add(new DropDownValue() { Value = "item5", Text = "List Item 5" });
DropDownListProperties obj = new DropDownListProperties();
obj.DataSource = data;
DropDownListFields fields = new DropDownListFields();
fields.Text = "Text";
fields.Value = "Value";
obj.DropDownListFields = fields;
ViewData["properties"] = obj;
}
```

In View, invoke the strongly typed DropDownListFor helper with the lambda expression to set the default value.

RAZOR

```
@model MvcApplication.Models. DropDownListModel
@Html.EJ().DropDownListFor(Model =>
Model.DropData,(Syncfusion.JavaScript.Models.DropDownListProperties)ViewData
["properties"])
```



Cascading operation can be performed in DropDownListFor.

Defines the Employee class as follows

C#

```
public class DropDownListModel
{
public List<Employee> DropData { get; set; }
public List<DeptDetails> getDept { get; set; }
}
public class Employee
{
public string Name { get; set; }
public string Department { get; set; }
}
public class DeptDetails
{
public string Department { get; set; }
public string Subject { get; set; }
}
```

In controller, defines Data source and properties for the both DropDownLists. The model will be passed to the view page. Here “DropDownListModel” class will be passed to the view.

C#

```
public ActionResult DropdownlistFeatures()
{
List<Employee> data = new List<Employee>() { };
data.Add(new Employee() { Name = "Nancy", Department = "Computer Science"
});
data.Add(new Employee() { Name = "Angel", Department = "Mechanics" });
data.Add(new Employee() { Name = "Prince", Department = "Electronics
Communication" });
data.Add(new Employee() { Name = "John", Department = "Information
Technology" });
data.Add(new Employee() { Name = "Daniha", Department = "Marine" });
DropDownListModel model = new DropDownListModel();
DropDownListProperties obj = new DropDownListProperties();
DropDownListFields fields = new DropDownListFields();
obj.DataSource = data;
obj.CascadeTo = "getDept";
fields.Text = "Name";
fields.Value = "Department";
obj.DropDownListFields = fields;
ViewData["properties"] = obj;
List<DeptDetails> dept = new List<DeptDetails>() { };
dept.Add(new DeptDetails() { Department = "Computer Science", Subject =
"Software Engineering" });
dept.Add(new DeptDetails() { Department = "Computer Science", Subject =
"Software Testing" });
dept.Add(new DeptDetails() { Department = "Computer Science", Subject = "Web
Technology" });
dept.Add(new DeptDetails() { Department = "Computer Science", Subject =
"Data Ware Housing" });
dept.Add(new DeptDetails() { Department = "Mechanics", Subject = "Thermal
Engineering" });
dept.Add(new DeptDetails() { Department = "Mechanics", Subject = "Materials
Engineering, Composites." });
dept.Add(new DeptDetails() { Department = "Mechanics", Subject = "Statics
and dynamics" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
dept.Add(new DeptDetails() { Department = "Mechanics", Subject = "Strength
of materials and solid mechanics" });
dept.Add(new DeptDetails() { Department = "Electronics Communication",
Subject = "Signals & Systems" });
dept.Add(new DeptDetails() { Department = "Electronics Communication",
Subject = "Electromagnetic Engineering" });
dept.Add(new DeptDetails() { Department = "Electronics Communication",
Subject = "Electronics Communication" });
dept.Add(new DeptDetails() { Department = "Information Technology", Subject
= "Analog Electronic Circuits" });
dept.Add(new DeptDetails() { Department = "Marine", Subject = "Effects of
waves" });
dept.Add(new DeptDetails() { Department = "Marine", Subject = "Towing and
mooring" });
dept.Add(new DeptDetails() { Department = "Marine", Subject = "Ocean
structures" });
DropDownListProperties deptProperties = new DropDownListProperties();
DropDownListFields objFields = new DropDownListFields();
deptProperties.DataSource = dept;
deptProperties.Enabled = false;
objFields.Text = "Subject";
objFields.Value = "Department";
deptProperties.DropDownListFields = objFields;
ViewData["Department"] = deptProperties;
return View(model);
}
```

C#

```
@Html.EJ().DropDownListFor(Model => Model.DropData,
(Syncfusion.JavaScript.Models.DropDownListProperties)ViewData["properties"])
@Html.EJ().DropDownListFor(Model => Model.getDept,
(Syncfusion.JavaScript.Models.DropDownListProperties)ViewData["Department"])
```

Localization

The DropDownList provides option to localize its strings, it is used to adapting the DropDownList to a particular local language. By default, the DropDownList will use the US English (en-US) as its language.

Note: The culture name has to be specified in a standard format such as [Language Code]-[County/Region Code].

To localize the dropdownlist's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the DropDownList in German language ("de-DE").

JAVASCRIPT

```
ej.DropDownList.Locale["de-DE"] = {
emptyResultText: "Keine Vorschläge" //replace with your text
};
```

Set the locale property of the DropDownList to the new language.

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().DropDownList("selectCar").Datasource((IEnumerable<Product>)ViewBa
g.data).DropDownListFields(df =>
df.Text("name").Value("name")).WatermarkText("Select a
Student").Width("100%").Locale("de-
DE").EnableFilterSearch(true).EnableIncrementalSearch(true)
```



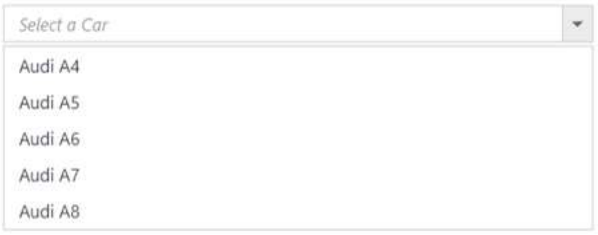
Load On Demand

Load on demand feature allows the DropDownList items load on request from the service/database, during only on click the DropDown icon or DropDownList. This functionality helps to improve performance on control initial rendering time. Because data items load on request.

The loadOnDemand property is used to enable or disable the load on demand functionality of the DropDownList.

ASPX-CS

```
@Html.EJ().DropDownList("selectCar").TargetID("carsList").Width("30%").Water
markText("Select a Car").LoadOnDemand(true)
<div id="carsList">
<ul>
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
<li>Audi A8</li>
</ul>
</div>
```



Accessibility

Keyboard Navigation

You can use the keyboard short cut keys as an alternative to the mouse and interact with DropDownList control.

Keyboard shortcut keys are,

Key	Single selection	Multi-selection
Access key + j	Sets focus to the input	Sets focus to the input
Up	Selects the previous item when the popup is opened/closed	Selects the previous item when the popup is opened/closed and clears all other selection
Down	Selects the next item when the popup is opened/closed	Selects the next item when the popup is opened/closed and clears all other selection
Page up	Selects the item in next page (i.e.,) scrolls down to next set of items when the popup is opened/closed	Selects the item in next page (i.e.,) scrolls down to next set of items when the popup is opened/closed and clear all existing selection
Page down	Selects the item in previous page (i.e.,) scrolls up to previous set of items when the popup is opened/closed	Selects the item in previous page (i.e.,) scrolls up to previous set of items when the popup is opened/closed and clear all existing selection
Left arrow	Selects the previous item when the popup is opened/closed. But in RTL mode selects next item	Selects the previous item when the popup is opened/closed and clears all other selection but in RTL mode selects the next item
Right arrow	Selects the next item when the popup is opened/closed. But in RTL mode selects previous item	Selects the next item when the popup is opened/closed and clears all other selection but in RTL mode selects the previous item
Backspace	No actions	Deletes the selected items from last when visual mode is enabled. (i.e.,) on first backspace key press, the last item will be selected and second backspace press will remove the corresponding item.
Tab	Close the popup if it is opened and focuses to next DOM element	Close the popup if it is opened and focuses to next DOM element
End	Selects the last list item	Select the last item and clear all selection
Home	Selects the first list item	Select the first item and clear all selection
Esc	Close the popup if itâ€™s opened	Close the popup if itâ€™s opened

Copyright © 2001 - 2020 Syncfusion Inc.

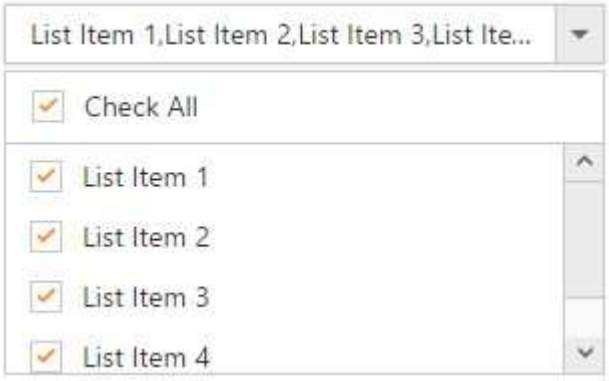
Space bar	Close the popup is itâ€™s opened	Check and uncheck the items which is focused.
Delete	No actions	Deletes the focused item in textbox when visual mode is enabled
Shift + up	Selects the previous item when the popup is opened/closed	Selects the immediate preceding item from current selected item
Shift + down	Selects the next item when the popup is opened/closed	Selects the immediate successive item from current selected item
Shift + home	Select the first list item	Select all items between focused item to first item
Shift + end	Selects the last list item	Select all items between focused item to last item
Shift + page up	Selects the item in previous page (i.e.,) scrolls down to previous set of items when the popup is opened/closed	Select all item between focused item to previous page or previous set of items when the popup is opened/closed
Shift + page down	Selects the item in next page (i.e.,) scrolls up to next set of items when the popup is opened/closed	Select all item between focused item to next page or next set of items when the popup is opened/closed
Ctrl + up	Selects the previous item when the popup is opened/closed	Maintain current selections and hovers to the previous item when the popup is opened/closed
Ctrl + down	Selects the next item when the popup is opened/closed	Maintain current selection and hovers to the next item when the popup is opened/closed
Alt + down	Opens the popup if itâ€™s closed	Open the popup if itâ€™s closed
Alt + up	Closes the popup if itâ€™s opened	Close the popup if itâ€™s opened
Enter	Closes the popup if itâ€™s opened	Selects the focused item when ctrl key pressed otherwise closes the popup if itâ€™s opened
Shift + Tab	Focuses the previous DOM element	Focuses the previous DOM element
Ctrl + shift + up	Selects the previous item when the popup is opened/closed	Maintains the existing items selection and selects the previous item from current selection.

Copyright © 2001 - 2020 Syncfusion Inc.

Ctrl + shift + down	Selects the next item when the popup is opened/closed	Maintain the existing items selection and selects next item from current selection.

Note:

i) *For Ctrl and Shift operations with multi-selection, the current focused item should be selected.*

ii) 

No round robin method is used on navigating through the list items in pop up, so that navigation stops when reaching the start/end of the popup list.

How To

Add check all option in popup list?

You can use HeaderTemplate property to add any HTML element. Code snippet to add check all option is given below,

HTML

```
@model MVCApplication.Controllers.HomeController
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).HeaderTemplate("<div class='temp'><input
id='check' type='checkbox' />
</div>").ShowCheckbox(true).Width("300").ClientSideEvents(clientEvent=>clien
tEvent.Create("create"))
```

CSS

```
.temp {
height: 30px;
display: block;
padding-left: 13px;
padding-top: 5px;
border-bottom: 1px solid #c8c8c8;
}
.e-chkbox-wrap .e-text {
font-size: 14px;
padding-left: 10px;
}
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
function create(args) {
$("#check").ejCheckBox({ text: "Check All", change: "Change" });
}
function Change(args) {
window.flag = true;
var obj = $("#DropDownList1").ejDropDownList("instance");
if (args.isChecked) obj.checkAll();
else obj.uncheckAll();
window.flag = false;
}
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "item1", Text = "List Item 1" });
DropdownData.Add(new Data { Value = "item2", Text = "List Item 2" });
DropdownData.Add(new Data { Value = "item3", Text = "List Item 3" });
DropdownData.Add(new Data { Value = "item4", Text = "List Item 4" });
DropdownData.Add(new Data { Value = "item5", Text = "List Item 5" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
public string Value { get; set; }
public string Text { get; set; }
}
```

The following screenshot exhibits the output of the above code,

Render the DropDownList in Partial View using normal mode

Create a Partial View and define as follows

RAZOR

```
@using DropdownList.Models
@Html.EJ().DropDownList("DropDown").Datasource((List<DropDownValue>)ViewData
["DataSrc"]).DropDownListFields(df=>df.Text("Text").Value("Value"))
@Html.EJ().ScriptManager()
```

Note: While rendering a control in the partial view, we need to specify @Html.EJ().ScriptManager() in the partial view page after the initializing the control.

Render the partial view into parental view

RAZOR

```
@Html.Partial("PartialView1")
```

Define DataSource and other required properties for rendering a simple DropDownList

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public ActionResult MainView()
{
List<DropDownValue> data = new List<DropDownValue>() { };
data.Add(new DropDownValue() { Value = "item1", Text = "List Item 1" });
data.Add(new DropDownValue() { Value = "item2", Text = "List Item 2" });
data.Add(new DropDownValue() { Value = "item3", Text = "List Item 3" });
data.Add(new DropDownValue() { Value = "item4", Text = "List Item 4" });
data.Add(new DropDownValue() { Value = "item5", Text = "List Item 5" });
ViewData["DataSrc"] = data;
return View();
}
```

Render the DropDownList in Partial View using Unobtrusive mode

When you enable the **unobtrusive** in your application, refer to the **ej.unobtrusive.min.js** file in the application.

You have to set the value of **UnobtrusiveJavaScriptEnabled** to **true** in Root directory **web.config** file as shown in the following code example.

XML

```
<appSettings>
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

After setting the value as **true,** refer to the unobtrusive script file in the **_Layout page** as shown in the following code example.

RAZOR

```
<head>
<script src="@Url.Content("~/Scripts/ej/ej.unobtrusive.min.js")"></script>
</head>
```

Create a Partial View and define as follows

RAZOR

```
@using DropdownList.Models
@Html.EJ().DropDownList("DropDown").Datasource((List<DropDownValue>)ViewData
["DataSrc"]).DropDownListFields(df=>df.Text("Text").Value("Value"))
```

Render the partial view into parental view.

RAZOR

```
@Html.Partial("PartialView1")
```

Define DataSource and other required properties for rendering a simple DropDownList

C#

```
public ActionResult MainView()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
List<DropDownValue> data = new List<DropDownValue>() { };
data.Add(new DropDownValue() { Value = "item1", Text = "List Item 1" });
data.Add(new DropDownValue() { Value = "item2", Text = "List Item 2" });
data.Add(new DropDownValue() { Value = "item3", Text = "List Item 3" });
data.Add(new DropDownValue() { Value = "item4", Text = "List Item 4" });
data.Add(new DropDownValue() { Value = "item5", Text = "List Item 5" });
ViewData["DataSrc"] = data;
return View();
}
```

Dynamically bind the data to ejDropDownList on AJAX call

Render the DropDownList with the empty DataSource, In client event called “click” of Button control, the AJAX post is called.

C#

```
@Html.EJ().DropDownList("bikeList").DropDownListFields(df =>
df.ID("id").Text("text")).ClientSideEvents(e => e.Create("onCreate"))<br />
@Html.EJ().Button("button").Text("click").ClientSideEvents(e =>
e.Click("onClick"))
```

HTML

```
public class DropDownController : Controller
{
// GET: DropDown
public ActionResult Index()
{
return View();
}
public ActionResult Add()
{
return RedirectToAction("Index", "DropDown");
}
}
```

Once the AJAX call is success, bind the data to the dropdown list on AJAX call by using the property “dataSource”.

JAVASCRIPT

```
function onClick() {
$.ajax({
url: '/DropDown/Add',
type: 'POST'
}).success(function (result) {
BikeList = [
{ id: "bk1", text: "Apache RTR" }, { id: "bk2", text: "CBR 150-R" }, { id:
"bk3", text: "CBZ Xtreme" },
{ id: "bk4", text: "Discover" }, { id: "bk5", text: "Dazzler" }, { id:
"bk6", text: "Flame" },
{ id: "bk7", text: "Fazer" }, { id: "bk8", text: "FZ-S" }, { id: "bk9",
text: "Pulsar" },
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ id: "bk10", text: "Shine" }, { id: "bk11", text: "R15" }, { id: "bk12",
text: "Unicorn" }
];
var data1 = $("#bikeList").data("ejDropDownList");
data1.option("dataSource", BikeList);
})
}
```

Remove validation error message on item selection

You can remove the validation error message of DropDownList on selecting an item through change event using the below code.

RAZOR

```
@model MVCApplication.Controllers.HomeController
<form id="form1">
@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewD
ata["DropDownSource"]).ClientSideEvents(e =>
e.Change("change")).DropDownListFields(Df =>
Df.Text("Text").Value("Value")).ValidationMessage(message =>
message.AddMessage("required", "* Required")).ValidationRules(vr =>
vr.AddRule("required", true))
<br /><input type="submit" value="Submit" />
</form>
```

JAVASCRIPT

```
$.validator.setDefaults({
ignore: [],
errorClass: 'e-validation-error', // to get the error message on jQuery
validation
errorPlacement: function (error, element) {
$(error).insertAfter(element.closest(".e-widget"));
}
// any other default options and/or rules
});
function change(args){
this.element.valid();  //validation of dropdownlist through change event.
}
```

C#

```
public ActionResult Index()
{
List<Data> DropdownData = new List<Data>();
DropdownData.Add(new Data { Value = "10", Text = "10" });
DropdownData.Add(new Data { Value = "20", Text = "20" });
DropdownData.Add(new Data { Value = "30", Text = "30" });
DropdownData.Add(new Data { Value = "40", Text = "40" });
DropdownData.Add(new Data { Value = "50", Text = "50" });
ViewData["DropDownSource"] = DropdownData;
return View();
}
public class Data
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public string Value { get; set; }
public string Text { get; set; }
}
```

Render DropDownList from code behind

DropDownList can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of DropDownList using DropDownList properties from code behind.

RAZOR

```
@{Html.EJ().DropDownList("Customer", Model).Render(); }
```

C#

```
public ActionResult Index()
{
DropDownListProperties DDLObj = new DropDownListProperties();
DDLObj.DataSource =
"http://js.syncfusion.com/ejServices/wcf/NorthWind.svc/";
DDLObj.Query = "ej.Query().from('Suppliers').select('SupplierID',
'ContactName').take(10)";
DDLObj.DropDownListFields.Text = "ContactName";
DDLObj.DropDownListFields.Value = "SupplierID";
return View(DDLObj);
}
```

FileExplorer

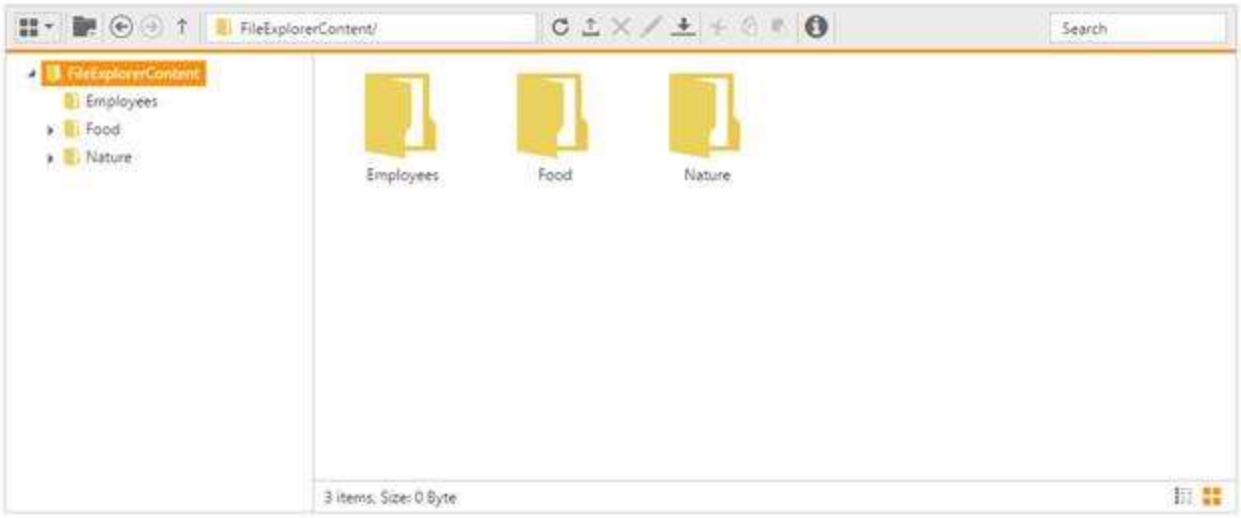
Overview

The EJ MVC FileExplorer component provides all the functionalities that is available in EJ FileExplorer, since it is a server side wrapper of EJ FileExplorer widget. It is an interface for managing the file system through any web application. It allows the user to perform the most common file operations such as browse files, open, create, rename, copy, paste or move, delete and file searching.

For a live demo of FileExplorer check the online sample from here.

Note: The FileExplorer control was officially added with the Essential Studio ASP.NET MVC package from the v13.1 release only.

Copyright © 2001 - 2020 Syncfusion Inc.



Key features

- Windows explorer like functionalities and appearances
- Handy file operations (copy, paste, move and delete)
- File download and upload
- File type restriction
- Easy UI customization
- Different layouts (grid, tile and large icons view)
- Context menu support
- Built-in image viewer support
- Keyboard navigation
- Right to Left alignment (RTL) support
- Localization support
- File access rules

Getting Started

The following section is briefly explain the things to get started with FileExplorer control.

A new MVC Application and required assemblies with dependent files

To get start with Essential ASP.NET MVC FileExplorer, create a new MVC Application and add the required assemblies in references and then refer the below specified dependent CSS file as well as scripts

To create a MVC Project and add necessary assemblies you can use the help of the given MVC-Getting Started Documentation.

Adding the references

To include the control in the application the following references need to be added:

- CSS references
- Script references

CSS file

- ej.web.all.min.css includes all widgets styles (To know more about theme refer Theming in Essential JavaScript Component

Note: Essential JS widgets having the support for 13 built-in themes, to know more please check here

Script references

The external script dependencies of the FileExplorer widget are,

- jQuery 1.7.1 or later versions.
- jsrender – for grid view template.

And the internal script dependencies of the FileExplorer widget are:

File	Description/Usage
ej.core.min.js	Must be referred always before using all the JS controls.
ej.data.min.js	Used to handle data operation and should be used while binding data to JS controls.
ej.touch.min.js	Used to handle touch operations
ej. draggable.min.js	Used to handle the drag and drop functionality
ej.scroller.min.js	Used to show the scrollbar in the layout area
ej.button.min.js	Used to display the buttons in the toolbar
ej.checkbox.min.js	Used to display the checkbox in files items
ej.splitbutton.min.js	Used to display the split buttons in the toolbar
ej.treeview.min.js	Used to display the â€œTreeViewâ€• in the navigation pane
ej.uploadbox.min.js	Used to perform the upload functionalityÂ
ej.waitingpopup.min.js	Used to showcase the waiting popup
ej.dialog.min.js	Used to create the alert windowsÂ
ej.splitter.min.js	Used as the body section to separate the navigation and layout area
ej.toolbar.min.js	Used to showcase the hearer section
ej.menu.min.js	Used to showcase the context menu
ej.grid.min.js	Used to showcase the grid layout view

Copyright © 2001 - 2020 Syncfusion Inc.

ej.fileexplorer.min.js	FileExplorer plugin
ej.globalize.min.js	Used to working with different localization culture formats

You can use the “**ej.web.all.min.js**” file, which encapsulates all the EJ MVC controls and frameworks in one single file

Note: To add required assembly references, scripts and CSS files automatically into your application, please refer following link http://help.syncfusion.com/aspnetmvc/getting-started#through-syncfusion-nuget-packages-2. Also, it configures the “web.config” file automatically.

FileExplorer using helper

In the view page, add FileExplorer helper as shown below.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
)
```

In above code block, “Path” denotes the URL of filesystem that are to be explored in "FileExplorer" and “AjaxAction” specifies the URL of server side AJAX handling method that handles the file operations of FileExplorer control. So “Path” and “AjaxAction” are the mandatory configuration at here.

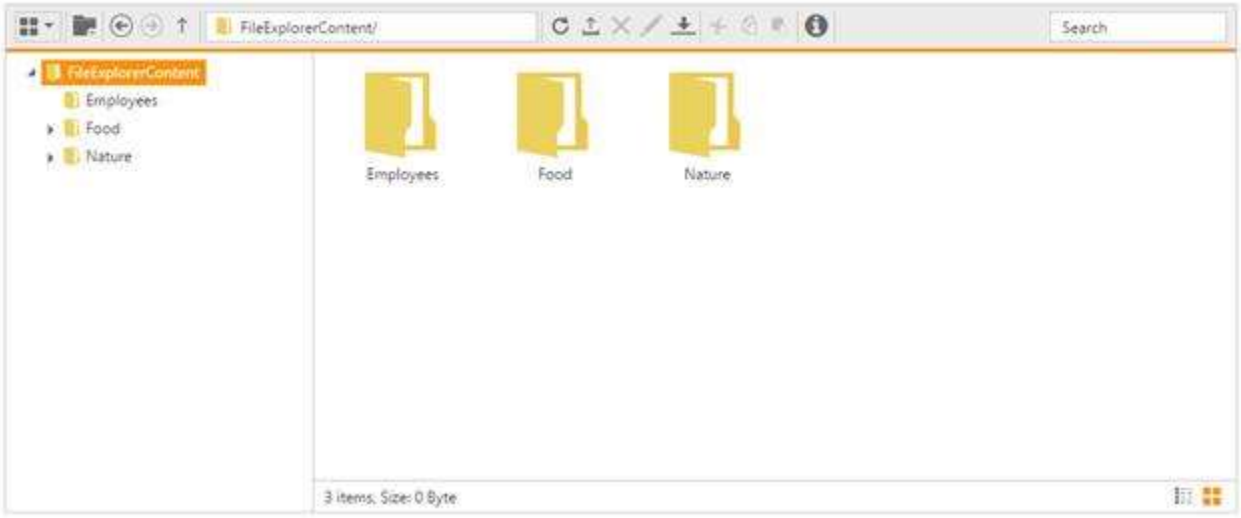
Add the following code example to the corresponding controller page.

C#

```
public ActionResult FileActionDefault(FileExplorerParams args)
{
FileExplorerOperations operation = new FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
case "Search":
```

```
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
}
return Json("");
}
```

Once you have completed the above steps, you will get an output like below.



Note: In the section, you can know the details about file handling operations, which are performed at server side of “FileExplorer”.

Note: If you are using 13.3.0.7rd or later version of script files and assemblies, above specified code blocks are helpful to render “FileExplorer” control. Also in future, we will provide the support based on this. If you are using the older version of Essential Studio (Before 13.3.0.7), refer following section to render “FileExplorer” control.

Behavior Settings

Here are the some properties to customize the behavior of FileExplorer.

File type restriction

FileExplorer control showcase all the files from the filesystem, here you can customize the file types that what you want to show by using “FileTypes” property. It filter the files based on the file extensions.

By default it having the value " \.\ ", so it allows all the file types. In case of image browser you can allow the image files only by setting “*.png,* .gif, *.jpg,* .jpeg”, so it doesn’t allow the other type of files.

In the view page, add FileExplorer helper and specify the file type restriction as shown below.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.FileTypes("*.png, *.gif, *.jpg, *.jpeg, *.docx")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Customize the AJAX request settings

As you already know FileExplorer is a client – server based control and each action performed in the client sends an AJAX request to the server to perform the server side operations. While the AJAX request, the AJAX configurations can be customized through "AjaxSettings" property.

The beforeAjaxRequest event will be triggered before the AJAX request is performed. You can modify the ajax request in this event.

You can see the following requests passed during the **client – server** actions:

- Read
- Create folder
- Remove
- Rename
- Paste
- Get details
- Upload
- Download
- Get Image
- Search

The actions "Read, CreateFolder, Remove, Rename, Paste, GetDetails, Search" are supported all the jQuery AJAX configurations. The remaining actions "Upload, Download, GetImage" are accepted the URL only.

If you want to customize read action alone, the AJAX "**Url**" and "**DataType**" are changed for the "Read" action.

In the view page, add FileExplorer helper and specify the AJAX settings for Read request as shown below.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/")
.AjaxAction("http://js.syncfusion.com/demos/ejServices/api/FileExplorer/File
Operations")
.AjaxSettings(settings =>
settings.Read(read =>
read.Url("http://js.syncfusion.com/demos/ejServices/api/FileExplorer/FileOpe
rationsCors")
.DataType("jsonp")
))
)
```

User Interface

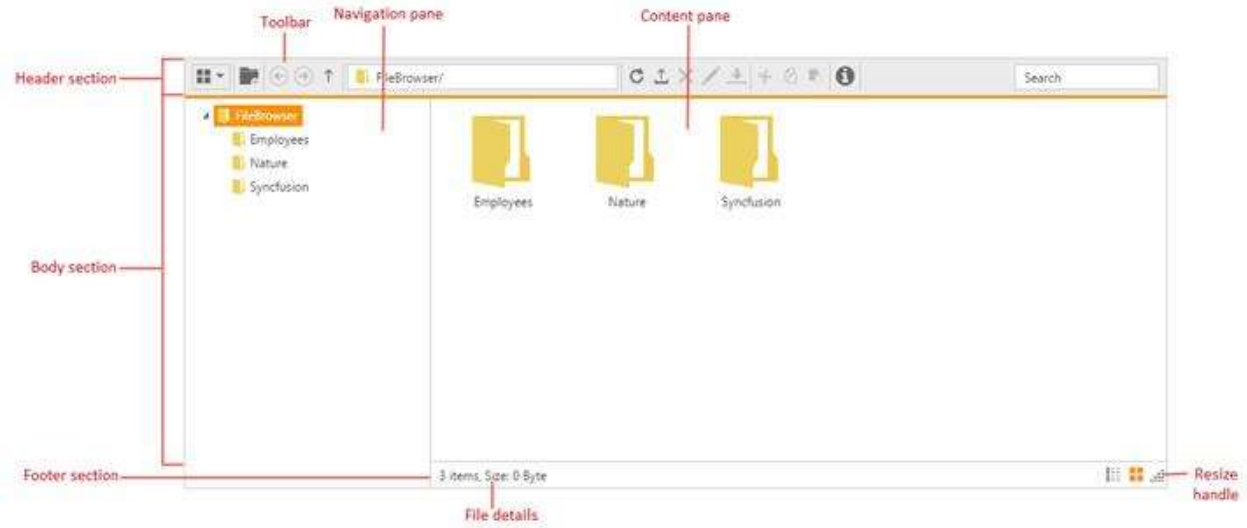
FileExplorer UI combined of several sections like header, body and footer. Each section having the corresponding components to perform the FileExplorer related operations.

- The **header section** contains the toolbar which having the list of tools to perform the file and navigation related operations.
- The **body section** is the main area which explores the filesystem contents. It separated into

Copyright © 2001 - 2020 Syncfusion Inc.

- **Navigation pane** which contains the tree view to showcase the folder hierarchy
- **Content pane** which show cases the files from the file system. The files in the content pane can be viewable in the following modes
- Grid
- Tile
- Large Icons
- The **footer section** contains the status bar which showcases the file details.

The following screenshot shows the diagrammatic detail of FileExplorer UI:



From above screenshot, you can see FileExplorer has several subcomponents for different functionalities. The upcoming sections explains the brief details of each components and their customizations.

File Actions

The FileExplorer provides the support for file and folder related operations and these can be done through the toolbar items as well as through the context menu items.

File action through Toolbar

The toolbar element is placed in the header section, and it having the list of tools to perform the file, navigation and search related operations.

For more details about file action through toolbar, please refer here.

File action through Context menu

The FileExplorer having the support for context menu which opens while right click on any file or folder. The context menu having the set of items which is used to perform the corresponding operations in a handy picture way. For example the user can delete the file through the context menu by right click on it.

In FileExplorer, in the following places the context menu appears.

- While right click on “TreeView” nodes (from navigation pane)
- While right click on file / folder
- While right click on layout (content pane)

Copyright © 2001 - 2020 Syncfusion Inc.

For more details about file action through context menu, please refer here.

Access Control

The FileExplorer provides method to assign permissions or access rights to specific users and group of users. It allows you to define access permissions for folders and files using a set of access rules to user(s).

The FileAccessInfo allows you to specify a Root **Path** and **Role**, and lists the permission (allow or deny) associated with this **Rules**. You can specify permissions to Copy, Download, Edit, Edit Contents, Path, Read, Role, and Upload a file/folder. The rule priority is determined by its index within an access rules collection.

Syntax

C#

```
// For File rule
new AccessRule { Path = "Documents/*.*", Role = "Document Manager", Read =
Permission.Allow, Edit = Permission.Allow, Copy = Permission.Allow, Download
= Permission.Allow },
// For Folder rule
new AccessRule { Path = "Documents/*", Role = "Document Manager", Read =
Permission.Allow, Edit = Permission.Allow, Copy = Permission.Allow,
EditContents = Permission.Allow, Upload = Permission.Allow },
```

The below table list out the FileAccessInfo properties:

Properties	Description
Role	Specifies the which role is applied
Rules	Specifies the collections of rules for folder and files
RootPath	Specifies the root path of the folderâ€™s which want to be secure

The below table below list out the AccessRule properties which is available for file and folder access rules.

Properties	Applicable for File	Applicable for Folder	Description
Copy	Yes	Yes	Specifies the permission to copy a file/folder
Download	Yes	No	Specifies the permission to download a file
Edit	Yes	Yes	Specifies the permission to edit a file/folder
EditContents	No	Yes	Specifies the permission to edit its content for folder
Path	Yes	Yes	Specifies the path to apply the rules which are defined
Read	Yes	Yes	Specifies the permission to read a file/folder content

Copyright © 2001 - 2020 Syncfusion Inc.

Role	Yes	Yes	Specifies the role to which the rule is applied
Upload	No	Yes	Specifies the permission to upload files to a folder

To specify the rule to be allow/deny, use the Permission enumeration values:

Permission	Description
Allow	Specify the rule to be allowed
Deny	Specify the rule to be denied

You can associate any number of access rules with specific security roles, it allows grouping related permissions together. To associate an access rule with a security role, assign the role's name to the AccessRule's Role property.

FileAccessOperations

The **FileAccessOperations** class allows user to perform the most common file operations such as read, create, rename, copy, paste or move, delete and file searching. The class helps to handle the file access operations in server end. You implement a new custom class for handling file operations in server end by inheriting this class.

By default, we send following parameters in data field of corresponding AJAX request to handle the server side operation. The request parameter and response data are explained in following table.

Operation	Default Request Parameter	Response data	Details
Read	string path, string filter, IEnumerable<object> selectedItems	It should be in JSON format with key name as files and cwd. The JSON fields contains the following field names 'name, isFile, hasChild, permission'. For example: {"cwd":{"name":"FileContent","type":"Directory","size":0,"dateModified":"3/30/2016 9:19:27 PM","hasChild":true,"isFile":false,"filterPath":null,"permission":{"Copy":false,"Download":false,"Edit":false,"EditContents":false,"Read":true,"Upload":false}}, "files":[{"name":"Documents","type":"Directory","size":0,"dateModified":"3/30/2016 9:20:10 PM","hasChild":false,"isFile":false,"filterPath":null,"permission":{"Copy":true,"Download":false,"Edit":false,"EditContents":true,"Read":true,"Upload":true}}, {"name":"Nature","type":"Directory","size":0,"dateModified":"3/30/2016 9:19:27 PM","hasChild":false,"isFile":false,"filterPath":null,"permission":{"Copy":false,"Download":false,"Edit":false,"EditContents":false,"Read":false,"Upload":false}}], "details":null, "error":null}	It used to get all immediate files and sub-folders of the given path and

			it returns the matched type of files only, which are specified in 'filter' parameter. (here cwd represents details of the given path and files represents details of the child files and folders of the

			given path)
CreateFolder	string path, string name, IEnumerable<object> selectedItems	It is in JSON format with key name as files. The 'name' and 'permission' fields are necessary on return JSON data. For example: {"cwd":null, "files":[{"name":"New Folder", "type":"Directory","size":0,"dateModified":"4/6/2016 2:55:22 PM","hasChild":true,"isFile":false,"filterPath":null,"permission":{"Copy":true, "Download":false,"Edit":true,"EditContents":true,"Read":true,"Upload":true} }], "details":null, "error":null}	It used to create a new folder in given path with specified name.(here files represents details of the newly added folderâ€™s)
Paste	string sourceDir, string backupDir, string[] names, string option, IEnumerable<CommonFileDetails> commonFiles, IEnumerable<o	It returns the pasted file details or null. For example: {"cwd":null, "files":[{"name":"bird(1).jpg","type":"File","size":102182,"dateModified":"3/23/2016 6:05:32 PM","hasChild":false,"isFile":true,"filterPath":null,"permission":{"Copy":true, "Download":true,"Edit":true,"EditContents":true,"Read":true,"Upload":true}}], "details":null, "error":null}	This method helps to copy or move files from

	bject> selectedItems, IEnumerable<object> targetFolder		one location to another location.(here files represents details of the pasted items)
Remove	string[] names, string path, IEnumerable<object> selectedItems	It returns the removed file details or null. For example: {"cwd":null, "files":[{"name":"New folder", "type":"Directory","size":0,"dateModified":"4/6/2016 3:05:17 PM","hasChild":true,"isFile":false,"filterPath":null,"permission":{"Copy":true, "Download":false,"Edit":true,"EditContents":true,"Read":true,"Upload":true} }], "details":null, "error":null}	It helps to remove the specified items from given path.(here files represents details of the removed

			item)
Rename	string path, string oldName, string newName, IEnumerable<CommonFileDetails> commonFiles, IEnumerable<object> selectedItems	It returns the renamed file details or null. For example: {"cwd":null, "files":[{"name":"my folder", "type":"Directory","size":0,"dateModified":"4/6/2016 3:05:17 PM","hasChild":true,"isFile":false,"filterPath":null,"permission":{"Copy":true, "Download":false,"Edit":true,"EditContents":true,"Read":true,"Upload":true} }], "details":null, "error":null}	This method helps to rename the file/folder, which is available in given path.(here files represents details of the renamed items)
GetDetails	string path, string[] names, IEnumerable<object> selectedItems	It is in JSON data with key name as details. For example: {"cwd":null, "files":null, "details":[{"Name":"green.jpg","Location":"C:\\Users\\XXX\\AppData\\Local\\\Syncfusion\\EssentialStudio\\14.1.0.40\\MVC\\Samples\\web\\FileContent \\Documents\\green.jpg","Type":".jpg","Size":89894,"Created":"3/30/2016 9:19:25 PM","Modified":"3/23/2016 6:05:32 PM","Permission":{"Copy":true,"Download":true,"Edit":true,"EditContents":true,"Read":true,"Upload":true}}], "error":null}	This method used to get the details of the specified

			file or directory. (here details represents details of the given path which is need to be showcased to user)
Download	string path, string[] names, IEnumerable<object> selectedItems	Void	This method helps to download the specified files in the given path.

Upload	IEnumerable<System.Web.HttpPostedFileBase> files, string path, IEnumerable<object> selectedItems	Void	This method helps to upload the specified files to given path.
Search	string path, string filter, string searchString, bool caseSensitive, IEnumerable<object> selectedItems	It returns JSON data with key name as files and JSON fields need to be with following field names 'name, isFile, hasChild, permission'. For example: {"cwd":{"name":"FileContent","type":"Directory","size":0,"dateModified":"3/30/2016 9:19:27 PM","hasChild":true,"isFile":false,"filterPath":null,"permission":{"Copy":false,"Download":false,"Edit":false,"EditContents":false,"Read":true,"Upload":false}}, "files":[{"name":"bird.jpg","type":"File","size":102182,"dateModified":"1/9/2016 6:48:42 AM","hasChild":false,"isFile":true,"filterPath":null,"permission":{"Copy":false,"Download":false,"Edit":false,"EditContents":true,"Read":true,"Upload":false}}, {"name":"sea.jpg","type":"File","size":97145,"dateModified":"1/9/2016 6:48:42 AM","hasChild":false,"isFile":true,"filterPath":null,"permission":{"Copy":false,"Download":false,"Edit":false,"EditContents":false,"Read":true,"Upload":true}}], "details":null, "error":null}	It used to search all the matched files and sub-folders in the given folder path also it filters the specified files using it

			type s.

Note: If needed, customer can also add additional data along with existing properties using **FileExplorerDirectoryContent** class

Example

We can make a FileExplorer with access rule by doing following steps,

1. Add the following code example to the corresponding View page to render the FileExplorer.

CSHARP

```
@Html.EJ().FileExplorer("fileExplorer").Path("~/FileContent/").AjaxAction(@Url.Content("FileAccessDefault"))
```

2. Add the following code example to the corresponding controller page. The FileAccessDefault method triggers from AJAX request with specific ActionType value as parameter.

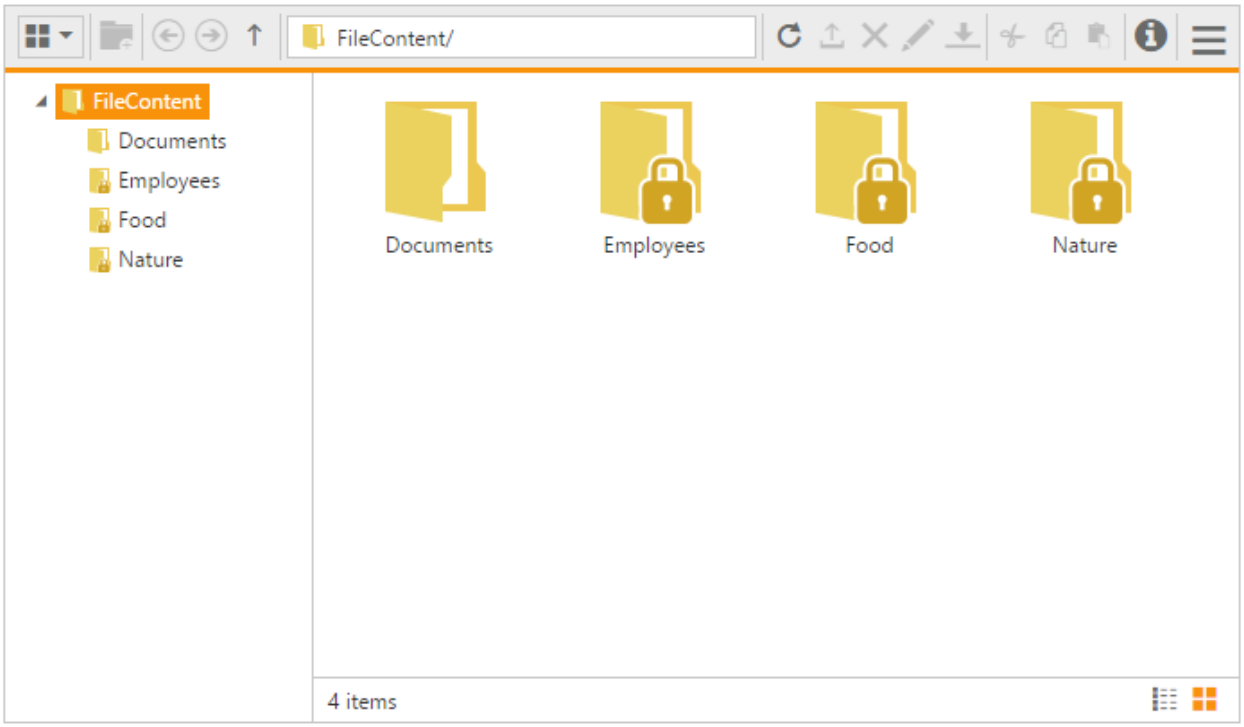
CSHARP

```
public ActionResult FileAccessDefault(FileExplorerParams args)
{
FileAccessOperations operation = new FileAccessOperations(GetRules());
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
case "Search":
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
}
return Json("");
}
public FileAccessInfo GetRules()
{
FileAccessInfo rules = new FileAccessInfo();
```

```
List<AccessRule> accessRules = new List<AccessRule> {
// For Default User
new AccessRule { Path = "*.*", Role = "Default User", Read =
Permission.Deny, Edit = Permission.Deny, Copy = Permission.Deny, Download =
Permission.Deny },
new AccessRule { Path = "*", Role = "Default User", Read = Permission.Deny,
Edit = Permission.Deny, Copy = Permission.Deny, EditContents =
Permission.Deny, Upload = Permission.Deny },
new AccessRule { Path = "", Role = "Default User", Read = Permission.Allow,
Edit = Permission.Deny, Copy = Permission.Deny, EditContents =
Permission.Deny, Upload = Permission.Deny },
// For Administrator
new AccessRule { Path = "*.*", Role = "Administrator", Read =
Permission.Allow, Edit = Permission.Allow, Copy = Permission.Allow, Download
= Permission.Allow },
new AccessRule { Path = "*", Role = "Administrator", Read =
Permission.Allow, Edit = Permission.Allow, Copy = Permission.Allow,
EditContents = Permission.Allow, Upload = Permission.Allow },
new AccessRule { Path = "", Role = "Administrator", Read = Permission.Allow,
Edit = Permission.Deny, Copy = Permission.Allow, EditContents =
Permission.Allow, Upload = Permission.Allow },
// For Document Manager
new AccessRule { Path = "*.*", Role = "Document Manager", Read =
Permission.Deny, Edit = Permission.Deny, Copy = Permission.Deny, Download =
Permission.Deny },
new AccessRule { Path = "Documents/*.*", Role = "Document Manager", Read =
Permission.Allow, Edit = Permission.Allow, Copy = Permission.Allow, Download
= Permission.Allow },
new AccessRule { Path = "*", Role = "Document Manager", Read =
Permission.Deny, Edit = Permission.Deny, Copy = Permission.Deny,
EditContents = Permission.Deny, Upload = Permission.Deny },
new AccessRule { Path = "", Role = "Document Manager", Read =
Permission.Allow, Edit = Permission.Deny, Copy = Permission.Deny,
EditContents = Permission.Deny, Upload = Permission.Deny },
new AccessRule { Path = "Documents", Role = "Document Manager", Read =
Permission.Allow, Edit = Permission.Deny, Copy = Permission.Allow,
EditContents = Permission.Allow, Upload = Permission.Allow },
new AccessRule { Path = "Documents/*", Role = "Document Manager", Read =
Permission.Allow, Edit = Permission.Allow, Copy = Permission.Allow,
EditContents = Permission.Allow, Upload = Permission.Allow },
};
rules.Rules = accessRules;
rules.Role = User.Identity.Name; // Specify the current user role
rules.RootPath = "~/FileContent/";
return rules;
}
```

The following screenshot displays the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



FileExplorer with Access Rule

Toolbar

The toolbar element having the number of tools, and each tool can be configurable.

Toolbar items

The toolbar having the list of items to perform various operations and grouped into some categorizes.

From below you can see the available toolbar items and its categorization:

RAZOR

```
@*all tools grouped under following categories*@
@{
List<String> layout = new List<string>() {
"Layout"
};
List<String> creation = new List<string>() {
"NewFolder"
};
List<String> navigation = new List<string>() {
"Back",
"Forward",
"Upward"
};
List<String> addressBar = new List<string>() {
"Addressbar"
};
List<String> editing = new List<string>() {
"Refresh",
"Upload",
"Delete",
"Rename",
```

```
"Download"
};
List<String> copyPaste = new List<string>() {
"Cut",
"Copy",
"Paste"
};
List<String> getProperties = new List<string>() {
"Details"
};
List<String> searchBar = new List<string>() {
"Searchbar"
};
List<String> sortBy = new List<string>() {
"SortBy"
};
}
@*all tools categories are placed in the toolbar by the below order*@
@{
List<String> toolsList = new List<string>() {
"layout",
"creation",
"navigation",
"addressBar",
"editing",
"copyPaste",
"getProperties",
"searchBar",
"sortBy"
};
}
```

The following table explains the functionality of each toolbar item:

Tool name	Description
Layout	In FileExplorer, files have been displayed in 3 types of views “Grid”, “Tile” and “Large Icons”. Using this tool, you can change the view type from one to another.
NewFolder	It creates a new folder on the current directory. While click on the NewFolder item, the dialog displays to get the folder name. Based on the user input, FileExplorer creates new folder on the current directory.Also createFolder event will be triggered when new folder is created successfully in the file system.
Back	It navigates to the previous directory from the user history. When the backward history is not available when it is disabled state.
Forward	It navigates to the next directory from the user history. When the forward history is not available when it is disable state.
Upward	It navigates to the parent directory of the current folder.

Copyright © 2001 - 2020 Syncfusion Inc.

Address bar	The “address bar” is the textbox which displays the path of the current directory, as a series of its parent directory separated by slash (“/”). And the address bar is an editable area, so you can enter any directory’s path to a quick navigation.
Refresh	It refreshes the current directory.
Upload	It uploads a file or list of files into the current directory. And you can customize the upload configurations, for details check here.
Delete	It delete the current selected file or folder. The delete icon is enable state if you select any file or folder. If you selected the multiple files, it deletes all the selected items.
Rename	This is used to rename the current selected file or folder. The rename icon is enable state if you select any file or folder. Even if you selected multiple files, it renames the last selected file only.
Download	It downloads the selected files. The download icon is enable state if you select any file or folder. The beforeDownload event will be triggered before the files are downloaded. If you select multiple files, it downloads all the files in a zip format.
Cut	It makes the copy of the selected files or folders into the clipboard. When the user paste the files in any location, the files are removed from the source location.The cut event will be triggered when files or folders are removed from the source.
Copy	It makes the copy of the selected files or folders into the clipboard. When the user paste the files, the copy of the files are pasted in the target location.The copy event will be triggered when file or folder is copied.
Paste	It pastes the files from the clipboard into the currently selected folder. Note: Only when the files are copied into the clipboard it is enabled.The paste event will be triggered when file or folder is pasted.
Details	It displays the details of the current selected file or folder.
Search bar	The Search bar is the textbox which is used to search the files from the current directory. It list the files based on the user search.

Copyright © 2001 - 2020 Syncfusion Inc.

	The search behavior of the “search bar” can be customize, for details check here.
Sort by	It's used to sorting the files from the current directory.The sorting can be done based on the columns available from grid,in both ascending and descending order.

Toolbar Visibility

The visibility of the toolbar can be customized through the “ShowToolbar” property. By disabling this property you can remove the toolbar from FileExplorer. Also you can remove the particular toolbar item by using removeToolbarItem method.

In the view page, add FileExplorer helper and specify “ShowToolbar” as false

RAZOR

```
@*hides the toolbar*@
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.ShowToolbar(false)
)
```

Toolbar Configuration

As you can see the available toolbar items from here. From the list of available items, you can configure and group your required toolbar items only through the “Tools” property. And also you can arrange the toolbar items by the “ToolsList” property.

In the view page, create the tools list and add FileExplorer helper as shown below

RAZOR

```
@{
List<String> toolsList = new List<string>() {
"creation",
"navigation",
"addressBar",
"copyPaste",
"searchBar"
};
List<String> creation = new List<string>() {
"NewFolder"
};
List<String> navigation = new List<string>() {
"Back",
"Forward",
"Upward"
};
List<String> addressBar = new List<string>() {
"Addressbar"
};
List<String> copyPaste = new List<string>() {
"Cut",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
"Copy",
"Paste"
};
List<String> searchBar = new List<string>() {
"Searchbar"
};
}
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.Tools(tools =>
tools.Creation(creation)
.AddressBar(addressBar)
.Navigation(navigation)
.CopyPaste(copyPaste)
.SearchBar(searchBar)
)
.ToolsList(toolsList)
)
```

Search bar

The Search bar can be customize through the "FilterSettings" property. By default the search doesn't consider the case sensitivity, and the search works based on "FilterType".

The FileExplorer allows the following filter types in the search functionality.

- "FilterOperator.StartsWith" – Supports to search text with starts with
- "FilterOperator.Contains" – Supports to search text with contains with
- "FilterOperator.EndsWith" – Supports to search text with ends with

In the view page, you can configure the filter type with enabling case sensitivity like below:

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.FilterSettings(settings =>
//it changes the search filter type as "startsWith"
settings.FilterType(FilterOperator.StartsWith)
//it enables the case sensitive search
.CaseSensitiveSearch(true)
)
)
```

Custom Tool in Toolbar

From the toolbar items you can see the list of built-in tools to perform the operations. Along with this built-in tools, you can add your custom tool with the custom functionality.

You can find an online demo sample of FileExplorer with custom tool from here.

In the view page, add FileExplorer helper and specify custom tool as shown below

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{
// denotes the order of the tools categories
List<String> toolsList = new List<string>() {
"creation",
"navigation",
"addressBar",
"copyPaste",
"searchBar",
"customTool"
};
}
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.ToolsList(toolsList)
//defines the tool items for customTool category
.Tools(tool =>
tool.CustomTool(custom =>
custom.Name("Help")
.Tooltip("Help")
.Css("e-fileExplorer-toolbar-icon Help")
.Action("dialogOpen").Add()
))
)
<script>
//click handler of custom tool
function dialogOpen() {
alert("custom tool item clicked");
}
</script>
@*Define the CSS that needs to apply for the custom tool.*@
<style class="cssStyles">
.e-fileExplorer-toolbar-icon {
height: 22px;
width: 22px;
font-family: 'ej-webfont';
font-size: 18px;
margin-top: 2px;
text-align: center;
}
.e-fileExplorer-toolbar-icon.Help:before {
content: "\e72b";
}
</style>
```

Enable / Disable the Toolbar Item

Each toolbar item can be enabled or disabled through the below client-side public methods.

- enableToolbarItem
- disableToolbarItem

These methods accepts the tool name as the parameter. It also allows the parameter as tool item index or the jQuery object of the tool item.

Copyright © 2001 - 2020 Syncfusion Inc.

JAVASCRIPT

```
$(function () {
var fileExpObj = $("#fileExplorer").data("ejFileExplorer");
// this disables the NewFolder item
fileExpObj.disableToolbarItem("NewFolder");
// this disables the Layout item (since index of Layout is 0)
fileExpObj.disableToolbarItem(0);
// this enables the NewFolder item
fileExpObj.enableToolbarItem("NewFolder");
});
```

Enable / Disable the custom added tool in Toolbar Item

If you want to enable / disable the custom added tool in toolbar, you need to pass the corresponding li elements of custom added tool in enableToolbarItem / disableToolbarItem method of FileExplorer. Since we have consider this custom tool as a object type.

JAVASCRIPT

```
var fileExpObj = $("#fileExplorer").data("ejFileExplorer");
//tool is a cssClass of FileExplorer
// this disables the custom tool item
var li = $(".tool").find(".Help").closest('li');
fileExpObj.disableToolbarItem(li);
// this enables the custom tool item
fileExpObj.enableToolbarItem(li);
```

Customizing the Upload Functionality

FileExplorer helps you to upload the file using "Upload" component. File upload can be done through the toolbar item or context menu item. The "UploadSettings" property is used to configure the upload functionalities.

This property has the below sub properties with the default values:

RAZOR

```
UploadSettings(settings =>
settings.AllowMultipleFile(true)
.FileSize(31457280)
.AutoUpload(false)
)
```

AllowMultipleFile: This property used to control the behavior of multiple files upload and this was enabled by default so you can upload multiple files at a time. If you disable this "AllowMultipleFile" property then you can upload only one file at a time.

MaxFileSize: The property limits the maximum file size to upload. It accepts the value in bytes.

If you want to upload more than 4 MB files in FileExplorer, you should specify the **maxRequestLength** attribute in webconfig file and specify the **MaxFileSize** property to upload it. Please check the below code block.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
<httpRuntime targetFramework="4.6"
maxRequestLength="2147483647"executionTimeout="1600"
requestLengthDiskThreshold="2147483647" />
<system.webServer>
<security>
<requestFiltering>
<requestLimits maxAllowedContentLength="3221225472" />
</requestFiltering>
</security>
…
</system.webServer>
```

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/Content/images/FileExplorer/")
.AjaxAction(@Url.Content("/FileExplorer/FileActionDefault"))
.UploadSettings(settings =>
settings.MaxFileSize(3221225472)
)
)
```

To know more details, please check the below reference link and sample.

Reference Link

Sample

AutoUpload: when you enable this property, the upload action performed automatically after select the files. When you disable this property, it shows a confirmation dialog with the selected file details and perform the upload action on press the “upload” button.

During upload process following events will be triggered, beforeUploadSend, beforeUploadDialogOpen, beforeUpload, uploadError, uploadSuccess and uploadComplete. You can customize the upload settings with these events.

In the view page, add FileExplorer helper and specify the upload settings as shown below

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.UploadSettings(settings =>
// it disables the multi file upload functionality
settings.AllowMultipleFile(false)
// it enables the auto upload functionality
.AutoUpload(true)
)
)
```

Context Menu

The context-menu has list of items which helps to perform FileExplorer operations, and it appears based on the target such as file or folder.

Copyright © 2001 - 2020 Syncfusion Inc.

Context menu items

The below table shows the context menu items corresponding to the location where it is opened:

Context menu location	Context menu items	Screenshot
While right click on tree view nodes (from navigation pane)	*New folder* Upload *Delete* Rename *Cut* Copy *Paste* Get info	
While right click on File / Folder	*Open* Download *Upload* Delete *Rename* Cut *Copy* Paste *Get info* Open folder location	
While right click on layout (content pane)	*Refresh* Paste *Sort by* New folder *UploadÂ* Get infoÂ	

The below table explains the behavior of each context menu item:

Open	It opens the selected folder. When an image file selected it opens the preview of the image. For the remaining files this option becomes disabled.
Download	It downloads the selected file. When a file or number of files selected at that time only download option enabled. If multiple files selected then it downloads all the files in a zip format.
Cut	It makes the copy of the selected files or folders into the clipboard. When the user paste the files in any location, the files are removed from the source location.

Copyright © 2001 - 2020 Syncfusion Inc.

Copy	It makes the copy of the selected files or folders into the clipboard. When the user paste the files, the copy of the files only pasted in the target location.
Paste	It paste the files from the clipboard into the current selected folder. Note that when the files are copied into the clipboard at that time only it enabled.
Delete	It deletes the current selected file or folder. When you select any file or folder at that time only this option gets enabled. If multiple files selected then it deletes all the selected items.
Rename	This is used to rename the current selected file or folder. When you select any file or folder at that time only this option gets enabled. Even multiple files selected it renames the single file only.
New Folder	It creates a new folder on the current directory. While click on the â€œNewFolderâ€• item a dialog appears to get the folder name. Based on the user input, a new folder create on the current directory.
Upload	It uploads a file or list of files into the current directory.
Get info	It displays the details of the current selected file or folder.
Sort By	It's used to sorting the files and folders from the current directory.The sorting can be done based on the columns available from grid,in both ascending and descending order.
Open folder location	It helps to open a file location from the filtered list.

Context menu Visibility

The presence of the context menu can be controlled by the “ShowContextMenu” property. This was enabled by default, and by disabling this property you can prevent our built-in context menu.

In the view page, add FileExplorer helper and specify the context menu visibility as false.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.ShowContextMenu(false)
)
```

Enable / Disable the Context menu Item

The context menu items can be enabled or disabled through the client side public methods. It enables or disables the item from all the context menu where it is present.

Copyright © 2001 - 2020 Syncfusion Inc.

For example, if you disable the “Upload” item, it disables in all places wherever it appears such as open the context menu on “TreeView”, open on file/folder, and open on layout.

- enableMenuItem
- disableMenuItem

These methods only accepts the context menu item name as the parameter.

JAVASCRIPT

```
$(function () {
var fileExpObj = $("#fileExplorer").data("ejFileExplorer");
// this disables the New Folder item
fileExpObj.disableMenuItem("New folder");
// this disables the Download item
fileExpObj.disableMenuItem("Download");
});
```

Context Menu Customization

You can customize the ContextMenu of FileExplorer control by using ContextMenuSettings property.

You can add your own context menu items and its action along with default context menu items of FileExplorer control. You can also remove the default context menu items in FileExplorer control.

To add/remove/re-arrange context menu items, you need to use ContextMenuSettings.items property and to bind required actions for newly added menu items and add sub menu items, use ContextMenuSettings.CustomMenuFields property.

Add the following code example to the corresponding controller page.

C#

```
public ActionResult FileActionContextMenu(FileExplorerParams args)
{
FileExplorerOperations operation = new FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
break;
case "Search":
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
}
return Json("");
}
```

In the view page, add FileExplorer helper and specify the **ContextMenuSettings** property.

RAZOR

```
@* removed the "NewFolder" item from NavigationPane ContextMenu *@
@{List<String> navigationBar = new List<string>() { "Upload", "|", "Delete",
"Rename", "|", "Cut", "Copy", "Paste", "|", "Getinfo" }; }
@* added the custom ContextMenu item (View) to Current working directory
ContextMenu *@
@{List<String> cwd = new List<string>() { "Refresh", "Paste", "|", "SortBy",
"View", "|", "NewFolder", "Upload", "|", "Getinfo" };}
@* removed "Upload" item from Selected files/ folder's ContextMenu *@
@{List<String> files = new List<string>() { "Open", "Download", "|",
"Delete", "Rename", "|", "Cut", "Copy", "Paste", "|",
"OpenFolderLocation", "Getinfo" }; }
@Html.EJ().FileExplorer("fileExplorer").Path("~/FileExplorerContent/").Conte
xtMenuSettings(
settings => settings.Items(
item => item.Navbar(navigationBar).Cwd(cwd).Files(files)
).CustomMenuFields(fields =>
{
fields.Add().Id("View").Text("View by").SpriteCssClass("custom-
grid").Child(child =>
{
child.Add().Id("tile").Text("Tile view").Action("onLayout");
child.Add().Id("grid").Text("Grid view").Action("onLayout");
child.Add().Id("largeIcons").Text("Large icons view").Action("onLayout");
});
})
).AjaxAction(@Url.Content("FileActionContextMenu")).ClientSideEvents(
e => e.MenuOpen("onMenuOpen").LayoutChange("onLayoutChange")
).Width("100%").IsResponsive(true).MinWidth("150px")
```

Icons of context menu items can be customized by overriding the default context menu item style. The following code example illustrates how to customize the icon of context menu items.

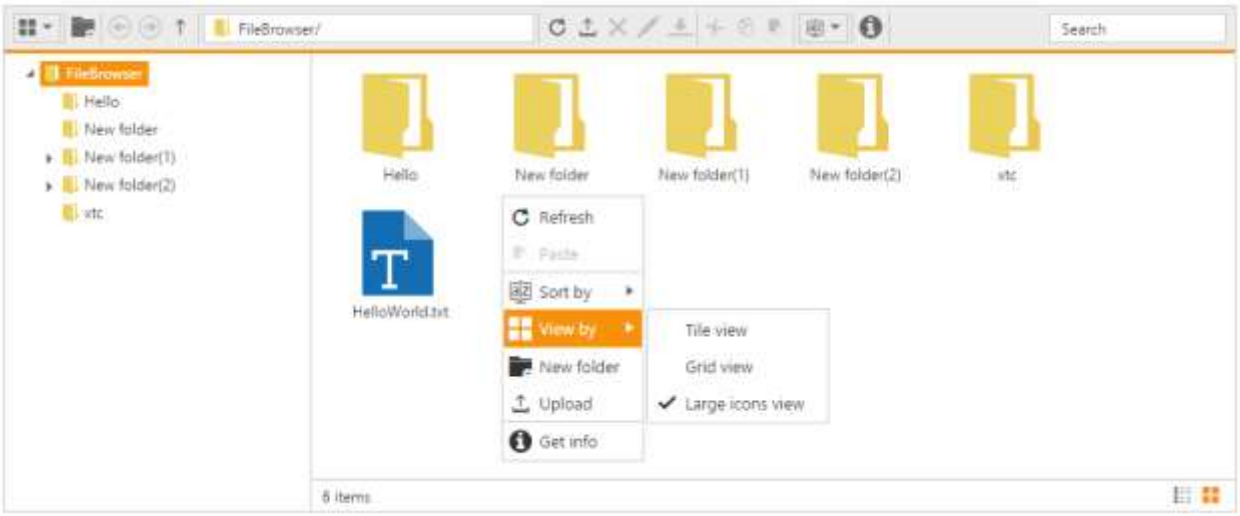
CSS

```
<style>
.fe-context-menu .custom-grid:before {
content: "\e7b9";
}
.fe-context-menu .custom-largeIcons:before {
content: "\e7bb";
}
.fe-context-menu .custom-tile:before {
content: "\e7be";
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</style>
```

The following screenshot displays the customization of context menu in FileExplorer control.



Context Menu Events

You would be notified with events when you try to open the context menu items (**MenuBeforeOpen**), after context menu items is opened (**MenuOpen**) and when you click the menu items (**MenuClick**). The following code example illustrates how to define those events.

RAZOR

```
@Html.EJ().FileExplorer("fileExplorer").Path("~/FileExplorerContent/").AjaxAction(@Url.Content("FileActionContextMenu")).ClientSideEvents(e => e.MenuBeforeOpen("menuBeforeOpen").MenuOpen("menuOpen").MenuClick("menuClick"))
```

JS

```
function menuBeforeOpen(args) {
//you add/remove the context menu items in run time
//do your custom action here.
args.dataSource.pop();
}
function menuOpen(args) {
//you can also identify which context menu is opened by
if (args.contextMenu == "cwd") {
//do your custom action here.
}
}
function menuClick(args) {
switch (args.text) {
case "largeIcons":
//do your custom action here.
break;
}
}
```

Customization

Dimension Customization

The dimension of the FileExplorer can be customized through “Height” and “Width” property. The dimension can be set in percentage (ex; width: “100 %”), so that the control inherits the width from the parent element.

In the view page, add FileExplorer helper with custom height and width

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.Height("300px")
.Width("900px")
)
```

Customizing the Navigation pane

The navigation pane contains the tree view element which displays all the folders from the filesystem in a hierarchical manner. This is useful to a quick navigation of any folder in the filesystem.

The visibility of the navigation pane can be controlled by the “ShowNavigationPane” property. By disabling this property, you can hide the navigation pane from FileExplorer.

In the view page, add FileExplorer helper and specify the navigation pane as false

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.ShowNavigationPane(false)
)
```

Customizing the Content pane

The content pane is the main part of the Fie explorer UI which displays all the files and folders from the filesystem. The content pane supports the following two types of layout views:

- Grid
- Tile
- Large Icons

The **grid view** displays the files and folders in a grid layout with the details in separate columns. By default the grid view having the four columns which displays the file name, type, date modified and size of the file. For more details about grid view customization, refer here.

The **tile view** display the files and folders like a small size icons in left side and file details in right. It allows the thumbnails for the image files so that you can view the tiny preview of all image files.

The **large icons view** display the files and folders like a large size icons with name. It allows the thumbnails for the image files so that you can view the tiny preview of all image files.

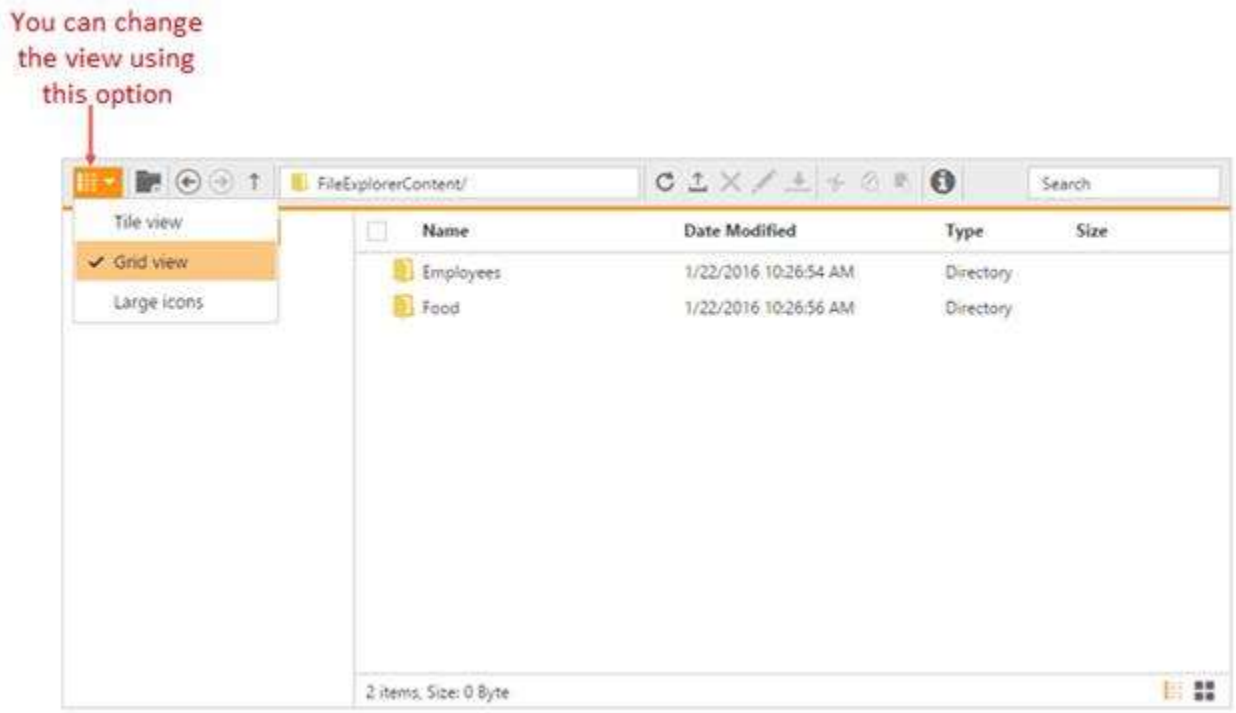
Changing the Layout views

Copyright © 2001 - 2020 Syncfusion Inc.

You can change the layout of current view by the switcher which displays at right-bottom of footer in the FileExplorer. By clicking the grid and large icons view buttons you can change the layout of current view.



You can change the layout using toolbar option. While clicking this tool, popup opens with different layout types. Here you can choose any options.



Also the layout views can be changed through the "Layout" property. The layoutChange event will be triggered whenever the layout view type is changed.

In the view page, add FileExplorer helper and specify the layout type as shown below

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
// while rendering it displays as tile view
.Layout(LayoutType.Tile)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Customizing the Grid view

By default sorting is enabled in grid view of FileExplorer, it helps you to sort each columns in ascending or descending order by pressing the corresponding column header. The sorting functionality can be disabled by setting "AllowSorting" property to false.

The behavior of the columns can be customized through the "Columns" property.

In the view page, add FileExplorer helper with custom grid settings

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
// under gridSettings you can customize the grid view functionalities
.GridSettings(settings =>
// disables the sorting functionality in grid view
settings.AllowSorting(true)
// it displays 3 columns only, like this you can display your desired
columns at here
.Column(column =>
{
column.Add()
.Field("name")
.HeaderText("Name")
.Width("150");
column.Add()
.Field("dateModified")
.HeaderText("Date Modified")
.Width("150");
column.Add()
.Field("size")
.HeaderText("Size")
.Width("90")
.TextAlign(TextAlign.Right)
.HeaderTextAlign(TextAlign.Left);
}
)
)
)
```

Footer Customization

The footer displays the details of the current selected files and folders, and the footer contains the switcher to change the layout view. The visibility of the footer can be customized by the "ShowFooter" property.

In the view page, add FileExplorer helper and hides the footer as shown below.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
// it hides the footer element
.ShowFooter(false)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Multiple Selection

The FileExplorer allows the user to select multiple files by enabling the "AllowMultiSelection" property. The multiple selection can be done by pressing the "**Ctrl"** key or "**Shift"** key. You can select all the files in the directory by pressing "**Ctrl + A**". The multiple selection is useful on copy pasting multiple files, deleting multiple files and downloading multiple files. In another way, multiple files can be selected by mouse down and drag over the desired files. In addition to that, additional files can be selected with the preselected file by holding the ctrl/shift key and drag the mouse over the files. Also, holding the ctrl/shift key and dragging over selected files will unselect the selected files.

The select event will be triggered when the items of FileExplorer control is selected.

Also unselect event will be triggered when the items of FileExplorer control is unselected.

Note: For selecting files by mouse down and drag set AllowMultiSelection property as true.

In the view page, add FileExplorer helper and specify the multi selection as true

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.AllowMultiSelection(true)
)
@*AllowMultiSelection property is true by default*@
```

Checkbox option

You can enable or disable the checkbox using "ShowCheckbox" API of FileExplorer

In the view page, add FileExplorer helper and disable the checkbox as shown below

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.ShowCheckbox(false)
)
```

Resizing

The FileExplorer has the resize support through the resize handle which appears at right-bottom corner of footer. By clicking the resize handle the user can resize the FileExplorer through the UI. While resizing the dimension of the FileExplorer is automatically adjusted.

The resize behavior can be enabled through the "EnableResize" property.

In the view page, add FileExplorer helper with resizable option as shown below

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
// it enables the resize functionality
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// and a resize handle added in the Footer element
.EnableResize(true)
)
```

Responsiveness

By enabling the "IsResponsive" property, you can make the FileExplorer as responsive. While resizing the FileExplorer component, the inner content and toolbar items are automatically adjusted to equalize the size. The toolbar items are displayed within Dropdown on enabling the responsive. Otherwise it floated to the next line to equalize the space.

In the view page, add FileExplorer helper with responsive option

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
// it enables the resize functionality
// and a resize handle added in the Footer element
.EnableResize(true)
// it enables the control responsiveness
.IsResponsive(true)
)
```

Restriction on Resize

You can restrict the dimension of the FileExplorer by setting the "MinHeight", "MaxHeight", "MinWidth" and "MaxWidth" properties while resizing the FileExplorer.

In the view page, add FileExplorer helper and set resize option within particular region.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.EnableResize(true)
.IsResponsive(true)
// restricts the dimensions for height and width
.MinHeight("200px")
.MaxHeight("400px")
.MinWidth("300px")
.MaxWidth("1200px")
)
```

Drag and Drop Support

The FileExplorer allows files to be moved from one folder to another by using drag and drop. It also supports uploading a file by dragging it from Windows Explorer to a folder in the FileExplorer control.

You can enable or disable this support by using "**AllowDragAndDrop**" API of FileExplorer.

The dragStart, drag, dragStop and drop events occur in the mentioned order when a drag and drop operation is performed.

In the view page, add "FileExplorer" helper and specify the drag and drop option as specified below.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Layout(LayoutType.Tile)
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileAction"))
.AllowDragAndDrop(true)
)
```

Thumbnail Image Compression Support

The “FileExplorer” allows thumbnail images to be compressed on the server side and loaded into the “FileExplorer” layout to improve performance when working with large size images.

You can enable this option using “**EnableThumbnailCompress**” API of FileExplorer. For enabling this API, showThumbnail must be set to true in order to render thumbnail preview of images in Tile and LargeIcons layout.

The beforeGetImage event is triggered before loading a requested image from the server and getImage event is triggered after the requested image is loaded.

Refer following code block that will be helpful to you.

In the view page, add “FileExplorer” helper and specify the thumbnail image compress option as specified below.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Layout(LayoutType.Tile)
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionThumbnailCompress"))
.EnableThumbnailCompress(true)
)
```

In the controller page, specify the “GetImage” handling operation as shown below. This handling function is necessary to compress and load the images in “FileExplorer”, while “EnableThumbnailCompress” option has been enabled.

C#

```
public ActionResult FileActionThumbnailCompress(FileExplorerParams args)
{
FileExplorerOperations operation  = new FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
case "Search":
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
case "GetImage":
//Helps to reduce thumbnail image size before loading it into FileExplorer
operation.GetImage(args.Path, args.CanCompress);
break;
}
return Json("");
}
```

Access shared folder using FileExplorer

Using “FileExplorer”, you can manage the files that are available in a shared folder of another system which is connected through LAN. Refer following steps, here you will see the details about accessing the shared folder with FileExplorer.

First you have to specify the shared folder path in any of following formats.

1. “\\Server\SharedFolder”
2. “\\Server”
3. “//Server/SharedFolder”
4. “//Server”

In the view page, add “FileExplorer” helper and specify the shared path as following format.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Layout(LayoutType.Tile)
.Path(@"\\Server\SharedFolder")
.AjaxAction(@Url.Content("FileActionDefault")))
```

In the controller page, specify the file handling operation as shown below. This handling functions are necessary to perform file operations in FileExplorer.

C#

```
public ActionResult FileActionDefault(FileExplorerParams args)
{
FileExplorerOperations operation = new FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
case "Search":
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
}
return Json("");
}
```

Authentication problem in shared folder

If shared folder is restricted with authentication, that time you are not able to access this folder using our FileExplorer. Here you will get an exception like "**System.UnauthorizedAccessException, Please add your windows credential details to open '\\server\'**". In order to solve this problem, you have to add the shared folder credential details in the windows credential manager that is available in your service hosted machine.



Localization

The FileExplorer control provides the translations for all static texts and messages. By default the FileExplorer uses the US English (en-US) language, and it having the built-in values of all texts with the corresponding key.

The below table lists out the keys with corresponding texts in English culture.

JAVASCRIPT

```
ej.FileExplorer.Locale["en-US"] = {
```

```
Back: "Backward",
Forward: "Forward",
Upward:"Upward",
Refresh: "Refresh",
Addressbar: "Address bar",
Upload: "Upload",
Rename: "Rename",
Delete: "Delete",
Download: "Download",
Error: "Error",
Cut: "Cut",
Copy: "Copy",
Paste: "Paste",
Details: "Details",
Searchbar: "Search bar",
Open: "Open",
Search: "Search",
EmptyResult: "No items match your search",
EmptyFolder: "This folder is empty",
NewFolder: "New folder",
Size: "Size",
RenameAlert: "Please enter new name",
NewFolderAlert: "Please enter new folder name",
ContextMenuOpen: "Open",
ContextMenuNewFolder: "New folder",
ContextMenuDelete: "Delete",
ContextMenuRename: "Rename",
ContextMenuUpload: "Upload",
ContextMenuDownload: "Download",
ContextMenuCut: "Cut",
ContextMenuCopy: "Copy",
ContextMenuPaste: "Paste",
ContextMenuGetinfo: "Get info",
ContextMenuRefresh: "Refresh",
ContextMenuOpenFolderLocation: "Open folder location",
Item: "item",
Items: "items",
Selected: "selected",
ErrorOnFolderCreation: "This destination already contains a folder named
'{0}'. Do you want to merge this folder content with already existing folder
'{0}'?",
InvalidFileName: "A file name can't contain any of the following characters:
\\/:*?\"<>|",
GeneralError: "Please see browser console window for more information",
ErrorPath: "FileExplorer can't find '{0}'. Check the spelling and try
again.",
ReplaceAlert: "File named '{0}' already exists. Replace old file with new
one?",
DuplicateAlert: "There is already a file with the same name '{0}'. Do you
want to create file with duplicate name",
DuplicateFileCreation: "There is already a file with the same name in this
location. Do you want to rename '{0}' to '{1}'?",
DeleteFolder: " Are you sure you want to delete ",
DeleteMultipleFolder: "Are you sure you want to delete these {0} items?",
CancelPasteAction: "The destination folder is a subfolder of source
folder.",
OkButton: "OK",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
CancelButton: "Cancel",
YesToAllButton: "Yes to all",
NoToAllButton: "No to all",
YesButton: "Yes",
NoButton: "No",
SkipButton:"Skip",
Grid: "Grid view",
Tile: "Tile view",
LargeIcons: "Large icons",
Name: "Name",
Location: "Location",
Type: "Item type",
Layout:"Layout",
Created: "Created",
Accessed: "Accessed",
Modified: "Modified",
Permission: "Permission",
DialogCloseToolTip: "Close",
UploadSettings: {
buttonText: {
upload: "Upload",
browse: "Browse",
cancel: "Cancel",
close: "Close"
},
dialogText: {
title: "Upload Box",
name: "Name",
size: "Size",
status: "Status"
},
dropAreaText: "Drop files or click to upload",
filedetail: "The selected file size is too large. Please select a file
within the valid size.",
denyError: "Files with #Extension extensions are not allowed.",
allowError: "Only files with #Extension extensions are allowed.",
cancelToolTip: "Cancel",
removeToolTip: "Remove",
retryToolTip: "Retry",
completedToolTip: "Completed",
failedToolTip: "Failed",
closeToolTip: "Close"
}
};
```

Change the Culture

The language of the FileExplorer can be changed by the "Locale" property. This property allows the culture code of the country as input. The culture code has to specify in the standard format as **[Language Code]-[County/Region Code]**.

For example the culture code for German is "**de-DE**".

You can customize built-in text and messages based on your culture. The below example shows the usage of FileExplorer with German (de-DE) culture.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/resources/")
.AjaxAction(@Url.Content("FileActionDefault"))
.Width("900")
.Locale("de-DE")
.GridSettings(settings =>
settings.Column(column =>
{
column.Add()
.Field("name")
.HeaderText("Name")
.Width("150");
column.Add()
.Field("dateModified")
.HeaderText("Date Modified")
.Width("150");
column.Add()
.Field("type")
.HeaderText("Art")
.Width("150");
column.Add()
.Field("size")
.HeaderText("Size")
.Width("90")
.TextAlign(TextAlign.Right)
.HeaderTextAlign(TextAlign.Left);
}
)
)
)
```

JAVASCRIPT

```
ej.FileExplorer.Locale["de-DE"] = {
Back: "rückwärts",
Forward: "Nach Vorne",
Upward: "nach oben",
Refresh: "erfrischen",
Addressbar: "Adressleiste",
Upload: "hochladen",
Rename: "umbenennen",
Delete: "löschen",
Download: "herunterladen",
Error: "Fehler",
Cut: "Schnitt",
Copy: "Kopie",
Paste: "kleben",
Details: "Einzelheiten",
Searchbar: "Searchbar",
Open: "geöffnet",
Search: "Suche",
EmptyResult: "Keine Artikel entsprechen Ihrer Suche nach",
EmptyFolder:"Dieser Ordner ist leer",
NewFolder: "neuer Ordner",
SelectedFileUrl: "Web-Adresse",
SelectedFileName: "Titel",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ImageWidth: "Breite",
ImageHeight: "Höhe",
Insert: "Insert",
Cancel: "Rückgängig Machen",
RenameAlert: "Bitte geben Sie neuen Namen",
NewFolderAlert: "Geben Sie den neuen Ordnernamen ein",
ContextMenuOpen: "geöffnet",
ContextMenuNewFolder: "Neuer Ordner",
ContextMenuDelete: "löschen",
ContextMenuRename: "umbenennen",
ContextMenuUpload: "hochladen",
ContextMenuDownload: "Herunterladen",
ContextMenuCut: "Schnitt",
ContextMenuCopy: "Kopie",
ContextMenuPaste: "kleben",
ContextMenuGetinfo: "Ausführliche Infos",
ContextMenuRefresh: "erfrischen",
ContextMenuOpenFolderLocation: "Öffnen Sie Ordner",
Item: " Artikel",
Items: " Angebote",
Selected: "ausgewählt",
Permission: "Erlaubnis",
OkButton: "Ok",
CancelButton: "Rückgängig machen",
YesToAllButton: "Ja zu allem",
NoToAllButton: "Nein, alle",
YesButton: "Ja",
NoButton: "Unterlassen Sie",
Size: "Größe",
Grid: "Gitter Ansicht",
Tile: "Kachelansicht",
LargeIcons: "Große Icons",
Layout: "Gestaltung",
SkipButton: "Überspringen",
ErrorOnFolderCreation: "Dieses Ziel ist bereits ein Ordner mit dem Namen
'{0}'. Möchten Sie diesen Ordner Inhalte mit bereits vorhandenen Ordner
'{0}' zusammenführen möchten?",
InvalidFileName: "Ein Dateiname darf keines der folgenden Zeichen enthalten:
\\/:*?\"<>|",
GeneralError: "Bitte beachten Sie Browser Konsolenfenster für weitere
Informationen",
ErrorPath: "Fileexplorer kann nicht finden '{0}'. Überprüfen Sie die
Schreibweise und versuchen Sie es erneut.",
ReplaceAlert: "Datei mit dem Namen '{0}' ist bereits vorhanden. Ersetzen Sie
alte Datei durch eine neue?",
DuplicateAlert: "Es gibt bereits eine Datei mit dem gleichen Namen '{0}'.
Möchten Sie diese Datei mit doppelten Namen erstellen möchten",
DuplicateFileCreation: "Es gibt bereits eine Datei mit dem gleichen Namen in
diesem Ort. Möchten Sie umbenennen '{0}' bis '{1}' suchen?",
DeleteFolder: " Sind Sie sicher, dass Sie löschen möchten ",
DeleteMultipleFolder: "Sind Sie sicher, dass Sie diese {0} Einträge
löschen?",
CancelPasteAction: "Der Zielordner ist ein Unterordner des Quellordners.",
Name: "Namen",
Location: "Ort",
Type: "Gegenstandsart",
Created: "Erstellt",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Modified: "geändert",
DialogCloseToolTip: "schließen",
UploadSettings: {
buttonText: {
upload: "hochladen",
browse: "blättern",
cancel: "Rückgängig Machen",
close: "schließen"
},
dialogText: {
title: "hochladen Box",
name: "Name",
size: "Größe",
status: "Status"
},
cancelToolTip: "stornieren",
removeToolTip: "entfernen",
retryToolTip: "wiederholen",
completedToolTip: "fertiggestellt",
failedToolTip: "fehlgeschlagen",
closeToolTip: "schließen"
}
};
```

Keyboard Navigation

The FileExplorer control can be fully accessible with the help of keyboard shortcuts and it can be allowed by specifying allowKeyboardNavigation as true. The below table shows the list of keyboard shortcuts to access the FileExplorer control.

Note: Set focus on FileExplorer component to access the keyboard shortcuts.

Key	Description
Ctrl + Shift + 1	Focuses the Toolbar
Ctrl + Shift + 2	Focuses the Tree view
Ctrl + Shift + 3	Focuses the Splitter
Ctrl + Shift + 4	Focuses the Grid view
Ctrl + Shift + 5	Focuses the Tile view
Ctrl + Shift + 6	Focuses the Large icons view
Ctrl + Shift + 7	Focuses the Status bar
Right	Focusing next item
Left	Focusing previous item

Copyright © 2001 - 2020 Syncfusion Inc.

Enter	Selection the focused item
Esc	Closes the opened dialog
F2	Renames the selected item
Ctrl + A	Selects the all items
Alt + N	Creates "New folder" dialog
Ctrl + U	Opens Upload dialog
F5	Refresh the content
Alt + Enter	Get the details of selected item
Ctrl + X	To cut the selected items
Ctrl + C	To copy the selected items
Ctrl + V	Pastes the copied items
Delete	Deletes the selected items
Shift + F10	Opens the Context Menu

Virtual Scrolling

Virtual Scrolling is introduced to provide support for loading files and folders on demand in order to improve performance in FileExplorer when large amount of files are present.

Virtual Scrolling can be enabled by using VirtualItemCount API. It specifies the total files to be loaded initially, and on each scroll the next set of files are loaded. For eg- If VirtualItemCount value is given as ”20”, then 20 items are loaded initially and on each scroll the next/previous 20 items are loaded. If the value of VirtualItemCount is 0, virtual scrolling will be disabled.

Note: Default value of `VirtualItemCount` is 0.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.VirtualItemCount(40)
)
```

How To

File handling operations

In controller page, we have specified some built-in classes at "**FileActionDefault**" action method. It helps to perform the server side operation of FileExplorer and you can find the details about these classes at below.

When you have made AJAX request on client-side, “FileActionDefault” method is triggered in controller part. Here received AJAX data will be stored in “FileExplorerParams” parameter. This “FileActionDefault” method finds out the specific operations using the “ActionType” property and calls the corresponding built-in methods of “FileExplorerOperations” class.

BasicFileOperations class

“BasicFileOperations” is an abstract class and it is useful for handling file operations in server end. By inheriting this class, easily you can implement a new custom class for handling file operations in server end. Refer class reference of “BasicFileOperations”

FileExplorerOperations class

This class is useful for handling file operations in server end. This class inherits the “BasicFileOperations” class and its abstract methods has been implemented here for managing files in **underlying machine’s physical file system**. Refer class reference of “FileExplorerOperations”

FileExplorerParams class

“FileExplorerParams” class is used to receive the AJAX data in server side. In server side, we have a common method to handle all the AJAX requests, which is raised from client part of FileExplorer. This method contains “FileExplorerParams” object as a parameter and it is used to store the AJAX request data that will be useful for handling file operations. Refer class reference of “FileExplorerParams”

Note: Here “ActionType” specifies following operations such as “Read”, “CreateFolder”, “Paste”, “Remove”, “Rename”, “GetDetails”, “Download”, “Upload”, “Search”. Following section contains the details about each operation.

Customizing AJAX handling functions

In FileExplorer, server side functionalities are necessary to handle the AJAX request of FileExplorer. Here “Syncfusion.EJ” assembly contains built-in “BasicFileOperations” abstract class and implementation of “FileExplorerOperations” class for providing the content to FileExplorer and handling AJAX requests. If is it necessary, you can customize the server side functionalities of FileExplorer as per your requirement.

Override the existing Syncfusion.JavaScript.FileExplorerOperations class

Using “FileExplorerOperations” class, you can manage the underlying machine's physical file system with the help of FileExplorer control. Here you can override the necessary methods that is available in “**FileExplorerOperations**” by sub classing the existing “**FileExplorerOperations**” class.

Below code example shows how to override “**GetDetails**” method, which is available in **“FileExplorerOperations**”

C#

```
public class FileExplorerCustomOperations : FileExplorerOperations
{
FileExplorerOperations operation = new FileExplorerOperations();
CustomFileExplorerResponse customDetails = new CustomFileExplorerResponse();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public override object GetDetails(string path, string[] names,
IEnumerable<object> selectedItems = null)
{
FileExplorerResponse response = (FileExplorerResponse)
operation.GetDetails(path, names);
FileDetails fDetails = response.details.ElementAt(0) ;
CustomFileDetails cfDetails =new CustomFileDetails() ;
CustomFileDetails[] responseDetails = new CustomFileDetails[1];
cfDetails.Name= fDetails.Name;
cfDetails.Location= fDetails.Location;
cfDetails.Size= fDetails.Size;
cfDetails.Created= fDetails.Created;
cfDetails.IsFile = (fDetails.Type == "File Folder"? "false" : "true");
responseDetails[0] = cfDetails;
customDetails.details = responseDetails;
return customDetails;
}
}
public class CustomFileDetails
{
public string Name { get; set; }
public string Location { get; set; }
public long Size { get; set; }
public string Created { get; set; }
public string IsFile { get; set; }
}
public class CustomFileExplorerResponse
{
public IEnumerable<FileExplorerDirectoryContent> files { get; set; }
public IEnumerable<CustomFileDetails> details { get; set; }
public object error { get; set; }
}
```

Implement a new custom class to handle AJAX request and provide content for FileExplorer

When overriding the particular methods were not enough, you can create a new custom class using "BasicFileOperations" abstract class. It helps to provide different file sources to be used the content for FileExplorer control, such as database system, physical system or online storage system such as Azure.

Notes to Implementers

When you implement a derived class of "BasicFileOperations", you must provide implementations for following methods and its details has been given in this section

- Read
- CreateFolder
- Remove
- Rename
- Paste
- Upload
- Download
- GetDetails
- GetImage
- Search

Copyright © 2001 - 2020 Syncfusion Inc.

Abstract methods in BasicFileOperations class

Operation and Request Parameter	**Response data**	**Details**
Operation: Read **Request Parameter** **Name**: path **Type :** String **Description:**Specifies the path of selected folder **Name**: filter **Type :** String **Description:** Specifies the file types that need to be filter **Name**: selectedItems **Type :** IEnumerable<object> **Description:** It contains the details about selected folder	Response data should be in JSON format with key name as "**files** " and JSON fields should be with following field names *â€œname, isFile, hasChildâ€*. *For example:* {"**files**":[{"**name":"bird.jpg**","type":"File","size":102182,"dateModified":"1/9/2016 6:48:42 AM","**hasChild":false,"isFile":true**,"filterPath":null},{"**name":"sea.jpg**","type":"File","size":97145,"dateModified":"1/9/2016 6:48:42 AM","**hasChild":false,"isFile":true**,"filterPath":null}],"details":null,"error":null} Note: If needed, customer can also add additional data along with existing properties of â€œFileExplorerDirectoryContentâ€• class.	It used to get all immediate files and sub-folders of the given path and it returns the matched type of files only, which are specified in â€œfilterâ€• parameter.
Operation: CreateFolder **Request Parameter** **Name**: path **Type :** String **Description:**Specifies the path of selected folder **Name**: name **Type :** String **Description:**specifies the name of new folder **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about selected folder	Response data should be in JSON format with key name as "**files**".In that returning JSON, â€œ**name**â€• field is necessary. *For example:* {"files":[{"**name":"New folder**","type":"Directory","size":0,"dateModified":"2/25/2016 7:31:02 AM","hasChild":true,"isFile":false,"filterPath":null}],"details":null,"error":null}	It used to create a new folder in given path with specified name

Copyright © 2001 - 2020 Syncfusion Inc.

Operation: Remove **Request Parameter** **Name:**names **Type :** String[] **Description:**specifies the selected item names, which is going to be removed **Name**: path **Type :** String **Description:**specifies the parent folder path of selected items **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about selected items, which is going to be removed	Here you may return the removed file details or empty.	It helps to remove the specified items from given path.
Operation: Rename **Request Parameter** **Name**: path **Type :** String **Description:**specifies the parent folder path of renaming item **Name**: oldName **Type :** String **Description:**specifies the existing file name **Name**: newName **Type :** String **Description:**specifies the new file name **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains	Here you may return the renamed file details or empty.	This method helps to rename the file/folder, which is available in given path.

the details about selected items, which is going to be renamed **Name**: commonFiles **Type :** IEnumerable **Description:**It contains the details about already existing files, which contains same name and type as given in new file with same parent folder.		
Operation: Paste **Request Parameter** **Name**: sourceDir **Type :** String **Description:**specifies the path of source directory **Name**: backupDir **Type :** String **Description:**specifies the path of destination directory **Name**: names **Type :** String[] **Description:**Specifies the name of file/ folders, which are going to be pasted in destination folder. **Name**: option **Type :** String **Description:**specifies the operation type â€œmoveâ€• or â€œcopyâ€•. **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about	Here you may return the pasted file details or empty.	This method helps to copy or move files from one location to another location.

selected items, which is going to be pasted **Name**: commonFiles **Type :** IEnumerable **Description:**It contains the details about already existing files in destination folder, which contains the files in same name and type as same as newly copied files. **Name**: targetFolder **Type :** IEnumerable<object> **Description:**It contains the details about target folder, where the files are going to be pasted.		
Operation: Upload **Request Parameter** **Name**: files **Type :** IEnumerable<System.Web.HttpPostedFileBase> **Description:**Specifies the file details that is going to be uploaded **Name**: path **Type :** String **Description:**specifies the path of destination directory, where the files need to be uploaded **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about selected items, which is going to be uploaded.	Return type of this method is â€œvoidâ€•	This method helps to upload the specified files to given directory

Copyright © 2001 - 2020 Syncfusion Inc.

Operation: Download **Request Parameter** **Name**: path **Type :** String **Description:**Specifies the parent directory path of selected files, which is going to be download **Name**: names **Type :** String [] **Description:**specifies the name of files that is need to be downloaded **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about files, which is going to be downloaded.	Return type of this method is â€œvoidâ€•	This method helps to download the specified files.
Operation: GetDetails **Request Parameter** **Name**: path **Type :** String **Description:**Specifies the parent directory path of selected file **Name**: names **Type :** String [] **Description:**Specifies the name of files in order to get itâ€™s details **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the basic details about selected files	Response data should be in JSON format like below {details : [{CreationTime:"4/28/2015 9:44:32 AM", Extension:".png", Format:"Archive", FullName:"F:\All samples\FileExplorer_Custom\FileExplorerContent\human.png", LastAccessTime:"4/28/2015 9:44:32 AM", LastWriteTime:"3/31/2015 3:16:35 PM", Length:11059, Name:"human.png"}]} Note: Here you may add additional date fields along with existing JSON data using â€œFileDetailsâ€• class.	This method used to get the details of the specified file or directory.

Operation: GetImage **Request Parameter** **Name**: path **Type :** String **Description:**specifies the path of image file **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about selected image	Return type of this method is void.	This method helps to get an image using â€œHttpResponseâ€• option
Operation: Search **Request Parameter** **Name**: path **Type :** String **Description:**specifies the directory path **Name**: filter **Type :** String **Description:**Specifies the file types that need to be filter **Name**: searchString **Type :** String **Description:**specifies the search string **Name**: caseSensitive **Type :** String **Description:**specifies the case sensitive option **Name**: selectedItems **Type :** IEnumerable<object> **Description:**It contains the details about selected folder	It should return data in JSON format with key name as "**files**" and JSON fields need to be with following field names. *â€œname, isFile, hasChild,filterPathâ€.* *For example:* {"**files**":[{"**name":"bird.jpg**","type":"File","size":102182,"dateModified":"1/9/2016 6:48:42 AM","**hasChild":false,"isFile":true**,"filterPath":null},{"**name":"sea.jpg**","type":"File","size":97145,"dateModified":"1/9/2016 6:48:42 AM","**hasChild":false,"isFile":true**,"filterPath":null }],"details":null,"error":null}	It used to search all the matched files and sub-folders in the given folder path also it filters the specified files using it types.

Copyright © 2001 - 2020 Syncfusion Inc.

Here you need to implement new class by sub classing the existing "**BasicFileOperations**" abstract class.

C#

```
public class NewClass : BasicFileOperations
{
public override object CreateFolder(string path, string name,
IEnumerable<object> selectedItems = null)
{
throw new NotImplementedException();
}
public override void Download(string path, string[] names,
IEnumerable<object> selectedItems = null)
{
throw new NotImplementedException();
}
public override object GetDetails(string path, string[] names,
IEnumerable<object> selectedItems = null)
{
throw new NotImplementedException();
}
public override void GetImage(string path, IEnumerable<object> selectedItems
= null)
{
throw new NotImplementedException();
}
public override object Paste(string sourceDir, string targetDir, string[]
names, string option, IEnumerable<CommonFileDetails> commonFiles,
IEnumerable<object> selectedItems = null, IEnumerable<object> targetFolder =
null)
{
throw new NotImplementedException();
}
public override object Read(string path, string filter, IEnumerable<object>
selectedItems = null)
{
throw new NotImplementedException();
}
public override object Remove(string[] names, string path,
IEnumerable<object> selectedItems = null)
{
throw new NotImplementedException();
}
public override object Rename(string path, string oldName, string newName,
IEnumerable<CommonFileDetails> commonFiles, IEnumerable<object>
selectedItems = null)
{
throw new NotImplementedException();
}
public override object Search(string path, string filter, string
searchString, bool caseSensitive, IEnumerable<object> selectedItems = null)
{
throw new NotImplementedException();
}
public override void Upload(IEnumerable<HttpPostedFileBase> files, string
path, IEnumerable<object> selectedItems = null)
{
```

```
throw new NotImplementedException();
}
}
```

Also we have option to configure the AJAX request in client side, please refer link: https://help.syncfusion.com/js/fileexplorer/behavior-settings#customize-the-ajax-request-settings

Managing files that is available in SQL database

You can manage the files that are available in database using our FileExplorer control. Here you may use this custom "**SQLFileExplorerOperations**" class for handling file management related operations using SQL database. This class is used to simplify the process on server side. It contains some built-in methods that are used to handle file operations (like read, copy, move, delete, etc.) using SQL database. This class is created by inheriting the abstract class "BasicFileOperations". If is it necessary, you may override the methods in "**SQLFileExplorerOperations**" class.

- To make connection with SQL database (FileManager.mdf) services, please specify connection string in "Web.config" file as specified in the following code example.

C#

```
<add name="FileExplorerConnection" connectionString="Data
Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\FileManager.mdf;Inte
grated Security=True;Trusted_Connection=true" />
```

- In controller part, you need to create an object of "SQLFileExplorerOperations" class as shown in the following code example.

C#

```
//Here "FileExplorerConnection" is a connection string name, which is
defined in Web.config file.
//"Product" is a table name, which is defined in SQL database
SQLFileExplorerOperations sqlObj = new
SQLFileExplorerOperations("FileExplorerConnection", "Product");
```

After creating the object for "SQLFileExplorerOperations" class in controller part, you have to call the corresponding built-in file operation handling methods available in "SQLFileExplorerOperations" class based on the file operation type.

We have prepared the following sample based on this, FileExplorer Sample

File Access permission for the files that is available in SQL database

The FileExplorer provides method to assign permissions or access rights to specific users and group of users. It allows you to define access permissions for folders and files using a set of access rules to user(s).

- As per the above details, In controller part, you need to create an object of "SQLFileExplorerOperations" class and additionally you need to pass the access rules as a constructor parameter.

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
//Here "FileExplorerConnection" is a connection string name, which is
defined in Web.config file.
//"Product" is a table name, which is defined in SQL database
// SQLGetRules() which a function which returns the access rules
SQLFileAccessOperations sqlObj = new
SQLFileAccessOperations("FileExplorerConnection", "Product", SQLGetRules());
```

For this, we have prepared the sample and you can get it from the below link

sample

Note: Here we have fetched the file permissions from the SQL table for the files located in SQL database.

To know more details about file access permission, please refer this link

Cross-origin resource sharing support for FileExplorer

Cross-origin request

The “FileExplorer” can browse and manage files on remote servers, which is located in other domains. If a server is located on a different domain, on a different port or using different protocol (HTTP / HTTPS) such requests are considered to be “**cross-origin requests**”. These type requests are prohibited by IE9 and its earlier browsers.

Enabling cross-origin request in IE8 and IE9

By default, Internet Explorer 9 and earlier prohibits cross-origin requests for Internet Zone, also it ignores "Access-Control-Allow" headers. To enable cross-origin in IE8 and IE9, we have specified two type of options. As per the requirements, you can use any option that is mentioned below.

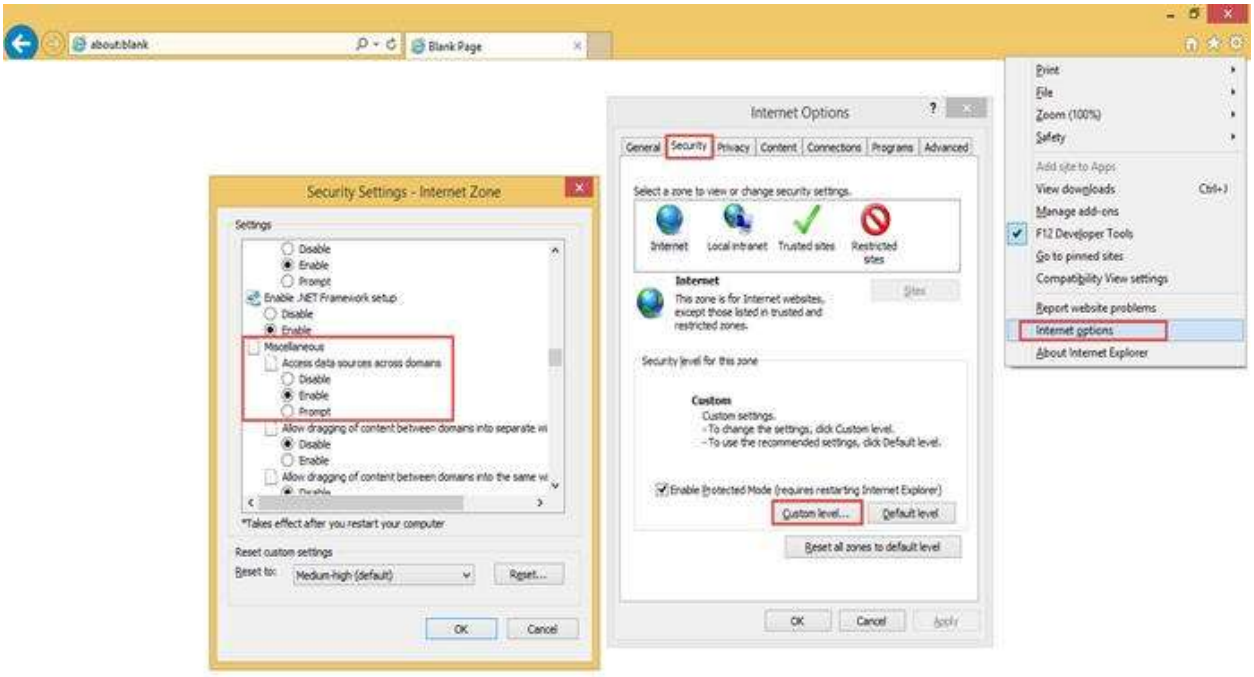
Option 1: Enabling cross-origin in IE through browser settings

To enable cross-origin access using settings of IE browser, go to Tools->Internet Options->Security tab, click on “Custom Level” button. Find the Miscellaneous and select “Enable” option, which is available under “Access data sources across domains” settings.

If your server is located in Intranet Zone, In IE Browser, confirmation dialog will be popped during first cross-domain request as “*This page is accessing information that is not under its control. This poses a security risk. Do you want to continue*?”

To suppress this warning, you need to specify the "*Access data sources across domains*" setting to “allow”.

Copyright © 2001 - 2020 Syncfusion Inc.



Option 2: Using JSONP for cross-origin request

Using JSONP data type, you can perform cross origin-request. To enable cross-origin request, in your "FileExplorer", you have to specify "**AjaxDataType"**as "**jsonp**". And we have provided "**doJSONPAction**" method to handle "jsonp" type AJAX request on server side. Please refer below code snippet to specify "**AjaxDataType"**as "**jsonp**".

Code example [CSHTML]:

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("http://js.syncfusion.com/demos/ejServices/Content/FileBrowser/")
.AjaxAction("http://js.syncfusion.com/demos/ejServices/api/FileExplorer/File
Operations")
.AjaxDataType("jsonp")
)
```

If we specify "AjaxDataType" as "jsonp", data will be received in string format while calling "doJSONPAction" method of Web API Controller, here you need to deserialize the received "localData" data into "FileExplorerParams" object. After performing corresponding operations, you have to specify the response data in serialized format with wrapped callback function. Please refer below code snippet to handle "jsonp" operations on server.

Code example [C#]:

C#

```
[HttpGet]
[ActionName("doJSONPAction")]
public object doJSONPAction(string callback, string localData)
{
```

```
object Data = null;
var serializer = new JavaScriptSerializer();
FileExplorerParams args =
(FileExplorerParams)serializer.Deserialize(localData,
typeof(FileExplorerParams));
try
{
switch (args.ActionType)
{
case "Read":
Data = FileExplorerOperations.ReadData(args.Path, args.ExtensionsAllow);
break;
case "Search":
Data = FileExplorerOperations.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive);
break;
case "CreateFolder":
Data = FileExplorerOperations.NewFolder(args.Path, args.Name);
break;
case "Paste":
Data = FileExplorerOperations.Paste(args.LocationFrom, args.LocationTo,
args.Names, args.Action, args.CommonFiles);
break;
case "Remove":
Data = FileExplorerOperations.Remove(args.Names, args.Path);
break;
case "Rename":
Data = FileExplorerOperations.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles);
break;
case "GetDetails":
Data = FileExplorerOperations.GetDetails(args.Path, args.Names);
break;
}
HttpContext.Current.Response.Write(string.Format("{0}({1});", callback,
serializer.Serialize(Data)));
return "";
}
catch (Exception e)
{
FileExplorerResponse Response = new FileExplorerResponse();
Response.error = e.GetType().FullName + ", " + e.Message;
HttpContext.Current.Response.Write(string.Format("{0}({1});", callback,
serializer.Serialize(Response)));
return "";
}
}
[HttpPost]
[ActionName("doJSONPAction")]
public void doJSONPAction()
{
if (HttpContext.Current.Request.Files.Count > 0)
FileExplorerOperations.Upload(HttpContext.Current.Request.Files,
HttpContext.Current.Request.QueryString.GetValues("Path")[0]);
}
[HttpGet]
[ActionName("doJSONPAction")]
```

```
public void doJSONPAction(string ActionType, string Path, string
SelectedItems)
{
if (ActionType == "Download")
FileExplorerOperations.Download(Path,
HttpContext.Current.Request.QueryString.GetValues("Names"));
else if (ActionType == "GetImage")
FileExplorerOperations.GetImage(Path);
}
```

In IE8 and IE9 Browser, These options helps to render our “FileExplorer” control with cross-origin resource support.

Access static physical path using FileExplorer control

Using “FileExplorer”, also you can manage the files that are available in local file system by referring its physical path. Please refer below example to achieve this.

In the view page, add “FileExplorer” helper and specify the physical path as below shown format.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("C:/Samples/web/FileExplorerContent")
.AjaxAction(@Url.Content("FileActionDefault"))
)
```

In the controller page, specify the “GetImage” handling operation as shown below. This handling function is necessary to render the images in FileExplorer, while physical path has been mentioned.

C#

```
public ActionResult FileActionDefault(FileExplorerParams args)
{
FileExplorerOperations operation = new FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
case "Search":
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
case "GetImage":
operation.GetImage(args.Path);
break;
}
return Json("");
}
```

Adding Custom column in grid view of FileExplorer

You can add custom column in grid view of “FileExplorer”. To achieve this requirement, in controller page you have to add the custom column with existing AJAX response by overriding the “Read” method, which is available in “FileExplorerOperations” class. Please refer following code example.

C#

```
public partial class FileExplorerController : Controller
{
public ActionResult Default()
{
return View();
}
public ActionResult FileActionDefault(FileExplorerParams args)
{
FileExplorerCustomOperations customOperation = new
FileExplorerCustomOperations();
FileExplorerOperations operation = new FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(customOperation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
case "Search":
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
case "GetImage":
operation.GetImage(args.Path);
break;
```

```
}
return Json("");
}
}
public class FileExplorerCustomOperations : FileExplorerOperations
{
public override object Read(string path, string filter, IEnumerable<object>
selectedItems = null)
{
CustomFileExplorerResponse ReadResponse = new CustomFileExplorerResponse();
try
{
var directory = new DirectoryInfo(GetPhysicalPath(ToAbsolute(path)));
filter = filter.Replace(" ", "");
var extensions = (filter ?? "*").Split(",|;".ToCharArray(),
System.StringSplitOptions.RemoveEmptyEntries);
var files = extensions.SelectMany(directory.GetFiles)
.Select(file => new CustomFileExplorerDirectoryContent
{
name = file.Name,
isFile = true,
size = file.Length,
type = "File",
dateModified = file.LastWriteTime.ToString(),
hasChild = false,
//assign the value in extension property
extension = file.Extension
});
var directories = directory.GetDirectories().Select(subDirectory => new
CustomFileExplorerDirectoryContent
{
name = subDirectory.Name,
size = 0,
isFile = false,
type = "Directory",
dateModified = subDirectory.LastWriteTime.ToString(),
hasChild = subDirectory.GetDirectories().Length > 0 ? true : false,
extension = ""
});
object fileDetails = files.Concat(directories);
ReadResponse.files =
(IEnumerable<CustomFileExplorerDirectoryContent>)fileDetails;
return ReadResponse;
}
catch (Exception e)
{
ReadResponse.error = e.GetType().FullName + ", " + e.Message;
return ReadResponse;
}
}
}
public class CustomFileExplorerDirectoryContent
{
public string name { get; set; }
public string type { get; set; }
public long size { get; set; }
public string dateModified { get; set; }
```

```
public bool hasChild { get; set; }
public bool isFile { get; set; }
//add custom property
public string extension { get; set; }
}
//customize the FileExplorer response
public class CustomFileExplorerResponse
{
public IEnumerable<CustomFileExplorerDirectoryContent> files { get; set; }
public IEnumerable<FileDetails> details { get; set; }
public object error { get; set; }
}
```

In the view page, add the custom column in grid using "GridSettings" API of "FileExplorer".

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.AjaxAction(@Url.Content("FileActionDefault"))
.GridSettings(grid =>
grid.Column(column=>
{
column.Add().Field("name").HeaderText("Name").Width("30%");
column.Add().Field("type").HeaderText("Type");
//adding new column
column.Add().Field("extension").HeaderText("Extension");
column.Add().Field("size").HeaderText("Size");
column.Add().Field("dateModified").HeaderText("Date Modified");
}
)
)
)
```

Create FileExplorer in older version

In older versions (Before 13.3.0.7), you can easily create the FileExplorer widget by using the following steps.

1. Add the following code example to the corresponding view page to render the FileExplorer.

RAZOR

```
@Html.EJ().FileExplorer("fileExplorer").Path("~/FileExplorerContent/").AjaxA
ction(@Url.Content("FileActionDefault"))
```

In above code block, "Path" denotes the filesystem path that are to be explored and "AjaxAction" specifies the URL of server side AJAX handling method that handles file operations of FileExplorer control. So "Path" and "AjaxAction" are the mandatory configuration at here.

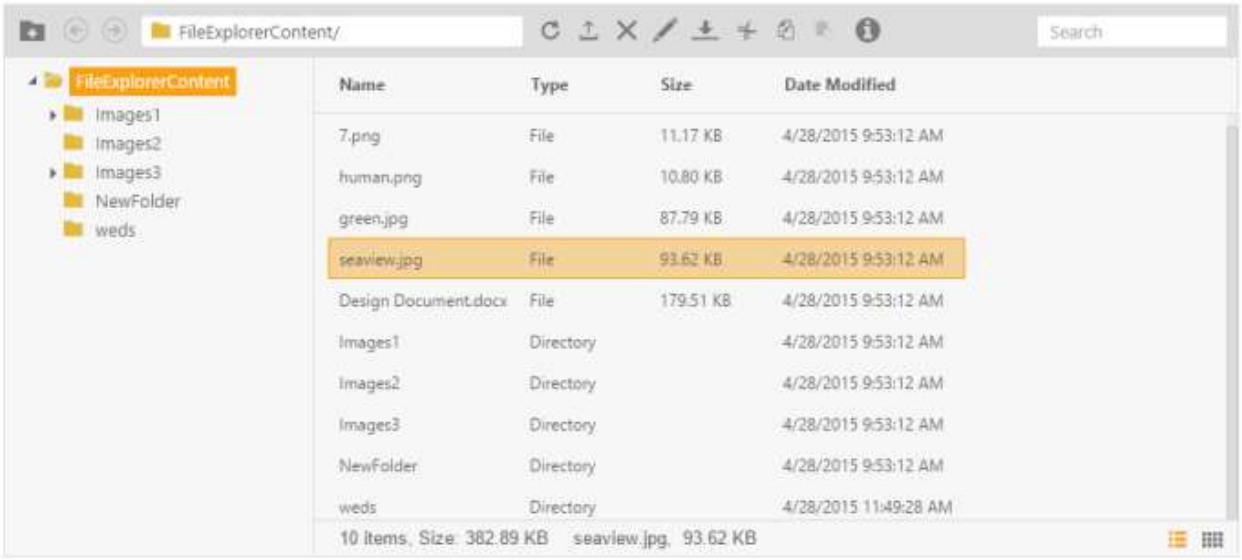
2. Add the following code example to the corresponding controller page

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult FileActionDefault(FileExplorerParams args)
{
switch (args.ActionType)
{
case "Read":
return Json(FileExplorerOperations.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(FileExplorerOperations.CreateFolder(args.Path, args.Name));
case "Paste":
FileExplorerOperations.Paste(args.LocationFrom, args.LocationTo,
args.Name.Split(','), args.Action, args.CommonFiles);
break;
case "Delete":
FileExplorerOperations.Delete(args.Name.Split(','), args.Path);
break;
case "Rename":
FileExplorerOperations.Rename(args.Path, args.PreviousName, args.NewName,
args.Type, args.CommonFiles);
break;
case "GetDetails":
return Json(FileExplorerOperations.GetDetails(args.Path, args.Name,
args.Type));
case "Download":
FileExplorerOperations.Download(args.Path);
break;
case "Upload":
FileExplorerOperations.Upload(args.FileUpload, args.Path);
break;
}
return Json("");
}
```

Once you have completed the above steps, you will get an output like below.



Copyright © 2001 - 2020 Syncfusion Inc.

FileExplorerOperations is a predefined class, which is used to perform File Explorer-based operations like read, createFolder, download, upload, rename, paste, getImage. It minimizes the work load on server-side.

By default, we send following parameters in data field of corresponding AJAX request. This helps to handle server side operation. Some Server side action method will only return the response data. This response data and request parameter are explained in following table.

Operation	Default Request Parameter	Response data	Details
Read	String ActionType,String Path,String ExtensionsAllow	Should return data in array of JSON format and JSON fields need to be in following field names. "**name, size, type, dateModified, hasChild**" *For example:*[{name: "7.png", type: "File", size: 11439, dateModified: "3/31/2015 3:16:38 PM", hasChild: false},{name: "human.png", type: "File", size: 11059, dateModified: "3/31/2015 3:16:35 PM", hasChild: false}]	Read the data from the given path
CreateFolder	String ActionType,String Path,String Name	Should return new folder name in string	Create the new folder to given path
Paste	String ActionType, String LocationFrom, String LocationTo, String Name,String Type, String Action	Void	Paste the content from source to target place
Delete	String ActionType, String Name,String Path	Void	Delete the data from the given path
Rename	String ActionType, String Path,String PreviousName, String NewName, String Type	Void	Rename the file or folder from the given path

GetDetails	String ActionType, String Path, String Name,String Type	Response data should be in JSON format like below {CreationTime:"4/28/2015 9:44:32 AM", Extension:".png", Format:"Archive", FullName:"F:\All samples\FileExplorer_Custom\FileExplorerContent\human.png", LastAccessTime:"4/28/2015 9:44:32 AM", LastWriteTime:"3/31/2015 3:16:35 PM", Length:11059, Name:"human.png"} Here you may add additional date fields with this JSON	To get the details of the data from the given path
Download	String ActionType,String Path	Void	To download the data from the given path
Upload	String ActionType, IEnumerable<HttpPostedFileBase> FileUpload, String Path	Void	To upload the data from the given path

Customize the Root Folder name in FileExplorer

You can set the alias name to the root folder of FileExplorer by using `rootFolderName` API. It is used to replace the actual root folder name in the FileExplorer UI. Refer to the below code to set the alias name for the root folder of FileExplorer.

RAZOR

```
@(Html.EJ().FileExplorer("fileExplorer")
.Path("~/FileExplorerContent/")
.RootFolderName ("This PC")
.AjaxAction(@Url.Content("FileActionDefault"))
)
```

Use the FTP connection using FileExplorer

You can access your FTP service using our FileExplorer component and can manage your files easily.

FTP Login

First you will need the FTP account login information. You can connect with the primary username and password or an FTP account. The FTP server is open to the public, using anonymous access. But most of the FTP servers are password protected.

Copyright © 2001 - 2020 Syncfusion Inc.

Username: username or user@example.com

Password: The password that was created for the user you are using.

FTP with FileExplorer server

In this FTP application, we have provided “FTPFileOperationController” file, which contains “FileActionDefault” action method. When you make AJAX request to this controller part, “FileActionDefault” method will be called after based on ActionType” parameter value, then it calls the corresponding built-in FTP file handling methods such as ‘Read’, ‘Search’, ‘Download’, ‘Upload’, ‘Remove’, ‘Rename’ which are available in our “FTPFileExplorerOperations” class.

Steps to create FileExplorer application to access FTP server

Step 1: Create a new ASP.NET MVC project using Visual Studio.

Step 2: Include **“FTPFileExplorerOperations”** file in that application. This file contains built-in file handling methods which helps to connect our FileExplorer with FTP service.

https://www.syncfusion.com/downloads/support/directtrac/general/ze/FTPFileExplorerOperations901960089.zip

Step 3: Add **“FileExplorerController.cs”** file in the controller part of FTP project.

This file, contains action handler methods. Based on the request parameters, it helps to call the built-in file handling methods of **FTPFileExplorerOperations** and perform corresponding actions.

C#

```
public ActionResult FileActionDefault(FileExplorerParams args)
{
FTPFileExplorerOperations operation= new FTPFileExplorerOperations();
// FTP Operations
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo, args.Names,
args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "GetImage":
operation.GetImage(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
case "Search":
```

```
return Json(operation.Search(args.Path, args.ExtensionsAllow,
args.SearchString, args.CaseSensitive));
}
return Json("");
}
```

Step 4: Add FTP Storage path that is viewed by FileExplorer.

You will create FTP URL like “ftp://localhost/FileBrowser/” (Port number may vary).

After that, you need to specify the FTP folder path and AJAX action handler name as shown in below code block.

RAZOR

```
@Html.EJ().FileExplorer("fileexplorer").Path("ftp://localhost/FileBrowser/")
.AjaxAction(@Url.Content("/FileExplorer/FileActionDefault"))
```

We have prepared a demo based on above steps, please refer this.

https://www.syncfusion.com/downloads/support/directtrac/general/ze/FTPMVC1739148525.zip

Gantt

Overview

The Essential JavaScript Gantt control is designed to visualize and edit the project schedule, and track the project progress. It helps to organize and schedule the projects and also you can update the project schedule through interactions like editing, dragging and resizing.

Key Features

- Sorting
- Editing
- Filtering
- Task relationship
- Stripline
- Baseline
- Timeline customization
- Work break down structure
- Taskbar customization
- Task label customization
- Localization
- Resources
- Resource allocation view
- Critical path
- Selection
- Task scheduling modes
- Work breakdown structure
- Exporting
- Row drag and drop
- Sequencing tasks

Copyright © 2001 - 2020 Syncfusion Inc.

- Searching
- Holidays
- Work and effort driven tasks

Getting Started

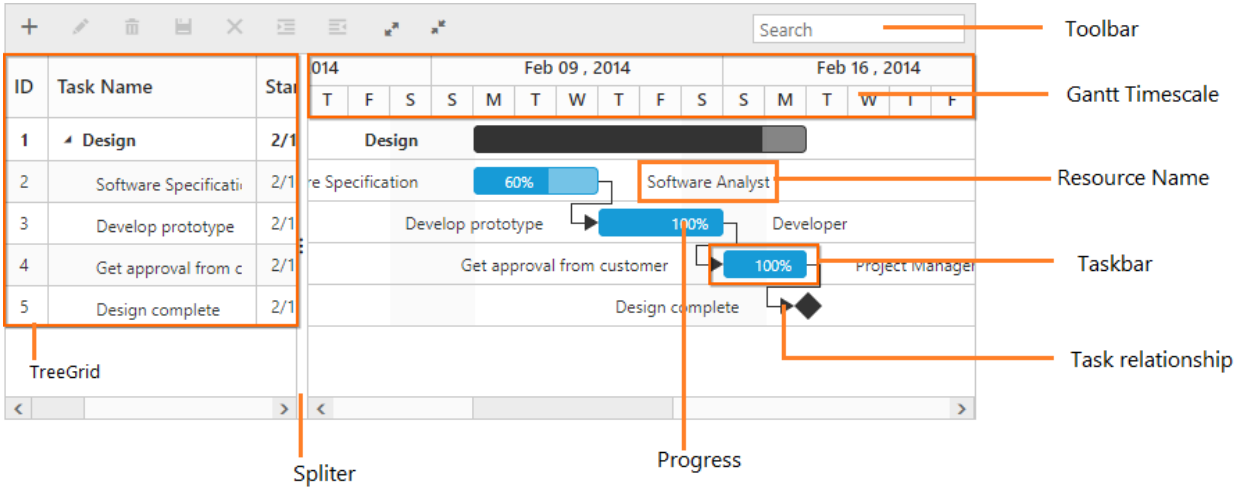
This section explains briefly about how to create a Gantt chart in your application with ASP.NET MVC.

Create your first Gantt in MVC

This section explains briefly about the control structure and how to create a Gantt chart.

Control Structure

Gantt chart is used to edit and visualize project schedule and also to track progress of a task. The following screen shot illustrates the elements of a Gantt chart.



Elements of a Gantt chart

- Toolbar – It is a collection of toolbar buttons to add, edit, delete a task. You can outdent and indent a task using outdent and indent buttons. Following screen shot illustrates the function of each toolbar button,

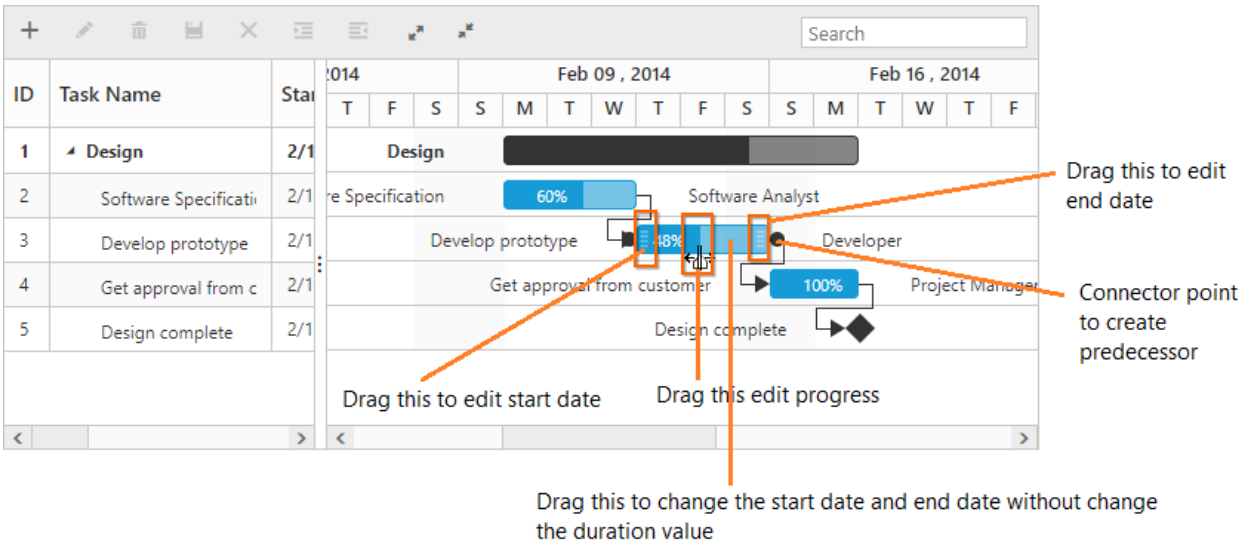


Toolbar of Gantt chart

- Search Textbox – It is used to search tasks, which contains the search string.
- Resource Names – It displays the names of the resources assigned to that task.
- Task bar – It is a graphical representation of the duration of task.
- Task Progress – It displays the percentage of the task completed.
- Header – It represents time scale based on which a task bar is drawn.
- Tree Grid – It displays the tasks and its sub tasks in hierarchical table.
- Task Relationship – It determines when to start or finish a task.

Copyright © 2001 - 2020 Syncfusion Inc.

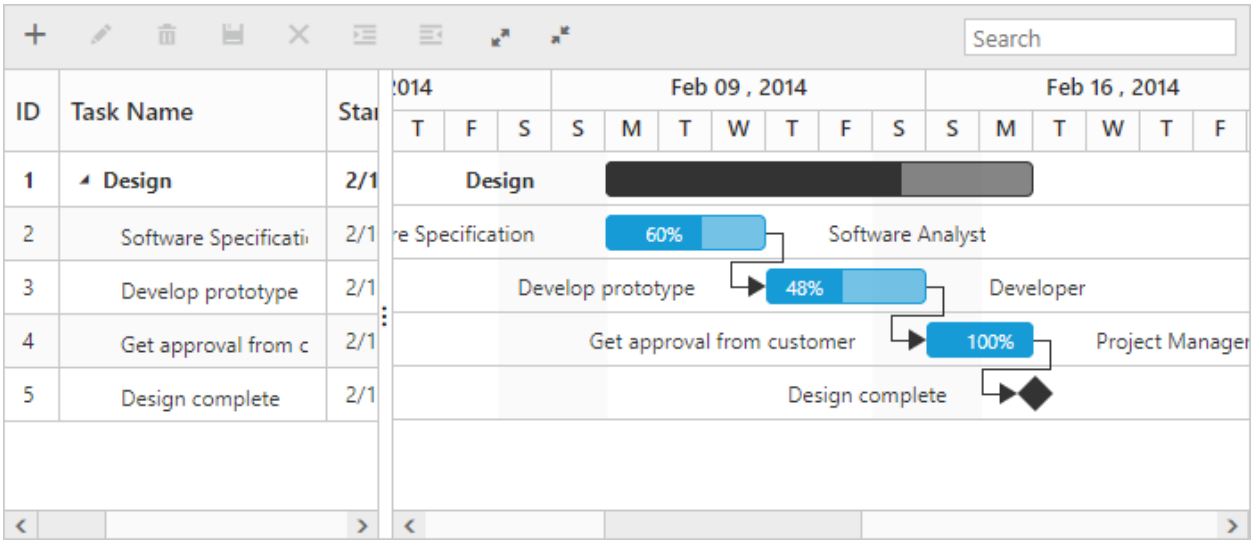
- Interactive Editing – You can edit the duration of a task by dragging or resizing the task bar. Following screen shot illustrates this.



Editing options Gantt chart

Create your Gantt chart

In this tutorial, you will learn how to create a simple Gantt chart, add tasks and subtasks, and set relationship between tasks for design phase of software project. The following screen shot displays the output after completing this tutorial.



Simple Gantt chart

1. First, create a new MVC Razor project; please refer MVC-Getting Started documentation to create new project and add necessary DLL's and script files.
2. Create a data source in Controllers/HomeController.cs file, We can bind datasource to Gantt using Datasource property.

Copyright © 2001 - 2020 Syncfusion Inc.

3. Add the following code example in the Index.cshtml file to create a Gantt control in the View page.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
.TaskIdMapping("TaskID")//Provide name of the property which contains task
id in the data source
.TaskNameMapping("TaskName")//Provide name of the property which contains
task name
.StartDateMapping("StartDate")//Provide name of the property which contains
start date of the task
.DurationMapping("Duration")//Provide name of the property which contains
duration of the task
.ProgressMapping("Progress")//Provide name of the property which contains
progress of the task
.ScheduleStartDate("02/01/2014")//Provide schedule header start date
.ScheduleEndDate("03/14/2014")//Provide schedule header end date
.ChildMapping("SubTasks")//Provide name of the property which contains
subtask of the task
.TreeColumnIndex(1)
.SizeSettings(ss=>ss.Width("700").Height("300"))
.Datasource(ViewBag.datasource) //Provides data source for Gantt
)
@(Html.EJ().ScriptManager())
```

C#

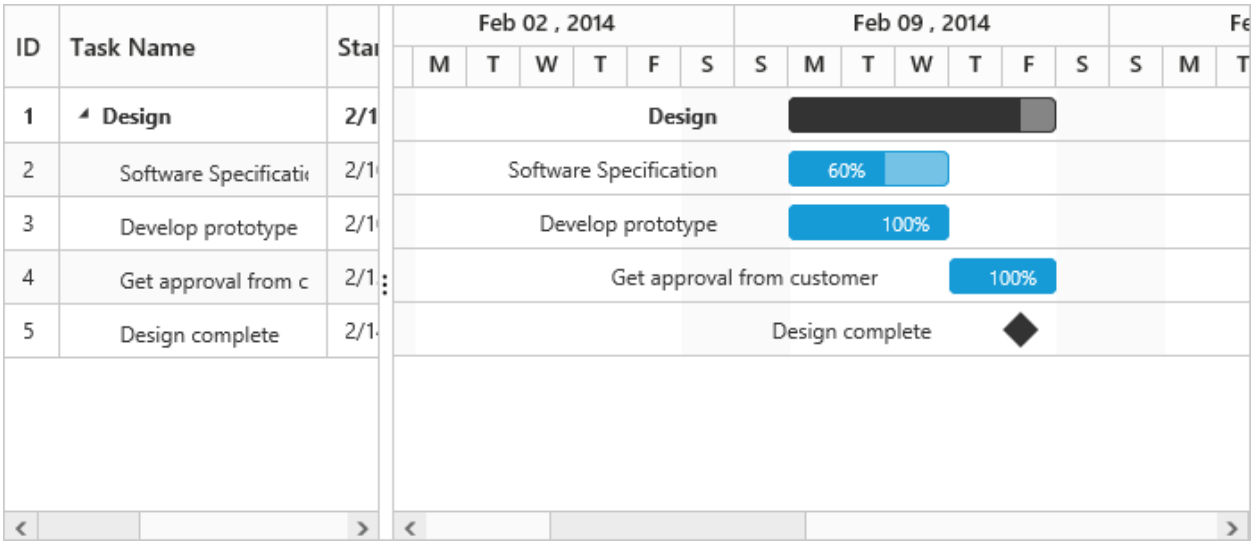
```
public class DefaultData
{
public string StartDate { get; set; }
public string EndDate { get; set; }
public int TaskID { get; set; }
public string TaskName { get; set; }
public int Duration { get; set; }
public int Progress { get; set; }
public List<DefaultData> SubTasks { get; set; }
public string Predecessors { get; set; }
public List<int> ResourceID { get; set; }
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> tasks = new List<DefaultData>();
tasks.Add(new DefaultData()
{
TaskID = 1,
TaskName = "Design",
StartDate = "02/10/2014",
EndDate = "02/14/2014",
Duration = 6,
});
tasks[0].SubTasks = new List<DefaultData>();
tasks[0].SubTasks.Add(new DefaultData()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
TaskID = 2,
TaskName = "Software Specification",
StartDate = "02/10/2014",
EndDate = "02/12/2014",
Duration = 3,
Progress = 60,
ResourceID = new List<int>() { 2 }
});
tasks[0].SubTasks.Add(new DefaultData()
{
TaskID = 3,
TaskName = "Develop prototype",
StartDate = "02/10/2014",
EndDate = "02/12/2014",
Duration = 3,
Progress = 100,
Predecessors = "2FS",
ResourceID = new List<int>() { 3 }
});
tasks[0].SubTasks.Add(new DefaultData()
{
TaskID = 4,
TaskName = "Get approval from customer",
StartDate = "02/13/2014",
EndDate = "02/14/2014",
Duration = 2,
Progress = 100,
Predecessors = "3FS",
ResourceID = new List<int>() { 1 }
});
tasks[0].SubTasks.Add(new DefaultData()
{
TaskID = 5,
TaskName = "Design complete",
StartDate = "02/14/2014",
EndDate = "02/14/2014",
Duration = 0,
Predecessors = "4FS"
});
return tasks;
}
}
public ActionResult SimpleGantt()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
return View();
}
```

The following screen shot displays the Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



Gantt chart

Enable Toolbar

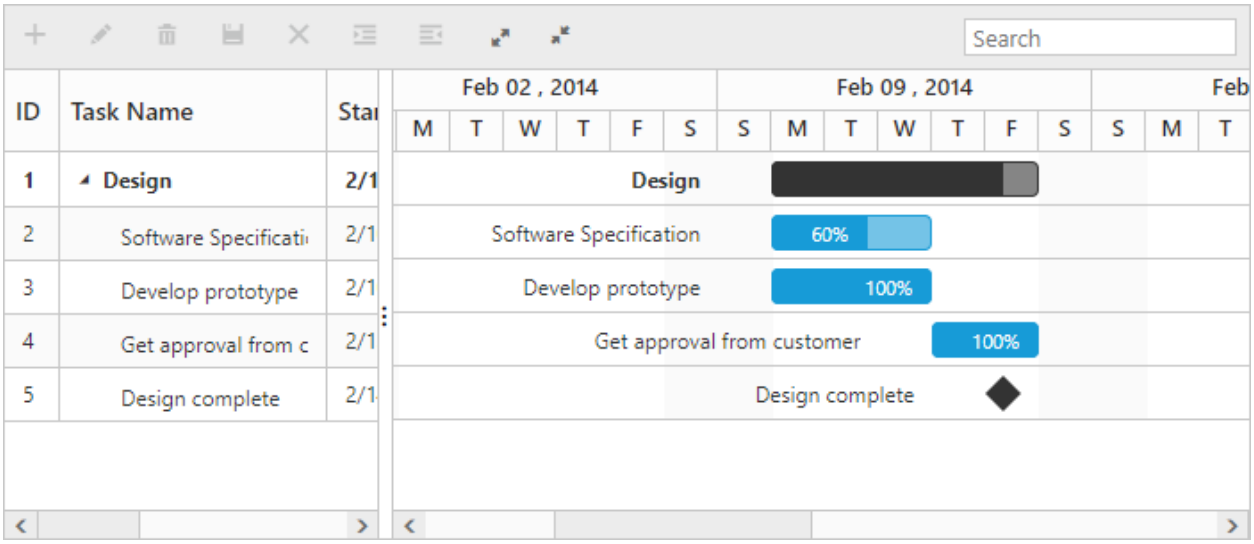
Gantt control contains the toolbar options to Add, Edit, Delete, Search, Indent, Outdent, Expand and Collapse all records. You can enable toolbar using ToolbarSettings.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
.ToolbarSettings(ts =>
{
ts.ShowToolbar(true);
ts.ToolbarItems(new List<GanttToolBarItems>()
{
GanttToolBarItems.Add,
GanttToolBarItems.Edit,
GanttToolBarItems.Delete,
GanttToolBarItems.Update,
GanttToolBarItems.Cancel,
GanttToolBarItems.Indent,
GanttToolBarItems.Outdent,
GanttToolBarItems.ExpandAll,
GanttToolBarItems.CollapseAll,
GanttToolBarItems.Search
});
})
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays a Tool bar in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



Toolbar in Gantt chart

Note: Add, Edit, Delete options are enabled when enabling the AllowEditing, AllowAdding, AllowDeleting in the EditSettings.

Enable sorting

Gantt control has the sorting functionality for arranging the tasks in ascending or descending based on the particular column.

Multicolumn sorting

Enable the multicolumn sorting in Gantt by setting AllowMultiSorting to true. You can sort multiple columns in Gantt by selecting the desired column header while holding the CTRL key.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.AllowSorting(true)
.AllowMultiSorting(true)
)
@(Html.EJ().ScriptManager())
```

Enable Editing

You can enable editing using EditSettings and AllowGanttChartEditing.

Cell Editing

Modify the task details through the grid cell editing by setting the EditMode as CellEditing.

Normal Editing

Modify the task details through the edit dialog by setting the EditMode as Normal.

Taskbar Editing

Modify the task details through user interaction like resizing and dragging the taskbar.

Predecessor Editing

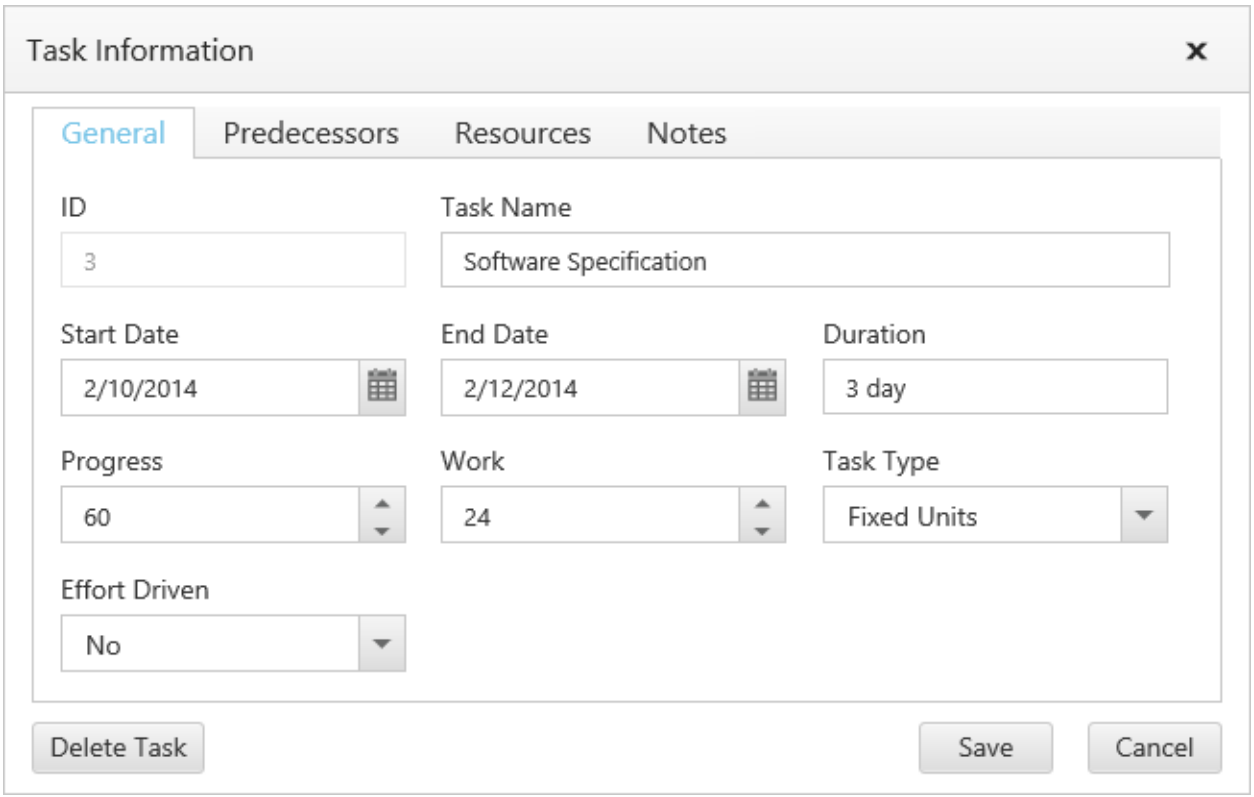
Modify the predecessor details of a task by using mouse interactions by setting AllowGanttChartEditing as true and setting the value for PredecessorMapping property.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.AllowGanttChartEditing(true)
.PredecessorMapping("Predecessor") // Predecessor editing
.EditSettings(edit =>
{
edit.AllowEditing(true);
edit.AllowAdding(true);
edit.AllowDeleting(true);
edit.EditMode("cellEditing");
edit.AllowIndent(true);
})
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays the Gantt control with dialog editing option.



Enable Context Menu

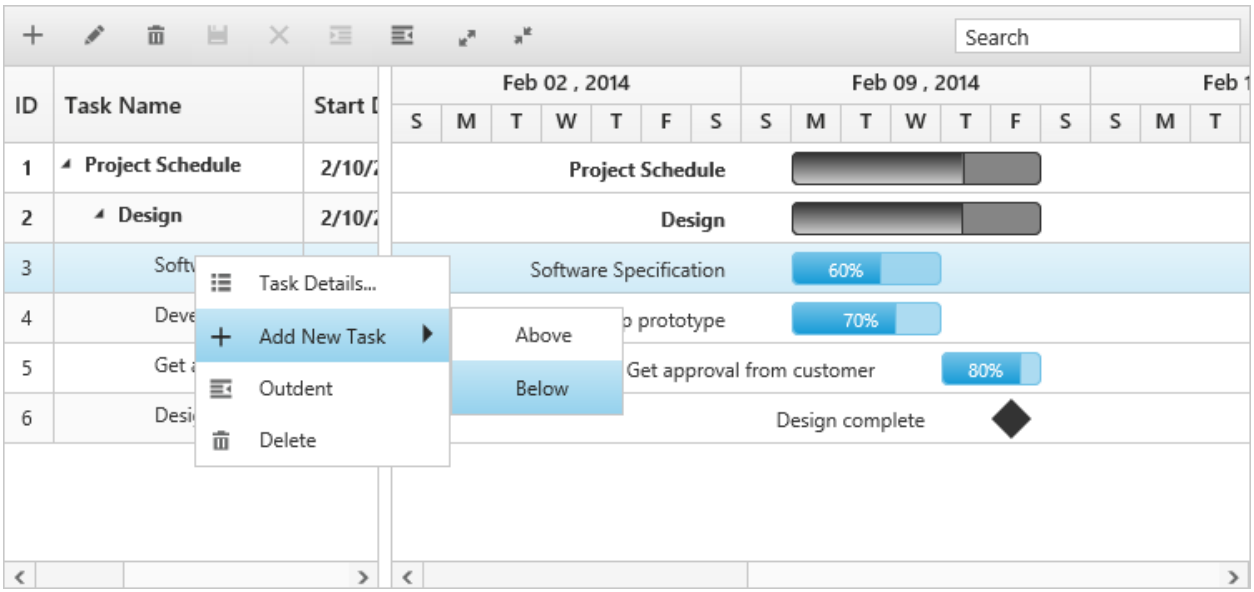
You can enable the context menu in Gantt by setting the `EnableContextMenu` as `true`.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.EnableContextMenu(true)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screen shot displays Gantt chart with Enable Context menu option.



Gantt chart with Enable Context menu option

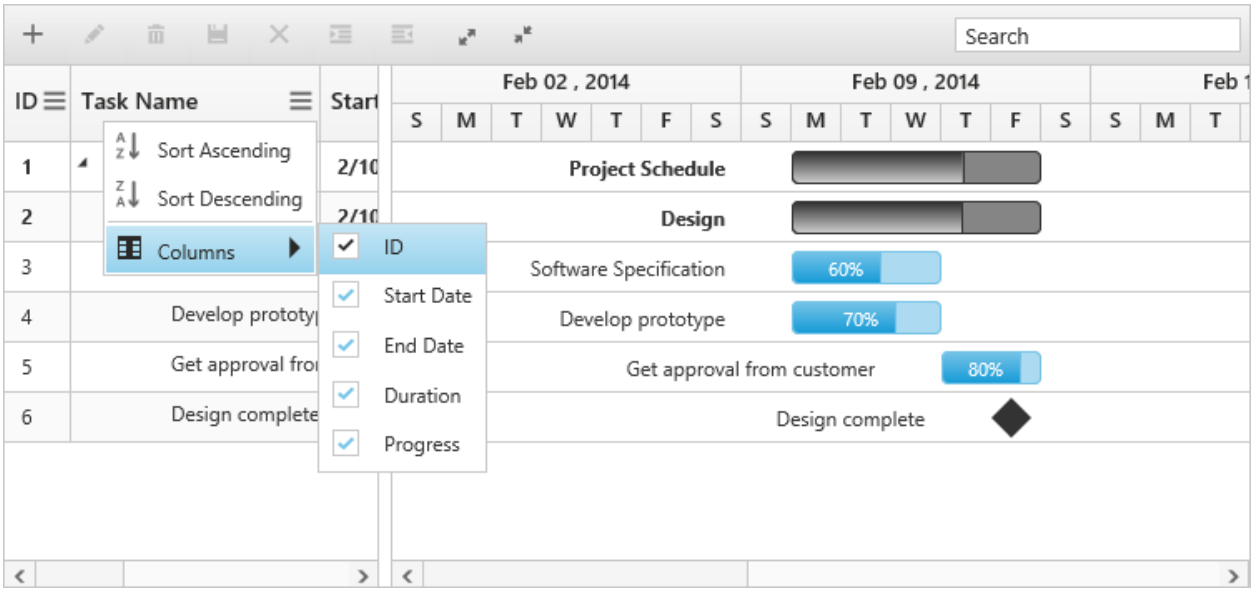
Enable Column Menu

You can enable the column menu in Gantt, by setting the ShowColumnChooser as true.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.ShowColumnChooser(true)
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays Gantt control in which column chooser option is enabled



Provide tasks relationship

In Gantt, you have the predecessor support to show the relationship between two different tasks.

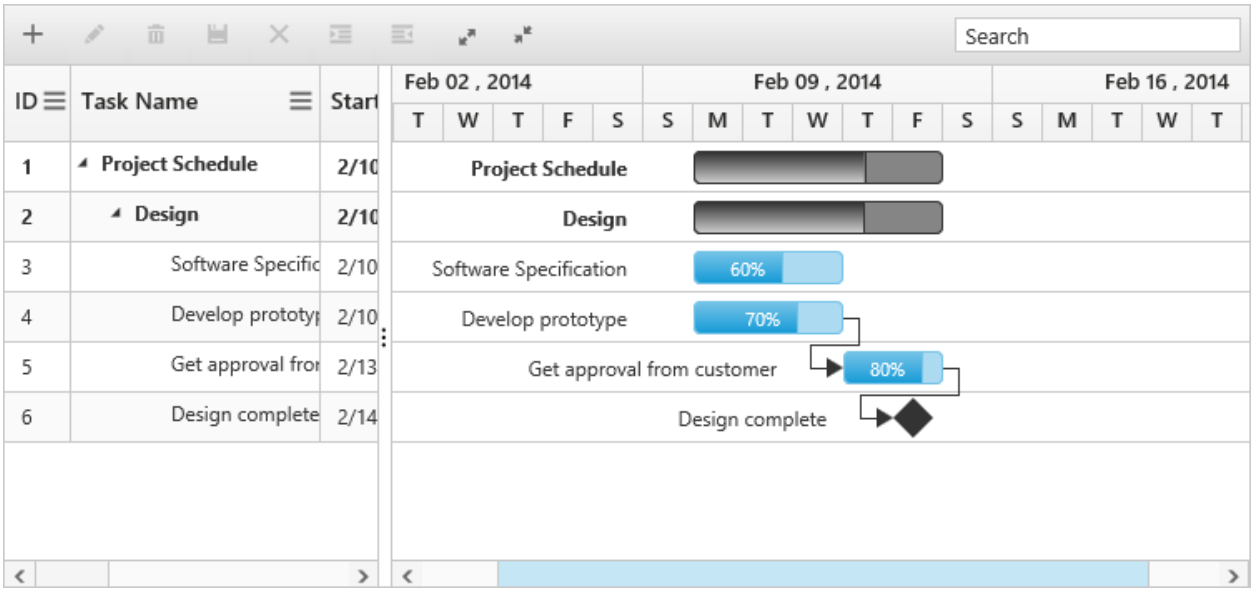
- **Start to Start (SS)** - You cannot start a task until the other task starts.
- **Start to Finish (SF)** - You cannot finish a task until the other task finishes.
- **Finish to Start (FS)** - You cannot start a task until the other task completes.
- **Finish to Finish (FF)** - You cannot finish a task until the other task completes.

You can show the relationship in tasks, by using the PredecessorsMapping property as shown in the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.PredecessorMapping("Predecessor")
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays the relationship between tasks.



Gantt chart with relationships between tasks

Provide Resources

In Gantt control, you can display and assign the resource for each task. Create a collection of JSON object, which contains id and name of the resource and assign it to the resources property. Then, specify the field name for id and name of the resource in the resource collection to ResourceIdMapping and ResourceNameMapping options. The name of the field, which contains the actual resources assigned for a particular task in the DataSource is specified using the ResourceInfoMapping.

C#

```
public class Resources
{
```

```
public int ResourceID { get; set; }
public string ResourceName { get; set; }
}
public class ResourceList
{
public static List<Resources> GetData()
{
List<Resources> resourceDetails = new List<Resources>();
resourceDetails.Add(new Resources() { ResourceID = 1, ResourceName =
"Project Manager" });
resourceDetails.Add(new Resources() { ResourceID = 2, ResourceName =
"Software Analyst" });
resourceDetails.Add(new Resources() { ResourceID = 3, ResourceName =
"Developer" });
resourceDetails.Add(new Resources() { ResourceID = 4, ResourceName =
"Testing Engineer" });
return resourceDetails;
}
}
public ActionResult SimpleGantt()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
var Resources = ResourceList.GetData();
ViewBag.resources = Resources;
return View();
}
```

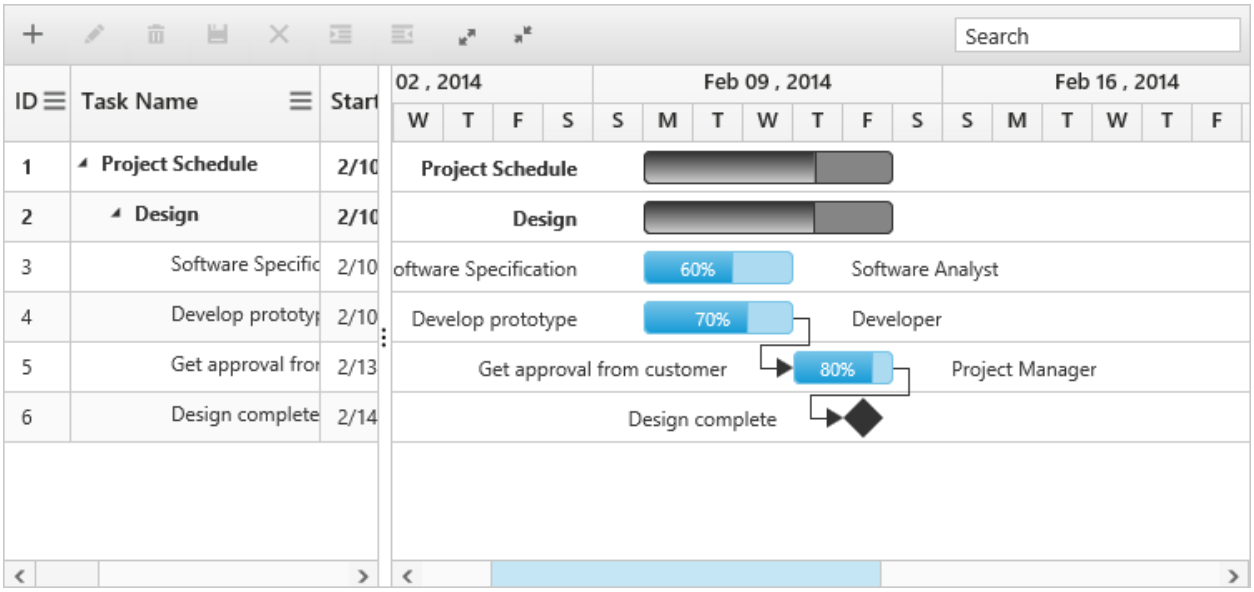
Add the following code example in the index.cshtml file, to create the Gantt control with resource in the View page.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.ResourceInfoMapping("ResourceID")//Field name which contains resource
details for the task
.ResourceNameMapping("ResourceName")//resource Name mapping
.ResourceIdMapping("ResourceID")//resource Id Mapping
//...
.Resources(ViewBag.resources)//resource collection dataSource
.Datasource(ViewBag.datasource) //Provides data source for Gantt
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays resource allocation for tasks in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



This section explains briefly about the control structure and how to create a Gantt chart.

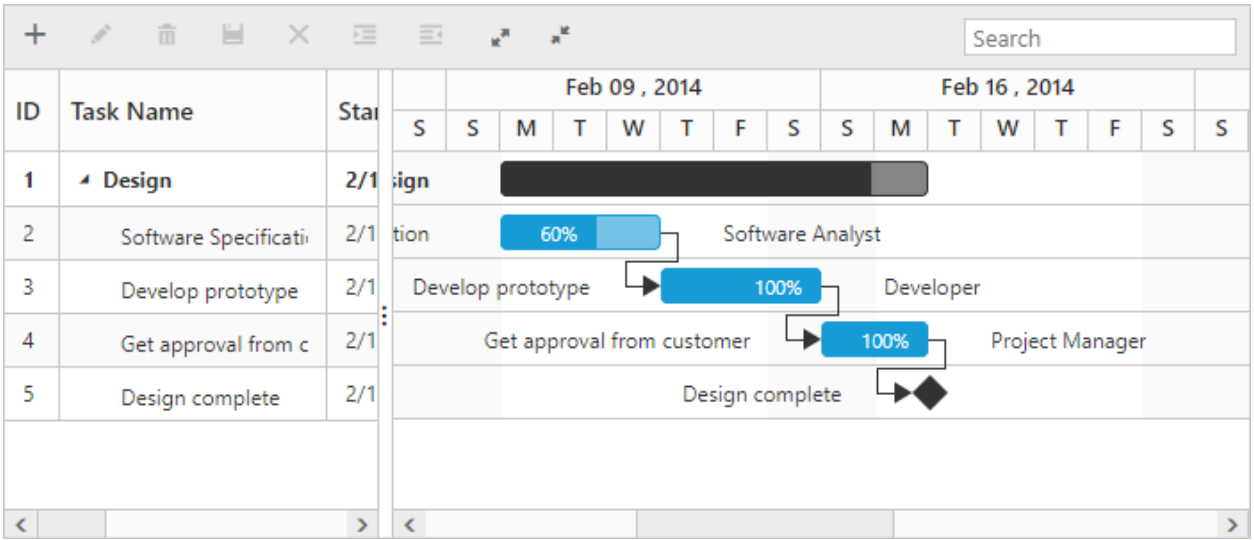
Highlight Weekend

In Gantt, you can on or off weekends high lighting by setting the HighlightWeekEnds as true or false.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.HighlightWeekends(false)
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays Gantt chart in which highlight weekends is disabled:



Define dimension of Gantt

By default Gantt control was rendered with 100% width and 450px height, we can define the dimension of Gantt control by using SizeSettingsproperty. Gantt control width and height can be

Copyright © 2001 - 2020 Syncfusion Inc.

defined by either Height and Width properties or by defining inline style in Gantt container element. The below code example shows how to define width and height for Gantt control.

ASPX-CS

```
<!--<div id="GanttContainer" style="width:700px;height:350px"></div>-->
<div id="GanttContainer"></div>
<script>
@(Html.EJ().Gantt("GanttContainer")
//...
.SizeSettings(size=>size.Height("350px").Width("700px"))
)
@(Html.EJ().ScriptManager())
</script>
```

Note: Gantt control will automatically update the width and height value based on container element on window resize action, this can be enabled by setting IsResponsive property as true for this Height and Width value will be defined in percentage.

Add notes in tasks

In Gantt, we can add additional information about the tasks, this information can be defined in data source and this field was mapped to Gantt control by using NotesMapping property. Notes values can be defined as string or in HTML string format. This notes content was displayed in Notes column and notes value can be updated by using cell editing and dialog editing. The following code example shows how to use the NotesMapping property.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.NotesMapping("notesContent")
)
@(Html.EJ().ScriptManager())
```

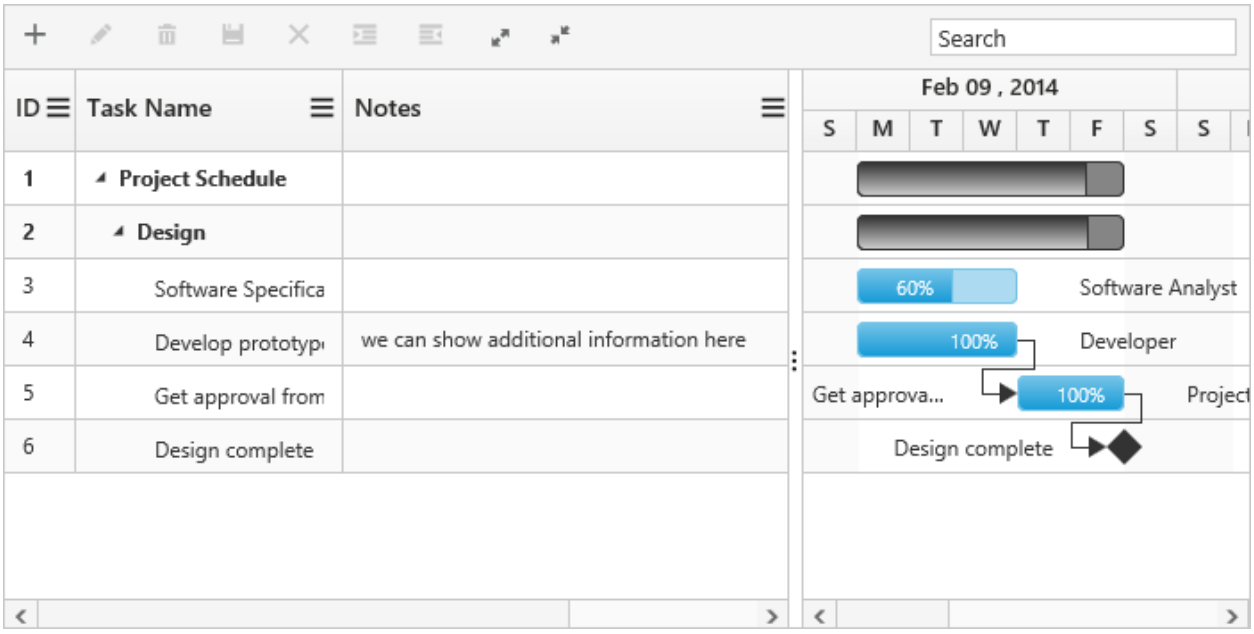
C#

```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
return View();
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
//..
list.Add(new DefaultData()
{
```

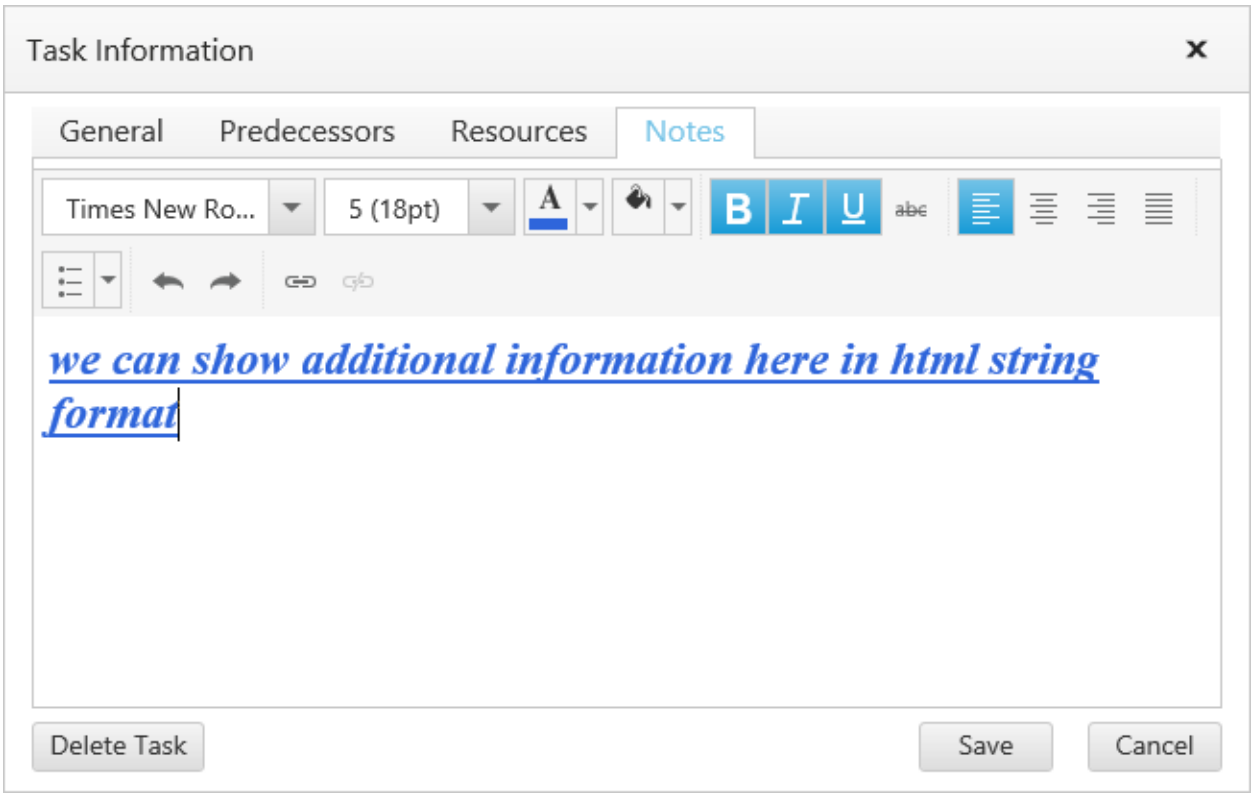
Copyright © 2001 - 2020 Syncfusion Inc.

```
Id = 4,
Name = "Develop prototype",
notesContent = "we can show additional information here",
//..
});
return list;
}
}
public class DefaultData
{
public int Id { get; set; }
public string Name { get; set; }
public string notesContent { get; set; }
}
}
```

The below screenshot shows the output of above code example.



Notes column in Gantt



Editing notes in edit dialog

Note: Notes value was displayed as plain text in Grid part and also when we edit the notes value by cell editing, value was stored as string value. We can use edit dialog to update the notes value of task in HTML string format.

Milestones in Gantt

Milestones are used to denote the important event/stages in project management. Milestones start date and end date value will be same and duration value will be 0. Using MilestoneMapping property we can map the datasource with Milestone mapping field and render Gantt tasks as milestone based on its mapping value. The below code example shows how to use this property.

ASPX-CS

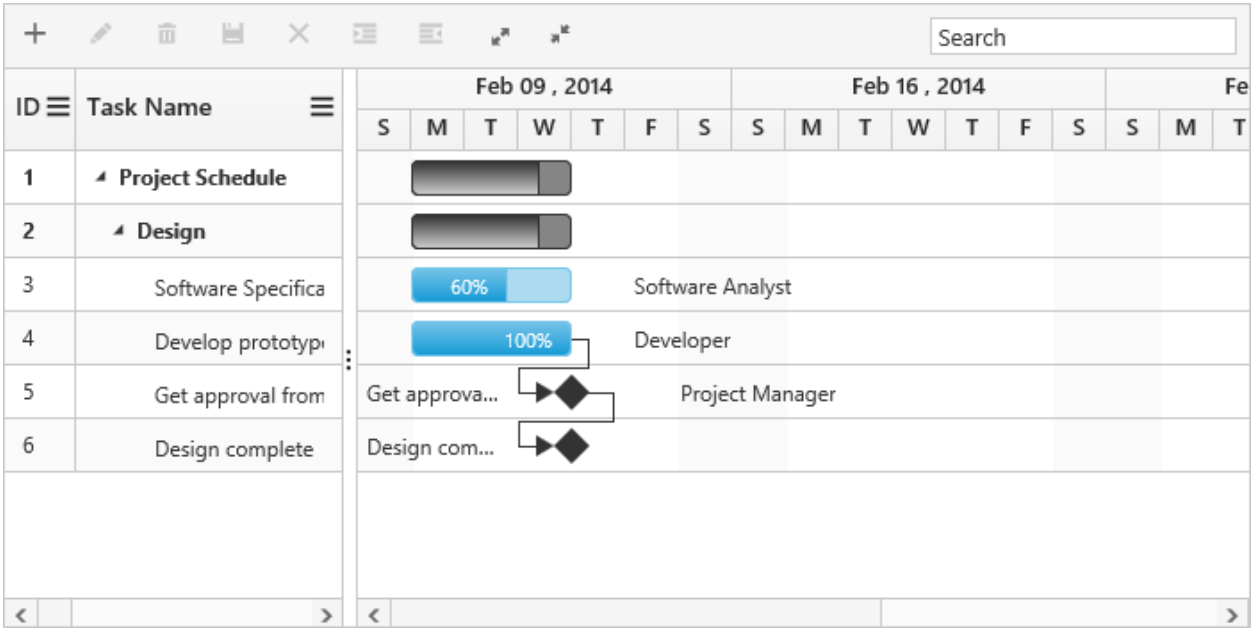
```
@(Html.EJ().Gantt("GanttContainer")
//...
.MilestoneMapping("isMileStone")
)
@(Html.EJ().ScriptManager())
```

C#

```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
```

```
ViewBag.datasource = DataSource;
return View();
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
//..
list.Add(new DefaultData()
{
Id = 4,
Name = "Develop prototype",
isMileStone = false,
//..
});
list.Add(new DefaultData()
{
Id = 5,
Name = "Get approval from customer",
StartDate = new Date("02/10/2014"),
EndDate = new Date("02/14/2014"),
Duration = 2,
isMileStone = true,
//..
});
return list;
}
}
}
```

The below screenshot shows the output of above code example.



Data Binding

Data binding is the process that establishes a connection between the application and different kinds of data sources such as business objects. And it is possible to bind local data and remote data in Gantt.

You can set the data source for Gantt control by using DataSource property.

Local Data Binding

In Local Data Binding, datasource for rendering the Gantt control is retrieved from the same application locally.

Two types of Data Binding are possible with Gantt control,

- Hierarchical datasource binding
- Self-referential data binding (Flat data)

Hierarchical data-source binding

In hierarchical data binding subtasks or child tasks collections are identified by using ChildMapping property.

The following code example explains how to bind the hierarchical data in Gantt.

ASPX-CS

```
@using Syncfusion.JavaScript
@using Syncfusion.MVC.EJ
<!DOCTYPE html>
<html>
<head>
@*Add script reference and style reference here*@
</head>
<body>
@(Html.EJ().Gantt("ganttsample2")
.TaskIdMapping("Id")
.TaskNameMapping("Name")
.StartDateMapping("StartDate")
.DurationMapping("Duration")
.ProgressMapping("PercentDone")
.TreeColumnIndex(1)
.ChildMapping("Children")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
</body>
</html>
```

C#

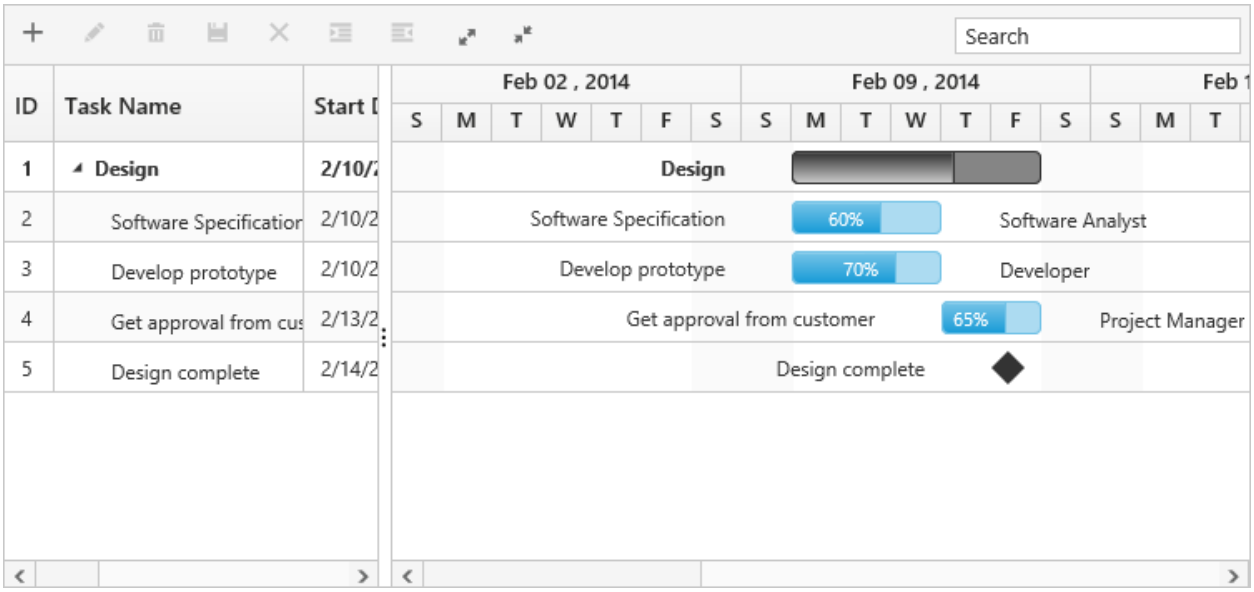
```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
list.Add(new DefaultData()
{
Id = 1,
Name = "Design",
StartDate = "02/10/2014",
EndDate = "02/14/2014",
Duration = 5,
PercentDone = 40,
Children = (new List<DefaultData>()
{
new DefaultData()
{
Id = 2,
Name = "Software Specification",
StartDate = "02/10/2014",
EndDate="02/12/2014",
Duration = 5,
PercentDone = 40
},
new DefaultData()
{
Id = 3,
Name = "Develop Prototype",
StartDate = "02/10/2014",
EndDate="02/12/2014",
Duration = 4,
PercentDone = 40,
Predecessor = "2"
},
//..
})
});
return list;
}
}
public class DefaultData
{
public string StartDate { get; set; }
public string EndDate { get; set; }
public int Id { get; set; }
public string Name { get; set; }
public int Duration { get; set; }
public int PercentDone { get; set; }
public List<DefaultData> Children { get; set; }
public string Predecessor { get; set; }
}
}
```

The output of the above steps is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Self-Referential Data Binding (Flat Data)

Gantt can be rendered from self-referential data structures, by mapping the task ID and parent task ID fields.

- Task ID field- This field must contain unique values to identify the nodes. It should be mapped to the `TaskIdMapping` property.
- Parent task ID field- This field must contain values to identify the parent nodes. It should be mapped to the `ParentTaskIdMapping` property.

ASPX-CS

```
@(Html.EJ().Gantt("ganttsample2")
.TaskIdMapping("Id")
.TaskNameMapping("Name")
.StartDateMapping("StartDate")
.DurationMapping("Duration")
.ProgressMapping("PercentDone")
.ParentTaskIdMapping("ParentId")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

```
public class GanttController : Controller
{
//
// GET: /GanttSelfReference/
public ActionResult GanttSelfReference()
{
var DataSource = GanttSelfData.GetData();
ViewBag.datasource = DataSource;
return View();
}
```

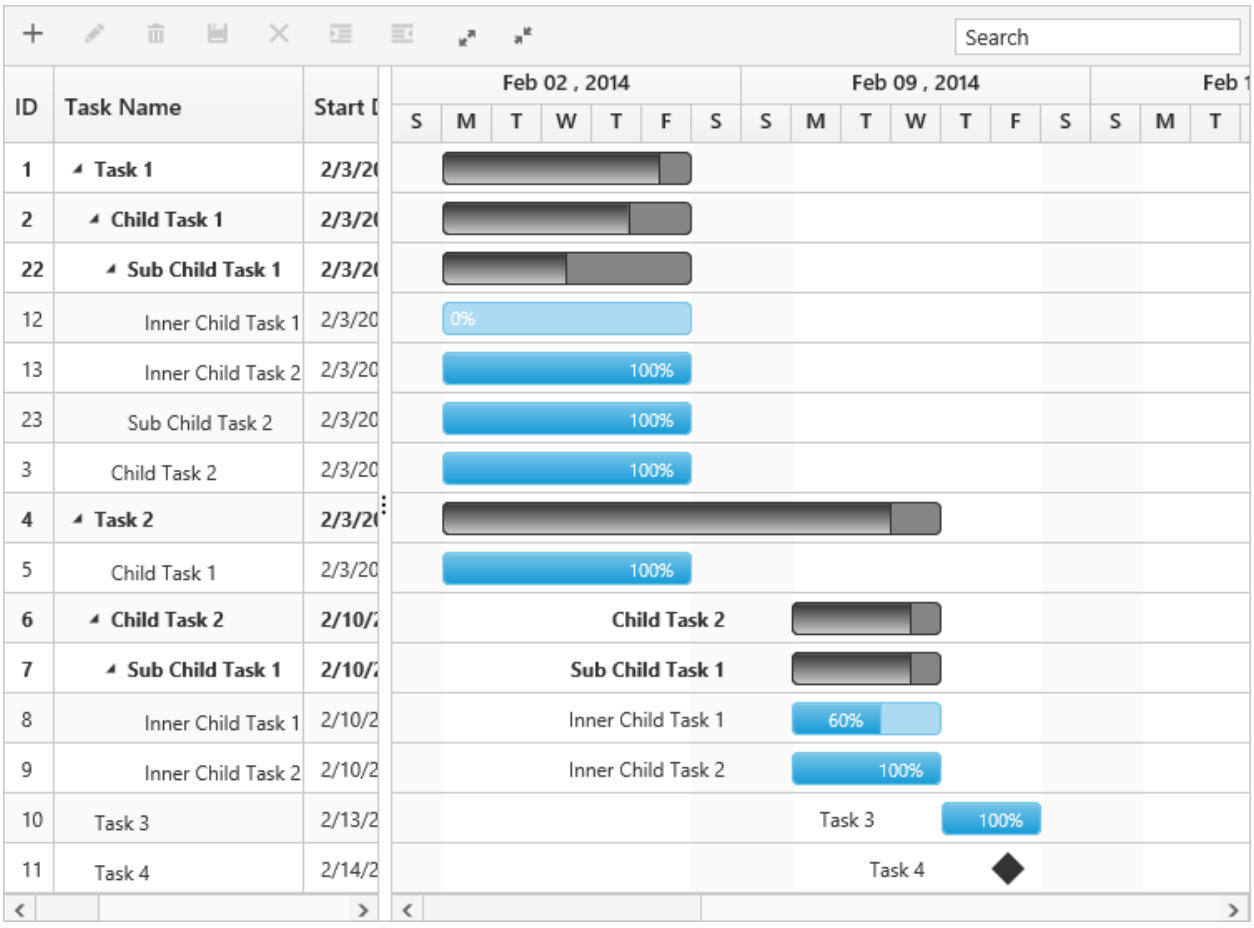
Copyright © 2001 - 2020 Syncfusion Inc.

```
public class GanttSelfData
{
public static List<SelfData> GetData()
{
List<SelfData> data = new List<SelfData>();
data.Add(new SelfData()
{
Id = 1,
Name = "Task 1",
StartDate = "02/03/2014",
Duration = 5,
PercentDone = 40,
ParentId = null
});
data.Add(new SelfData()
{
Id = 2,
Name = "Child Task 1",
StartDate = "02/03/2014",
Duration = 5,
PercentDone = 40,
ParentId = 1
});
data.Add(new SelfData()
{
Id = 3,
Name = "Child Task 2",
StartDate = "02/03/2014",
Duration = 5,
PercentDone = 40,
ParentId = 1
});
data.Add(new SelfData()
{
Id = 22,
Name = "Sub Child Task 1",
StartDate = "02/03/2014",
Duration = 5,
PercentDone = 40,
ParentId = 2
});
data.Add(new SelfData()
{
Id = 23,
Name = "Sub Child Task 2",
StartDate = "02/03/2014",
Duration = 5,
PercentDone = 40,
ParentId = 2
});
data.Add(new SelfData()
{
Id = 12,
Name = "Inner Child Task 1",
StartDate = "02/03/2014",
Duration = 5,
PercentDone = 40,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ParentId = 5
});
//..
return data;
}
}
public class SelfData
{
public string StartDate { get; set; }
public int Id { get; set; }
public string Name { get; set; }
public int Duration { get; set; }
public int PercentDone { get; set; }
public string Predecessor { get; set; }
public int? ParentId { get; set; }
}
}
```

The following screenshot shows the output of the above steps.



You can find the online demo sample for binding self-referential data here

Remote data

OData

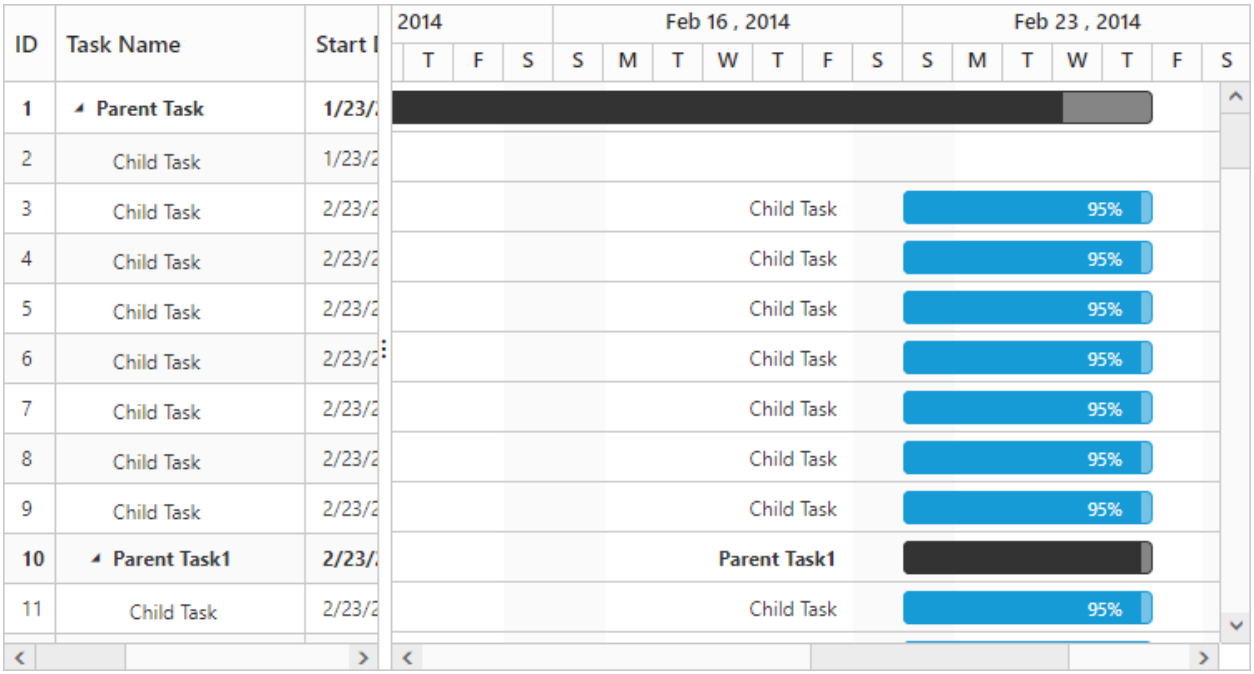
OData is a standardized protocol for creating and consuming data. You can provide the OData service URL directly to the ej.DataManager class and then you can assign it to Gantt dataSource.

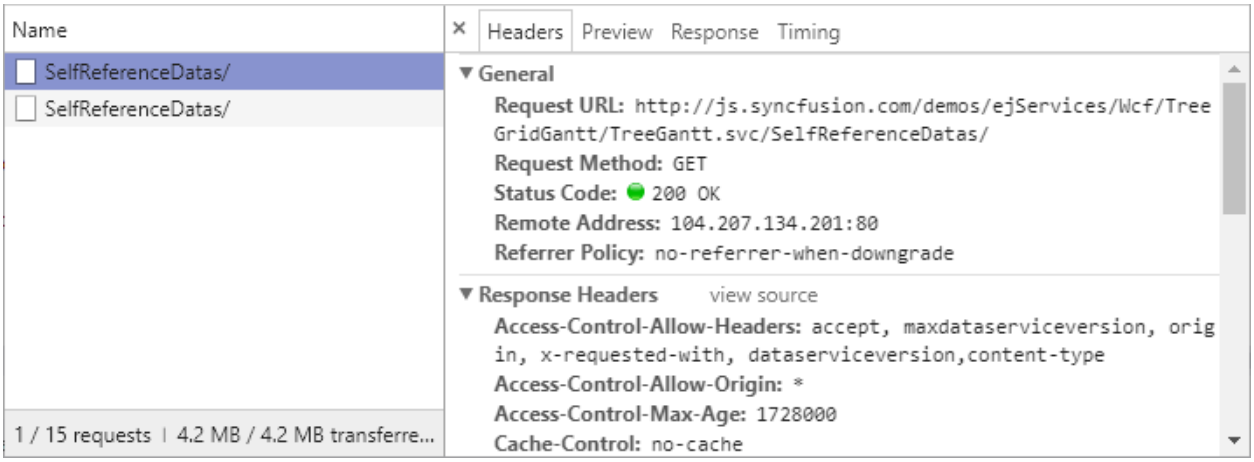
The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.ParentTaskIdMapping("ParentID")
.StartDateMapping("StartDate")
.EndDateMapping("EndDate")
.Datasource(ds =>
ds.URL("http://js.syncfusion.com/demos/ejServices/Wcf/TreeGridGantt/TreeGantt.svc/SelfReferenceDatas"))
)
@(Html.EJ().ScriptManager())
```

The following output is displayed for the code above,





WebAPI

You can bind WebApi service data to Gantt. The data from WebApi service must be returned as object that has property Items with its value as datasource and this object can be pass to dataSource property of Gantt control.

The following code example describes the above behavior.

ASPX-CS

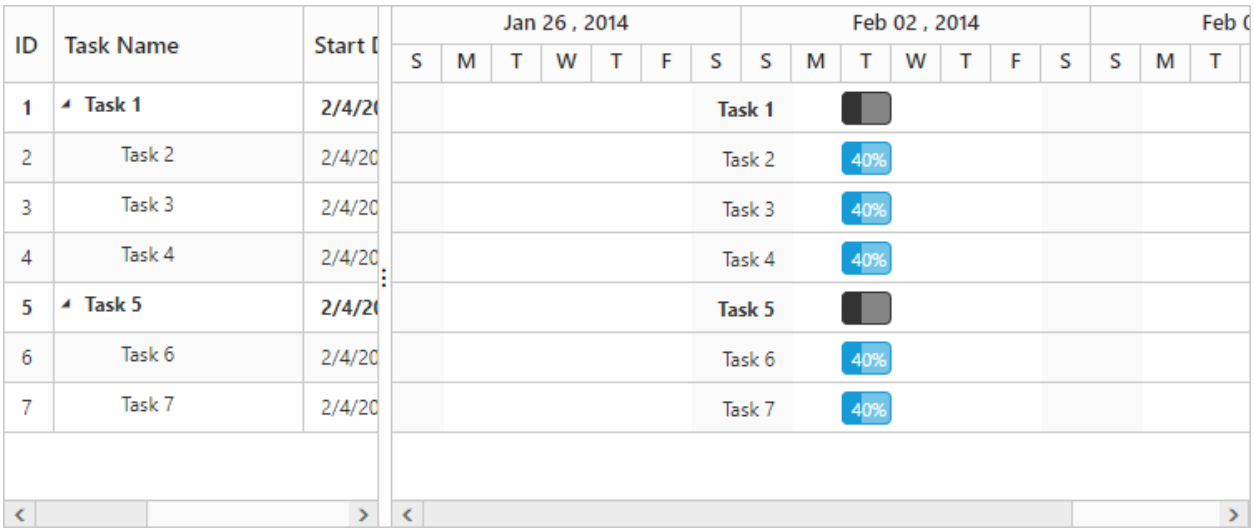
```
@(Html.EJ().Gantt("GanttContainer")
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.ParentTaskIdMapping("ParentID")
.StartDateMapping("StartDate")
.EndDateMapping("EndDate")
.Datasource(ds => ds.URL("api/Home/GetGanttData"))
)
@(Html.EJ().ScriptManager())
```

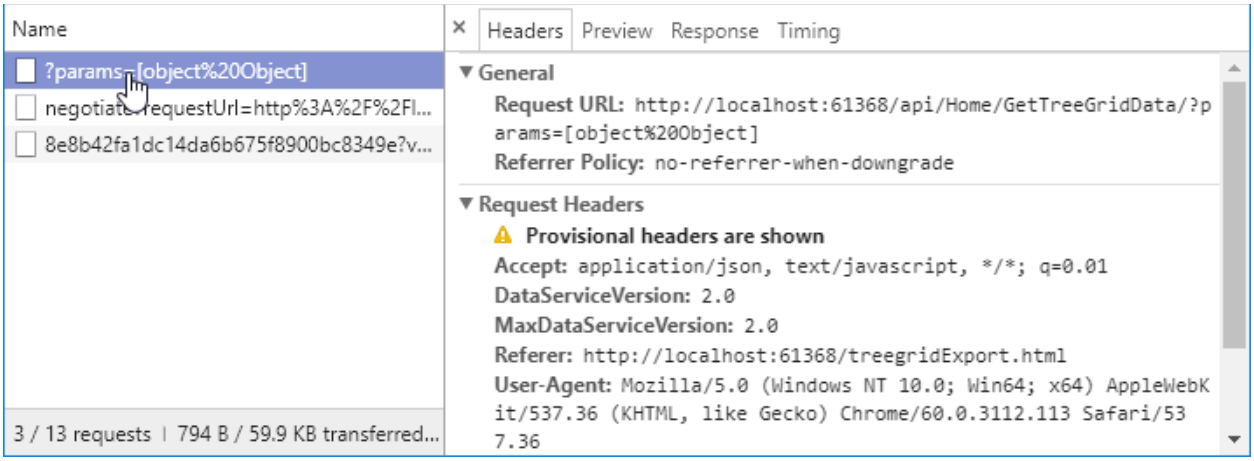
C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using GanttExportService.Models;
namespace GanttExportService {
public class HomeController: ApiController {
GanttDataEntities data = new GanttDataEntities();
public object GetGanttData() {
var Data = data.Table1.ToList();
return Data;
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.





Virtualization mode in Gantt

Virtualization support was used to render large number tasks in Gantt with effective performance, in this mode all the tasks are fetched from data source initially then some of the records are rendered in DOM which are compact to the current viewport area.

While scrolling tasks are updated in DOM as per current viewport position.

This mode can be enabled by setting `EnableVirtualization` property as `true`.

The below code example shows how to use this property.

ASPX-CS

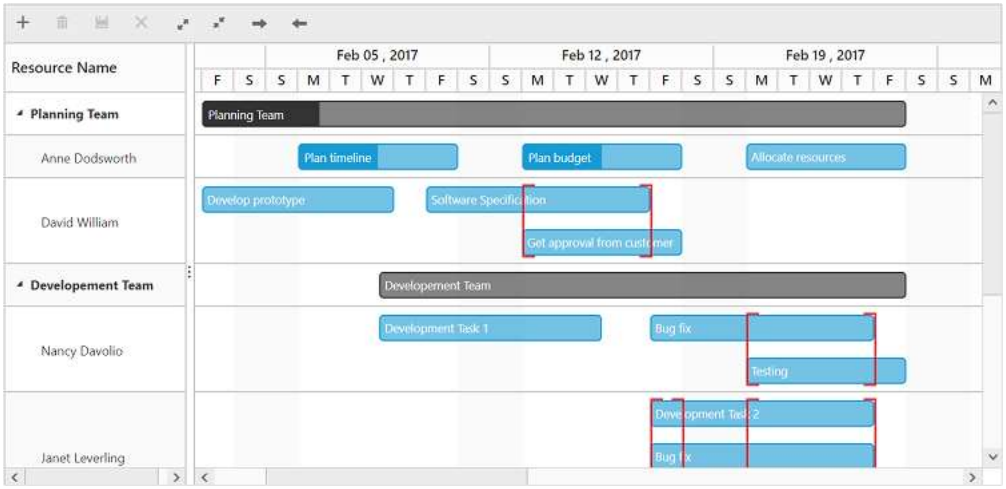
```
@(Html.EJ().Gantt("GanttContainer")
.EnableVirtualization(true)
//..
)
@(Html.EJ().ScriptManager())
```

You can find the online demo sample of Gantt with virtualization mode here.

Copyright © 2001 - 2020 Syncfusion Inc.

Resource Allocation View

Resource allocation view is used to visualize the resource usage in a project. Resources are displayed in separate rows and all the tasks assigned to a specific resource will be displayed in the chart side. In this view, a user can able to easily identify the overallocation of resources in a project.



Data Binding

User can bind two types of data structure to Gantt in resource allocation view,

- Hierarchical data source binding
- Flat data binding

Hierarchical data source binding

The following code example explains how to bind the hierarchical data.

ASPX-CS

```
@(Html.EJ().Gantt("ResourceViewGantt")
.ViewType(GanttViewType.ResourceView)
.TaskCollectionMapping("Tasks")
.ResourceNameMapping("ResourceName")
//...
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

```
public class TaskDetails
{
public int TaskID { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
public string EndDate { get; set; }
public int Duration { get; set; }
public string Progress { get; set; }
public List<TaskDetails> SubTasks { get; set; }
public List<object> ResourceID { get; set; }
public string Predecessors { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public class ResourceDetails
{
public string ResourceName { get; set; }
public List<TaskDetails> Tasks { get; set; }
}
public List<ResourceDetails> GetResourceData()
{
List<ResourceDetails> data = new List<ResourceDetails>() {
new ResourceDetails()
{
ResourceName = "Anne Dodsworth",
Tasks = new List<TaskDetails>()
{
new TaskDetails() { TaskID = 1, TaskName= "Plan timeline", StartDate=
"02/06/2017", EndDate = "02/10/2017", Duration= 5, Progress= "50"},
new TaskDetails() { TaskID= 2, TaskName= "Plan budget", StartDate=
"02/13/2017", EndDate= "02/17/2017", Duration= 5, Progress= "50"},
new TaskDetails() { TaskID= 3, TaskName= "Allocate resources", StartDate=
"02/20/2017", EndDate= "02/24/2017", Duration= 5, Progress= "0"}
}
}
};
//...
return data;
}
public ActionResult ResourceView()
{
ViewBag.datasource = GetResourceData();
return View();
}
```

Flat data binding

The following code example explains how to bind the flat data.

ASPX-CS

```
@(Html.EJ().Gantt("ResourceViewGantt")
.ViewType(GanttViewType.ResourceView)
.ResourceIdMapping("ResourceId")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.TaskIdMapping("TaskID")
//...
.Resources(ViewBag.resources)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

```
public class TaskDetails
{
public int TaskID { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public string EndDate { get; set; }
public int Duration { get; set; }
public string Progress { get; set; }
public string Priority { get; set; }
public bool Approved { get; set; }
public List<TaskDetails> SubTasks { get; set; }
public List<Object> ResourceID{ get; set; }
public bool IsManual { get; set; }
}
public class GanttResources
{
public string ResourceName { get; set; }
public int ResourceId { get; set; }
public int TeamId { get; set; }
}
public List<GanttResources> GetResources()
{
List<GanttResources> data = new List<GanttResources>(){
new GanttResources(){ ResourceId = 1, ResourceName = "Anne Dodsworth" },
new GanttResources(){ ResourceId = 2, ResourceName = "David William"},
new GanttResources(){ ResourceId = 3, ResourceName = "Nancy Davolio"},
//...
};
return data;
}
public List<TaskDetails> GetResourceFlatData()
{
List<TaskDetails> data = new List<TaskDetails>() {
new TaskDetails() { TaskID = 1, TaskName= "Plan timeline", StartDate=
"02/06/2017", EndDate = "02/10/2017", Duration= 5, Progress= "50",
ResourceID= new List<object>(){1}},
new TaskDetails() { TaskID= 2, TaskName= "Plan budget", StartDate=
"02/13/2017", EndDate= "02/17/2017", Duration= 5, Progress= "50",
ResourceID= new List<object>(){1}},
new TaskDetails() { TaskID= 3, TaskName= "Allocate resources", StartDate=
"02/20/2017", EndDate= "02/24/2017", Duration= 5, Progress= "0", ResourceID=
new List<object>(){1}},
//..
}
return data;
}
public ActionResult ResourceView()
{
ViewBag.datasource = GetResourceFlatData();
ViewBag.resources = GetResources();
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Grouping

The resources and the tasks assigned to those resources can be grouped into categories. The property GroupCollection is used to define the groups to be displayed in the project. The properties GroupIdMapping and GroupNameMapping are used to map the group details.

The below code snippet explains grouping of the resources in resource allocation view.

ASPX-CS

```
@(Html.EJ().Gantt("ResourceViewGantt")
.ViewType(GanttViewType.ResourceView)
.ResourceIdMapping("ResourceId")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.GroupNameMapping("TeamName")
.GroupIdMapping("TeamId")
.TaskIdMapping("TaskID")
//...
.GroupCollection(ViewBag.groups)
.Resources(ViewBag.resources)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

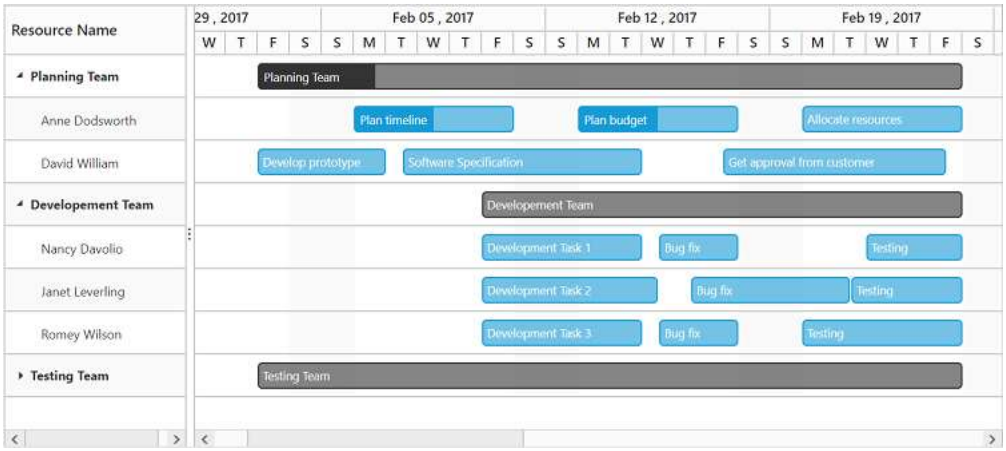
```
public class TaskDetails
{
public int TaskID { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
public string EndDate { get; set; }
public int Duration { get; set; }
public string Progress { get; set; }
public string Priority { get; set; }
public bool Approved { get; set; }
public List<TaskDetails> SubTasks { get; set; }
public List<Object> ResourceID{ get; set; }
public bool IsManual { get; set; }
}
public class GanttResources
{
public string ResourceName { get; set; }
public int ResourceId { get; set; }
public int TeamId { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public List<GanttResources> GetResources()
{
List<GanttResources> data = new List<GanttResources>(){
new GanttResources(){ ResourceId = 1, ResourceName = "Anne
Dodsworth",TeamId=1 },
new GanttResources(){ ResourceId = 2, ResourceName = "David William",
TeamId=1},
new GanttResources(){ ResourceId = 3, ResourceName = "Nancy Davolio",
TeamId=2},
//...
};
return data;
}
public class ResourceGroupDetails
{
public string TeamName { get; set; }
public int TeamId { get; set; }
}
public List<ResourceGroupDetails> GetResourceGroups()
{
List<ResourceGroupDetails> data = new List<ResourceGroupDetails>(){
new ResourceGroupDetails(){ TeamName="Planning Team", TeamId=1},
new ResourceGroupDetails(){ TeamName="Development Team", TeamId=2},
new ResourceGroupDetails(){ TeamName="Testing Team", TeamId=3}
};
return data;
}
public List<TaskDetails> GetResourceFlatData()
{
List<TaskDetails> data = new List<TaskDetails>() {
new TaskDetails() { TaskID = 1, TaskName= "Plan timeline", StartDate=
"02/06/2017", EndDate = "02/10/2017", Duration= 5, Progress= "50",
ResourceID= new List<object>(){1}},
new TaskDetails() { TaskID= 2, TaskName= "Plan budget", StartDate=
"02/13/2017", EndDate= "02/17/2017", Duration= 5, Progress= "50",
ResourceID= new List<object>(){1}},
new TaskDetails() { TaskID= 3, TaskName= "Allocate resources", StartDate=
"02/20/2017", EndDate= "02/24/2017", Duration= 5, Progress= "0", ResourceID=
new List<object>(){1}},
//..
}
return data;
}
public ActionResult ResourceView()
{
ViewBag.datasource = GetResourceFlatData();
ViewBag.resources = GetResources();
ViewBag.groups = GetResourceGroups();
return View();
}
```

The below screen shot depicts a resource view in a project with 3 groups.

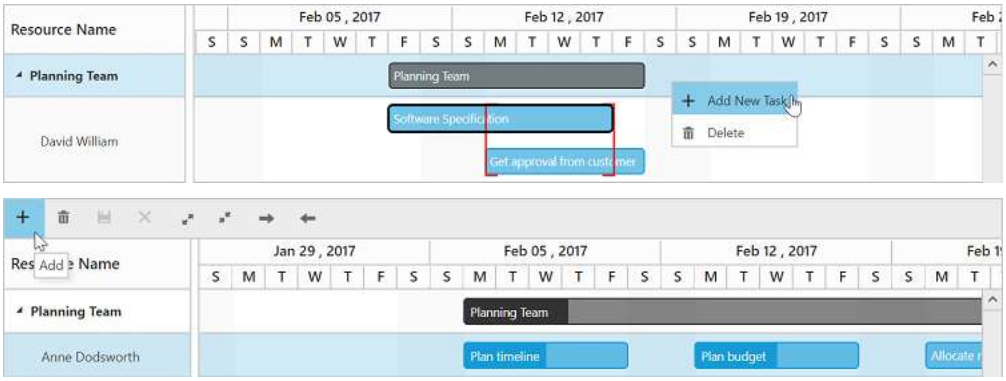
Copyright © 2001 - 2020 Syncfusion Inc.



CRUD data operations

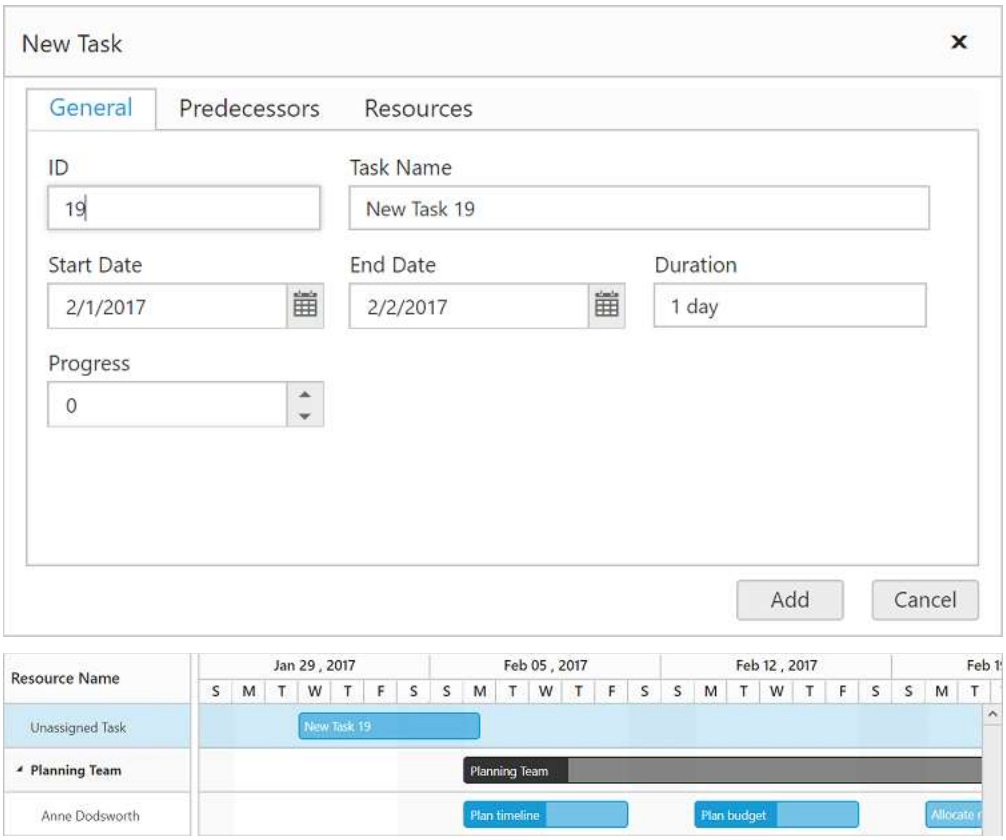
Adding a task

It is possible to add a task dynamically either by using toolbar insert icon or by context menu add option. The below screen shot shows the toolbar add and context menu add options.

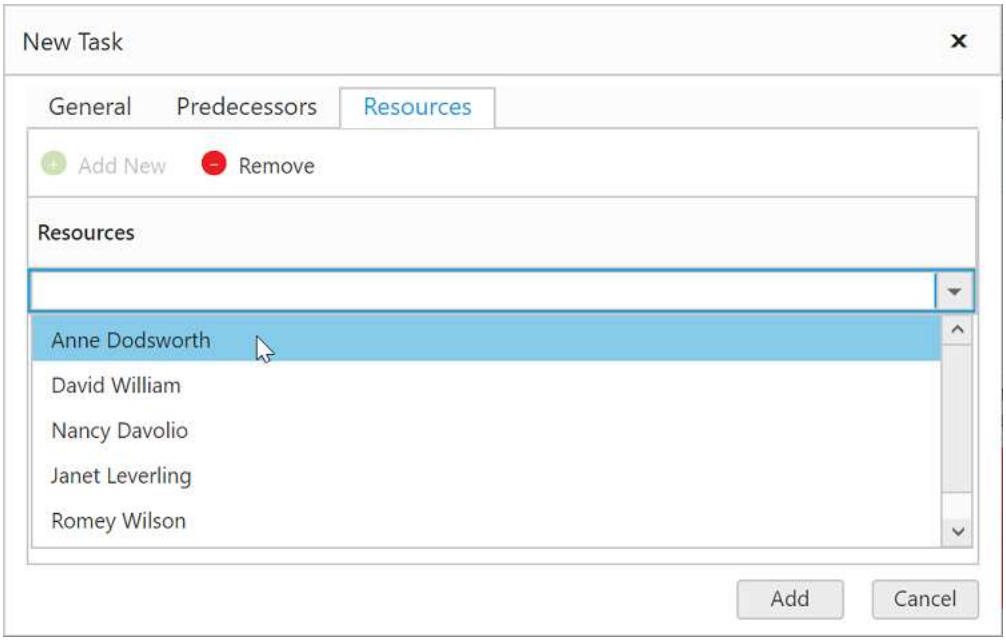


When the insert option is clicked, the add dialog will be displayed prompting the user to provide the task details. After providing the task details, when user click the save button without providing the resource information or without assigning any resource to the task, the task will be added at the top by default.

Copyright © 2001 - 2020 Syncfusion Inc.

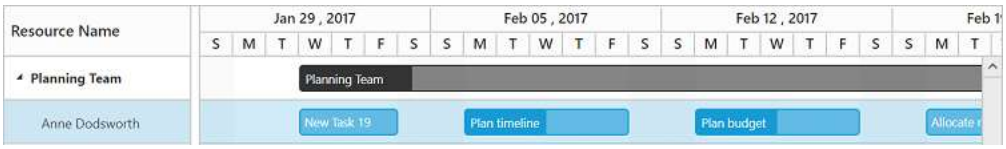


If the newly added task should be assigned to a specific resource, user can able to select the resource information in the resource tab of add dialog.



The below screen shot depicts a new task being assigned to a resource Anne Dodsworth.

Copyright © 2001 - 2020 Syncfusion Inc.



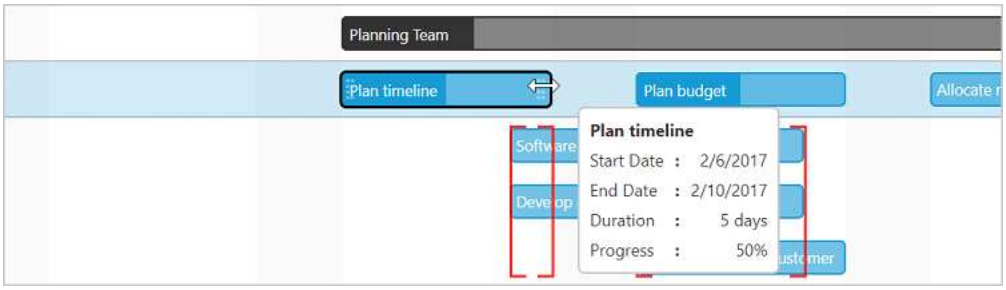
Note: It is not possible to add a resource to the project dynamically.

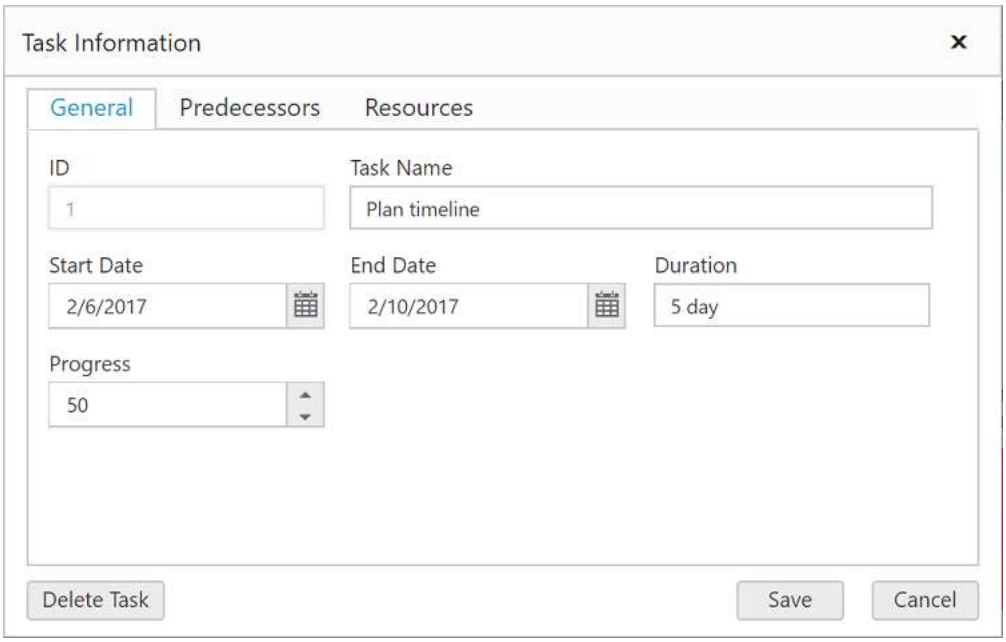
A task can be assigned to one or more resource while adding.

Editing

Editing a Task

A task can be edited either by using edit dialog or by mouse interactions. A user can edit a task using edit dialog by performing double click action on the taskbar, user can then edit the task information in the edit dialog.





A task's schedule can be changed by left resizing, right resizing or dragging the task across the dates.



Copyright © 2001 - 2020 Syncfusion Inc.

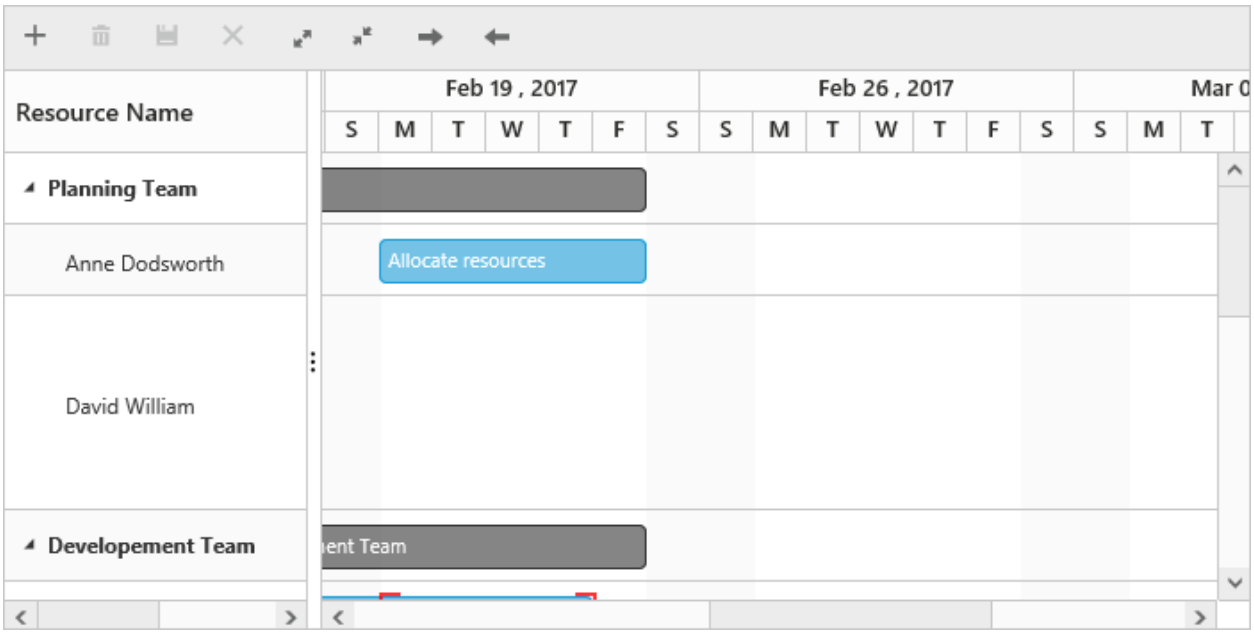
Update task values dynamically

In resource allocation view also, we can update the task details dynamically on any action like external button click by using updateRecordByTaskId method. The below code example explains how to use this method in resource allocation view.

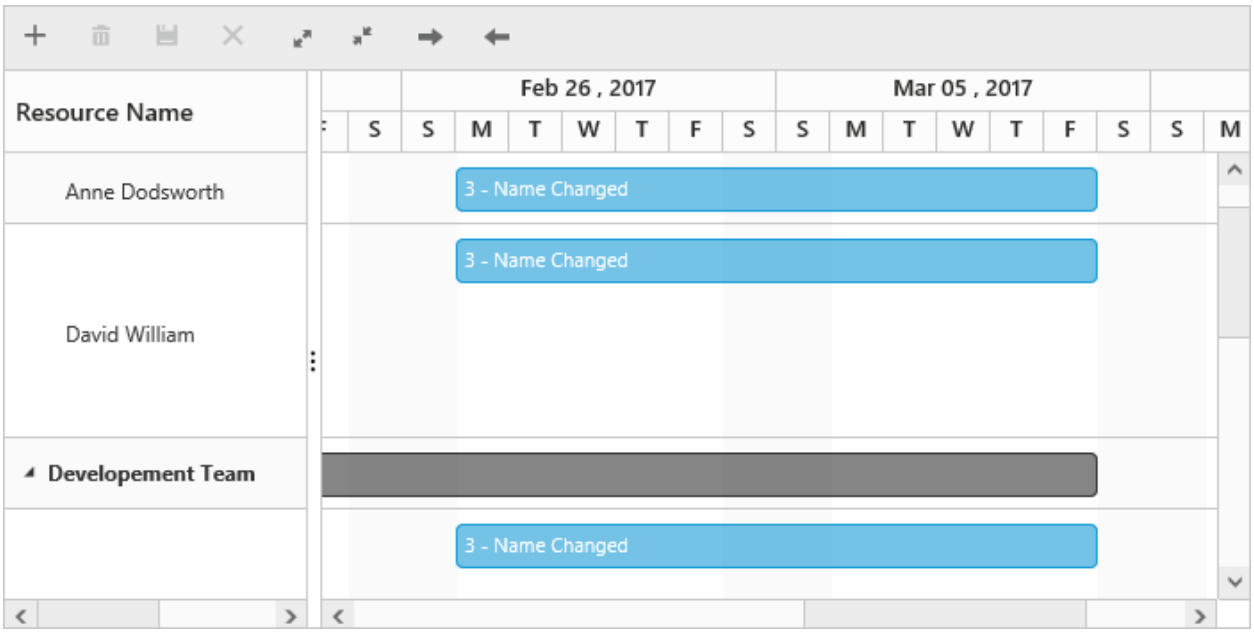
ASPX-CS

```
@(Html.EJ().Gantt("ResourceViewGantt")
.ViewType(GanttViewType.ResourceView)
//...
)
<button id="update">Update Tasks</button>
<script type="text/javascript">
$("#update").click(function () {
var ganttObject = $("#ResourceViewGantt").ejGantt("instance");
var task = {};
task.TaskID = 3;
task.ResourceID = [1, 2, 3];
task.StartDate = new Date("02/25/2017")
task.Duration = 10;
task.TaskName = task.TaskID + " - Name Changed";
ganttObject.updateRecordByTaskId(task);
});
</script>
```

The below screenshot shows the result of above code example.



Before update

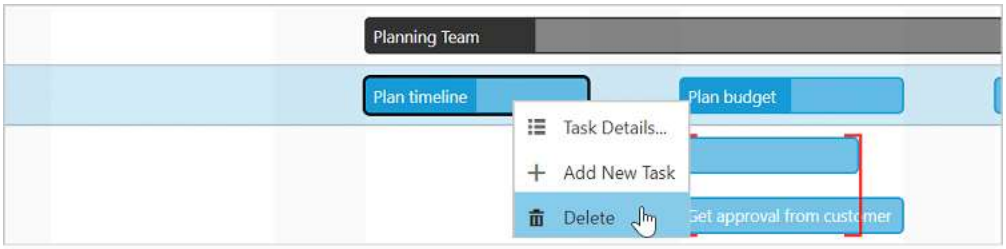


After update

Deleting

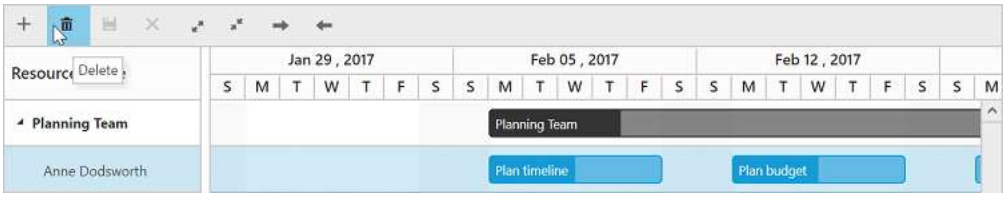
Deleting Task

User should select a task to be removed from the project, and then the task can be deleted using context menu delete button.



Deleting Resource

User can able to delete a resource and the tasks assigned to it using toolbar and context menu delete options. After selecting the resource row, when clicking the delete toolbar icon, the resource will be removed from the project and the tasks assigned to that resource will be listed in unassigned tasks' list. The same will happen for context menu delete option.



The below screen shot depicts the tasks of the removed resource added as unassigned tasks to the project.

Copyright © 2001 - 2020 Syncfusion Inc.



Unassigned Tasks and Resources

There may be some instances, where one or more resources are not yet included in the project plan. Therefore, those resources are left with no tasks assigned to them. Those resources are termed as unassigned resource. The right side of the resource row will be left blank with no tasks.

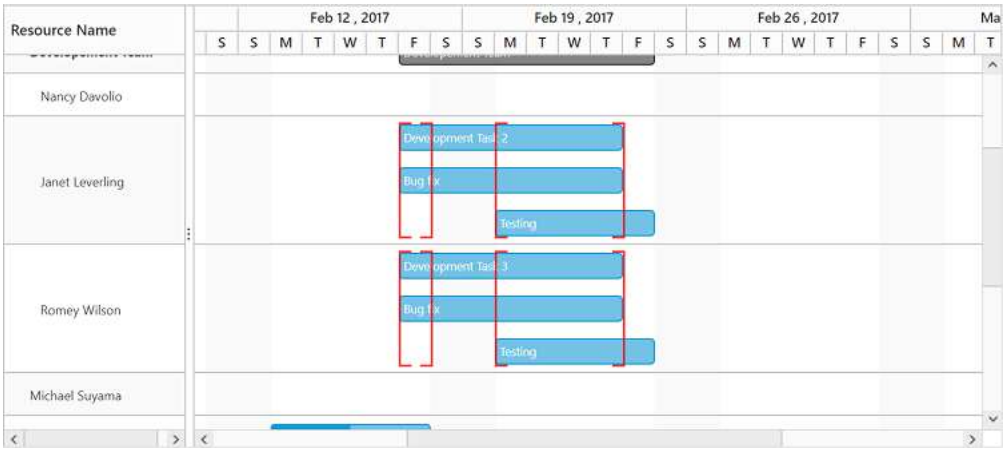
The below code snippet shows a resource with no tasks assigned to it.

C#

```
public List<ResourceDetails> GetResourceData()
{
List<ResourceDetails> data = new List<ResourceDetails>() {
new ResourceDetails()
{
ResourceName = "Anne Dodsworth",
Tasks = new List<TaskDetails>()
{
new TaskDetails() { TaskID = 1, TaskName= "Plan timeline", StartDate=
"02/06/2017", EndDate = "02/10/2017", Duration= 5, Progress= "50"},
new TaskDetails() { TaskID= 2, TaskName= "Plan budget", StartDate=
"02/13/2017", EndDate= "02/17/2017", Duration= 5, Progress= "50"},
new TaskDetails() { TaskID= 3, TaskName= "Allocate resources", StartDate=
"02/20/2017", EndDate= "02/24/2017", Duration= 5, Progress= "0"}
}
},
new ResourceDetails()
{
ResourceName = "Nancy Davolio",
Tasks = new List<TaskDetails>() {}
}
//...
};
return data;
}
```

In the below screen shot, the resource Nancy Davolio is the unassigned resource.

Copyright © 2001 - 2020 Syncfusion Inc.



There may be instances where some tasks are not yet assigned with resources. Those tasks that are not assigned to any resources are termed as unassigned tasks. Unassigned tasks are displayed at bottom, with resource name as Unassigned Task. Unassigned tasks can be assigned to any resources in the project.

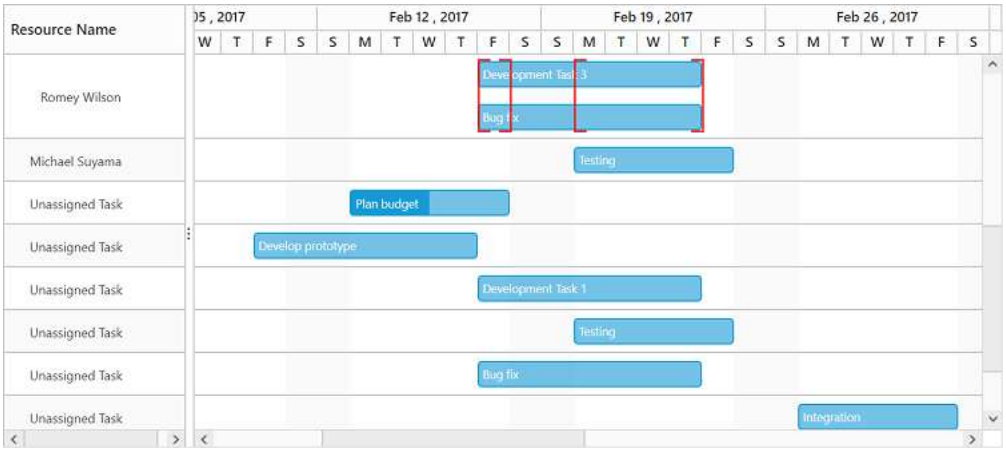
The below code snippet shows a task with no resource assigned to it.

C#

```
public class GanttResources
{
public string ResourceName { get; set; }
public int ResourceId { get; set; }
public int TeamId { get; set; }
}
public List<GanttResources> GetResources()
{
List<GanttResources> data = new List<GanttResources>(){
new GanttResources(){ ResourceId = 1, ResourceName = "Anne
Dodsworth",TeamId=1 },
new GanttResources(){ ResourceId = 2, ResourceName = "David William",
TeamId=1},
new GanttResources(){ ResourceId = 3, ResourceName = "Nancy Davolio",
TeamId=2},
//...
};
return data;
}
public List<TaskDetails> GetResourceFlatData()
{
List<TaskDetails> data = new List<TaskDetails>() {
//...
new TaskDetails() { TaskID= 2, TaskName= "Plan budget", StartDate=
"02/13/2017", EndDate= "02/17/2017", Duration= 5, Progress= "50"},
//..
}
return data;
}
```

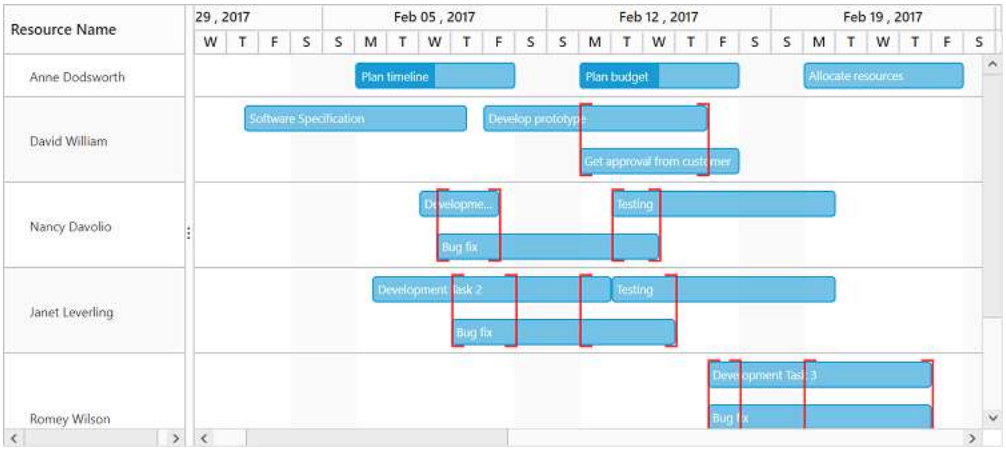
In the below screen shot, there more instances of unassigned tasks.

Copyright © 2001 - 2020 Syncfusion Inc.



Resource Overallocation

By default, a resource can work 8 hours a work day, but when a resource is forced to work more than 8 hours it is termed as overallocation. Or when a resource is assigned with two or more tasks which is occurring in a same date range, this state is termed as over allocation for a resource. The overlapped tasks are highlighted like in the below screen shot. The overlapped tasks will be displayed one below the other with an overlapped highlighted region.



Resource Histogram

A resource histogram displays the scheduled working time range of resources for its assigned tasks. Using this, you can easily identify the resource's availability and its working time. The resource histogram is used to allocate the resources properly throughout the project and find the details about allocated resources and the allocated time ranges of a specific resource.

Databinding

You can render the histogram view for both project view Gantt and resource view Gantt by using the `ViewType` as `GanttViewType.HistogramView`.

The following code example explains how to bind the resource histogram view for project view Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
```

```
.DurationMapping("Duration")
.ProgressMapping("Progress")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.ResourceIdMapping("ResourceID")
.ChildMapping("SubTasks")
.ViewType(GanttViewType.ProjectView)
//..
.Resources(ViewBag.resources)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().Gantt("HistogramContainer")
.ViewType(GanttViewType.HistogramView)
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
.DurationMapping("Duration")
.ProgressMapping("Progress")
.ChildMapping("SubTasks")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.ResourceIdMapping("ResourceID")
//..
.Resources(ViewBag.resources)
.Datasource(ViewBag.histogramData)
)
@(Html.EJ().ScriptManager())
```

Note: You should bind the data source and necessary mapping properties to render the histogram view. You can differentiate whether the bound data source is project view data or resource view data using the ganttObject.isProjectViewData boolean property and Load event. Set this property to true if the provided data is project view and set to false for resource view data.

Synchronize resource histogram view Gantt with other views

You can synchronize the splitter position and horizontal scroller position in chart side and task values of the project view/resource view Gantt control with resource histogram view Gantt by using the ActionComplete and SplitterResized events.

Synchronize with project view

The following code snippet shows how to synchronize the resource histogram view Gantt with project view Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
.DurationMapping("Duration")
.ProgressMapping("Progress")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.ResourceIdMapping("ResourceID")
.ChildMapping("SubTasks")
.ViewType(GanttViewType.ProjectView)
```

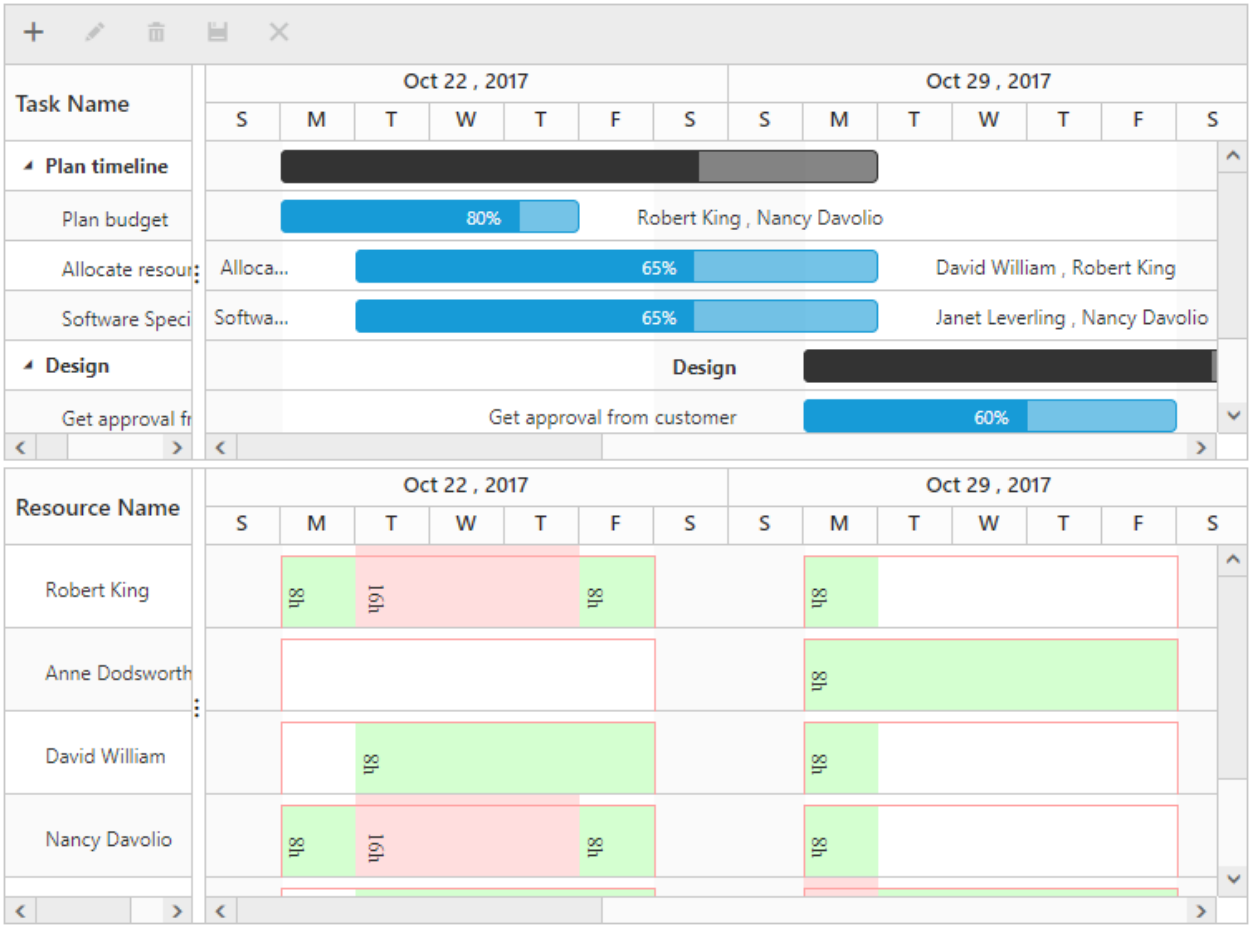
Copyright © 2001 - 2020 Syncfusion Inc.

```
.ClientSideEvents(cs => {
cs.SplitterResized("splitterResized").ActionComplete("actionComplete");
})
//..
.Resources(ViewBag.resources)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().Gantt("HistogramContainer")
.ViewType(GanttViewType.HistogramView)
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
.DurationMapping("Duration")
.ProgressMapping("Progress")
.ChildMapping("SubTasks")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.ResourceIdMapping("ResourceID")
.ClientSideEvents(cs => {
cs.Load("load").SplitterResized("splitterResized").ActionComplete("actionCom
plete");
})
//..
.Resources(ViewBag.resources)
.Datasource(ViewBag.histogramData)
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function load(args) {
this.isProjectViewData = true;
}
function splitterResized(args) {
if (args.isOnResize == false) return;
if (this._id == "GanttContainer") {
$("#HistogramContainer").ejGantt("setSplitterPosition",
args.currentSplitterPosition);
} else if (this._id == "HistogramContainer") {
$("#GanttContainer").ejGantt("setSplitterPosition",
args.currentSplitterPosition);
}
}
function actionComplete(args) {
if (args.requestType == "scroll" && args.scrollDirection == "horizontal") {
var scrollLeft = args.scrollLeft;
if (this._id == "GanttContainer" && !args.isScrollByMethod) {
$("#HistogramContainer").ejGantt("setChartScrollLeft", scrollLeft);
} else if (this._id == "HistogramContainer" && !args.isScrollByMethod) {
$("#GanttContainer").ejGantt("setChartScrollLeft", scrollLeft);
}
} else if (args.requestType == "recordUpdate") {
$("#HistogramContainer").ejGantt("updateHistogramTask", args.data,
"update");
if (args.updatedRecords && args.updatedRecords.length > 0) {
for (var count = 0; count < args.updatedRecords.length; count++) {
$("#HistogramContainer").ejGantt("updateHistogramTask",
args.updatedRecords[count], "update");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
} else if (args.requestType == "save" && args.modifiedRecord) {
$("#HistogramContainer").ejGantt("updateHistogramTask", args.modifiedRecord,
"update");
} else if (args.requestType == "save" && args.addedRecord) {
$("#HistogramContainer").ejGantt("updateHistogramTask", args.addedRecord,
"add");
} else if (args.requestType == "delete") {
$("#HistogramContainer").ejGantt("updateHistogramTask", args.data,
"delete");
}
}
</script>
```

The following screenshot shows the resource histogram with project view Gantt.



Please find our online demo sample for further reference

[ResourceHistogram]

Synchronize with resource view

The following code snippet shows how to synchronize the resource histogram view Gantt with resource view Gantt.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

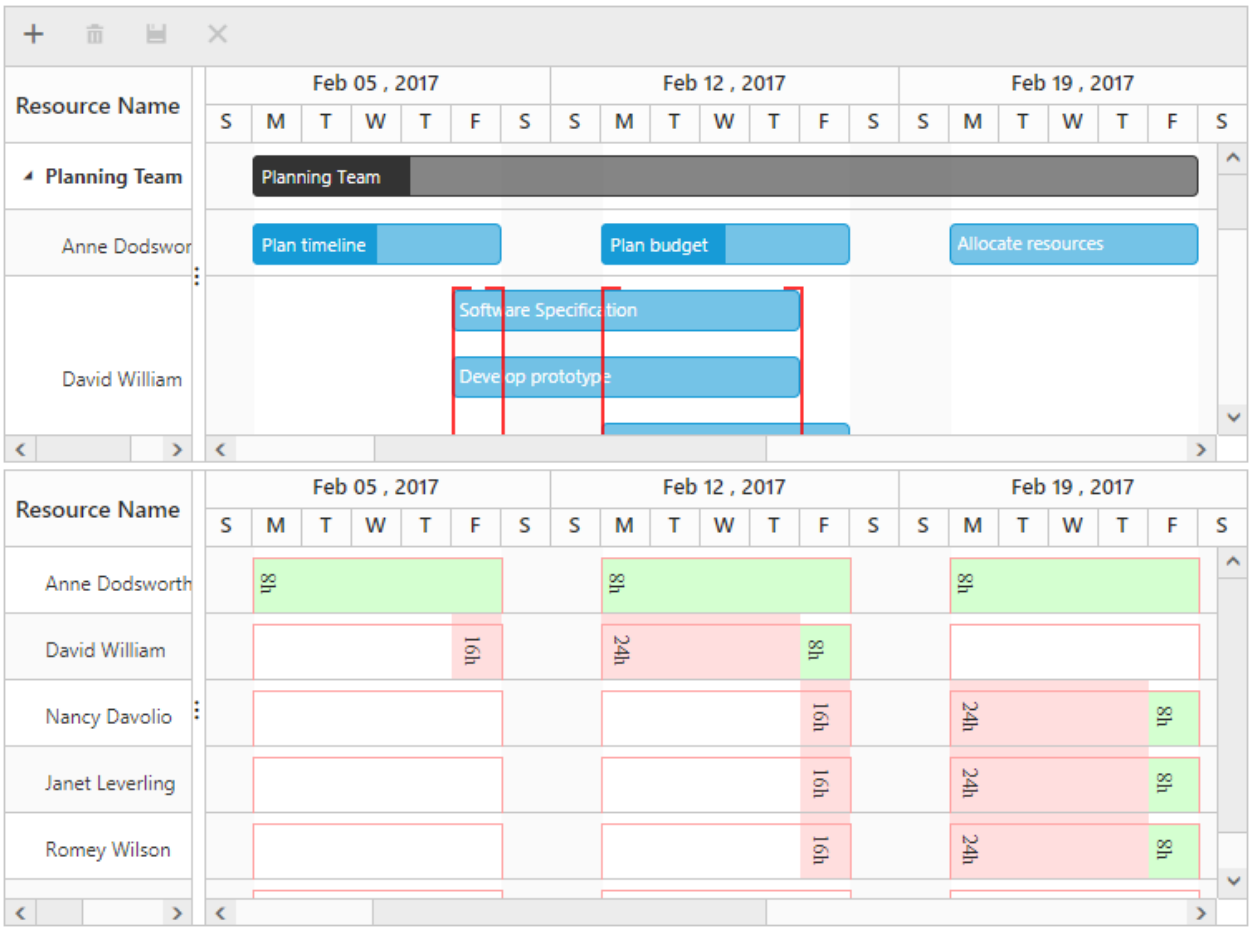
```
@(Html.EJ().Gantt("GanttContainer")
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
.EndDateMapping("EndDate")
.DurationMapping("Duration")
.ProgressMapping("Progress")
.GroupNameMapping("TeamName")
.GroupIdMapping("TeamId")
.ViewType(GanttViewType.ResourceView)
.ResourceIdMapping("ResourceId")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
//..
.ClientSideEvents(cs => {
cs.ActionComplete("actionComplete");
cs.SplitterResized("splitterResized");
})
.GroupCollection(ViewBag.groups)
.Resources(ViewBag.resources)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().Gantt("HistogramContainer")
.TaskIdMapping("TaskID")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
.EndDateMapping("EndDate")
.DurationMapping("Duration")
.ProgressMapping("Progress")
.GroupNameMapping("TeamName")
.GroupIdMapping("TeamId")
.ViewType(GanttViewType.HistogramView)
.ResourceIdMapping("ResourceId")
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
//..
.ClientSideEvents(cs => {
cs.Load("load");
cs.ActionComplete("actionComplete");
cs.SplitterResized("splitterResized");
})
.GroupCollection(ViewBag.groups)
.Resources(ViewBag.resources)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function load(args) {
this.isProjectViewData = false;
}
function splitterResized(args) {
if (args.isOnResize == false) return;
if (this._id == "GanttContainer") {
$("#HistogramContainer").ejGantt("setSplitterPosition",
args.currentSplitterPosition);
} else if (this._id == "HistogramContainer") {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#GanttContainer").ejGantt("setSplitterPosition",
args.currentSplitterPosition);
}
}
function actionComplete(args) {
if (args.requestType == "scroll" && args.scrollDirection == "horizontal") {
var scrollLeft = args.scrollLeft;
if (this._id == "GanttContainer" && !args.isScrollByMethod) {
$("#HistogramContainer").ejGantt("setChartScrollLeft", scrollLeft);
} else if (this._id == "HistogramContainer" && !args.isScrollByMethod) {
$("#GanttContainer").ejGantt("setChartScrollLeft", scrollLeft);
}
}
//task drag and drop action and edit action
else if (args.requestType == "save" && args.modifiedRecord ||
args.requestType == "recordUpdate") {
var data = args.requestType == "save" ? args.modifiedRecord : args.item ?
args.item : args.data;
$("#HistogramContainer").ejGantt("updateHistogramTask", data, "update");
//row delete & group delete
if (args.updatedRecords) {
for (var i = 0; i < args.updatedRecords.length; i++) {
var data = args.updatedRecords[i];
$("#HistogramContainer").ejGantt("updateHistogramTask", data, "update");
}
}
}
//add row
else if (args.requestType == "save" && args.addedRecord) {
$("#HistogramContainer").ejGantt("updateHistogramTask", args.addedRecord,
"add");
}
//task delete
else if (args.requestType == "delete") {
$("#HistogramContainer").ejGantt("updateHistogramTask", args.data,
"delete");
}
}
</script>
```

The following screenshot shows the resource histogram for resource view Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



Editing

The Gantt control provides in-built support to add, insert and update the tasks. The following are the types of editing available in Gantt.

- Cell Editing
- Normal Editing
- Taskbar Editing
- Dependency Editing

Cell Editing

Initialize cell edit mode

Update the task details through grid cell editing by setting `EditSettings.EditMode` as `cellEditing`.

The following code example shows you how to enable the `cellEditing` in Gantt control.

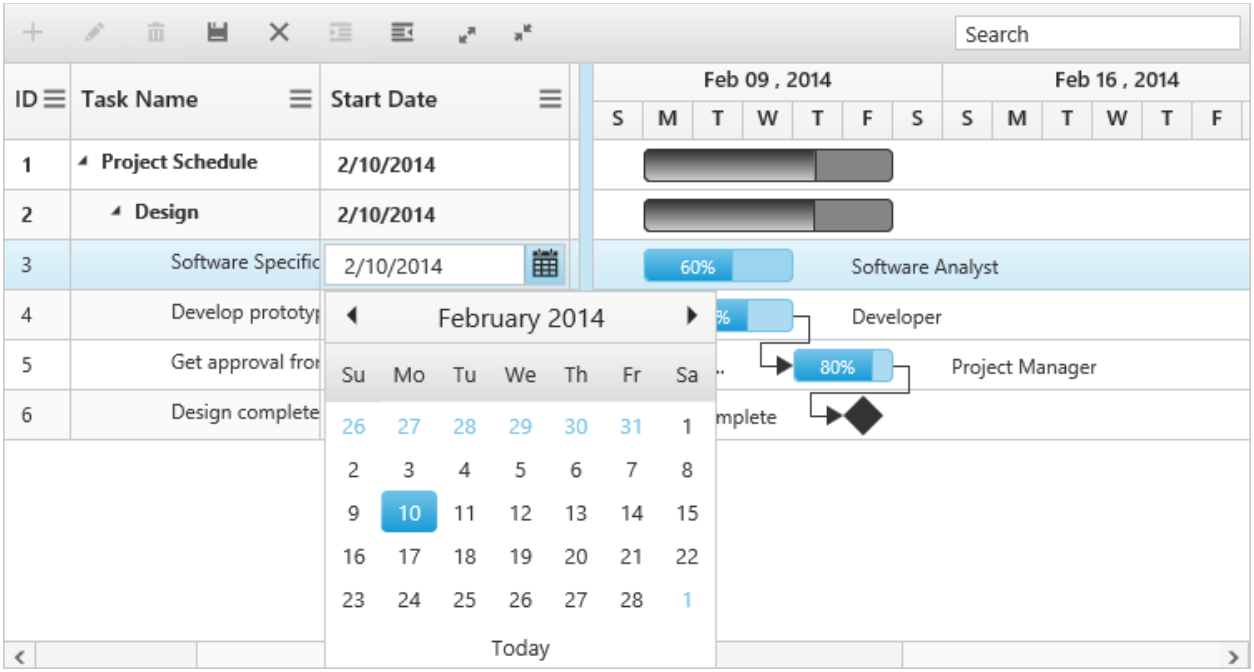
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("cellEditing");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The output of Gantt with cellEditing is as follows.



Save editing cell

Currently editing cell value can be saved by clicking the save toolbar icon or clicking on any other cell element in the grid and also editing cell can be saved by using saveEdit method, using this method we can save the current editing cell dynamically on any action like external button click, please refer the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("cellEditing");
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#save_button").click(function () {
$("#Gantt").ejGantt("instance").saveEdit();
});
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Cancel editing cell

Currently editing cell value can be restored with old value by using cancel toolbar icon or by pressing Esc key. And also this can be done by using cancelEdit method on any other external actions like button click, please refer the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("cellEditing");
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#cancel_button").click(function () {
$("#Gantt").ejGantt("instance").cancelEdit();
});
</script>
```

Prevent Cell Editing

In cell editing action BeginEdit and EndEdit client side events are triggered before and after the editing action. Cell editing for particular column or specific cell can be prevented by using BeginEdit client side event, please refer following code example for this.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("cellEditing");
})
.ClientSideEvents(cs => cs.BeginEdit("beginEdit"))
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
function beginEdit(args) {
if (args.columnIndex == 1)
args.cancel = true;
}
</script>
```

Normal Editing

Initialize Normal edit mode

Update the task details through edit dialog by setting EditMode as normal. The following code example shows you how to enable normal editing in Gantt control.

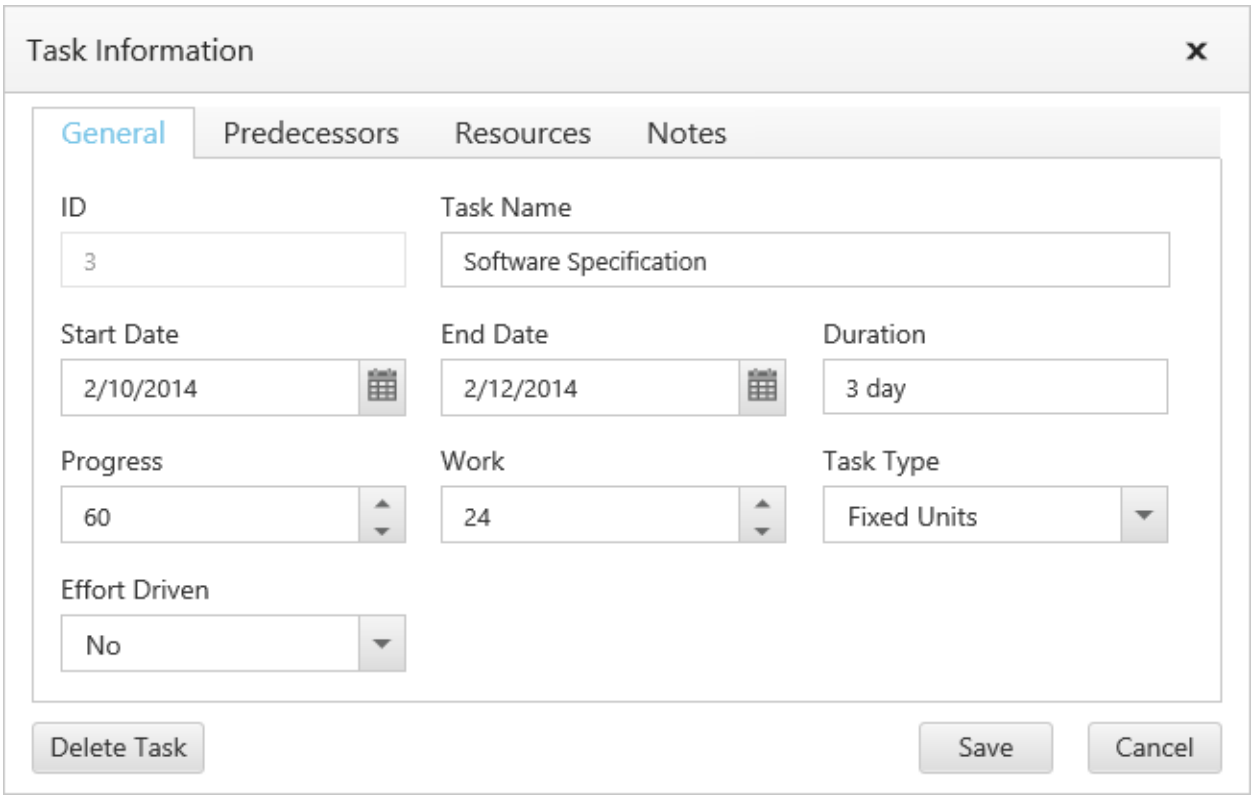
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("normal");
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of normal editing.



Define required fields in add/edit dialog

In Gantt we can define the editing fields available in add and edit dialogs by using AddDialogFields and EditDialogFields properties. Each editing fields are defined with Field and EditType properties. The following code example shows how to define EditDialogFields property.

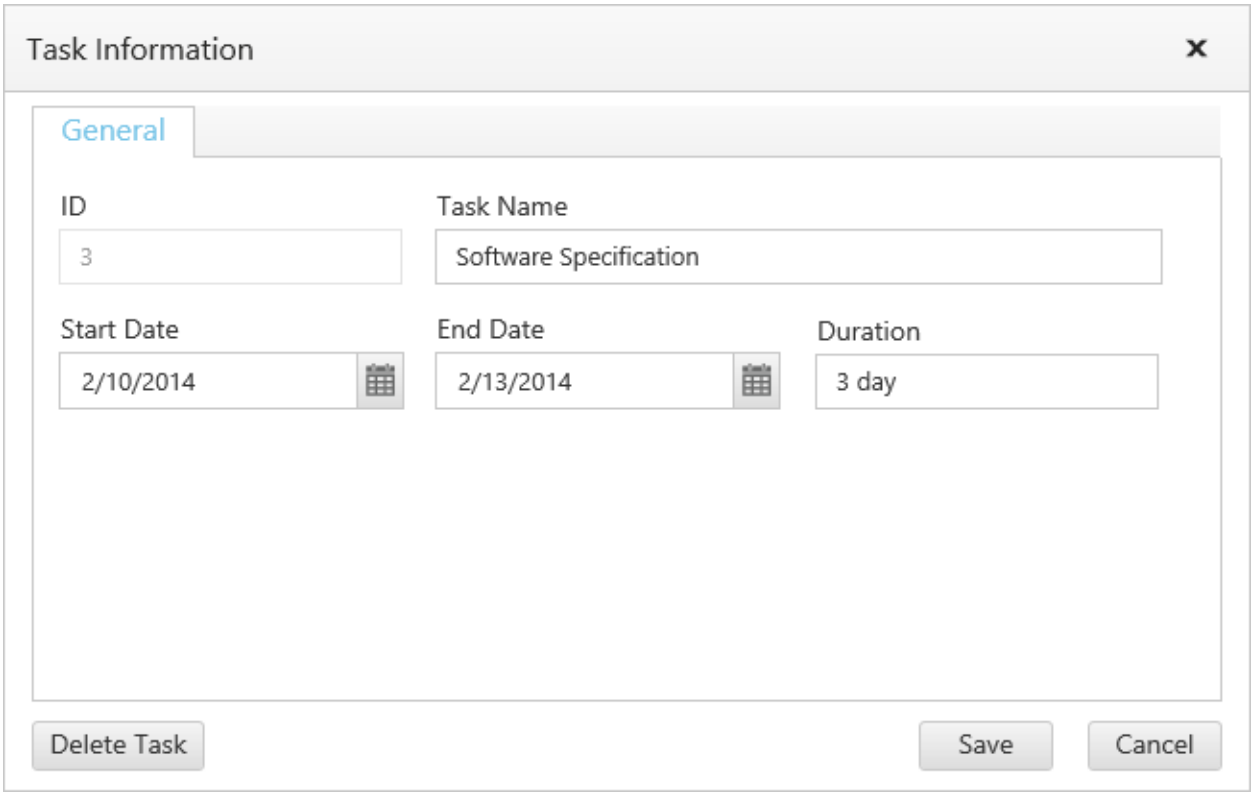
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("normal");
})
.EditDialogFields(editDialog =>
{
editDialog.Field("TaskID").EditType("stringedit").Add();
editDialog.Field("TaskName").EditType("stringedit").Add();
editDialog.Field("StartDate").EditType("datepicker").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
editDialog.Field("EndDate").EditType("datepicker").Add();
editDialog.Field("Duration").EditType("stringedit").Add();
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following screenshot show the output of above code example.



Note: Similarly we can define the required fields in add dialog with AddDialogFields.Field and AddDialogFields.EditType properties.

Add custom column fields in General tab

By default custom column fields are included in add/edit dialog's custom field tab but we can display this column field in General tab by setting DisplayInGeneralTab property as true in AddDialogFields and EditDialogFields properties. The following code example shows how to add custom column field in General tab of edit dialog.

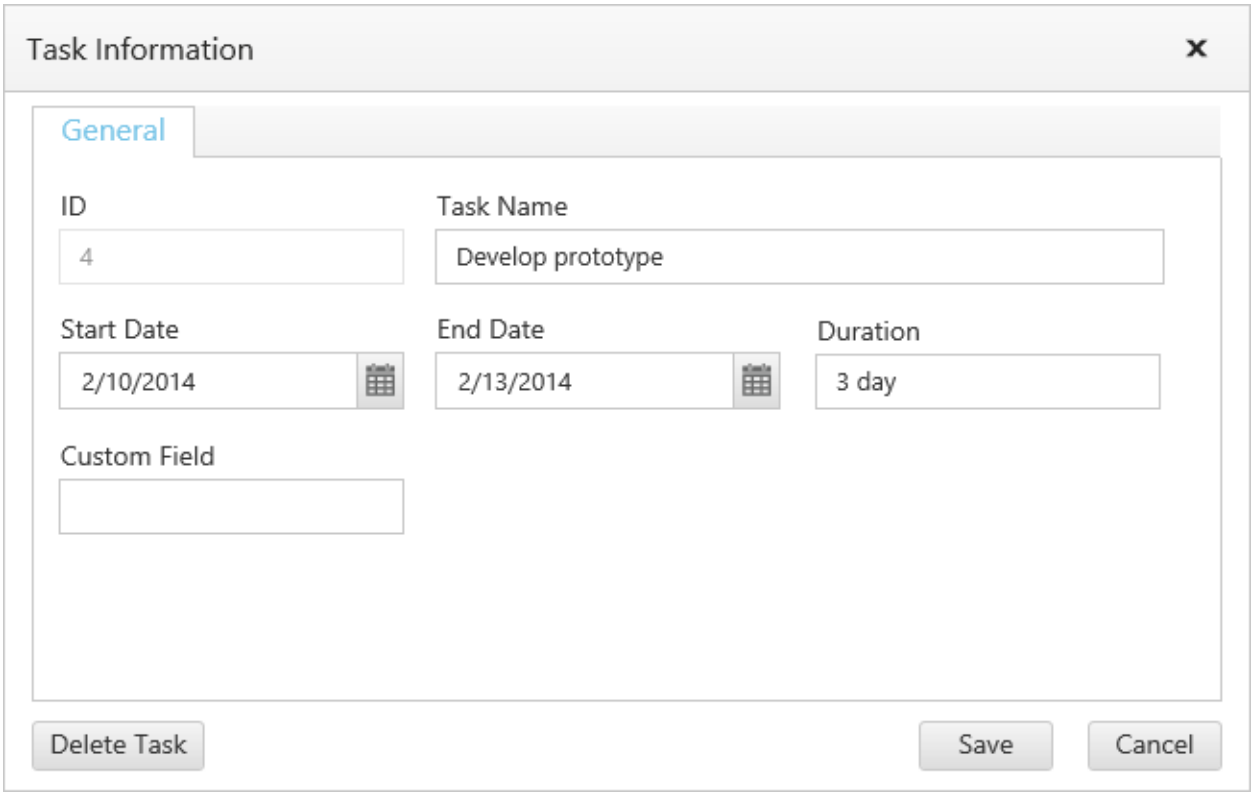
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("normal");
})
.EditDialogFields(editDialog =>
{
editDialog.Field("TaskID").EditType("stringedit").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
editDialog.Field("TaskName").EditType("stringedit").Add();
editDialog.Field("StartDate").EditType("datepicker").Add();
editDialog.Field("EndDate").EditType("datepicker").Add();
editDialog.Field("Duration").EditType("stringedit").Add();
editDialog.Field("CustomField").EditType("stringedit").DisplayInGeneralTab(t
rue).Add();
})
.ClientSideEvents(cs => cs.Load("load"))
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
function load(args) {
var column = { field: "CustomField", mappingName: "CustomField", headerText:
"Custom Field" };
this.getColumns().push(column);
}
</script>
```

The following screenshot show the output of above code example.



Note: Similarly we can include custom fields in add dialog's General tab by setting `DisplayInGeneralTab` as `true` in `AddDialogFields` collection.

Note: Default columns predecessors, resources and notes are displayed in separate tabs, we can't include this in General tab of add/edit dialog.

Copyright © 2001 - 2020 Syncfusion Inc.

Open add/edit dialog dynamically

Gantt add and edit dialogs can be opened dynamically by using openAddDialog and openEditDialog methods. The following code example shows how to open add and dialog on separate button click actions.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit=>{
edit.AllowEditing(true);
edit.EditMode("normal");
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#openAddDialog").click(function(){
$("#Gantt").ejGantt("instance").openAddDialog();
});
$("#openEditDialog").click(function(){
$("#Gantt").ejGantt("instance").openEditDialog();
});
</script>
```

Note: We should select any one of the row in Gantt to open the edit dialog.

Taskbar Editing

Update the task details by interactions such as resizing and dragging the taskbar. Taskbar editing can be enabled by setting AllowGanttChartEditing as true. The following code example shows you how to enable taskbar resizing in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowGanttChartEditing(true)
)
@(Html.EJ().ScriptManager())
```

You can also enable or disable the progress bar resizing by using the EnableProgressBarResizing. The following code example shows how to disable this property.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EnableProgressBarResizing(false)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Prevent taskbar editing for particular task

On taskbar edit action TaskbarEditing and TaskbarEdited client side event will be triggered. We can prevent the taskbar from editing by using TaskbarEditing event and we can revert taskbar edit action by using TaskbarEdited event. The following code example shows how to achieve this.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowGanttChartEditing(true)
.ClientSideEvents(cs =>
cs.TaskbarEditing("taskbarEditing").TaskbarEdited("taskbarEdited"))
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
function taskbarEditing(args) {
if (args.rowData.taskId == 4) // We can't edit Task Id 4
args.cancel = true;
}
function taskbarEdited(args) {
if (args.data.taskId == 5) // We can edit task id 5 task but changes
reverted on mouse up action
args.cancel = true;
}
</script>
```

Dependency Editing

In Gantt, we can add, edit, update the task dependencies by mouse interactions, edit dialog and public methods. The code example shows how to enable dependency editing in Gantt.

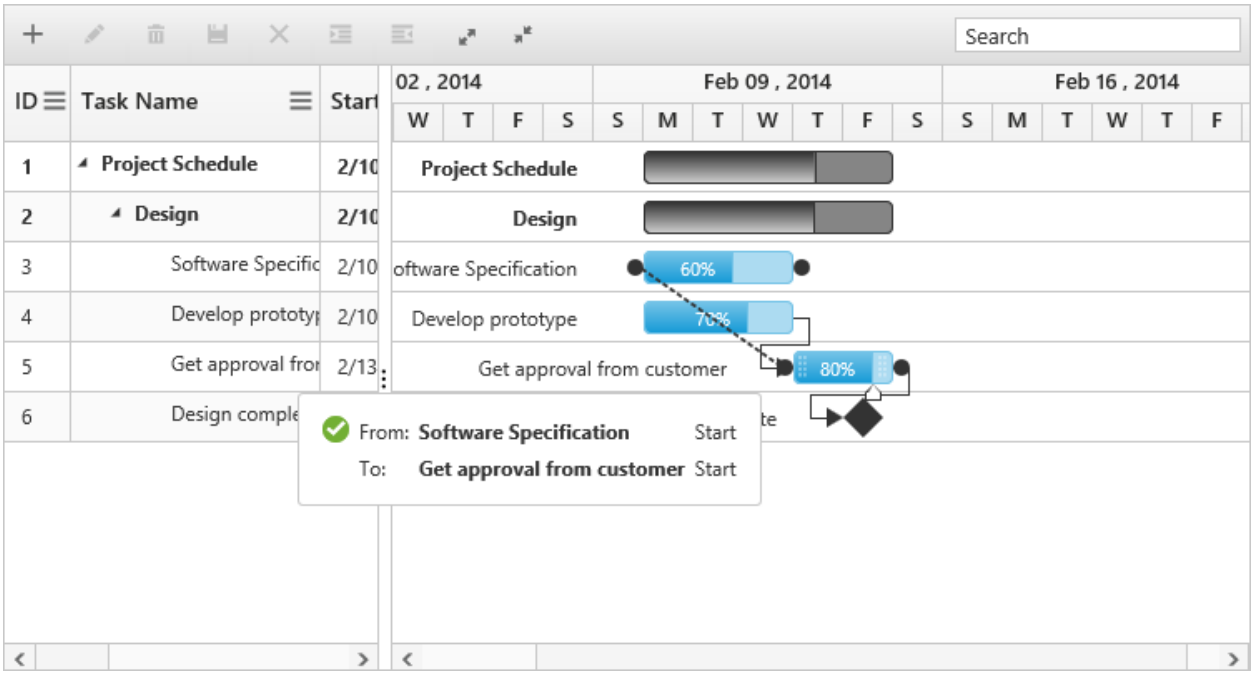
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowGanttChartEditing(true)
.PredecessorMapping("Predecessors")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Add Dependency

Task dependency can be added by mouse interactions by connecting connector points in predecessor and successor tasks. The following screen shot shows the add dependency action.

Copyright © 2001 - 2020 Syncfusion Inc.



Edit Dependency

Task dependency value can be edited by using edit dialog and updateDependency method. Dependency edit dialog can be opened by double clicking on corresponding dependency line.

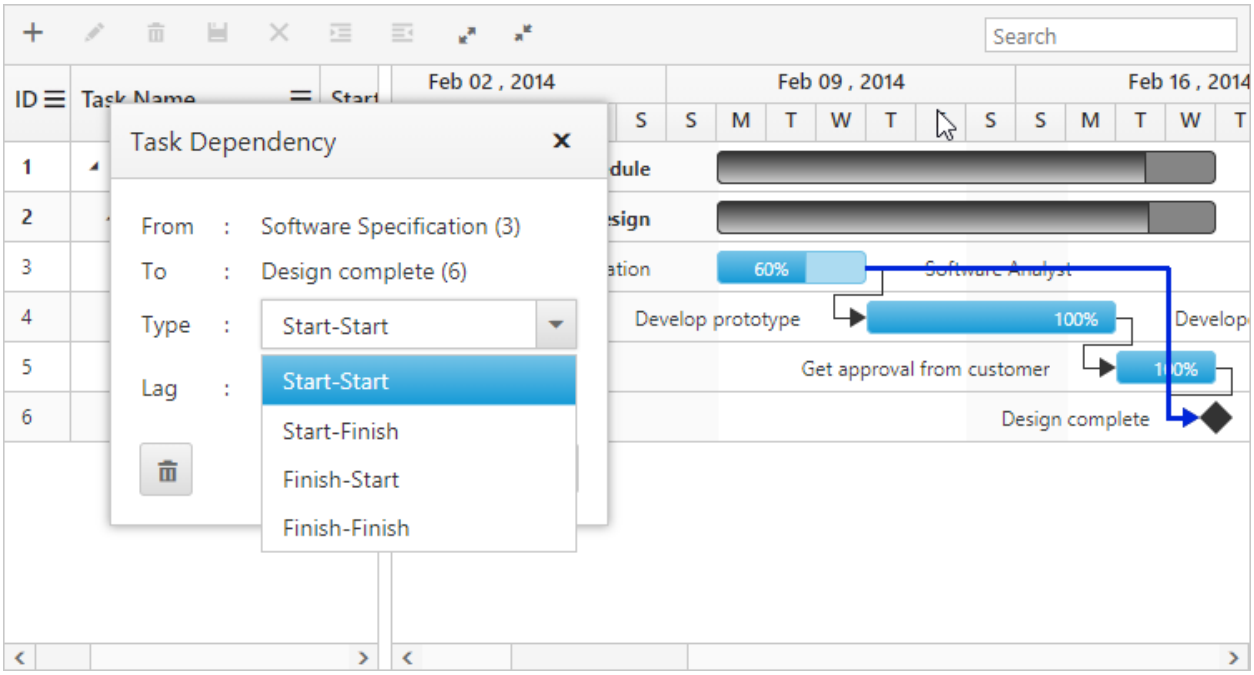
Client side ActionBegin event will be triggered twice with different requestType argument values while opening of dependency dialog. First time, event will be triggered with requestType argument values as beforeDependencyEditDialogOpen, in this event we can get the current editing dependency line information, next ActionBegin client side event will be triggered with requestType argument value as afterDependencyEditDialogOpen, in this event we can get the information about current editing dependency line and editing elements in edit dialog. Using this event we can customize the dependency edit dialog elements.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowGanttChartEditing(true)
.PredecessorMapping("Predecessors")
.ClientSideEvents(cs => cs.ActionBegin("actionBegin"))
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
function actionBegin(args) {
if(args.requestType == "beforeDependencyEditDialogOpen") {
}
else if(args.requestType == "afterDependencyEditDialogOpen") {
}
}
</script>
```

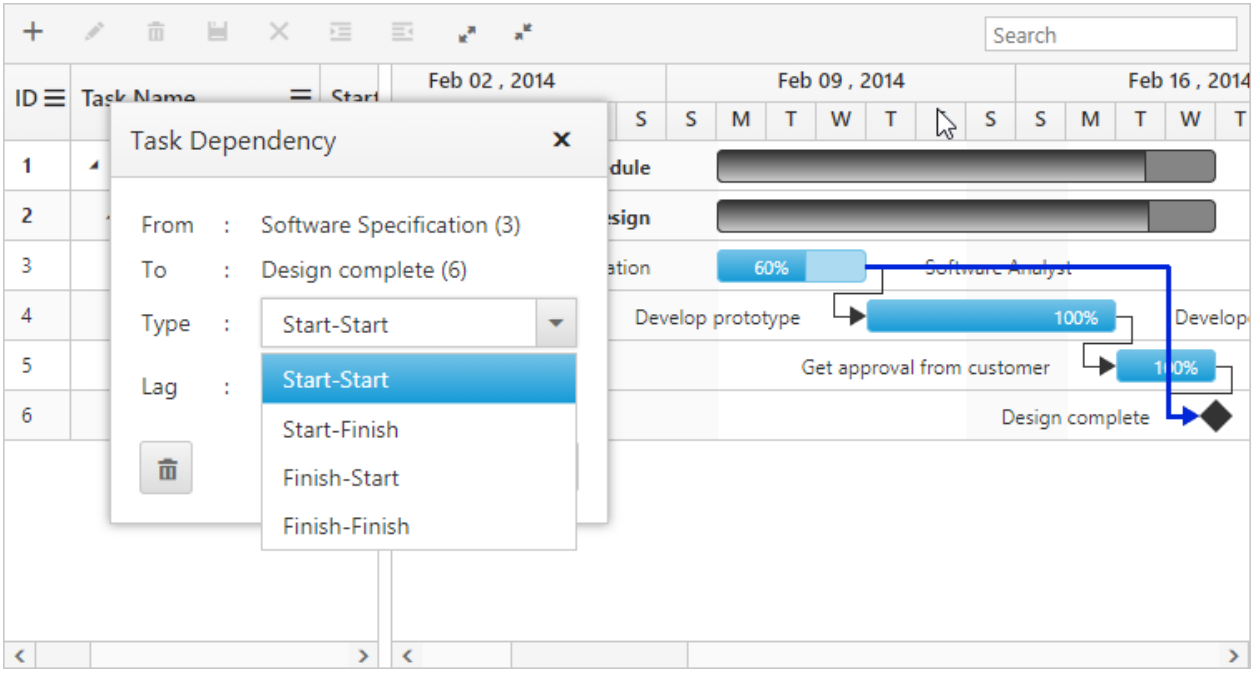
The following screen shot shows the dependency edit dialog.

Copyright © 2001 - 2020 Syncfusion Inc.



Delete Dependency

Task dependency can be deleted by using edit dialog and deleteDependency method. The following screen shot shows the dependency edit dialog with delete option.



Click here to view the online demo sample for editing in Gantt.

Update Gantt record value dynamically

Gantt record's value can be dynamically updated by using updateRecordByTaskId or updateRecordByIndex method. updateRecordByTaskId method used to update the Gantt record by using it's task id value and updateRecordByIndex method was used to update Gantt record value by row

index value. We can invoke this method on any external button click action, the following code example shows how to use this methods.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#updateRecordByTaskId").click(function () {
var ganttObj = $("#Gantt").ejGantt("instance"),
data = {
taskID: 4,
taskName: "Updated by task id",
startDate: new Date("02/10/2014"),
endDate: new Date("02/14/2014"),
duration: 5,
progress: "80",
resourceId: [2]
};
ganttObj.updateRecordByTaskId(data);
});
$("#updateRecordByRowIndex").click(function () {
var ganttObj = $("#Gantt").ejGantt("instance"),
data = {
taskID: 5,
taskName: "Updated by index value",
startDate: new Date("02/10/2014"),
endDate: new Date("02/14/2014"),
duration: 3, progress: "100",
predecessor: "4SS",
resourceId: [2]
};
ganttObj.updateRecordByIndex(4, data);
});
</script>
```

Note: Using these methods we can't update the task id value.

Delete confirmation message

Delete confirmation message is used to get the confirmation from the user before deleting the record. This confirmation message can be enabled by setting `EditSettings.ShowDeleteConfirmDialog` property as `true`.

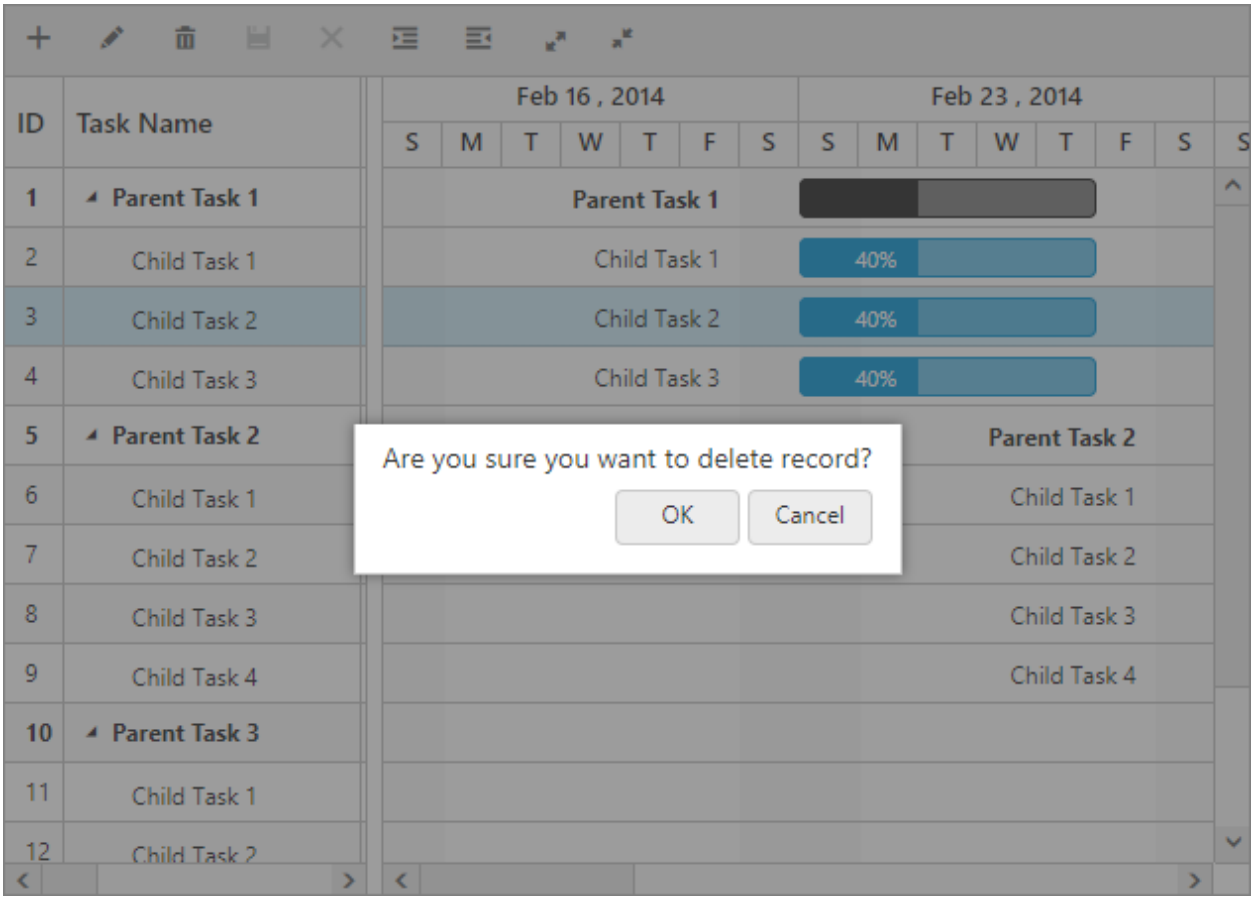
The following code snippet explains how to enable delete confirmation message in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit =>
{
edit.AllowDeleting(true);
edit.ShowDeleteConfirmDialog(true);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
@(Html.EJ().ScriptManager())
```



The above screen shot shows the appearance of delete confirmation message in Gantt.

Task Indent

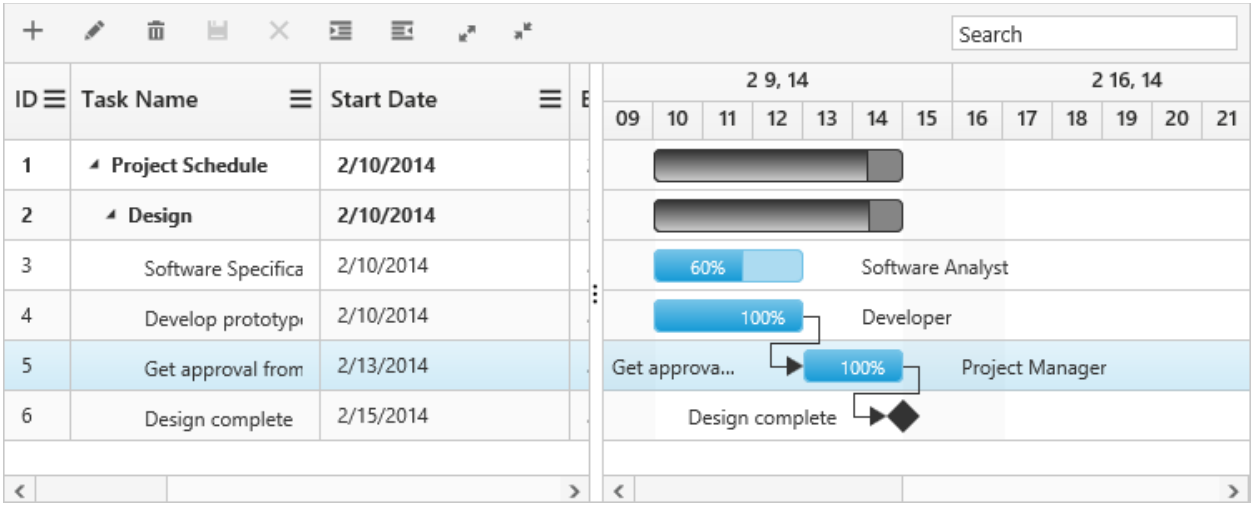
Task indent option in Gantt was enabled by setting EditSettings.AllowIndent as true. Tasks can be indented by clicking on indent toolbar item or by using indentItem method. We can invoke this method dynamically on any action like external button click. The below code example shows how to enable indent option in Gantt.

ASPX-CS

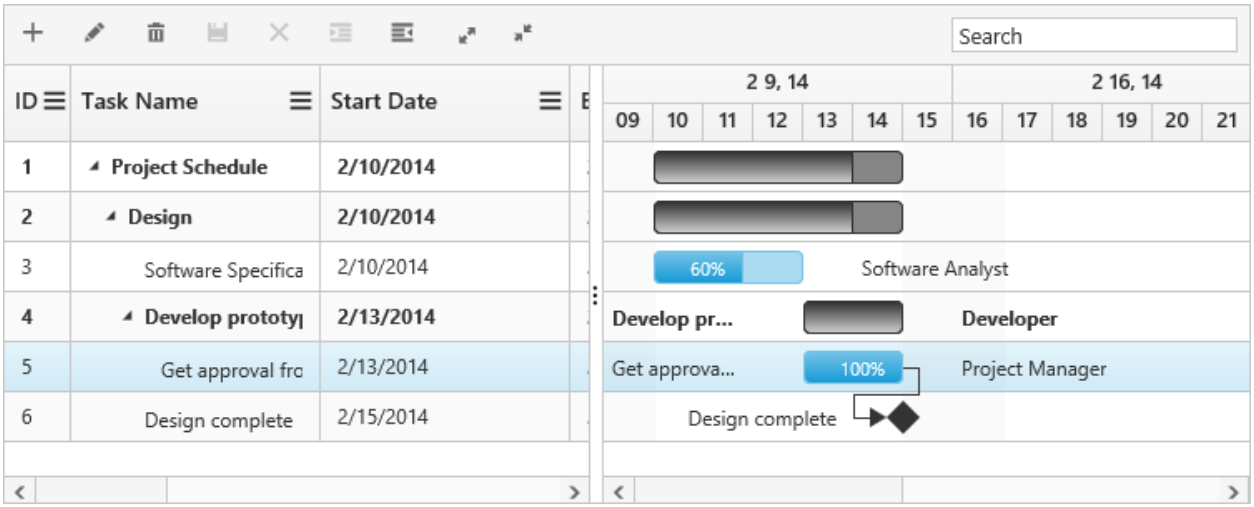
```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit =>
{
edit.AllowIndent(true);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true);
toolbar.ToolbarItems(new List<GanttToolBarItems>()
{
//...
```

```
GanttToolBarItems.Indent,
});
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#indentTask").click(function () {
var ganttObj = $("#Gantt").ejGantt("instance");
if (ganttObj.selectedRowIndex() != -1)
ganttObj.indentItem();
});
</script>
```

The following screenshots shows the output of above code example.



Before Indent



After Indent

Note: We should select any one of the row in Gantt to perform task indent action.

Copyright © 2001 - 2020 Syncfusion Inc.

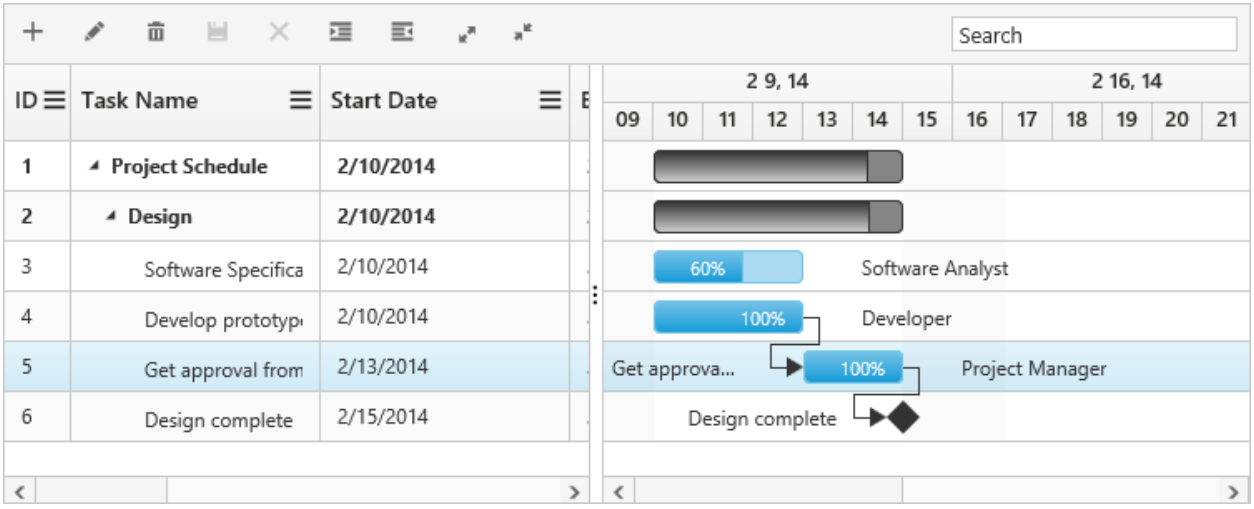
Task Outdent

Task outdent option in Gantt was enabled by setting EditSettings.AllowIndent as true. Tasks can be outdent by clicking on outdent toolbar item or by using outdentItem method. We can invoke this method dynamically on any action like external button click. The below code example shows how to enable outdent option in Gantt.

ASPX-CS

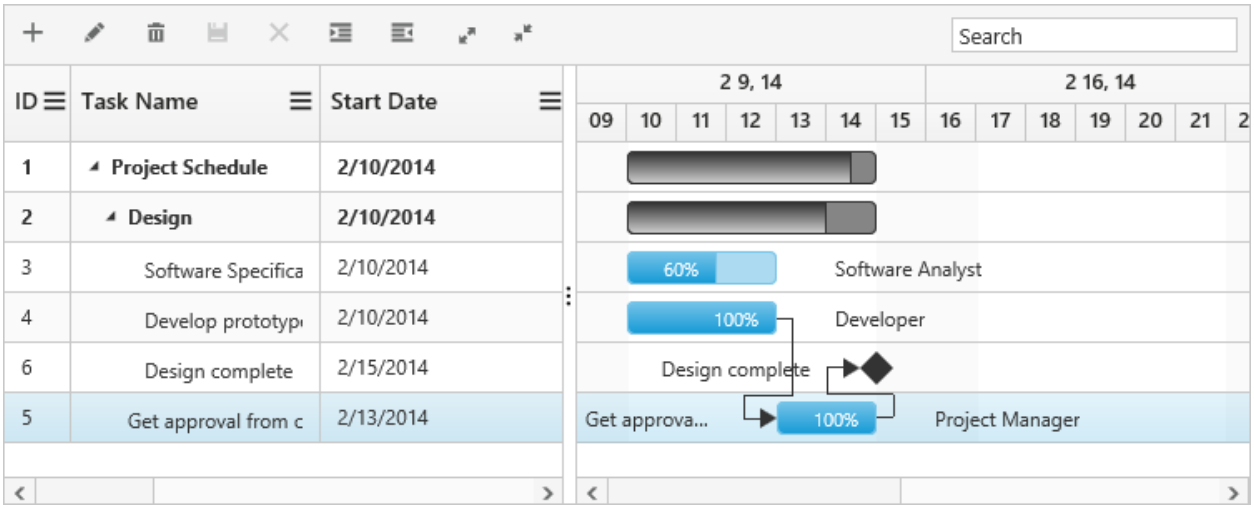
```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit =>
{
edit.AllowIndent(true);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true);
toolbar.ToolbarItems(new List<GanttToolBarItems>()
{
//...
GanttToolBarItems.Outdent,
});
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#outdentTask").click(function () {
var ganttObj = $("#Gantt").ejGantt("instance");
if (ganttObj.selectedRowIndex() != -1)
ganttObj.outdentItem();
});
</script>
```

The following screenshots shows the output of above code example.



Before Outdent

Copyright © 2001 - 2020 Syncfusion Inc.



After Outdent

Note: We should select any one of the row in Gantt to perform task outdent action.

Task Delete

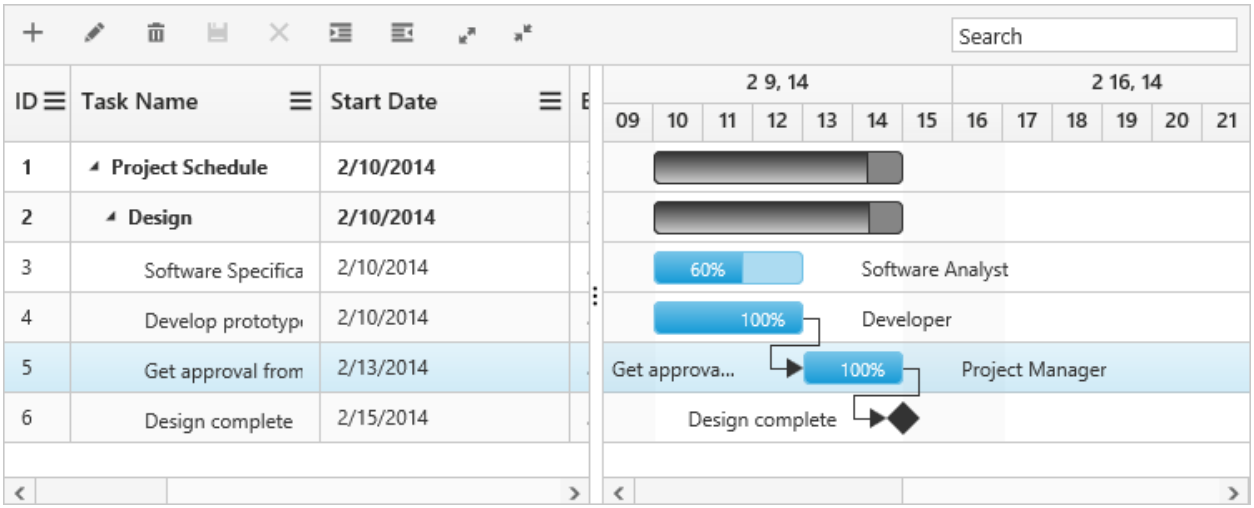
Task delete option in Gantt was enabled by setting `EditSettings.AllowDeleting` as `true`. Tasks can be delete by clicking on delete toolbar item or by using deleteItem method. We can invoke this method dynamically on any action like external button click. The below code example shows how to enable delete option in Gantt.

ASPX-CS

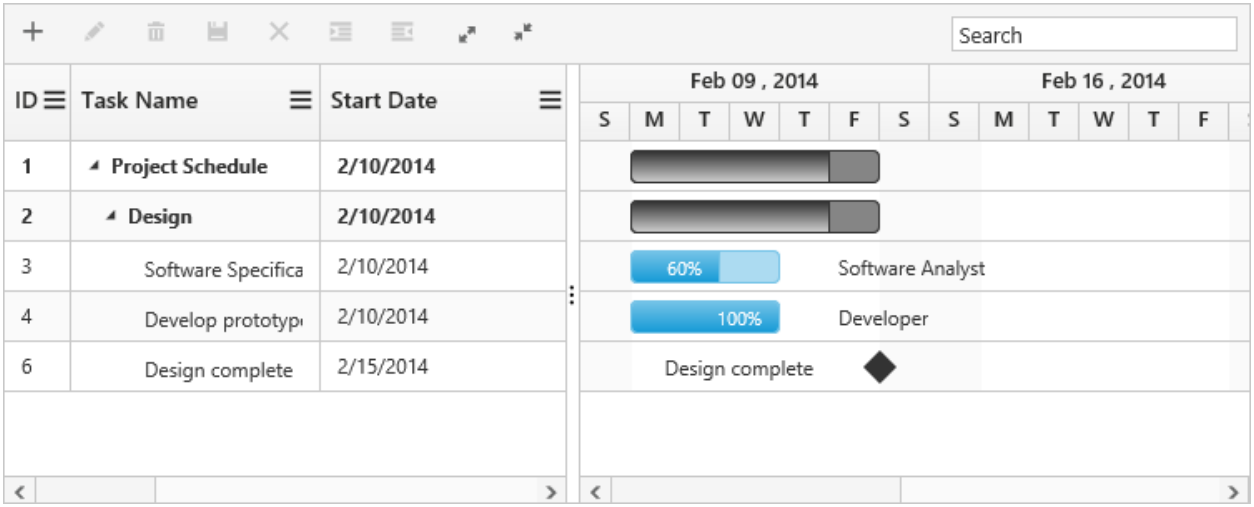
```
@(Html.EJ().Gantt("Gantt")
//...
.EditSettings(edit =>
{
edit.AllowDeleting(true);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true);
toolbar.ToolbarItems(new List<GanttToolBarItems>()
{
//...
GanttToolBarItems.Delete,
});
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script>
$("#deleteTask").click(function () {
var ganttObj = $("#Gantt").ejGantt("instance");
if (ganttObj.selectedRowIndex() != -1)
ganttObj.deleteItem();
});
</script>
```

The following screenshots shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Before Delete



After Delete

Note: We should select any one of the row in Gantt to perform task delete action.

Read-only Gantt

In Gantt, all editing options can be disabled by setting ReadOnly property as true. The following code example shows how to make Gantt control as read-only.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ReadOnly(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Timescale customization

Gantt contains built-in support to switch over to various schedule mode. You can achieve this by defining a schedule header type for the Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.

Schedule Header Types

Gantt contains the following built-in schedule header types:

- Hour – Minute
- Day – Hour
- Week – Day
- Month – Week
- Year – Month

Time scale mode in Gantt can be defined by using `ScheduleHeaderType` property and

the following code example illustrates you on how to change the schedule mode.

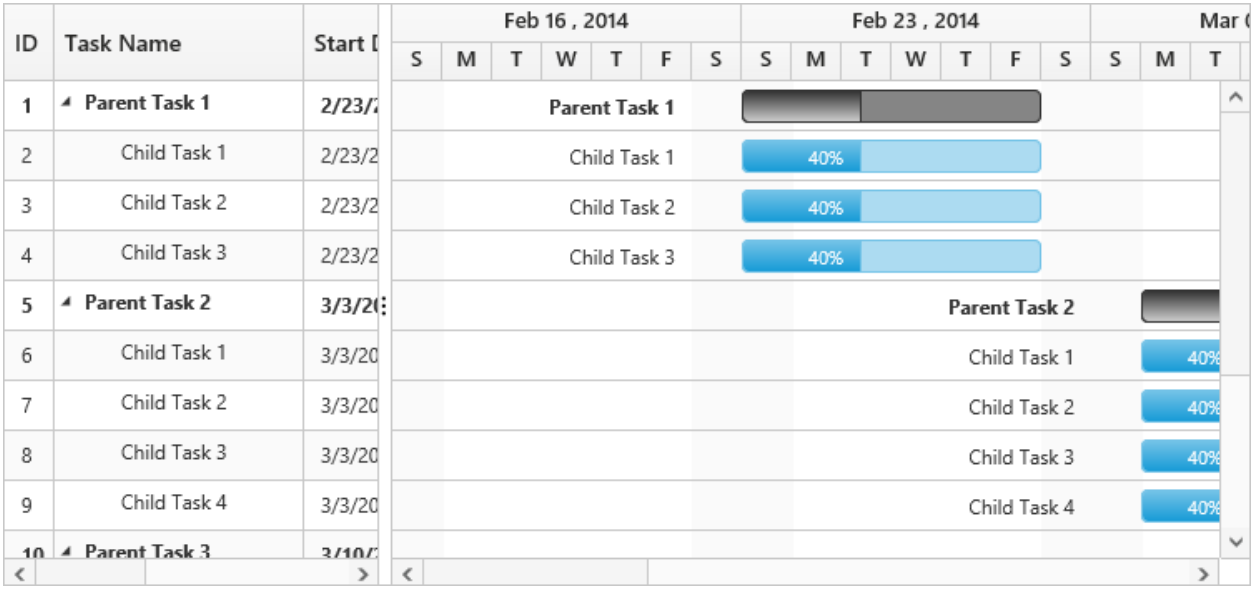
Week Schedule Mode

In the Week schedule mode, the upper part of the schedule header displays the weeks whereas the bottom half of the header displays the days. Refer the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ScheduleHeaderSettings(sh=>sh.ScheduleHeaderType(GanttScheduleHeaderType.We
ek)
.WeekHeaderFormat("MMM dd , yyyy ")
.DayHeaderFormat("ddd "))
)
@(Html.EJ().ScriptManager())
```

The following screenshot illustrates the Week Schedule in Gantt control.



Week Schedule in Gantt control

Month Schedule Mode

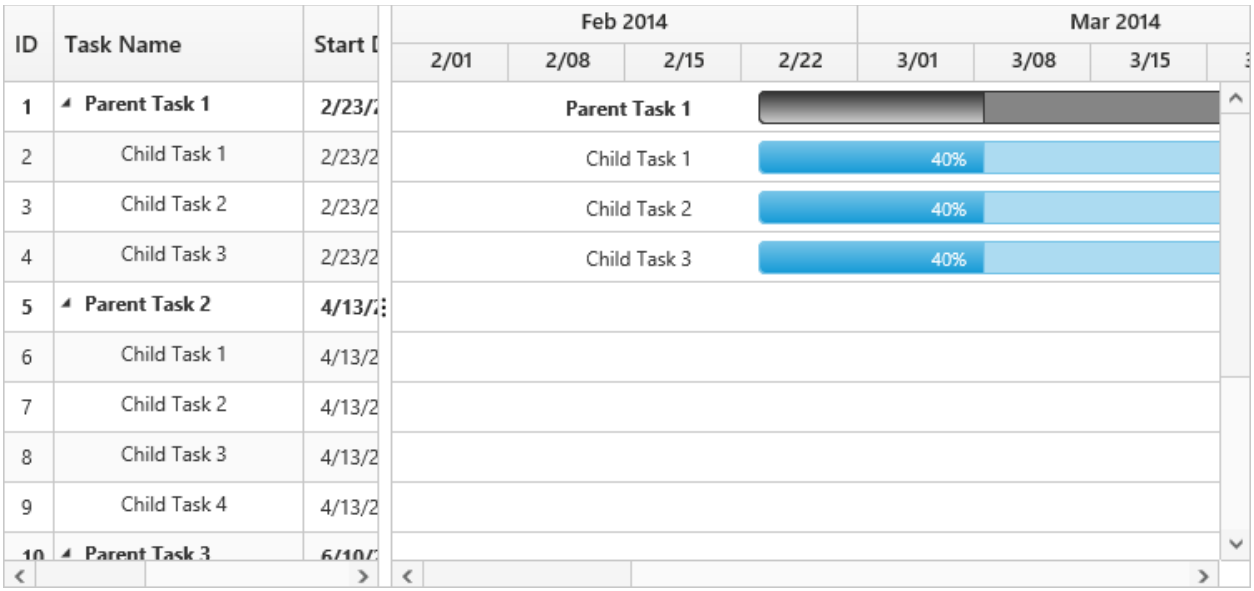
In the Week schedule mode, the upper part of the schedule header displays the Months whereas the bottom header of the schedule displays its corresponding Weeks. Refer the following code example.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ScheduleHeaderSettings(sh=>sh.ScheduleHeaderType(GanttScheduleHeaderType.Mo
nth)
.MonthHeaderFormat("MMM yyyy")
.WeekHeaderFormat("M/dd"))
)
@(Html.EJ().ScriptManager())
```

The following screenshot illustrates the Month Schedule in Gantt control.



Month Schedule in Gantt control

Year Schedule Mode

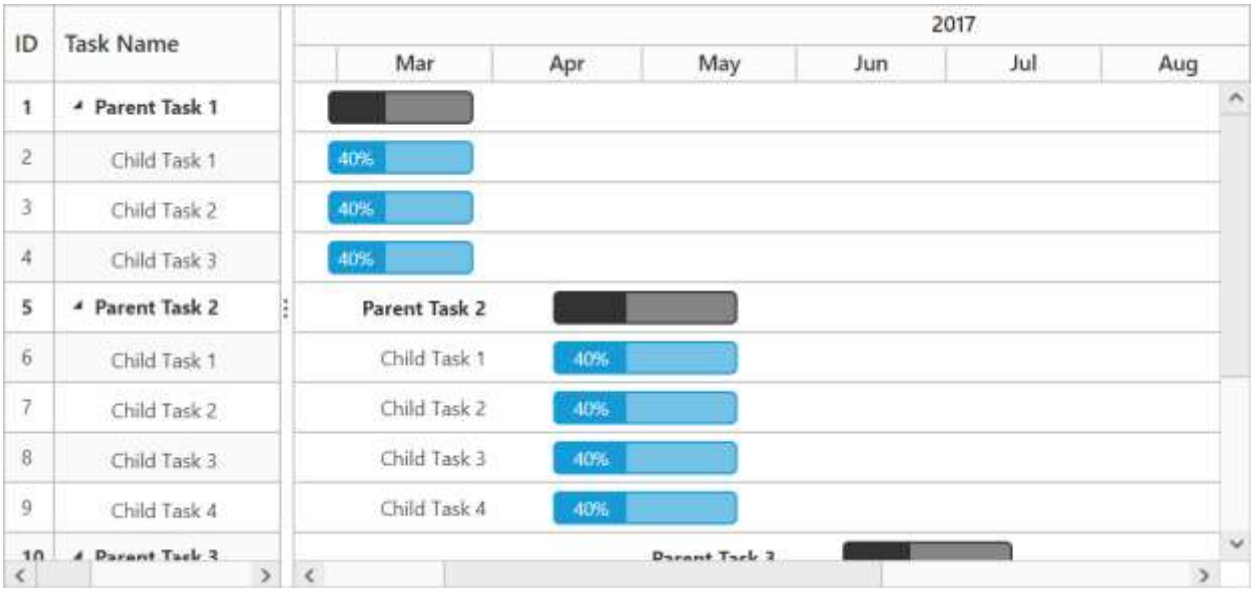
In the Week schedule mode, the upper schedule header displays the Years whereas the bottom header displays its corresponding Months. Refer the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ScheduleHeaderSettings(sh=>sh.ScheduleHeaderType(GanttScheduleHeaderType.Ye
ar)
.YearHeaderFormat("yyyy")
.MonthHeaderFormat("MMM"))
)
@(Html.EJ().ScriptManager())
```

The following screen shot shows the Year Schedule in Gantt control.

Copyright © 2001 - 2020 Syncfusion Inc.



Year Schedule in Gantt control

Day Schedule Mode

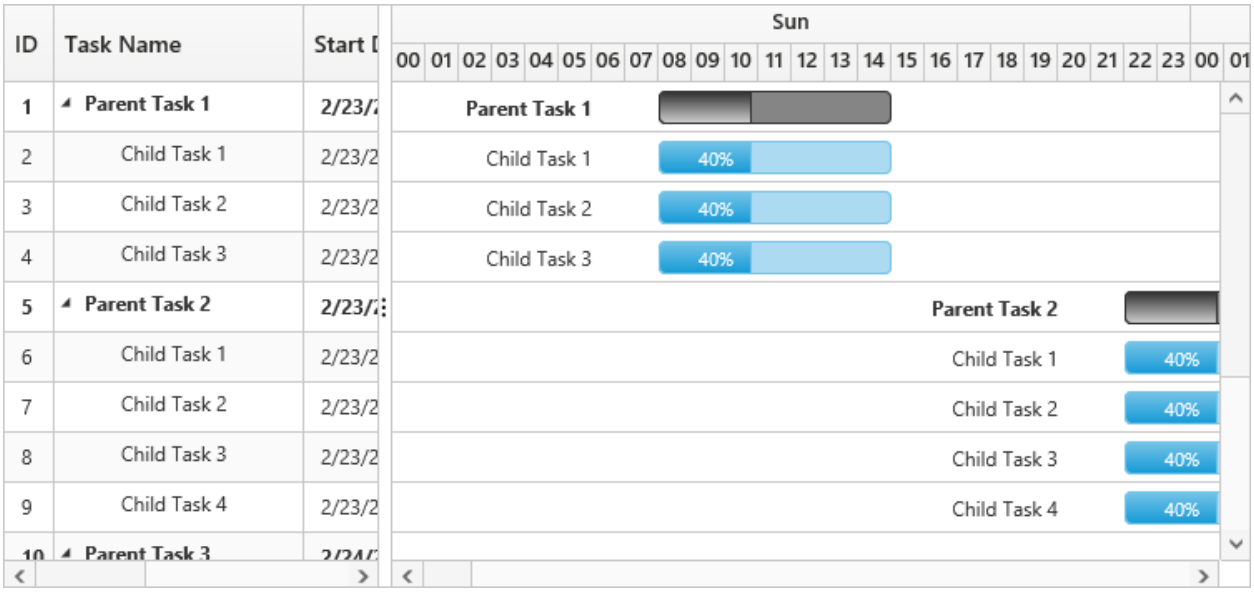
In the Week schedule mode, the upper part of the header displays the Days whereas the bottom schedule header displays its corresponding Hours. Refer the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ScheduleHeaderSettings(sh=>sh.ScheduleHeaderType(GanttScheduleHeaderType.Day)
.DayHeaderFormat("ddd ")
.HourHeaderFormat("HH"))
)
@(Html.EJ().ScriptManager())
```

The following screenshot illustrates the Day Schedule in Gantt control.

Copyright © 2001 - 2020 Syncfusion Inc.



Day Schedule in Gantt control

Hour Schedule Mode

An Hour-Minute Schedule Mode tracks the tasks in minutes scale. In this mode, upper schedule header displays hour scale and the lower schedule header displays its corresponding Minutes. The minute split-up in the lower schedule header can be defined by using the MinutesPerIntervalEnum property. The enumeration values of the MinutesPerInterval are,

- Auto
- OneMinute
- FiveMinutes
- FifteenMinutes
- ThirtyMinutes

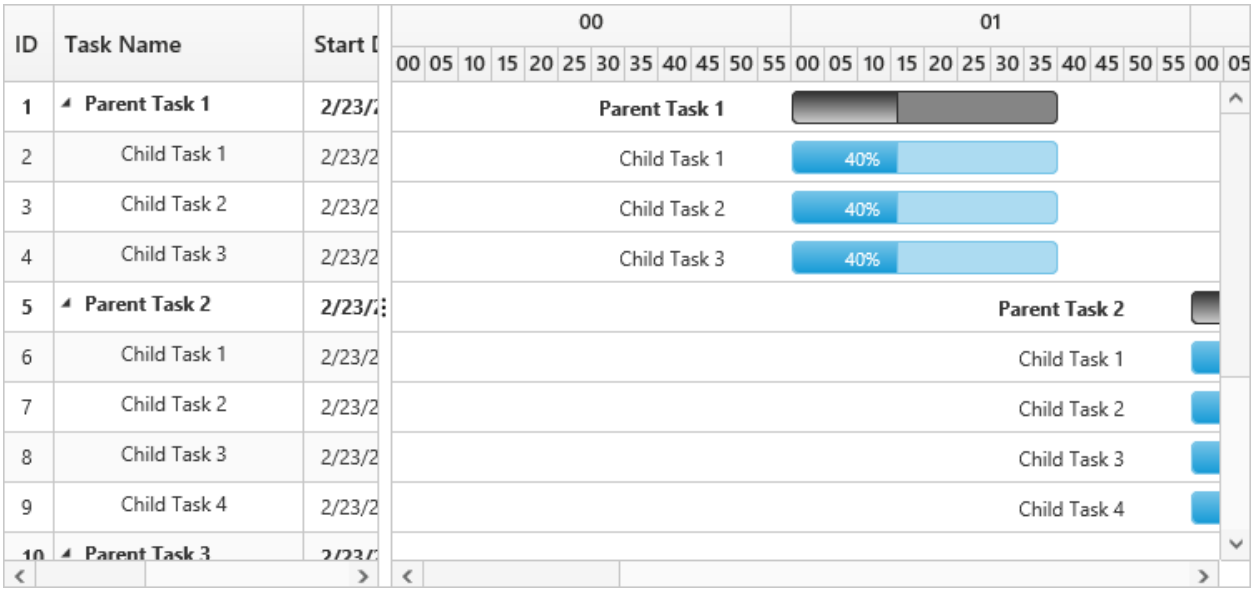
The value, Auto, automatically calculates the interval depending upon the ScheduleStartDate and ScheduleEndDate, whereas the other enumeration values splits up accordingly.

The Hour Schedule Mode supports both the Minute and Hour duration units.

ASPX-CS

```
@(Html.EJ().Gantt("ganttContainer")
.DateFormat("M/d/yyyy hh:mm:ss tt")
.DurationUnit(GanttDurationUnit.Minute)
.ScheduleHeaderSettings(sh =>
{
sh.ScheduleHeaderType(GanttScheduleHeaderType.Hour);
sh.MinutesPerInterval(GanttMinutesPerInterval.FiveMinutes);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Hour-Minute schedule mode in Gantt control

Week start day customization

In Gantt, we can customize week start day by using WeekStartDay property.

By default the weekStartDay will be assigned with 0 which specifies the start day of the week.

In week schedule mode, week starts with Sunday by default. But we can customize the week start day by using below code example

ASPX-CS

```
@(Html.EJ().Gantt("ganttContainer")
.ScheduleHeaderSettings(sh =>
{
sh.ScheduleHeaderType(GanttScheduleHeaderType.Week);
sh.WeekStartDay(3);
})
)
@(Html.EJ().ScriptManager())
```

Rounding off timescale (schedule) start date

You can able to round off the schedule start date in a project by using the TimescaleStartDateMode property. It is possible to set the following values to the property,

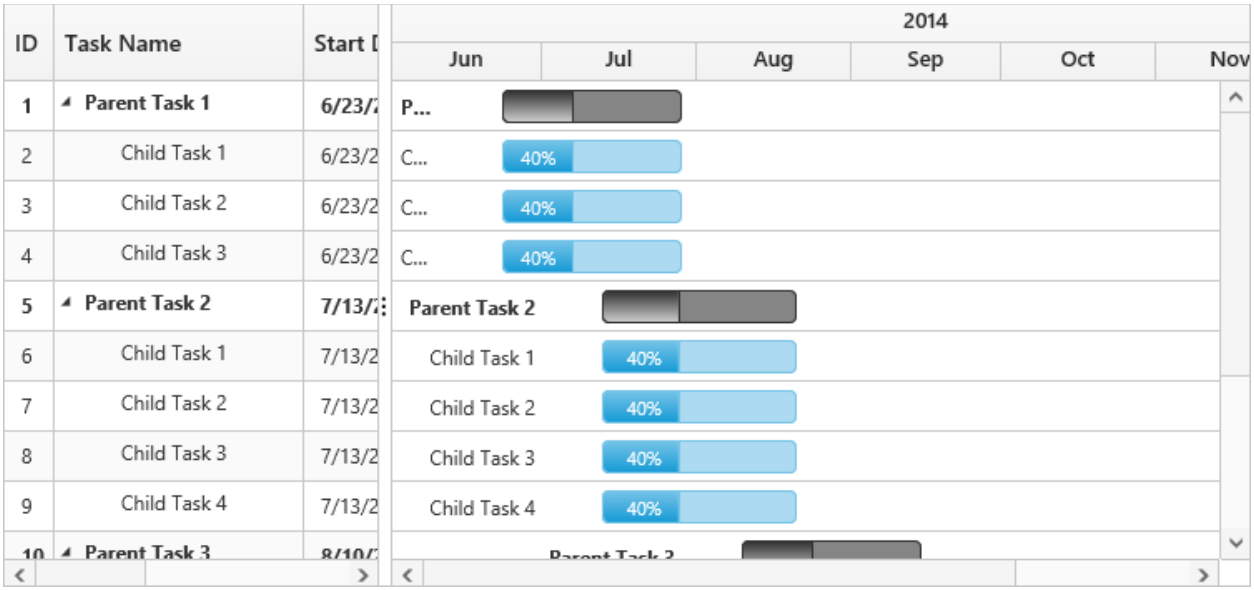
- Auto
- Month
- Week
- Year

The value Auto, automatically calculates the schedule header depending on the datasource values, whereas the other enumeration values rounds off the schedule header accordingly. For Instance, in year schedule if you set TimescaleStartDateMode as Month then the schedule header will start from the immediate month of the schedule instead of starting from beginning of the year.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Gantt("ganttContainer")
.ScheduleHeaderSettings(sh =>
{
sh.ScheduleHeaderType(GanttScheduleHeaderType.Year);
sh.TimescaleStartDateMode(GanttTimescaleRoundMode.Month);
})
)
@(Html.EJ().ScriptManager())
```



Click here to view the timescale modes in Gantt.

Customize automatic timescale update action

In Gantt, schedule timeline will be automatically updated when the tasks are edited beyond the schedule start date and end date range. This can be enabled/disabled by using UpdateTimescaleView property.

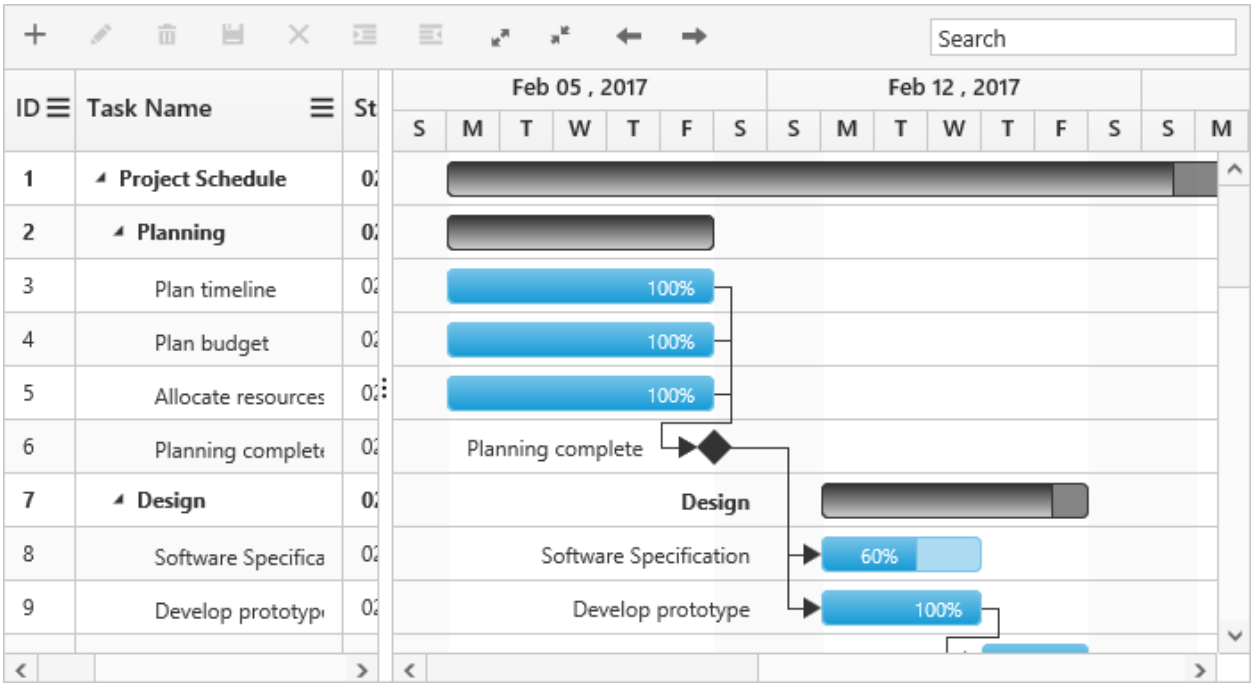
The following code snippets shows how to prevent the automatic timescale update in Gantt.

ASPX-CS

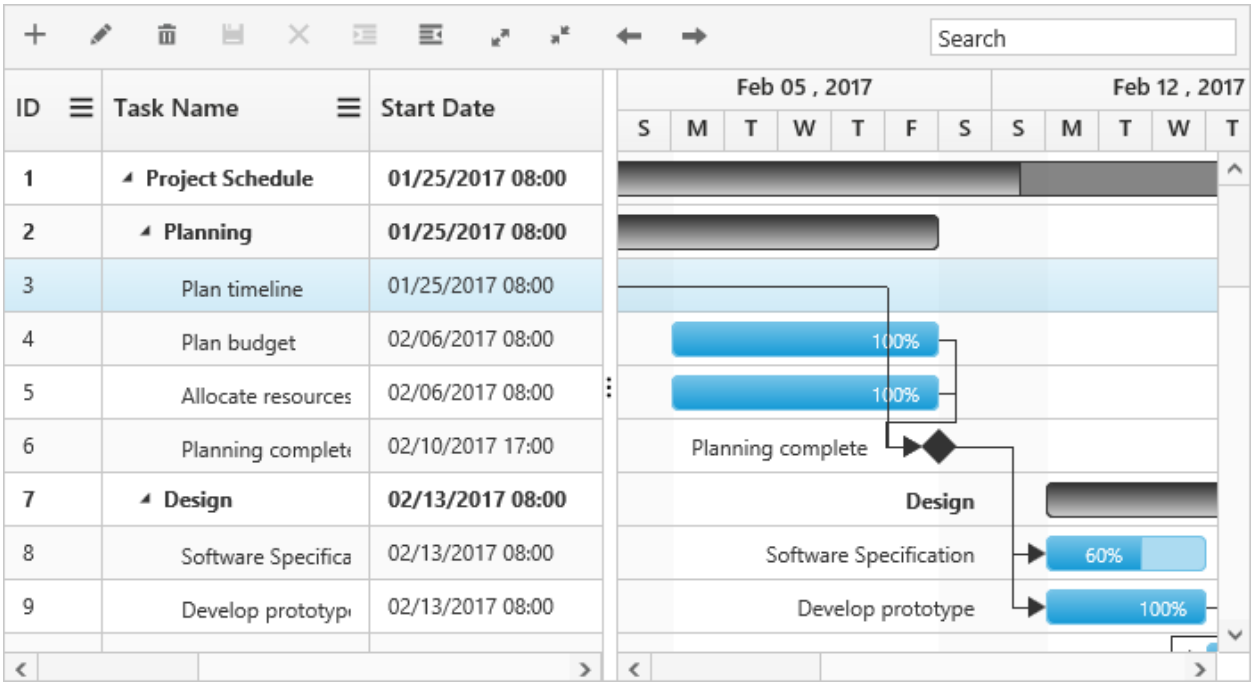
```
@(Html.EJ().Gantt("ganttContainer")
.ScheduleHeaderSettings(sh =>
{
sh.ScheduleHeaderType(GanttScheduleHeaderType.Week);
sh.UpdateTimescaleView(false);
})
)
```

The following screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

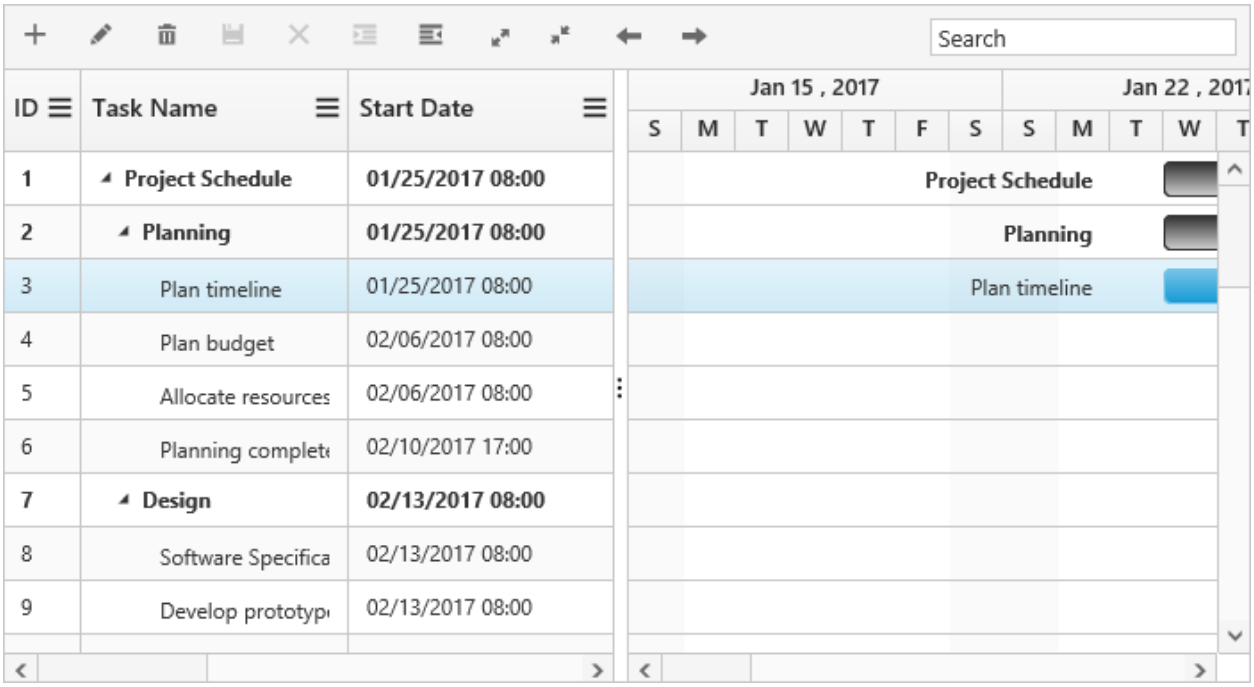


At Initial load



`updateTimescaleView` property as `false`

Copyright © 2001 - 2020 Syncfusion Inc.



updateTimescaleView property as true

Critical Path

The critical path in a project is indicated by a single task or a series of tasks in a project, critical path describes the calculated start date or end date of a project. If a task in a critical path is delayed, then the entire project will be delayed.

The critical path can be enabled in Gantt by using built-in toolbar button or showCriticalPath method.

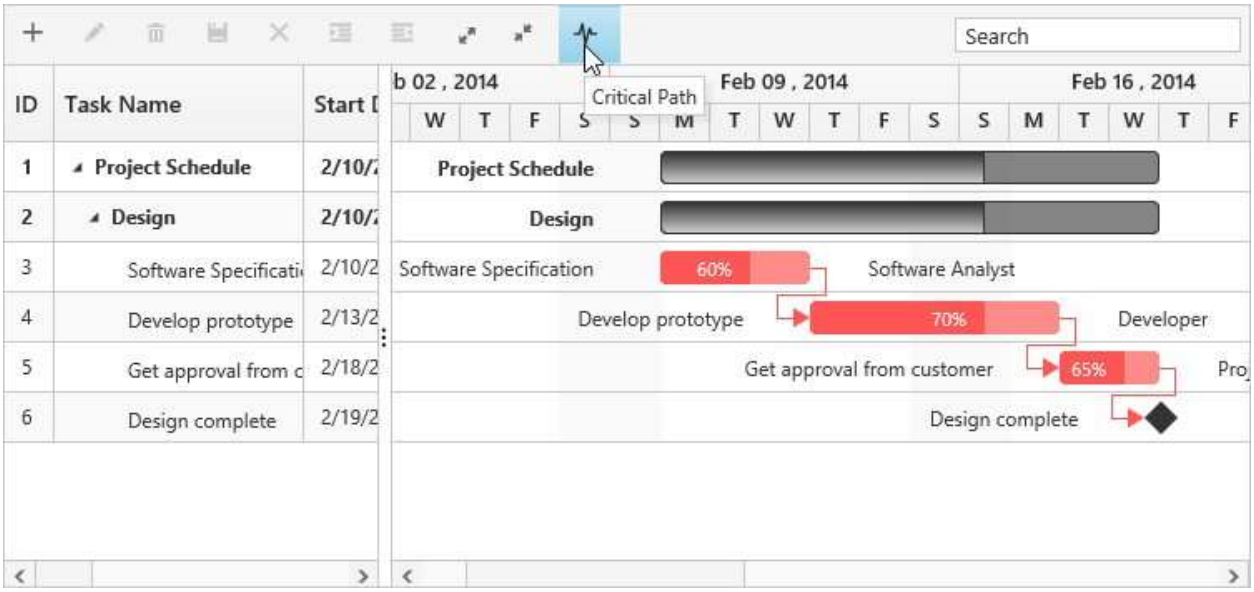
Using Toolbar Icon

You can enable/disable the critical path in Gantt by using toolbar button, and this button can be enabled in the Gantt toolbar by using the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true);
toolbar.ToolbarItems(new List<GanttToolBarItems>()
{
//..
GanttToolBarItems.CriticalPath
});
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample for critical path in Gantt.

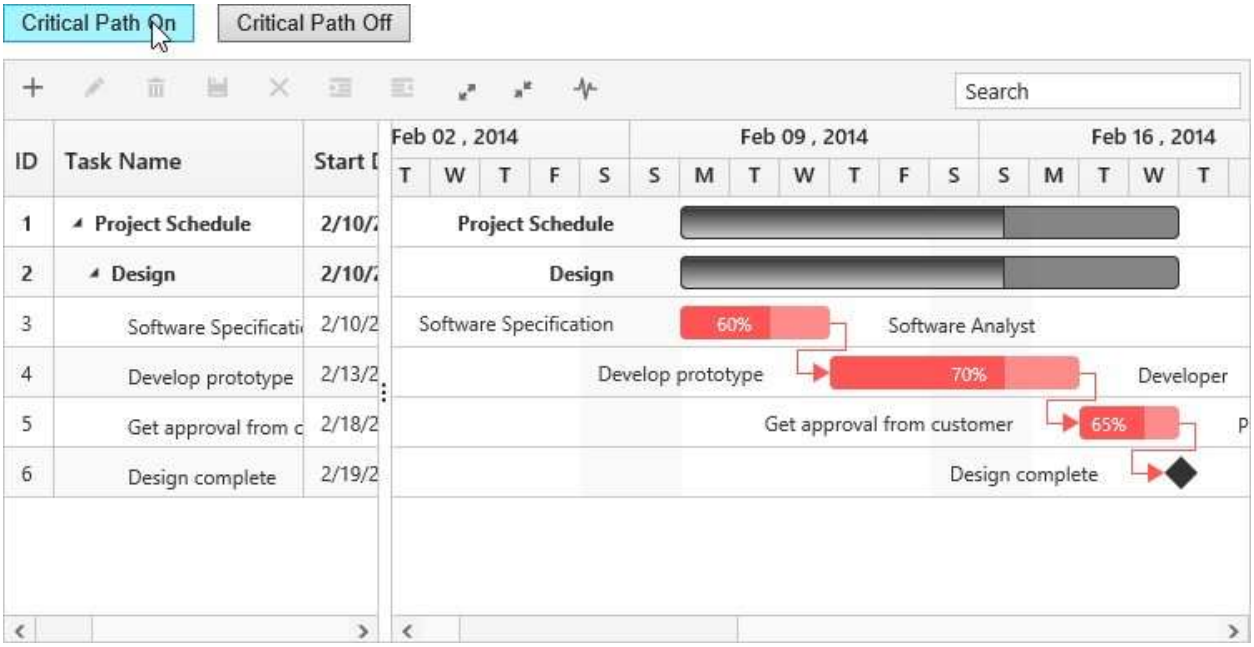
Using Method

You can enable the critical path by using showCriticalPath method with parameter as true. And critical path can be disabled using the same method with parameter as false.

ASPX-CS

```
<script>
$("#buttonOn").click(function (args) {
ganttObj = $("#Gantt").data("ejGantt");
ganttObj.showCriticalPath(true);
})
$("#buttonOff").click(function (args) {
ganttObj = $("#Gantt").data("ejGantt");
ganttObj.showCriticalPath(false);
})
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing critical path background

You can customize the critical task background by setting custom color codes to the critical task element's class names,

Taskbar–.e-ganttchart .e-criticaltaskbar

Progressbar–.e-ganttchart .e-criticalprogressbar

Connector line–.e-ganttchart .e-criticalconnectorline

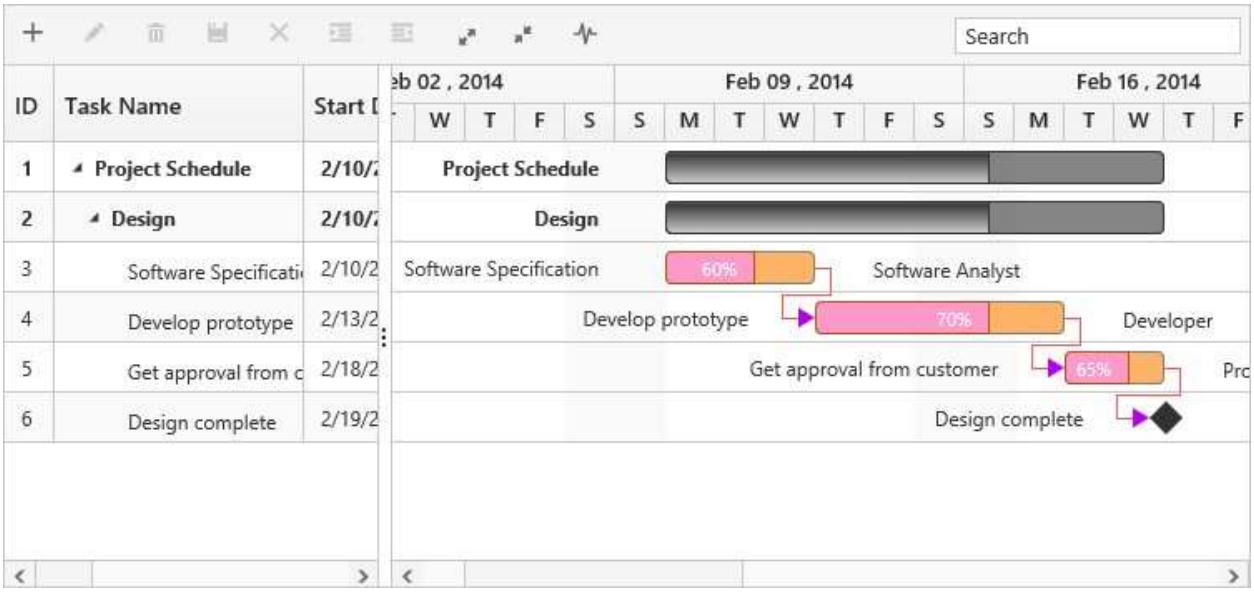
Connector line left arrow– .e-ganttchart .e-criticalconnectorlineleftarrow

Connector line right arrow– .e-ganttchart .e-criticalconnectorlinerightarrow

ASPX-CS

```
<style>
.e-ganttchart .e-criticaltaskbar {
background-color : #ffb366!important;
border-color : gray!important
}
.e-ganttchart .e-criticalprogressbar {
background-color : #ff99cc!important;
border-color : #b35900!important
}
.e-ganttchart .e-criticalconnectorline {
background-color : #b800e6!important;
}
.e-ganttchart .e-criticalconnectorlineleftarrow {
border-right-color : #b800e6!important;
}
.e-ganttchart .e-criticalconnectorlinerightarrow{
border-left-color : #b800e6!important;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Gantt Columns

Gantt column displays the information from a bounded data source and it will be editable to update the task details through TreeGrid.

Column Edit Types

Gantt supports the following types of column editors,

- String
- Date
- Datetime
- Numeric
- Maskedit
- Currency
- Dropdown

You can change the width of the column in TreeGrid to show the entire text of the column by resizing the column. The following code example shows you how to enable the column resize feature at Gantt initialize.

Format Column

It is possible to format a column using Load event. The following code examples show how to format the progress column with percentage value.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ClientSideEvents(eve=>
{
eve.Load("load");
})
)
@(Html.EJ().ScriptManager())
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function load(args){
var columns = this.getColumns();
columns[5]["format"] = "{0:p0}";
}
</script>
```

Note: For more numeric format strings, please refer to this link.

Note: For more date format strings, please refer to this link.

Column Resizing

You can change the width of the column to show the entire text of the column by resizing the column.

In Gantt column resizing can be enabled by setting `AllowColumnResize` property as `true`.

The following code example shows you how to enable the column resize feature in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowColumnResize(true)
)
@(Html.EJ().ScriptManager())
```

Column Template

Column template is used to customize the column's look and feel, based on requirement.

The following code example shows you how to display a column with resource images.

ASPX-CS

```
<script type="text/x-jsrender" id="columnTemplate">
{{"{{"}}if #data['resourceNames']{{}}}}
<div style="display:inline-block;position:relative;left:10px;top:1px">
<img src="images/Gantt/{{'{{'}}:#data['resourceNames']{{}}}}.png"
height="40px" />
</div>
<div style='display:inline-
block;width:100%;position:relative;left:10px;top:2px'>{{:#data['resourceName
s']{{}}}}</div>
{{"{{"}}/if{{}}}}
</script>
@(Html.EJ().Gantt("Gantt")
//...
.ResourceInfoMapping("ResourceID")
.ResourceNameMapping("ResourceName")
.ResourceIdMapping("ResourceID")
.ClientSideEvents(eve=>
{
eve.Load("load");
})
.Resources(ViewBag.datasource1)
)
@(Html.EJ().ScriptManager())
<script>
function load(args){
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var Gantt = $("#Gantt").ejGantt("instance");
var columns = Gantt.getColumns();
columns[2].visible = columns[3].visible = false;
columns[4].isTemplateColumn = true;
columns[4].templateID = "columnTemplate";
columns[4].width = "172";
}
</script>
```

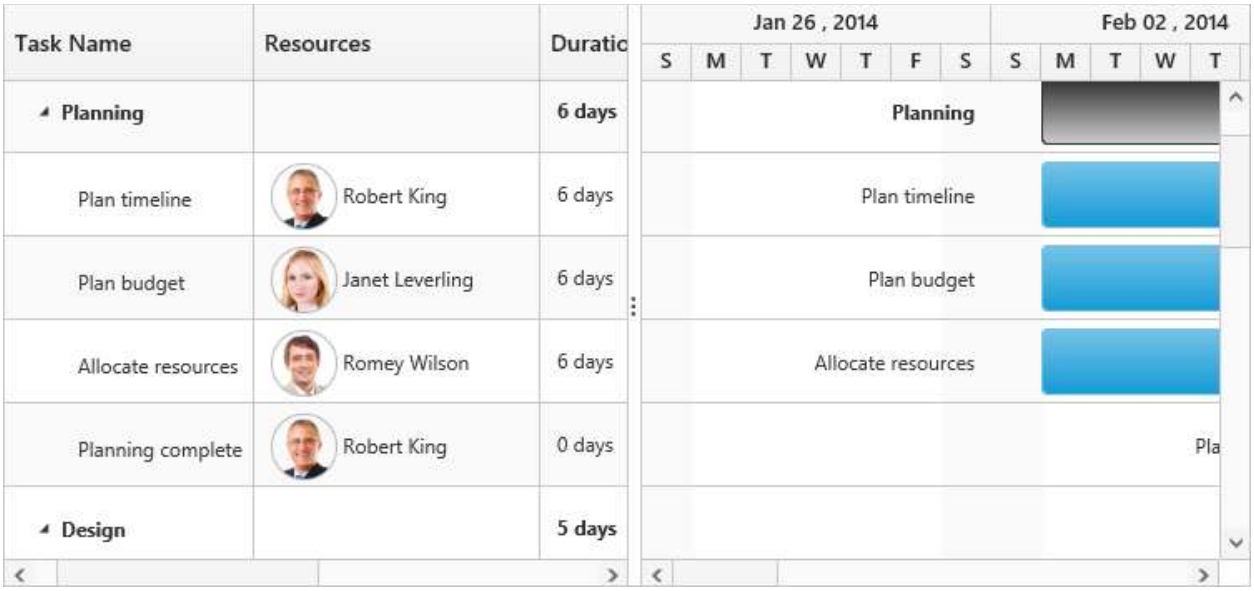
C#

```
public class GanttController : Controller
{
// GET: /GanttColumnTemplate/
public ActionResult GanttColumnTemplate()
{
var DataSource = GetTaskData();
ViewBag.datasource = DataSource;
return View();
}
public class TaskDetails
{
public int TaskID { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
public string EndDate { get; set; }
public int Duration { get; set; }
public string Progress { get; set; }
public List<TaskDetails> SubTasks { get; set; }
public List<object> ResourceID { get; set; }
public string Predecessors { get; set; }
}
public List<TaskDetails> GetTaskData()
{
List<TaskDetails> tasks = new List<TaskDetails>();
tasks.Add(new TaskDetails()
{
TaskID = 1,
TaskName = "Project Schedule",
StartDate = "02/06/2017",
EndDate = "03/10/2017"
});
tasks[0].SubTasks = new List<TaskDetails>();
tasks[0].SubTasks.Add(new TaskDetails()
{
TaskID = 2,
TaskName = "Planning",
StartDate = "02/06/2017",
EndDate = "02/10/2017"
});
tasks[0].SubTasks[0].SubTasks = new List<TaskDetails>();
tasks[0].SubTasks[0].SubTasks.Add(new TaskDetails()
{
TaskID = 3,
TaskName = "Plan timeline",
StartDate = "02/06/2017",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
EndDate = "02/10/2017",
Duration = 5,
Progress = "100",
ResourceID = new List<object>() { 1 }
});
//..
}
}
```

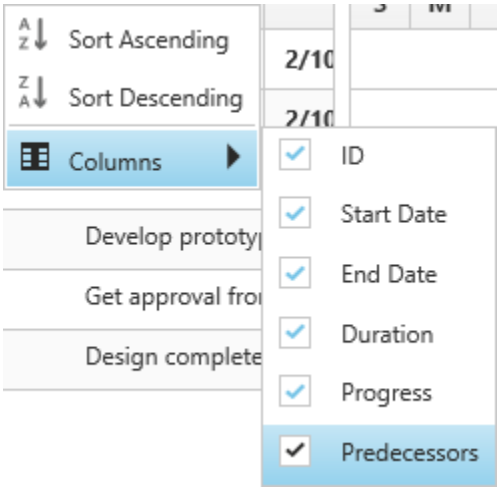
The following screenshot displays the customized column in Gantt control.



Column Menu

Show column chooser

Gantt supports enabling and disabling the visibility of the columns dynamically with the ShowColumnChooser property. The visibility of the custom columns can also be toggled with this property. Column chooser option is rendered as a sub menu item within the column menu in the Gantt columns.



The column menu is enabled with the ShowColumnChooser property, where the default value for this property is false.

The column menu provides the following options:

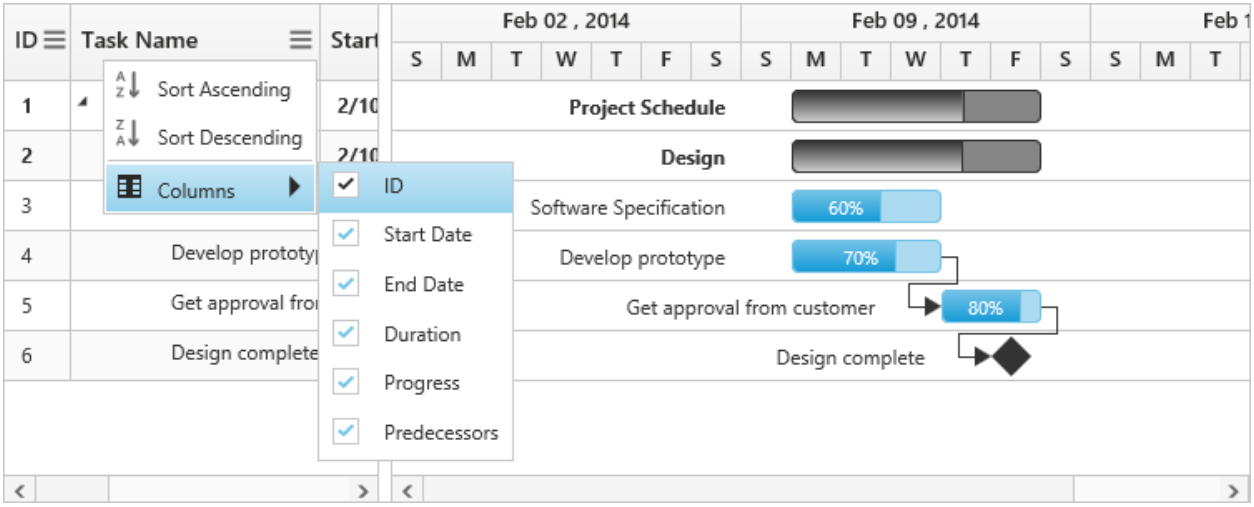
- Sort Ascending
- Sort Descending
- Columns

Sort Ascending and Sort Descending options can be enabled or disabled with the AllowSorting property. Single level sorting can be performed with these options. To perform multilevel sorting, the AllowMultiSorting property should be enabled. You can also disable the visibility of a particular column in the column collection manually by setting the visible property to false.

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
// ...
.ShowColumnChooser(true)
.AllowSorting(true)
.AllowMultiSorting(true)
)
@(Html.EJ().ScriptManager())
```

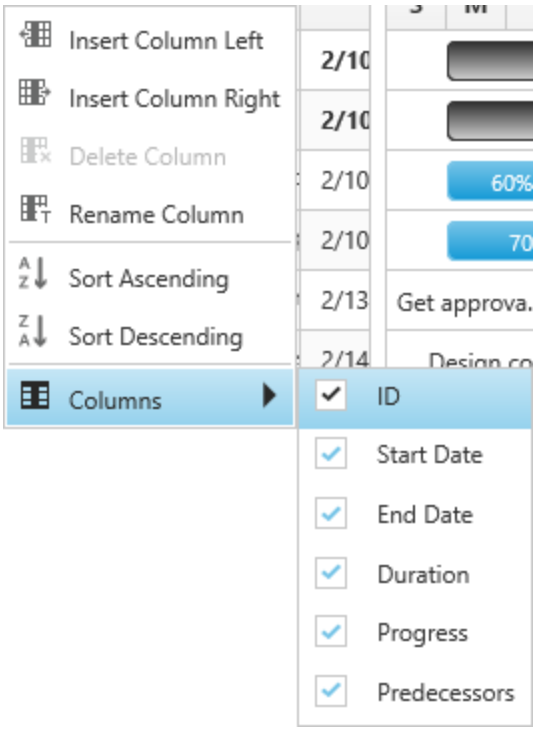
The following screenshot displays the column chooser in the Gantt control.



Show Column Options

You can customize the column with some more options with the ShowColumnOptions property. Use this property to insert a new column, delete a column and to update the header text of the column.

Copyright © 2001 - 2020 Syncfusion Inc.



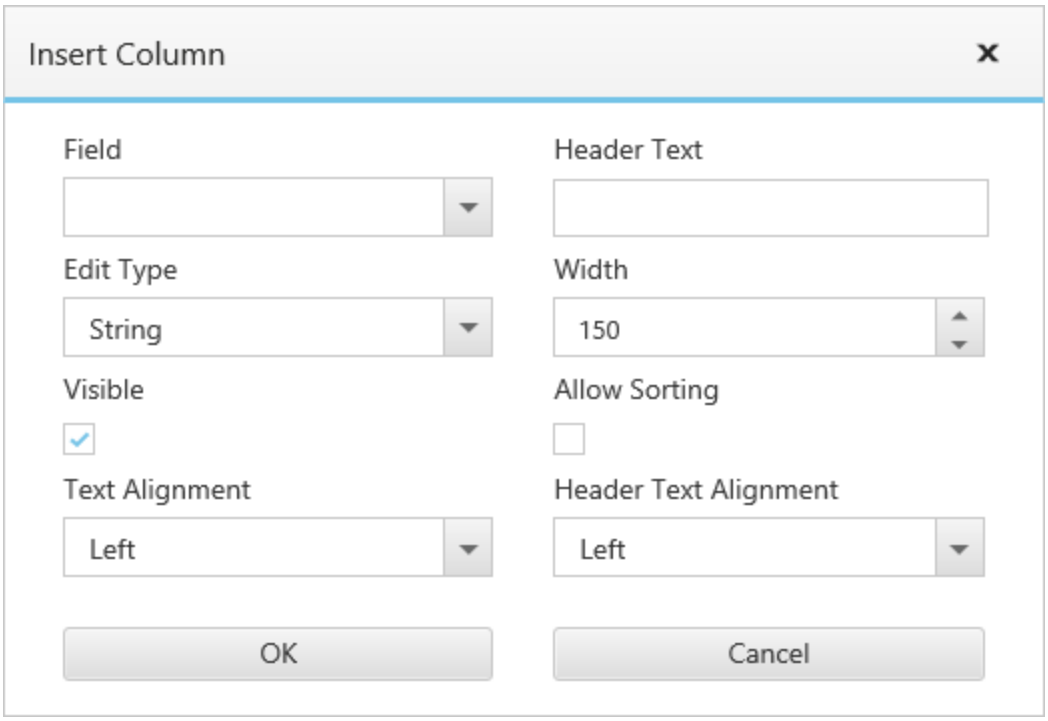
The column options can be enabled or disabled with the `ShowColumnOptions` property, where the default value for this property is `false`.

The column options provide the following options:

- Insert column left
- Insert column right
- Delete column
- Rename column

Inserting column provides the dialog to enter the details for the column.

Copyright © 2001 - 2020 Syncfusion Inc.

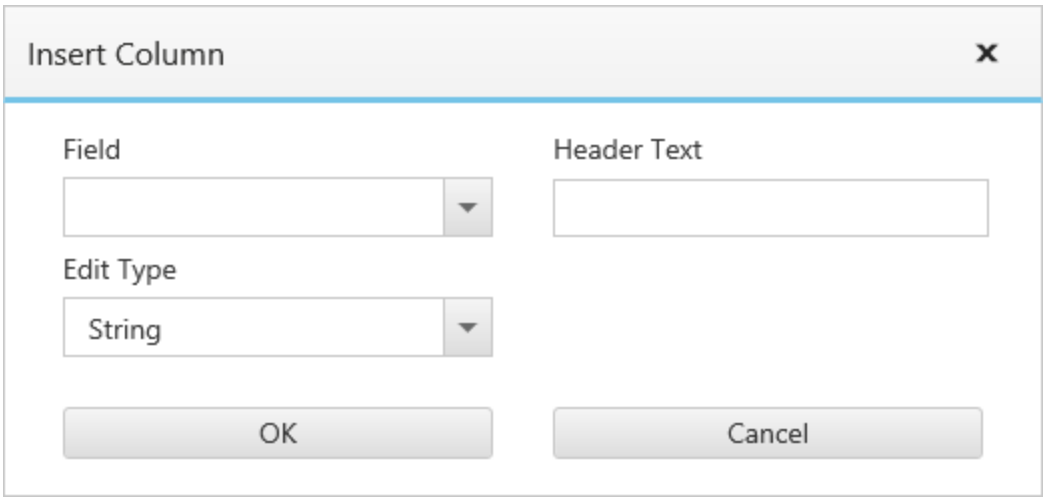


These fields can be customized with the ColumnDialogFields property. The following code sample shows you how to customize these fields.

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
// ...
.ShowColumnChooser(true)
.ShowColumnOptions(true)
.ColumnDialogFields(new List <GanttColumnDialogFields>(){
GanttColumnDialogFields.Field,
GanttColumnDialogFields.HeaderText,
GanttColumnDialogFields.EditType
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Change visibility of the columns dynamically

Gantt columns visibility can be changed dynamically by using showColumn and hideColumn methods. The below code example shows how to change the visibility of the column in Gantt dynamically.

ASPX-CS

```
<button id="hide_column">Hide</button>
<button id="show_column">Show</button>
@(Html.EJ().Gantt("gantt")
//..
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#hide_column").click(function () {
var ganttObj = $("#gantt").ejGantt("instance");
var column = ganttObj.getColumns()[0];
ganttObj.hideColumn(column.headerText);
});
$("#show_column").click(function () {
var ganttObj = $("#gantt").ejGantt("instance");
var column = ganttObj.getColumns()[0];
ganttObj.showColumn(column.headerText);
});
</script>
```

Change Tree/Expander Column

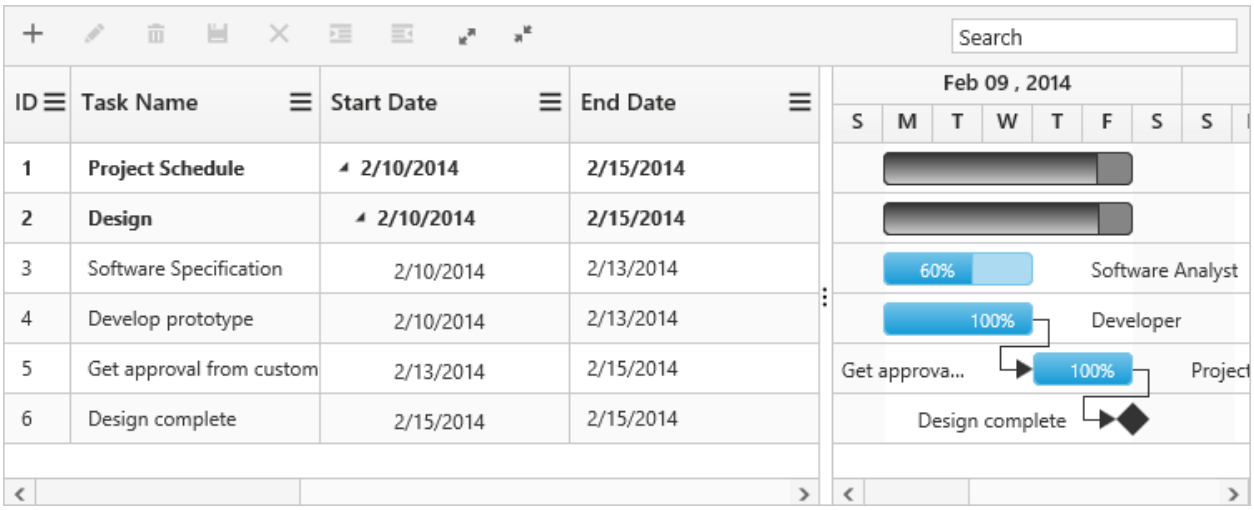
Tree/Expander column is a column in Gantt which has icons to expand/collapse the parent records. We can define the tree column index in Gantt by using `TreeColumnIndex` property, default value of this property was `0`. The following code example shows how to use this property.

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
// ...
.TreeColumnIndex(2)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The below screenshot shows the output of above code example.



Sequencing Tasks

How to enable serial number column in Gantt?

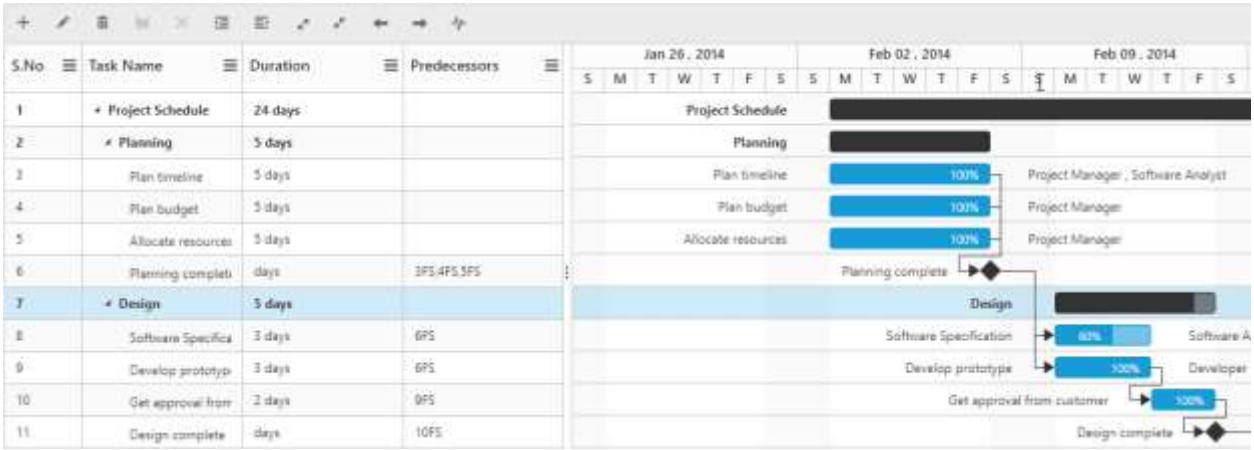
The serial or sequence number support in Gantt is used to index the tasks in a project. The Serial number column can be rendered by enabling the `EnableSerialNumber` property. On enabling this property the serial number column will be displayed and the Task Id column will be hidden, the tasks will be identified using the serial numbers. Further the column values for task predecessors will also be displayed using the serial numbers of the corresponding tasks, instead of task IDs.

Code snippets for enabling the serial number is as follows.

ASPX-CS

```
@(Html.EJ().Gantt("GanttSerialNumber")
//...
.EnableSerialNumber(true)
)
@(Html.EJ().ScriptManager())
```

The following screenshot displays the Serial number column in Gantt control.



The serial number column will be resequenced automatically on performing any actions which will change the row indexes of the tasks such as row drag and drop, deleting, adding.

Rows

Row represents a task information from the data source, and it is possible to perform the following actions in Gantt rows.

Adding a row

A row can be added in the following ways in Gantt.

- Toolbar
- Context menu
- Adding a row programmatically

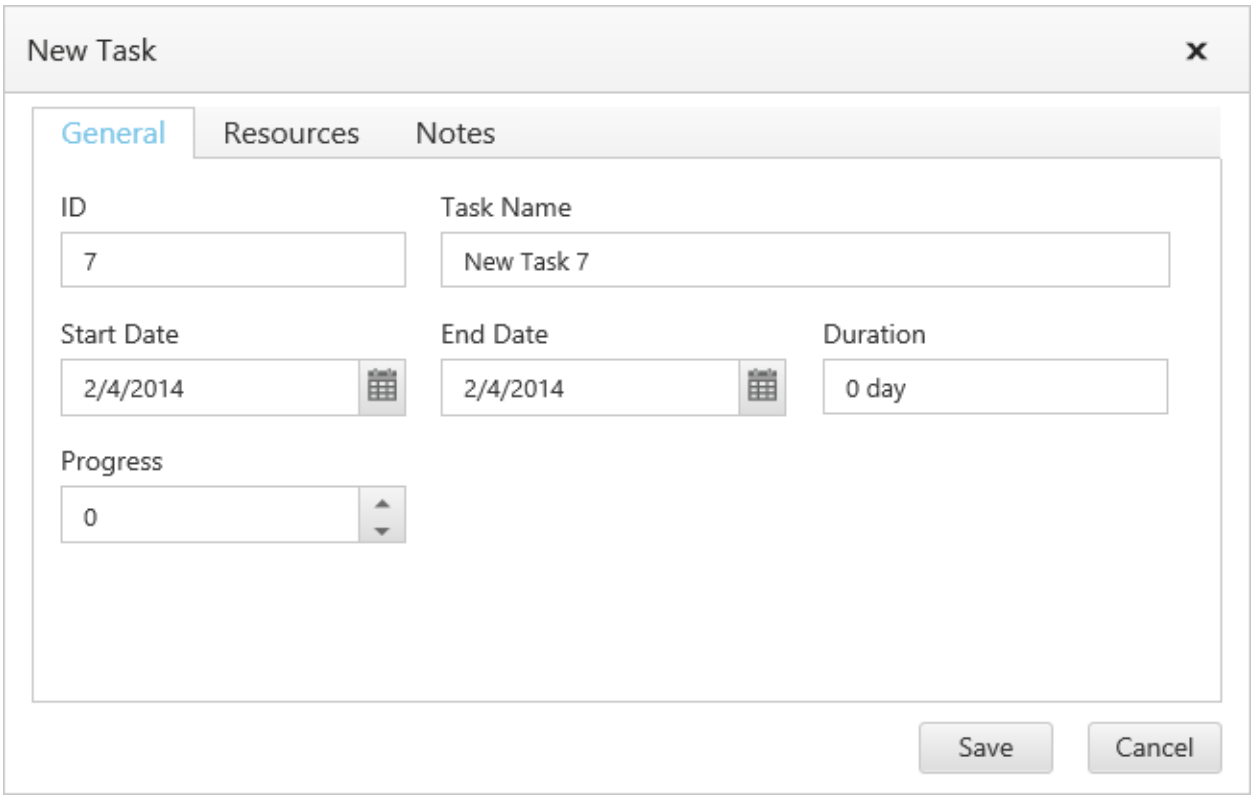
Toolbar adding

Row can be added in Gantt from toolbar while the EditSettings.AllowAdding property is set to true. On clicking the toolbar add icon, you need to provide the task information in the add dialog. If a row is previously selected, then the new row will be added below and in the same hierarchical level as that of the selected row. If there are no rows selected in Gantt, by default the new row will be added as the top most row in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.EditSettings(edit=>
{
edit.AllowAdding(true);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



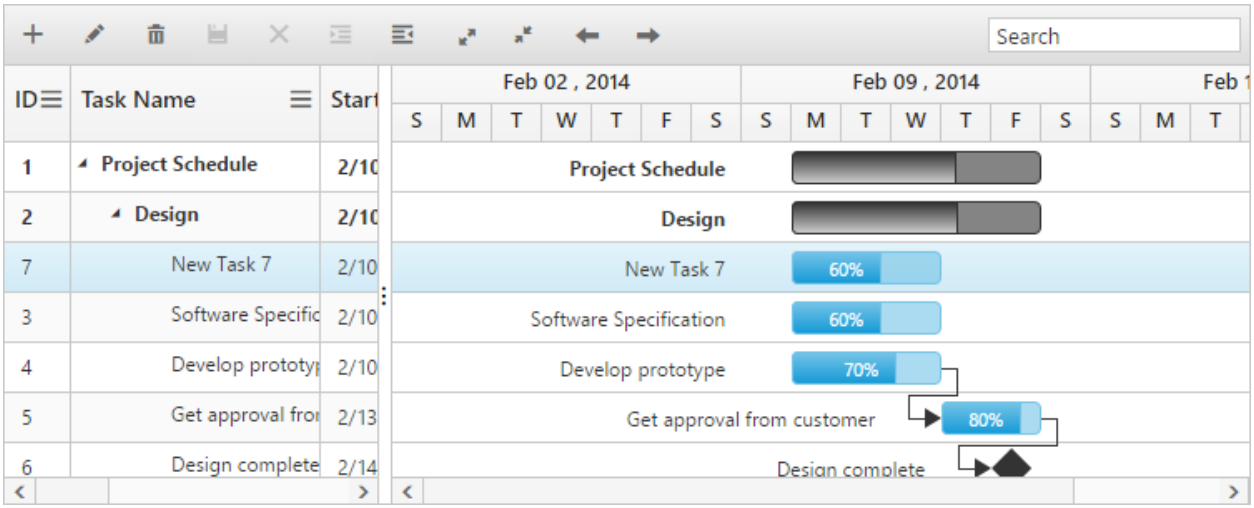
Context menu adding

New rows can either be added above or below the selected rows by using the default context menu, while the EnableContextMenu is set to true. The new row added will have the same task information similar to the selected row.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.EnableContextMenu(true)
.EditSettings(edit=>
{
edit.AllowAdding(true);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Adding a row programmatically

You can add rows in the below positions dynamically using addRecord public method and we can define the default new record add position by using rowPosition property.

- Top of all the rows
- Bottom to all the existing rows
- Above the selected row
- Below the selected row
- As child to the selected row

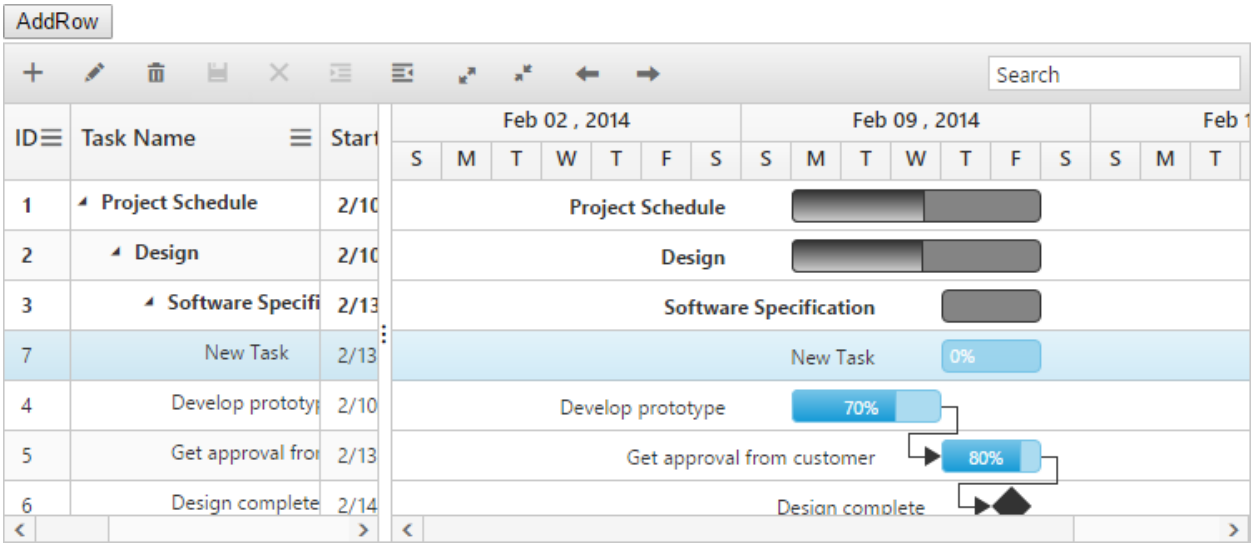
The below code example explains on adding a row using a custom button.

ASPX-CS

```
<button id="addRow"
style="top:27px;left:50px;position:absolute">AddRow</button>
@(Html.EJ().Gantt("GanttContainer")
//...
)
@(Html.EJ().ScriptManager())
$("#addRow").click(function(args) {
//Create Gantt object
var ganttObj = $("#GanttContainer").data("ejGantt");
// data to be added
var data = {
taskID: 7,
taskName: "New Task",
startDate: "02/13/2014",
endDate: "02/14/2014",
duration: 2
};
ganttObj.addRecord(data, ej.Gantt.AddRowPosition.Child);
})
```

The following screen shot shows to add new row as child.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: While adding a new record ActionComplete event will be triggered with arguments addedRecord value as new record and requestType value as save, using this event we can update new record information in server database.

Row drag and drop

It is possible to dynamically re-arrange the rows in the Gantt control by using the AllowDragAndDrop property. With this property, row drag and drop can be enabled or disabled. Rows can be inserted above, below as a sibling or as a child to the existing row with the help of this feature. A default tooltip is rendered while dragging the Gantt row and this tooltip can be customized by the DragTooltip property. This property has inner properties such as ShowTooltip, TooltipItems and TooltipTemplate.

The ShowTooltip property is used to enable or disable the tooltip. By default, this property value is false.

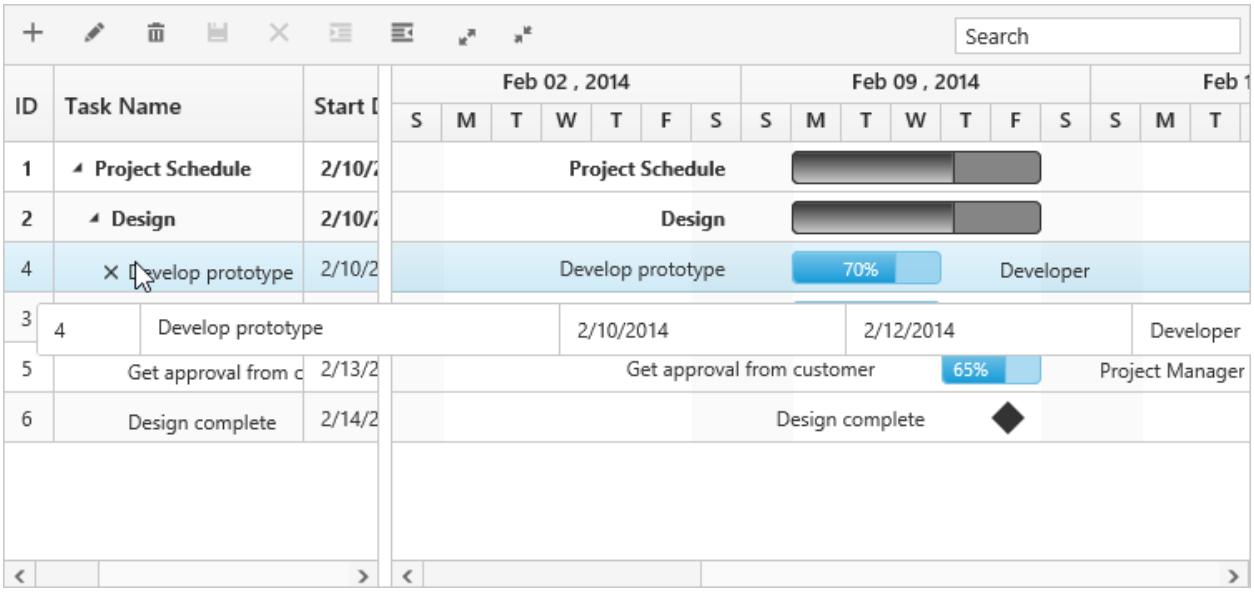
The following code explains about enabling the row drag and drop with the default tooltip in the Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowDragAndDrop(true)
.DragTooltip(dt =>
{
dt.ShowTooltip(true);
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot depicts a row drag and drop in the Gantt widget.

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample for row drag and drop in Gantt.

Customizing drag tooltip

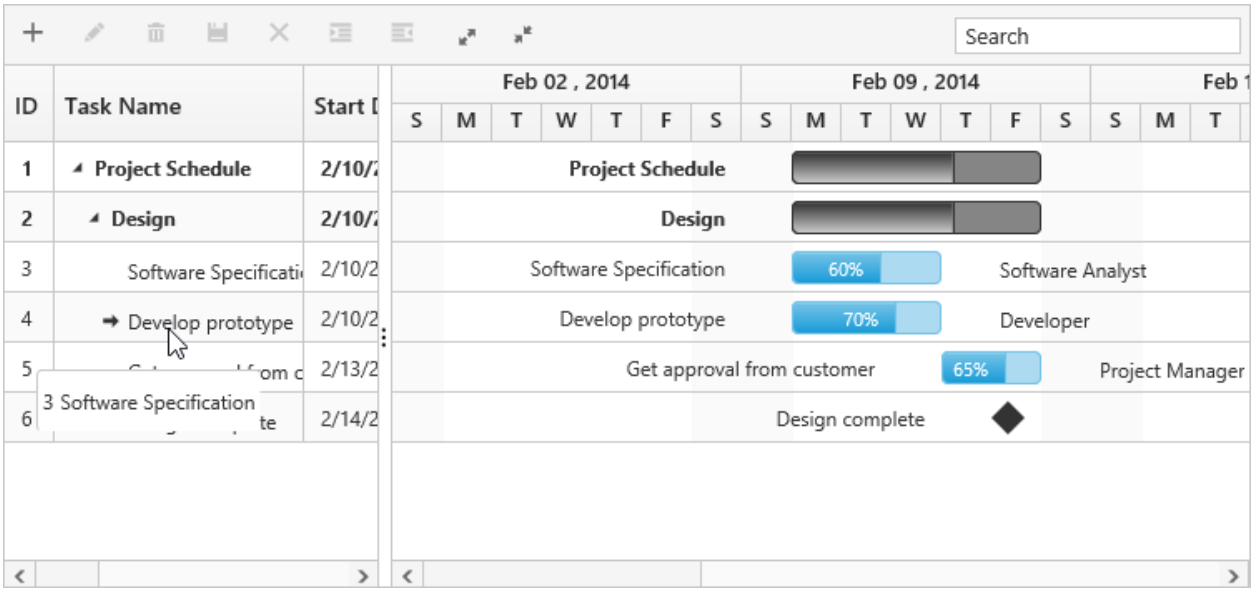
The TooltipItems property is used to customize the tooltip items. By using this property, specific fields can be rendered in the tooltip. By default this property value is null, and all the defined field items are rendered in the tooltip.

The TooltipTemplate property renders the template tooltip for row drag and drop in the Gantt control by using the JsRender template. You can provide either the id value of the script element or the script element to the property.

The following code shows how to render row drag tooltip with tooltip template.

ASPX-CS

```
<script id="customTooltip" type="text/x-jsrender">
<tr>
<td class="border" style='height:30px;'>
<div>{{"{{"}}:#data['TaskId'] {{}}}}</div>
</td>
<td class="border" style='height:30px;'>
<div>{{"{{"}}:#data['TaskName'] {{}}}}</div>
</td>
</tr>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.AllowDragAndDrop(true)
.DragTooltip(dt =>
{
dt.ShowTooltip(true);
dt.TooltipTemplate("#customTooltip");
})
)
@(Html.EJ().ScriptManager())
```



Customize row drag and drop action

In Gantt, RowDragStart, RowDrag and RowDragStop events are triggered on row drag and drop action. Using this event we can prevent drag and drop action of particular task and validate the drop position on particular row. The below code example shows how to use this events.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowDragAndDrop(true)
.ClientSideEvents(eve =>
{
eve.RowDrag("rowDrag");
eve.RowDragStart("rowDragStart");
eve.RowDragStop("rowDragStop");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function rowDrag(args) {
if (args.targetRow.taskId == 5 && args.dropPosition == "insertAsChild") //
Can't drop task as child on Task Id 5
args.canDrop = false;
}
function rowDragStart(args) {
if (args.draggedRow.taskId == 6) // Task Id 6 can't be dragged
args.cancel = true;
}
function rowDragStop(args) {
if (args.targetRow.taskId == 6) // Can't drop any task on Task Id 6
args.cancel = true;
}
```

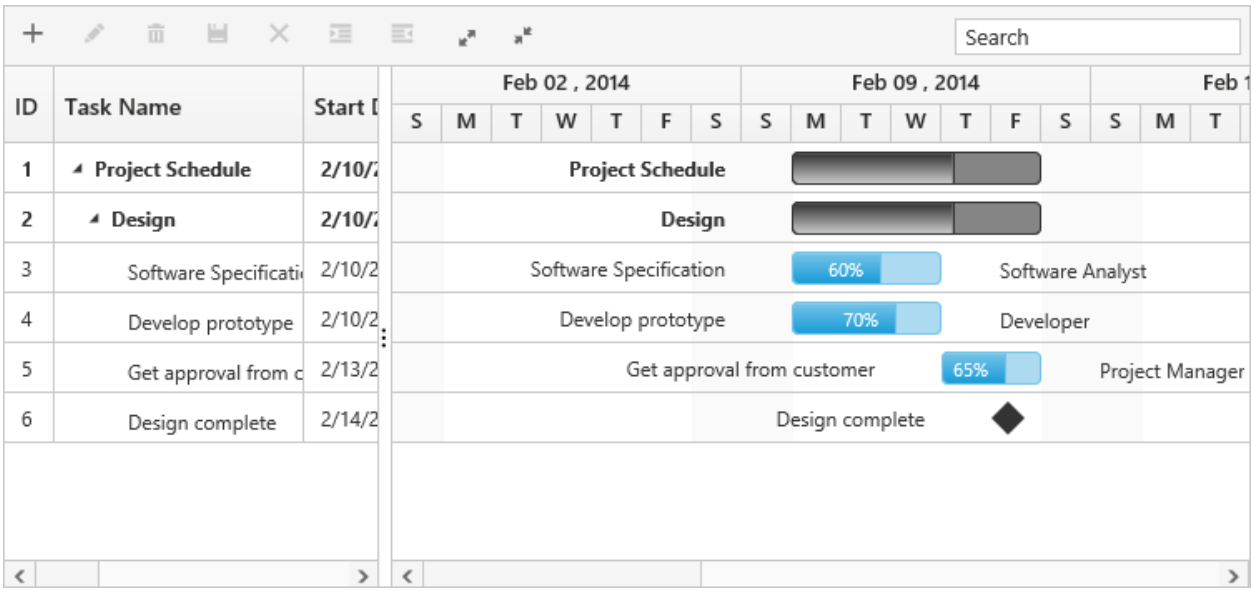
Copyright © 2001 - 2020 Syncfusion Inc.

Alternate row background

In Gantt, it is possible to enable or disable the alternate row background using the EnableAltRow property. The following code example shows you to disable the alternate row color in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EnableAltRow(false)
)
@(Html.EJ().ScriptManager())
```



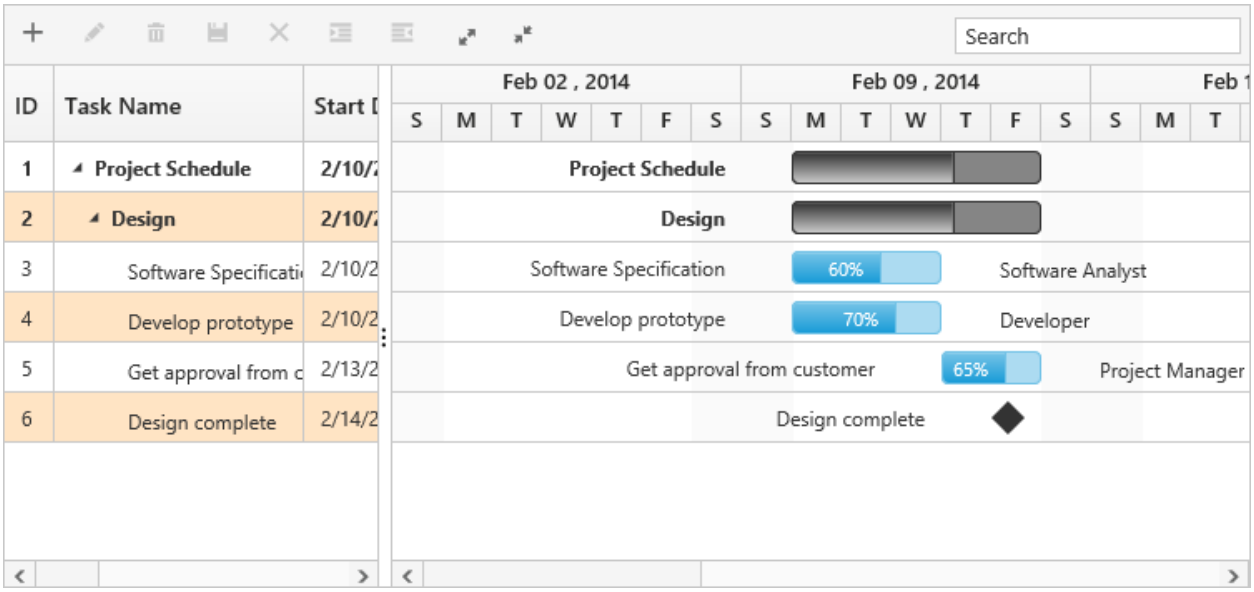
Change alternate rows background

The alternate rows background can be changed by setting the background color for the alternate row elements using CSS. The following code example shows you how to change the alternate rows background color in Gantt.

ASPX-CS

```
<head>
<style>
.e-treegrid .e-alt-row {
background-color: Bisque;
}
</style>
</head>
@(Html.EJ().Gantt("Gantt")
//...
.EnableAltRow(true)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

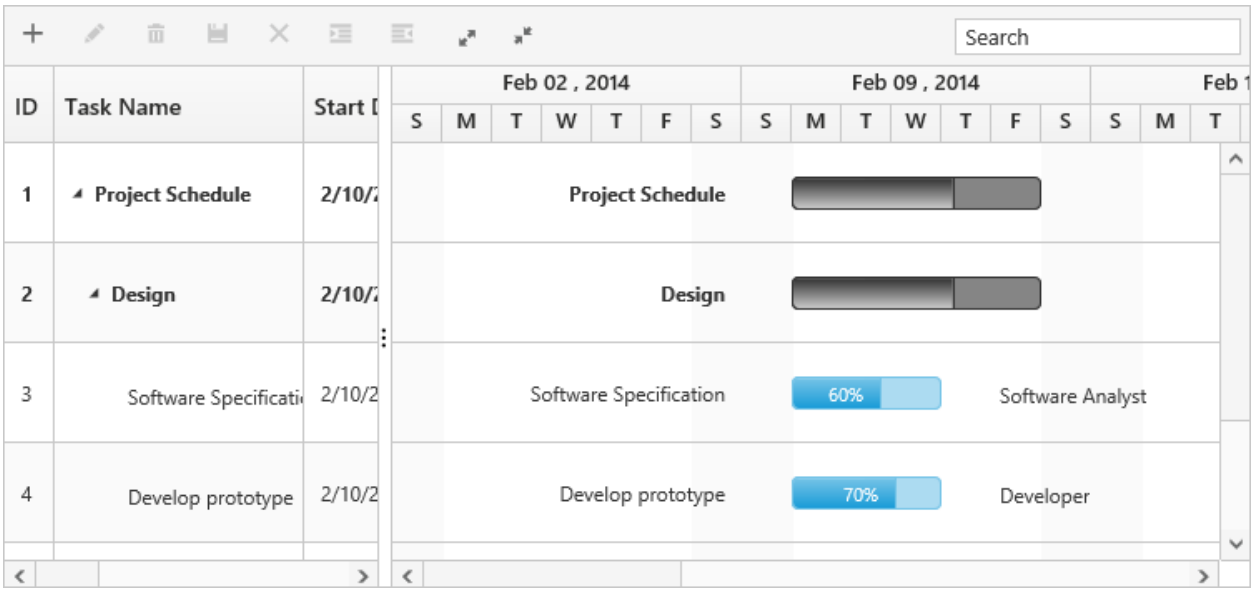


Row height

It is possible to change the height of the row in Gantt by setting row height in pixels to the RowHeight property. The following code example explains how to change the row height in Gantt at load time.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.RowHeight(60)
)
@(Html.EJ().ScriptManager())
```



Copyright © 2001 - 2020 Syncfusion Inc.

Expand/Collapse Row

In Gantt parent tasks are expanded/collapsed by using expand/collapse icons, expand all/collapse all toolbar items and by using public methods. By default all tasks in Gantt was rendered in expanded state but we can change this status in Gantt.

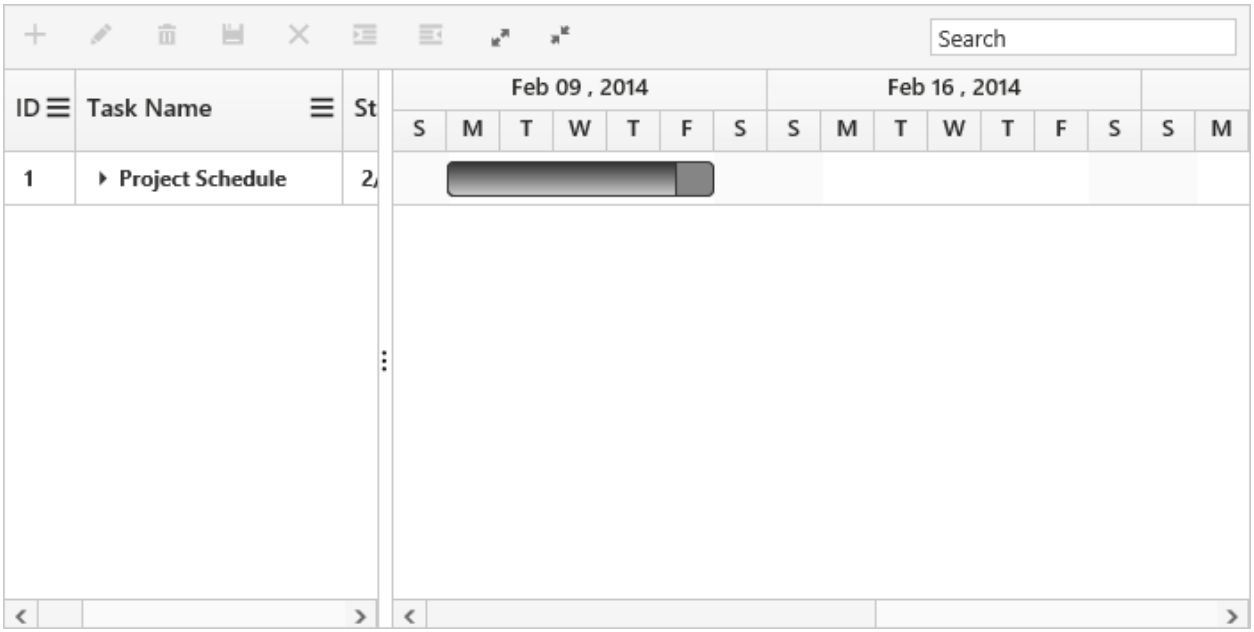
Collapse all tasks at Gantt load

All tasks available in Gantt was rendered in collapsed state by setting `EnableCollapseAll` property as `true`. The following code example shows how to use this property.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EnableCollapseAll(true)
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.



Define expand/collapse status of tasks at Gantt load

In Gantt, we can render some tasks in collapsed state and some tasks in expanded state, this can done by defining expand status of the task in data source. This value was mapped to Gantt control by using `ExpandStateMapping` property. The following code example shows how to use this property.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ExpandStateMapping("ExpandState")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

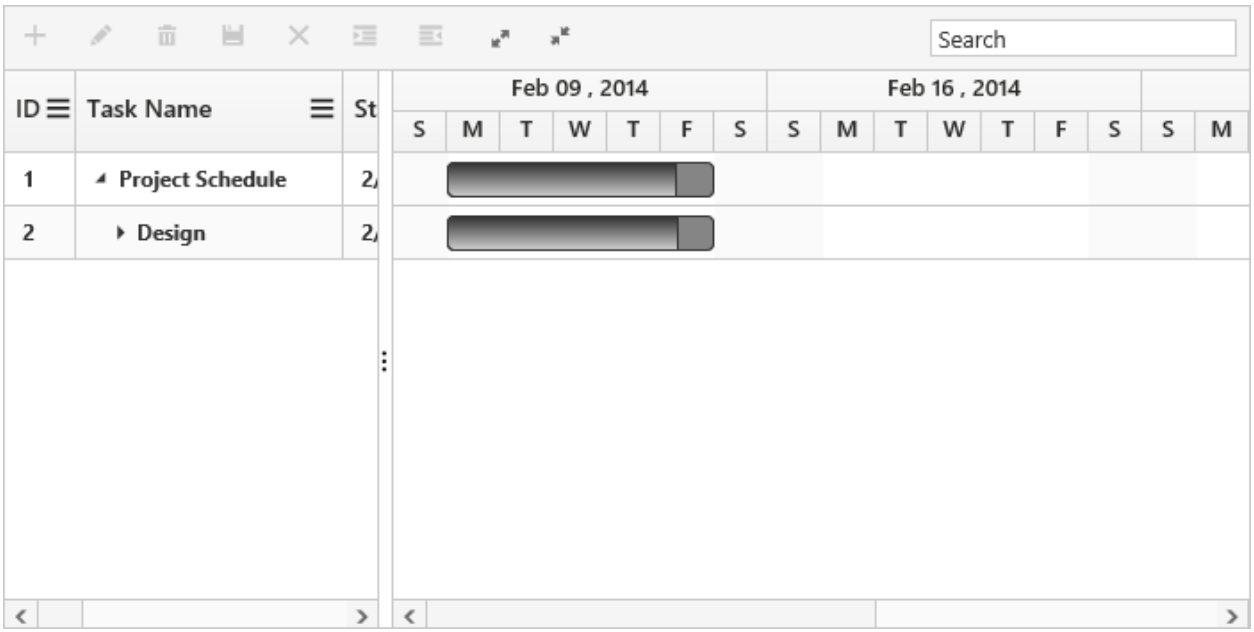
C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
return View();
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
list.Add(new DefaultData()
{
Id = 1,
Name = "Project Schedule",
ExpandState = true,
Children = (new List<DefaultData>()
{
new DefaultData()
{
Id = 2,
Name = "Design",
ExpandState = false,
},
//..
})
});
return list;
}
}
}
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Expand/Collapse the task dynamically

Gantt tasks can be expanded/collapsed dynamically by using expandCollapseRecord method. The following code example shows how to use this method.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#expandCollapseTask").click(function () {
var ganttObj = $("#GanttContainer").ejGantt("instance");
ganttObj.expandCollapseRecord(2);
});
</script>
```

Note: This method was used to toggle the expand status of Gantt task, when we pass the id of task which is in expanded state to this method and this task will be collapsed, similarly collapsed task will be expanded.

Expand/Collapse all the tasks dynamically

All the tasks in Gantt will be expanded/collapsed by clicking expandAll and collapseAll toolbar items or by using expandAllItems and collapseAllItems methods. We can invoke this methods dynamically on any action like external button click. The below code example shows how to use this methods.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.ToolbarSettings(ts =>
{
ts.ShowToolbar(true);
ts.ToolbarItems(new List<GanttToolBarItems>()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
//..
GanttToolBarItems.ExpandAll,
GanttToolBarItems.CollapseAll
});
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#expandAllTasks").click(function () {
var ganttObj = $("#GanttContainer").ejGantt("instance");
ganttObj.expandAllItems();
});
$("#collapseAllTasks").click(function () {
var ganttObj = $("#GanttContainer").ejGantt("instance");
ganttObj.collapseAllItems();
});
</script>
```

Customize expand/collapse action

On expand action Expanding and Expanded event will be triggered with current expanding row's information. Similarly on collapse action Collapsing and Collapsed event will be triggered. Using this events and it's arguments we can customize the expand/collapse action. The following code example shows how to prevent the particular row from expand/collapse action using Expanding and Collapsing event.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ExpandStateMapping("ExpandState")
.ClientSideEvents(eve =>
{
eve.Expanding("expanding");
eve.Collapsing("collapsing");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function expanding(args) {
if (args.data.taskId == 2) // we can't expand Task Id 2
args.cancel = true;
}
function collapsing(args) {
if (args.data.taskId == 1) // we can't collapse Task Id 1
args.cancel = true;
</script>
```

Task Dependencies

Task dependencies or task relationship can be established between two task in Gantt. This dependencies effect the project schedule, if we change the predecessor of one task it will affect the successor task, which effect next task and so on.

Copyright © 2001 - 2020 Syncfusion Inc.

Task Relationship Types

Task relationships are categorized into four types based on the start and finish date of the task.

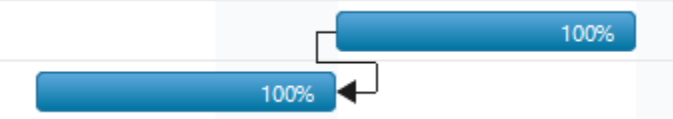
Start to Start(SS)

You cannot start a task until the other task also starts.



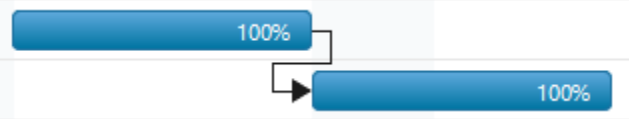
Start to Finish(SF)

You cannot finish a task until the other task is started.



Finish to Start(FS)

You cannot start a task until the other task is completed.



Finish to Finish(FF)

You cannot finish a task until the other task is completed.



Define task relationship

Task relationships are defined in data source as string value and this values are mapped to Gantt control by using PredecessorMapping property. The following code example shows you how to enable the predecessor in the Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.PredecessorMapping("Predecessor")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

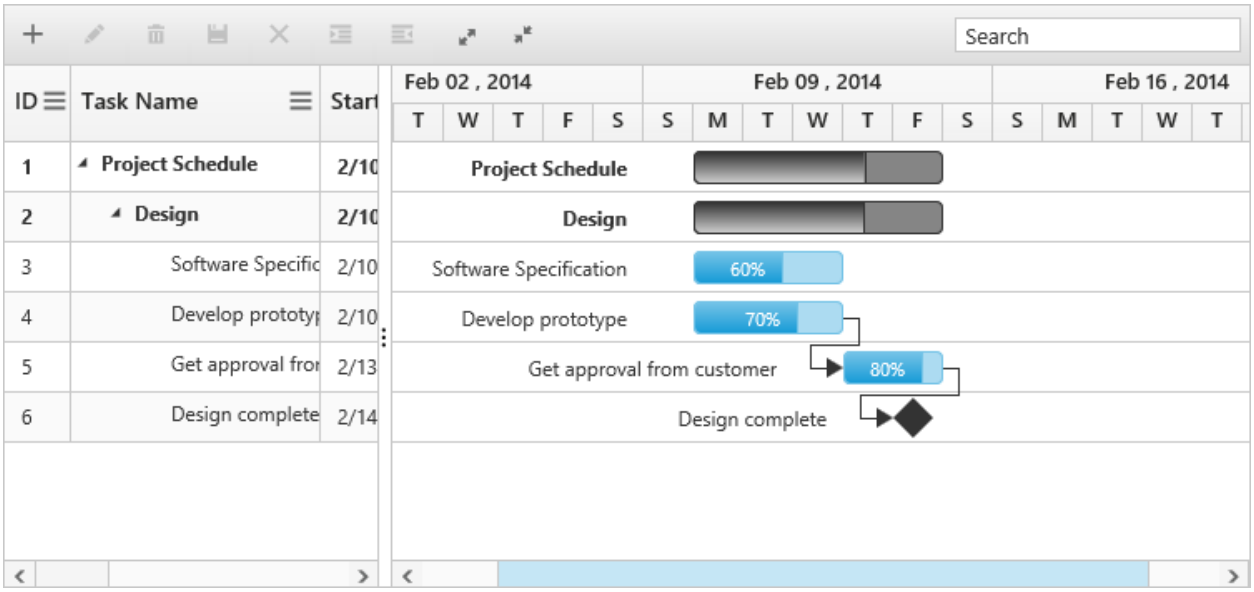
```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewBag.datasource = DataSource;
return View();
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
list.Add(new DefaultData()
{
//..
Children = (new List<DefaultData>()
{
new DefaultData()
{
Id = 4,
Name = "Develop prototype"
},
new DefaultData()
{
Id = 5,
Name = "Get approval from customer",
Predecessor: "4FS",
},
new DefaultData()
{
Id = 6,
Name = "Design complete",
Predecessor: "5FS",
},
//..
})
});
return list;
}
}
}
```

The following screenshot displays the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Task relationship

Predecessor offset with duration units

In Gantt, predecessor offset can be defined with the following duration units.

- Day
- Hour
- Minute

We can define offset with various offset duration units for predecessors by using the below code example.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.PredecessorMapping("Predecessor")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

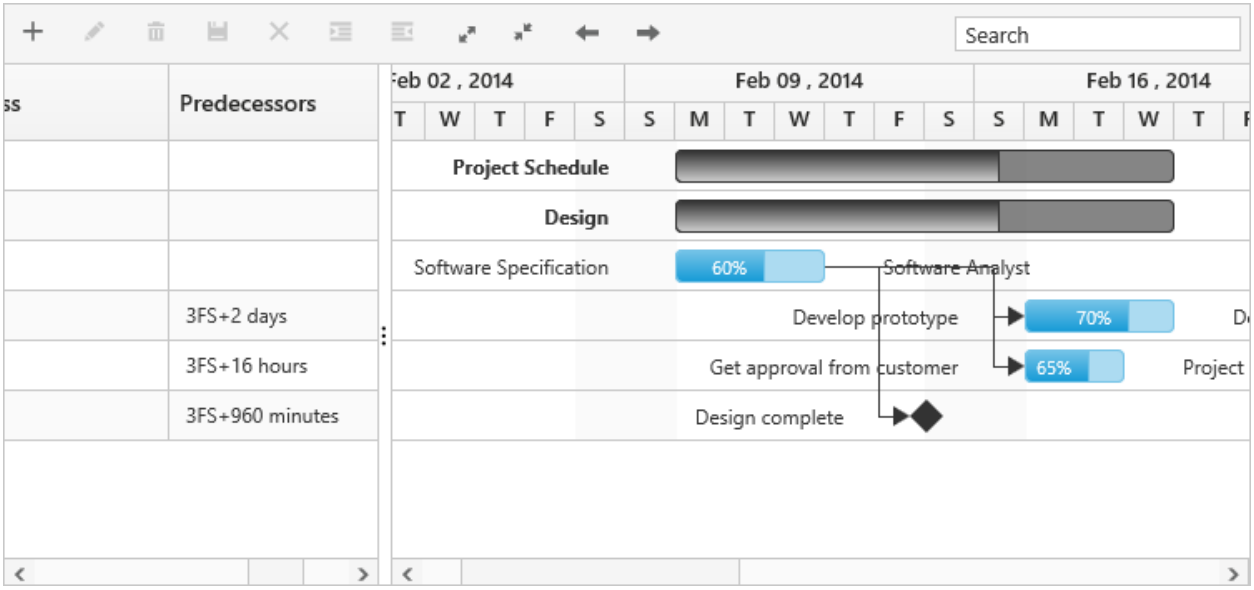
C#

```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
return View();
}
public class GanttDefaultData
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
list.Add(new DefaultData()
{
//..
Children = (new List<DefaultData>()
{
new DefaultData()
{
//..
Predecessor: "3FS+2d",
},
new DefaultData()
{
//..
Predecessor: "3FS+16h",
},
new DefaultData()
{
//..
Predecessor: "3FS+960m",
},
//..
})
});
return list;
}
}
}
```

The following screen shot depicts the duration unit support in the predecessor offset.



Copyright © 2001 - 2020 Syncfusion Inc.

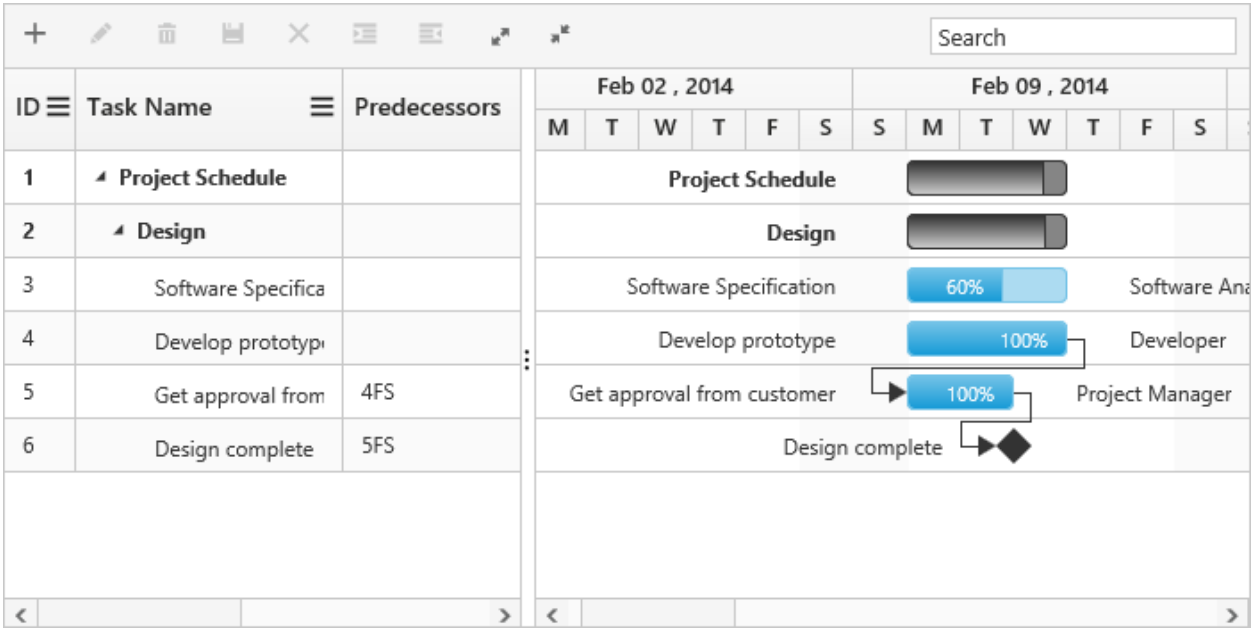
Enable/disable predecessor validation

By default Gantt tasks date values are validated as per predecessor task and it's relationship type, we can disable/enable this validation by using `EnablePredecessorValidation` property. The following code example shows how to disable predecessor validation in Gantt.

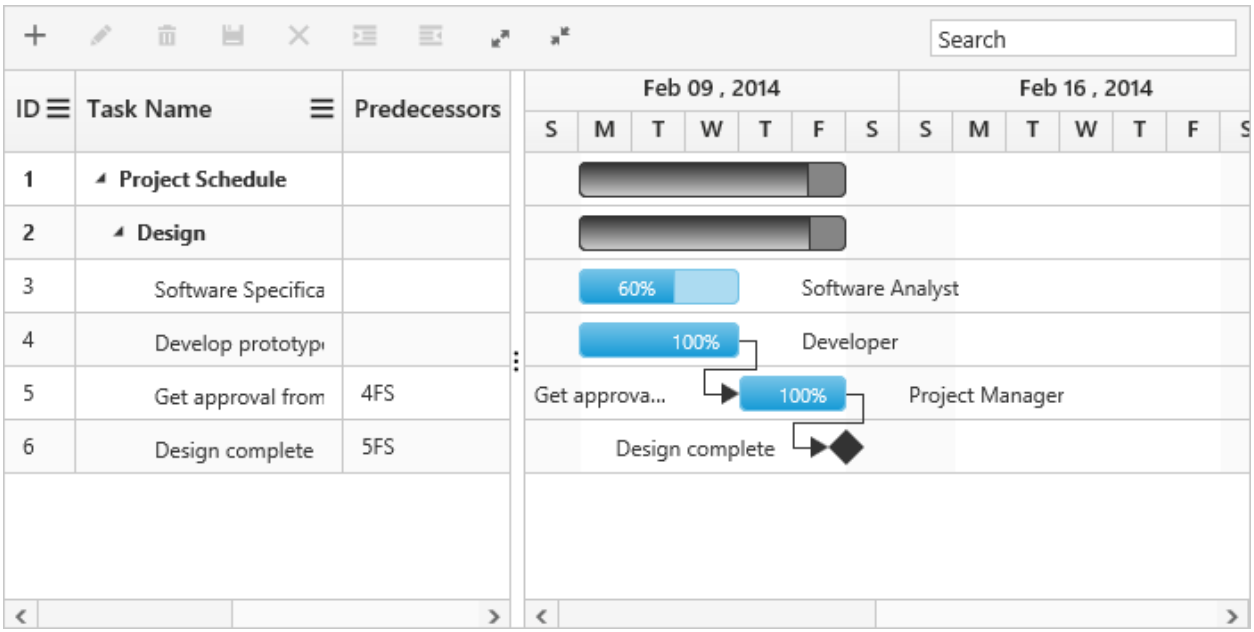
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EnablePredecessorValidation(false)
.PredecessorMapping("Predecessor")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.



Predecessor validation disabled mode



Predecessor validation enabled mode

Validate predecessor links on editing

In Gantt, it is possible to validate the taskbar editing based on the predecessor connections. Below are the following ways to validate the taskbar editing.

- Using ActionBegin event
- Using Validation dialog

Using ActionBegin event

The ActionBegin event with requestType argument as **validateLinkedTask** will be triggered when editing a task with predecessor links.

It is possible to validate the editing within the ActionBegin event using the **validateMode** event argument. The validateMode event argument has the following properties.

Argument	Default value	Description
args.validateMode.respectLink	false	In this validation mode, the predecessor links will be considered as high priority. With this mode enabled, when the successor task is moved before predecessor task's end date, the editing will be reverted and dates will be validated based on the dependency links.
args.validateMode.removeLink	false	In this validation mode, the taskbar editing will be considered as high priority, where in the case of inappropriate task dates the dependency links will be removed and tasks will be moved to the edited date.

Copyright © 2001 - 2020 Syncfusion Inc.

args.validateMode.preserveLinkWithEditing	true	In this validation mode, taskbar editing will be considered along with the dependency links. There will be no validations in task editing.

By default, the **preserveLinkWithEditing** validation mode will be enabled thus the validations will not occur when editing the linked tasks.

The below code snippet explains enabling the **respectLink** validation mode while editing the linked tasks in the ActionBegin event.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ClientSideEvents(eve =>
{
eve.ActionBegin("actionBegin");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function actionBegin(args) {
if (args.requestType == "validateLinkedTask") {
args.validateMode.respectLink = true;
}
}
</script>
```

Using validation dialog

When disabling all the validation modes in the ActionBegin event, a validation popup will be displayed prompting the user to select the validation mode to validate taskbar editing.

This validation popup will display different options based on the successor task's start date after editing.

If the user moved the successor task, that starts after predecessor task's end date then the dialog will be rendered with below options,

- Cancel, Keep the existing link.
- Remove the link and move the task to start on edited date.
- Move the task to start on edited date and keep the link.

If the user moved the successor task, that starts before the predecessor task's end date then the dialog will be rendered with below options.

If the user moved the successor task, that starts before the predecessor task's end date then the dialog will be rendered with below options.

- Cancel, Keep the existing link.
- Remove the link and move the task to start on edited date.

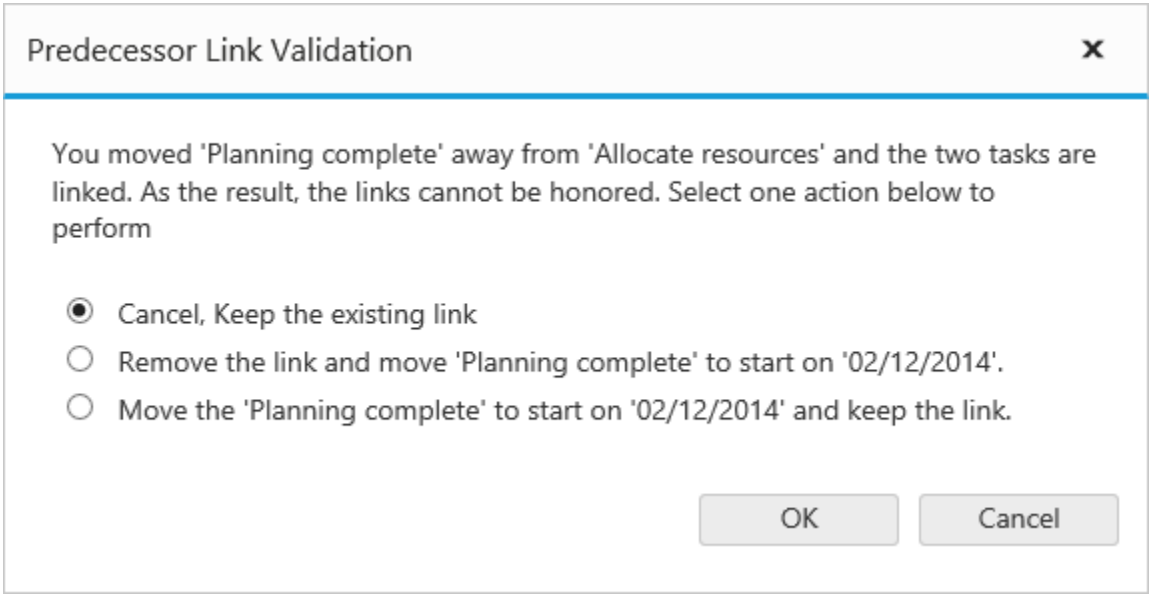
Copyright © 2001 - 2020 Syncfusion Inc.

The following code example explains this.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ClientSideEvents(eve =>
{
eve.ActionBegin("actionBegin");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function actionBegin(args) {
if (args.requestType == "validateLinkedTask") {
args.validateMode.preserveLinkWithEditing = false;
}
}
</script>
```

In this case, if the user dragging action violated the predecessor type then the following dialog will be rendered to perform operation.



Time Options

Start date and End date in the datasource defines the starting point and ending point of a task in the project. It can be of two data types,

- Date
- Datetime

The datetime data type for start date and end date defines the exact starting point and ending point of a task along with time details of a day. There are two types of time scales available based on working hours,

- TimeScale8Hours for 8 hour working scale
- TimeScale24Hours for 24 hour working scale

Use the following code example for setting different time scales of working hours.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.WorkingTimeScale(GanttWorkingTimeScale.TimeScale24Hours)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Use the following code example for including time options in start date and end date of a task.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.TaskIdMapping("TaskId")
.TaskNameMapping("TaskName")
.StartDateMapping("StartDate")
.DurationMapping("Duration")
.ProgressMapping("Progress")
.DateFormat("M/d/yyyy hh:mm:ss tt")
.ScheduleStartDate("02/01/2014 00:00:00 AM")
.WorkingTimeScale(GanttWorkingTimeScale.TimeScale24Hours)
.ScheduleEndDate("03/14/2016 00:00:00 PM")
.ChildMapping("SubTasks")
.TreeColumnIndex(1)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

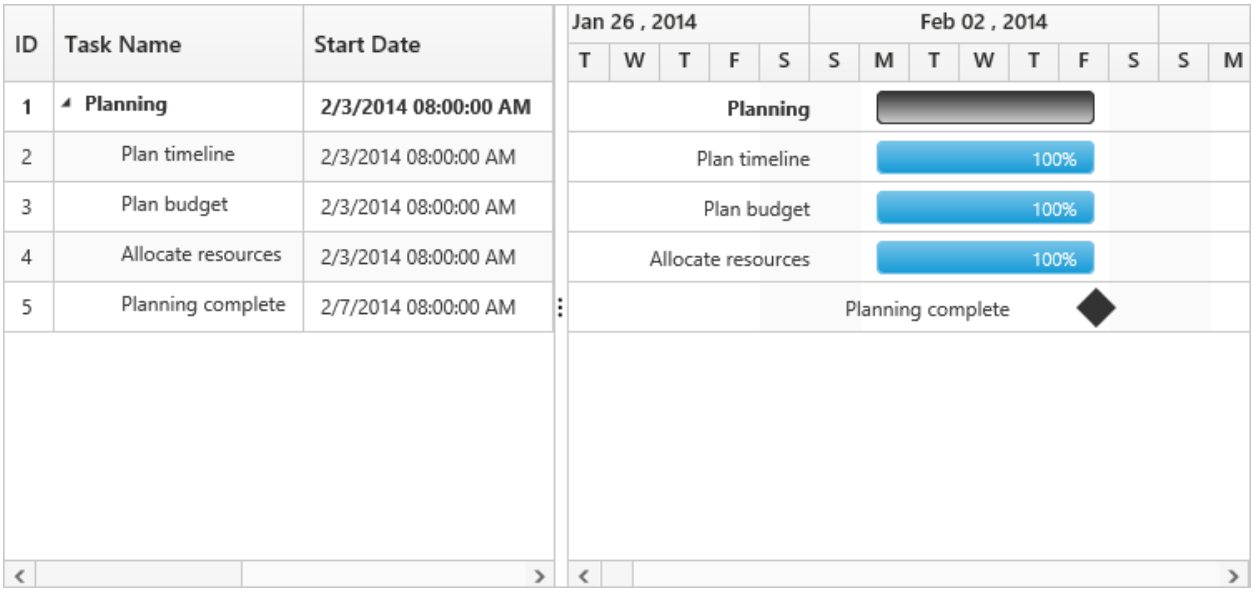
```
public class GanttController : Controller
{
// GET: /Gantt/
public ActionResult Gantt()
{
ViewBag.datasource = this.GetDataSource();
return View();
}
private List<Data> GetDataSource()
{
List<Data> DataCollection = new List<Data>();
Data ParentData = new Data()
{
TaskId = 1,
TaskName = "Planning",
StartDate = "02/03/2014 08:00:00 AM",
EndDate = "02/07/2014 05:00:00 PM",
Duration = 5
};
Data ChildData = new Data()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
TaskId = 2,
TaskName = "Plan timeline",
StartDate = "02/03/2014 08:00:00 AM",
EndDate = "02/07/2014 05:00:00 PM",
Duration = 5,
Progress = "100",
};
ParentData.SubTasks = new List<Data>();
ParentData.SubTasks.Add(ChildData);
ChildData = new Data()
{
TaskId = 3,
TaskName = "Plan Budget",
StartDate = "02/03/2014 08:00:00 AM",
EndDate = "02/07/2014 05:00:00 PM",
Duration = 5,
Progress = "100",
};
ParentData.SubTasks.Add(ChildData);
ChildData = new Data()
{
TaskId = 4,
TaskName = "Allocate Resources",
StartDate = "02/03/2014 08:00:00 AM",
EndDate = "02/07/2014 05:00:00 PM",
Duration = 5,
Progress = "100",
};
ParentData.SubTasks.Add(ChildData);
ChildData = new Data()
{
TaskId = 5,
TaskName = "Planing complete",
StartDate = "02/07/2014 08:00:00 AM",
EndDate = "02/07/2014 05:00:00 PM",
Duration = 0,
};
ParentData.SubTasks.Add(ChildData);
DataCollection.Add(ParentData);
return DataCollection;
}
public class Data
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public string  StartDate { get; set; }
public string  EndDate { get; set; }
public int Duration { get; set; }
public List<Data> SubTasks { get; set; }
public string Progress { get; set; }
}
}
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Validating Schedule Dates

Validating schedule dates is used to change the schedule start date and end date dynamically. By this support, ScheduleStartDate and ScheduleEndDate can be automatically updated from the data source. When you change the date of any task item to the date that is beyond ScheduleStartDate or ScheduleEndDate through cell editing, taskbar editing, dialog editing or toolbar operation, then the ScheduleStartDate and ScheduleEndDate can be dynamically updated based on that task item's date.

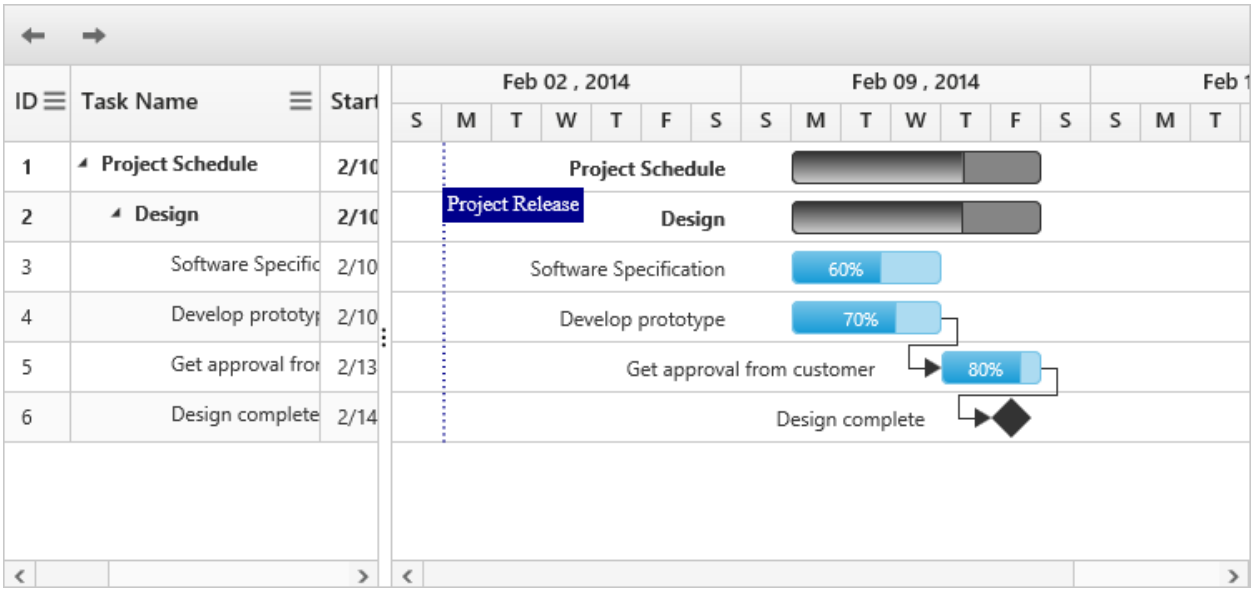
PrevTimeSpan and NextTimeSpan toolbar items are used to create new time span based on the schedule mode.

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
// ...
.ScheduleStartDate("02/01/2014")
.ScheduleEndDate("03/14/2014")
.ToolbarSettings(tbs=>
{
tbs.ShowToolbar(true);
tbs.ToolbarItems(new List<GanttToolBarItems>(){
GanttToolBarItems.PrevTimeSpan,
GanttToolBarItems.NextTimeSpan,
});
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot illustrates the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Scheduling Modes

The Gantt provides support for automatic and manual task scheduling modes. Scheduling mode of a task is used to indicate whether the start and end dates of a task will be automatically validated or not. Using the property TaskSchedulingMode we can able to change the scheduling mode of a task. The following are the enumeration values that can be set to the property TaskSchedulingMode.

- Auto
- Manual
- Custom

Automatically Scheduled Tasks

When the TaskSchedulingMode property is set as Auto scheduling mode, all the tasks in the project will be rendered as automatically scheduled tasks. Thus the start and end dates of the tasks in the project will be automatically validated. The tasks will be automatically scheduled based on the factors such as dependencies between the tasks, non-working days like holidays and weekends. Tasks automatically recalculate the scheduling date when its predecessor task has been affected. But still we can schedule the tasks and reset the start and end dates by manual editing. Summary tasks will also be automatically scheduled, but its start date, end date and duration values cannot be edited manually.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.TaskSchedulingMode(GanttTaskSchedulingMode.Auto)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Note: Automatic scheduling mode is the default task scheduling mode in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.

Manually Scheduled Tasks

When the TaskSchedulingMode property is set as Manual scheduling mode, all the tasks in the project will be rendered as manually scheduled tasks. Thus the dates of the tasks will not get validated automatically by the system. The tasks will not get rescheduled and dates will not be recalculated automatically based on the factors such as task dependencies and non-working days and hence manual scheduled tasks will lie on weekends and holidays. We can restrict this mode in predecessor editing alone, that is we can make to automatically validate the dates of the manual tasks on predecessor editing by enabling the property ValidateManualTasksOnLinking. By enabling this property, the dates of the manual tasks will recalculate automatically, when its predecessor task's dates have been changed.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.TaskSchedulingMode(GanttTaskSchedulingMode.Manual)
.ValidateManualTasksOnLinking(false)
)
@(Html.EJ().ScriptManager())
```

Custom

When the TaskSchedulingMode property is set as Custom, the scheduling mode for each tasks will be mapped form the data source field. The property TaskSchedulingModeMapping is used to map the scheduling mode field from the data source.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.TaskSchedulingMode(GanttTaskSchedulingMode.Manual)
.ValidateManualTasksOnLinking(false)
)
@(Html.EJ().ScriptManager())
```

C#

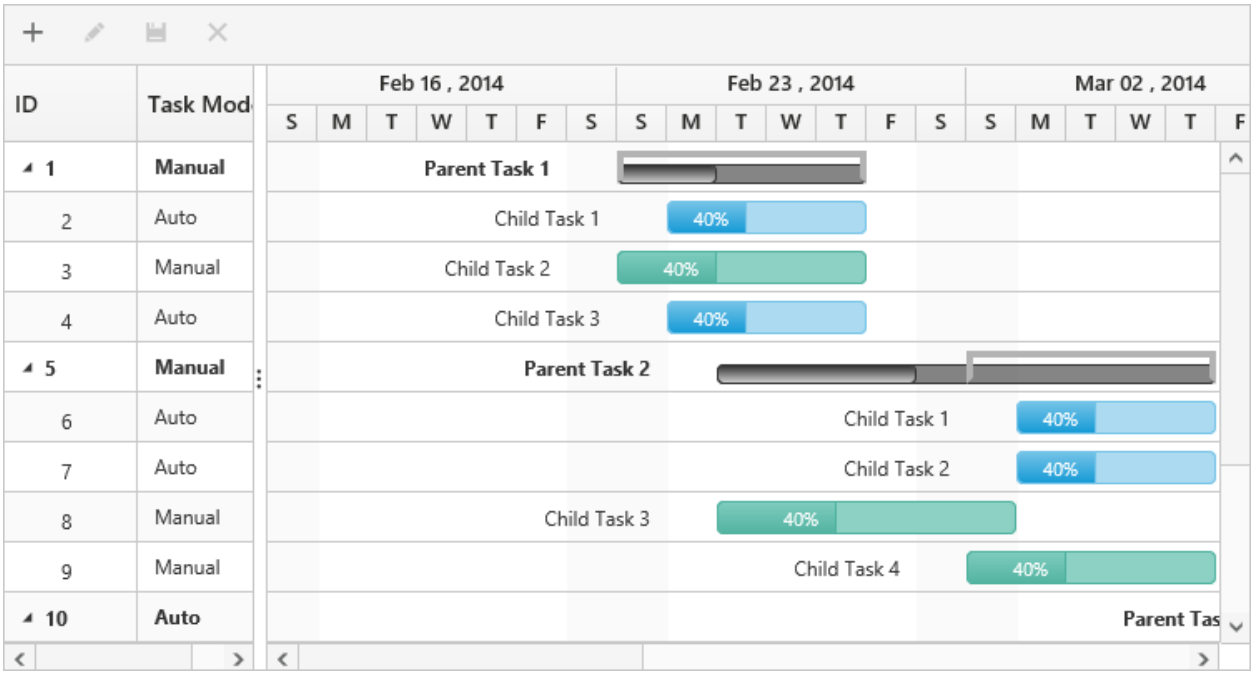
```
public class GanttController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = GanttDefaultData.GetData();
ViewBag.datasource = DataSource;
return View();
}
public class GanttDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
list.Add(new DefaultData()
{
Id = 1,
Name = "Design",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
StartDate = "02/10/2014",
EndDate = "02/14/2014",
isManual = true,
Children = (new List<DefaultData>()
{
new DefaultData()
{
Id = 2,
Name = "Software Specification",
StartDate = "02/10/2014",
EndDate = "02/12/2014",
},
new DefaultData()
{
Id = 3,
Name = "Develop Prototype",
StartDate = "02/10/2014",
EndDate = "02/12/2014",
isManual = true,
},
//..
})
});
return list;
}
}
}
```

Click here to view the online demo sample for task scheduling modes.

The following screen shot depicts a project with custom scheduling mode, where both automatic and manual scheduling modes are mapped to the tasks.



Unscheduled Tasks

Unscheduled tasks are planned for a project but do not have definite schedule dates. Now, the Gantt control supports rendering the unscheduled tasks. You can create or update the tasks with anyone of start date, end date and duration values or none. You can enable or disable the unscheduled tasks by using the AllowUnscheduledTask property.

Unscheduled Task Types

Unscheduled tasks have various task types with only either start date, end date or duration.

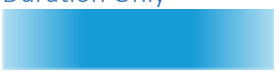
Start Date Only



End Date Only



Duration Only



Milestone

The milestone task, one without a start and end date, but having a duration value of zero is represented as follows.



Define unscheduled tasks in data source

You can define the various types of unscheduled tasks in the data source as follows.

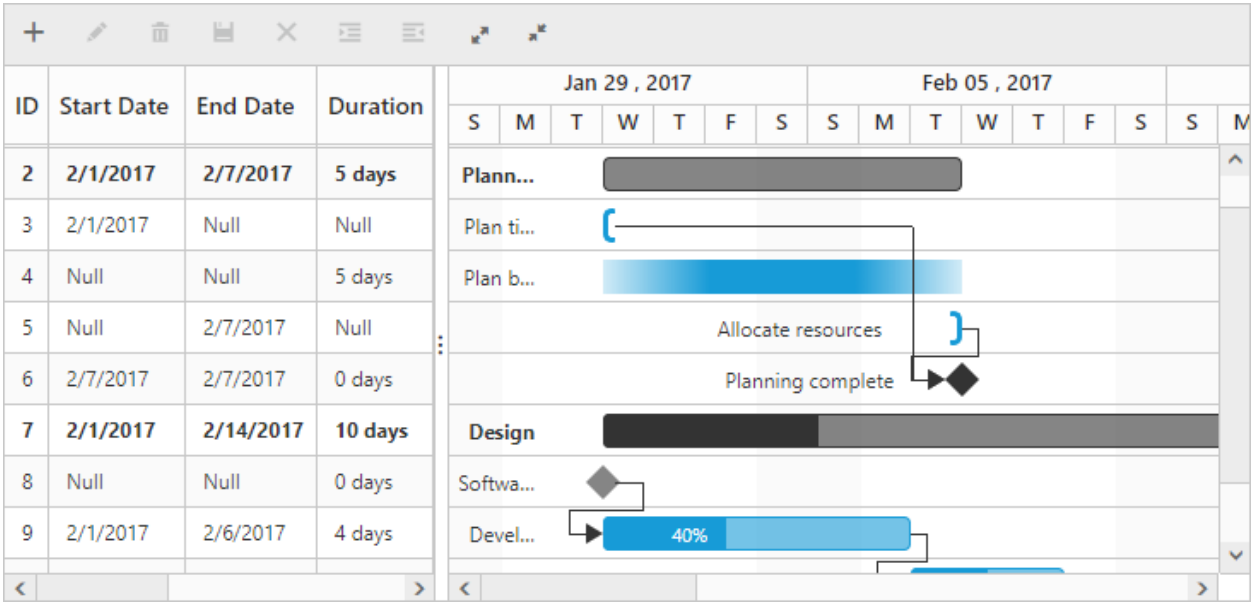
ASPX-CS

```
public List<GanttTaskDetails> GetUnscheduledData()
{
List<GanttTaskDetails> tasks = new List<GanttTaskDetails>();
// Start Date only
tasks[0].SubTasks[0].SubTasks.Add(new GanttTaskDetails()
{
TaskID = 3,
TaskName = "Plan timeline",
StartDate = "02/01/2017",
Duration = null,
EndDate = null,
Progress = "100",
ResourceID = new List<object>() { 1 }
});
//Duration only
tasks[0].SubTasks[0].SubTasks.Add(new GanttTaskDetails()
{
TaskID = 4,
TaskName = "Plan budget",
StartDate = null,
EndDate = null,
Duration = 5,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Progress = "100",
ResourceID = new List<object>() { 5 }
});
//End date only
tasks[0].SubTasks[0].SubTasks.Add(new GanttTaskDetails()
{
TaskID = 5,
TaskName = "Allocate resources",
StartDate = null,
EndDate = "02/7/2017",
Duration = null,
Progress = "100",
ResourceID = new List<object>() { 6 }
});
// Milestone
tasks[0].SubTasks[1].SubTasks.Add(new GanttTaskDetails()
{
TaskID = 8,
TaskName = "Software Specification",
StartDate = null,
EndDate = null,
Duration = 0,
Progress = "60",
ResourceID = new List<object>() { 2 }
});
//...
return tasks;
}
@(Html.EJ().Gantt("GanttContainer")
//... .
.AllowUnscheduledTask(true)
)
@(Html.EJ().ScriptManager())
```

The following screenshot displays the output of the above code.



Copyright © 2001 - 2020 Syncfusion Inc.

Note: If the AllowUnscheduledTask property is set to false for unplanned task, the Gantt control will automatically calculate the scheduled dates with a default value of duration 1 and the project scheduled start date is considered as the start date for the task.

Show/hide null text in Gantt columns

You can show/hide the null text cell value for start date, end date and duration columns by using the showNullText column property. This can be customized in load event of Gantt. You can change the Null string by using the nullText property in the localization text file of the Gantt control. The following code snippets explain this behavior.

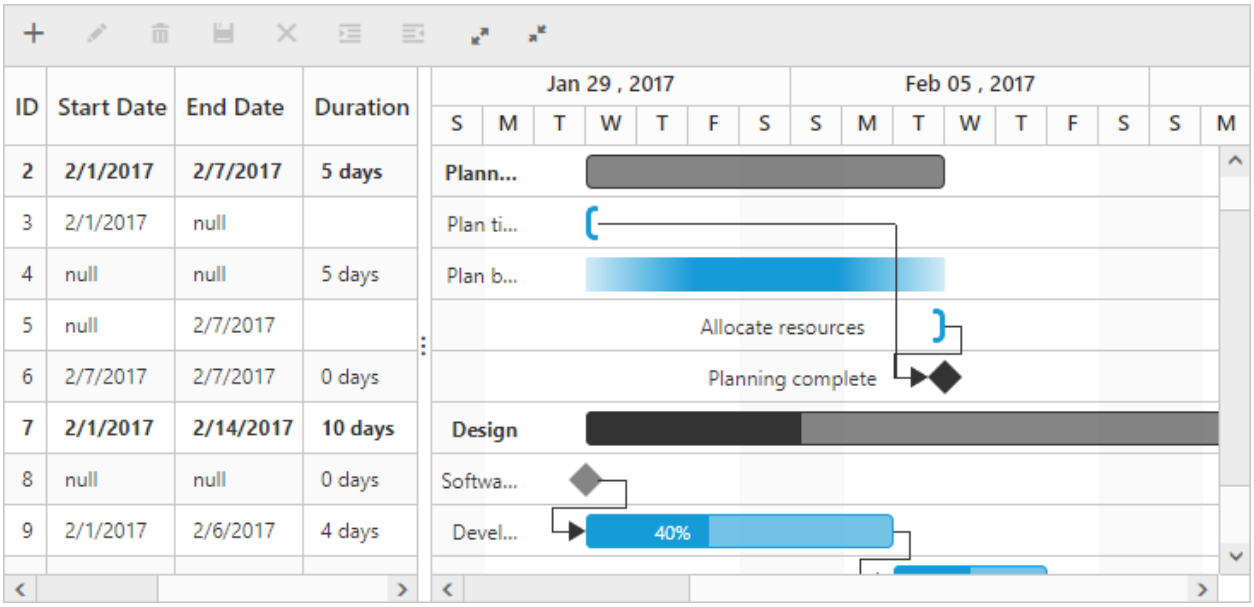
ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//... .
.AllowUnscheduledTask(true)
.ClientSideEvents(eve =>{ eve.Load("loadEvent"); })  // Client-side event
triggered at load time.
)
@(Html.EJ().ScriptManager())
```

JAVASCRIPT

```
function loadEvent(args) {
//…
var columns = this.getColumns();
columns[1].showNullText = true;
columns[2].showNullText = true;
column[3].showNullText = false;
}
ej.Gantt.Locale["en-Us"] = {
nullText: "null"
}
```

The following screenshot displays the output of the above code.



Copyright © 2001 - 2020 Syncfusion Inc.

Context Menu

Default Menu Items

Context menu in Gantt has the following default menu items.

- Task Details
- Add New Task
- Indent
- Outdent
- Delete

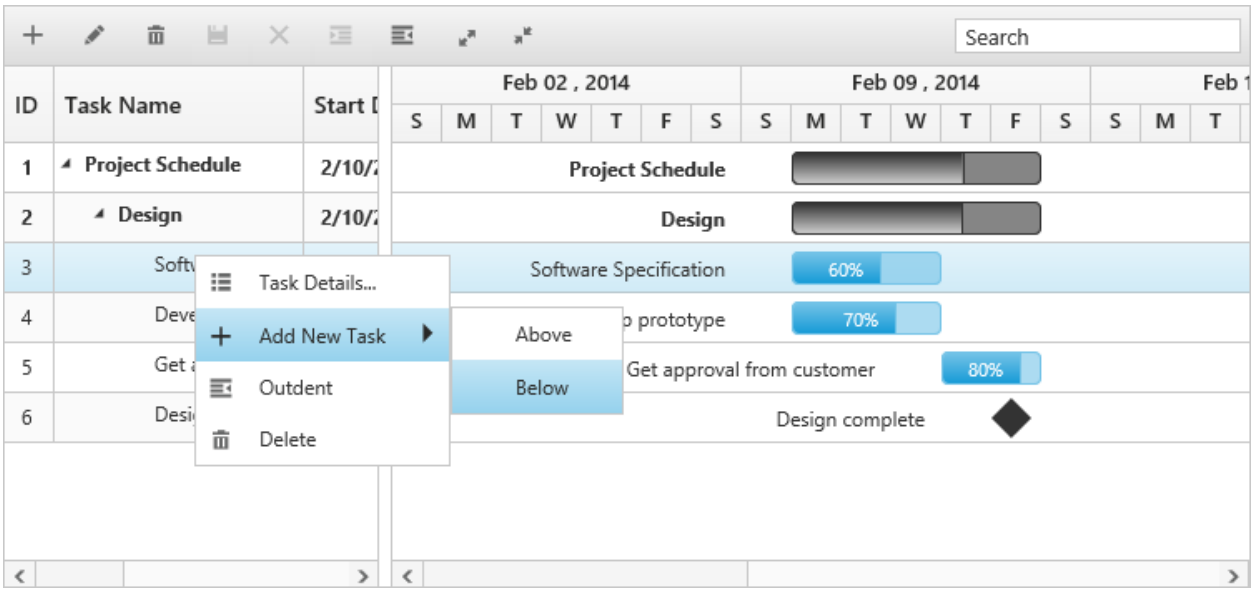
Context menu in Gantt can be enabled by using `EnableContextMenu` property.

The following code example explains how to enable the context menu in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//.
.EnableContextMenu(true)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the default context menu in Gantt control.



Custom Menu Item

It is possible to add a custom context menu item in Gantt control by using `ContextMenuOpen` event. The following code example explains on how to add the custom context menu item.

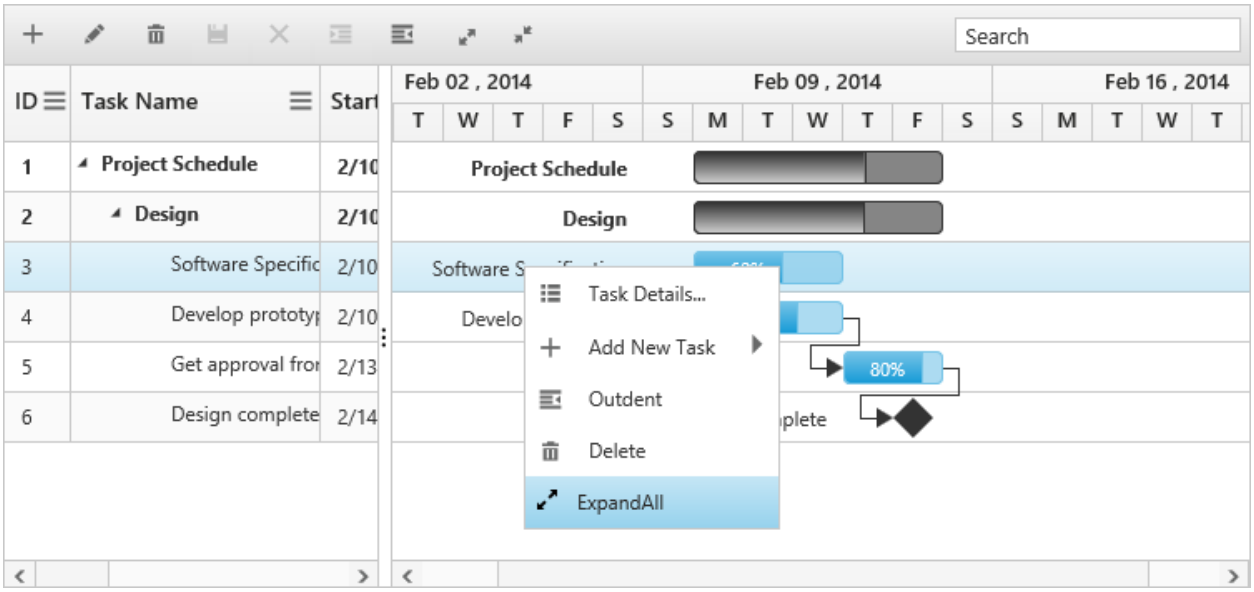
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.EnableContextMenu(true)
.ClientSideEvents(eve=>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
eve.ContextMenuOpen("contextMenuOpen");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function contextMenuOpen(args) {
args.contextMenuItems.push({
headerText: "Expand/Collapse",
menuId: "expand",
iconPath: "url(Expand-02-WF.png)",
eventHandler: function () {
//event handler for custom menu items
}
})
}
</script>
```

The screenshot of the custom context menu items in Gantt control is as follows.



Custom menu item with sub menu item

It is possible to create a custom menu item with a sub menu, by mapping the parentMenuId property from the contextMenuItems argument in the ContextMenuOpen event.

The following code example explains on how to add sub context menu for custom menu items.

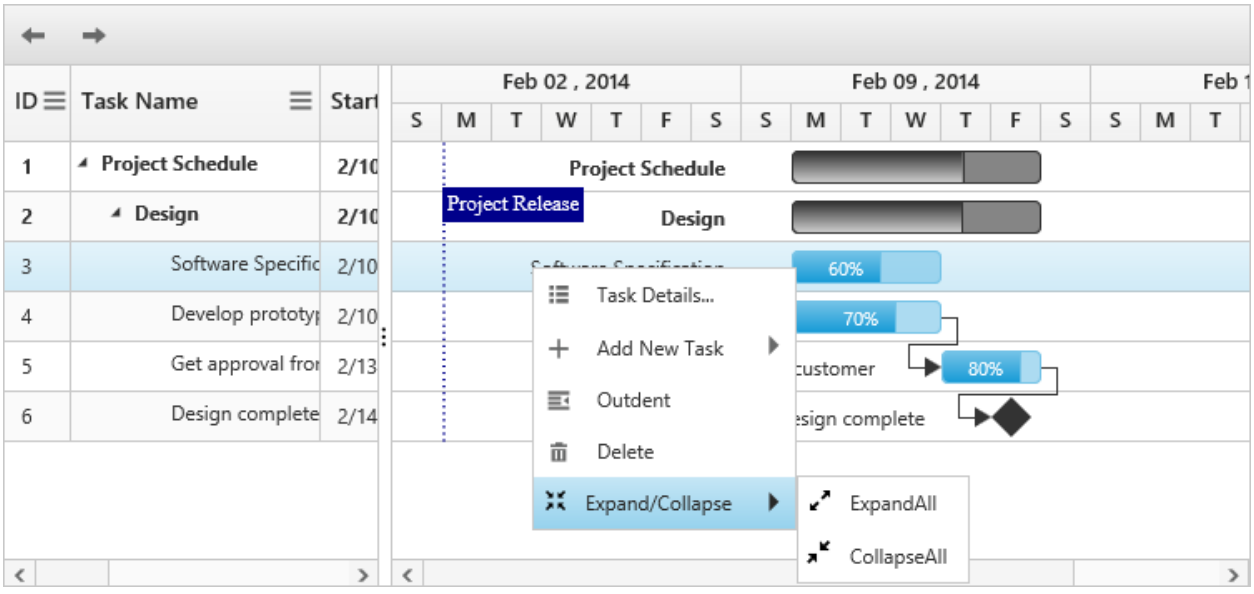
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.EnableContextMenu(true)
.ClientSideEvents(eve=>
{
eve.ContextMenuOpen("contextMenuOpen");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
args.contextMenuItems.push({
headerText: "Expand/Collapse",
menuId: "expand",
iconPath: "url(Navigation-Up-02-WF.png)",
eventHandler: function () {
//event handler for custom menu items
}
});
args.contextMenuItems.push({
headerText: "ExpandAll",
menuId: "expandAll",
parentMenuId: "expand",
iconPath: "url(Expand-02-WF.png)",
eventHandler: function () {
//event handler for custom menu items
}
});
args.contextMenuItems.push({
headerText: "CollapseAll",
menuId: "collapseAll",
parentMenuId: "expand",
iconPath: "url(shrink2.png)",
eventHandler: function () {
//event handler for custom menu items
}
});
}
</script>
```

The screenshot of the custom context menu items in Gantt control is as follows.



Click here to view the online demo sample for custom context menu in Gantt.

Customizations

Gantt provides support for the following UI customizations,

- Taskbar template
- Task label template
- Tooltip template

Taskbar template

You can design your own taskbars to view the tasks in Gantt by using TaskbarTemplate property. And it is possible to map the JsRender script or script element's ID to this property. It is also possible to customize the parent taskbars and milestones with custom templates by using ParentTaskbarTemplate and MilestoneTemplate properties.

The following code example shows how to define template for taskbars in Gantt.

ASPX-CS

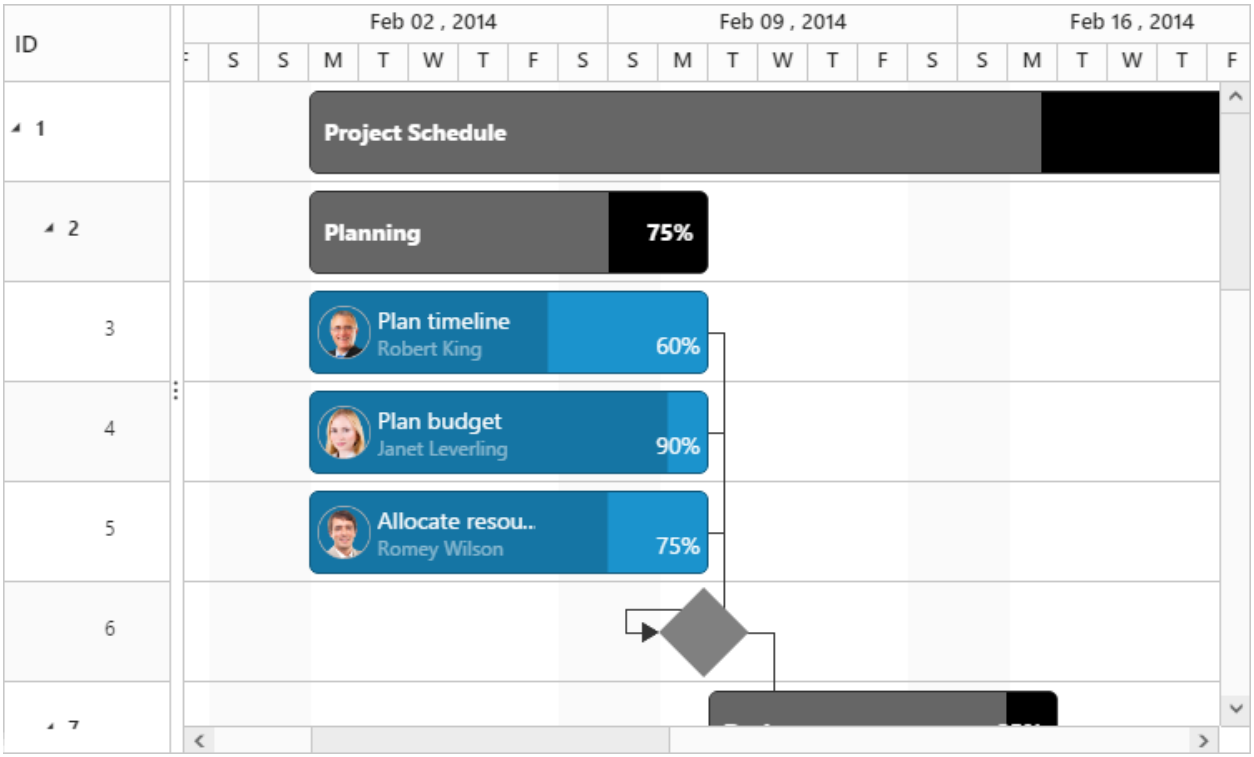
```
<script type="text/x-jsrender" id="taskbarTemplate">
<div class="e-gantt-template-taskbar bg-color">
<div>
//…
</div>
<div class="e-gantt-template-progressbar">
</div>
</div>
</script>
<script type="text/x-jsrender" id="parentTaskbarTemplate">
<div class="e-gantt-template-taskbar">
//…
<div class="e-gantt-template-progressbar">
</div>
</div>
</script>
<script type="text/x-jsrender" id="milestoneTemplate">
<div class="e-gantt-template-milestone" style="background-
color:transparent;">
<div class="e-gantt-milestone milestone-top"></div>
<div class="e-gantt-milestone milestone-bottom"></div>
</div>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.TaskbarTemplate("#taskbarTemplate")
.ParentTaskbarTemplate("#parentTaskbarTemplate")
.MilestoneTemplate("#milestoneTemplate")
)
@(Html.EJ().ScriptManager())
```

The DOM structure and class names mentioned in the above code snippet is mandatory for providing the editing support in custom templates.

Click here to view the online demo sample for Taskbar templates in Gantt.

The following screenshot shows the template for taskbars in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



Task label template

By default, task name will be displayed to the left and resource names will be displayed to the right of the taskbars as task labels. We can enable/disable this default task labels by using ShowTaskNames and ShowResourceNames properties. But these task labels are customizable.

Mapping data source fields as task labels

It is also possible to set any data source fields as task labels using RightTaskLabelMapping and LeftTaskLabelMapping properties.

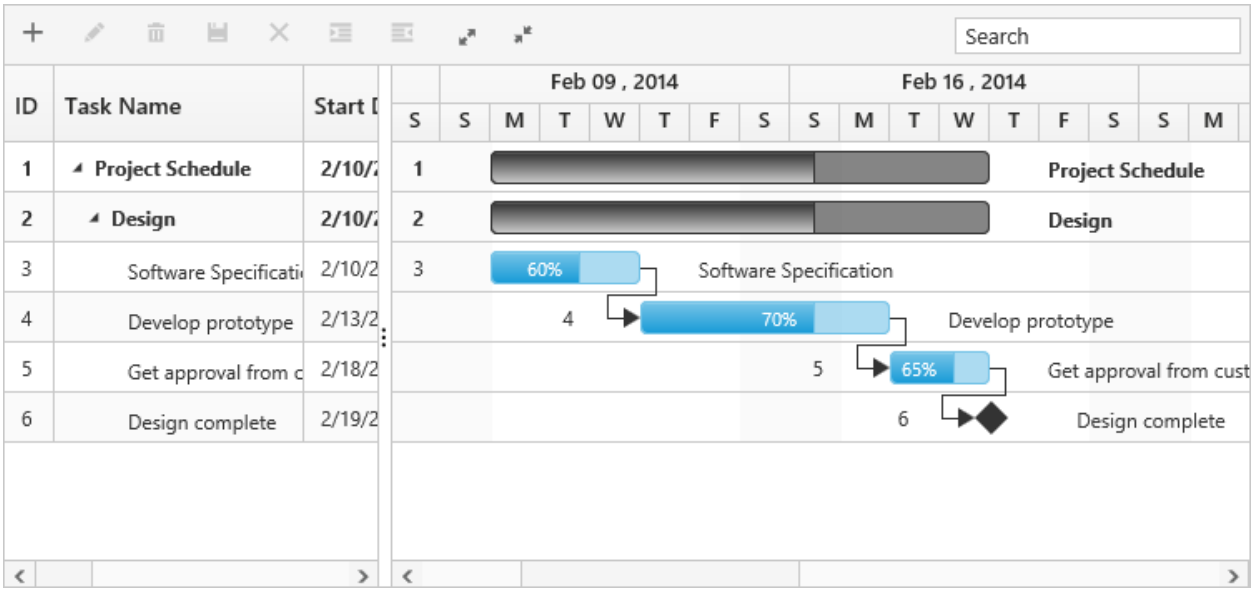
The following code example explains how to set task name field as right label and task ID field as left label,

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.LeftTaskLabelMapping("TaskName")
.RightTaskLabelMapping("TaskID")
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows Gantt with task labels mapped with different data source fields

Copyright © 2001 - 2020 Syncfusion Inc.



Task label templates

It is possible to customize the task labels with templates, by using RightTaskLabelTemplate and LeftTaskLabelTemplate properties. The following code example explains how to map custom templates to task labels.

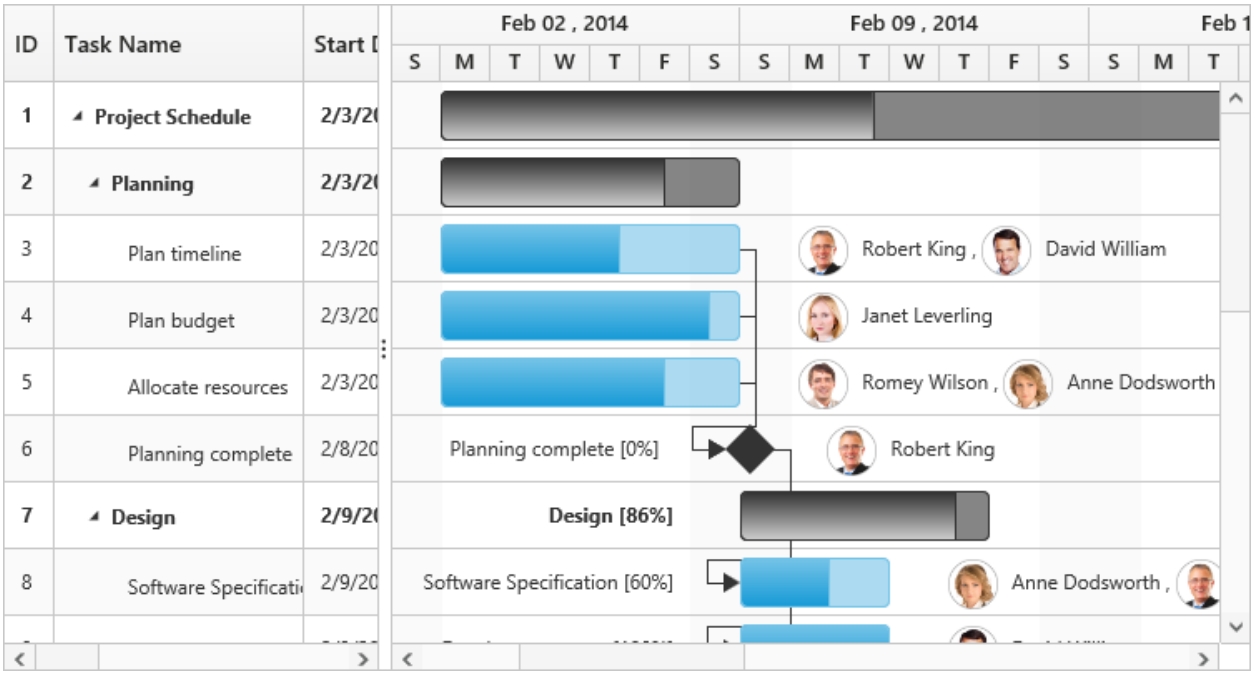
ASPX-CS

```
<script id="rightLabelTemplate" type="text/x-jsrender">
{{"{{"}}if #data['resourceNames']{{}}}}
<div>
{{"{{"}}for resourceInfo{{}}}}
<img src="themes/web/content/images/gantt/{{"{{"}}:resourceName{{}}}}.png"
height="30px" />
<span style="margin-left:5px;">{{"{{"}}:resourceName{{}}}}</span>
{{"{{"}}:~_getSeparator(#get("array").data.length,#index){{}}}}
{{"{{"}}/for{{}}}}
</div>
{{/if}}
</script>
<script id="leftLabelTemplate" type="text/x-jsrender">
<div style="padding-top:5px;">
<span>{{"{{"}}:#data['taskName']{{}}}}  [{{"{{"}}:status{{}}}}%]</span>
</div>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.RightTaskLabelTemplate("#rightLabelTemplate")
.LeftTaskLabelTemplate("#leftLabelTemplate")
)
@(Html.EJ().ScriptManager())
```

You can find the online demo sample for task label templates in Gantt here

The following screenshot shows Gantt with task label templates.

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip template

Taskbar tooltip

The default tooltip in Gantt can be customized by using the `TaskbarTooltipTemplateId` property. We need to map the JsRender script element's ID value to this property.

The following code example shows how to customize the tooltip.

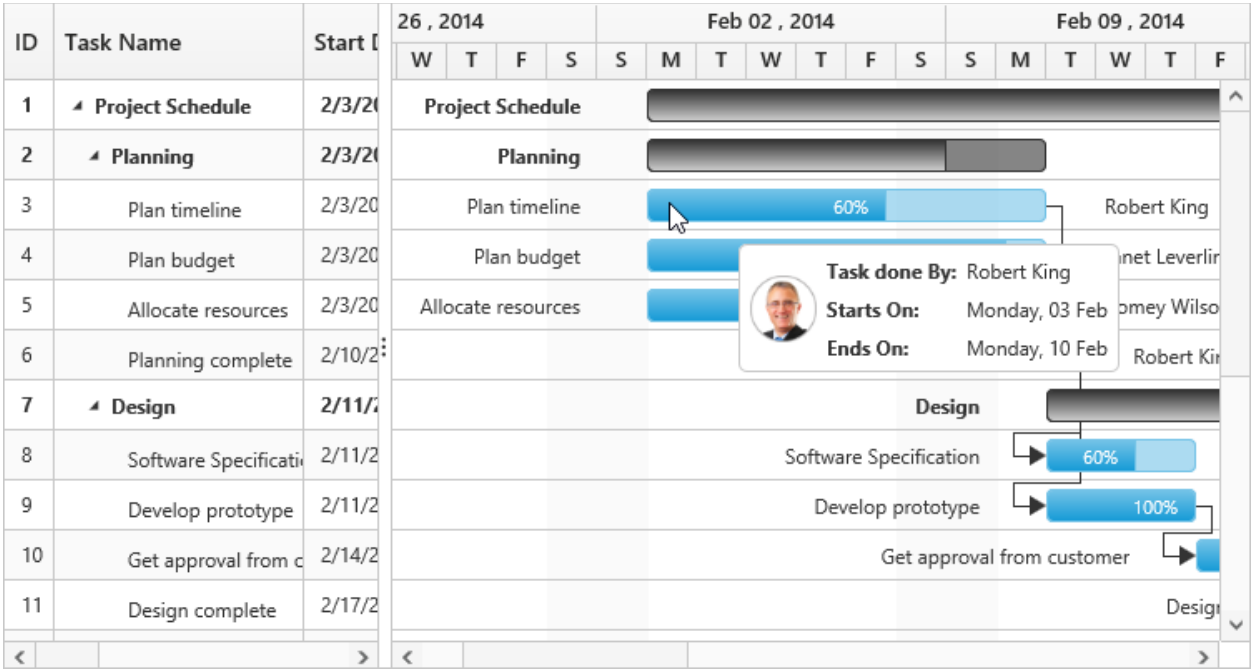
ASPX-CS

```
<script type="text/x-jsrender" id="tooltipTemplate">
<table>
{{"{{"}}if #data['resourceNames']{{}}}}
<tr>
<td rowspan="3" style="padding:3px"><img
src="themes/web/content/images/gantt/{{"{{"}}:#data['resourceNames']{{}}}}.p
ng" height="40px" /></td>
<td style="padding:3px"><b>Task done By:</b></td>
<td style="padding:3px">{{"{{"}}:#data['resourceNames']{{}}}}</td>
</tr>
{{/if{{}}}}
<tr>
<td style="padding:3px"><b>Starts On:</b></td>
<td
style="padding:3px">{{"{{"}}:~_ganttDateFormatter("startDate"){{}}}}</td>
</tr>
<tr>
<td style="padding:3px"><b>Ends On:</b></td>
<td style="padding:3px">{{"{{"}}:~_ganttDateFormatter("endDate"){{}}}}</td>
</tr>
</table>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.TaskbarTooltipTemplateId("tooltipTemplate")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows Gantt with task tooltip customization.



Dependency tooltip

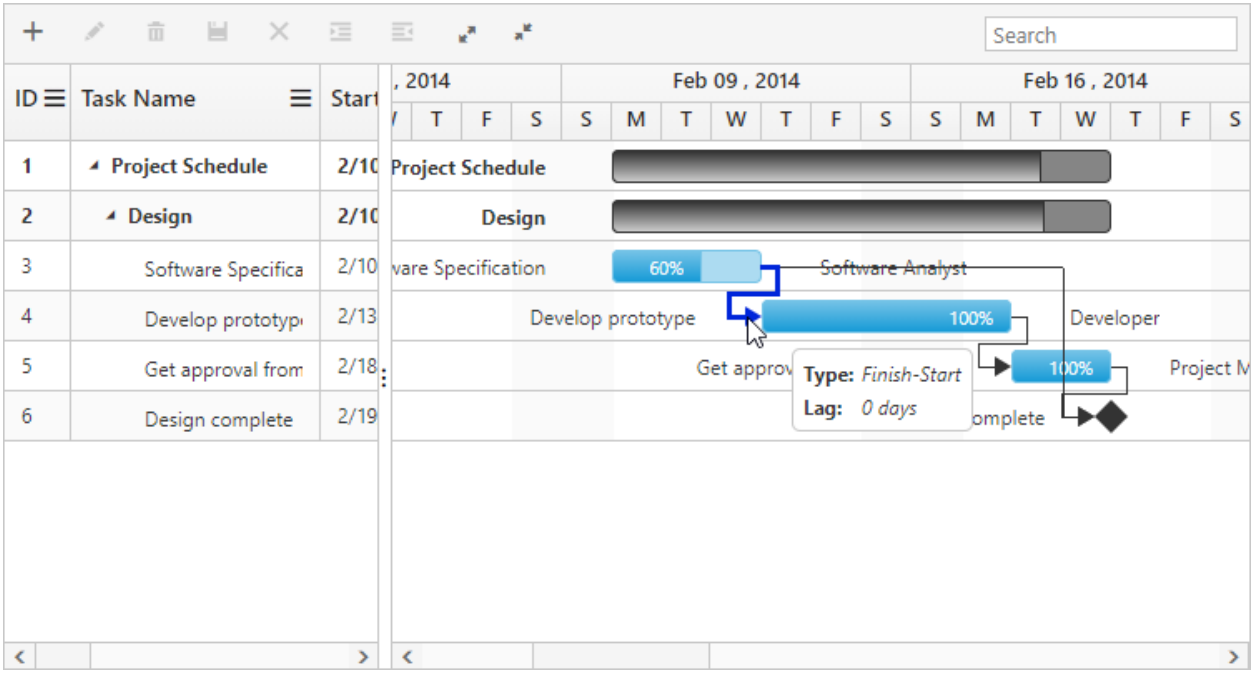
The default dependency tooltip in Gantt can be customized by using PredecessorTooltipTemplate property. We can map value to this property as JsRender template script id with prefix of '#' or HTML elements in string format. The following code example shows how to use the PredecessorTooltipTemplate property.

ASPX-CS

```
<script type="text/javascript">
$.views.helpers({
_Type: getType,
_Lag: getLag
});
function getType() {
return this.data.linkText;
}
function getLag() {
return this.data.offset + " " + this.data.offsetUnit;
}
</script>
<script id="ToolTipTemplate" type="text/x-jsrender">
<table>
<tr>
<td><b>Type:</b></td>
<td><i>{{:~_Type()}}</i></td>
</tr>
<tr>
```

```
<td><b>Lag:</b></td>
<td><i>{{:~_Lag()}}</i></td>
</tr>
</table>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.EnableTaskbarTooltip(true)
.PredecessorTooltipTemplate("#ToolTipTemplate")
)
@(Html.EJ().ScriptManager())
```

The following screenshot show the output of above code example.



Cell tooltip

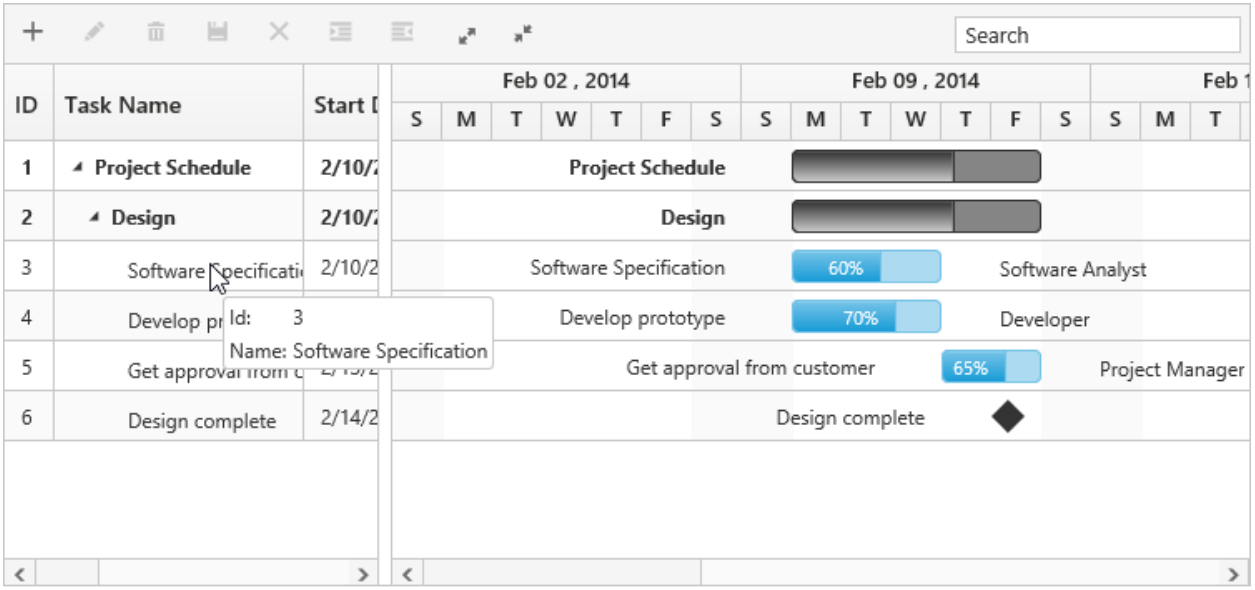
TreeGrid part tooltip can also be customized using CellTooltipTemplate property. We need to map the script element or element id to this property. The following code explains how to customize the cell tooltip in Gantt.

ASPX-CS

```
<script type="text/javascript">
$.views.helpers({
_TaskID: getTaskID,
_TaskName: getTaskName
});
function getTaskID() {
return this.data.record["taskId"];
}
function getTaskName() {
return this.data.record["taskName"];
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script id="CustomToolTip" type="text/x-jsrender">
<table>
<tr>
<td>Id:</td>
<td>{{"{{"}}:~_TaskID(){{}}}}</td>
</tr>
<tr>
<td>Name:</td>
<td>{{"{{"}}:~_TaskName(){{}}}}</td>
</tr>
</table>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.ShowGridCellTooltip(true)
.CellTooltipTemplate("#CustomToolTip")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```



You can find the online demo sample for tooltip templates for taskbars here

Taskbar Editing Tooltip

Editing tooltip is used to show the updated start date, end date, duration and progress values of a task while resizing, dragging and progress bar resizing actions. Currently two editing tooltips are available in Gantt.

- Taskbar editing tooltip
- Progress bar editing tooltip

We can customize the default taskbar editing tooltip and progress bar editing tooltip in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.

Customize taskbar editing tooltip

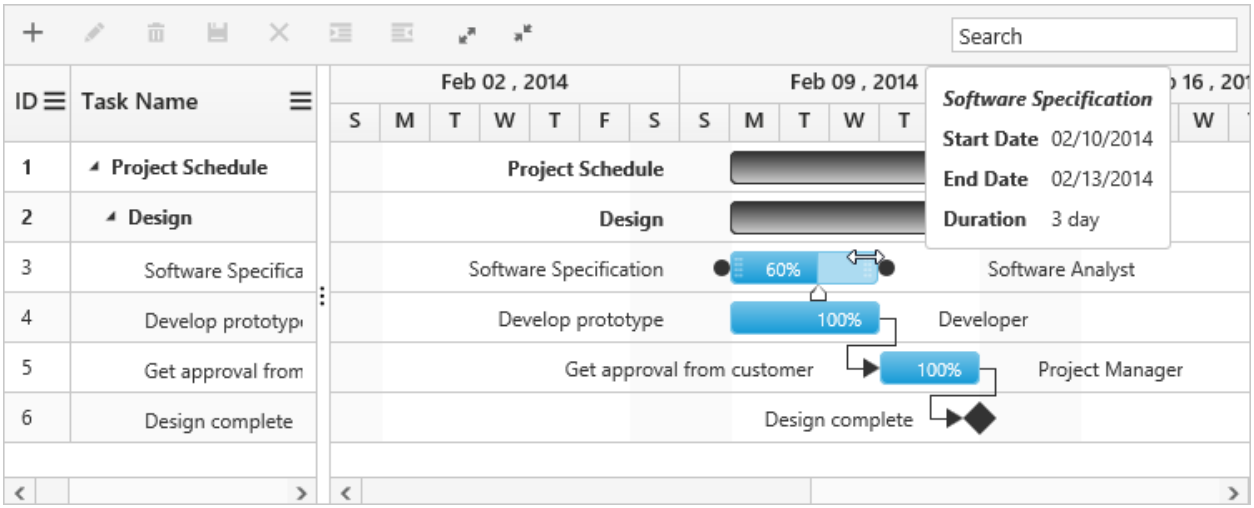
Taskbar editing tooltip can be customized by using TaskbarEditingTooltipTemplate and TaskbarEditingTooltipTemplateId properties. The below code example shows how to customize the taskbar editing tooltip in Gantt.

ASPX-CS

```
<script type="text/javascript">
$.views.helpers({
getStartDate: function () {
return ej.format(this.data.startDate, "MM/dd/yyyy", "en-US");
},
getEndDate: function () {
return ej.format(this.data.endDate, "MM/dd/yyyy", "en-US");
}
});
</script>
<script id="taskbar_editing_tooltip_template" type="text/x-jsrender">
<table>
<tr>
<td colspan="2" style="padding:3px;font-weight:bold;font-
style:italic">{{:taskName}}</td>
</tr>
<tr>
<td style="padding:3px;font-weight:bold">Start Date</td>
<td style="padding:3px">{{:~getStartDate(#data)}}</td>
</tr>
<tr>
<td style="padding:3px;font-weight:bold">End Date</td>
<td style="padding:3px">{{:~getEndDate(#data)}}</td>
</tr>
<tr>
<td style="padding:3px;font-weight:bold">Duration</td>
<td style="padding:3px">{{:duration}} {{:durationUnit}}</td>
</tr>
</table>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.TaskbarEditingTooltipTemplateId("taskbar_editing_tooltip_template")
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Customize progress bar editing tooltip

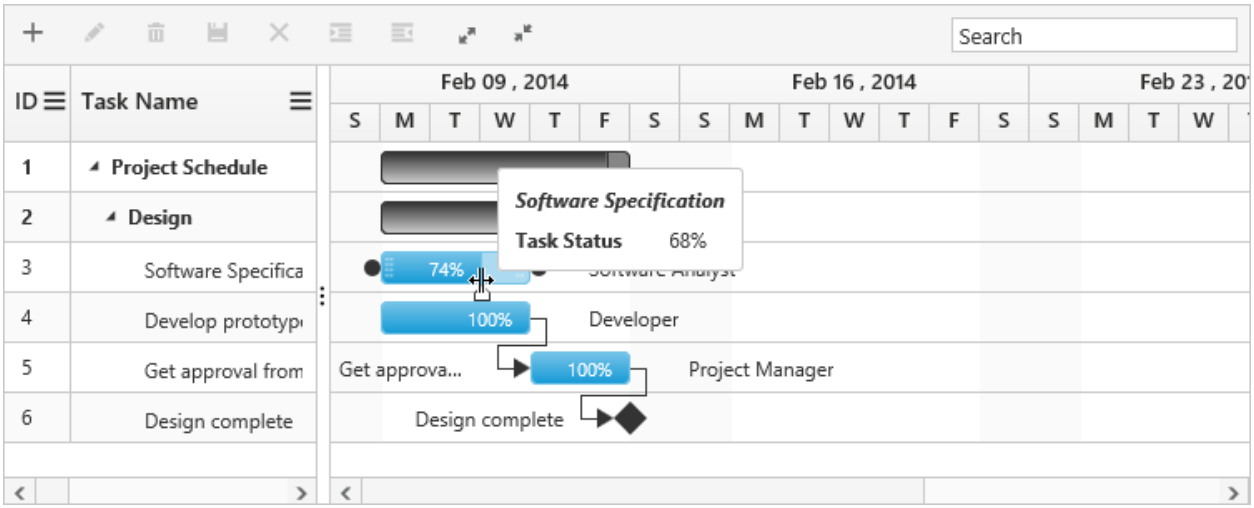
Progress bar editing tooltip can be customized by using `ProgressbarTooltipTemplate` and `ProgressbarTooltipTemplateId` properties. The below code example shows how to customize the progress bar editing tooltip in Gantt.

ASPX-CS

```
<script id="progressbar_editing_tooltip_template" type="text/x-jsrender">
<table>
<tr>
<td colspan="2" style="padding:3px;font-weight:bold;font-
style:italic">{{:taskName}}</td>
</tr>
<tr>
<td style="padding:3px;font-weight:bold">Task Status</td>
<td style="padding:3px">{{:status}}%</td>
</tr>
</table>
</script>
@(Html.EJ().Gantt("Gantt")
//...
.ProgressbarTooltipTemplateId("progressbar_editing_tooltip_template")
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Selection

Row selection

The row selection in Gantt can be enabled or disabled, by using the `AllowSelection` property. You can able to get the selected row object using the `selectedItem` property from the Gantt model. The following code example shows how to disable the row selection in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowSelection(true)
)
@(Html.EJ().ScriptManager())
```

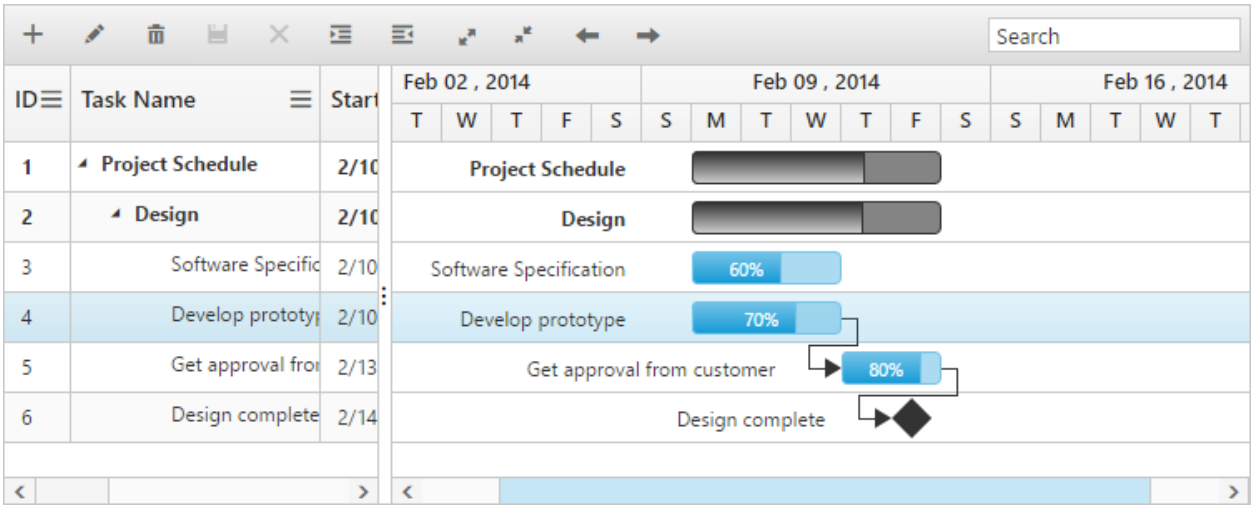
Selecting a row on initial load

You can select a row on load time by setting the index of the row to `SelectedRowIndex` property. Find the following code example for details.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.SelectedRowIndex(3)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Selecting a row programmatically

You can also select a row programmatically by setting index of the row value to SelectedRowIndex property. The following code shows to select a row programmatically with a custom button click action,

ASPX-CS

```
<button onclick="selectRow()">SelectRow</button>
@(Html.EJ().Gantt("Gantt")
//...
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function selectRow() {
$("#Gantt ").ejGantt("option", "selectedRowIndex", 4);
}
</script>
```

Multiple row selection

It is also possible to select multiple rows by setting SelectionType as Multiple. You can select more than one row by holding down CTRL key while selecting multiple rows.

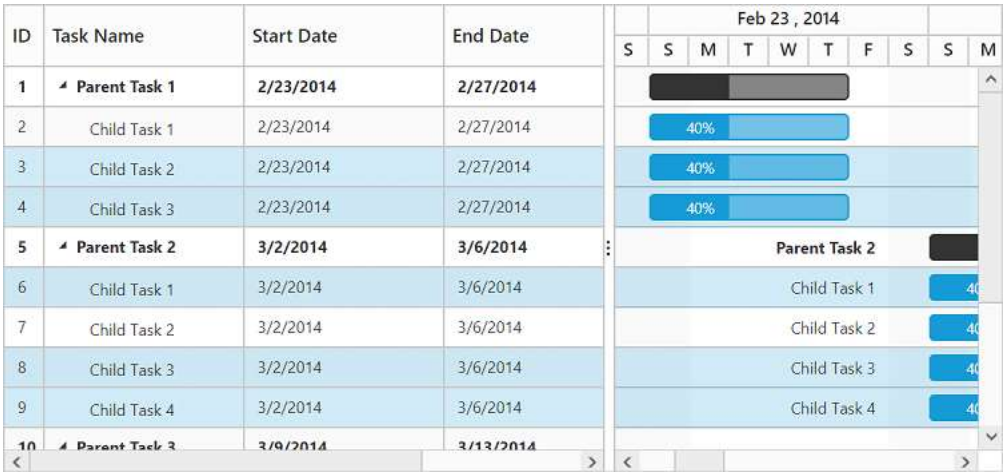
The following code example explains how to enable multiple selection in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowSelection(true)
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Row)
)
@(Html.EJ().ScriptManager())
```

The output of the Gantt with multiple row selection is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Selecting multiple rows programmatically

You can also select multiple rows programmatically by using selectMultipleRows public method. The following code example explains how to enable multiple selection in Gantt.

ASPX-CS

```
<button onclick="selectMultipleRow()">selectMultipleRow</button>
@(Html.EJ().Gantt("Gantt")
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Row)
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function selectMultipleRow() {
var ganttObj = $("#Gantt").data("ejGantt"),
multipleRowIndex = [1,0,5,7];
ganttObj.selectMultipleRows(multipleRowIndex);
}
<script>
```

Customize row selection action

While selecting a row in Gantt, RowSelecting and RowSelected event will be triggered. Row selecting event will be triggered on initialization of row selection action. In RowSelecting event we can get the previously selected row and current selecting row's information, using this information we can prevent selection of particular row. The RowSelected event will be triggered on completion of row selection action, in this event we can get the current selected row's information. The following code example shows how to prevent the selection of particular row using RowSelecting event.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//
.AllowSelection(true)
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Row)
.ClientSideEvents(eve =>
{
eve.RowSelecting("rowSelecting");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function rowSelecting(args) {
if (args.data.taskId == 5) // prevent selection of Task id 5
args.cancel = true;
}
</script>
```

Cell selection

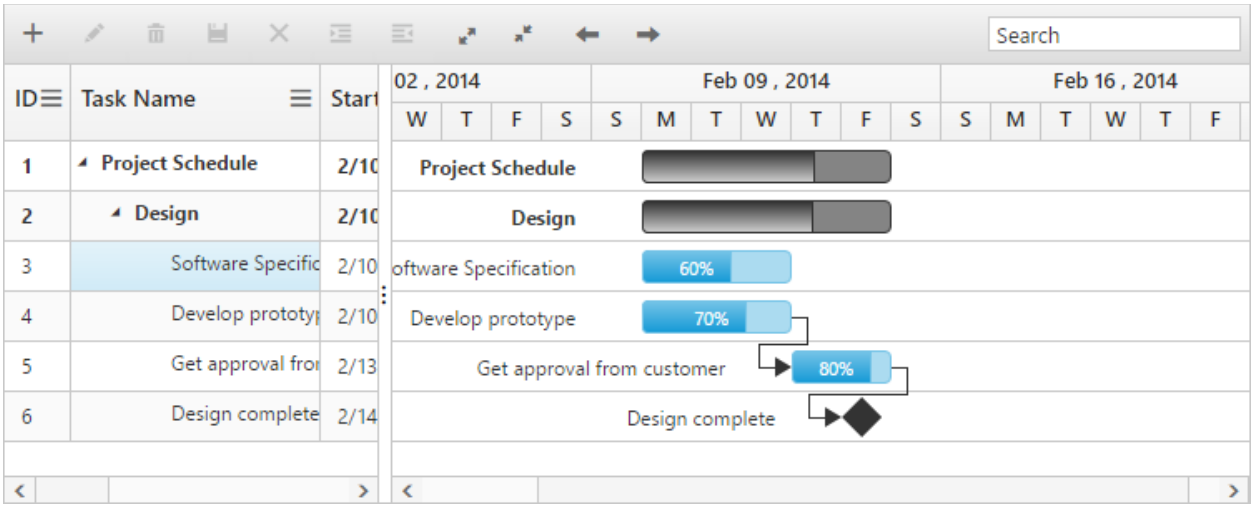
You can select a cell in Gantt by setting SelectionMode property as Cell. And you can able to get the selected cell information using selectedCellIndexes property from the Gantt object. selectedCellIndexes is an object collection, which has the cell index and row index information of the selected cells.

Find the code example below to enable the cell selection in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//
.SelectionMode(GanttSelectionMode.Cell)
)
@(Html.EJ().ScriptManager())
```

The following screen shots shows you cell selection.



Selecting multiple cells

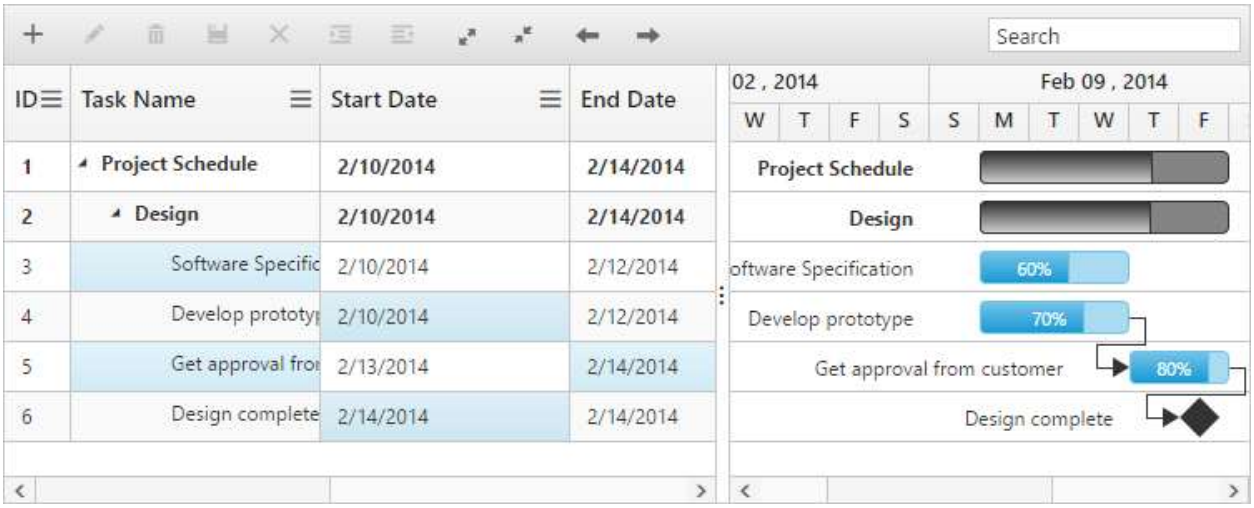
You can also select multiple cells by setting SelectionType property as Multiple while SelectionMode property is set to cell. Multiple cells can be selected by holding the ctrl key and to click on the cells. The following code example shows you to select multiple cells.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Cell)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
```



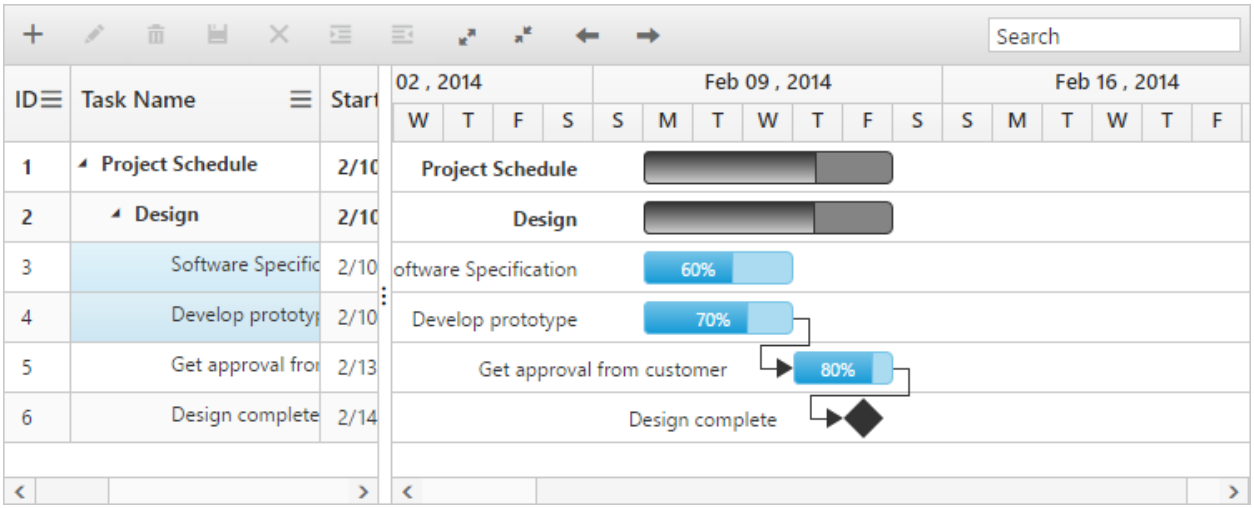
Select cells programmatically

You can select the cells programmatically using selectCells public method. Find the code example below for details.

ASPX-CS

```
<button onclick="selectCells()">selectCells</button>
@(Html.EJ().Gantt("Gantt")
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Cell)
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function selectCells() {
var ganttObj = $("#GanttContainer").data("ejGantt");
cellIndex = [{
rowIndex: 2,
cellIndex: 1
}, {
rowIndex: 3,
cellIndex: 1
}];
ganttObj.selectCells(cellIndex);
}
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customize cell selection action

While selecting a cell in Gantt, CellSelecting and CellSelected event will be triggered.

Cell selecting event will be triggered on initialization of cell selection action.

In CellSelecting event we can get the current selecting cell information, using this information we can prevent selection of particular cell in particular row.

The CellSelected event will be triggered on completion of cell selection action, in this event we can get the current selected cell's information. The following code example shows how to prevent the selection of particular cell using CellSelecting event.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//
.AllowSelection(true)
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Row)
.ClientSideEvents(eve =>
{
eve.CellSelecting("cellSelecting");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function cellSelecting(args) {
if (args.data.taskId == 5 && args.cellIndex == 1) // prevent selection of
Task Name cell of Task id 5
args.cancel = true;
</script>
```

MultiSelection – Touch Option

It is possible to select rows using touch action in Gantt. Gantt provides support for both single selection and multiple row selection using touch action. For multiple row selection, when we tap on a cell, a helper icon will be displayed using which we can select multiple rows.

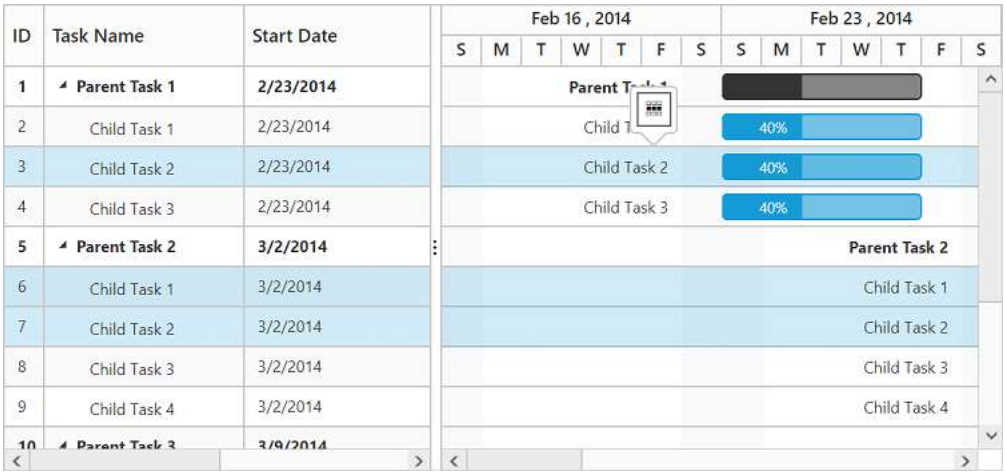
The following code example describes how to enable multiple selection in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.SelectionType(GanttSelectionType.Multiple)
.SelectionMode(GanttSelectionMode.Row)
)
@(Html.EJ().ScriptManager())
```

The following output is displayed the result of multiple selection in touch device environment.



Export

Gantt provides support to export the contents in PDF and excel.

To export the contents, the ExcelExport and PdfExport toolbar items must be included in the ToolbarSettings.ToolbarItems property.

And you need to call the export method with the export mapper as parameter in the toolbar button click action. We can export multiple Gantt control in same file by using multiple exporting support, this can be enabled by setting AllowMultipleExporting property as true.

The below code snippet explains the above behavior,

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<GanttToolBarItems>()
{
GanttToolBarItems.PdfExport,
GanttToolBarItems.ExcelExport
});
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

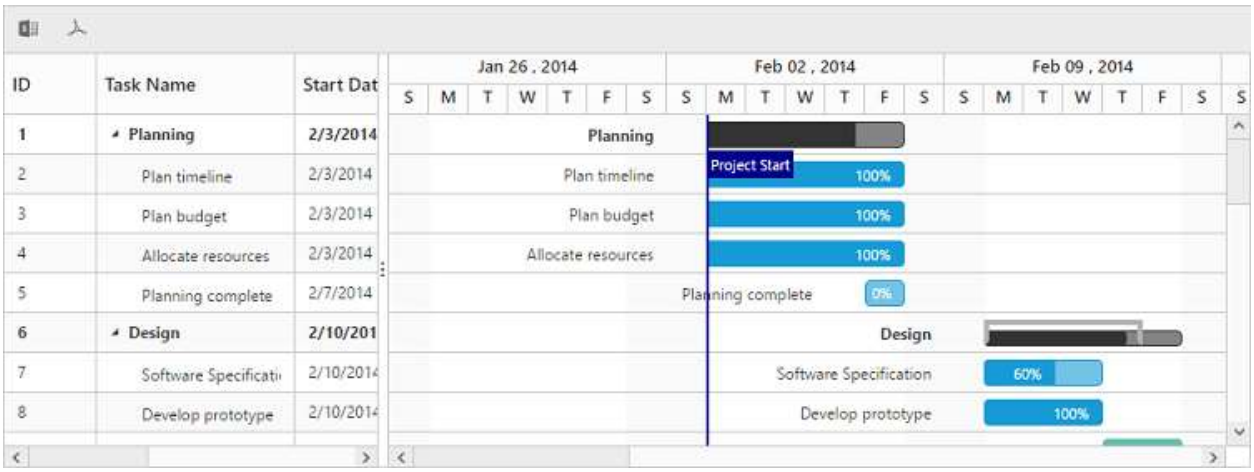
C#

```
public class GanttController : Controller
{
public ActionResult GanttExporting()
{
ViewBag.datasource = this.GetDataSource();
return View();
}
public void ExportToPdf(string GanttModel)
{
PdfExport exp = new PdfExport();
var datasource = this.DataSource();
GanttProperties obj = ConvertGanttObject(GanttModel);
GanttPdfExportSettings settings = new GanttPdfExportSettings();
settings.Theme = GanttExportTheme.FlatSaffron;
settings.Locale = Request.Form["locale"];
exp.Export(obj, datasource, settings, "Gantt");
}
public void ExportToExcel(string GanttModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = this.GetEditingDataSource();
GanttProperties obj = ConvertGanttObject(GanttModel);
exp.Export(obj, DataSource, "GanttExport.xlsx", ExcelVersion.Excel2010, new
GanttExportSettings() { Theme = ExportTheme.FlatSaffron });
}
private GanttProperties ConvertGanttObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GanttProperties gridProp = new GanttProperties();
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(ds.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
return gridProp;
}
}
```

The PDF and Excel exporting services for Gantt are explained in detail here.

The below screen shot shows Gantt with excel and PDF exporting enabled.

Copyright © 2001 - 2020 Syncfusion Inc.



Server dependencies

Export Helper functions are available in the Assembly Syncfusion.EJ.Export, which is available in the Essential Studio & Essential ASP.NET MVC builds. The list of assemblies needed for Gantt Export as follows

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.Linq.Base
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base
- Syncfusion.PDF.Base

Supported Export Types

Currently server helper function allows following two types of exporting.

- Excel
- PDF

Export Mapper in MVC

Mappers is used to change the default routing path for exporting. By using Mappers you can change any action name in controller and the action can be in any controller (Need not to be in Gantt View Page Controller).

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.Mappers(map =>
map.ExportToPdfAction("PdfAction").ExportToExcelAction("ExcelAction"))
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<GanttToolBarItems>()
{
GanttToolBarItems.PdfExport,
GanttToolBarItems.ExcelExport
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
//...
)
@(Html.EJ().ScriptManager())
```

C#

```
public partial class GanttController : Controller
{
public void PdfAction(string GanttModel)
{
PdfExport exp = new PdfExport();
var DataSource = this.GetEditingDataSource();
GanttProperties obj = ConvertGanttObject(GanttModel);
GanttPdfExportSettings settings = new GanttPdfExportSettings();
settings.Theme = GanttExportTheme.FlatSaffron;
settings.Locale = sRequest.Form["locale"];
exp.Export(obj, DataSource, settings, "Gantt");
}
public void ExcelAction(string GanttModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = this.GetEditingDataSource();
GanttProperties obj = ConvertGanttObject(GanttModel);
exp.Export(obj, DataSource, "GanttExport.xlsx", ExcelVersion.Excel2010, new
GanttExportSettings() { Theme = ExportTheme.FlatSaffron });
}
}
```

Multiple exporting

Multiple export is used for export more than one Gantt object in the same file. Multiple export in Gantt can be enabled by setting AllowMultipleExporting property to true. Gantt properties of all the Gantt which are available in current page are passed as string array parameter to controller action method.

The following code example describes exporting multiple Gantt in PDF format

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.AllowMultipleExporting(true)
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<GanttToolBarItems>()
{
GanttToolBarItems.PdfExport
});
})
.Mappers(map => map.ExportToPdfAction("MultipleExportToPDF"))
.Datasource(ViewBag.datasource)
)
@(Html.EJ().Gantt("GanttContainer1")
//...
.Datasource(ViewBag.datasource1)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
@(Html.EJ().Gantt("GanttContainer2")
//...
.Datasource(ViewBag.datasource2)
)
@(Html.EJ().ScriptManager())
```

C#

```
public partial class GanttController : Controller
{
public void MultipleExportToPDF(string[] GanttModel)
{
PdfExport exp = new PdfExport();
var PlanData = this.GetPlanDataSource();
var DesignData = this.GetDesignDataSource();
var ImplementationData = this.GetImplementationDataSource();
PdfDocument document = null;
GanttPdfExportSettings settings = new GanttPdfExportSettings();
settings.Theme = GanttExportTheme.FlatSaffron;
settings.Locale = Request.Form["locale"];
int controlCount = 1;
foreach (string ganttProperty in GanttModel)
{
GanttProperties ganttProp = this.ConvertGanttObject(ganttProperty);
if (controlCount == 1)
{
document = exp.Export(ganttProp, (IEnumerable)PlanData, settings, false);
}
else if (controlCount == 2)
{
document = exp.Export(ganttProp, (IEnumerable)DesignData, settings,
document, false);
}
else
{
exp.Export(ganttProp, (IEnumerable)ImplementationData, settings, "Gantt",
document, true);
}
count++;
}
}
}
```

Click here to view the online demo sample for multiple exporting.

Export Theme

The Gantt export supports the below themes,

- flat-azure
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron

Copyright © 2001 - 2020 Syncfusion Inc.

- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap-theme

The desired theme should be passed as a parameter to the Export method and the code snippet for this as follows

C#

```
public void ExportToExcel(string GanttModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = this.GetEditingDataSource();
GanttProperties obj = ConvertGanttObject(GanttModel);
exp.Export(obj, DataSource, "GanttExport.xlsx", ExcelVersion.Excel2010, new
GanttExportSettings() { Theme = ExportTheme.FlatSaffron });
}
```

Customization

In Gantt, we can customize Grid cells, taskbars in Excel and PDF files by using exporting events. While exporting the Gantt as Excel or PDF files, events are triggered to customize the Grid cells and taskbars.

Customize Excel cell

Excel cells can be customized by using ServerExcelQueryCellInfo event, in this event we can get the details about current record, Excel cell and current column information. Using this information we can customize the background color, font color and value of Excel cell, please find the event argument details below.

Name	Description	Type
Data	Returns the current row details	GanttRecordDetails
Cell	Returns current Excel cell information	IRange
Column	Returns current column information	GanttColumn

The following code snippets shows how to bind ServerExcelQueryCellInfo event in Web API controller and how to customize Excel cell using this event.

C#

```
public partial class GanttController : Controller
{
public ActionResult GanttConditionalExporting()
{
ViewBag.dataSource = GetExportDataSource();
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
GanttProperties.GanttEJExportEventHandler queryExcelCellDelegate = new
GanttProperties.GanttEJExportEventHandler(GanttExcelQueryCell);
public void ConditionalExcel(string GanttModel)
{
ExcelExport export = new ExcelExport();
var DataSource = GetExportDataSource();
GanttProperties obj = ConvertToGanttObject(GanttModel);
`obj.ServerExcelQueryCellInfo = queryExcelCellDelegate;`
export.Export(obj, DataSource, "GanttExport.xlsx", ExcelVersion.Excel2010,
new GanttExportSettings() { Theme = ExportTheme.FlatSaffron });
}
public static void GanttExcelQueryCell(object model, object args)
{
var record = (GanttRecordDetails)((Dictionary<string, object>)args)["Data"];
var cell = (IRange)((Dictionary<string, object>)args)["Cell"];
var column = (GanttColumn)((Dictionary<string, object>)args)["Column"];
var ganttModel = (GanttProperties)model;
if (column.MappingName == ganttModel.ProgressMapping && !record.IsParentRow)
{
if (float.Parse(cell.Value) > 80)
{
cell.CellStyle.Color = new PdfColor(165, 105, 189);
cell.CellStyle.Font.Color = ExcelKnownColors.White;
}
else if (float.Parse(cell.Value) < 20)
{
cell.CellStyle.Color = new PdfColor(240, 128, 128);
cell.CellStyle.Font.Color = ExcelKnownColors.White;
}
}
}
private GanttProperties ConvertToGanttObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GanttProperties gridProp = new GanttProperties();
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(ds.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
return gridProp;
}
}
```

Click here to view the online demo sample with above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Refer this link to know more about what are the properties are available in Excel cell and it's type values.

Customize PDF cell

PDF cells in Gantt can be customized by using ServerPdfQueryCellInfo event, in this event we can get the details about current record, PDF cell and current column information. Using this information we can customize the background color and font color of PDF cells, please find the event argument details below.

Name	Description	Type
Data	Returns the current row details	GanttRecord
Cell	Returns current Excel cell information	PdfTreeGridCell
Column	Returns current column information	GanttColumn

The following code snippets shows how to bind ServerPdfQueryCellInfo event in Web API controller and how to customize PDF cell using this event.

C#

```
public partial class GanttController : Controller
{
public ActionResult GanttConditionalExporting()
{
ViewBag.datasource = GetExportDataSource();
return View();
}
GanttProperties.GanttEJExportEventHandler queryExcelCellDelegate = new
GanttProperties.GanttEJExportEventHandler(GanttExcelQueryCell);
public void ConditionalPdf(string GanttModel)
{
PdfExport exp = new PdfExport();
var DataSource = GetExportDataSource();
GanttProperties obj = ConvertToGanttObject(GanttModel);
`obj.ServerPdfQueryCellInfo = queryCellDelegate;`
GanttPdfExportSettings settings = new GanttPdfExportSettings();
settings.Theme = GanttExportTheme.FlatSaffron;
settings.Locale = Request.Form["locale"];
settings.IsFitToWidth = true;
exp.Export(obj, DataSource, settings, "Gantt");
}
public static void GanttPdfQueryCell(object model, object args)
{
var record = (GanttRecord)((Dictionary<string, object>)args)["Data"];
var cell = (PdfTreeGridCell)((Dictionary<string, object>)args)["Cell"];
var column = (GanttColumn)((Dictionary<string, object>)args)["Column"];
var ganttModel = (GanttProperties)model;
if (column.MappingName == ganttModel.ProgressMapping && !record.IsParentRow)
{
if (record.Progress > 80)
{
PdfBrush color = new PdfSolidBrush(new PdfColor(165, 105, 189));
cell.Style.BackgroundBrush = color;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
else if (record.Progress < 20)
{
PdfBrush color = new PdfSolidBrush(new PdfColor(240, 128, 128));
cell.Style.BackgroundBrush = color;
}
}
}
private GanttProperties ConvertToGanttObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GanttProperties gridProp = new GanttProperties();
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(ds.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
return gridProp;
}
}
```

Click here to view the online demo sample with above code example.

Note: Refer this link to know more about what are the properties are available in PDF cell and it's type values.

Customize PDF taskbar

PDF taskbars in Gantt can be customized by using ServerPdfQueryTaskbarInfo event, in this event we can get the details about current record and taskbar information. Using this information we can customize the background color of taskbar and progress bar, please find the event argument details below.

Name	Description	Type
Data	Returns the current row details	GanttRecord
Taskbar	Returns current Excel cell information	PdfGanttTaskbar

The following code snippets shows how to bind ServerPdfQueryTaskbarInfo event in Web API controller and how to customize PDF taskbar using this event.

C#

```
public partial class GanttController : Controller
{
```

```
public ActionResult GanttConditionalExporting()
{
ViewBag.datasource = GetExportDataSource();
return View();
}
GanttProperties.GanttEJExportEventHandler queryTaskbarDelegate = new
GanttProperties.GanttEJExportEventHandler(GanttQueryTaskbar);
public void ConditionalPdf(string GanttModel)
{
PdfExport exp = new PdfExport();
var DataSource = GetExportDataSource();
GanttProperties obj = ConvertToGanttObject(GanttModel);
`obj.ServerPdfQueryTaskbarInfo = queryTaskbarDelegate;`
GanttPdfExportSettings settings = new GanttPdfExportSettings();
settings.Theme = GanttExportTheme.FlatSaffron;
settings.Locale = Request.Form["locale"];
settings.IsFitToWidth = true;
exp.Export(obj, DataSource, settings, "Gantt");
}
public static void GanttQueryTaskbar(object model, object args)
{
var record = (GanttRecord)((Dictionary<string, object>)args)["Data"];
var taskbar = (PdfGanttTaskbar)((Dictionary<string,
object>)args)["Taskbar"];
if (!record.IsParentRow)
{
if (record.Progress > 80)
{
taskbar.ProgressColor = new PdfColor(108, 52, 131);
taskbar.TaskBorderColor = taskbar.TaskColor = new PdfColor(165, 105, 189);
}
else if (record.Progress < 20)
{
taskbar.ProgressColor = new PdfColor(205, 92, 92);
taskbar.TaskBorderColor = taskbar.TaskColor = new PdfColor(240, 128, 128);
}
}
}
private GanttProperties ConvertToGanttObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GanttProperties gridProp = new GanttProperties();
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(ds.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
return gridProp;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
```

Click here to view the online demo sample with above code example.

Sorting

The Gantt control for JavaScript has built-in support for sorting one or more columns.

Sorting Columns

Gantt allows the tasks to be sorted in ascending or descending order based on the selected column by enabling the AllowSorting option in Gantt control. The following code example shows you how to enable sorting in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.AllowSorting(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Multicolumn sorting

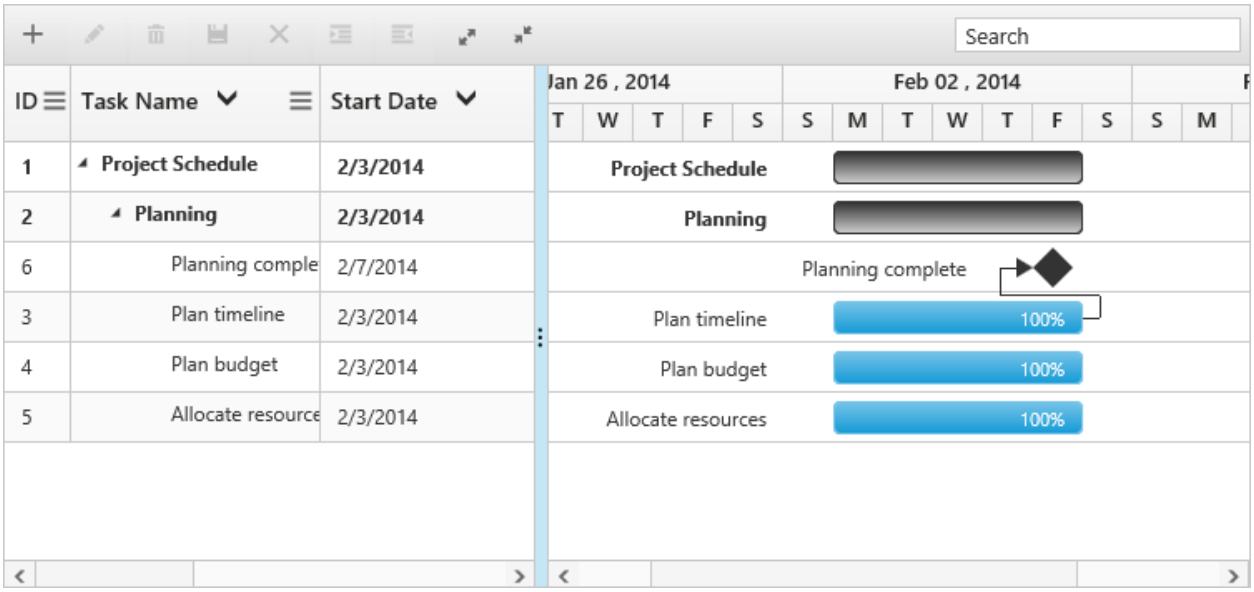
Gantt allows you to sort multiple columns by clicking the desired column headers while holding the CTRL key with AllowMultiSorting as true . The following code example shows you how to enable multicolumn sorting in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowSorting(true)
.AllowMultiSorting(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of multicolumn sorting in Gantt control.

Copyright © 2001 - 2020 Syncfusion Inc.



Multicolumn Sorting

Sorting column on Gantt initialization

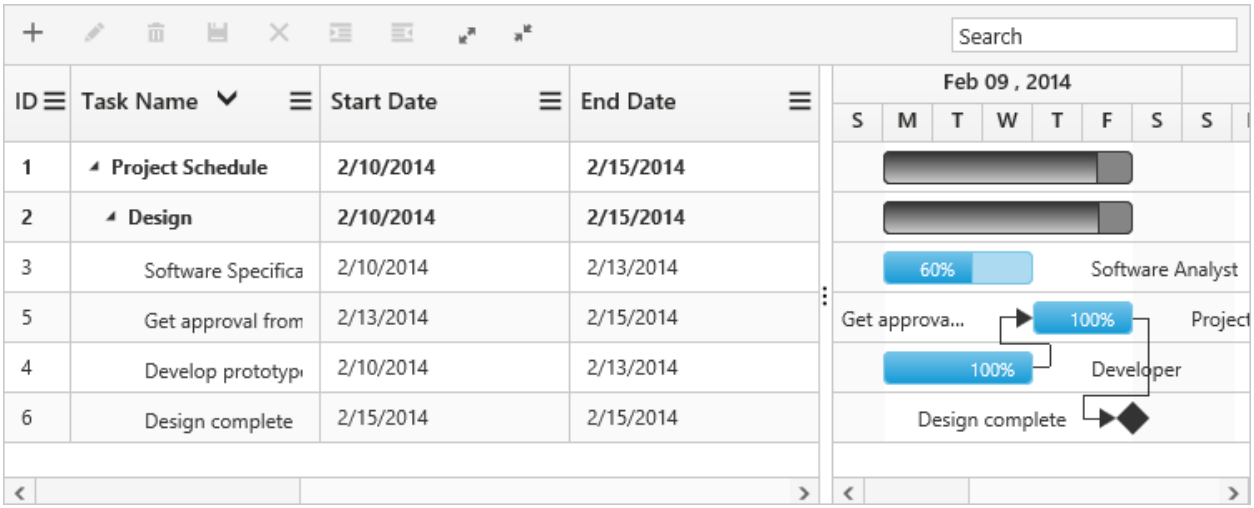
Gantt control can be rendered with sorted columns initially, this can be achieved by using SortSettings property. We can add columns which are sorted initially in SortedColumns collection. SortedColumns collection was defined with Field and Direction properties. The following code example shows how to add sorted column on Gantt initialization.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.AllowSorting(true)
.SortSettings(sort=>
{
sort.SortedColumns(sorted =>
sorted.Field("taskName").Direction(SortOrder.Descending).Add());
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



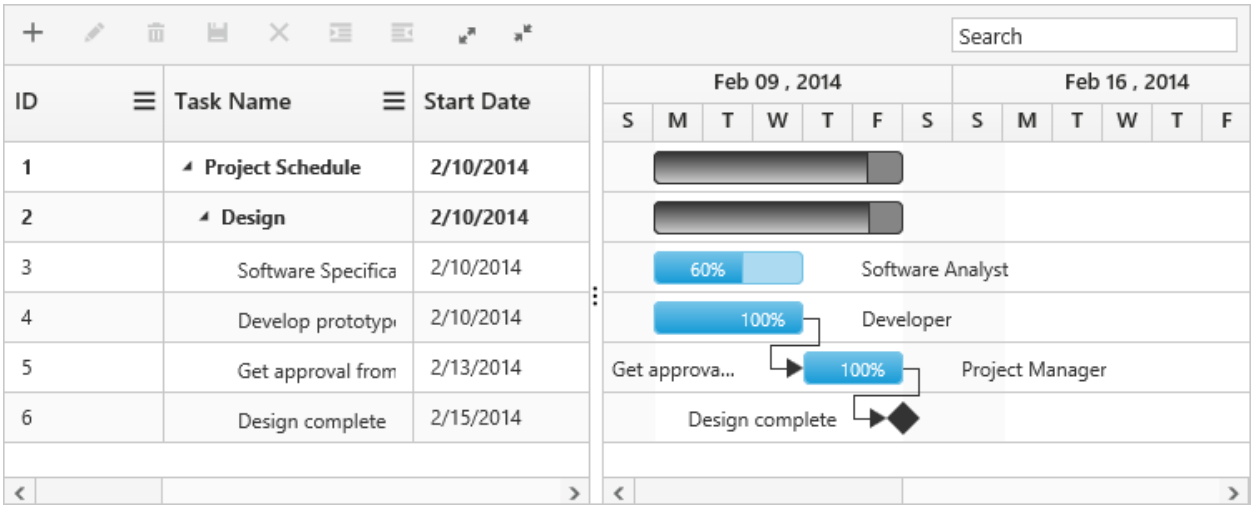
Sorting column dynamically

Columns in Gantt control can be sorted dynamically by using sortColumn method. The following code example shows how to invoke the sortColumn method on button click action.

ASPX-CS

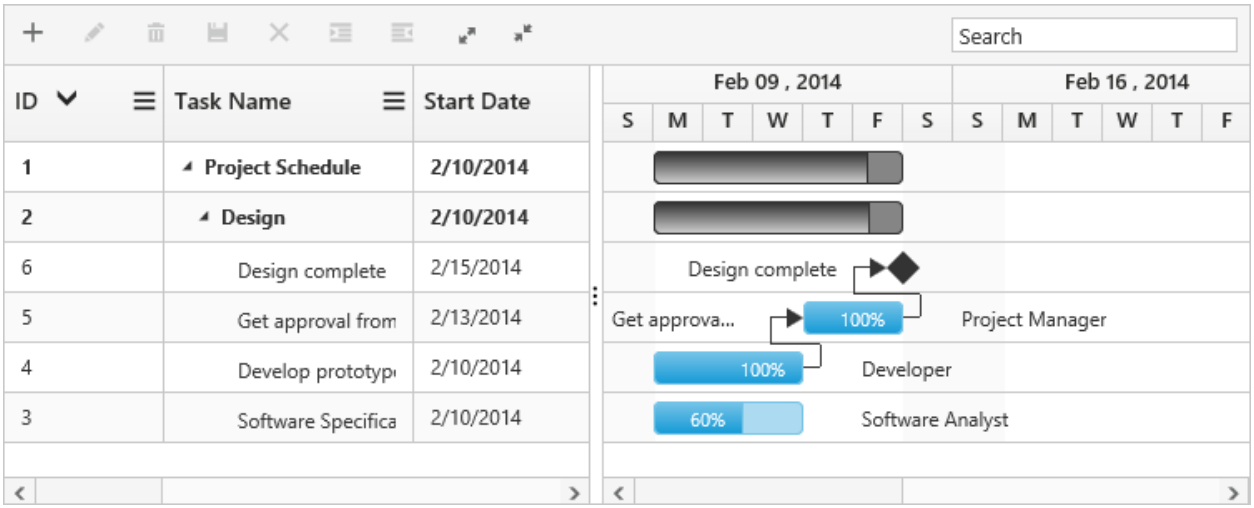
```
@(Html.EJ().Gantt("Gantt")
//...
.AllowSorting(true)
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#sort_column").click(function(){
$("#Gantt").ejGantt("sortColumn", "taskID", ej.sortOrder.Descending);
});
</script>
```

The following screenshot shows the output of above code example.



Before Sorting

Copyright © 2001 - 2020 Syncfusion Inc.



After Sorting

Note: when sorting a column using this method, previously sorted columns are cleared.

Sorting multiple columns dynamically

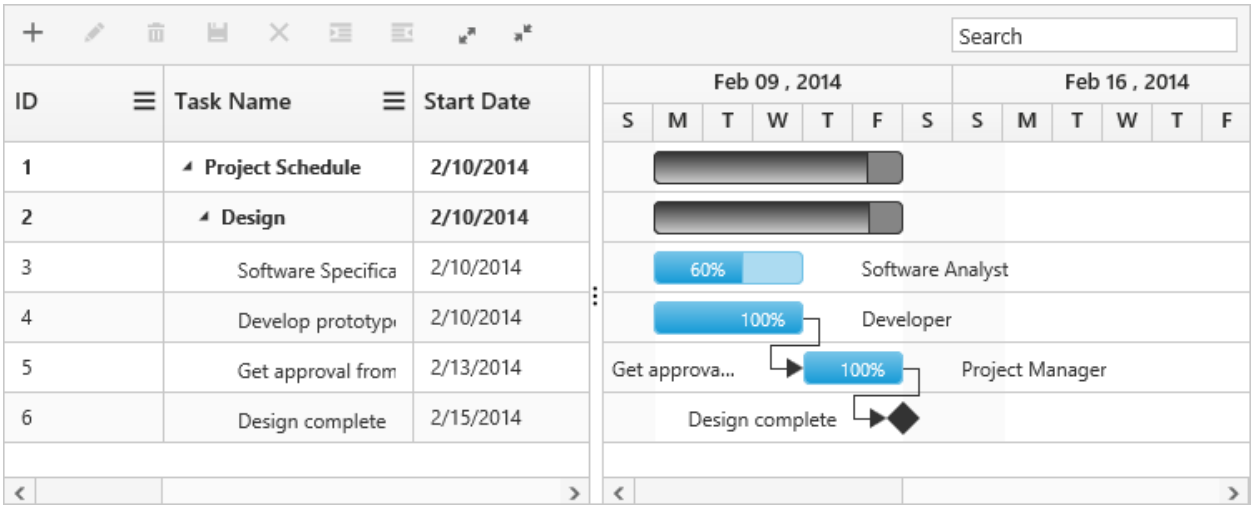
Multiple columns in Gantt can be sorted dynamically by using setModel method with sortSettingsproperty. The following code example shows how to use this method.

ASPX-CS

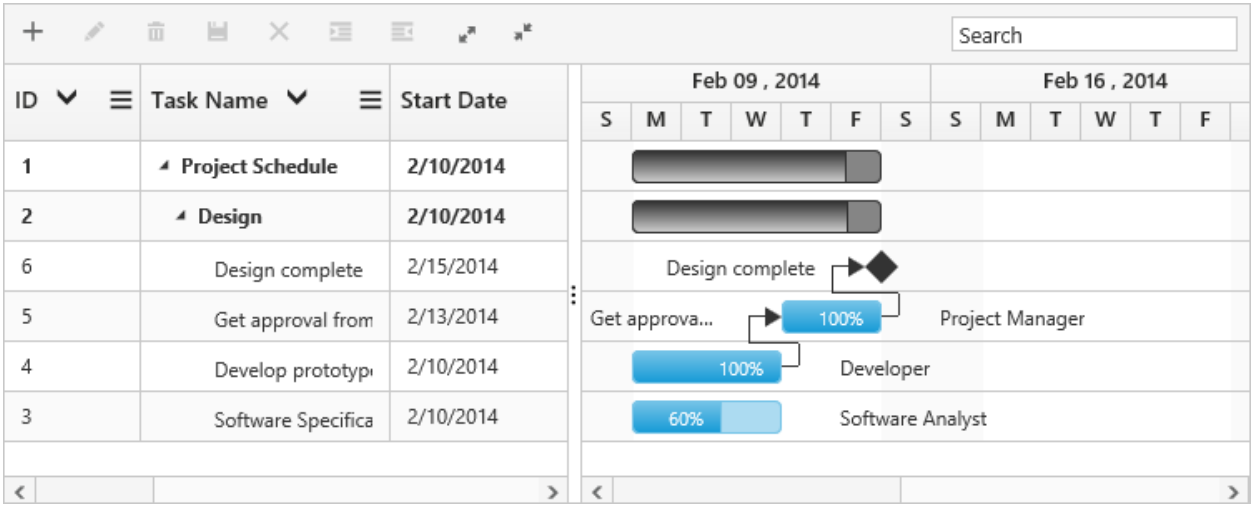
```
@(Html.EJ().Gantt("Gantt")
//...
.AllowSorting(true)
.AllowMultiSorting(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#sort_column").click(function() {
var sortedColumns = [
{ field: "taskID", direction: ej.sortOrder.Descending },
{ field: "taskName", direction: ej.sortOrder.Descending }
];
var sortSettings = {
sortedColumns: sortedColumns
};
var ganttObj = $("#Gantt").ejGantt("instance");
ganttObj.setModel({ "sortSettings": sortSettings });
});
</script>
```

The following screenshot shows the output of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Before Sorting



After Sorting

Searching

The Gantt control for JavaScript has in-built support for searching any content in Gantt.

Searching for content columns

In Gantt, we can search the content using the JavaScript method searchItem with search key as parameter. Also, we can integrate the search text box in Gantt toolbar by adding search toolbar item in ToolbarSettings.ToolbarItems property.

The following code example shows you how to add search option in Gantt toolbar.

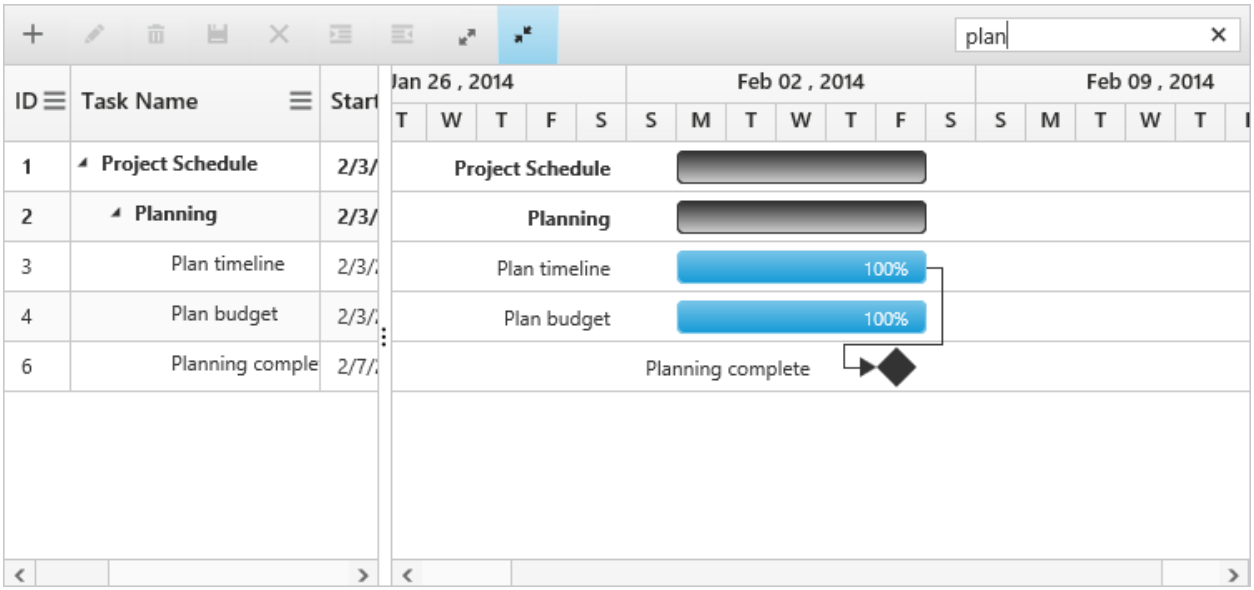
ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.ToolbarSettings(toolbar=>
{
toolbar.ShowToolbar(true);
toolbar.ToolbarItems(new List<GanttToolBarItems>(){
GanttToolBarItems.Search   //To search the task
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of searching for string in Gantt control.



Filtering

Filtering helps to view specific or related records from data source which meets a given filtering criteria. The FilterSettings property in Gantt is used to set the filtering criteria at load time.

Filter columns at initial load

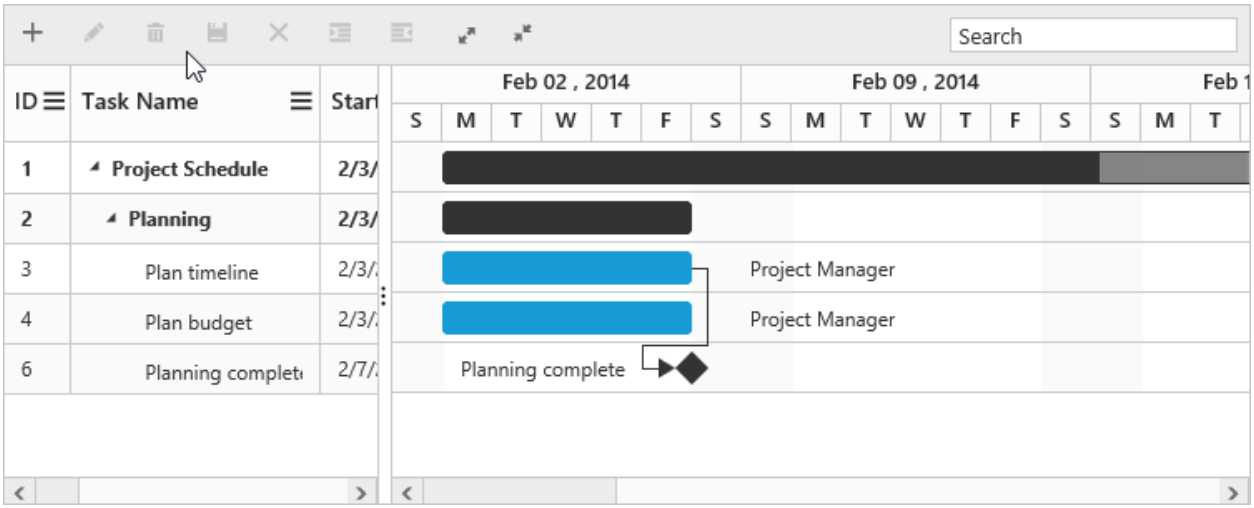
It is possible to filter one or more columns at initial load by providing the Field, Value,Predicate and Operator to the FilteredColumns property. The following code example explains how to filter a column on initial load.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
.FilterSettings(filter =>
{
filter.FilteredColumns(filtered =>
{
filtered.Value("plan").Field("taskName").Predicate("and").Operator(FilterOpe
ratorType.StartsWith).Add();
});
})
)
@(Html.EJ().ScriptManager())
```

The output of the filtering applied for a specific column is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Filtering a specific column by public method

It is possible to filter columns dynamically by using the filterColumn public method.

The below code snippet explains the above behavior.

JS

```
<button id="filterColumn">Filter Column</button>
@(Html.EJ().Gantt("GanttContainer")
//..
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#filterColumn").click(function (args) {
var obj = $("#GanttContainer").ejGantt("instance");
obj.filterColumn("taskName", "startswith", "plan");
})
</script>
```

Clearing the filter applied to Gantt

You can clear all the filtering condition done in the Gantt by using the clearFilter public method.

The below code snippet explains the above behavior.

JS

```
<button id="clearFilter">Clear Filter</button>
@(Html.EJ().Gantt("GanttContainer")
//..
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#clearFilter").click(function (args) {
var obj = $("#GanttContainer").ejGantt("instance");
obj.clearFilter();
})
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Toolbar

In Gantt we can show/hide the Toolbar by using ShowToolbar property.We can add default toolbar items by ToolbarItems. User can also create a custom toolbar items by using CustomToolbarItems.

Default Toolbar Items

Using Gantt default toolbar items we can perform below operations.

- **Add**- To add new task.
- **Edit**-To edit a selected task.
- **Delete**- To delete a selected task.
- **Cancel**- To cancel the edited changes in a task.
- **Update**- To save the edited changes in a task.
- **ExpandAll**- To expand all the Gantt rows.
- **CollapseAll**- To collapse all the Gantt rows.
- **Indent**- To indent the selected task in Gantt.
- **Outdent**- To outdent the selected task in Gantt.
- **CriticalPath**- To support critical path in Gantt.
- **PrevTimeSpan**- To navigate the Gantt timeline to previous time span.
- **NextTimeSpan**- To navigate the Gantt timeline to Next time span.
- **Search**- To perform search operation in Gantt.
- **ExcelExport**- To export Gantt in Excel format.
- **PdfExport**- To export Gantt in PDF format.

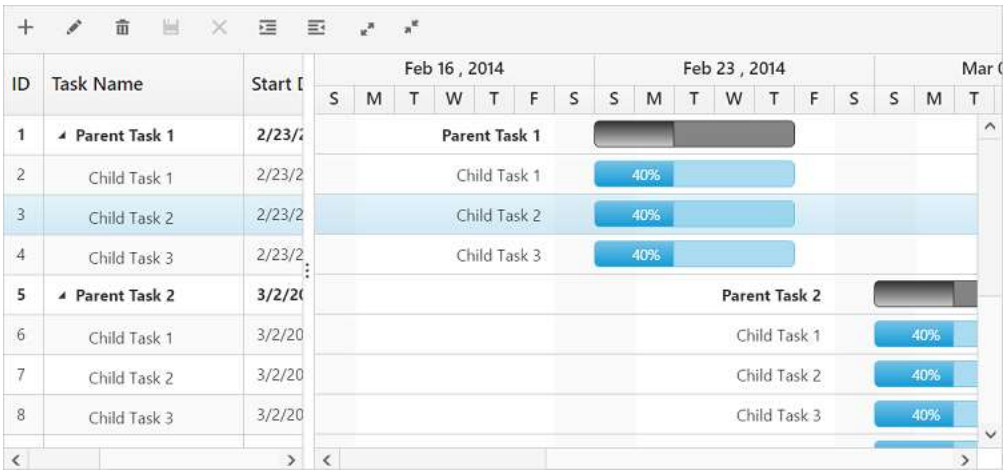
We can enable Gantt toolbar by using below code example:

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true);
toolbar.ToolbarItems(new List<GanttToolBarItems>()
{
GanttToolBarItems.Add,
GanttToolBarItems.Edit,
GanttToolBarItems.Delete,
GanttToolBarItems.Update,
GanttToolBarItems.Cancel,
GanttToolBarItems.Indent,
GanttToolBarItems.Outdent,
GanttToolBarItems.ExpandAll,
GanttToolBarItems.CollapseAll,
GanttToolBarItems.NextTimeSpan,
GanttToolBarItems.PrevTimeSpan,
GanttToolBarItems.Search,
GanttToolBarItems.PdfExport,
GanttToolBarItems.ExcelExport,
GanttToolBarItems.CriticalPath
});
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays the toolbar option in Gantt control.



Note: To perform add,edit,delete,cancel,update,indent,outdent using Toolbar items we need to enable add/edit/delete/indent using EditSettings.

Custom Toolbar Items

CustomToolbarItems allows us to insert custom icons and custom template in Gantt toolbar. By using below properties we can customize Gantt toolbar as per our requirement.

- **Text**- To insert the custom icons in toolbar using CSS class name selector.
- **TemplateID**-To insert the custom icons in toolbar using script templates. Using this property we can bind HTML elements and other EJ controls to Gantt toolbar.
- **TooltipText**-Displays tooltip text for the custom icons.

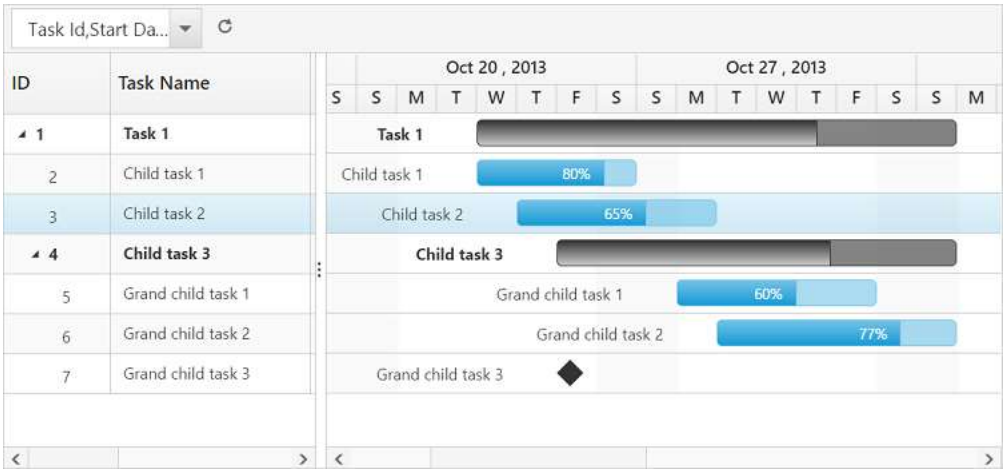
To insert EJ Controls in Gantt toolbar we need to initiate the control in Create client side event.In ToolbarClick client side event we can bind actions to the custom toolbar items.

ASPX-CS

```
@(Html.EJ().Gantt("ToolbarTemplate")
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true);
toolbar.CustomToolbarItems(ct =>
{
ct.TemplateID("#ColumnVisibility").TooltipText("Column Visibility").Add();
ct.Text("Reset").TooltipText("Reset").Add();
});
})
.ClientSideEvents(cs =>
{
cs.ToolbarClick("toolbarClick");
cs.Create("create");
})
)
@(Html.EJ().ScriptManager())
<script id="ColumnVisibility" type="text/x-jsrender">
<input id="dropdownContainer" />
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function toolbarClick(args) {
if (args.itemName == "Reset") {
//we can bind the custom actions here
}
}
//Here we can append custom EJ controls
function create(args) {
$("#dropdownContainer").ejDropDownList({ });
}
</script>
<style type="text/css" class="cssStyles">
#ToolbarTemplate_ColumnVisibility {
padding-top: 2px;
padding-bottom: 0px;
}
.Reset:before {
content: "\e677";
}
</style>
```



Click here to view the demo sample for custom toolbar item

Baseline

Baseline is used to describe the original plan of the task and it can be the same as current duration of the task or different.

Baseline can be enabled by setting `RenderBaseline` as `true` and baseline color can be changed by `BaselineColor` property.

To render the baseline you have to map the baseline start date and baseline end date values from data source.

This can be done by using `BaselineStartDateMapping` and `BaselineEndDateMapping` properties.

The following code example shows how to enable baseline in Gantt control.

ASPX-CS

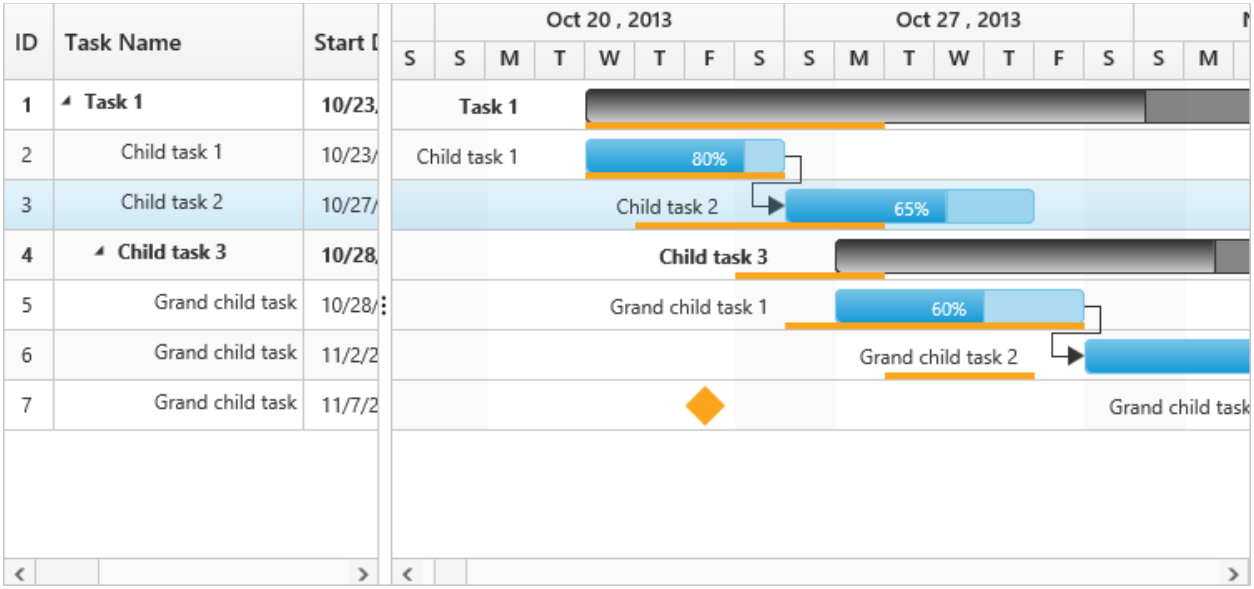
```
@(Html.EJ().Gantt("Gantt")
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.BaselineStartDateMapping("baselineStartDate") //Mapping BaselineStartDate
to Gantt
.BaselineEndDateMapping("baselineEndDate") //Mapping BaselineEndDate to
Gantt
.RenderBaseline(true) //Show/Hide the baseline in Gantt
.BaselineColor("#fba41c")
)
@(Html.EJ().ScriptManager())
```

Click here to view the online demo sample for baselines in Gantt.

The following screenshot shows the baseline in Gantt control.



Work Breakdown Structure

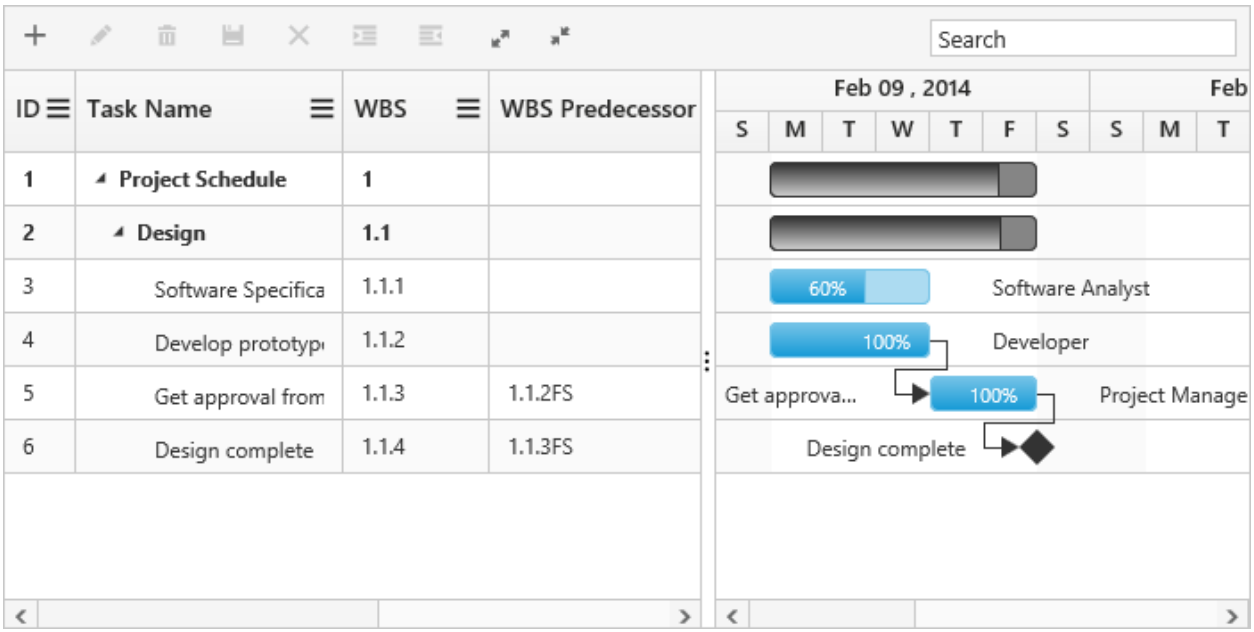
Work Breakdown Structure(WBS) in Gantt represents the entire project activities in various sub modules. It is used to split the large tasks into manageable small tasks. WBS value and WBS predecessor value of Gantt tasks are displayed in WBS column and WBS predecessor column. This can be enabled in Gantt by using `EnableWBS` and `EnableWBSPredecessor` properties. The following code example shows how to enable WBS columns in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
// ...
.EnableWBS(true)
.EnableWBSPredecessor(true)
)
@(Html.EJ().ScriptManager())
```

The below screenshot depicts the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample for WBS in Gantt.

Resources

Resources are represented by staff, equipment and materials etc. In Gantt control you can show /allocate the resources (human resources) for each task.

Resource Collection

The resource collection contains details about the resources that are used in the project. Resources are List object that contains id and name of the resources and this collection is mapped to the Gantt control using Resources property.

Id and name field of the resources are mapped by using the ResourceIdMapping and ResourceNameMapping properties.

The following code snippets shows resource collection object and how it assinged to Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//…
.ResourceIdMapping("ResourceId")
.ResourceNameMapping("ResourceName")
.Resources(ViewBag.resource)
)
@(Html.EJ().ScriptManager())
```

C#

```
public ActionResult Resource()
{
ViewBag.resource = this.GetResources();
//...
return View();
}
// Create resource collection
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class Resource
{
public int ResourceId { get; set; }
public string ResourceName { get; set; }
}
//...
public List<Resource> GetResources()
{
List<Resource> ResourceCollection = new List<Resource>();
ResourceCollection.Add(new Resource() { ResourceId = 1, ResourceName =
"Project Manager" });
ResourceCollection.Add(new Resource() { ResourceId = 2, ResourceName =
"Software Analyst" });
ResourceCollection.Add(new Resource() { ResourceId = 3, ResourceName =
"Developer" });
ResourceCollection.Add(new Resource() { ResourceId = 4, ResourceName =
"Testing Engineer" });
return ResourceCollection;
}
```

Assign Resource

We can assign resources for a task at initial load, using the resource id value of the resources as a collection. This collection is mapped from the DataSource to the Gantt control using the ResourceInfoMapping property.

The following code snippet shows how to assign the resource for each task and map to Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//…
.ResourceInfoMapping("Resources")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

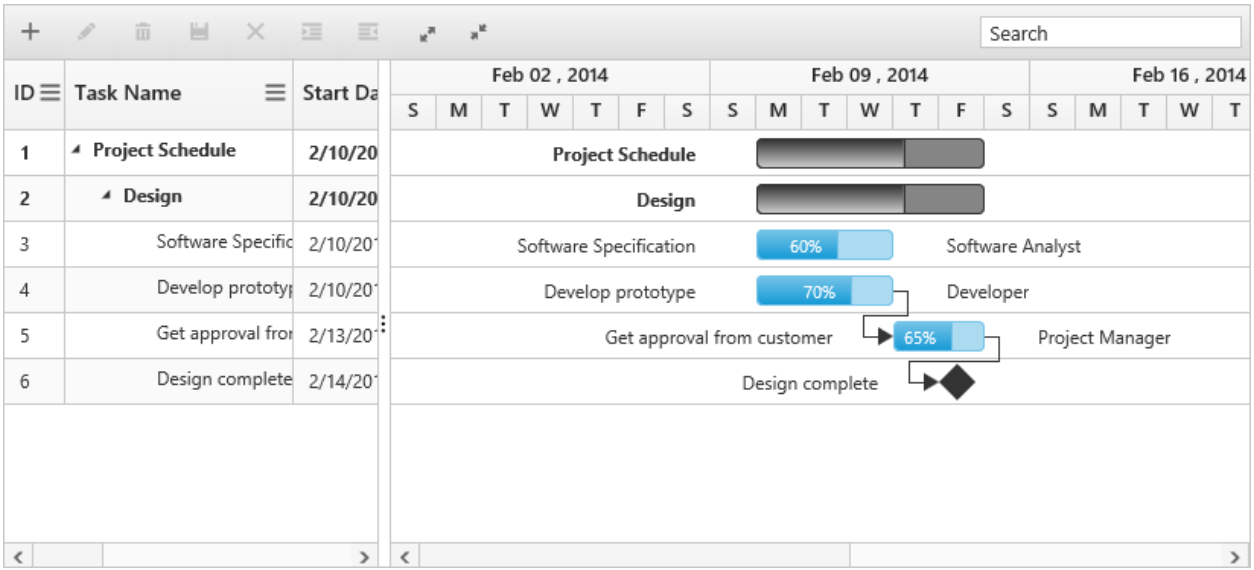
C#

```
public ActionResult Resource()
{
ViewBag.datasource = GetData();
//...
return View();
}
public class ResourceData
{
//...
public List<int> Resources { get; set; }
}
public List<ResourceData> GetData()
{
//..
List<ResourceData> list = new List<ResourceData>();
list.Add(new ResourceData()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//...
Resources =new List<int>(){2},
//..
});
return list;
}
```

The following screenshot shows Gantt control with resources.



Assign Resources with Unit

Resource units indicates the amount of work done by a resource for the task. We can specify the resource unit for the each assigned resource for a particular task.

Resource unit value is mapped to the Gantt tasks from the datasource using `ResourceUnitMapping` property.

The below code snippets shows how to assign resource unit value.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//…
.ResourceInfoMapping("Resources")
.ResourceUnitMapping("ResourceUnit")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

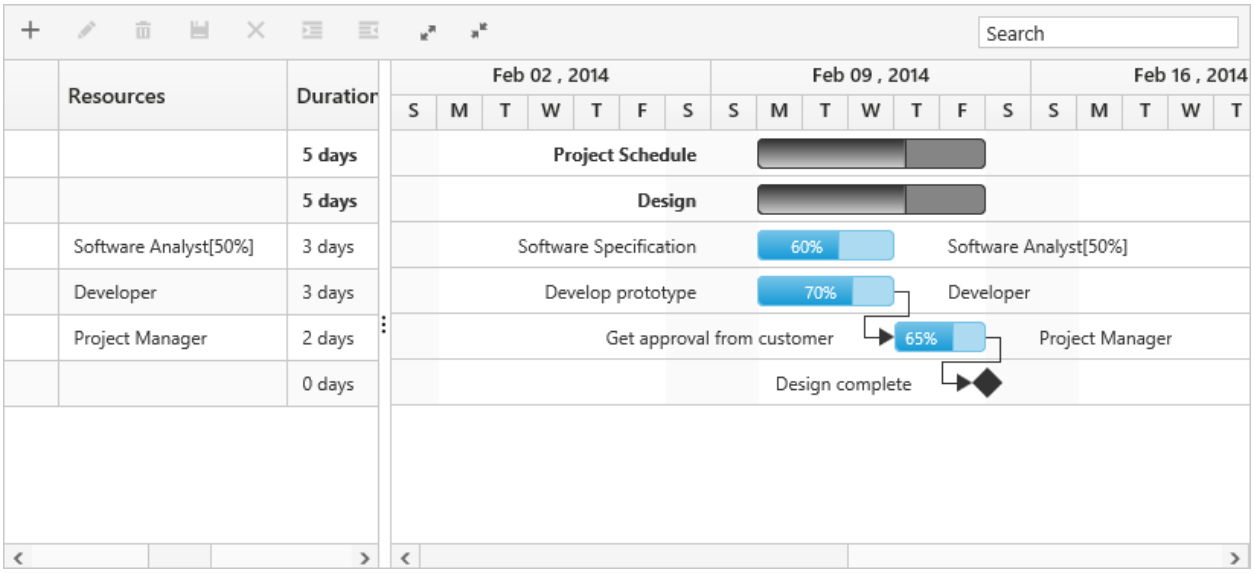
C#

```
public ActionResult Resource()
{
ViewBag.datasource = GetData();
//...
return View();
}
public class ResourceObject
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public int ResourceId { get; set; }
public int ResourceUnit { get; set; }
}
public class ResourceData
{
//...
public List<ResourceObject> Resources { get; set; }
}
public List<ResourceData> GetData()
{
//..
List<ResourceData> list = new List<ResourceData>();
list.Add(new ResourceData()
{
//...
Resources = new List<ResourceObject>(){ new ResourceObject(){ ResourceId=2,
ResourceUnit=50}},
//..
});
return list;
}
```

The following screenshot shows Gantt control with resource units.



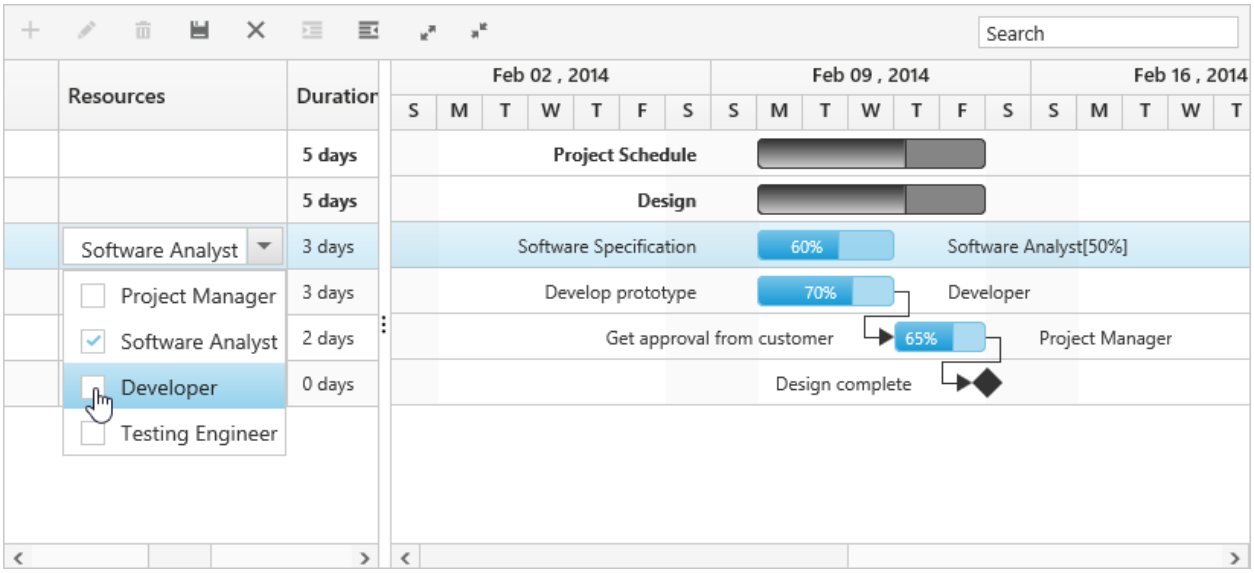
Edit Resource Collection

By using cell edit option we can add/remove the resource for particular task and by using dialog edit support we can modify the resource unit value also.

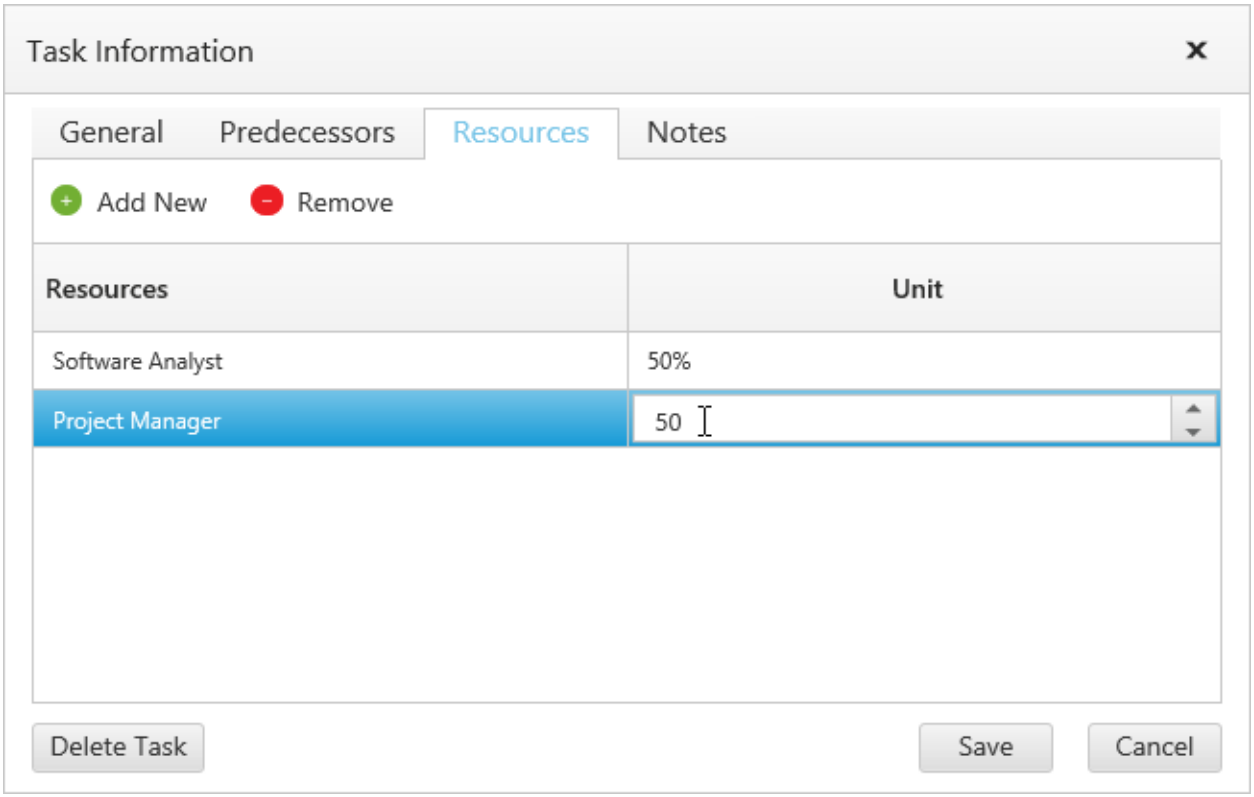
Refer this link to know more about editing in Gantt control.

The following screenshot shows the resource edit option in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



Editing resource with cell edit



Editing resource with edit dialog

Note: While editing, resource values are saved with resource unit value when resource unit is not equal to 100% like below.

C#

```
public ActionResult Resource()
{
ViewBag.datasource = GetData();
```

```
//...
return View();
}
public class ResourceObject
{
public int ResourceId { get; set; }
public int ResourceUnit { get; set; }
}
public class ResourceData
{
//...
public List<ResourceObject> Resources { get; set; }
}
public List<ResourceData> GetData()
{
//..
List<ResourceData> list = new List<ResourceData>();
list.Add(new ResourceData()
{
//...
Resources = new List<ResourceObject>(){ new ResourceObject(){ ResourceId=2,
ResourceUnit=50}},
//..
});
return list;
}
```

Note: When resource unit is 100%, resource value is stored with ID only.

C#

```
public ActionResult Resource()
{
ViewBag.datasource = GetData();
//...
return View();
}
public class ResourceData
{
//...
public List<int> Resources { get; set; }
}
public List<ResourceData> GetData()
{
//..
List<ResourceData> list = new List<ResourceData>();
list.Add(new ResourceData()
{
//...
Resources =new List<int>(){2},
//..
});
return list;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The unit values are updated to the resources assigned to the Gantt tasks, only when mapping resource units data source field using the ResourceUnitMapping property.

Resource unit will be updated while editing the duration and work fields with respective to task type.

Work

The Work is the total labor hours necessary to complete a task. Work can be mapped from the data source field using the property WorkMapping and when the work value is mapped from the data source, the end date and duration of the task will be calculated automatically based on the work and resource unit values from the data source.

The below code snippet explains mapping the work value from the data source.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
// ...
.WorkMapping("estimatedWork")
)
@(Html.EJ().ScriptManager())
```

Note:

- When the work is mapped from the data source and if the resource unit is not defined for a task, the work will be calculated as 0 and task will be displayed as a milestone.
- When the work field is mapped from the data source, the default task type will be fixed work.

When the work value for a task is not mapped from the data source using the WorkMapping property, it will be calculated internally based on the number of resources assigned to the task and the duration of the task.

Note:

- By Default, the work field value for a task will be calculated based on the duration and assigned resource unit.
- If an assigned resource has resource unit value of 100% for a task, i.e. if a resource is dedicatedly assigned to a task, the work per day will be **8 hours**.

Work can be measured in hours, days and minutes, and it is measured in 'hours' scale by default and this can be changed by using the workUnit property.

The below code snippet explains how to change work unit property in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
// ...
.WorkUnit(GanttWorkUnit.Minute)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Task type

The work, duration and resource unit fields of a task depends upon each other values and will change automatically on editing any one of these fields. But we can also set these field's values as constant using the taskType property. The following values can be set to the type property,

Fixed Duration - Duration task field will remain constant while updating resource unit or work field.

Fixed Work - Work field will remain constant while updating resource unit or duration fields.

Fixed Unit - Resource units will remain constant while updating duration or work field.

The following code snippet explains how to set taskType property in Gantt,

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
// ...
.TaskType(GanttTaskType.FixedWork)
)
@(Html.EJ().ScriptManager())
```

Following table explains how the work, duration and resource unit fields will gets updated on changing any of the fields

Task Type	Changes in Duration	Changes in Work	Changes in Resource units
Fixed Duration	Work field updates	Resource unit updates	Work field updates
Fixed Work	Resource unit updatesNote: For manually scheduled task work will update.	Duration field updatesNote: For manually scheduled task resource unit updates.	Duration will updateNote: For manually scheduled task work field updates.
Fixed Unit	Work field updates	Duration field updatesNote: For manually scheduled task resource unit updates.	Duration will updateNote: For manually scheduled task work field updates.

Note:

- **Fixed Unit** is the default taskType in Gantt.
- The above calculations are not applicable for Milestones.

Effort driven tasks

The effortDriven field is used to define whether the total labor hours or the work need to split or to be extended among the resources when adding or removing a resource to a task.

Note:

- Effort driven field is effective only on adding or removing a resource in a task.

The below table illustrates the effects of effort driven field with TaskType property.

Task Type	Effort driven tasks	Task without effort driven

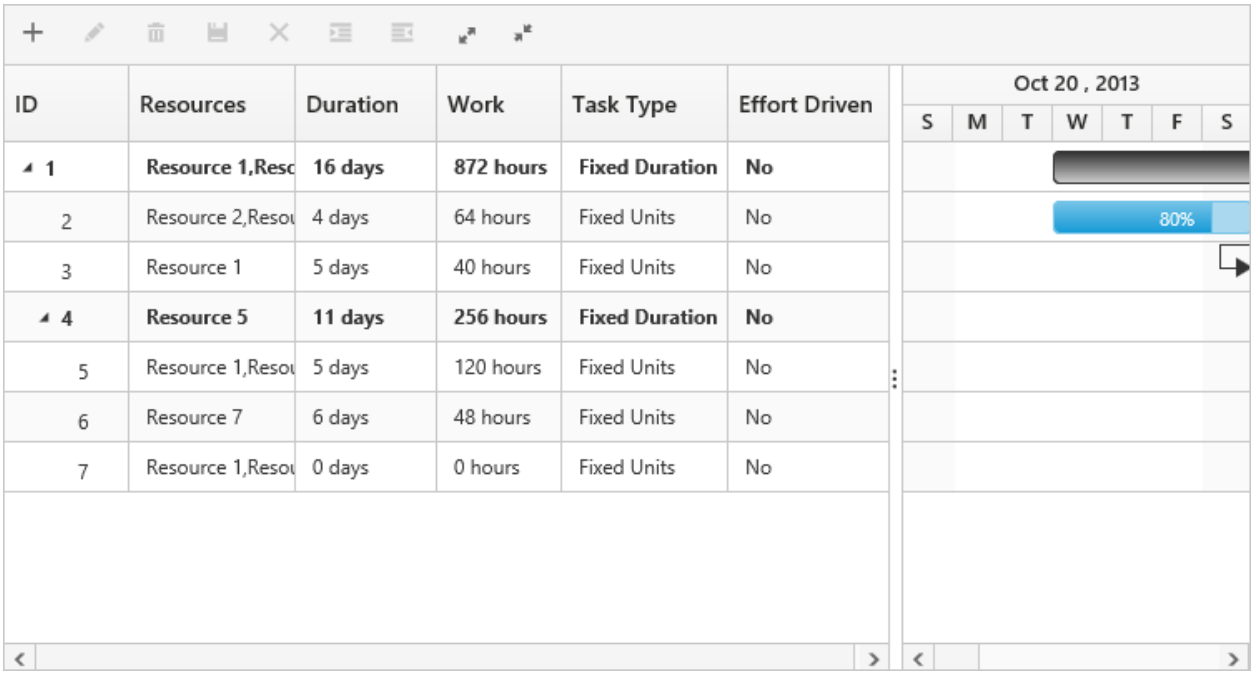
Copyright © 2001 - 2020 Syncfusion Inc.

Fixed Duration	Resource unit updates	Work field updates
Fixed Work	Duration field updatesNote: For Fixed Work effort driven is always true.In this task type the effort driven field is non-editable.	Not applicable
Fixed Unit	Duration field updates	Work field updates

Notes:

- The above calculations will not be considered for Milestones.
- For manually scheduled task effort driven will not be considered while adding or removing resources.
- For manually scheduled task the taskType and effortDriven fields are non-editable.
- When assigning a first resource for the task the effortDriven and taskType fields are not considered, the work field alone will be calculated.

The following screen shot depicts the work, task type and effort driven fields in Gantt.



Stripline

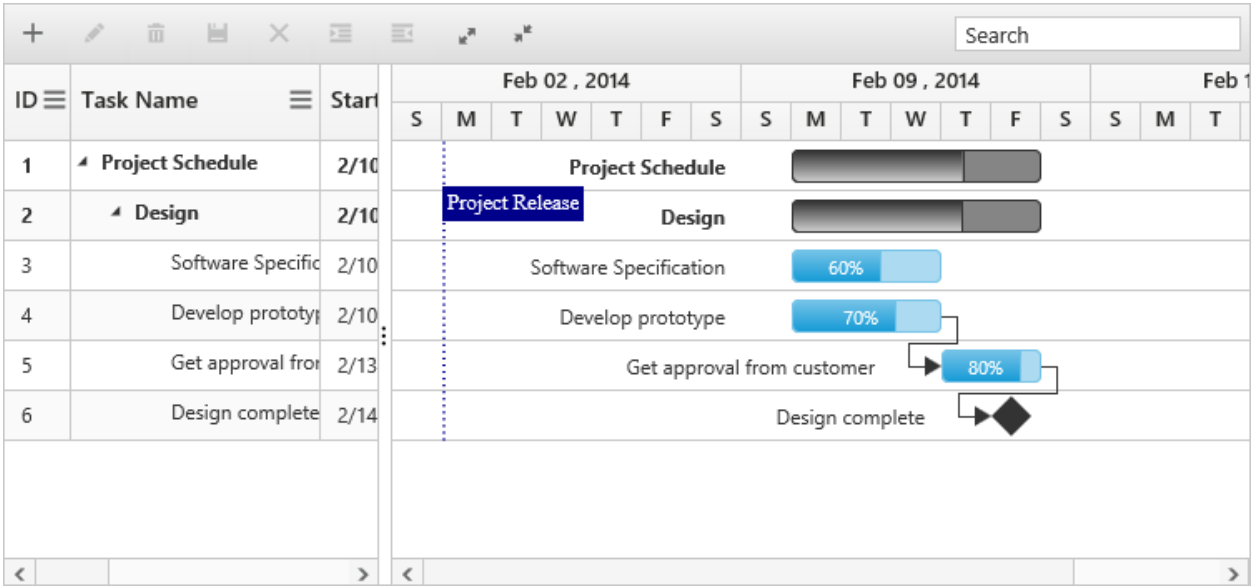
Stripline in Gantt control is used to highlight the important event in Gantt chart part. By using this feature, you can add the Striplines to highlight important days in your project. The following code example shows you how to add the Stripline in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.StripLines(new    List<Syncfusion.JavaScript.Models.StripLine>
{
```

```
new Syncfusion.JavaScript.Models.StripLine()
{
Day="03/02/2014",
Label="Project Release",
LineStyle="dotted",
LineColor="blue",
LineWidth=2 },
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows stripline in Gantt control.



Holidays

The Holidays in Gantt control is used to highlight the non-working days in Gantt control and it can be initialized by using Holidays property.

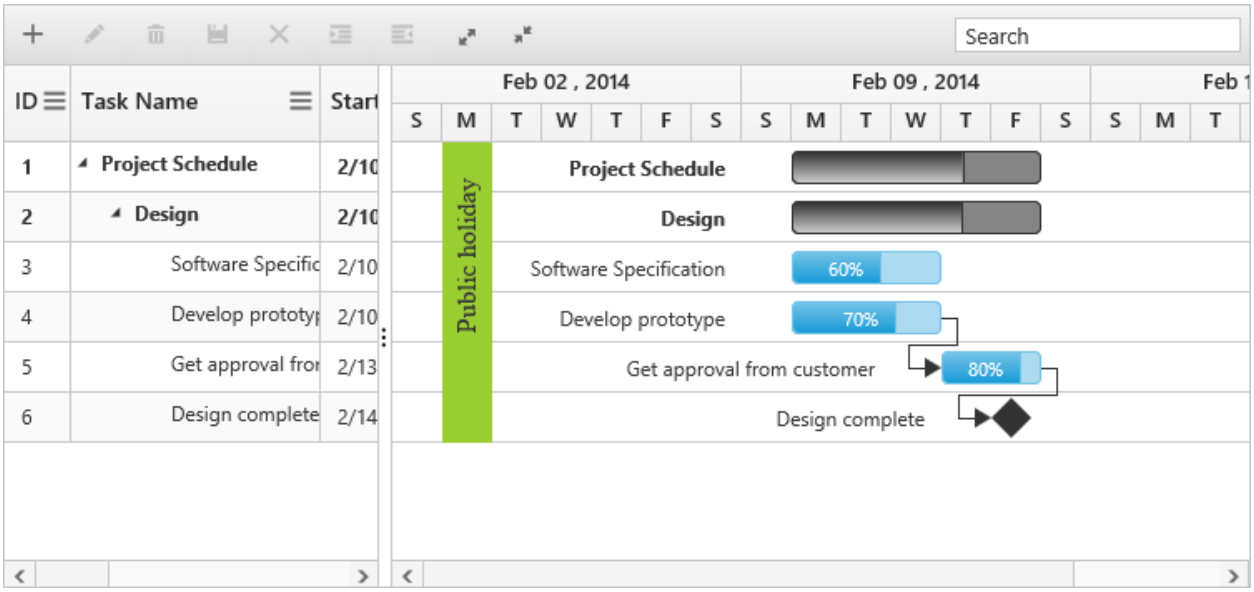
Each holiday can be defined with Day, Label, Background properties. The following code example shows how to initialize the Holidays in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.Holidays(new List<Syncfusion.JavaScript.Models.Holiday>
{
new Syncfusion.JavaScript.Models.Holiday()
{
Day="2/03/2014",
Background="YellowGreen",
Label="Public holiday"
},
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot shows the output of Holidays in Gantt control.



We can also load holidays from SQL database into Gantt. Click here to view our knowledge base for this.

Localization

Localization is the process of customizing the User Interface (UI) based on a culture, specific to a particular country or region, in order to display regional data. The culture is represented by a unique string like en-US for US English and fr-FR for French.

Localization is the key feature that provides solutions to global customers with the help of localized control. It is necessary to include the specific culture script files (ej.culture.fr-FR.min.js file for French culture) in the reference section, which is available in the following location.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Also it is possible to localize all the texts in the Gantt control with specific culture by referring the ej.localetexts.fr-FR.min.js file which is available in the following location.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i10n

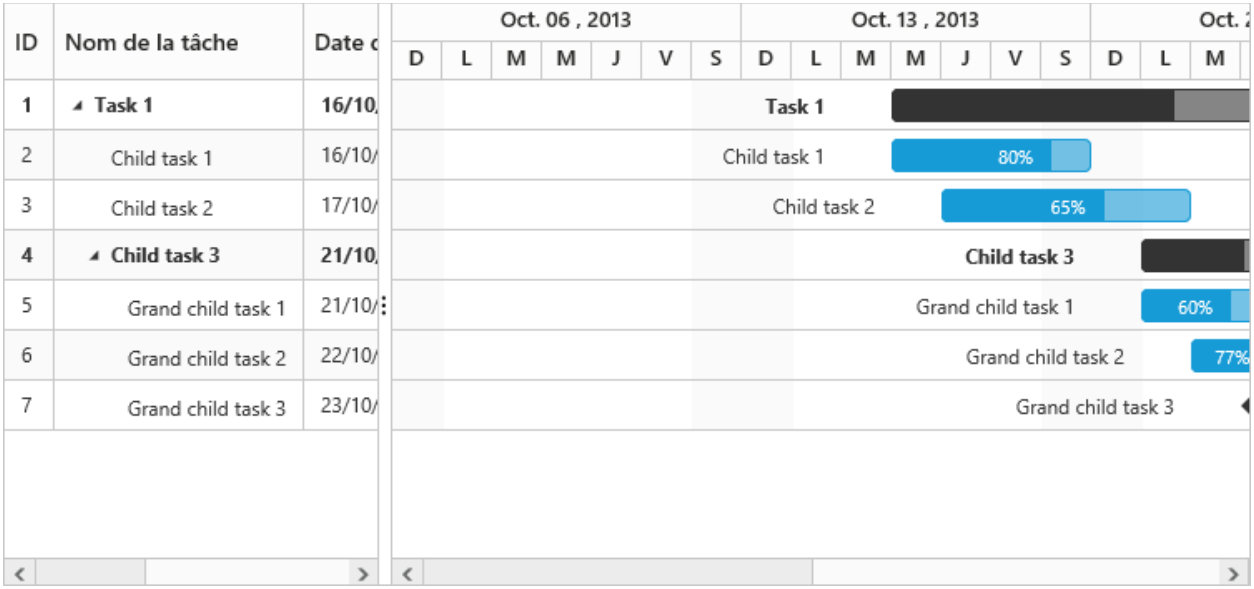
Culture of the Gantt control can be defined by using Locale property.The following code example explains on how to localize the control in French culture

JAVASCRIPT

```
<head>
//…
<script src="Scripts/ej.culture.fr-FR.min.js"></script>
<script src="Scripts/ej.localetexts.fr-FR.js"></script>
</head>
@(Html.EJ().Gantt("Gantt")
//…
.Locale("fr-FR")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
@(Html.EJ().ScriptManager())
```



Which will localize all the text and date formats to French culture. If we still need to customize the localized text, we can define them locally in the sample as explained below.

JAVASCRIPT

```
@(Html.EJ().Gantt("Gantt")
//…
.Locale("fr-FR")
)
@(Html.EJ().ScriptManager())
<script type = "text/javascript">
ej.Gantt.Locale["fr-FR"] = {
emptyRecord: "Aucun enregistrement à afficher",
alertTexts: {
indentAlert: "Il n'y a aucune trace de Gantt est sélectionné pour effectuer
le retrait",
outdentAlert: "Il n'y a aucune trace de Gantt est sélectionné pour effectuer
le retrait négatif",
predecessorEditingValidationAlert: "Cyclique dépendance survenu, S'il vous
plaît Consultez le prédécesseur",
predecessorAddingValidationAlert: "Remplissez toutes les colonnes dans la
table prédécesseur",
idValidationAlert: "Duplicate ID",
dateValidationAlert: "Invalid Date de fin",
dialogResourceAlert: "Remplissez toutes les colonnes du tableau des
ressources"
},
//headerText to be displayed in TreeGrid
columnHeaderTexts: {
taskId: "Tâche Ia",
taskName: "Tâche Tâche",
startDate: "Démarrer Date",
endDate: "Fin Date",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
resourceInfo: "Ressources",
duration: "Durée",
status: "Progrès",
predecessor: "Prédécesseur",
baselineStartDate: "Baseline Démarrer Date",
baselineEndDate: "Baseline Fin Date"
},
//string to display in dialog
editDialogTexts: {
addFormTitle: "Nouveau Tâche",
editFormTitle: "Modifier Tâche",
saveButton: "Sauver",
cancelButton: "Annuler"
},
columnDialogTexts: {
field: "Champ",
headerText: "En-tête",
editType: "Modifier le type",
filterEditType: "Modifier le type de filtre",
allowFiltering: "Autoriser le filtrage",
allowFilteringBlankContent: "Autoriser le filtrage du contenu Blank",
allowSorting: "Autoriser tri",
visible: "Visible",
width: "Largeur",
textAlign: "Alignement du texte",
headerTextAlign: "Alignement du texte en-tête",
columnsDropdownData: "Colonne Chute de données vers le bas",
dropdownTableText: "Texte",
dropdownTableValue: "Valeur",
addData: "Ajouter",
deleteData: "Retirer",
allowCellSelection: "Autoriser la sélection de cellules",
clipMode: "Mode Clip",
tooltip: "Info-bulle",
headerTooltip: "Header Tooltip"
},
editTypeTexts: {
string: "Chaîne",
numeric: "Numérique",
datePicker: "Sélecteur de date",
dateTimePicker: "Date Time Picker",
dropdown: "Menu déroulant",
boolean: "Boolean"
},
textAlignTypes: {
right: "Droite",
left: "La gauche",
center: "centre"
},
clipModeTexts: {
clip: "Agrafe",
ellipsis: "Ellipse"
},
toolboxTooltipTexts: {
addTool: "Ajouter",
editTool: "modifier",
saveTool: "Mettre à jour",
```

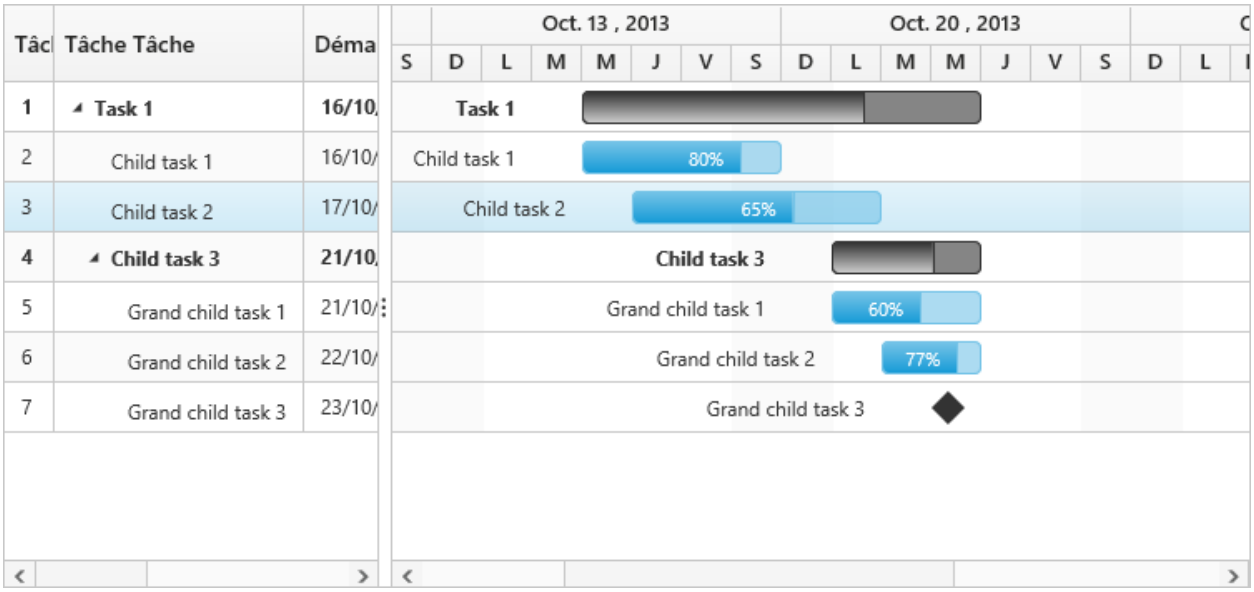
Copyright © 2001 - 2020 Syncfusion Inc.

```
deleteTool: "Effacer",
cancelTool: "Annuler",
searchTool: "Chercher",
indentTool: "tiret",
outdentTool: "Retrait négatif",
expandAllTool: "Développer tout",
collapseAllTool: "Réduire tout",
nextTimeSpanTool: "Suivant Timespan",
prevTimeSpanTool: "Précédent Timespan",
criticalPathTool: "Chemin critique",
excelExportTool: "Exportation Excel",
pdfExportTool: "Exportation PDF"
},
durationUnitTexts: {
days: "journées",
hours: "heures",
minutes: "minutes",
day: "journée",
hour: "heure",
minute: "minute"
},
durationUnitEditText: {
minute: ["m", "min", "minute", "minutes"],
hour: ["h", "heure", "heure", "heures"],
day: ["ré", "dy", "journée", "journées"]
},
workUnitTexts: {
days: "journées",
hours: "heures",
minutes: "minutes"
},
taskTypeTexts: {
fixedWork: "travail fixe",
fixedUnit: "Unités fixes",
fixedDuration: "Durée fixe"
},
effortDrivenTexts: {
yes: "Oui",
no: "Non"
},
contextMenuTexts: {
taskDetailsText: "Détails de la tâche ...",
addNewTaskText: "Ajouter une nouvelle tâche",
indentText: "tiret",
outdentText: "Retrait négatif",
deleteText: "Effacer",
aboveText: "Au dessus de",
belowText: "Au dessous de"
},
newTaskTexts: {
newTaskName: "Nouvelle tâche"
},
columnMenuTexts: {
sortAscendingText: "Trier par ordre croissant",
sortDescendingText: "Trier par ordre décroissant",
columnsText: "colonnes",
insertColumnLeft: "Insérez la colonne de gauche",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
insertColumnRight: "Insérez la colonne de droite",
deleteColumn: "Supprimer la colonne",
renameColumn: "Renommer la colonne"
},
taskModeTexts: {
manual: "Manuel",
auto: "Auto"
},
columnDialogTitle: {
insertColumn: "Insérer une colonne",
deleteColumn: "Supprimer la colonne",
renameColumn: "Renommer la colonne"
},
deleteColumnText: "Êtes-vous sûr de vouloir supprimer cette colonne?",
okButtonText: "D'accord",
cancelButtonText: "Annuler",
confirmDeleteText: "Confirmation de la suppression",
predecessorEditingTexts: {
fromText: "De",
toText: "À"
},
dialogTabTitleTexts: {
generalTabText: "Général",
predecessorsTabText: "Prédécesseurs",
resourcesTabText: "Ressources",
customFieldsTabText: "Les champs personnalisés",
notesTabText: "Remarques"
},
predecessorCollectionText: [{
id: "SS",
text: "Démarrer-Démarrer",
value: "Démarrer-Démarrer"
},
{
id: "SF",
text: "Démarrer-terminer",
value: "Démarrer-terminer"
},
{
id: "FS",
text: "terminer-Démarrer",
value: "terminer-Démarrer"
},
{
id: "FF",
text: "terminer-terminer",
value: "terminer-terminer"
}
],
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



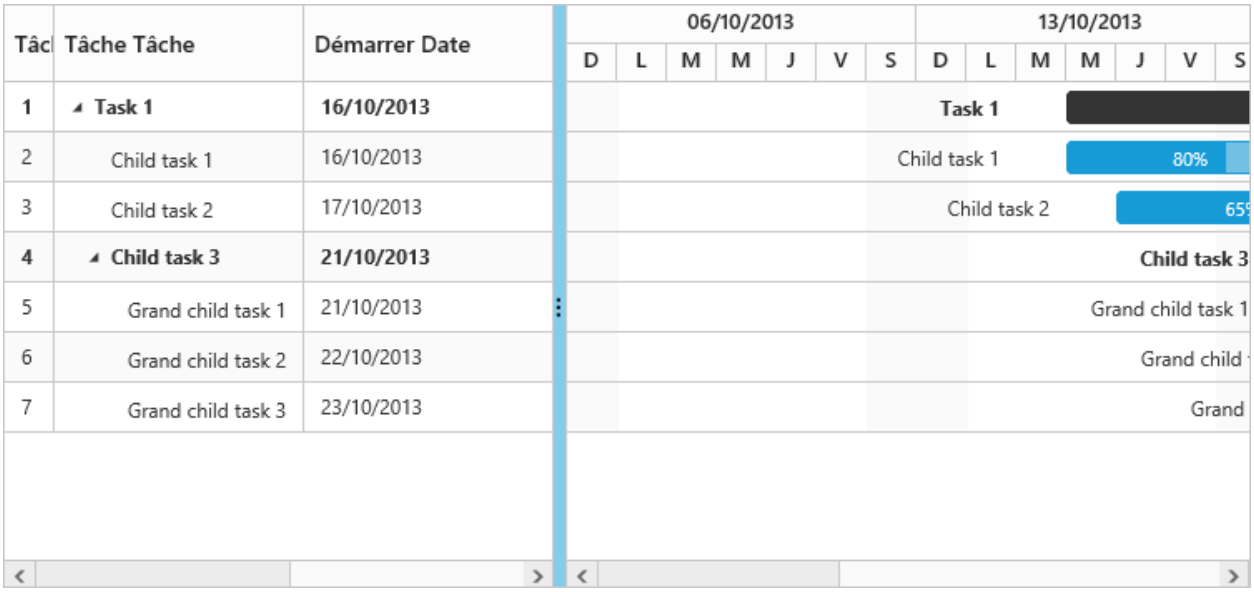
Date format

The default date format used in the Gantt control is “MM/dd/yyyy”. Date formats will be changed based on the culture referred in the control. But if you still need to change the date format we can define the desired format using the dateFormat property. The same have been explained in the following code example.

JAVASCRIPT

```
@(Html.EJ().Gantt("Gantt")
//…
.DateFormat("dd/MM/yyyy")
.ScheduleHeaderSettings(ScheduleHeader =>
ScheduleHeader.WeekHeaderFormat("dd/MM/yyyy"))
.Locale("fr-FR")
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Keyboard navigation

Gantt control supports the following keyboard shortcuts for performing various actions and keyboard navigation can be enabled/disabled by AllowKeyboardNavigation property

Alt + j : To focus the Gantt control.

Home : To select the first row.

End : To select the last row.

F2 : For editing the focused cell of the selected row.

Enter : To save the edited cell.

Delete : To delete the selected row.

Ctrl + Down Arrow : To expand all the parent tasks.

Ctrl + Up Arrow : To collapse all the parent tasks.

Left Arrow : To collapse the selected parent task.

Right Arrow : To expand the selected parent task.

Click here to view the sample for keyboard navigation.

Tooltip

The Gantt has support to display tooltip for both taskbars and for column cells.

Taskbar and dependency line tooltip

In Gantt, you can enable or disable taskbar and dependency line mouse hover tooltip by using the EnableTaskbarTooltip property. The default value of this property is true. The following code example shows, how to enable the taskbar and dependency line tooltip in Gantt.

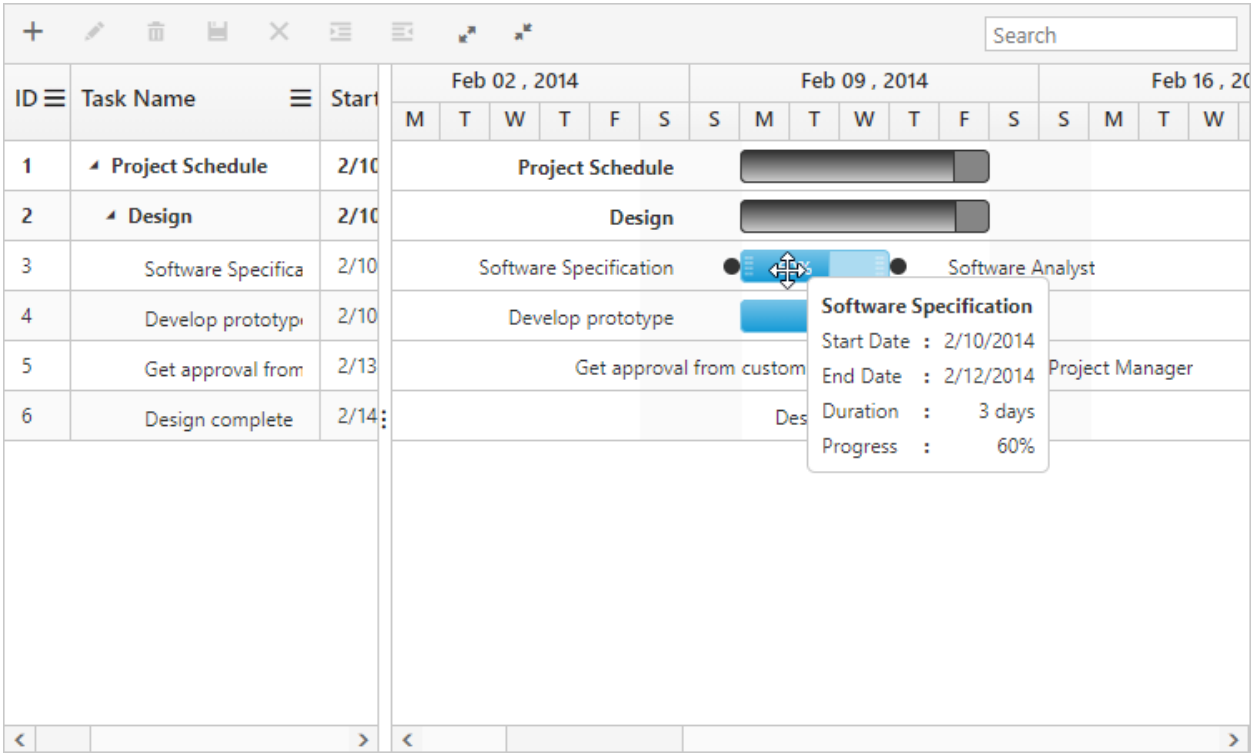
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EnableTaskbarTooltip(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

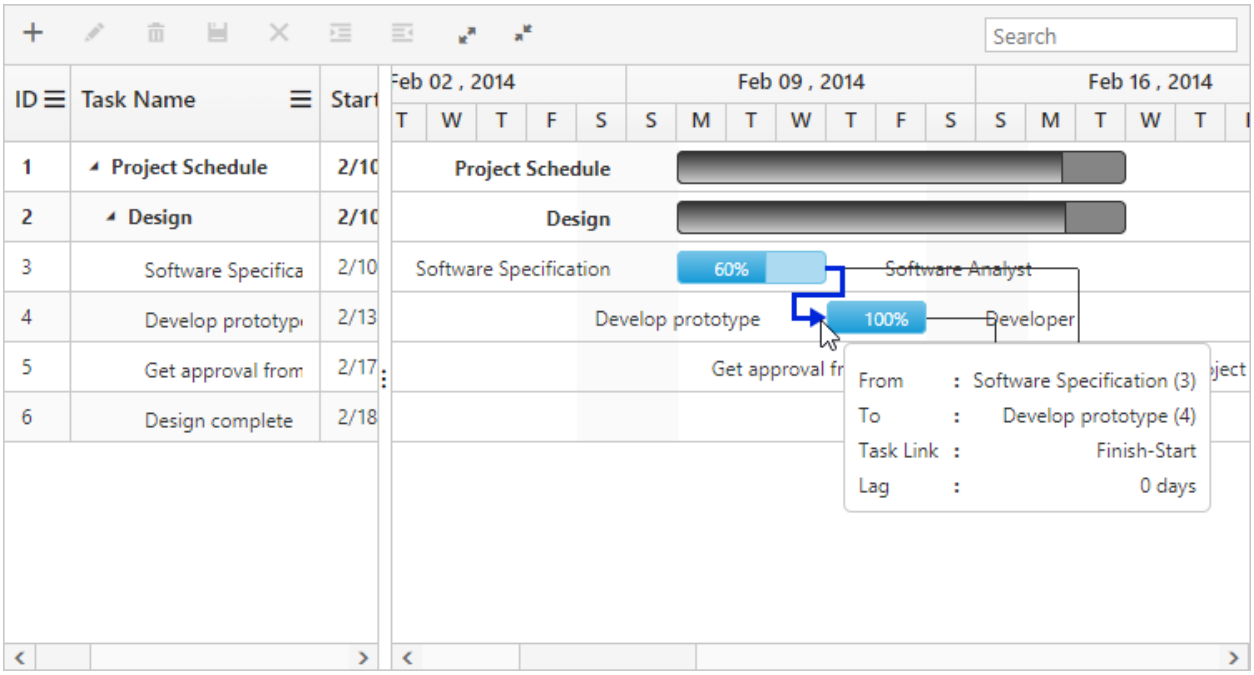
```
.PredecessorMapping("Predecessors")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Taskbar tooltip can be customized by using the TaskbarTooltipTemplate property and dependency line tooltip can be customized by using the PredecessorTooltipTemplate property, these properties are described briefly in the customization section.



Taskbar Tooltip

Copyright © 2001 - 2020 Syncfusion Inc.



Dependency Tooltip

Taskbar drag tooltip

It is possible to enable or disable the tooltip while performing editing actions on taskbar (left resizing, right resizing, dragging and progress resizing) by using the `EnableTaskbarDragTooltip` property. By default, this property is set to `true`. The following code example explains this behavior.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.EnableTaskbarDragTooltip(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Cell tooltip

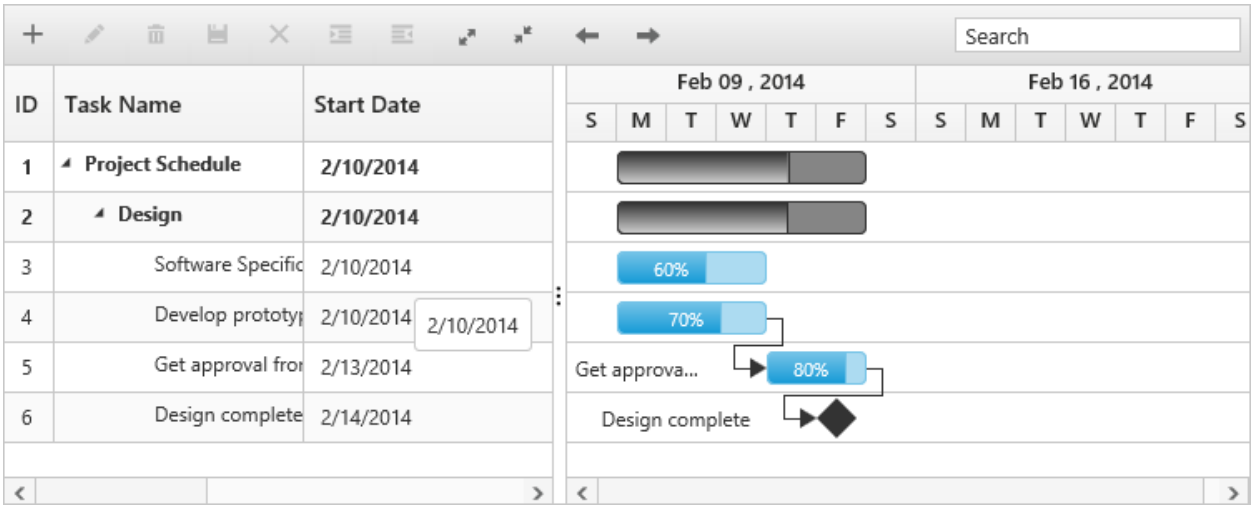
It is possible to enable or disable the TreeGrid cell tooltip in mouse hover by using the `ShowGridCellTooltip` property. By default, this property is set to `true`. The following code example explains how to enable disable this property.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ShowGridCellTooltip(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

This tooltip can be customized using the `CellTooltipTemplate` property, which is described briefly in the customization section.

Copyright © 2001 - 2020 Syncfusion Inc.

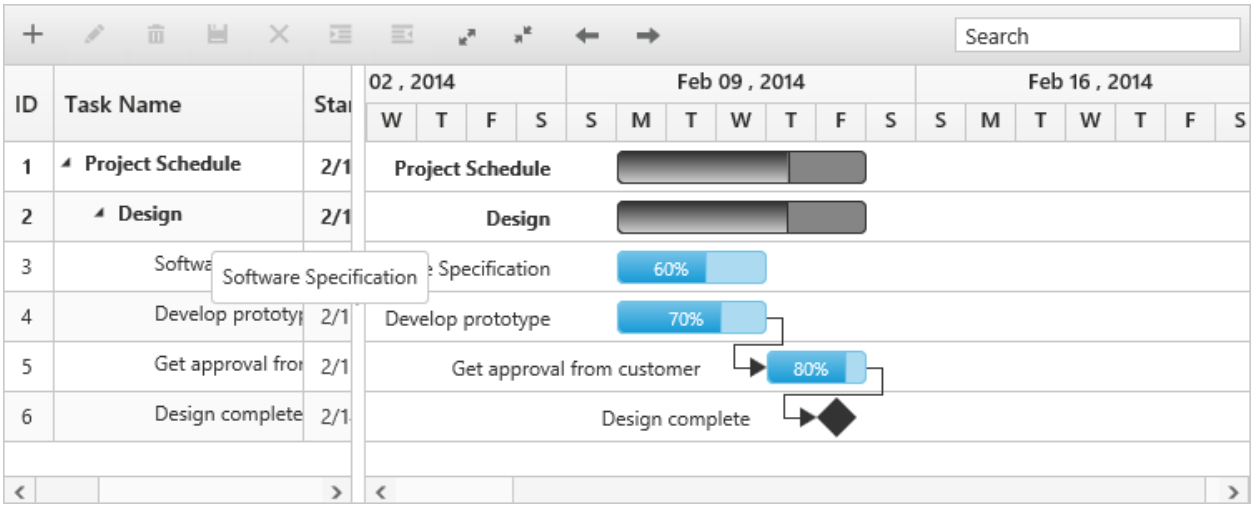


Tree column (Expander column) tooltip

It is also possible to display tooltip only for expander column by setting the ShowGridExpandCellTooltip property. The following code example shows you to enable expander column tooltip in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//...
.ShowGridExpandCellTooltip(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```



Duration Units

In Gantt, the task's duration value can be measured by the following duration units,

- Day
- Hour
- Minute

Copyright © 2001 - 2020 Syncfusion Inc.

Each task in the project can be defined with different duration units and the duration unit of a task can be defined by the following ways,

1. Using DurationUnitMapping property, to map the duration unit data source field.
2. Defining the duration unit value along with the duration field in the data source.

Mapping the Duration Unit field

The below code snippet explains the mapping of duration unit data source field to the Gantt control using the DurationUnitMapping property.

ASPX-CS

```
@(Html.EJ().Gantt("ganttsample1")
//..
.DurationMapping("Duration")
.DurationUnitMapping("DurationUnit")
)
@(Html.EJ().ScriptManager())
```

C#

```
public class GanttController : Controller
{
//
// GET: /DurationUnits/
public ActionResult GanttDurationUnits()
{
ViewBag.DataSource = GetData();
return View();
}
public static List<GanttDuratioUnitData> GetData()
{
List<GanttDuratioUnitData> list = new List<GanttDuratioUnitData>();
list.Add(new GanttDurationUnitData()
{
TaskID = 1,
TaskName = "Parent Task 1",
StartDate = "02/27/2017",
Duration = "5day",
Progress = 40,
Children = (new List<GanttDurationUnitData>()
{
new GanttDurationUnitData()
{
TaskID = 2,
TaskName = "Child Task 1",
StartDate = "02/27/2017",
Duration = "5d",
Progress = 40
},
new GanttDurationUnitData()
{
TaskID = 3,
TaskName = "Child Task 2",
StartDate = "02/27/2017",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Duration = "40",
DurationUnit="hour",
Progress = 40,
},
new GanttDurationUnitData()
{
TaskID = 4,
TaskName = "Child Task 3",
StartDate = "02/27/2017",
Duration = "2400",
DurationUnit = "minute",
Progress = 40
}
})
//..
});
return list;
}
public class GanttDurationUnitData
{
//..
public string Duration { get; set; }
public string DurationUnit { get; set; }
}
}
```

Note: 1. The default value of the DurationUnit property is GanttDurationUnit.Day.

Defining duration unit along with duration field

Duration units for the tasks can also be defined along with the duration values, the below code snippet explains the duration unit for a task along with duration value,

ASPX-CS

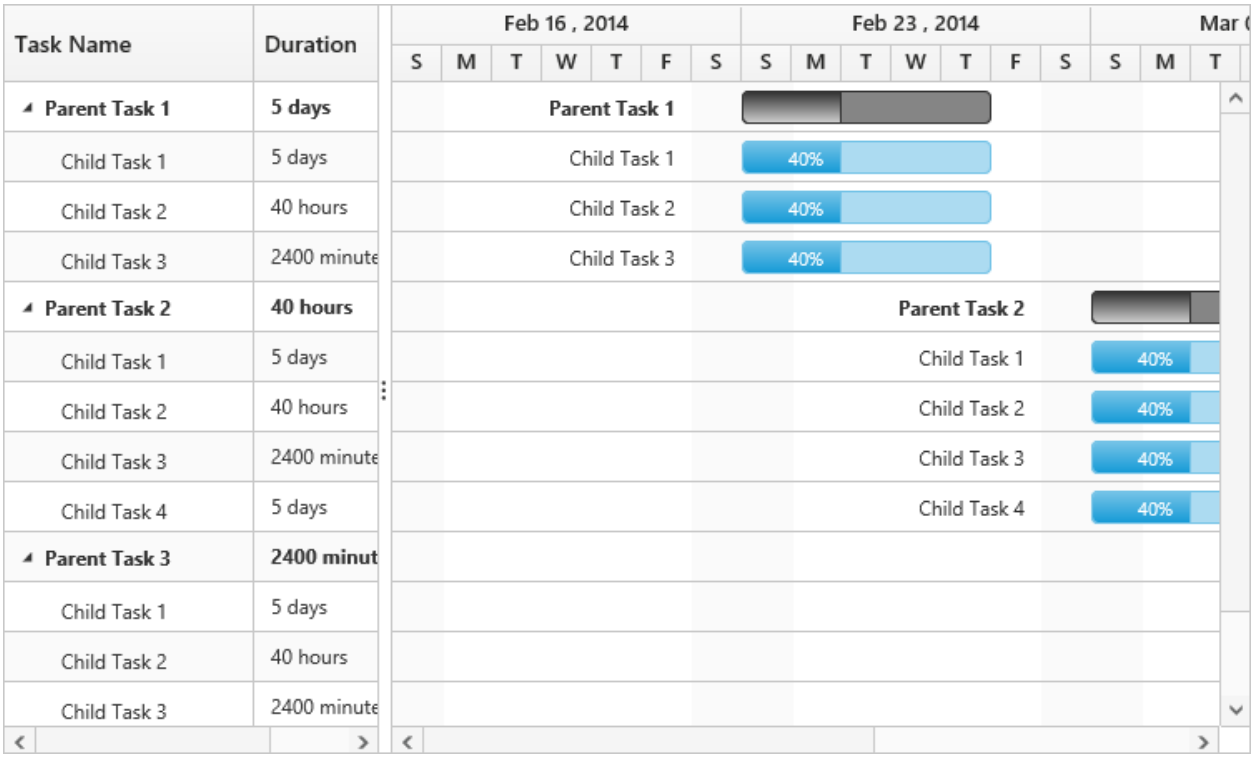
```
@(Html.EJ().Gantt("ganttsample1")
//..
.DurationMapping("Duration")
)
@(Html.EJ().ScriptManager())
```

The below sample explains the task's duration unit in Gantt.

[Duration Unit]

The below screen shot shows different duration unit support in Gantt control.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: 1. The edit type of the duration column in Gantt is string, to support editing the duration field along with duration units.

Localizing the duration unit

We can localize the duration unit texts like below,

ASPX-CS

```
ej.Gantt.Locale["en-US"] = {
//...
durationUnitTexts: {
days: "days",
hours: "hours",
minutes: "minutes",
day: "day",
hour: "hour",
minute: "minute"
},
durationUnitEditText: {
minute: ["m", "min", "minute", "minutes"],
hour: ["h", "hr", "hour", "hours"],
day: ["d", "dy", "day", "days"]
}
};
```

Note: **durationUnitTexts** property is used to display the unit of duration in columns and taskbar tooltips.

durationUnitEditText property is used to save the edited duration unit value on editing action.

Click here to view the online demo sample for mapping the duration units.

Copyright © 2001 - 2020 Syncfusion Inc.

Working Time Range

In Gantt control, working hours in a day for a project can be defined by using the DayWorkingTime property. Based on the working hours, automatic date scheduling and duration validations for a task are performed.

The below code snippet explains on how to define the working time range for the project in Gantt,

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
// ...
.DayWorkingTime(dt=>
{
dt.From("08:00 AM").To("12:00 PM").Add();
dt.From("01:00 PM").To("05:00 PM").Add();
})
)
@(Html.EJ().ScriptManager())
```

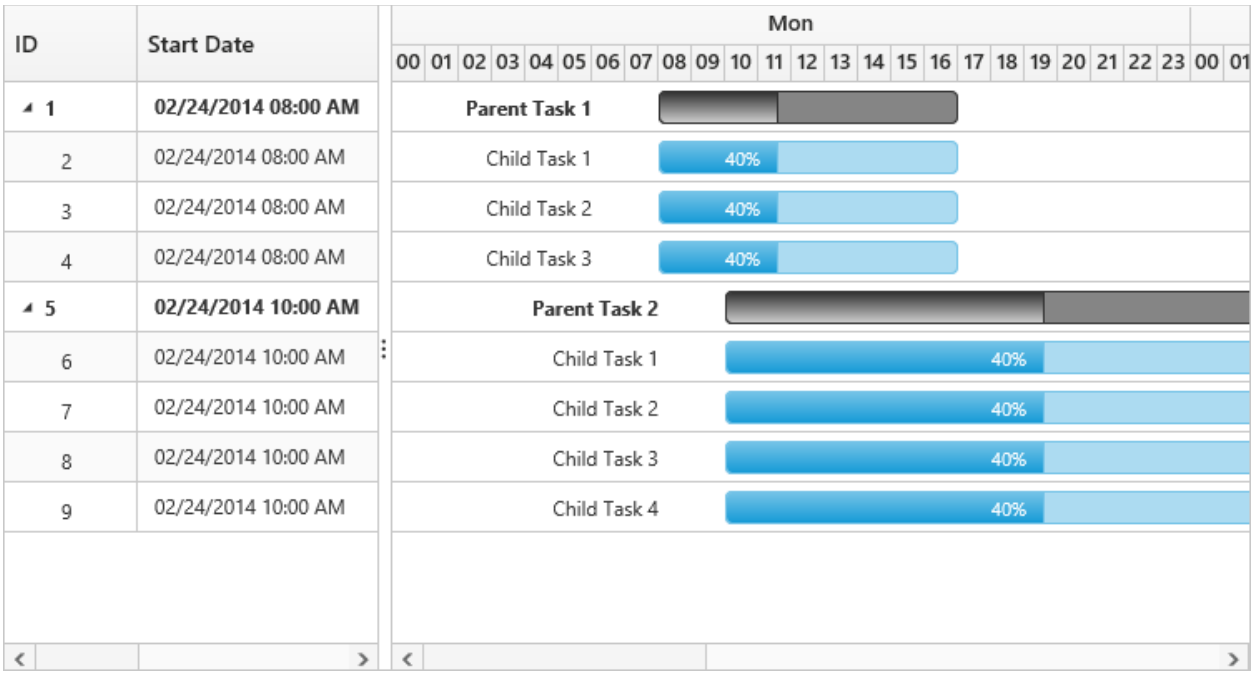
Note: 1. Individual tasks can lie between any time within the defined working time range of the project.

Note: 2. DayWorkingTime property is used to define the working time for the whole project.

The below demo explains the working time range in Gantt.

Working Time Range

The following screen shot shows working time range in Gantt control.



Highlight working time range

Highlight the working time range with a background color by using the DayWorkingTime.Background property. You can highlight the non-working time ranges in a day. To do this, set the

Copyright © 2001 - 2020 Syncfusion Inc.

HighlightNonWorkingTime property to true. To customize the non-working time background, use the NonWorkingBackground property.

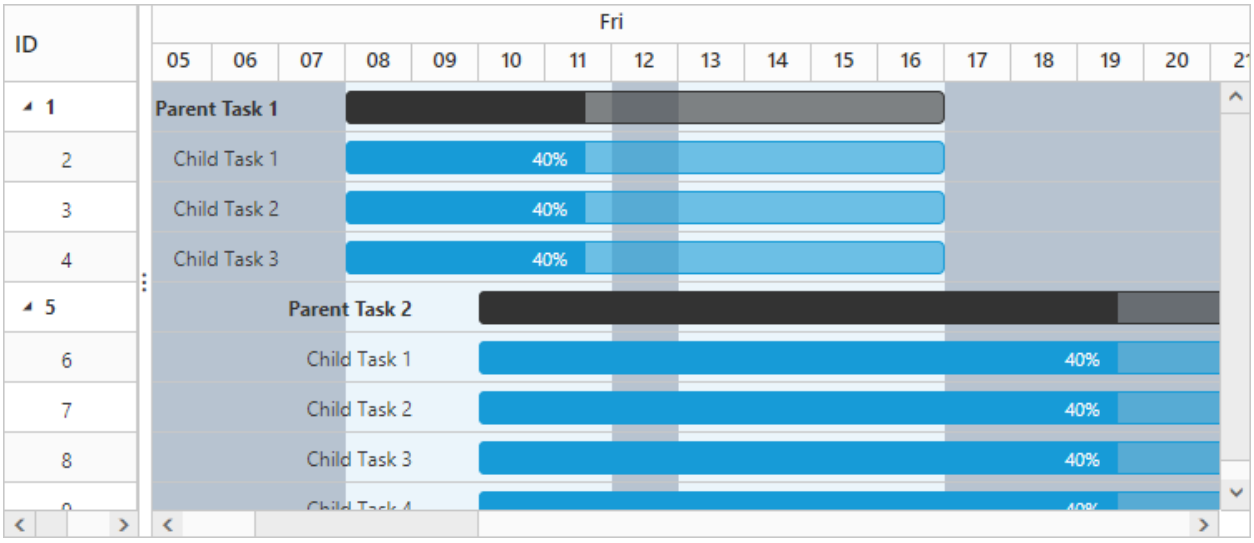
The following code snippet explains how to define the working time range with background in Gantt.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
// ...
.HighlightNonWorkingTime(true)
.NonWorkingBackground("#B7C3D0")
.DayWorkingTime(dt => {
dt.From("08:00 AM").To("12:00 PM").Background("#EBF5FB").Add();
dt.From("01:00 PM").To("05:00 PM").Background("#EBF5FB").Add();
})
)
@(Html.EJ().ScriptManager())
```

Note: The background colors of working time range are highlighted only when the ScheduleHeaderSettings.ScheduleHeaderType value as day.

The following screenshot shows the working time range with background colors.



Appearance and Styling

The look and feel of the Gantt control can be customized by applying themes and formatting the schedule header.

Schedule Header Customization

Schedule Header Unit Format

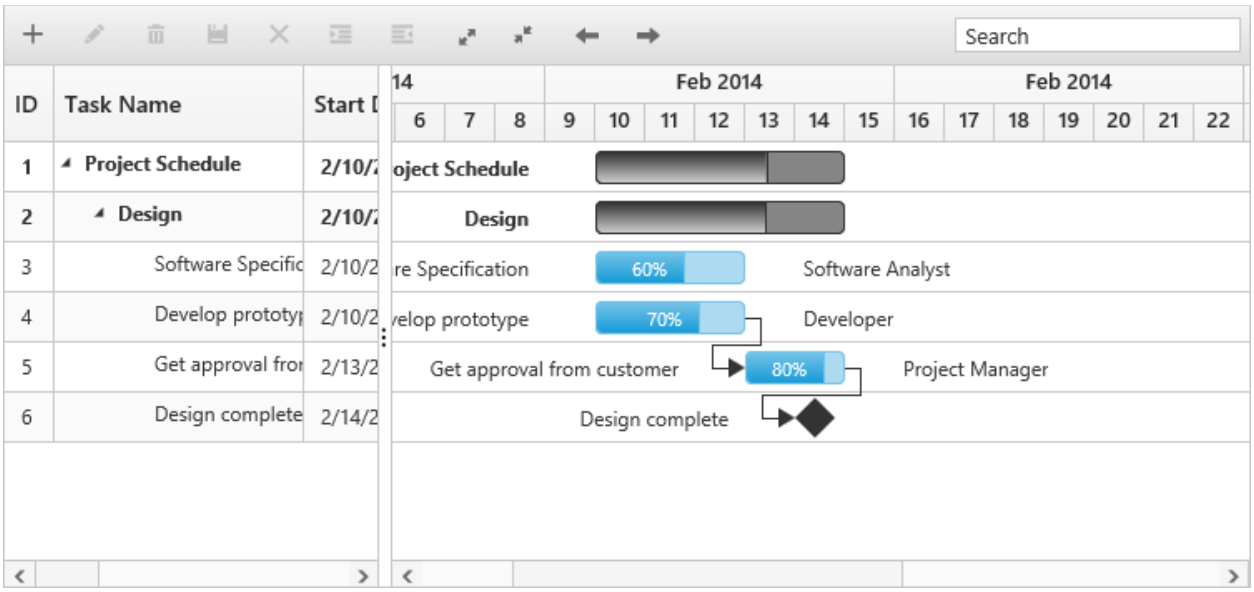
You can change the format of schedule headers in various timescale modes by using DayHeaderFormat, HourHeaderFormat, WeekHeaderFormat, MonthHeaderFormat and YearHeaderFormat properties available in ScheduleHeaderSettings property.

And you can change the background of weekends available in timescale by using WeekendBackground property, please refer the following code example.

ASPX-CS

```
@(Html.EJ().Gantt("GanttContainer")
//...
.ScheduleHeaderSettings(sh=>
{
sh.WeekHeaderFormat("MMM yyyy");
sh.DayHeaderFormat("d");
sh.HourHeaderFormat("HH");
sh.MonthHeaderFormat("MMM");
sh.YearHeaderFormat("yyyy");
sh.WeekendBackground("#F2F2F2");
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the customized format schedule header in Gantt control.



Schedule Header Unit Width

Schedule header units width value can be customized by using ScheduleHeaderSettings.TimescaleUnitSize property. The default value of this property was 100%, we can set value for this property from 50% to 500%.

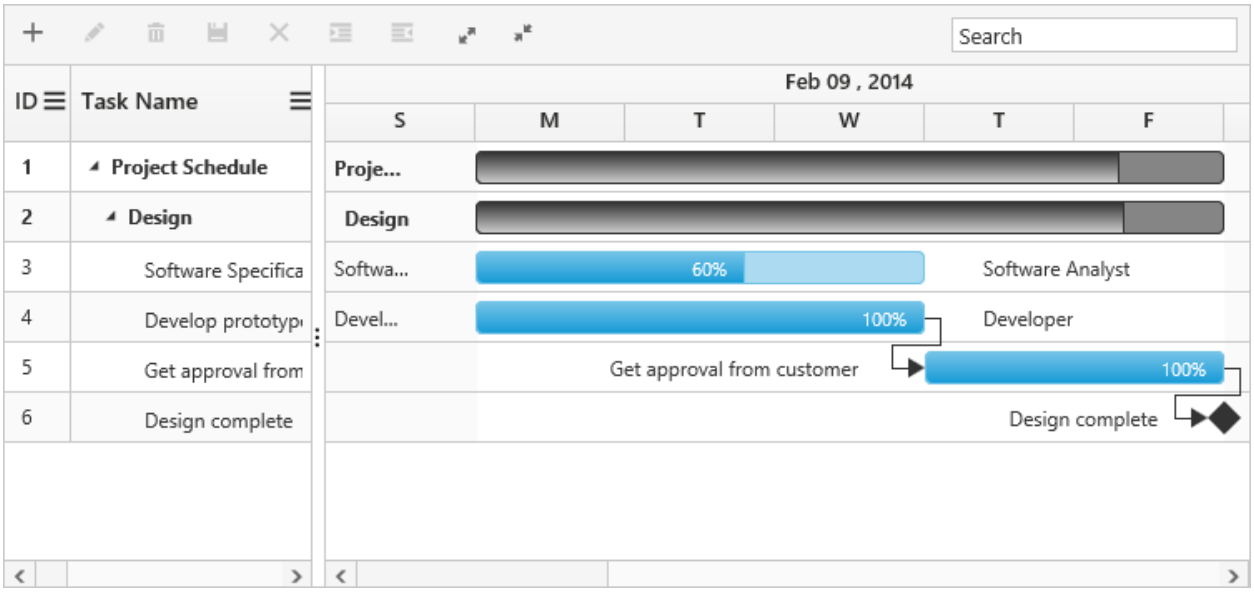
The following code example shows how to use this properties.

ASPX-CS

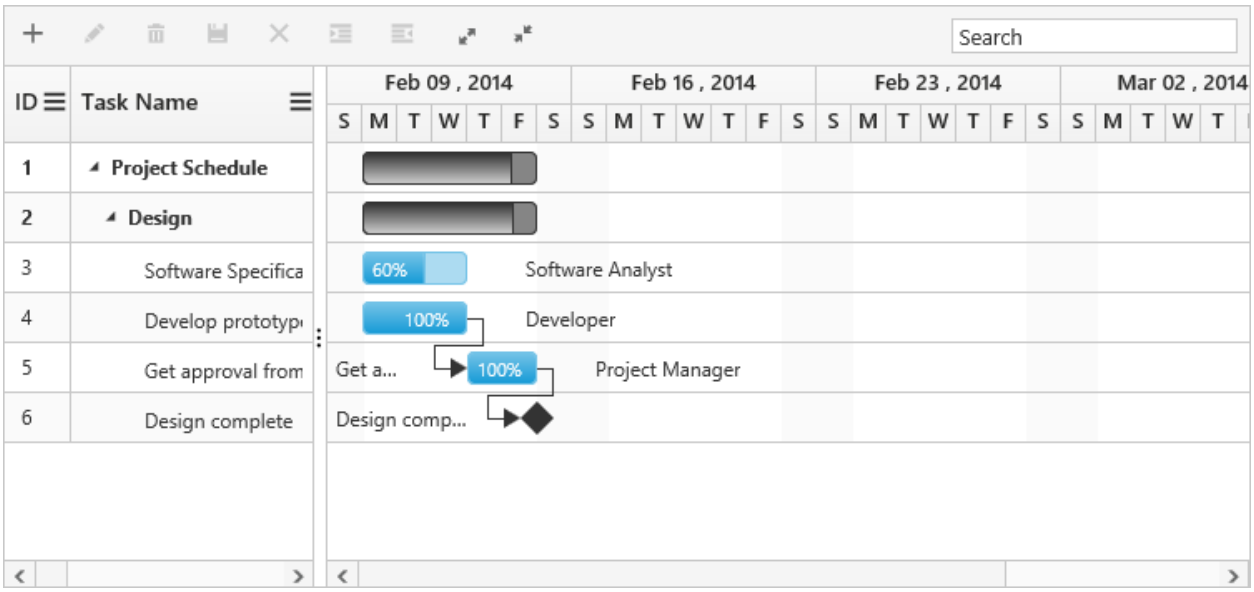
```
@(Html.EJ().Gantt("Gantt")
//..
.ScheduleHeaderSettings(sh=>
{
sh.TimescaleUnitSize("70%");
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Schedule header units with 300% width value



Schedule header units with 70% width value

Taskbar Customization

Taskbar Background

Background color of child taskbars and parent taskbars can be customized by using TaskbarBackground and ParentTaskbarBackground properties. The following code example shows how to use this properties.

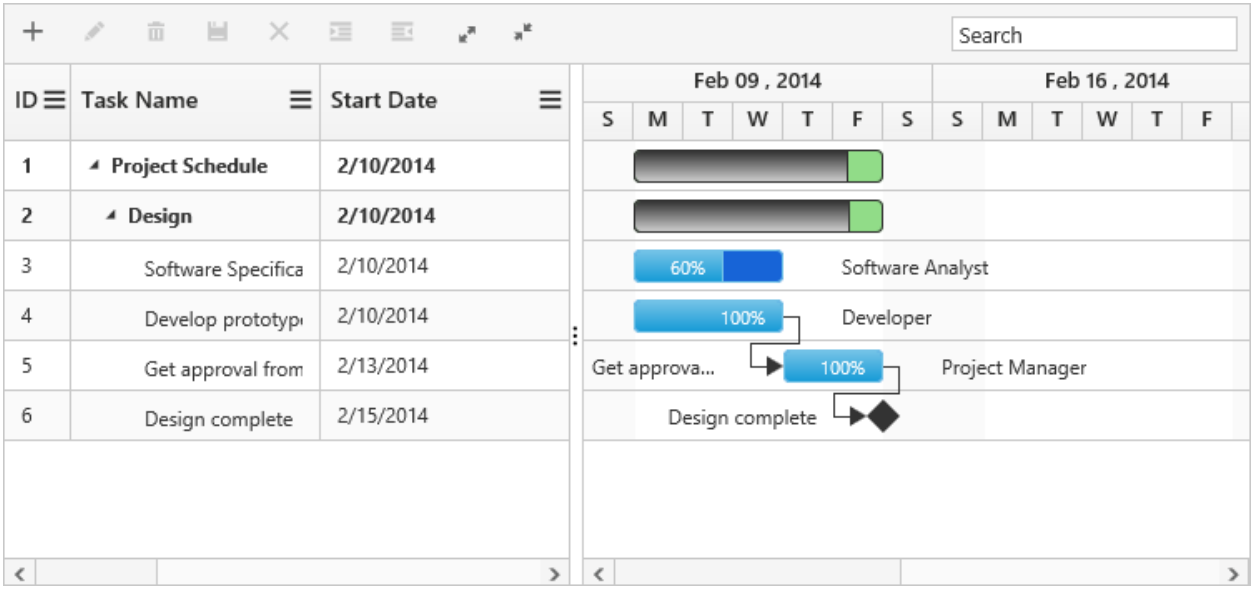
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.TaskbarBackground("#1764d7")
.ParentTaskbarBackground("#91dc88")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
```

The following screenshot shows the customized parent and child taskbars in Gantt.



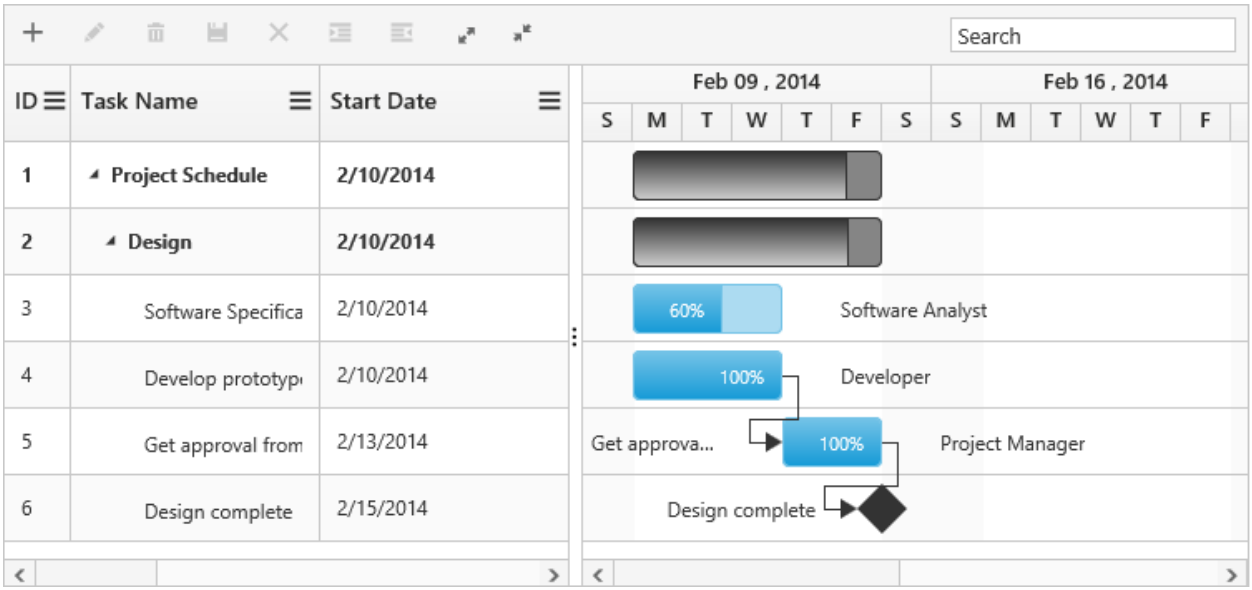
Taskbar Height

Height of child taskbars and parent taskbars can be customized by using TaskbarHeight property. The following code example shows how to use the property.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.TaskbarHeight(30)
.RowHeight(40)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of above code example.



Note: TaskbarHeight value should be lower than RowHeight property value.

Progressbar Customization

Background color of child task's progress bar and parent task's progress bar can be customized by using ProgressbarBackground and ParentProgressbarBackground properties. Progress bars height can be changed by using ProgressbarHeight property. The visibility of progress status label inside the taskbars can be changed by using ShowProgressStatus property.

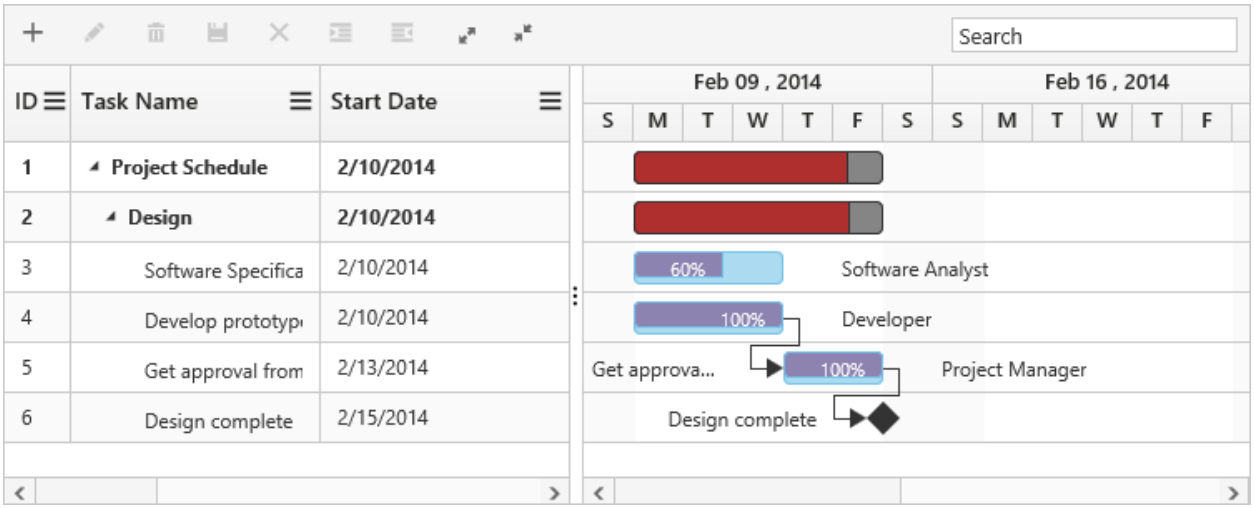
The following code example shows how to use this properties.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.ProgressbarBackground("#8c83b1")
.ParentProgressbarBackground("#af2f2f")
.ProgressbarHeight(80)
.ShowProgressStatus(true)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the customized progress bar of parent and child tasks in Gantt.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: ProgressbarHeight property value should be in 0 to 100, because this value was considered as percentage value of taskbar height value.

Conditional Formatting

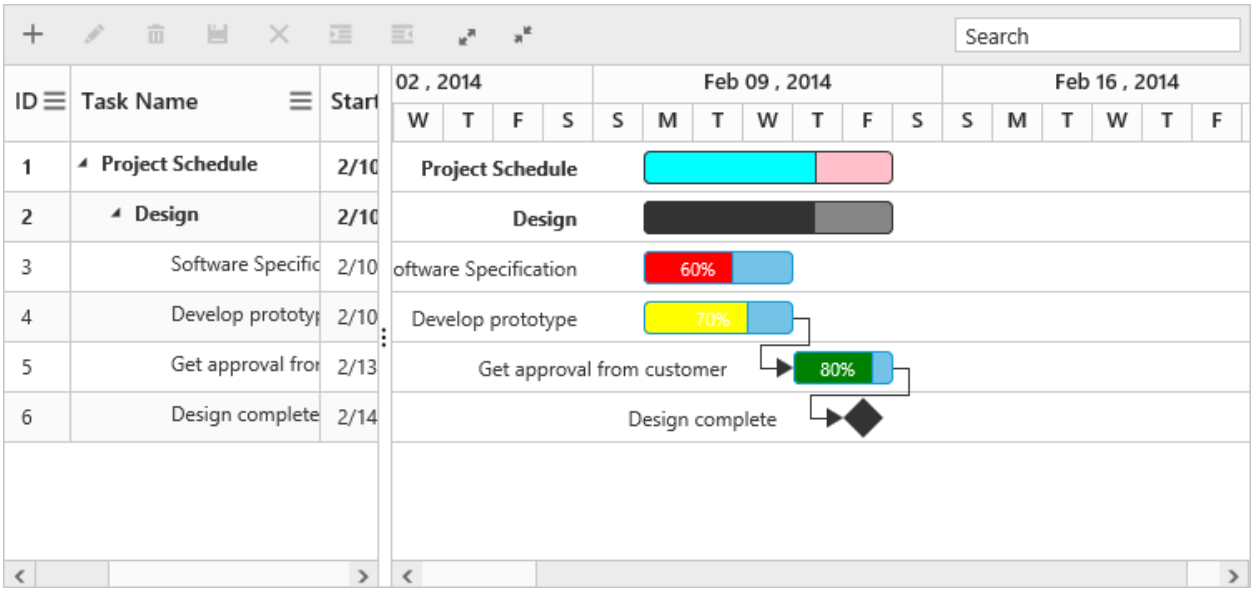
The Taskbar can be customized based on the task information in Gantt control by using QueryTaskbarInfo event. The following code example shows how to customize the Taskbar in Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ClientSideEvents(eve =>
{
eve.QueryTaskbarInfo("queryTaskbarInfo");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function queryTaskbarInfo(args){
//queryTaskbarInfo will be triggered when a taskbar is rendered
if (args.data.level === 0) {
args.parentTaskbarBackground = "pink";
args.parentProgressbarBackground = "cyan";
} else {
if (args.data.status == "60") {
args.progressbarBackground = "red";
} else if (args.data.status == "70") {
args.progressbarBackground = "yellow";
} else if (args.data.status == "80") {
args.progressbarBackground = "green";
}
}
}
</script>
```

The following screenshot shows the customized taskbar in Gantt control.

Copyright © 2001 - 2020 Syncfusion Inc.



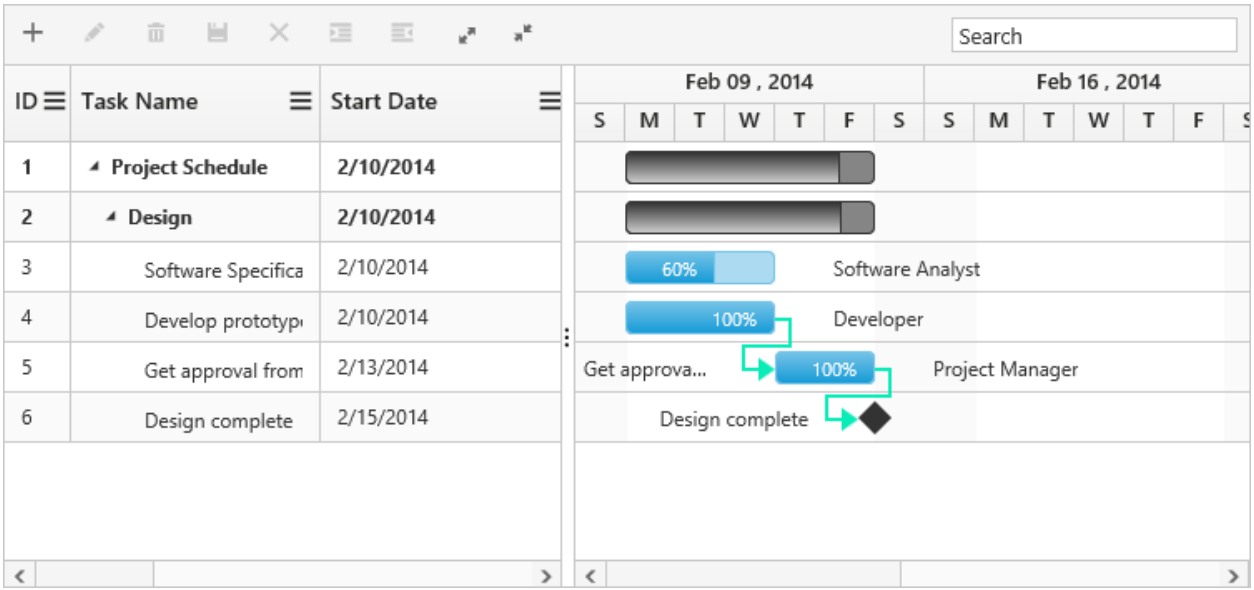
Dependency line customization

Width and background color of dependency line in Gantt can be customized by using `ConnectorlineWidth` and `ConnectorLineBackground` properties. The following code example shows how to use this properties.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ConnectorlineWidth(2)
.ConnectorLineBackground("#0aecb8")
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the customized dependency lines in Gantt.



Copyright © 2001 - 2020 Syncfusion Inc.

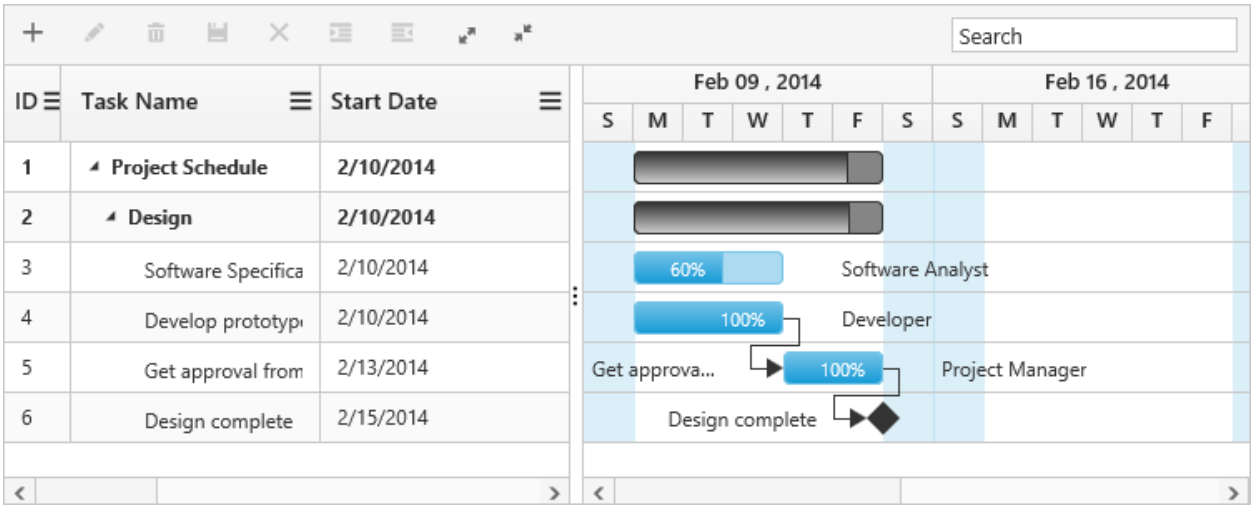
Weekend background

Background color of weekends in Gantt can be changed by using weekendBackground property. The following code example shows how to use this properties.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
//..
.WeekendBackground("rgba(116, 195, 231, 0.26)")
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the customized dependency lines in Gantt.



Themes

The following are the types of themes available in Gantt control.

1.Flat Azure 2.Flat Azure Dark 3.Flat Lime 4.Flat Lime Dark 5.Flat Saffron 6.Flat Saffron Dark 7.Gradient Azure 8.Gradient Azure Dark 9.Gradient Lime 10.Gradient Lime Dark 11.Gradient Saffron 12.Gradient Saffron Dark 13.Bootstrap 14.High Contrast 01 15.High Contrast 02 16.Material 17.Office-365

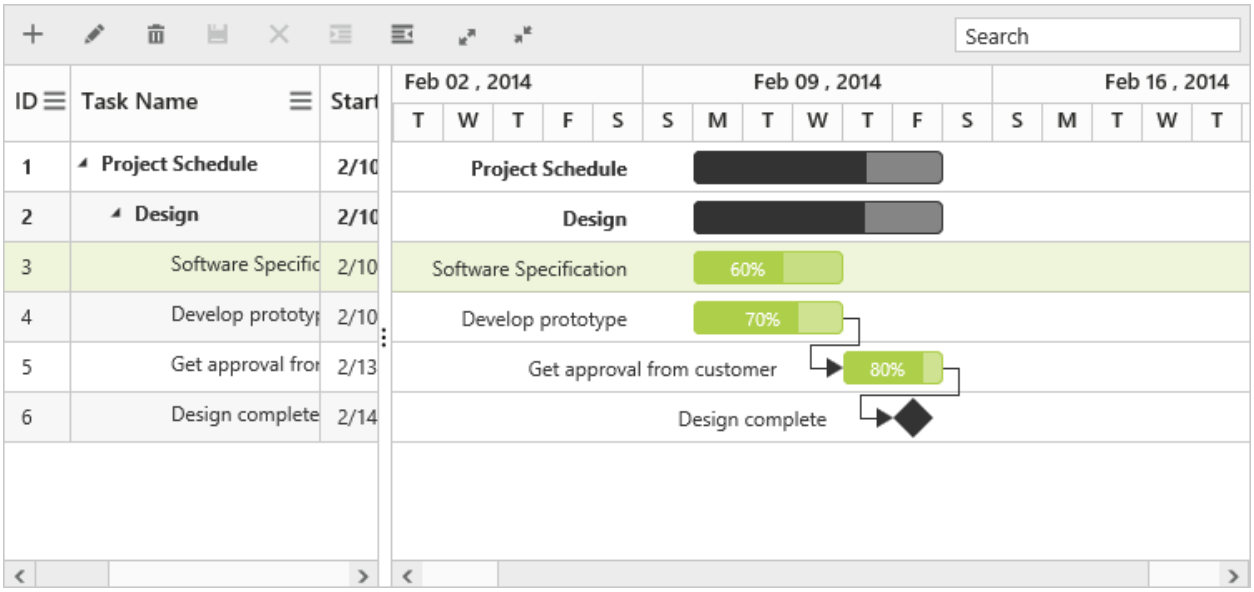
The theme (Gradient lime) can be applied to the Gantt control by using the style sheet from the online link as follows.

HTML

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Getting Started with Gantt Control for JavaScript</title>
<!-- style sheet for default theme(gradient lime) -->
<link href="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
//...
</html>
```

The following screenshot shows the Gantt control with Gradient-lime theme.

Copyright © 2001 - 2020 Syncfusion Inc.



Configuring CSS Class

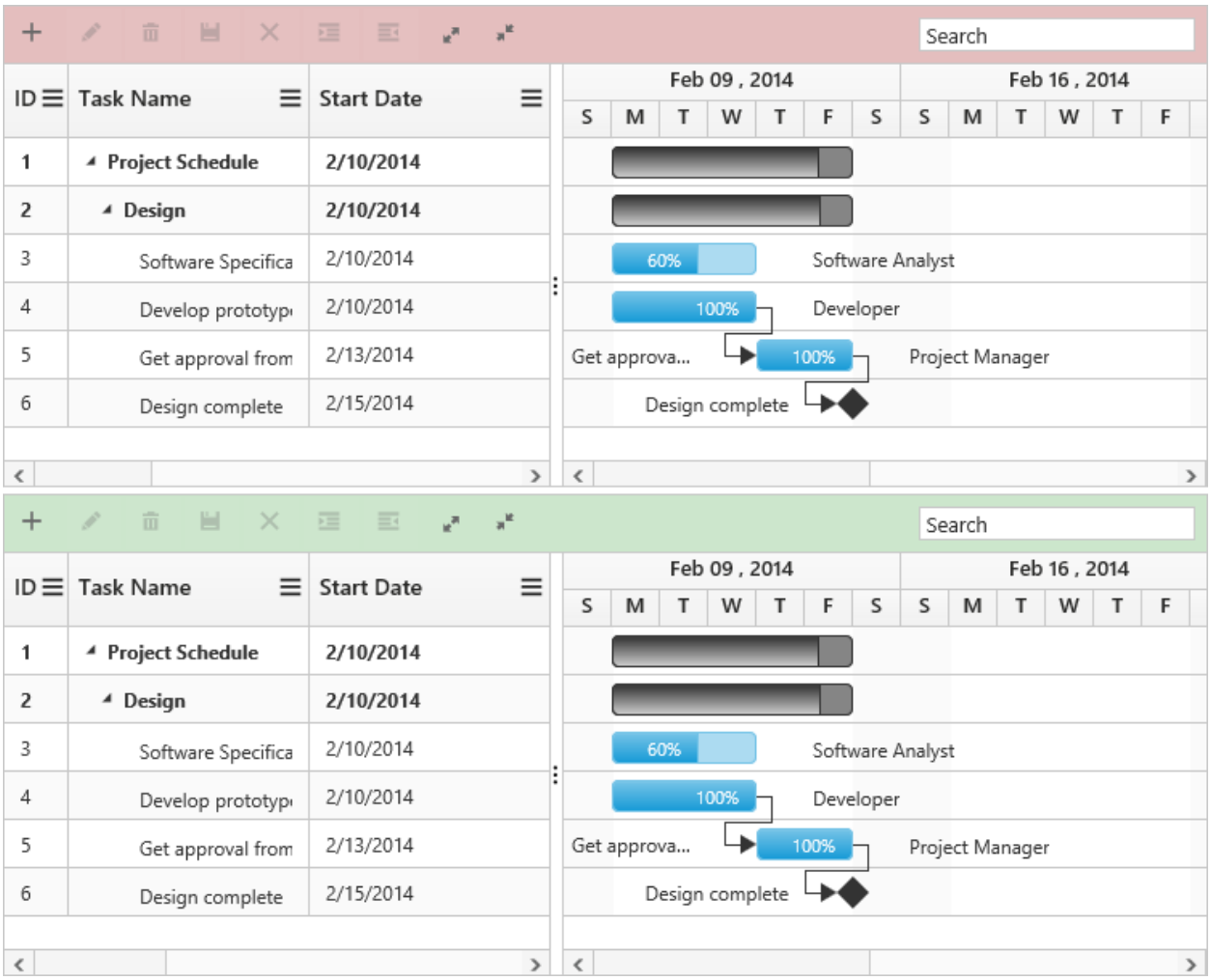
In Gantt CssClass property was used to apply different customized styles to multiple Gantt controls available in same page. The following code example shows how to apply different background color for each Gantt control's toolbar element.

HTML

```
<style>
.c-class1.e-gantt .e-toolbar {
background-color: rgba(169, 45, 45, 0.31);
}
.c-class2.e-gantt .e-toolbar {
background-color: rgba(0, 128, 0, 0.2);
}
</style>
@(Html.EJ().Gantt("GanttContainer")
.CssClass("c-class1")
)
@(Html.EJ().Gantt("GanttContainer1")
.CssClass("c-class2")
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Customize rows and cells

While rendering the rows in Grid part of Gantt RowDataBound event and QueryCellInfo event will be triggered for each rows and cells. Using this event we can customize the rows and cells. The below code example shows how to customize the cell and row element using this events.

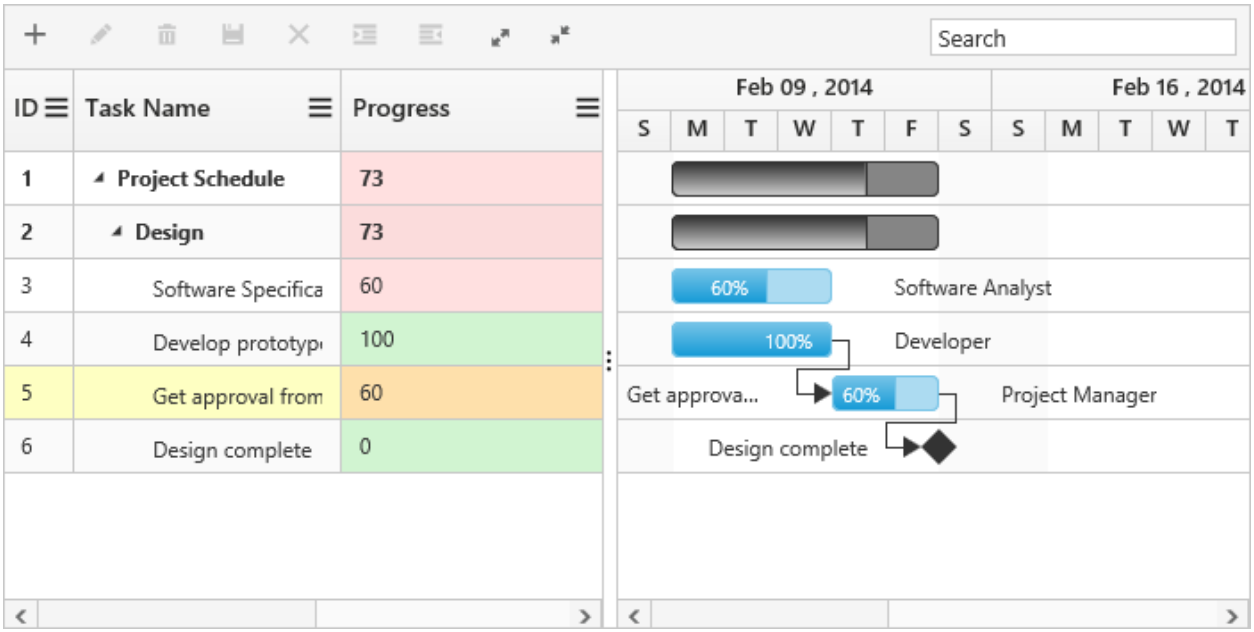
ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ClientSideEvents(eve =>
{
eve.QueryCellInfo("queryCellInfo");
eve.RowDataBound("rowDataBound");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function queryCellInfo(args) {
if (args.column.mappingName == "progress") {
if (args.data.item["progress"] < 80)
$(args.cellElement).css("background-color", "rgba(255, 0, 0, 0.12)");
else
$(args.cellElement).css("background-color", "rgba(86, 226, 86, 0.25)");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
function rowDataBound(args) {
if (args.data.item["taskID"] == 5)
$(args.rowElement).css("background-color", "rgba(251, 255, 0, 0.24)");
}
</script>
```

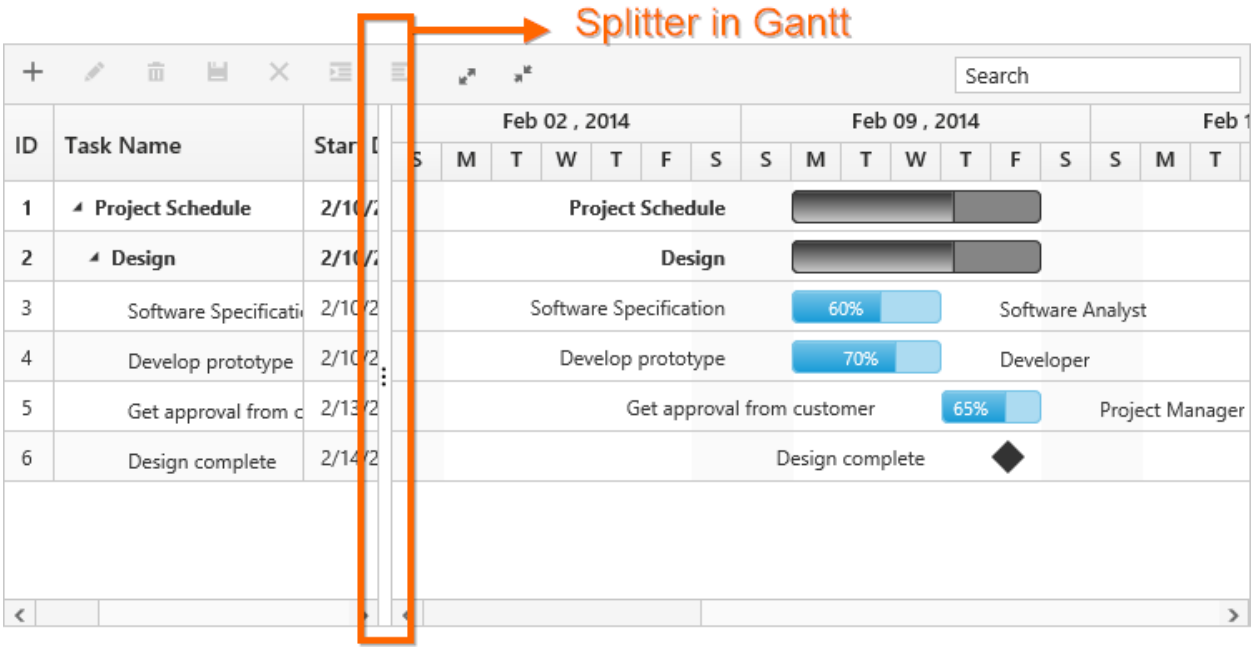
The below screenshot shows the output of above code example.



How to

Change Splitter position

In Gantt control, Splitter separates the TreeGrid section from the Chart section.



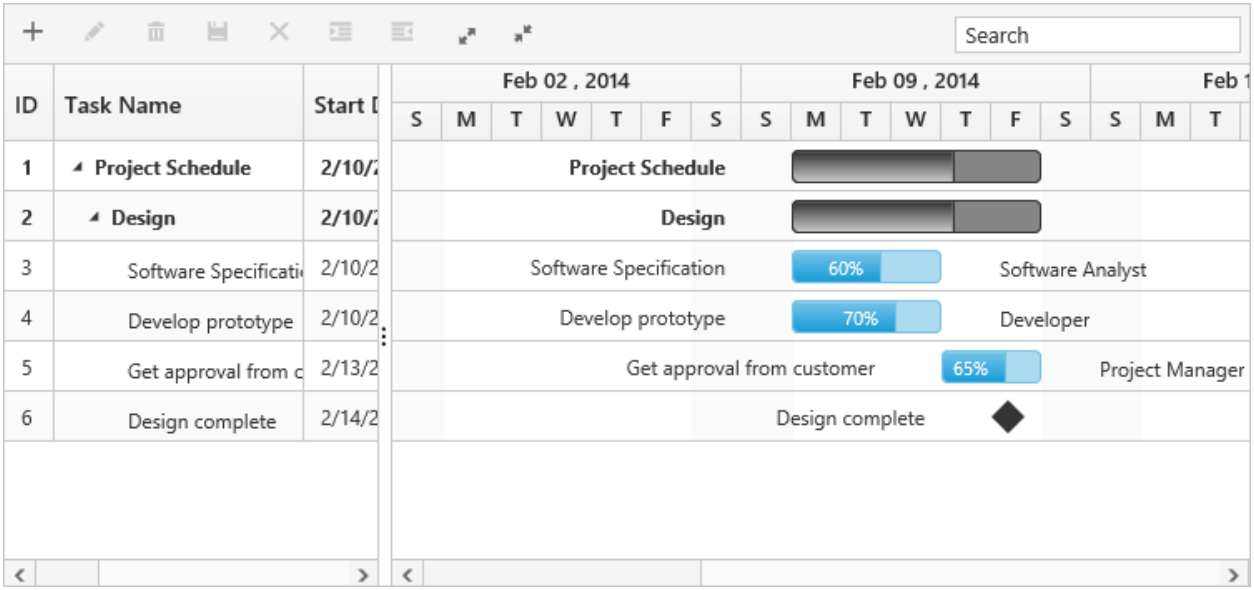
Copyright © 2001 - 2020 Syncfusion Inc.

It is possible to change the position of the Splitter while loading the Gantt by using the SplitterPosition property, thereby varying the width of the TreeGrid and Chart sections in the control. SplitterPosition property denotes the percentage of the TreeGrid section's width to be rendered and this property supports both pixels and percentage values.

The following code example explains how to define the SplitterPosition property in the Gantt.

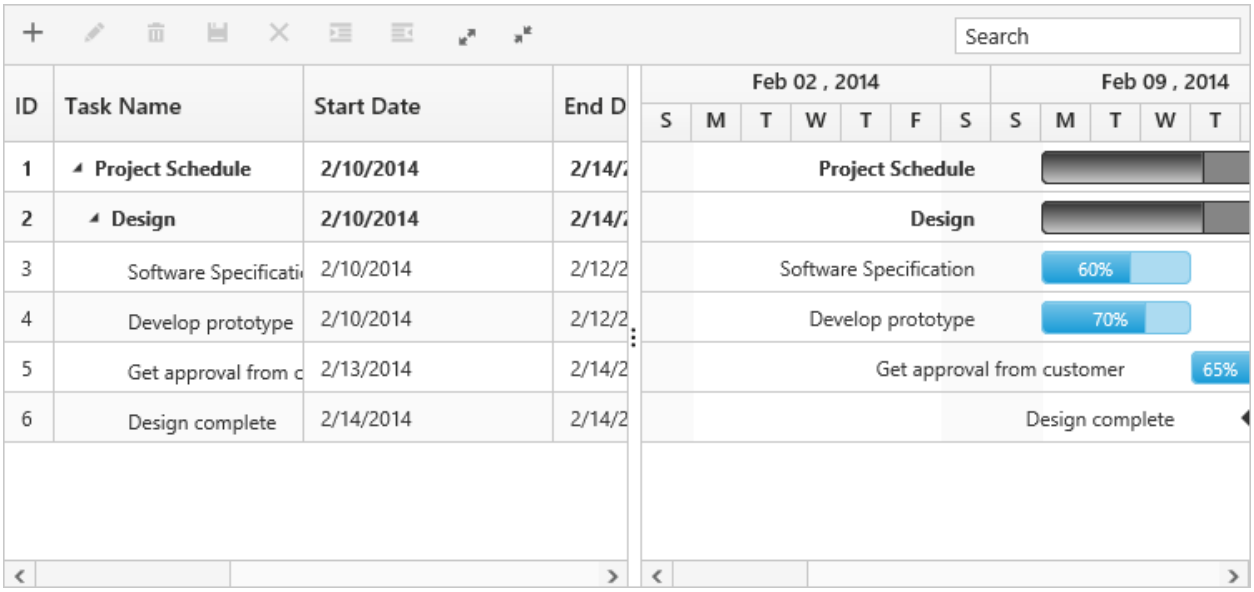
ASPX-CS

```
@(Html.EJ().Gantt("gantt")
.SplitterPosition("50%")
//also you can define with pixel value as
//.SplitterPosition(”650”) (or) .SplitterPosition(”650px”)
)
@(Html.EJ().ScriptManager())
```

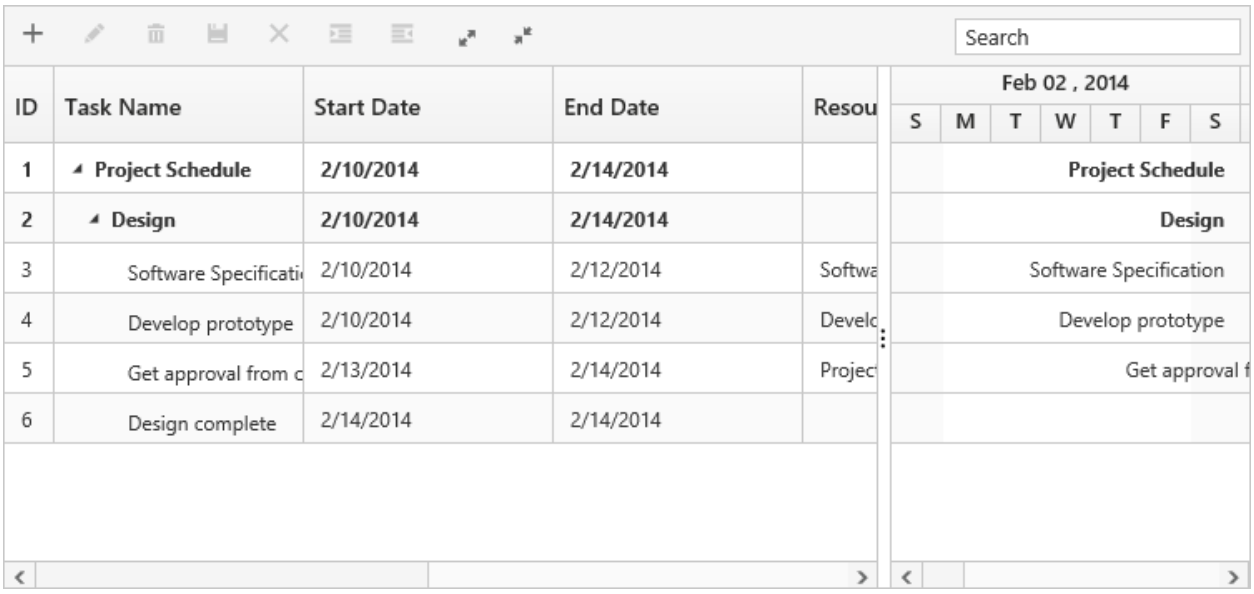


Gantt with 30 % splitter position

Copyright © 2001 - 2020 Syncfusion Inc.



Gantt with 50% splitter position



Gantt with 600px splitter position

Change workweek/working days in a week

Non-working days/weekend are used to represent the non-productive days in a project. It is possible to change the non-working days in a week using the `WorkWeek` property in Gantt.

By default, Saturdays and Sundays are considered as non-working days/weekend in a project.

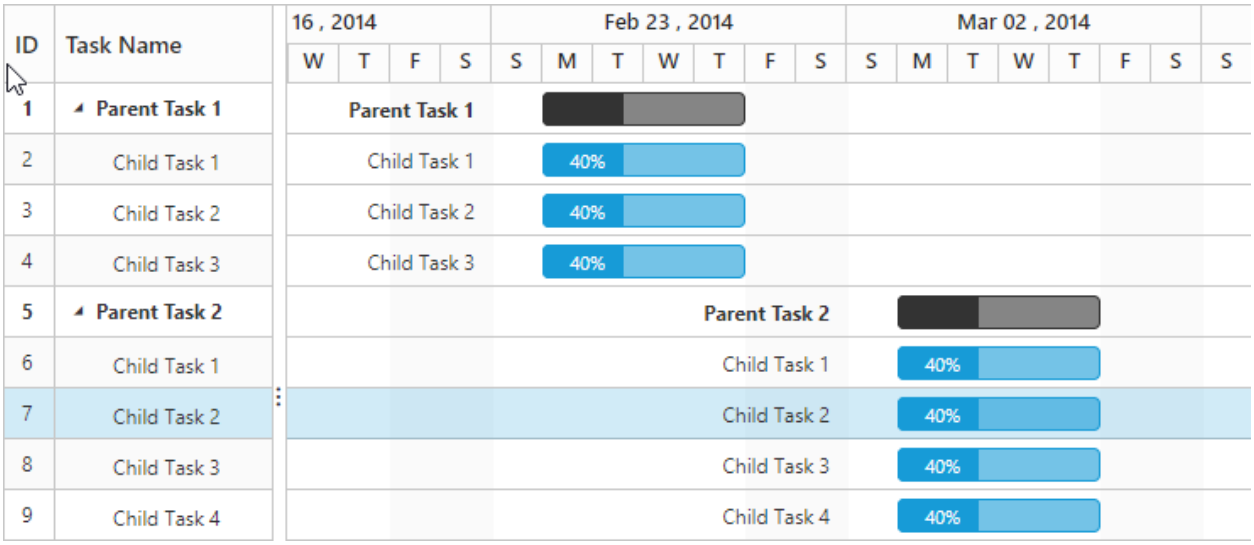
The following code example explains how to change weekend/non-working days

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
.WorkWeek(new List<string>() {
"Sunday","Monday","Tuesday","Wednesday","Thursday" })
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
```



The above screen shot will be displayed after changing the non-working days in Gantt.

Destroy Gantt Control

Gantt control is composed with more sub-controls like TreeGrid, Splitter, Dialog etc. So while destroying the Gantt control we need to

remove all sub-control elements and unbind all events bound by this control. By using destroy public method, we can properly remove the Gantt control from DOM.

The following code example explains how to destroy Gantt control.

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
//..
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#gantt").ejGantt("destroy");
<script>
```

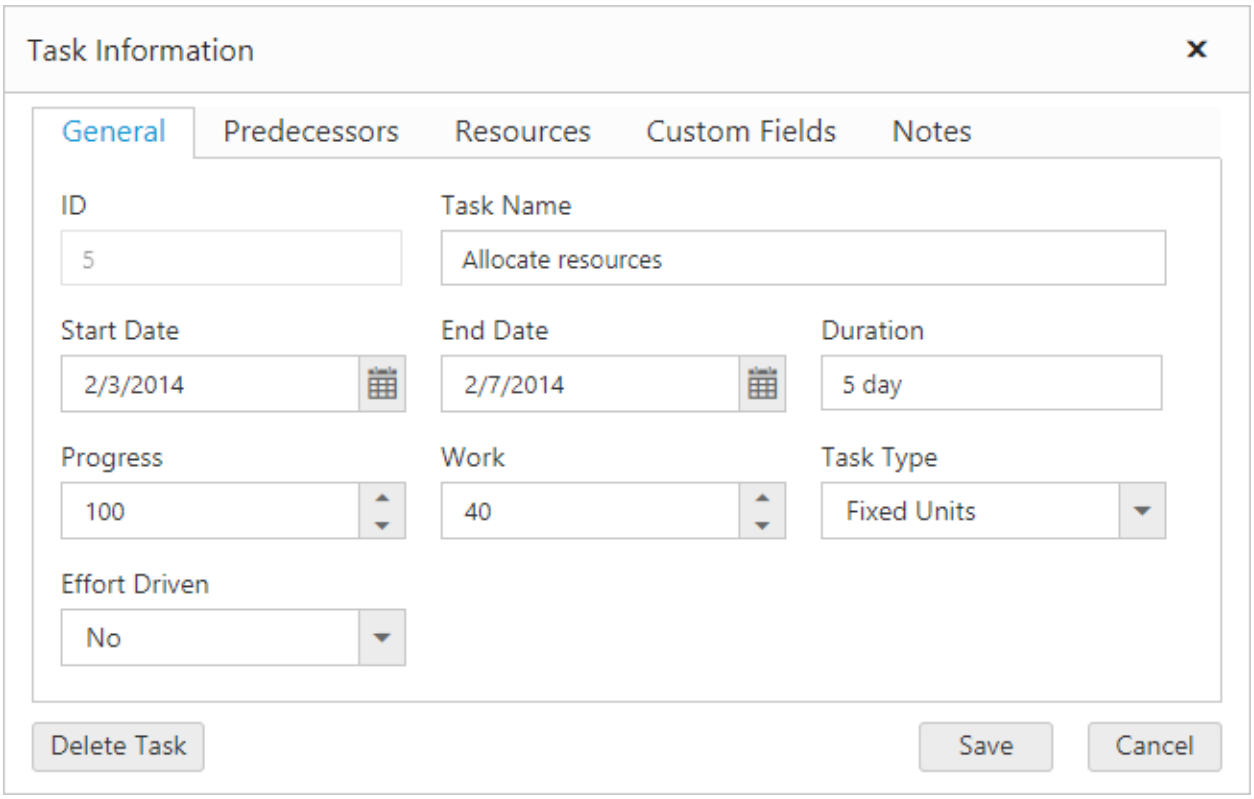
Note: If you want to re-initialize Gantt control on same div element,

we need to perform destroy action before re-initialize the Gantt.

Display custom fields in general tab

Gantt add and edit dialogs groups the data sources fields in five different tabs such as General, Predecessors, Resources, Custom Fields and Notes.

Copyright © 2001 - 2020 Syncfusion Inc.



The custom fields are usually displayed in Custom Fields tab, but it is also possible to display the custom fields in general tab using DisplayInGeneralTab property in Gantt. By default, its value is false.

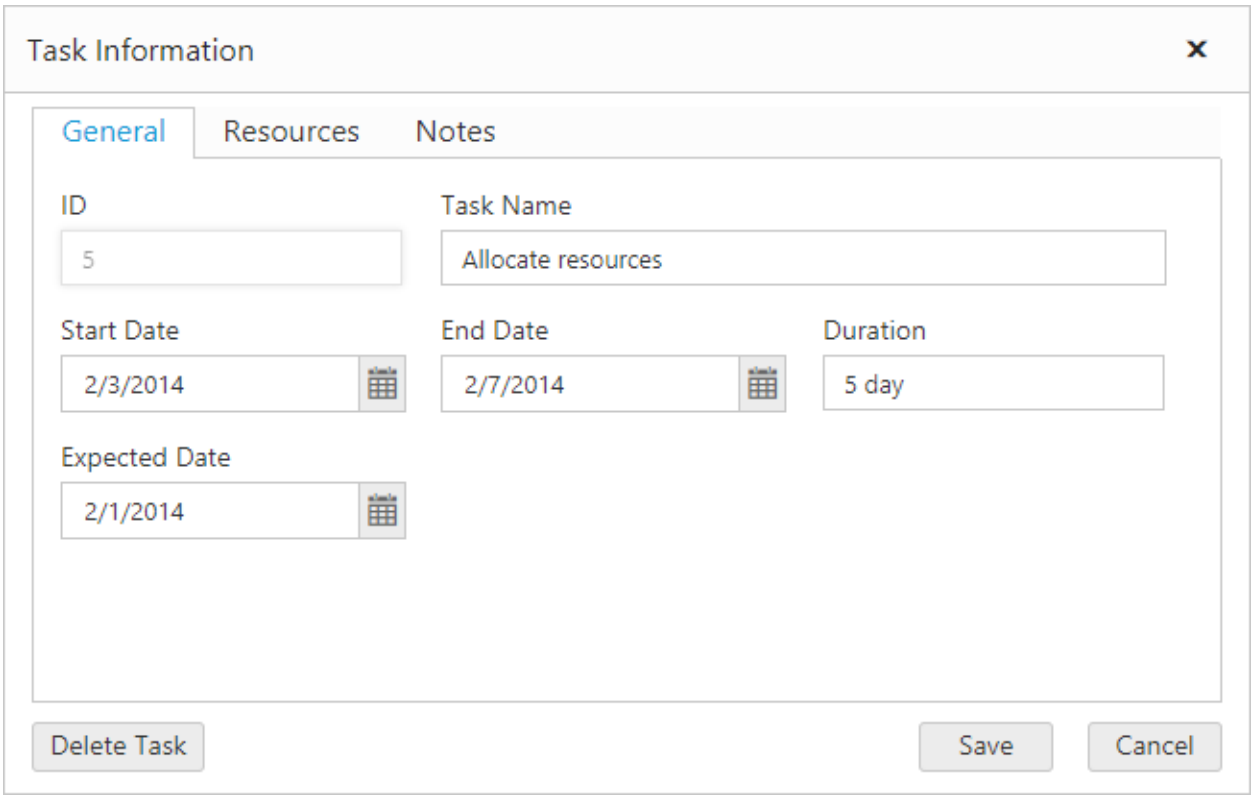
The following code example explains how to display custom fields in general tab

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
.AddDialogFields(ads =>
{
ads.Field("TaskID").Add();
ads.Field("TaskName").Add();
ads.Field("EndDate").Add();
ads.Field("Duration").Add();
ads.Field("Notes").Add();
ads.Field("ExpectedDate").DisplayInGeneralTab(true).Add();
})
.EditDialogFields(eds =>
{
eds.Field("TaskID").Add();
eds.Field("TaskName").Add();
eds.Field("EndDate").Add();
eds.Field("Duration").Add();
eds.Field("Notes").Add();
eds.Field("ExpectedDate").DisplayInGeneralTab(true).Add();
})
)
@(Html.EJ().ScriptManager())
```

The above code example makes Fridays and Saturdays as non-working days in a week.

Copyright © 2001 - 2020 Syncfusion Inc.



The above screen shot shows custom column field “Expected Date” displayed in General tab.

Change week start day in month timescale mode

Using start date mode as Month

When setting the timescaleStartDateMode property as month, the project will start from the first date of the same month of the first task in a project. Using below code example we can change the week start day of the project start date in month timescale mode.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ScheduleHeaderSettings(sh =>
{
sh.ScheduleHeaderType(GanttScheduleHeaderType.Month);
sh.TimescaleStartDateMode(GanttTimescaleRoundMode.Month);
sh.MonthHeaderFormat("MMM yyyy");
sh.WeekHeaderFormat("M/dd");
sh.WeekStartDay(1);
})
.ClientSideEvents(cs =>
{
cs.Load("load");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function load(args) {
var ganttObj = $("#Gantt").data("ejGantt");
ganttObj._enableMonthStart = false;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script>
```


Using start date mode as Year

When setting the timescaleStartDateMode property as Year, the project will start from the first date of the same year to which the first task in a project starts. Using below code example we can change the week start day of the project start date in year timescale mode.

ASPX-CS

```
@(Html.EJ().Gantt("Gantt")
.ScheduleHeaderSettings(sh =>
{
sh.ScheduleHeaderType(GanttScheduleHeaderType.Month);
sh.TimescaleStartDateMode(GanttTimescaleRoundMode.Year);
sh.WeekHeaderFormat("M/dd");
sh.WeekStartDay(1);
})
.ClientSideEvents(cs =>
{
cs.Load("load");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function load(args) {
var ganttObj = $("#Gantt").data("ejGantt");
ganttObj._enableMonthStart = false;
}
<script>
```


By default *enableMonthStart property will be true. Week header in month schedule mode will be rendered with month/year start day. To customize the week start day in month mode we need to set* enableMonthStart as false.

Round-off start date, end date and duration value on taskbar editing

In Gantt start date, end date and duration values can be round-off as per current ScheduleHeaderSettings.ScheduleHeaderType value on taskbar resizing and dragging actions. This can be achieved by setting roundOffDuration argument value as true in TaskbarEditing event.

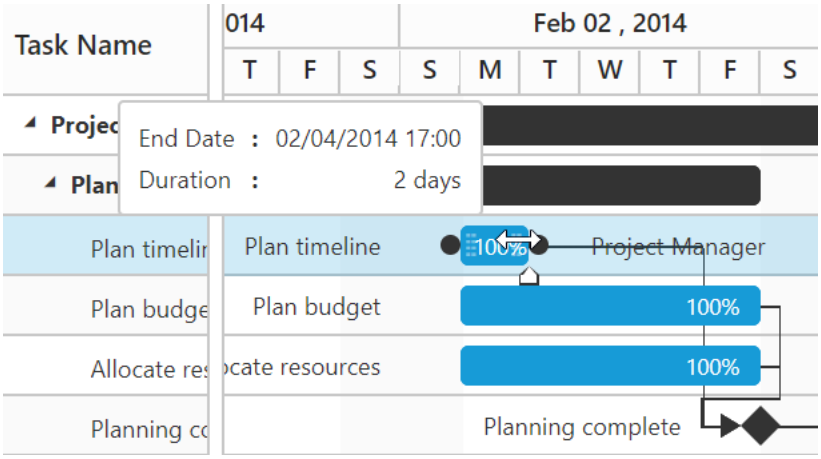
The below code example explains how to achieve this requirement.

ASPX-CS

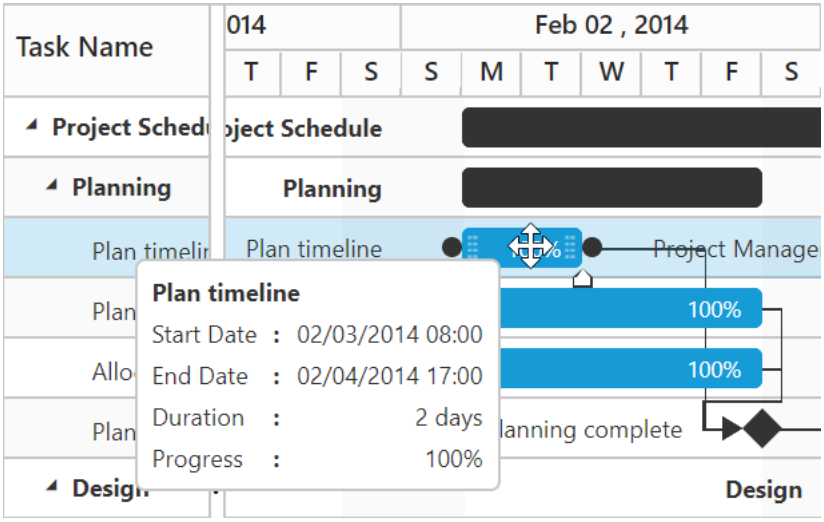
```
@(Html.EJ().Gantt("gantt")
.ClientSideEvents(eve =>
{
eve.TaskbarEditing("taskbarEditing");
})
)
<script type="text/javascript">
function taskbarEditing(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
args.roundOffDuration = true;
}
<script>
```



Before resizing



After resizing

Rendering the Gantt control using GanttProperties in server-side

The GanttProperties helps to render the Gantt control in server-side.

The following code example explains how to render the Gantt control in server-side.

RAZOR

```
@(Html.EJ().Gantt("GanttControl", ViewBag.GanttModel))
```

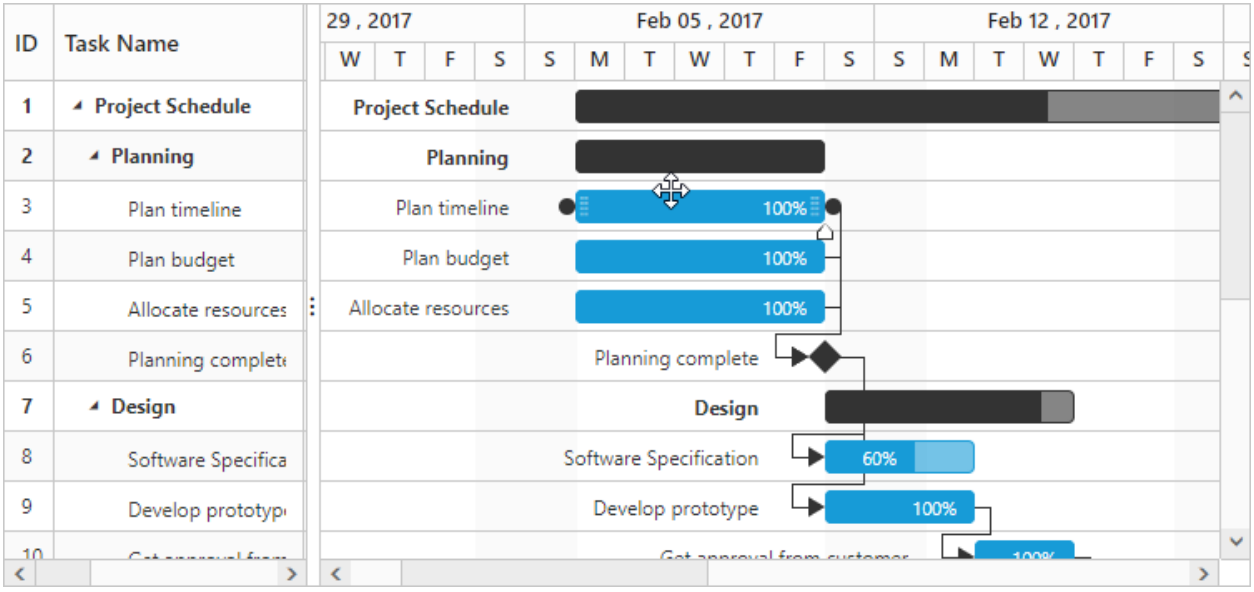
C#

```
namespace Gantt_MVC.Controllers
{
public class GanttController : Controller
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Gantt()
{
GanttProperties gantt = new GanttProperties();
gantt.TaskIdMapping = "TaskID";
gantt.TaskNameMapping = "TaskName";
gantt.StartDateMapping = "StartDate";
gantt.DurationMapping = "Duration";
gantt.ProgressMapping = "Progress";
gantt.ScheduleStartDate = "02/01/2017";
gantt.ScheduleEndDate = "03/14/2017";
gantt.ChildMapping = "SubTasks";
gantt.PredecessorMapping = "Predecessors";
gantt.TreeColumnIndex = 1;
gantt.SizeSettings.Width = "750px";
gantt.SizeSettings.Height = "350px";
gantt.DataSource = GetTaskData();
ViewBag.GanttModel = gantt;
return View();
}
}
}
```

The below screenshot shows the output of above code example.



Update task id value

Id value of Gantt tasks can be changed dynamically by using updateTaskId method. The below code example shows how to invoke this method on button click action.

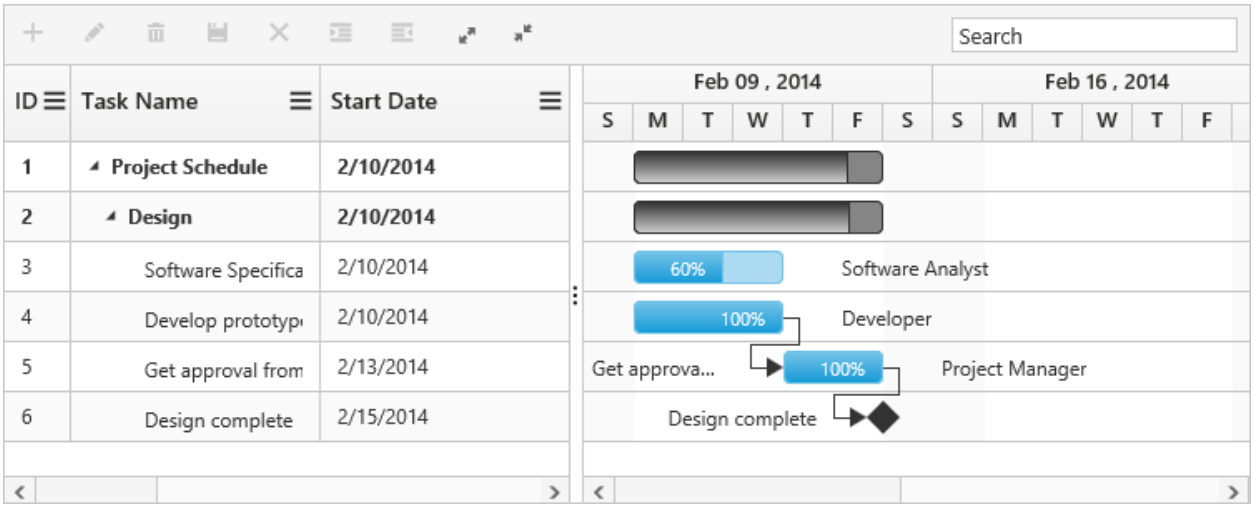
ASPX-CS

```
@(Html.EJ().Gantt("gantt")
//..
)
@(Html.EJ().ScriptManager())
$("#update_task_id").click(function () {
var ganttObj = $("#gantt").ejGantt("instance");
```

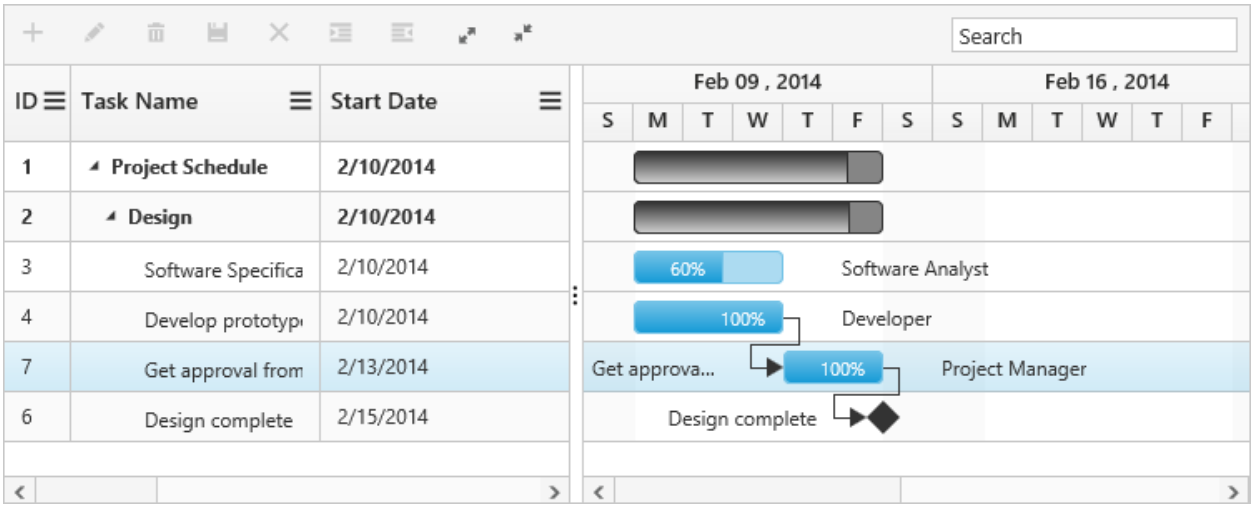
Copyright © 2001 - 2020 Syncfusion Inc.

```
ganttObj.updateTaskId(5, 7);
});
```

The below screen shows the output of above code example.



Before id update



After id update

Get taskbar information on click action

When we click on the taskbar, TaskbarClick event will be triggered, using this event and it's arguments, we can do any action related to taskbar. The following code example shows how to use this event.

ASPX-CS

```
@(Html.EJ().Gantt("gantt")
.ClientSideEvents(eve =>
{
eve.TaskbarClick("taskbarClick");
})
)
<script type="text/javascript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function taskbarClick(args) {
var taskbarElement = args.taskbarElement,
taskData = args.data,
target = args.target;
//...
}
<script>
```

Grid

Grid

The grid control for ASP.NET MVC is an efficient display engine for tabular data. It will pull from a datasource, such as List of collections, OData web services, or DataManager, binding data fields to columns and displaying a column header to identify the field. It is a feature-rich control that provides extensive appearance customization options with support for grouped records. This grid is very useful for generating complex grid-based reports with rich formatting. The most important features available in the grid control for ASP.NET MVC are paging, sorting, filtering, searching, grouping, and editing.

Key features

Some important features of the grid control for JavaScript are:

- Data sources—Bind the grid control with a List of collections or ej.DataManager.
- Sorting and grouping—Supports *n* levels of sorting and grouping.
- Filtering—Offers Excel-like filtering for filter data.
- Editing—Offers two editing modes for inserting, editing, and deleting records in a grid.
- Paging—Provides the option to easily switch between pages using the pager bar.
- Reordering—Allows you to drag any column and drop it at any position in the grid, allowing columns to be repositioned at the required place.
- Resize columns—grid provides option for resizing the columns.
- Summary support—Offers options for specifying summary rows and columns.
- Detail template—Offers to render the detail row for the corresponding expanded master row.
- Unbound columns—Offers the option to specify unbound columns.
- RTL support—Provides a full-fledged right-to-left mode which aligns content in the Grid control from right to left.
- Localization—Provides inherent support to localize the UI.

Getting Started

This section explains briefly about how to create a Grid in your application with ASP.NET MVC, and also explains about how to enable basic grid operations like paging, filtering, grouping and summary. The following screenshot illustrates the grid control.

Copyright © 2001 - 2020 Syncfusion Inc.

ItemID	ItemName	ItemType	Price	Stock
1	Apple Pie	Veg	15	3
2	Bread	Veg	10	5
3	Chocolate cake	Veg	20	2
4	Cream cookies	Veg	12	7
5	Doughnut	Veg	16	12
6	Green pepper	Veg	18	5
7	Nuggets	Veg	14	3
8	Chicken soup	Non-Veg	20	8
9	Pancake	Veg	17	6
10	Pomegranate	Veg	20	1
11	Sugar cookie	Veg	22	9
12	Wheat	Veg	12	13

Create your first grid in MVC

The following steps explain creating first grid with an empty datasource:

1. Create Syncfusion ASP.NET MVC application. You can refer to MVC Getting Started documentation to create a new project and add necessary dll's and script files.
2. Add a grid control in view file with an empty datasource as like as follows. In Columns definition, the TextAlign property allows you to align text of the columns, the Width property is used to define width of the columns and Format property allows you to format the particular columns value.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
})
)
```

You can execute the above code sample to render an empty grid is rendered with specified column headers.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Ship Name	Ship City	Freight
No records to display				

Data binding

You can bind the data to grid control using the DataSource property.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Ship Name	Ship City	Freight
10248	VINET	Vins et alcools Chevalier	Reims	$32.380
10249	TOMSP	Toms Spezialitäten	Münster	$11.610
10250	HANAR	Hanari Carnes	Rio de Janeiro	$65.830
10251	VICTE	Victuailles en stock	Lyon	$41.340
10252	SUPRD	Suprêmes délices	Charleroi	$51.300
10253	HANAR	Hanari Carnes	Rio de Janeiro	$58.170
10254	CHOPS	Chop-suey Chinese	Bern	$22.980
10255	RICSU	Richter Supermarkt	Genève	$148.330
10256	WELLI	Wellington Importadora	Resende	$13.970
10257	HILAA	HILARION-Abastos	San Cristóbal	$81.910
10258	ERNSH	Ernst Handel	Graz	$140.510
10259	CENTC	Centro comercial Moctez	México D.F.	$3.250
10260	OTTIK	Ottilies Käseladen	Köln	$55.090
10261	QUEDE	Que Delícia	Rio de Janeiro	$3.050
10262	RATTC	Rattlesnake Canyon Groc	Albuquerque	$48.290
10263	ERNSH	Ernst Handel	Graz	$146.060
10264	FOLKO	Folk och fä HB	Bräcke	$3.670
10265	BLONP	Blondel père et fils	Strasbourg	$55.280
10266	WARTH	Wartian Herkku	Oulu	$25.730
10267	FRANK	Frankenversand	München	$208.580
10268	GROSR	GROSELLA-Restaurante	Caracas	$66.290
10269	WHITC	White Clover Markets	Seattle	$4.560
10270	WARTH	Wartian Herkku	Oulu	$136.540

Enable paging

The Paging feature can be enabled by the AllowPaging property of grid control. This adds the pager in the bottom of the grid, using that pager you can display the grid records in paged view. The page size can be customized using the PageSize

and PageSettings property.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

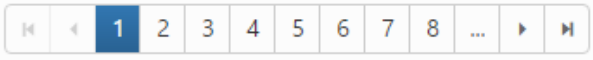
```
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Note: *Pager settings can be customized by using the 'PageSize' and 'PageCount' properties. When it is not given the default values for PageSize and PageCount are 12 and 8 respectively.*

Order ID	Customer ID	Ship Name	Ship City	Freight
10248	VINET	Vins et alcools Chevalier	Reims	$32.380
10249	TOMSP	Toms Spezialitäten	Münster	$11.610
10250	HANAR	Hanari Carnes	Rio de Janeiro	$65.830
10251	VICTE	Victuailles en stock	Lyon	$41.340
10252	SUPRD	Suprêmes délices	Charleroi	$51.300
10253	HANAR	Hanari Carnes	Rio de Janeiro	$58.170
10254	CHOPS	Chop-suey Chinese	Bern	$22.980
10255	RICSU	Richter Supermarkt	Genève	$148.330
10256	WELLI	Wellington Importadora	Resende	$13.970
10257	HILAA	HILARION-Abastos	San Cristóbal	$81.910
10258	ERNSH	Ernst Handel	Graz	$140.510
10259	CENTC	Centro comercial Moctez	México D.F.	$3.250



1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Enable filtering

The Filtering feature in the grid is used to facilitate the extraction of a subset of records that meets certain criteria. You can apply Filter to one or more columns. Filtering feature can be enabled by the AllowFiltering property. By default, the filter bar row is displayed to perform filtering, you can change the filter type by using the FilterType property of FilterSettings.

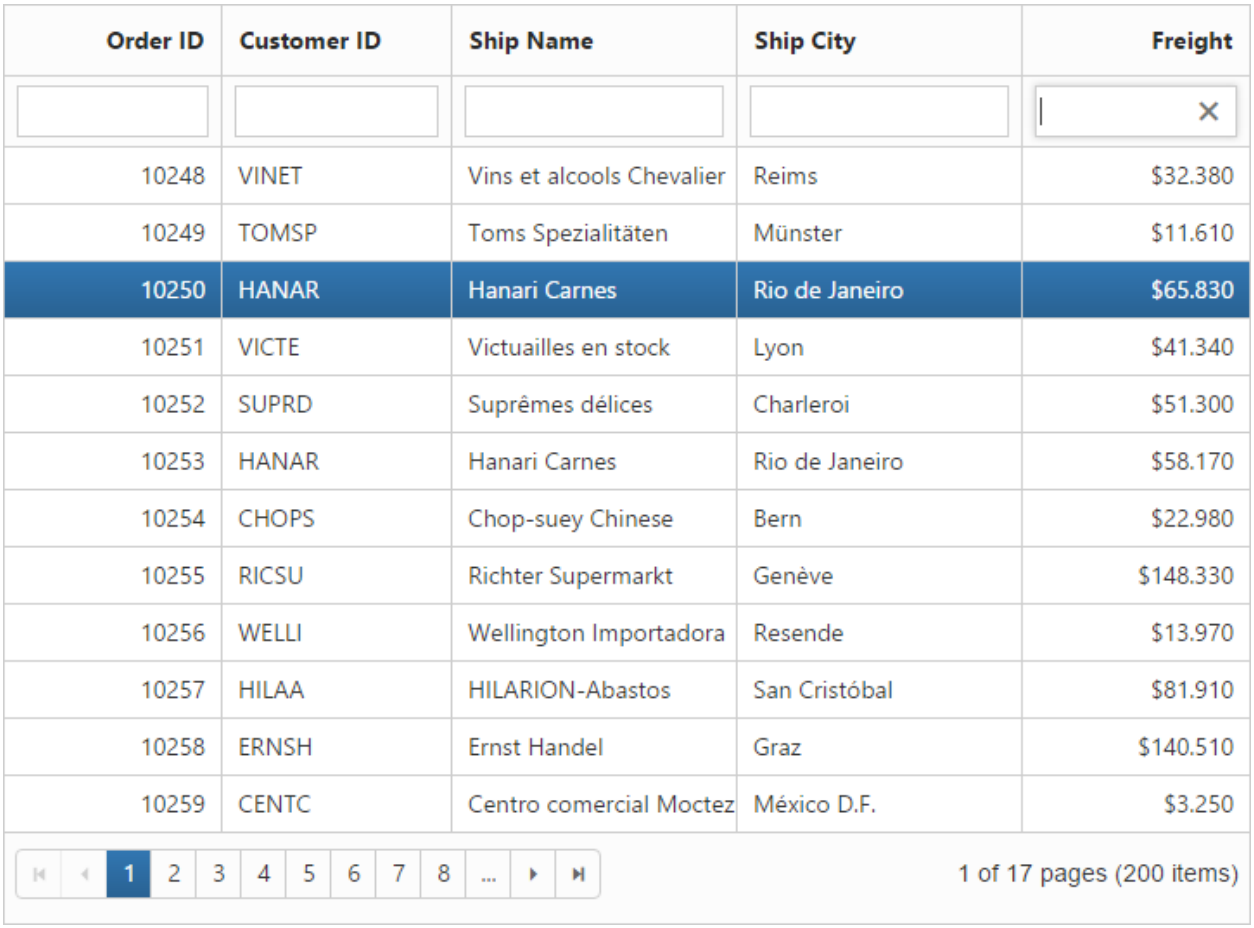
RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowFiltering()
.FilterSettings(d => d.FilterType(FilterType.FilterBar))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Order ID	Customer ID	Ship Name	Ship City	Freight
10248	VINET	Vins et alcools Chevalier	Reims	$32.380
10249	TOMSP	Toms Spezialitäten	Münster	$11.610
10250	HANAR	Hanari Carnes	Rio de Janeiro	$65.830
10251	VICTE	Victuailles en stock	Lyon	$41.340
10252	SUPRD	Suprêmes délices	Charleroi	$51.300
10253	HANAR	Hanari Carnes	Rio de Janeiro	$58.170
10254	CHOPS	Chop-suey Chinese	Bern	$22.980
10255	RICSU	Richter Supermarkt	Genève	$148.330
10256	WELLI	Wellington Importadora	Resende	$13.970
10257	HILAA	HILARION-Abastos	San Cristóbal	$81.910
10258	ERNSH	Ernst Handel	Graz	$140.510
10259	CENTC	Centro comercial Moctez	México D.F.	$3.250

Enable grouping

The 'Grouping' feature in the grid is used to consolidate grid data into groups. Grouping allows the categorization of records based on specified columns. You can enable grouping feature by the AllowGrouping property. Columns can be grouped dynamically by drag and drop of the grid column header to the group drop area. The initial grouping can be done by adding required column names in the GroupedColumns property of GroupSettings property.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => group.GroupedColumns(col=>col.Add("ShipName") ))
.AllowFiltering()
.FilterSettings(d => d.FilterType(FilterType.FilterBar))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
```

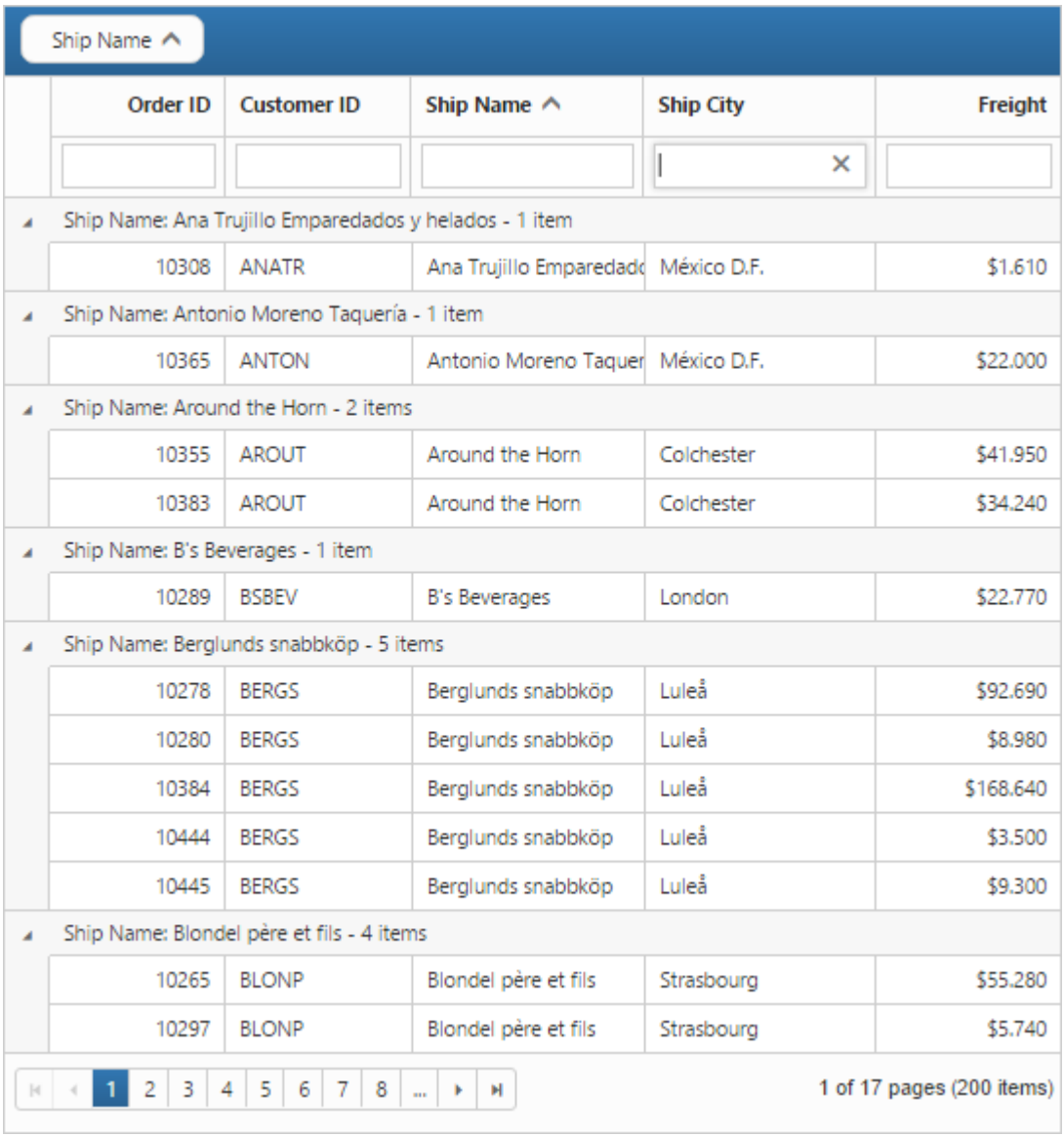
Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Add summaries

The Summaries can be enabled by setting the ShowSummary to true and adding required summary rows and columns in the SummaryRows property.

The following code example shows the option to enable group summary.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowGrouping()
.ShowSummary()
.GroupSettings(group => group.GroupedColumns(col=>col.Add("ShipName") ))
.AllowFiltering()
.FilterSettings(d => d.FilterType(FilterType.FilterBar))
.SummaryRow(row =>
{
```

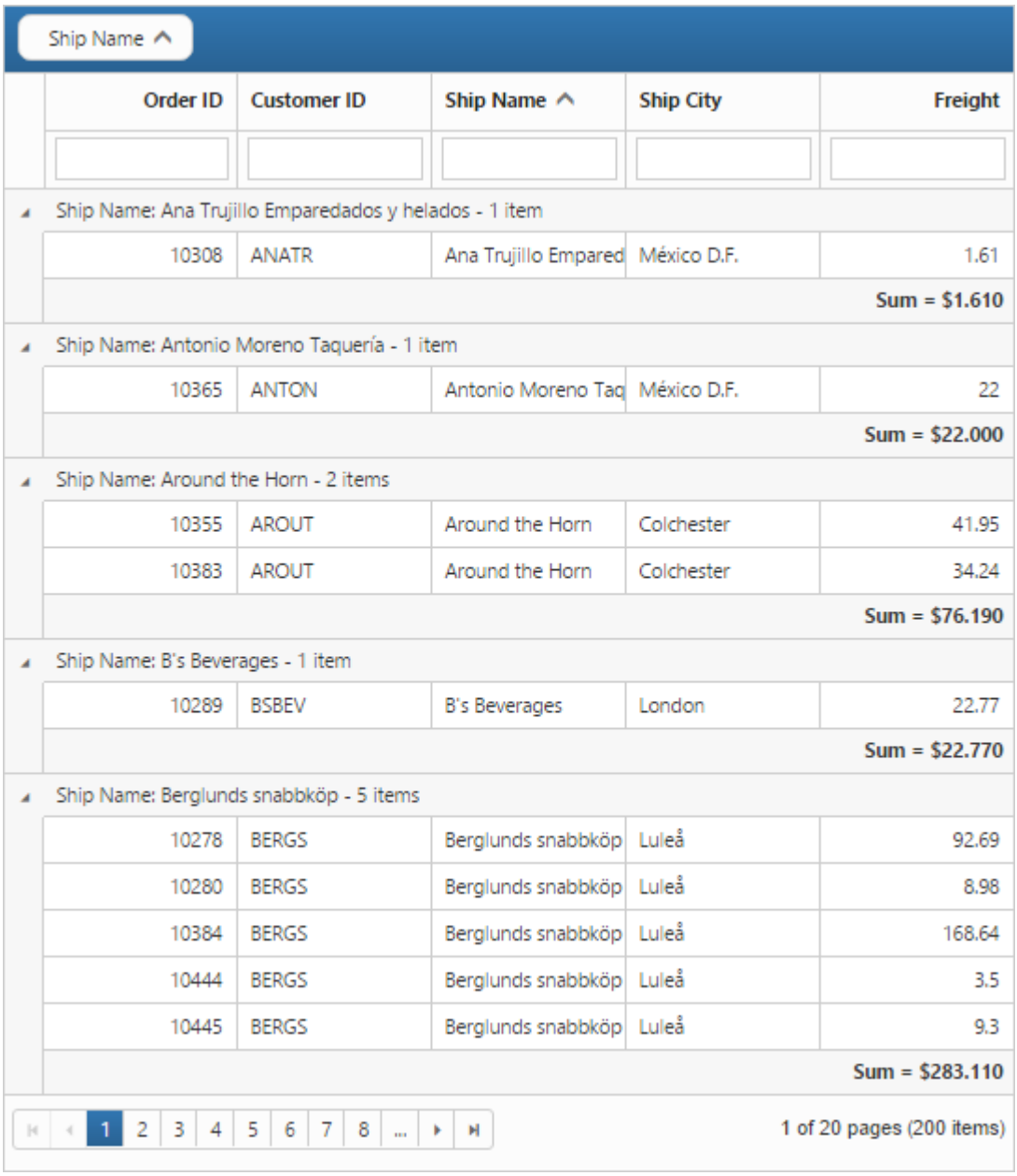
Copyright © 2001 - 2020 Syncfusion Inc.

```
row.ShowTotalSummary(false)
.SummaryColumns(col =>
{
col.SummaryType(SummaryType.Sum)
.DisplayColumn("Freight")
.DataMember("Freight")
.Prefix("Sum = ")
.Format("{0:c3}")
.Add();
}).Add();
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").Format("{0:c3}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Grid Dependencies

ej.web.all.js is a bundle of all EssentialJavaScript controls. When you use ej.web.all.js in your application, you can leave this section or else you can try to render the grid in your application by using ej.grid.min.js file. You can refer to the following frameworks and controls in your project.

Grid Dependency

FileÂ Â	Description/Usage
ej.core.min.js	Must be referred always before using all the JS controls.
ej.data.min.js	Used to handle dataManager operation and should be used while binding data to JS controls.

ej.grid.min.js	Should be referred when using grid control.
ej.pager.min.js	Should be referred when using paging in grid. Â
ej.scroller.min.js	Should be referred when using scrolling in grid. Â
ej.waitingpopup.min.js	Should be referred when using the remote data binding in grid. The waiting popup will show while requesting the server for data.
ej.gridresize.min.js	Need to refer when using the resizing feature in grid.
ej.dropdownlist.min.js	These files are used while enable Editing and Filtering feature in grid.
ej.dialog.min.js	
ej.button.min.js	
ej.autocomplete.min.js	
ej.datepicker.min.js	
ej.timepicker.min.js	
ej.datetimepicker.min.js	
ej.checkbox.min.js	
ej.editor.min.js	

Note: _Please refers ej.unobtrusive.min.js while enable the UnobtrusiveJavaScriptEnabled in web.config file.

Data Binding

Grid DataSource property allows to bind datasource as the instance of one of the following types.

Collection that implements IEnumerable or IEnumerable<T>. DataTable. *ITypedList.* REST Service URL as string. *Table – Allows to bind HTML Table and it accepts table template script "ID".* ORM components such as Entity Framework/Linq to SQL.

We can also bind the above type of datasource by using lambda Expressions of Grid DataSource Property.

Note: 1. To update the dataSource after Grid rendered, use dataSource method of Grid. Refer to this link for more information.

Note: 2. DateTime values, retrieved from server-end or database, will be converted based on the local time zone. To avoid the local time zone conversion, refer this knowledge base link.

In the following section, let us see on how to bind various datasources to Grid using DataSource API.

IEnumerable

The Grid can be bound with either non-generic collection or generic collection that implements IEnumerable interface. It can be assigned to Grid's DataSource property.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: When using DataSource builder, the IEnumerable datasource can be passed as parameter either directly to the DataSource builder or to the Json property of the DataSource builder.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Person>("Grid")
.Datasource((IEnumerable<Person>)ViewBag.datasource)
.Columns(col =>
{
col.Field(p => p.FirstName).HeaderText("First
Name").TextAlign(TextAlign.Left).Add();
col.Field(p => p.LastName).HeaderText("Last
Name").TextAlign(TextAlign.Left).Add();
col.Field(p => p.Email).HeaderText("Email").TextAlign(TextAlign.Left).Add();
}))
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
List<Person> Persons = new List<Person>();
Persons.Add(new Person() { FirstName = "John", LastName = "Beckett", Email =
"john@syncfusion.com" });
Persons.Add(new Person() { FirstName = "Ben", LastName = "Beckett", Email =
"ben@syncfusion.com" });
Persons.Add(new Person() { FirstName = "Andrew", LastName = "Beckett", Email
= "andrew@syncfusion.com" });
ViewBag.datasource = Persons;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

First Name	Last Name	Email
John	Beckett	john@syncfusion.com
Ben	Beckett	ben@syncfusion.com
Andrew	Beckett	andrew@syncfusion.com

DataTable

A DataTable, which represents one table of in-memory relational data that has in-built schema to work easily with data column and data row objects.

Copyright © 2001 - 2020 Syncfusion Inc.

Binding DataTable to Grid is a very simpler way that you only need to set DataTable model to Grid DataSource API.

The following code example describes the above behavior.

RAZOR

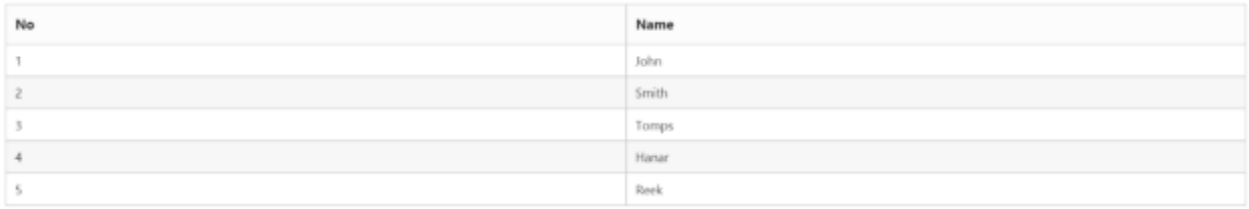
```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((System.Data.DataTable)ViewBag.dataSource)
)
```

C#

```
namespace EJGrid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
DataTable dt = new DataTable("Table1");
DataColumn cl = new DataColumn("No");
dt.Columns.Add(cl);
cl = new DataColumn("Name");
dt.Columns.Add(cl);
DataRow dataRow = dt.NewRow();
dataRow[0] = 1;
dataRow[1] = "John";
dt.Rows.Add(dataRow);
dataRow = dt.NewRow();
dataRow[0] = 2;
dataRow[1] = "Smith";
dt.Rows.Add(dataRow);
dataRow = dt.NewRow();
dataRow[0] = 3;
dataRow[1] = "Tomps";
dt.Rows.Add(dataRow);
dataRow = dt.NewRow();
dataRow[0] = 4;
dataRow[1] = "Hanar";
dt.Rows.Add(dataRow);
dataRow = dt.NewRow();
dataRow[0] = 5;
dataRow[1] = "Reek";
dt.Rows.Add(dataRow);
ViewBag.dataSource = dt;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



No	Name
1	John
2	Smith
3	Tomps
4	Hanar
5	Reek

Entity Framework

The collection of business objects generated from Entity Framework ORM can be used as Grid`s datasource. The Grid can be used with Entity Framework using either code first technique or Database first technique.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<OrdersView>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field(p => p.CustomerID).HeaderText("Customer ID").Add();
col.Field(p => p.EmployeeID).HeaderText("Employee
ID").TextAlign(TextAlign.Right).Add();
col.Field(p =>
p.Freight).HeaderText("Freight").Format("{0:C2}").TextAlign(TextAlign.Right)
.Add();
})
)
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = new NORTHWNDEntities().OrdersView.ToList();
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Freight
10643	ALFKI	6	$29.46
10692	ALFKI	4	$61.02
10702	ALFKI	4	$23.94
10835	ALFKI	1	$69.53
10952	ALFKI	1	$40.42
11011	ALFKI	3	$1.21
10926	ANATR	4	$39.92
10759	ANATR	3	$11.99
10625	ANATR	3	$43.90
10308	ANATR	7	$1.61
10365	ANTON	3	$22.00
10507	ANTON	7	$47.45

1 2 3 4 5 6 7 8 ... 1 of 70 pages (830 items)

See Also

For more information on creating and consuming Entity Framework, please refer this link.

LINQ to SQL

The LINQ to SQL can be used as the data source of the Grid. And the Object Relation class which is used to map with Relational table, can be used as the type of the Grid to leverage the Data Annotation.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<OrdersView>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field(p => p.CustomerID).HeaderText("Customer ID").Add();
col.Field(p => p.EmployeeID).HeaderText("Employee
ID").TextAlign(TextAlign.Right).Add();
col.Field(p =>
p.Freight).HeaderText("Freight").Format("{0:C2}").TextAlign(TextAlign.Right)
.Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = new NorthwindDataContext().OrdersViews.ToList();
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Order ID	Customer ID	Employee ID	Freight
10643	ALFKI	6	$29.46
10692	ALFKI	4	$61.02
10702	ALFKI	4	$23.94
10835	ALFKI	1	$69.53
10952	ALFKI	1	$40.42
11011	ALFKI	3	$1.21
10926	ANATR	4	$39.92
10759	ANATR	3	$11.99
10625	ANATR	3	$43.90
10308	ANATR	7	$1.61
10365	ANTON	3	$22.00
10507	ANTON	7	$47.45

1 2 3 4 5 6 7 8 ... 1 of 70 pages (830 items)

See Also

For more information on using LINQ to SQL please refer to this link

Copyright © 2001 - 2020 Syncfusion Inc.

ITypedList Binding

ITypedList provides functionality to discover the schema for a binding list, where the properties available to bind differ from the public properties of the object to bind.

To implement ITypedList binding, create a generic type named class that derives from ITypedList interface. Define the named class based on properties descriptor of the Grid Model class, to return list according to the custom implementation.

Create a collection of ITypedList and bind it to Grid using DataSource property.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.dataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Add();
col.Field("Freight").Add();
col.Field("ShipCountry").Add();
col.Field("ShipCity").Add();
})
)
```

C#

```
[Serializable()]
public class SortableBindingList<T> : BindingList<T>, ITypedList
{
[NonSerialized()]
private PropertyDescriptorCollection properties;
public SortableBindingList()
: base()
{
// Get the 'shape' of the list.
// Only get the public properties marked with Browsable = true.
PropertyDescriptorCollection propertyDescriptorCollection =
TypeDescriptor.GetProperties(
typeof(T),
new Attribute[] { new BrowsableAttribute(true) });
// Sort the properties.
properties = propertyDescriptorCollection.Sort();
}
public PropertyDescriptorCollection GetItemProperties(PropertyDescriptor[]
listAccessors)
{
PropertyDescriptorCollection propertyDescriptorCollection;
if (listAccessors != null && listAccessors.Length > 0)
{
// Return child list shape.
propertyDescriptorCollection =
ListBindingHelper.GetListItemProperties(listAccessors[0].PropertyType);
}
else
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
// Return properties in sort order.
propertyDescriptorCollection = properties;
}
return propertyDescriptorCollection;
}
// This method is only used in the design-time Framework
// and by the obsolete DataGrid control.
public string GetListName(PropertyDescriptor[] listAccessors)
{
return typeof(T).Name;
}
}
namespace EJGrid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var data = OrderRepository.GetAllRecords();
SortableBindingList<EditableOrder> order = new
SortableBindingList<EditableOrder>();
foreach (var temp in data)
{
order.Add(temp);
}
ViewBag.dataSource = order;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



See Also

For more information on ITypedList interface please refer to this link.

Complex Binding

The Grid can display nested or navigation properties in the column that would provide the way to display the field from another entity. The complex property can be provided in the Field property builder either as string value concatenated by dot or using lambda Expression.

Note: 1. To use lambda Expression, the Grid should be associated with some business object and Field definition using lambda expression will not work for dynamic object.

2. To display navigation properties using complex binding, the corresponding property should be eager loaded. Lazy loading is not supported for complex binding.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Person>("Grid")
.Datasource((IEnumerable<Person>)ViewBag.datasource)
.Columns(col =>
{
col.Field(p => p.FirstName).HeaderText("First
Name").TextAlign(TextAlign.Left).Add();
col.Field(p => p.LastName).HeaderText("Last
Name").TextAlign(TextAlign.Left).Add();
col.Field(p =>
p.Designation.Position).HeaderText("Designation").TextAlign(TextAlign.Left).
Add();
col.Field(p => p.Email).HeaderText("Email").TextAlign(TextAlign.Left).Add();
})
)
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
List<Person> Persons = new List<Person>();
Persons.Add(new Person() { FirstName = "John", LastName = "Beckett", Email =
"john@syncfusion.com", Designation = new Designation{ Position = "Manager"
} });
Persons.Add(new Person() { FirstName = "Ben", LastName = "Beckett", Email =
"ben@syncfusion.com", Designation = new Designation{ Position = "Asst.
Manager" } });
Persons.Add(new Person() { FirstName = "Andrew", LastName = "Beckett", Email
= "andrew@syncfusion.com", Designation = new Designation{ Position =
"Engineer"  } });
ViewBag.datasource = Persons;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

First Name	Last Name	Designation	Email
John	Beckett	Manager	john@syncfusion.com
Ben	Beckett	Asst. Manager	ben@syncfusion.com
Andrew	Beckett	Engineer	andrew@syncfusion.com

Waring: A navigation property that represents the "many" end of a relationship cannot be specified using lambda expression. Instead Field name can be provide as string in such as case.

WCF DataService / OData Service

WCF Data Services is a component of the .NET Framework that enables you to create services that use the Open Data Protocol (OData) to expose and consume data over the Web or intranet by using the semantics of representational state transfer (REST).

To consume WCF DataService in Grid control, provide the service link directly to the Grid DataSource property.

We have an online OData Service http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders created specifically for Syncfusion Controls

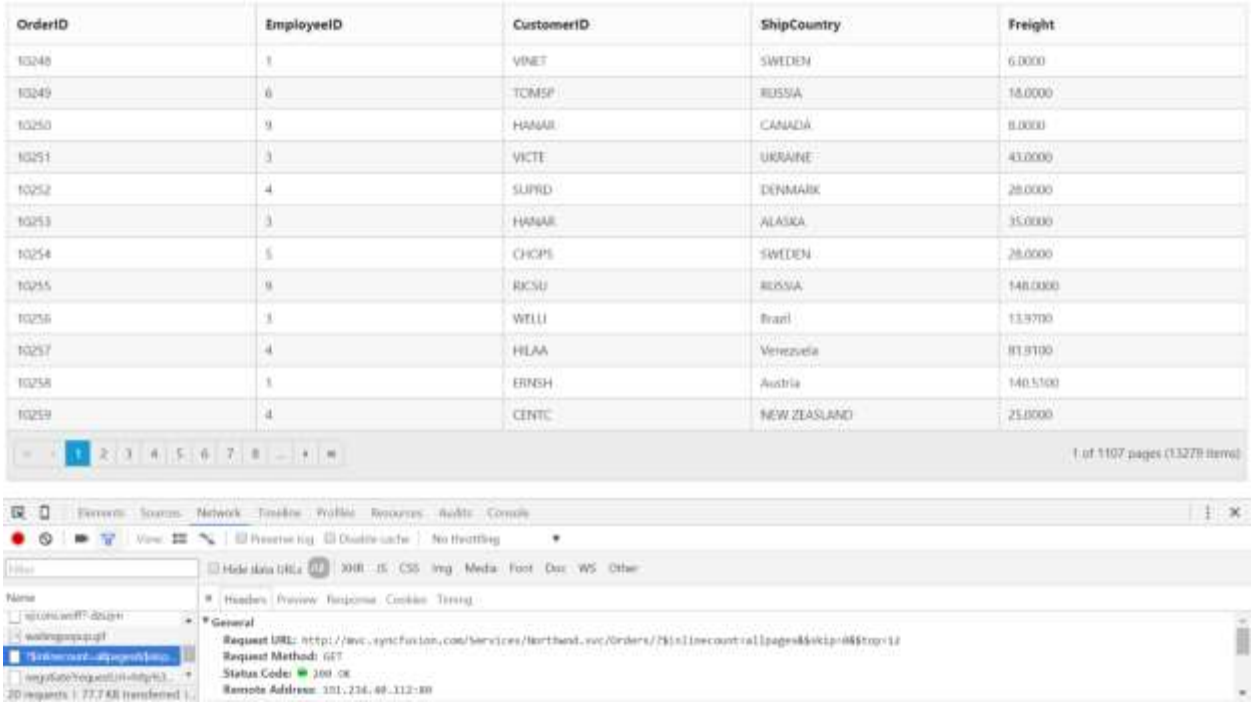
The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders")
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



See Also

To create WCF Data Service in your web application refer here.

ODataV4 Service

ODataV4Adaptor is used for consuming data from OData V4 Service. To consume OData V4 service, set the service link to the URL property of Grid DataSource and set AdaptorType.ODataV4Adaptor to the Adaptor Property of Grid DataSource

Waring: ODL supports to parse the $search query option, however, Web API OData doesn't support it so far.

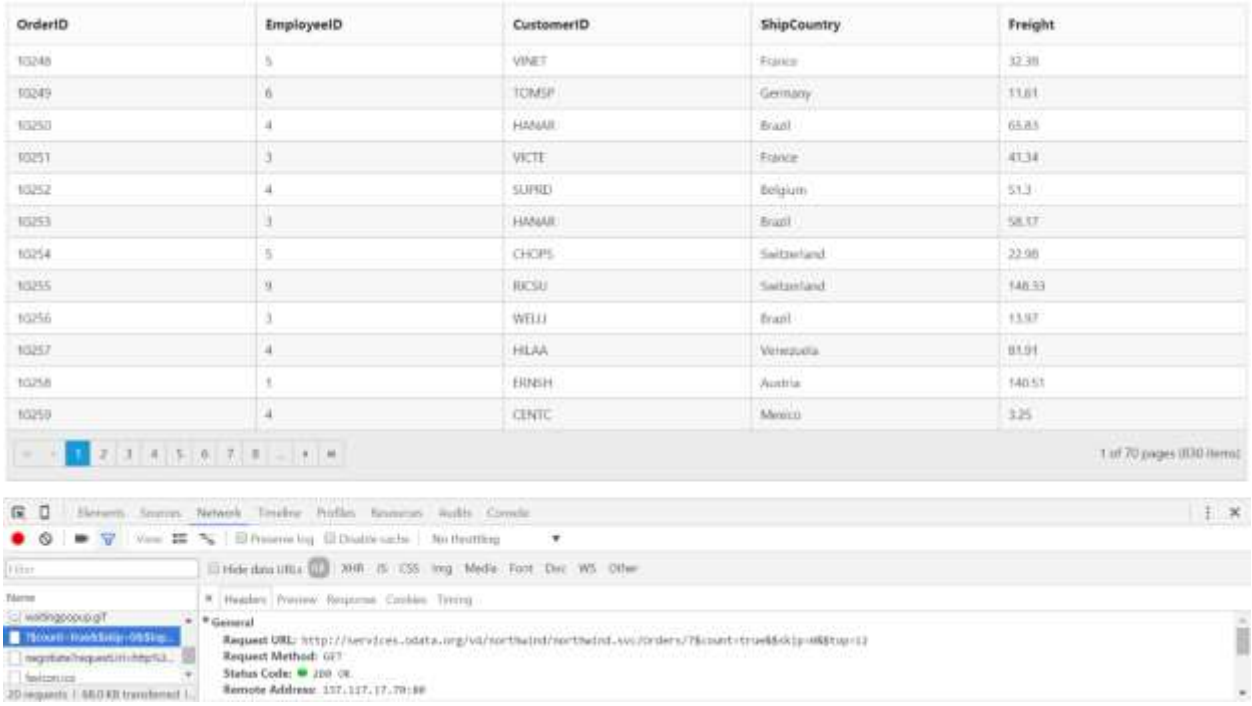
The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("http://services.odata.org/v4/northwind/northwind.svc/Orders"
).Adaptor(AdaptorType.ODataV4Adaptor))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Web API Service

Web API Adaptor is used for processing request and response messages from Web API Service.

To consume Web API service, set the service link to the URL property of Grid DataSource and you can set adaptor type as AdaptorType.WebApiAdaptor to the Adaptor Property of Grid DataSource

Information: The datasource from Web API service must be returned as object that has property Items with its value as datasource and another property Count with its value as datasource's total records count.

DataOperation queries such as sorting, filtering, etc., would be sent to Web API Service corresponding to Grid actions performed and they need to be handled manually as Web API Service does not process it by default.

Note: In ASP.NET core default casing as camelCase.So, we need to return the data as JSON and the JSON object must contain a property as result with dataSource as its value and one more property count with the dataSource total records count as its value.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("/api/Orders").Adaptor(AdaptorType.WebApiAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
```

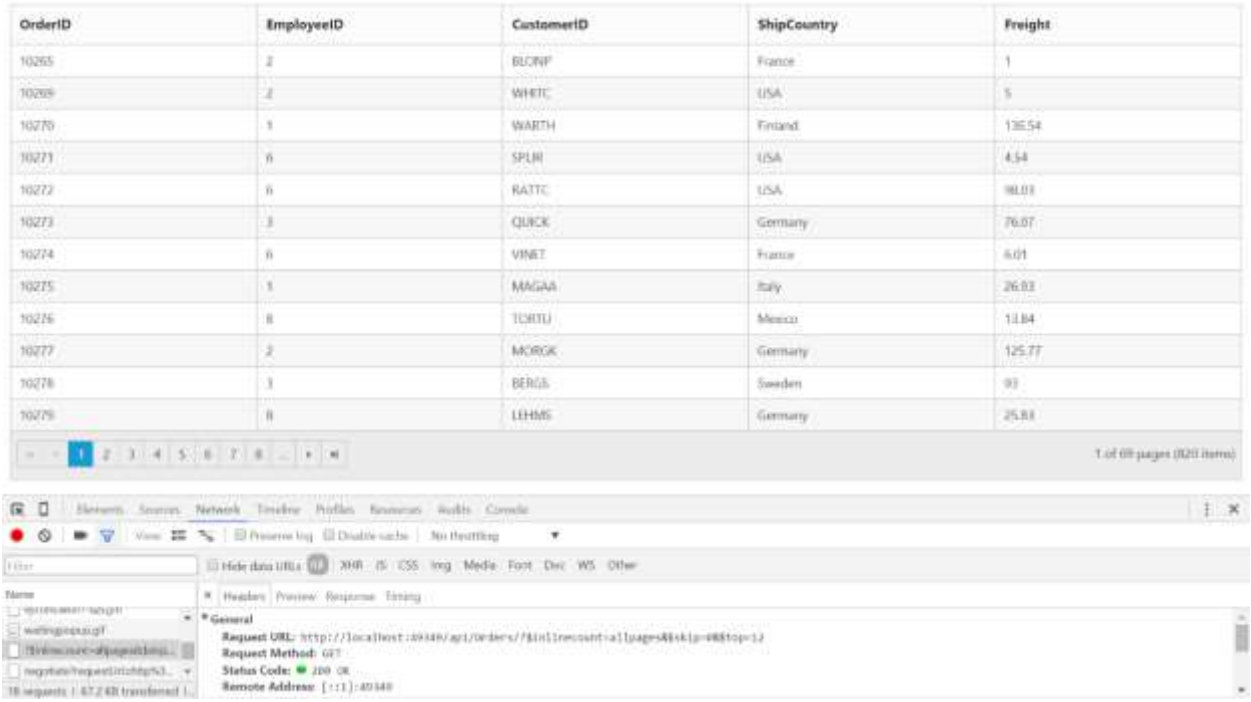
Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").Add();
})
)
```

C#

```
namespace EJGrid.Controllers
{
public class OrdersController: ApiController
{
// GET: api/Orders
NORTHWNDEntities db = new NORTHWNDEntities();
public object Get()
{
var queryString = HttpContext.Current.Request.QueryString;
int skip = Convert.ToInt32(queryString["$skip"]);
int take = Convert.ToInt32(queryString["$top"]);
var data = db.Orders.ToList();
return new { Items = data.Skip(skip).Take(take), Count = data.Count() };
}
}
}
```

The following output is displayed as a result of the above code example.



HTML Table binding

HTML table can be set as a data source for grid. The ID of the HTML table should be assigned to the `Table` property of the `DataSource` API.

Information: HTML table is the only valid element to use through `DataManager`.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("Grid")
.Datasource(datasource => datasource.Table("#GridTable"))
.Columns(col =>
{
col.Field("Laptop").HeaderText("Laptop").TextAlign(TextAlign.Left).Add();
col.Field("Model").HeaderText("Model").TextAlign(TextAlign.Left).Add();
col.Field("Price").HeaderText("Price").Add();
col.Field("OS").HeaderText("OS").TextAlign(TextAlign.Left).Add();
col.Field("RAM").HeaderText("RAM").TextAlign(TextAlign.Left).Add();
col.Field("ScreenSize").HeaderText("Screen size").Add();
})
)
```

JS

```
<script id="GridTable" type="text/template">
<table>
<thead>
<tr>
<th>Laptop</th>
<th>Model</th>
<th>Price</th>
<th>OS</th>
<th>RAM</th>
<th>ScreenSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Vostro</td>
<td>2520</td>
<td>39990</td>
<td>Windows 8</td>
<td>4GB</td>
<td>15.6</td>
</tr>
<tr>
<td>HP Pavilion Sleekbook</td>
<td>14-B104AU</td>
<td>22800</td>
<td>Windows 8</td>
<td>2GB</td>
<td>14</td>
</tr>
<tr>
<td>Sony Vaio</td>
<td>E14A15</td>
<td>42500</td>
<td>Windows 7 Home Premium</td>
<td>4GB DDR3 RAM</td>
<td>14</td>
</tr>
<tr>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<td>Lenovo</td>
<td>Yoga 13</td>
<td>57000</td>
<td>Windows 8 RT</td>
<td>2GB DDR3 RAM</td>
<td>11.6</td>
</tr>
<tr>
<td>Toshiba</td>
<td>L850-Y3110</td>
<td>57700</td>
<td>Windows 8 SL</td>
<td>8GB DDR3 RAM</td>
<td>15.6</td>
</tr>
</tbody>
</table>
</script>
```

The following output is displayed as a result of the above code example.

Laptop	Model	Price	OS	RAM	Screen size
Dell Vostro	2520	$39,990.00	Windows 8	4GB	15.6
HP Pavilion Sleek	14-B104AU	$22,800.00	Windows 8	2GB	14
Sony Vaio	E14A15	$42,500.00	Windows 7 Hom	4GB DDR3 RAM	14
Lenovo	Yoga 13	$57,000.00	Windows 8 RT	2GB DDR3 RAM	11.6
Toshiba	L850-Y3110	$57,700.00	Windows 8 SL	8GB DDR3 RAM	15.6

Miscellaneous

Load On Demand

By default, Grid with remote data binding will work in On-Demand concept for which either Paging or VirtualScrolling feature should be enabled in Grid. It helps improving performance of loading a large data set.

The following code example describes the above behavior.

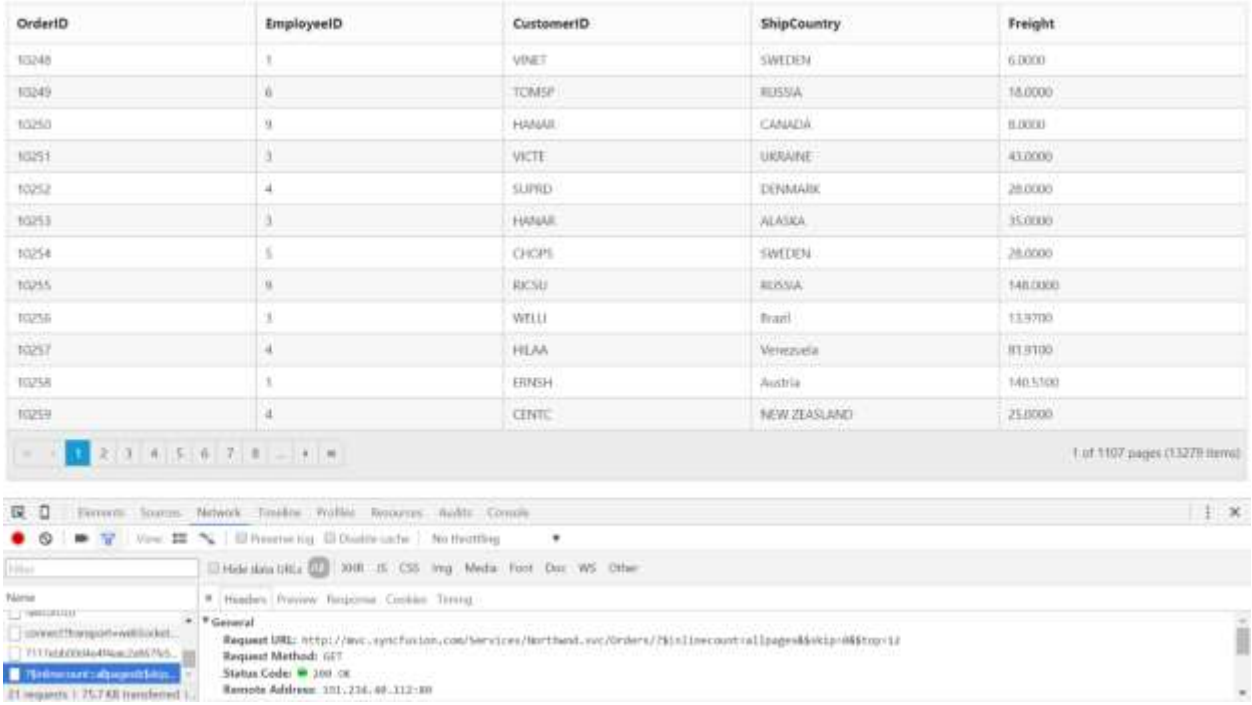
RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders")
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```

The following output is displayed as a result of the above code example.



Load at once

Load at once concept in Grid can be used to load all data from remote service at a single request and all further Grid action will be performed at client-side on the cached data.

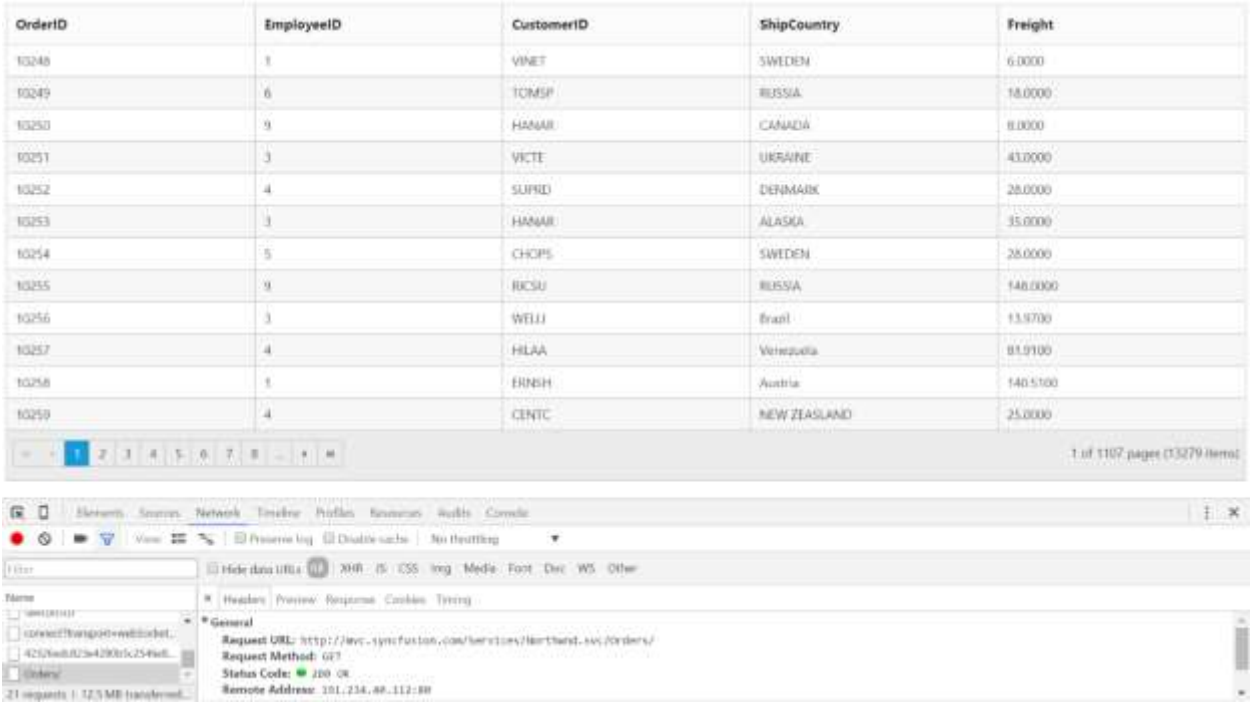
Offline property of Grid DataSource is used to enable Load at once in Grid control

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/
Orders").Offline(true))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

The following output is displayed as a result of the above code example.



Data caching

The DataManager can cache loaded data. The caching functionality can be enabled by setting the EnableCaching property in DataManager.

The TimeTillExpiration and CachingPageSize properties are used to control the expiration time of data and the cache page size settings respectively.

Note: window`s localStorage is used to cache the loaded data.

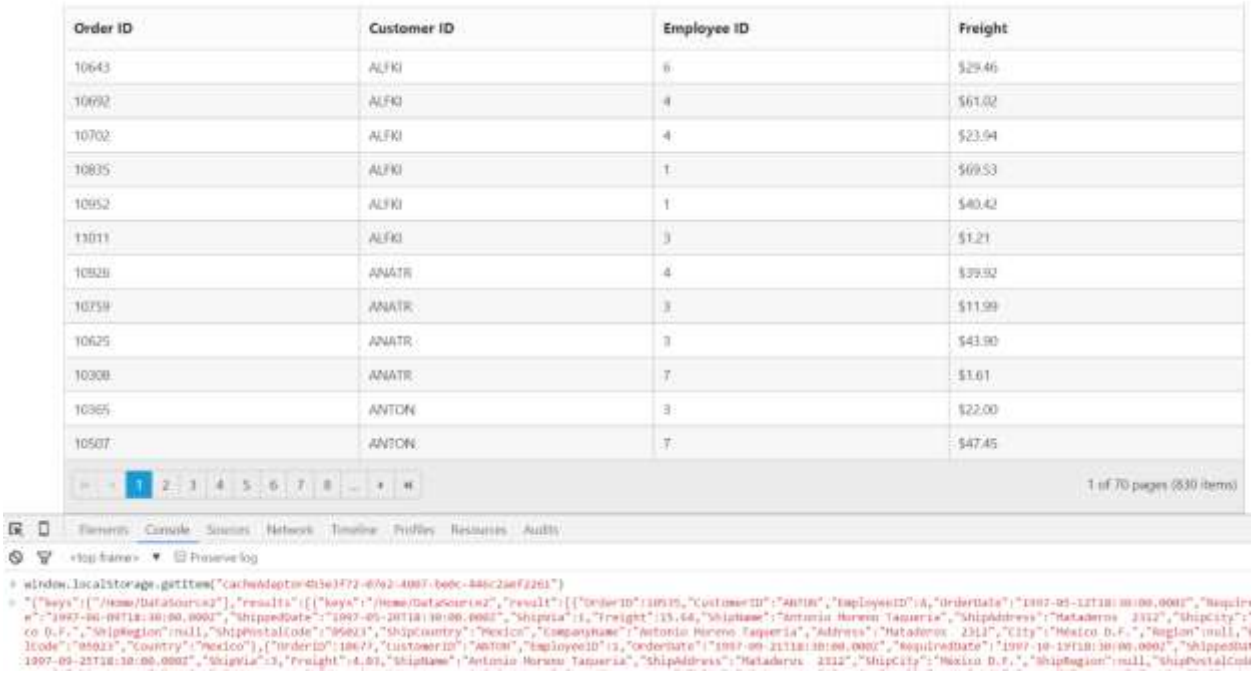
The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource => datasource.URL(@Url.Action("DataSource"))
.EnableCaching()
.CachingPageSize(4)
.TimeTillExpiration(120000)
.Adaptor(AdaptorType.UrlAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order ID").Add();
col.Field(p => p.CustomerID).HeaderText("Customer
ID").TextAlign(TextAlign.Left).Add();
col.Field(p => p.EmployeeID).HeaderText("Employee ID").Add();
col.Field(p => p.Freight).HeaderText("Freight").Format("{0:C2}").Add();
})
)
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Custom request parameters and HTTP Header

Adding request parameters

The 'addParams' function in DataManager's Query class can be used to add additional custom parameter in data requests. The Grid has an option to set default Query which can be used to add custom parameter.

The following code example describes the above behavior.

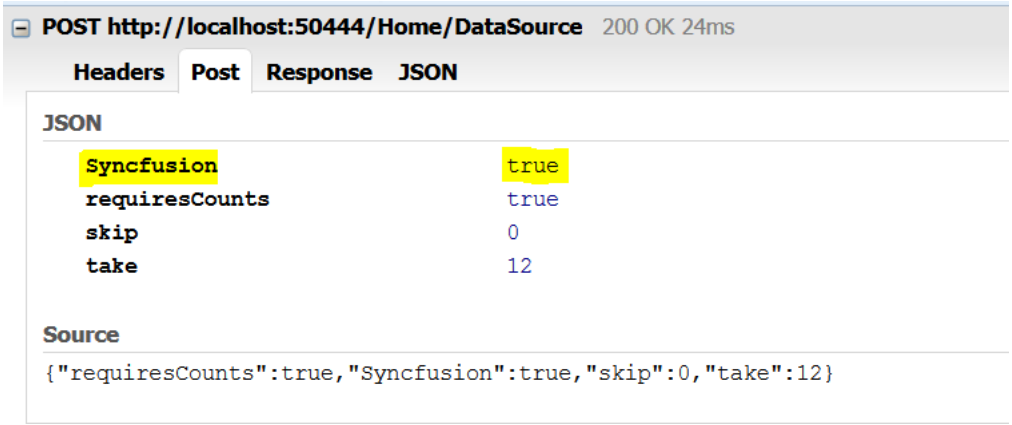
RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource => datasource.URL(@Url.Action("DataSource"))
.Adaptor(AdaptorType.UrlAdaptor))
.Query("ej.Query().addParams('Syncfusion', true)")
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order ID").Add();
col.Field(p => p.CustomerID).HeaderText("Customer
ID").TextAlign(TextAlign.Left).Add();
col.Field(p => p.EmployeeID).HeaderText("Employee ID").Add();
col.Field(p => p.Freight).HeaderText("Freight").Format("{0:C2}").Add();
})
)
```

Attempting to add custom parameters with key name same as any default AJAX parameters used by DataManager will results in error.

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding custom HTTP headers

The Custom header can be added through DataManager Headers options. While performing, CRUD operations, the addParams cannot be used to send additional parameters to the server in such cases the parameters can be send as custom header.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource => datasource.URL(@Url.Action("DataSource"))
.Headers(new { Syncfusion = false })
.Adaptor(AdaptorType.UrlAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order ID").Add();
col.Field(p => p.CustomerID).HeaderText("Customer
ID").TextAlign(TextAlign.Left).Add();
col.Field(p => p.EmployeeID).HeaderText("Employee ID").Add();
col.Field(p => p.Freight).HeaderText("Freight").Format("{0:C2}").Add();
})
)
```

Note: To add custom headers to the DataManager through JavaScript, refer to this link

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Handling HTTP Errors

During server interaction from the Grid, there may occur some server-side exceptions and you can acquire those error messages or exception details in client-side using ActionFailure event of Grid control.

The argument passed to the ActionFailure Grid event contains the error details returned from server. Please refer the following table for some error details that would be acquired in client-side event arguments.

Parameter	Description
arguments.error.status	It returns the response error code.
arguments.error.statusText	It returns the error message.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource => datasource.URL(@Url.Action("DataSources"))
.Adaptor(AdaptorType.UrlAdaptor))
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order ID").Add();
col.Field(p => p.CustomerID).HeaderText("Customer
ID").TextAlign(TextAlign.Left).Add();
col.Field(p => p.EmployeeID).HeaderText("Employee ID").Add();
col.Field(p => p.Freight).HeaderText("Freight").Format("{0:C2}").Add();
})
.ClientSideEvents(event => event.ActionFailure("OnActionFailure"))
)
```

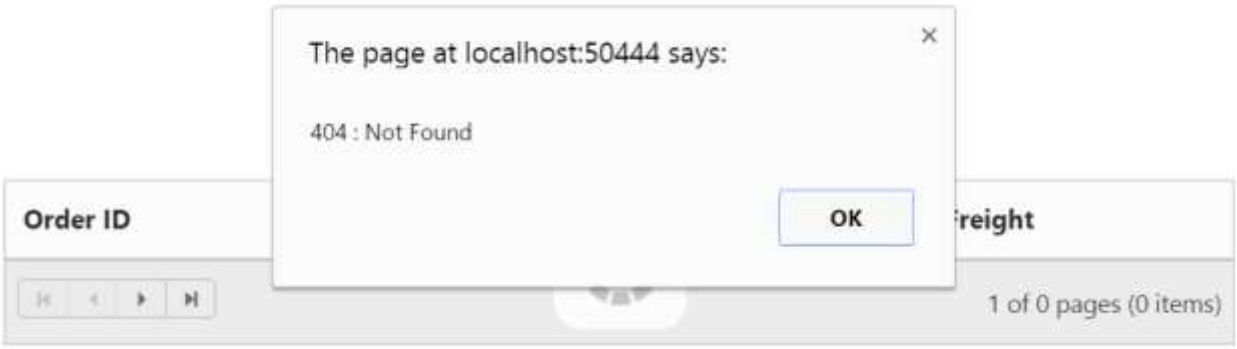
JS

```
<script>
function OnActionFailure(args) {
alert(args.error.status + " : " + args.error.statusText);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
```

The following output is displayed as a result of the above code example.



Data Adaptors

The Data Adaptor is a mechanism through which the DataManager interact with the remote service or local dataSource. The DataManager has several preconfigured Data Adaptors, refer to the link.

In Grid, the Adaptor can be specified using the Adaptor property of the DataSource builder.

Note: For the Adaptor builder, you can assign Adaptor name either as string value (”JsonAdaptor”) or Enum value as AdaptorType.JsonAdaptor.

JSON Adaptor

The JSON Adaptor is used when the grid is bound with local datasource. It can be enabled in Grid using Adaptor property as AdaptorType.JsonAdaptor. The JsonAdaptor is the default Adaptor used by DataManager when bind with local dataSource such as IEnumerable, DataTable etc.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Person>("Grid")
.Datasource(datasource =>
datasource.Json((IEnumerable<Person>)ViewBag.datasource).Adaptor(AdaptorType
.JsonAdaptor))
.Columns(col =>
{
col.Field(p => p.FirstName).HeaderText("First
Name").TextAlign(TextAlign.Left).Add();
col.Field(p => p.LastName).HeaderText("Last
Name").TextAlign(TextAlign.Left).Add();
col.Field(p => p.Email).HeaderText("Email").TextAlign(TextAlign.Left).Add();
})
)
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
```

```
{
public ActionResult Index()
{
List<Person> Persons = new List<Person>();
Persons.Add(new Person() { FirstName = "John", LastName = "Beckett", Email =
"john@syncfusion.com" });
Persons.Add(new Person() { FirstName = "Ben", LastName = "Beckett", Email =
"ben@syncfusion.com" });
Persons.Add(new Person() { FirstName = "Andrew", LastName = "Beckett", Email
= "andrew@syncfusion.com" });
ViewBag.datasource = Persons;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

First Name	Last Name	Email
John	Beckett	john@syncfusion.com
Ben	Beckett	ben@syncfusion.com
Andrew	Beckett	andrew@syncfusion.com

Url Adaptor

The Url Adaptor is the base Adaptor that would interact with remote services. It can be enabled in Grid using Adaptor property of DataSource as AdaptorType.UrlAdaptor. For every operations, an AJAX post will be send to the specified data service.

Note: When using UrlAdaptor, grid actions such as Paging, Filtering and Sorting should be handled at the server side itself. We have DataOperation class to do these server side operations. Please refer to the Knowledge Base link. These operations can also be done using the QueryableDataOperations class which accepts the IQueryable Data. Please refer the Kb link.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource => datasource.URL(@Url.Action("DataSource"))
.Adaptor(AdaptorType.UrlAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order ID").Add();
col.Field(p => p.CustomerID).HeaderText("Customer
ID").TextAlign(TextAlign.Left).Add();
col.Field(p => p.EmployeeID).HeaderText("Employee ID").Add();
col.Field(p => p.Freight).HeaderText("Freight").Format("{0:C2}").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

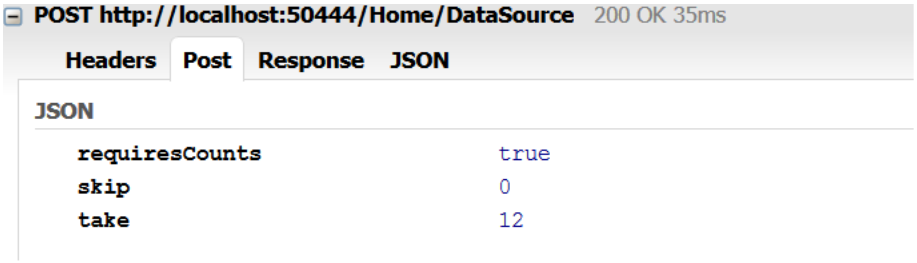
```
namespace Grid.Controllers
{
using Syncfusion.JavaScript;
using Syncfusion.JavaScript.DataSources;
public class HomeController : Controller
{
public IEnumerable OrderData = new
NorthwindDataContext().OrdersViews.ToList();
public ActionResult Index()
{
return View();
}
public ActionResult DataSource(DataManager dataManager)
{
IEnumerable data = OrderData;
DataOperations operation = new DataOperations();
if (dataManager.Sorted != null && dataManager.Sorted.Count > 0) //Sorting
{
data= operation.PerformSorting(data, dataManager.Sorted);
}
if (dataManager.Where != null && dataManager.Where.Count > 0) //Filtering
{
data= operation.PerformWhereFilter(data, dataManager.Where,
dataManager.Where[0].Operator);
}
int count = data.Cast<OrdersView>().Count();
if (dataManager.Skip != 0)
{
data= operation.PerformSkip(data, dataManager.Skip);
}
if (dataManager.Take != 0)
{
data= operation.PerformTake(data, dataManager.Take);
}
return Json(new { result = data, count = count });
}
}
}
```

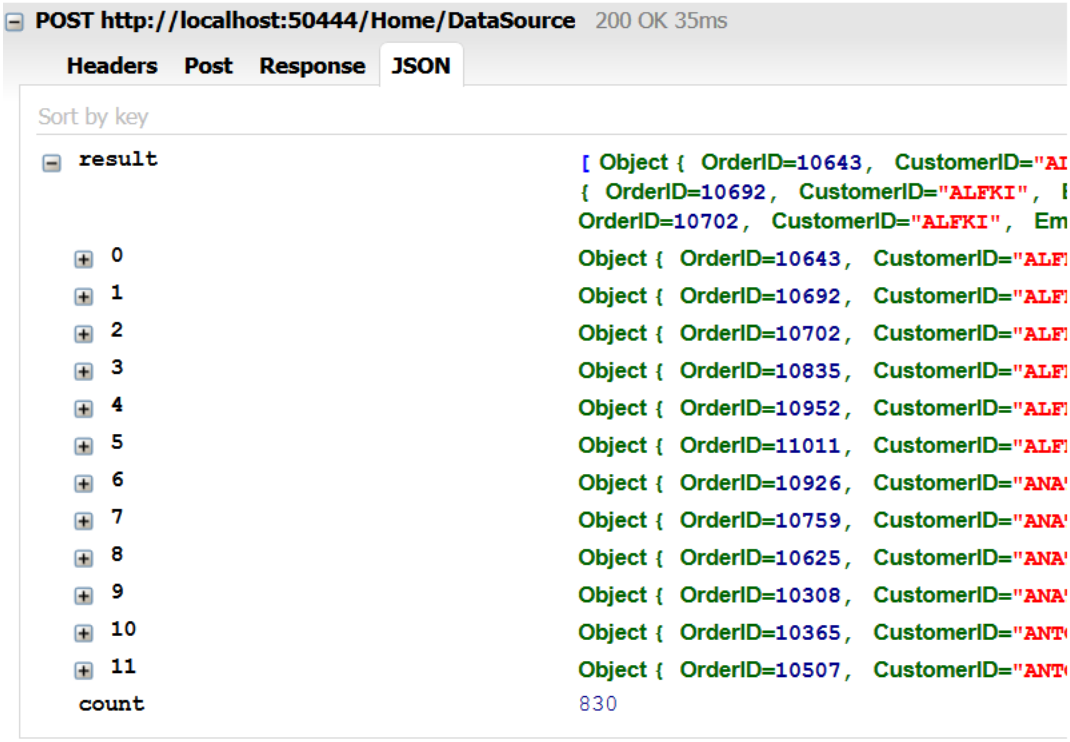
Information: 1. The response from server should be wrapped in an object with properties named result to hold the data and count to hold the total records count.

Information: 2. The count must be returned along with response when paging is enabled in Grid.

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.





OData Adaptor

OData Adaptor that is extended from UrlAdaptor, is used for consuming data through OData Service.

We have an online OData Service http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders created specifically for Syncfusion Controls

Note: ODataAdaptor is the default Adaptor of DataManager and so no need to specify Adaptor when binding OData service

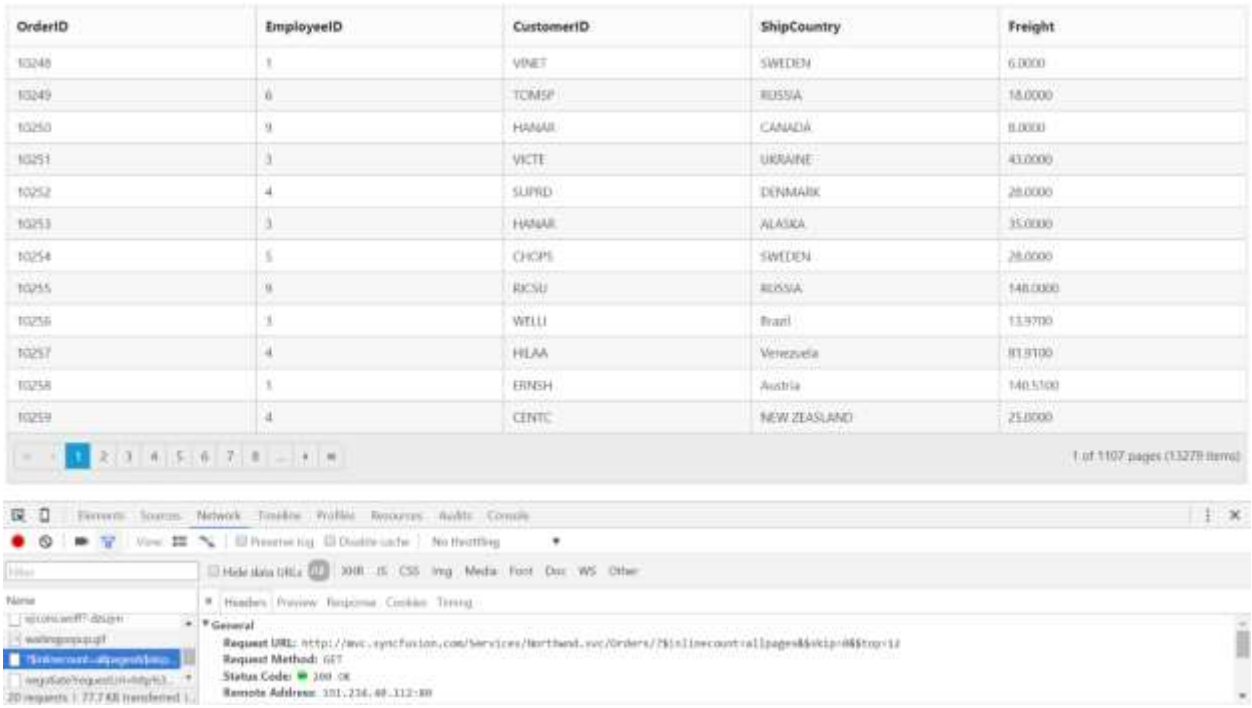
You can use the following code example to use OData Adaptor.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/
Orders").Adaptor(AdaptorType.ODataAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
```

```
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

The following output is displayed as a result of the above code example.



ODataV4 Adaptor

ODataV4 Adaptor that is extended from ODataAdaptor, is used for consuming data from ODataV4 Service

To consume ODataV4 service, set the service link to the Url property of Grid DataSource and you can set Adaptor type as AdaptorType.ODataV4Adaptor to the Adaptor Property of Grid DataSource

You can use the following code example to use ODataV4 Adaptor.

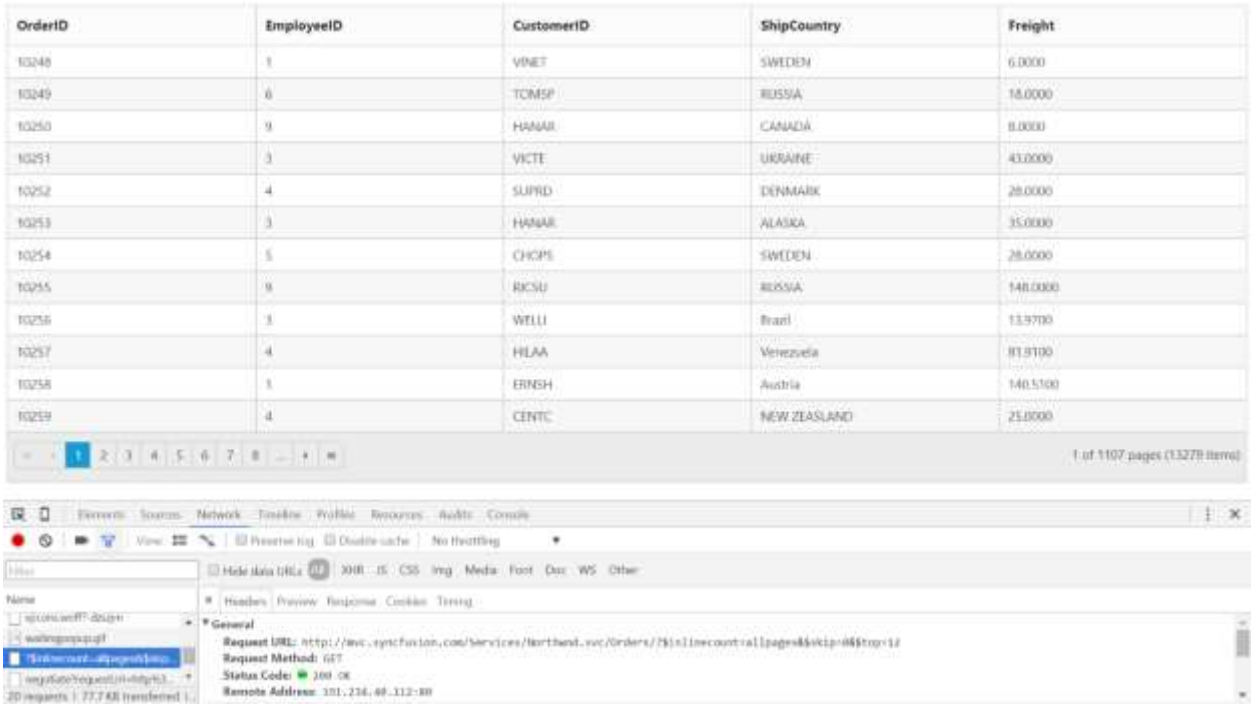
RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("http://services.odata.org/v4/northwind/northwind.svc/Orders"
).Adaptor(AdaptorType.ODataV4Adaptor))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

The following output is displayed as a result of the above code example.



WebAPI Adaptor

WebAPI Adaptor that is extended from ODataAdaptor, is used for consuming data from WebAPI Service

To consume Web API service, set the service link to the Url property of Grid DataSource and you can set Adaptor type as AdaptorType.WebApiAdaptor to the Adaptor Property of Grid DataSource.

You can use the following code example to use WebAPI Adaptor.

RAZOR

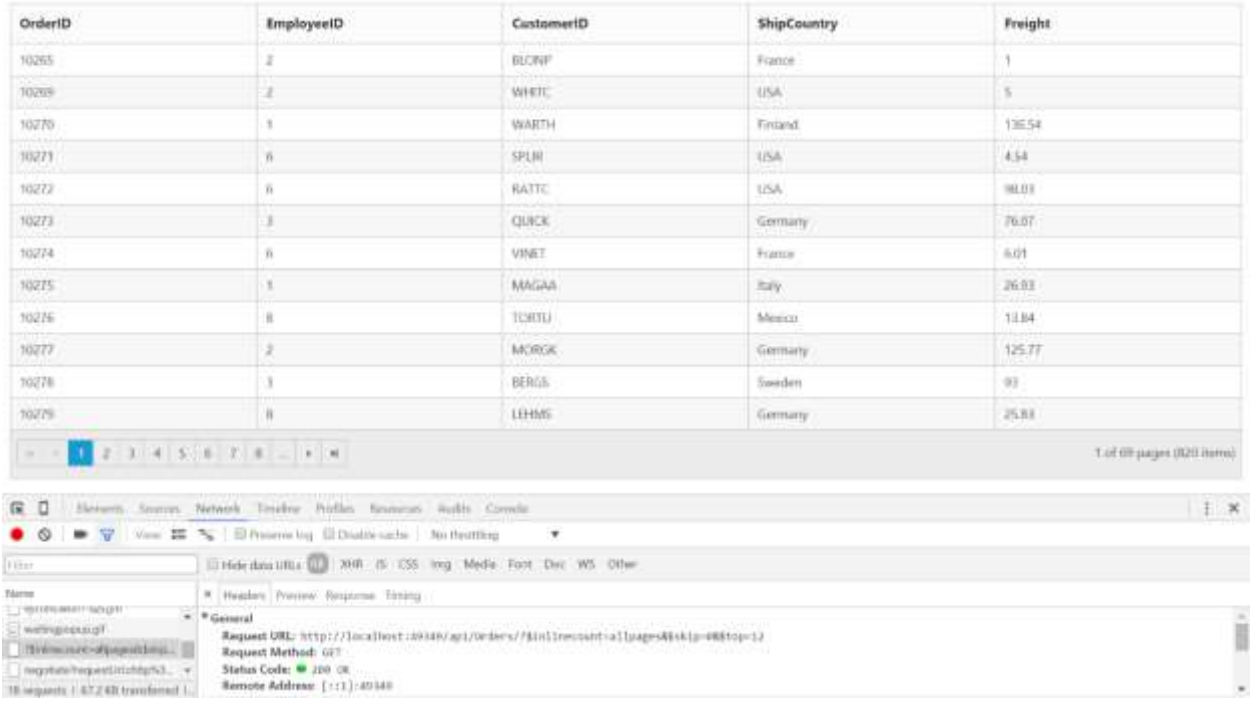
```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("/api/Orders").Adaptor(AdaptorType.WebApiAdaptor))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace EJGrid.Controllers
```

```
{
public class OrdersController: ApiController
{
// GET: api/Orders
NORTHWNDEntities db = new NORTHWNDEntities();
public object Get()
{
var queryString = HttpContext.Current.Request.QueryString;
int skip = Convert.ToInt32(queryString["$skip"]);
int take = Convert.ToInt32(queryString["$top"]);
var data = db.Orders.ToList();
return new { Items = data.Skip(skip).Take(take), Count = data.Count() };
}
}
}
```

The following output is displayed as a result of the above code example.



RemoteSave Adaptor

Sometimes you may need to perform all Grid Actions in client-side except the CRUD operations that should be interacted with server-side to persist data. It can be achieved in Grid by using RemoteSaveAdaptor.

Datasource must be set to Json Property and set Adaptor type as AdaptorType.RemoteSaveAdaptor to the Adaptor Property of Grid DataSource. CRUD operations can be mapped to server-side using UpdateUrl, InsertUrl, RemoveUrl, BatchUrl, CrudUrl properties.

You can use the following code example to use RemoteSaveAdaptor.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
```

```
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.datasource).UpdateURL("/Home/Up
date")
.InsertURL("/Home/Insert").RemoveURL("/Home/Remove").Adaptor(AdaptorType.Rem
oteSaveAdaptor))
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

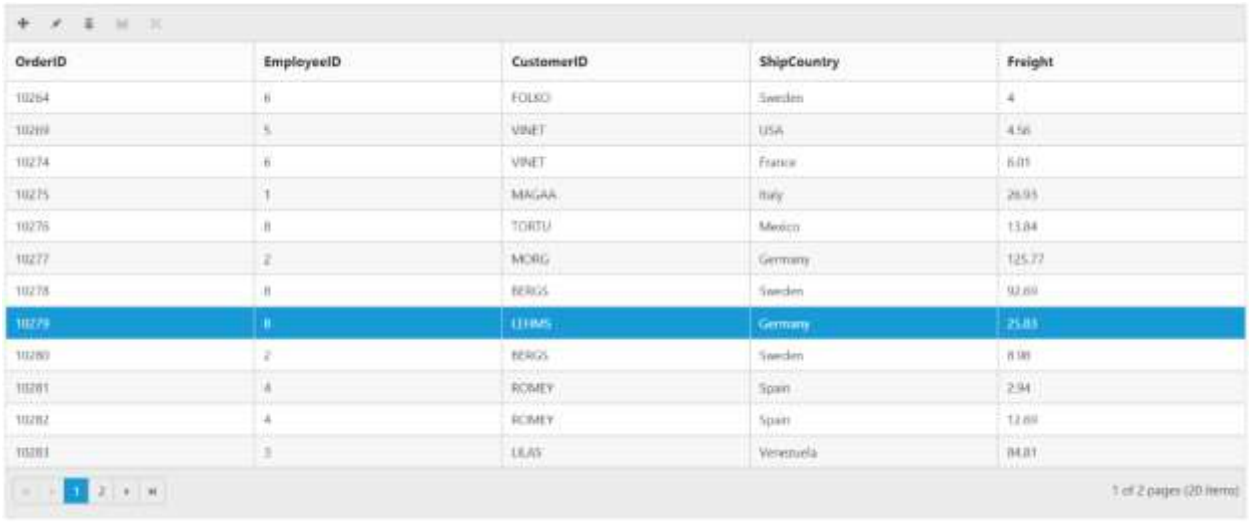
C#

```
namespace EJGrid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var data = OrderRepository.GetAllRecords();
ViewBag.dataSource = data;
return View();
}
public ActionResult Update(EditableOrder value)
{
OrderRepository.Update(value);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(EditableOrder value)
{
OrderRepository.Add(value);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
public ActionResult Remove(int key)
{
OrderRepository.Delete(key);
```

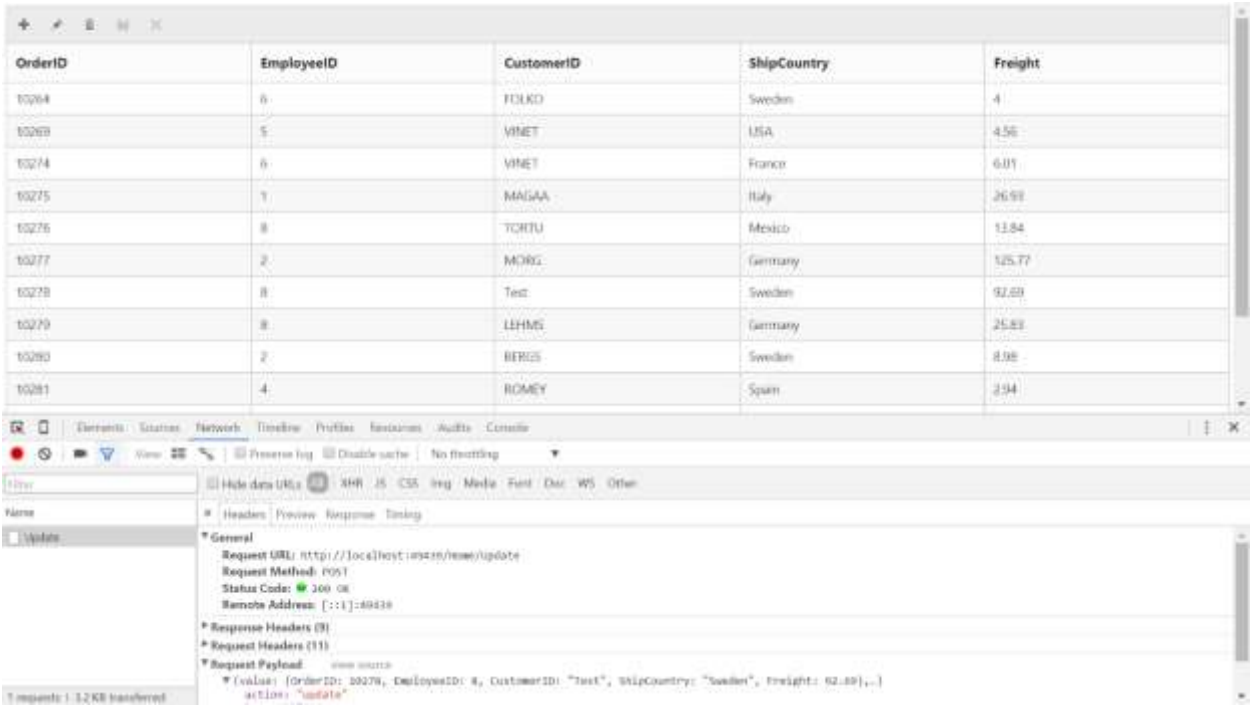
Copyright © 2001 - 2020 Syncfusion Inc.

```
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
}
}
```

The following output is displayed as a result of the above code example.



On performing CRUD operations in Grid, the record changes will be sent to server-side as in the following screenshot.



Foreign Key Adaptor

The Grid can have a look up column. The Foreign key column using ForeignKeyField has some limitations such as sort/group operations on column will happen based on Field instead of ForeignKeyField. The ForeignKeyAdaptor can be used to overcome this limitation.

Note: It works by specifying a virtual column (which is not in the grid datasource) in the Grid. This Adaptor should be initialized in the Load event of the grid. ForeignKeyAdaptor supported for only local data binding.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<OrdersView>)ViewBag.datasource)
.AllowPaging()
.AllowSorting()
.Columns(col =>
{
col.Field(p => p.OrderID).HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field(p => p.CustomerID).HeaderText("Customer ID").Add();
col.Field("FirstName").HeaderText("Name").Add();
col.Field(p =>
p.Freight).HeaderText("Freight").TextAlign(TextAlign.Right).Format("{0:C2}")
.Add();
})
.ClientSideEvents(event => event.Load("onLoad"))
)
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.employeeData = new JavaScriptSerializer().Serialize(new
NorthwindDataContext().EmployeeViews.ToList());
return View();
}
}
}
```

JS

```
<script>
var employeeData = @Html.Raw(ViewBag.employeeData);
function onLoad(args) {
this.model.dataSource.adaptor = new ej.ForeignKeyAdaptor([{ foreignKeyField:
"EmployeeID", foreignKeyValue: "FirstName", dataSource: employeeData }],
"JsonAdaptor");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

Information: 1. The Field name of the virtual column should be the name of the field to display from foreign datasource.

Information: 2. By default, the ForeignKeyAdaptor uses JsonAdaptor, to use other Adaptors specify the Adaptor name as the second argument during initialization.

The following output is displayed as a result of the above code example.

Order ID	Customer ID	Name	Freight
10280	BERGS	Andrew	$8.98
10265	BLONP	Andrew	$55.28
10663	BONAP	Andrew	$113.15
10982	BOTTM	Andrew	$14.01
10949	BOTTM	Andrew	$74.44
10471	BSBEV	Andrew	$45.59
10819	CACTU	Andrew	$19.76
11042	COMMI	Andrew	$29.99
10462	CONSH	Andrew	$6.17
10683	DUMON	Andrew	$4.40
10595	ERNSH	Andrew	$96.78
10368	ERNSH	Andrew	$101.95

1 2 3 4 5 6 7 8 ... 1 of 70 pages (830 items)

Custom Adaptor

Instead of using the pre-configured Adaptors, the DataManager allow us to write our own data Adaptor.

Please refer here for more details about creating custom Adaptor.

The custom Adaptor should be assigned to the Grid DataSource in the Load event.

For instance, though we have not provided in-built support to bind XML data, you can achieve by creating custom Adaptor as explained in below KB.

https://www.syncfusion.com/kb/3375/how-to-process-xml-data-from-server-using-datamanager-and-bound-to-grid

Copyright © 2001 - 2020 Syncfusion Inc.

Note: To create a custom Adaptor for remote service, the UrlAdaptor can be used as base Adaptor.

Data Annotation

Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end users.

You can enable Data Annotation by binding the corresponding Class to Grid helper and thus data annotations attributes will be mapped to the corresponding Grid Column property.

Please find the following list of annotation attributes that are supported in Grid Control.

Attribute Name	Functionality in Grid
BindAttribute - Exclude	To exclude the corresponding field from Grid Columns property in AutoGenerate Grid
DisplayName	It sets HeaderText property of Grid Column
ReadOnly	It sets AllowEditing for a particular column
Key	To Set PrimaryKey in Grid Columns
ScaffoldColumn	It sets Visible property of Grid Columns which is used to hide or show a Column in Grid
DisplayFormat - DataFormatString	To sets Format property of Grid Column that renders corresponding Grid Column data in respective format.
DatabaseGenerated(DatabaseGeneratedOption.Identity)	To set isIdentity Property of Grid Column
Validation • RequiredAttribute • StringLengthAttribute • RangeAttribute • RegularExpressionAttribute • MinLengthAttribute • MaxLengthAttribute • CompareAttribute • DataTypeAttribute • DataType.Custom • DataType.Date • DataType.DateTime • DataType.EmailAddress • DataType.ImageUrl • DataType.Url • Custom DataType	The data annotation validations attribute would used as validation rules in Grid CRUD operations
ForeignKey	Set ForeignKeyField Property of Grid Column

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Grid Properties has more priority than Data Annotation. For Instance, if DisplayName Attribute is set to a Field in Grid Model class and also we set different value to the respective Grid Columns property HeaderText, then the value of HeaderText property will be considered and shown in Grid header.

The following code example shows how data annotation works in Grid Control.

RAZOR

```
@model IEnumerable<EditableOrder>
@(Html.EJ().Grid<EditableOrder>("FlatGrid")
.Datasource(Model)
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.AllowPaging()
)
```

C#

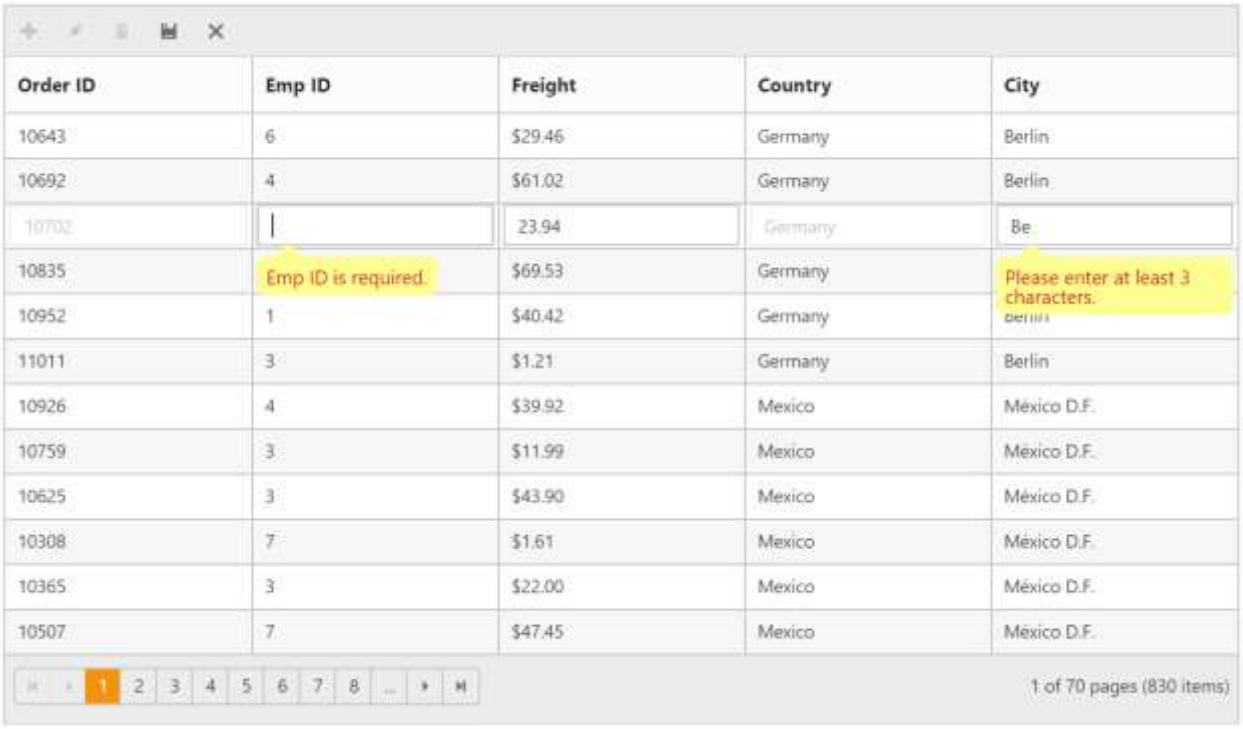
```
namespace EJGrid.Controllers
{
public class EditableOrder
{
[Display(Name = "Order ID")]
[Key]
[DatabaseGenerated(DatabaseGeneratedOption.Identity)]
public int OrderID
{
get;
set;
}
[Display(Name = "Emp ID")]
[Required]
public int? EmployeeID
{
get;
set;
}
[Display(Name = "Freight")]
[DisplayFormat(DataFormatString = "{0:C2}")]
public decimal? Freight
{
get;
set;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
[Display(Name = "Country")]
[Editable(false)]
public string ShipCountry
{
get;
set;
}
[Display(Name = "City")]
[MinLength(3)]
[MaxLength(10)]
public string ShipCity
{
get;
set;
}
[Display(Name = "Ship Name")]
[ScaffoldColumn(false)]
public string ShipName
{
get;
set;
}
}
public class HomeController : Controller
{
public ActionResult Index()
{
List<EditableOrder> data = OrderRepository.GetAllRecords() ;
return View(data);
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Columns

Column definitions are used as the DataSource schema in the grid and it plays vital role in rendering the column values in required format and sorting, filtering, editing based on its type. The Field property of the columns is necessary to map the datasource values in grid columns.

Note: 1. The column with Field which are not in the datasource, then the column values will be displayed as empty.

2. If the Field name contains "dot" then it is considered as complex binding.

Auto Generation

The columns are automatically generated when Columns declaration is empty or undefined while initializing the grid. Also, all the columns which are in DataSource are bound as a grid Columns.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	Freight	ShipCountry	ShipCity
10248	5	32.38	France	Reims
10249	6	11.61	Germany	Münster
10250	4	65.83	Brazil	Rio de Janeiro
10251	3	41.34	France	Lyon
10252	4	51.3	Belgium	Charleroi
10253	3	58.17	Brazil	Rio de Janeiro
10254	5	22.98	Switzerland	Bern
10255	9	148.33	Switzerland	Genève
10256	3	13.97	Brazil	Resende
10257	4	81.91	Venezuela	San Cristóbal
10258	1	140.51	Austria	Graz
10259	4	3.25	Mexico	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

How to set the `IsPrimaryKey` for auto generated columns when editing is enabled:

Using the DataBound event, you can set the IsPrimaryKey value as true by two ways. The following code example demonstrates the above behavior.

1. If primary key "column index" is known then refer to the following code example.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit =>
{
edit.AllowEditing(true);
})
.ClientSideEvents(eve => { eve.DataBound("dataBound"); })
)
<script type="text/javascript">
function dataBound(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var column = args.model.columns[0];
//(or)
var column = this.getColumnByIndex(0);
column.isPrimaryKey = true;
//Here columns method used to update the particular column
this.columns(column, "update");
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

2. If primary key "column field name" is known then refer to the following code example.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit =>
{
edit.AllowEditing(true);
})
.ClientSideEvents(eve => { eve.DataBound("dataBound"); })
)
<script type="text/javascript">
function dataBound(args) {
var column = this.getColumnByField("OrderID");
column.isPrimaryKey = true;
//Here columns method used to update the particular column
this.columns(column, "update");
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

```
function dataBound(args) {
var column = this.getColumnByField("OrderID");
column.isPrimaryKey = true;
//Here columns method used to update the particular column
this.columns(column, "update");
}
```

Headers

HeaderText

It represents the title for particular column. To enable header text, set the HeaderText property of Columns. The following code example describes the above behavior.

Note: If HeaderText is not defined then the Field name is considered as header text for that particular column. If Field name and HeaderText also not defined then the column is rendered with "empty" header text.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("Freight").HeaderText("Freight").Add();
col.Field("ShipCountry").HeaderText("Country").Add();
col.Field("ShipCity").HeaderText("City").Add();
}))
```

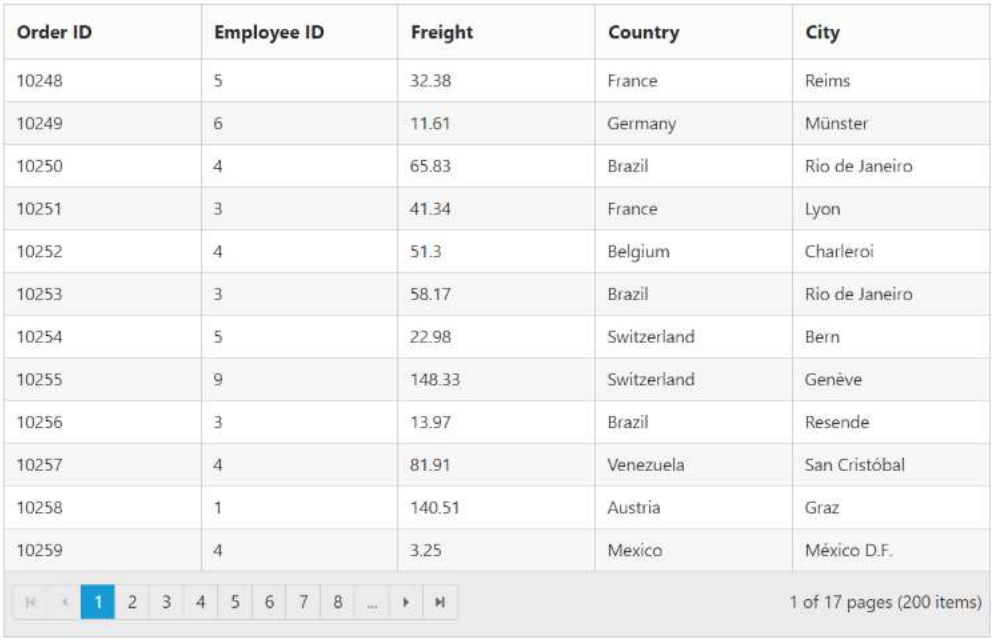
C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

The following output is displayed as a result of the above code example.



Order ID	Employee ID	Freight	Country	City
10248	5	32.38	France	Reims
10249	6	11.61	Germany	Münster
10250	4	65.83	Brazil	Rio de Janeiro
10251	3	41.34	France	Lyon
10252	4	51.3	Belgium	Charleroi
10253	3	58.17	Brazil	Rio de Janeiro
10254	5	22.98	Switzerland	Bern
10255	9	148.33	Switzerland	Genève
10256	3	13.97	Brazil	Resende
10257	4	81.91	Venezuela	San Cristóbal
10258	1	140.51	Austria	Graz
10259	4	3.25	Mexico	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Header Text alignment

Align the header text of column header using the HeaderTextAlign property of the Columns. There are four possible ways to align header text, they are.

1. Right
2. Left
3. Center
4. Justify

Note: For the HeaderTextAlign property you can assign enum value (TextAlign.Right).

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderTextAlign(TextAlign.Right).HeaderText("Order
ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("Freight").HeaderText("Freight").Add();
col.Field("ShipCountry").HeaderText("Country").Add();
col.Field("ShipCity").HeaderText("City").Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Order ID	Employee ID	Freight	Country	City
10248	5	32.38	France	Reims
10249	6	11.61	Germany	Münster
10250	4	65.83	Brazil	Rio de Janeiro
10251	3	41.34	France	Lyon
10252	4	51.3	Belgium	Charleroi
10253	3	58.17	Brazil	Rio de Janeiro
10254	5	22.98	Switzerland	Bern
10255	9	148.33	Switzerland	Genève
10256	3	13.97	Brazil	Resende
10257	4	81.91	Venezuela	San Cristóbal
10258	1	140.51	Austria	Graz
10259	4	3.25	Mexico	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Header Template

The template design that applies on for the column header. To render template, use the HeaderTemplateID property of Columns.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this link.

Note: It's a standard way to enclose the template within the script tag with type as text/x-jsrender.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderTemplateID("#employeeTemplate").Add();
col.Field("Freight").HeaderText("Freight").Add();
col.Field("ShipCountry").HeaderText("Country").Add();
col.Field("ShipCity").HeaderText("City").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

```
<script id="employeeTemplate" type="text/x-jsrender">
Employee ID
<span class="e-userlogin e-icon employee"></span>
</script>
```

The following output is displayed as a result of the above code example.

Order ID	Employee ID	Freight	Country	City
10248	5	32.38	France	Reims
10249	6	11.61	Germany	Münster
10250	4	65.83	Brazil	Rio de Janeiro
10251	3	41.34	France	Lyon
10252	4	51.3	Belgium	Charleroi
10253	3	58.17	Brazil	Rio de Janeiro
10254	5	22.98	Switzerland	Bern
10255	9	148.33	Switzerland	Genève
10256	3	13.97	Brazil	Resende
10257	4	81.91	Venezuela	San Cristóbal
10258	1	140.51	Austria	Graz
10259	4	3.25	Mexico	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Text alignment

You can align both content and header text of particular column using the TextAlign property. There are four possible ways to align content and header text of column, they are.

1. Right
2. Left
3. Center
4. Justify

Note: 1. For TextAlign property you can assign the enum value (TextAlign.Right).

2. The TextAlign property will affect both content and header text of the grid.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").TextAlign(TextAlign.Right).Add();
col.Field("EmployeeID").TextAlign(TextAlign.Right).Add();
col.Field("Freight").TextAlign(TextAlign.Right).Add();
col.Field("ShipCountry").TextAlign(TextAlign.Center).Add();
col.Field("ShipCity").TextAlign(TextAlign.Justify).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	Freight	ShipCountry	ShipCity
10248	5	32.38	France	Reims
10249	6	11.61	Germany	Münster
10250	4	65.83	Brazil	Rio de Janeiro
10251	3	41.34	France	Lyon
10252	4	51.3	Belgium	Charleroi
10253	3	58.17	Brazil	Rio de Janeiro
10254	5	22.98	Switzerland	Bern
10255	9	148.33	Switzerland	Genève
10256	3	13.97	Brazil	Resende
10257	4	81.91	Venezuela	San Cristóbal
10258	1	140.51	Austria	Graz
10259	4	3.25	Mexico	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Format

Format is the process of customizing the particular column data with specified jQuery recognized globalize formats, such as currency, numeric, decimal, percentage or dates. To specify the globalize format, by using Format property of Columns.

The Format value should be wrapped within "{0:" and "}". (For ex: "{0:C3}"). The data format strings available for the Date and Number types.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("Freight").Format("{0:C2}").Add();
col.Field("OrderDate").Format("{0:dd/MM/yyyy}").Add();
col.Field("ShipCity").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	Freight	OrderDate	ShipCity
10248	5	$32.38	04/07/1996	Reims
10249	6	$11.61	05/07/1996	Münster
10250	4	$65.83	08/07/1996	Rio de Janeiro
10251	3	$41.34	08/07/1996	Lyon
10252	4	$51.30	09/07/1996	Charleroi
10253	3	$58.17	10/07/1996	Rio de Janeiro
10254	5	$22.98	11/07/1996	Bern
10255	9	$148.33	12/07/1996	Genève
10256	3	$13.97	15/07/1996	Resende
10257	4	$81.91	16/07/1996	San Cristóbal
10258	1	$140.51	17/07/1996	Graz
10259	4	$3.25	18/07/1996	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Width

You can specify the width for the particular column by setting Width property of Columns as in pixel (ex: 100) or in percentage (ex: 40%).

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Width("10%").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").Width("15%").Add();
col.Field("Freight").Width(100).Add();
col.Field("ShipCity").Width(150).Add();
col.Field("ShipCountry").Width(100).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	Freight	ShipCity	ShipCountry
10248	5	32.38	Reims	France
10249	6	11.61	Münster	Germany
10250	4	65.83	Rio de Janeiro	Brazil
10251	3	41.34	Lyon	France
10252	4	51.3	Charleroi	Belgium
10253	3	58.17	Rio de Janeiro	Brazil
10254	5	22.98	Bern	Switzerland
10255	9	148.33	Genève	Switzerland
10256	3	13.97	Resende	Brazil
10257	4	81.91	San Cristóbal	Venezuela
10258	1	140.51	Graz	Austria
10259	4	3.25	México D.F.	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Resizing

The AllowResizing property enables the grid to set the width to columns based on resizing the grid column manually.

Copyright © 2001 - 2020 Syncfusion Inc.

Resizing modes

ResizeMode property of ResizeSettings is used to change the resizing modes. It indicates whether to define mode of resizing.

Name	Description
Normal	New column size will be adjusted by all other Columns
NextColumn	New column Size will be adjusted using next column
Control	New column Size will be adjusted using entire control

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<EmployeeView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowResizing()
.ResizeSettings(emp => { emp.ResizeMode(ResizeMode.NextColumn); })
.Columns(col =>
{
col.Field("ShipCity").HeaderText("Ship
City").IsPrimaryKey(true).Width(80).Add();
col.Field("ShipPostalCode").HeaderText("Ship Postal Code").Width(40).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(40).Add();
col.Field("ShipAddress").HeaderText("Ship Address").Width(100).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Resize to fit

The AllowResizeToFit property enable the grid to set width to columns based on maximum width of the particular column's content to facilitate full visibility of data in all the grid rows and this automatic behavior is applicable only for the columns which does not have width specified.

On columns where "width is defined", double click on the particular column header's resizer symbol to resize the column to show the whole text. For example, refer to the "ShipCity" column in the below code snippet and output screen shot.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowResizeToFit()
.Columns(col =>
{
col.Field("OrderID").Width(100).Add();
col.Field("EmployeeID").Add();
col.Field("Freight").Width(75).Add();
col.Field("ShipCity").Width(50).Add();
col.Field("ShipAddress").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	Freight	ShipCity	ShipAddress
10248	5	32.38	Reims	59 rue de l'Abbaye
10249	6	11.61	Münster	Luisenstr. 48
10250	4	65.83	Rio de Ja	Rua do Paço, 67
10251	3	41.34	Lyon	2, rue du Commerce
10252	4	51.3	Charleroi	Boulevard Tirou, 255
10253	3	58.17	Rio de Ja	Rua do Paço, 67
10254	5	22.98	Bern	Hauptstr. 31
10255	9	148.33	Genève	Starenweg 5
10256	3	13.97	Resende	Rua do Mercado, 12
10257	4	81.91	San Cristó	Carrera 22 con Ave. Carlos Soublette #8-35
10258	1	140.51	Graz	Kirchgasse 6
10259	4	3.25	México D	Sierras de Granada 9993

Reorder

Reordering can be done by drag and drop of the particular column header from one index to another index within the grid. Reordering can be enabled by setting the AllowReordering property as true.

The following code example describes the above behavior.

RAZOR

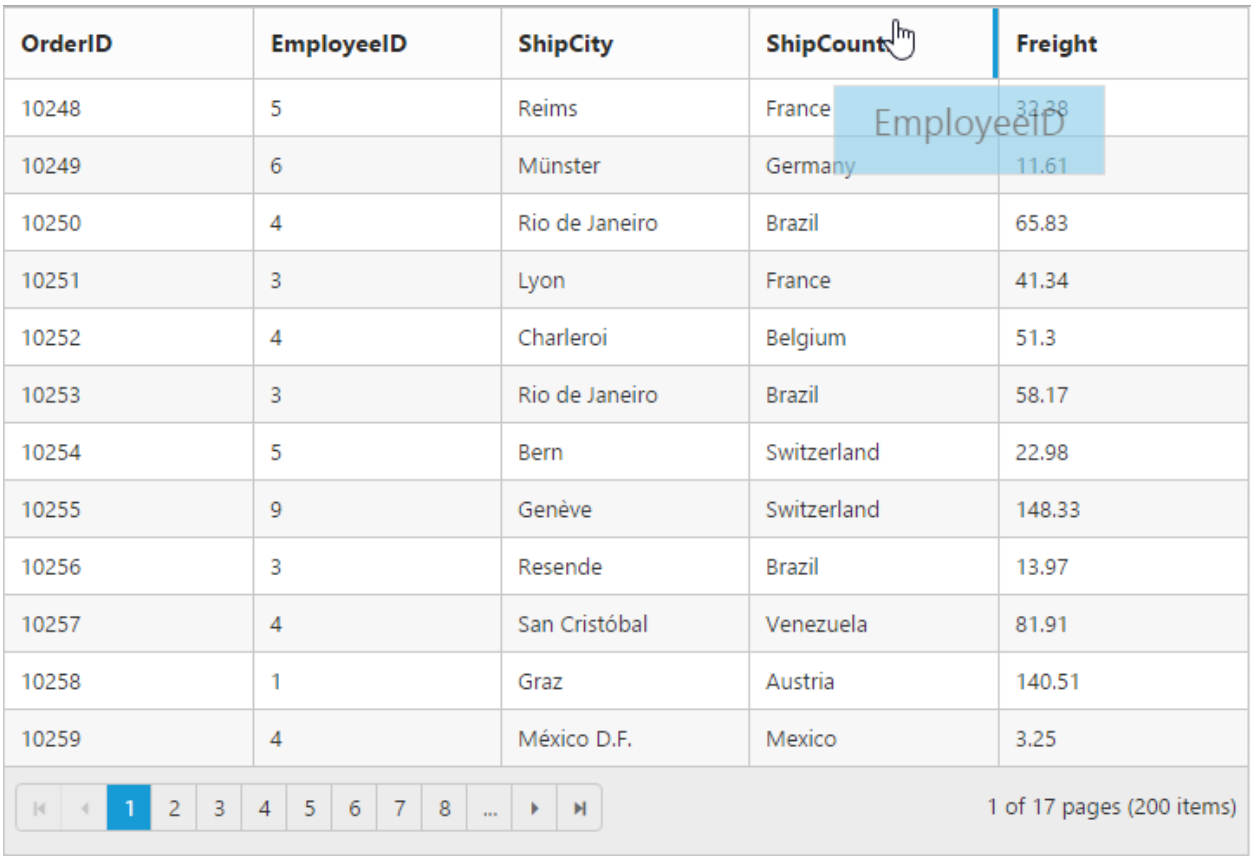
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowReordering()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
```

```
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

Visibility

You can hide particular column in grid view by setting the Visible property of it as false.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("EmployeeID").Add();
col.Field("OrderID").Visible(false).Add();
col.Field("Freight").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

EmployeeID	Freight	ShipCity	ShipCountry
5	32.38	Reims	France
6	11.61	Münster	Germany
4	65.83	Rio de Janeiro	Brazil
3	41.34	Lyon	France
4	51.3	Charleroi	Belgium
3	58.17	Rio de Janeiro	Brazil
5	22.98	Bern	Switzerland
9	148.33	Genève	Switzerland
3	13.97	Resende	Brazil
4	81.91	San Cristóbal	Venezuela
1	140.51	Graz	Austria
4	3.25	México D.F.	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Unbound Column

You can define the unbound columns in grid by not defining the Field property for that particular. Value for this columns can be populated either by manually using the QueryCellInfo event or by using the ColumnTemplate.

Note: Editing, grouping, filtering, sorting, summary and searching support are not available for unbound columns.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowDeleting(); })
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("Freight").Add();
col.HeaderText("").Format("<a onclick=\"click(this)\"
href=#>Delete</a>").Add();
})
)
```

C#

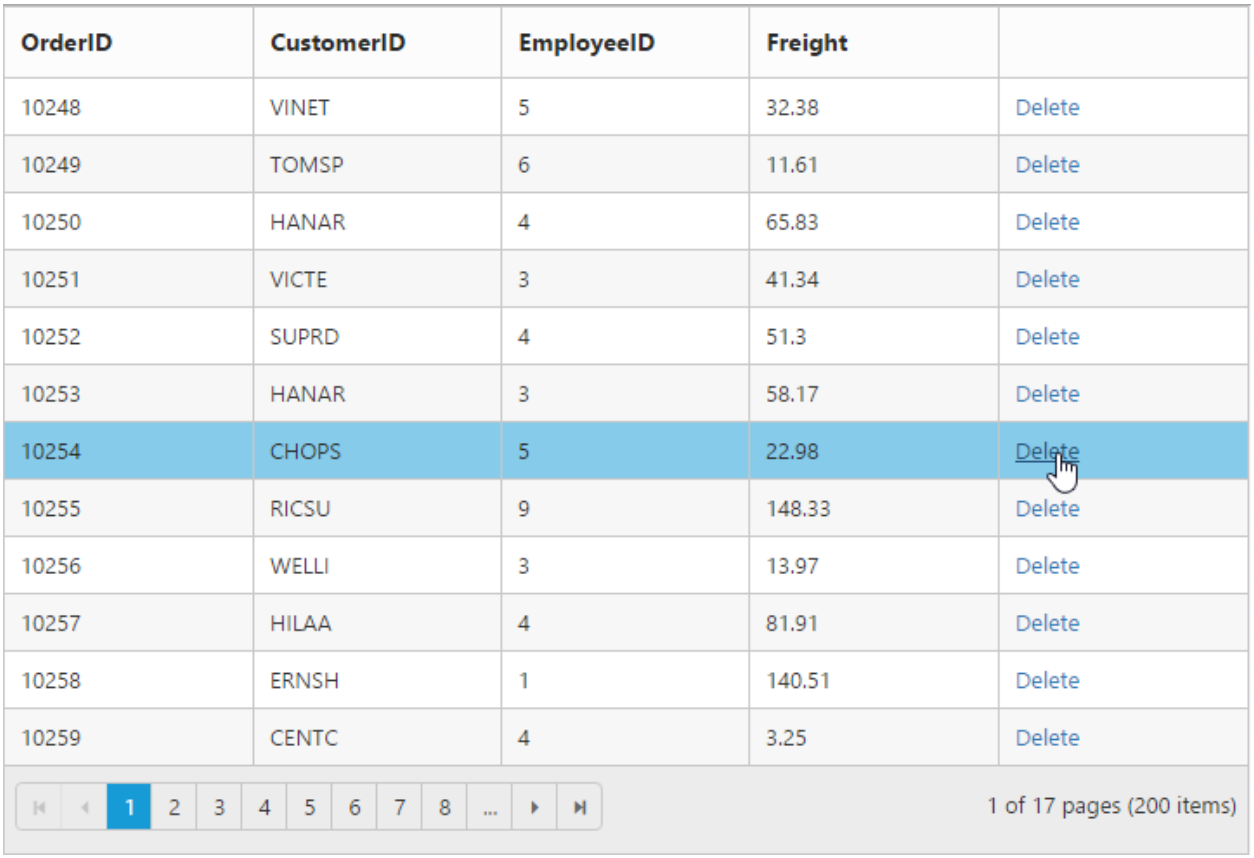
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/javascript">
function click(e) {
var obj = $("#FlatGrid").data("ejGrid");
obj.deleteRecord("OrderID", obj.getSelectedRecords()[0]);
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	CustomerID	EmployeeID	Freight	
10248	VINET	5	32.38	Delete
10249	TOMSP	6	11.61	Delete
10250	HANAR	4	65.83	Delete
10251	VICTE	3	41.34	Delete
10252	SUPRD	4	51.3	Delete
10253	HANAR	3	58.17	Delete
10254	CHOPS	5	22.98	Delete
10255	RICSU	9	148.33	Delete
10256	WELLI	3	13.97	Delete
10257	HILAA	4	81.91	Delete
10258	ERNSH	1	140.51	Delete
10259	CENTC	4	3.25	Delete

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Column Template

HTML templates can be specified in the `Template` property of the particular column as a string (html element) or ID of the template's HTML element.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this link.

For template manipulation using JavaScript, either you can use JsRender helper function or `TemplateRefresh` grid event. For more information on `TemplateRefresh` event, refer this link.

Note: If `Field` is not specified, you will not able to perform editing, grouping, filtering, sorting, search and summary functionalities in particular column.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.PageSettings(page => { page.PageSize(4); })
.Columns(col =>
{
col.HeaderText("Photo").Template("<img style='width: 75px; height: 70px'
src='/13.2.0.29/themes/web/images/employees/{{"{{"}}:EmployeeID{{}}}}.png'
alt='{{"{{"}}:EmployeeID{{}}}}.png' />").Add();
col.Field("EmployeeID").Add();
col.Field("FirstName").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("LastName").Add();
col.Field("Country").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Photo	EmployeeID	FirstName	LastName	Country
	1	Nancy	Davolio	USA
	2	Andrew	Fuller	USA
	3	Janet	Leverling	USA
	4	Margaret	Peacock	USA

1 2 3 1 of 3 pages (9 items)

Controlling Grid actions

You can control the grid actions of a particular column by setting the AllowSorting,AllowGrouping, AllowFiltering and AllowEditing properties of it as false.

The following code example describes the above behavior.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowResizing()
.AllowSorting()
.AllowGrouping()
.AllowFiltering()
.EditSettings(edit => { edit.AllowEditing(); })
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").AllowEditing(false).AllowResizing(false).AllowSortin
g(false).AllowGrouping(false).AllowFiltering(false).Add();
col.Field("Freight").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
}) )
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

Read only

To make a column as "read-only" then set AllowEditing property of Columns as false.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowEditing(); })
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").AllowEditing(false).Add();
col.Field("Freight").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
})
)
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	Freight	ShipCity	ShipCountry
10248	5	32.38	Reims	France
10249	6	11.61	Münster	Germany
10250	4	65.83	Rio de Janeiro	Brazil
10251	3	41.34	Lyon	France
10252	4	51.3	Charleroi	Belgium
10253	3	58.17	Rio de Janeiro	Brazil
10254	5	22.98	Bern	Switzerland
10255	9	148.33	Genève	Switzerland
10256	3	13.97	Resende	Brazil
10257	4	81.91	San Cristóbal	Venezuela
10258	1	140.51	Graz	Austria
10259	4	3.25	México D.F.	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Expression Column

Expression column is possible only for Template column. You can use JsRender syntax in the template.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this link.

Note: This expression column is supported at read only mode.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
```

```
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("FoodName").Add();
col.Field("Protein").Add();
col.Field("Fat").Add();
col.Field("Carbohydrate").Add();
col.HeaderText("Calories In Take").Template("<span>{{"{{"}}:Protein * 4  +
Fat * 4 + Carbohydrate * 9 {{}}}}</span>").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().FoodInformation.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

FoodName	Protein	Fat	Carbohydrate	Calories In Take
CHEESE BURGER	15	15	28	372
PIZZA	15	9	39	447
CHICKEN NOODLE	4	2	9	105
YOGURT	10	2	43	435
BEEF SANDWICH	22	13	34	446
CHICKEN BURGER	15	10	25	325

1 1 of 1 pages (6 items)

Command Column

Default action buttons

Using Command column, you can add CRUD action buttons as one of the grid column, through Type property of the Commands. The Type property supports the below default UnboundType buttons.

1. Edit

Copyright © 2001 - 2020 Syncfusion Inc.

2. Save
3. Delete
4. Cancel

Through ButtonOptions property of Commands, you can specify all the button options which are supported by Essential Studio ASP.NET MVC Button control.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowEditing().AllowDeleting().AllowAdding();
})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").Add();
col.Field("Freight").EditType(EditingType.NumericEdit).Add();
col.Field("ShipCountry").Add();
col.HeaderText("Manage Records").Commands(command =>
{
command.Type(UnboundType.Edit)
.ButtonOptions(new Syncfusion.JavaScript.Models.ButtonProperties()
{
Text = "Edit"
}).Add();
command.Type(UnboundType.Delete)
.ButtonOptions(new Syncfusion.JavaScript.Models.ButtonProperties()
{
Text = "Delete"
}).Add();
command.Type(UnboundType.Save)
.ButtonOptions(new Syncfusion.JavaScript.Models.ButtonProperties()
{
Text = "Save"
}).Add();
command.Type(UnboundType.Cancel)
.ButtonOptions(new Syncfusion.JavaScript.Models.ButtonProperties()
{
Text = "Cancel"
}).Add();
}).Width(150).Add();
}) )
```

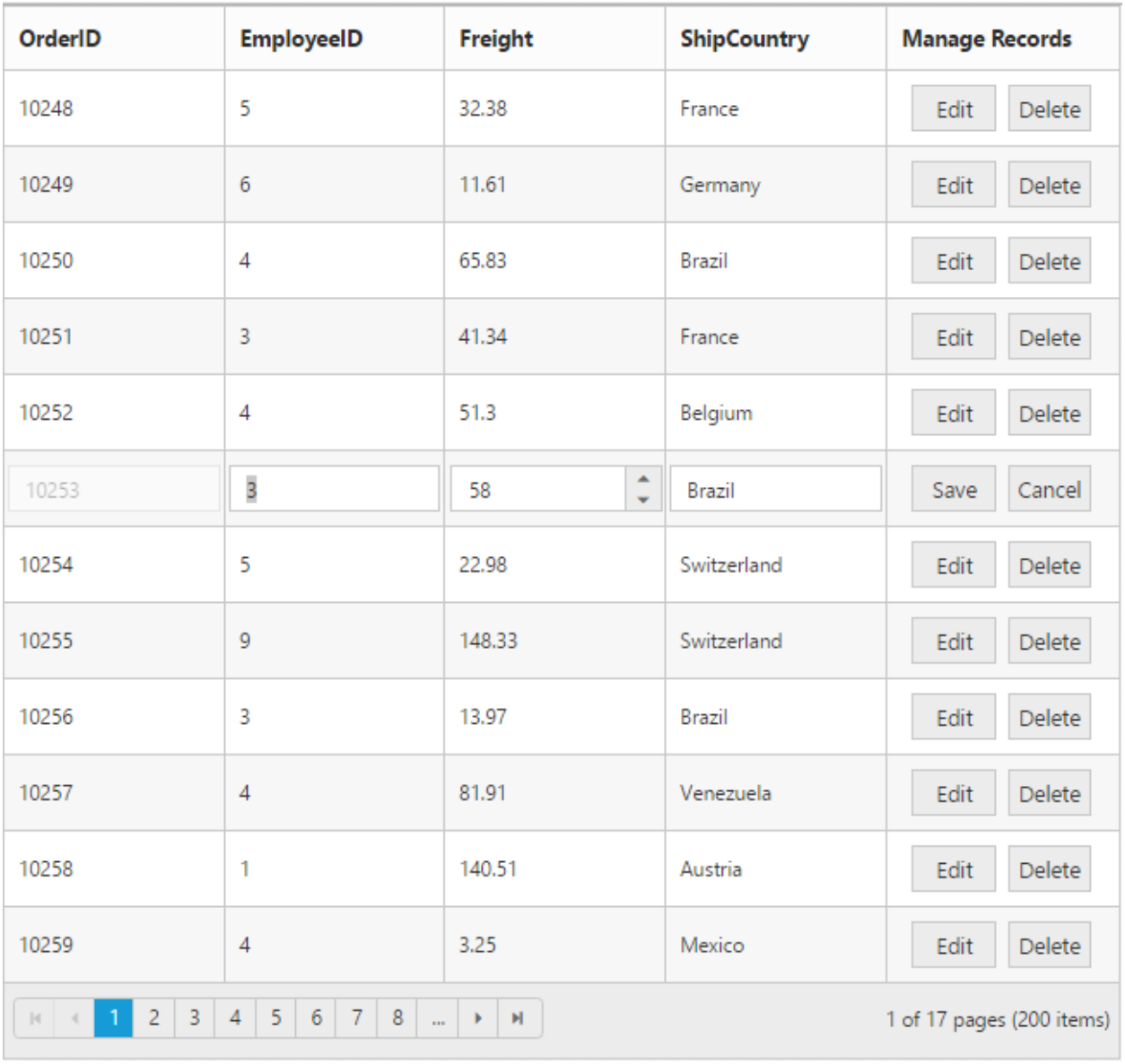
C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersView.ToList();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



OrderID	EmployeeID	Freight	ShipCountry	Manage Records
10248	5	32.38	France	Edit Delete
10249	6	11.61	Germany	Edit Delete
10250	4	65.83	Brazil	Edit Delete
10251	3	41.34	France	Edit Delete
10252	4	51.3	Belgium	Edit Delete
10253	3	58	Brazil	Save Cancel
10254	5	22.98	Switzerland	Edit Delete
10255	9	148.33	Switzerland	Edit Delete
10256	3	13.97	Brazil	Edit Delete
10257	4	81.91	Venezuela	Edit Delete
10258	1	140.51	Austria	Edit Delete
10259	4	3.25	Mexico	Edit Delete

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Custom buttons

You can add custom button in the command column by specifying the `Type` property of the `Commands` as `empty` or any other `string` instead of `enum` values.

Note: 1. For the Type property, assign any string value other than the default UnboundType.

2. In command column you can add only buttons.

The following code example describes the above behavior.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("EmployeeID").Add();
col.HeaderText("Employee Details").Commands(command =>
{
command.Type("detail")
.ButtonOptions(new Syncfusion.JavaScript.Models.ButtonProperties()
{
Text = "Details",
Width = "100px",
Click = "onClick"
}).Add();
})
.TextAlign(TextAlign.Center)
.Width(150)
.Add();
})
)
```

C#

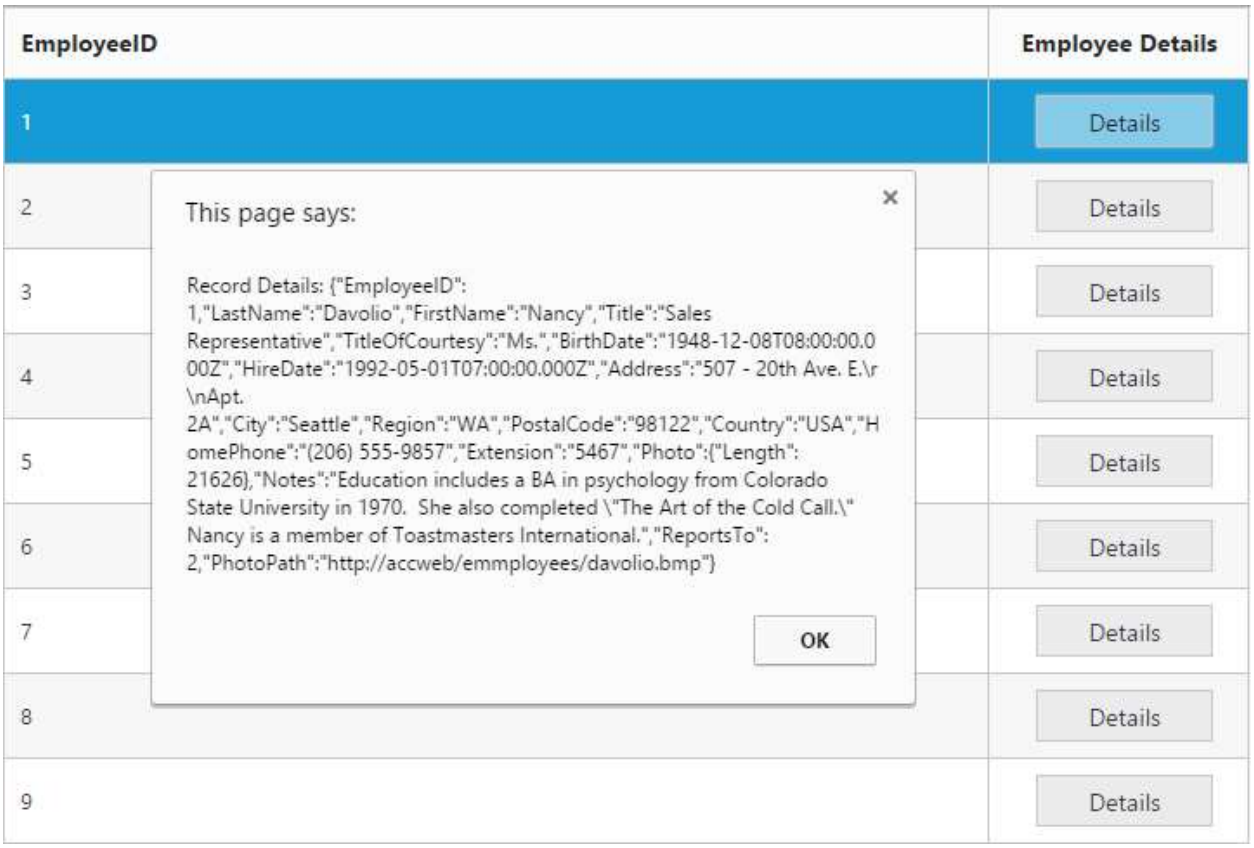
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersView.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/javascript">
function onClick(args) {
var grid = $("#FlatGrid").ejGrid("instance");
var index = this.element.closest("tr").index();
var record = grid.getCurrentViewData()[index];
alert("Record Details: " + JSON.stringify(record));
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Column Chooser

Column chooser contains all the columns which are defined in the Columns property, using this you can control the visibility of columns in grid. You can prevent to show the particular column in column chooser by setting the ShowInColumnChooser property of Columns as false. It can be shown in the right corner of grid. To enable column chooser, set the ShowColumnChooser property as true.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.ShowColumnChooser()
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").ShowInColumnChooser(false).Add();
col.Field("Freight").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
}) )
```

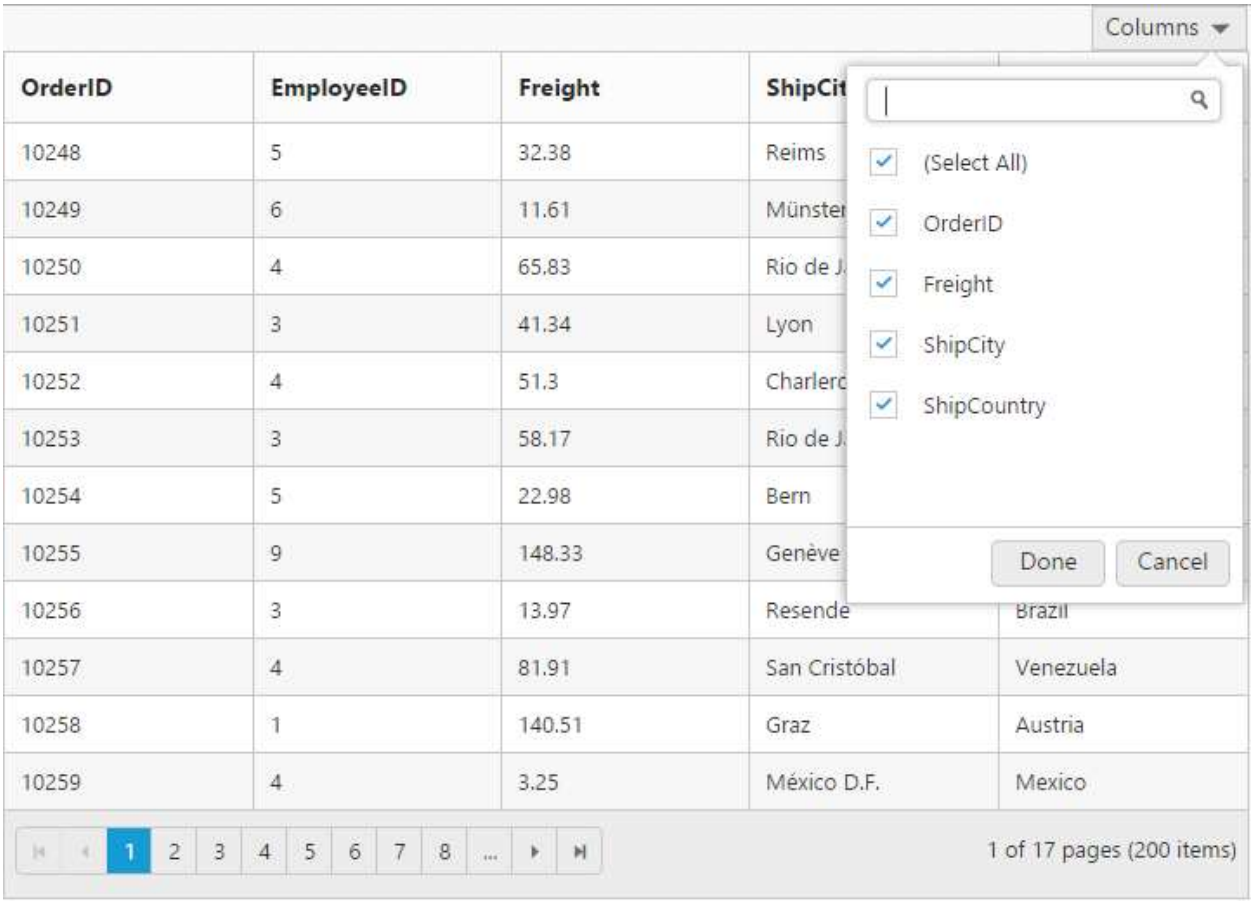
C#

```
namespace MVCSampleBrowser.Controllers
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersView.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Foreign Key Column

Lookup data source can be bound to DataSource property of Columns. Data field and text can be set using the ForeignKeyField and ForeignKeyValue property of Columns.

In the DataSource property, we can bound local and remote data.

Information: For foreign key column the sorting and grouping is based on the ForeignKeyField instead of ForeignKeyValue. You can refer Foreign Key Adaptor to sort and group the foreign key column based on ForeignKeyValue.

Note: In remote data, server should be configured to perform select and filter operations since the Grid will try to fetch required columns using select operation and the required data using filter operation.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: To render a Hierarchy Grid with different ForeignKeyField in parent and child table, click here.

The following code example describes the above behavior.

RAZOR

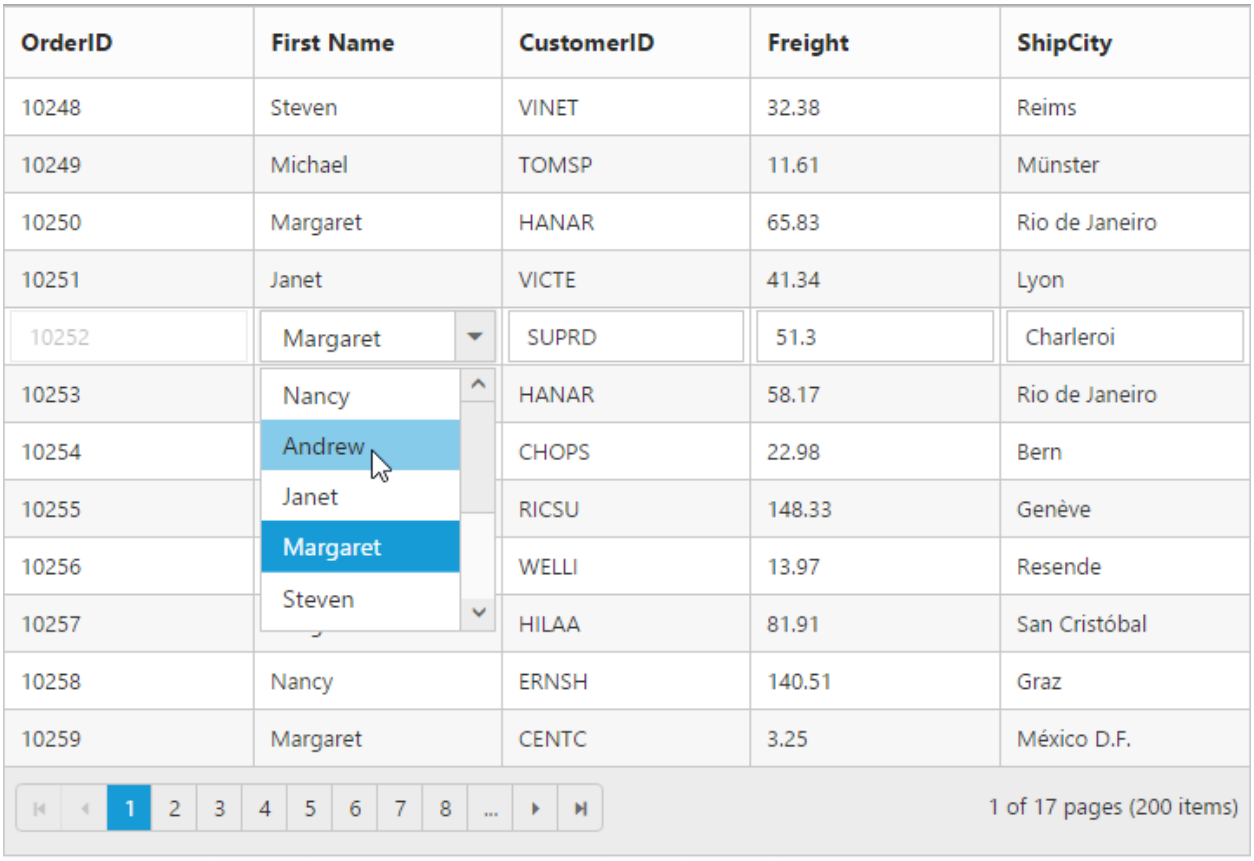
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource1)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").HeaderText("First
Name").ForeignKeyField("EmployeeID").ForeignKeyValue("FirstName").DataSource
((IEnumerable<object>)ViewBag.DataSource2).Add();
//(or)
col.Field("EmployeeID").HeaderText("First
Name").ForeignKeyField("EmployeeID").ForeignKeyValue("FirstName").DataSource
("http://mvc.syncfusion.com/Services/Northwnd.svc/Employees/").Add();
col.Field("CustomerID").Add();
col.Field("Freight").Add();
col.Field("ShipCity").Add();
}) )
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource1 = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource1 = DataSource1;
var DataSource2 = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.DataSource2 = DataSource2;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Customize column

You can customize the header and content of that particular column by the CssClass property of the column.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Add();
col.Field("EmployeeID").CssClass("customizes").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var DataSource1 = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource1;
return View();
}
}
}
```

CSS

```
.customizes.e-headercell {
background-color: #2382c3;
color: white;
font-family: 'Bell MT';
font-size: 20px;
}
.customizes.e-rowcell {
background-color: #ecedee;
font-family: 'Bell MT';
color: red;
font-size: 20px;
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	CustomerID	EmployeeID	Freight
10248	VINET	5	32.38
10249	TOMSP	6	11.61
10250	HANAR	4	65.83
10251	VICTE	3	41.34
10252	SUPRD	4	51.3
10253	HANAR	3	58.17
10254	CHOPS	5	22.98
10255	RICSU	9	148.33
10256	WELLI	3	13.97
10257	HILAA	4	81.91
10258	ERNSH	1	140.51
10259	CENTC	4	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Type

Used to define the type of the particular column data. If the Type property of Columns is not specified then its type is automatically defined based on the first row data of that column.

Note: The Type is needed for filtering feature when first row of the data is null or empty.

The available column data type is tabulated as follows.

Type	Description
string	Gets or sets the type of the column value as string.
number	Gets or sets the type of the column value as number.
date	Gets or sets the type of the column value as date.
datetime	Gets or sets the type of the column value as datetime.
boolean	Gets or sets the type of the column value as true or false.
guid	Gets or sets the type of the column value as guid.

Copyright © 2001 - 2020 Syncfusion Inc.

checkbox	Gets or sets the type of the column value as checkbox for row selection.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Type("string").Add();
col.Field("EmployeeID").Type("number").Add();
col.Field("Freight").Add();
col.Field("ShipCountry").Add();
}) )
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource1 = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource1;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	CustomerID	EmployeeID	Freight	ShipCountry
10248	VINET	5	32.38	France
10249	TOMSP	6	11.61	Germany
10250	HANAR	4	65.83	Brazil
10251	VICTE	3	41.34	France
10252	SUPRD	4	51.3	Belgium
10253	HANAR	3	58.17	Brazil
10254	CHOPS	5	22.98	Switzerland
10255	RICSU	9	148.33	Switzerland
10256	WELLI	3	13.97	Brazil
10257	HILAA	4	81.91	Venezuela
10258	ERNSH	1	140.51	Austria
10259	CENTC	4	3.25	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Column Layout

You can set the Grid's columns layout based on either Grid width or its columns width using the ColumnLayout property of Grid. There are two ways to set the column layout, they are.

1. Auto
2. Fixed

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.ColumnLayout(ColumnLayout.Fixed)
.Columns(col =>
{
col.Field("OrderID").Width(80).Add();
col.Field("EmployeeID").Width(80).Add();
col.Field("ShipCity").Width(90).Add();
col.Field("ShipName").Width(110).Add();
col.Field("ShipCountry").Width(100).Add();
col.Field("Freight").Width(80).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	Employeel	ShipCity	ShipName	ShipCountry	Freight
10248	5	Reims	Vins et alcools Ch	France	32.38
10249	6	Münster	Toms Spezialitäte	Germany	11.61
10250	4	Rio de Janeiro	Hanari Carnes	Brazil	65.83
10251	3	Lyon	Victuailles en stoc	France	41.34
10252	4	Charleroi	Suprêmes délices	Belgium	51.3
10253	3	Rio de Janeiro	Hanari Carnes	Brazil	58.17
10254	5	Bern	Chop-suey Chine:	Switzerland	22.98
10255	9	Genève	Richter Supermar	Switzerland	148.33
10256	3	Resende	Wellington Impor	Brazil	13.97
10257	4	San Cristóbal	HILARION-Abasto	Venezuela	81.91
10258	1	Graz	Ernst Handel	Austria	140.51
10259	4	México D.F.	Centro comercial	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Row

It represents the record details that are fetched from the datasource.

Row hover

You can see the mouse hovering effect on the corresponding Grid rows using the EnableRowHover property. By default its value is true.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.EnableRowHover(true)
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Default()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Details template

It provides a detailed view /additional information about each row of the grid. You can render any type of JsRender template and assign the script template id in the DetailsTemplate property. Also, you can change HTML elements in detail template row into JavaScript controls using the DetailsDataBound event.

On enabling details template, new column will be added in the grid with an expander button in it and that can be expanded or collapsed to show or hide the underlying details row respectively.

Note: It's a standard way to enclose the template within the script tag with type as "text/x-jsrender".

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<EmployeeView>("DetailTemplate")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.DetailsTemplate("#tabGridContents")
.ClientSideEvents(eve => { eve.DetailsDataBound("detailGridData"); })
.Columns(col =>
{
col.Field("EmployeeID").Add();
col.Field("FirstName").Add();
col.Field("Title").Add();
col.Field("City").Add();
col.Field("Country").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult DetailTemplate()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script id="tabGridContents" type="text/x-jsrender">
<div class="tabcontrol" id="Test">
<ul>
<li><a href="#gridTab{{"{{"}}:EmployeeID {{}}}}">Stock Grid</a></li>
</ul>
<div id="gridTab{{"{{"}}:EmployeeID {{}}}}">
<div id="detailGrid">
</div>
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
<script src="~/Scripts/jsondata.min.js"></script>
<script type="text/javascript">
function detailGridData(e) {
var filteredData = e.rowData["EmployeeID"];
// the datasource "window.ordersView" is referred from jsondata.min.js
var data =
ej.DataManager(window.ordersView).executeLocal(ej.Query().where("EmployeeID"
, "equal", parseInt(filteredData), true).take(5));
e.detailsElement.find("#detailGrid").ejGrid({
dataSource: data,
columns: [
{field: "OrderID"},
{field: "EmployeeID"},
{field: "ShipCity"},
{field: "ShipCountry"},
{field: "Freight"}
]
});
e.detailsElement.find(".tabcontrol").ejTab();
}
</script>
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.

	EmployeeID	FirstName	Title	City	Country
◢	1	Nancy	Sales Representative	Seattle	USA

Stock Grid

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10835	1	Berlin	Germany	69.53
10952	1	Berlin	Germany	40.42
10677	1	México D.F.	Mexico	4.03
10558	1	Colchester	UK	72.97
10453	1	Colchester	UK	25.36

▸	2	Andrew	Vice President, Sales	Tacoma	USA
▸	3	Janet	Sales Representative	Kirkland	USA
▸	4	Margaret	Sales Representative	Redmond	USA
▸	5	Steven	Sales Manager	London	UK
▸	6	Michael	Sales Representative	London	UK
▸	7	Robert	Sales Representative	London	UK
▸	8	Laura	Inside Sales Coordinat	Seattle	USA
▸	9	Anne	Sales Representative	London	UK

Row Template

Row template enables you to set the customized look and behavior to grid all rows. The `RowTemplate` property can be used bind the `id` of HTML template.

The following code example describes the previous behavior.

JS

```
<script id="templateData" type="text/x-jsrender">
<tr>
<td class="photo">
<img style="width:130px;height: 160px"
src="/13.2.0.29/themes/web/images/employees/{{"{{"}}:EmployeeID {{}}}}.png"
alt="{{"{{"}}:EmployeeID {{}}}}" />
</td>
<td class="details">
<table class="CardTable" cellpadding="3" cellspacing="2">
<colgroup>
<col width="50%">
<col width="50%">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</colgroup>
<tbody>
<tr>
<td class="CardHeader">First Name</td>
<td>{{"{{"}}:FirstName {{}}}} </td>
</tr>
<tr>
<td class="CardHeader">Last Name</td>
<td>{{"{{"}}:LastName {{}}}}</td>
</tr>
<tr>
<td class="CardHeader">Title</td>
<td>{{"{{"}}:Title {{}}}}</td>
</tr>
</tbody>
</table>
</td>
</tr>
</script>
```

CSS

```
<style>
.photo img {
width: 130px;
}
.photo, .details {
border-color: #c4c4c4;
border-style: solid;
}
.photo {
border-width: 1px 0px 0px 0px;
}
.details {
border-width: 1px 0px 0px 1px;
}
.details > table {
width: 100%;
}
.CardHeader {
font-weight: bolder;
}
</style>
```

RAZOR

```
@(Html.EJ().Grid<EmployeeView>("RowTemplate")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Query("new ej.Query().take(2)")
.Columns(col =>
{
col.HeaderText("Photo").Width(30).Add();
col.HeaderText("Employee Details").Width(70).Add();
})
.RowTemplate("#templateData")    // row template
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult RowTemplate()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Photo	Employee Details	
	First Name Last Name Title	Nancy Davolio Sales Representative
	First Name Last Name Title	Andrew Fuller Vice President, Sales

Alternate row styling

Alternate row styling enhances the readability of grid rows by setting different background color for every alternate row. You can enable the alternative row styling in grid by using the EnableAltRow property.

By default its value is true, so the following code example describes the how to turn off alternate row behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.EnableAltRow(false)
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Default()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Drag-and-Drop

The grid rows can be reordered, dropped to another grid or custom control by enabling the AllowRowDragAndDrop grid property.

Note: To enable selection of multiple rows by mouse dragging on grid rows, the SelectionType property of grid must be set to the “multiple”.

Reorder

By simply enabling the property AllowRowDragAndDrop, grid rows can be reordered within the same grid.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.datasource).Adaptor(AdaptorType
.RemoteSaveAdaptor))
.AllowSorting()
.AllowPaging()
.SelectionType(SelectionType.Multiple)
.AllowRowDragAndDrop()
.RowDropSettings(drop => drop.RowDropMapper("RowDropHandler"))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
```

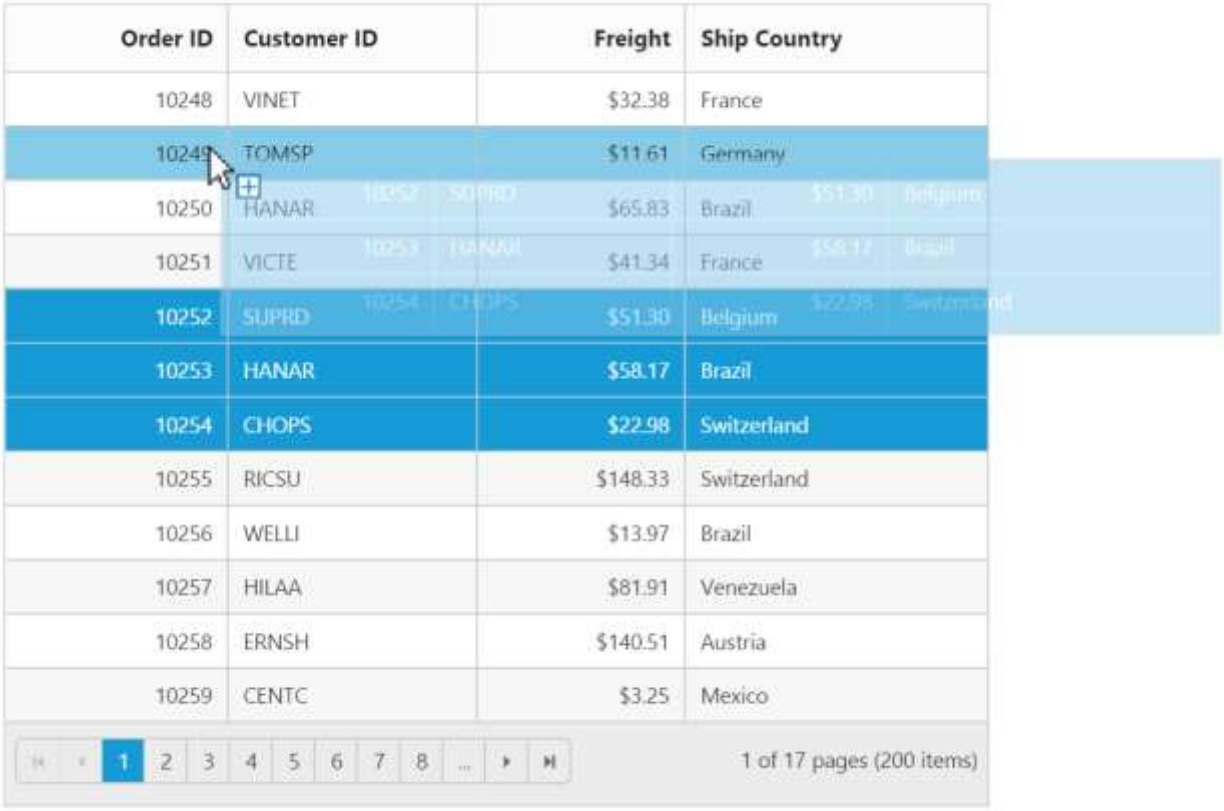
C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Default()
{
ViewBag.datasource =OrderRepository.GetAllRecords();
return View();
}
public ActionResult RowDropHandler(List<EditableOrder> changed)
{
JavaScriptSerializer ser = new JavaScriptSerializer();
RowDropModel dropDetails =
(RowDropModel)ser.Deserialize(Request.Headers["rowDropDetails"],
typeof(RowDropModel));
var count = 0;
foreach (var item in changed)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
EditableOrder result = OrderRepository.GetAllRecords().Where(o => o.OrderID
== item.OrderID).FirstOrDefault();
OrderRepository.GetAllRecords().Remove(result);
OrderRepository.GetAllRecords().Insert(dropDetails.DestinationRowIndex +
count, item);
count++;
}
return Json(changed, JsonRequestBehavior.AllowGet);
}
}
}
```

The following output is displayed before reordering rows.

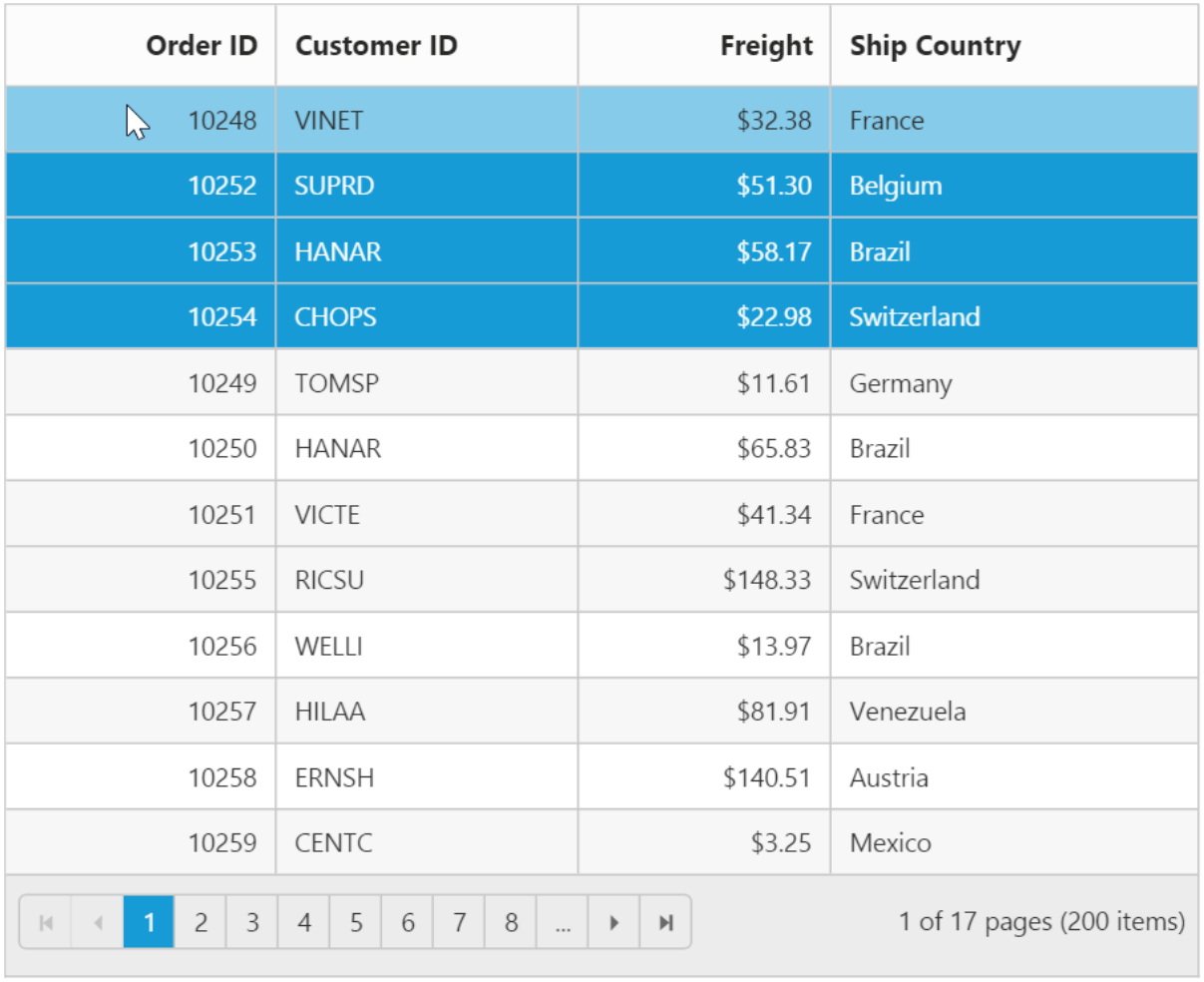


Order ID	Customer ID	Freight	Ship Country
10248	VINET	$32.38	France
10249	TOMSP	$11.61	Germany
10250	HANAR	$65.83	Brazil
10251	VICTE	$41.34	France
10252	SUPRD	$51.30	Belgium
10253	HANAR	$58.17	Brazil
10254	CHOPS	$22.98	Switzerland
10255	RICSU	$148.33	Switzerland
10256	WELLI	$13.97	Brazil
10257	HILAA	$81.91	Venezuela
10258	ERNSH	$140.51	Austria
10259	CENTC	$3.25	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

The following output is displayed after reordering rows.

Copyright © 2001 - 2020 Syncfusion Inc.



Order ID	Customer ID	Freight	Ship Country
10248	VINET	$32.38	France
10252	SUPRD	$51.30	Belgium
10253	HANAR	$58.17	Brazil
10254	CHOPS	$22.98	Switzerland
10249	TOMSP	$11.61	Germany
10250	HANAR	$65.83	Brazil
10251	VICTE	$41.34	France
10255	RICSU	$148.33	Switzerland
10256	WELLI	$13.97	Brazil
10257	HILAA	$81.91	Venezuela
10258	ERNSH	$140.51	Austria
10259	CENTC	$3.25	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Grid-to-grid

To drag and drop rows between two grid, enable the grid property AllowRowDragAndDrop and specify the target grid ID in DropTargetID property of the grid RowDropSettings.

Dragged and Dropped rows can be mapped to server-side using the RowDragMapper and RowDropMapper property of grid RowDropSettings.

The following code example describes the previous behavior.

RAZOR

```
<div style="float:left;width:49%">
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.datasource).Adaptor(AdaptorType
.RemoteSaveAdaptor))
.AllowSorting()
.AllowPaging()
.SelectionType(SelectionType.Multiple)
.AllowRowDragAndDrop()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.RowDropSettings(drop =>
drop.RowDragMapper("RowDragHandler").RowDropMapper("RowDropHandler").DropTar
getID("#DesignationGrid"))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
</div>
<div style="float:right;width:49%">
@(Html.EJ().Grid<OrdersView>("DesignationGrid")
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.datasource2).Adaptor(AdaptorTyp
e.RemoteSaveAdaptor))
.AllowSorting()
.AllowPaging()
.SelectionType(SelectionType.Multiple)
.AllowRowDragAndDrop()
.RowDropSettings(drop =>
drop.RowDragMapper("RowDragHandler2").RowDropMapper("RowDropHandler2").DropT
argetID("#Grid"))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
</div>
```

C#

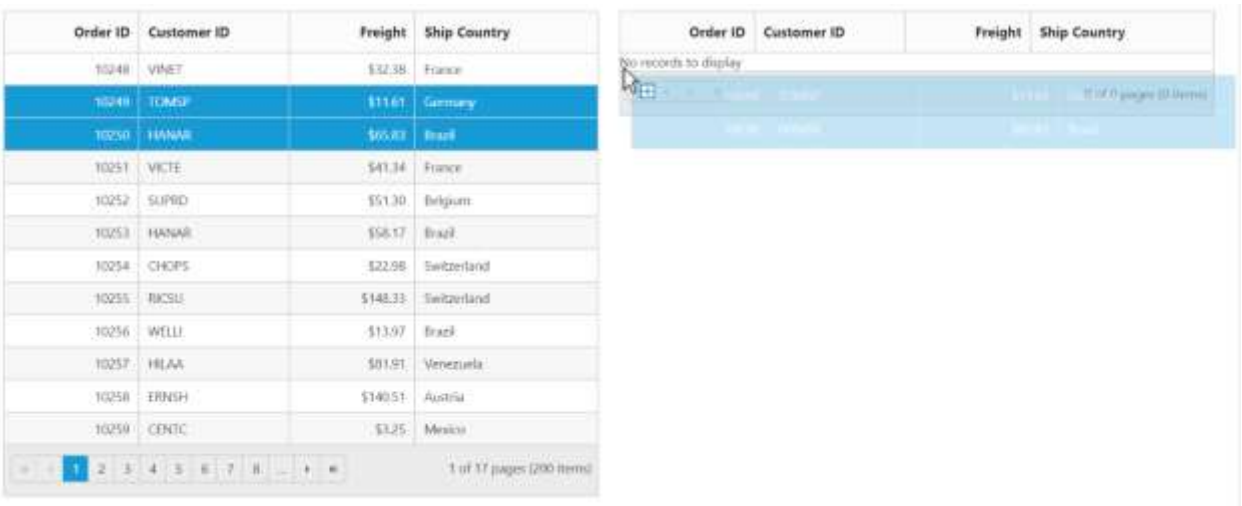
```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
JavaScriptSerializer ser = new JavaScriptSerializer();
public ActionResult DragAndDrop()
{
ViewBag.datasource2 = OrderRepository.GetAllRecords2();
ViewBag.datasource = OrderRepository.GetAllRecords();
return View();
}
public ActionResult RowDragHandler(List<EditableOrder> deleted)
{
OrderRepository.Delete(deleted);
```

Copyright © 2001 - 2020 Syncfusion Inc.

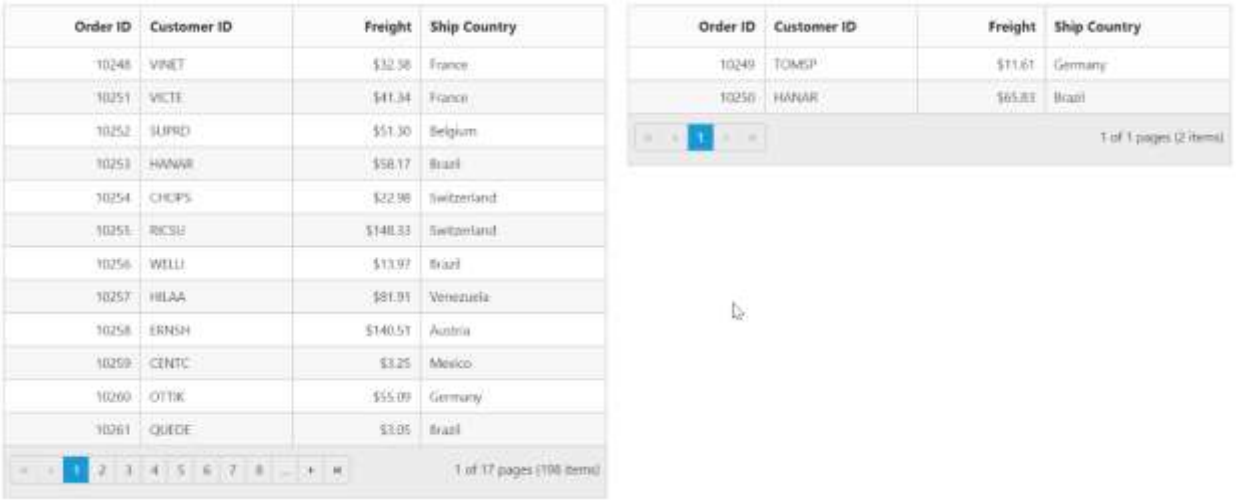
```
return Json(deleted, JsonRequestBehavior.AllowGet);
}
public ActionResult RowDragHandler2(List<EditableOrder> deleted)
{
OrderRepository.Delete2(deleted);
return Json(deleted, JsonRequestBehavior.AllowGet);
}
public ActionResult RowDropHandler(List<EditableOrder> added)
{
RowDropModel dropDetails =
(RowDropModel)ser.Deserialize(Request.Headers["rowDropDetails"],
typeof(RowDropModel));
var count = 0;
var data = OrderRepository.GetAllRecords();
if (added != null)
{
foreach (var item in added)
{
data.Insert(dropDetails.DestinationRowIndex + count, item);
count++;
}
}
return Json(added, JsonRequestBehavior.AllowGet);
}
public ActionResult RowDropHandler2(List<EditableOrder> added)
{
RowDropModel dropDetails =
(RowDropModel)ser.Deserialize(Request.Headers["rowDropDetails"],
typeof(RowDropModel));
var count = 0;
var data = OrderRepository.GetAllRecords2();
foreach (var item in added)
{
data.Insert(dropDetails.DestinationRowIndex + count, item);
count++;
}
return Json(added, JsonRequestBehavior.AllowGet);
}
}
}
```

The following output is displayed before dropping grid rows.

Copyright © 2001 - 2020 Syncfusion Inc.



The following output is displayed after dropping grid rows.



Grid-to-custom control

You can also drag and drop grid rows to any custom control. For instance, let it be a form.

Enable the grid property AllowRowDragAndDrop and specify the target form element ID in DropTargetID property of grid RowDropSettings.

On dropping the grid records on form element, grid event “RowDrop” would be triggered by which we can populate input elements in form.

The following code example describes the previous behavior.

RAZOR

```
<div style="float:left;width:49%">
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.datasource).Adaptor(AdaptorType
.RemoteSaveAdaptor))
.AllowSorting()
.AllowPaging()
.AllowRowDragAndDrop()
```

```
.RowDropSettings(drop => drop.DropTargetID("#DropForm"))
.ClientSideEvents(eve => eve.RowDrop("rowDropHandler"))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
}))
</div>
<div style="float:right;width:38%">
<form role="form" id="dropForm" style="width:98%">
<fieldset style="text-align:center; font-weight:700"><legend>Record
Details</legend></fieldset>
<div class="form-group row">
<label for="OrderID">Order ID:</label>
<input class="form-control" name="OrderID">
</div>
<div class="form-group row">
<label for="CustomerID">Customer ID:</label>
<input name="CustomerID" class="form-control">
</div>
<div class="form-group row">
<label for="EmployeeID">Employee ID:</label>
<input name="EmployeeID" class="form-control">
</div>
<div class="form-group row">
<label for="Freight">Freight:</label>
<input name="Freight" class="form-control">
</div>
<div class="form-group row">
<label for="ShipCity">Ship City:</label>
<input name="ShipCity" class="form-control">
</div>
<br />
</form>
</div>
```

C#

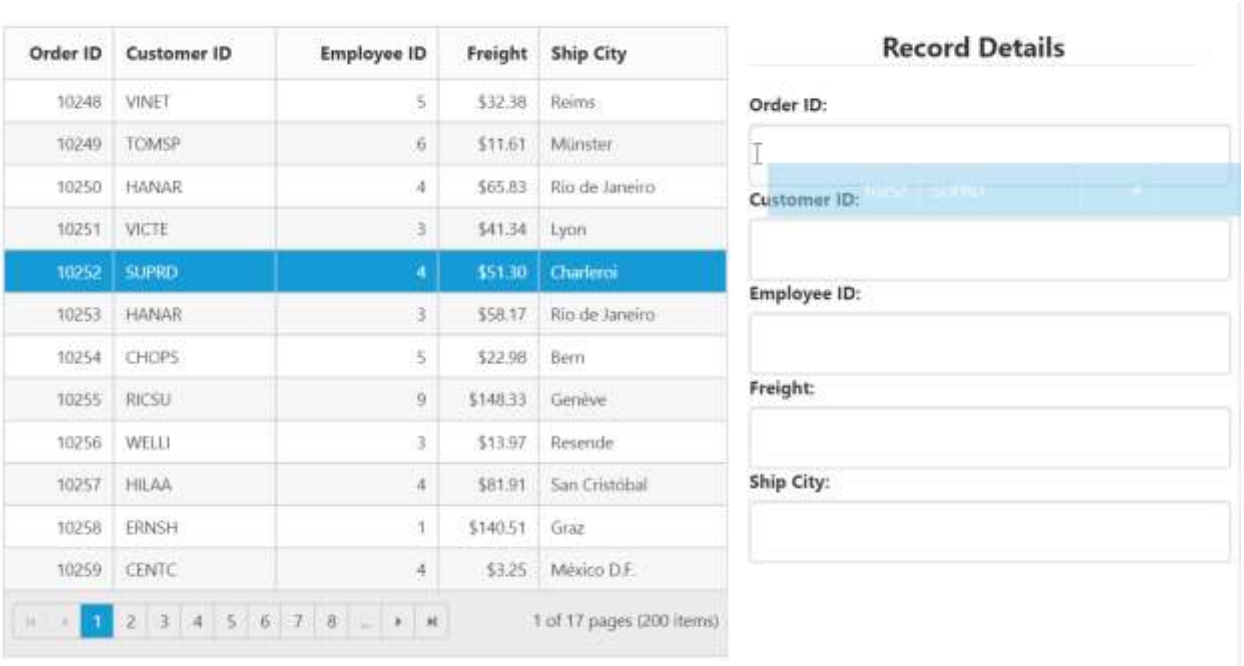
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Default()
{
ViewBag.datasource =OrderRepository.GetAllRecords();
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

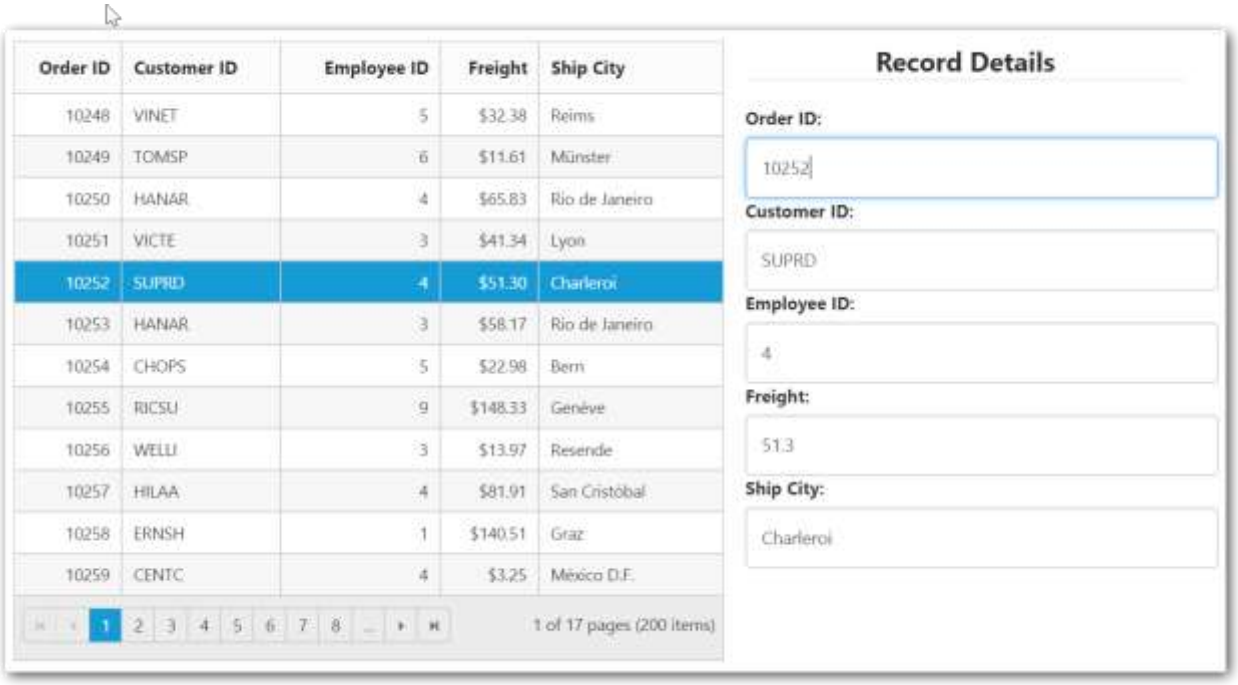
JS

```
<script type="text/javascript">
function rowDropHandler(args) {
for (var key in args.rowData[0]) {
$('#dropForm input[name=' + key + ']').val(args.rowData[0][key]);
}
}
</script>
```

The following output is displayed before dropping the rows on form.



The following output is displayed after dropping the rows on form.



Note: The default behavior of drag and drop between grid or any other controls is as cut and paste. For copy and paste behavior specify the drag behavior in the DragBehavior property of RowDropSettings as "DragBehavior.Copy".

Cell

Auto wrap

Auto wrap enables the Grid to wrap the cell content or header content to next line when the content exceeds the boundary of the cell width. To enable auto wrap, set the AllowTextWrap property as true.

We can specify the mode of auto wrap using WrapMode property of the TextWrapSettings.

Three types of WrapMode are available and they are.

1. Both
2. Header
3. Content

Note: 1. By default the WrapMode will be set as Both.

2. While using the TextWrapSettings then it is must to set the AllowTextWrap as true.

3. For the WrapMode property you can assign enum value (Syncfusion.JavaScript.WrapMode.Both).

Both

When the WrapMode of TextWrapSettings property is set as Both then the auto wrap will be enabled for both grid content and header.

The following code example describes the above behavior.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowTextWrap()
.TextWrapSettings(wrap => { wrap.WrapMode(WrapMode.Both); })
.Columns(col =>
{
col.Field("OrderID").Width(100).Add();
col.Field("EmployeeID").Width(100).Add();
col.Field("Freight").Width(100).Add();
col.Field("ShipCity").Width(150).Add();
col.Field("ShipAddress").HeaderText("Ship Address").Width(100).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	Freight	ShipCity	Ship Address
10248	5	32.38	Reims	59 rue de l'Abbaye
10249	6	11.61	Münster	Luisenstr. 48
10250	4	65.83	Rio de Janeiro	Rua do Paço, 67
10251	3	41.34	Lyon	2, rue du Commerce
10252	4	51.3	Charleroi	Boulevard Tirou, 255
10253	3	58.17	Rio de Janeiro	Rua do Paço, 67
10254	5	22.98	Bern	Hauptstr. 31

1 2 3 4 5 6 7 8 ... 1 of 29 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Header

When the WrapMode of TextWrapSettings property is set as Header then the auto wrap will be enabled only for Grid header alone.

The following code example describes the above behavior.

RAZOR

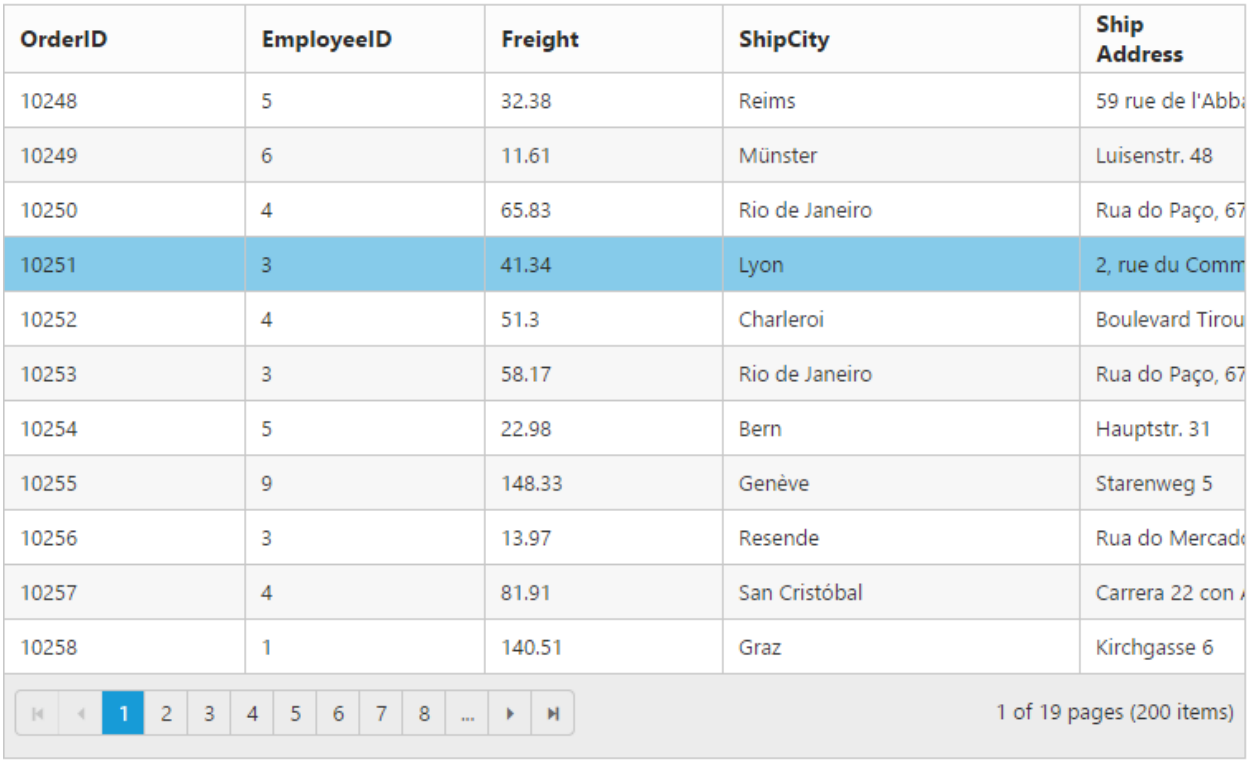
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowTextWrap()
.TextWrapSettings(wrap => { wrap.WrapMode(WrapMode.Header); })
.Columns(col =>
{
col.Field("OrderID").Width(100).Add();
col.Field("EmployeeID").Width(100).Add();
col.Field("Freight").Width(100).Add();
col.Field("ShipCity").Width(150).Add();
col.Field("ShipAddress").HeaderText("Ship Address").Width(100).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	Freight	ShipCity	Ship Address
10248	5	32.38	Reims	59 rue de l'Abb
10249	6	11.61	Münster	Luisenstr. 48
10250	4	65.83	Rio de Janeiro	Rua do Paço, 67
10251	3	41.34	Lyon	2, rue du Comm
10252	4	51.3	Charleroi	Boulevard Tirou
10253	3	58.17	Rio de Janeiro	Rua do Paço, 67
10254	5	22.98	Bern	Hauptstr. 31
10255	9	148.33	Genève	Starenweg 5
10256	3	13.97	Resende	Rua do Mercad
10257	4	81.91	San Cristóbal	Carrera 22 con
10258	1	140.51	Graz	Kirchgasse 6

1 2 3 4 5 6 7 8 ... 1 of 19 pages (200 items)

Content

When the WrapMode of TextWrapSettings property set as Content then the auto wrap will be enabled only for the Grid content alone.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowTextWrap()
.TextWrapSettings(wrap => { wrap.WrapMode(WrapMode.Header); })
.Columns(col =>
{
col.Field("OrderID").Width(100).Add();
col.Field("EmployeeID").Width(100).Add();
col.Field("Freight").Width(100).Add();
col.Field("ShipCity").Width(150).Add();
col.Field("ShipAddress").HeaderText("Ship Address").Width(100).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	Freight	ShipCity	Ship Address
10248	5	32.38	Reims	59 rue de l'Abbaye
10249	6	11.61	Münster	Luisenstr. 48
10250	4	65.83	Rio de Janeiro	Rua do Paço, 67
10251	3	41.34	Lyon	2, rue du Commerce
10252	4	51.3	Charleroi	Boulevard Tirou, 255
10253	3	58.17	Rio de Janeiro	Rua do Paço, 67
10254	5	22.98	Bern	Hauptstr. 31

1 2 3 4 5 6 7 8 ... 1 of 29 pages (200 items)

Cell Merging

The grid has options to merge the grid cells based on the required conditions. This can be enabled by setting the `AllowCellMerging` property as `true` and the merge conditions can be defined in `MergeCellInfo` event. In this event, you can get the column details and data of that particular row and column which is helpful in defining conditions.

You can merge the rows and cells of grid, using the `rowMerge`, `colMerge` and `merge` functions available in `MergeCellInfo` event's argument.

Note: The following features are not supported with Cell Merging

1. Normal Mode Editing
2. Inline Mode Editing
3. Inline TemplateForm Mode Editing
4. Grouping
5. Virtual Scrolling
6. Frozen Columns
7. Cell Selection Modes

Copyright © 2001 - 2020 Syncfusion Inc.

8. Column Selection

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowCellMerging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
.ClientSideEvents(eve => { eve.MergeCellInfo("mergeCellInfo"); })
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

C#

```
<script type="text/javascript">
function mergeCellInfo(args)
{
if (args.column.field == "EmployeeID" && args.rowData.OrderID == 10248)
args.rowMerge(3);
else if (args.column.field == "ShipCity" && args.rowData.OrderID == 10252)
args.colMerge(3);
else if (args.column.field == "ShipCity" && args.rowData.OrderID == 10255)
args.merge(0, 3);
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249		Münster	Germany	11.61
10250		Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi		
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3		Brazil	13.97
10257	4		Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Custom Attribute

You can add custom attribute for the particular column's td element by using CustomAttributes property of the column.

Based on custom attribute you can customize the style and appearance of the td element or handling jQuery functionalities.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this link.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").CustomAttributes(custom => {
custom.AddAttribute("title","'{{"{{"}}:ShipCity {{}}}}'"); }).Add();
col.Field("ShipCountry").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource1 = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource1;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	CustomerID	EmployeeID	ShipCity	ShipCountry
10248	VINET	5	Reims	France
10249	TOMSP	6	Münster	Germany
10250	HANAR	4	Rio de Janeiro	Brazil
10251	VICTE	3	Lyon	France
10252	SUPRD	4	Charle	Belgium
10253	HANAR	3	Rio de Janeiro	Brazil
10254	CHOPS	5	Bern	Switzerland
10255	RICSU	9	Genève	Switzerland
10256	WELLI	3	Resende	Brazil
10257	HILAA	4	San Cristóbal	Venezuela
10258	ERNSH	1	Graz	Austria
10259	CENTC	4	México D.F.	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Displaying HTML content

This will helps you to show actual HTML value in grid content and header. To disable HTML code, set the DisableHtmlEncode property of Columns as true.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").HeaderText("<div>Customer
ID</div>").disableHtmlEncode(true).Add();
col.Field("EmployeeID").HeaderText("<div>Employee
ID</div>").disableHtmlEncode(false).Add();
col.Field("ShipCountry").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource1 = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource1;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	Customer ID	Employee ID	ShipCountry
10248	VINET	5	France
10249	TOMSP	6	Germany
10250	HANAR	4	Brazil
10251	VICTE	3	France
10252	SUPRD	4	Belgium
10253	HANAR	3	Brazil
10254	CHOPS	5	Switzerland
10255	RICSU	9	Switzerland
10256	WELLI	3	Brazil
10257	HILAA	4	Venezuela
10258	ERNSH	1	Austria
10259	CENTC	4	Mexico

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Tooltip

When you move the cursor over the particular cell it provides an information about the corresponding cell value.

Template

Copyright © 2001 - 2020 Syncfusion Inc.

HTML templates can be specified in the Tooltip property of the particular column cell as a string (HTML element) or ID of the template's HTML element.You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this link.

Note: It's a standard way to enclose the template within the script tag with type as "text/x-jsrender".

The Tooltip template must contain value property to bind the corresponding cell text in tooltip

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Tooltip("#colTip").Add();
col.Field("Freight").Add();
})
)
```

C#

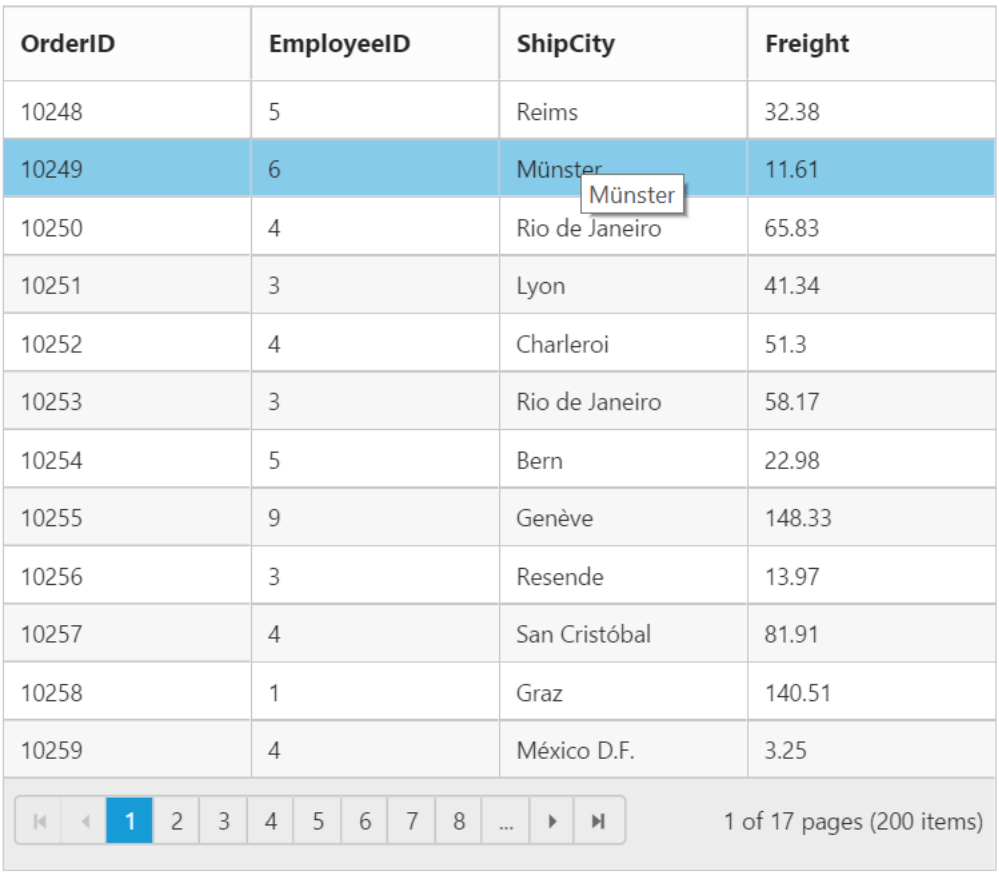
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult DetailTemplate()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/template" id="colTip">
{{"{{"}}:value{{}}}}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



ClipMode

When the cell value contains a long text that will not fit into the grid column cell, the ClipMode property is used. By using the ClipMode, the cell value will be displayed with ellipsis or with clipped content when the text overflows inside a column cell.

Note: By default the ClipMode will be set as Clip.

List of types

1. Clip
2. Ellipsis
3. EllipsisWithTooltip

Clip

When the content overflows, the remaining content will be hidden in the particular cell

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
```

```
col.Field("ShipCity").Add();
col.Field("ShipName").ClipMode(ClipMode.Clip).Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult DetailTemplate()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	ShipCity	ShipName	Freight
10248	Reims	Vins et alcools Cl	32.38
10249	Münster	Toms Spezialitäte	11.61
10250	Rio de Janeiro	Hanari Carnes	65.83
10251	Lyon	Victuailles en sto	41.34
10252	Charleroi	Suprêmes délices	51.3
10253	Rio de Janeiro	Hanari Carnes	58.17
10254	Bern	Chop-suey Chine	22.98
10255	Genève	Richter Superma	148.33
10256	Resende	Wellington Impo	13.97
10257	San Cristóbal	HILARION-Abast	81.91
10258	Graz	Ernst Handel	140.51
10259	México D.F.	Centro comercial	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Ellipsis

Ellipsis will be displayed when the content overflows its column width. Here the Tooltip will not be shown for corresponding columns.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

RAZOR

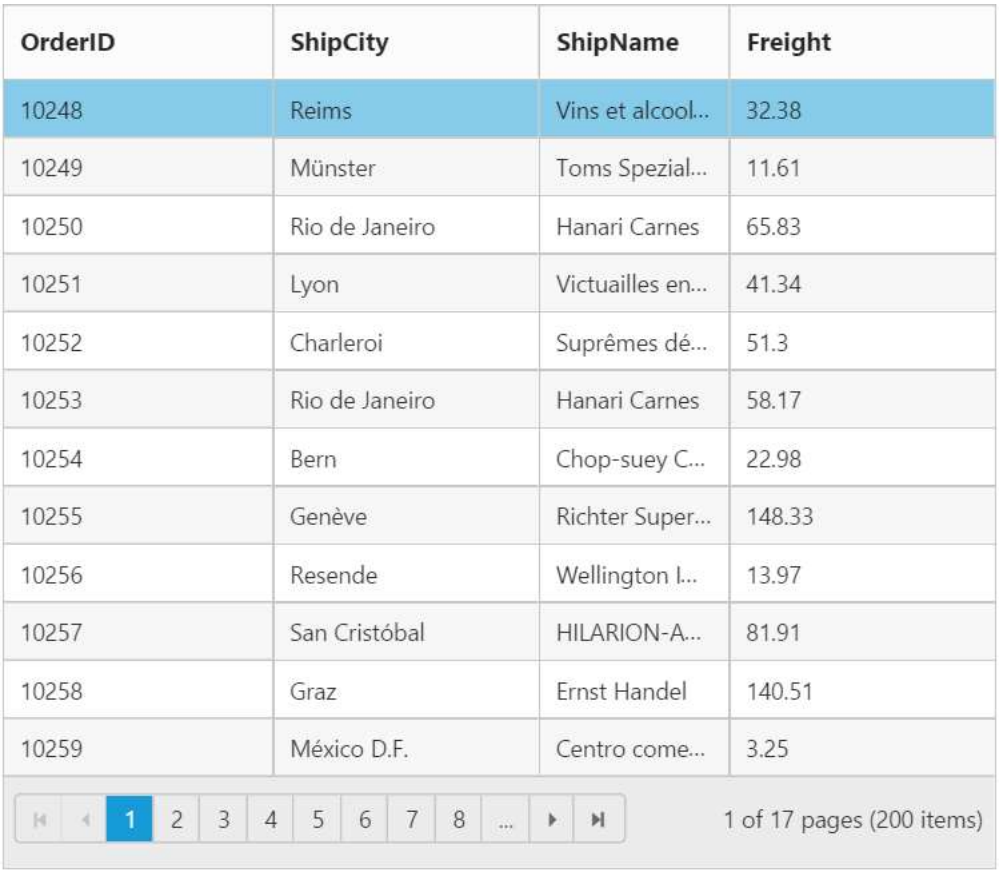
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("ShipCity").Add();
col.Field("ShipName").ClipMode(ClipMode.Ellipsis).Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult DetailTemplate()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Ellipsis With Tooltip

Ellipsis will be displayed when the content overflows its column width. Here tooltip will be shown only for the corresponding column cells that shows ellipsis.

Note: If `ClipMode` is set as `EllipsisWithTooltip`, then `Tooltip` must be given.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("ShipCity").Add();
col.Field("ShipName").Tooltip("#colTip").ClipMode(ClipMode.EllipsisWithTooltip).Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class GridController : Controller
{
public ActionResult DetailTemplate()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/template" id="colTip">
{{"{{"}}:value{{}}}}
</script>
```

The following output is displayed as a result of the above code example.

OrderID	ShipCity	ShipName	Freight
10248	Reims	Vins et alcool...	32.38
10249	Münster	Toms Spezial	11.61
10250	Rio de Janeiro	Hanari Carnes	65.83
10251	Lyon	Victuailles en...	41.34
10252	Charleroi	Suprêmes dé...	51.3
10253	Rio de Janeiro	Hanari Carnes	58.17
10254	Bern	Chop-suey C...	22.98
10255	Genève	Richter Super...	148.33
10256	Resende	Wellington I...	13.97
10257	San Cristóbal	HILARION-A...	81.91
10258	Graz	Ernst Handel	140.51
10259	México D.F.	Centro come...	3.25

Vins et alcools Chevalier

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Sorting

The Grid control has support to sort data bound columns in ascending or descending order. This can be achieved by setting the AllowSorting property as true.

To dynamically sort a particular column, click on its column header. The order switch between ascending and descending each time you click a column header for sorting.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the previous behavior.

RAZOR

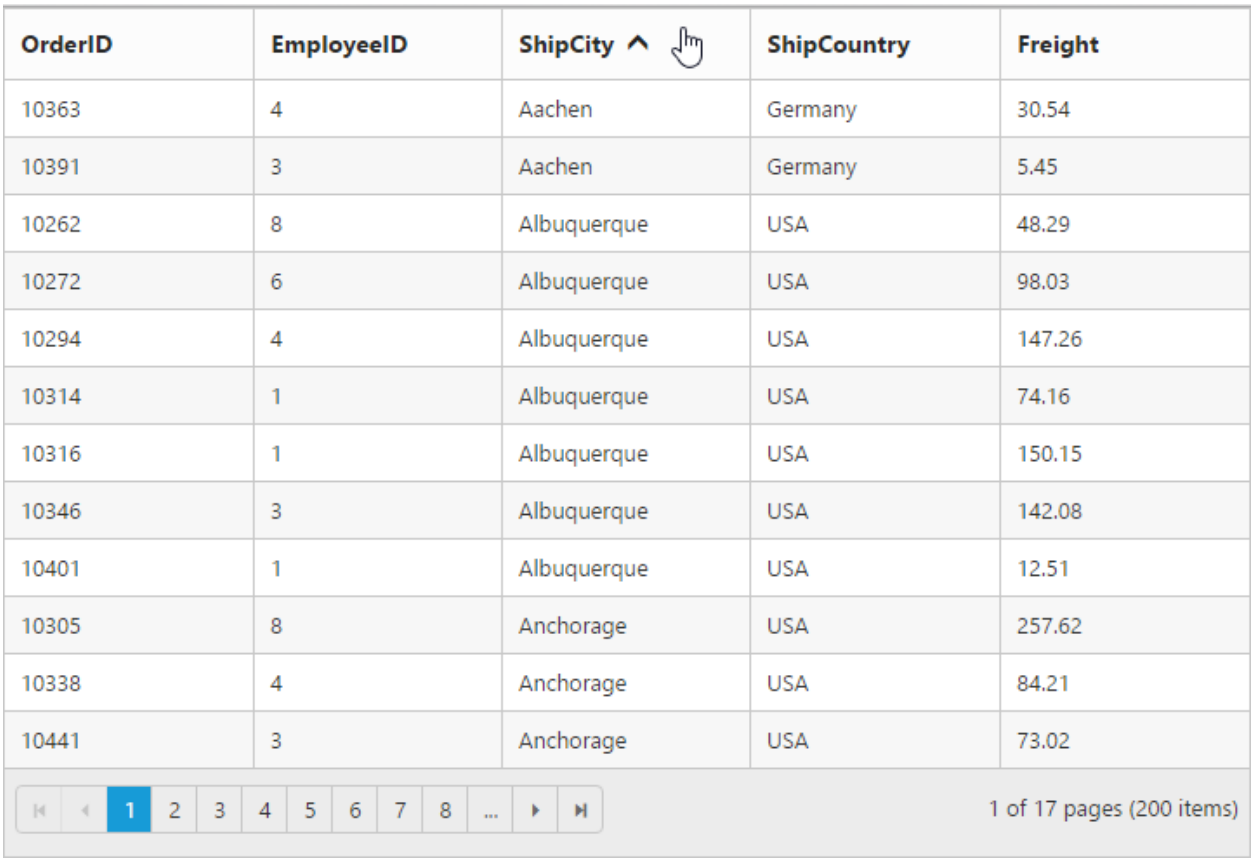
```
@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSorting()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = OrderRepository.GetAllRecords();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code sample.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10363	4	Aachen	Germany	30.54
10391	3	Aachen	Germany	5.45
10262	8	Albuquerque	USA	48.29
10272	6	Albuquerque	USA	98.03
10294	4	Albuquerque	USA	147.26
10314	1	Albuquerque	USA	74.16
10316	1	Albuquerque	USA	150.15
10346	3	Albuquerque	USA	142.08
10401	1	Albuquerque	USA	12.51
10305	8	Anchorage	USA	257.62
10338	4	Anchorage	USA	84.21
10441	3	Anchorage	USA	73.02

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Initial sorting

Through the SortedColumns property of SortSettings, you can sort the columns while initializing the grid itself. You need to specify the Field (Columns) name and Direction in the SortedColumns.

Note: 1. For the Direction property you can assign either string value ("Descending") or enum value (SortOrder.Descending).

2. You can add multiple columns in the SortedColumns for multi column sorting while initializing the grid itself.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSorting()
.SortSettings(sort => sort.SortedColumns(col =>
col.Field("EmployeeID").Direction(SortOrder.Descending).Add()))
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = OrderRepository.GetAllRecords();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	CustomerID	ShipCountry	Freight
10255	9	RICSU	Switzerland	148.33
10263	9	ERNSH	Austria	146.06
10324	9	SAVEA	USA	214.27
10331	9	BONAP	France	10.19
10386	9	FAMIA	Brazil	13.99
10411	9	BOTTM	Canada	23.65
10262	8	RATTC	USA	48.29
10268	8	GROSR	Venezuela	66.29
10276	8	TORTU	Mexico	13.84
10278	8	BERGS	Sweden	92.69
10279	8	LEHMS	Germany	25.83
10286	8	QUICK	Germany	229.24

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Multi-column sorting

Sort multiple columns in the grid by setting the AllowMultiSorting property as true. The sorting order is displayed in the header while doing multi sorting.

You can sort more than one column by pressing the "Ctrl key + mouse left click" on the column header. To clear sorting for particular column, press the "Shift + mouse left click".

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The AllowSorting must be true while enabling multi sort.

The following code example describes the previous behavior.

RAZOR

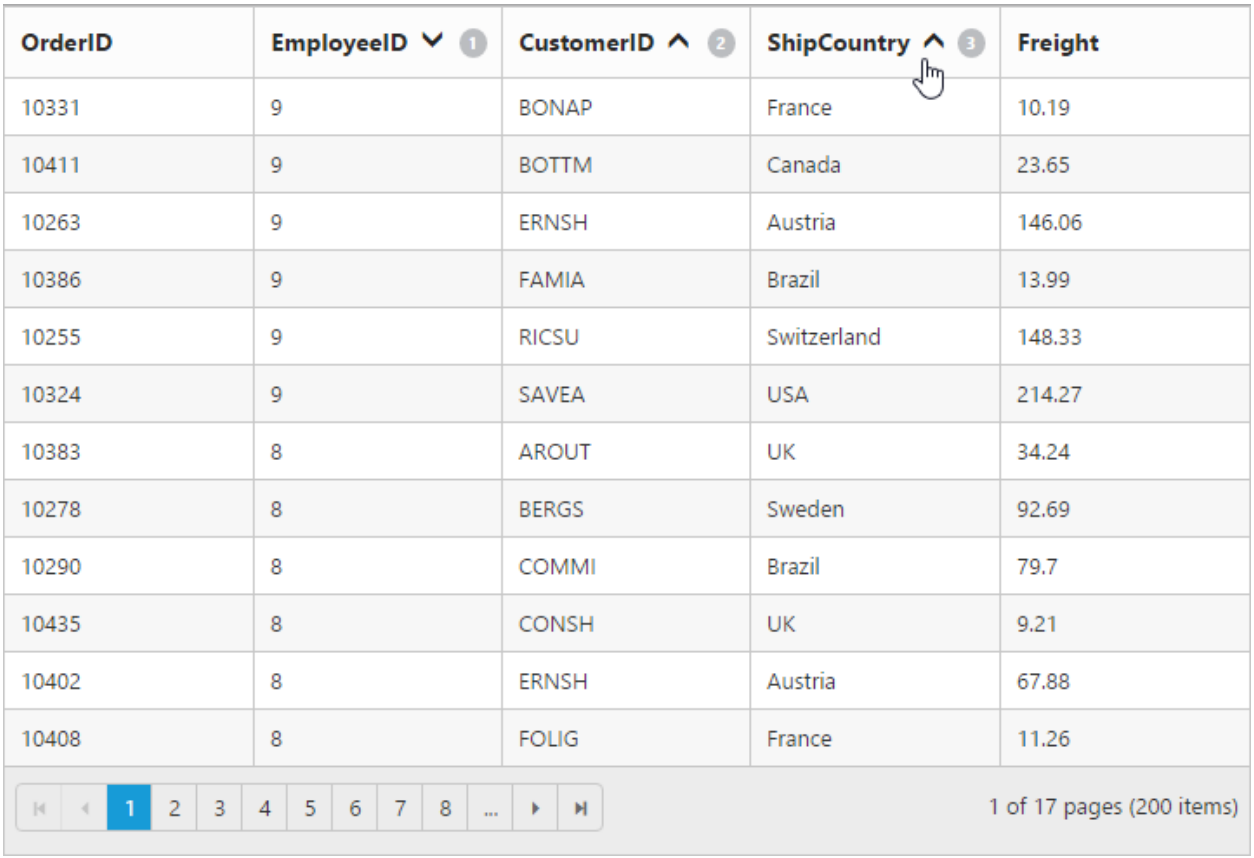
```
@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSorting()
.AllowMultiSorting()
.SortSettings(sort =>
{
sort.SortedColumns(col =>
col.Field("EmployeeID").Direction(SortOrder.Descending).Add());
sort.SortedColumns(col => col.Field("CustomerID").Add());
})
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = OrderRepository.GetAllRecords();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	CustomerID	ShipCountry	Freight
10331	9	BONAP	France	10.19
10411	9	BOTTM	Canada	23.65
10263	9	ERNSH	Austria	146.06
10386	9	FAMIA	Brazil	13.99
10255	9	RICSU	Switzerland	148.33
10324	9	SAVEA	USA	214.27
10383	8	AROUT	UK	34.24
10278	8	BERGS	Sweden	92.69
10290	8	COMMI	Brazil	79.7
10435	8	CONSH	UK	9.21
10402	8	ERNSH	Austria	67.88
10408	8	FOLIG	France	11.26

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Stable sorting

For sorting, grid uses default browser's sort function for better performance. On multi column sorting in some browsers like chrome, the records order will be different due to unstable implementation of sorting algorithm in it.

To resolve this, you need to set `ej.support.stableSort` as `false`.

This will tell the "DataManager" to use custom sort function for sorting data.

Please refer to the link, to know more information about stable sort.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSorting()
.AllowMultiSorting()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = OrderRepository.GetAllRecords();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/javascript">
ej.support.stableSort = true
</script>
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10409	3	Buenos Aires	Argentina	29.83
10258	1	Graz	Austria	140.51
10263	9	Graz	Austria	146.06
10351	1	Graz	Austria	162.33
10353	7	Salzburg	Austria	360.63
10368	2	Graz	Austria	101.95
10382	4	Graz	Austria	94.77
10390	6	Graz	Austria	126.38
10392	2	Salzburg	Austria	122.46
10402	8	Graz	Austria	67.88
10403	4	Graz	Austria	73.79
10427	4	Salzburg	Austria	31.29

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Touch options

While using the grid in a touch device, you have an option for multi sorting in single tap on the grid header. By tapping on the grid header, it will show the toggle button in small popup with sort icon. Now, tap the button to enable multi sorting in single tap.

Again if you tap the popup symbol, then the single tap multi sorting will be disabled.

Note: The AllowMultiSorting and AllowSorting should be true then only the popup will be shown.

The following code example describes the previous behavior.

RAZOR

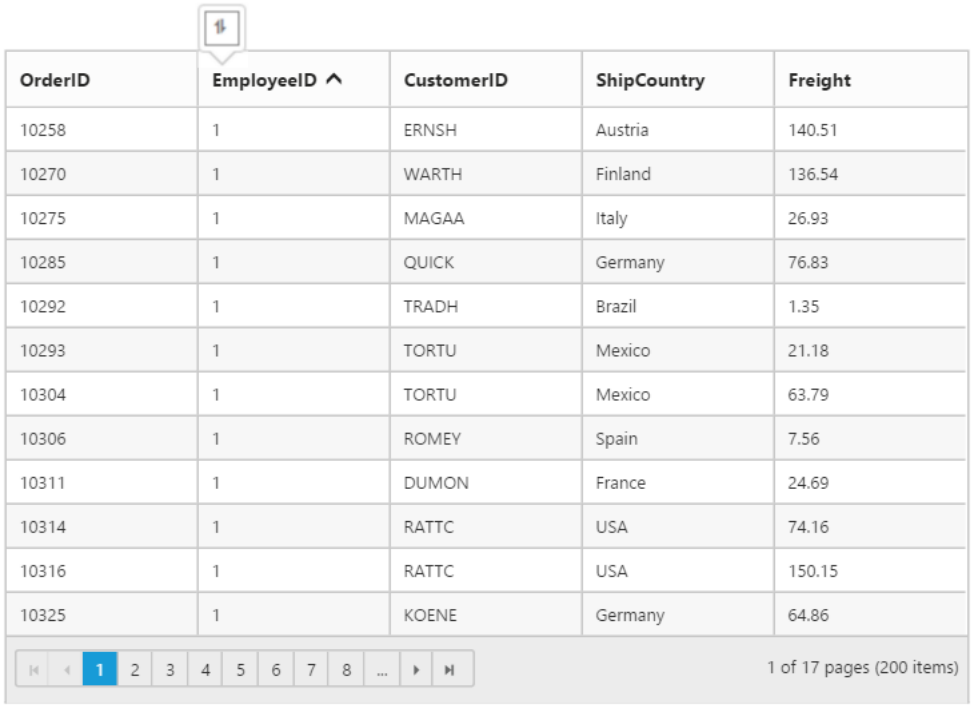
```
@(Html.EJ().Grid<object>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowMultiSorting()
.AllowSorting()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = OrderRepository.GetAllRecords();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	CustomerID	ShipCountry	Freight
10258	1	ERNSH	Austria	140.51
10270	1	WARTH	Finland	136.54
10275	1	MAGAA	Italy	26.93
10285	1	QUICK	Germany	76.83
10292	1	TRADH	Brazil	1.35
10293	1	TORTU	Mexico	21.18
10304	1	TORTU	Mexico	63.79
10306	1	ROMEY	Spain	7.56
10311	1	DUMON	France	24.69
10314	1	RATTC	USA	74.16
10316	1	RATTC	USA	150.15
10325	1	KOENE	Germany	64.86

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Note: To get the sorted data of the grid after sorting a column you can refer the How To.

Filtering

Filtering helps to view the particular or related records from dataSource which meets a given filtering criteria. To enable filter, set the AllowFiltering property astrue.

The Grid supports three types of filter, they are the following.

1. Filter Bar
2. Menu
3. Excel

Four types of filter menu is available in all filter types, they are the following.

1. String
2. Numeric
3. Date
4. Boolean

The corresponding filter menu is opened based on the column type.

Note: 1. Need to specify the Type of column, when first record data value is empty or null otherwise the filter menu is not opened.

2. The default filter type is Filter bar, when the AllowFiltering is enabled and FilterType is not set.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowFiltering()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	CustomerID	ShipCountry	Freight
10248	5	VINET	France	32.38
10249	6	TOMSP	Germany	11.61
10250	4	HANAR	Brazil	65.83
10251	3	VICTE	France	41.34
10252	4	SUPRD	Belgium	51.3
10253	3	HANAR	Brazil	58.17
10254	5	CHOPS	Switzerland	22.98
10255	9	RICSU	Switzerland	148.33
10256	3	WELLI	Brazil	13.97
10257	4	HILAA	Venezuela	81.91
10258	1	ERNSH	Austria	140.51
10259	4	CENTC	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Menu filter

You can enable menu filter by setting FilterType as Menu in FilterSettings

There is an option to show or hide the additional filter options in the menu by setting the ShowPredicate as true or false in FilterSettings respectively.

Note: For the FilterType property you can assign either string value ("Menu") or enum value (Syncfusion.JavaScript.FilterType.Menu).

A specified range of values can be filtered by using the between operator for the column type number, date and datetime.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowFiltering()
.FilterSettings(filter => { filter.FilterType(FilterType.Menu); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
```

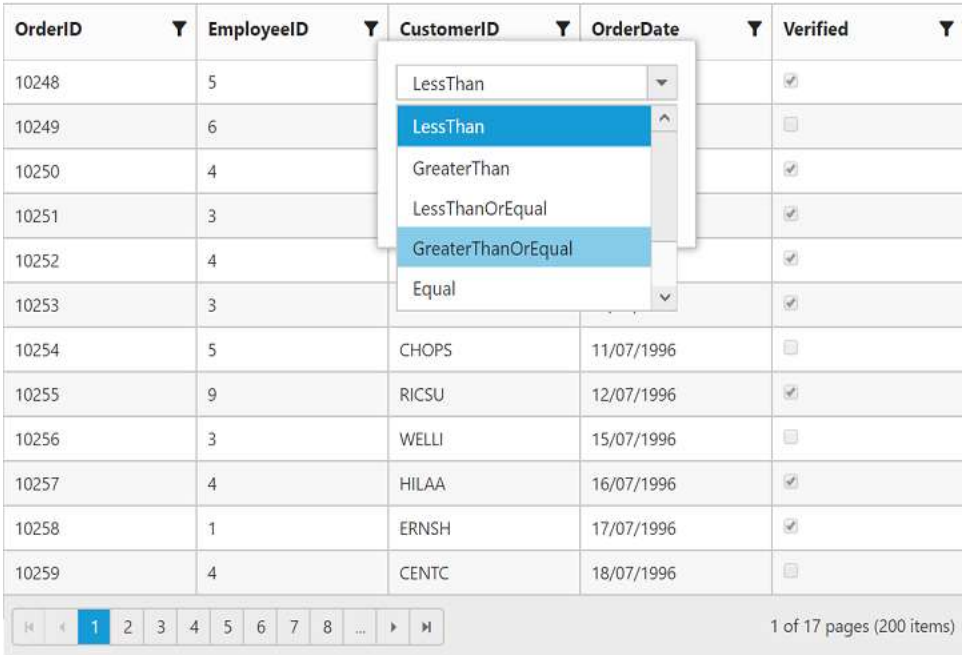
Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

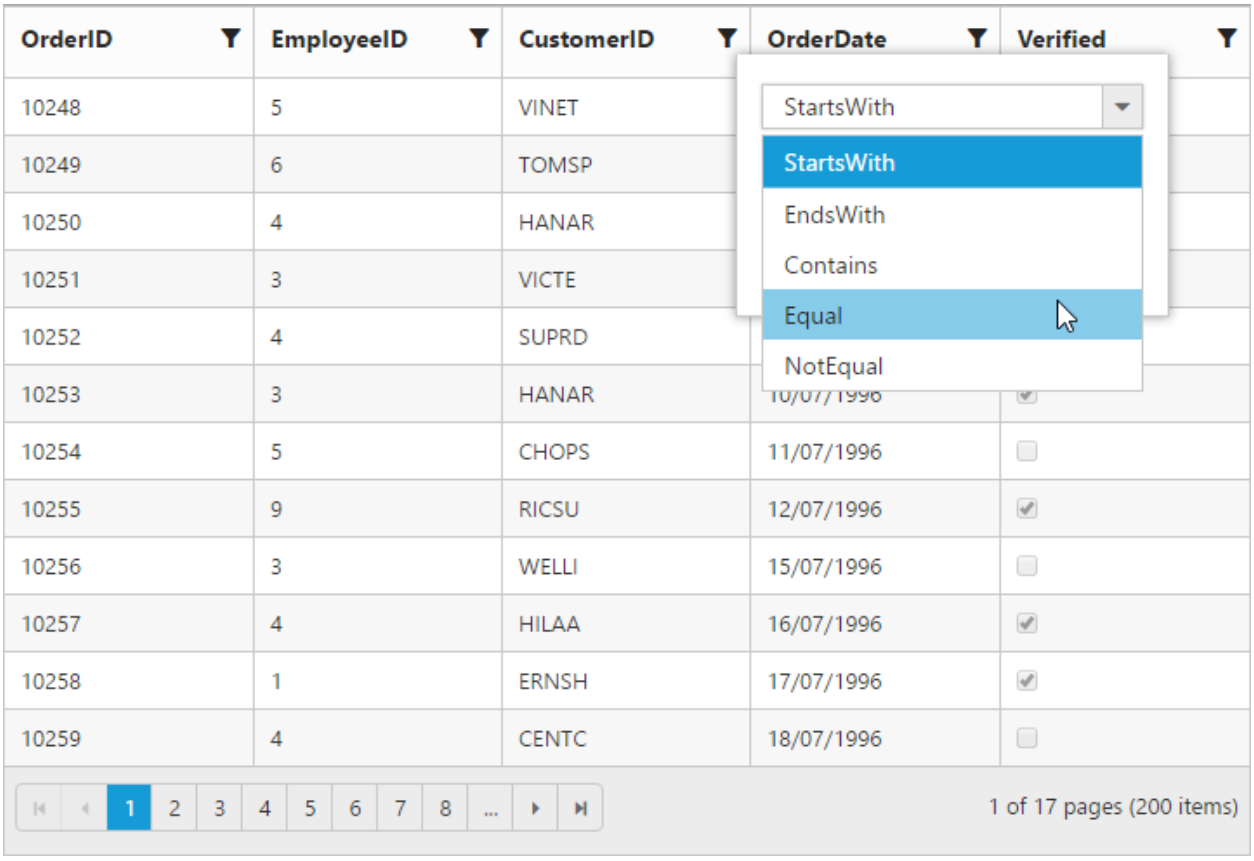
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

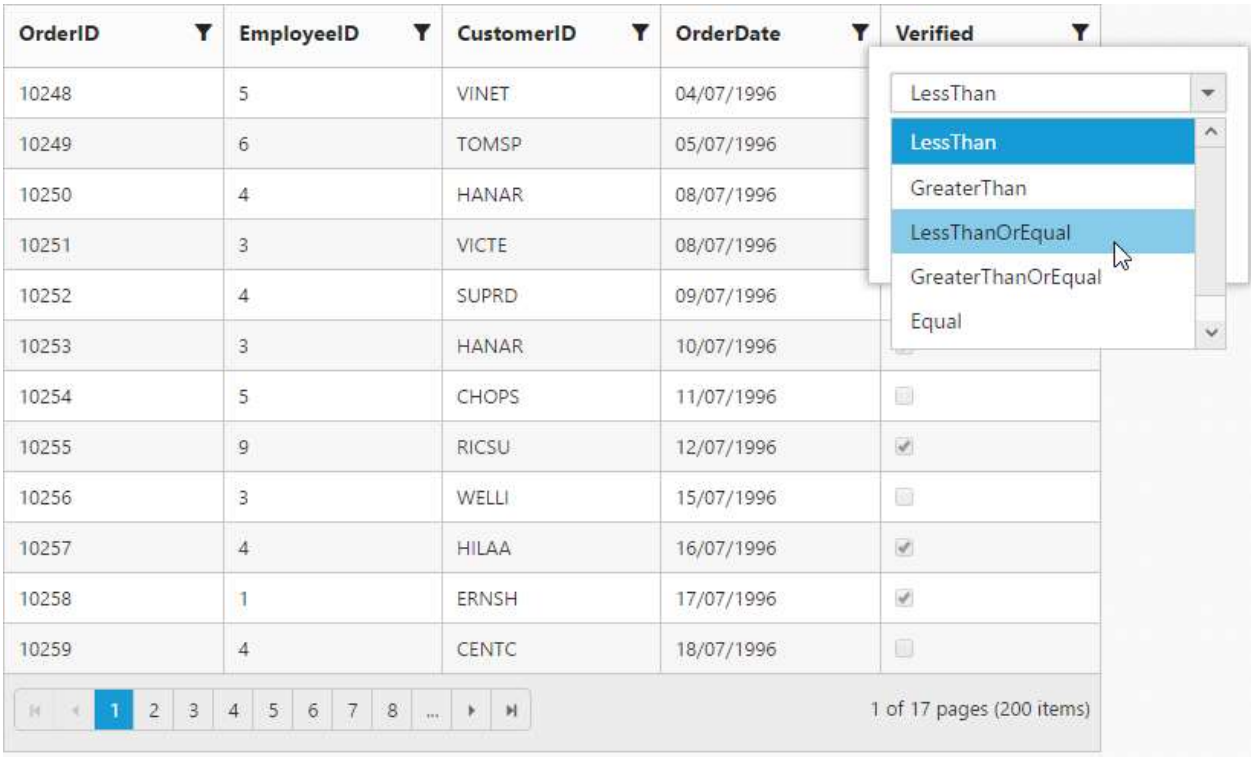


Numeric Filter

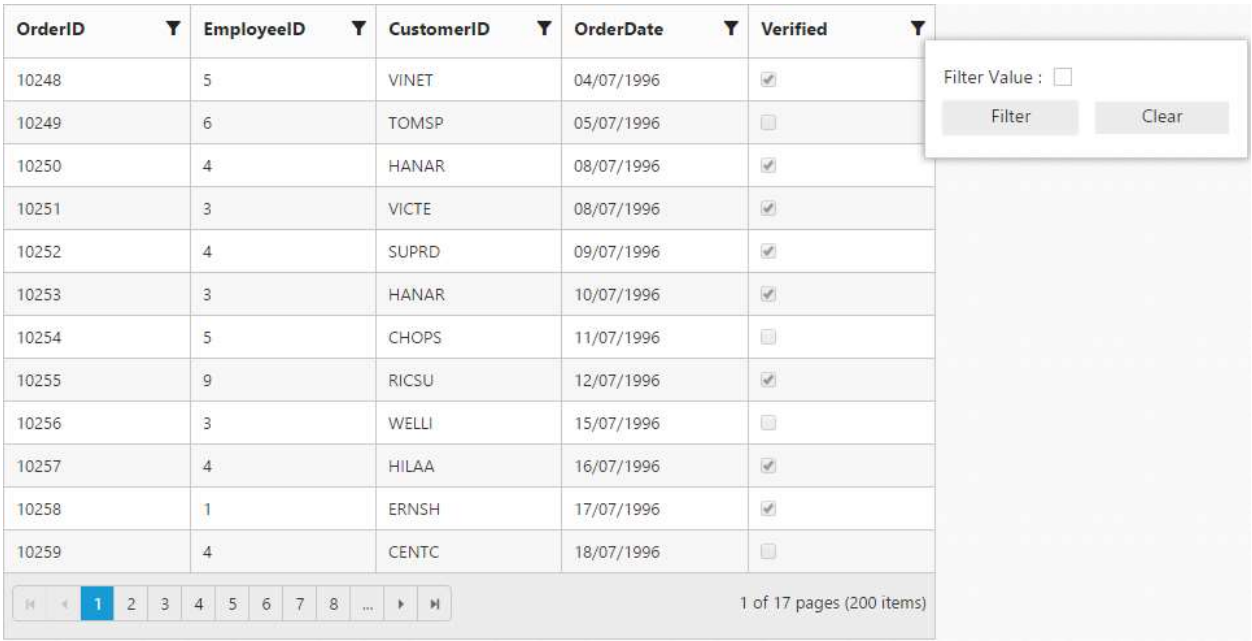
Copyright © 2001 - 2020 Syncfusion Inc.



String Filter



Date Filter



Boolean Filter

Excel-like filter

You can enable excel menu by setting `FilterType` as `Excel` in the `FilterSettings` . The excel menu contains an option such as sorting, clear filter, submenu for advanced filtering.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowFiltering()
.FilterSettings(filter => { filter.FilterType(FilterType.Excel); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

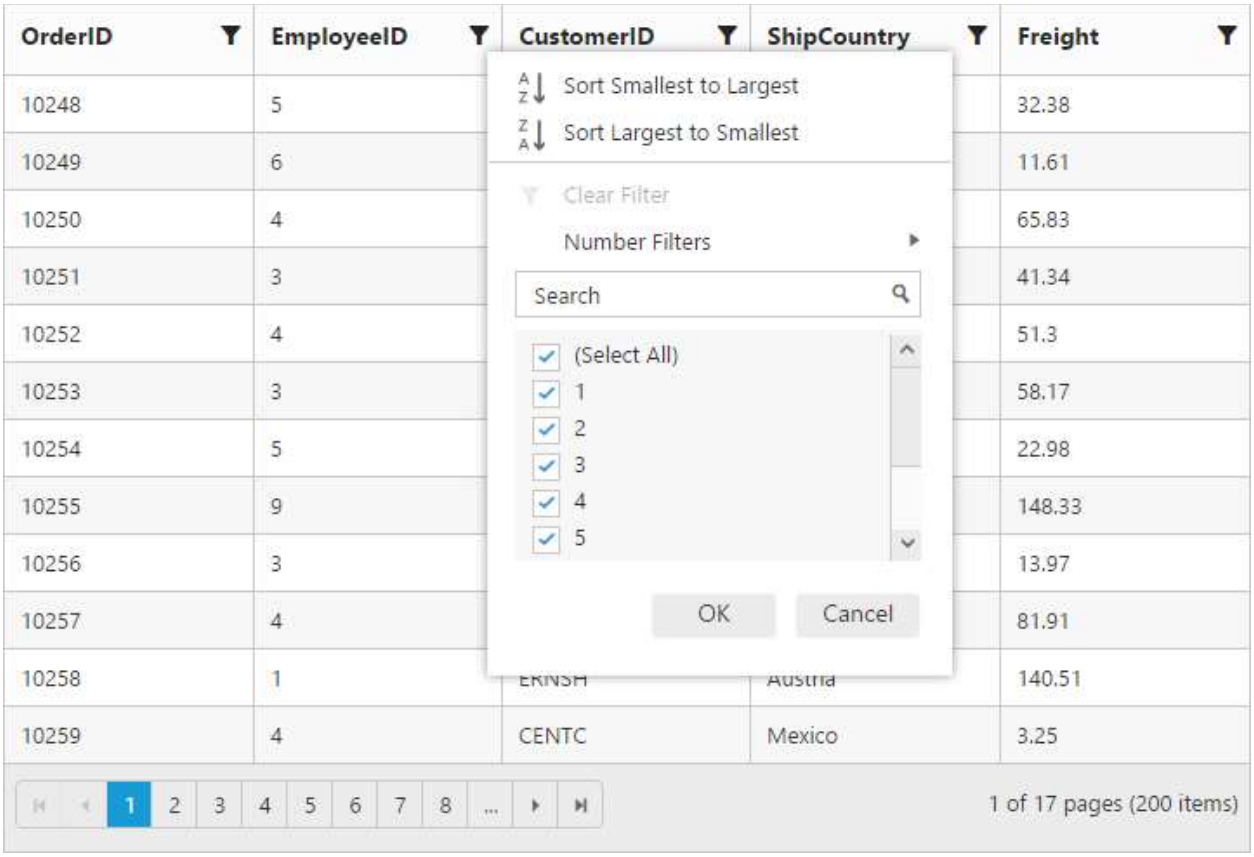
C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

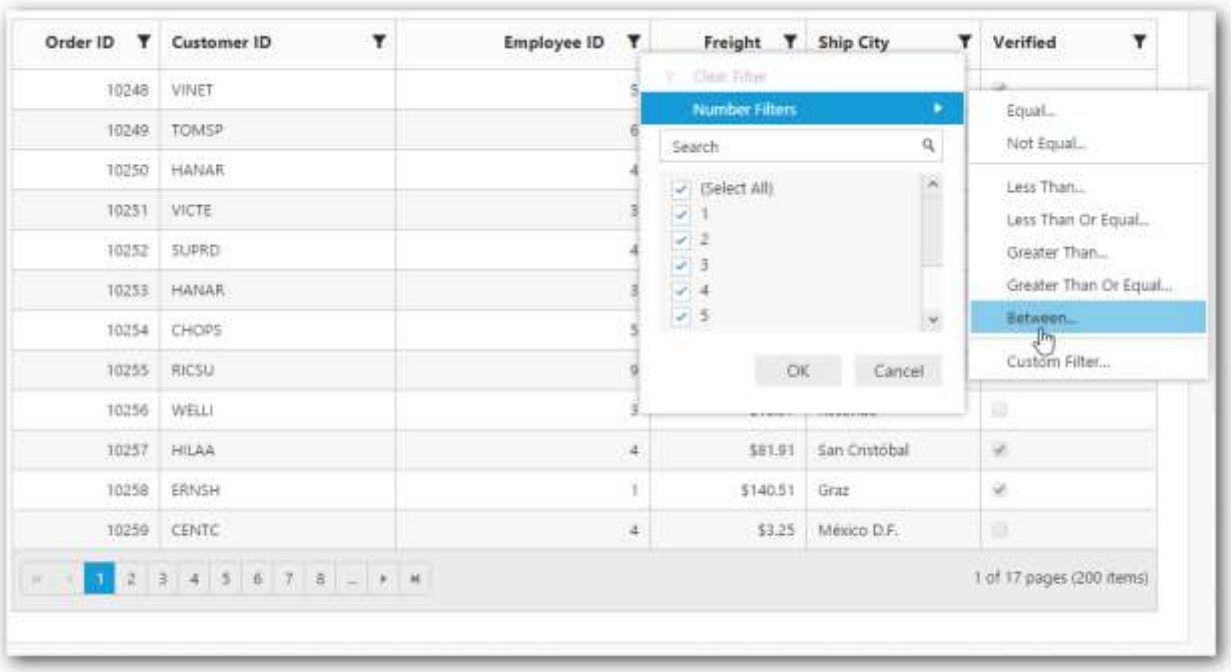
```
}
}
}
```

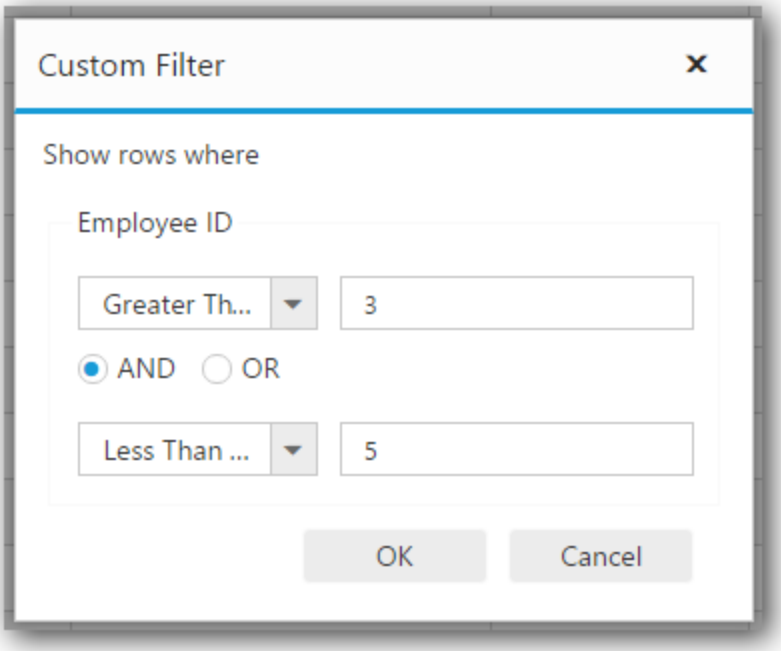
The following output is displayed as a result of the previous code example.



Filtering between values

By using the custom filter feature of the Excel filter, we can filter between values of the column. The following screenshot depicts the usage of "Between" option of the custom filter dialog.





Checkbox list generation:

By default, the checkbox list is generated from the distinct values of the filter column from dataSource which gives an option to search and select the required items.

Also, on checkbox list generation, if the number of distinct values are greater than 1000, then the excel filter will display only first 1000 values and show "Not all items shown" label to ensure the best performance on rendering and searching. However this limit has been customized according to your requirement by setting the MaxFilterChoices with required limit in integer.

Note: 1. Using excel filter events you can change the dataSource of the checkbox list.

Copyright © 2001 - 2020 Syncfusion Inc.

2. Query of checkbox list can also be changed using excel filter events.

The following code example describes the previous behavior.

RAZOR

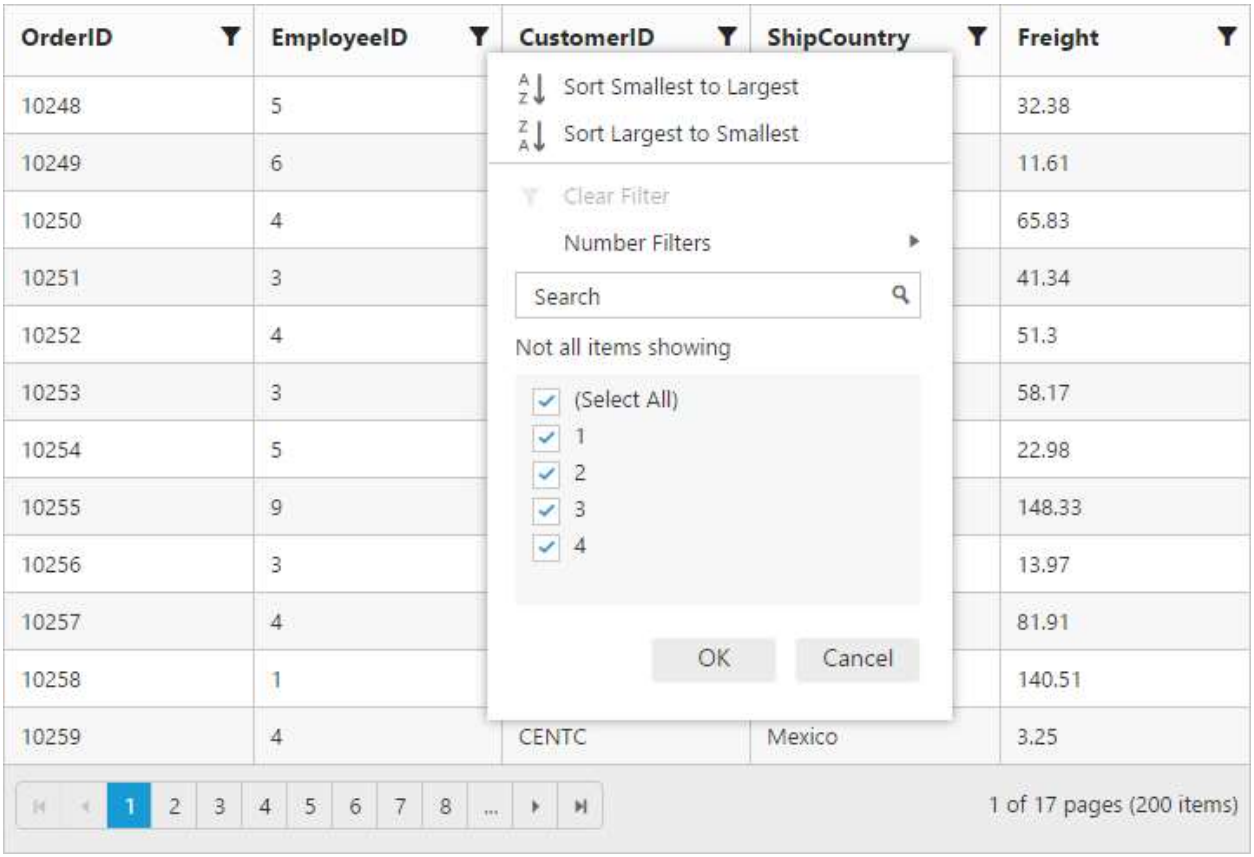
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowFiltering()
.FilterSettings(filter => {
filter.FilterType(FilterType.Excel).MaxFilterChoices(4); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

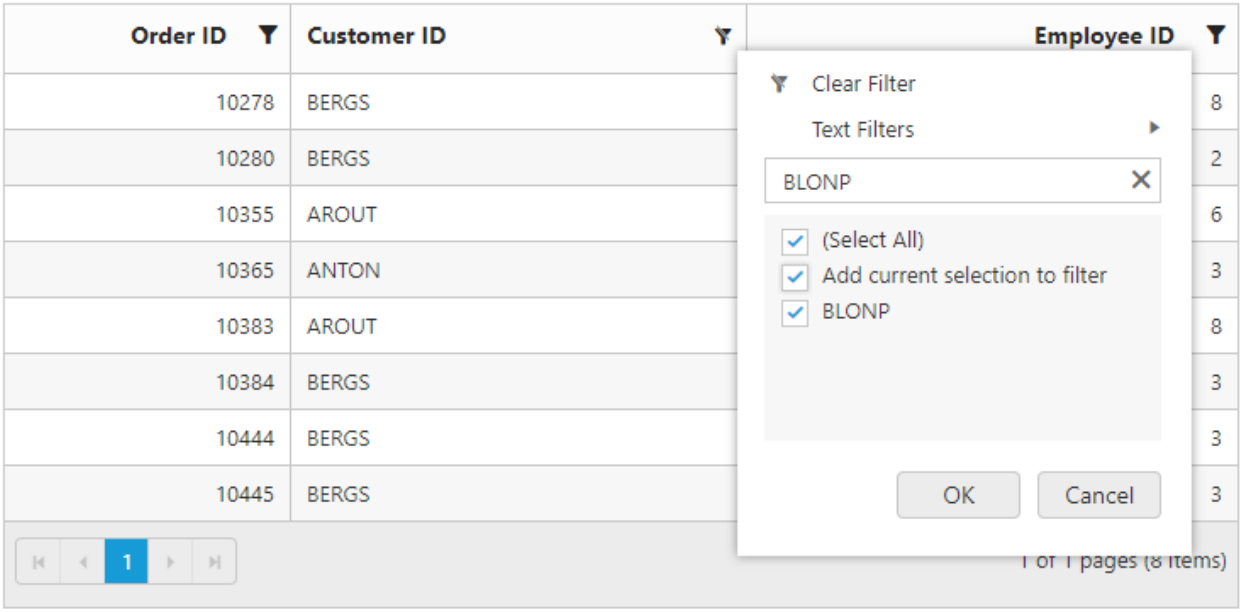
Copyright © 2001 - 2020 Syncfusion Inc.



Add current selection to filter checkbox:

When filtering is done multiple times on the same column then the previously filtered values on the column will be cleared. So, to retain the old values Add current selection to filter checkbox can be used which is displayed when data is searched in the search bar.

The following image describes the previous mentioned behavior.



Copyright © 2001 - 2020 Syncfusion Inc.

Case sensitivity

To perform filter operation with case sensitive in excel styled filter menu mode by setting EnableCaseSensitivity as true.

The following code example describes the previous behavior.

RAZOR

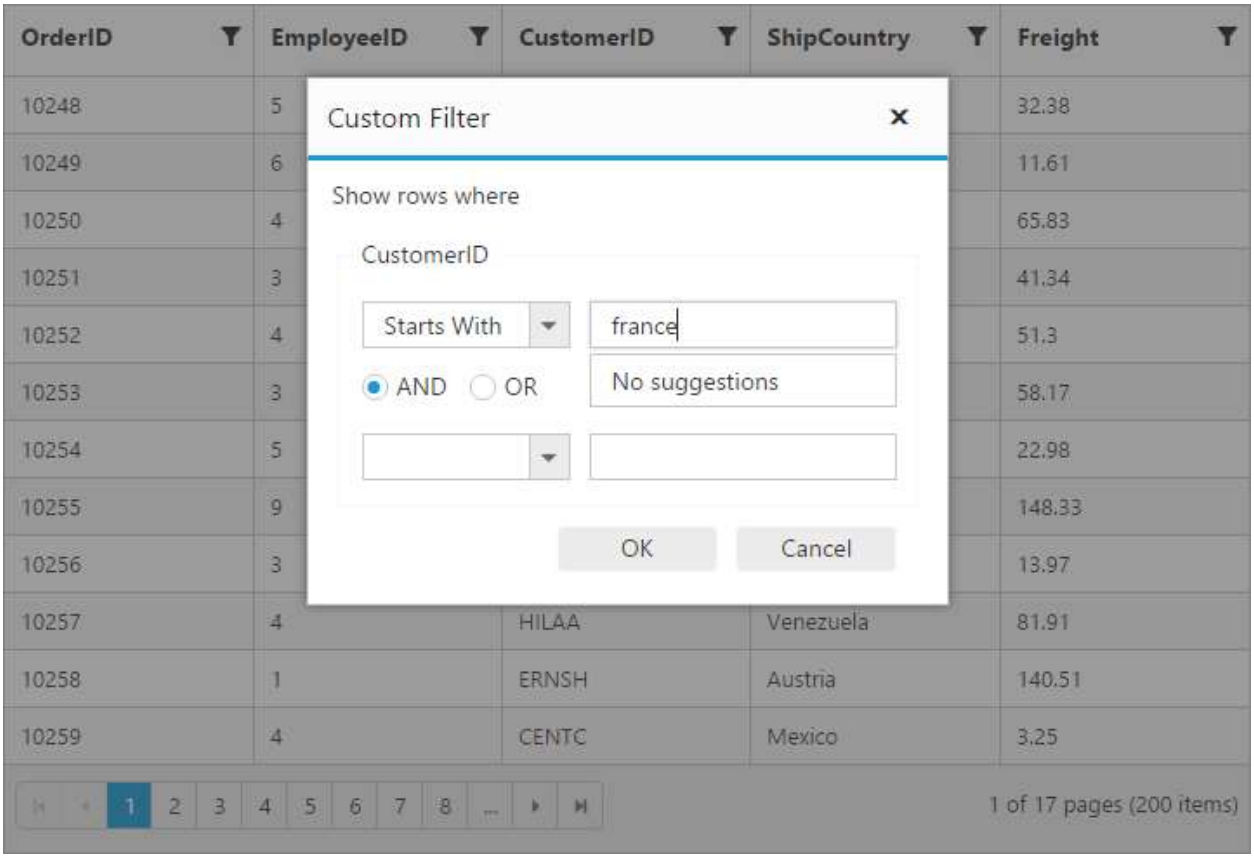
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowFiltering()
.FilterSettings(filter => {
filter.FilterType(FilterType.Excel).EnableCaseSensitivity(true);  })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Filter bar

The 'Filter bar' row is located next to column header of grid. You can filter the records with different expressions depending upon the column type. To show the filter bar row, set the `FilterType` as `FilterBar`.

List of filter bar expressions:

You can enter the below filter expressions manually in the filter bar.

Expression	Example	Description	Column Type
=	= value	Equal	Numeric
!=	!= value	NotEqual	
>	> value	GreaterThan	
<	< value	LessThan	
>=	>= value	GreaterThanOrEqual	
<=	<= value	LessThanOrEqual	
N/A	N/A	Always `StartsWith` operator will be used for the string filter	String
N/A	N/A	Always `Equal` operator will be used for the Date filter	Date
N/A	N/A	Always `Equal` operator will be used for the Boolean filter	Boolean

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowFiltering()
.FilterSettings(filter => { filter.FilterType(FilterType.FilterBar); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	CustomerID	ShipCountry	Freight
	>5			
10249	6	TOMSP	Germany	11.61
10255	9	RICSU	Switzerland	148.33
10262	8	RATTC	USA	48.29
10263	9	ERNSH	Austria	146.06
10264	6	FOLKO	Sweden	3.67
10268	8	GROSR	Venezuela	66.29
10271	6	SPLIR	USA	4.54
10272	6	RATTC	USA	98.03
10274	6	VINET	France	6.01
10276	8	TORTU	Mexico	13.84
10278	8	BERGS	Sweden	92.69
10279	8	LEHMS	Germany	25.83

1 2 3 4 5 6 1 of 6 pages (66 items)

EmployeeID: >5

Filter bar modes:

This specifies the grid to start the filter action while typing in the filter bar or after pressing the enter key based on the FilterBarMode.There are two types of FilterBarMode, they are the following.

1. OnEnter
2. Immediate

Note: For the FilterBarMode property you can assign either string value (OnEnter) or enum value (Syncfusion.JavaScript.FilterBarMode.OnEnter).

Filter bar message:

The filter bar message is supported only for the FilterType as 'FilterBar'. The filtered data with column name is displayed in the grid pager itself. By default the ShowFilterBarStatus is 'true'.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowFiltering()
.FilterSettings(filter => { filter.ShowFilterBarStatus(true); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	CustomerID	ShipCountry	Freight
	5			
10248	5	VINET	France	32.38
10254	5	CHOPS	Switzerland	22.98
10269	5	WHITC	USA	4.56
10297	5	BLONP	France	5.74
10320	5	WARTH	Finland	34.57
10333	5	WARTH	Finland	0.59
10358	5	LAMAI	France	19.64
10359	5	SEVES	UK	288.43
10372	5	QUEEN	Brazil	890.78
10378	5	FOLKO	Sweden	5.44
10397	5	PRINI	Portugal	60.26

1 1 of 1 pages (11 items)

EmployeeID: 5

Filter Operators

The grid controls uses filter operators from the DataManager, which are used at the time of filtering.

List of column type and filter operators.

Column Type	Filter Operators
Number	FilterOperatorType.GreaterThan
	FilterOperatorType.GreaterThanOrEqual
	FilterOperatorType.LessThan
	FilterOperatorType.LessThanOrEqual
	FilterOperatorType.Equal
	FilterOperatorType.NotEqual
String	FilterOperatorType.StartsWith
	FilterOperatorType.EndsWith
	FilterOperatorType.Contains
	FilterOperatorType.Equal

Copyright © 2001 - 2020 Syncfusion Inc.

	FilterOperatorType.NotEqual
Boolean	FilterOperatorType.Equal
	FilterOperatorType.NotEqual
Date	FilterOperatorType.GreaterThan
	FilterOperatorType.GreaterThanOrEqual
	FilterOperatorType.LessThan
	FilterOperatorType.LessThanOrEqual
	FilterOperatorType.Equal
	FilterOperatorType.NotEqual

FilterBar template

Usually enabling the AllowFiltering, will create default textbox in grid FilterBar. So, Using the [FilterBarTemplate] property of Columns we can render any other controls like AutoComplete, DropDownList etc in filterbar to filter the grid data for the particular column.

It has three functions. They are the following:

1. create - It is used to create the control at time of initialize.
2. read - It is used to read the filter value selected.
3. write - It is used to render the control and assign the value selected for filtering.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<EditableOrder>("Filtering")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowFiltering()
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("CustomerID").FilterBarTemplate(filterbar
=>
filterbar.Create("autoComplete_create").Write("autoComplete_write").Read("au
toComplete_read")).Width(90).Add();
col.Field("EmployeeID").HeaderText("EmployeeID").FilterBarTemplate(filterbar
=>
filterbar.Write("dropdown_write").Read("dropdown_read")).TextAlign(TextAlign
.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").FilterBarTemplate(filterbar =>
filterbar.Write("numeric_write").Read("numeric_read")).TextAlign(TextAlign.R
ight).Width(75).Format("{0:C}").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(90).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Verified").HeaderText("Verified").TextAlign(TextAlign.Center).Fil
terBarTemplate(filterbar =>
filterbar.Write("check_write").Read("check_read")).Width(80).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Filtering()
{
var DataSource = OrderRepository.GetAllRecords().ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

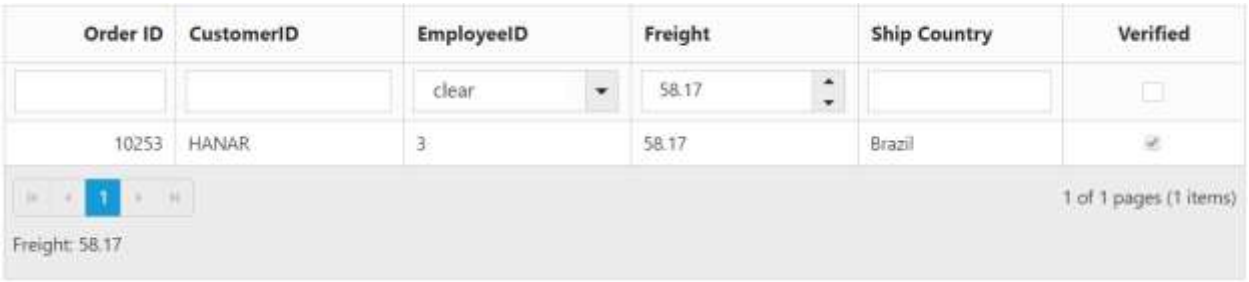
JS

```
<script type="text/javascript">
function autoComplete_create(args) {
return "<input>"
}
function autoComplete_write(args) {
var gridObj = $("#Filtering").ejGrid("instance");
var data = ej.DataManager(gridObj.model.dataSource).executeLocal(new
ej.Query().select("CustomerID"));
args.element.ejAutocomplete({ width: "100%", dataSource: data,
enableDistinct: true, focusOut: ej.proxy(args.column.filterBarTemplate.read,
this, args) });
}
function autoComplete_read(args) {
this.filterColumn(args.column.field, "equal", args.element.val(), "and",
true)
}
function dropdown_write(args) {
var data = [{ text: "clear", value: "clear" }, { text: "1", value: 1 }, {
text: "2", value: 2 }, { text: "3", value: 3 }, { text: "4", value: 4 },
{ text: "5", value: 5 }, { text: "6", value: 6 }, { text: "7", value: 7 }, {
text: "8", value: 8 }, { text: "9", value: 9 }
]
args.element.ejDropDownList({ width: "100%", dataSource: data, change:
ej.proxy(args.column.filterBarTemplate.read, this, args) })
}
function dropdown_read(args) {
if (args.element.val() == "clear") {
this.clearFiltering(args.column.field);
args.element.val("")
}
this.filterColumn(args.column.field, "equal", args.element.val(), "and",
true)
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function numeric_write(args) {
args.element.ejNumericTextbox({ width: "100%",decimalPlaces: 2, focusOut:
ej.proxy(args.column.filterBarTemplate.read, this, args) });
}
function numeric_read(args) {
this.filterColumn(args.column.field, "equal", args.element.val(), "and",
true)
}
function check_write(args) {
args.element.ejCheckBox({ change:
ej.proxy(args.column.filterBarTemplate.read, this, args) });
}
function check_read(args) {
this.filterColumn(args.column.field, "equal",
args.element.parent().attr('aria-checked'), "and", true)
}
</script>
```

The following output is displayed as a result of the previous code example.



After Filtering

Editing

The grid control has support for dynamic insertion, updating and deletion of records. You can start the edit action either by double clicking the particular row or by selecting the required row and clicking on Edit icon in toolbar. Similarly, you can add new record to grid either by clicking on insert icon in toolbar or on an external button which is bound to call addRecord method of grid. Save and Cancel while on edit mode is possible using respective toolbar icon in grid.

Deletion of the record is possible by selecting the required row and clicking on Delete icon in toolbar.

The primary key for the data source should be defined in Columns definition, for editing to work properly. In Columns definition, particular primary column's IsPrimaryKey property should be set to true. Refer the Knowledge base link for more information.

Note: 1. In grid, the primary key column will be automatically set to read only while editing the row, but you can specify primary key column value while adding a new record.

2. The column which is specified as IsIdentity will be in readonly mode both while editing and adding a record. Also, auto incremented value is assigned to that IsIdentity column.

Copyright © 2001 - 2020 Syncfusion Inc.

Toolbar with edit option

Using toolbar which is rendered at the top of the grid header, you can show all the CRUD related action. To enable toolbar and toolbar items, set `ShowToolbar` property as true and `ToolbarItems`. The default toolbar items are `Add`, `Edit`, `Delete`, `Update` and `Cancel`.

Note: For `ToolbarItems` property you can assign either `string` value ("Add") or `enum` value (`Syncfusion.JavaScript.ToolBarItems.Add`).

The following code example describes the above behavior.

RAZOR

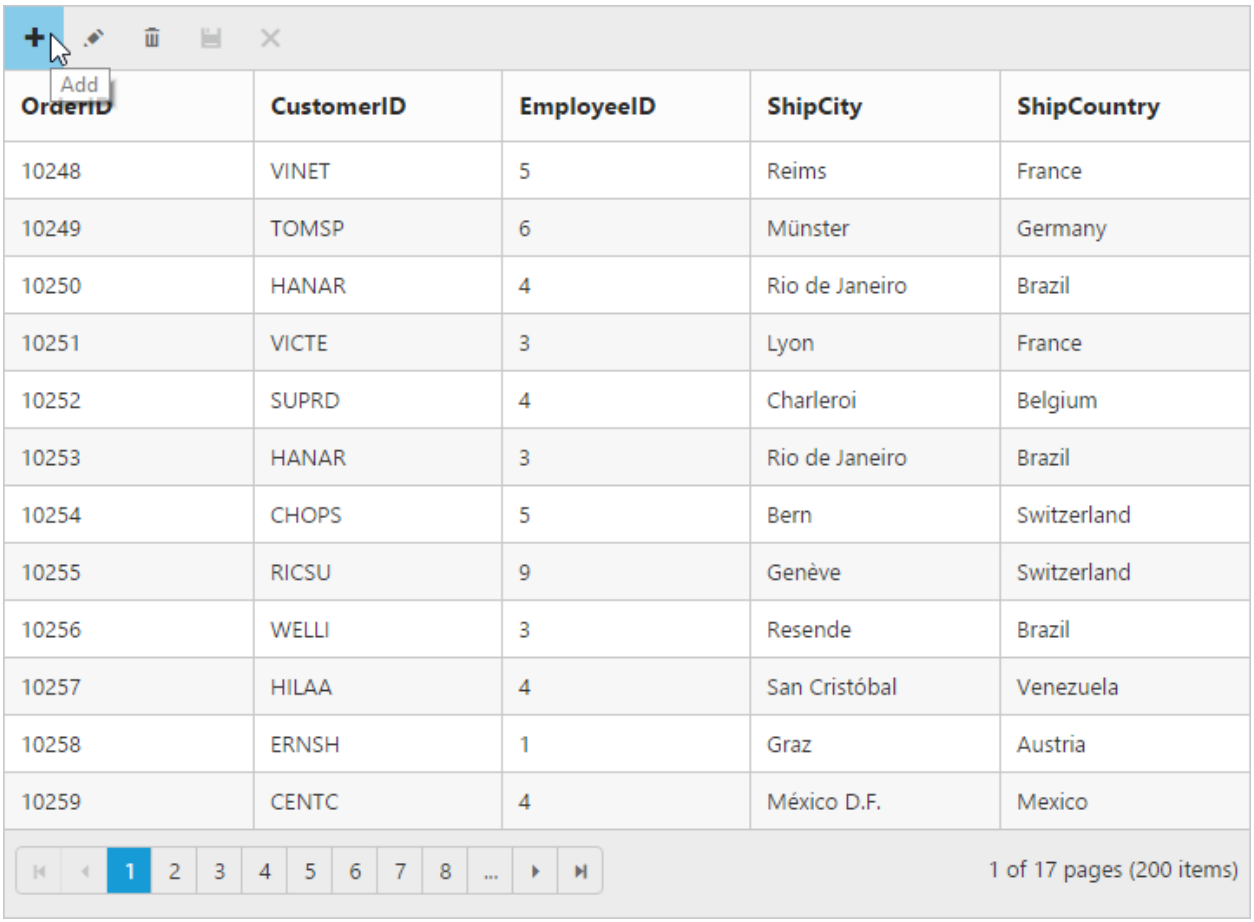
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Cell edit type and its edit options

The edit type of bound column can be customized using `EditType` property of `Columns`. The following Essential JavaScript controls are supported built-in by `EditType`. You can set the `EditType` based on specific data type of the column.

- `NumericTextBox` control for integers, double, and decimal data types.
- `DatePicker` control for date data type.
- `DateTimePicker` control for date-time data type.
- `DropDownList` control for list of data type.

And also you can define the model for all the editTypes controls while editing through EditOptions.

The following table describes `EditType` and their corresponding EditOptions of the specific data type of the column.

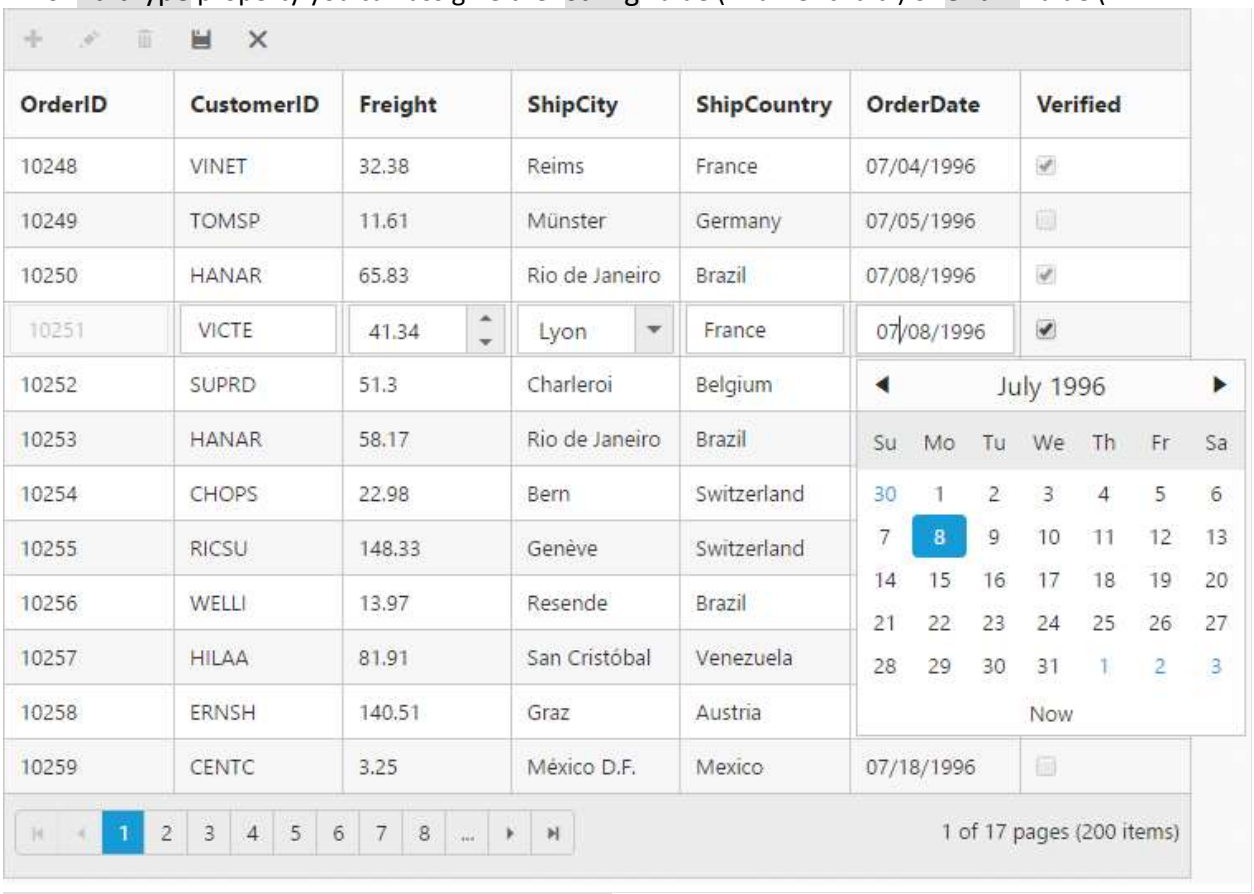
EditType	EditParams	Example
NumericTextBox	TextBoxes	NumericEditOptions(new EditorProperties() { DecimalPlaces = 2, })
DatePicker	DatePicker	DateEditOptions(new DatePickerProperties() { ButtonText="Now"})
DateTimePicker	DateTimePicker	DateTimeEditOptions(new DateTimePickerProperties() {Enabled=False})

Copyright © 2001 - 2020 Syncfusion Inc.

DropDownList	DropDownList	<DropDownEditOptions(new DropDownListProperties() {ShowCheckbox=True})/>

Note: 1. If EditType is not set, then by default it will display the input element ("stringEdit") while editing a column.

2. For EditType property you can assign either string value ("NumericEdit") or enum value (



Syncfusion.JavaScript.EditingType.NumericEdit).

The following code example describes the above behavior

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCity").HeaderText("Ship
City").EditType(EditingType.DropdownEdit).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).DateEditOptions(new
DatePickerProperties() { ButtonText="Now"}).Format("{0:MM/dd/yyyy}").Add();
col.Field("Verified").HeaderText("Verified").EditType(EditingType.BooleanEdi
t).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

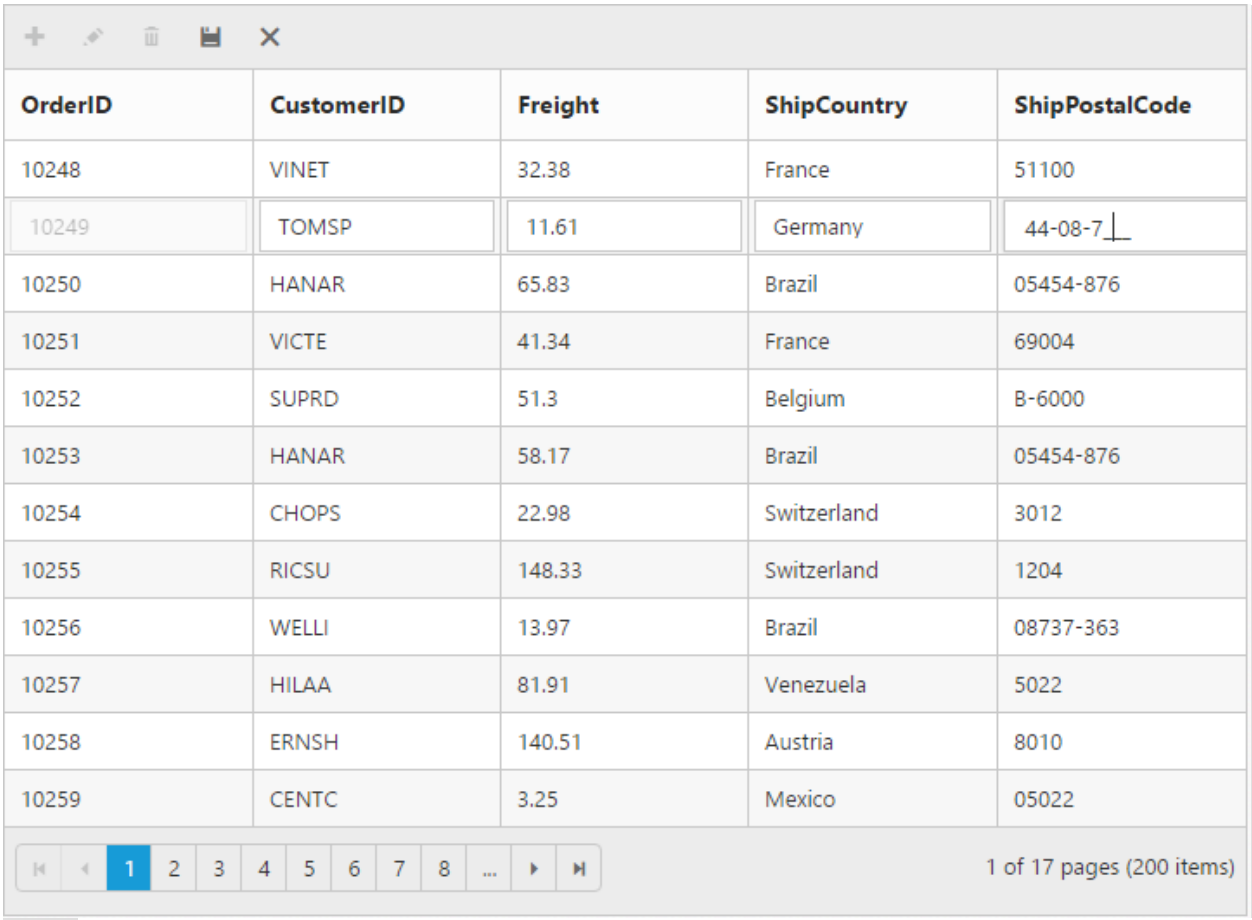
The following output is displayed as a result of the above code example.

Cell Edit Template

On editing the column values, custom editor can be created by using EditTemplate property of Columns. It has three functions, they are

1. Create - It is used to create the control at time of initialize.
2. Read - It is used to read the input value at time of save.

Copyright © 2001 - 2020 Syncfusion Inc.

3. 

OrderID	CustomerID	Freight	ShipCountry	ShipPostalCode
10248	VINET	32.38	France	51100
10249	TOMSP	11.61	Germany	44-08-7
10250	HANAR	65.83	Brazil	05454-876
10251	VICTE	41.34	France	69004
10252	SUPRD	51.3	Belgium	B-6000
10253	HANAR	58.17	Brazil	05454-876
10254	CHOPS	22.98	Switzerland	3012
10255	RICSU	148.33	Switzerland	1204
10256	WELLI	13.97	Brazil	08737-363
10257	HILAA	81.91	Venezuela	5022
10258	ERNSH	140.51	Austria	8010
10259	CENTC	3.25	Mexico	05022

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Write - It is used to assign the value to control at time of editing.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").HeaderText("Freight").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("ShipPostalCode").HeaderText("Ship Postal Code").EditTemplate(a =>
{ a.Create("create").Read("read").Write("write"); }).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

```
<script id="template" type="application/javascript">
function create()
{
return $("<input>");
}
function write(args)
{
args.element.ejMaskEdit({
maskFormat : "99-99-9999",
value : args.rowdata["ShipPostalCode"]
});
}
function read(args)
{
return args.ejMaskEdit("get_StrippedValue");
}
</script>
```

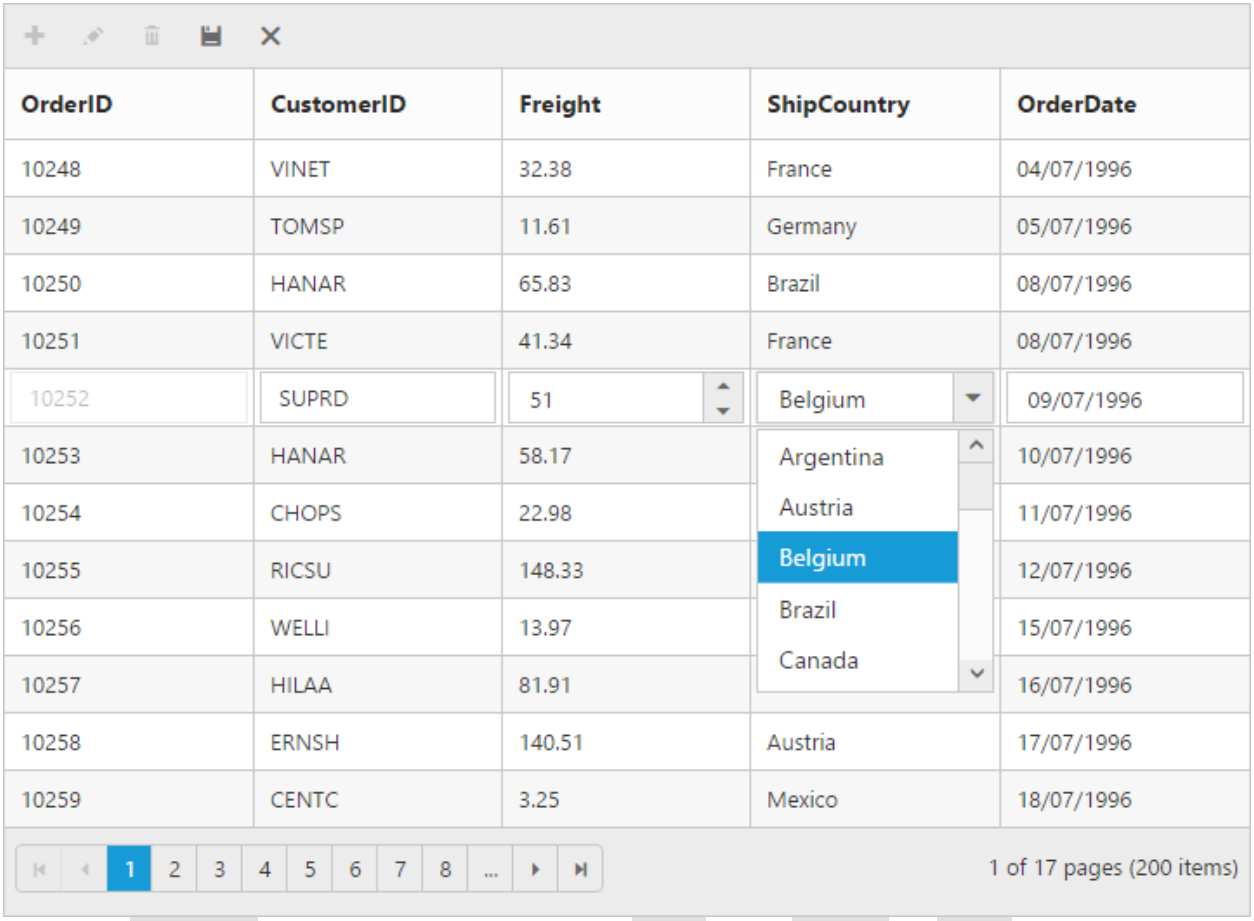
The following output is displayed as a result of the above code example.

Edit Modes

Inline

Set `EditMode` as `Normal`, then the row itself is changed as edited row.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: For EditMode property you can assign either string value (Normal) or enum value (Syncfusion.JavaScript.EditMode.Normal).

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.Normal);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship
Country").EditType(EditingType.DropdownEdit).Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

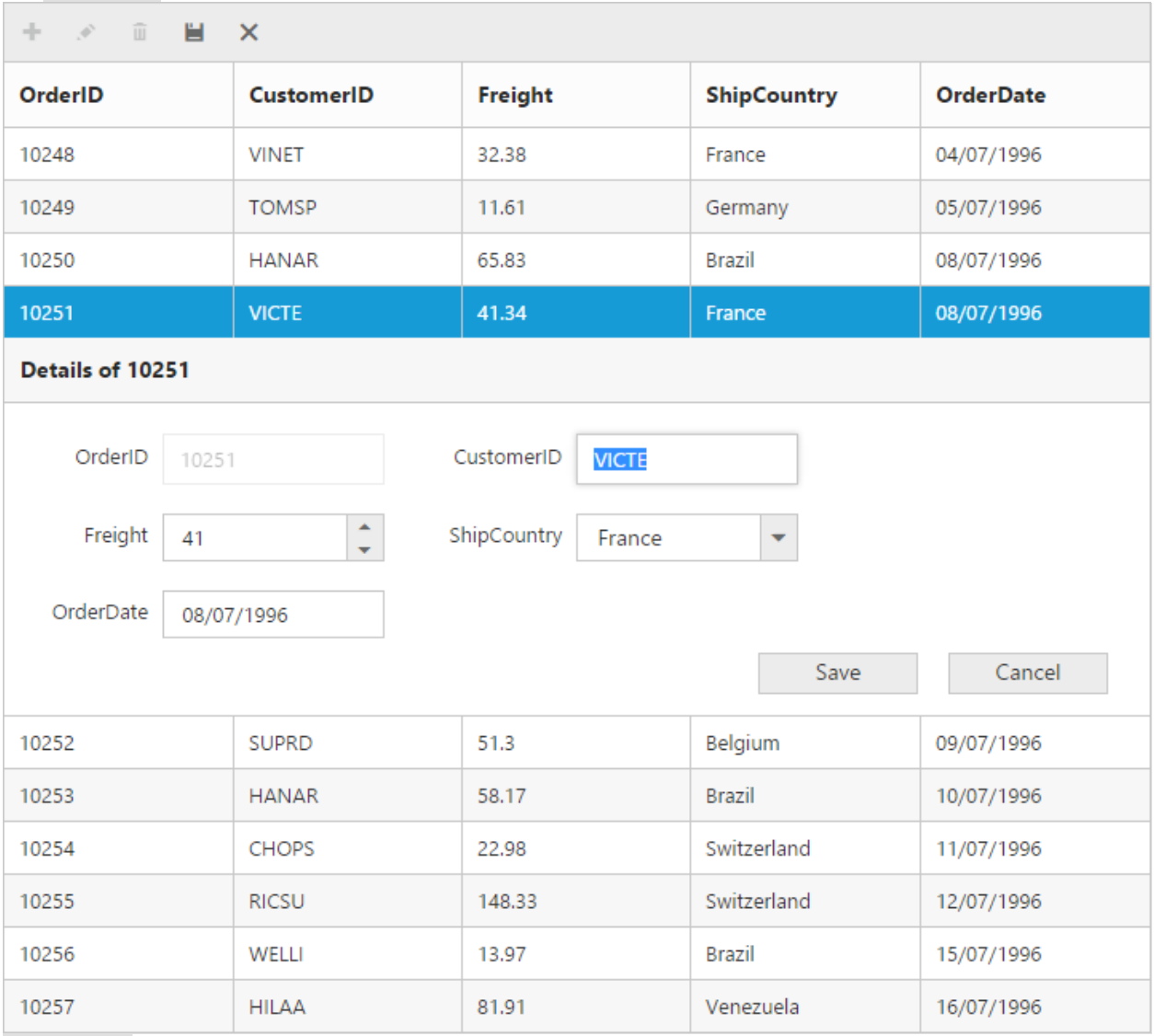
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

Inline Form

Set EditMode as



InlineForm, then edit form will be inserted next to the row which is to be edited.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.InlineFo
rm); })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Inline Template Form

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit the fields that are not bound to grid columns.

To edit the records using Inline template form, set `EditMode` as `InlineFormTemplate` and specify the template ID to `InlineFormTemplateID` property of `EditSettings`.

While using template form, you can change the HTML elements to appropriate JS controls based on the column type. This can be achieved by using `ActionComplete` event of grid.

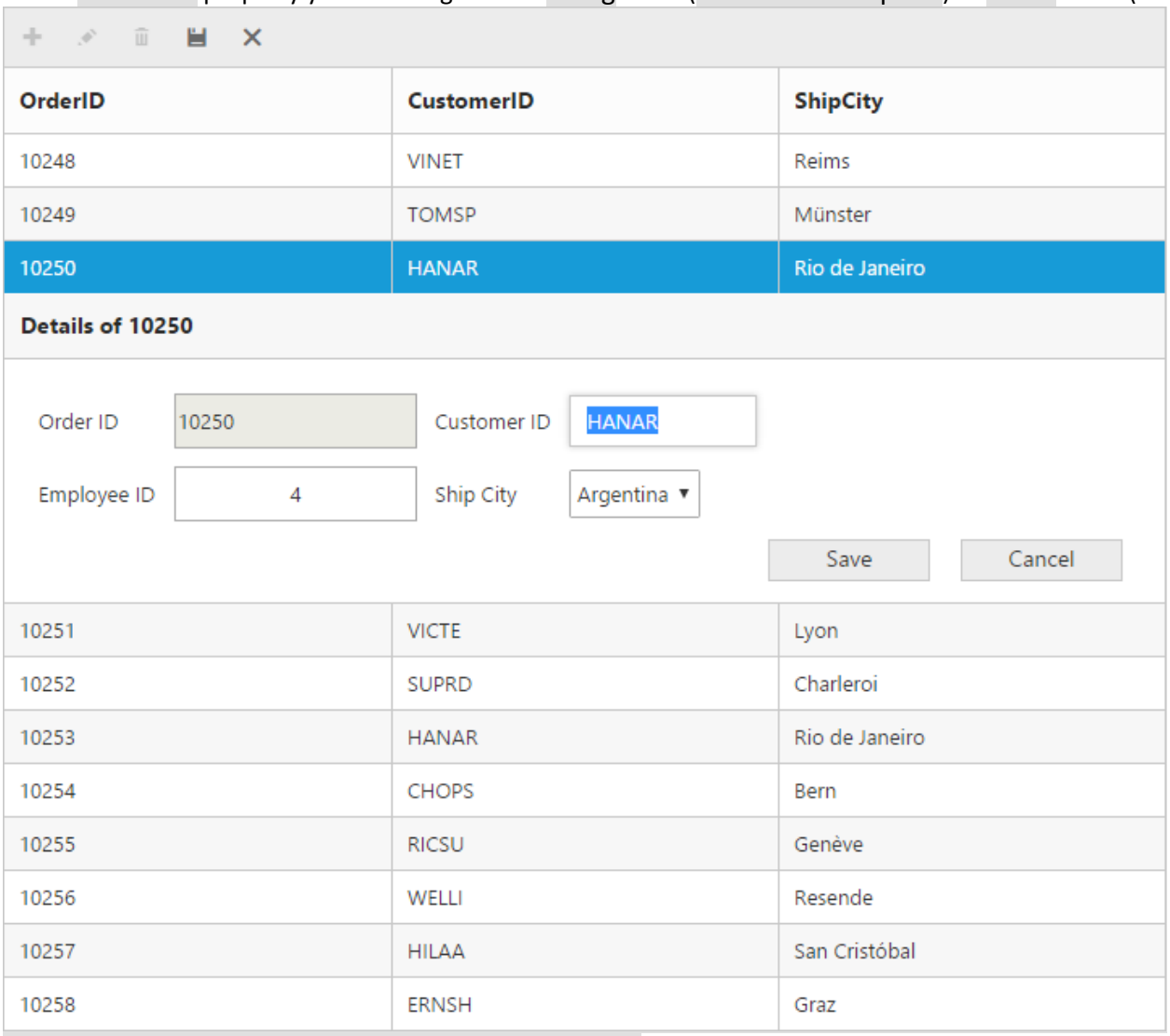
Note: 1. `value` attribute is used to bind the corresponding field value while editing.

2. `name` attribute is used to get the changed field values while saving the edited record.

3. It's a standard way to enclose the `Template` within the `script` tag with `type` as "text/x-jsrender".

Copyright © 2001 - 2020 Syncfusion Inc.

4. For EditMode property you can assign either string value (InlineFormTemplate) or enum value (



Syncfusion.JavaScript.EditMode.InlineTemplateForm)

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.ClientSideEvents(eve => { eve.ActionComplete("complete"); })
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.InlineFo
rmTemplate).InlineFormTemplateID("#template");})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("ShipCity").HeaderText("Ship
City").EditType(EditingType.DropdownEdit).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

```
<script id="template" type="text/template">
<table cellspacing="10">
<tr>
<td>Order ID</td>
<td>
<input id="OrderID" name="OrderID" disabled="disabled"
value="{{"{{"}}:OrderID{{}}}}" class="e-field e-ejinputtext"
style="width:116px;height:28px" />
</td>
<td>Customer ID</td>
<td>
<input id="CustomerID" name="CustomerID" value="{{"{{"}}:CustomerID{{}}}}"
class="e-field e-ejinputtext" style="width: 116px; height: 28px" />
</td>
</tr>
<tr>
<td>Employee ID</td>
<td>
<input type="text" id="EmployeeID" name="EmployeeID"
value="{{"{{"}}:EmployeeID{{}}}}" />
</td>
<td>Ship City</td>
<td>
<select id="ShipCity" name="ShipCity">
<option value="Argentina">Argentina</option>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<option value="Austria">Austria</option>
<option value="Belgium">Belgium</option>
<option value="Brazil">Brazil</option>
<option value="Canada">Canada</option>
<option value="Denmark">Denmark</option>
</select>
</td>
</tr>
</table>
</script>
<script>
function complete(args) {
$("#EmployeeID").ejNumericTextbox();
$("#Freight").ejNumericTextbox();
$("#ShipCity").ejDropDownList();
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.

![](Editing

OrderID	CustomerID	ShipCity
10248	VINET	Reims
10249	TOMSP	Münster
10250	HANAR	Rio de Janeiro
Details of 10250		
Order ID: 10250	Customer ID: HANAR	
Employee ID: 4	Ship City:	
Save	Cancel	
10251	VICTE	Lyon
10252	SUPRD	Charleroi
10253	HANAR	Rio de Janeiro
10254	CHOPS	Bern
10255	RICSU	Genève
10256	WELLI	Resende
10257	HILAA	San Cristóbal
10258	ERNSH	Graz

*images/Editing*img6.png)

Before the template elements are converted to JS controls

After the template elements are converted to JS controls using actionComplete event.

Note: To render the EJ MVC form controls like DatePicker, Dropdownlist, NumericTextBox in the template edit form, from the partial view, refer the KB How render ejControls in the Edit Form Template of the MVC Grid.

Dialog

Set EditMode as Dialog to edit data using a dialog box, which displays the fields associated with the data record being edited.

The following code example describes the above behavior.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

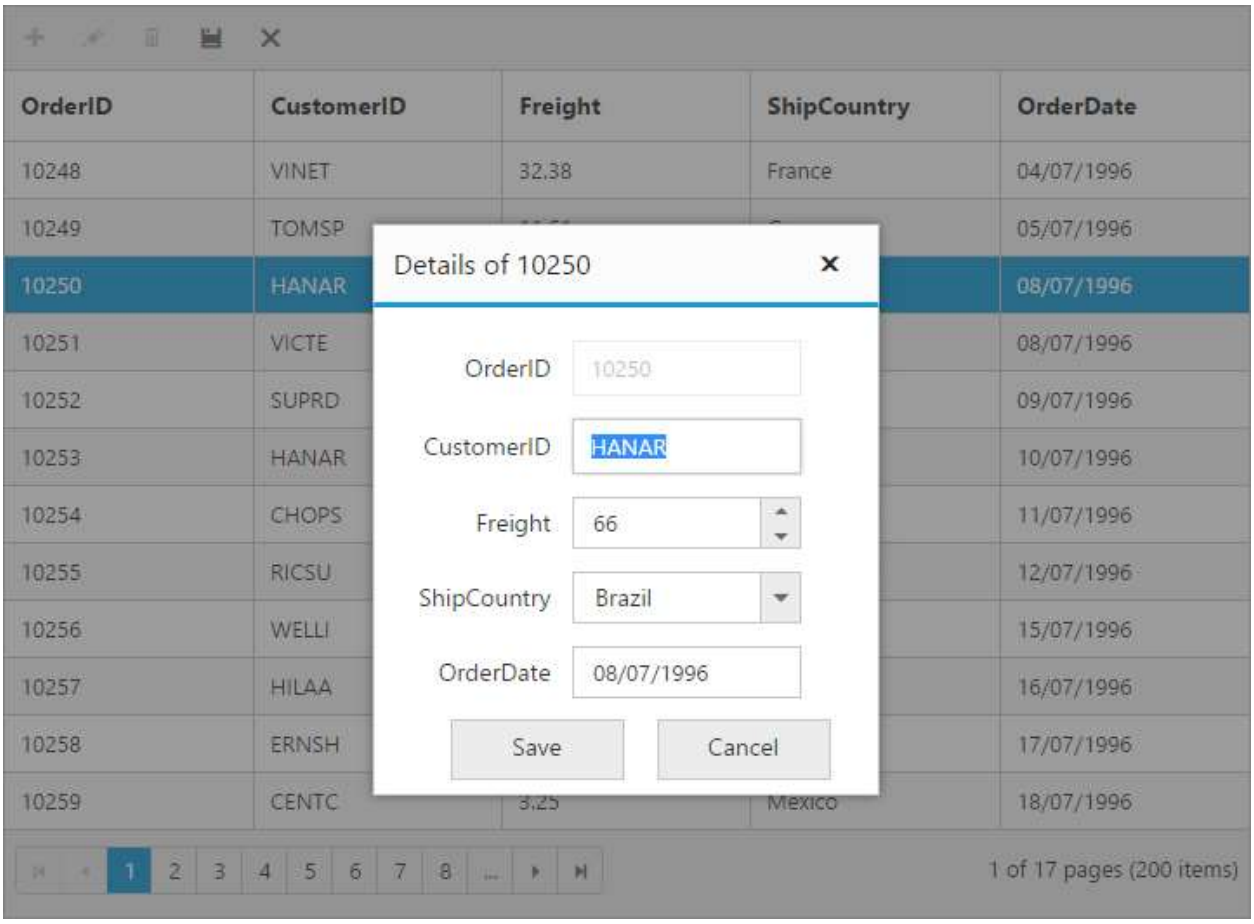
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.Dialog);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Dialog Template Form

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit the fields that are not bound to grid columns.

To edit the records using Dialog template form, set EditMode as 'DialogTemplate' and specify the template id to DialogEditorTemplateID property of EditSettings.

While using template, you can change the elements that are defined in the Template, to appropriate JS controls based on the column type. This can be achieved by using ActionComplete event of grid.

Note: 1. value attribute is used to bind the corresponding field value while editing.

2. name attribute is used to get the changed field values while save the edited record.

3. For EditMode property you can assign either string value (DialogTemplate) or enum value (Syncfusion.JavaScript.EditMode.DialogTemplate).

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.ClientSideEvents(eve => { eve.ActionComplete("complete"); })
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.DialogTe
mplate).DialogEditorTemplateID("#template"); })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("ShipCity").HeaderText("Ship
City").EditType(EditingType.DropdownEdit).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

JS

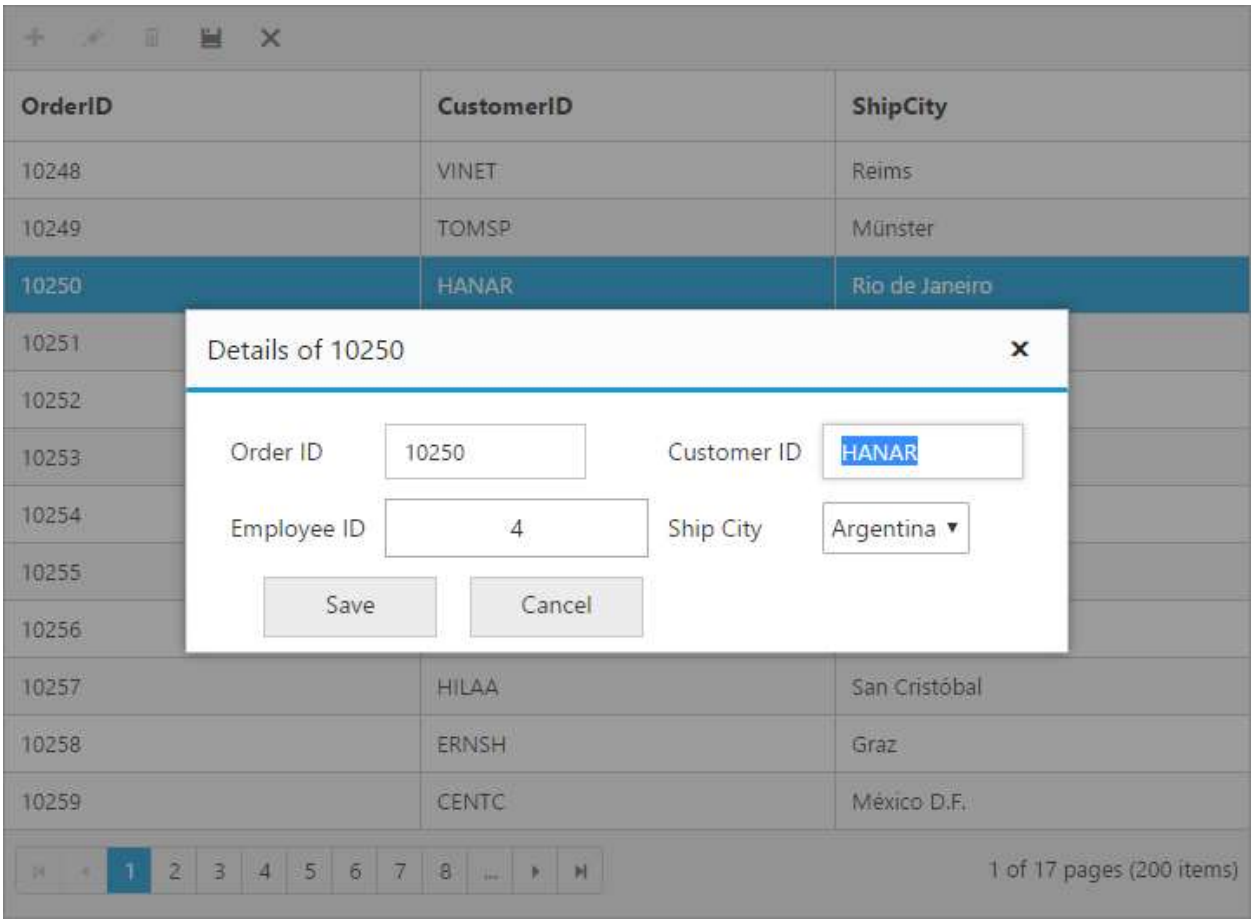
```
<script id="template" type="text/template">
<table cellspacing="10">
<tr>
<td>Order ID</td>
<td>
<input id="OrderID" name="OrderID" disabled="disabled"
value="{{"{{"}}:OrderID{{}}}}" class="e-field e-ejinputtext"
style="width:116px;height:28px" />
</td>
<td>Customer ID</td>
<td>
<input id="CustomerID" name="CustomerID" value="{{"{{"}}:CustomerID{{}}}}"
class="e-field e-ejinputtext" style="width: 116px; height: 28px" />
</td>
</tr>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<tr>
<td>Employee ID</td>
<td>
<input type="text" id="EmployeeID" name="EmployeeID"
value="{{"{{"}}:EmployeeID{{}}}}" />
</td>
<td>Ship City</td>
<td>
<select id="ShipCity" name="ShipCity">
<option value="Argentina">Argentina</option>
<option value="Austria">Austria</option>
<option value="Belgium">Belgium</option>
<option value="Brazil">Brazil</option>
<option value="Canada">Canada</option>
<option value="Denmark">Denmark</option>
</select>
</td>
</tr>
</table>
</script>
<script>
function complete(args) {
$("#EmployeeID").ejNumericTextbox();
$("#Freight").ejNumericTextbox();
$("#ShipCity").ejDropDownList();
}
</script>
```

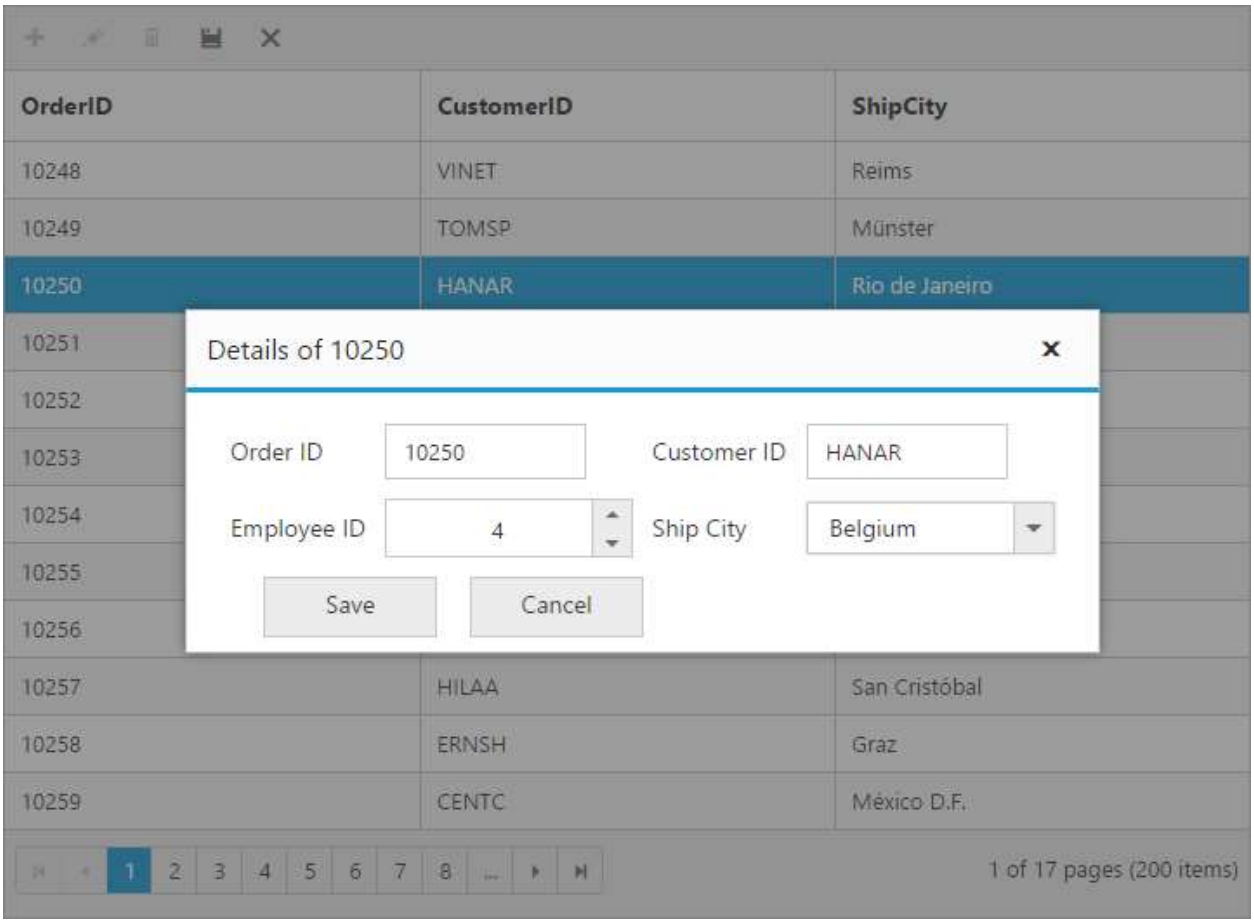
The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Before the template elements are converted to JS controls

Copyright © 2001 - 2020 Syncfusion Inc.



After the template elements are converted to JS controls using actionComplete event.

External Form

By setting the EditMode as ExternalForm, the edit form is opened outside the grid content.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.External
Form); })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

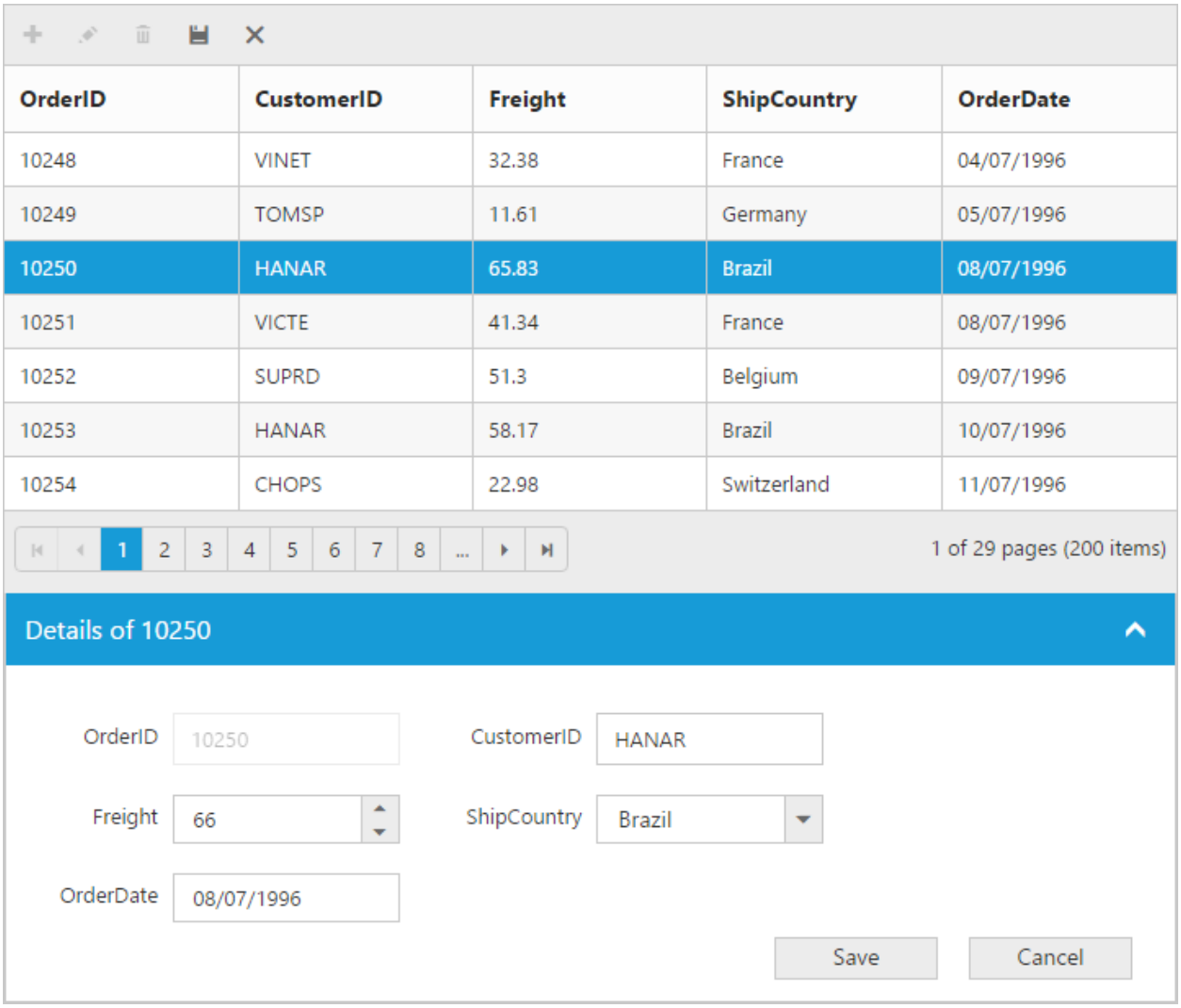
```
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Form Position:

You can position an External edit form in the following two ways.

1. Top-right
2. Bottom left

This can be achieved by setting the FormPosition property of EditSettings as 'TopRight' or 'BottomLeft'.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.External
Form).FormPosition(FormPosition.TopRight);  })
.ToolbarSettings(toolbar =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

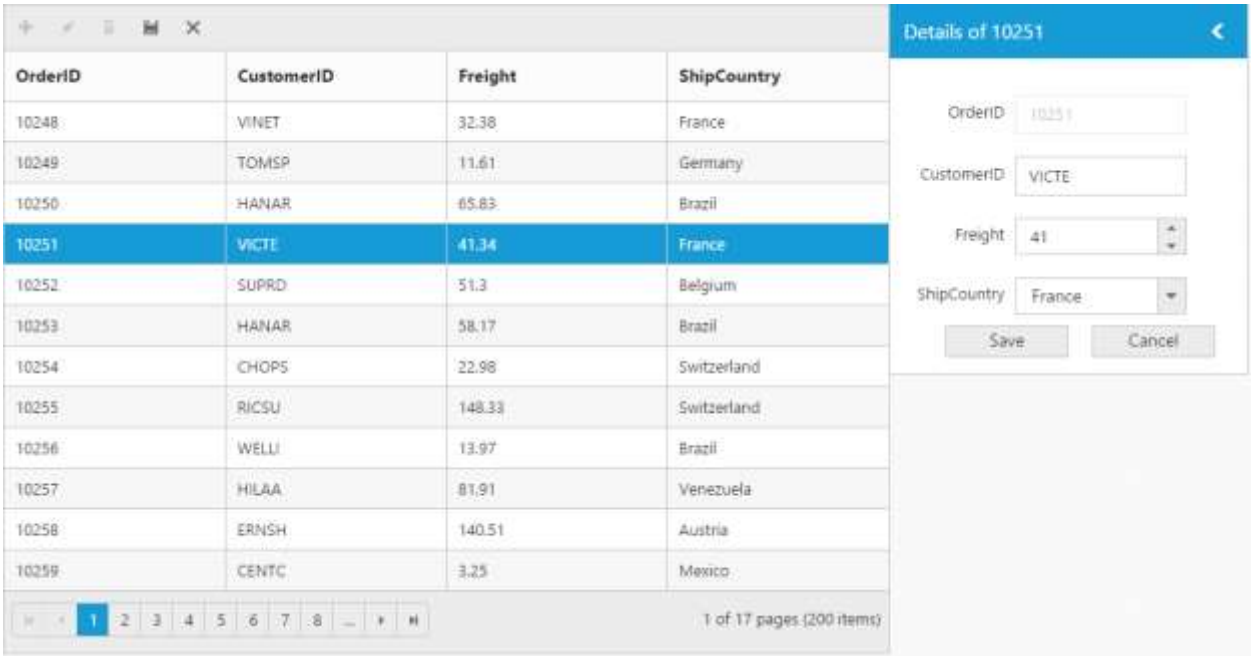
```
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



External Template Form

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit the fields that are not bound to grid columns.

To edit the records using External template form, set EditMode as ExternalFormTemplate and specify the template id to ExternalFormTemplateID property of EditSettings.

While using template, you can change the elements that are defined in the template, to appropriate JS controls based on the column type. This can be achieved by using ActionComplete event of grid.

Note: 1. value attribute is used to bind the corresponding field value while editing.

2. name attribute is used to get the changed field values while save the edited record.

3. For EditMode property you can assign either string value (ExternalFormTemplate) or enum value (Syncfusion.JavaScript.EditMode.ExternalFormTemplate).

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.ClientSideEvents(eve => { eve.ActionComplete("complete"); })
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.External
FormTemplate).ExternalFormTemplateID("#template");})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("ShipCity").HeaderText("Ship
City").EditType(EditingType.DropdownEdit).Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

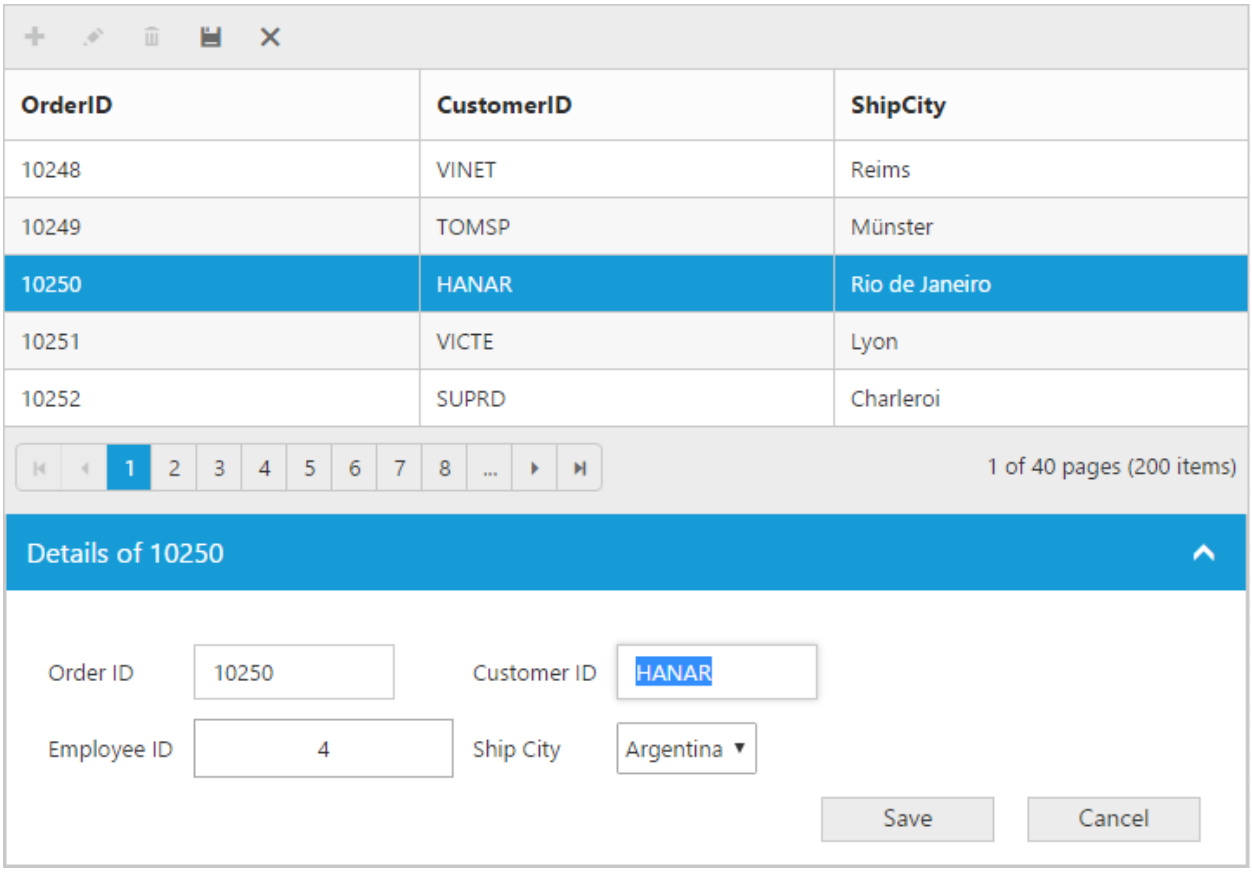
JS

```
<script id="template" type="text/template">
<table cellspacing="10">
<tr>
<td>Order ID</td>
<td>
<input id="OrderID" name="OrderID" disabled="disabled"
value="{{"{{"}}:OrderID{{}}}}" class="e-field e-ejinputtext"
style="width:116px;height:28px" />
</td>
<td>Customer ID</td>
<td>
<input id="CustomerID" name="CustomerID" value="{{"{{"}}:CustomerID{{}}}}"
class="e-field e-ejinputtext" style="width: 116px; height: 28px" />
</td>
</tr>
<tr>
<td>Employee ID</td>
<td>
<input type="text" id="EmployeeID" name="EmployeeID"
value="{{"{{"}}:EmployeeID{{}}}}" />
</td>
<td>Ship City</td>
<td>
```

Copyright © 2001 - 2020 Syncfusion Inc.

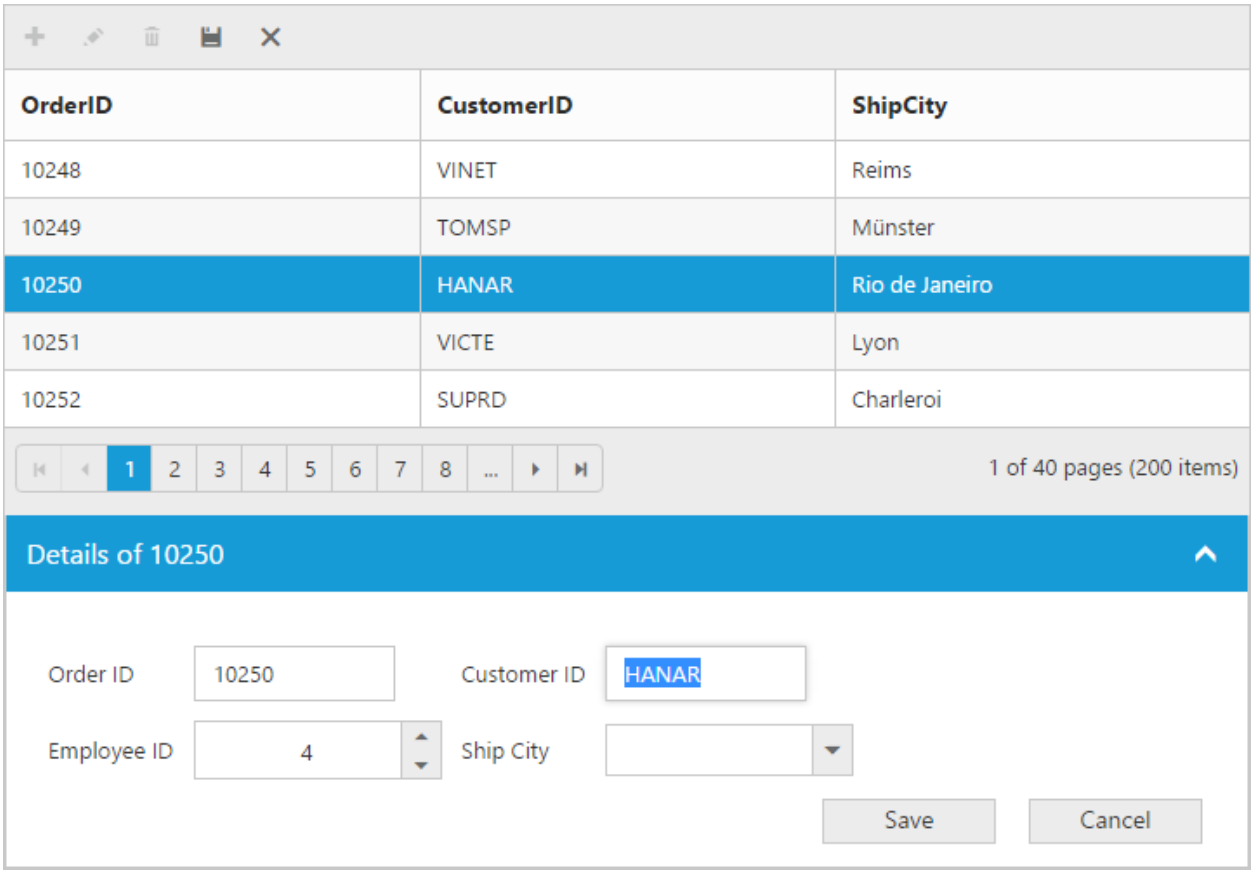
```
<select id="ShipCity" name="ShipCity">
<option value="Argentina">Argentina</option>
<option value="Austria">Austria</option>
<option value="Belgium">Belgium</option>
<option value="Brazil">Brazil</option>
<option value="Canada">Canada</option>
<option value="Denmark">Denmark</option>
</select>
</td>
</tr>
</table>
</script>
<script>
function complete(args) {
$("#EmployeeID").ejNumericTextbox();
$("#Freight").ejNumericTextbox();
$("#ShipCity").ejDropDownList();
}
</script>
```

The following output is displayed as a result of the above code example.



Before the template elements are converted to JS controls

Copyright © 2001 - 2020 Syncfusion Inc.



After the template elements are converted to JS controls using actionComplete event.

Note: To render the EJ MVC form controls like DatePicker, Dropdownlist, NumericTextBox in the template edit form, from the partial view, refer the KB How render ejControls in the Edit Form Template of the MVC Grid.

Batch / Excel-like

Users can start editing by clicking a cell and typing data into it. Edited cell will be marked while navigating to next cell or any other row, so that you know which fields or cells has been edited. Set `EditMode` as `Batch` to enable batch editing.

Note: `getBatchChanges` method of grid holds the unsaved record changes.

Refer the KB link for "How to suppress grid confirmation messages" in batch mode.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.Batch);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

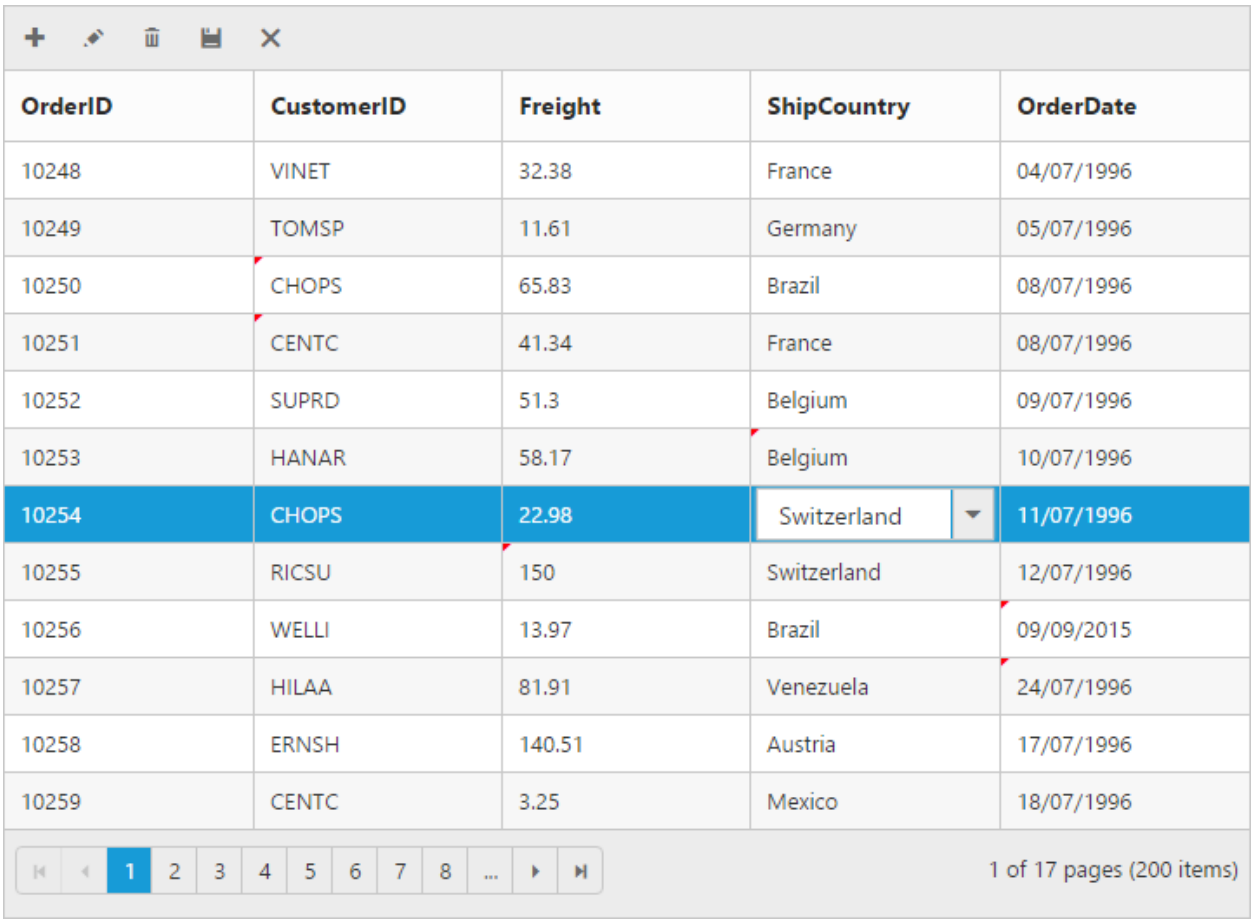
```
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	CustomerID	Freight	ShipCountry	OrderDate
10248	VINET	32.38	France	04/07/1996
10249	TOMSP	11.61	Germany	05/07/1996
10250	CHOPS	65.83	Brazil	08/07/1996
10251	CENTC	41.34	France	08/07/1996
10252	SUPRD	51.3	Belgium	09/07/1996
10253	HANAR	58.17	Belgium	10/07/1996
10254	CHOPS	22.98	Switzerland	11/07/1996
10255	RICSU	150	Switzerland	12/07/1996
10256	WELLI	13.97	Brazil	09/09/2015
10257	HILAA	81.91	Venezuela	24/07/1996
10258	ERNSH	140.51	Austria	17/07/1996
10259	CENTC	3.25	Mexico	18/07/1996

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Confirmation messages

To show the confirm dialog while saving or discarding the Batch changes (discarding during the grid action like filtering, sorting and paging), set ShowConfirmDialog as true.

Note: ShowConfirmDialog property is only for Batch editing mode.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.Batch).S
howConfirmDialog(); })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

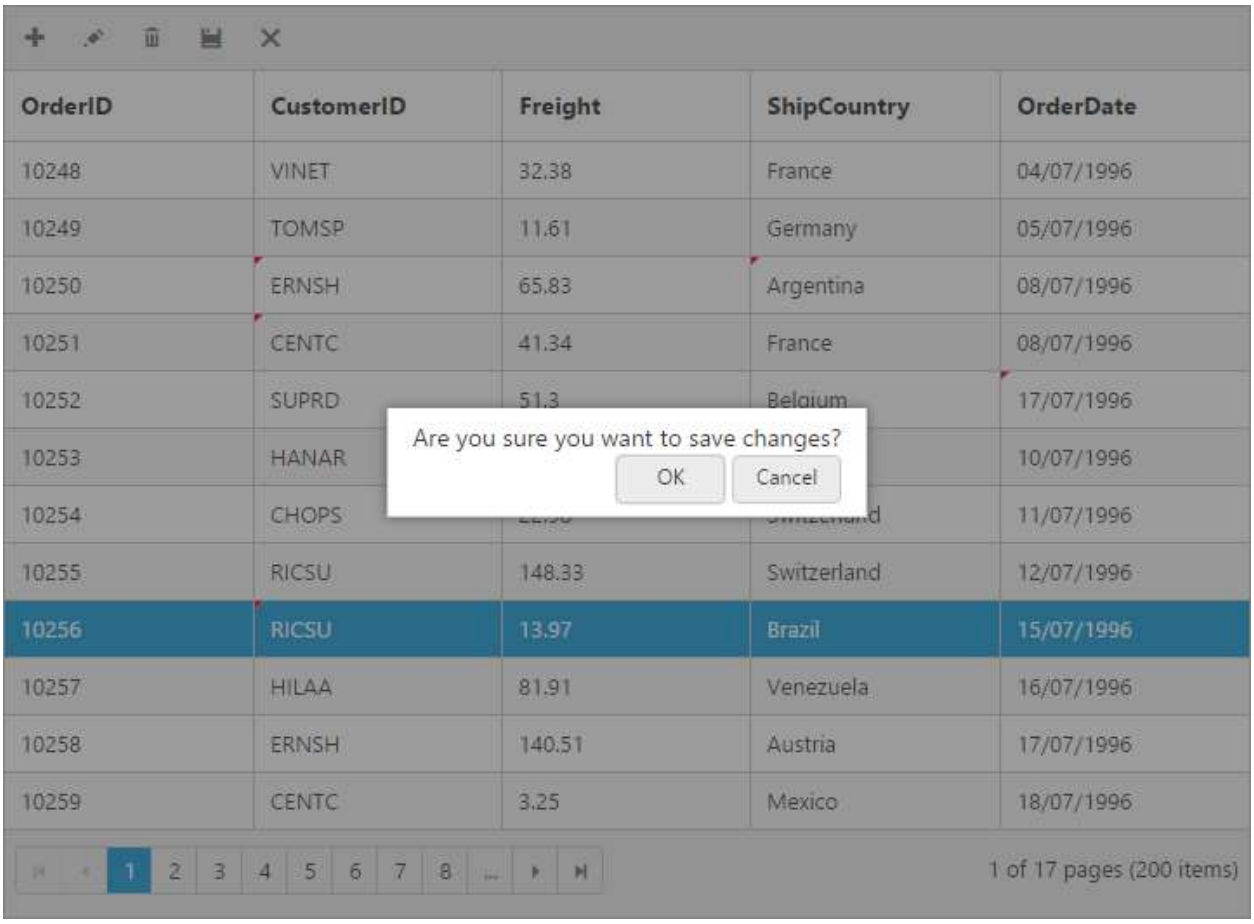
```
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



To show delete confirm dialog while deleting a record, set ShowDeleteConfirmDialog as true.

Note: ShowDeleteConfirmDialog property is for all type of EditMode.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().ShowDeleteConfirmDialog();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

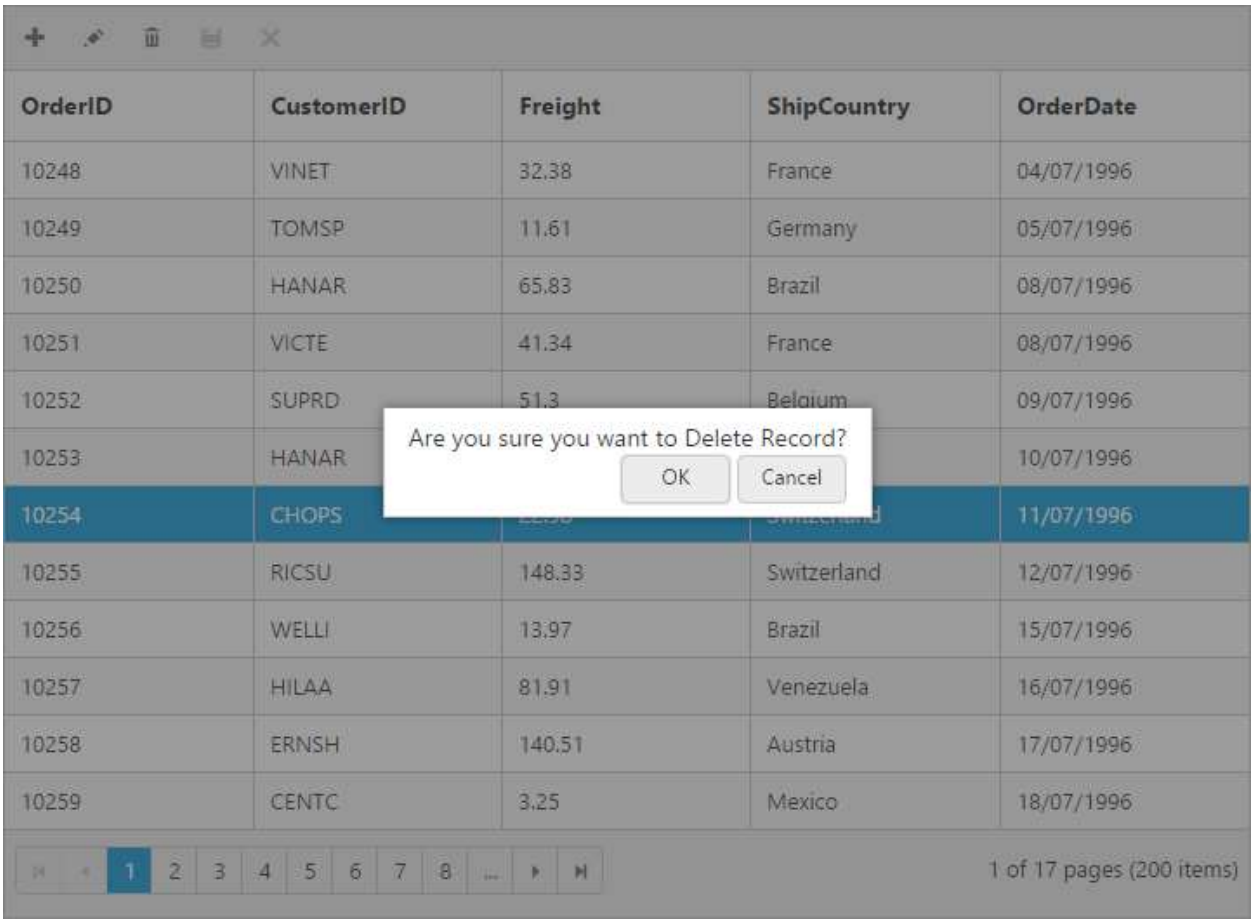
```
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 }).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("OrderDate").HeaderText("Order
Date").EditType(EditingType.Datepicker).Format("{0:MM/dd/yyyy}").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Column Validation

We can validate the value of the added or edited record cell before saving.

The below validation script files are needed when editing is enabled with validation.

1. jquery.validate.min.js
2. jquery.validate.unobtrusive.min.js

jQuery Validation

You can set validation rules using `ValidationRules` property of `Columns`. The following are jQuery validation methods.

List of Jquery validation methods

Rules	Description
required	Requires an element.
remote	Requests a resource to check the element for validity.
minlength	Requires the element to be of given minimum length.
maxlength	Requires the element to be of given maximum length.

Copyright © 2001 - 2020 Syncfusion Inc.

range	Requires the element to be in given value range.
min	The element requires a given minimum.
max	The element requires a given maximum.
email	The element requires a valid email.
url	The element requires a valid URL
date	Requires the element to be a date.
dateISO	The element requires an ISO date.
number	The element requires a decimal number.
digits	The element requires digits only.
creditcard	Requires the element to be a credit card number.
equalTo	Requires the element to be the same as another.

Grid supports all the standard validation methods of jQuery, please refer the jQuery validation documentation link for more information.

To return custom validation message from server end, please refer this KB link.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).ValidationRules(v =>
v.AddRule("required",true).AddRule("number", true)).Add();
col.Field("CustomerID").HeaderText("Customer ID").ValidationRules(v =>
v.AddRule("required", true).AddRule("minlength", 3)).Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.ValidationRules(v => v.AddRule("range", "[0,1000]")).Add();
```

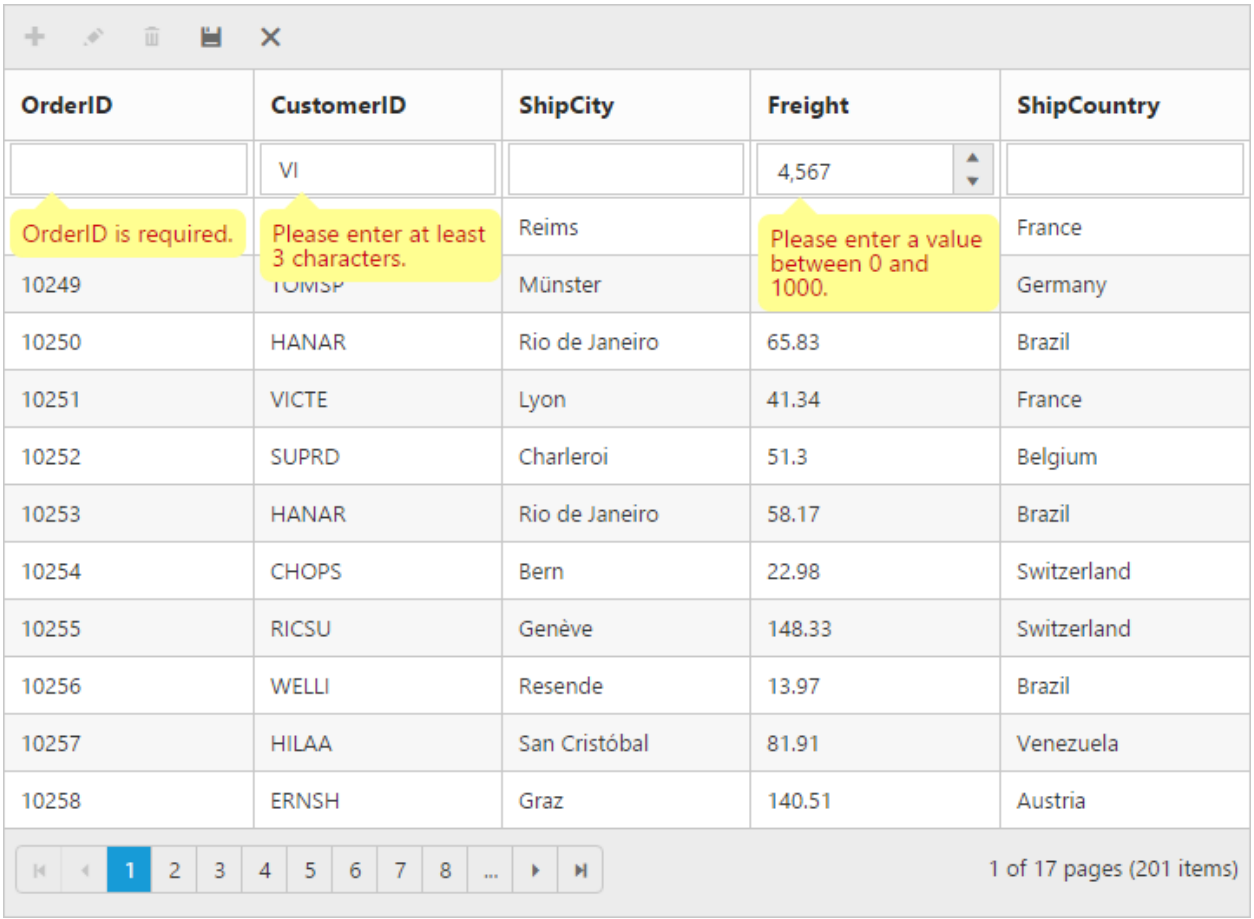
Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Note: 1. Refer this Knowledge Base link to perform server side validation in Grid.

Custom Validation

In addition to jQuery validation methods, you can also add your own custom validation methods for a specific column. Function call to custom validator function to be mentioned within ValidationRules property of Columns.

Using messages property of ValidationRules you can specify the error message for that column.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").ValidationRules(v =>
v.AddRule("customRegex", 5)).Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.ValidationRules(rule => rule.AddRule("customCompare", new List<object>() {
0, 1000 })).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

JS

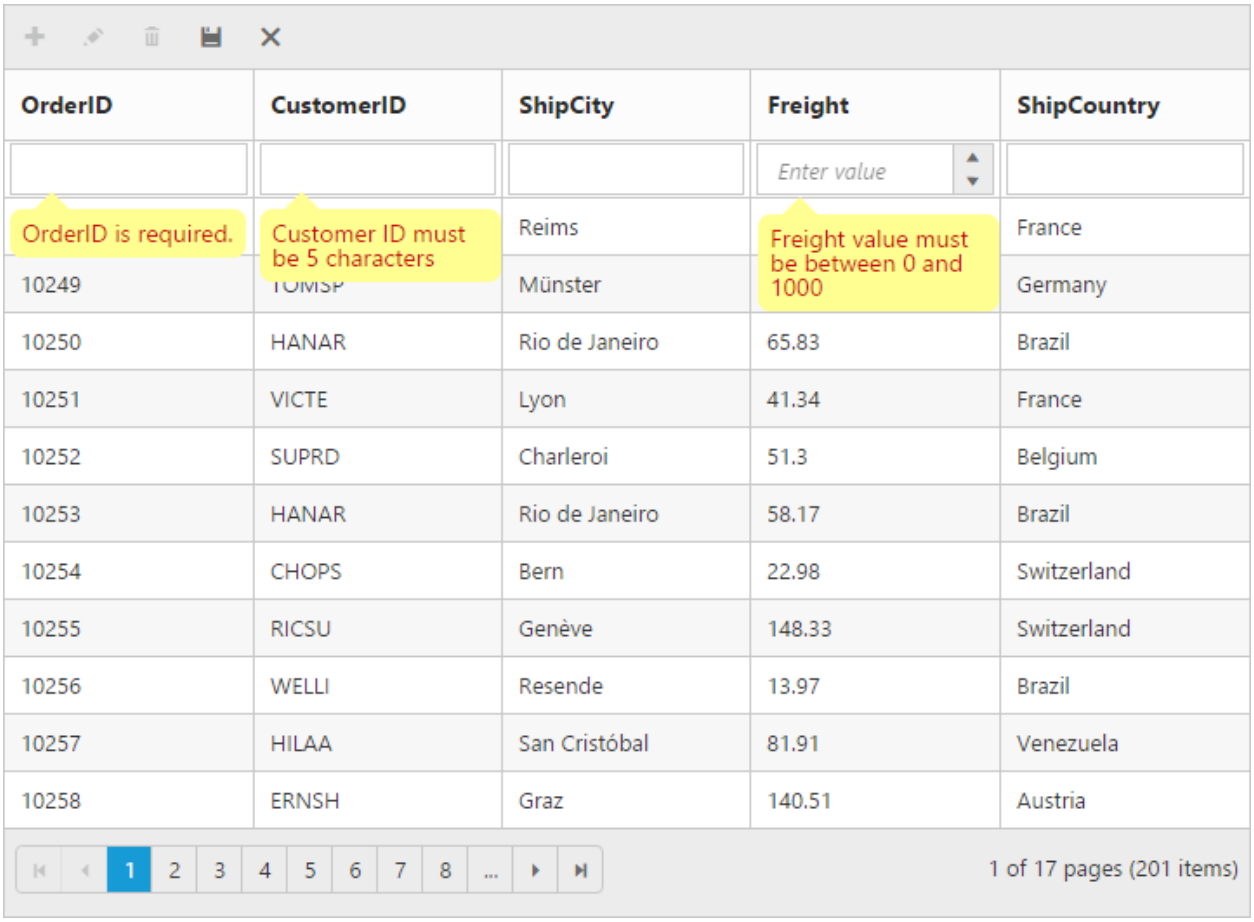
```
<script type="text/javascript">
$(function () {
$.validator.addMethod("customCompare", function (value, element, params) {
return element.value > params[0] && element.value < params[1];
}, "Freight value must be between 0 and 1000");
$.validator.addMethod("customRegex", function (value, element, params) {
if (element.value.length == params)
return true;
return false;
}, "Customer ID must be 5 characters");
});
</script>
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Persisting data in Server

Edited data can be persisted in database using RESTful web services.

All the CRUD operations in grid are done through DataManager. DataManager have an option to bind all the CRUD related data in server side. Please refer the ['link'] to know about the DataManager.

For you information ODataAdaptor persist data in server as per OData protocol.

In the below section, we have explained how to get the edited data details at the server side using URLAdaptor.

Copyright © 2001 - 2020 Syncfusion Inc.

URL Adaptor

You can use the UrlAdaptor of DataManger when binding datasource from remote data. At initial load of Grid, using URL property of DataManager, data are fetched from remote data and bound to Grid. You can map CRUD operation in Grid to Server-Side Controller action using the properties InsertURL, RemoveURL, UpdateURL, CrudURL and BatchURL.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("Home/DataSource").UpdateURL("Home/Update").InsertURL("Home/I
nsert").RemoveURL("Home/Delete").Adaptor(AdaptorType.UrlAdaptor))
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

Also when you use UrlAdaptor, you need to return the data as JSON and the JSON object must contain a property as result with dataSource as its value and one more property count with the dataSource total records count as its value.

The following code example describes the above behavior.

C#

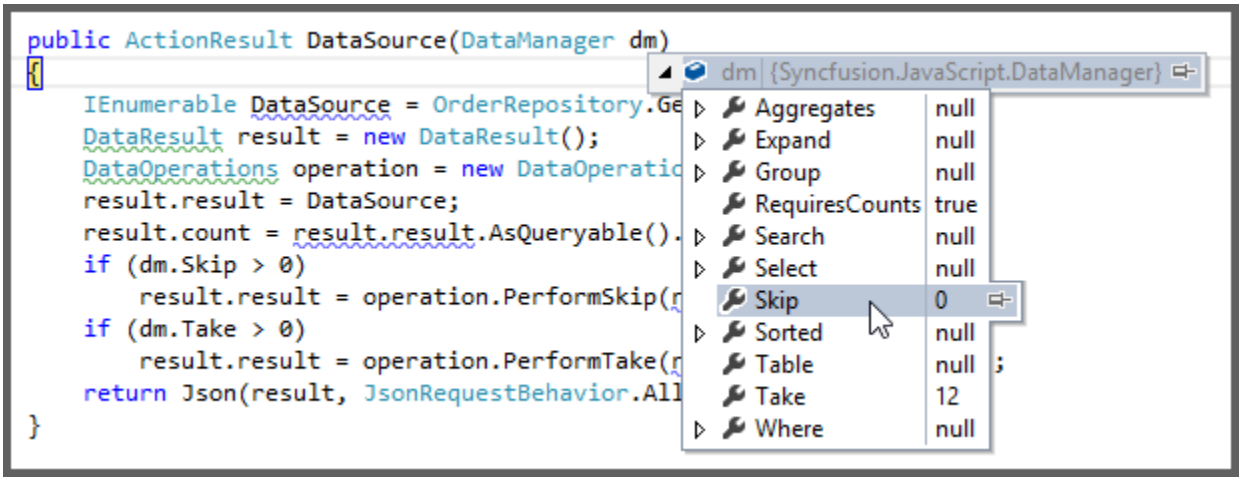
```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult DataSource(DataManager dataManager)
{
IEnumerable DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
DataOperations operation = new DataOperations();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
result.result = DataSource;
result.count = result.result.AsQueryable().Count();
if (dataManager.Skip > 0)
result.result = operation.PerformSkip(result.result, dataManager.Skip);
if (dataManager.Take > 0)
result.result = operation.PerformTake(result.result, dataManager.Take);
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable result { get; set; }
public int count { get; set; }
}
}
}
```

The grid actions (sorting, filtering, paging, searching, and aggregates) details are obtained in the 'DataManager' class. While initializing the grid, paging only enabled hence in the below screen shot paging details are bound to the DataManager class.

Please refer the below screen shot.



Also, using 'DataOperations' helper class you can perform grid action at server side. The in-built methods that we have provided in the DataOperations class are listed below.

1. PerformSorting
2. PerformFiltering
3. PerformSearching
4. PerformSkip
5. PerformTake
6. PerformWhereFilter
7. PerformSelect
8. Execute

RemoteSave Adaptor:

RemoteSaveAdaptor is used for binding local data and performs all data operations in client-side. It interacts with server-side only for CRUD operations to pass the modified records.

Copyright © 2001 - 2020 Syncfusion Inc.

Refer to the following code example

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource(dataSource =>
dataSource.Json((IEnumerable<object>)ViewBag.datasource).UpdateURL("/Home/Up
date")
.InsertURL("/Home/Insert").RemoveURL("/Home/Remove").Adaptor(AdaptorType.Rem
oteSaveAdaptor))
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

Accessing CRUD action request details in server side:

The 'Server-Side' function must be declared with the following parameter name for each editing functionality.

Parameters Table

Action	Parameter Name	Example
Update,Insert	value	public ActionResult Update(EditableOrder value){ }
	public ActionResult Insert(EditableOrder value){ }	
Remove	key	public ActionResult Remove(int key){ }
Batch Add	added	public ActionResult BatchUpdate(string action, List <EditableOrder> added, List <EditableOrder> changed, List <EditableOrder> deleted, int? key){ }
Batch Update	changed	
Batch Delete	deleted	

Copyright © 2001 - 2020 Syncfusion Inc.

Crud Update,Crud Insert	value, action	public ActionResult CrudUrl(EditableOrder value, string action){ }
Crud Remove	action, key, keyColumn	public ActionResult CrudUrl(string action, int? key, string keyColumn){ }
Crud Remove - Multi Delete	action, key, deleted	public ActionResult CrudUrl(string action, string key, List <EditableOrder> deleted){ }

Insert Record:

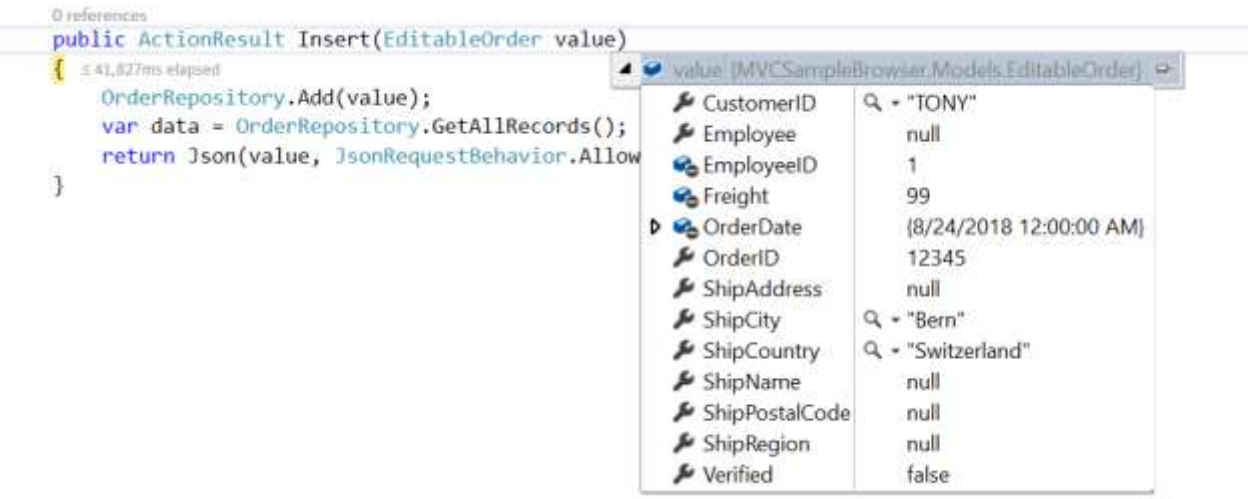
Using InsertURL property, you can specify the controller action mapping URL to perform insert operation at server side.

The following code example describes the above behavior.

C#

```
public ActionResult Insert(EditableOrder value)
{
OrderRepository.Add(value);
var data = OrderRepository.GetAllRecords();
return Json(value, JsonRequestBehavior.AllowGet);
}
```

The newly added record details are bound to the 'value' parameter. Please refer the below image.



Update Record:

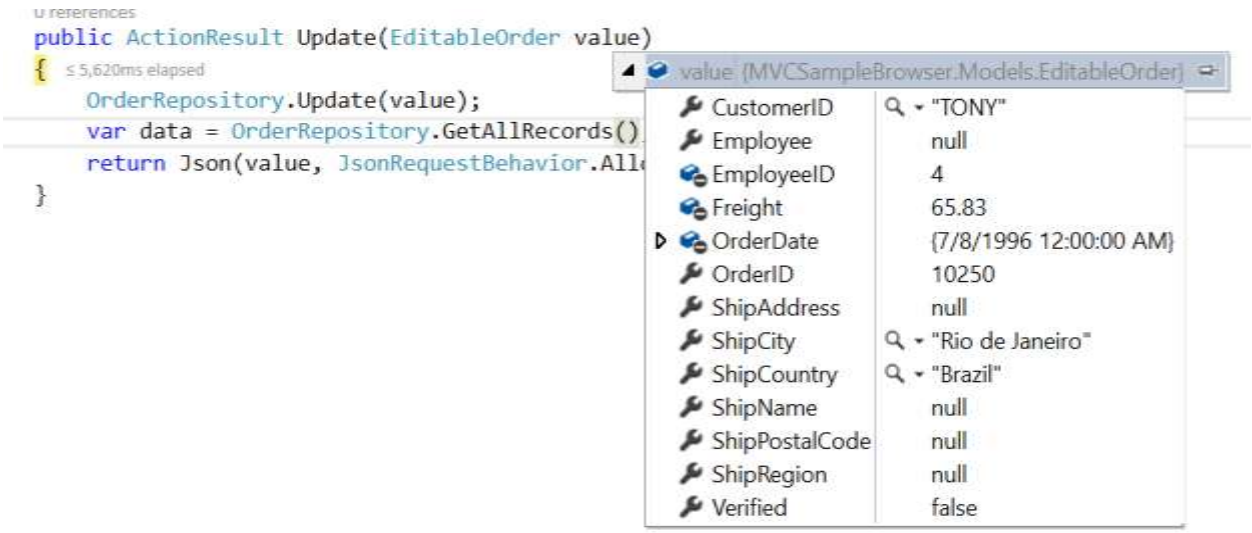
Using UpdateURL property, you can specify the controller action mapping URL to perform save/update operation at server side.

The following code example describes the above behavior.

C#

```
public ActionResult Update(EditableOrder value)
{
OrderRepository.Update(value);
var data = OrderRepository.GetAllRecords();
return Json(value, JsonRequestBehavior.AllowGet);
}
```

The updated record details are bound to the 'value' parameter. Please refer the below image.



Delete Record:

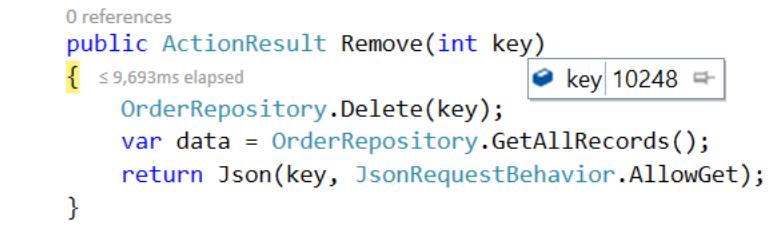
Using RemoveURL property, you can specify the controller action mapping URL to perform delete operation at server side.

The following code example describes the above behavior.

C#

```
public ActionResult Remove(int key)
{
OrderRepository.Delete(key);
var data = OrderRepository.GetAllRecords();
return Json(key, JsonRequestBehavior.AllowGet);
}
```

The deleted record primary key value is bound to the 'key' parameter. Please refer the below image.



Copyright © 2001 - 2020 Syncfusion Inc.

CRUD URL:

Instead of specifying separate controller action method for CRUD (insert, update and delete)operation, using CrudURL property you can specify the controller action mapping URL to perform all the CRUD operation at server side using single method.

The action parameter of CrudURL is used to get the corresponding CRUD action.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource(datasource =>
datasource.URL("Home/DataSource").CrudURL("Home/CrudUpdate").Adaptor(Adaptor
Type.UrlAdaptor))
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

C#

```
public ActionResult CrudUpdate(EditableOrder value, string action, int key)
{
if (action == "update")
OrderRepository.Update(value);
else if (action == "insert")
OrderRepository.Add(value);
else if (action == "remove")
OrderRepository.Delete(key);
return Json(value, JsonRequestBehavior.AllowGet);
}
```

Please refer the below image to know about the action parameter

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult CrudUrl(EditableOrder value, int? key, string action)
{
    if (action == "update")
        OrderRepository.Update(value);
    else if (action == "insert")
        OrderRepository.Add(value);
    else if (action == "remove")
        OrderRepository.Delete(key);
    return Json(value, JsonRequestBehavior.AllowGet);
}
```

Note: If you specify InsertURL along with CrudURL then while adding InsertURL only called.

Batch URL:

The BatchURL property supports only for batch editing mode. You can specify the controller action mapping URL to perform Batch operation at server side.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.dataSource).BatchURL("Home/Batc
hUpdate").Adaptor(AdaptorType.UrlAdaptor))
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

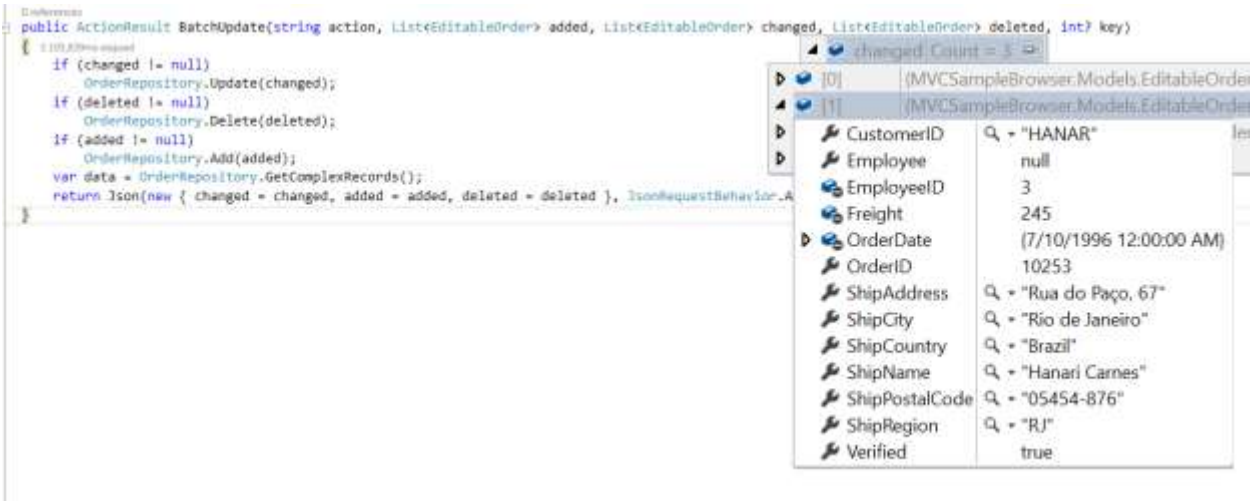
C#

```
public ActionResult BatchUpdate(string action, List<EditableOrder> added,
List<EditableOrder> changed, List<EditableOrder> deleted, int? key)
{
if (changed != null)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
OrderRepository.Update(changed);
if (deleted != null)
OrderRepository.Delete(deleted);
if (added != null)
OrderRepository.Add(added);
var data = OrderRepository.GetComplexRecords();
return Json(new { changed = changed, added = added, deleted = deleted },
JsonRequestBehavior.AllowGet);
}
```

Please refer the below image for more information about batch parameters



Performing CRUD Operation on ODataV4 Adaptor

ODataV4 Adaptor that is extended from ODataAdaptor, is used for consuming data from ODataV4 Service

To consume ODataV4 service, set the service link to the Url property of Grid DataSource and you can set Adaptor type as AdaptorType.ODataV4Adaptor to the Adaptor Property of Grid DataManager. We can bind this DataManager as dataSource for the Grid.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("Grid")
.Datasource(datasource => {
datasource.URL("/odata/Orders").Adaptor(AdaptorType.ODataV4Adaptor); })
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().EditMode(EditMode.Normal);
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).IsIdentity(true).Width(50).Add();
col.Field("EmployeeID").HeaderText("Employee ID").Width(50).Add();
col.Field("ShipCity").HeaderText("ShipCity").Width(50).Add();
})
)
```

To perform the CRUD Operation in ODataV4 we need not explicitly mention any URL for Insert, Update or Delete operations in grid instead we can retrieve it using the PUT, POST and DELETE semantics of the OData which will be handled on the URL property already specified.

The following code example describes the above behavior.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : ODataController
{
// GET: Orders
NORTHWNDEntities db = new NORTHWNDEntities();
[EnableQuery]
public IQueryable<Order> Get()
{
return db.Orders;
}
}
}
```

The Get method returns the entire Order collection. This Get Method returns the data to the Grid from the database.

Insert Record:

To add a new data to the database, add the following method to controller page.

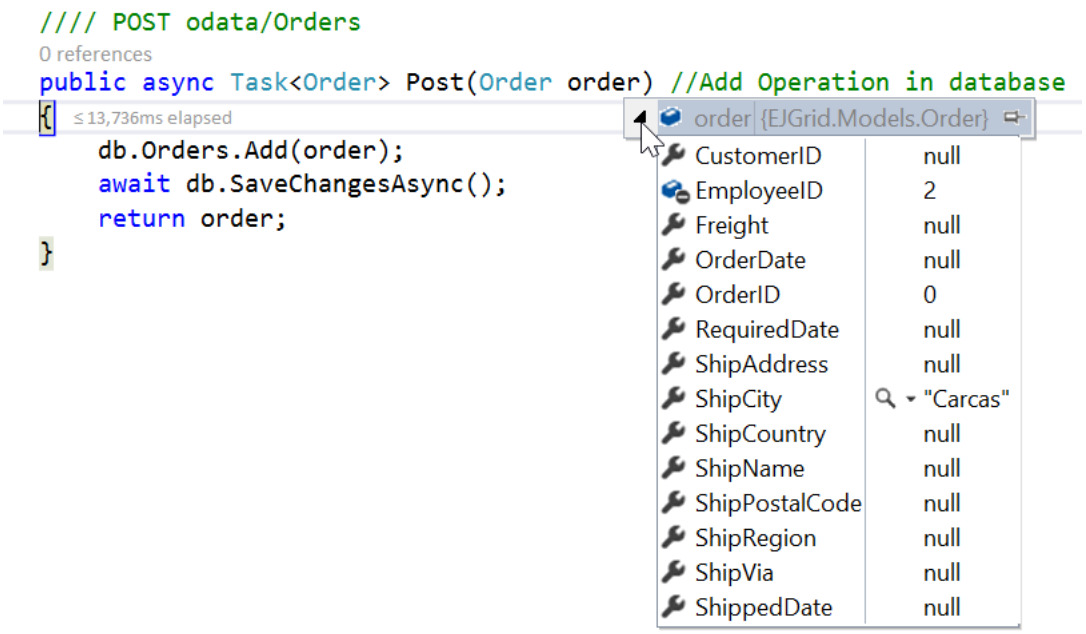
The following code example describes the above behavior.

C#

```
// POST odata/Orders
public async Task<Order> Post(Order order)
{
db.Orders.Add(order);
await db.SaveChangesAsync();
return order;
}
```

Please refer the below image.

Copyright © 2001 - 2020 Syncfusion Inc.



Here the newly added record details are bound to the 'order' parameter.

Update Record:

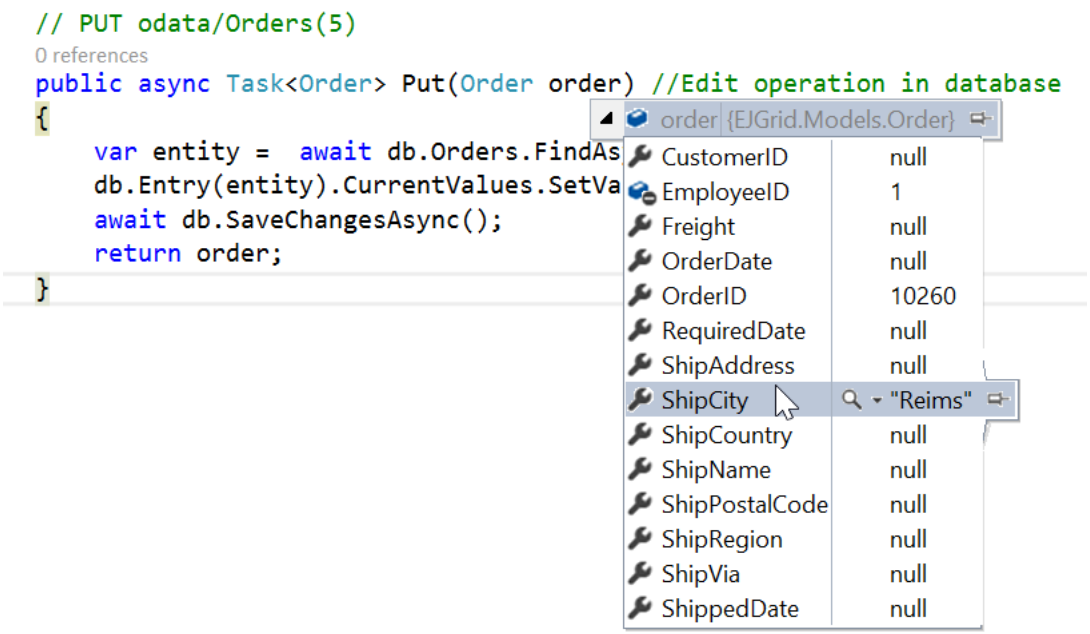
To update an existing data from the database, add the following method to Controller page.

The following code example describes the above behavior.

C#

```
// PUT odata/Orders(5)
public async Task<Order> Put(Order order)
{
var entity =  await db.Orders.FindAsync(order.OrderID);
db.Entry(entity).CurrentValues.SetValues(order);
await db.SaveChangesAsync();
return order;
}
```

Please refer the below image.



Here the modified row data will be obtained in the order parameter.

Delete Record:

To delete a data from the database, add the following method to controller.

The following code example describes the above behavior.

C#

```
// DELETE odata/Orders(5)
public async Task<IHttpActionResult> Delete([FromODataUri] int key)
{
var data = await db.Orders.FindAsync(key);
if (data == null)
{
return NotFound();
}
db.Orders.Remove(data);
await db.SaveChangesAsync();
return StatusCode(HttpStatusCode.NoContent);
}
```

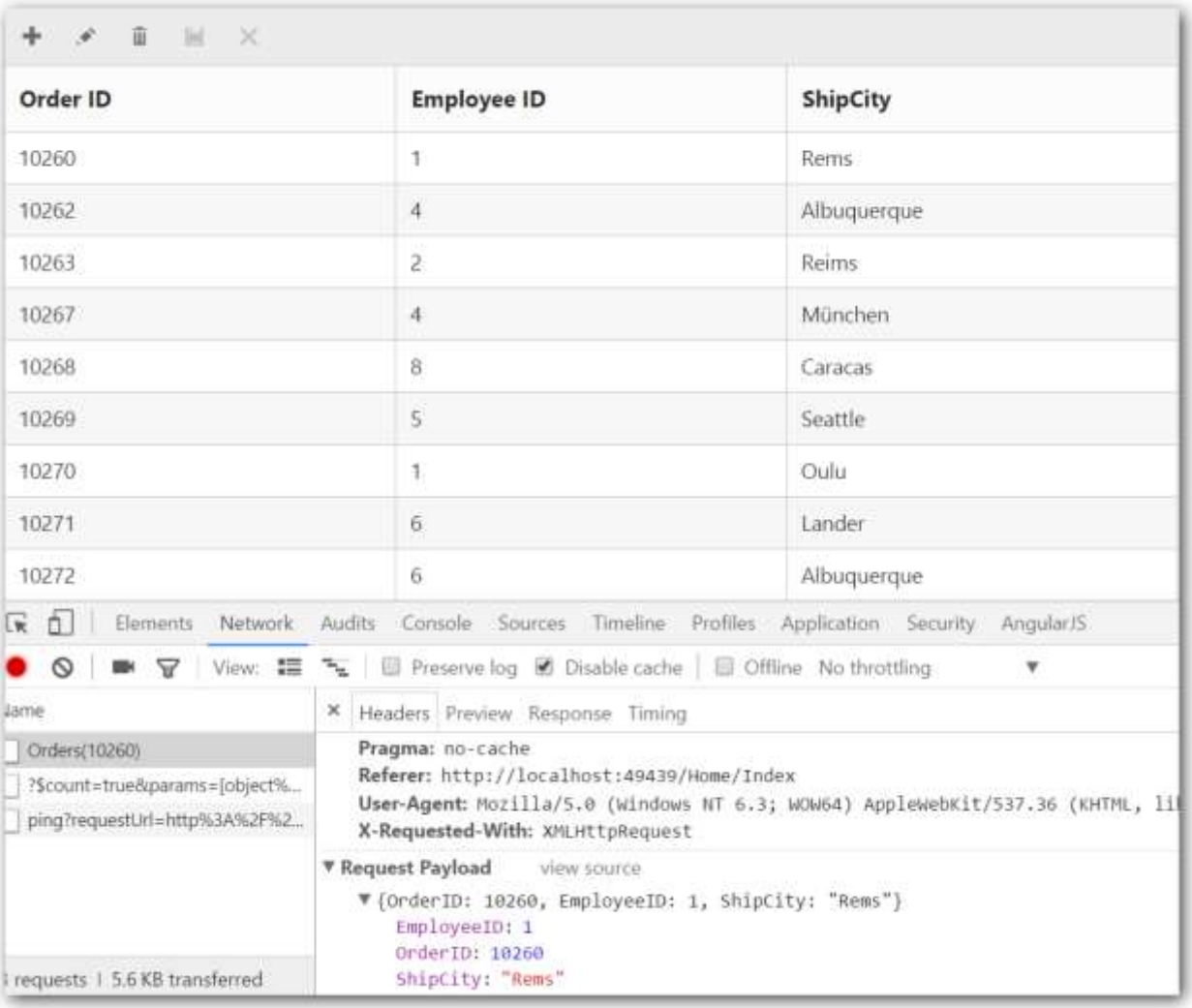
Please refer the below image.

Copyright © 2001 - 2020 Syncfusion Inc.

```
// DELETE odata/Orders(5)
 0 references
public async Task<IHttpActionResult> Delete([FromODataUri] int key)
{  ≤5,179ms elapsed                                          key 10260
    var od = await db.Orders.FindAsync(key);
     if (od == null)
     {
         return NotFound();
     }
     db.Orders.Remove(od);
     await db.SaveChangesAsync();
     return StatusCode(HttpStatusCode.NoContent);
}
```

The deleted record primary key value is bound to the ‘key’ parameter.

Please refer the below image.



Copyright © 2001 - 2020 Syncfusion Inc.

Adding New Row Position

To add new row in the top or bottom position of grid content, set RowPosition property of EditSettings depending on the requirement.

The following code example describes the above behavior.

RAZOR

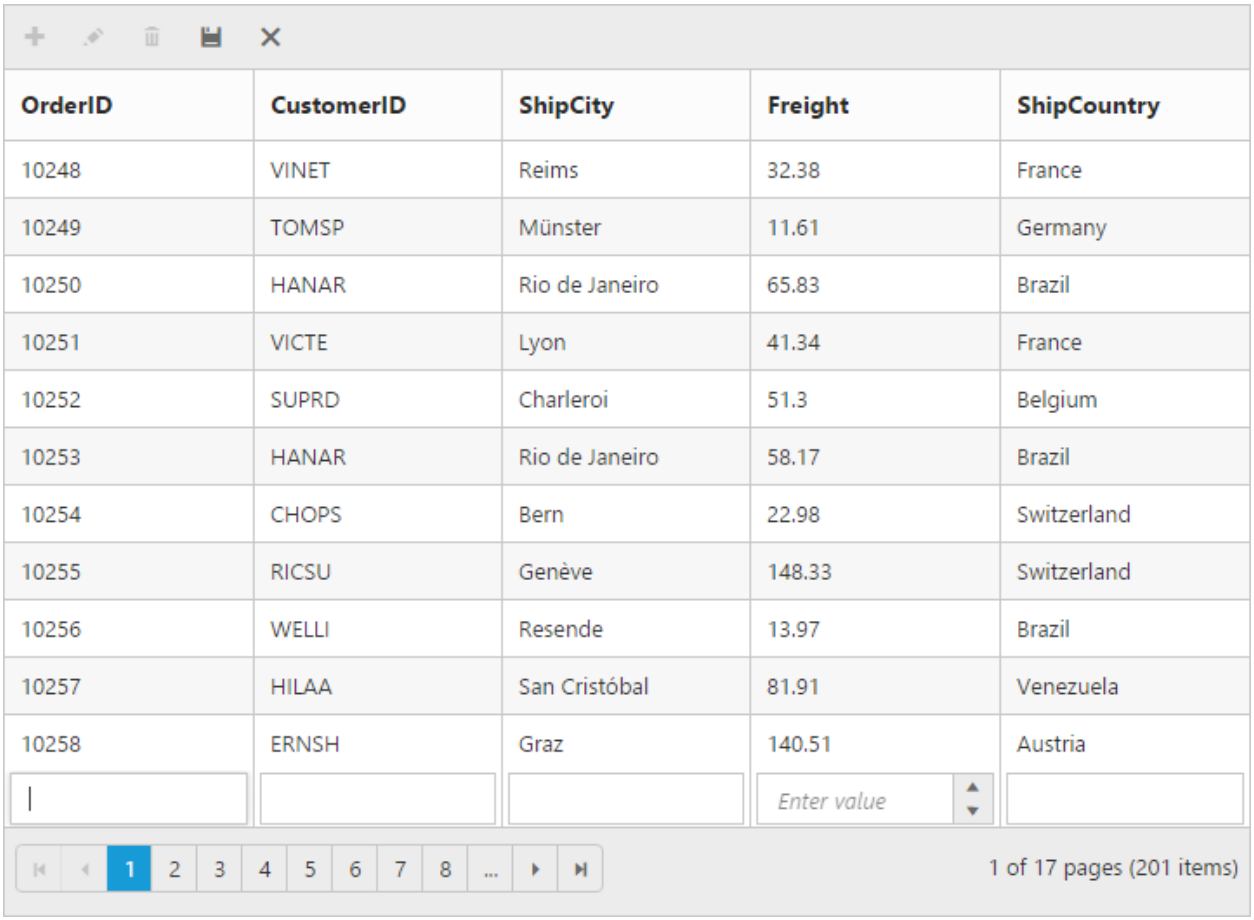
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().RowPosition(RowPosition.Bo
ttom) })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Render with blank row for easy add new

The blank add new row is displayed in the grid content during grid initialization itself to add a new record easily. To enable show add new row by default, set ShowAddNewRow property of EditSettings as true.

The blank add new row is displayed either in the top or bottom of the corresponding page, its position is based on the RowPosition property of EditSettings.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().ShowAddNewRow(); })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
```

Copyright © 2001 - 2020 Syncfusion Inc.

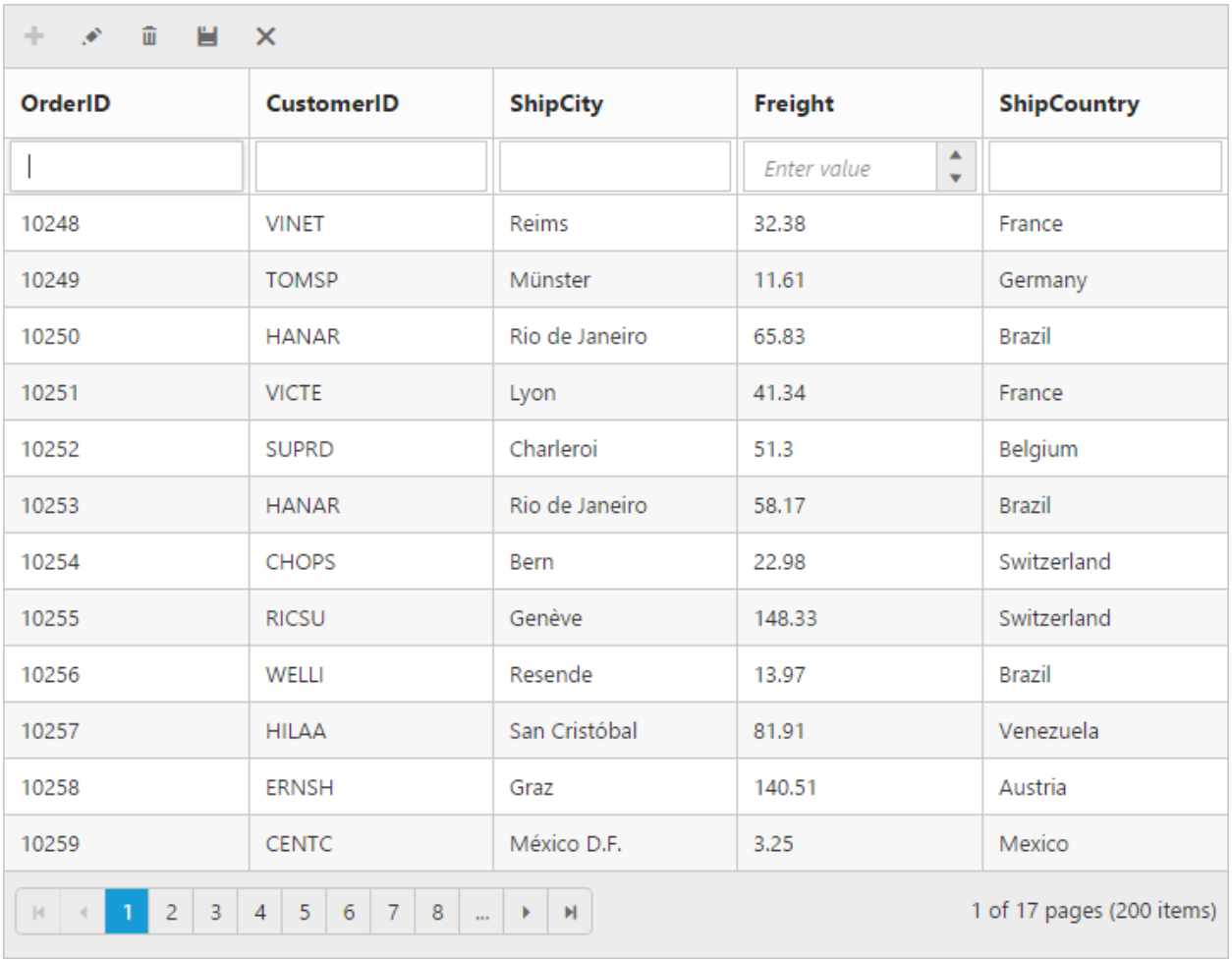
```
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: 1. If it is remote, then the newly added record is placed based on the index from current view data.

2. If it is local, then the newly added record is added at the top of the page even if the added new `RowPosition` is mentioned as "Bottom".

Default column values on add new

While adding new record in grid, there is an option to set the default value for the columns. Using `DefaultValue` property of `Columns` you can set the default values for that particular column while editing or adding a new row.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.EditSettings(edit => {
edit.AllowAdding().AllowDeleting().AllowEditing().ShowAddNewRow(); })
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

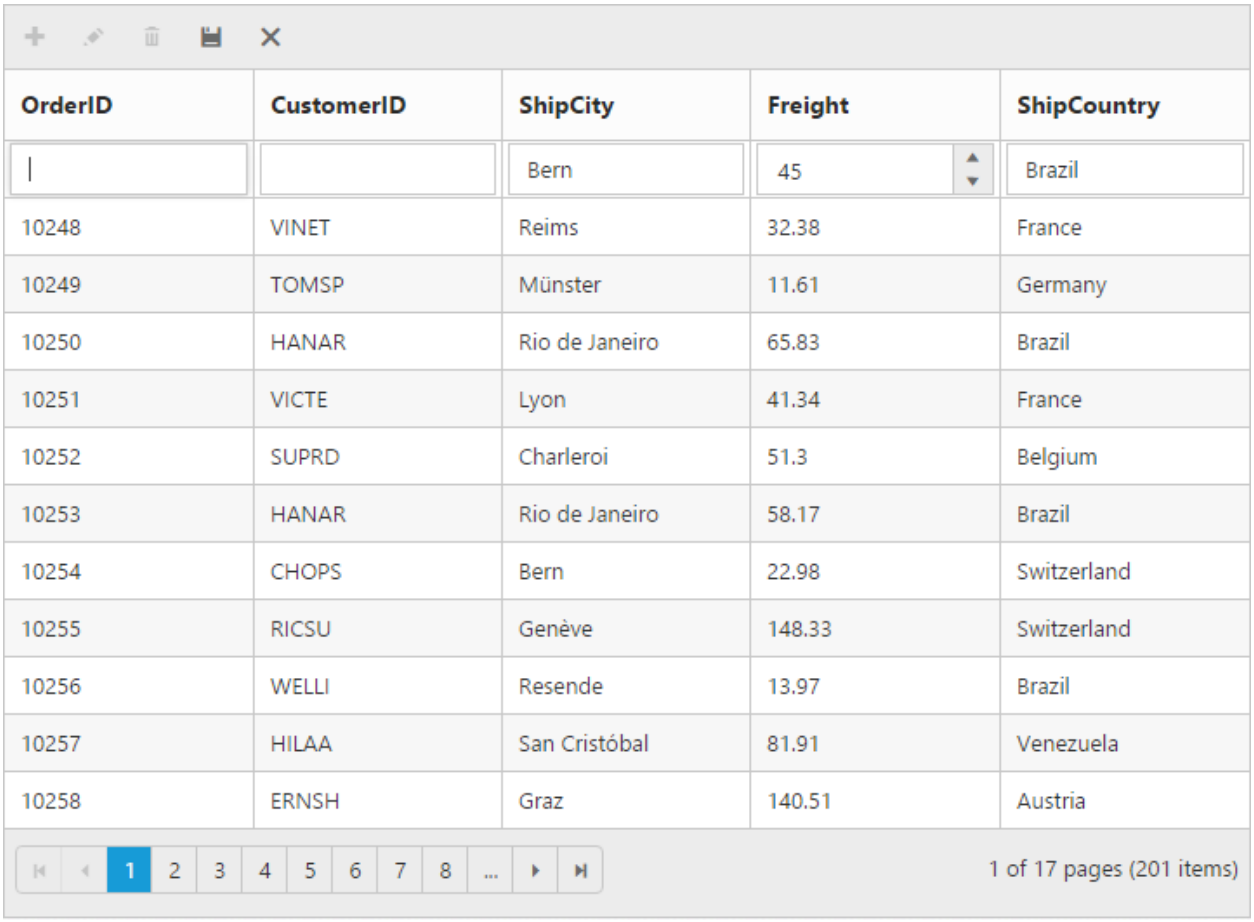
```
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCity").HeaderText("Ship City").DefaultValue("Bern").Add();
col.Field("Freight").HeaderText("Freight").EditType(EditingType.NumericEdit)
.DefaultValue(45).Add();
col.Field("ShipCountry").HeaderText("Ship
Country").DefaultValue("Brazil").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Grouping

The Grid control has options to group the records based on the required column. When grouping is applied, grouped records are organized into a hierarchical structure to facilitate easier expand and collapse of records. To enable grouping, set AllowGrouping property as true.

Columns can be grouped by simply dragging the column header and drop on the group drop area or simply click the group button which is displayed in the column. By default, sorting is done while grouping the column.

The following code example describes the above behavior.

RAZOR

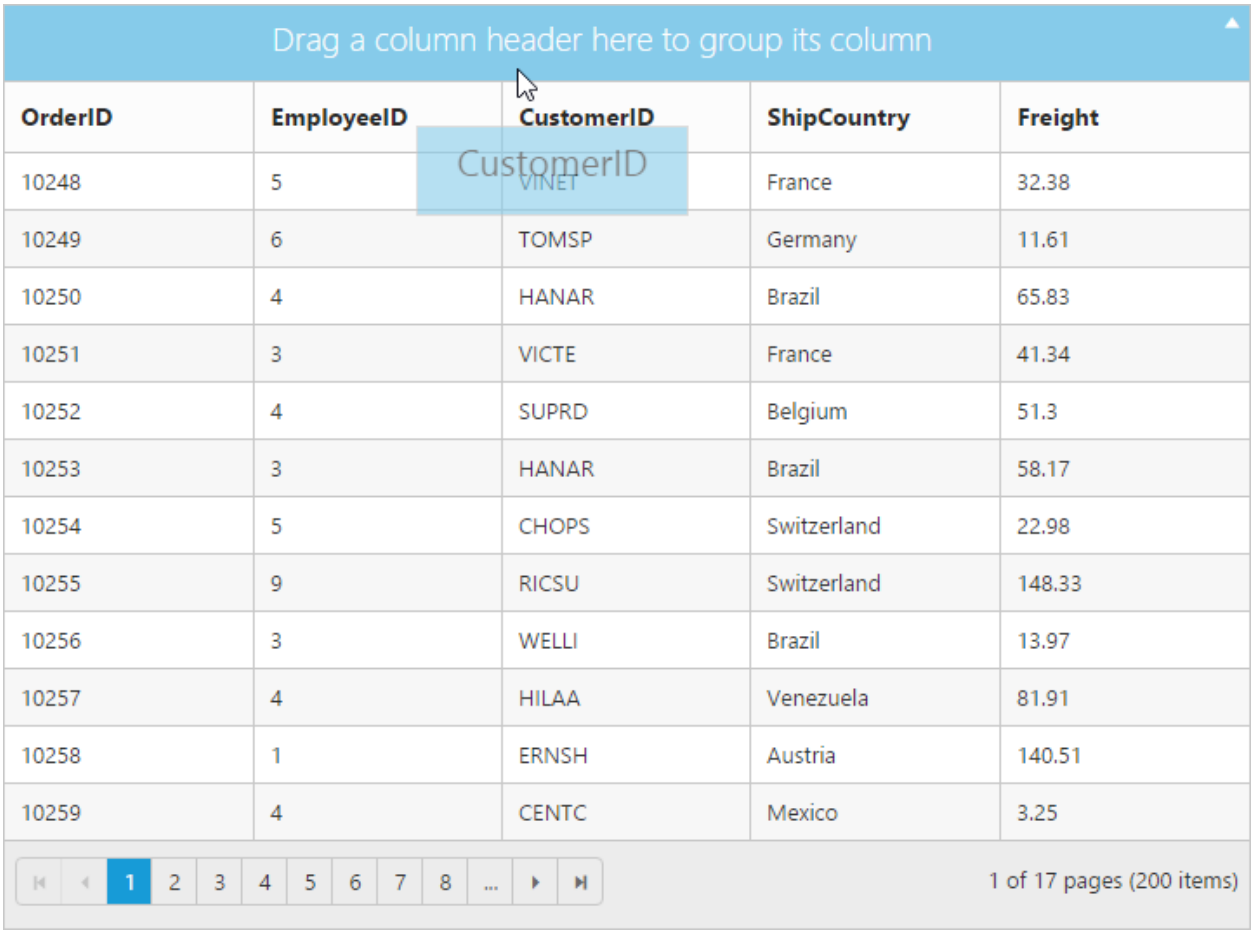
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Initial Grouping

While initializing the grid itself, there is an option to group the column and display it in a hierarchical structure. To enable initial grouping, set array of column's `Field` name to be grouped in `GroupedColumns` property of `GroupSettings`.

The following code example describes the above behavior.

RAZOR

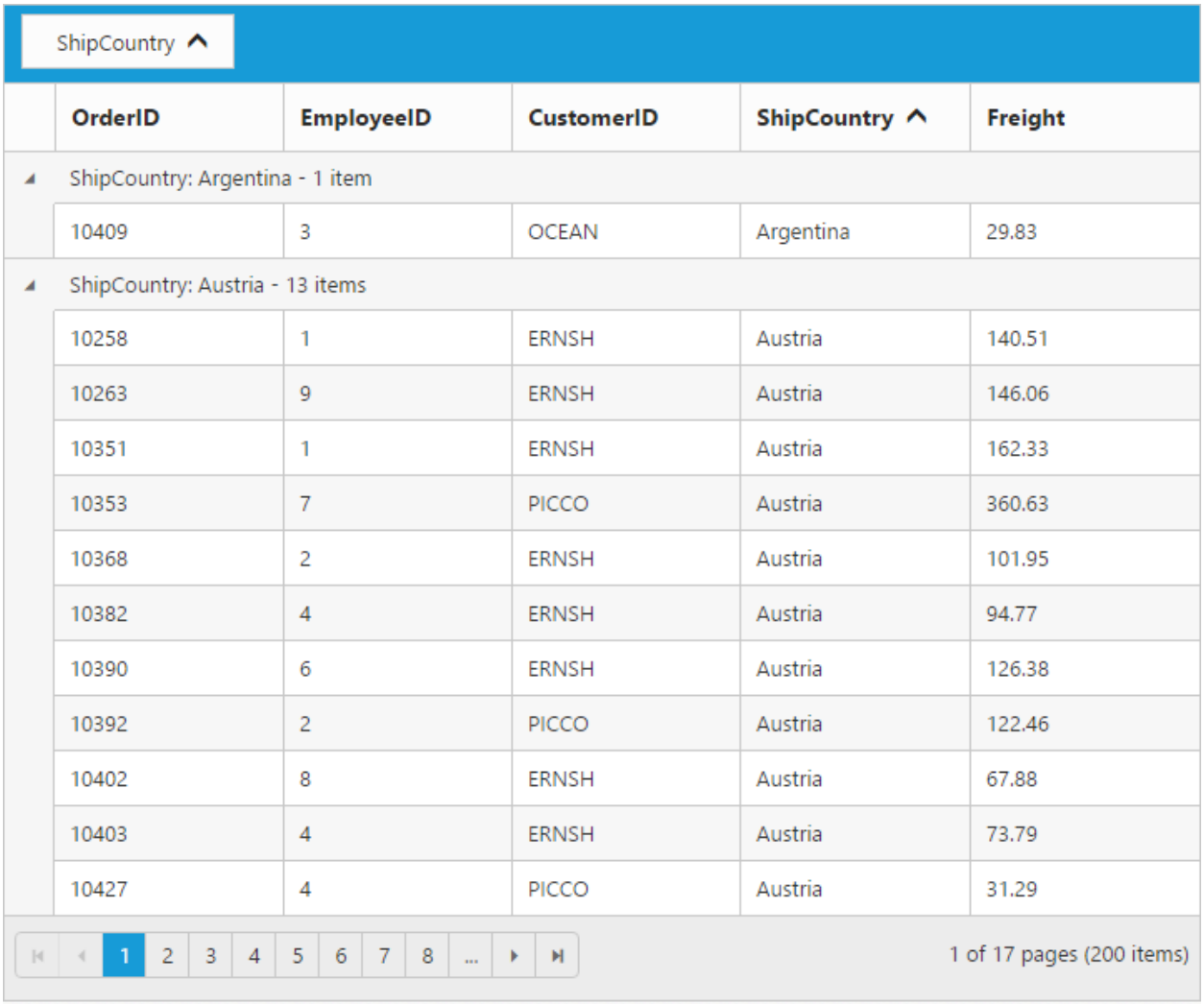
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.GroupedColumns(col => {
col.Add("ShipCountry"); }); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Multi-Column Grouping

Group multiple columns by simply drag and drop the columns one by one from column header into group drop area.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.GroupedColumns(col => {
col.Add("ShipCountry"); col.Add("CustomerID"); }); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

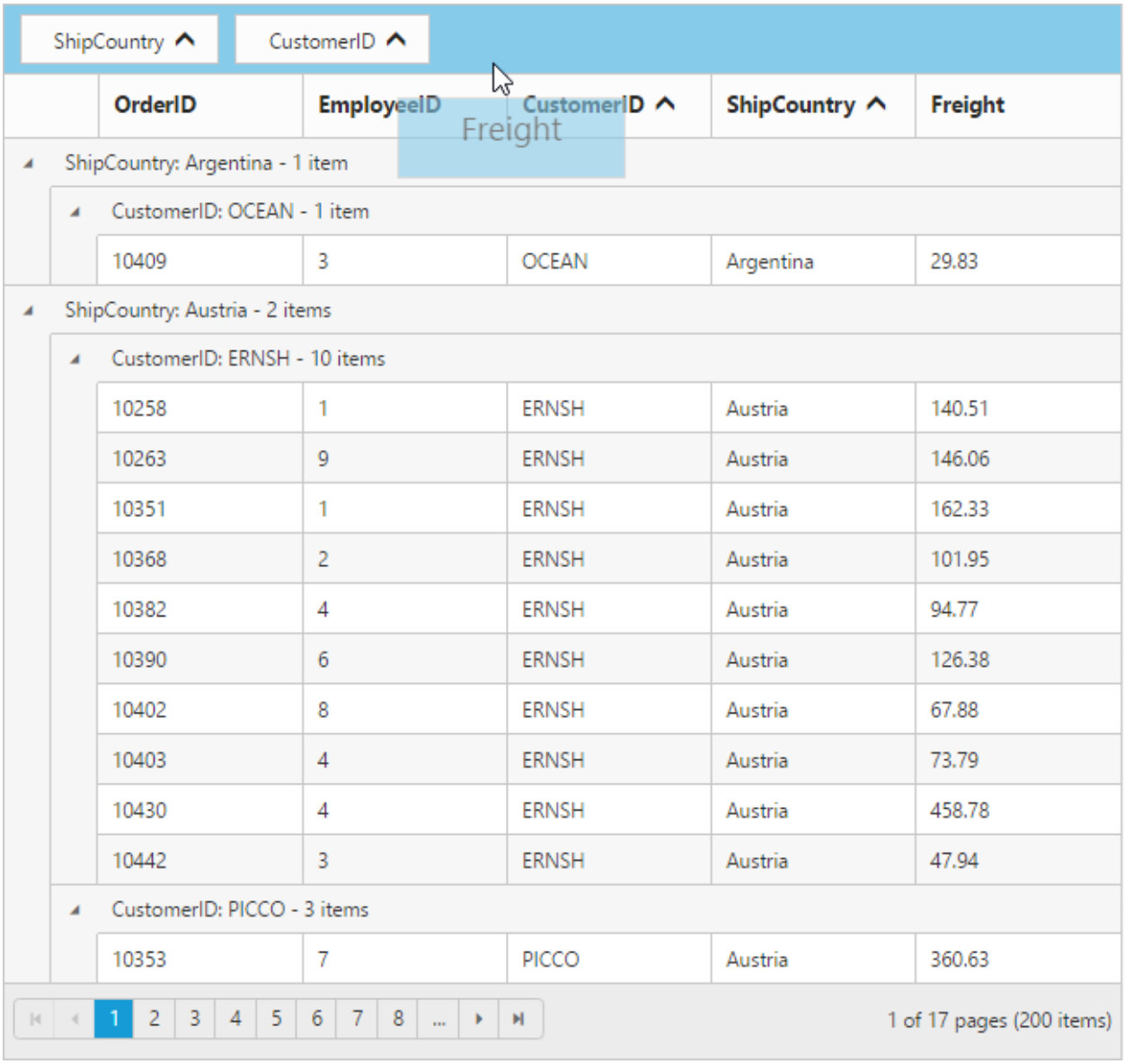
Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Group Buttons

To do grouping easily without doing drag and drop column header by setting ShowToggleButton property of GroupSettings as true.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.ShowToggleButton(true); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

OrderID	EmployeeID	CustomerID	ShipCountry	Freight
10248	5	VINET	France	32.38
10249	6	TOMSP	Germany	11.61
10250	4	HANAR	Brazil	65.83
10251	3	VICTE	France	41.34
10252	4	SUPRD	Belgium	51.3
10253	3	HANAR	Brazil	58.17
10254	5	CHOPS	Switzerland	22.98
10255	9	RICSU	Switzerland	148.33
10256	3	WELLI	Brazil	13.97
10257	4	HILAA	Venezuela	81.91
10258	1	ERNSH	Austria	140.51
10259	4	CENTC	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Hide Ungroup Button

Hide ungroup button from grouped columns which is in the group drop area by setting the ShowUngroupButton property of GroupSettings as false.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

RAZOR

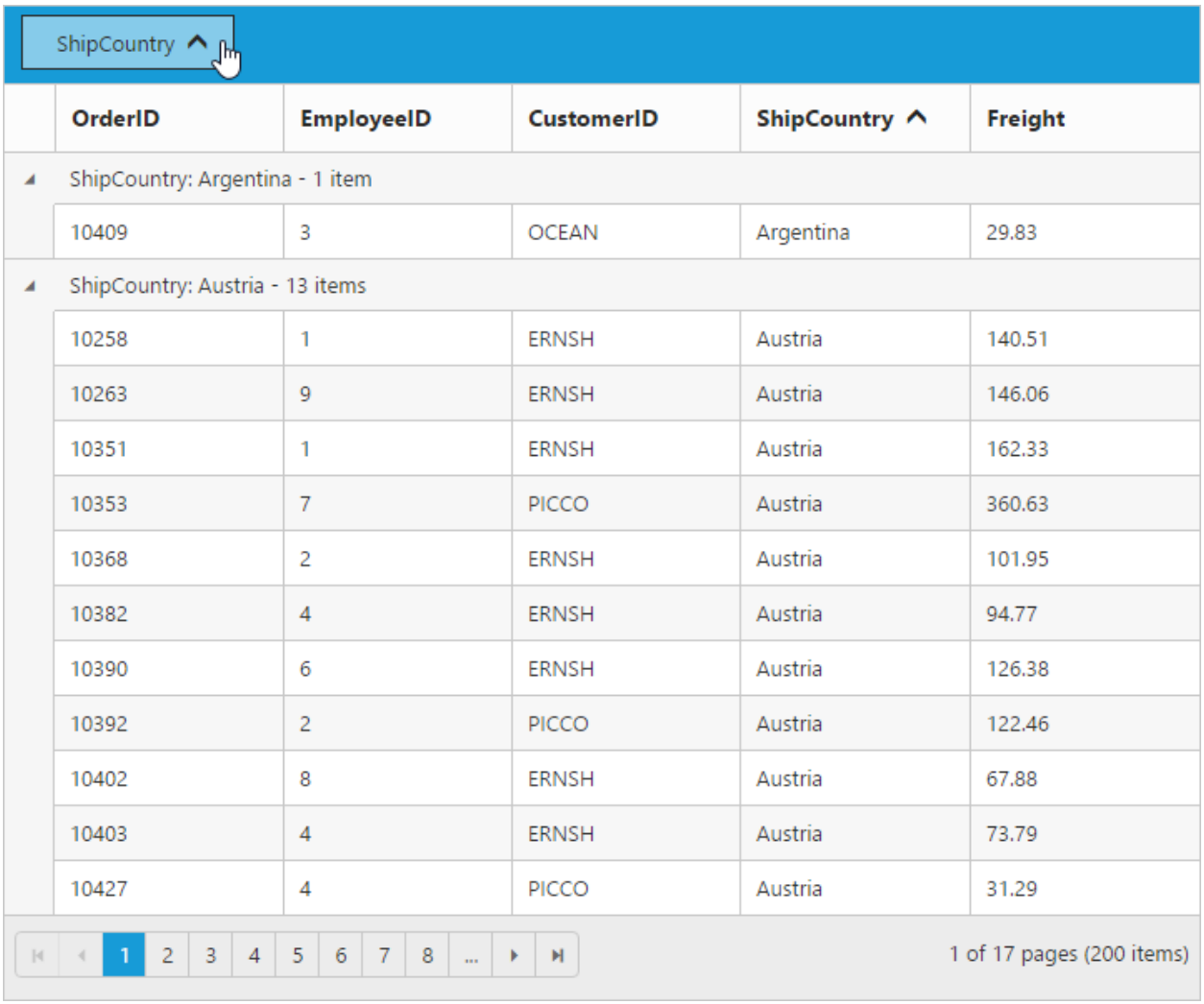
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.ShowUngroupButton(false); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Hide Grouped Column

While grouping a particular column, there is an option to hide the grouped columns from grid. To enable hide grouped column option, set `ShowGroupedColumn` property of `GroupSettings` as `false`.

The following code example describes the above behavior.

RAZOR

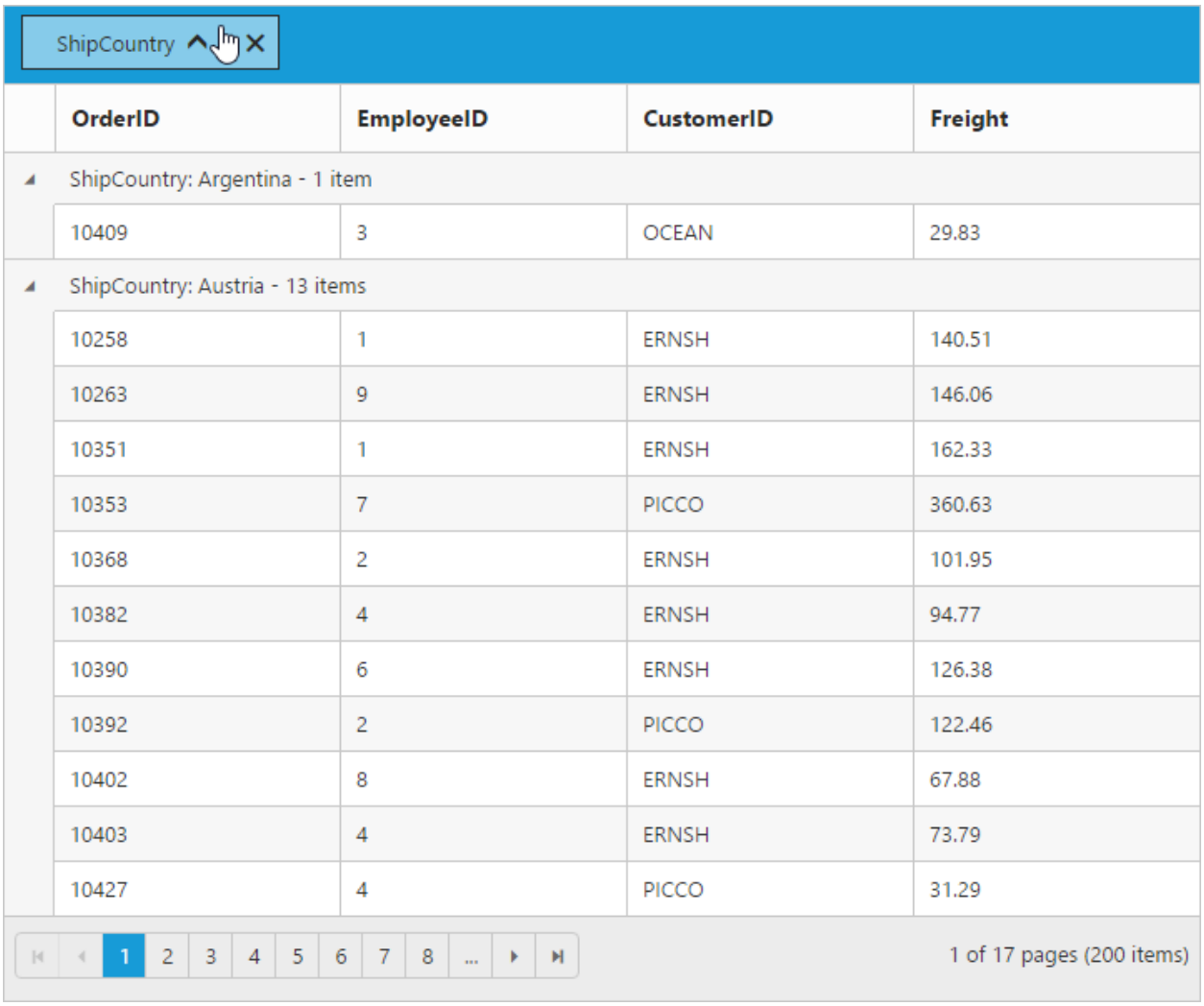
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.ShowGroupedColumn(false); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



AutoSize Drop Area

Drag any column header and move it to the group drop area, then its portion expands smoothly. Stop this animation by setting `EnableDropAreaAutoSizing` property of `GroupSettings` as `false`.

The following code example describes the above behavior.

RAZOR

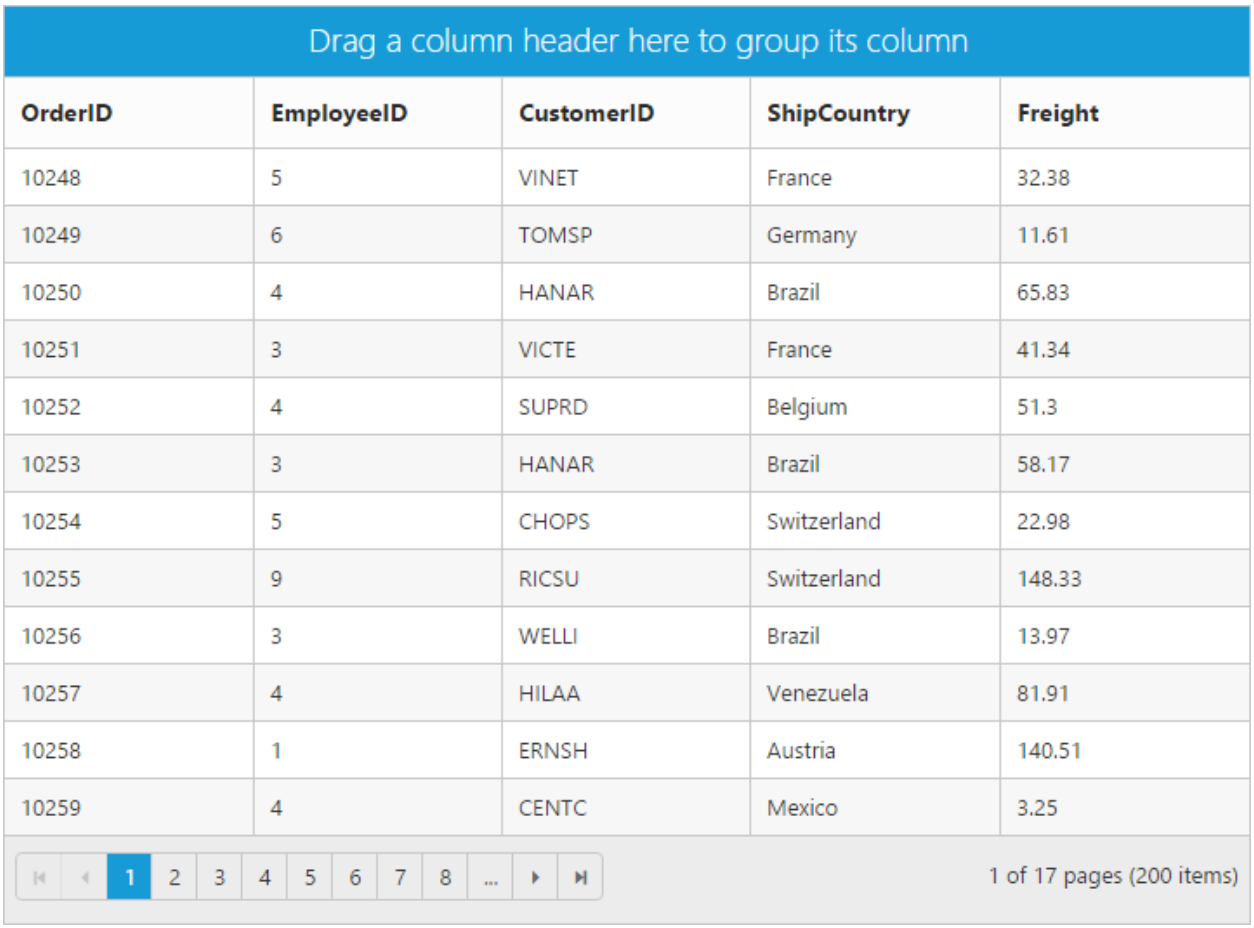
```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.EnableDropAreaAutoSizing(false); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Hide Drop Area

To avoid ungrouping or further grouping of a column after an initial column grouping by setting ShowDropArea property of GroupSettings as false.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.ShowDropArea(false).GroupedColumns(col => {
col.Add("ShipCountry"); }); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

	OrderID	EmployeeID	CustomerID	ShipCountry	Freight
▲	ShipCountry: Argentina - 1 item				
	10409	3	OCEAN	Argentina	29.83
▲	ShipCountry: Austria - 13 items				
	10258	1	ERNSH	Austria	140.51
	10263	9	ERNSH	Austria	146.06
	10351	1	ERNSH	Austria	162.33
	10353	7	PICCO	Austria	360.63
	10368	2	ERNSH	Austria	101.95
	10382	4	ERNSH	Austria	94.77
	10390	6	ERNSH	Austria	126.38
	10392	2	PICCO	Austria	122.46
	10402	8	ERNSH	Austria	67.88
	10403	4	ERNSH	Austria	73.79
	10427	4	PICCO	Austria	31.29

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Group Caption Format/Group Caption Template

Using CaptionFormat property of GroupSettings you can render any type of JsRender templates or customizing the group caption text.

You can use JsRender syntax in the template.For more information about JsRender syntax, please refer the link.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: 1. It's a standard way to enclose the template within the script tag with type as "text/x-jsrender".

2. Using locale property of GroupCaptionFormat, you can only customize the default group caption text.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.AllowGrouping()
.GroupSettings(group => { group.CaptionFormat("#template"); })
.ClientSideEvents(eve => { eve.ActionComplete("complete"); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("ShipCountry").HeaderText("Ship Country").Add();
col.Field("Freight").HeaderText("Freight").Add();
}))
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
}
}
```

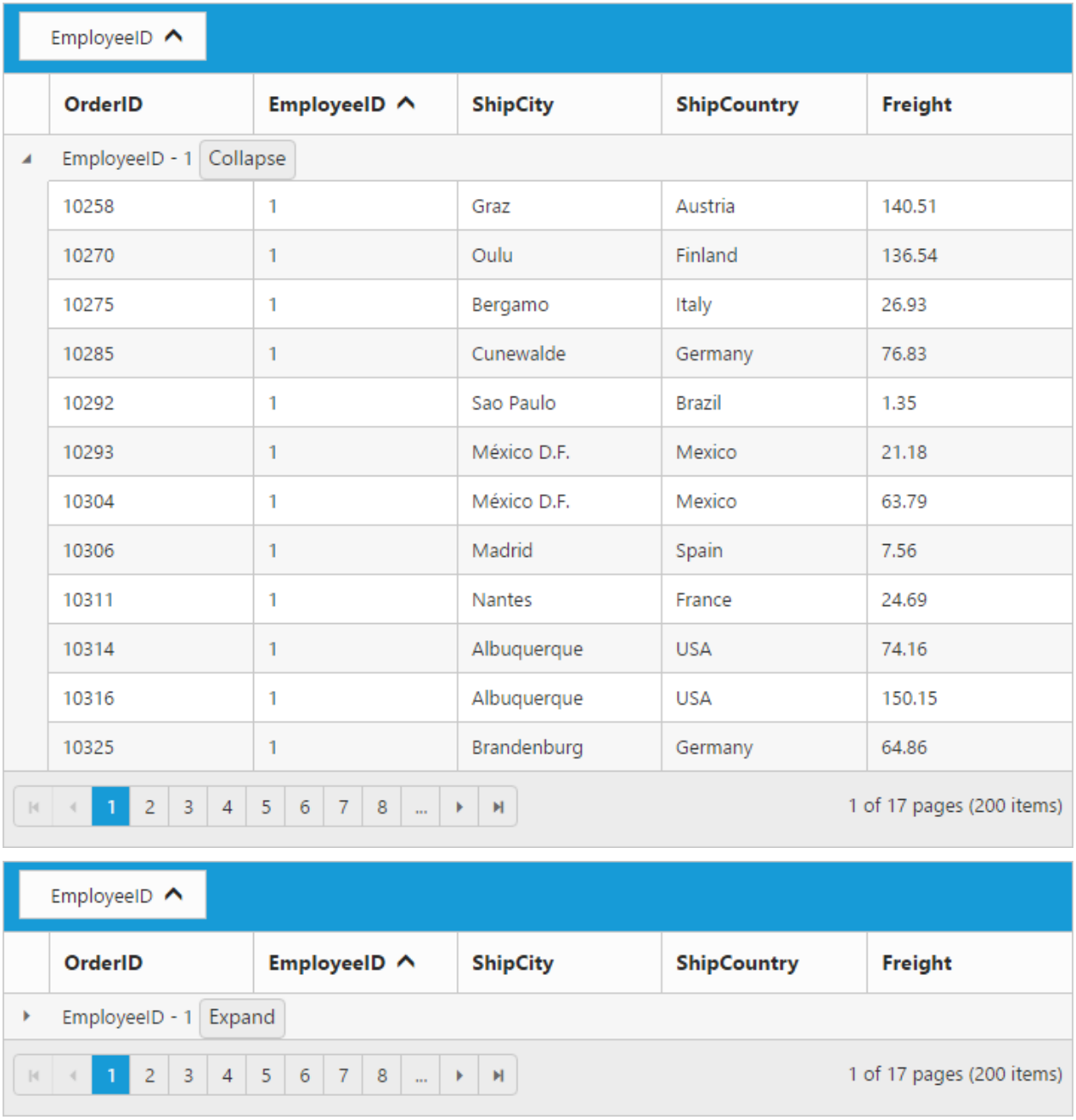
JS

```
<script id="template" type="text/x-jsrender">
"{{"{{"}}:field{{}}}}-"{{"{{"}}:key{{}}}}
<button id="btn" class="btn">Collapse</button>
</script>
<script>
function complete(args) {
$(".btn").ejButton({
click: "btnClick"
});
}
function btnClick(args) {
var gridObj = $("#FlatGrid").data("ejGrid");
gridObj.expandCollapse(this.element.parent().prev());
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#btn").ejButton("model.text", args.model.text == "Collapse" ? "Expand" :
"Collapse");
}
</script>
```

The following output is displayed as a result of the above code example.



Handling grouped records count in server-side

When binding remote data to grid with on-demand data loading, only current page data knowledge is available to grid and so grouped records count would be shown based on current Page only.

Copyright © 2001 - 2020 Syncfusion Inc.

This can be rectified when binding data to grid using UrlAdaptor of DataManager. The grouped column values should be passed into the groupDs property of return object from server-side along with datasource and count.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grouping")
.Datasource(datasource =>
datasource.URL("/Grid/UrlDataSource").Adaptor("UrlAdaptor"))
.AllowSorting()
.AllowGrouping()
.GroupSettings(group => { group.GroupedColumns(col => {
col.Add("EmployeeID"); }); })
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("OrderDate").HeaderText("Order
Date").TextAlign(TextAlign.Right).Width(80).Format("{0:MM/dd/yyyy}").Add();
})
)
```

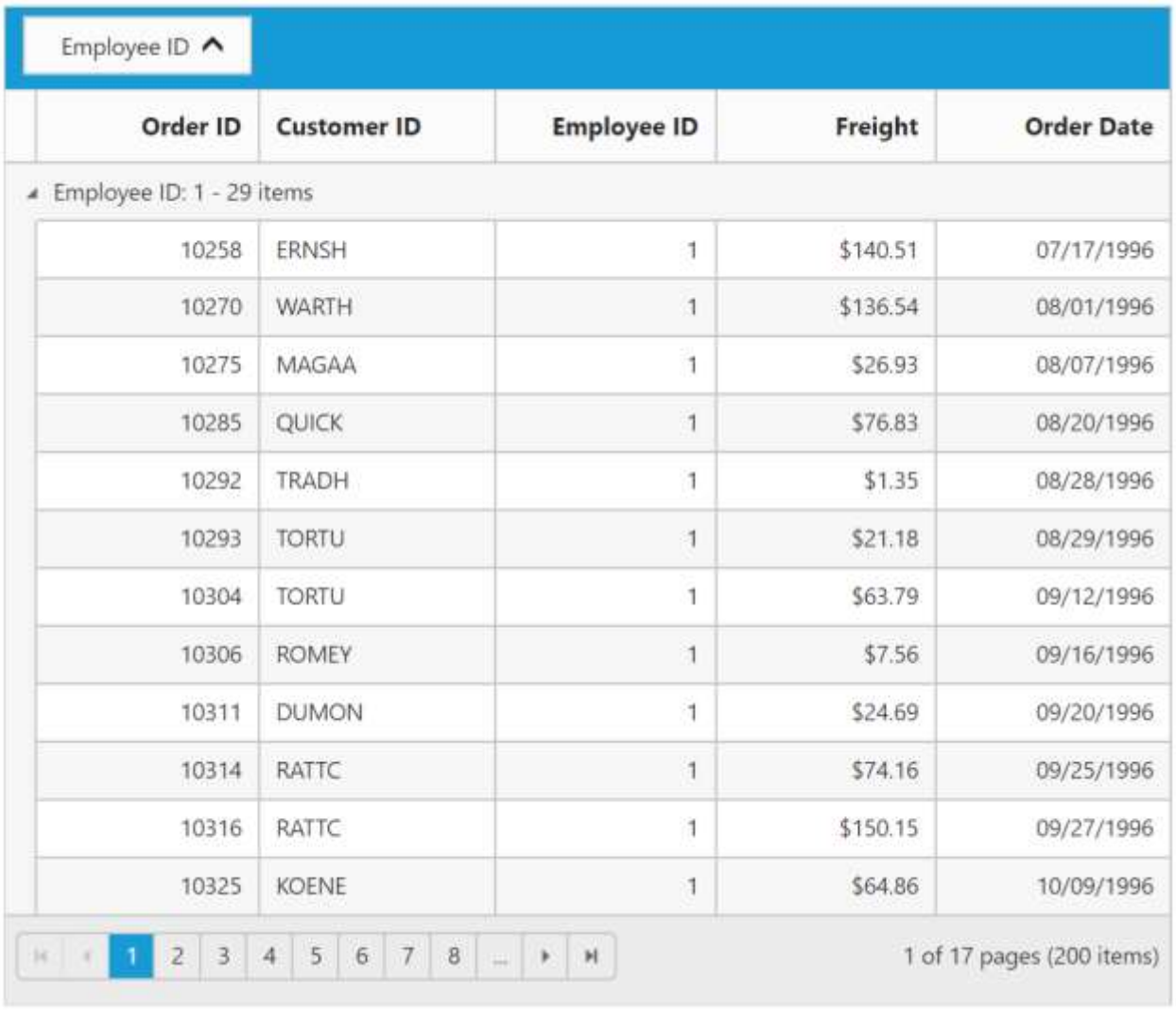
C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
public ActionResult UrlDataSource(DataManager dm)
{
IEnumerable DataSource = new NorthwindDataContext().OrdersViews.ToList();
int count = DataSource.AsQueryable().Count();
IEnumerable GroupDs = new List<object>(); ;
DataOperations ds = new DataOperations();
List<string> str = new List<string>();
if (dm.Group != null)
GroupDs = ds.PerformSelect(DataSource, dm.Group); //Pass grouped column
records
if (dm.Sorted != null)
DataSource = ds.PerformSorting(DataSource, dm.Sorted);
DataSource = ds.PerformSkip(DataSource, dm.Skip);
DataSource = DataSource.AsQueryable().Take(dm.Take);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return Json(new {result = DataSource, count =count, groupDs = GroupDs },
JsonRequestBehavior.AllowGet);
}
}
}
```

The following output is displayed as a result of the above code example.



Summary

Summary rows visibility can be controlled by the ShowSummary property and it can be added to grid by using the SummaryRow array property. The following code sample describes this.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
```

```
row.Title("Sum").SummaryColumns(col => {
col.SummaryType(SummaryType.Sum).Format("{0:C}").DisplayColumn("Freight").Da
taMember("Freight").Add(); }).Add();
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(100).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Employee ID	Ship City	Ship Country	Freight
10248	5	Reims	France	$32.38
10249	6	Münster	Germany	$11.61
10250	4	Rio de Janeiro	Brazil	$65.83
10251	3	Lyon	France	$41.34
10252	4	Charleroi	Belgium	$51.30
10253	3	Rio de Janeiro	Brazil	$58.17
10254	5	Bern	Switzerland	$22.98
10255	9	Genève	Switzerland	$148.33
10256	3	Resende	Brazil	$13.97
10257	4	San Cristóbal	Venezuela	$81.91
10258	1	Graz	Austria	$140.51
10259	4	México D.F.	Mexico	$3.25
Sum				**$13,126.33**

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Supported aggregates

Following are the supported list of aggregates:

- Sum
- Average
- Maximum
- Minimum
- False Count
- True Count

Sum, average, maximum and minimum

Summaries with Sum,average,maximum and minimum aggregate can be defined by using the SummaryType in SummaryColumns collections. These aggregate are used in the Number column.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
row.Title("Sum").SummaryColumns(col => {
col.SummaryType(SummaryType.Sum).Format("{0:C}").DisplayColumn("Freight").Da
taMember("Freight").Add(); }).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
row.Title("Average").SummaryColumns(col => {
col.SummaryType(SummaryType.Average).Format("{0:C}").DisplayColumn("Freight"
).DataMember("Freight").Add(); }).Add();
row.Title("Maximum").SummaryColumns(col => {
col.SummaryType(SummaryType.Maximum).Format("{0:C}").DisplayColumn("Freight"
).DataMember("Freight").Add(); }).Add();
row.Title("Minimum").SummaryColumns(col => {
col.SummaryType(SummaryType.Minimum).Format("{0:C}").DisplayColumn("Freight"
).DataMember("Freight").Add(); }).Add();
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(100).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Employee ID	Ship City	Ship Country	Freight
10248	5	Reims	France	$32.38
10249	6	Münster	Germany	$11.61
10250	4	Rio de Janeiro	Brazil	$65.83
10251	3	Lyon	France	$41.34
10252	4	Charleroi	Belgium	$51.30
10253	3	Rio de Janeiro	Brazil	$58.17
10254	5	Bern	Switzerland	$22.98
10255	9	Genève	Switzerland	$148.33
10256	3	Resende	Brazil	$13.97
10257	4	San Cristóbal	Venezuela	$81.91
10258	1	Graz	Austria	$140.51
10259	4	México D.F.	Mexico	$3.25
Sum				**$13,126.33**
Average				**$65.63**
Maximum				**$890.78**
Minimum				**$0.12**

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

True and False count

Summaries with True and False count aggregate can be defined by using the SummaryType,SummaryColumns collections. The True and False count aggregates are used for boolean columns.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
row.Title("False Count").SummaryColumns(col => {
col.SummaryType(SummaryType.FalseCount).DisplayColumn("Verified").DataMember
("Verified").Add(); }).Add();
row.Title("True Count").SummaryColumns(col => {
col.SummaryType(SummaryType.TrueCount).DisplayColumn("Verified").DataMember(
"Verified").Add(); }).Add();
})
.AllowPaging()
.Columns(col =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(100).Add();
col.Field("Verified").HeaderText("Verified").Width(80).Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Employee ID	Ship City	Ship Country	Verified
10248	5	Reims	France	☑
10249	6	Münster	Germany	☐
10250	4	Rio de Janeiro	Brazil	☑
10251	3	Lyon	France	☑
10252	4	Charleroi	Belgium	☑
10253	3	Rio de Janeiro	Brazil	☑
10254	5	Bern	Switzerland	☐
10255	9	Genève	Switzerland	☑
10256	3	Resende	Brazil	☐
10257	4	San Cristóbal	Venezuela	☑
10258	1	Graz	Austria	☑
10259	4	México D.F.	Mexico	☐
False Count				**91**
True Count				**109**

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Custom summary

Custom summary can be used to create summary values based on your required custom logic and calculations. To enable custom summary, the SummaryType should be Custom and CustomSummaryValue property need to defined as function. In the CustomSummaryValue function, you need to use the grid instance to access model.dataSource and model.currentViewData. After the custom calculation, the returned value will be displayed in corresponding summary cell.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
row.Title("Currency").SummaryColumns(col => {
col.SummaryType(SummaryType.Custom).CustomSummaryValue("currency").DisplayCo
lumn("Freight").Format("{0:C2}").Add(); }).Add(););
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(70).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("CustomerID").HeaderText("CustomerID").TextAlign(TextAlign.Right).
Width(70).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(70).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(70).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
70).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/javascript">
function currency(args,data) {
//ej.sum is aggregate to add data's of freight from datasource
return ej.sum(data, "Freight");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Customer ID	Employee ID	Ship City	Freight
10248	VINET	5	Reims	$32.38
10249	TOMSP	6	Münster	$11.61
10250	HANAR	4	Rio de Janeiro	$65.83
10251	VICTE	3	Lyon	$41.34
10252	SUPRD	4	Charleroi	$51.30
10253	HANAR	3	Rio de Janeiro	$58.17
10254	CHOPS	5	Bern	$22.98
10255	RICSU	9	Genève	$148.33
10256	WELLI	3	Resende	$13.97
10257	HILAA	4	San Cristóbal	$81.91
10258	ERNSH	1	Graz	$140.51
10259	CENTC	4	México D.F.	$3.25
Currency				**$13,126.33**

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Group summary

Group summary is used to summarize values of a particular column based on group and it shows at bottom of each group. To enable group Summary for particular group, you need to define the ShowTotalSummary as false.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
row.ShowTotalSummary(false).SummaryColumns(col =>
{
col.SummaryType(SummaryType.Sum)
.Format("{0:C2}")
.DisplayColumn("Freight")
.DataMember("Freight")
.Prefix("Sum = ")
.Add();
}).Add();
})
.AllowPaging()
.AllowSorting()
.AllowGrouping()
```

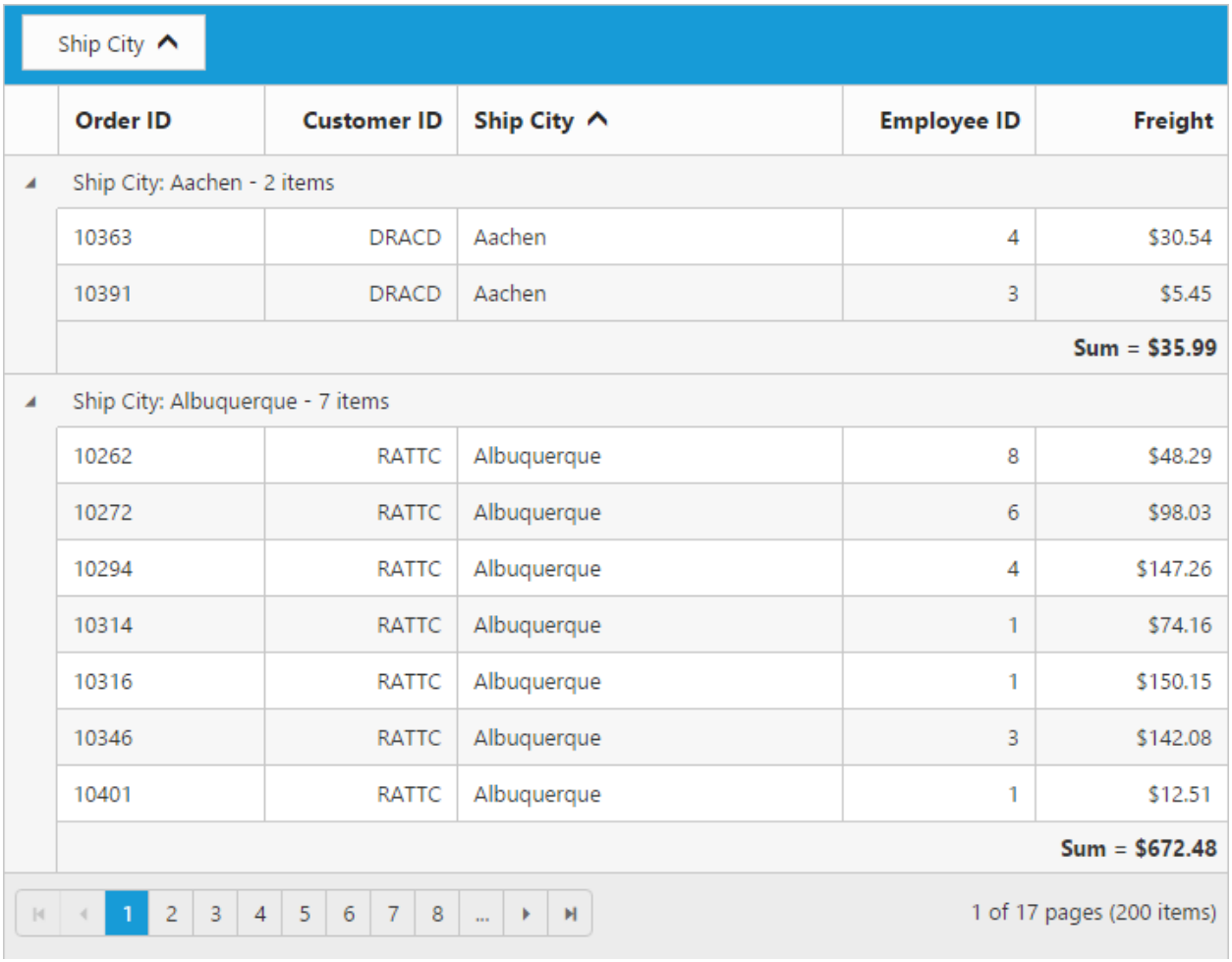
Copyright © 2001 - 2020 Syncfusion Inc.

```
.GroupSettings(group => { group.GroupedColumns(col => {
col.Add("CustomerID"); }); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width(80).Add();
col.Field("CustomerID").HeaderText("CustomerID").TextAlign(TextAlign.Right).
Width(75).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(150).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Waring: Minimum one column should be grouped to show the summary details.

Group caption summary

To show summaries in each group's caption row, particular summary row should have the ShowTotalSummary as false and the ShowCaptionSummary as true.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
row.ShowTotalSummary(false).SummaryColumns(col =>
{
col.SummaryType(SummaryType.Average)
.Format("{0:C2}")
.DisplayColumn("Freight")
.DataMember("Freight")
.Prefix("Average = ")
.Add();
}).Add();
})
.AllowPaging()
```

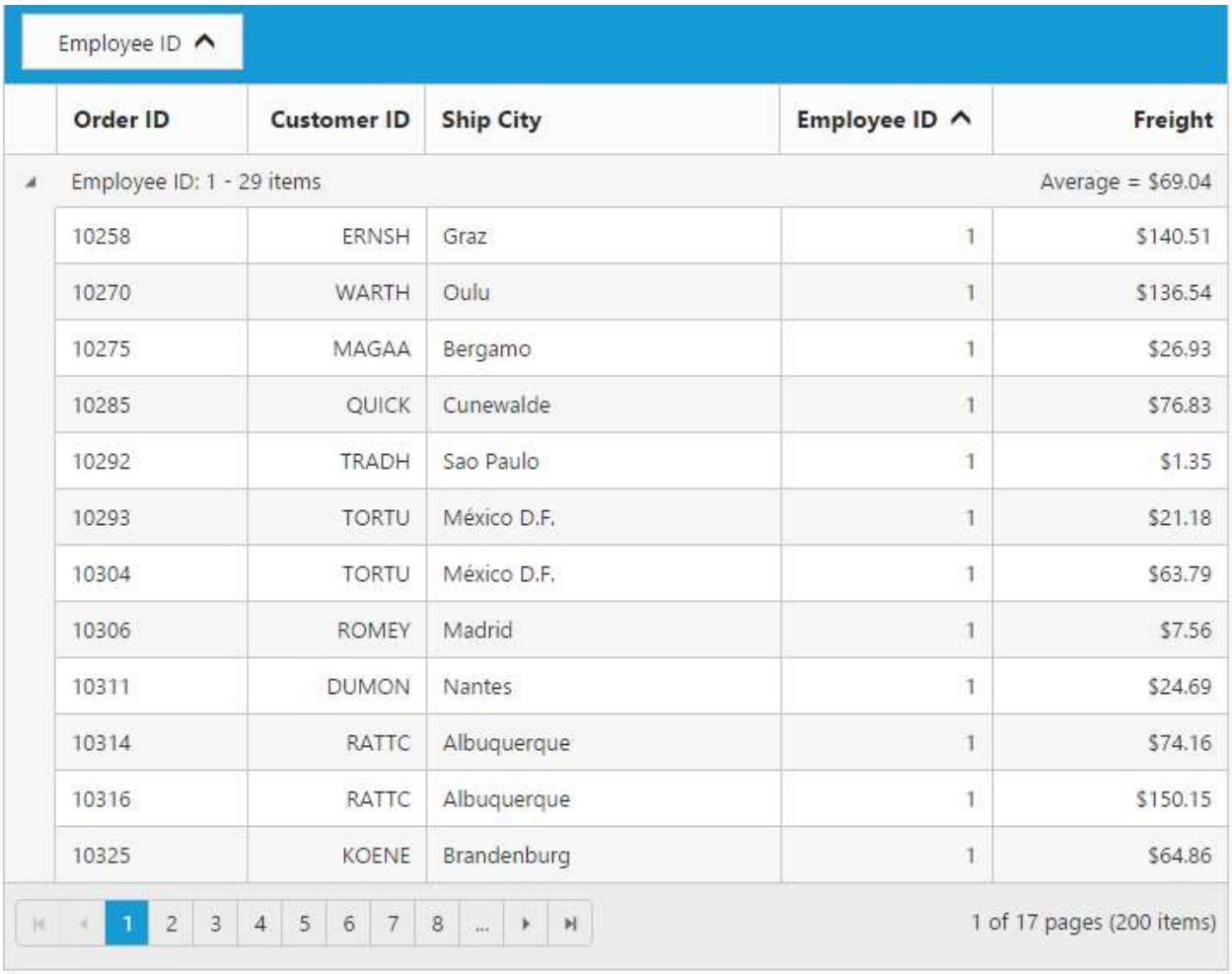
Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowSorting()
.AllowGrouping()
.GroupSettings(group => { group.GroupedColumns(col => {
col.Add("EmployeeID"); }); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Width(80).Add();
col.Field("CustomerID").HeaderText("CustomerID").TextAlign(TextAlign.Right).
Width(75).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(150).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Waring: Minimum one column should be grouped to the show summary details.

Summary template

Using the Template property of SummaryColumns you can render any type of JsRender templates or customizing the summary value.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowSummary()
.SummaryRow(row =>
{
row.Title("Average").SummaryColumns(col =>
{
col.SummaryType(SummaryType.Average)
.Format("{0:C2}")
.DisplayColumn("Freight")
.DataMember("Freight")
.Template("#templateData")
.Add();
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.Take(5).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script id="templateData" type="text/x-jsrender">
Freight has Average of {{"{{"}}:summaryValue{{"}}"}} in  dollars
</script>
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	Freight
10248	5	$32.38
10249	6	$11.61
10250	4	$65.83
10251	3	$41.34
10252	4	$51.30
Average		**Freight has Average of $40.49 in dollars**

Format

To format summary values, the Format property needs to be assigned in SummaryColumns collection object. To know more about formatting options. Please refer to the globalize.js

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource((IEnumerable<object>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ShowSummary()
.SummaryRow(row =>
{
row.Title("Sum").SummaryColumns(col => {
col.SummaryType(SummaryType.Sum).Format("{0:C}").DisplayColumn("Freight").Da
taMember("Freight").Add(); }).Add();
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(100).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Employee ID	Ship City	Ship Country	Freight
10248	5	Reims	France	$32.38
10249	6	Münster	Germany	$11.61
10250	4	Rio de Janeiro	Brazil	$65.83
10251	3	Lyon	France	$41.34
10252	4	Charleroi	Belgium	$51.30
10253	3	Rio de Janeiro	Brazil	$58.17
10254	5	Bern	Switzerland	$22.98
10255	9	Genève	Switzerland	$148.33
10256	3	Resende	Brazil	$13.97
10257	4	San Cristóbal	Venezuela	$81.91
10258	1	Graz	Austria	$140.51
10259	4	México D.F.	Mexico	$3.25
Sum				**$13,126.33**

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Handling aggregation in server-side

The Aggregation at server-side is handled by using the aggregate key. While using remote data, summary row must be handled by returning summary column datasource into the aggregate property of result object.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Summary")
.Datasource(datasource =>
datasource.URL("/Grid/DataSource").Adaptor("UrlAdaptor"))
.ShowSummary()
.AllowPaging()
.SummaryRow(row =>
{
row.ShowTotalSummary(true)
row.Title("Sum").SummaryColumns(col ={
col.SummaryType(SummaryType.Sum)
.DisplayColumn("Freight")
.DataMember("Freight")
.Format("{0:C}")
.Add();
}).Add();
})
.Columns(col =>
```

```
{
col.Field("OrderID").HeaderText("Order ID").Width(100).Add();
col.Field("EmployeeID").HeaderText("Employee ID").Width(100).Add();
col.Field("Freight").HeaderText("Freight").Width(100).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
})
)
}
```

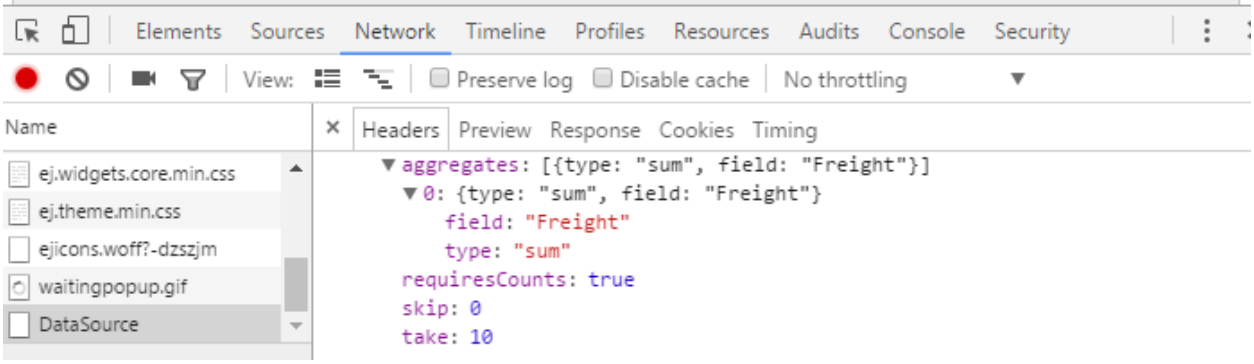
C#

```
namespace MvcApplication4.Controllers
{
public partial class GridController: Controller
{
public ActionResult GridFeatures()
{
ViewBag.dataSource = OrderRepository.GetAllRecords().ToList();
return View();
}
public ActionResult DataSource(DataManager dataManager)
{
IEnumerable DataSource = OrderRepository.GetAllRecords();
DataResult result = new DataResult();
DataOperations dataOperations = new DataOperations();
List<string> aggregateFields = new List<string>();
if (dataManager.Aggregates != null)
{
for (var i = 0; i < dataManager.Aggregates.Count; i++)
string.Add(dataManager.Aggregates[i].Field);
result.aggregate = dataOperations.PerformSelect(DataSource,
aggregateFields);
}
DataSource = dataOperations.PerformSkip(DataSource, dataManager.Skip);
result.result = dataOperations.PerformTake(DataSource, dataManager.Take);
result.count = DataSource.AsQueryable().Count();
return Json(result, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable result { get; set; }
public int count { get; set; }
public IEnumerable aggregate { get; set; }
public IEnumerable groupDs { get; set; }
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.

Order ID	Employee ID	Freight	Ship City
10248	5	32.38	Reims
10249	6	11.61	Münster
10250	4	65.83	Rio de Janeiro
10251	3	41.34	Lyon
10252	4	51.3	Charleroi
10253	3	58.17	Rio de Janeiro
10254	5	22.98	Bern
10255	9	148.33	Genève
10256	3	13.97	Resende
10257	4	81.91	San Cristóbal
Sum		**$64,942.69**	



Paging

You can display the grid records in paged view, by setting the AllowPaging property as true.

The code sample to enable paging is as follows.

RAZOR

```
@(Html.EJ().Grid<Object>("Paging")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	CustomerID	ShipCountry	Freight
10248	5	VINET	France	32.38
10249	6	TOMSP	Germany	11.61
10250	4	HANAR	Brazil	65.83
10251	3	VICTE	France	41.34
10252	4	SUPRD	Belgium	51.3
10253	3	HANAR	Brazil	58.17
10254	5	CHOPS	Switzerland	22.98
10255	9	RICSU	Switzerland	148.33
10256	3	WELLI	Brazil	13.97
10257	4	HILAA	Venezuela	81.91
10258	1	ERNSH	Austria	140.51
10259	4	CENTC	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Pager with query string

You can pass the current page information as a query string while navigating to other page. To enable query string, set the `EnableQueryString` property of `PageSettings` as `true`.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<Object>("Paging")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
```

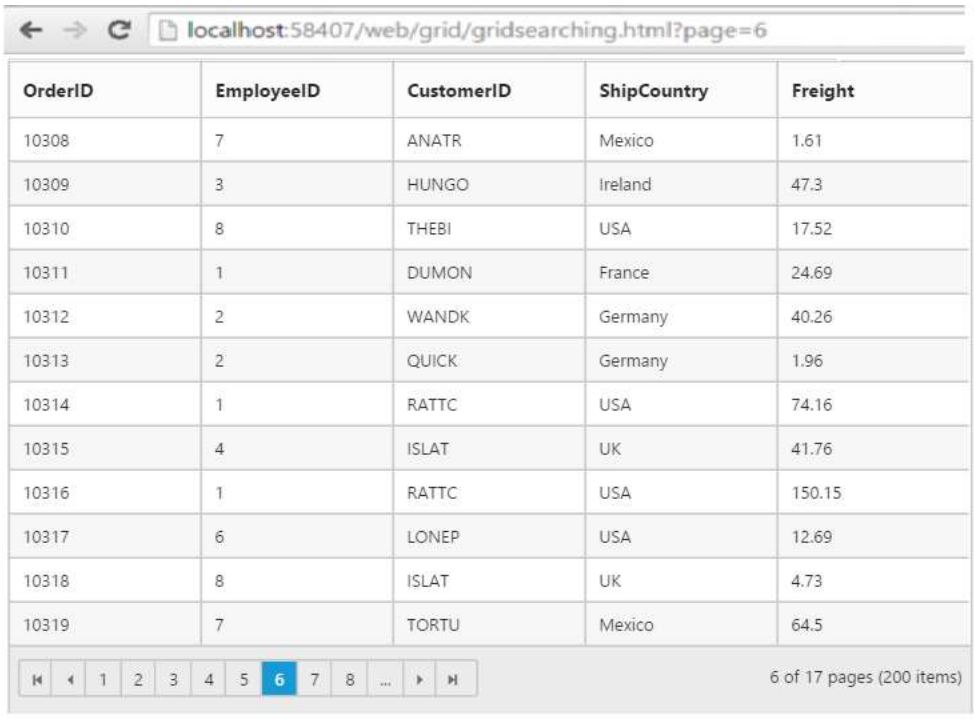
Copyright © 2001 - 2020 Syncfusion Inc.

```
.PageSettings(Page => { Page.EnableQueryString(true); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



Pager template

Apart from default pager, there is an option to render a specific custom template in a grid pager. To render template in pager, set `EnableTemplates` as true and `Template` properties of `PageSettings`.

Prevent to show the default pager while enabling the pager Template by setting ShowDefaults property of PageSettings as false.

Note: It's a standard way to enclose the Template within the script tag with type as "text/x-jsrender".

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Paging")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.PageSettings(Page => {
Page.EnableTemplates().Template("#template").ShowDefaults(false); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Format("{0:C}").Add();
}).ClientSideEvents(e=>e.Create("create"))
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script id="template" type="text/x-jsrender">
<input id="currentPage" type="text" style="text-align: center; width: 32px;
height: 21px; background: white;" value="1" />
<label>of 200</label>
<button id="btn">Go to</button>
</script>
<script>
function create(args) {
$("#btn").ejButton({
click : "btnClick"
});
}
function btnClick(args) {
var val = $("#currentPage").val();
var gridObj = $("#Grid").data("ejGrid");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
gridObj.gotoPage(args.val);
}
</script>
```

CSS

```
.e-grid .e-pager .e-pagercontainer {
border-width: 0px;
overflow: visible;
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 **of 200** Go to

Pager with pageSize drop down

There is an option to set the size of page by means of selecting from the page size values available at the dropdown in pager. Once selected it updates the pageSize value accordingly. To render drop down in pager, provide the pageSize values you wish to display in drop down as `List` to the `PageSizeList` property of `PageSettings`.

The following code sample describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<object>("Paging")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.PageSettings(p => { p.PageSizeList(new List<int>() { 7, 2 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Add();
col.Field("EmployeeID").Add();
col.Field("Freight").Format("{0:C}").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	CustomerID	EmployeeID	Freight	ShipCity	ShipCountry
10248	VINET	5	$32.38	Reims	France
10249	TOMSP	6	$11.61	Münster	Germany
10250	HANAR	4	$65.83	Rio de Janeiro	Brazil
10251	VICTE	3	$41.34	Lyon	France
10252	SUPRD	4	$51.30	Charleroi	Belgium
10253	HANAR	3	$58.17	Rio de Janeiro	Brazil
10254	CHOPS	5	$22.98	Bern	Switzerland

1 2 3 4 5 6 7 8 7 1 of 8 pages (50 items)

7
2

Pager with pageSettings

We can customize the default page settings, such as the pageCount, pageSize of the grid's pager by using the pageSettings property of grid control.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<object>("Paging")
.Datasource((IEnumerable<object>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowPaging()
.PageSettings(page => page.PageSize(8).PageCount(3))
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Add();
col.Field("EmployeeID").Add();
col.Field("Freight").Format("{0:C}").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Paging()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	CustomerID	OrderDate	Verified
10248	5	VINET	04/07/1996	☑
10249	6	TOMSP	05/07/1996	☐
10250	4	HANAR	08/07/1996	☑
10251	3	VICTE	08/07/1996	☑
10252	4	SUPRD	09/07/1996	☑
10253	3	HANAR	10/07/1996	☑
10254	5	CHOPS	11/07/1996	☐
10255	9	RICSU	12/07/1996	☑

1 2 3 ... 1 of 25 pages (200 items)

Scrolling

Scrolling can be enabled by setting the AllowScrolling as true. The height and width can be set to grid by using the properties Height and Width property of the ScrollSettings.

Note: If Width and Height is not defined in the ScrollSettings property then the horizontal and vertical scrollbar is enabled, only when the grid width exceeds the browser width.

Copyright © 2001 - 2020 Syncfusion Inc.

The height and width can be set in percentage and pixel. The default value for Height and Width in ScrollSettings is 0 and auto respectively.

The following code example describes the previous behavior.

RAZOR

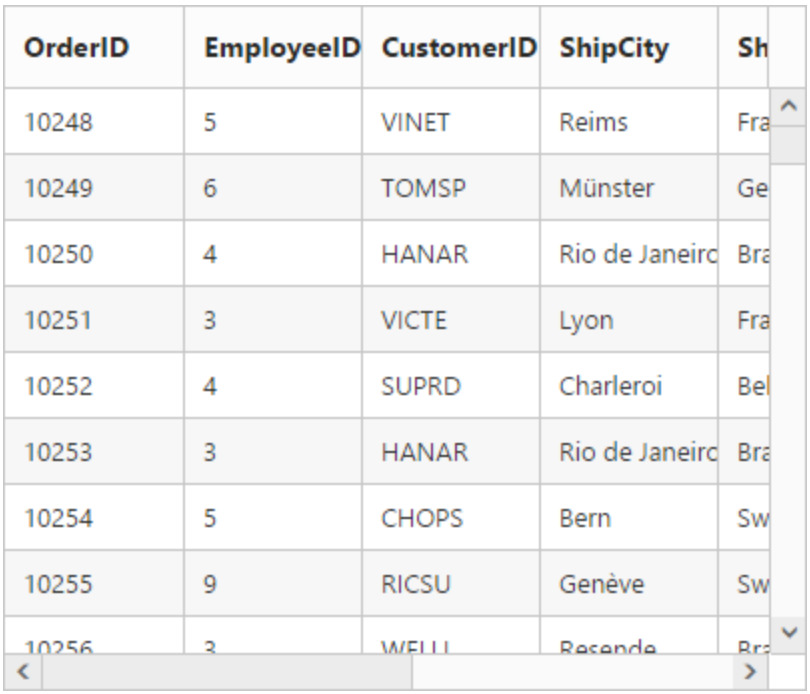
```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width(400).Height(300); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add()
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Set width and height in pixel

To specify the Width and Height property of ScrollSettings in pixel, by set the pixel value as integer.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width(500).Height(300); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

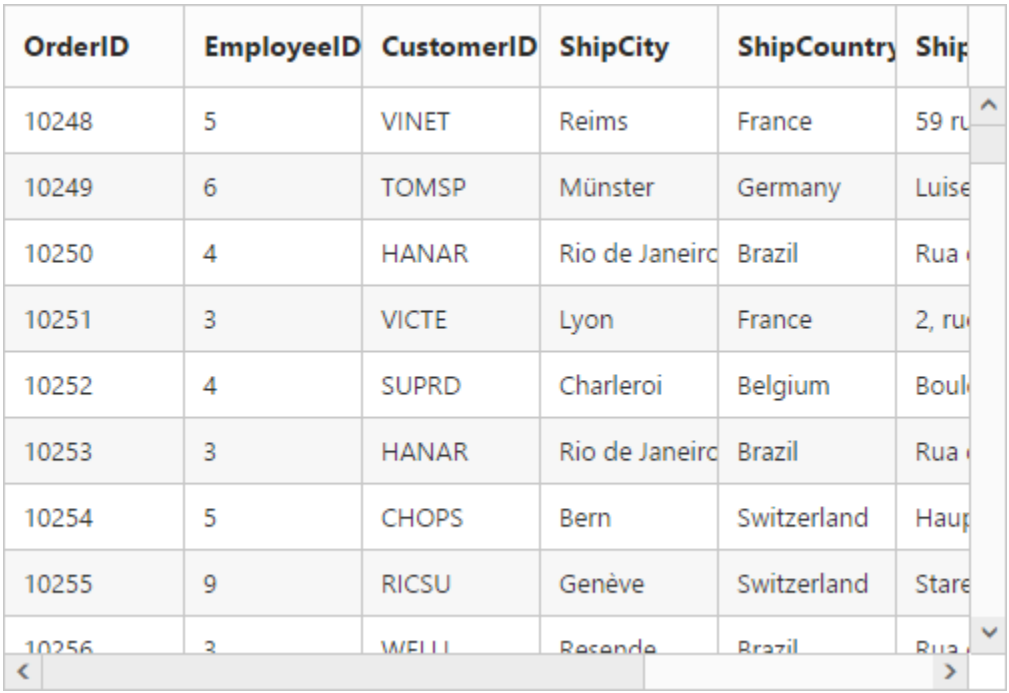
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



OrderID	EmployeeID	CustomerID	ShipCity	ShipCountry	Ship
10248	5	VINET	Reims	France	59 ru
10249	6	TOMSP	Münster	Germany	Luise
10250	4	HANAR	Rio de Janeirc	Brazil	Rua
10251	3	VICTE	Lyon	France	2, ru
10252	4	SUPRD	Charleroi	Belgium	Boul
10253	3	HANAR	Rio de Janeirc	Brazil	Rua
10254	5	CHOPS	Bern	Switzerland	Haup
10255	9	RICSU	Genève	Switzerland	Stare
10256	3	WELLI	Resende	Brazil	Rua

Set width and height in percentage

To specify the `Width` and `Height` property of `ScrollSettings` in percentage, by set the percentage value as string.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width("70%").Height("5%"); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



OrderID	EmployeeID	CustomerID	ShipCity	ShipCountry	ShipA
10248	5	VINET	Reims	France	59 rue
10249	6	TOMSP	Münster	Germany	Luisens
10250	4	HANAR	Rio de Janeirc	Brazil	Rua do
10251	3	VICTE	Lyon	France	2, rue c
10252	4	SUPRD	Charleroi	Belgium	Boulev
10253	3	HANAR	Rio de Janeirc	Brazil	Rua do
10254	5	CHOPS	Bern	Switzerland	Haupts
10255	9	RICSU	Genève	Switzerland	Staren
10256	3	WELLI	Resende	Brazil	Rua do
10257	4	HILAA	San Cristóbal	Venezuela	Carrer

Set width as auto

Specify the Width property of ScrollSettings as auto, then the scrollbar is rendered only when the grid width exceeds the browser window width.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width("auto").Height(300); })
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

OrderID	EmployeeID	CustomerID	ShipCity	ShipCountry	ShipAddress	ShipPostalC	Freight
10248	5	VINET	Reims	France	59 rue de l'Ab	51100	32.38
10249	6	TOMSP	Münster	Germany	Luisenstr. 48	44087	11.61
10250	4	HANAR	Rio de Janeirc	Brazil	Rua do Paço,	05454-876	65.83
10251	3	VICTE	Lyon	France	2, rue du Com	69004	41.34
10252	4	SUPRD	Charleroi	Belgium	Boulevard Tirc	B-6000	51.3
10253	3	HANAR	Rio de Janeirc	Brazil	Rua do Paço,	05454-876	58.17
10254	5	CHOPS	Bern	Switzerland	Hauptstr. 31	3012	22.98
10255	9	RICSU	Genève	Switzerland	Starenweg 5	1204	148.33
10256	3	WELLI	Resende	Brazil	Rua do Merca	08737-363	13.97

Frozen columns

Specify the `FrozenColumns` property of `ScrollSettings` to freeze the columns(upto the specified frozenColumns value) at the time of scrolling. Horizontal scrollbar must be enabling while specifying the `FrozenColumns` then only you can scroll and see the remaining columns with freeze pane.

Note: The `AllowScrolling` must be `true` while specifying `FrozenColumns` .

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width(550).Height(300).FrozenColumns(2); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Freeze particular columns:

To freeze selected columns in grid at the time of scrolling, by setting the `IsFrozen` property of columns as `true`. The `IsFrozen` columns are rendered first in the grid even the columns index is different while declaring the `Columns` .

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width(550).Height(400); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("Freight").Format("{0:C}").IsFrozen(true).Add();
col.Field("OrderDate").Format("{0:dd/MM/yyyy}").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Width(100).IsFrozen(true).Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



Freight	ShipCountry	OrderID	Employeel	Customerl	OrderDate	Ship
32.38	France	10248	5	VINET	04/07/1996	Reim
11.61	Germany	10249	6	TOMSP	05/07/1996	Mün
65.83	Brazil	10250	4	HANAR	08/07/1996	Rio d
41.34	France	10251	3	VICTE	08/07/1996	Lyon
51.3	Belgium	10252	4	SUPRD	09/07/1996	Char
58.17	Brazil	10253	3	HANAR	10/07/1996	Rio d
22.98	Switzerland	10254	5	CHOPS	11/07/1996	Bern
148.33	Switzerland	10255	9	RICSU	12/07/1996	Gene
13.97	Brazil	10256	3	WELLI	15/07/1996	Rese
81.91	Venezuela	10257	4	HILAA	16/07/1996	San
140.51	Austria	10258	1	ERNSH	17/07/1996	Graz
3.25	Mexico	10259	4	CENTC	18/07/1996	Méx

Frozen columns alert messages:

1. If the AllowScrolling is false while using FrozenColumns then "Enable AllowScrolling while using frozen Columns" alert message is thrown.
2. If the FrozenColumns is specified out of the grid column view then "Frozen columns should be in grid view area" alert message is thrown.
3. Frozen Rows and Columns are not supported in the following features.

Grouping

Row Template

Detail Template

Copyright © 2001 - 2020 Syncfusion Inc.

Hierarchy Grid

Batch Editing

Virtual Scrolling

If any one of the previous feature is enabled along with Frozen rows and columns, then the "Frozen Columns and Rows are not supported for Grouping, Row Template, Detail Template, Hierarchy Grid and Batch Editing" alert message is thrown.

Frozen rows

Specify the FrozenRows property of ScrollSettings to freeze rows(upto the specified FrozenRows value) at the time of scrolling. Vertical scrollbar must be enabling while specifying the FrozenRows then only you can scroll and see the remaining rows with freeze pane.

Note: The AllowScrolling must be true while specifying FrozenRows .

The following code example describes the previous behavior.

RAZOR

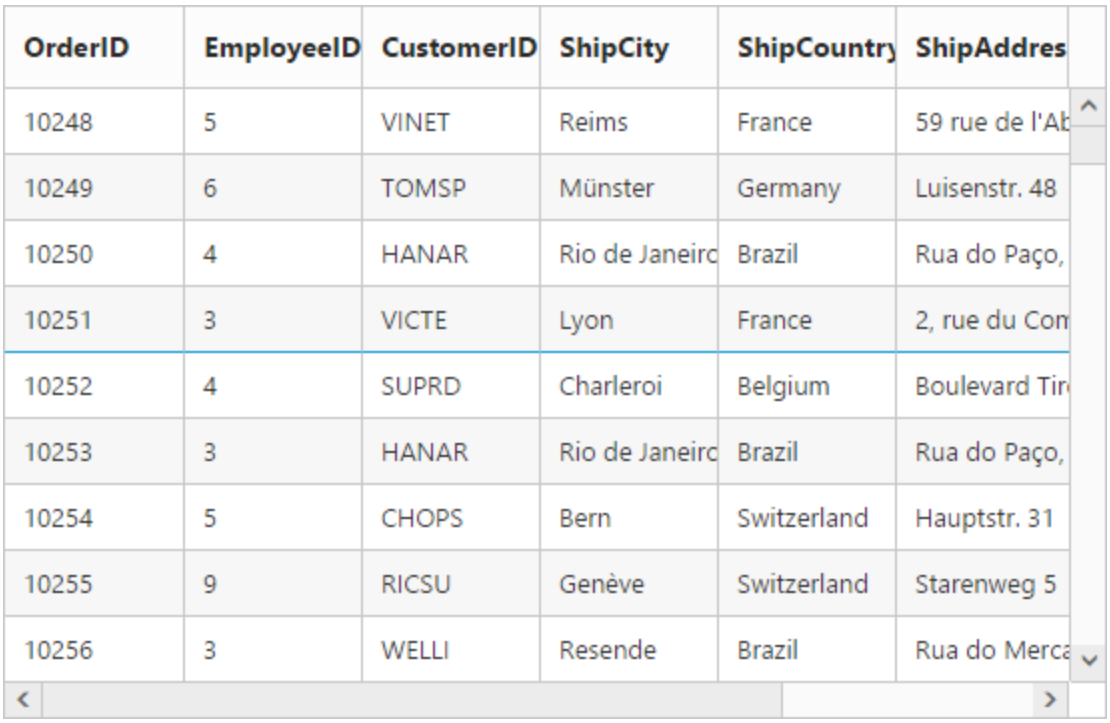
```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => { col.Width(550).Height(300).FrozenRows(4); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	CustomerID	ShipCity	ShipCountry	ShipAddres
10248	5	VINET	Reims	France	59 rue de l'Ab
10249	6	TOMSP	Münster	Germany	Luisenstr. 48
10250	4	HANAR	Rio de Janeirc	Brazil	Rua do Paço,
10251	3	VICTE	Lyon	France	2, rue du Con
10252	4	SUPRD	Charleroi	Belgium	Boulevard Tir
10253	3	HANAR	Rio de Janeirc	Brazil	Rua do Paço,
10254	5	CHOPS	Bern	Switzerland	Hauptstr. 31
10255	9	RICSU	Genève	Switzerland	Starenweg 5
10256	3	WELLI	Resende	Brazil	Rua do Merca

Touch scroll

In touch supported devices you can scroll and show the content by swipe left, right, top and bottom. Disable touch scroll by setting the `EnableTouchScroll` property of `ScrollSettings` as `false`.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => {
col.Width(550).Height(300).EnableTouchScroll(false); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Virtual scrolling

The virtual scrolling support allows you to load data that you require (load data based on page size) without buffering the entire huge database. To enable virtual scrolling by setting the AllowVirtualScrolling property of ScrollSettings as true.

We also have an enhanced virtual scrolling feature with an improvised virtual scrolling performance. To enable improvised virtual scrolling feature by setting EnableVirtualization property of ScrollSettings as true and it doesn't requires AllowVirtualScrolling to enabled. It allows you to load the grid with data while scrolling. In order to enable this, you need to enable EnableVirtualization property of the ScrollSettings. Some of the relevant functionalities of this are,

1. White space will not be appeared in the grid. 2. Improved page rendering performance. 3. It can render nearly 500 thousand records.

It supports two mode of virtualization. They are:

1. Normal Mode
2. Continuous Mode

Note: Enhanced Virtual Scrolling supports only normal mode

The following features are not supported by virtual scrolling

1. Grouping

2. Frozen Rows

3. Cell merging

4. Detail template

5. Hierarchy

6. Editing

Normal mode:

It allows you to load the grid with data while scrolling. This can be achieved by setting the VirtualScrollMode as Normal.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ScrollSettings(col => {
col.Width(550).Height(300).AllowVirtualScrolling(true).VirtualScrollMode(Vir
tualScrollMode.Normal); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Enhanced virtual scrolling:

In order to enable this, you need to set the EnableVirtualization property of the ScrollSettings as true.

The following code example describes the previous behavior.

RAZOR

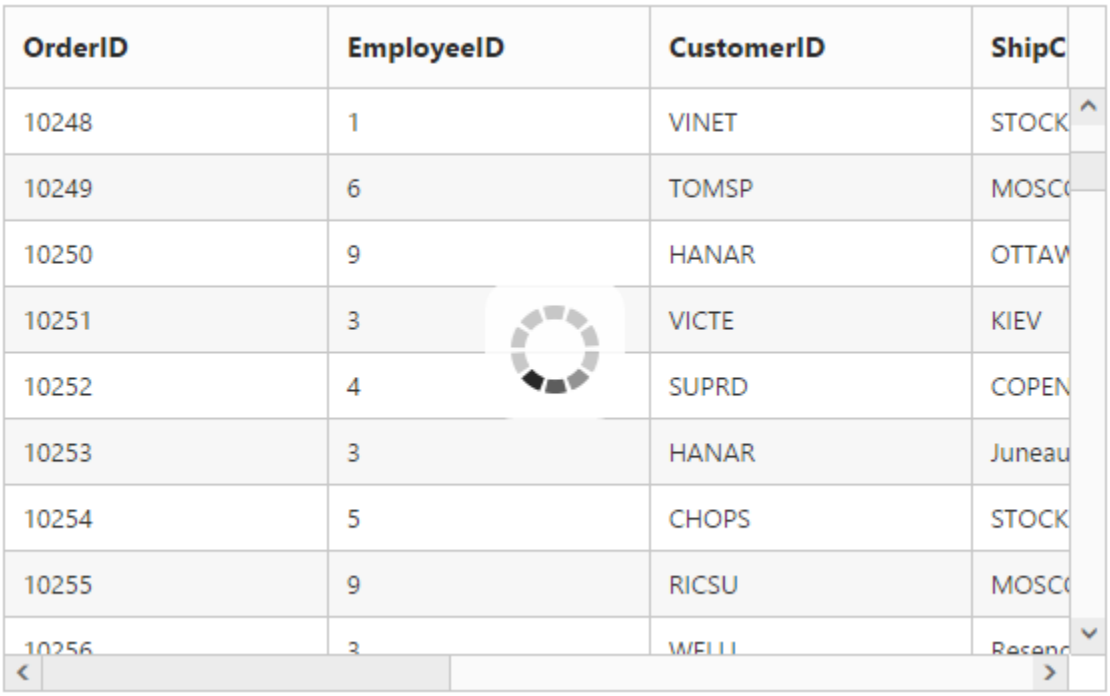
```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => {
col.Width(550).Height(300).EnableVirtualization(true); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
```

```
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



OrderID	EmployeeID	CustomerID	ShipC
10248	1	VINET	STOCK
10249	6	TOMSP	MOSC
10250	9	HANAR	OTTAW
10251	3	VICTE	KIEV
10252	4	SUPRD	COPEN
10253	3	HANAR	Juneau
10254	5	CHOPS	STOCK
10255	9	RICSU	MOSC
10256	3	WELL	Resen

Continuous mode:

In continuous mode, the data is loaded in grid when the scrollbar reaches the end. You can enable the continuous mode by setting the VirtualScrollMode property as Continuous.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Scrolling")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling()
.ScrollSettings(col => {
col.Width(550).Height(300).AllowVirtualScrolling(true).VirtualScrollMode(Vir
tualScrollMode.Continuous); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("CustomerID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("ShipAddress").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("ShipPostalCode").Add();
col.Field("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Scrolling()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



OrderID	EmployeeID	CustomerID	ShipCity	ShipCountry	ShipAddres
10249	6	TOMSP	MOSCOW	RUSSIA	CENTRAL ST
10250	9	HANAR	OTTAWA	CANADA	Hamilton
10251	3	VICTE	KIEV	UKRAINE	Khreschatik
10252	4	SUPRD	COPENHAGEN	DENMARK	Connaught Pl
10253	3	HANAR	Juneau	Venesuela	NORTH SLOP
10254	5	CHOPS	STOCKHOLM	SWEDEN	OLDTOWN
10255	9	RICSU	MOSCOW	RUSSIA	CENTRAL ST
10256	3	WELLI	Resende	Brazil	Rua do Merca
10257	4	HILAA	San Cristóbal	Venezuela	Carrera 22 co

1 of 1330 pages (13294 items)

Responsive

The Grid control has support for responsive behavior based on the client browser's width and height. To enable responsive, the IsResponsive property should be true. In Desktop and Tablet mode, to render scroller set MinWidth property. There are three modes of responsive layout is available in grid based on client width. They are.

Copyright © 2001 - 2020 Syncfusion Inc.

- Mobile(<321px)
- Tablet (321px to 800px)
- Desktop(>800px)

Note: The following features are not supported by grid's responsive:

1. Virtual Scrolling

2. Frozen Rows and Frozen Columns

3. Hierarchy

4. Detail template

Mobile layout

If client width is less than 321px, the grid will render in mobile mode. In which, you can see that grid user interface is customized and redesigned for best view in small screens. The customized features includes responsive row rendering, filtering, sorting, searching and editing.

Responsive row

Enabling responsive row makes the grid to render the record values in vertical order which removes the need for horizontal scrolling to view complete record details. It can be enabled by defining the `EnableResponsiveRow` property as `true`.

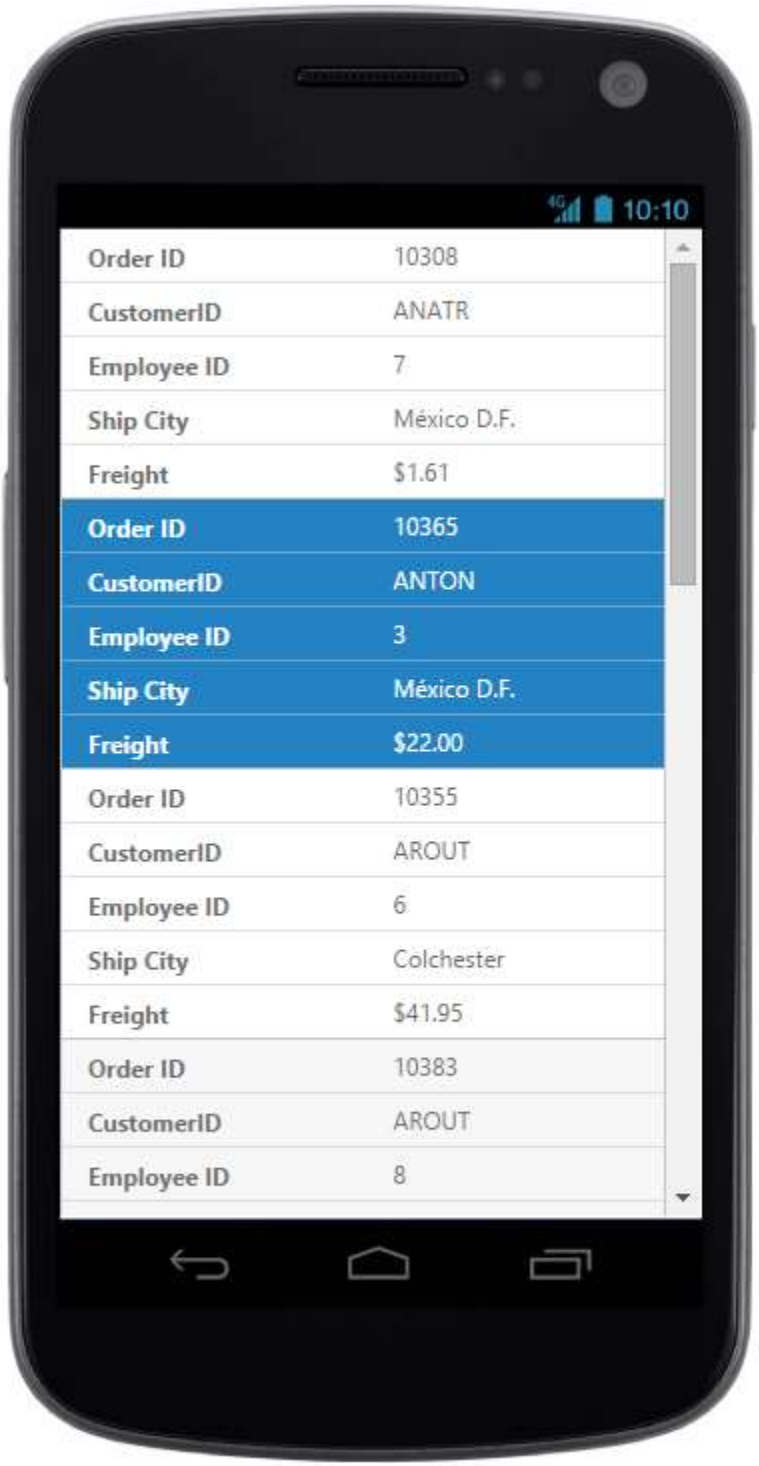
RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.IsResponsive(true)
.EnableResponsiveRow(true)
.AllowPaging()
.PageSettings(p => p.PageCount(3).PageSize(7))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").IsPrimaryKey(true).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee ID").Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
col.Field("Freight").HeaderText("Freight").Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

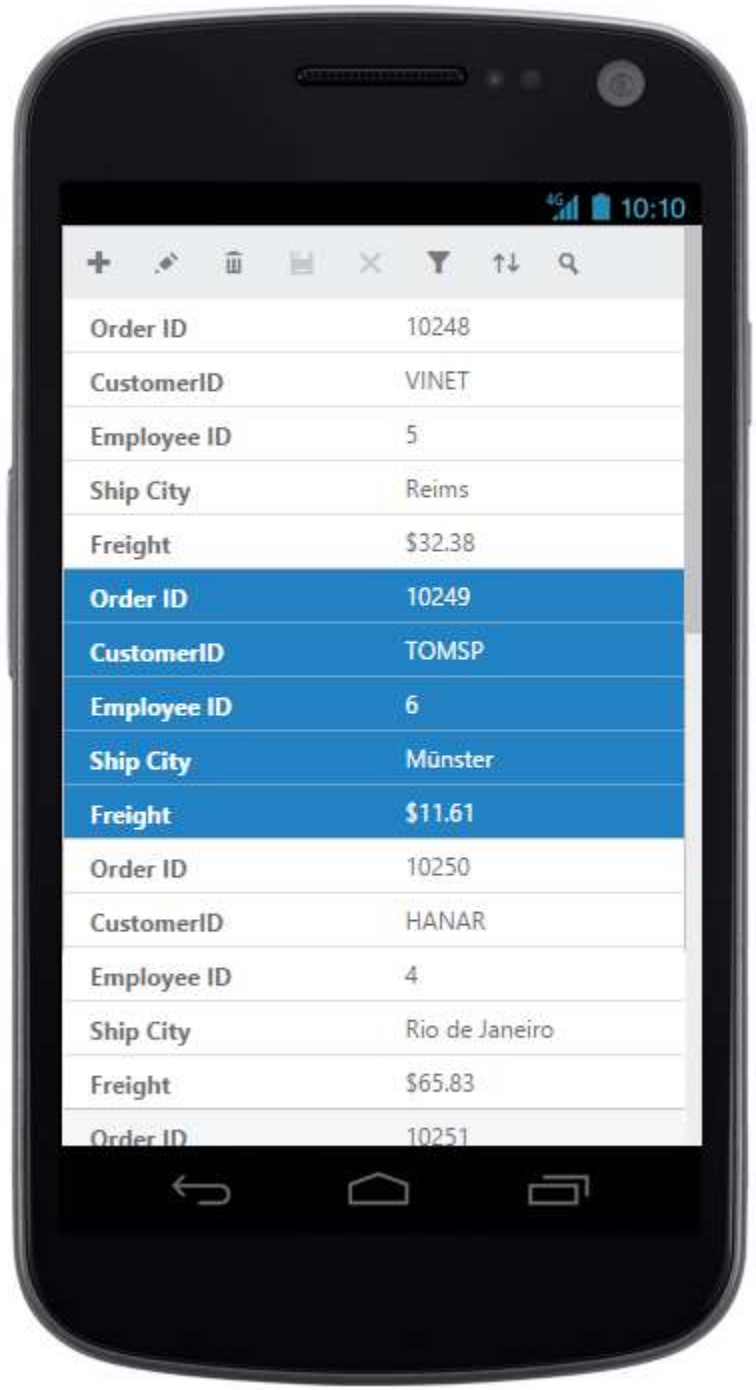


Waring: IE8 and IE9 does not support responsive row. The ejgrid.responsive.css should be referred to display responsive row.

Copyright © 2001 - 2020 Syncfusion Inc.

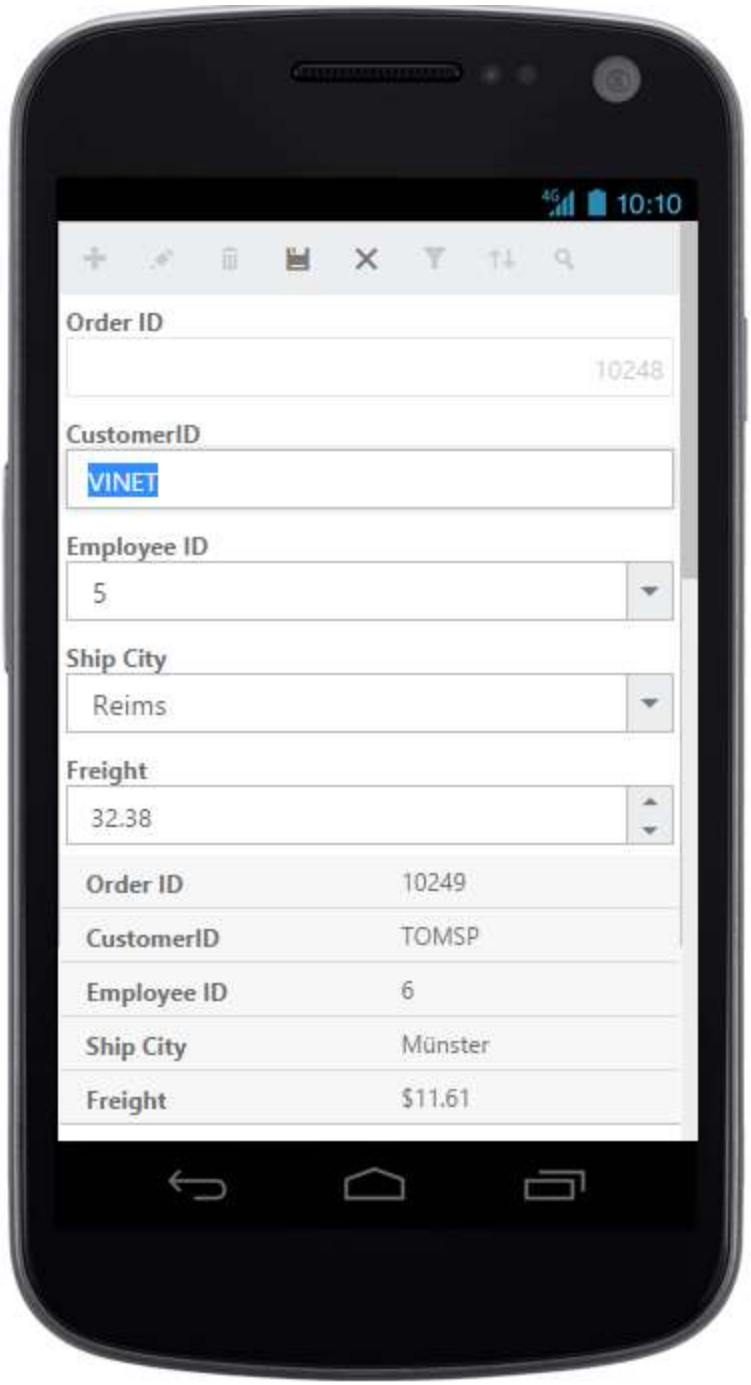
Customized features

The customized layout for filtering, sorting, searching and CRUD operations in mobile device can be seen in the following screen shots.



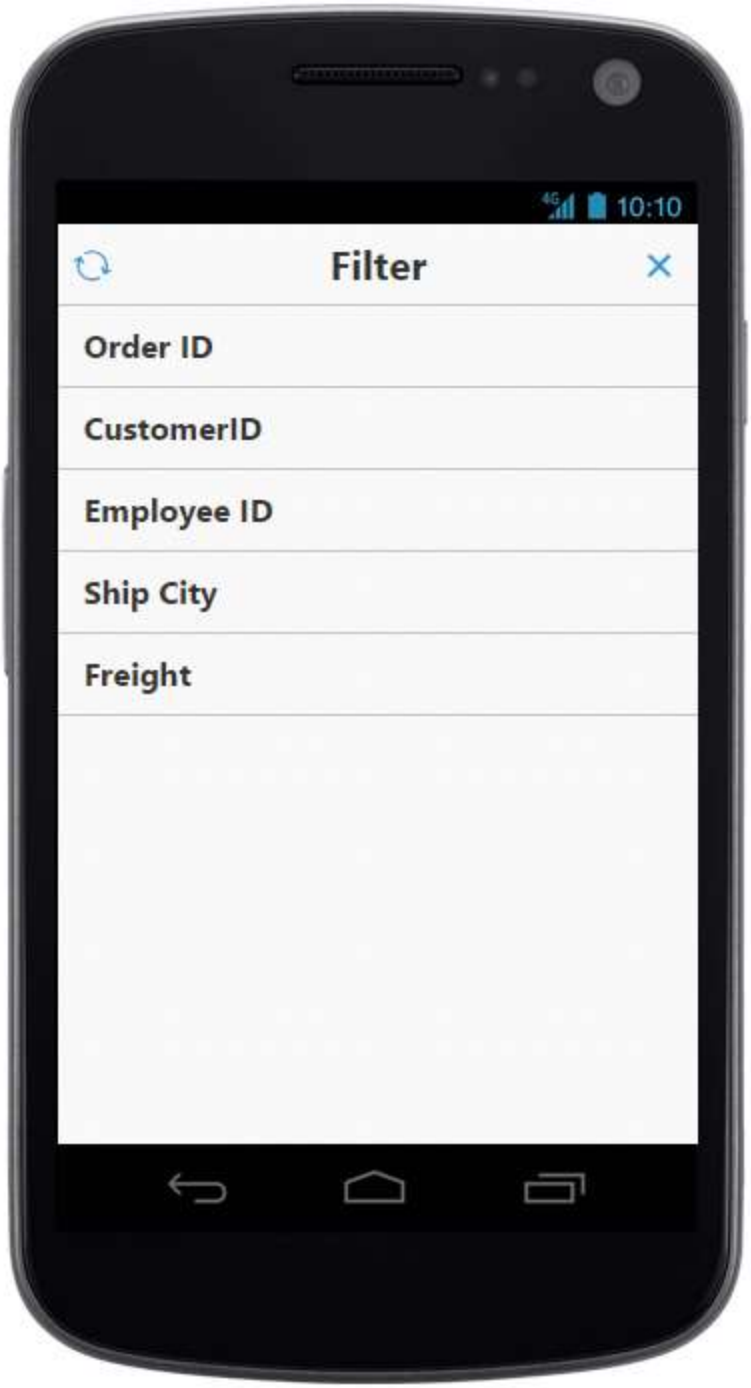
Responsive row with filtering , sorting and searching

Copyright © 2001 - 2020 Syncfusion Inc.



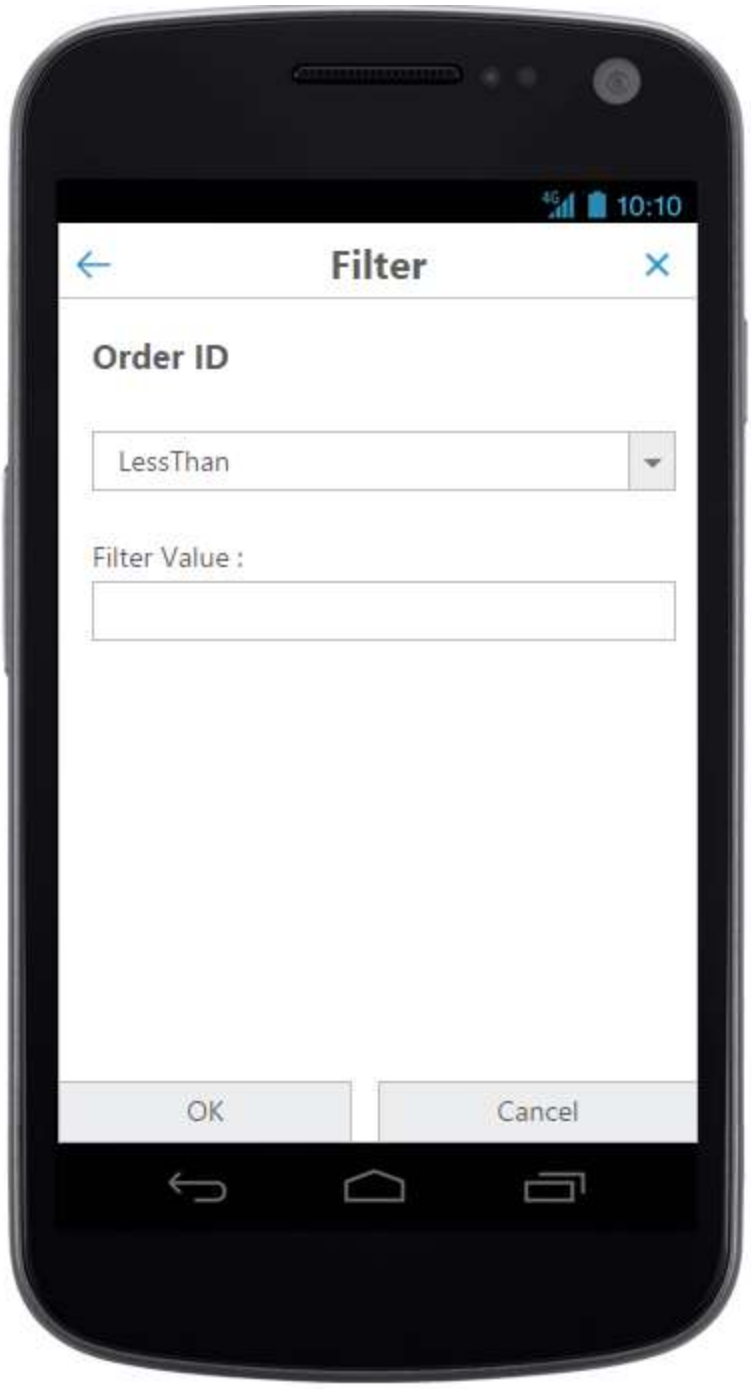
CRUD in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



Filtering in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



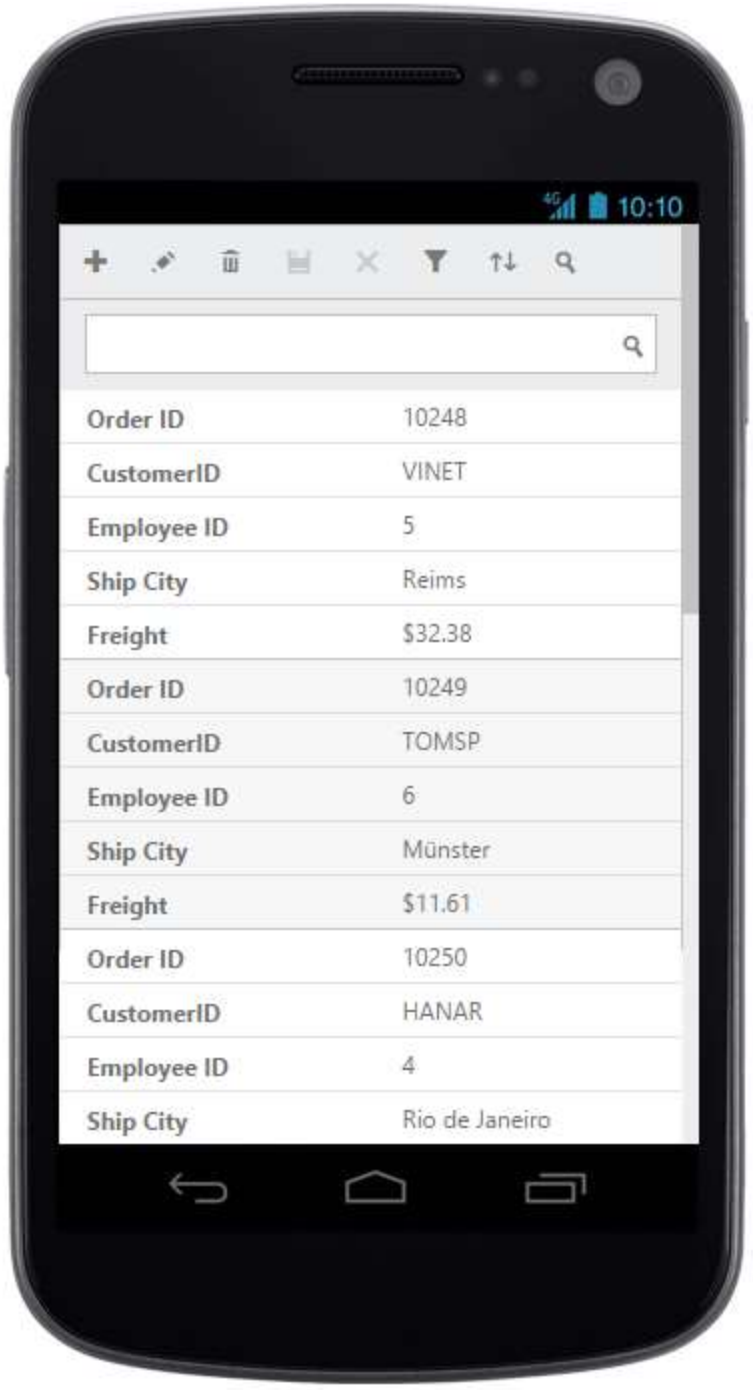
Filtering in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



Sorting in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



Searching in mobile layout

RAZOR

```
@using Syncfusion.JavaScript.Models
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.IsResponsive(true)
.EnableResponsiveRow(true)
.AllowPaging()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.EditSettings(d =>
d.AllowAdding(true).AllowDeleting(true).AllowEditing(true))
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
items.AddTool(ToolBarItems.Search);
});
})
.PageSettings(p => p.PageCount(3).PageSize(7))
.AllowFiltering()
.AllowSorting()
.AllowMultiSorting()
.FilterSettings(filterSettings =>
filterSettings.FilterType(FilterType.Menu))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").Width("90").IsPrimaryKey(true).ValidationRules(v =>
v.AddRule("required", true).AddRule("number",
true)).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").Width("100").ValidationRules(v => v.AddRule("required", true)).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").EditType(EditingType.DropdownEdit).TextAlign(TextAlign.Right).Width("90
").Add();
col.Field("ShipCity").HeaderText("Ship
City").Width("120").EditType(EditingType.DropdownEdit).Add();
col.Field("Freight").HeaderText("Freight").Width("110").EditType(EditingType
.NumericEdit).NumericEditOptions(new EditorProperties() { DecimalPlaces = 2
}).Format("{0:C}").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Tablet layout

If the client width is between 321px and 800px, then the grid will render in tablet mode. Also, it has customized filtering design to match tablet screen size.

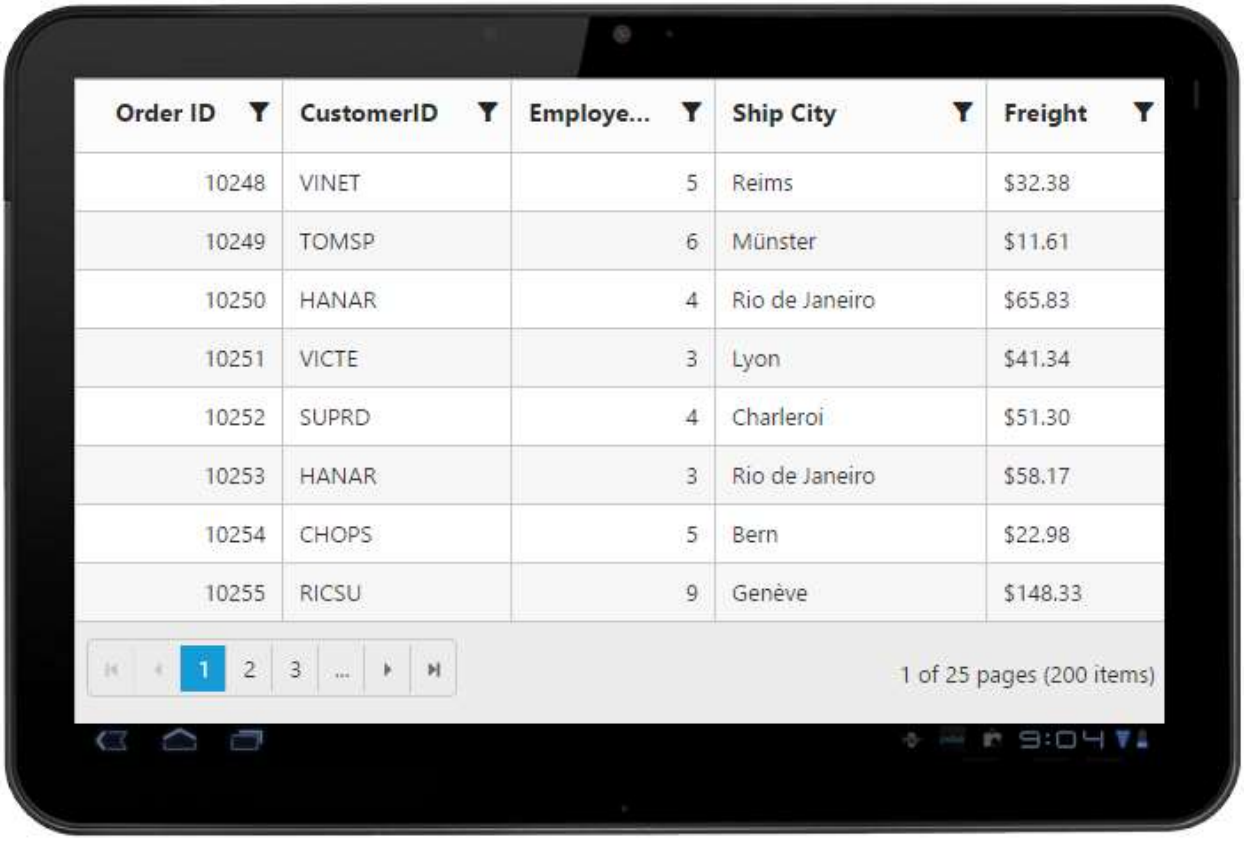
RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.IsResponsive(true)
.AllowFiltering()
.FilterSettings(filterSettings =>
filterSettings.FilterType(FilterType.Menu))
.AllowPaging()
.PageSettings(p => p.PageCount(3).PageSize(8))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width("90").Add();
col.Field("CustomerID").HeaderText("Customer ID").Width("100").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width("90").Add();
col.Field("ShipCity").HeaderText("Ship City").Width("120").Add();
col.Field("Freight").HeaderText("Freight").Width("80").Format("{0:C}").Add()
;
})
)
```

C#

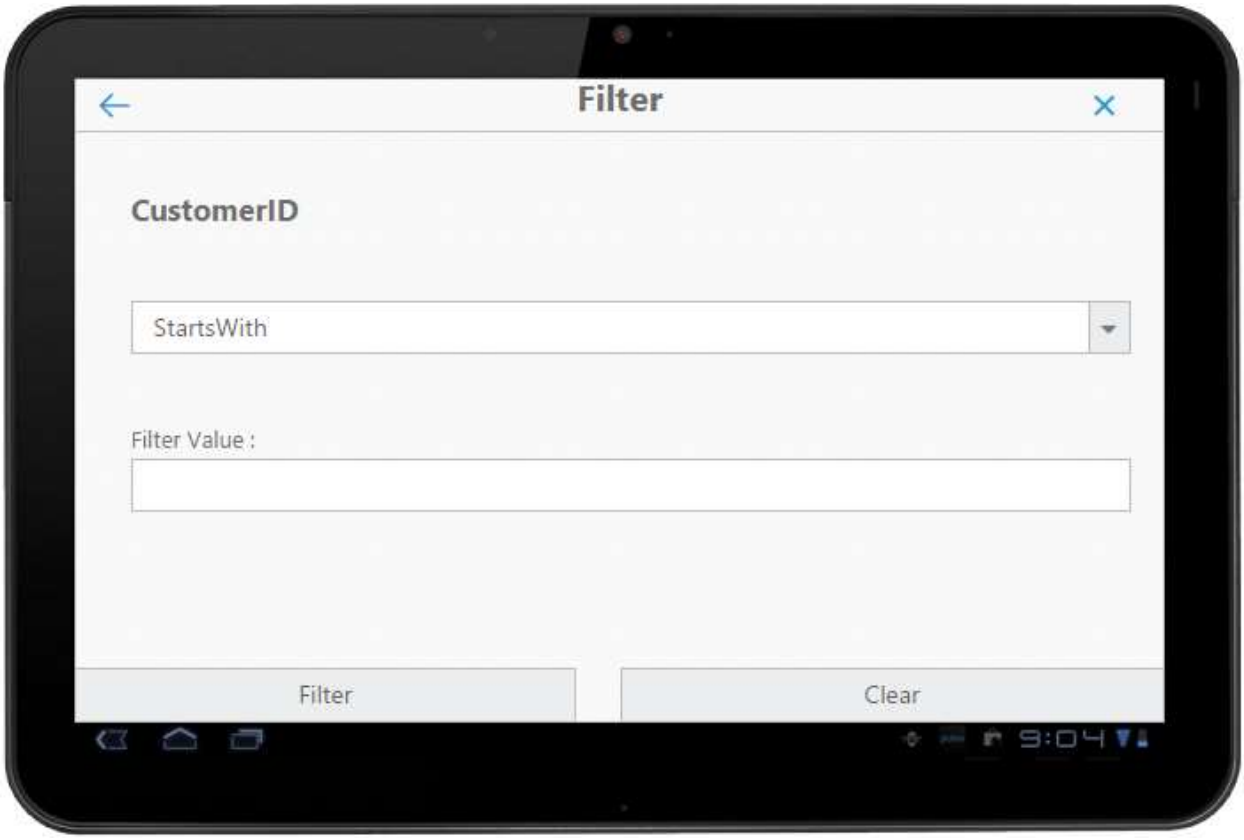
```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Default tab layout

Copyright © 2001 - 2020 Syncfusion Inc.



Filtering design in tab layout.

Width

By default, the grid is adaptable to its parent container. It can adjust its width of columns based on parent container width. You can also assign the Width of Columns in percentage.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width("10%").Add();
col.Field("CustomerID").HeaderText("Customer ID").Width("15%").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width("10%").Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
```

```
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Information: The AllowScrolling should be false while defining Width in percentage.

Min width

Min width is used to maintain minimum width for the grid. To enable min width, MinWidth should be defined. If the grid width is less than MinWidth then the scrollbar will be displayed to maintain minimum width.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.MinWidth(700)
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width("90").Add();
col.Field("CustomerID").HeaderText("Customer ID").Width("100").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width("90").Add();
col.Field("ShipCity").HeaderText("Ship City").Width("120").Add();
col.Field("Freight").HeaderText("Freight").Width("110").Format("{0:C}").Add(
);
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

MinWidth set to grid

Note: We can render the grid with height responsive by setting the scrollSettings heightas 100%. If the grid height is greater than its parent container's height then the vertical scrollbar will be displayed to view the content.

Copyright © 2001 - 2020 Syncfusion Inc.

Priority for columns

Priority makes column to be visible or hidden based on the Priority value and browser's width to best accommodate the possible columns. To enable Priority for Columns, Priority needs to be defined in columns collection. These Priority values are from one to six.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).Priority(1).TextAlign(TextAlign.Right).Width("90").A
dd();
col.Field("CustomerID").HeaderText("Customer
ID").Width("100").Priority(2).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Priority(1).Width("90").Add();
col.Field("ShipCity").HeaderText("Ship
City").Width("120").Priority(3).Add();
col.Field("Freight").HeaderText("Freight").Width("110").Format("{0:C}").Prio
rity(4).Add();
})
)
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Information: The ejgrid.responsive.css should be referred.

Selection

Selection provides an interactive support to highlight the row, cell or column that you select. Selection can be done through simple Mouse down or Keyboard interaction. To enable selection, set the AllowSelection as true.

Types of selection

There are two types of selections available in grid. They are:

1. Single
2. Multiple

Copyright © 2001 - 2020 Syncfusion Inc.

Single selection

Single selection is an interactive support to select a specific row, cell or column in the grid by mouse or keyboard interactions. To enable single selection by setting the SelectionType property as Single and also set the AllowSelection property as true.

Multiple selections

Multiple selections is an interactive support to select a group of rows, cells or columns in grid by mouse or keyboard interactions. To enable multiple selections by set the SelectionType property as Multiple and also set the AllowSelection property as true.

Row selection

Row selection is enabled by setting the SelectionMode property of SelectionSettings as Row. For random row selection, press the **"Ctrl + mouse left"** click and for continuous row selection press the **"Shift + mouse left"** click on the grid rows. To unselect the selected rows, press the **"Ctrl + mouse left"** click on selected row.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSelection()
.SelectionType(SelectionType.Multiple)
.SelectionSettings(select => { select.SelectionMode(mode => {
mode.AddMode(SelectionMode.Row); }); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Multiple row selection using checkbox column

Select multiple rows in grid by using the checkbox column and it can be enabled by setting the column Type as checkbox. It also provides the option to select/deselect all the rows in the grid using a checkbox in the corresponding column header.

If the Field property of Checkbox column is not defined, then it acts as a template column. So Field property is necessary to perform grid actions like sorting, editing, etc., for the corresponding checkbox column.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSelection()
.Columns(col =>
{
col.Type("checkbox").Width(50).Add();
col.Field("OrderID").IsPrimaryKey(true).Width(80).TextAlign(TextAlign.Right)
.Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(75).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").Width(75).TextAlign(TextAlign.Right).Add();
col.Field("Freight").HeaderText("Freight").Format("{0:C}").Width(75).TextAli
gn(TextAlign.Right).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example



Cell selection

Cell selection is enabled by setting the SelectionMode property of SelectionSettings as Cell. For random cell selection, press the **"Ctrl + mouse left"** click and for continuous cell selection, press the **"Shift + mouse left"** click on the grid cells. To unselect selected cells, press the **"Ctrl + mouse left"** on selected cell click.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSelection()
.SelectionType(SelectionType.Multiple)
.SelectionSettings(select => { select.SelectionMode(mode => {
mode.AddMode(SelectionMode.Cell); }); })
.Columns(col =>
```

```
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Cell selection mode

There are two types of cell selection available in grid. They are:

1. Continuous Selection
2. Box Selection

Box cell selection is to select multiple cells vertically based on the initial column index selection.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSelection()
.SelectionType(SelectionType.Multiple)
.SelectionSettings(select => { select.SelectionMode(mode => {
mode.AddMode(SelectionMode.Cell);}).CellSelectionMode(CellSelectionMode.Box)
; })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example

Copyright © 2001 - 2020 Syncfusion Inc.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Column selection

Column selection can be enabled by setting the SelectionMode property of SelectionSettings as Column. For random column selection, press the **“Ctrl + mouse left click”** and for continuous column selection, press the **“Shift + mouse left click”** on the top of the column header. To unselect selected columns, press the **“Ctrl + mouse left click”** on top of the selected column header.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSelection()
.SelectionType(SelectionType.Multiple)
.SelectionSettings(select => { select.SelectionMode(mode => {
mode.AddMode(SelectionMode.Column); }); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code sample.

OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	148.33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Touch options

While using the grid in a touch device environment, there is an option for multi selection through single tap on the row and it will shows a popup with multi-selection symbol. Tap the icon to enable multi selection in a single tap.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

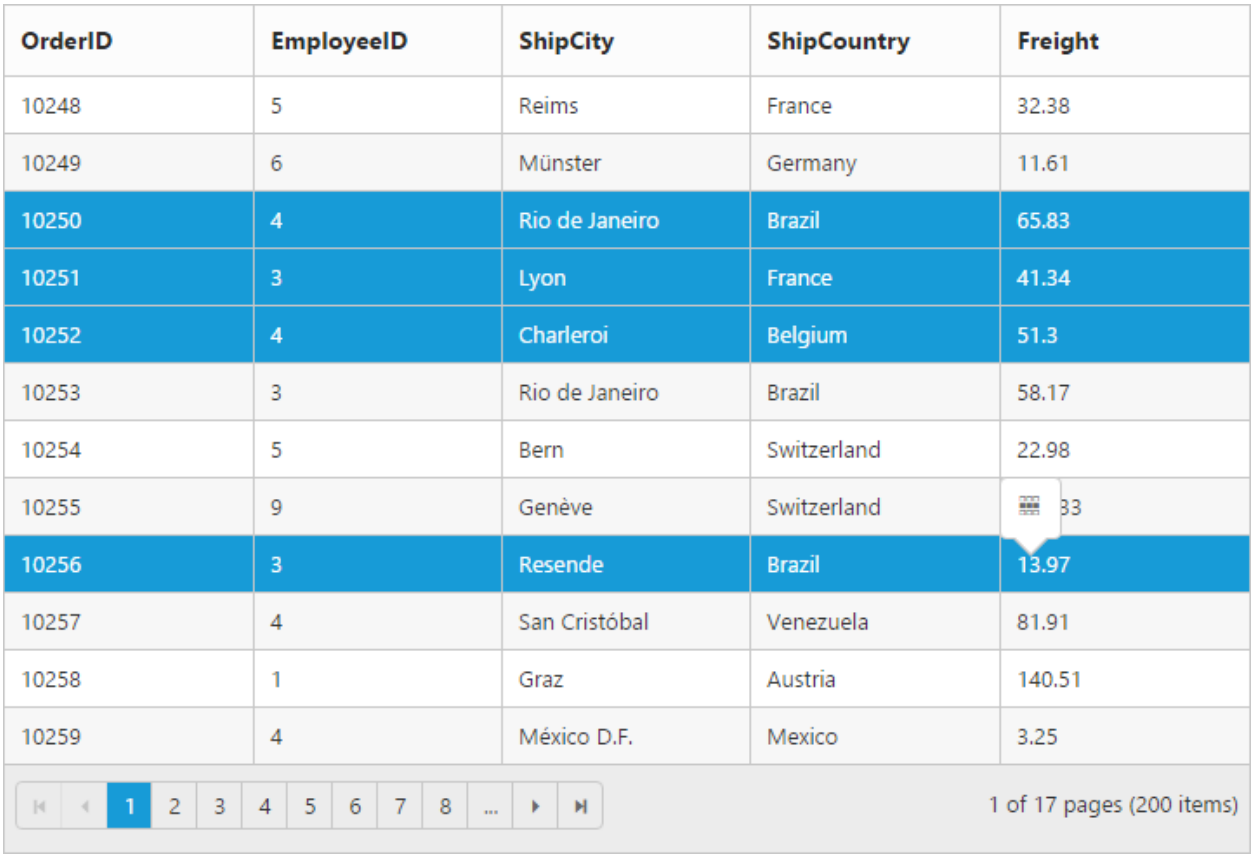
```
.AllowPaging()
.AllowSelection()
.SelectionType(SelectionType.Multiple)
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



OrderID	EmployeeID	ShipCity	ShipCountry	Freight
10248	5	Reims	France	32.38
10249	6	Münster	Germany	11.61
10250	4	Rio de Janeiro	Brazil	65.83
10251	3	Lyon	France	41.34
10252	4	Charleroi	Belgium	51.3
10253	3	Rio de Janeiro	Brazil	58.17
10254	5	Bern	Switzerland	22.98
10255	9	Genève	Switzerland	33
10256	3	Resende	Brazil	13.97
10257	4	San Cristóbal	Venezuela	81.91
10258	1	Graz	Austria	140.51
10259	4	México D.F.	Mexico	3.25

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Toggle selection

The Toggle selection allows to perform selection and unselection of the particular row, cell or column. To enable toggle selection, set EnableToggle property of SelectionSettings as true. If you click on the selected row, cell or column then it will be unselected and vice versa.

Note: If multi selection is enabled, then in first click on any selected row (without pressing Ctrl key), it will clear multi selection and in second click on the same row, it will be unselected.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Selection")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.EnableRowHover(false)
.SelectionSettings(select => { select.EnableToggle(true); })
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("EmployeeID").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
col.Field("Freight").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Selection()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Toolbar

Toolbar can be shown by defining the ShowToolbar property in ToolbarSettings. Toolbar has option to add default items in the ToolbarItems and customized items in the CustomToolbarItems.

Default toolbar items

The following table shows default toolbar items and its action.

Default toolbar items.	Action
Add	Add a new row.
Edit	Edit an existing.
Delete	Delete a row.
Update	Update edited or added row.
Cancel	Cancel edited or added row.
Search	Search text in records.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SyncfusionMvcApplication1;
using SyncfusionMvcApplication1.Models;
namespace SyncfusionMvcApplication1.Controllers
{
public class GridController : Controller
{
//
// GET: /Grid/
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
```

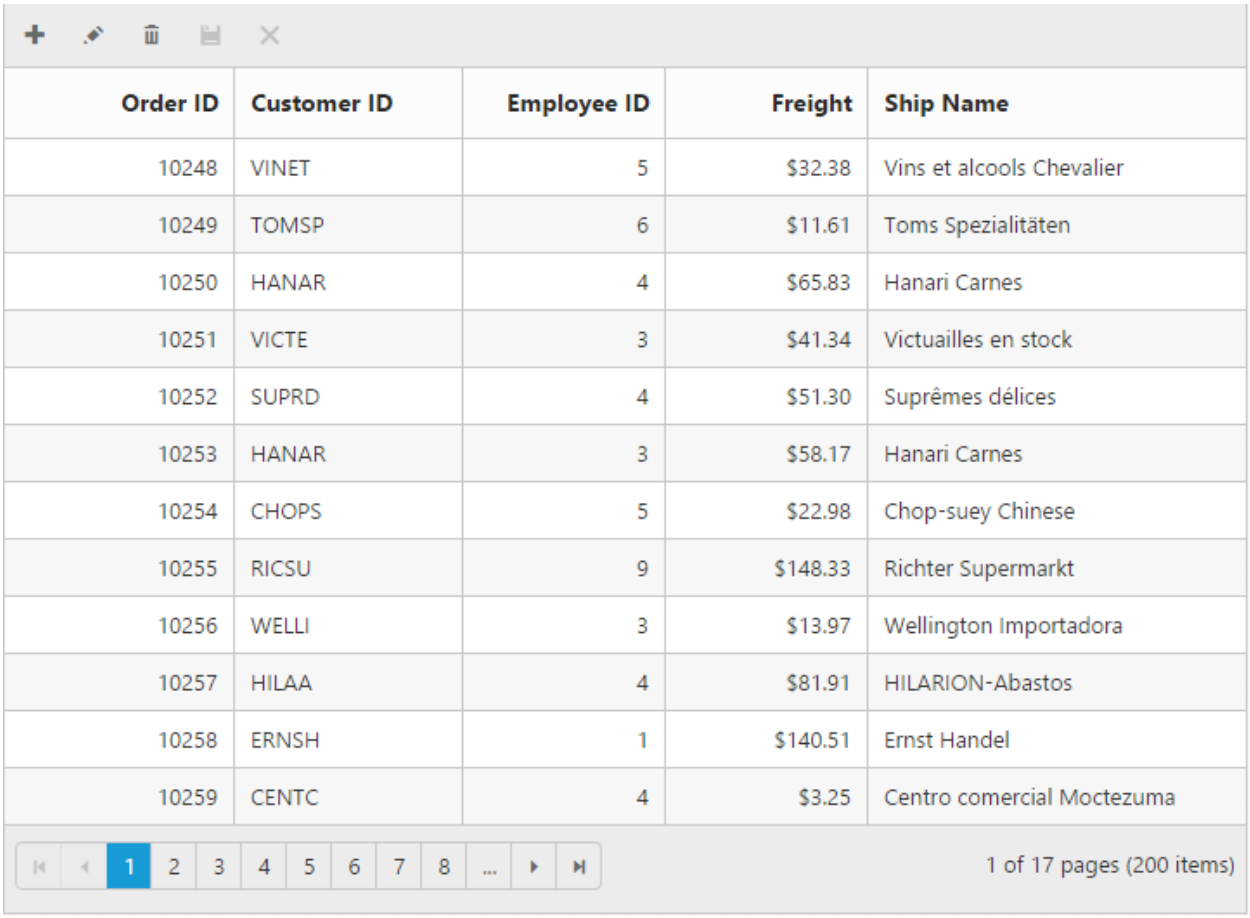
Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
}
}
```

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(90).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Format("{0:C}").Add();
col.Field("ShipName").HeaderText("Ship Name").Width(150).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Order ID	Customer ID	Employee ID	Freight	Ship Name
10248	VINET	5	$32.38	Vins et alcools Chevalier
10249	TOMSP	6	$11.61	Toms Spezialitäten
10250	HANAR	4	$65.83	Hanari Carnes
10251	VICTE	3	$41.34	Victuailles en stock
10252	SUPRD	4	$51.30	Suprêmes délices
10253	HANAR	3	$58.17	Hanari Carnes
10254	CHOPS	5	$22.98	Chop-suey Chinese
10255	RICSU	9	$148.33	Richter Supermarkt
10256	WELLI	3	$13.97	Wellington Importadora
10257	HILAA	4	$81.91	HILARION-Abastos
10258	ERNSH	1	$140.51	Ernst Handel
10259	CENTC	4	$3.25	Centro comercial Moctezuma

Information: The AllowAdding, AllowEditing and AllowDeleting need to be enabled for add, delete, edit and save & cancel in the ToolbarItems. The AllowSearching to be enabled while adding Search in toolbar to perform search action.

Custom toolbar items

Custom toolbar is used to create your own toolbar items in toolbar. It can be added by defining CustomToolbarItems in the ToolbarSettings. Actions for this customized toolbar is defined in the ToolbarClick event.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SyncfusionMvcApplication1;
using SyncfusionMvcApplication1.Models;
namespace SyncfusionMvcApplication1.Controllers
{
public class GridController : Controller
{
//
// GET: /Grid/
public ActionResult GridFeatures()
```

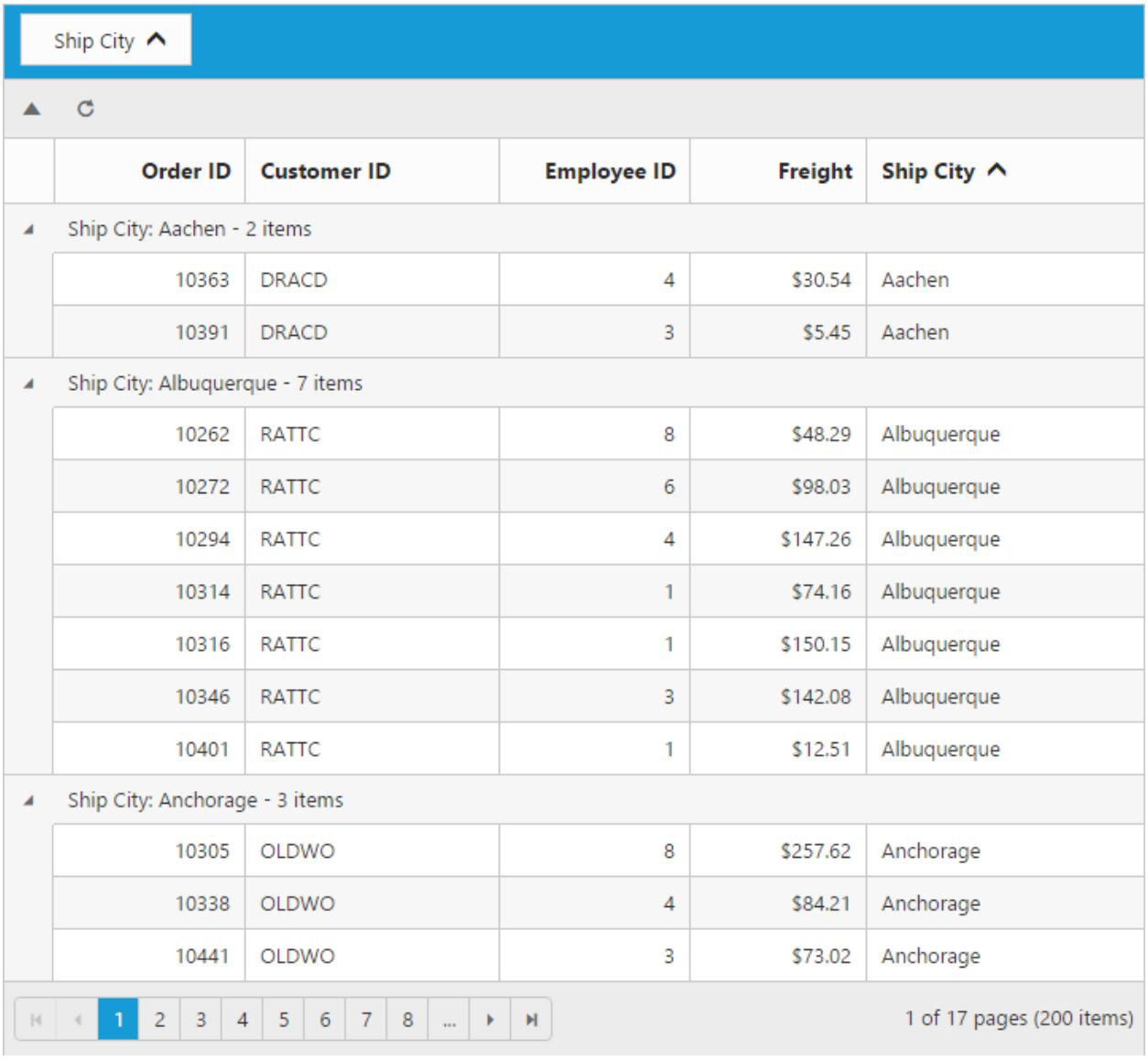
```
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar(true).CustomToolbarItems(new List<object>() {
"Collapse", new Syncfusion.JavaScript.Models.CustomToolbarItem() {
TemplateID = "#Refresh" } });
})
.AllowGrouping()
.GroupSettings(group => { group.GroupedColumns(new List<String>() {
"ShipCity" }); })
.ClientSideEvents(eve => { eve.ToolbarClick("onToolBarClick"); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(100).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
70).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
})
)
<style type="text/css" class="cssStyles">
.Collapse:before {
content: "\e625";
}
.refresh:before {
content: "\e677";
}
</style>
<script id="Refresh" type="text/x-jsrender">
<a class="e-toolbaricons e-icon refresh" />
</script>
<script type="text/javascript">
function onToolBarClick(args) {
var toolbar = $(args.target),
grid = this;
if (toolbar.hasClass("Collapse")) grid.collapseAll(); //collapse Grid using
grid instance, `this` is grid instance
else grid.refreshContent(); //refresh content using grid instance
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```



Web Accessibility

Web Accessibility is achieved in Grid control through the Keyboard Navigation and WAI-ARIA Standard.

Keyboard Navigation

Supported Keyboard Interactions keys with its description are tabulated as follows.

Interaction Keys	Description
Alt + j	Focus the Grid
Insert	Insert record in Grid
Delete	Delete record in Grid
F2	Edit record in Grid

Copyright © 2001 - 2020 Syncfusion Inc.

Enter	Save edited or added
Esc	Cancel add or edit state
PgDn	Go to next Page
PgUp	Go to previous Page
Ctrl +Alt +PgDn	Go to last page
Ctrl + Alt + PgUp	Go to first page
Alt + PgDown	Go to next Pager
Alt + PgUp	Go to previous Pager
Home	Go to first cell
End	Go to last cell
Ctrl + Home	Go to first row
Ctrl + End	Go to last row
Up arrow	Move to up cell selection
Down arrow	Move to down cell selection
Right arrow	Move to right cell selection
Left arrow	Move to left cell selection
Tab	Go to next cell
Shift + tab	Go to previous cell
Alt + DownArrow	Expand selected group
Ctrl + DownArrow	Expand All visible groups
Alt + UpArrow	Collapse selected group
Ctrl + UpArrow	Collapse All visible groups

WAI - ARIA

This helps to enable better user interaction in ejGrid and uses the W3C's Widget Design Patterns in the WAI specification and added customize attributes. Please find the list of ARIA attribute used in Grid.

- grid (role)
- row (role)
- gridcell (role)
- aria-selected (attribute)

Copyright © 2001 - 2020 Syncfusion Inc.

Styling

List of classes and its purposes

To modify the Grid appearance, you need to override default CSS of the grid. Please find the list of CSS classes and its corresponding section in the grid. Also you have an option to create your own custom theme for all the ASP.NET MVC controls using our Theme Studio.

Section	CSS class	Purpose of the CSS class
Root	e-grid	This classes are in this root element (div) of the grid control.
	e-js	
Header	e-gridheader	This is class is added in the root element of header element.
	e-table	This class is added at 'table' of grid header. This CSS class makes table width as 100 %.
	e-columnheader	This class is added at 'tr' of the grid header.
	e-headercell	This class is added in 'th' element of grid header. You can override the background color of header and border color.
	e-headercelldiv	This class is add in div which present 'th' element in header. It is recommended to use the e-headercelldiv to override skeleton of header.
Body	e-gridcontent	This class is added at root of body content. This is to override the background color of body.
	e-table	This class is added to table of content. This CSS class makes the table width as 100 %.
	e-alt_row	This class is added to alternate rows of grid. This is to override the alternate row color of grid.
	e-rowcell	This class is added to all cells in the grid. This is to override cell's appearance and styling.
	e-groupcaption	This class is added to 'td' of group caption which is to change the background color of caption cell.
	e-selectionbackground	This class is added to the row cells of grid. This is override selection.
	e-hover	This class adds to the row of grid when hovering on grid rows.
Pager	e-pager	This class is added to root element of the pager. This to change appearance of background color and color of font.
	e-pagercontainer	This class is added to numeric items of the pager.
	e-parentmsgbar	This class is added to pager info of the pager.
Summary	e-gridfooter	This class is added to root of the summary div.

Copyright © 2001 - 2020 Syncfusion Inc.

	e-gridsummary	This class is added to table of summary. This CSS class makes table width as 100 %.
	e-gridSummaryRows	This class is added to rows of grid summary.
	e-summaryrow	This class is added to cells of summary row. This to override the background color of summary.

Toolbar customization

To customize toolbar, you need to use the toolbar default CSS class to override icon in toolbar.

customize toolbar

Context Menu

Context menu is used to improve the user action with Grid using popup menu. It can be shown by defining the EnableContextMenu property of ContextMenuSettings as true. Context menu has option to add default items in the ContextMenuItems property of ContextMenuSettings and customized items in the CustomContextMenuItems property of ContextMenuSettings.

Default Context Menu items

Please find the below table for default context menu items and its actions.

Section	Context menu items	Action
Header	Sort in Ascending Order	Sort column in Ascending order.
	Sort in Descending Order	Sort column in Descending order.
	Group	Group the current column.
	Ungroup	Ungroup the current column if already grouped.
Body	Add Record	Start Add new record.
	Edit Record	Start Edit in current record.
	Delete Record	Delete the current record.
	Save	Save the record if Add/Edit record is started.
	Cancel	Cancel Added/Edited state.
Pager	Next Page	Go to Next Page.
	Last Page	Go to Last page.
	Previous page	Go to previous page.
	First page	Go to first page.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
```

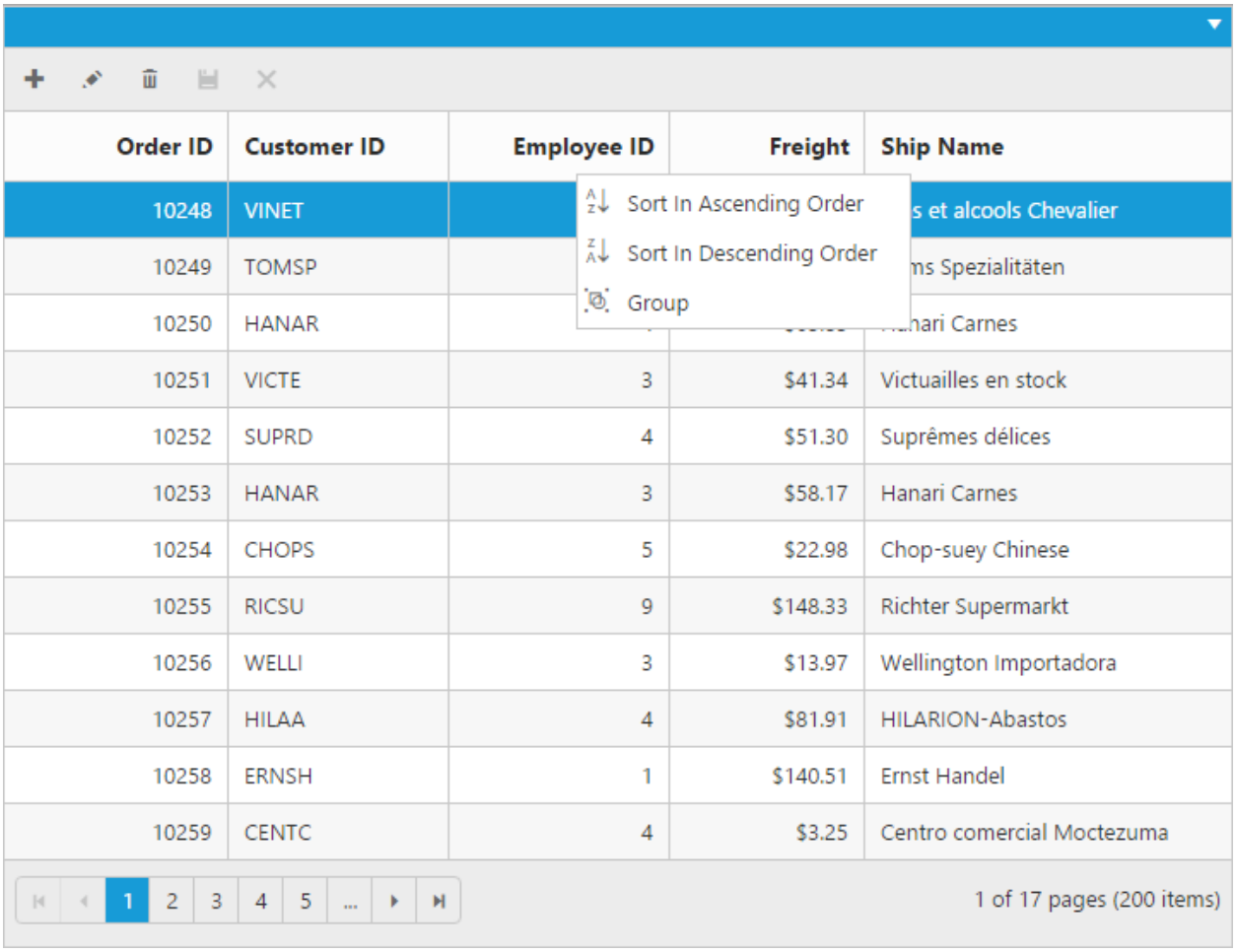
Copyright © 2001 - 2020 Syncfusion Inc.

```
.EditSettings(edit =>
{
edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.ContextMenuSettings(contextMenu =>
{
contextMenu.EnableContextMenu();
})
.AllowPaging()
.AllowSorting()
.AllowGrouping()
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("Freight").Format("{0:c2}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(150).Add();
})
)
```

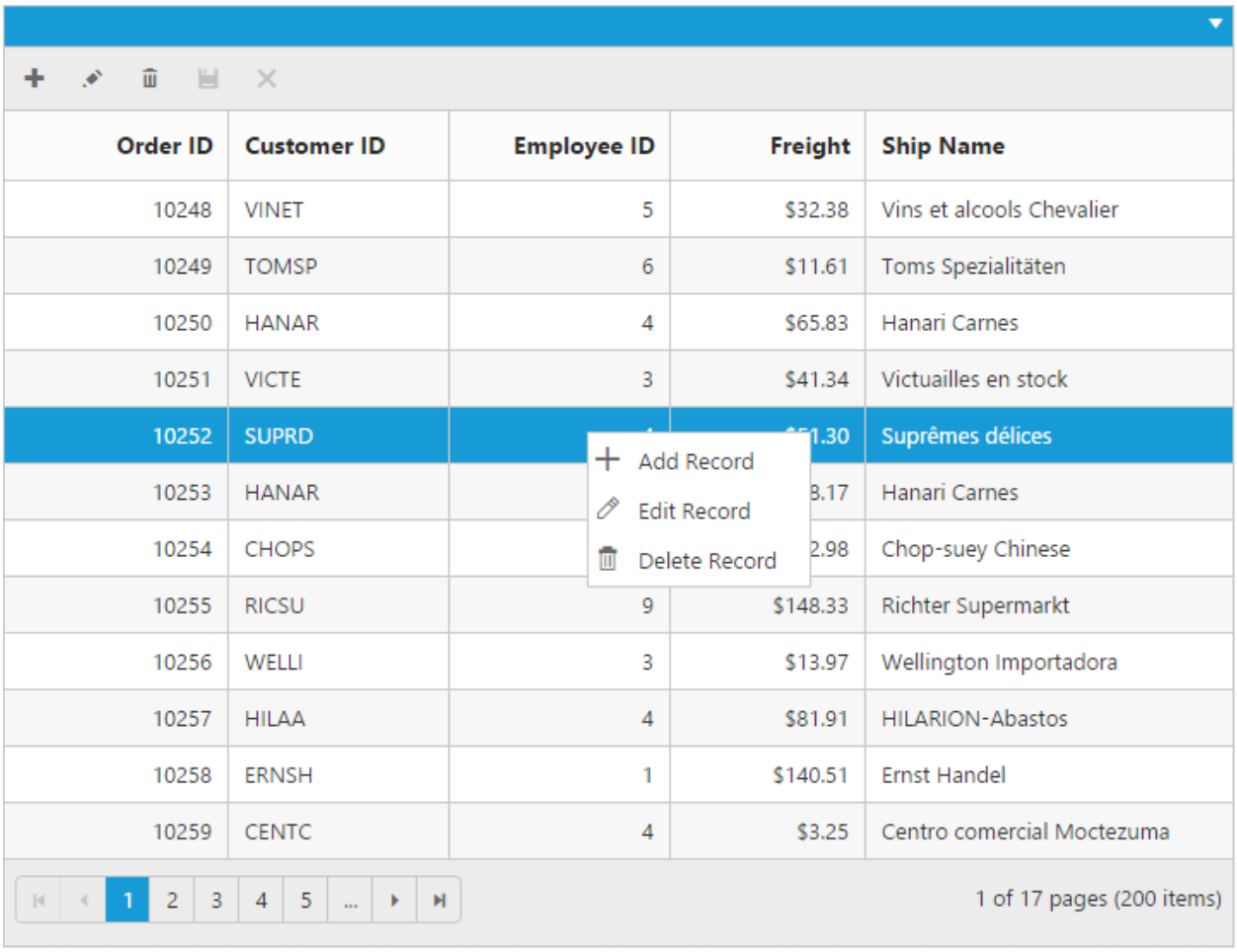
C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Index(){
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

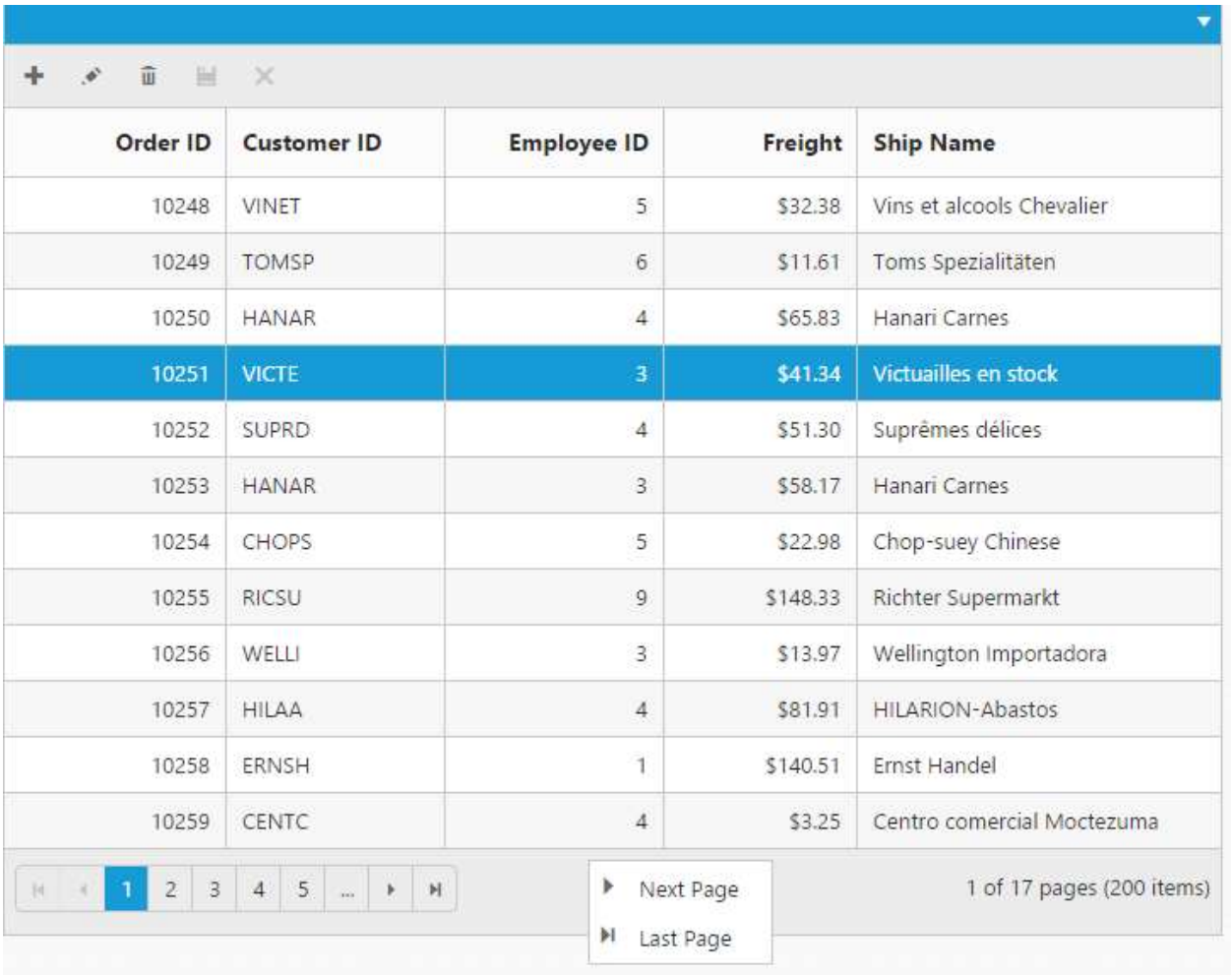


Context menu at header



Context menu at body

Copyright © 2001 - 2020 Syncfusion Inc.



Context menu at pager

Note: AllowGrouping, AllowSorting should be enabled to perform default context menu actions in Grid header. AllowEditing, AllowDeleting and AllowAdding should be enabled to perform default actions in body.

Custom Context Menu

Custom context menu is used to create your own menu item and its action. To add customized context menu items, you need to use the ContextMenuSettings.CustomContextMenuItems property and to bind required actions for this, use the ContextClick event.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ContextMenuSettings(contextMenu =>
{
contextMenu.EnableContextMenu();
contextMenu.DisableDefaultItems();
contextMenu.CustomContextMenuItems(new List<CustomContextMenuItems> { new
CustomContextMenuItems() { Id = "clear", Text = "Clear Selection" }});
})
.AllowPaging()
```

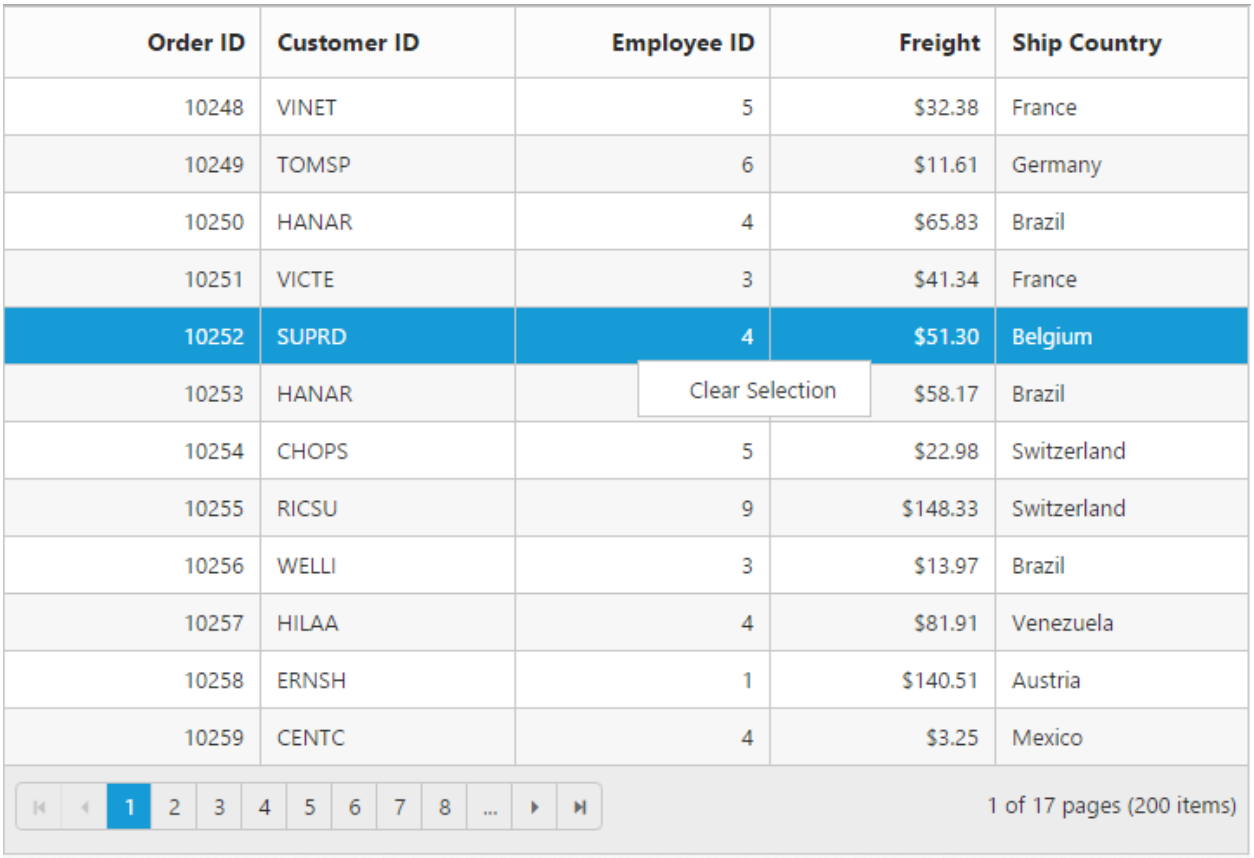
Copyright © 2001 - 2020 Syncfusion Inc.

```
.ClientSideEvents(eve => {eve.ContextClick("context_click");})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("Freight").Format("{0:c2}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(150).Add();
})
)
<script type="text/javascript">
function context_click(args) {
if (args.text == "Clear Selection")
this.clearSelection();
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Index(){
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Sub Context Menu

Sub context menu is used to add customized sub menu to the custom context menu item. To add a sub context menu, you need to use the ContextMenuSettings.SubContextMenu property and to bind required actions for this, use the ContextClick event.

RAZOR

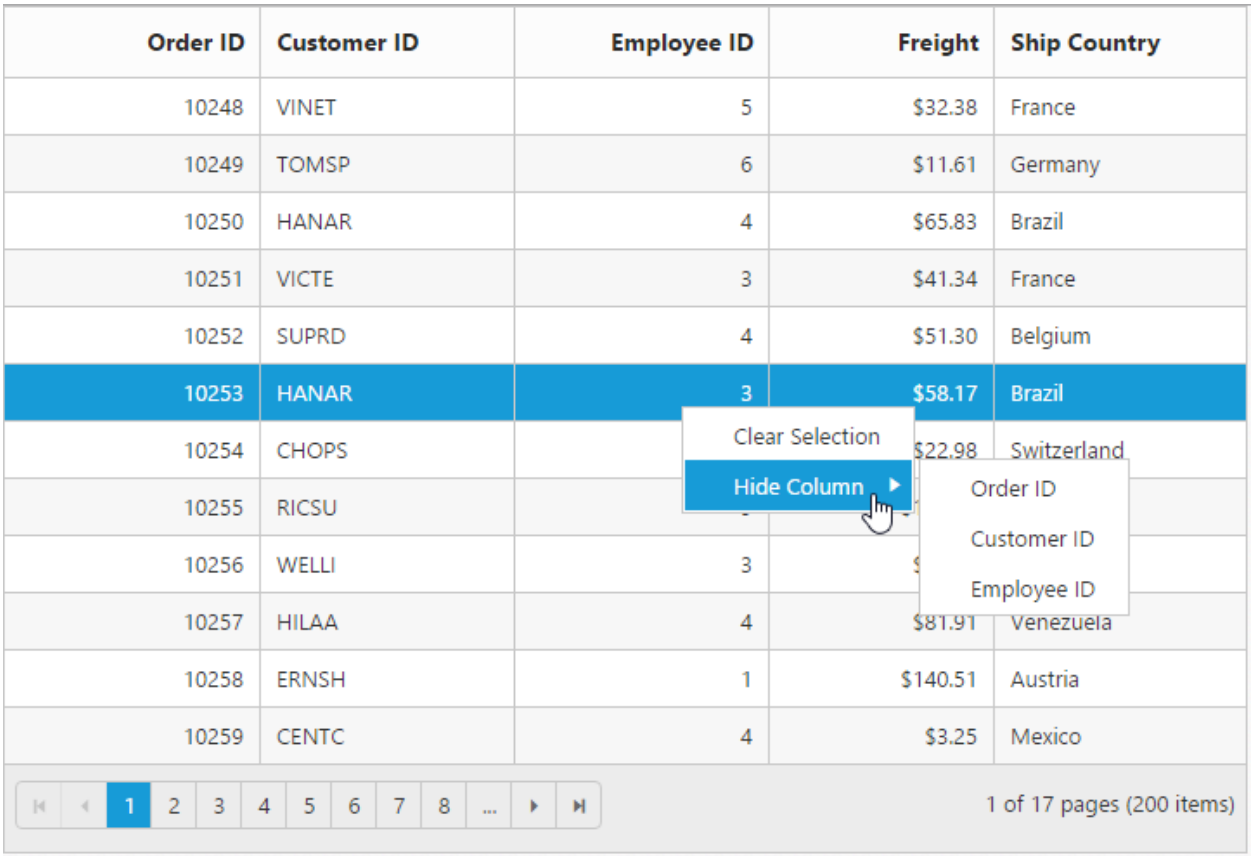
```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ContextMenuSettings(contextMenu =>
{
contextMenu.EnableContextMenu();
contextMenu.DisableDefaultItems();
contextMenu.CustomContextMenuItems(new List<CustomContextMenuItems> { new
CustomContextMenuItems() { Id = "clear", Text = "Clear Selection" }, new
CustomContextMenuItems() { Id = "hide", Text = "Hide Column" } });
contextMenu.SubContextMenu(submenu =>
{
submenu.ContextMenuItem("hide").SubMenu(new List<string>() { "Order ID",
"Customer ID", "Employee ID" }).Add();
});
})
.AllowPaging()
.ClientSideEvents(eve => {eve.ContextClick("context_click");})
.Columns(col =>
{
```

```
col.Field("OrderID").IsPrimaryKey(true).HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("Freight").Format("{0:c2}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(150).Add();
})
)
<script type="text/javascript">
function context_click(args) {
if (args.text == "Clear Selection")
this.clearSelection();
else if (args.text == "Hide Column")
this.hideColumns(args.text);
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Index(){
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Sub Context Menu with Template

On rendering the Sub context menu items, the customized sub menu items is created by using the ContextMenuSettings.SubContextMenu.Template property.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ContextMenuSettings(contextMenu =>
{
contextMenu.EnableContextMenu();
contextMenu.DisableDefaultItems();
contextMenu.CustomContextMenuItems(new List<CustomContextMenuItems> { new
CustomContextMenuItems() { Id = "clear", Text = "Clear Selection" }, new
CustomContextMenuItems() { Id = "hide", Text = "Hide Column" } });
contextMenu.SubContextMenu(submenu =>
{
submenu.ContextMenuItem("hide").Template("#template").Add();
});
})
.AllowPaging()
.ClientSideEvents(eve => {eve.ContextClick("context_click");})
.Columns(col =>
{
col.Field("OrderID").IsPrimaryKey(true).HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("Freight").Format("{0:c2}").HeaderText("Freight").Width(80).TextAl
ign(TextAlign.Right).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(150).Add();
})
)
```

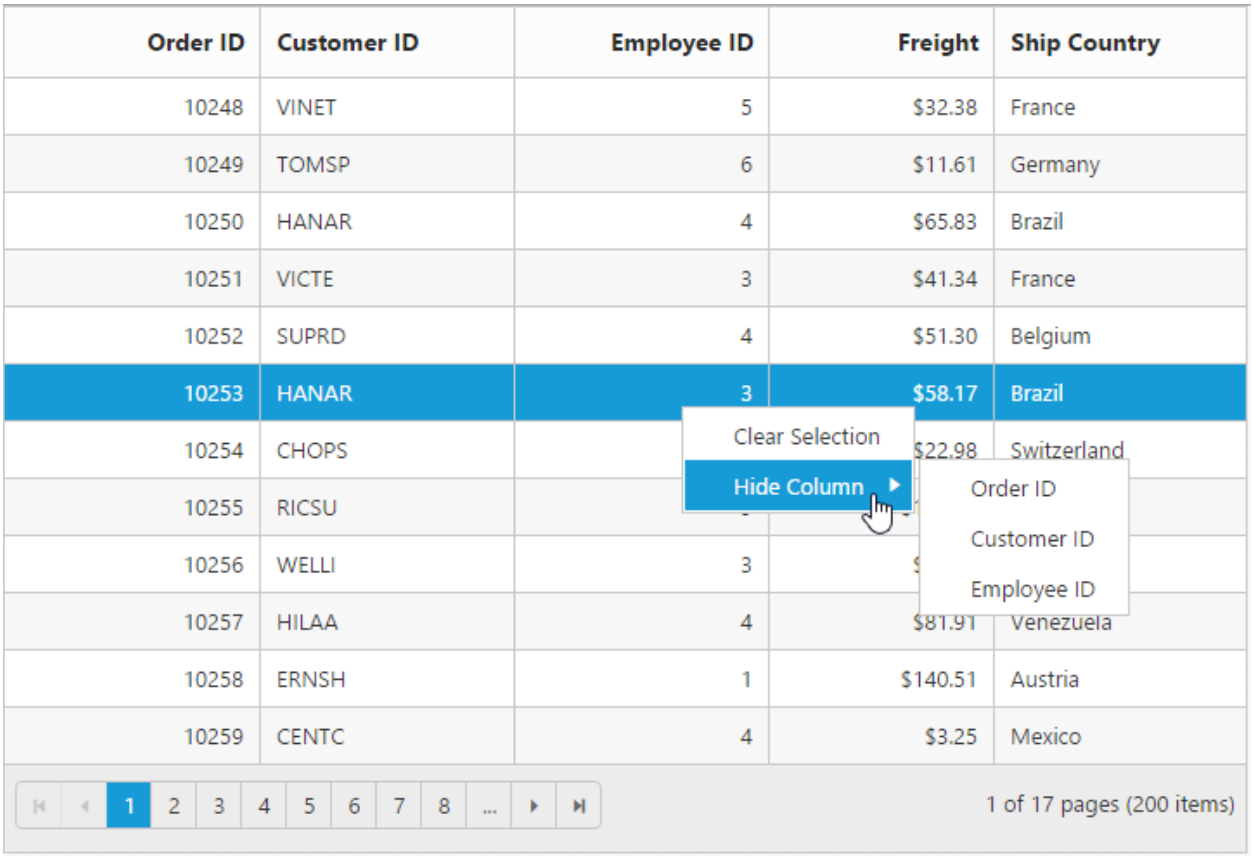
JS

```
<script type="text/javascript">
function context_click(args) {
if (args.text == "Clear Selection")
this.clearSelection();
else if (args.text != "Hide Column")
this.hideColumns(args.text);
}
</script>
<script type= "text/x-jsrender" id=template>
<ul>
<li><a>OrderID</a></li>
<li><a>CustomerID</a></li>
<li><a>EmployeeID</a></li>
</ul>
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class GridController : Controller
{
public ActionResult Index(){
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Hierarchical Bindings

Hierarchical binding can be used to create the Grid with parent and child relation, this facilitate you to view the child records for a particular row by clicking on the Expander button present in first column of each grid row. This can be enabled by defining ChildGrid and QueryString.

The ChildGrid property is used to define the model properties that has to be applied on the child grid. The ChildGrid is the extended class of the base class grid such that it holds all the properties of the grid. The QueryString is a property that has to be specified within the ChildGrid, which defines the relation between the parent and child grid. The QueryString property is used to denote the primaryKey field of the parent grid which is to be mapped with the foreignKey field of the child grid. Based on the mapping, the child grid records are filtered from the table and is bound as datasource for the child grid.

If the foreign key column name differs for parent and child grid then use ForeignKeyField property of child grid. Refer here for more information.

Note: The responsive and exporting support is not applicable for Hierarchical binding.

RAZOR

```
@(Html.EJ().Grid<EmployeeView>("HierarchyGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(85).Add();
col.Field("FirstName").HeaderText("First Name").Width(100).Add();
col.Field("Title").Width(120).Add();
```

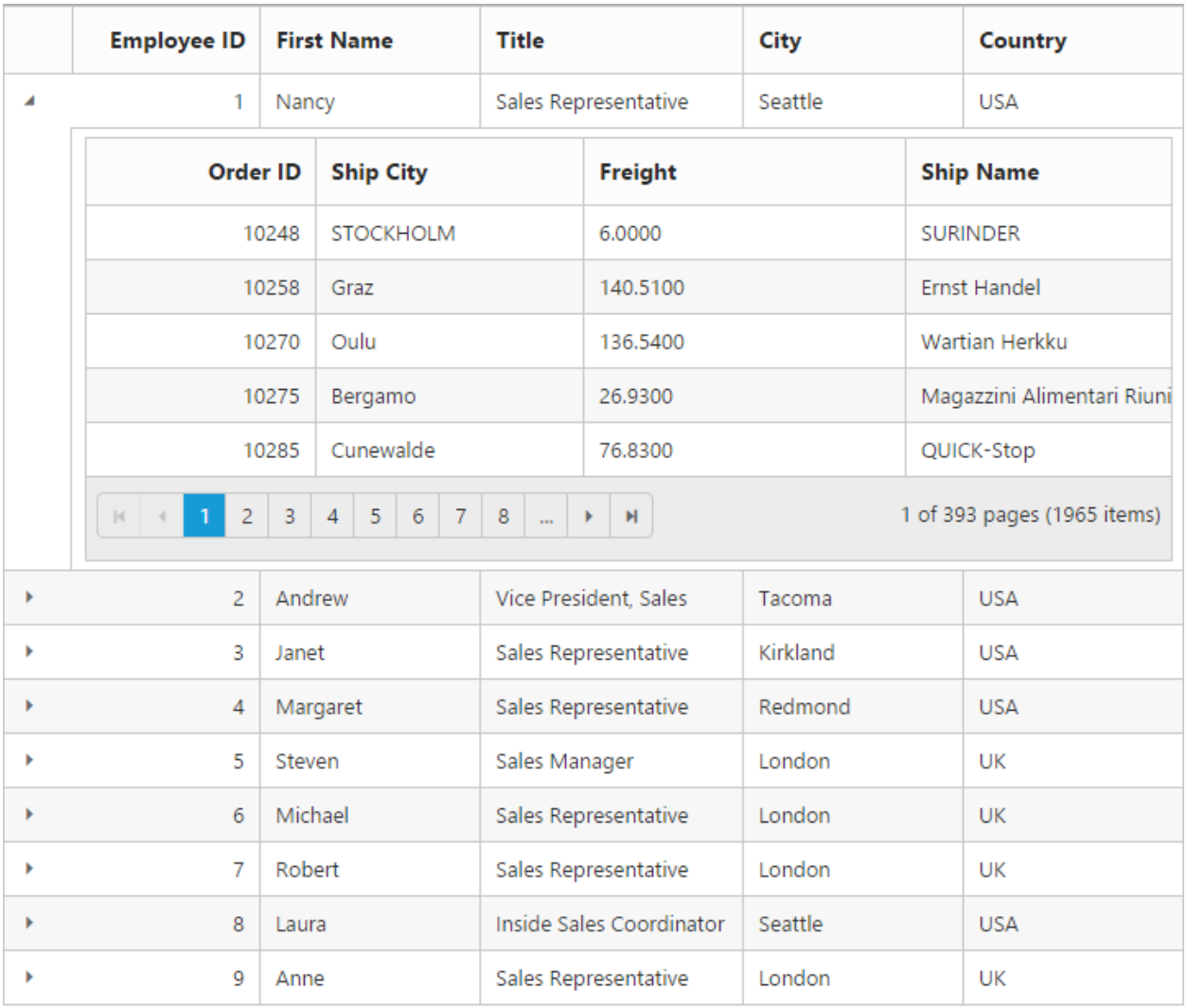
Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("City").Width(100).Add();
col.Field("Country").Width(100).Add();
})
.ChildGrid(child =>
{
child.Datasource("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders")
.QueryString("EmployeeID")
.AllowPaging()
.PageSettings(page => page.PageSize(5))
.Columns(col =>
{
col.Field("OrderID").HeaderText("OrderID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("ShipCity").HeaderText("ShipCity").Width(100).Add();
col.Field("Freight").Width(120).Add();
col.Field("ShipName").Width(100).Add();
});
})
)
```

C#

```
public partial class GridController : Controller
{
//
// GET: /HierarchyGrid/
public ActionResult HierarchyGrid()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Expand or collapse all child's

The grid can able to expand and collapse all the ChildGrid through programmatically using the expandAll and collapseAll method.

RAZOR

```
<button id="expand">expandAll</button>
<button id="collapse">collapseAll</button>
@(Html.EJ().Grid<EmployeeView>("HierarchyGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ChildGrid(child =>
{
child.Datasource((IEnumerable<object>)ViewBag.datasource2)
.QueryString("EmployeeID")
.AllowPaging()
.PageSettings(page => page.PageSize(5))
.Columns(col =>
{
col.Field("OrderID").HeaderText("OrderID").TextAlign(TextAlign.Right).Width(
75).Add();
col.Field("ShipCity").HeaderText("ShipCity").Width(100).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").Width(120).Add();
col.Field("ShipName").Width(100).Add();
});
})
)
<script type="text/javascript">
$("#expand,#collapse").ejButton({
showRoundedCorner: true,
size: "mini",
width: 150,
click: function (args) {
$("#HierarchyGrid").ejGrid(args.model.text); //invokes expandAll &
collapseAll method based on button name
}
});
</script>
```

C#

```
public partial class GridController : Controller
{
//
// GET: /HierarchyGrid/
public ActionResult HierarchyGrid()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
var DataSource2 = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
ViewBag.datasource2 = DataSource2;
return View();
}
}
```

Stacked Headers

The stacked headers helps you to group the logical columns in the Grid. It can be shown by setting the ShowStackedHeader as true and by defining the StackedHeaderRows.

Adding stacked header columns

To stack the columns in stacked header, you need to define the Column property in StackedHeaderColumns with field names of visible columns.

ASPX-CS

```
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ShowStackedHeader()
.StackedHeaderRows(row =>
{
row.StackedHeaderColumns(column =>
{
column.HeaderText("OrderDetails").Column(col =>
{
col.Add("OrderID");
col.Add("OrderDate");
col.Add("Freight");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
column.HeaderText("Ship Details").Column(col =>
{
col.Add("ShipName");
col.Add("ShipCity");
col.Add("ShipCountry");
}).Add();
}).Add();
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Width(80).Add();
col.Field("OrderDate").HeaderText("Order
Date").Width(80).Format("{0:MM/dd/yyyy}").TextAlign(TextAlign.Right).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipName").HeaderText("Ship Name").Width(110).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(110).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(110).Add();
}))
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Order Details			Ship Details		
Order ID	**Order Date**	**Freight**	**Ship Name**	**Ship City**	**Ship Country**
10248	07/04/1996	$32.38	Vins et alcools Chevali	Reims	France
10249	07/05/1996	$11.61	Toms Spezialitäten	Münster	Germany
10250	07/08/1996	$65.83	Hanari Carnes	Rio de Janeiro	Brazil
10251	07/08/1996	$41.34	Victuailles en stock	Lyon	France
10252	07/09/1996	$51.30	Suprêmes délices	Charleroi	Belgium
10253	07/10/1996	$58.17	Hanari Carnes	Rio de Janeiro	Brazil
10254	07/11/1996	$22.98	Chop-suey Chinese	Bern	Switzerland
10255	07/12/1996	$148.33	Richter Supermarkt	Genève	Switzerland
10256	07/15/1996	$13.97	Wellington Importador	Resende	Brazil
10257	07/16/1996	$81.91	HILARION-Abastos	San Cristóbal	Venezuela
10258	07/17/1996	$140.51	Ernst Handel	Graz	Austria
10259	07/18/1996	$3.25	Centro comercial Moct	México D.F.	Mexico
10260	07/19/1996	$55.09	Ottilies Käseladen	Köln	Germany
10261	07/19/1996	$3.05	Que Delícia	Rio de Janeiro	Brazil
10262	07/22/1996	$48.29	Rattlesnake Canyon Gr	Albuquerque	USA
10263	07/23/1996	$146.06	Ernst Handel	Graz	Austria
10264	07/24/1996	$3.67	Folk och fä HB	Bräcke	Sweden

Exporting

The Exporting feature provides support to export Grid data into excel, word and PDF files. To export the grid, the export JavaScript method should be called with export action as parameter. To make it work from grid tool bar the ExcelExport, WordExport and PdfExport toolbar items needs to be added in grid tool bar using the ToolbarItems property of ToolbarSettings which are used to perform exporting. When you click the toolbar exporting icon, it internally invokes the export public method of grid object to export. The code sample for this is as follows.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
col.Field("OrderDate").HeaderText("Order Date").TextAlign(TextAlign.Right).
Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "flat-saffron");
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.docx", false, false, "flat-saffron");
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.pdf", false, false, "flat-saffron");
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Hierarchy grid exporting

The grid will be exported with its child grid. This can be achieved by enabling the IncludeChildGrid property of the respective exporting classes like GridExcelExport, GridWordExport and GridPdfExport and include the dataSource needed for ChildGrid in the GridProperties object after deserializing them. Remaining procedures will be same as the normal grid exporting.

Note: Excel file will be exported in the collapsed state with the expand/collapse icon whereas other file-formats like PDF and Word will be exported in the expanded state.

RAZOR

```
@(Html.EJ().Grid<object>("HierarchyGrid")
.Datasource((IEnumerable<object>)ViewBag.parentData)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("FirstName").HeaderText("First Name").Width(100).Add();
col.Field("Title").Width(120).Add();
col.Field("City").Width(100).Add();
col.Field("Country").Width(100).Add();
})
.ChildGrid(child =>
{
child.Datasource((IEnumerable<object>)ViewBag.childData)
.QueryString("EmployeeID")
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("Freight").TextAlign(TextAlign.Right).Format("{0:C2}").Width(120).
Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
});
})
)
```

C#

```
public class HomeController : Controller
{
public ActionResult Index()
{
var order = new NorthWndDataContext().Orders.Take(100).ToList();
var emp = new NorthWndDataContext().Employees.ToList();
ViewBag.childData = order;
ViewBag.parentData = emp;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthWndDataContext().Employees.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.ChildGrid.DataSource = new
NorthWndDataContext().Orders.Take(100).ToList();
GridExcelExport expo = new GridExcelExport();
expo.IncludeChildGrid = true;
exp.Export(obj, DataSource, expo);
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthWndDataContext().Employees.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.ChildGrid.DataSource = new
NorthWndDataContext().Orders.Take(100).ToList();
GridWordExport expo = new GridWordExport();
expo.IncludeChildGrid = true;
exp.Export(obj, DataSource, expo);
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthWndDataContext().Employees.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.ChildGrid.DataSource = new
NorthWndDataContext().Orders.Take(100).ToList();
GridPdfExport expo = new GridPdfExport();
expo.IncludeChildGrid = true;
expo.Unicode = true;
exp.Export(obj, DataSource, expo);
}
private GridProperties ConvertGridObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GridProperties gridProp = new GridProperties();
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
object value = null;
string serialize = serializer.Serialize(ds.Value);
if (ds.Key == "childGrid")
value = ConvertGridObject(serialize);
else
value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return gridProp;
}
}
```

Note: 1. The dataSource passed to the exporting method is compared with grid column while exporting, when a column is not defined in the grid dataSource, it is obtained as null value and hence throws a null exception. To avoid the above exception ensure that the columns bound to the grid are defined in the grid dataSource before exporting the grid.

Note: 2. To export the dynamic list of objects in the grid use the TypeBuilder concept refer to this link

Server dependencies

Export Helper functions are available in the assembly Syncfusion.EJ.Export, which is available in the Essential Studio builds. Full list of assemblies needed for the grid export is as follows.

1. Syncfusion.EJ
2. Syncfusion.EJ.Export
3. Syncfusion.Linq.Base
4. Syncfusion.Compression.Base
5. Syncfusion.DocIO.Base
6. Syncfusion.XlsIO.Base
7. Syncfusion.PDF.Base

Support export types

Currently server helper function allows following three types of exporting.

1. Word
2. Excel
3. PDF

Server side handlers

In MVC, exporting is achieved by using action controller method. In controller method, grid property is passed as string parameter and you need to deserialize it into the grid Property. By using the Export server method, you can export the grid into excel, PDF and word documents.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
col.Field("OrderDate").HeaderText("Order Date").TextAlign(TextAlign.Right).
Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "flat-saffron");
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.docx", false, false, "flat-saffron");
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.pdf", false, false, "flat-saffron");
}
}
```

On exporting, the default routing path to server-side that contains the action name as ExportToExcel for Excel exporting, ExportToWord for Word exporting and ExportToPdf for PDF exporting. The default controller name in routing path is the grid view page's controller name. For instance, when the grid is rendered in GridFeatures View Page of the Home Controller, then on Excel exporting grid Content the default routing path will be ~/Home/ExportToExcel.

Copyright © 2001 - 2020 Syncfusion Inc.

Export mapper in MVC

The Mappers grid property is used to modify the default routing path during exporting. By using Mappers, you can use any action name in controller for exporting and the action can be in any controller (Need not to be in grid view page controller).

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Mappers(map => map.ExportToExcelAction("ExcelAction")
.ExportToPdfAction("PdfAction").ExportToWordAction("WordAction")
)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
col.Field("OrderDate").HeaderText("Order Date").TextAlign(TextAlign.Right).
Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExcelAction(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "flat-saffron");
}
public void WordAction(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.docx", false, false, "flat-saffron");
}
public void PdfAction(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
exp.Export(obj, DataSource, "Export.pdf", false, false, "flat-saffron");
}
}
```

ColumnTemplate exporting

To export the grid with columnTemplate we have to set the IsTemplateColumnIncluded as true in the parameter of the export method. You can handle the template elements in server-side event while exporting the grid to various files such as Excel, PDF and Word.

The server-side events available in template column exporting and its argument types are listed in the following table.

Event Name	Argument	Description
ExcelColumnTemplateInfo	currentCell, Row	It returns the current cell and row of excel sheet.
WordColumnTemplateInfo	currentCell, Row	It returns the current cell and row of word.
PdfColumnTemplateInfo	currentCell, Row	It returns the current cell and row of PDF.
ExcelChildGridInfo	current row, row data, GridProperties	Customize the cell and child grid
PdfChildGridInfo	current row, row data, GridProperties	Customize the cell and child grid
WordChildGridInfo	current row, row data, GridProperties	Customize the cell and child grid

You can modify the template column of exporting files using server events. The code sample for this is as follows.

RAZOR

```
<script type="text/x-jsrender" id="columnTemplate">
<a href="https://www.syncfusion.com">{{:FirstName}}</a>
</script>
@(Html.EJ().Grid<EmployeeView>("ColumnTemplate")
.Datasource((IEnumerable<object>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowPaging()
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Mappers(map =>
map.ExportToExcelAction("ColumnTemplateExportToExcel").ExportToPdfAction("Co
lumnTemplateExportToPdf").ExportToWordAction("ColumnTemplateToWord"))
.Columns(col =>
{
col.HeaderText("First
Name").Template("#columnTemplate").TextAlign(TextAlign.Center).Width(80).Add
();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(100).Add();
col.Field("LastName").HeaderText("Last Name").Width(100).Add();
col.Field("BirthDate").HeaderText("Birth
Date").TextAlign(TextAlign.Right).Width(100).Format("{0:MM/dd/yyyy}").Add();
col.Field("Country").Width(100).HeaderText("Country").Add();
})
)
```

C#

```
public partial class GridController : Controller
{
public void ColumnTemplateExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource =  new NorthwindDataContext().EmployeeViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.ExcelColumnTemplateInfo = templateInfo;
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
true, "flat-saffron");
}
public void ColumnTemplateToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource =  new NorthwindDataContext().EmployeeViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.WordColumnTemplateInfo = WordTemplateInfo;
exp.Export(obj, DataSource, "Export.docx", false, true, "flat-saffron");
}
public void ColumnTemplateExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.PdfColumnTemplateInfo = PdfTemplateInfo;
exp.Export(obj, DataSource, "Export.pdf", false, true, "flat-saffron");
}
public void templateInfo(object currentCell, object row)
{
IRange range = (IRange)currentCell;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
object templates;
foreach (var data in row.GetType().GetProperties())
{
if (range.Value.Contains(data.Name))
{
templates = row.GetType().GetProperty(data.Name).GetValue(row, null);
range.Value = range.Value.Replace(data.Name, templates.ToString());
var regex = new Regex("<a
[^>]*href=(?:'(?<href>.*?)'|(?:\"(?<href>.*?)\")",
RegexOptions.IgnoreCase);
var urls = regex.Matches(range.Value.ToString()).OfType<match>().Select(m =>
m.Groups["href"].Value).SingleOrDefault();
IHyperLink hyperlink = (range.Parent as
Syncfusion.XlsIO.Implementation.WorksheetImpl).HyperLinks.Add(range);
hyperlink.Type = ExcelHyperLinkType.Url;
hyperlink.TextToDisplay = templates.ToString();
hyperlink.Address = urls;
}
}
}
public void WordTemplateInfo(object currentCell, object row)
{
WTableCell wCell = (WTableCell)currentCell;
object templates;
foreach (var data in row.GetType().GetProperties())
{
if (wCell.LastParagraph.Text.ToString().Contains(data.Name))
{
templates = row.GetType().GetProperty(data.Name).GetValue(row, null);
var regex = new Regex("<a
[^>]*href=(?:'(?<href>.*?)'|(?:\"(?<href>.*?)\")",
RegexOptions.IgnoreCase);
var urls = regex.Matches(wCell.LastParagraph.Text).OfType<Match>().Select(m
=> m.Groups["href"].Value).SingleOrDefault();
wCell.LastParagraph.Text = "";
IWField field = wCell.LastParagraph.AppendHyperlink(urls,
templates.ToString(), HyperlinkType.WebLink);
}
}
}
public void PdfTemplateInfo(object currentCell, object row)
{
Syncfusion.Pdf.Grid.PdfGridCell range =
(Syncfusion.Pdf.Grid.PdfGridCell)currentCell;
object templates;
range.Value = Uri.UnescapeDataString(range.Value.ToString());
foreach (var data in row.GetType().GetProperties())
{
if (range.Value.ToString().Contains(data.Name))
{
templates = row.GetType().GetProperty(data.Name).GetValue(row, null);
var regex = new Regex("<a
[^>]*href=(?:'(?<href>.*?)'|(?:\"(?<href>.*?)\")",
RegexOptions.IgnoreCase);
var urls = regex.Matches(range.Value.ToString()).OfType<Match>().Select(m =>
m.Groups["href"].Value).SingleOrDefault();
RectangleF rectangle = new RectangleF(10, 40, 30, 30);
```

```
PdfUriAnnotation uriAnnotation = new PdfUriAnnotation(rectangle, urls);
uriAnnotation.Text = templates.ToString();
range.Value = uriAnnotation;
}
}
}
}
```

DetailTemplate exporting

To export the grid with detail template we have to set the IncludeDetailRow as true in the parameter of the export method. You can handle template elements using server-side event while exporting grid to various files such as Excel, PDF and Word.

The server-side events available in detail template exporting and its argument types are listed in the following table.

Event Name	Argument	Description
ExcelDetailTemplateInfo	currentCell, Row	It returns the current cell and row of excel sheet.
WordDetailTemplateInfo	currentCell, Row	It returns the current cell and row of word.
PdfDetailTemplateInfo	currentCell, Row	It returns the current cell and row of PDF.

You can modify the detailTemplate of exporting files using server events. The code sample for this is as follows:

RAZOR

```
<script id="tabGridContents" type="text/x-jsrender">
<b> More Details: </b> <br /><br /> <b class='detail'>Last Name</b> -
{{:LastName}} <br /> <br /> <b class='detail'>Home Phone</b> -
{{:HomePhone}} <br /> <br /> <b class='detail'>Extension No.</b> -
{{:Extension}}
</script>
@(Html.EJ().Grid<EmployeeView>("DetailTemplate")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("FirstName").HeaderText("First Name").Width(100).Add();
col.Field("Title").Width(120).Add();
col.Field("City").Width(100).Add();
col.Field("Country").Width(100).Add();
})
.DetailsTemplate("#tabGridContents")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.dataSource = DataSource;
return View();
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
GridExcelExport exp2 = new GridExcelExport() { IncludeDetailRow = true,
Theme = "flat-saffron", FileName = "Export.xlsx" };
obj.ExcelDetailTemplateInfo = templateInfo;
exp.Export(obj, DataSource, exp2);
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.WordDetailTemplateInfo = WordDetailTemplateInfo;
GridWordExport exp1 = new GridWordExport() { IncludeDetailRow = true, Theme
= "flat-saffron", FileName = "Export.docx" };
exp.Export(obj, DataSource, exp1);
}
public void ExportToPDF(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
obj.PdfDetailTemplateInfo = PdfDetailTemplateInfo;
GridPdfExport exp3 = new GridPdfExport() { IncludeDetailRow = true, Theme =
"flat-saffron", FileName = "Export.pdf" };
exp.Export(obj, DataSource, exp3);
}
public void templateInfo(object currentCell, object row)
{
IRange range = (IRange)currentCell;
object templates;
foreach (var data in row.GetType().GetProperties())
{
if (range.Value.Contains(data.Name))
{
templates = row.GetType().GetProperty(data.Name).GetValue(row, null);
range.Value = range.Value.Replace(data.Name, templates.ToString());
var charsToRemove = new string[] { '{', '}', '<b>', ':', '</b>', '<br />',
'style', '=', 'class', '</div>', '<p>', '</p>', 'detail', '<b', '>', };
foreach (var c in charsToRemove)
{
range.Value = range.Value.ToString().Replace(c, string.Empty);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
range.HorizontalAlignment = ExcelHAlign.HAlignCenter;
}
}
}
public void WordDetailTemplateInfo(object currentCell, object row)
{
WTableCell wCell = (WTableCell)currentCell;
object templates;
foreach (var data in row.GetType().GetProperties())
{
if (wCell.LastParagraph.Text.ToString().Contains(data.Name))
{
templates = row.GetType().GetProperty(data.Name).GetValue(row, null);
wCell.LastParagraph.Text =
wCell.LastParagraph.Text.ToString().Replace(data.Name,
templates.ToString());
var charsToRemove = new string[] { '{', '}', '<b>', ':', '</b>', '<br />',
'style', '=', 'class', '</div>', '<p>', '</p>', 'detail', '<b', '>', };
foreach (var c in charsToRemove)
{
wCell.LastParagraph.Text  = wCell.LastParagraph.Text.ToString().Replace(c,
string.Empty); //
}
}
}
}
public void PdfDetailTemplateInfo(object currentCell, object row)
{
Syncfusion.Pdf.Grid.PdfGridCell range =
(Syncfusion.Pdf.Grid.PdfGridCell)currentCell;
object templates;
foreach (var data in row.GetType().GetProperties())
{
if (range.Value.ToString().Contains(data.Name))
{
templates = row.GetType().GetProperty(data.Name).GetValue(row, null);
range.Value = range.Value.ToString().Replace(data.Name,
templates.ToString());
var charsToRemove = new string[] { '{', '}', '<b>', ':', '</b>', '<br />',
'style', '=', 'class', '</div>', '<p>', '</p>', 'detail', '<b', '>', };
foreach (var c in charsToRemove)
{
range.Value = range.Value.ToString().Replace(c, string.Empty);
}
}
}
}
private GridProperties ConvertGridObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GridProperties gridProp = new GridProperties();
foreach (KeyValuePair<string, object> data in div)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var property = gridProp.GetType().GetProperty(data.Key,
BindingFlags.Instance | BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(data.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
return gridProp;
}
}
```

Exporting with Custom Summary

In Exporting, custom summary needs to be handled using QueryCustomSummaryInfo server-side event.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.ShowSummary()
.SummaryRow(row =>
{
row.Title("Currency").SummaryColumns(col => {
col.SummaryType(SummaryType.Custom).CustomSummaryValue("currency").DisplayCo
lumn("Freight").Format("{0:C2}").Add();
}).Add();
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("OrderDate").HeaderText("Order
Date").Width(80).TextAlign(TextAlign.Right).Format("{0:MM/dd/yyyy}").Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipName").HeaderText("Ship Name").Width(110).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(90).Add();
}))
```

C#

```
namespace Grid.Controllers
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
GridExcelExport GridExp = new GridExcelExport();
GridExp.QueryCustomSummaryInfo = SummaryCellInfo;
GridExp.Theme = "flat-saffron";
GridExp.FileName = "Export.xlsx";
var DataSource = new NorthwindDataContext().OrdersViews.Take(100).ToList();
GridProperties obj = ConvertGridObject(GridModel);
exp.Export(obj, DataSource, GridExp);
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
GridWordExport GridExp = new GridWordExport();
GridExp.QueryCustomSummaryInfo = SummaryCellInfo;
GridExp.Theme = "flat-saffron";
GridExp.FileName = "Export.docx";
var DataSource = new NorthwindDataContext().OrdersViews.Take(100).ToList();
GridProperties obj = ConvertGridObject(GridModel);
exp.Export(obj, DataSource, GridExp);
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
GridPdfExport GridExp = new GridPdfExport();
GridExp.QueryCustomSummaryInfo = SummaryCellInfo;
GridExp.Theme = "flat-saffron";
GridExp.FileName = "Export.pdf";
var DataSource = new NorthwindDataContext().OrdersViews.Take(100).ToList();
GridProperties obj = ConvertGridObject(GridModel);
exp.Export(obj, DataSource, GridExp);
}
private object SummaryCellInfo(IQueryable arg1, SummaryColumn arg2)
{
var rs = 0; double value = 0;
if (arg2.DisplayColumn == "Freight")
{
rs = 100000;
value = 0.017;
}
return (rs * value);
}
private GridProperties ConvertGridObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
GridProperties gridProp = new GridProperties();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key, BindingFlags.Instance
| BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(ds.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
}
}
return gridProp;
}
}
}
```

JS

```
<script type="text/javascript">
function currency() {
var rs = 100000;
var value = 0.017;
return (rs * value);
}
</script>
```

Multiple exporting

The AllowMultipleExporting property allows you to export multiple grids into the same file. Once you enable the AllowMultipleExporting, grid properties of all the grid which are available in current page are passed as string array parameter to controller action method.

In controller action method you are able to export all the grids available in the current page. The code sample for this is as follows.

RAZOR

```
<div class="label1">
Employee Grid
</div>
@(Html.EJ().Grid<EmployeeView>("MasterGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource1)
.SelectedRowIndex(0)
.AllowMultipleExporting()
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Mappers(map =>
map.ExportToExcelAction("MultipleExportToExcel").ExportToPdfAction("Multiple
ExportToPdf").ExportToWordAction("MultipleExportToWord"))
.Columns(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(100).Add();
col.Field("FirstName").HeaderText("First Name").Width(100).Add();
col.Field("LastName").HeaderText("Last Name").Width(100).Add();
col.Field("Title").HeaderText("Title").Width(90).Add();
col.Field("BirthDate").HeaderText("Birth
Date").TextAlign(TextAlign.Right).Width(100).Format("{0:MM/dd/yyyy}").Add();
col.Field("Country").Width(120).HeaderText("Country").Add();
})
)
<div class="label1">
Orders Grid
</div>
@(Html.EJ().Grid<OrdersView>("DetailGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource2)
.AllowPaging(false)
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(125).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(100).Add();
col.Field("Freight").Width(100).Format("{0:C}").TextAlign(TextAlign.Right).A
dd();
col.Field("ShipCity").HeaderText("Ship City").Width(150).Add();
col.Field("OrderDate").HeaderText("Order
Date").TextAlign(TextAlign.Right).Width(100).Format("{0:MM/dd/yyyy}").Add();
col.Field("ShipCountry").Width(100).HeaderText("Ship Country").Add();
})
)
```

C#

```
// Excel export
public void MultipleExportToExcel(string[] GridModel)
{
ExcelExport exp = new ExcelExport();
var EmployeeData = new
NorthwindDataContext().EmployeeViews.Take(5).ToList();
var OrderData = new NorthwindDataContext().OrdersViews.Take(5).ToList();
bool initial = true;
IWorkbook book = null;
foreach(string gridProperty in GridModel)
{
GridProperties gridProp =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), gridProperty);
if (initial)
{
book = exp.Export(gridProp, EmployeeData, "Export.xlsx",
ExcelVersion.Excel2010, true, true, "flat-saffron", true);
initial = false;
}
else
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
exp.Export(gridProp, OrderData, "Export.xlsx", ExcelVersion.Excel2010, true,
true, "flat-saffron", false, book, MultipleExportType.AppendToSheet, "Second
Grid");
}
}
}
// Word export
public void MultipleExportToWord(string[] GridModel)
{
WordExport exp = new WordExport();
var EmployeeData = new
NorthwindDataContext().EmployeeViews.Take(5).ToList();
var OrderData = new NorthwindDataContext().OrdersViews.Take(5).ToList();
IWordDocument document = null;
bool initial = true;;
foreach(string gridProperty in GridModel)
{
GridProperties gridProp =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), gridProperty);
if (initial)
{
document = exp.Export(gridProp, EmployeeData, "Export.docx", true, true,
"flat-saffron", true);
initial = false;
} else
{
exp.Export(gridProp, OrderData, "Export.docx", true, true, "flat-saffron",
false, document, "Second Grid");
}
}
}
// PDF export
public void MultipleExportToPdf(string[] GridModel)
{
PdfExport exp = new PdfExport();
var EmployeeData = new
NorthwindDataContext().EmployeeViews.Take(5).ToList();
var OrderData = new NorthwindDataContext().OrdersViews.Take(5).ToList();
PdfDocument document = null;
bool initial = true;
foreach(string gridProperty in GridModel)
{
GridProperties gridProp =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), gridProperty);
if (initial)
{
document = exp.Export(gridProp, EmployeeData, "Export.pdf", true, true,
"flat-saffron", true);
initial = false;
} else
{
exp.Export(gridProp, OrderData, "Export.pdf", true, true, "flat-saffron",
false, document, "Second Grid");
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

DataTable exporting

The grid has support to export DataTable type datasource. This support helps when a grid has no view model, and a DataTable is bound to a grid.

RAZOR

```
@(Html.EJ().Grid<object>("DataTableGrid")
.Datasource((DataTable)ViewBag.dataTable)
.AllowPaging()
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Columns(col =>
{
col.Field("ProductID").HeaderText("Product
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("ProductName").HeaderText("Product Name").Width(80).Add();
col.Field("SupplierID").HeaderText("Supplier
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CategoryID").HeaderText("Category ID").Width(80).Add();
})
)
```

C#

```
public class GridController : Controller
{
DataTable _dTable = new DataTable();
List<Product> data = new List<Product>();
public void GetGridDT()
{
var connection =
ConfigurationManager.ConnectionStrings["NORTHWNDConnectionString"].Connectio
nString;
using (var dataAdapter = new SqlDataAdapter("SELECT * from Products",
connection))
{
dataAdapter.Fill(_dTable);
}
}
public ActionResult GridFeatures()
{
GetGridDT();
ViewBag.dataTable = _dTable;
return View();
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
GetGridDT();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataTable DataSource = _dTable;
GridProperties properties =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
exp.Export(properties, DataSource, "Export.xlsx", ExcelVersion.Excel2010,
false, false, "flat-saffron");
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
GetGridDT();
DataTable DataSource = _dTable;
GridProperties properties =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
exp.Export(properties, DataSource, "Export.docx", false, false, "flat-
saffron");
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
GetGridDT();
DataTable DataSource = _dTable;
GridProperties properties =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
exp.Export(properties, DataSource, "Export.pdf", false, false, "flat-
saffron");
}
}
```

Export required grid in single file

You can export required grids in single file using the ej.Grid.exportAll method. This method can be used with export action and array of jQuery selector of grid which need to be export. The code sample for it is as follows:

JS

```
$('#exportAll').click(function(){
ej.Grid.exportAll('MultipleExportToExcel',['Grid1', 'Grid2']);
});
```

List of properties ignored on export

Following are the list of properties that are excluded during grid export, to reduce the unwanted data transfer to server.

- dataSource
- query
- rowTemplate
- detailsTemplate
- pageSettings
- enableRTL
- cssClass

Copyright © 2001 - 2020 Syncfusion Inc.

Export only visible records

By default the pageSettings is ignored in export to facilitate all pages export. To achieve current visible page record export, the pageSettings should be removed from ignore list using the following code.

The sample for this is as follows:

JS

```
var grid = $('#Grid').ejGrid('instance');
grid.ignoreOnExport.splice(grid.ignoreOnExport.indexOf('pageSettings'), 1);
```

On server before calling the Export function, the data source should be processed using the DataOperations's execute function.

C#

```
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties properties
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
var dataSource = new DataOperations().Execute(GridModel, DataSource);
exp.Export(properties, dataSource, "Export.xlsx", ExcelVersion.Excel2010,
false, false, "flat-saffron");
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties properties
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
var dataSource = new DataOperations().Execute(GridModel, DataSource);
exp.Export(properties, dataSource, "Export.docx", false, false, "flat-
saffron");
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties properties =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
var dataSource = new DataOperations().Execute(GridModel, DataSource);
exp.Export(properties, dataSource, "Export.pdf", false, false, "flat-
saffron");
}
```

Local data

By default, client data source is not sent to server to prevent the unwanted data transfer since all data origin is in server. In case, if you don't want to query the data source again for exporting in server, the

Copyright © 2001 - 2020 Syncfusion Inc.

grid's client dataSource can be sent to server on export PostBack by removing the dataSource property in grid's ignore list. The code sample for this is as follows.

JS

```
var grid = $('#GridId').ejGrid('instance');
grid.ignoreOnExport.splice(grid.ignoreOnExport.indexOf('dataSource'), 1);
```

Themes

The grid export have 13 theme option to match with our built-in control themes. They are as follows.

Enum	Equivalent string input
ExportTheme.FlatAzure	default-theme
ExportTheme.FlatSaffron	flat-saffron
ExportTheme.FlatLime	flat-lime
ExportTheme.FlatDarkAzure	flat-azure-dark
ExportTheme.FlatDarkSaffron	flat-saffron-dark
ExportTheme.FlatDarkLime	flat-lime-dark
ExportTheme.GradientAzure	gradient-azure
ExportTheme.GradientSaffron	gradient-saffron
ExportTheme.GradientLime	gradient-lime
ExportTheme.GradientDarkAzure	gradient-azure-dark
ExportTheme.GradientDarkSaffron	gradient-saffron-dark
ExportTheme.GradientDarkLime	gradient-lime-dark
ExportTheme.BootstrapTheme	bootstrap-theme
ExportTheme.None	none

Also, it has `none` option which will export the grid without any theme. The desired theme name can be passed as a parameter to `Export` method and the code sample for this is as follows.

C#

```
public partial class GridController : Controller
{
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties),GridModel);
```

Copyright © 2001 - 2020 Syncfusion Inc.

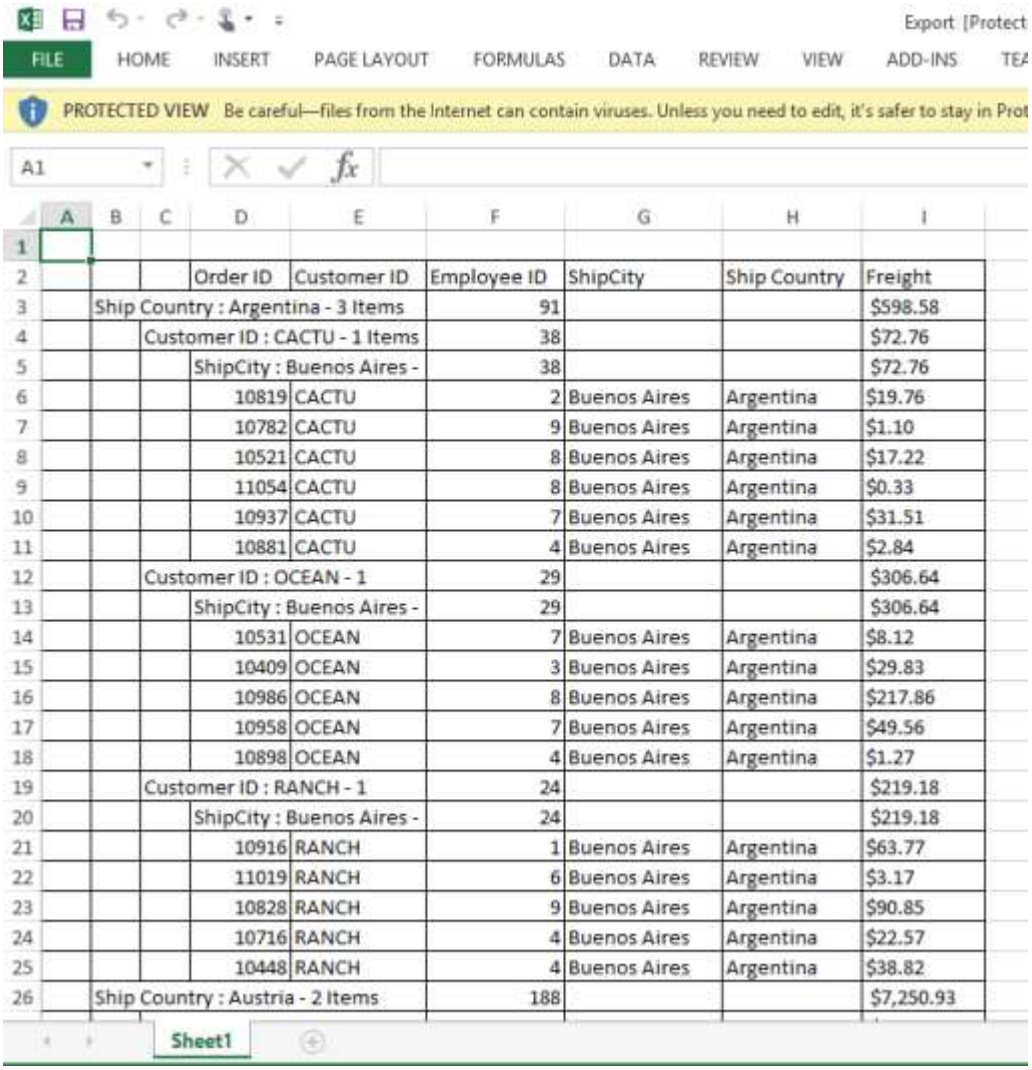
```
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "none");
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties),GridModel);
exp.Export(obj, DataSource, "Export.docx", false, false,
ExportTheme.FlatSaffron);
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties),GridModel);
exp.Export(obj, DataSource, "Export.pdf", false, false, "flat-saffron");
}
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
col.Field("OrderDate").HeaderText("Order Date").TextAlign(TextAlign.Right).
Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

When the theme is set as none and the autoFormat is not set to the grid, then no theme is applied to the exported grid. The grid is exported without any theme as in the following screenshots:



Order ID	Customer ID	Employee ID	ShipCity	Ship Country	Freight
Ship Country : Argentina - 3 Items		91			$598.58
Customer ID : CACTU - 1 Items		38			$72.76
ShipCity : Buenos Aires -		38			$72.76
10819	CACTU	2	Buenos Aires	Argentina	$19.76
10782	CACTU	9	Buenos Aires	Argentina	$1.10
10521	CACTU	8	Buenos Aires	Argentina	$17.22
11054	CACTU	8	Buenos Aires	Argentina	$0.33
10937	CACTU	7	Buenos Aires	Argentina	$31.51
10881	CACTU	4	Buenos Aires	Argentina	$2.84
Customer ID : OCEAN - 1		29			$306.64
ShipCity : Buenos Aires -		29			$306.64
10531	OCEAN	7	Buenos Aires	Argentina	$8.12
10409	OCEAN	3	Buenos Aires	Argentina	$29.83
10986	OCEAN	8	Buenos Aires	Argentina	$217.86
10958	OCEAN	7	Buenos Aires	Argentina	$49.56
10898	OCEAN	4	Buenos Aires	Argentina	$1.27
Customer ID : RANCH - 1		24			$219.18
ShipCity : Buenos Aires -		24			$219.18
10916	RANCH	1	Buenos Aires	Argentina	$63.77
11019	RANCH	6	Buenos Aires	Argentina	$3.17
10828	RANCH	9	Buenos Aires	Argentina	$90.85
10716	RANCH	4	Buenos Aires	Argentina	$22.57
10448	RANCH	4	Buenos Aires	Argentina	$38.82
Ship Country : Austria - 2 Items		188			$7,250.93

AutoFormat class

The **AutoFormat** class can be used to customize the styles or themes applied to the exported grid. With the autoFormat class, you can provide required color to the grid content, alt row background or border color.

The various properties available under the autoFormat class are listed in the following table.

Properties	Type	Description
FontFamily	String	The font name
HeaderBorderColor	System.Drawing.Color	The border color of the grid header.
HeaderFontSize	int	The size of the grid header font.
GHeaderBgColor	System.Drawing.Color	The column header cell color and the group header indent cell color.

Copyright © 2001 - 2020 Syncfusion Inc.

ContentBgColor	System.Drawing.Color	The background color of the content.
ContentBorderColor	System.Drawing.Color	The border color of the grid content.
ContentFontSize	int	The font size of the grid content.
GContentFontColor	System.Drawing.Color	The font color of the record field cell.
GCaptionBorderColor	System.Drawing.Color	The border color of the group caption cell.
AltRowBgColor	System.Drawing.Color	The background color of the alternative row of the grid content.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
col.Field("OrderDate").HeaderText("Order Date").TextAlign(TextAlign.Right).
Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
}))
```

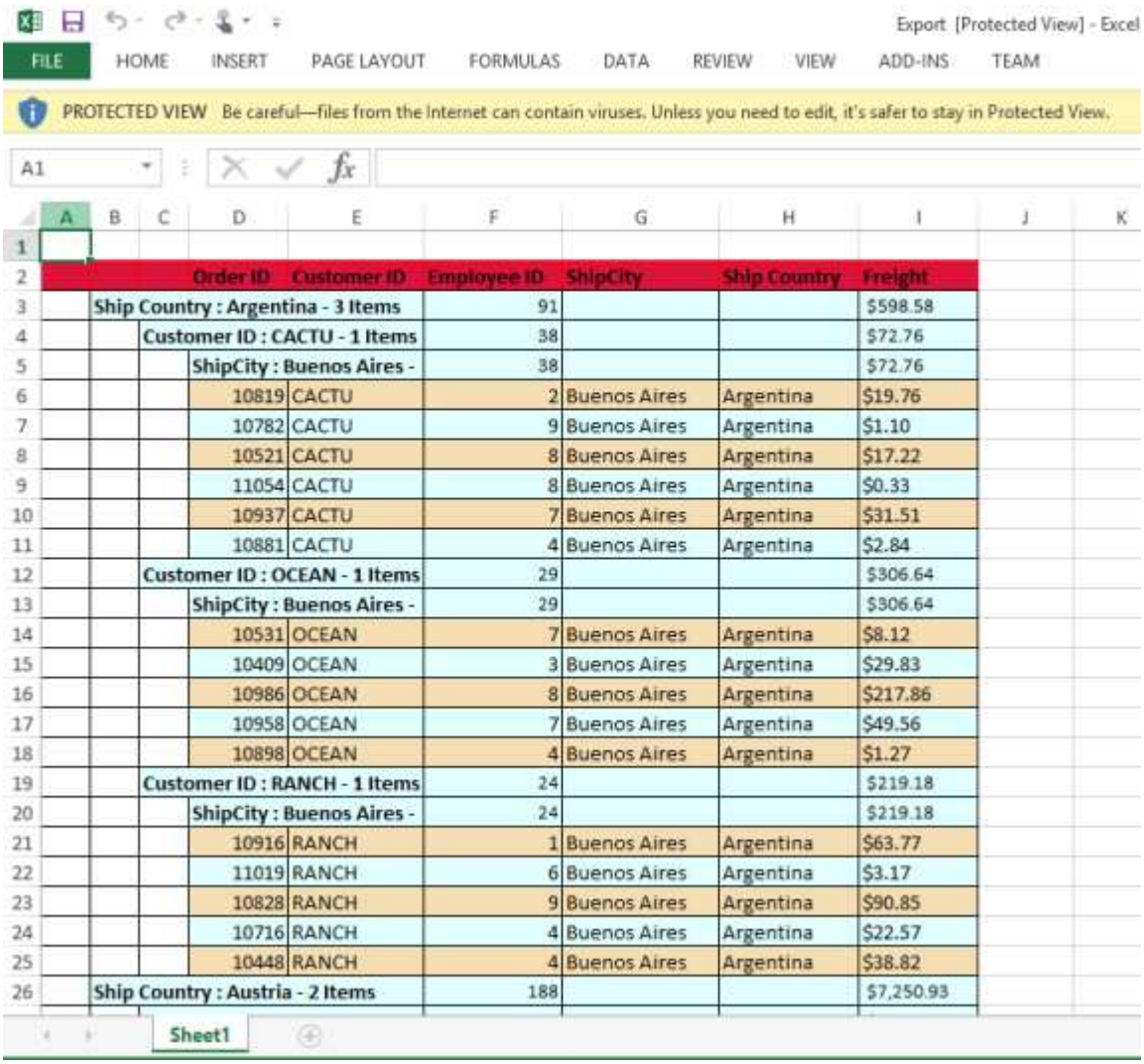
C#

```
public partial class GridController : Controller
{
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "flat-saffron");
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
exp.Export(obj, DataSource, "Export.docx", false, false,
ExportTheme.FlatSaffron);
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj = ConvertGridObject(GridModel);
exp.Export(obj, DataSource, "Export.pdf", false, false, "flat-saffron");
}
private GridProperties ConvertGridObject(string gridProperty)
{
GridProperties gridProp =
Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridProperties),gridPr
operty);
GridExtensions ext = new GridExtensions();
AutoFormat auto = new AutoFormat();
auto.FontFamily = "Arial";
auto.ContentBorderColor = Color.Brown;
auto.ContentFontSize = 10;
auto.GCaptionBorderColor = Color.Cornsilk;
auto.GContentFontColor = Color.DarkBlue;
auto.HeaderFontSize = 12;
auto.HeaderBorderColor = Color.Red;
auto.ContentBgColor = Color.Wheat;
auto.GHeaderBgColor = Color.Crimson;
auto.AltRowBgColor = Color.LightCyan;
ext.SetTheme(auto, "flat-saffron");
gridProp.AutoFormat = auto;
return gridProp;
}
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Order ID	Customer ID	Employee ID	ShipCity	Ship Country	Freight
Ship Country : Argentina - 3 Items		91			$598.58
Customer ID : CACTU - 1 Items		38			$72.76
ShipCity : Buenos Aires -		38			$72.76
10819	CACTU	2	Buenos Aires	Argentina	$19.76
10782	CACTU	9	Buenos Aires	Argentina	$1.10
10521	CACTU	8	Buenos Aires	Argentina	$17.22
11054	CACTU	8	Buenos Aires	Argentina	$0.33
10937	CACTU	7	Buenos Aires	Argentina	$31.51
10881	CACTU	4	Buenos Aires	Argentina	$2.84
Customer ID : OCEAN - 1 Items		29			$306.64
ShipCity : Buenos Aires -		29			$306.64
10531	OCEAN	7	Buenos Aires	Argentina	$8.12
10409	OCEAN	3	Buenos Aires	Argentina	$29.83
10986	OCEAN	8	Buenos Aires	Argentina	$217.86
10958	OCEAN	7	Buenos Aires	Argentina	$49.56
10898	OCEAN	4	Buenos Aires	Argentina	$1.27
Customer ID : RANCH - 1 Items		24			$219.18
ShipCity : Buenos Aires -		24			$219.18
10916	RANCH	1	Buenos Aires	Argentina	$63.77
11019	RANCH	6	Buenos Aires	Argentina	$3.17
10828	RANCH	9	Buenos Aires	Argentina	$90.85
10716	RANCH	4	Buenos Aires	Argentina	$22.57
10448	RANCH	4	Buenos Aires	Argentina	$38.82
Ship Country : Austria - 2 Items		188			$7,250.93

Exporting server events

The Exporting feature supports server-side event handler. You can handle server-side event while exporting grid to various files such as Excel, PDF and Word. The various server-side events available in exporting and its argument types are listed in the following table.

Event Name	Argument	Description
ServerExcelQueryCellInfo	currentCell	It returns the current cell of excel sheet.
ServerExcelRowInfo	range	It returns the current row of excel sheet.
ServerWordQueryCellInfo	currentCell	It returns the current cell of word.
ServerWordRowInfo	range	It returns the current row of word.
ServerPdfQueryCellInfo	currentCell	It returns the current cell of PDF.
ServerPdfRowInfo	range	It returns the current row of PDF.

Copyright © 2001 - 2020 Syncfusion Inc.

You can customize the particular cell or particular row of exporting files using server events. The code sample for this is as follows.

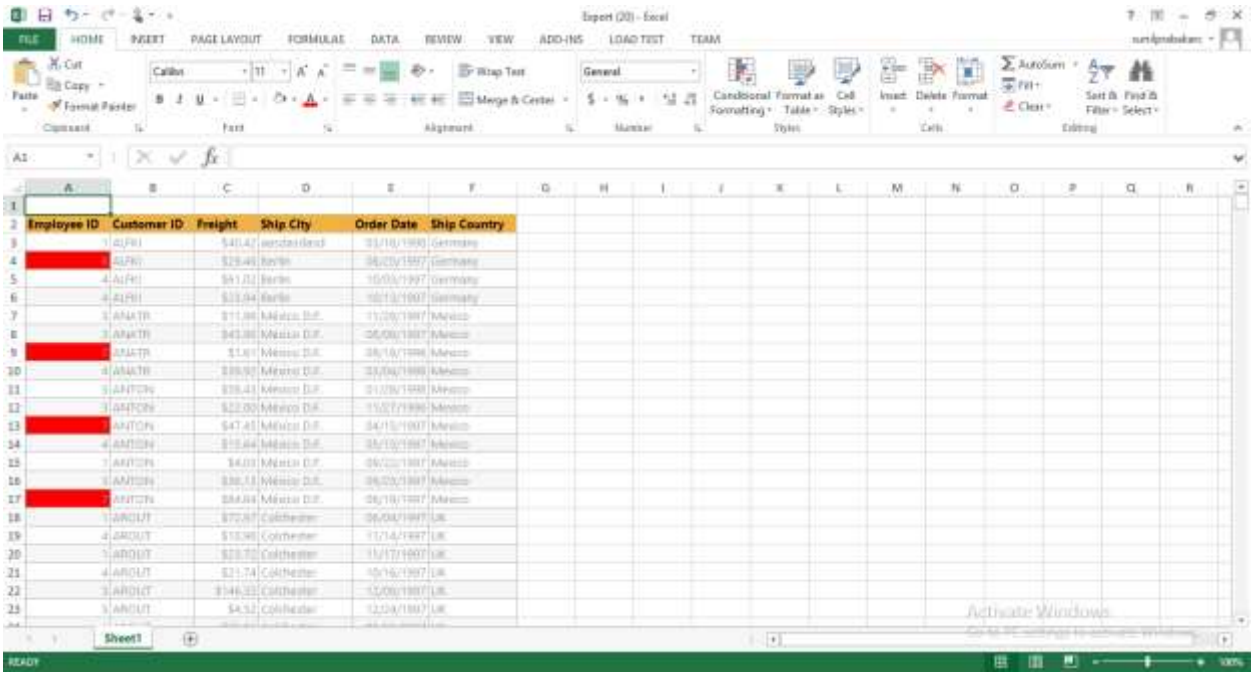
RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
col.Field("OrderDate").HeaderText("Order Date").TextAlign(TextAlign.Right).
Add();
col.Field("ShipCity").HeaderText("Ship City").Add();
}))
```

C#

```
public partial class GridController : Controller
{
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = new
NorthwindDataContext().EmployeeViews.Take(100).ToList();
GridProperties obj
=(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridP
roperties), GridModel);
obj.ServerExcelQueryCellInfo = QueryCellInfo;
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "flat-saffron");
}
public void QueryCellInfo(object currentCell)
{
IRange range = (IRange)currentCell;
if (range.Column==1 && int.Parse(range.Value) == 5)
range.CellStyle.Color = Color.Red;
}
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Print

Use the print() method from Grid instance to print the grid. You can add Print option in Toolbar item by adding the PrintGrid in ToolbarItems.

RAZOR

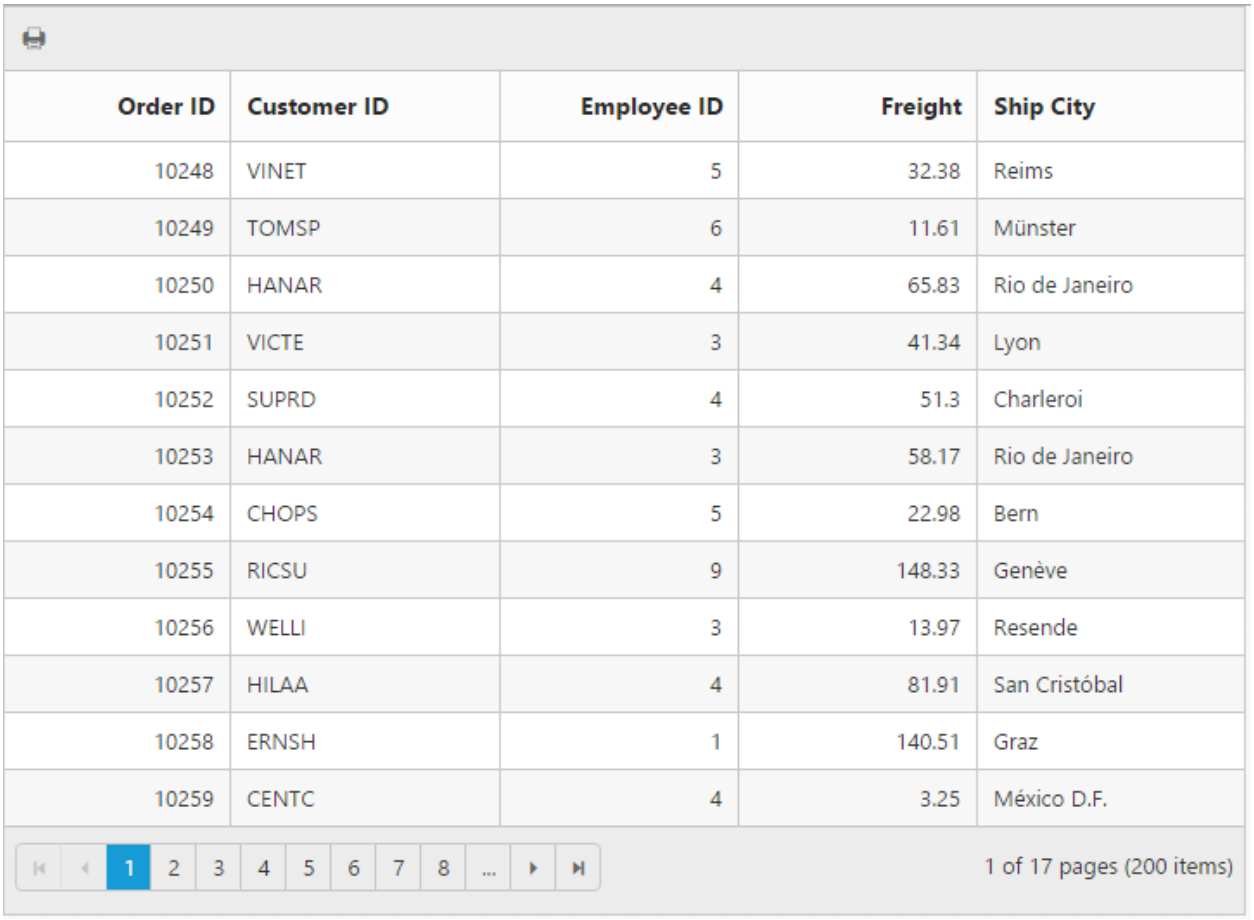
```
@(Html.EJ().Grid<OrdersView>("PrintGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.PrintGrid);
});
})
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
}))
```

C#

```
using System;
using System.Collections.Generic;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SyncfusionMvcApplication1;
using SyncfusionMvcApplication1.Models;
namespace SyncfusionMvcApplication1.Controllers
{
public class GridController : Controller
{
//
// GET: /Grid/
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```



Order ID	Customer ID	Employee ID	Freight	Ship City
10248	VINET	5	32.38	Reims
10249	TOMSP	6	11.61	Münster
10250	HANAR	4	65.83	Rio de Janeiro
10251	VICTE	3	41.34	Lyon
10252	SUPRD	4	51.3	Charleroi
10253	HANAR	3	58.17	Rio de Janeiro
10254	CHOPS	5	22.98	Bern
10255	RICSU	9	148.33	Genève
10256	WELLI	3	13.97	Resende
10257	HILAA	4	81.91	San Cristóbal
10258	ERNSH	1	140.51	Graz
10259	CENTC	4	3.25	México D.F.

Page setup

Some of the print options are not configurable through JavaScript code. You need to customize layout, paper size and margins options through browser's page setup dialog. Please find the following guidelines link to browser page setup.

Copyright © 2001 - 2020 Syncfusion Inc.

- Chrome
- Firefox
- Safari
- IE

Print on external button click

By default, the grid can be print from toolbar. To print from external button action, you need to call the grid's print() method from the required button event.

RAZOR

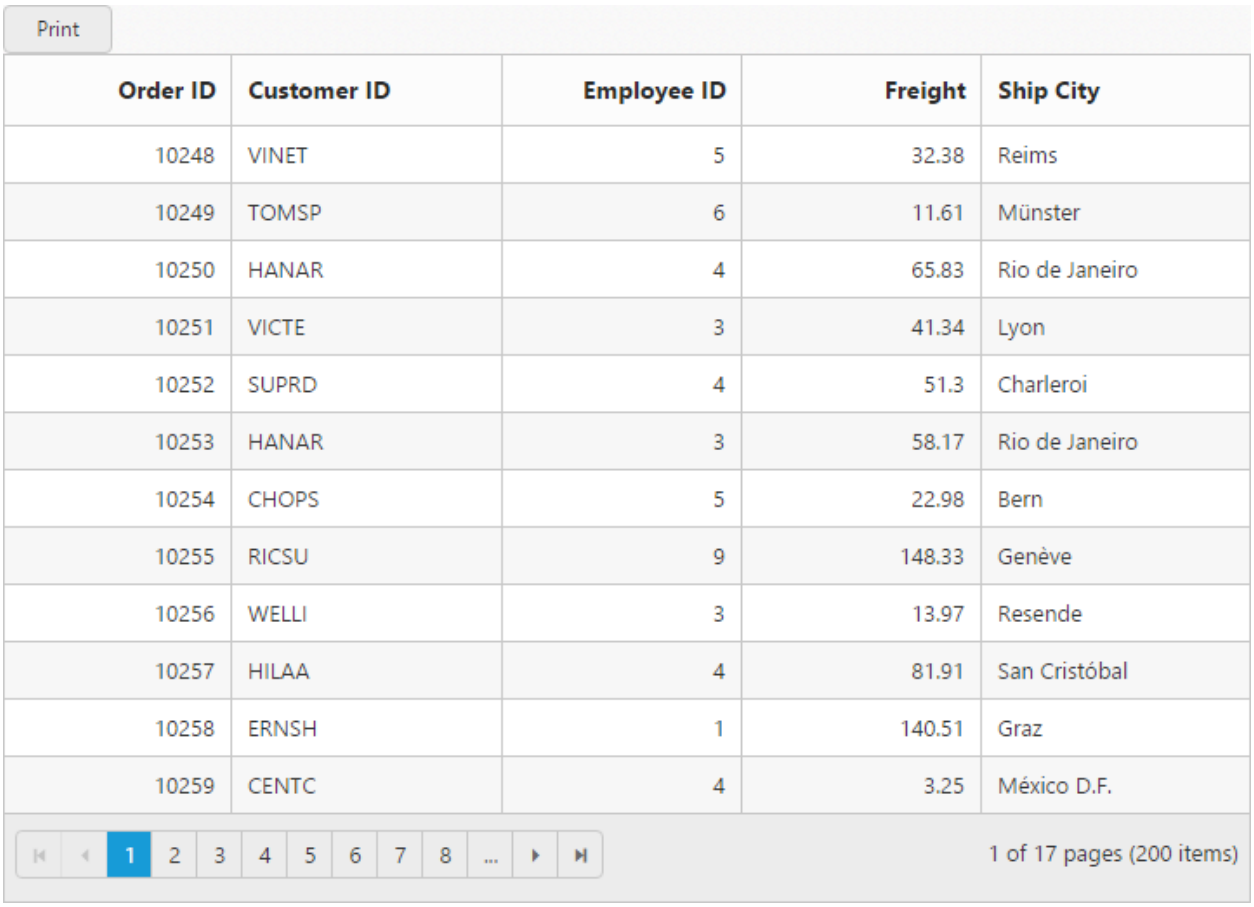
```
<button id="print">Print</button>
@(Html.EJ().Grid<OrdersView>("PrintGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.EnableHeaderHover()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
}))
<script type="text/javascript">
$("#print").ejButton({
showRoundedCorner: true,
size: "mini",
click: function () {
$("#PrintGrid").ejGrid("print");
}
});
</script>
```

C#

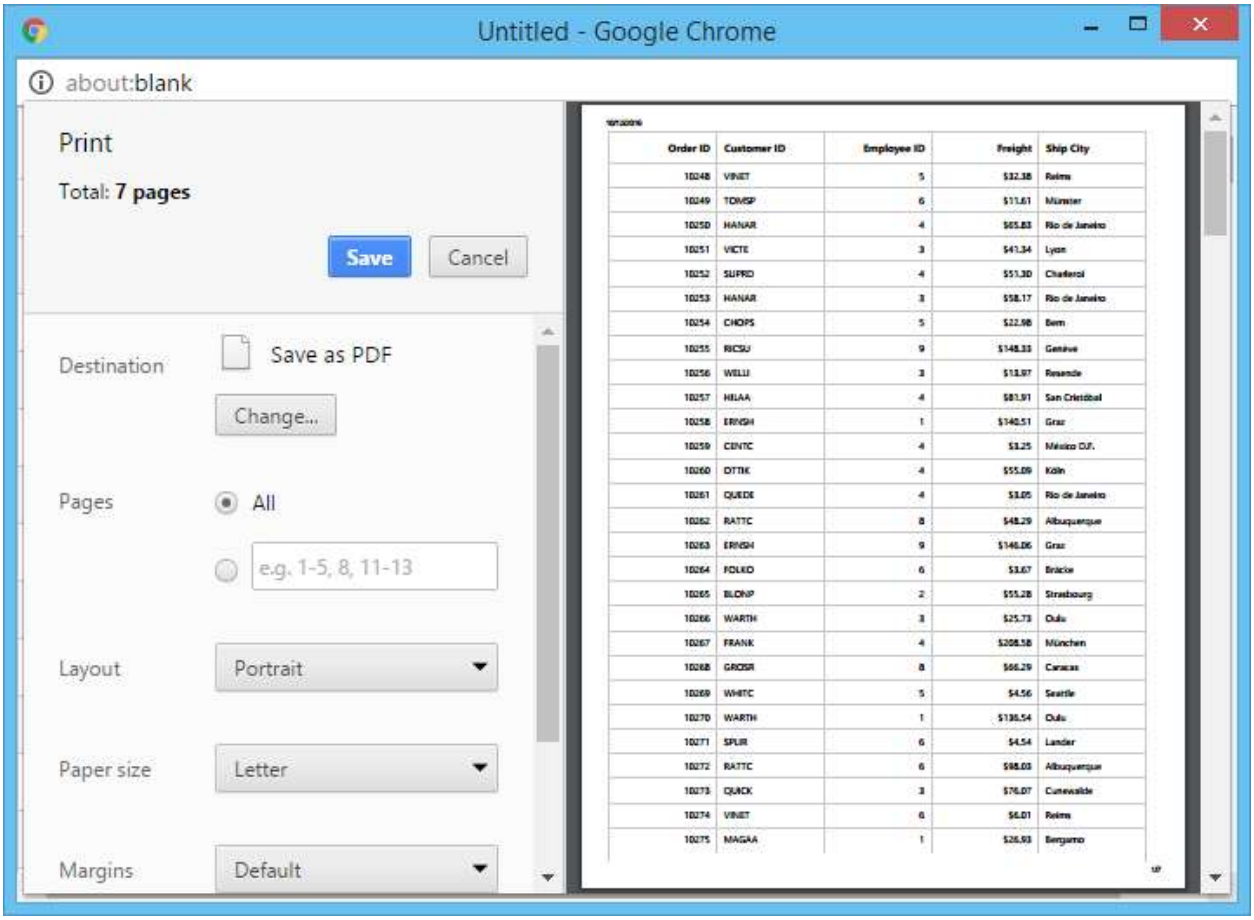
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SyncfusionMvcApplication1;
using SyncfusionMvcApplication1.Models;
namespace SyncfusionMvcApplication1.Controllers
{
public class GridController : Controller
{
//
// GET: /Grid/
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewBag.datasource = DataSource;
return View();
}
}
}
```



Grid with external button for Print



Print dialog in Chrome browser

Print visible page

By default, the grid will print all records. To print current page, you need to set the PrintMode as CurrentPage in the PageSettings property.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("PrintGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.PageSettings(page=> page.PrintMode(PrintMode.CurrentPage))
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.PrintGrid);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(90).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(80).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
}))
```

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using SyncfusionMvcApplication1;
using SyncfusionMvcApplication1.Models;
namespace SyncfusionMvcApplication1.Controllers
{
public class GridController : Controller
{
//
// GET: /Grid/
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

State Persistence

State persistence is to maintain the grid state in browser's local storage even if browser refreshes or move to the next page. State persistence stores grid's model object in local storage while defining the enablePersistence as true.

Information: The localStorage is not supported below IE9 then grid state persistence technique is fallback to cookie.

List of properties are not persisted by default

The following properties are not included while maintaining the grid state in local storage to keep localStorage compact.

- Query
- isEdit
- toolbarClick
- queryCellInfo
- mergeCellInfo
- currentViewData
- enableAltRow
- enableRTL
- contextClick
- contextOpen
- rowDataBound

Copyright © 2001 - 2020 Syncfusion Inc.

- rowTemplate
- detailsDataBound
- detailsTemplate
- childGrid
- summaryRows
- toolbarSettings
- editSettings
- allowMultiSorting
- enableAutoSaveOnSelectionChange
- locale
- allowScrolling
- allowCellMerging
- allowTextWrap
- cssClass
- dataSource
- groupSettings.enableDropAreaAnimation
- enableRowHover
- showSummary
- allowGrouping
- enableHeaderHover
- allowKeyboardNavigation
- scrollSettings.frozenRows
- scrollSettings.frozenColumns
- enableTouch
- editSettings.rowPosition
- editSettings.showAddNewRow
- contextMenuSettings.enableContextMenu

Information: The given excluded properties can be included in persist state using the `_ignoreOnPersist` in grid prototype.

Accessing currently stored state

The persisted state can be accessed through local storage using the corresponding key name. Key name is the combination of plugin name and control id.

JS

```
var gridStateString = window.localStorage.ejGridGrid; // grid state as
string
var gridStateObject = JSON.parse(window.localStorage.ejGridGrid);//grid
state as object
```

Information: In the above example, the "ejGrid" is plugin name and "Grid" is control id.

Globalization and localization

Localization

All text in the grid can be localized using the `ej.Grid.Locale` object. Please find the table with list of properties and its value in locale object.

Copyright © 2001 - 2020 Syncfusion Inc.

Locale key words	Text
EmptyRecord	No records to display
PagerInfo	{0} of {1} pages ({2} items)
GroupDropArea	Drag a column header here to group its column
DeleteOperationAlert	No records selected for delete operation
EditOperationAlert	No records selected for edit operation
SaveButton	Save
OKButton	OK
CancelButton	Cancel
EditFormTitle	Details of
AddFormTitle	Add New Record
GroupCaptionFormat	{{"{{"}}:headerText{{}}}}: {{"{{"}}:key{{}}}} - {{"{{"}}:count{{}}}} {{"{{"}}if count == 1 {{}}}} item {{"{{"}}else{{}}}} items {{"{{"}}/if{{}}}}
BatchSaveConfirm	Are you sure you want to save changes?
BatchSaveLostChanges	Unsaved changes will be lost. Are you sure you want to continue?
ConfirmDelete	Are you sure you want to Delete Record?
FrozenColumnsViewAlert	Frozen columns should be in grid view area
FrozenColumnsScrollAlert	Enable allowScrolling while using frozen Columns
FrozenNotSupportedException	Frozen Columns and Rows are not supported for Grouping, Row Template, Detail Template, Hierarchy Grid and Batch Editing
Add	Add
Edit	Edit
Delete	Delete
Update	Update
Cancel	Cancel
Done	Done
Columns	Columns
PrintGrid	Print
ExcelExport	Excel Export
WordExport	Word Export

Copyright © 2001 - 2020 Syncfusion Inc.

PdfExport	PDF Export
StringMenuOptions	[{text: "StartsWith",value: "StartsWith"},{text: "EndsWith",value: "EndsWith"},{text: "Contains",value: "Contains"},{text: "Equal",value: "Equal"},{text: "NotEqual",value: "NotEqual"}]
NumberMenuOptions	[{text: "LessThan",value: "LessThan"},{text: "GreaterThan",value: "GreaterThan"},{text: "LessThanOrEqual",value: "LessThanOrEqual"},{text: "GreaterThanOrEqual",value: "GreaterThanOrEqual"},{text: "Equal",value: "Equal"},{text: "NotEqual",value: "NotEqual"}]
PredicateAnd	AND
PredicateOr	OR
Filter	Filter
FilterMenuCaption	Filter Value
FilterbarTitle	's filter bar cell
MatchCase	Match Case
Clear	Clear
ResponsiveFilter	Filter
ResponsiveSorting	Sort
Search	Search
DatePickerWaterMark	Select date
EmptyDataSource	DataSource must not be empty at initial load since columns are generated from the dataSource in AutoGenerate Column Grid
True	True
False	False
UnGroup	Click here to ungroup
AddRecord	Add Record
EditRecord	Edit Record
DeleteRecord	Delete Record
Save	Save
Grouping	Grouping
Ungrouping	Ungrouping
SortInAscendingOrder	Sort In Ascending Order

Copyright © 2001 - 2020 Syncfusion Inc.

SortInDescendingOrder	Sort In Descending Order

Pager Localization

Paging in Grid can also be localized using the ej.Pager.Locale object. Please find the table with list of properties and its value in locale object.

Locale key words	Text
PagerInfo	{0} of {1} pages ({2} items)
firstPageTooltip	Go to first page
lastPageTooltip	Go to last page
nextPageTooltip	Go to next page
previousPageTooltip	Go to previous page
nextPagerTooltip	Go to next Pager
previousPagerTooltip	Go to previous Pager

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Localization")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowGrouping()
.AllowPaging()
.Locale("de-DE")
.GroupSettings(group=>group.EnableDropAreaAnimation(false))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(95).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(95).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(80).Add();
})
)
```

C#

```
public partial class GridController : Controller
{
//
// GET: /Localization/
public ActionResult Localization()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.dataSource = DataSource;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
}
```

JS

```
<script type="text/javascript">
ej.Grid.Locale["de-DE"] = {
EmptyRecord: "Keine Aufzeichnungen angezeigt",
GroupDropArea: "Ziehen Sie eine Spaltenüberschrift hier",
DeleteOperationAlert: "Keine Einträge für Löschvorgang ausgewählt",
EditOperationAlert: "Keine Einträge für Bearbeiten Betrieb ausgewählt",
SaveButton: "Speichern",
CancelButton: "stornieren",
EditFormTitle: "Korrektur von",
GroupCaptionFormat: "{{"{{"}}:field{{}}}}: {{"{{"}}:key{{}}}} -
{{"{{"}}:count{{}}}} {{"{{"}}if count ==
1{{}}}}Beiträge{{"{{"}}else{{}}}}Beiträges{{"{{"}}/if{{}}}}",
UnGroup: "Klicken Sie hier, um die Gruppierung aufheben"
};
ej.Pager.Locale["de-DE"] = {
pagerInfo: "{0} von {1} Seiten ({2} Beiträge)",
firstPageTooltip: "Zur ersten Seite",
lastPageTooltip: "Zur letzten Seite",
nextPageTooltip: "Zur nächsten Seite",
previousPageTooltip: "Zurück zur letzten Seite",
nextPagerTooltip: "Zum nächsten Pager",
previousPagerTooltip: "Zum vorherigen Pager"
};
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Information: You need to change pager locale in `ej.Pager.Locale` object.

Excel-filter Localization

All text in Excel-filter can be localized using the `ej.ExcelFilter.Locale` object. Please find the table with list of properties and its value in locale object.

Locale key words	Text
SortNoSmaller	Sort Smallest to Largest
SortNoLarger	Sort Largest to smallest
SortTextAscending	Sort A to Z
SortTextDescending	Sort Z to A
SortDateOldest	Sort By Oldest
SortDateNewest	Sort By Newest
SortByColor	Sort By Color
SortByCellColor	Sort By Cell Color
SortByFontColor:	Sort By Font Color:

Copyright © 2001 - 2020 Syncfusion Inc.

FilterByColor	Filter By Color
SortColorOptions:	[{ id: 1, background:"#FFFFFF"}, {id: 2, background:"#5EABDA"}],
CustomSort	Custom Sort
FilterColorOptions	{ id: 1, background:"#FFFFFF"}, {id: 2, background:"#5EABDA"}],
FilterByCellColor	Filter By Cell Color
FilterByFontColor	Filter By Font Color
ClearFilter	Clear Filter
NumberFilter	Number Filter
TextFilter	Text Filter
DateFilter	Date Filter
DateTimeFilter	Date Time Filters
GuidFilter	Guid Filters
StringMenuOptions	[{ text:"Equal",value:"equal"},{ text:"Not Equal", value:"notequal"},{ text:"Starts With",value:"startswith"}, { text:"Ends With",value:"endswith"},{ text:"Contains",value:"contains"}, {text:"Custom Filter", value:"customfilter"}],
NumberMenuOptions	[{text:"Equal",value:"equal"}, {text:"Not Equal",value:"notequal"}, { text:"Less Than",value:"lessthan"}, {text:"Less Than Or Equal", value:"lessthanorequal"}, {text:"Greater Than",value:"greaterthan"},{ text:"Greater Than Or Equal", value:"greaterthanorequal"}, { text:"Between",value:"between"},{ text:"Custom Filter", value:"customfilter"}]
DateMenuOptions	[{ text:"Equal", value:"equal"}, {text:"Not Equal",value:"notequal"},{text:"Less Than",>value:"lessthan"}, {text:"Less Than Or Equal",value:"lessthanorequal"}, {text:"Greater Than",value:"greaterthan"},{text:"Greater Than Or Equal", value:"greaterthanorequal"}, { text:"Between",value:"between"},{ text:"Custom Filter", value:"customfilter"}]
DatetimeMenuOptions	[{ text: "Equal", value: "equal" }, { text: "Not Equal", value: "notequal" }, { text: "Less Than", value: "lessthan" }, { text: "Less Than Or Equal", value: "lessthanorequal" }, { text: "Greater Than", value: "greaterthan" }, { text: "Greater Than Or Equal", value: "greaterthanorequal" }, { text: "Between", value: "between" }, { text: "Custom Filter", value: "customfilter" }]
Top10MenuOptions	[{ text:"Top", value:"top"},{text:"Bottom", value:"bottom"}]
GuidMenuOptions	[{ text: "Equal", value: "equal" }, { text: "Not Equal", value: "notequal" }, { text: "Custom Filter", value: "customfilter" }]

Copyright © 2001 - 2020 Syncfusion Inc.

title	Custom Filter
PredicateOr	OR
PredicateAnd	AND
OK	OK
MathCase	Match Case
Cancel	Cancel
NoResult	No Match Found
CheckBoxStatusMsg	Not all items showing
DatePickerWaterMark	Select date
DateTimePickerWaterMark	Select date time
True	True
False	False
SelectAll	Select All
Blanks	Blanks
Showrowswhere	Show rows where
NumericTextboxWaterMark	Enter value
Search	Search
AddToFilter	Add current selection to filter

Please find the code

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Localization")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.Locale("de-DE")
.AllowFiltering()
.FilterSettings(filter => filter.FilterType(FilterType.Excel))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(95).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(95).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(80).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

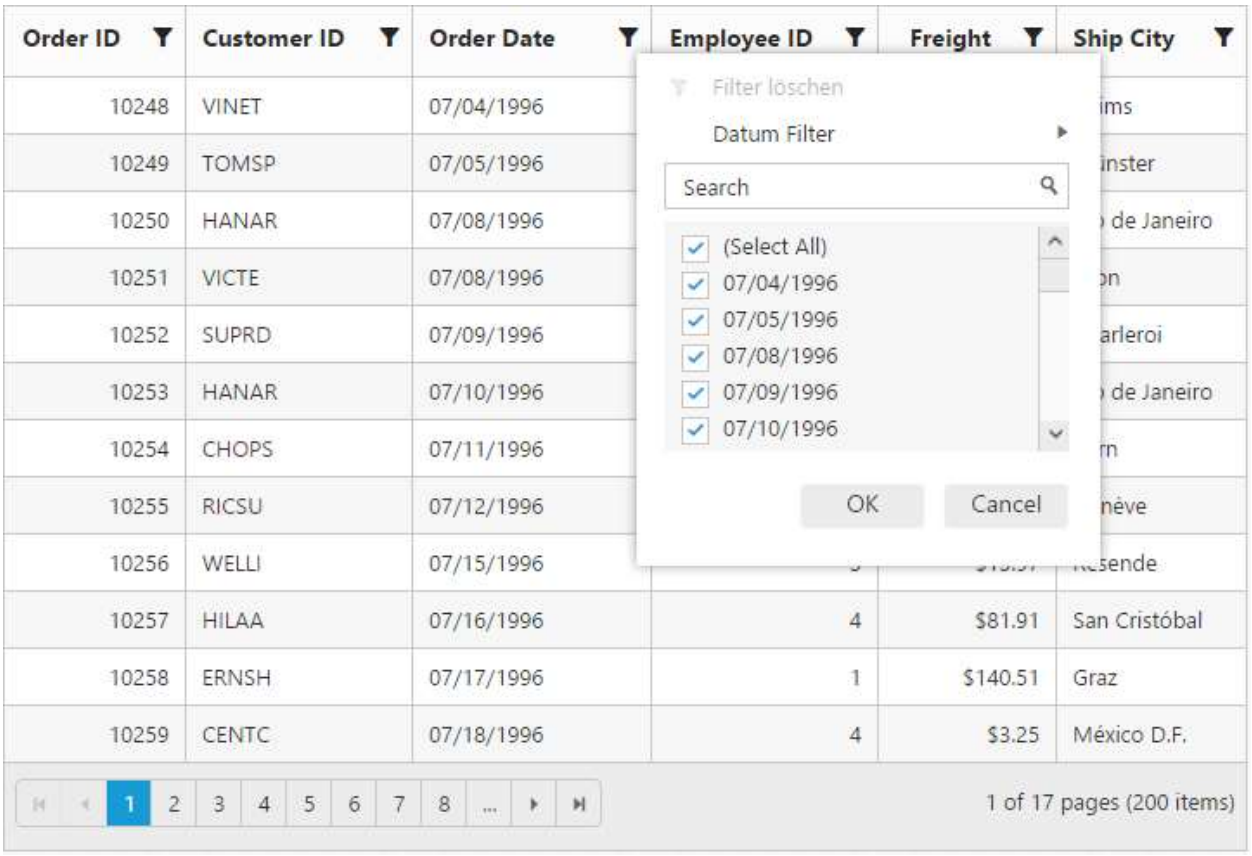
C#

```
public partial class GridController : Controller
{
//
// GET: /Localization/
public ActionResult Localization()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.dataSource = DataSource;
return View();
}
}
```

JS

```
<script type="text/javascript">
ej.ExcelFilter.Locale["de-DE"] = {
SortNoSmaller: "Art Anzahl kleiner",
SortNoLarger: "Art Anzahl größer",
SortTextAscending: "Sortieren aufsteigend Text",
SortTextDescending: "Sortieren absteigend Text",
SortDateOldest: "Sortieren Datum Älteste",
SortDateNewest: "Datum sortieren Neueste",
ClearFilter: "Filter löschen",
DateFilter: "Datum Filter"
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Information: We have uploaded the pre-defined language packs for some commonly used cultures in this github location. Refer to the github location for getting the pre-defined language packs for the corresponding culture. The culture file has localized texts for all the Syncfusion controls.

Globalization

The ej.globalize library is used to globalize numeric values in the grid control using Format property in Columns. Globalize values will be automatically used when Locale property is set with locale string value for example en-US.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Localization")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Locale("de-DE")
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(95).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(95).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(80).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public partial class GridController : Controller
{
//
// GET: /Localization/
public ActionResult Localization()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.dataSource = DataSource;
return View();
}
}
```

Order ID	Customer ID	Employee ID	Freight	Ship City
10248	VINET	5	32,38 €	Reims
10249	TOMSP	6	11,61 €	Münster
10250	HANAR	4	65,83 €	Rio de Janeiro
10251	VICTE	3	41,34 €	Lyon
10252	SUPRD	4	51,30 €	Charleroi
10253	HANAR	3	58,17 €	Rio de Janeiro
10254	CHOPS	5	22,98 €	Bern
10255	RICSU	9	148,33 €	Genève
10256	WELLI	3	13,97 €	Resende
10257	HILAA	4	81,91 €	San Cristóbal
10258	ERNSH	1	140,51 €	Graz

Information: To translate our control content from default English to any of the culture, say For example - German language, then you need to refer the **ej.culture.de-DE.min.js** file in your application, after the reference of ej.web.all.min.js file. For all culture files, please download from the GitHub location.

localization

Right to left - RTL

By default, the grid render its text and layout from left to right. To customize grid's direction, you can change direction from LTR to RTL by using EnableRTL as true.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("Localization")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.EnableRTL()
.Columns(col =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(120).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(120).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(120).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
120).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(120).Add();
})
)
```

C#

```
public partial class GridController : Controller
{
//
// GET: /Localization/
public ActionResult Localization()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.dataSource = DataSource;
return View();
}
}
```

Ship City	Freight	Employee ID	Customer ID	Order ID
Reims	$32.38	5	VINET	10248
Münster	$11.61	6	TOMSP	10249
Rio de Janeiro	$65.83	4	HANAR	10250
Lyon	$41.34	3	VICTE	10251
Charleroi	$51.30	4	SUPRD	10252
Rio de Janeiro	$58.17	3	HANAR	10253
Bern	$22.98	5	CHOPS	10254
Genève	$148.33	9	RICSU	10255
Resende	$13.97	3	WELLI	10256
San Cristóbal	$81.91	4	HILAA	10257
Graz	$140.51	1	ERNSH	10258
.México D.F	$3.25	4	CENTC	10259

(of 17 pages (200 items 1 ... 8 7 6 5 4 3 2 1

Copyright © 2001 - 2020 Syncfusion Inc.

How to

Apply formatting for columns dynamically

Column format can be used dynamically to change data values format with the help of the public method. The following code sample illustrates the Essential JavaScript with column formatting in public method.

JS

```
var grid = $("#Grid").data("ejGrid");
var column = grid.getColumnByField("fieldName");
column.format = "{0:n2}";
grid.refreshContent(true);
```

Disable double-click edit

The AllowEditOnDblClick property can be set as true to enable editing the record by double-clicking it. When it is set as false, it cannot be edited by double-clicking it. In that case, you can edit the record by using the Toolbar option only.

RAZOR

```
@(Html.EJ().Grid<object>("Editing")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing()
.AllowEditOnDblClick(false);})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.AllowPaging(true)
.PageSettings(page => { page.PageSize(8); })
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer
ID").EditType(EditingType.StringEdit).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).EditType(EditingType.NumericEdit).Add();
col.Field("ShipCity").HeaderText("Ship
City").EditType(EditingType.DropdownEdit).Add();
})
)
```

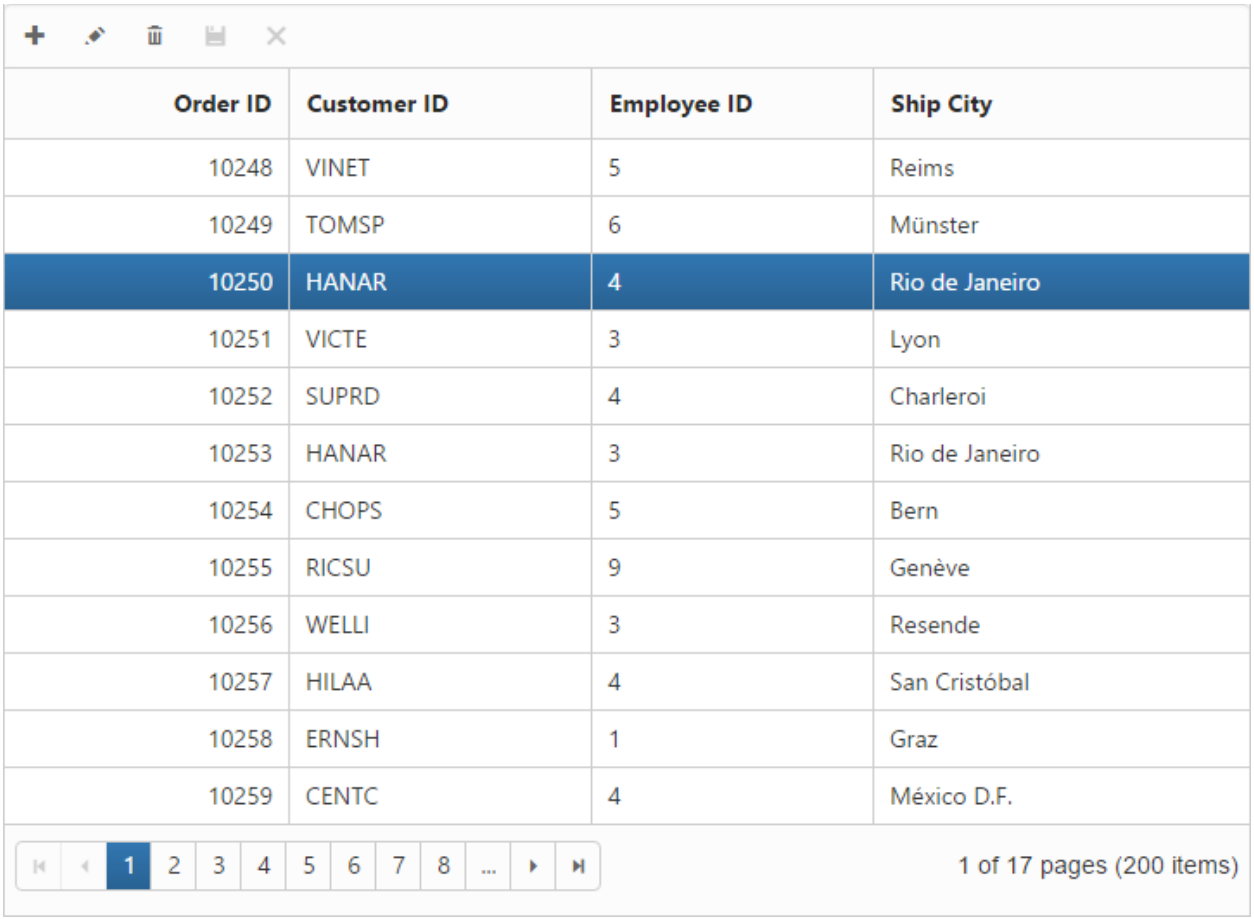
C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public ActionResult Index()
{
ViewBag.datasource = OrderRepository.GetAllRecords();
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



Order ID	Customer ID	Employee ID	Ship City
10248	VINET	5	Reims
10249	TOMSP	6	Münster
10250	HANAR	4	Rio de Janeiro
10251	VICTE	3	Lyon
10252	SUPRD	4	Charleroi
10253	HANAR	3	Rio de Janeiro
10254	CHOPS	5	Bern
10255	RICSU	9	Genève
10256	WELLI	3	Resende
10257	HILAA	4	San Cristóbal
10258	ERNSH	1	Graz
10259	CENTC	4	México D.F.

1 2 3 4 5 6 7 8 ... 1 of 17 pages (200 items)

Customize drag and drop element while grouping

In this section, you can learn how to customize drag and drop element. This drag and drop element is framed by using CSS classes with default values. When you want to change or customize drag and drop element, then just override default values of CSS class values. The e-cloneproperties is the name of drag and drop element in CSS class.

RAZOR

```
<style type="text/css">
.e-grid .e-cloneproperties
{
background-color: black;
}
</style>
```

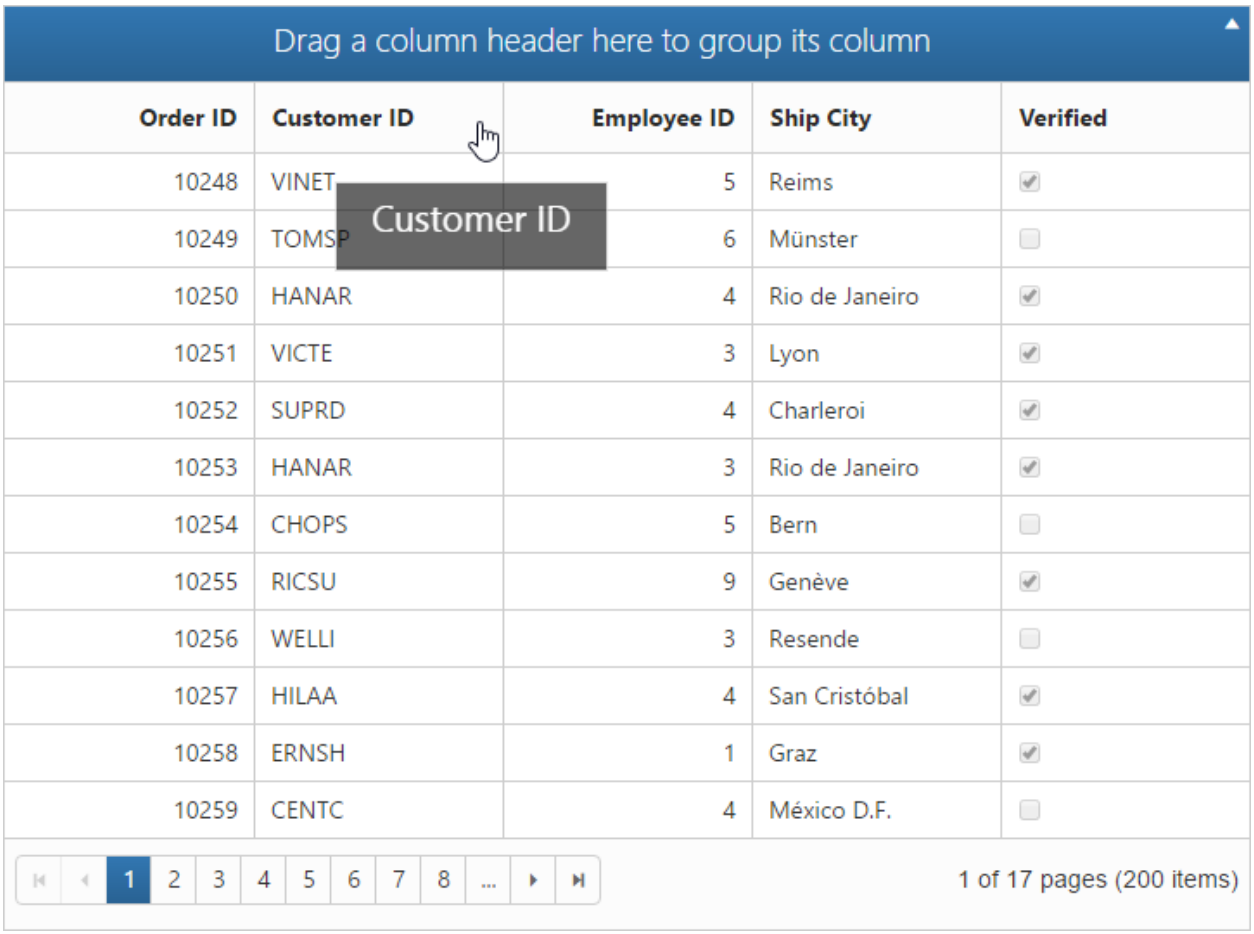
Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Grid<EditableOrder>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowGrouping(true)
.AllowPaging()
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
public ActionResult Default()
{
ViewBag.datasource = OrderRepository.GetAllRecords();
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.



Copyright © 2001 - 2020 Syncfusion Inc.

Display tooltip

To apply tooltip for cells, You need to use the CustomAttributes in columns. For more reference, about the CustomAttributes.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer
ID").Width(80).CustomAttributes(customAttributes =>
customAttributes.AddAttribute("title","")).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").Width(75).TextAlign(TextAlign.Right).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
// GET: /Custom Attributes/
public ActionResult Default()
{
// Data gets from DataContext
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Set dynamic dataSource to grid

Grid control is capable of updating its dataSource as and when required. The grid method “DataSource” helps in achieving this and in this method parameter, you have to pass the new dataSource as list collection.

For instance, consider a textbox above the grid and depending on its value, you can update a new datasource to the grid dynamically.

RAZOR

```
Enter EmployeeID Field Value:
<input type="text" id="colValue" />
<input type="button" id="customButton" value="Change DataSource">
@(Html.EJ().Grid<EJGrid.Models.Order>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

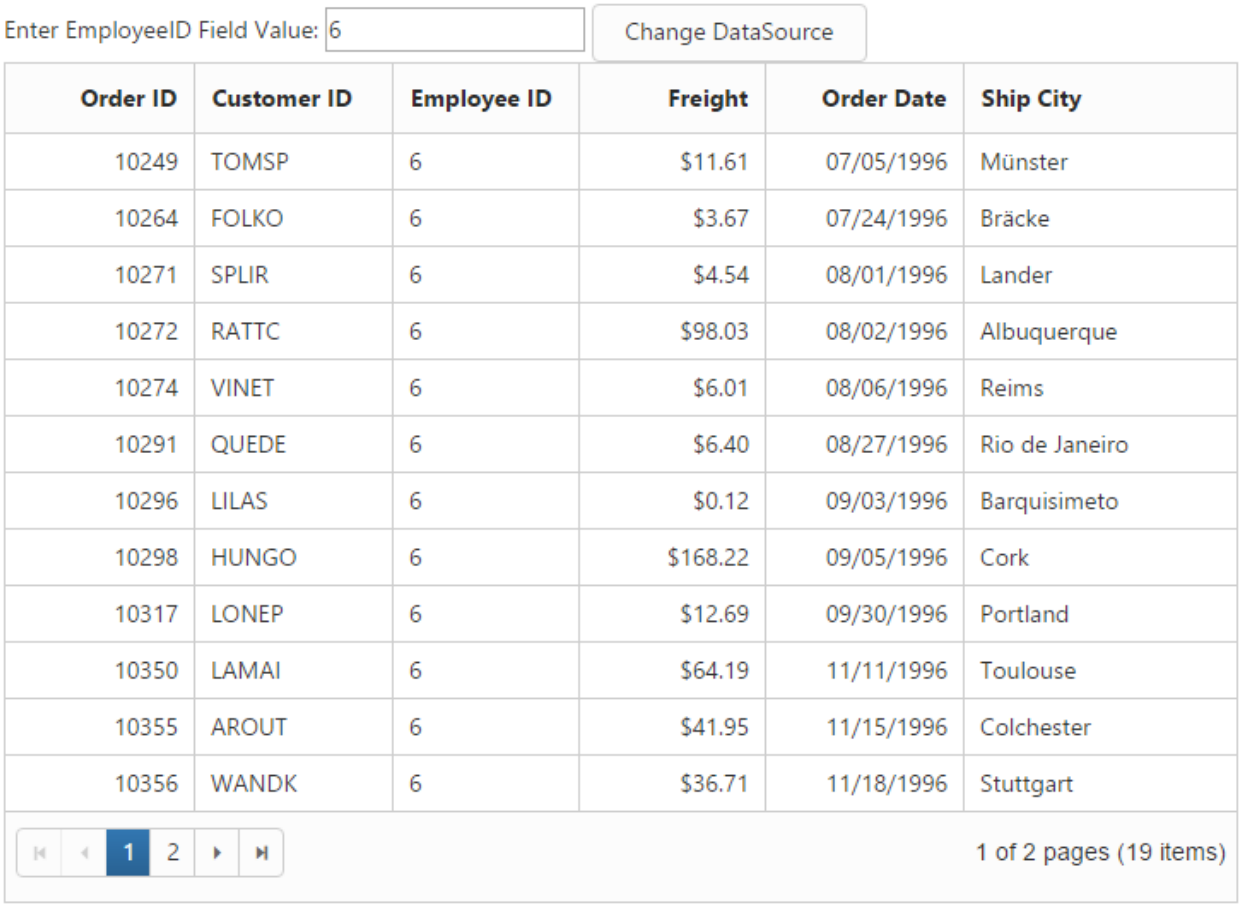
```
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(90).Add();
col.Field("Freight").HeaderText("Freight").Format("{0:c}").TextAlign(TextAli
gn.Right).Width(90).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(90).Add();
col.Field("Child.Test").HeaderText("TEst").Format("{0:c}").Width(90).Add();
col.Field("ShipCountry").HeaderText("Ship Country").Width(90).Add();
})
)
<script>
$("#customButton").ejButton({
size: "Normal", click: function (args) {
var obj = $("#Grid").ejGrid("instance");
var value = $("#colValue").val();
//Add custom parameter to the server
var query = new ej.Query().addParams("EmployeeID", value);
//Creating ejDataManager with UrlAdaptor
var dataManager = ej.DataManager({ url: "/Home/GetData", adaptor: new
ej.UrlAdaptor() });
var promise = dataManager.executeQuery(query);
promise.done(function (e) {
//Assign the result to the grid dataSource using "dataSource" method.
obj.dataSource(e.result);
});
}})
</script>
```

C#

```
namespace EJGrid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = null;
return View();
}
public JsonResult GetData(int EmployeeID)
{
var data = new DataClasses1DataContext().Orders.Where(datasource =>
datasource.EmployeeID ==        EmployeeID).ToList();
return Json(data, JsonRequestBehavior.AllowGet);
}
}
}
```

The following screenshot illustrates the output.

Copyright © 2001 - 2020 Syncfusion Inc.



Custom binding for grid CRUD operation

In the grid control DataManager is used for data processing. The adaptors of dataManager are customizable that can be extended for custom binding with server-side for grid CRUD operation.

For instance bind the data to grid by using the “remoteSaveAdaptor” and extend it to modify its update method to bind edited record values of grid as “FormCollection” in the server-side.

RAZOR

```
@(Html.EJ().Grid<EJGrid.Models.Order>("Grid")
.Datasource(datasource =>
datasource.Json((IEnumerable<object>)ViewBag.datasource).UpdateURL("Home/Upd
ate")
.InsertURL("Home/Insert").RemoveURL("Home/Remove").Adaptor("remoteSaveAdapto
r"))
.AllowSorting()
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
```

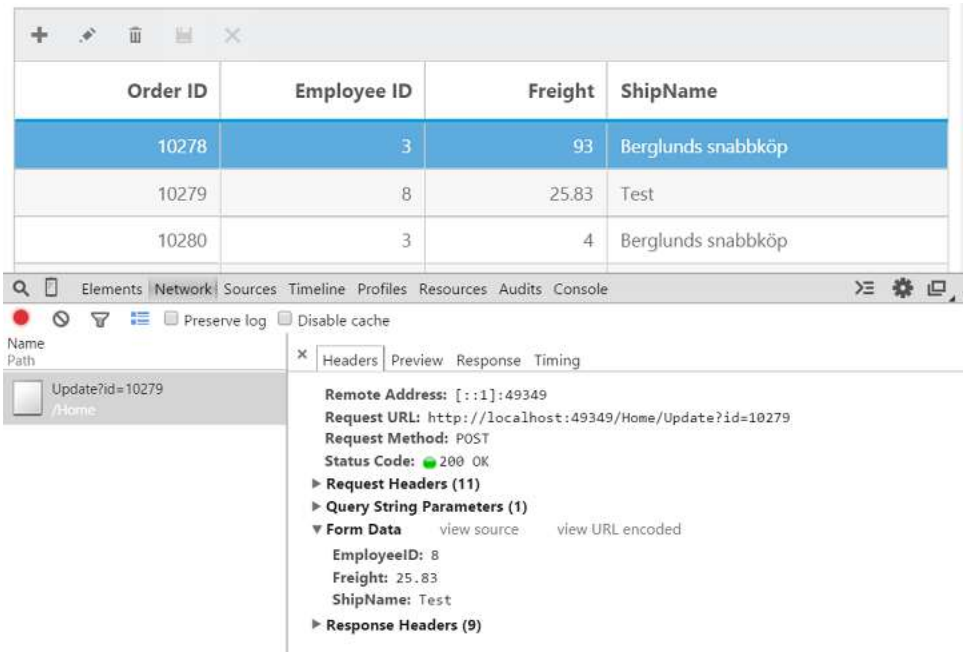
Copyright © 2001 - 2020 Syncfusion Inc.

```
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(90).ValidationRules
(v => v.AddRule("required", true).AddRule("number", true)).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(90).ValidationRules(v =>
v.AddRule("required", true)).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
80).Add();
col.Field("ShipName").HeaderText("ShipName").Width(150).Add();
})
.ClientSideEvents(eve => eve.Load("load"))
)
var adaptor = new ej.remoteSaveAdaptor().extend({
insert: function (dataManager, data, tableName) {
return {
url: dataManager.dataSource.insertUrl,
dataType: 'json',
contentType: "application/x-www-form-urlencoded; charset=utf-8",
data: $("#GridEditForm").serialize()
};
},
update: function (dataManager, keyField, value, tableName) {
return {
type: "POST",
url: dataManager.dataSource.updateUrl+"?id="+value.OrderID,
dataType: 'json',
contentType: "application/x-www-form-urlencoded; charset=utf-8",
data: $("#GridEditForm").serialize()
};
},
})
function load(args) {
this.model.dataSource.adaptor = new adaptor();
}
```

C#

```
namespace EJGrid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = OrderRepository.GetAllRecords();
return View();
}
public ActionResult Update(int id, FormCollection value)
{
var order = new EditableOrder();
order.OrderID = id;
```

```
if (value["EmployeeID"] != "")
order.EmployeeID = Convert.ToInt32(value["EmployeeID"]);
if (value["Freight"] != "")
order.Freight = Convert.ToDecimal(value["Freight"]);
if (value["ShipCity"] != null)
order.ShipCity = value["ShipCity"];
OrderRepository.Update(order);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(FormCollection value)
{
var order = new EditableOrder();
order.OrderID = Convert.ToInt32(value["OrderID"]);
if (value["EmployeeID"] != "")
order.EmployeeID = Convert.ToInt32(value["EmployeeID"]);
if (value["Freight"] != "")
order.Freight = Convert.ToDecimal(value["Freight"]);
if (value["ShipCity"] != null)
order.ShipCity = value["ShipCity"];
OrderRepository.Add(order);
var data = OrderRepository.GetAllRecords();
return Json(data, JsonRequestBehavior.AllowGet);
}
}
}
```



Binding SignalR endpoint

Grid supports the SignalR features for live updates in record. Please find the below option to configure signalR with Grid.

1) Before configure SignalR with the ejGrid. You need to setup the SignalR configuration in Visual Studio project. For reference, please find the link given below.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: signalR

2) After configuration of SignalR, you have to create Hub for communication between different actions of grid.

C#

```
public class SignalHub: Hub
{
public void modify(string action, string details)
{
Clients.All.modify(action, details);
}
}
```

3) Implementation of SignalR communication with Grid through Hub.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
75).Format("{0:C}").Add();
col.Field("ShipCity").HeaderText("Ship City").Width(80).Add();
})
)
<!--Reference the SignalR library. -->
<script src="~/Scripts/jquery.signalR-1.1.4.min.js"></script>
<!--Reference the autogenerated SignalR hub script. -->
<script src="@Url.Content("~/signalr/hubs")"
type="text/javascript"></script>
<script>
$(function () {
window.signal = $.connection.signalHub;
window.signal.client.modify = function (action, details) {
details = JSON.parse(details);
if (action == "add") $("#FlatGrid").ejGrid("addRecord", details);
else if (action == "beginedit") $("#Editing").ejGrid("updateRecord",
"OrderID", details);
else $("#Editing").ejGrid("deleteRecord", "OrderID", details);
};
$.connection.hub.start().done(function () {
window.actionComplete = function (args) {
if (args.requestType == "save" || args.requestType == "delete")
window.signal.server.modify(args.requestType == "delete" ? args.requestType
: window.previousAction, JSON.stringify(args.rowData));
if (args.requestType != "delete") window.previousAction = args.requestType;
}
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class GridController : Controller
{
// GET: /Custom Attributes/
public ActionResult Default()
{
// Data gets from DataContext
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copy data from Excel to grid

This blog is about conversion of Excel to JSON data. After got JSON data you can bind it to the grid.

Use NHibernate with grid

NHibernate

NHibernate is an object-relational mapping (ORM) solution for the Microsoft .NET platform. It provides a Framework for mapping an object-oriented domain model to a traditional relational database.

Populate Grid with data by using NHibernate

You can populate the grid with data obtained by using the NHibernate. The UrlAdaptor can be used for data binding and to handle the request and response from the server. You can use the CRUD URL mappers (InsertUrl/UpdateUrl/RemoveUrl/CrudUrl) to provide the action/method to be called while editing.

The grid initialization is as follows.

RAZOR

```
@(Html.EJ().Grid<EmployeeMap>("Grid")
.Datasource(datasource =>
datasource.URL("Home/GetData") //Action which returns data
.InsertURL("Home/PerformInsert")
.UpdateURL("Home/PerformUpdate")
.RemoveURL("Home/PerformDelete")
.Adaptor(AdaptorType.UrlAdaptor)
)
.AllowPaging()
.EditSettings(edit =>
edit.AllowEditing().AllowAdding().AllowDeleting().EditMode(EditMode.Normal))
.ToolbarSettings(tools => tools.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
items.AddTool(ToolBarItems.Cancel);
}))
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(75).Add();
col.Field("FirstName").HeaderText("First Name").Width(80).Add();
col.Field("LastName").HeaderText("Last Name").Add();
col.Field("Designation").HeaderText("Designation"). Width(75).Add();
})
)
```

C#

```
namespace UsingNHibernate.Controllers
{
public class HomeController : Controller
{
public IEnumerable<EmployeeMap> list { get; set; }
public ActionResult Index()
{
return View();
}
//PROVIDE GRID DATASOURCE
public ActionResult GetData()
{
using (ISession session = AppSession.OpenSession())
{
list = session.Query<EmployeeMap>().ToList();
}
return Json(new { result = list, count = list.Count() },
JsonRequestBehavior.AllowGet);
}
//PERFORM INSERT
public ActionResult PerformInsert(string action, EmployeeMap value)
{
using (ISession session = AppSession.OpenSession())
{
using (ITransaction transaction = session.BeginTransaction())
{
session.Save(value);
transaction.Commit();
}
list = session.Query<EmployeeMap>().ToList();
}
return Json(new { result = value, count = list.Count() },
JsonRequestBehavior.AllowGet);
}
//PERFORM UPDATE
public ActionResult PerformUpdate(int key, EmployeeMap value)
{
using (ISession session = AppSession.OpenSession())
{
var employeeUpdate = session.Get<EmployeeMap>(key);
employeeUpdate.Designation = value.Designation;
employeeUpdate.FirstName = value.FirstName;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
employeeUpdate.LastName = value.LastName;
using (ITransaction transaction = session.BeginTransaction())
{
session.Save(employeeUpdate);
transaction.Commit();
}
list = session.Query<EmployeeMap>().ToList();
}
return Json(new { result = value, count = list.Count() },
JsonRequestBehavior.AllowGet);
}
//PERFORM DELETE
public ActionResult PerformDelete(int key)
{
using (ISession session = AppSession.OpenSession())
{
using (ITransaction transaction = session.BeginTransaction())
{
session.Delete(session.Get<EmployeeMap>(key));
transaction.Commit();
}
list = session.Query<EmployeeMap>().ToList();
}
return Content("Success"); // Json(new { result = list, count = list.Count()
}, JsonRequestBehavior.AllowGet);
}
}
}
```

The SessionFactory used by the application is as follows.

C#

```
public static class AppSession
{
public static ISession OpenSession()
{
var configuration = new Configuration();
var configurationPath =
HttpContext.Current.Server.MapPath(@"~\Models\nHibernateConfig\Employee.cfg.
xml");
configuration.Configure(configurationPath);
var employeeConfigurationFile =
HttpContext.Current.Server.MapPath(@"~\Models\nHibernateConfig\EmployeeMap.h
bm.xml");
configuration.AddFile(employeeConfigurationFile);
ISessionFactory sessionFactory = configuration.BuildSessionFactory();
return sessionFactory.OpenSession();
}
}
```

Note: In the previous code example, the ORM mapping is performed by using the XML-documents and the mapping files are Employee.cfg.xml and EmployeeMap.hbm.xml

Copyright © 2001 - 2020 Syncfusion Inc.

Adding comments in the exported file (Excel, Word or PDF)

We can add comments in the exported file while exporting the grid control.

Adding comments in Excel sheet

Using the Range property and AddComment method of the XlsIO IRange Class, we will add comments to any cell of particular range in Excel sheet.

RAZOR

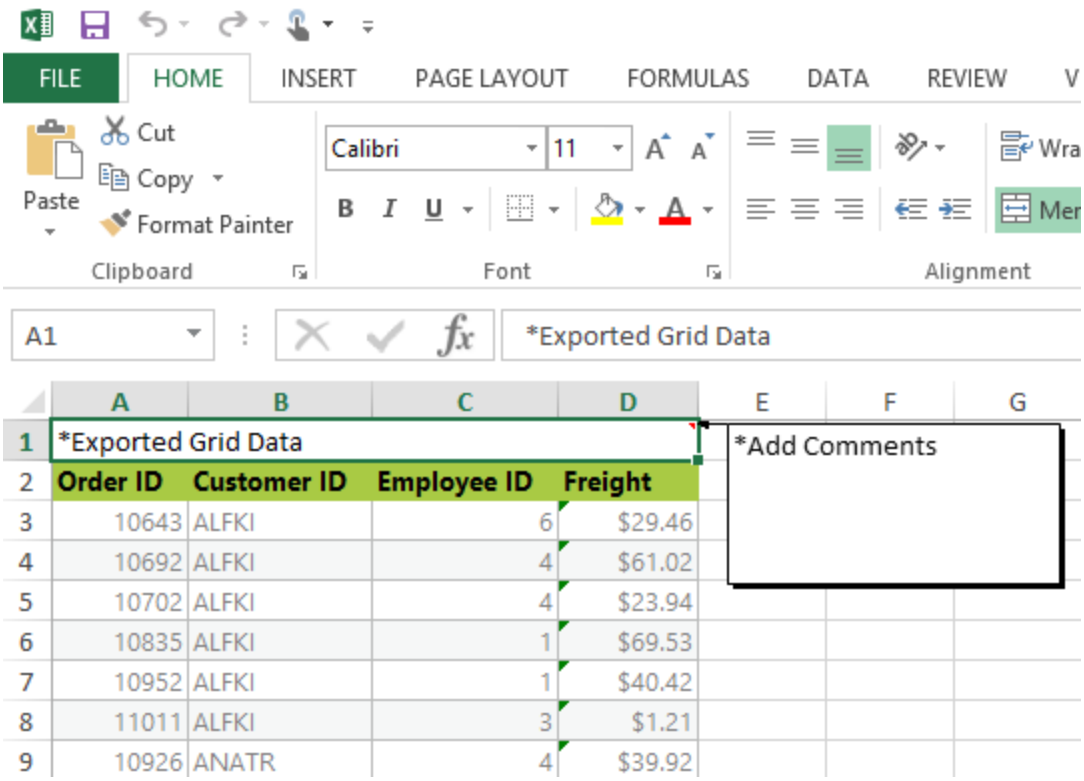
```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExportToExcel(string GridModel)
{
GridProperties gridProperty =
(GridProperties)Utils.DeserializeToModel(typeof(GridProperties), GridModel);
ExcelExport exp = new ExcelExport();
IEnumerable data = new NorthwindDataContext().OrdersViews.ToList();
IWorkbook book = exp.Export(gridProperty, (IEnumerable)data, "Export.xlsx",
ExcelVersion.Excel2010, false, false, "flat-lime", true);
book.ActiveSheet.Range["A1:D1"].Merge();
book.ActiveSheet.Range["A1"].Text = "*Exported Grid Data";//Adding text to
the cells of particular range
book.ActiveSheet.Range["A1"].AddComment().Text = "*Add Comments";//Adding
comments to cells of particular range
book.SaveAs("Export.xlsx", ExcelSaveType.SaveAsXLS,
System.Web.HttpContext.Current.Response, ExcelDownloadType.Open);
}
}
```

The following screenshot displays the exported grid with comments added to cells

Copyright © 2001 - 2020 Syncfusion Inc.



Adding comments in exported Word document

We can add comments to the word document using the AppendComment method in the WComment class of the DocIO namespace. We can traverse through the required row/cell with the help of WTable class and thus can append the comment to the table cell.

RAZOR

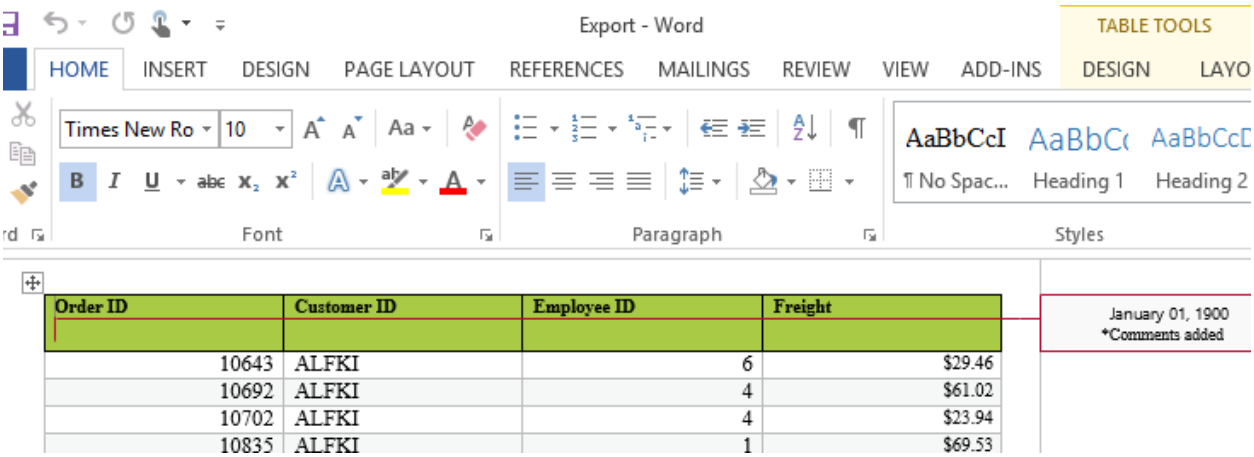
```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.WordExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
```

```
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties gridProperty =
(GridProperties)Utils.DeserializeToModel(typeof(GridProperties), GridModel);
IWordDocument document = exp.Export(gridProperty, (IEnumerable)data,
"Export.docx", false, false, "flat-lime", true);
var table = document.Sections[0].Tables[0];
table.AddRow();//Add new row to the grid table
var para = table.Rows[0].Cells[0].AddParagraph().AppendComment("*Comments
added");
document.Save("Export.docx", FormatType.Docx,
System.Web.HttpContext.Current.Response, HttpContentDisposition.Attachment);
}
}
```

The following screenshot shows the exported grid with comments added to the document.



Adding comments in exported PDF document

We can add comments to a PDF documents using the annotation support provided in the PDF namespace. The annotation added can be of any format such as freeTextAnnotation, popUpAnnotation and LineAnnotation etc., based on the requirement we can add annotation which is available under Syncfusion.Pdf.Interactive namespace.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.Columns(col =>
{
```

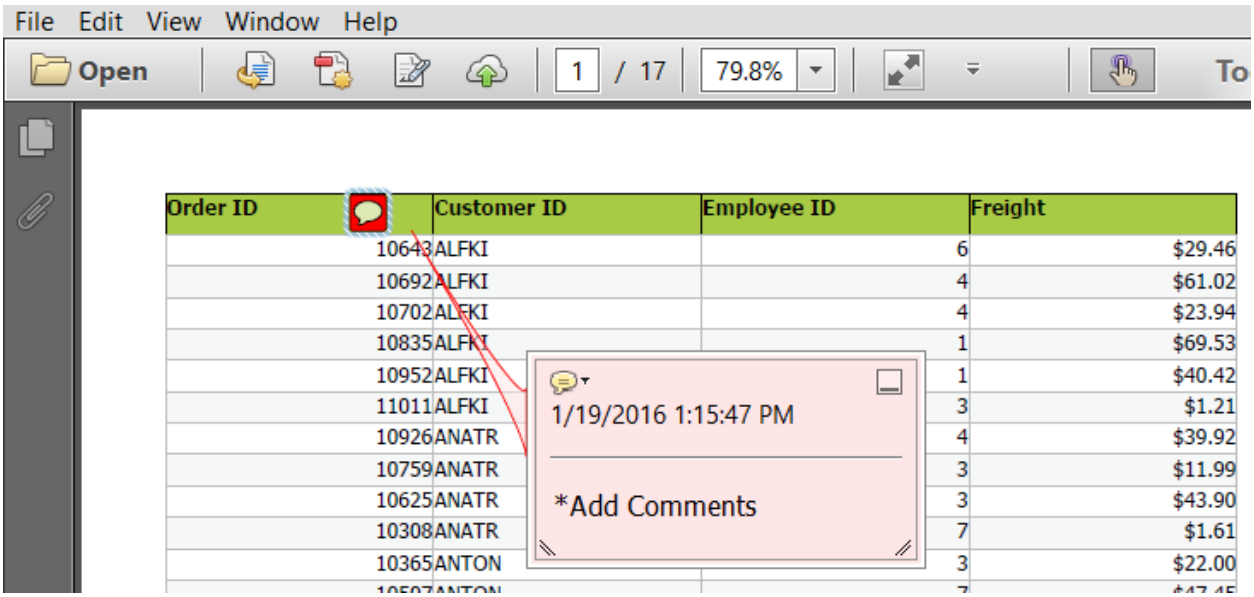
```
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties gridProperty =
(GridProperties)Utils.DeserializeToModel(typeof(GridProperties), GridModel);
PdfDocument document = exp.Export(gridProperty, (IEnumerable)data,
"Export.pdf", false, false, "flat-lime", true);
RectangleF rectangle = new RectangleF(120, 20, 20, 125);
//Creates a new pop-up annotation.
PdfPopupAnnotation popupAnnotation = new PdfPopupAnnotation(rectangle,
"*Comments added");
popupAnnotation.Border.Width = 2;
popupAnnotation.Border.HorizontalRadius = 10;
popupAnnotation.Border.VerticalRadius = 20;
//Sets the PDF pop-up icon.
popupAnnotation.Icon = PdfPopupIcon.Comment;
//Adds the annotation to page.
document.Pages[0].Annotations.Add(popupAnnotation);
document.Save("Export.pdf", Response, HttpReadType.Save);
}
}
```

The following screenshot displays the exported grid with comments added to cells.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding header and footer in the exported file (Excel, Word or PDF)

We can add header and footer in the exported file while exporting the grid control.

Adding header and footer in Excel sheet

Using the Range Text property and SetValue method of the XlsIO IRange Class, we can add headers and footers in Excel sheet.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public void ExportToExcel(string GridModel)
{
GridProperties gridProperty =
(GridProperties)Utils.DeserializeToModel(typeof(GridProperties), GridModel);
ExcelExport exp = new ExcelExport();
IEnumerable data = new NorthwindDataContext().OrdersViews.ToList();
IWorkbook book = exp.Export(gridProperty, (IEnumerable)data, "Export.xlsx",
ExcelVersion.Excel2010, false, false, "flat-lime", true);
book.ActiveSheet.InsertRow(1);
// Merging the sheet from Range A1 to D1 for adding title space
book.ActiveSheet.Range["A1:D1"].Merge();
//Adding the title using Text property
book.ActiveSheet.Range["A1"].Text = "Grid Order Data";
book.ActiveSheet.Range["A1"].CellStyle.HorizontalAlignment =
ExcelHAlign.HAlignCenter;//set text alignment
//Adding footer using SetValue method
book.ActiveSheet.SetValue(book.ActiveSheet.Rows.Length + 2,
book.ActiveSheet.Columns.Length - 3, "CopyRights");
book.SaveAs("Export.xlsx", ExcelSaveType.SaveAsXLS,
System.Web.HttpContext.Current.Response, ExcelDownloadType.Open);
}
}
```

Adding header and footer in exported Word document

We can add header and footer to the word document using the HeadersFooters property in the WHeadersFooters class of the DocIO namespace. We can create an instance of the IWParagraph class and append the header/footer text to it using the AppendText method.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.WordExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
}))
```

C#

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties gridProperty =
(GridProperties)Utils.DeserializeToModel(typeof(GridProperties), GridModel);
IWordDocument document = exp.Export(gridProperty, (IEnumerable)DataSource,
"Export.docx", false, false, "flat-lime", true);
IWParagraph para = new WParagraph(document);
//Add header to the word document
para = document.Sections[0].HeadersFooters.Header.AddParagraph();
//Insert the header text using AppendText method
para.AppendText("[Header]");
//Add footer to the word document
para = document.Sections[0].HeadersFooters.Footer.AddParagraph();
//Insert the footer text using AppendText method
para.AppendText("[Footer]");
//Adding Title to the Grid
var index =
document.LastSection.Body.ChildEntities.IndexOf(document.LastSection.Tables[
0]);
WParagraph para1 = new WParagraph(document);
para1.Text = "Grid Title";
document.LastSection.Body.ChildEntities.Insert(index, para1);
document.Save("Export.docx", FormatType.Docx,
System.Web.HttpContext.Current.Response, HttpContentDisposition.Attachment);
}
}
```

Adding header and footer in the exported PDF document

We can add header/footer to a PDF documents using the PdfPageTemplateElement class. The header and footer can contain any types of element including dynamic fields.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.PdfExport);
}))
.AllowPaging()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right). Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
}))
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public partial class GridController : Controller
{
public ActionResult ExportingGrid()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
public ActionResult ExportToPdf(string GridModel)
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties gridProperty =
(GridProperties)Utils.DeserializeToModel(typeof(GridProperties), GridModel);
PdfExport exp = new PdfExport();
PdfDocument document = exp.Export(gridProperty, (IEnumerable)DataSource,
"Export.pdf", false, false, "flat-lime", true);
RectangleF rect = new RectangleF(0, 0, document.PageSettings.Width, 50);
//create a header pager template
PdfPageTemplateElement header = new PdfPageTemplateElement(rect);
//create a footer pager template
PdfPageTemplateElement footer = new PdfPageTemplateElement(rect);
Font f = new Font("Helvetica", 10, System.Drawing.FontStyle.Bold);
PdfFont font = new PdfTrueTypeFont(f, true);
header.Graphics.DrawString("Demo Report", font, PdfBrushes.Black, new
Point(250, 0)); //Add custom text to the Header
document.Template.Top = header; //Append custom template to the document
footer.Graphics.DrawString("CopyRights", font, PdfBrushes.Gray, new
Point(250, 0));//Add Custom text to footer
document.Template.Bottom = footer;//Add the footer template to document
document.Save(Server.MapPath("/Output/Export.pdf"));
return RedirectToAction("Index");
}
}
```

External search in grid

Using search method of grid, you can search the string in grid externally without using in-built toolbar search support. While using the search method it is necessary to set the AllowSearching property as true. The following code example explains the previous behavior.

RAZOR

```
<input type="text" id="search" class="e-ejinputtext" />
@(Html.EJ().Button("search")
.Text("Searching")
.ClientSideEvents(eve => { eve.Click("onSearching"); })
)
@(Html.EJ().Grid<OrdersView>("Grid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSearching()
.Columns(col =>
{
col.Field("OrderID").Add();
col.Field("CustomerID").Add();
col.Field("EmployeeID").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("Freight").Add();
col.Field("ShipCity").Add();
col.Field("ShipCountry").Add();
})
)
```

JS

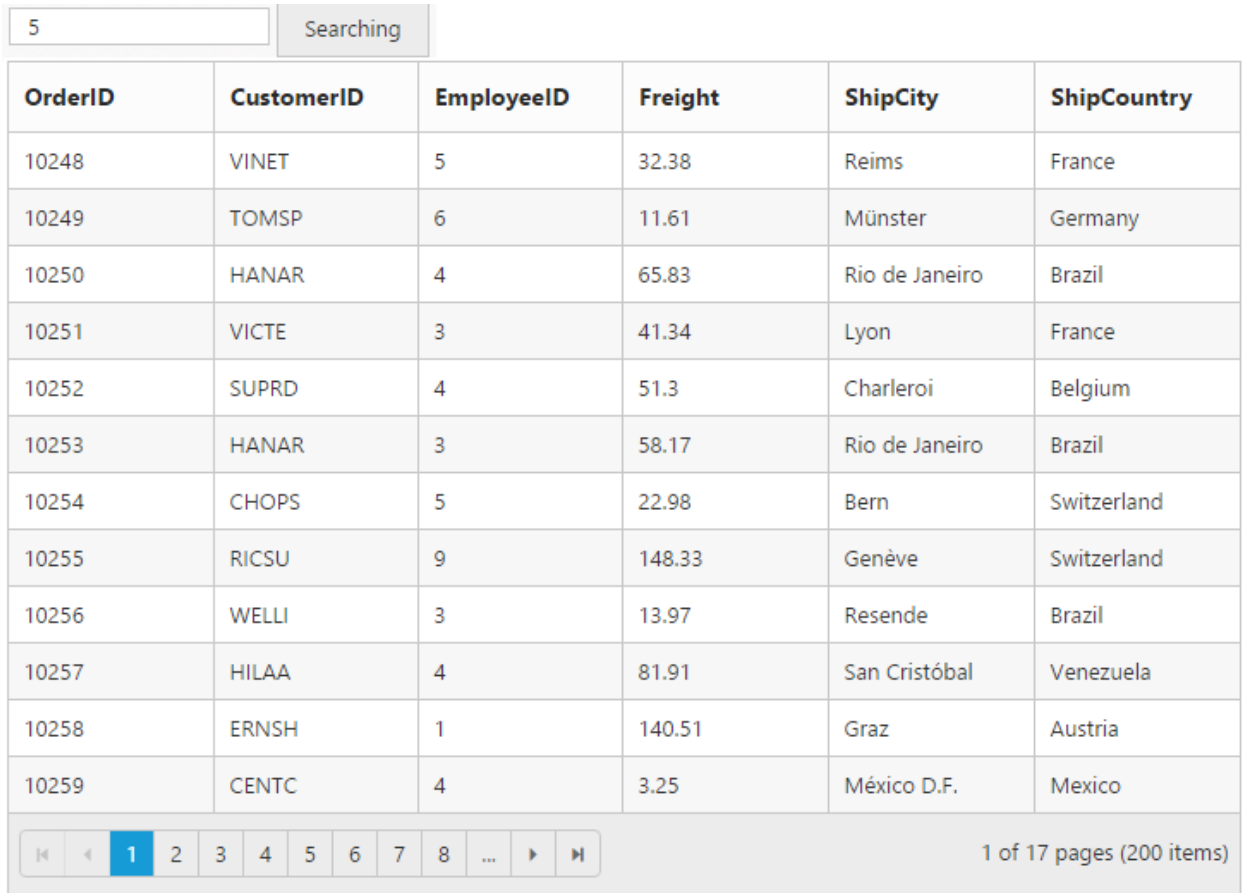
```
function onSearching(args) {
var obj = $("#Grid").ejGrid("instance");
var val = $("#search").val();
obj.search(val);
}
```

C#

```
namespace SyncfusionMvcApplication3.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = OrderRepository.GetAllRecords();
return View();
}
}
}
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Configure grid through the grid model properties

There comes a time when you want to set grid properties other than using grid builder i.e., in the Controller side. At the time you can pass GridProperties Model instance as a parameter to grid Helper.

We can set grid properties in server-side using the GridProperties class and use it in view page using grid Helper overload.

In addition to id parameter of grid helper we can also pass GridProperties Model as another parameter to grid helper.

The following code example shows you how to set grid properties such as AllowPaging, DataSource and Columns in Controller using GridProperties class.

RAZOR

```
@model Syncfusion.JavaScript.Models.GridProperties
@(Html.EJ().Grid<object>("FlatGrid",Model))
```

C#

```
namespace EJGrid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
```

```
List<Column> cols = new List<Column>();
cols.Add(new Column() { Field = "OrderID" });
cols.Add(new Column() { Field = "EmployeeID" });
cols.Add(new Column() { Field = "ShipCity" });
cols.Add(new Column() { Field = "ShipCountry" });
cols.Add(new Column() { Field = "Freight" });
GridProperties prop = new GridProperties();
prop.DataSource = OrderRepository.GetAllRecords();
prop.Columns = cols;
prop.AllowPaging = true;
return View(prop);
}
}
}
```

The following output is displayed as a result of the previous code example.

Work with partial views

The grid can be rendered in either Unobtrusive or Non-unobtrusive mode.

In unobtrusive mode, to load grid from partial view via AJAX, in the success callback, the partial view should be parsed using the ej.widget.init method.

Note: The ej.widget.init method is available in ej.unobtrusive.min.js file.

Note: If the InitUnobtrusiveInScriptManager key is set in appSettings of web config then the Html.EJ().ScriptManager() should be placed in the partial view.

The following code example describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("PartialGrid")
.Datasource((IEnumerable<OrdersView>)Model)
.AllowPaging()
.Columns(col =>
{
col.Field(o => o.OrderID).HeaderText("Order
ID").Width(70).TextAlign(TextAlign.Right).Add();
col.Field(o => o.EmployeeID).HeaderText("Employee
ID").Width(70).TextAlign(TextAlign.Right).Add();
col.Field(o => o.CustomerID).HeaderText("Customer ID").Width(70).Add();
col.Field(o =>
o.Freight).HeaderText("Freight").Width(70).TextAlign(TextAlign.Right).Add();
})
)
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
```

```
{
return View();
}
public ActionResult GetPartial()
{
var model = new NorthwindDataContext().OrdersViews.ToList();
return PartialView("_GridPartial", model);
}
}
}
```

HTML

```
<div id="target"></div>
```

JS

```
<script>
$(function(){
$.ajax({
url:'@Url.Action("GetPartial")',
success: function(doc){
$("#target").html(doc);
ej.widget.init($("#target"));
}
});
});
</script>
```

The following output is displayed as a result of the previous code example.

In non-unobtrusive mode, to load grid from partial view via AJAX, the ScriptManager HTML helper should be placed at the bottom of the partial view.

Note: The Html.EJ().ScriptManager() which is placed in the Layout page will not initiate control rendering in partial view hence every partial view should have their own ScriptManager.

The following code sample describes the previous behavior.

RAZOR

```
@(Html.EJ().Grid<OrdersView>("PartialGrid")
.Datasource((IEnumerable<OrdersView>)Model)
.AllowPaging()
.Columns(col =>
{
col.Field(o => o.OrderID).HeaderText("Order
ID").Width(70).TextAlign(TextAlign.Right).Add();
col.Field(o => o.EmployeeID).HeaderText("Employee
ID").Width(70).TextAlign(TextAlign.Right).Add();
col.Field(o => o.CustomerID).HeaderText("Customer ID").Width(70).Add();
col.Field(o =>
o.Freight).HeaderText("Freight").Width(70).TextAlign(TextAlign.Right).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
@(Html.EJ().ScriptManager())
```

C#

```
namespace Grid.Controllers
{
public class HomeController : Controller
{
public ActionResult Index()
{
return View();
}
public ActionResult GetPartial()
{
var model = new NorthwindDataContext().OrdersViews.ToList();
return PartialView("_GridPartial", model);
}
}
}
```

HTML

```
<div id="target"></div>
```

JS

```
<script>
$(function(){
$.ajax({
url:'@Url.Action("GetPartial")',
success: function(doc){
$("#target").html(doc);
}
});
});
</script>
```

The following output is displayed as a result of the previous code example.

Rendering the grid using grid properties in server-side

The GridProperties helps to render the grid control in server-side.

The following code example which will explain to render the grid control in server-side.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid", Model))
```

C#

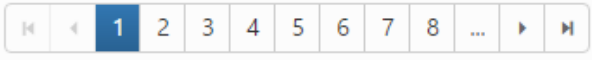
```
namespace Grid.Controllers
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public class GridController : Controller
{
public ActionResult GridFeatures()
{
GridProperties grid = new GridProperties();
List<Column> colList = new List<Column>();
colList.Add(new Column() { Field = "OrderID", HeaderText="Order ID",
TextAlign = Syncfusion.JavaScript.TextAlign.Right, Width = 75 });
colList.Add(new Column() { Field = "CustomerID", HeaderText = "Customer ID",
Width = 80 });
colList.Add(new Column() { Field = "ShipName", HeaderText = "Ship Name",
Width = 100 });
colList.Add(new Column() { Field = "ShipCity", HeaderText = "Ship City",
Width = 100 });
colList.Add(new Column() { Field = "Freight", HeaderText = "Freight", Width
= 80, TextAlign = Syncfusion.JavaScript.TextAlign.Right });
grid.Columns = colList;
grid.AllowPaging = true;
grid.DataSource = OrderRepository.GetAllRecords().ToList();
return View(grid);
}
}
}
```

The following output is displayed as a result of the previous code example.

Order ID	Customer ID	Ship Name	Ship City	Freight
10248	VINET	Vins et alcools Chevalier	Reims	$32.380
10249	TOMSP	Toms Spezialitäten	Münster	$11.610
10250	HANAR	Hanari Carnes	Rio de Janeiro	$65.830
10251	VICTE	Victuailles en stock	Lyon	$41.340
10252	SUPRD	Suprêmes délices	Charleroi	$51.300
10253	HANAR	Hanari Carnes	Rio de Janeiro	$58.170
10254	CHOPS	Chop-suey Chinese	Bern	$22.980
10255	RICSU	Richter Supermarkt	Genève	$148.330
10256	WELLI	Wellington Importadora	Resende	$13.970
10257	HILAA	HILARION-Abastos	San Cristóbal	$81.910
10258	ERNSH	Ernst Handel	Graz	$140.510
10259	CENTC	Centro comercial Moctez	México D.F.	$3.250



1 of 17 pages (200 items)

Copyright © 2001 - 2020 Syncfusion Inc.

Export grid to target location without download prompt(Excel, Word or PDF)

In the Export method we have assigned the target location to save the exported file in the specified path. The target location is placed using the Server.MapPath.

The following code example shows how to save the exported files in a target location.

RAZOR

```
@(Html.EJ().Grid<EmployeeView>("MasterGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource)
.AllowPaging()
.ToolbarSettings(toolBar => toolBar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.ExcelExport);
items.AddTool(ToolBarItems.WordExport);
items.AddTool(ToolBarItems.PdfExport);
}))
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Add();
col.Field("CustomerID").HeaderText("Customer ID").Add();
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Add();
})
)
```

C#

```
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
ViewBag.DataSource = DataSource;
return View();
}
public void ExportToExcel(string GridModel)
{
ExcelExport exp = new ExcelExport();
string targetFolder = Server.MapPath("") + "\\New folder\\";     // Set the
target folder using Server.MapPath
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, false,
false, "flat-saffron", true, targetFolder);    //specify the path which
helps to save the excel file in local folder
}
public void ExportToWord(string GridModel)
{
WordExport exp = new WordExport();
string targetFolder = Server.MapPath("") + "\\New folder\\";   // Set the
target folder using Server.MapPath
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
GridProperties obj =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
exp.Export(obj, DataSource, "Export.docx", false, false, "flat-saffron",
true, targetFolder);   //specify the path which helps to save the excel file
in local folder
}
public void ExportToPdf(string GridModel)
{
PdfExport exp = new PdfExport();
string targetFolder = Server.MapPath("") + "\\New folder\\";    // Set the
target folder using Server.MapPath
var DataSource = new NorthwindDataContext().OrdersViews.ToList();
GridProperties obj =
(GridProperties)Syncfusion.JavaScript.Utils.DeserializeToModel(typeof(GridPr
operties), GridModel);
exp.Export(obj, DataSource, "Export.pdf", false, false, "flat-saffron",
true, targetFolder);     //specify the path which helps to save the excel
file in local folder
}
```

See Also

For more information on enable unobtrusive please refer this link.

Hierarchy grid with different foreignKeyField in parent and child table

The QueryString property is used to filter the childGrid data based on value in parent grid data. But when the field name provided in the QueryString does not exists in Child grid, then the ForeignKeyField property is used to filter the childGrid data. If the foreign key column name differs for parent and child grid then use the ForeignKeyField property of grid.

The following code example explains the previous behavior.

RAZOR

```
@(Html.EJ().Grid<EmployeeView>("HierarchyGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.Field("EmployeeID").HeaderText("Employee
ID").TextAlign(TextAlign.Right).Width(85).Add();
col.Field("FirstName").HeaderText("First Name").Width(100).Add();
col.Field("City").Width(100).Add();
col.Field("Country").Width(100).Add();
})
.ChildGrid(child =>
{
child.Datasource("http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.sv
c/Orders")
.QueryString("FirstName")
.ForeignKeyField("CustomerName")
.AllowPaging()
.PageSettings(page => page.PageSize(5))
.Columns(col =>
{
```

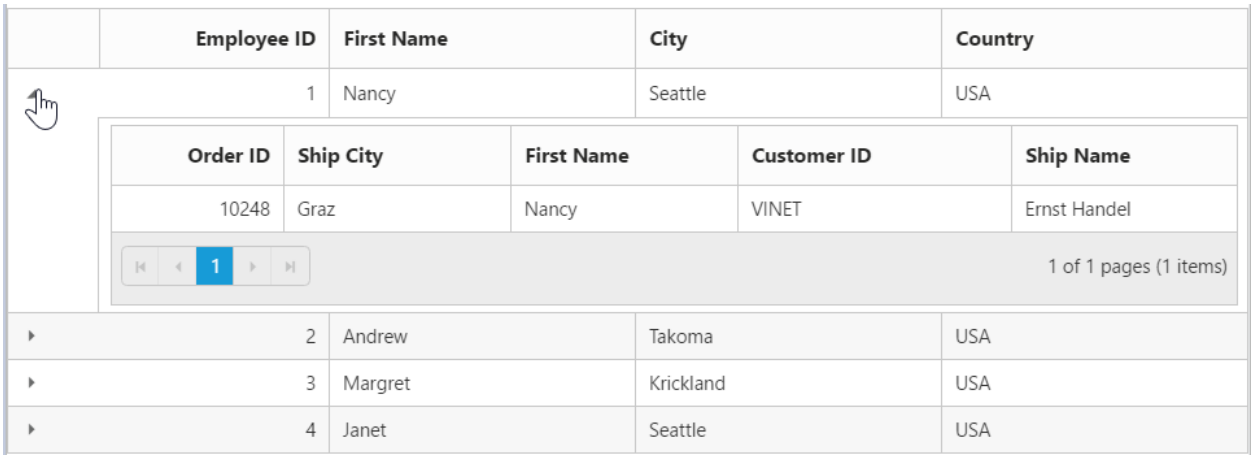
Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("OrderID").HeaderText("Order
ID").TextAlign(TextAlign.Right).Width(75).Add();
col.Field("ShipCity").HeaderText("Ship City").Width(100).Add();
col.Field("CustomerName").HeaderText("First Name").Width(100).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(120).Add();
col.Field("ShipName").HeaderText("Ship Name").Width(100).Add();
});
})
)
```

C#

```
public partial class GridController : Controller
{
//
// GET: /HierarchyGrid/
public ActionResult HierarchyGrid()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

The following output is displayed as a result of the previous code example.



Display other Syncfusion controls in grid columns

We can display the other Syncfusion controls using the Template property of grid columns and TemplateRefresh event of ejGrid control.

RAZOR

```
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.ClientSideEvents(event => event.TemplateRefresh("template"))
.Columns(col =>
{
col.HeaderText("Employee
Rating").Template("#columnTemplate").Width(100).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("EmployeeID").HeaderText("Employee ID").Width(90).Add();
col.Field("FirstName").HeaderText("First Name").Width(90).Add();
col.Field("LastName").HeaderText("Last Name").Width(90).Add();
col.Field("Country").HeaderText("Country").Width(80).Add();
})
)
```

C#

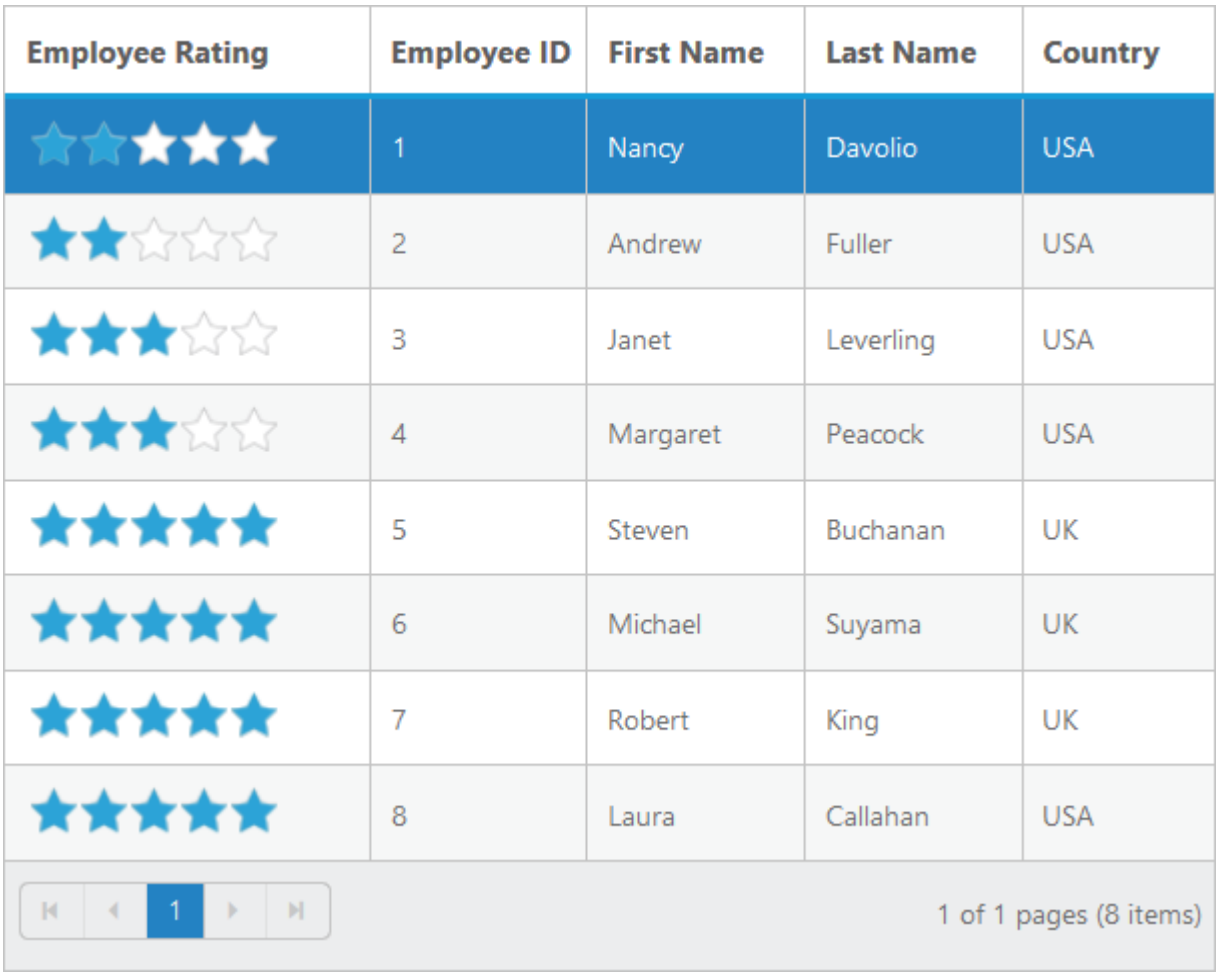
```
namespace Grid.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/x-jsrender" id="columnTemplate">
{{"{{"}}if EmployeeID<3{{}}}}
<input type="text" class="rating" value="3" />
{{"{{"}}else EmployeeID>2 && EmployeeID<5{{}}}}
<input type="text" class="rating" value="3" />
{{"{{"}}else EmployeeID>4{{}}}}
<input type="text" class="rating" value="5" />
{{"{{"}}/if{{}}}}
</script>
<script type="text/javascript">
function template(args) {
$(args.cell).find(".rating").ejRating({ allowReset: false });
}
</script>
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Perform grid actions on external button click

CRUD operations

Using the addRecord method of grid, you can add a record to a grid externally without using in-built toolbar add support. While using the addRecord method it is necessary to set the AllowAdding property as true.

Using deleteRecord method of grid, you can delete a record to a grid externally without using in-built toolbar delete support. While using the deleteRecord method it is necessary to set the AllowDeleting property as true.

Using updateRecord method of grid, you can update a record to a grid externally without using in-built toolbar update support. While using the updateRecord method it is necessary to set the AllowEditing property as true.

Filtering

Using the filterColumn method of grid, you can filter the data in the grid externally without using in-built filter support. While using filterColumn method it is necessary to set AllowFiltering property as true.

Copyright © 2001 - 2020 Syncfusion Inc.

Grouping

Using the groupColumn and ungroupColumn method of grid, you can group/ungroup the grid externally without using in-built grouping support. While using the groupColumn and ungroupColumn method it is necessary to set the AllowGrouping property as true.

Sorting

Using the sortColumn method of grid, you can sort the grid externally without using in-built sorting support. While using the sortColumn method it is necessary to set the yAllowSorting property as true.

The following code example explains the previous behavior.

HTML

```
<table>
<tr>
<td>
<b>CRUD</b><br>@(Html.EJ().Button("AddRecord")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("addRecord"); })
.Text("AddRecord")
)
@(Html.EJ().Button("UpdateRecord")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("updateRecord"); })
.Text("UpdateRecord")
)
@(Html.EJ().Button("DeleteRecord")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("deleteRecord"); })
.Text("DeleteRecord")
)
</td>
<td>
<b>Filtering</b><br><br>
@(Html.EJ().DropDownList("FilterColumnOne")
.TargetID("Order")
.Width("120px")
.WatermarkText("Select filter value one")
.SelectedIndex(0)
)
@(Html.EJ().DropDownList("FilterColumnTwo")
.TargetID("Employee")
.Width("120px")
.WatermarkText("Select filter value two")
.SelectedIndex(0)
)
@(Html.EJ().Button("filter")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("Filter"); })
.Text("Filter")
)
@(Html.EJ().Button("ClearFilter")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("clearFilter"); })
.Text("Clear")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</td>
<div
id="Order"><ul><li>10001</li><li>10249</li><li>10250</li><li>10251</li><li>1
0252</li></ul></div>
<div
id="Employee"><ul><li>1</li><li>2</li><li>3</li><li>4</li><li>5</li></ul></d
iv>
<td>
<b>Grouping</b><br><br>
<div id="columnName">
<ul>
<li>OrderID</li>
<li>CustomerID</li>
<li>Freight</li>
<li>Verified</li>
<li>ShipName</li>
</ul>
</div><br>
@(Html.EJ().DropDownList("GroupColumnName")
.TargetID("columnName")
.Width("115px")
.SelectedIndex(0)
.ClientSideEvents(eve => { eve.Change("OnChange"); })
)
@(Html.EJ().Button("groupColumn")
.Width("100px")
.Size(ButtonSize.Medium)
.ClientSideEvents(eve => { eve.Click("clickToGroup"); })
.Text("GroupColumn")
)
@(Html.EJ().Button("unGroupColumn")
.Width("115px")
.Size(ButtonSize.Medium)
.ClientSideEvents(eve => { eve.Click("clickToGroup"); })
.Text("UnGroupColumn")
)
</td>
<td>
<b>Sorting</b><br><br>
<div id="sortColumnName">
<ul>
<li>Order ID</li>
<li>Customer ID</li>
<li>Employee ID</li>
<li>Freight</li>
<li>Ship City</li>
</ul>
</div>
@(Html.EJ().DropDownList("SortColumnName")
.TargetID("sortColumnName")
.Width("120px")
.SelectedIndex(0)
)
<div id="directions">
<ul>
<li>Ascending</li>
<li>Descending</li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</ul>
</div>
@(Html.EJ().DropDownList("Directions")
.TargetID("directions")
.Width("120px")
.SelectedIndex(0)
)
@(Html.EJ().Button("doSorting")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("Sort"); })
.Text("Sort")
)
@(Html.EJ().Button("clearSort")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("Sort"); })
.Text("Clear")
)
</td>
</tr>
</table>
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowGrouping()
.AllowFiltering()
.AllowPaging()
.AllowSorting()
.EditSettings(edit => { edit.AllowAdding().AllowDeleting().AllowEditing();
})
.ToolbarSettings(toolbar =>
{
toolbar.ShowToolbar().ToolbarItems(items =>
{
items.AddTool(ToolBarItems.Add);
items.AddTool(ToolBarItems.Edit);
items.AddTool(ToolBarItems.Delete);
items.AddTool(ToolBarItems.Update);
items.AddTool(ToolBarItems.Cancel);
});
})
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order
ID").IsPrimaryKey(true).TextAlign(TextAlign.Right).Width(150).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(200).Add();
col.Field("EmployeeID").HeaderText("Employee ID").Width(150).Add();
col.Field("Freight").HeaderText("Freight").TextAlign(TextAlign.Right).Width(
150).Format("{0:C}").Add();
col.Field("OrderDate").HeaderText("Order Date").Width(200).Add();
})
)
```

C#

```
public partial class GridController : Controller
{
//
```

```
// GET: /ExternalButton/
public ActionResult ExternalButton()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

JS

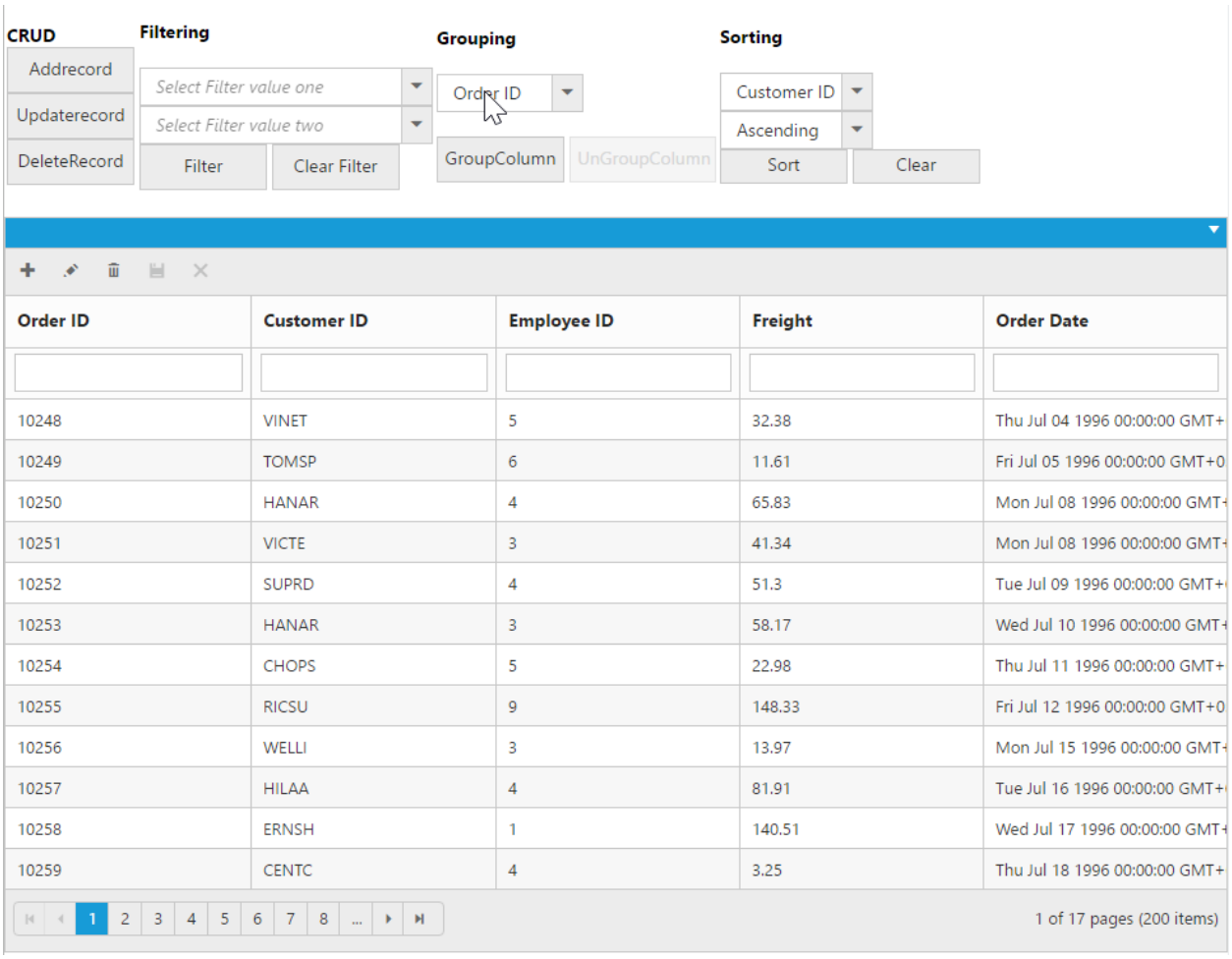
```
<script type="text/javascript">
function addRecord(){
var gridObj = $('#FlatGrid').data("ejGrid");
gridObj.addRecord({ "OrderID": 12333 });
}
function deleteRecord(){
var gridObj = $('#FlatGrid').data("ejGrid");
gridObj.deleteRecord("OrderID", { OrderID:
gridObj.model.dataSource[gridObj.model.selectedRowIndex].OrderID });
}
function updateRecord(){
var gridObj = $('#FlatGrid').data("ejGrid");
gridObj.updateRecord("OrderID", { OrderID: 10002, EmployeeID: 6 });
}
function Filter(args) {
var obj = $('#FlatGrid').data("ejGrid");
var one = $('#FilterColumnOne').data("ejDropDownList");
var two = $('#FilterColumnTwo').data("ejDropDownList");
var One = one.getValue();
var Two = two.getValue();
obj.filterColumn([{field:"OrderID",operator:"equal",value:One,predicate:"and
",
matchcase:true},{field:"EmployeeID",operator:"equal",value:Two,predicate:"an
d", matchcase:true}]);
}
function clearFilter(args) {
var gridObj = $("#FlatGrid").data("ejGrid");
gridObj.clearFiltering();
}
function Sort(args) {
var gridObj = $("#FlatGrid").data("ejGrid");
if (gridObj != undefined) {
gridObj.clearSorting();
if (this.element.attr('id').indexOf("doSorting") != -1) {
var name = $('#SortColumnName').data("ejDropDownList");
var direction = $('#Directions').data("ejDropDownList");
var columnName = name.getValue().replace(/\s*/g, "");
var sortDirection = direction.getValue().toLowerCase();
gridObj.sortColumn(columnName, sortDirection);
}
}
}
function clickToGroup(args) {
var gridObj = $("#FlatGrid").data("ejGrid");
var columnName = $("#GroupColumnName").ejDropDownList("getSelectedValue");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (this.element.attr("id") == "groupColumn") {
gridObj.groupColumn(columnName);
$("#groupColumn").ejButton("disable");
$("#unGroupColumn").ejButton("enable");
}
else {
gridObj.ungroupColumn(columnName);
group = true;
$("#unGroupColumn").ejButton("disable");
$("#groupColumn").ejButton("enable");
}
}
function OnChange() {
var gridObj = $("#FlatGrid").data("ejGrid");
var columnName = $("#GroupColumnName").ejDropDownList("getSelectedValue");
if ($.inArray(columnName, gridObj.model.groupSettings.groupedColumns) != -1)
{
$("#unGroupColumn").ejButton("enable");
$("#groupColumn").ejButton("disable");
}
else {
$("#groupColumn").ejButton("enable");
$("#unGroupColumn").ejButton("disable");
}
}
</script>
```

The following output is displayed as a result of the previous code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Getting Datasource of Grid in Sorted Order

Grid column can be sorted and after sorting, the datasource can be obtained in the same order using `sortBy` query and `executeLocal` method of DataManager.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Button("Sorting")
.Width("100px")
.ClientSideEvents(eve => { eve.Click("GetSortedData"); })
.Text("GetSortedData")
)
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.datasource)
.AllowPaging()
.AllowSorting()
.AllowMultiSorting()
.Columns(col =>
{
col.Field("OrderID").HeaderText("Order ID").Width(75).Add();
col.Field("CustomerID").HeaderText("Customer ID").Width(80).Add();
col.Field("EmployeeID").HeaderText("Employee ID").Width(75).Add();
col.Field("ShipCity").HeaderText("Verified").Width(110).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
```

C#

```
namespace Grid.Controllers
{
public class GridController : Controller
{
public ActionResult GridFeatures()
{
var DataSource = new NorthwindDataContext().EmployeeViews.ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JS

```
<script type="text/javascript">
function GetSortedData(args) {
var obj = $("#FlatGrid").ejGrid("instance");
var Sort = obj.model.sortSettings.sortedColumns;
var query = ej.Query();
if(obj.model.sortSettings.sortedColumns.length){
for(var i=Sort.length-1;i>=0;i--){
query.sortBy(Sort[i].field, Sort[i].direction);
}
var SortedDatasource =
ej.DataManager(obj.model.dataSource).executeLocal(query);
}
}
</script>
```

Note: This solution is applicable only for local data.

GroupButton

Overview

The Group button control is a collection of buttons grouped together that allows you to select the items and perform action based on the item selected. It can be customized to behave either as RadioButtons or Checkboxes. You can add as many group button items as possible by connecting it to local or remote datasource.

You can customize the groupbutton items by using the spriteCSS, Image position properties available in Groupbutton.

Key Features

- Trendy Look: Rich Appearance with Theme Support
- RTL: Supports for Right to Left alignment of Groupbutton items.
- Text and Image: Supports both text and image as Groupbutton item content
- Built-in Icon: Supports the built-in icon libraries

Copyright © 2001 - 2020 Syncfusion Inc.

- Easy Customization: The customization of Groupbutton control to any form is made simple

Getting Started

This section provides details on how to create and customize Groupbutton control in an ASP.NET MVC application.

Dependencies

The following assemblies are needs to referenced to render a Groupbutton control in your application.

1. Syncfusion.EJ.dll
2. Syncfusion.EJ.MVC.dll

The script and CSS dependencies for rendering Groupbutton is given in the JS help document.

Create a simple GroupButton control

Create an ASP.NET MVC project and add necessary assemblies and script dependencies with the help of given Getting started document.

Add the below code in the view page to render a simple groupbutton control.

HTML

```
@Html.EJ().GroupButton("GroupButton").Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```

Run the project to get the following output



Behavior settings

Groupbutton provides options through which you can customize the default behavior of Groupbutton control.

Groupbutton Mode

The default behavior of Groupbutton is like Radiobutton, i.e., only one item can be selected at a time.

HTML

```
@Html.EJ().GroupButton("GroupButton").GroupButtonMode(GroupButtonMode.RadioB
utton).Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

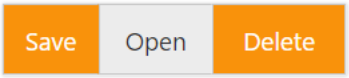


RadioButton mode {:.caption}

To enable selection of one or more Groupbutton items at a time, you can set the Groupbutton mode to Checkbox. With this, you can toggle the each button's state to perform actions, since all the button will behave like individual buttons.

HTML

```
@Html.EJ().GroupButton("GroupButton").GroupButtonMode(GroupButtonMode.CheckB
ox).Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```



CheckBox mode {:.caption}

Easy Customization

The GroupButton wrapper and each GroupButton items can be easily customized according to your need using the options available in Groupbutton control.

Button Size

You can render the Groupbutton in different sizes by using the predefined size options for rendering a groupbutton with different sizes. Each size option has different height and width. Mainly it avoids the complexity in rendering groupbutton with complex CSS class.

Normal	Creates button with content size.
Mini	Creates button with Built-in mini size height, width specified.
Small	Creates button with Built-in small size height, width specified.
Medium	Creates button with Built-in medium size height, width specified.
Large	Creates button with Built-in large size height, width specified.

Apart from the above mentioned predefined size option, you can set your own width and height of GroupButton by using Height and Width property.

In the View page, add the following GroupButton elements to configure GroupButton for configure the size

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

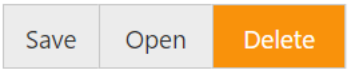
```
<%--Set the different size options and custom size Groupbutton control as
follows--%>
@Html.EJ().GroupButton("GroupButton").Size(ButtonSize.Mini).Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```

Mini sized Groupbutton {:.caption}



HTML

```
<%--Set the different size options and custom size Groupbutton control as
follows--%>
@Html.EJ().GroupButton("GroupButton").Size(ButtonSize.Medium).Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```



Medium sized Groupbutton {:.caption}

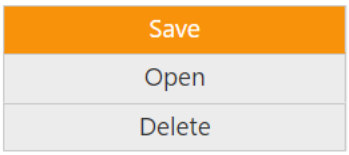
Orientation

Groupbutton element can be rendered either in vertical or horizontal orientation by means of the Orientation property available in GroupButton.

You can change the orientation of Groupbutton as given in the below code snippet.

HTML

```
@Html.EJ().GroupButton("GroupButton").Orientation(Orientation.Vertical).Item
s(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```



Copyright © 2001 - 2020 Syncfusion Inc.

DataSource

GroupButton can populate the groupbutton items based on datasource and by specifying the associated fields. You can bind data from both remote and local sources.

Refer the below table to know about the available fields

Field is a property that includes the object type. Fields are used to bind the data source and it includes following field members to make binding easier.

List of Field members

Name	Description
DataSource	Datasource receives Essential DataManager object and JSON object.
Id	Specifies the id to Groupbutton item
Text	Specifies the text of Groupbutton item
SpriteCssClass	Specifies the sprite CSS class to â€œLIâ€• item list
Url	Specifies the href attribute of "A" tag in item list
PrefixIcon	Specifies the PrefixIcon class for Groupbutton item list
SuffixIcon	Specifies the SuffixIcon class for Groupbutton item list
ImagePosition	Specifies the position of Image in Groupbutton item list
LinkAttribute	Specifies the link attribute to â€œAâ€• tag in item list
HtmlAttribute	Specifies the HTML attribute to â€œliâ€• tag in item list
ContentTemplate	Specifies the template content for customizing the Groupbutton items
Selected	Specifies initial selection state of GroupButton item
ContentType	Specifies Content type of Groupbutton Item. The possible values are 1. TextOnly 2. ImageOnly 3. ImageBoth 4. TextAndImage 5. ImageTextImage

Local data

To populate the Groupbutton items with the local data, map the field names mentioned in the above table with the property names.

1. Add the following code in your view page to render GroupButton with local data

ASPX-CS

```
// Add the following code in your CSHTML page.
```

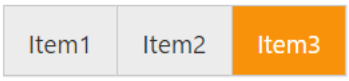
Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().GroupButton("groupbutton11").Datasource((IEnumerable<Groupbutton_
mvc.ButtonController.Groupbutton>)ViewBag.Model).GroupButtonFields(e =>
e.Text("text"))
```

C#

```
using System;
using System.Collections.Generic;
using System.Web.Mvc;
using Groupbutton_mvc.Models;
namespace Groupbutton_mvc
{
public class ButtonController: Controller
{
public class Groupbutton
{
public string text { get; set; }
public string id { get; set; }
}
public List<Groupbutton> model = new List<Groupbutton>();
public ActionResult ButtonFeatures()
{
model.Add(new Groupbutton { text = "Item1" });
model.Add(new Groupbutton { text = "Item2" });
model.Add(new Groupbutton { text = "Item3" });
ViewBag.Model = model;
return View();
}
}
}
```

The following screenshot displays the output of the above code.



Local data of GroupButton

RTL Support

In some cases, where you want to display the content in Right to Left direction, you can use RTL support by using the EnableRTL property. In RTL mode, when you have more than one content (image/text, image/image) in button, then these content are aligned in right to left format. With this property enabled, the Groupbutton items are rendered in reverse order.

In the view page, add the following Groupbutton to configure the Items with Right to left alignment

HTML

```
<%--Enable the alignment format for GroupButton control as follows--%>
@Html.EJ().GroupButton("GroupButton").EnableRTL(true).Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
```

Here the items are rendered in the reverse order.



Miscellaneous

Text

You can display your own text for Button. By using the Text property, you can easily set text content for button. This text property overwrites the text that is provided on input button element.

In the view page, add the following button elements to configure Button with text

HTML

```
<%--Set the text for button control as follows--%>
@Html.EJ().GroupButton("GroupButton").Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```



Show Rounded Corner

Customizes the corners of Groupbutton control to be rendered with Rounded corners.

In the view page, add the following button elements to get rounded Button

HTML

```
<%--Enable the rounded corner for button control as follows--%>
@Html.EJ().GroupButton("GroupButton").ShowRoundedCorner(true).Items(item =>
{
item.Add().Text("Save");
item.Add().Text("Open");
item.Add().Text("Delete");
})
```



How to?

Render GroupButton from Code behind

GroupButton can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of GroupButton properties in the controller.

C#

```
//Namespace to get the JavaScript (GroupButton) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class GroupButtonController : Controller
{
public class Groupbutton
{
public string text { get; set; }
public string id { get; set; }
}
public List<Groupbutton> model = new List<Groupbutton>();
// GET: GroupButton
public ActionResult GroupButtonFeatures()
{
model.Add(new Groupbutton { text = "Item1" });
model.Add(new Groupbutton { text = "Item2" });
model.Add(new Groupbutton { text = "Item3" });
//Initializing the GroupButton model
GroupButtonProperties BtnObj = new GroupButtonProperties();
//Initializing the datasource and other properties
BtnObj.ShowRoundedCorner=true;
BtnObj.DataSource = model;
BtnObj.GroupButtonFields.Text = "text";
//Passing GroupButton properties using the ViewData
ViewData["BtnModel"] = BtnObj;
return View();
}
}
}
```

Binding the GroupButton properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

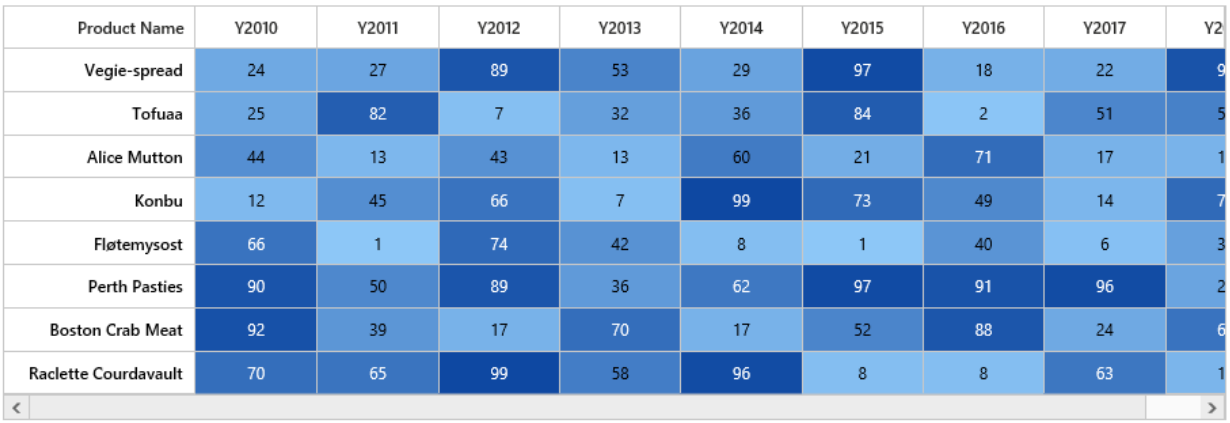
```
@{
Html.EJ().GroupButton("searchCustomer",(Syncfusion.JavaScript.Models.GroupBu
ttonProperties)ViewData["BtnModel"]).Render();
}
```

HeatMap

Overview

Essential HeatMap MVC represents tabular data values as gradient colors instead of numbers. Low and high values are different colors with different gradients.

Copyright © 2001 - 2020 Syncfusion Inc.



Product Name	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017	Y2
Vegie-spread	24	27	89	53	29	97	18	22	9
Tofuaa	25	82	7	32	36	84	2	51	5
Alice Mutton	44	13	43	13	60	21	71	17	1
Konbu	12	45	66	7	99	73	49	14	7
Fløtemysost	66	1	74	42	8	1	40	6	3
Perth Pasties	90	50	89	36	62	97	91	96	2
Boston Crab Meat	92	39	17	70	17	52	88	24	6
Raclette Courdavault	70	65	99	58	96	8	8	63	1

Key features:

- 2 types of data source mapping (TableMapping, CellMapping)
- Color mapping
- Legend
- Virtualization

Getting Started

This section explains briefly about how to create a HeatMap in your application with ASP.NET MVC.

Initialize the HeatMap

1. Create Syncfusion ASP.NET MVC application. You can refer MVC Getting Started documentation to create new project and add necessary dll's and script files.
2. Add a HeatMap control in View file as like as follows.

RAZOR

```
<div>
@Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"] as
Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
</div>
```

Prepare and Populate data

Populate product information in a collection

C#

```
public ActionResult Default()
{
CellMapping CellMapping = new CellMapping();
CellMapping.Column = new PropertyMapping() { PropertyName = "ProductName",
DisplayName = "Product Name" };
CellMapping.Row = new PropertyMapping() { PropertyName = "Year", DisplayName
= "Year" };
CellMapping.Value = new PropertyMapping() { PropertyName = "Value" };
Collection columnMapping = new Collection();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
columnMapping.Add(new HeaderMapping() { PropertyName = "Veggie-spread",
DisplayName = "Veggie-spread" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Tofu", DisplayName =
"Tofu" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Alice Mutton",
DisplayName = "Alice Mutton" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Donut", DisplayName
= "Donut" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Burger", DisplayName
= "Burger" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Perth Pasties",
DisplayName = "Perth Pasties" });
CellMapping.ColumnMapping = columnMapping;
HeaderMapping headerMapping = new HeaderMapping() { PropertyName = "Year",
DisplayName = "Year", ColumnStyle = new ColumnStyle() { Width = 105,
TextAlign = HeatMapTextAlign.Right } };
CellMapping.HeaderMapping = headerMapping;
ViewData["HeatMapModel"] = Heatmap;
return View();
}
```

Map data into HeatMap

Now data is ready, next we need to configure data source and map rows and columns to visualize.

- Prepare `ItemsMapping` add it in resource.

C#

```
public ActionResult Default()
{
CellMapping CellMapping = new CellMapping();
CellMapping.Column = new PropertyMapping() { PropertyName = "ProductName",
DisplayName = "Product Name" };
CellMapping.Row = new PropertyMapping() { PropertyName = "Year", DisplayName
= "Year" };
CellMapping.Value = new PropertyMapping() { PropertyName = "Value" };
Collection columnMapping = new Collection();
columnMapping.Add(new HeaderMapping() { PropertyName = "Veggie-spread",
DisplayName = "Veggie-spread" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Tofu", DisplayName =
"Tofu" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Alice Mutton",
DisplayName = "Alice Mutton" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Donut", DisplayName
= "Donut" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Burger", DisplayName
= "Burger" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Perth Pasties",
DisplayName = "Perth Pasties" });
CellMapping.ColumnMapping = columnMapping;
HeaderMapping headerMapping = new HeaderMapping() { PropertyName = "Year",
DisplayName = "Year", ColumnStyle = new ColumnStyle() { Width = 105,
TextAlign = HeatMapTextAlign.Right } };
CellMapping.HeaderMapping = headerMapping;
Heatmap.ItemsMapping = CellMapping;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ViewData["HeatMapModel"] = Heatmap;
return View();
}
```

- Set items source and mapping

C#

```
public ActionResult Default()
{
Heatmap.ItemsMapping = CellMapping;
ViewData["HeatMapModel"] = Heatmap;
return View();
}
public Collection GetCellSource()
{
Collection collection = new Collection();
string[] name = { "Veggie-spread", "Tofu", "Alice Mutton", "Donut",
"Burger", "Perth Pasties" };
Random random = new Random();
foreach (string item in name)
{
for (int i = 0; i < 6; i++)
{
double value = random.Next(0, random.Next(0, 100));
collection.Add(new SampleCellData() { ProductName = item, Year = "Y" + (2011
+ i), Value = value });
}
}
return collection;
}
```

Product Name	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017
Vegie-spread	43	23	51	23	87	22	59	15
Tofuaa	50	92	28	18	58	26	81	99
Alice Mutton	15	81	56	35	53	14	47	35
Konbu	22	85	49	69	25	48	18	23
Fløtemysost	35	81	78	35	90	61	2	39
Perth Pasties	42	64	81	36	3	17	84	15
Boston Crab Meat	82	76	86	25	56	33	83	90
Raclette Courdavault	74	53	19	46	88	43	83	22

Color Mapping

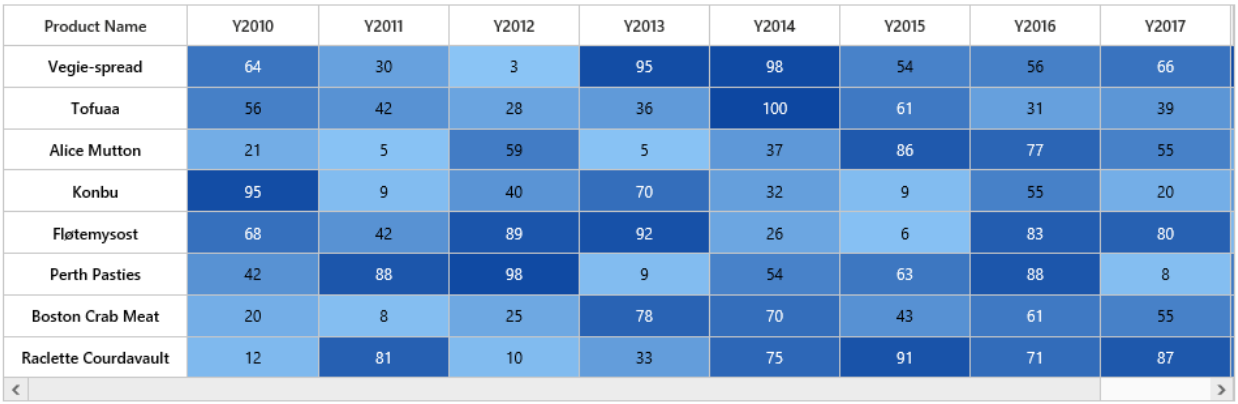
Next we can configure color range for these values using color mapping

- Configure items mapping based on items source.

C#

```
public ActionResult Default()
{
HeatMapProperties Heatmap = new HeatMapProperties();
List<HeatMapColorMapping> colorCollection = new List<HeatMapColorMapping>();
colorCollection.Add(new HeatMapColorMapping() { Color = "#8ec8f8", Label =
new HeatMapLabel() { Text = "0" }, Value = 0 });
colorCollection.Add(new HeatMapColorMapping() { Color = "#0d47a1", Label =
new HeatMapLabel() { Text = "100" }, Value = 100 });
Heatmap.ColorMappingCollection = colorCollection;
ViewData["HeatMapModel"] = Heatmap;
return View();
}
```

- This will show the grid data with color based on the range given.



Product Name	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017
Vegie-spread	64	30	3	95	98	54	56	66
Tofuaa	56	42	28	36	100	61	31	39
Alice Mutton	21	5	59	5	37	86	77	55
Konbu	95	9	40	70	32	9	55	20
Fløtemysost	68	42	89	92	26	6	83	80
Perth Pasties	42	88	98	9	54	63	88	8
Boston Crab Meat	20	8	25	78	70	43	61	55
Raclette Courdavault	12	81	10	33	75	91	71	87

Legend

A legend control is used to represent range value in a gradient, create a legend with the same color mapping as shown below.

HTML

```
<div>
@Html.EJ().HeatMapLegend("heatmapLegend", ViewData["HeatMapLegendModel"] as
Syncfusion.JavaScript.DataVisualization.Models.HeatMapLegendProperties)
</div>
```

- Final code files looks like this.

C#

```
public ActionResult Default()
{
HeatMapProperties Heatmap = new HeatMapProperties();
List<HeatMapColorMapping> colorCollection = new List<HeatMapColorMapping>();
colorCollection.Add(new HeatMapColorMapping() { Color = "#8ec8f8", Label =
new HeatMapLabel() { Text = "0" }, Value = 0 });
colorCollection.Add(new HeatMapColorMapping() { Color = "#0d47a1", Label =
new HeatMapLabel() { Text = "100" }, Value = 100 });
Heatmap.ColorMappingCollection = colorCollection;
```

Copyright © 2001 - 2020 Syncfusion Inc.

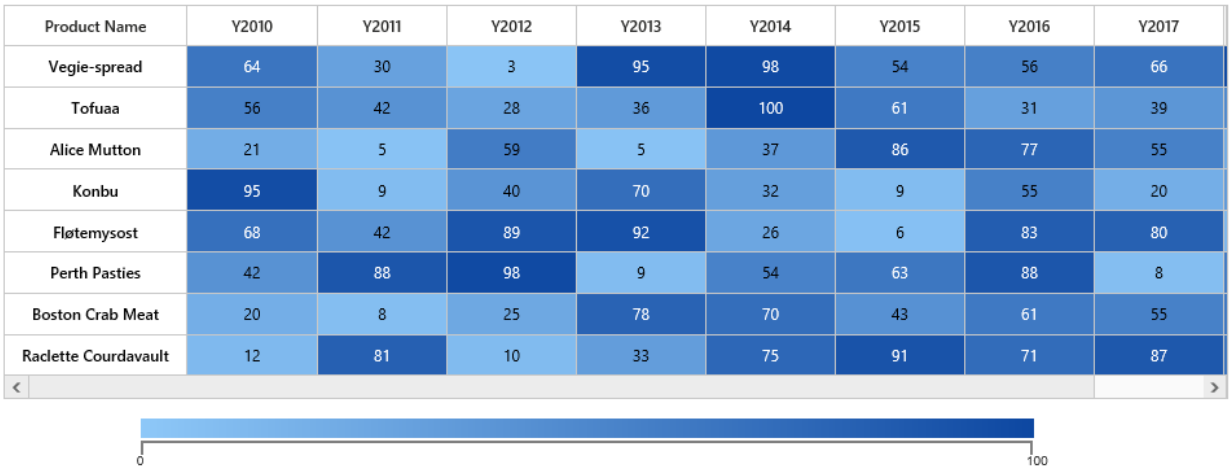
```
Heatmap.LegendCollection.Add("heatmapLegend");
Heatmap.Width = "830";
CellMapping CellMapping = new CellMapping();
CellMapping.Column = new PropertyMapping() { PropertyName = "ProductName",
DisplayName = "Product Name" };
CellMapping.Row = new PropertyMapping() { PropertyName = "Year", DisplayName
= "Year" };
CellMapping.Value = new PropertyMapping() { PropertyName = "Value" };
Collection columnMapping = new Collection();
columnMapping.Add(new HeaderMapping() { PropertyName = "Veggie-spread",
DisplayName = "Veggie-spread" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Tofu", DisplayName =
"Tofu" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Alice Mutton",
DisplayName = "Alice Mutton" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Donut", DisplayName
= "Donut" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Burger", DisplayName
= "Burger" });
columnMapping.Add(new HeaderMapping() { PropertyName = "Perth Pasties",
DisplayName = "Perth Pasties" });
CellMapping.ColumnMapping = columnMapping;
HeaderMapping headerMapping = new HeaderMapping() { PropertyName = "Year",
DisplayName = "Year", ColumnStyle = new ColumnStyle() { Width = 105,
TextAlign = HeatMapTextAlign.Right } };
CellMapping.HeaderMapping = headerMapping;
Heatmap.ItemsSource = GetCellSource();
Heatmap.ItemsMapping = CellMapping;
Heatmap.IsResponsive = true;
ViewData["HeatMapModel"] = Heatmap;
HeatMapLegendProperties legend = new HeatMapLegendProperties();
legend.ColorMappingCollection = colorCollection;
legend.Orientation = HeatMapLegendOrientation.Horizontal;
legend.LegendMode =
Syncfusion.JavaScript.DataVisualization.HeatMapEnums.HeatMapLegendMode.Gradi
ent;
legend.Height = "50px";
legend.Width = "75%";
legend.IsResponsive = true;
ViewData["HeatMapLegendModel"] = legend;
return View();
}
public Collection GetCellSource()
{
Collection collection = new Collection();
string[] name = { "Veggie-spread", "Tofu", "Alice Mutton", "Donut",
"Burger", "Perth Pasties" };
Random random = new Random();
foreach (string item in name)
{
for (int i = 0; i < 6; i++)
{
double value = random.Next(0, random.Next(0, 100));
collection.Add(new SampleCellData() { ProductName = item, Year = "Y" + (2011
+ i), Value = value });
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return collection;
}
```

HTML

```
<div>
@Html.EJ().HeatMap("heatmap", ViewData["HeatMapModel"] as
Syncfusion.JavaScript.DataVisualization.Models.HeatMapProperties)
</div>
<div style="height: 15px; width: 100%;"></div>
<div>
@Html.EJ().HeatMapLegend("heatmapLegend", ViewData["HeatMapLegendModel"] as
Syncfusion.JavaScript.DataVisualization.Models.HeatMapLegendProperties)
</div>
```



Items Mapping

External data source can be mapped with HeatMap using ItemsMapping property. It supports 2 kind of data source.

- In TableMapping rows represents an objects in collection, columns represents numerical properties of that object.
- In CellMapping each cell represent an object in collection, this collection is grouped based on specific property to form as rows and columns.

Let us see the difference between two types of mapping. Following table represents two different data structure to represent the same HeatMap.

CellMapping	TableMapping
{% highlight c# %}public ActionResult Default(){ HeatMapProperties Heatmap = new HeatMapProperties(); CellMapping CellMapping = new CellMapping(); CellMapping.Column = new PropertyMapping() { PropertyName = "ProductName", DisplayName = "Product Name" };	{% highlight c# %}public ActionResult TableDataBinding(){ HeatMapProperties Heatmap = new HeatMapProperties(); TableMapping TableMapping = new TableMapping(); TableMapping.HeaderMapping = new HeaderMapping() { PropertyName =

Copyright © 2001 - 2020 Syncfusion Inc.

CellMapping.Row = new PropertyMapping() { PropertyName = "Year", DisplayName = "Year" }; CellMapping.Value = new PropertyMapping() { PropertyName = "Value" }; Collection columnMapping = new Collection(); columnMapping.Add(new HeaderMapping() { PropertyName = "Veggie-spread", DisplayName = "Veggie-spread" }); columnMapping.Add(new HeaderMapping() { PropertyName = "Tofu", DisplayName = "Tofu" }); columnMapping.Add(new HeaderMapping() { PropertyName = "Alice Mutton", DisplayName = "Alice Mutton" }); columnMapping.Add(new HeaderMapping() { PropertyName = "Donut", DisplayName = "Donut" }); columnMapping.Add(new HeaderMapping() { PropertyName = "Burger", DisplayName = "Burger" }); columnMapping.Add(new HeaderMapping() { PropertyName = "Perth Pasties", DisplayName = "Perth Pasties" }); CellMapping.ColumnMapping = columnMapping; HeaderMapping headerMapping = new HeaderMapping() { PropertyName = "Year", DisplayName = "Year", ColumnStyle = new ColumnStyle() { Width = 105, TextAlign = HeatMapTextAlign.Right } }; CellMapping.HeaderMapping = headerMapping; Heatmap.ItemsSource = GetCellSource(); Heatmap.ItemsMapping = CellMapping; ViewData["HeatMapModel"] = Heatmap; return View();}public Collection GetCellSource(){ Collection collection = new Collection(); string[] name = { "Veggie-spread", "Tofu", "Alice Mutton", "Donut", "Burger", "Perth Pasties" }; Random random = new Random(); foreach (string item in name) { for (int i = 0; i < 6; i++) { double value = random.Next(0, random.Next(0, 100)); collection.Add(new SampleCellData() { ProductName = item, Year = "Y" + (2011 + i), Value = value }); } } return collection;}{% endhighlight %}	"ProductName", DisplayName = "Product Name", ColumnStyle = new ColumnStyle() { Width = 140, TextAlign = HeatMapTextAlign.Right } }; TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2010", DisplayName = "Y2010" }); TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2011", DisplayName = "Y2011" }); TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2012", DisplayName = "Y2012" }); TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2013", DisplayName = "Y2013" }); TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2014", DisplayName = "Y2014" }); TableMapping.ColumnMapping.Add(new HeaderMapping() { PropertyName = "Y2015", DisplayName = "Y2015" }); Heatmap.ItemsMapping = TableMapping; Heatmap.ItemsSource = GetTableSource(); ViewData["HeatMapModel"] = Heatmap; return View(); }public Collection GetTableSource(){ Collection collection = new Collection(); Random random = new Random(); string[] rows = { "Veggie-spread", "Tofu", "Alice Mutton", "Donut", "Burger", "Perth Pasties", "Boston Crab Meat", "Racket Burger" }; for (int i = 0; i < 8; i++) { collection.Add(new SampleTableData() { ProductName = rows[i], Y2010 = random.Next(0, 100), Y2011 = random.Next(0, 100), Y2012 = random.Next(0, 100), Y2013 = random.Next(0, 100), Y2014 = random.Next(0, 100), Y2015 = random.Next(0, 100) }); } return collection;}{% endhighlight %}
Here, a single ProductInfo object represent a value for a particular product in a particular year	Here, a single ProductInfo object represents value for a particular product from year 2010 to 2015.

Copyright © 2001 - 2020 Syncfusion Inc.

ProductName	2010	2011
Vegie-spread	64	56
Tofuaa	26	93
Alice Mutton	47	67
Konbu	67	36

ProductName	2010	2011
Vegie-spread	64	56
Tofuaa	26	93
Alice Mutton	47	67
Konbu	67	36

Color Mapping

Color mapping is used to indicate values as colors instead of numerical values. For example, if a HeatMap represents a data from 0 to 100. ColorMapping is used to specify a color for lower value and higher value. For any value between two values, a medium color will be automatically be chosen.

In color mapping, when white color is set to value 0 and red color is set for value 30, as shown below.

C#

```
public ActionResult Default()
{
HeatMapProperties Heatmap = new HeatMapProperties();
List<HeatMapColorMapping> colorCollection = new List<HeatMapColorMapping>();
colorCollection.Add(new HeatMapColorMapping() { Color = "#8ec8f8", Label =
new HeatMapLabel() { Text = "0" }, Value = 0 });
colorCollection.Add(new HeatMapColorMapping() { Color = "#0d47a1", Label =
new HeatMapLabel() { Text = "100" }, Value = 100 });
Heatmap.ColorMappingCollection = colorCollection;
ViewData["HeatMapModel"] = Heatmap;
return View();
}
```

Resultant HeatMap will be as shown below.

Product Name	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016	Y2017
Vegie-spread	84	65	55	80	13	100	41	96
Tofuaa	84	83	94	29	82	2	9	20
Alice Mutton	24	39	49	16	53	74	34	52
Konbu	57	1	95	47	39	39	98	52
Fløtemysost	13	40	57	71	27	22	42	19
Perth Pasties	30	70	38	78	5	27	45	63
Boston Crab Meat	56	52	60	7	74	68	45	100
Raclette Courdavault	90	2	58	90	80	94	30	77

Legend

Legend is a control used to summarize the range of colors in HeatMap. This gives visual guideline for mapping between value and color.

Copyright © 2001 - 2020 Syncfusion Inc.

Create Legend

Legend can be created with color mapping as shown below.

C#

```
public ActionResult TableDataBinding()
{
List<HeatMapColorMapping> colorCollection = new List<HeatMapColorMapping>();
colorCollection.Add(new HeatMapColorMapping() { Color = "#fe0002", Label =
new HeatMapLabel() { Text = "Poor" }, Value = 0 });
colorCollection.Add(new HeatMapColorMapping() { Color = "#ffff01", Label =
new HeatMapLabel() { Text = "Average" }, Value = 3 });
colorCollection.Add(new HeatMapColorMapping() { Color = "#13ab11", Label =
new HeatMapLabel() { Text = "Good" }, Value = 6 });
colorCollection.Add(new HeatMapColorMapping() { Color = "#005ba2", Label =
new HeatMapLabel() { Text = "Excellent" }, Value = 10 });
Heatmap.ColorMappingCollection = colorCollection;
HeatMapLegendProperties legend = new HeatMapLegendProperties();
legend.ColorMappingCollection = colorCollection;
legend.Orientation = HeatMapLegendOrientation.Horizontal;
legend.LegendMode = HeatMapLegendMode.Gradient;
legend.Height = "50px";
legend.Width = "75%";
ViewData["HeatMapLegendModel"] = legend;
return View();
}
```

Resultant legend will be like following image.



Legend Mode

There are two modes for Legend

- Gradient
- List

Gradient:

C#

```
public ActionResult TableDataBinding()
{
HeatMapLegendProperties legend = new HeatMapLegendProperties();
legend.Orientation = HeatMapLegendOrientation.Horizontal;
legend.LegendMode = HeatMapLegendMode.Gradient;
ViewData["HeatMapLegendModel"] = legend;
return View();
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

List:

C#

```
public ActionResult TableDataBinding()
{
HeatMapLegendProperties legend = new HeatMapLegendProperties();
legend.Orientation = HeatMapLegendOrientation.Horizontal;
legend.LegendMode = HeatMapLegendMode.List;
ViewData["HeatMapLegendModel"] = legend;
return View();
}
```



Orientation

There are 2 types of Orientation, applicable for Gradient and List Mode

- Horizontal
- Vertical

Horizontal:

C#

```
public ActionResult TableDataBinding()
{
HeatMapLegendProperties legend = new HeatMapLegendProperties();
legend.Orientation = HeatMapLegendOrientation.Horizontal;
legend.LegendMode = HeatMapLegendMode.List;
ViewData["HeatMapLegendModel"] = legend;
return View();
}
```



Vertical:

C#

```
public ActionResult TableDataBinding()
{
HeatMapLegendProperties legend = new HeatMapLegendProperties();
legend.Orientation = HeatMapLegendOrientation.Vertical;
legend.LegendMode = HeatMapLegendMode.List;
ViewData["HeatMapLegendModel"] = legend;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Kanban Board

Overview

The Kanban control for ASP.NET MVC is an efficient way to visualize the workflow at each stage along its path to completion. The most important features available are Swim lane, filtering, and editing.

Some important features of the Kanban control are:

** DataSource - Bind the Kanban control with an array of JSON objects or ej.DataManager which support OData and remote web service binding. * Swim lane – Supports Swim lane grouping. Filters – Supports filtering based on user Query. Editing - Offers card editing, inserting, and deleting using Dialog. Card template - Offers to render the card based on user template. RTL support - Provides a full-fledged right-to-left mode which aligns content in the Kanban control from right to left. * Localization - Provides inherent support to localize the UI.*

Getting Started

This section explains briefly how to create a Kanban in your application with ASP.NET MVC and also explains about how to enable basic Kanban features like Swim lane, Filtering.

Create your first Kanban in MVC

The following steps explain creating first Kanban with an empty columns.

1. Create Syncfusion ASP.NET MVC application. You can refer MVC Getting Started documentation to create new project and add necessary dll's and script files. 2. Add a Kanban control in View file with an empty columns as like as follows. In Columns definition, the HeaderText property allows you to set text to column headers.

HTML

```
@(Html.EJ().Kanban("Kanban")
.Columns(col =>
{
col.HeaderText("Backlog").Add();
col.HeaderText("In Progress").Add();
col.HeaderText("Done").Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Data Binding

You can bind the data to Kanban control by either locally or remotely. Assign the local data to DataSource property of Kanban control to bind local data.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Add();
col.HeaderText("In Progress").Add();
col.HeaderText("Done").Add();
})
)
```

C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
Task.Add(new Tasks(6, "Close", "Arrange a web meeting with the customer to
get the login page requirements.", "Others", "Low", "Meeting", 2, "Janet",
"../content/images/kanban/6.png", 1));
Task.Add(new Tasks(7, "Validate", "Validate new requirements",
"Improvement", "Low", "Validation", 1.5, "Janet",
"../content/images/kanban/7.png", 4));
Task.Add(new Tasks(8, "Close", "Login page validation", "Story", "Release
Breaker", "Validation,Fix", 2.5, "Andrew Fuller",
"../content/images/kanban/8.png", 2));
Task.Add(new Tasks(9, "Testing", "Fix the issues reported in Safari
browser.", "Bug", "Release Breaker", "Fix,Safari", 1.5, "Janet",
"../content/images/kanban/1.png", 2));
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Task.Add(new Tasks(10, "Close", "Test the application in the IE browser.",
"Story", "Low", "Testing,IE", 5.5, "Andrew Fuller",
"../content/images/kanban/4.png", 3));
ViewBag.datasource = Task;
return View();
}
}
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public string Image { get; set; }
public int RankId { get; set; }
}
```



Note: ODataAdaptor is the default adaptor used within DataManager. While binding to other web services, proper data adaptor needs to be set for adaptor option of DataManager.

Mapping Values

In order to display cards in Kanban control, you need to map the database fields to Kanban cards and columns. The required mapping field are listed as follows

- KeyField - Map the column name to use as Key values to columns.
- Columns - Map the corresponding Key values of KeyField column to each columns
- Content - Map the column name to use as content to cards in the Fields.
- PrimaryKey - Map the column name to use as Primary Key in Fields.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Color("Type")
.Content("Summary")
.PrimaryKey("Id");
})
)
```

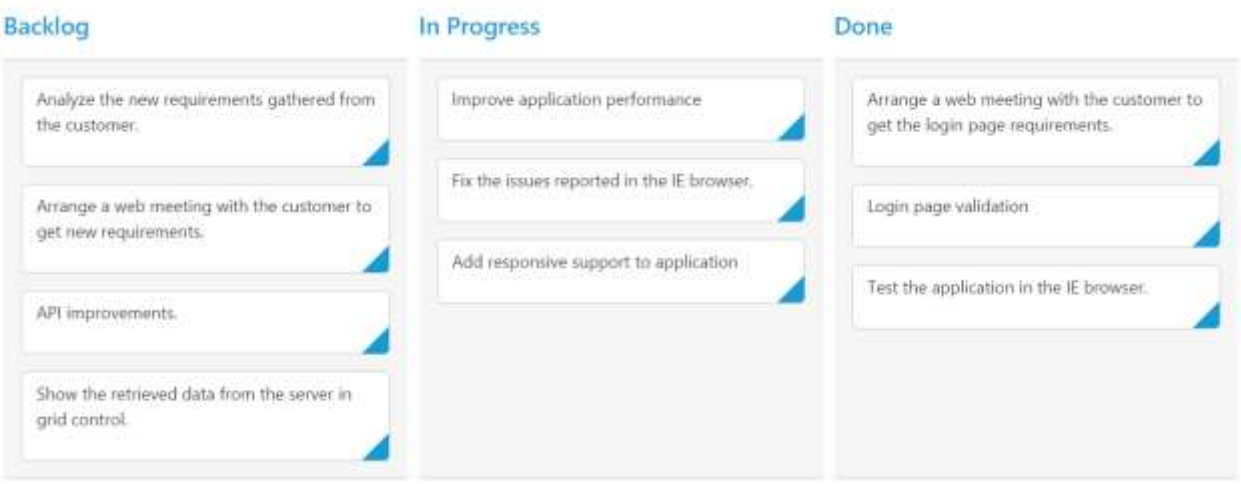
C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
Task.Add(new Tasks(6, "Close", "Arrange a web meeting with the customer to
get the login page requirements.", "Others", "Low", "Meeting", 2, "Janet",
"../content/images/kanban/6.png", 1));
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Task.Add(new Tasks(7, "Validate", "Validate new requirements",
"Improvement", "Low", "Validation", 1.5, "Janet",
"../content/images/kanban/7.png", 4));
Task.Add(new Tasks(8, "Close", "Login page validation", "Story", "Release
Breaker", "Validation,Fix", 2.5, "Andrew Fuller",
"../content/images/kanban/8.png", 2));
Task.Add(new Tasks(9, "Testing", "Fix the issues reported in Safari
browser.", "Bug", "Release Breaker", "Fix,Safari", 1.5, "Janet",
"../content/images/kanban/1.png", 2));
Task.Add(new Tasks(10, "Close", "Test the application in the IE browser.",
"Story", "Low", "Testing,IE", 5.5, "Andrew Fuller",
"../content/images/kanban/4.png", 3));
ViewBag.datasource = Task;
return View();
}
}
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public string Image { get; set; }
public int RankId { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Note: PrimaryKey field is mandatory for "Drag and Drop","Selection" and "Editing" Features.

Enable Swimlane

Swimlane can be enabled by mapping the SwimlaneKey to appropriate column name in DataSource. This enables the grouping of the cards based on the mapped column values.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Color("Type")
.Content("Summary")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
)
```

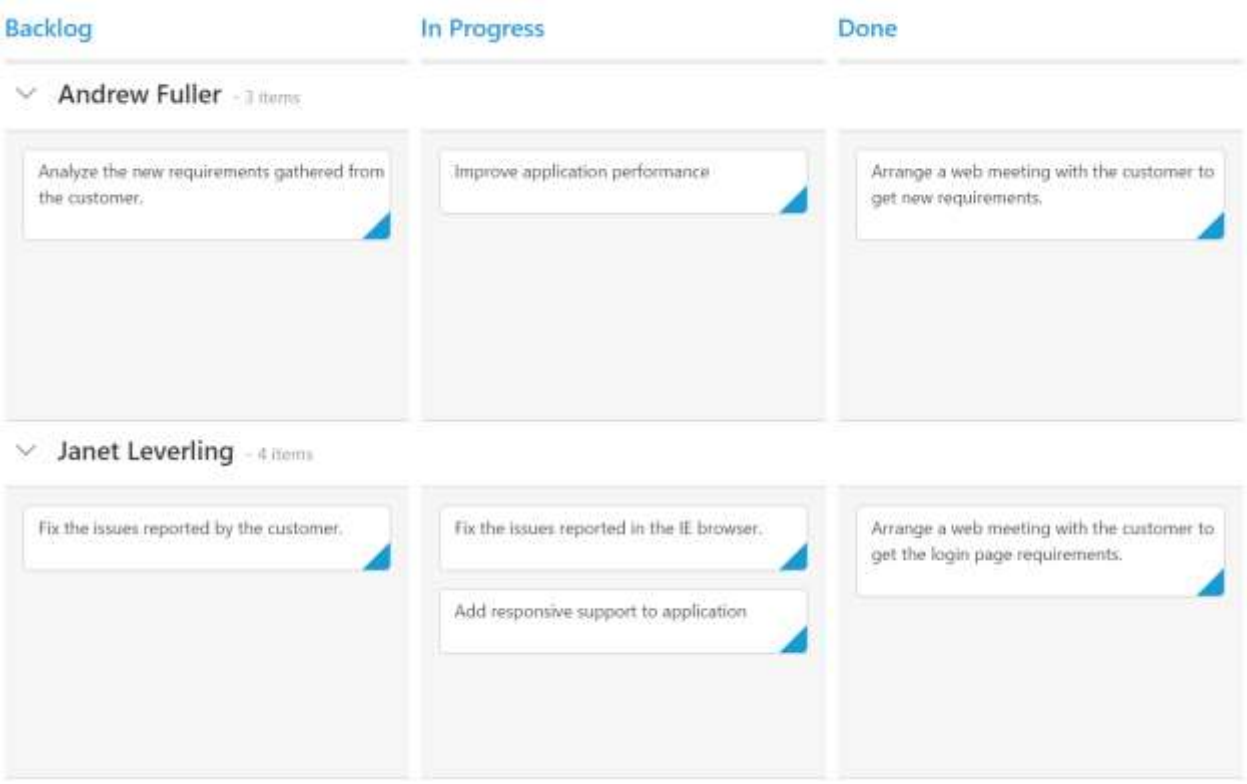
C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
```

```
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
Task.Add(new Tasks(6, "Close", "Arrange a web meeting with the customer to
get the login page requirements.", "Others", "Low", "Meeting", 2, "Janet",
"../content/images/kanban/6.png", 1));
Task.Add(new Tasks(7, "Validate", "Validate new requirements",
"Improvement", "Low", "Validation", 1.5, "Janet",
"../content/images/kanban/7.png", 4));
Task.Add(new Tasks(8, "Close", "Login page validation", "Story", "Release
Breaker", "Validation,Fix", 2.5, "Andrew Fuller",
"../content/images/kanban/8.png", 2));
Task.Add(new Tasks(9, "Testing", "Fix the issues reported in Safari
browser.", "Bug", "Release Breaker", "Fix,Safari", 1.5, "Janet",
"../content/images/kanban/1.png", 2));
Task.Add(new Tasks(10, "Close", "Test the application in the IE browser.",
"Story", "Low", "Testing,IE", 5.5, "Andrew Fuller",
"../content/images/kanban/4.png", 3));
ViewBag.datasource = Task;
return View();
}
}
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public double Estimate { get; set; }
public string Assignee { get; set; }
public string ImgUrl { get; set; }
public int RankId { get; set; }
}
```



Adding Filters

Filters allows to filter the collection of cards from DataSource which meets the predefined query in the filters collection. To enable filtering, define FilterSettings collection with display Text and ej.Query.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Color("Type")
.Content("Summary")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
.FilterSettings(filter =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

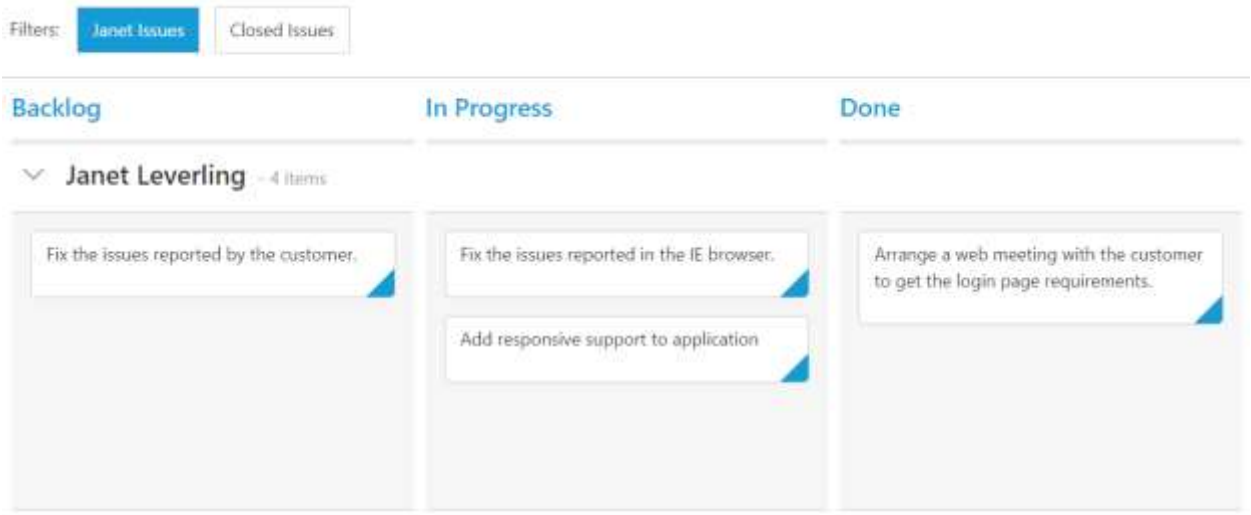
```
{
filter.Text("Janet Issues").Query("new ej.Query().where('Assignee', 'equal',
'Janet')").Description("Displays issues which matches the assignee as
'Janet").Add();
filter.Text("Closed Issues").Query("new ej.Query().where('Status', 'equal',
'Close')").Description("Display the issues of 'Testing'").Add();
})
)
```

C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
Task.Add(new Tasks(6, "Close", "Arrange a web meeting with the customer to
get the login page requirements.", "Others", "Low", "Meeting", 2, "Janet",
"../content/images/kanban/6.png", 1));
Task.Add(new Tasks(7, "Validate", "Validate new requirements",
"Improvement", "Low", "Validation", 1.5, "Janet",
"../content/images/kanban/7.png", 4));
Task.Add(new Tasks(8, "Close", "Login page validation", "Story", "Release
Breaker", "Validation,Fix", 2.5, "Andrew Fuller",
"../content/images/kanban/8.png", 2));
Task.Add(new Tasks(9, "Testing", "Fix the issues reported in Safari
browser.", "Bug", "Release Breaker", "Fix,Safari", 1.5, "Janet",
"../content/images/kanban/1.png", 2));
Task.Add(new Tasks(10, "Close", "Test the application in the IE browser.",
"Story", "Low", "Testing,IE", 5.5, "Andrew Fuller",
"../content/images/kanban/4.png", 3));
ViewBag.datasource = Task;
return View();
}
}
public class Tasks
{
public Tasks()
{
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public string ImgUrl { get; set; }
public int RankId { get; set; }
}
```



Columns

Column fields are present in the dataSource schema and it is rendering cards based its mapping column values.

Key Mapping

To render Kanban with simple cards, you need to map the dataSource fields to Kanban cards and Columns. The required mapping field are listed as follows

Mapping Fields	Description

KeyField	Map the column name to use as Key values to columns.
Columns.Key	Map the corresponding key values of KeyField column to each columns.
Columns.HeaderText	It represents the title for particular column
Fields.Content	Map the column name to use as content to cards.

Note: 1. If the column with KeyField is not in the dataSource and Key values specified will not available in column values, then the cards will not be rendered.

2. If the Fields.Content is not in the dataSource, then empty cards will be rendered.

The following code example describes the above behavior.

RAZOR

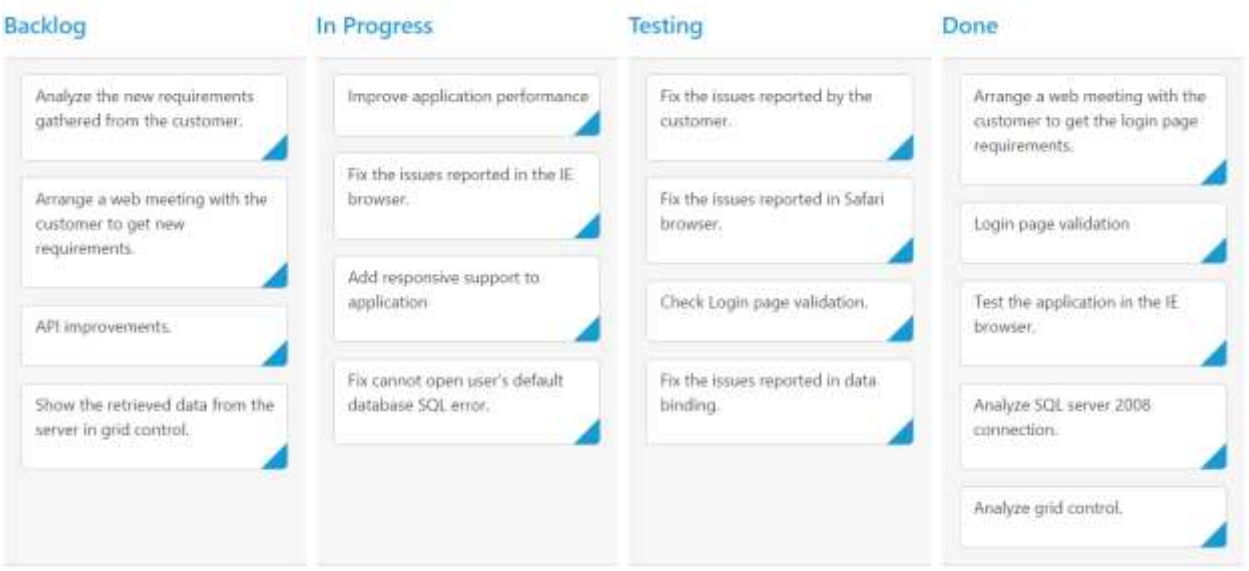
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Key Mapping

You can map more than one datasource fields as Key values to show different key cards into single column. For e.g , you can map "Validate,In progress" keys under "In progress" column.

The following code example and screenshot which describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress or Validate").Key("InProgress,Validate").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

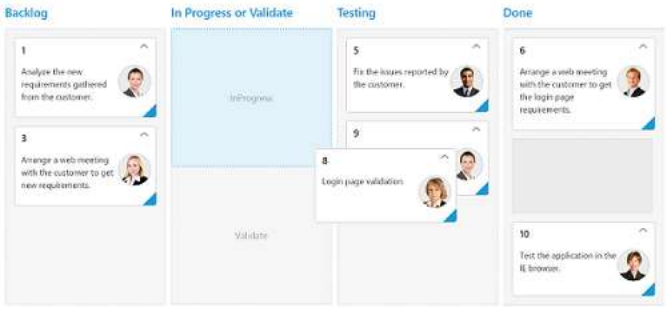
C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

The following output is displayed as a result of the above code example.



Headers

Header Template

The template design that applies on for the column header. To render template, set HeaderTemplate property of the Columns.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer the link.

The following code example describes the above behavior.

HTML

```
<!-- Column Template -->
<script id="column1" type="text/x-jsrender">
<span class="e-backlog e-icon"></span> Backlog
</script>
<div id="column4">
<span class="e-done e-icon"></span> Done
</div>
<div id='Kanban'></div>
```

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").HeaderTemplate("#column1").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").HeaderTemplate("#column4").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

C#

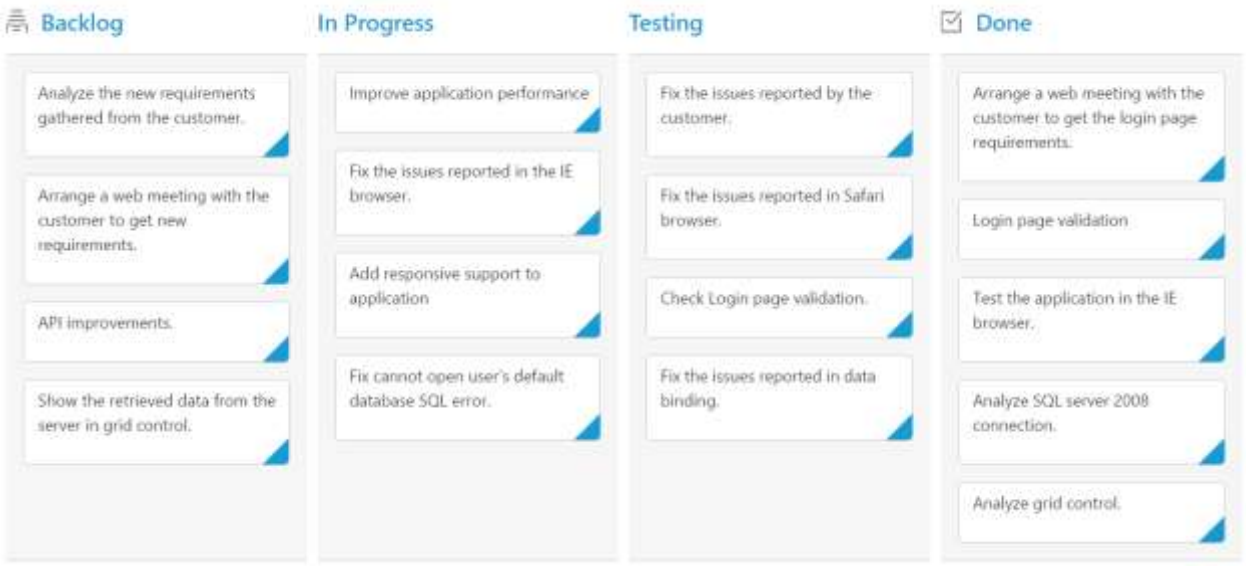
Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

CSS

```
/*CSS for Header template icon*/
.e-backlog,.e-done {
font-size: 16px;
padding-right: 5px;
display: inline-block;
}
.e-backlog:before {
content: "\e807";
}
.e-done:before {
content: "\e80a";
}
```

The following output is displayed as a result of the above code example.



Width

You can specify the width for particular column by setting Width property of Columns as in pixel (ex: 100) or in percentage (ex: 40%).

The following code example describes the above behavior.

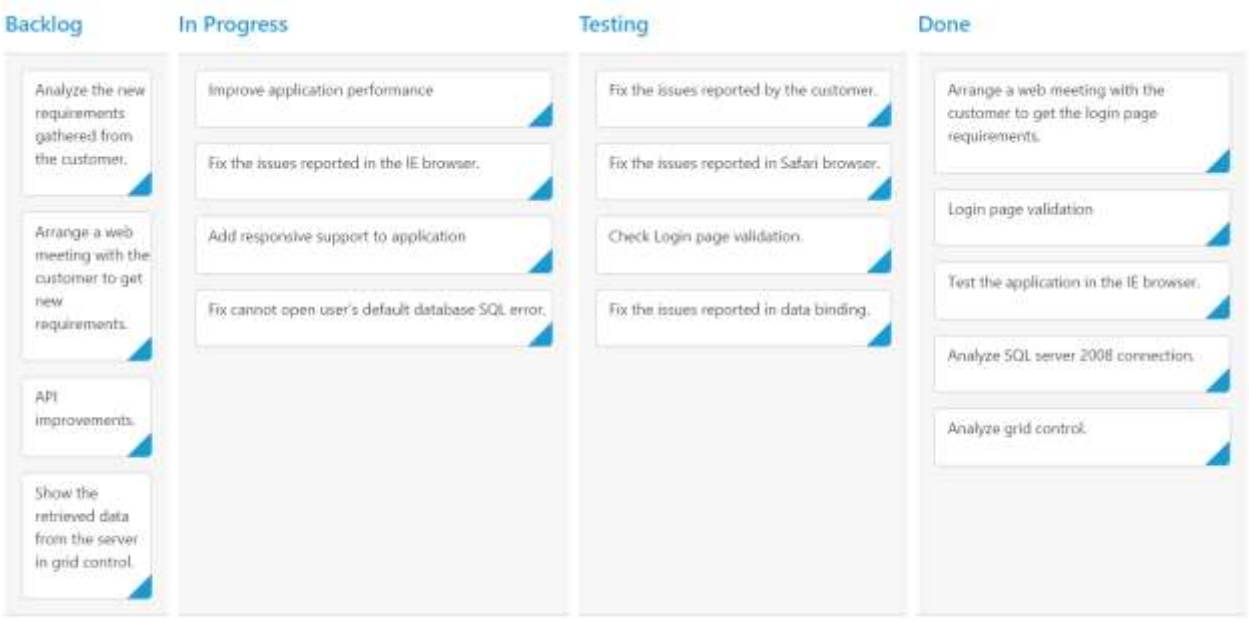
RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Width(70).Add();
col.HeaderText("In Progress").Key("InProgress").Width("40%").Add();
col.HeaderText("Testing").Key("Testing").Width(90).Add();
col.HeaderText("Done").Key("Close").Width(90).Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Copyright © 2001 - 2020 Syncfusion Inc.

Visibility

You can hide particular column in Kanban by setting Visible property of it as false.

The following code example describes the above behavior.

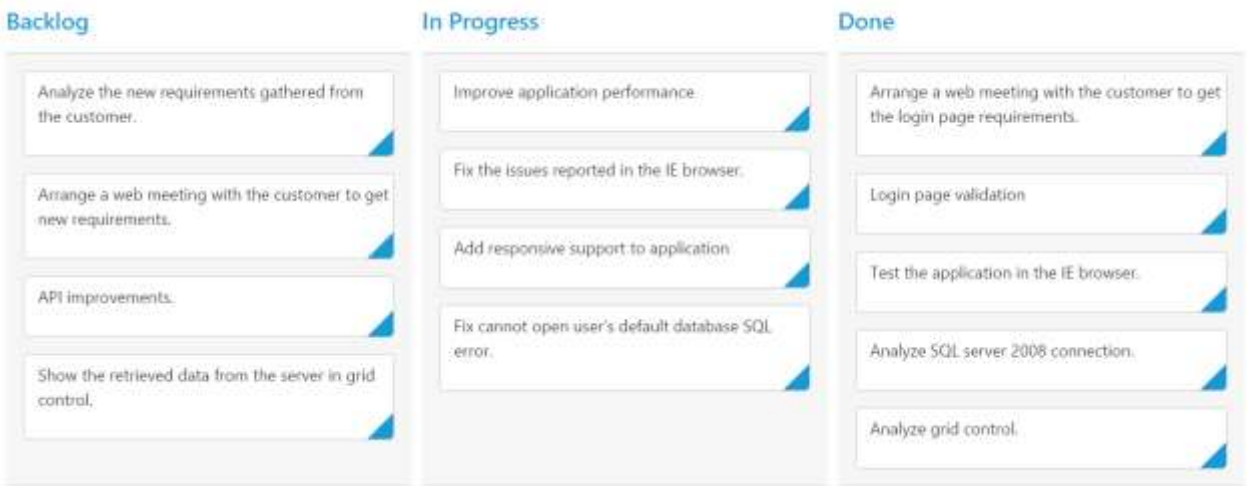
RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Visible(false).Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Copyright © 2001 - 2020 Syncfusion Inc.

Toggle

You can set particular column collapsed state in Kanban by setting IsCollapsed property of it as true. You need to set AllowToggleColumn as true to use Expand/Collapse Column.

The following code example describes the above behavior.

RAZOR

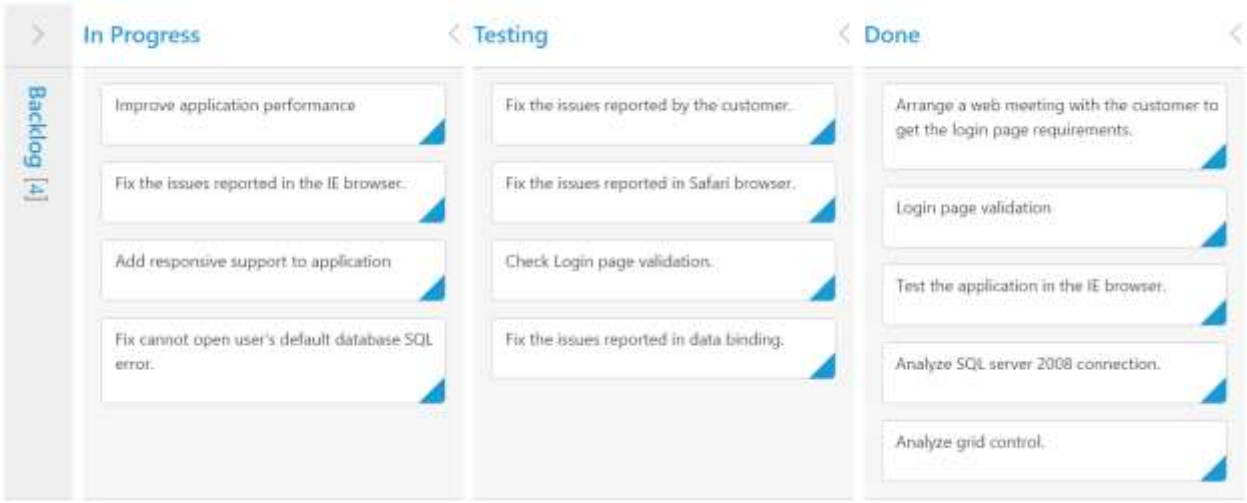
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowToggleColumn(true)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").IsCollapsed(true).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
field.PrimaryKey("Id");
})
```

C#

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowToggleColumn(true)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").IsCollapsed(true).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
field.PrimaryKey("Id");
})
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Allow Dragging

You can enable and disable drag behavior to the cards in the Kanban columns using the AllowDrag property and the default value is true.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").AllowDrag(false).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Priority("RankId")
.PrimaryKey("Id");
})
)
```

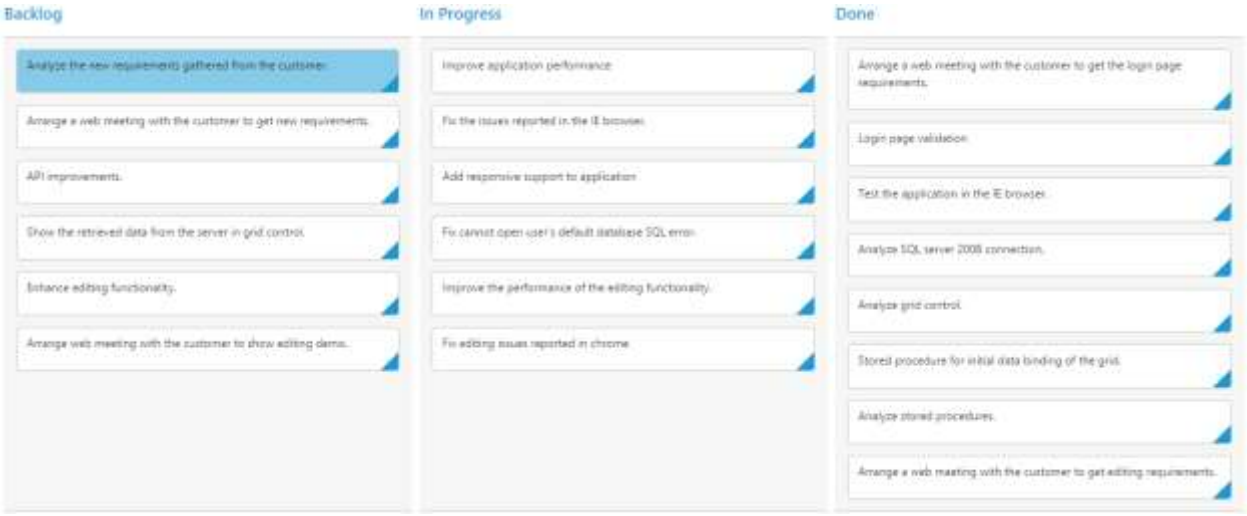
C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
```

The following output is displayed as a result of the above code example.



Allow Dropping

You can enable and disable drop behavior to the cards in the Kanban columns using the AllowDrop property and the default value is true.

The following code example describes the above behavior.

RAZOR

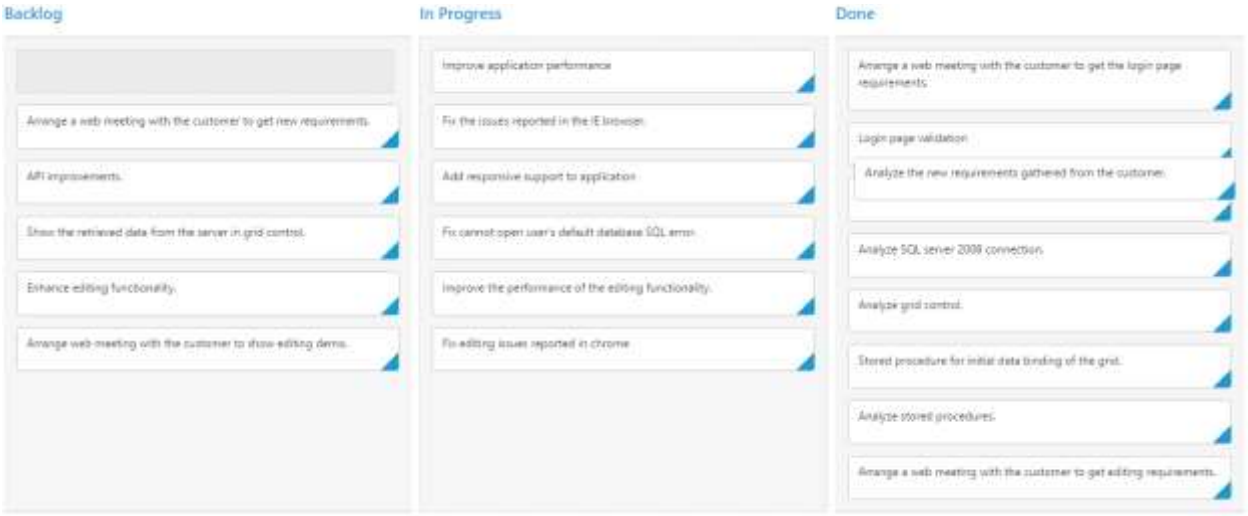
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").AllowDrop(false).Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Priority("RankId")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
```

```
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Items Count

You can show total cards count in each column's header using the property EnableTotalCount and the default value is false.

The following code example describes the above behavior.

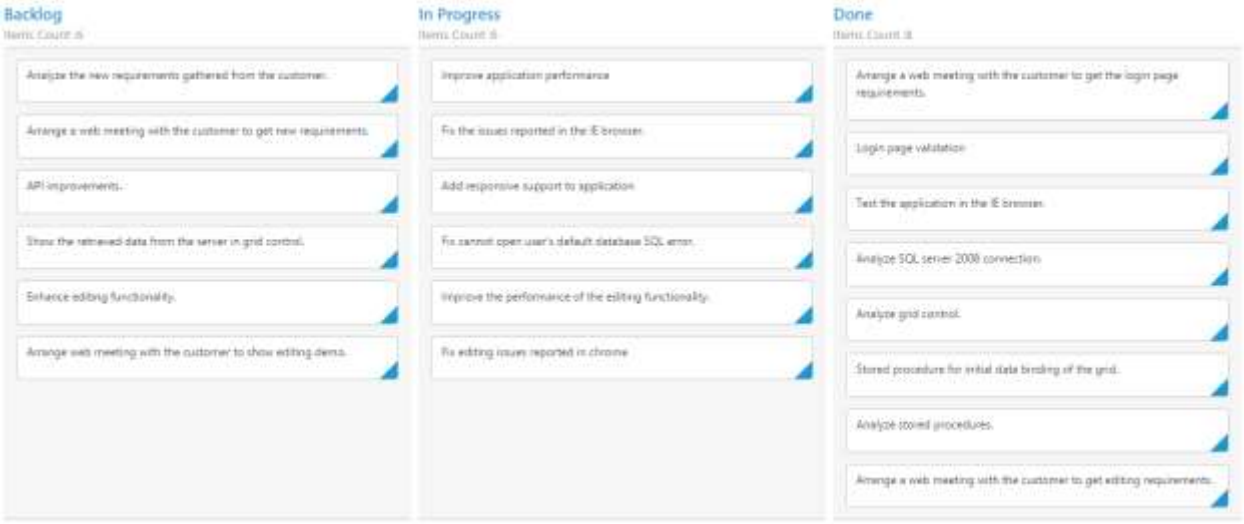
RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.EnableTotalCount(true)
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Customize Items Count Text

You can customize the Items count text using the property TotalCount.Text.

The following code example describes the above behavior.

RAZOR

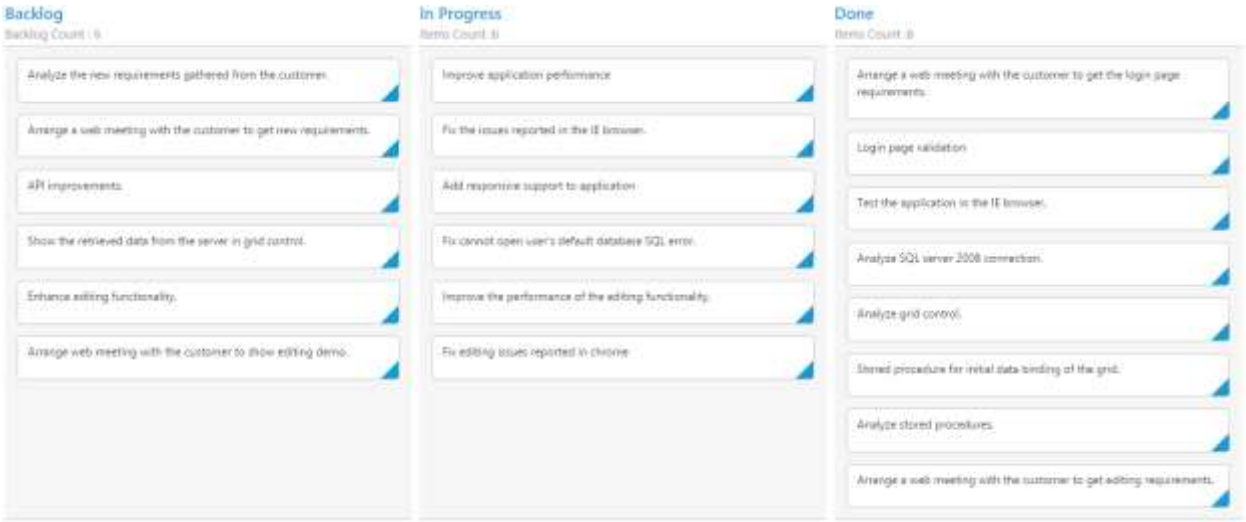
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").TotalCount(
t => {
t.Text("Backlog Count");
}
).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.EnableTotalCount(true)
.Fields(field =>
```

```
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Display Empty Column

On binding empty data source to Kanban control, it displays "No cards to display" text. Using ShowColumnWhenEmpty property, we can render empty kanban columns. Set AllowEditing and AllowAdding property as true to add new data into the empty kanban column.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ShowColumnWhenEmpty(true)
.Columns(col =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

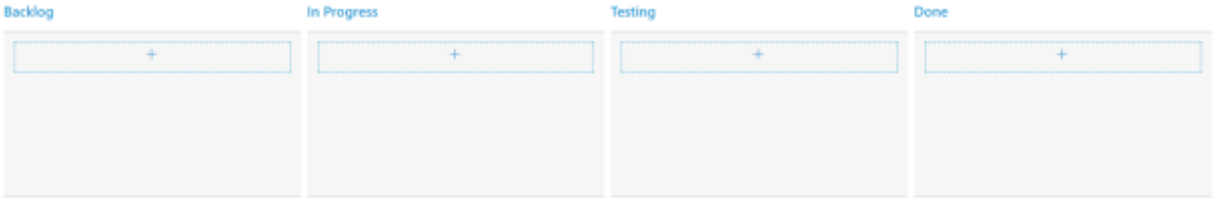
```
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").ValidationRules(rule => { rule.AddRule("required",
true).AddRule("number", true); }).Add();
e.Field("Status").Add();
e.Field("Assignee").Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties() { DecimalPlaces = 2 }).ValidationRules(rule => {
rule.AddRule("range", "[0,1000]"); }).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).ValidationRules(rule
=> { rule.AddRule("required", true); }).Add();
}).EditMode(KanbanEditMode.Dialog);
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(0).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Data Binding

Kanban DataSource property allows to bind datasource as the instance of one of the following types.

- Collection that implements IEnumerable or IEnumerable.
- REST Service URL as string.
- ORM components such as Entity Framework/Linq to SQL.

We can also bind the above type of data source by using lambda Expressions of Kanban DataSource Property.

In the following section, let us see on how to bind various data sources to Kanban using DataSource API.

IEnumerable

The Kanban can be bound with collection that implements IEnumerable interface. It can be assigned to Kanban DataSource property.

Note: When using DataSource builder, the IEnumerable data source can be passed as parameter either directly to the DataSource builder or to the JSON property of the DataSource builder.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<Tasks>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
List<Tasks> Task = new List<Tasks>();
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Nancy"));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Andrew Fuller"));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.","Janet"));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Janet"));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the
customer.","Steven walker"));
ViewBag.datasource = Task;
return View();
}
[Serializable]
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Assignee)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Assignee = Assignee;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Assignee { get; set; }
}
}
}
```

The following output is displayed as a result of the above code example.



Entity Framework

The collection of business objects generated from Entity Framework ORM can be used as Kanban data source. The Kanban can be used with Entity Framework using either code first technique or Database first technique.

The following code example describes the above behavior.

RAZOR

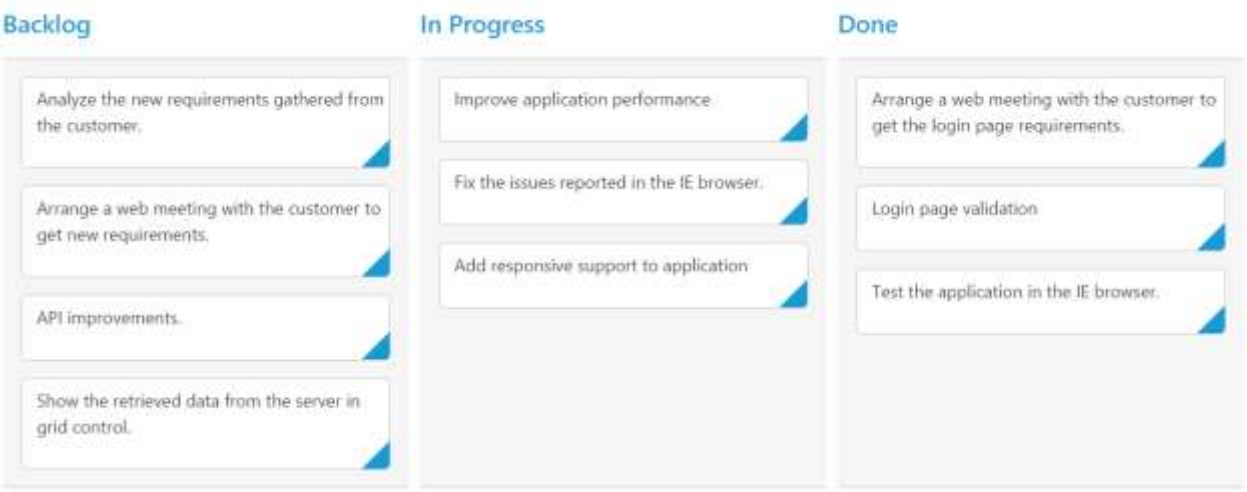
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



LINQ to SQL

The LINQ to SQL can be used as the data source of the Kanban. And the Object Relation class which is used to map with Relational table.

The following code example describes the above behavior.

RAZOR

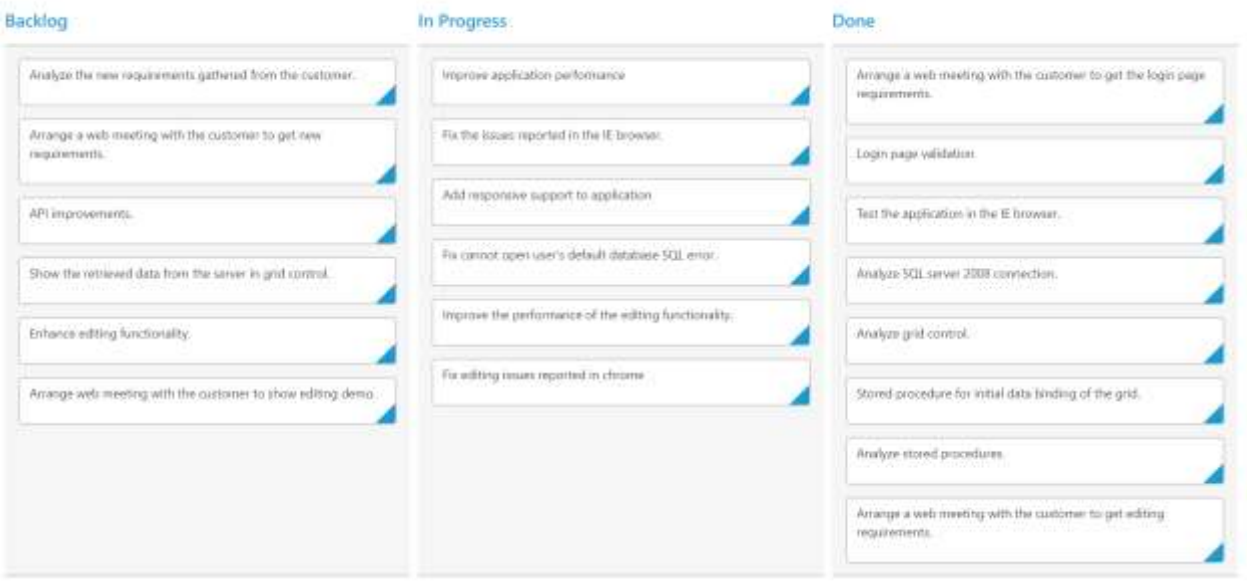
```
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



WCF DataService / OData Service

WCF Data Services is a component of the .NET Framework that enables you to create services that use the Open Data Protocol (OData) to expose and consume data over the Web or intranet by using the semantics of representational state transfer (REST).

To consume WCF DataService in Kanban control, provide the service link directly to the Kanban DataSource property.

We have an online OData Service http://mvc.syncfusion.com/Services/Northwnd.svc/Tasks created specifically for Syncfusion Controls

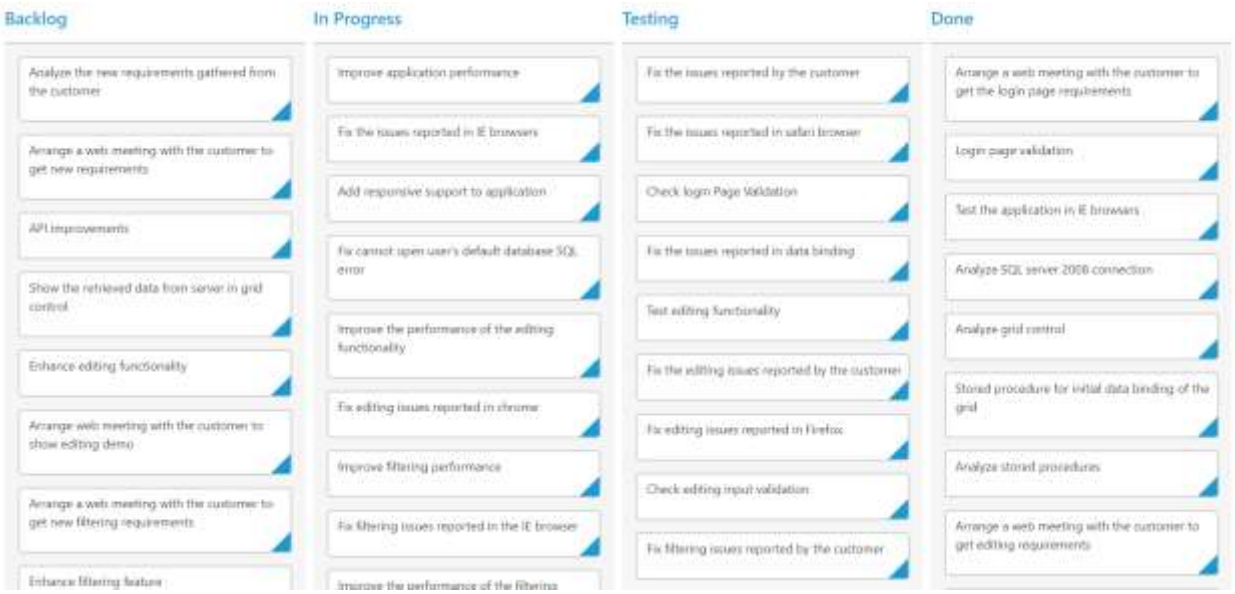
The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource("http://mvc.syncfusion.com/Services/Northwnd.svc/Tasks")
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



ODataV4 Service

ODataV4Adaptor is used for consuming data from OData V4 Service. To consume OData V4 service, set the service link to the URL property of Kanban DataSource and set AdaptorType.ODataV4Adaptor to the Adaptor Property of Kanban DataSource.

Waring: ODL supports to parse the $search query option, however, Web API OData doesn't support it so far.

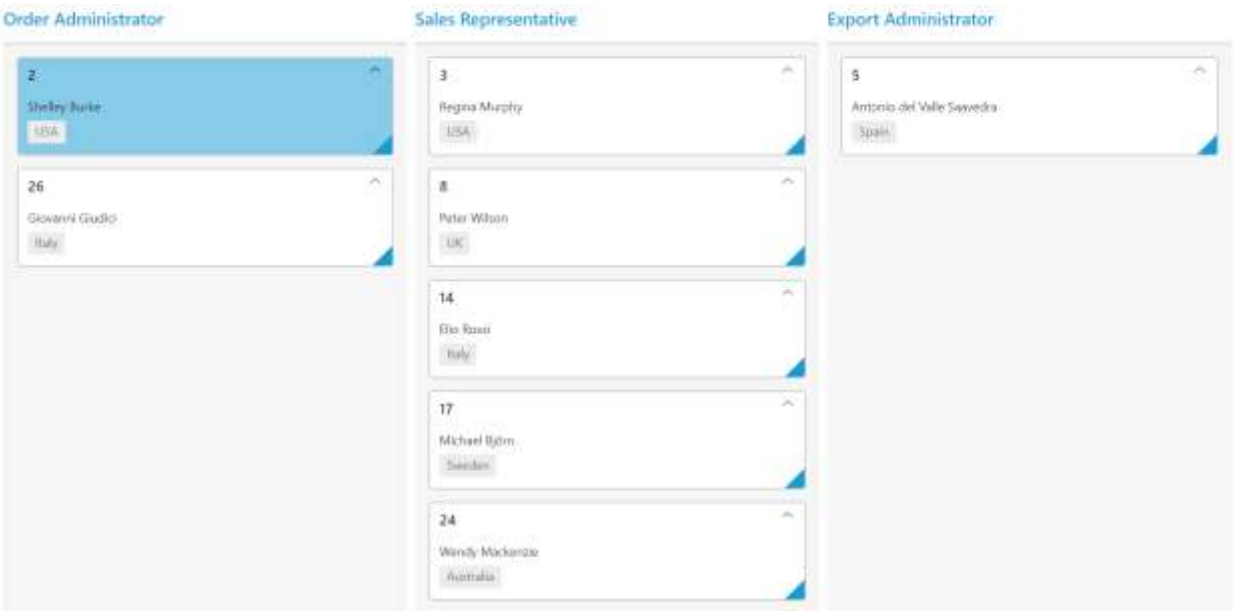
The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource(data =>
data.URL("http://services.odata.org/v4/northwind/northwind.svc/Suppliers").A
daptor(AdaptorType.ODataV4Adaptor))
.Columns(col =>
{
col.HeaderText("Order Administrator ").Key("Order Administrator").Add();
col.HeaderText("Sales Representative").Key("Sales Representative").Add();
col.HeaderText("Export Administrator").Key("Export Administrator").Add();
})
.KeyField("ContactTitle")
.AllowTitle(true)
.Fields(field =>
{
field.Content("ContactName")
.Tag("Country")
.PrimaryKey("SupplierID");
})
)
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



WebAPI

Using ej.WebApiAdaptor, you can bind WebApi service data to Kanban. The data from WebApi service must be returned as object that has property Items with its value as dataSource and another property Count with its value as dataSource's total cards count.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource(data =>
data.URL("/api/Orders").Adaptor(AdaptorType.WebApiAdaptor))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Backlog ").Add();
col.HeaderText("In Progress").Key("InProgress ").Add();
col.HeaderText("Close").Key("Done ").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public class OrdersController : ApiController
{
NorthwindDataContext db = new NorthwindDataContext();
public object Get()
{
```

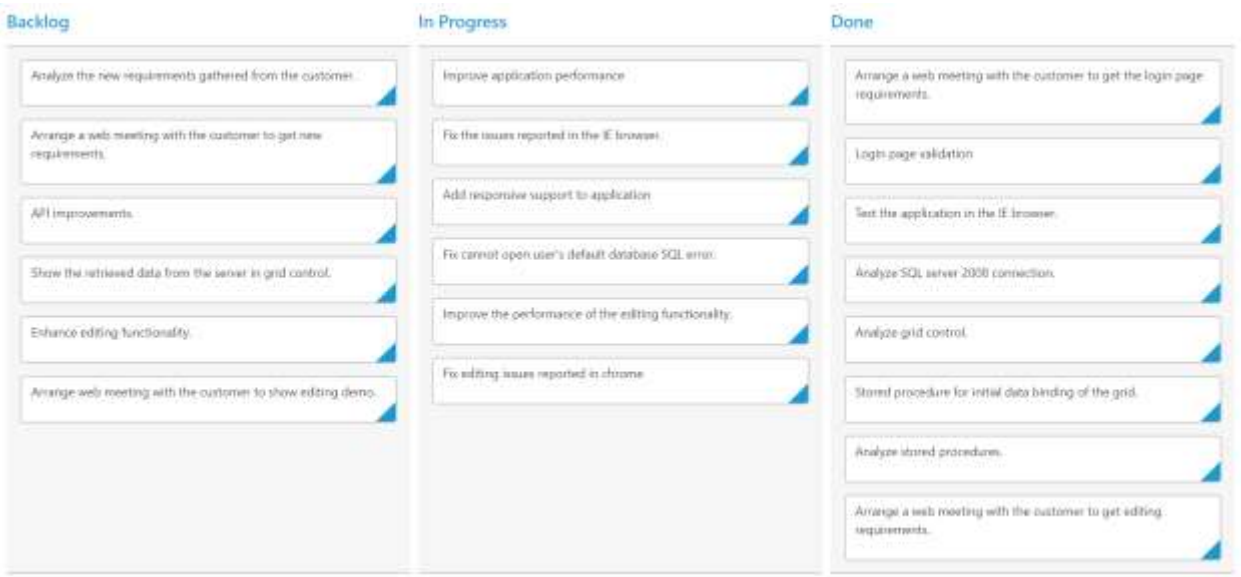
Copyright © 2001 - 2020 Syncfusion Inc.

```
var data = db.Tasks.Take(10).ToList();
return new
{
Items = data,
Count = data.Count()
};
}
}
}
```

Please refer the following link for common WebAPI creation.

http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api

The following output is displayed as a result of the above code example.



URL Adaptor

You can use the UrlAdaptor of DataManager when binding DataSource from remote data. At initial load of Kanban, using URL property of DataManager, data are fetched from remote data and bound to Kanban. You can map CRUD operation in Kanban to Server-Side Controller action using the property CrudURL.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource(data =>
data.URL("GetData").CrudURL("Crud").Adaptor(AdaptorType.UrlAdaptor))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").ShowAddButton(true).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.CustomToolbarItems(custom =>
{
custom.Template("#Delete").Add();
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").Add();
e.Field("Assignee").Add();
e.Field("Estimate").Add();
e.Field("Summary").Add();
}).EditMode(KanbanEditMode.Dialog);
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id")
.Priority("RankId");
})
.ClientSideEvents(eve => eve.ToolBarClick("toolbarClick"))
)
<script type="text/javascript">
function toolbarClick(args) {
if (args.itemName == "Delete" && this.element.find(".e-
kanbancard").hasClass("e-cardselection")) {
var selected = this.element.find(".e-cardselection");
this.KanbanEdit.deleteCard(selected.attr("id"));
}
}
</script>
<script id="Delete" type="text/template">
<a class="e-customdelete  e-icon" />
</script>
<style type="text/css" class="cssStyles">
.e-customdelete:before {
content: "\e800";
line-height: 26px;
min-height: 26px;
min-width: 14px;
display: inline-block;
}
</style>
```

Also when you use UrlAdaptor, you need to return the data as JSON and the JSON object must contain a properties result & count. The result holds the dataSource as its value and count holds the total cards count as its value.

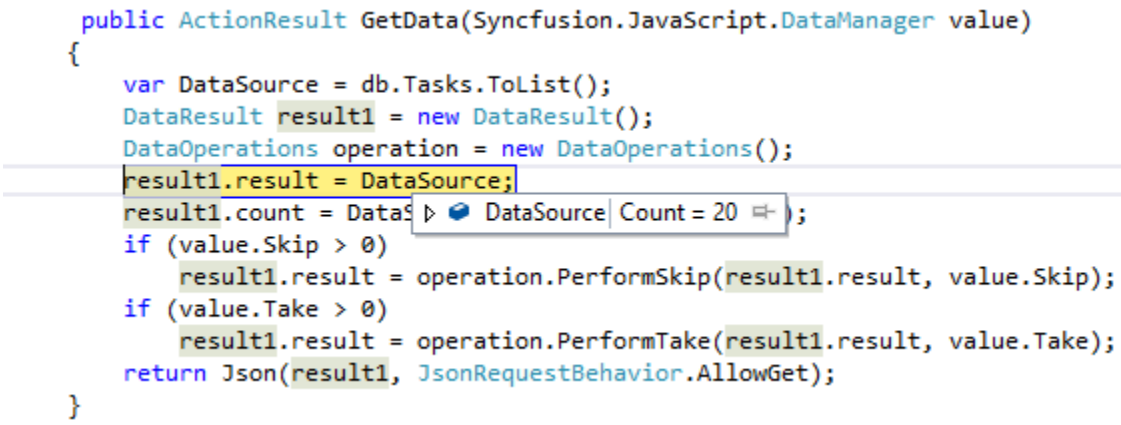
The following code example describes the above behavior.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult GetData(Syncfusion.JavaScript.DataManager value)
{
var DataSource = db.Tasks.ToList();
DataResult result1 = new DataResult();
DataOperations operation = new DataOperations();
result1.result = DataSource;
result1.count = DataSource.AsQueryable().Count();
if (value.Skip > 0)
result1.result = operation.PerformSkip(result1.result, value.Skip);
if (value.Take > 0)
result1.result = operation.PerformTake(result1.result, value.Take);
return Json(result1, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable result { get; set; }
public int count { get; set; }
}
```

Please refer to the below screenshot.



Using DataOperations helper class you can perform Kanban action at server side. The in-built methods that we have provided in the DataOperations class are listed below.

1. PerformTake 2. PerformSelect 3. Execute

For more information about the Server-Side operation you can refer the below link.

Persisting Data in Server

Workflows

Workflows can be defined to set the flow of card moving between the Kanban column statuses and it is applicable to drag and drop and context menu features.

You can set Workflows as array of Objects which consists of Key and AllowedTransitions properties. The AllowedTransitions accepts more than one transition of the specific column key mentioned in Key property.

If a card is to be dragged to not allowed transition columns , then not supported warning symbol will be displayed for denoting the error.

The following code example describes the above Workflow functionality.

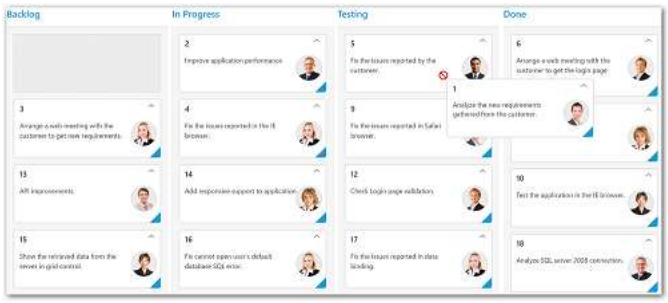
HTML

```
<div id='Kanban'></div>
```

JAVASCRIPT

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Workflows(workflow =>
{
workflow.Key("Open").AllowedTransitions("InProgress").Add();
workflow.Key("InProgress").AllowedTransitions("Testing,Close").Add();
})
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
})
)
```

The following output is displayed as a result of the above code example.



Swim lanes

Swim lanes are a horizontal categorization of issues in the Kanban control which brings transparency to the workflow. This can be enabled by mapping the SwimlaneKey to appropriate column name in the DataSource.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

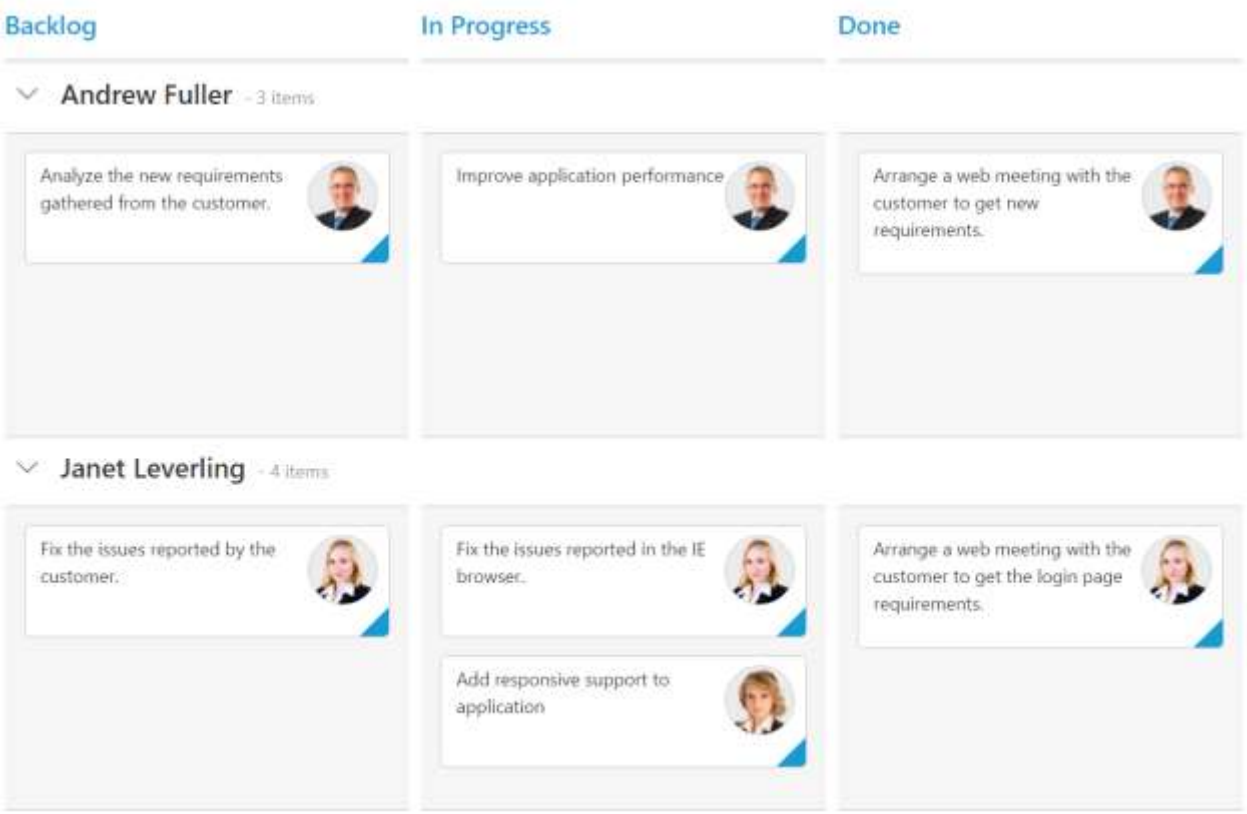
```
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.SwimlaneKey("Assignee")
.ImageUrl("Image")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Customized swimlane header text

You can change the swimlane row header text using the swimlane Headers property. In this property, the text is changed using the Text property and the corresponding value is mapped into the Key property based on SwimlaneKey datasource mapping.

Refer to the following code example.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.AllowTitle(true)
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
.SwimlaneSettings(swim =>
{
swim.Headers(header => {
header.Text("Andrew").Key("Andrew Fuller").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
header.Text("Janet").Key("Janet").Add();
});
})
)
```

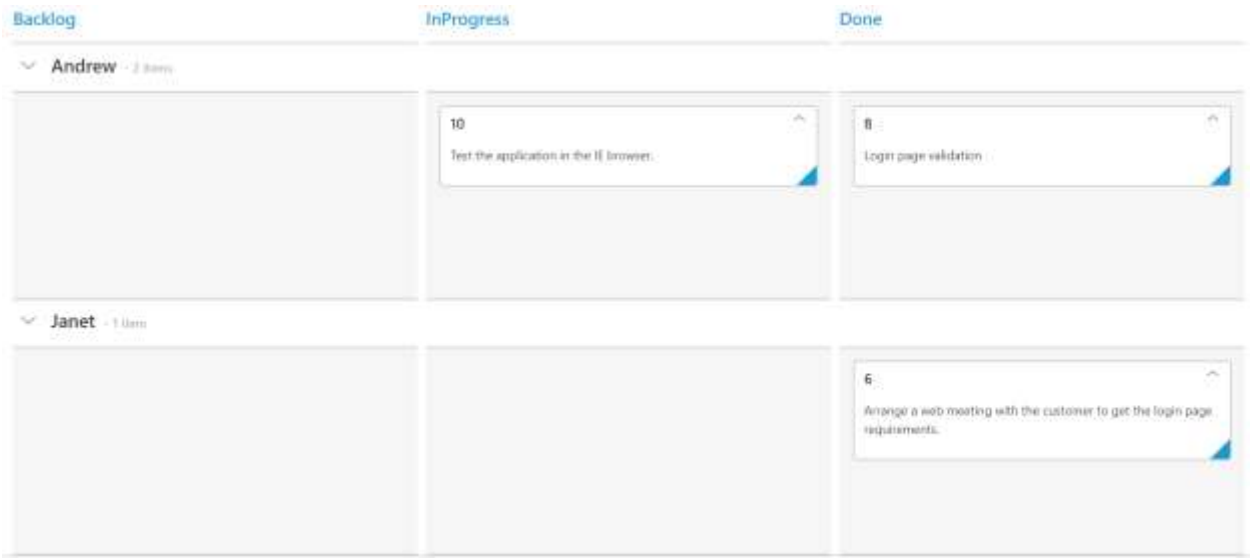
C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(6, "Close", "Arrange a web meeting with the customer to
get the login page requirements.", "Others", "Low", "Meeting", 2, "Janet",
"../content/images/kanban/3.png", 1));
Task.Add(new Tasks(7, "Validate", "Validate new requirements",
"Improvement", "Low", "Validation", 1.5, "Janet",
"../content/images/kanban/3.png", 4));
Task.Add(new Tasks(8, "Close", "Login page validation", "Story", "Release
Breaker", "Validation,Fix", 2.5, "Andrew Fuller",
"../content/images/kanban/1.png", 2));
Task.Add(new Tasks(9, "Testing", "Fix the issues reported in Safari
browser.", "Bug", "Release Breaker", "Fix,Safari", 1.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(10, "InProgress", "Test the application in the IE
browser.", "Story", "Low", "Testing,IE", 5.5, "Andrew Fuller",
"../content/images/kanban/1.png", 3));
ViewBag.datasource = Task;
return View();
}
}
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public string Image { get; set; }
public int RankId { get; set; }
}
```

The following output is displayed as a result of the above code example.



Empty swimlane row on Kanban board

You can create an empty swimlane row by enabling the ShowEmptySwimlane property based on swimlane headers Key value mapping. If no data is present, then the empty swimlane row is rendered on the Kanban board based on the specified swimlane headers Key.

Refer to the following code.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.AllowTitle(true)
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
.SwimlaneSettings(swim =>
{
```

```
swim.ShowEmptySwimlane(true)
.Headers(header => {
header.Text("Andrew").Key("Andrew Fuller").Add();
header.Text("Janet").Key("Janet Leverling").Add();
});
})
)
```

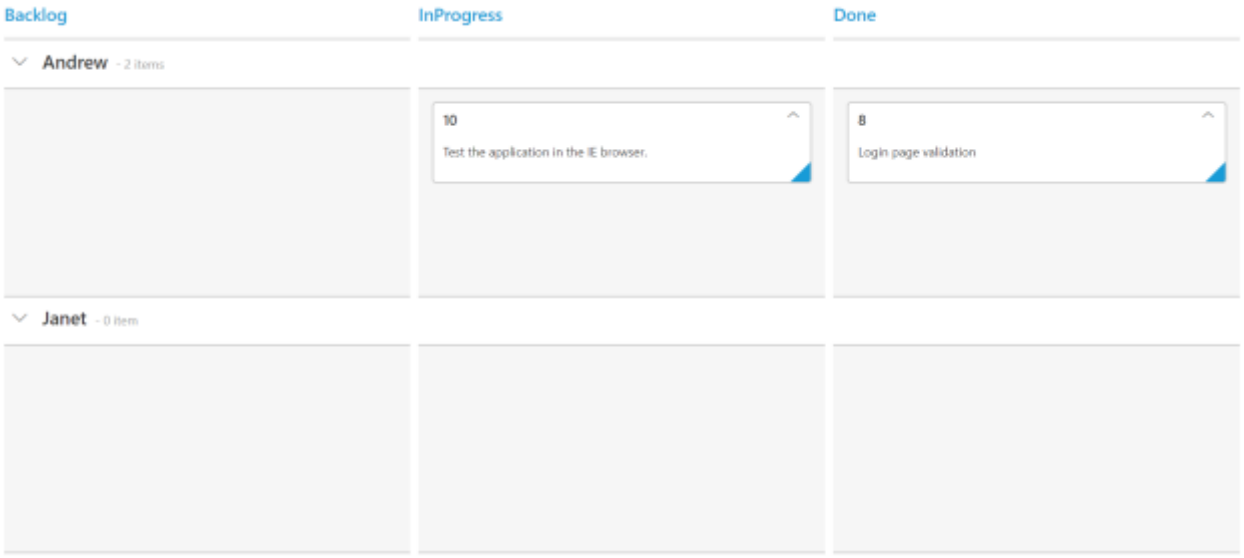
C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(8, "Close", "Login page validation", "Story", "Release
Breaker", "Validation,Fix", 2.5, "Andrew Fuller",
"../content/images/kanban/1.png", 2));
Task.Add(new Tasks(9, "Testing", "Fix the issues reported in Safari
browser.", "Bug", "Release Breaker", "Fix,Safari", 1.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(10, "InProgress", "Test the application in the IE
browser.", "Story", "Low", "Testing,IE", 5.5, "Andrew Fuller",
"../content/images/kanban/1.png", 3));
ViewBag.datasource = Task;
return View();
}
}
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public string Assignee { get; set; }
public string Image { get; set; }
public int RankId { get; set; }
}
```

The following output is displayed as a result of the above code example.



Drag And Drop between swim lanes

You can set 'AllowDragAndDrop' property of 'SwimlaneSettings' as true to enable Drag and Drop between the swim lanes.

If a card is to be dragged in the same swim lane, only a droppable target cell is added to the dotted line border.If a card is dragged from one swim lane to another, all the Kanban cells will be added to the dotted line borders, except the dragged card cell.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.SwimlaneSettings(swimSettings =>
{
swimSettings.AllowDragAndDrop(true);
})
.Fields(field =>
{
field.Content("Summary")
.SwimlaneKey("Assignee")
.ImageUrl("Image")
```

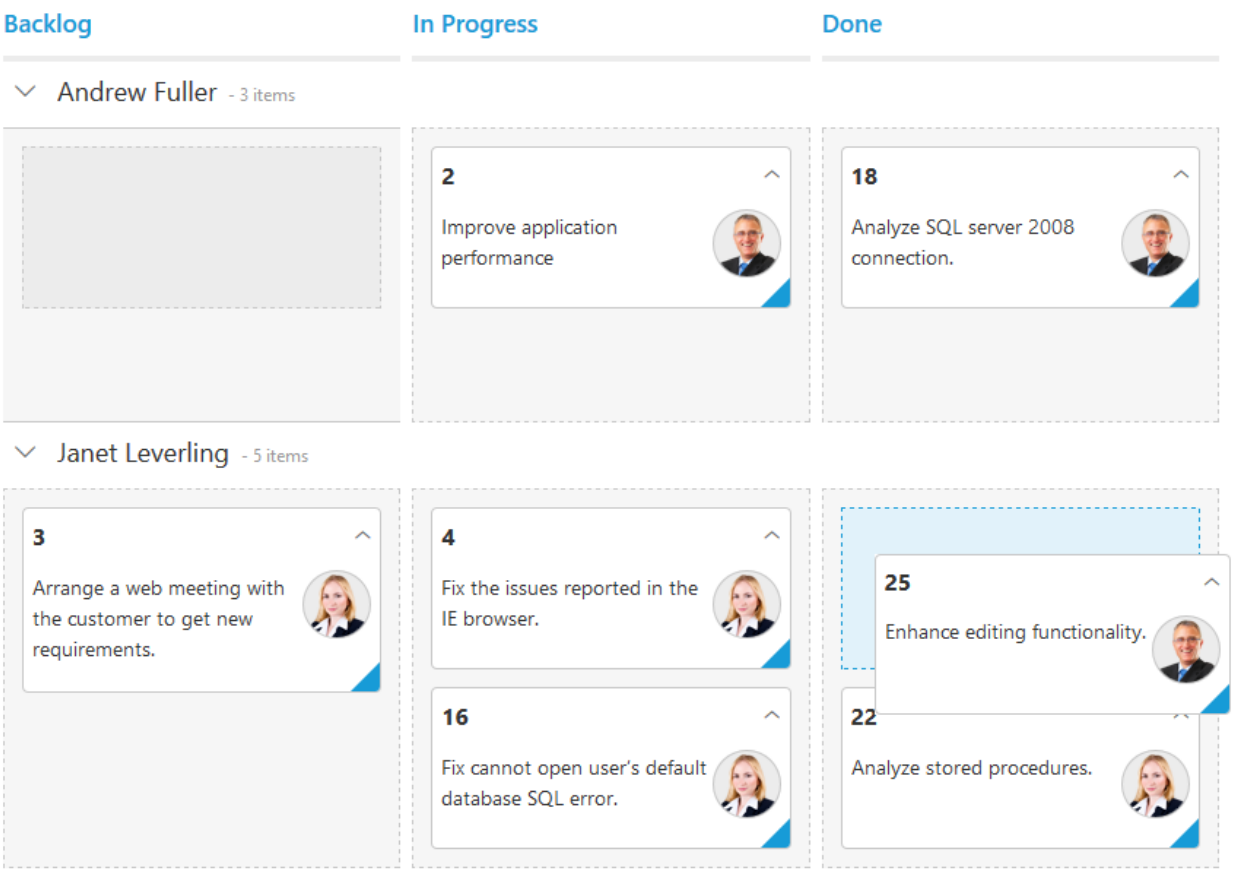
Copyright © 2001 - 2020 Syncfusion Inc.

```
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Unassigned swim lane group

Unassigned swim lane feature provides option to group some common swim lane key values as separate swim lane group. You can enable and disable this behavior using the property Enable.

User can use default common key values or user defined key values.

Using default values Using user defined values

Note: By default, given common keys are grouped under the swim lane name Unassigned, user can customize the name using localization.

Using default values

By default, the swim lane keys of card which is having null, undefined, empty string ("") values will be grouped as unassigned category when Enable property is set as true.

Default values in the Keys collection are null, undefined, empty string ("").

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.SwimlaneKey("Assignee")
.ImageUrl("Image")
.PrimaryKey("Id");
})
)
```

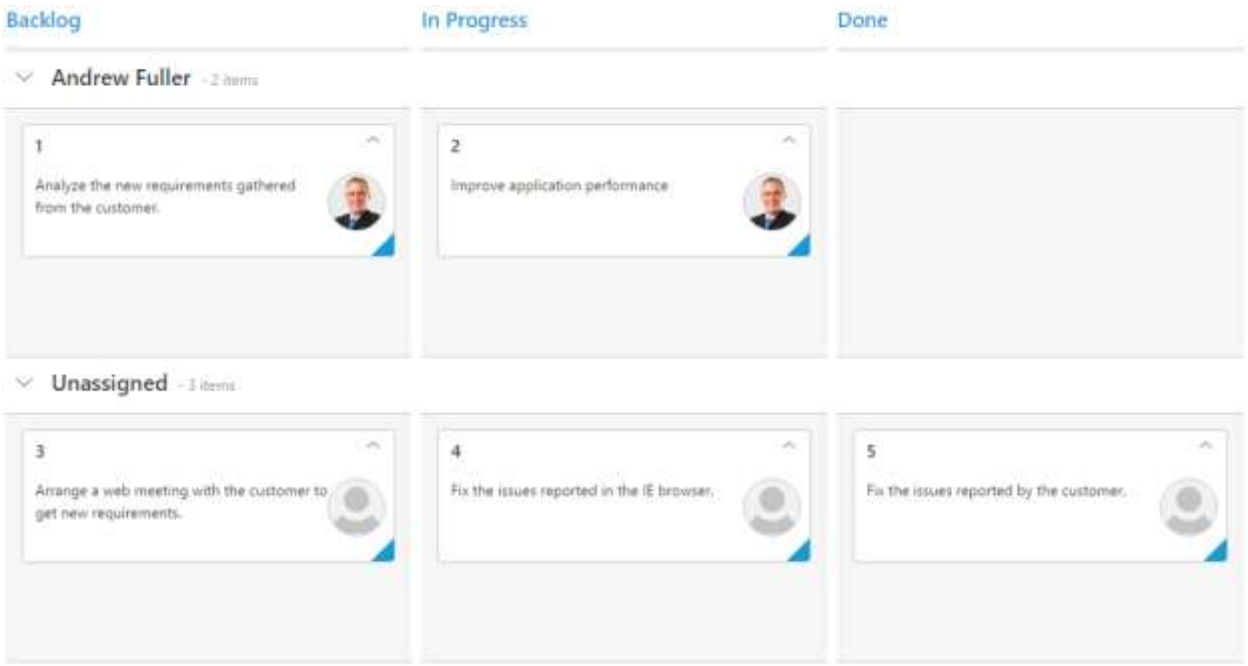
C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
List<Tasks> Task = new List<Tasks>();
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "", "", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, null, "", 2));
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Task.Add(new Tasks(5, "Close", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "", "", 1));
ViewBag.datasource = Task;
return View();
}
}
}
```

The output of the above code example.



Using user defined values

You can override default values for unassigned swim lane group using the property Keys.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.SwimlaneKey("Assignee")
.ImageUrl("Image")
.PrimaryKey("Id");
})
.SwimlaneSettings(ss =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

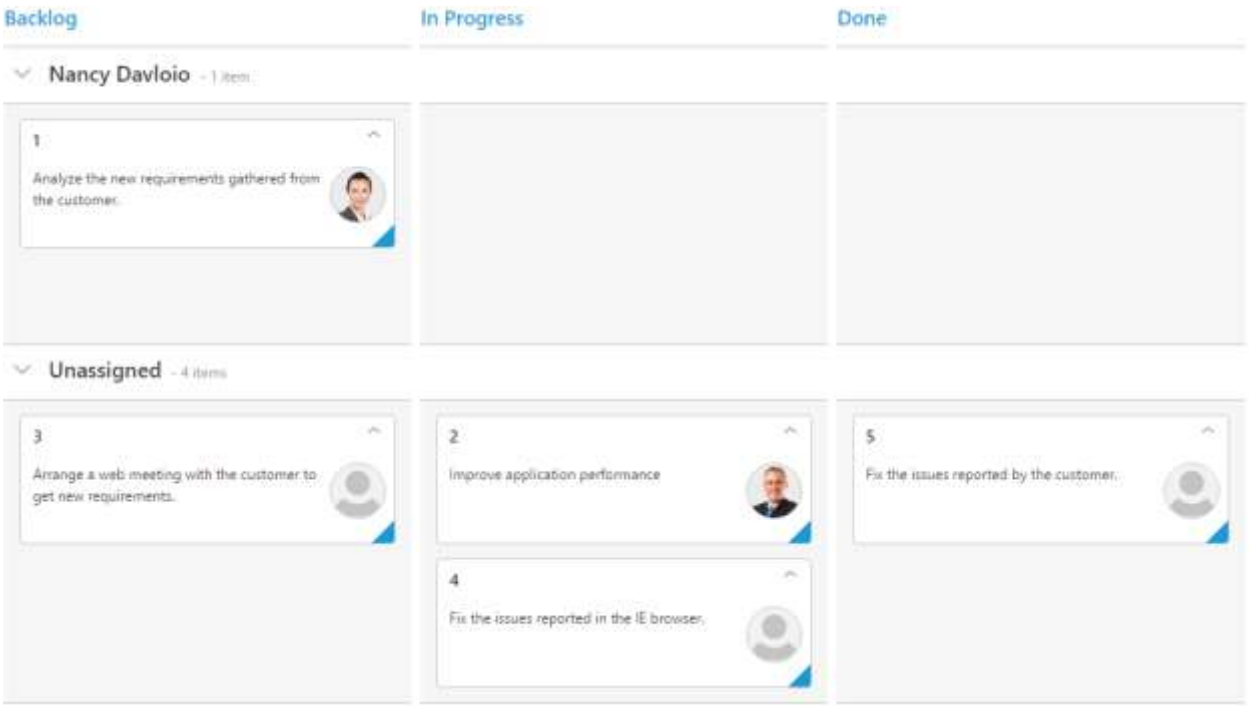
```
ss.UnassignedGroup(ug =>
ug.Keys(key =>
{
key.Add("Andrew Fuller");
key.Add("");
key.Add("null");
}
)
)
)
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
List<Tasks> Task = new List<Tasks>();
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Nancy",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "", "", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, null, "", 2));
Task.Add(new Tasks(5, "Close", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "", "", 1));
ViewBag.datasource = Task;
return View();
}
}
}
```

The output of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Drag and Drop

By default, AllowDragAndDrop is true. Cards can be transited from one column to another column, by dragging and dropping. And it has drop position indicator which enables easier positioning of cards

Note: It is not possible transit cards to other swim lanes through Drag and Drop.

Drag and Drop to external control

Kanban is now provided support to drag and drop cards to external controls, when AllowExternalDragAndDrop is set to true. Along with that you need to specify the ID of the component to which the card must be dropped in CardSettings property ExternalDropTarget.

Here, in order to drag and drop cards between two Kanban, enable the Kanban property AllowExternalDragAndDrop and specify the target Kanban ID in ExternalDropTarget.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Open").Key("Open,InProgress").Add();
col.HeaderText("Done").Key("Testing,Close").Add();
})
.AllowTitle(true)
.KeyField("Status")
.AllowExternalDragAndDrop(true)
.CardSettings(card =>
{
card.ExternalDropTarget("#Kanban1");
})
.Fields(field =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
@(Html.EJ().Kanban("Kanban1")
.DataSource((IEnumerable<object>)ViewBag.datasource1)
.Columns(col =>
{
col.HeaderText("Open").Key("Open,InProgress").Add();
col.HeaderText("Done").Key("Testing,Close").Add();
})
.AllowTitle(true)
.AllowExternalDragAndDrop(true)
.CardSettings(card =>
{
card.ExternalDropTarget("#Kanban");
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
)
```

C#

```
public partial class KanbanController : Controller
{
List<Tasks> Task = new List<Tasks>();
List<Tasks> Task1 = new List<Tasks>();
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
Task.Add(new Tasks(6, "Close", "Arrange a web meeting with the customer to
get the login page requirements.", "Others", "Low", "Meeting", 2, "Janet",
"../content/images/kanban/6.png", 1));
Task1.Add(new Tasks(7, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
```

Copyright © 2001 - 2020 Syncfusion Inc.

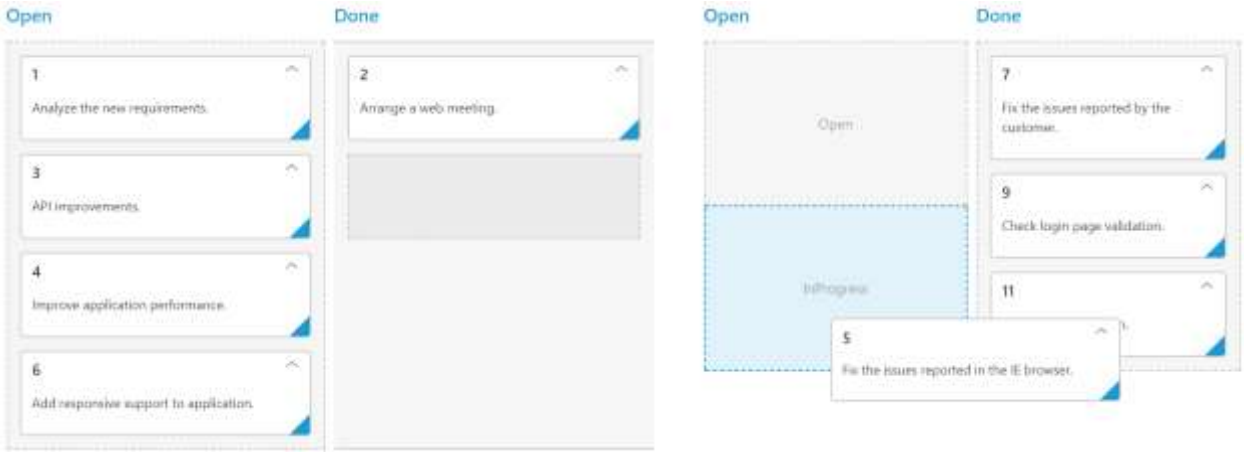
```
Task1.Add(new Tasks(8, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task1.Add(new Tasks(9, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task1.Add(new Tasks(10, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task1.Add(new Tasks(11, "Testing", "Fix the issues reported by the
customer.", "Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
ViewBag.datasource = Task;
ViewBag.datasource1 = Task1;
return View();
}
public class Tasks
{
public Tasks()
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public string Image { get; set; }
public int RankId { get; set; }
}
}
```

CSS

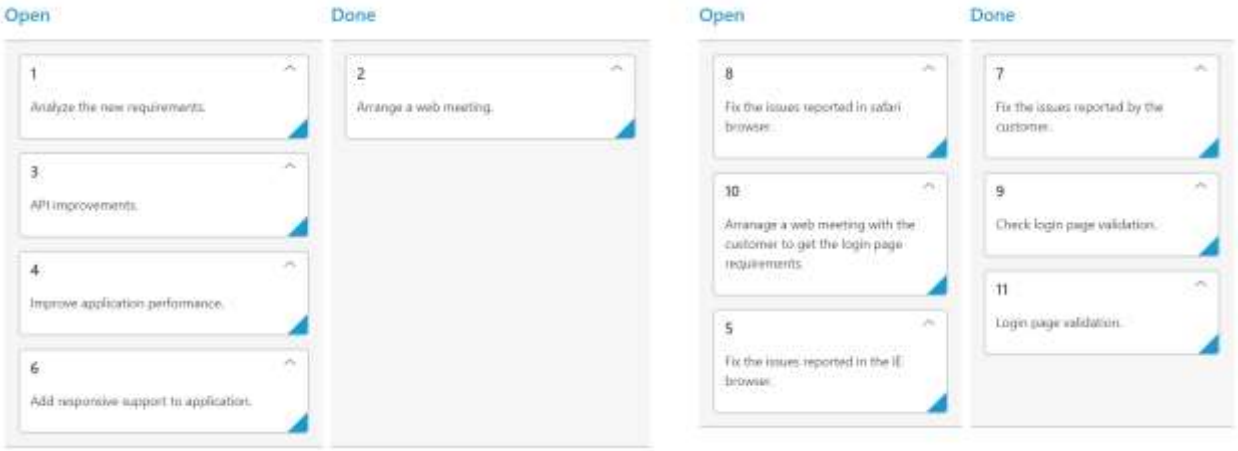
```
.e-kanban .e-draggedcard {
z-index:1;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output is displayed before dropping Kanban cards.



The following output is displayed after dropping Kanban cards.



See Also

You can also drag and drop Kanban cards to any custom control. For instance, let it be a Grid control. Enable the Kanban property AllowExternalDragAndDrop and specify the target Grid element ID in ExternalDropTarget.

Note: The target control must have droppable feature to handle the card dropped from Kanban and customize it based on corresponding control.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Open").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.AllowTitle(true)
.KeyField("Status")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowExternalDragAndDrop(true)
.CardSettings(card =>
{
card.ExternalDropTarget("#Kanban1");
})
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.ClientSideEvents(eve => { eve.CardDragStop("cardDragStop"); })
)
@(Html.EJ().Grid<object>("FlatGrid")
.Datasource((IEnumerable<object>)ViewBag.DataSource1)
.AllowSorting()
.AllowPaging()
.EditSettings(edit => { edit.AllowAdding(); })
.Columns(col =>
{
col.Field("Id").HeaderText("Card ID").IsPrimaryKey(true).Add();
col.Field("Status").HeaderText("Status").Add();
col.Field("Assignee").HeaderText("Assignee").Add();
col.Field("RankId").HeaderText("Rank ID").Add();
})
)
```

C#

```
public partial class KanbanController: Controller
{
List<Tasks> Task = new List<Tasks>();
List<Tasks> Task1 = new List<Tasks>();
public ActionResult KanbanFeatures()
{
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Close", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
ViewBag.datasource = Task;
ViewBag.datasource1 = Task1;
return View();
}
public class Tasks
{
public Tasks()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
}
public Tasks(int Id, string Status, string Summary, string Type, string
Priority, string Tags, double Estimate, string Assignee, string Image, int
RankId)
{
this.Id = Id;
this.Status = Status;
this.Summary = Summary;
this.Type = Type;
this.Priority = Priority;
this.Tags = Tags;
this.Estimate = Estimate;
this.Assignee = Assignee;
this.Image = Image;
this.RankId = RankId;
}
public int Id { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public string Image { get; set; }
public int RankId { get; set; }
}
}
```

JAVASCRIPT

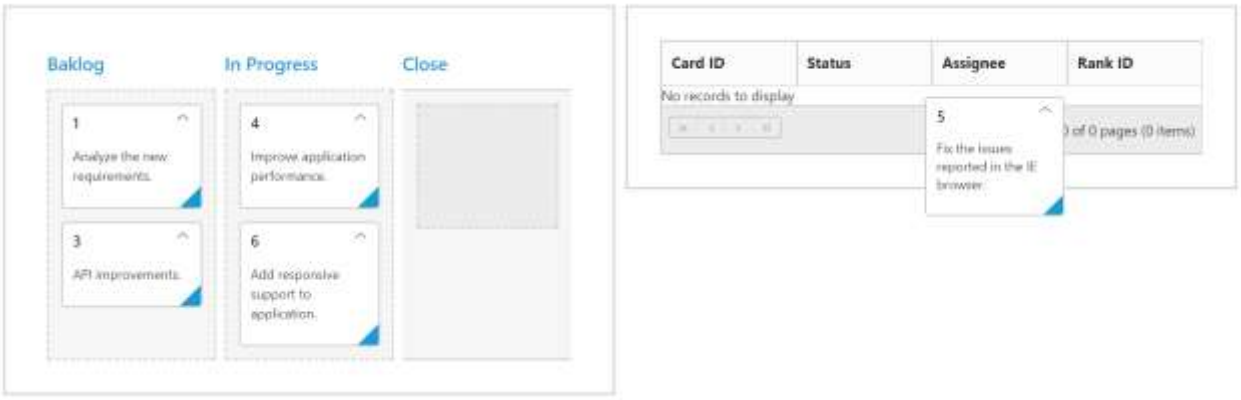
```
function cardDragStop(args) {
if ($(args.dropTarget).parents(".e-grid").length != 0) {
var gridObj = $("#FlatGrid").data("ejGrid");
gridObj.addRecord(args.data[0][0]);
gridObj.refreshContent();
}
}
```

CSS

```
.e-kanban .e-draggedcard {
z-index:1;
}
```

The following output is displayed before dropping the cards on Grid.

Copyright © 2001 - 2020 Syncfusion Inc.



The following output is displayed after dropping the cards on Grid.



Cards

Customization

Cards can be customized with appropriate mapping fields from the database. The customizable mapping properties are listed as follows

Mapping Fields	Description
Content	Map the column name to use as content to cards.
Tag	Map the column name to use as tag. Multiple tags can be given with comma separated. E.g. "API","SQL, Database".
Color	Map the column name to use as colors to highlight cards left border.
ColorMapping	Map the colors to use with column values which is mapped with fields.color.
ImageUrl	Map the column name to use as image to cards.
PrimaryKey	Map the column name to use as primary key to cards.
PriorityKey	Map the column name to use as priority key to cards.

Copyright © 2001 - 2020 Syncfusion Inc.

Title	Map the column name to use as title to cards. Default title is primaryKey.
AllowTitle	Set as true to enable title for card.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.AllowTitle(true)
.AllowSelection(true)
.KeyField("Status")
.Fields(field =>
{
field.Color("Type")
.Content("Summary")
.ImageUrl("Image")
.Tag("Tags")
.Priority("RankId")
.PrimaryKey("Id");
})
.CardSettings(card=>
{
card.ColorMapping(color =>
{
color.Add("#cb2027", "Bug,Story")
.Add("#67ab47", "Improvement")
.Add("#fbae19", "Epic")
.Add("#6a5da8", "Others");
});
})
)
```

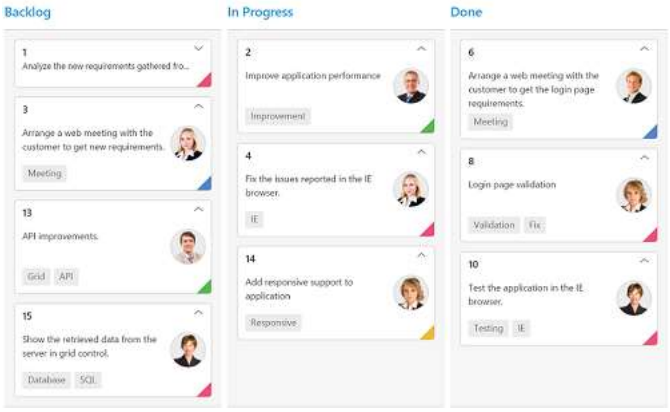
C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
```

The following output is displayed as a result of the above code example.



Template

Templates are used to create custom card layout as per the user convenient. HTML templates can be specified in the `Template` property of the `CardSettings` as an ID of the template's HTML element.

You can use JsRender syntax in the template. For more information about JsRender syntax, please refer this link.

The following code example describes the above behavior.

HTML

```
<script id="template" type="text/x-jsrender">
<table>
<tr>
<td class="photo">
<img src="{{"{"}}:Priority{{}}}}.png">
</td>
<td class="details">
<table>
<colgroup>
<col width="30%">
<col width="70%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">   Name: </td>
<td>{{"{"}}:Assignee{{}}}}</td>
</tr>
<tr>
<td class="CardHeader">   Task: </td>
<td>{{"{"}}:Type{{}}}}</td>
</tr>
</tbody>
</table>
</td>
</tr>
</table>
```

```
</script>
```

CSS

```
<!--CSS for card template-->
<style>
.e-templatetable {
width: 100%;
}
.details > table {
margin-left: 2px;
border-collapse: separate;
border-spacing: 2px;
width: 100%;
}
.details td {
vertical-align: top;
}
.details {
padding: 8px 8px 10px 0;
}
.photo {
padding: 8px 6px 10px 6px;
text-align: center;
}
.CardHeader {
font-weight: bolder;
padding-right: 10px;
}
</style>
```

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Color("Type")
.PrimaryKey("Id");
})
.CardSettings(card=>
{
card.ColorMapping(color =>
{
color.Add("#cb2027", "Bug,Story")
.Add("#67ab47", "Improvement")
.Add("#fbae19", "Epic")
.Add("#6a5da8", "Others");
});
```

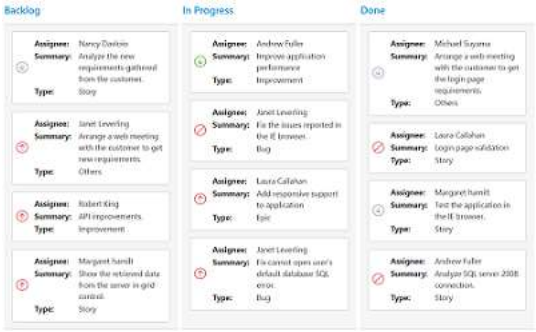
Copyright © 2001 - 2020 Syncfusion Inc.

```
card.Template("#template");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Tooltip

You can enable HTML tooltip for Kanban card elements by setting 'Enable property as true in TooltipSettings`.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.TooltipSettings(tooltip=>tooltip.Enable(true))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
```

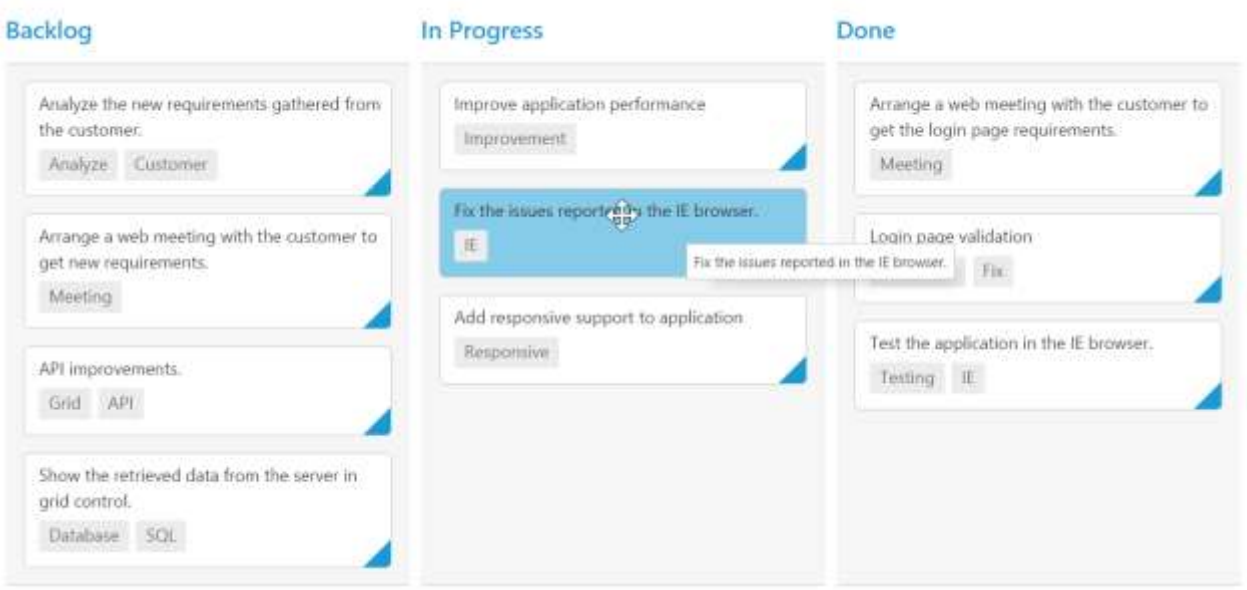
Copyright © 2001 - 2020 Syncfusion Inc.

```
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Template

By making use of template feature with tooltip, all the field names that are mapped from the dataSource can be accessed to define the Template tooltip for card. The TooltipSettings.Enable must be enabled first.

The following code example describes the tooltip template.

HTML

```
<script id="tooltipTemp" type="text/x-jsrender">
<div class='e-kanbantooltiptemplate'>
```

```
<table>
<tr>
<td class="photo">
<img src="{{"{{"}}:ImgUrl{{}}}}">
</td>
<td class="details">
<table>
<colgroup>
<col width="30%">
<col width="70%">
</colgroup>
<tbody>
<tr>
<td class="CardHeader">Assignee:</td>
<td>{{"{{"}}:Assignee{{}}}}</td>
</tr>
<tr>
<td class="CardHeader">Type:</td>
<td>{{"{{"}}:Type{{}}}}</td>
</tr>
<tr>
<td class="CardHeader">Estimate:</td>
<td>{{"{{"}}:Estimate{{}}}}</td>
</tr>
<tr>
<td class="CardHeader">Summary:</td>
<td>{{"{{"}}:Summary{{}}}}</td>
</tr>
</tbody>
</table>
</td>
</tr>
</table>
</div>
</script>
```

CSS

```
<!--toolTip template releated css -->
<style>
.details >table {
width: 100%;
margin-left:2px;
border-collapse: separate;
border-spacing: 1px;
}
.e-kanbantooltiptemplate {
width: 250px;
padding: 3px;
}
.e-kanbantooltiptemplate > table {
width: 100%;
}
.e-kanbantooltiptemplate td {
vertical-align: top;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
td.details {
padding-left: 10px;
}
.CardHeader {
font-weight: bolder;
}
</style>
```

RAZOR

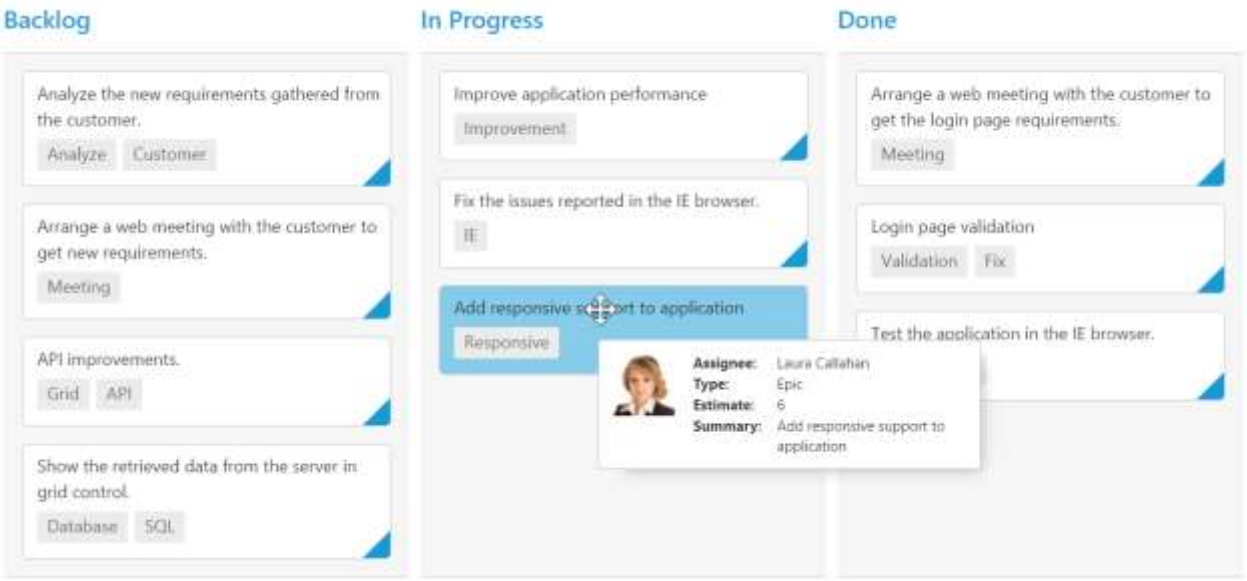
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.TooltipSettings(tooltip=>tooltip.Enable(true).Template("#tooltipTemp"))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Collapsible Cards

You can set particular cards collapsed state in Kanban by defining the CollapsibleCards property. Based on the CollapsibleCards object value, it maps the cards to the collapsible area.

You can set CollapsibleCards as object which consists of Field and Key properties. The Field property map the datasource field to be used in CollapsibleCards. The Key property map the specific column key to be in collapsed state.

Mapping Fields	Description
CollapsibleCards.Field	Map the collapsible card's field mapping.
CollapsibleCards.Key	Map the collapsible card's key mapping which is available in datasource value of field mapped in collapsibleCards.field.

Note: 1. If the CollapsibleCards with Field is in the dataSource and Key values specified will available in column values, then the cards will be rendered inside the collapsible card's division.

The following code example describes the collapsible cards.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Andrew").Key("Andrew Fuller").Add();
col.HeaderText("Janet").Key("Janet Leverling").Add();
col.HeaderText("Nancy").Key("Nancy Davloio").Add();
})
.AllowTitle(true)
.AllowSelection(false)
.KeyField("Assignee")
.Fields(field =>
```

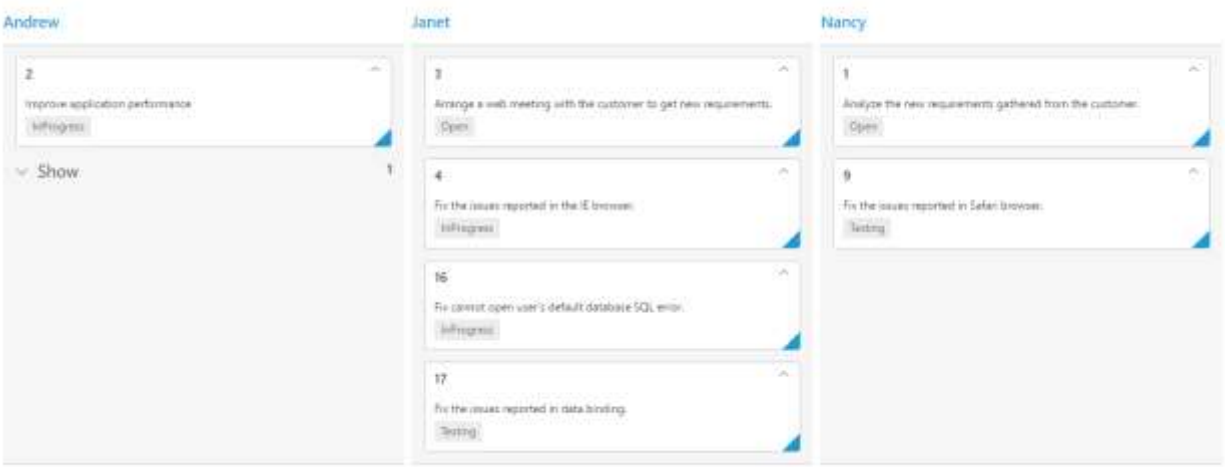
Copyright © 2001 - 2020 Syncfusion Inc.

```
{
field.PrimaryKey("Id");
field.Content("Summary");
field.Tag("Status");
field.CollapsibleCards(collapsibleCards =>
{
collapsibleCards.Field("Status").Key("Close");
});
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Prioritization of cards

Prioritizing cards is easy with drag-and-drop re-ordering that helps you to categorize most important work at the top. Cards can be categorized with priority by mapping specific database field into Priority property.

RankId defined in the dataSource which is consist priority of cards. The RankId will be changed while card ordering changes through Drag and Drop and Editing.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

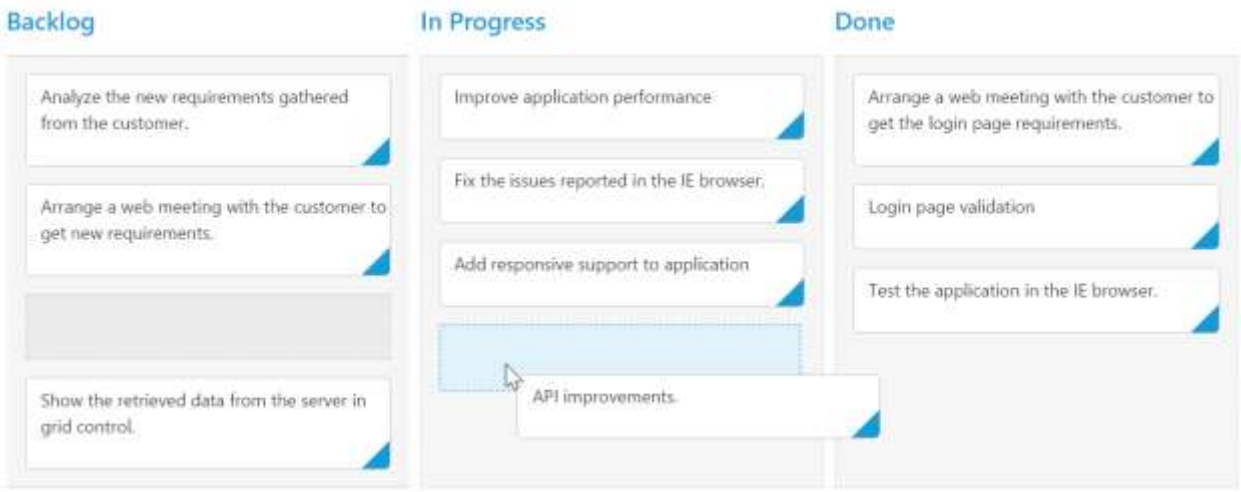
RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Priority("RankId")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Copyright © 2001 - 2020 Syncfusion Inc.

Note: For Drag and Drop event handling, please refer this API.

Note: If the Priority property is not set in the Kanban fields, the cards are dropped based on the specifying JSON data orders in a particular column. If not set, drop the card in a particular position based on the previous card Priority value and the next card's Priority values will change dynamically.

Editing

The Kanban control has support for dynamic insertion, updating and deletion of cards.

Set AllowEditing and AllowAdding property as true to enable editing/inserting respectively. The primary key for the data source should be defined in PrimaryKey, for editing to work properly.

You can start the edit action by double clicking the particular card. Similarly, you can add new card to Kanban either by double clicking the particular cell or on an external button which is bound to call addCard method of Kanban.

Deletion of the card is possible by using deleteCard by passing primary key as attribute.

Note: In Kanban, the primary key column will be automatically set to read only while editing the card which is to avoid duplicate entry in the cards.

Configuring Edit Items

You need to configure the list of data source fields that are allowable in editing state using EditItems property. The Field property of EditItems needs to be mapped with data source fields.

You can map the data source field as title to edit form using Title property of Fields. By default, it's mapped with PrimaryKey.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Assignee").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties()
```

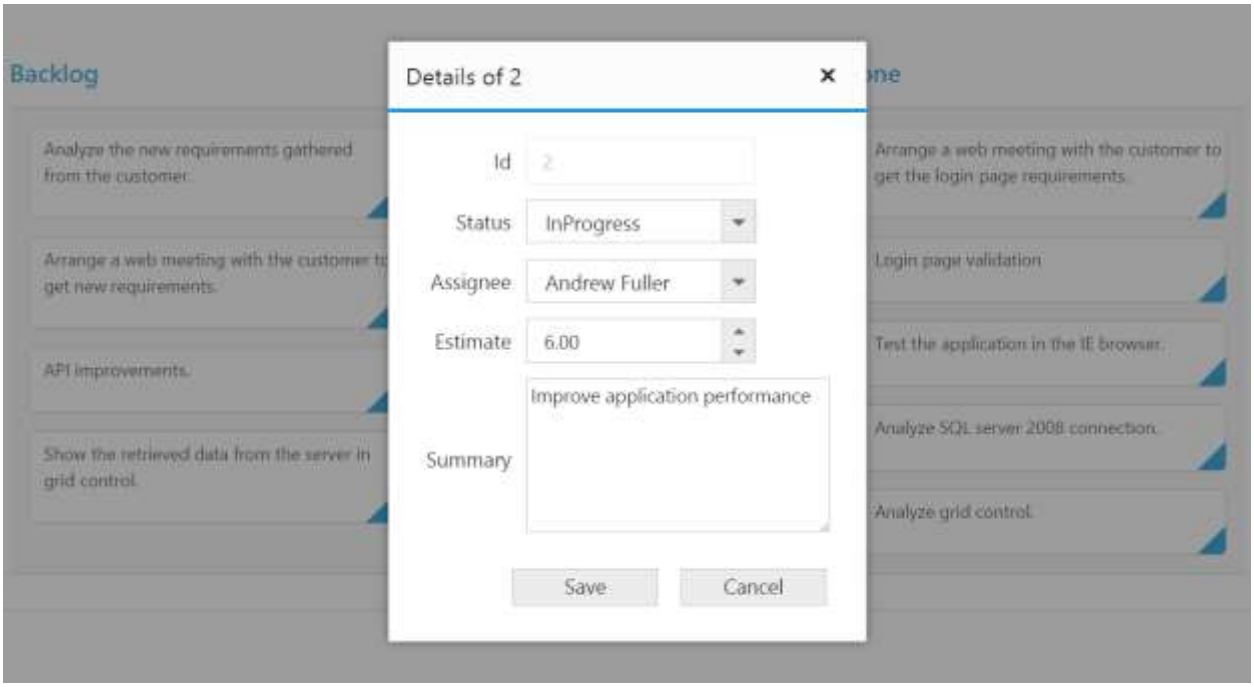
Copyright © 2001 - 2020 Syncfusion Inc.

```
{
DecimalPlaces = 2
}).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).Add();
});
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Edit modes

Dialog

Set EditMode as Dialog to edit data using a dialog box, which displays the fields associated with the data card being edited. Default value is Dialog.

Note: For EditMode property you can assign either string value (“Dialog”) or enum value (ej.Kanban.EditMode.Dialog).

The following code example describes the above behavior.

RAZOR

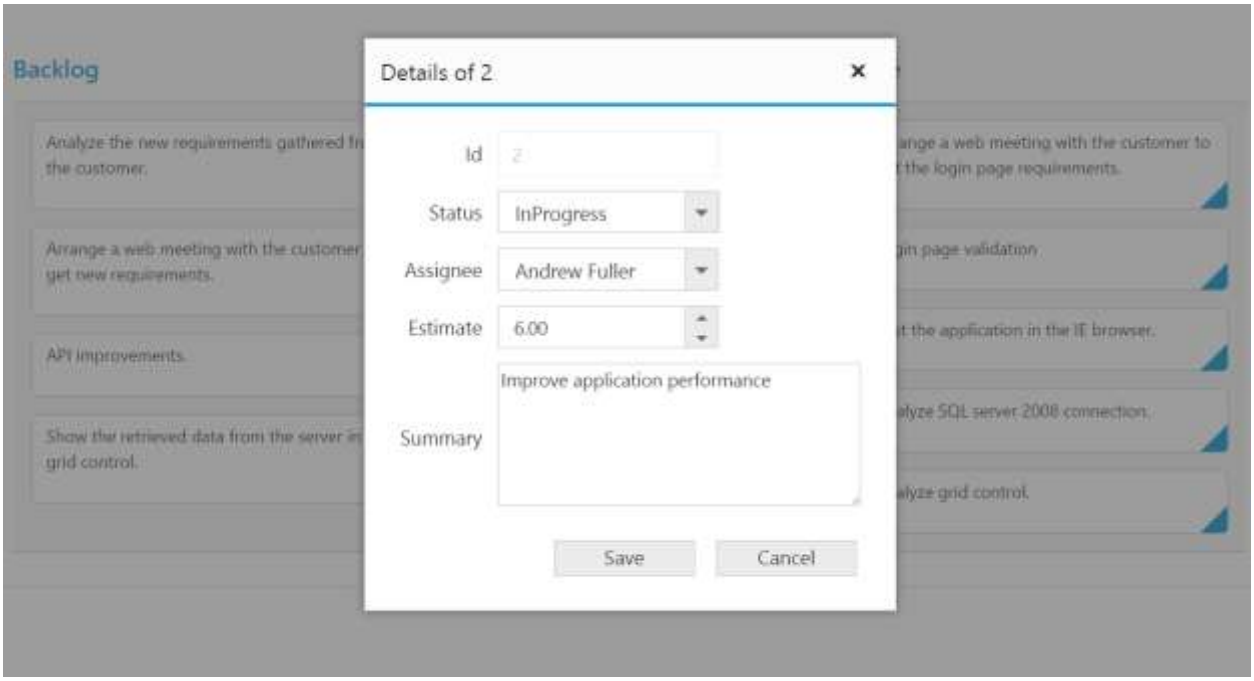
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Assignee").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties()
{
DecimalPlaces = 2
}).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).Add();
});
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output is displayed as a result of the above code example.



Dialog Template Form

You can edit any of the fields pertaining to a single card of data and apply it to a template so that the same format is applied to all the other cards that you may edit later.

Using this template support, you can edit the fields that are not bound to EditItems.

To edit the cards using Dialog template form, set EditMode as DialogTemplate and specify the template id to DialogTemplate property of EditSettings.

Note: 1. value attribute is used to bind the corresponding field value while editing.

2. name attribute is used to get the changed field values while save the edited card.

3. For EditMode property you can assign enum value (KanbanEditMode.DialogTemplate).

The following code example describes the above behavior.

HTML

```
<script id="template" type="text/template">
<table cellspacing="10">
<tr>
<td style="text-align: right;">Id
</td>
<td style="text-align: left">
<input id="Id" name="Id" value="{{: Id}}" class="e-field e-ejinputtext valid
e-disable" style="text-align: right; width: 175px; height: 28px"
disabled="disabled" />
</td>
<td style="text-align: right;">Status
</td>
```

```
<td style="text-align: left">
<select id="Status" name="Status">
<option value="Close">Close</option>
<option value="InProgress">InProgress</option>
<option value="Open">Open</option>
</select>
</td>
</tr>
<tr>
<td style="text-align: right;">Estimate
</td>
<td style="text-align: left">
<input type="text" id="Estimate" name="Estimate" value="{{:Estimate}}" />
</td>
<td style="text-align: right;">Assignee
</td>
<td style="text-align: left">
<select id="Assignee" name="Assignee">
<option value="Nancy Davloio">Nancy</option>
<option value="Andrew Fuller">Andrew Fuller</option>
<option value="Janet Leverling">Janet</option>
<option value="Margaret hamilt">Margaret</option>
<option value="Steven walker">Steven walker</option>
<option value="Michael Suyama">Michael</option>
<option value="Robert King">Robert King</option>
<option value="Laura Callahan">Laura Callahan</option>
</select>
</td>
</tr>
</table>
</script>
```

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ClientSideEvents(eve => eve.ActionComplete("complete"))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditMode(KanbanEditMode.DialogTemplate)
.DialogTemplate("#template");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

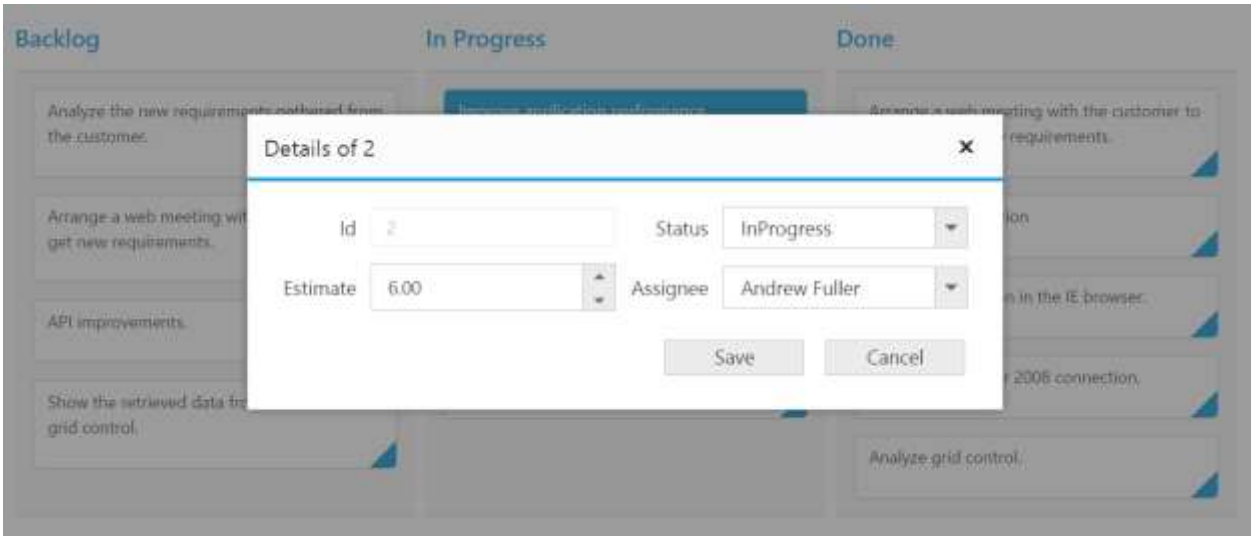
While using template, you can change the elements that are defined in the template, to appropriate Syncfusion JS controls based on the column type. This can be achieved by using ActionComplete event of Kanban. Please refer to following code snippets.

JAVASCRIPT

```
function complete(args) {
if ((args.requestType == "beginedit" || args.requestType == "add") &&
args.model.editSettings.editMode == "dialogtemplate") {
$("#Estimate").ejNumericTextbox({ value: parseFloat($("#Estimate").val()),
width: "175px", height: "34px", decimalPlaces: 2 });
$("#Assignee").ejDropDownList({ width: '175px' });
$("#Status").ejDropDownList({ width: '175px' });
$("#Priority").ejDropDownList({ width: '175px' });
if (args.requestType == "beginedit" || args.requestType == "add") {
$("#Assignee").ejDropDownList("setSelectedValue", args.data['Assignee']);
$("#Priority").ejDropDownList("setSelectedValue", args.data['Priority']);
$("#Status").ejDropDownList("setSelectedValue", args.data['Status']);
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



External Form

Set the EditMode as ExternalForm to open the edit form in outside kanban content.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).Add();
}).EditMode(KanbanEditMode.ExternalForm);
})
)
```

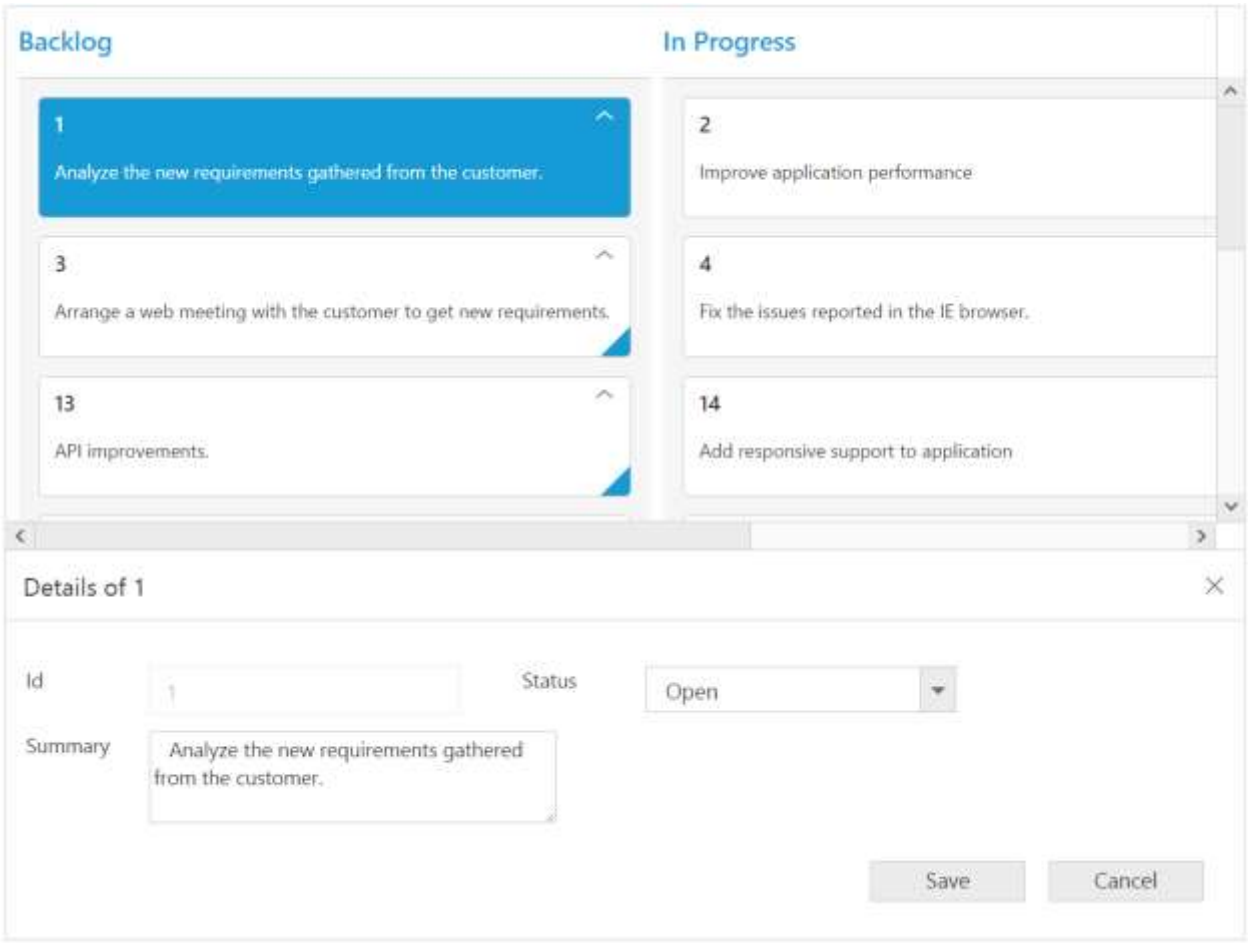
C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Form Position:

Form Position can be customized by setting the FormPosition property of `EditSettings' as "right" or "bottom".

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

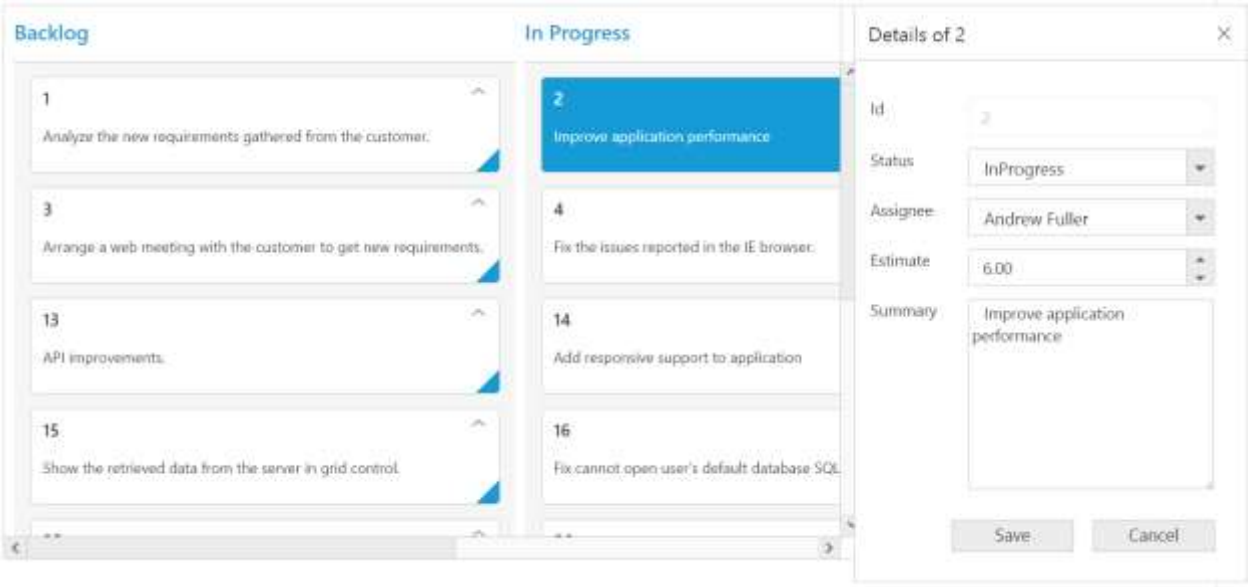
```
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.FormPosition(KanbanFormPosition.Right)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Assignee").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties()
{
DecimalPlaces = 2
}).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).Add();
}).EditMode(KanbanEditMode.ExternalForm);
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



External Template Form

You can edit any of the fields pertaining to a single card of data and apply it to a template so that the same format is applied to all the other cards that you may edit later.

Using this template support, you can edit the fields that are not bound to Kanban Edit Items.

To edit the cards using External template form, set `EditMode` as `ExternalFormTemplate` and specify the template id to `ExternalFormTemplate` property of `EditSettings`.

While using template, you can change the elements that are defined in the template, to appropriate Syncfusion JS controls based on the column type. This can be achieved by using `ActionComplete` event of Kanban.

Note: 1. `value` attribute is used to bind the corresponding field value while editing.

2. `name` attribute is used to get the changed field values while save the edited card.

3. For `EditMode` property you can assign `enum` value (`KanbanEditMode.ExternalFormTemplate`).

The following code example describes the above behavior.

HTML

```
<script id="template" type="text/template">
<table cellspacing="10">
<tr>
<td style="text-align: right;">Id
</td>
<td style="text-align: left">
<input id="Id" name="Id" value="{{: Id}}" class="e-field e-ejinputtext valid
e-disable" style="text-align: right; width: 175px; height: 28px"
disabled="disabled" />
</td>
<td style="text-align: right;">Status
</td>
<td style="text-align: left">
<select id="Status" name="Status">
<option value="Close">Close</option>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<option value="InProgress">InProgress</option>
<option value="Open">Open</option>
</select>
</td>
</tr>
<tr>
<td style="text-align: right;">Assignee
</td>
<td style="text-align: left">
<select id="Assignee" name="Assignee">
<option value="Nancy">Nancy</option>
<option value="Andrew Fuller">Andrew Fuller</option>
<option value="Janet">Janet</option>
<option value="Margaret">Margaret</option>
<option value="Steven walker">Steven walker</option>
<option value="Michael">Michael</option>
<option value="Robert King">Robert King</option>
<option value="Laura Callahan">Laura Callahan</option>
</select>
</td>
</tr>
</table>
</script>
```

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ClientSideEvents(eve => eve.ActionComplete("complete"))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditMode(KanbanEditMode.ExternalFormTemplate)
.ExternalFormTemplate("#template");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

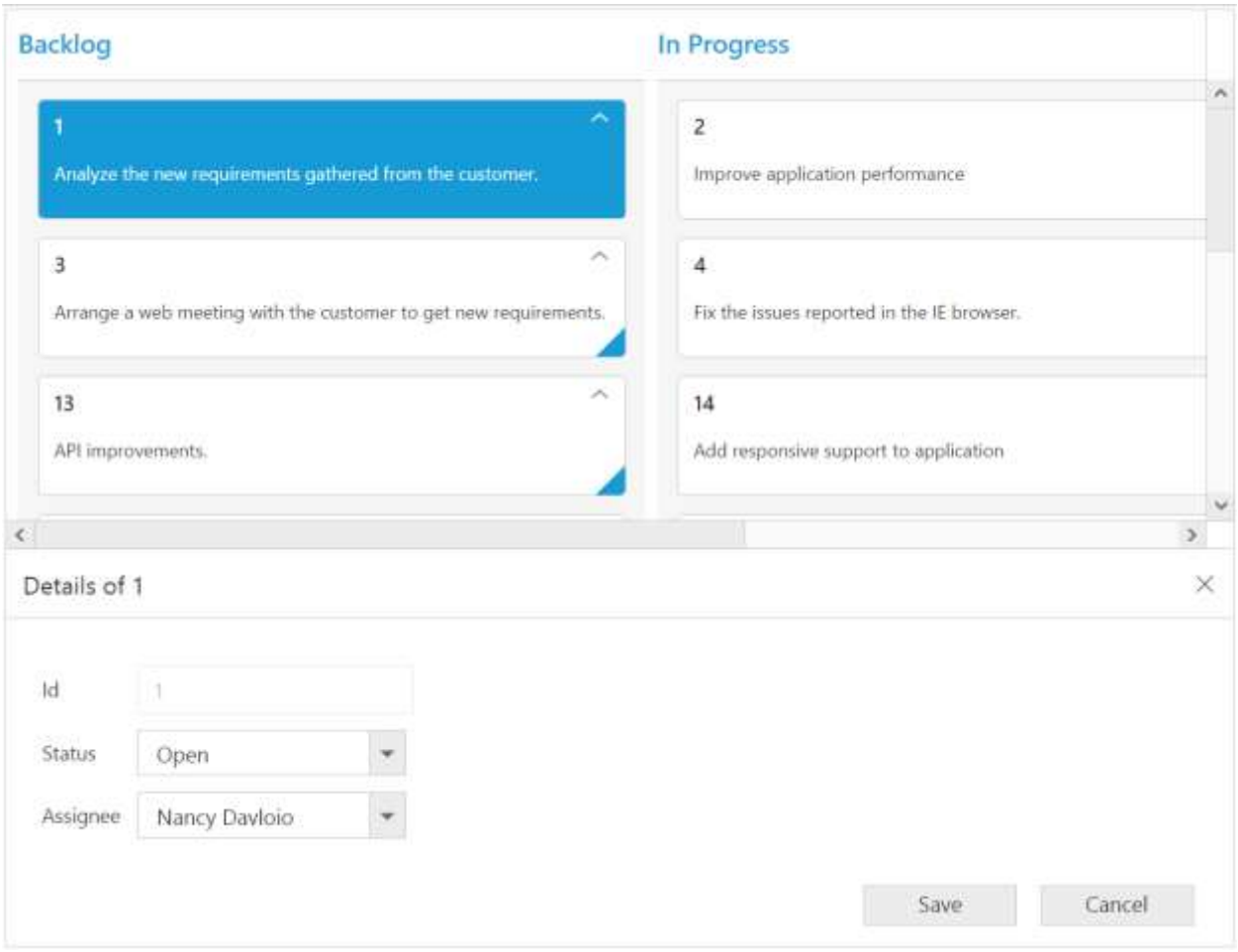
```
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

JAVASCRIPT

```
function complete(args) {
if ((args.requestType == "beginedit" || args.requestType == "add") &&
args.model.editSettings.editMode == "externalformtemplate") {
$("#Assignee").ejDropDownList({ width: '175px' });
$("#Status").ejDropDownList({ width: '175px' });
if (args.requestType == "beginedit" || args.requestType == "add") {
$("#Assignee").ejDropDownList("setSelectedValue", args.data['Assignee']);
$("#Status").ejDropDownList("setSelectedValue", args.data['Status']);
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Cell edit type and its params

The edit type of bound column can be customized using `EditType` property of `Columns`. The following Essential JavaScript controls are supported built-in by `EditType`. You can set the `EditType` based on specific data type of the column.

EditType	EditParams	Description	Example
Numeric	TextBoxes	control for integers, double, and decimal dataâ€™s	NumericEditOptions(new EditorProperties() { DecimalPlaces = 2 })
String	HTML Textbox	HTML Textbox	-
DatePicker	DatePicker	control for date data	DateEditOptions(new DatePickerProperties() { ButtonText="Now"})
DateTimePicker	DateTimePicker	control for date data-time data	DateTimeEditOptions(new DateTimePickerProperties() { Enabled=true})

Copyright © 2001 - 2020 Syncfusion Inc.

DropDown	DropDownList	control for list of data	DropdownEditOptions(new DropDownListProperties() { AllowGrouping=true})
RTE	RTE	control for customizing text in RTE format	RTEEditOptions(new RTEproperties() { EnableResize=true})
TextArea	HTML TextArea	Control for multi-line plain-text editing	

Note: 1. If EditType is not set, then by default it will display HTML textbox while editing a card.

2. For EditType property you can assign either string value (numericedit) or enum value (ej.Kanban.EditingType.Numeric).

The following code example describes the above behavior.

RAZOR

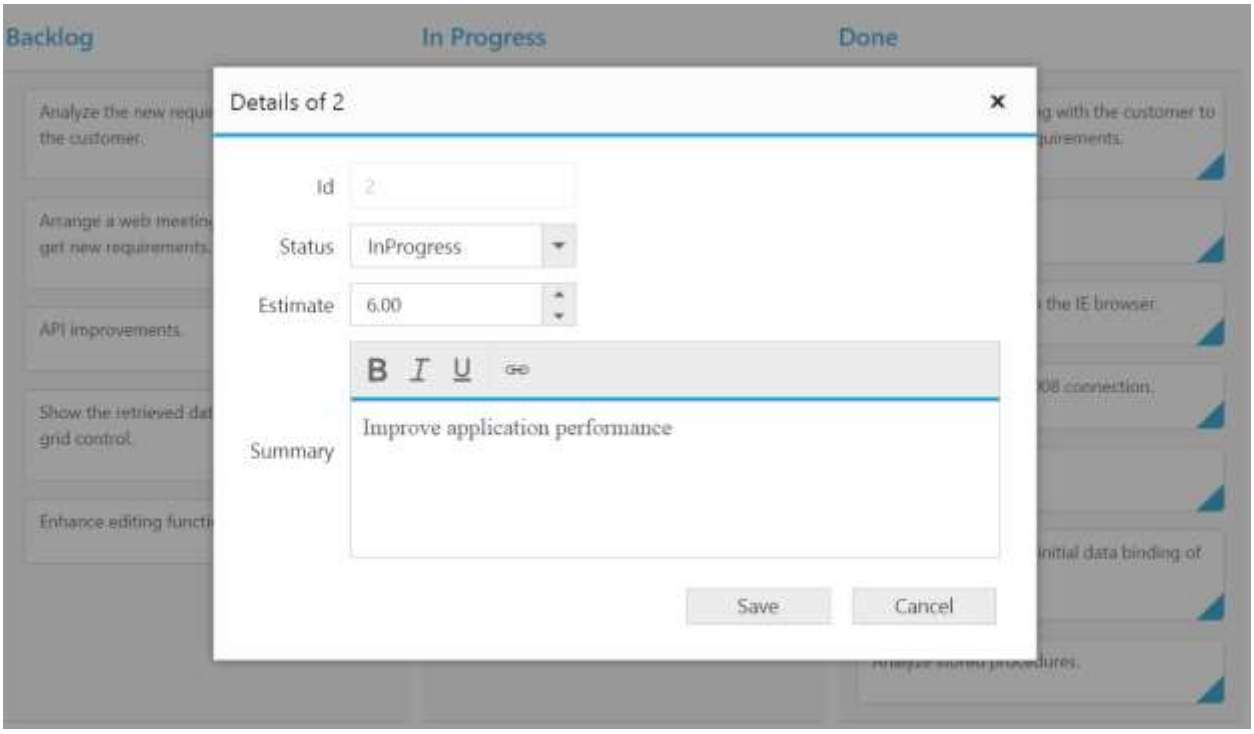
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties()
{
DecimalPlaces = 2
}).Add();
e.Field("Summary").EditType(KanbanEditingType.RTE).RTEEditOptions(new
RTEproperties() { Height = "150",MinHeight="100"}).Add();
});
})
)
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Column Validation

We can validate the value of the added or edited card cell before saving.

The below validation script files are needed when editing is enabled with validation.

1. jquery.validate.min.js 2. jquery.validate.unobtrusive.min.js

Note: If you enabled the unobtrusive option, then need to refer the jquery.validate.unobtrusive.min.js file in your application along with the other script.

jQuery Validation

You can set validation rules using ValidationRules property of Columns. The following are jQuery validation methods.

Copyright © 2001 - 2020 Syncfusion Inc.

List of jQuery validation methods

Rules	Description
required	Requires an element.
remote	Requests a resource to check the element for validity.
minlength	Requires the element to be of given minimum length.
maxlength	Requires the element to be of given maximum length.
rangelength	Requires the element to be in given value range.
min	The element requires a given minimum.
max	The element requires a given maximum.
range	Requires the element to be in a given value range.
email	The element requires a valid email.
url	The element requires a valid url.
date	Requires the element to be a date.
dateISO	The element requires an ISO date.
number	The element requires a decimal number.
digits	The element requires digits only.
creditcard	Requires the element to be a credit card number.
equalTo	Requires the element to be the same as another.

Kanban supports all the standard validation methods of jQuery, please refer the jQuery validation documentation link for more information.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id");
})
.EditSettings(edit =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

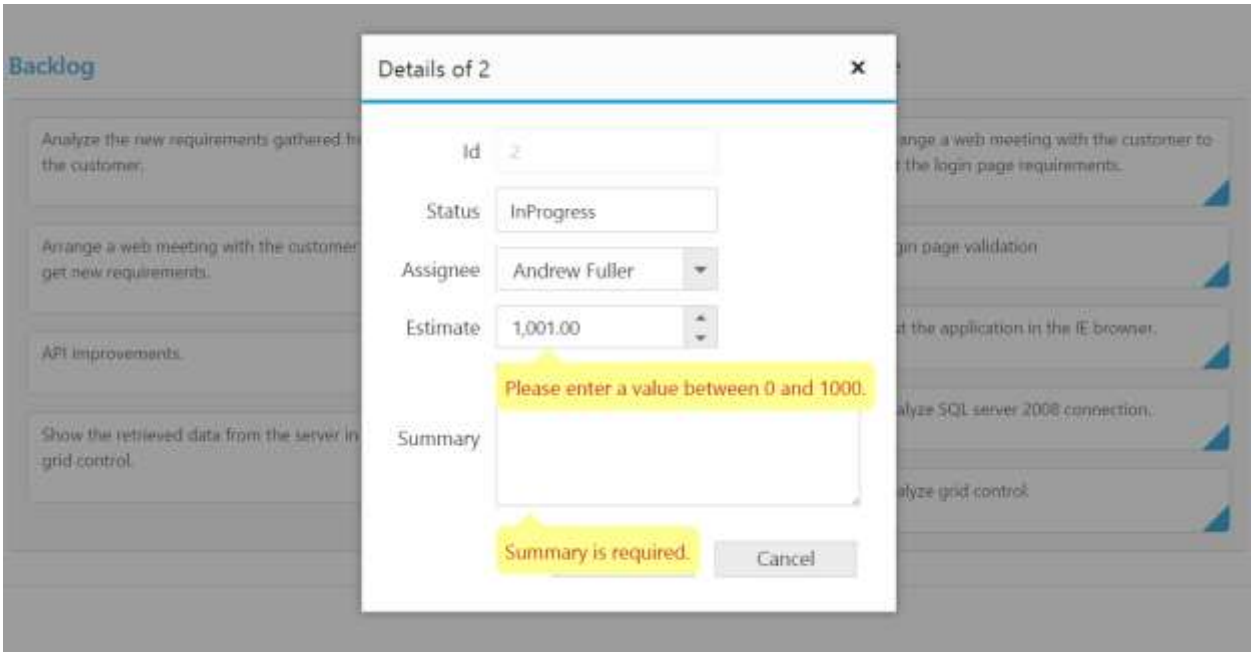
```
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Assignee").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties() { DecimalPlaces = 2 }).ValidationRules(rule => {
rule.AddRule("range", "[0,1000]"); }).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).ValidationRules(rule
=> { rule.AddRule("required", true); }).Add();
});
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Persisting data in server

Edited data can be persisted in database using RESTful web services.

All the CRUD operations in Kanban are done through DataManager. DataManager have an option to bind all the CRUD related data in server side. Please refer the link to know about the DataManager.

URL Adaptor

You can use the UrlAdaptor of DataManager when binding DataSource from remote data. At initial load of Kanban, using URL property of DataManager, data are fetched from remote data and bound to Kanban. You can map CRUD operation in Kanban to Server-Side Controller action using the property CrudURL.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource(data =>
data.URL("GetData").CrudURL("Crud").Adaptor(AdaptorType.UrlAdaptor))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").ShowAddButton(true).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.CustomToolbarItems(custom =>
{
custom.Template("#Delete").Add();
})
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
e.Field("Id").Add();
e.Field("Status").Add();
e.Field("Assignee").Add();
e.Field("Estimate").Add();
e.Field("Summary").Add();
}).EditMode(KanbanEditMode.Dialog);
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.PrimaryKey("Id")
.Priority("RankId");
})
.ClientSideEvents(eve => eve.ToolBarClick("toolbarClick"))
)
<script type="text/javascript">
function toolbarClick(args) {
if (args.itemName == "Delete" && this.element.find(".e-
kanbancard").hasClass("e-cardselection")) {
var selected = this.element.find(".e-cardselection");
this.KanbanEdit.deleteCard(selected.attr("id"));
}
}
</script>
<script id="Delete" type="text/template">
<a class="e-customdelete  e-icon" />
</script>
<style type="text/css" class="cssStyles">
.e-customdelete:before {
content: "\e800";
line-height: 26px;
min-height: 26px;
min-width: 14px;
display: inline-block;
}
</style>
```

Also when you use UrlAdaptor, you need to return the data as JSON and the JSON object must contain a properties result & count. The result holds the dataSource as its value and count holds the total cards count as its value.

The following code example describes the above behavior.

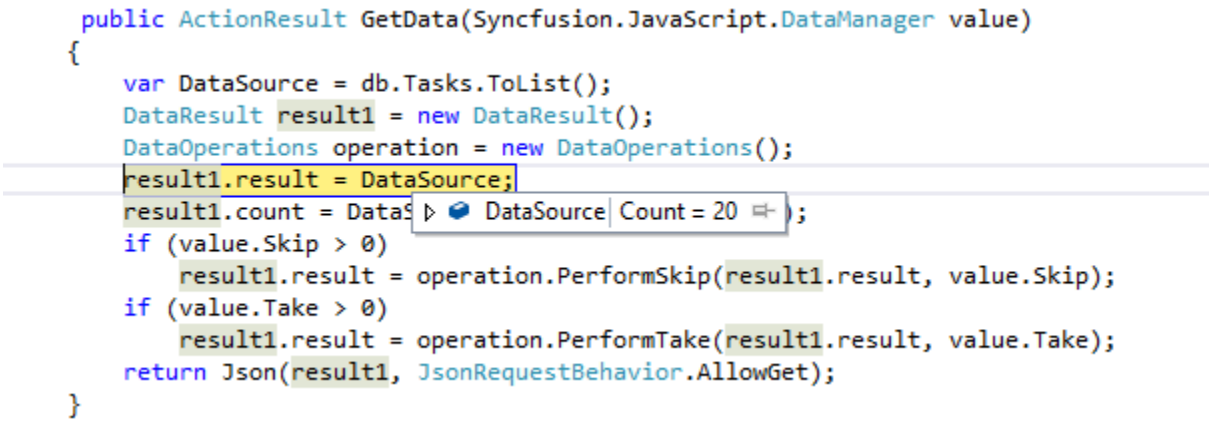
C#

```
public ActionResult GetData(Syncfusion.JavaScript.DataManager value)
{
var DataSource = db.Tasks.ToList();
DataResult result1 = new DataResult();
DataOperations operation = new DataOperations();
result1.result = DataSource;
result1.count = DataSource.AsQueryable().Count();
if (value.Skip > 0)
result1.result = operation.PerformSkip(result1.result, value.Skip);
if (value.Take > 0)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
result1.result = operation.PerformTake(result1.result, value.Take);
return Json(result1, JsonRequestBehavior.AllowGet);
}
public class DataResult
{
public IEnumerable result { get; set; }
public int count { get; set; }
}
```

Please refer to the below screenshot.



Using DataOperations helper class you can perform Kanban action at server side. The in-built methods that we have provided in the DataOperations class are listed below.

1. PerformTake 2. PerformSelect 3. Execute

Accessing CRUD action request details in server side

The Server-Side function must be declared with the following parameter name for each editing functionality.

Parameters Table

Action	Parameter Name	Example
Update, Insert, Remove, Crud Update(Multiple cards data will be passed to the server when drag and drop enabled with priority.)	Changed values,Added values,Deleted value	public ActionResult Crud(List changed, List added, List deleted)

Insert Card

Using CrudURL property, you can specify the controller action mapping URL to perform insert operation at server side.

The following code example describes the above behavior.

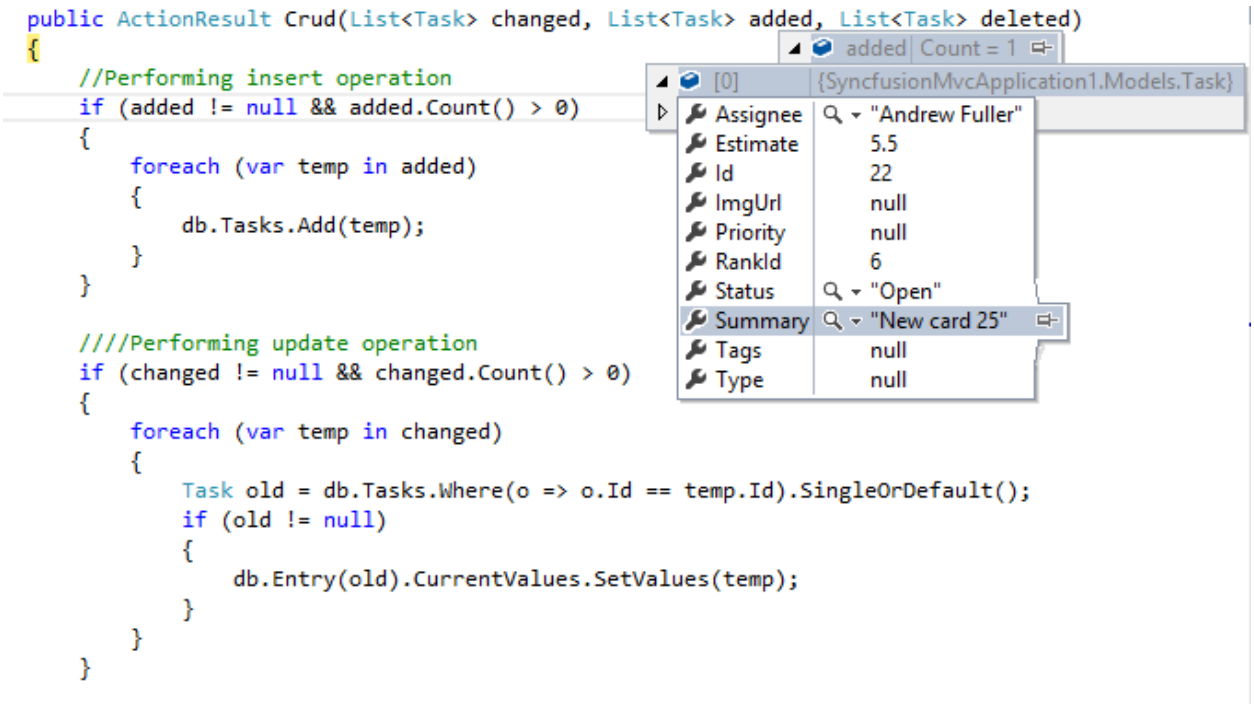
C#

```
public ActionResult Crud(List<Task> changed, List<Task> `added`, List<Task>
deleted)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
// Insert card in the database.
if (added != null && added.Count() > 0)
{
foreach (var temp in added)
{
db.Tasks.Add(temp);
}
}
db.SaveChanges();
var data = db.Tasks.ToList();
return Json(data, JsonRequestBehavior.AllowGet);
}
```

The newly added card details are bound to the added parameter. Please refer the below image.



Update Card

Using CrudURL property, you can specify the controller action mapping URL to perform save/update operation at server side.

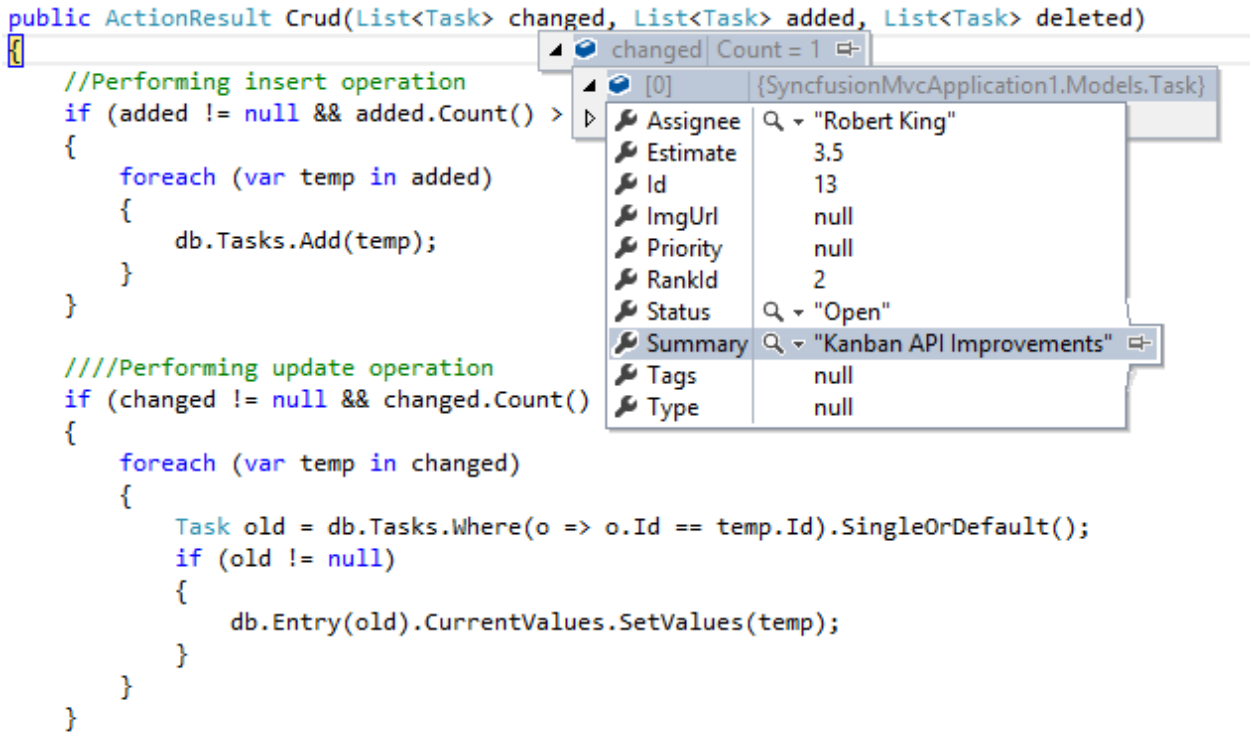
The following code example describes the above behavior.

C#

```
public ActionResult Crud(List<Task> `changed`, List<Task> added, List<Task>
deleted)
{
//Update card in the database.
if (changed != null && changed.Count() > 0)
{
foreach (var temp in changed)
{
Task old = db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault();
```

```
if (old != null)
{
db.Entry(old).CurrentValues.SetValues(temp);
}
}
}
db.SaveChanges();
var data = db.Tasks.ToList();
return Json(data, JsonRequestBehavior.AllowGet);
}
```

The updated card details are bound to the changed parameter. Please refer the below image.



Delete Card

Using CrudURL property, you can specify the controller action mapping URL to perform delete operation at server side.

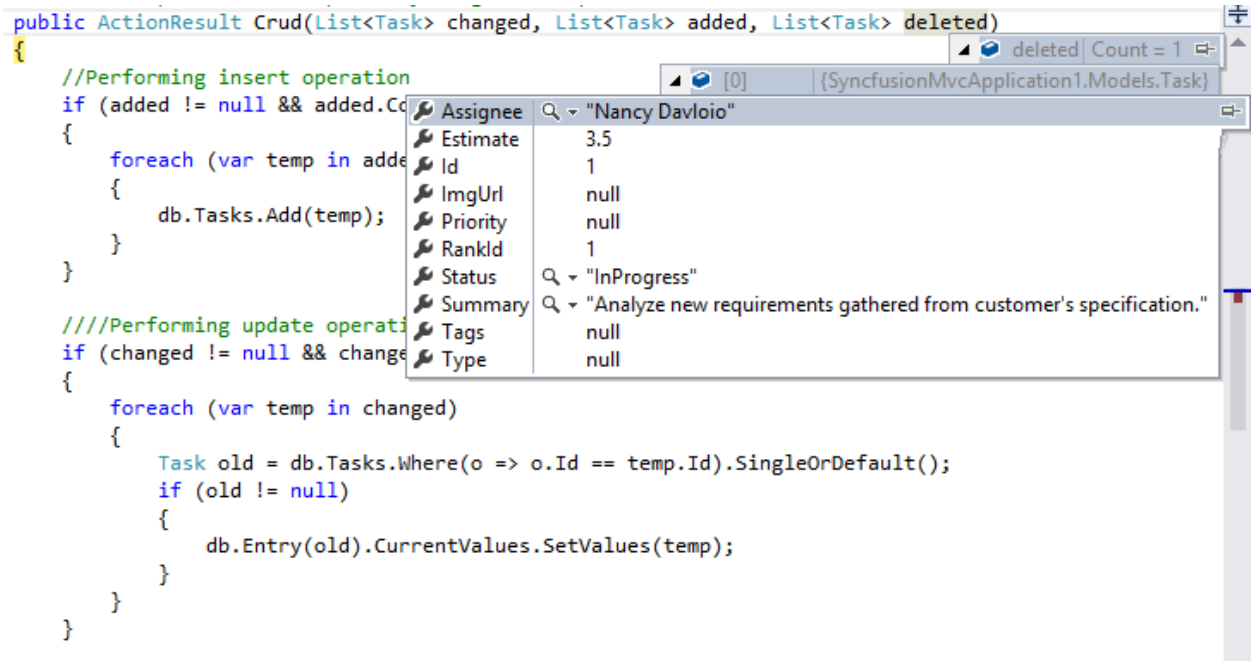
The following code example describes the above behavior.

C#

```
public ActionResult Crud(List<Task> changed, List<Task> added, List<Task>
deleted)
{
//Delete card in the database.
if (deleted != null && deleted.Count() > 0)
{
foreach (var temp in deleted)
{
db.Tasks.Remove(db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault());
}
```

```
}
db.SaveChanges();
var data = db.Tasks.ToList();
return Json(data, JsonRequestBehavior.AllowGet);
}
```

The deleted card details are bound to the deleted parameter. Please refer the below image.



Filtering

Filtering allows to filter the collection of cards from dataSource which meets the predefined query in the quick filters collection. To enable filtering, define FilterSettings collection with display Text and ej.Query.

You can also define display tip to describe filter definition to user using property Description. If the Description property is not defined, given Text will act as display tip.

We can also customize filter option through external button or customToolbarItems by using filterCards() method.

The following code example describes the above behavior.

RAZOR

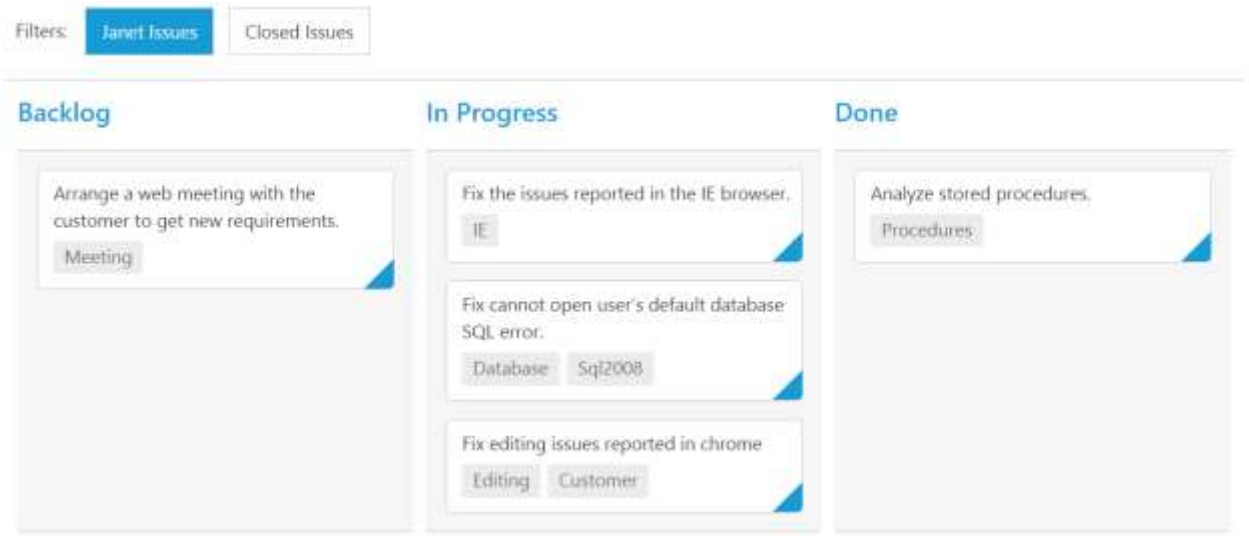
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
```

```
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
.FilterSettings(filter =>
{
filter.Text("Janet Issues").Query("new ej.Query().where('Assignee', 'equal',
'Janet')").Description("Displays issues which matches the assignee as
'Janet").Add();
filter.Text("Closed  Issues").Query("new ej.Query().where('Status', 'equal',
'Close')").Description("Display the 'Closed' issues").Add();
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Scrolling

Scrolling can be enabled by setting AllowScrolling as true. The height and width can be set to Kanban by using the properties ScrollSettings.Height and ScrollSettings.Width respectively.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The height and width can be set in percentage and pixel. The default value for Height and Width in ScrollSettings is 0 and auto respectively.

Set width and height in pixel

To specify the ScrollSettings.Width and ScrollSettings.Height in pixel, by set the pixel value as integer.

The following code example describes the above behavior.

RAZOR

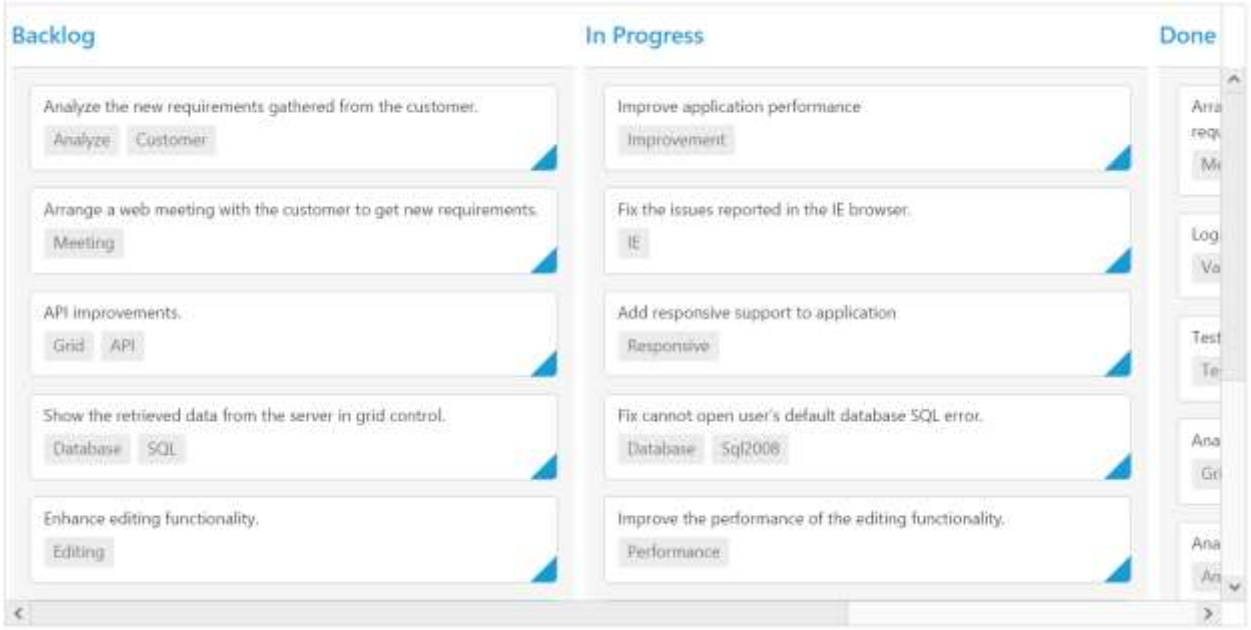
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling(true)
.ScrollSettings(scroll=>scroll.Width(900).Height(450))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Set height and width in percentage

To specify the ScrollSettings.Width and ScrollSettings.Height in percentage, by set the percentage value as string.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling(true)
.ScrollSettings(scroll => scroll.Width("70%").Height("70%"))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

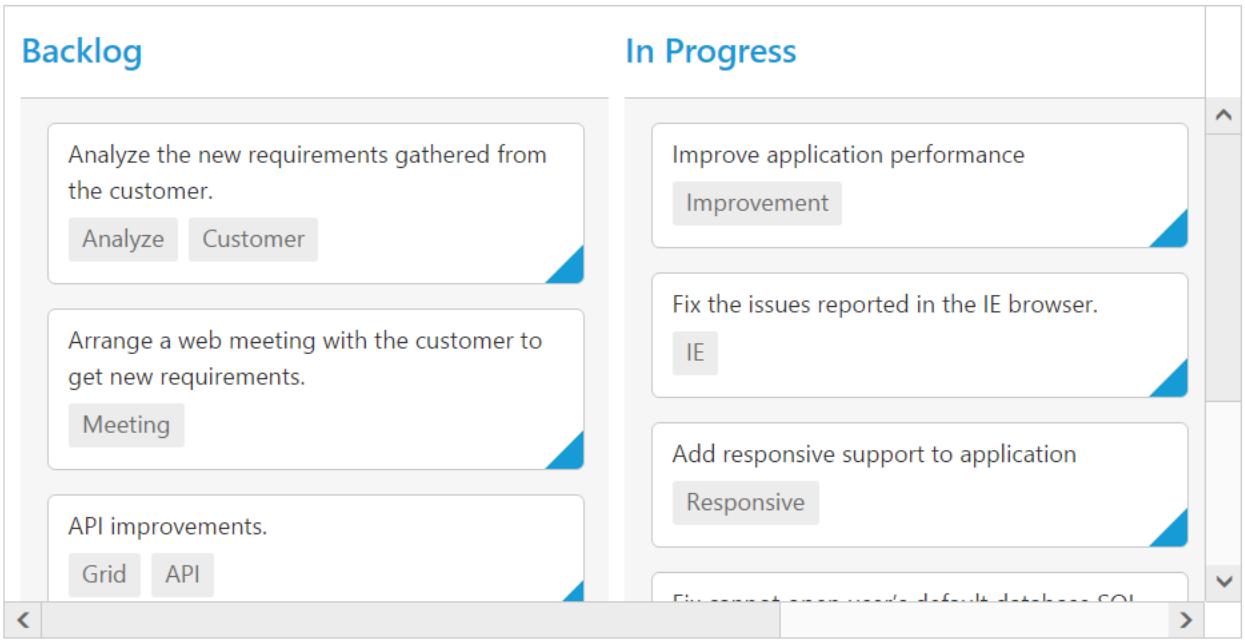
C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Set width as auto

Specify Width property of ScrollSettings as auto, then the scrollbar is rendered only when the Kanban width exceeds the browser window width.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling(true)
.ScrollSettings(scroll => scroll.Width("auto").Height("500"))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
```

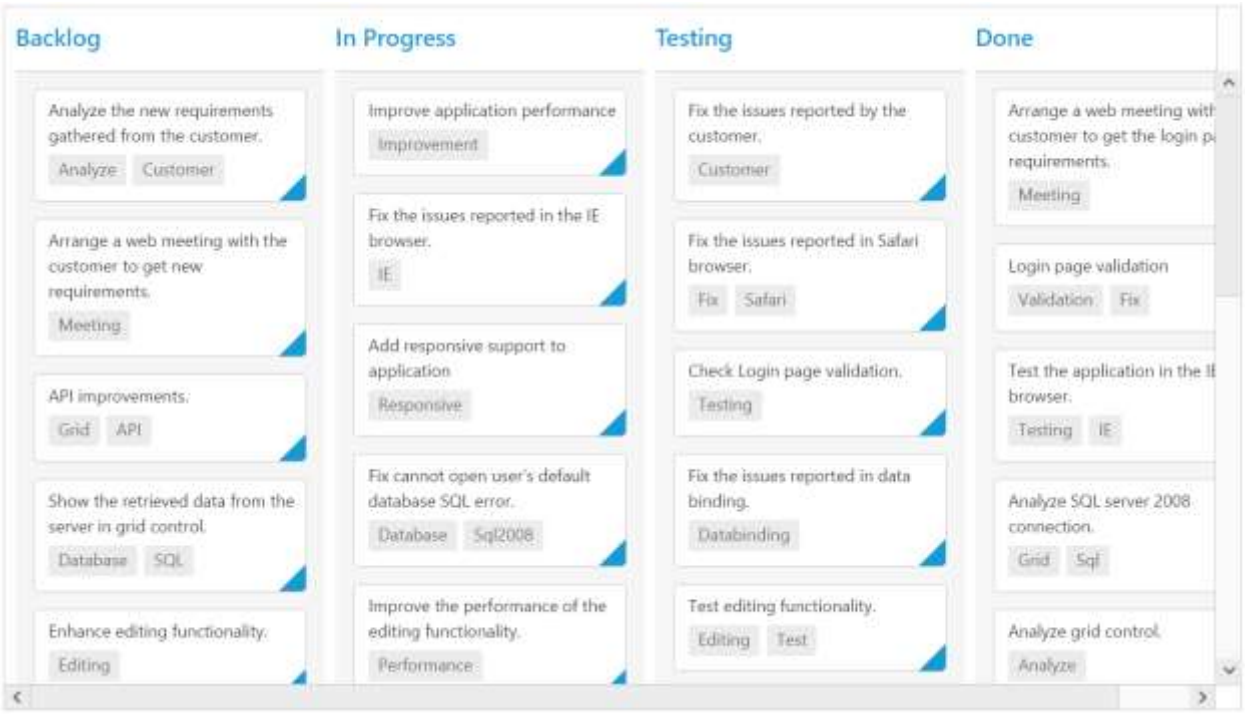
Copyright © 2001 - 2020 Syncfusion Inc.

```
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Enabling freeze swim lane

Set AllowFreezeSwimlane as true. This enables scrolling with freezing of swim lane until you scroll to the next Swim lane, which helps user to aware of current swim lane target.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
```

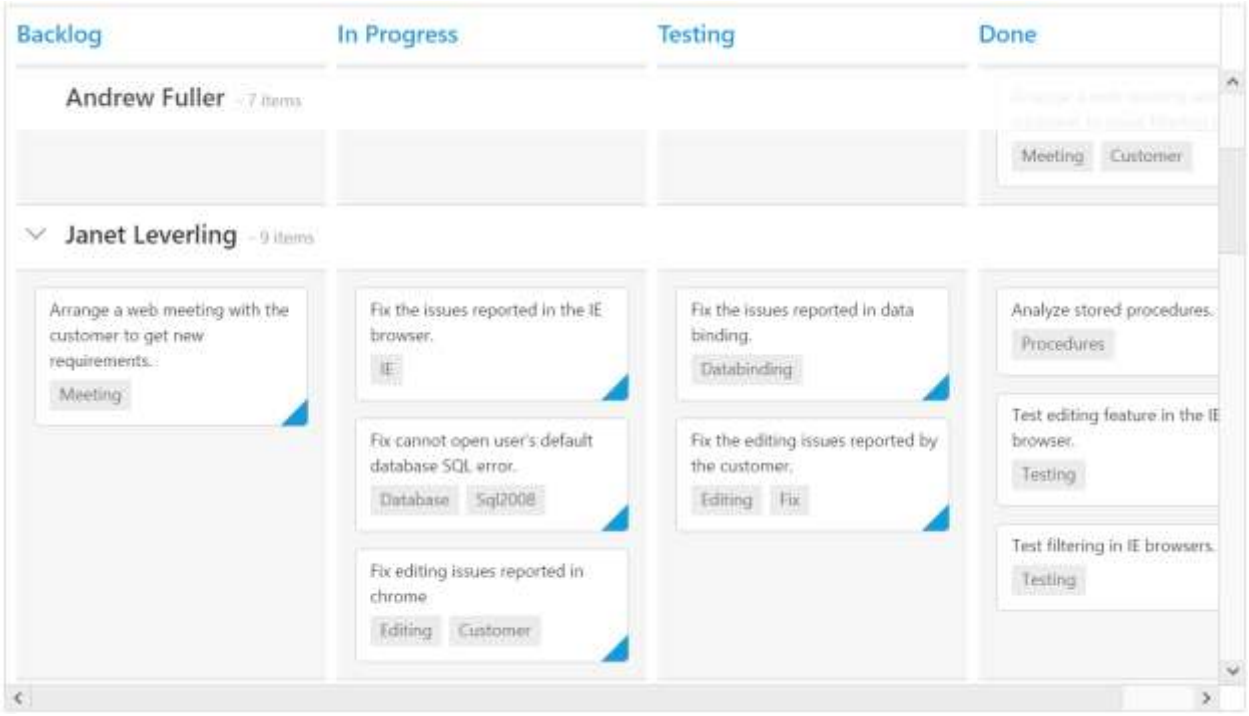
```
.DataSource((IEnumerable<object>)ViewBag.datasource)
.AllowScrolling(true)
.ScrollSettings(scroll =>
scroll.Width("auto").Height("500").AllowFreezeSwimlane(true))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: AllowFreezeSwimlane is applicable when swim lane grouping enabled by setting SwimlaneKey.

Selection & Hovering

Selection provides an interactive support to highlight the card that you select. Selection can be done through simple Mouse down or Keyboard interaction. To enable selection, set AllowSelection as true.

You can see the mouse hovering effect on the corresponding cards using AllowHover property. By default selection and hovering is true.

Types of Selection

Two types of selections available in Kanban are,

1. Single 2. Multiple

Single Selection

To enable single selection by setting SelectionType property as single.

Multiple Selection

Multiple selections are an interactive support to select a group of cards in Kanban by mouse or keyboard interactions. To enable multiple selections by set SelectionType property as multiple.

You can select multiple random cards below key press.

Keys	Description
Ctrl + mouse left	To select multiple random cards.
Shift + mouse left	To continuous cards select.

To unselect selected cards, by press “Shift + mouse left” click on selected row.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

RAZOR

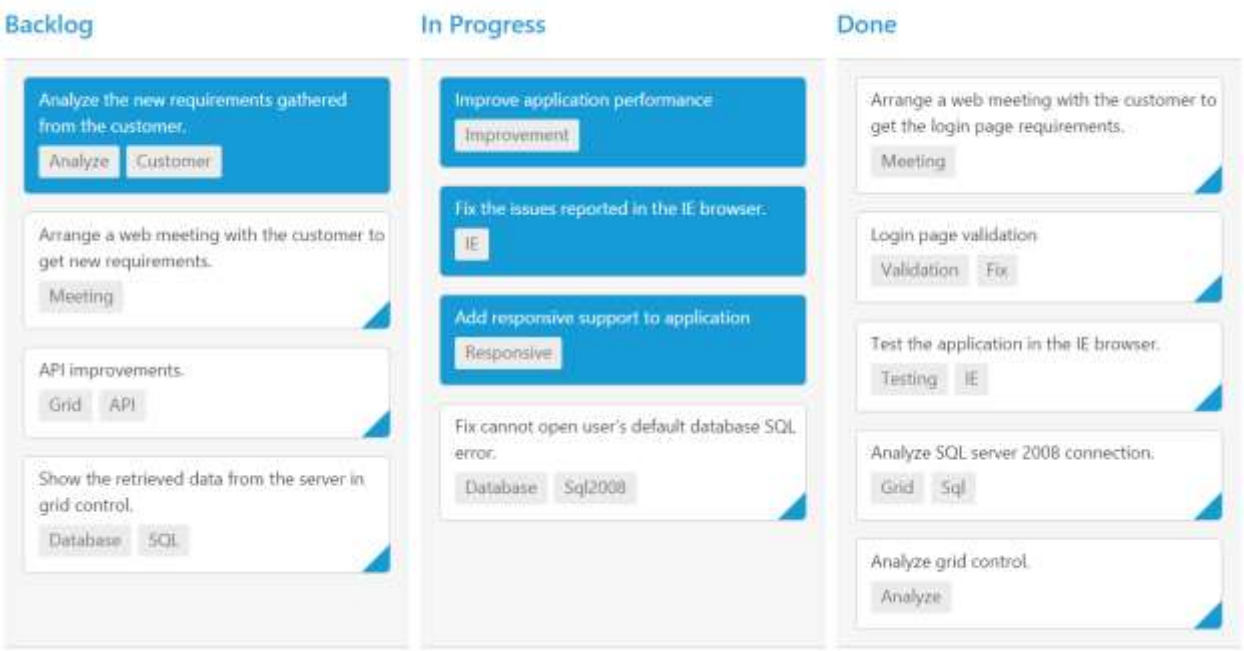
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
.SelectionType(SelectionType.Multiple)
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



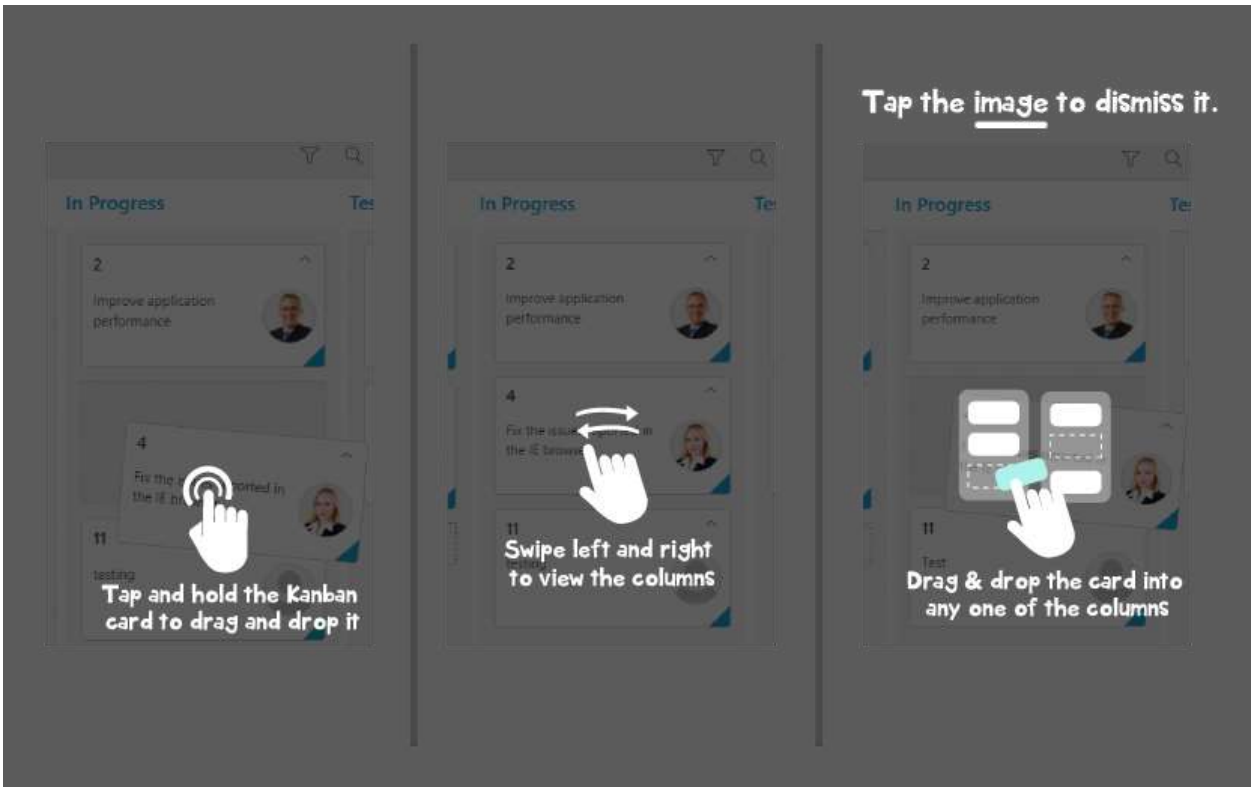
Responsive

The Kanban control has support for responsive behavior based on client browser's width and height. To enable responsive, IsResponsive property should be true.There are two modes of responsive layout is available in Kanban based on client width. They are.

- Mobile(<480px)
- Desktop(>480px)

You can check the image representation of touch actions from the below image.

Copyright © 2001 - 2020 Syncfusion Inc.



Mobile Layout

If client width is less than 480px, the Kanban will render in mobile mode. In which, you can see that kanban user interface is customized and redesigned for best view in small screens.To enable responsive, `IsResponsive` property should be true.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.IsResponsive(true)
.AllowSelection(false)
.AllowKeyboardNavigation(true)
.AllowTitle(true)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").ShowAddButton(true).Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.EditSettings(edit =>
{
edit.AllowAdding(true)
.AllowEditing(true)
.EditItems(e =>
{
e.Field("Id").ValidationRules(rule => { rule.AddRule("required",
true).AddRule("number", true); }).Add();
e.Field("Status").EditType(KanbanEditingType.Dropdown).Add();
```

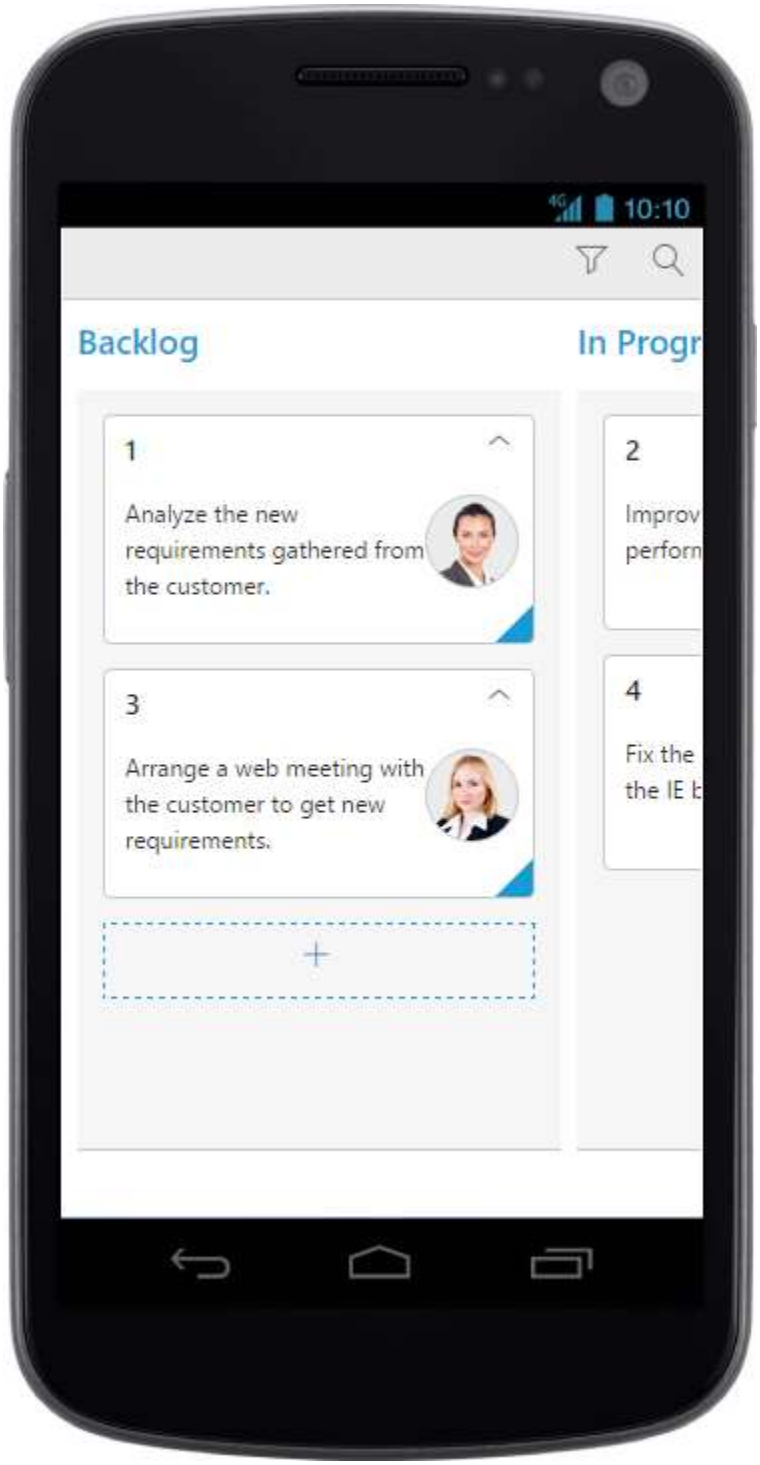
Copyright © 2001 - 2020 Syncfusion Inc.

```
e.Field("Assignee").EditType(KanbanEditingType.Dropdown).Add();
e.Field("Estimate").EditType(KanbanEditingType.Numeric).NumericEditOptions(n
ew EditorProperties() { DecimalPlaces = 2 }).ValidationRules(rule => {
rule.AddRule("range", "[0,1000]"); }).Add();
e.Field("Summary").EditType(KanbanEditingType.TextArea).ValidationRules(rule
=> { rule.AddRule("required", true); }).Add();
}).EditMode(KanbanEditMode.Dialog);
})
.AllowSearching(true)
.FilterSettings(filter =>
{
filter.Text("Janet Issues").Query("new ej.Query().where('Assignee', 'equal',
'Janet')").Description("Displays issues which matches the assignee as
'Janet").Add();
filter.Text("Testing Issues").Query("new ej.Query().where('Status', 'equal',
'Testing')").Description("Display the issues of 'Testing'").Add();
})
.Fields(field =>
{
field.Color("Type")
.PrimaryKey("Id")
.Content("Summary")
.ImageUrl("Image");
})
)
```

C#

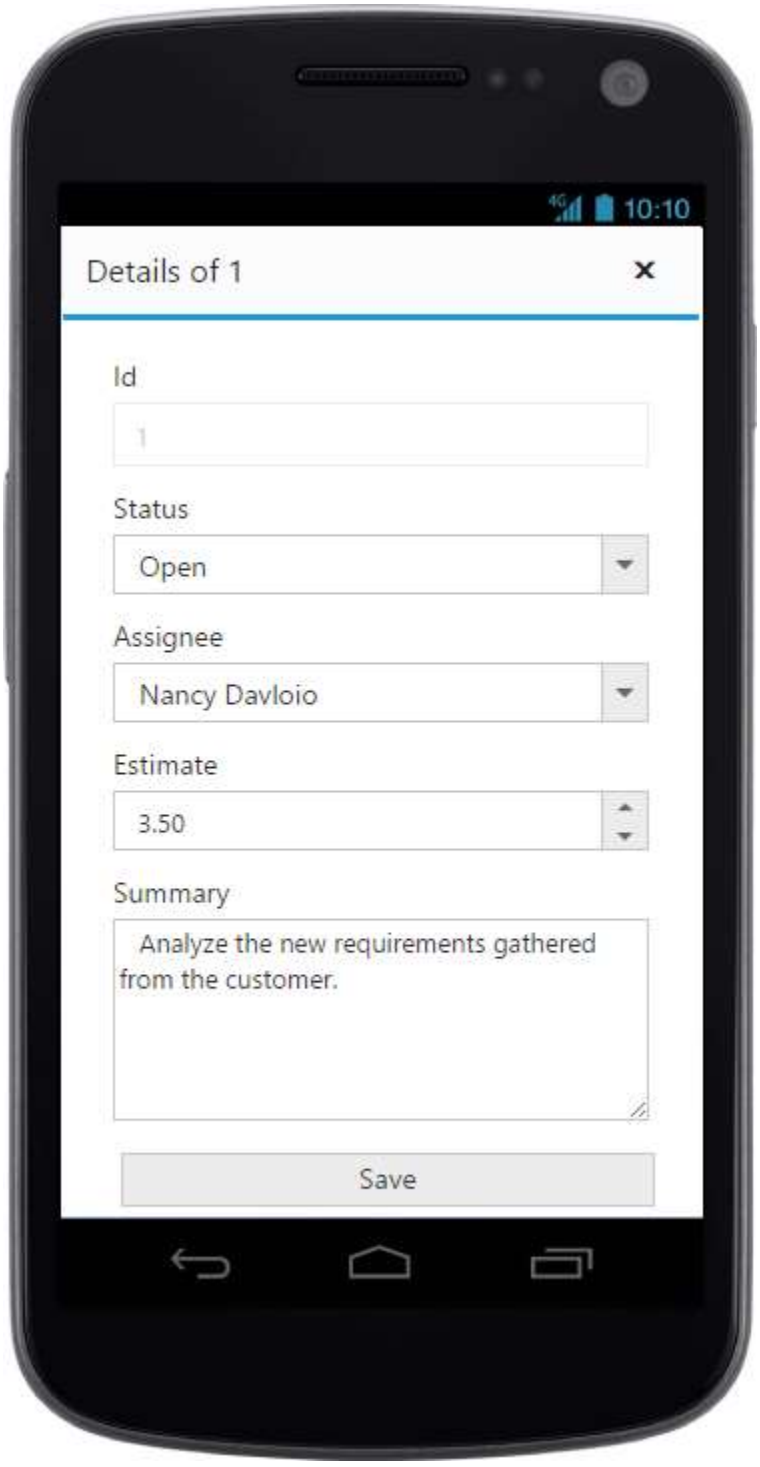
```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



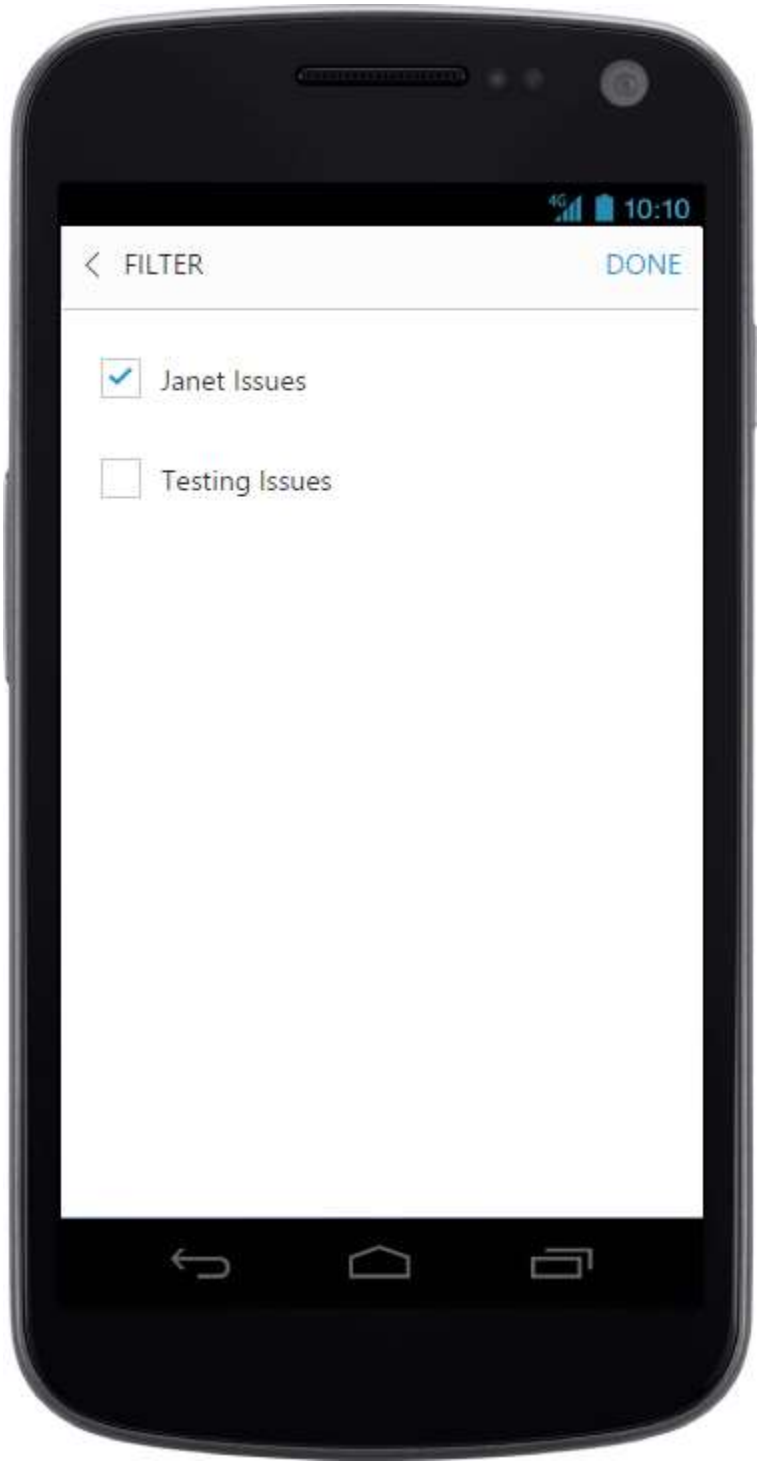
Waring: IE8 and IE9 does not support responsive kanban. `ej.responsive.css` should be referred to display Responsive Kanban.

Copyright © 2001 - 2020 Syncfusion Inc.



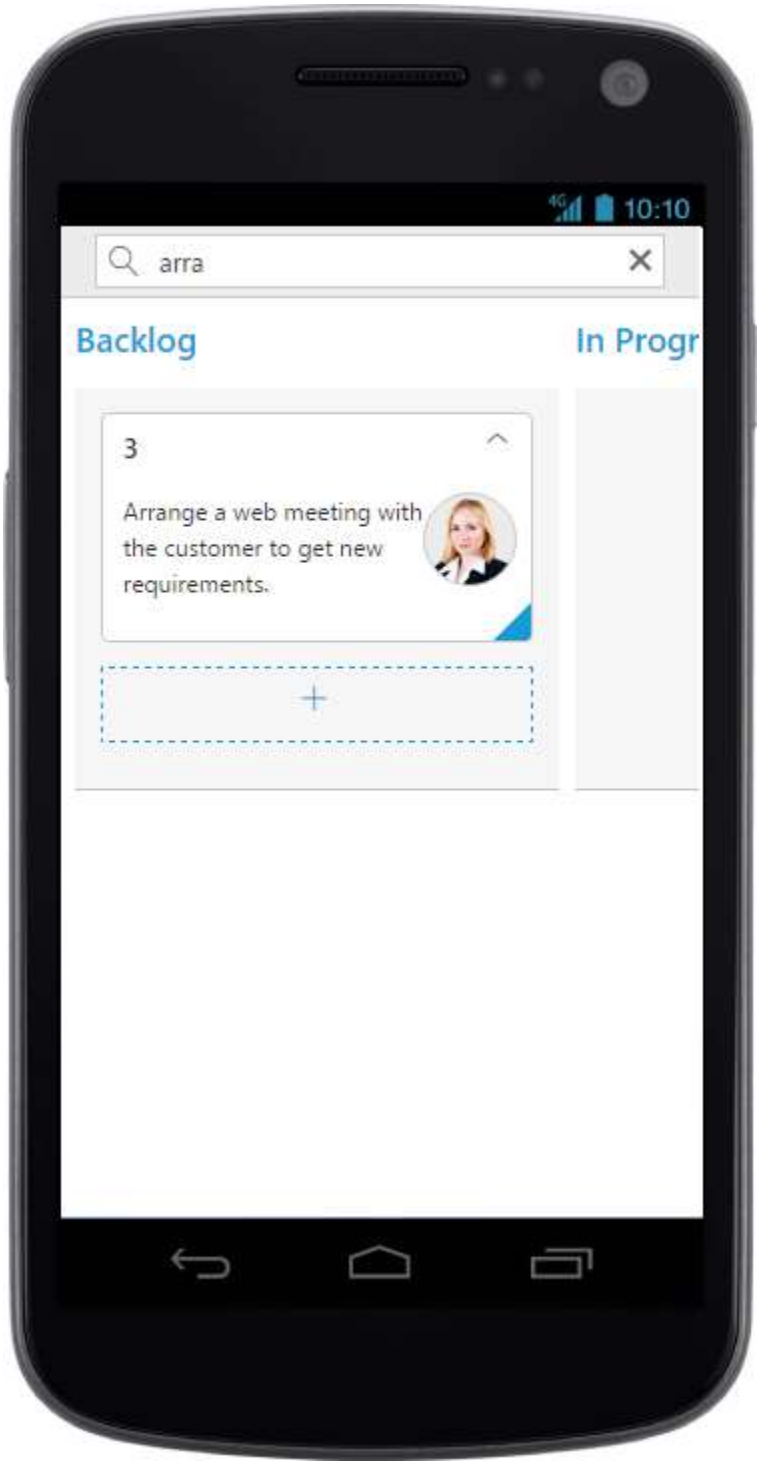
CRUD in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



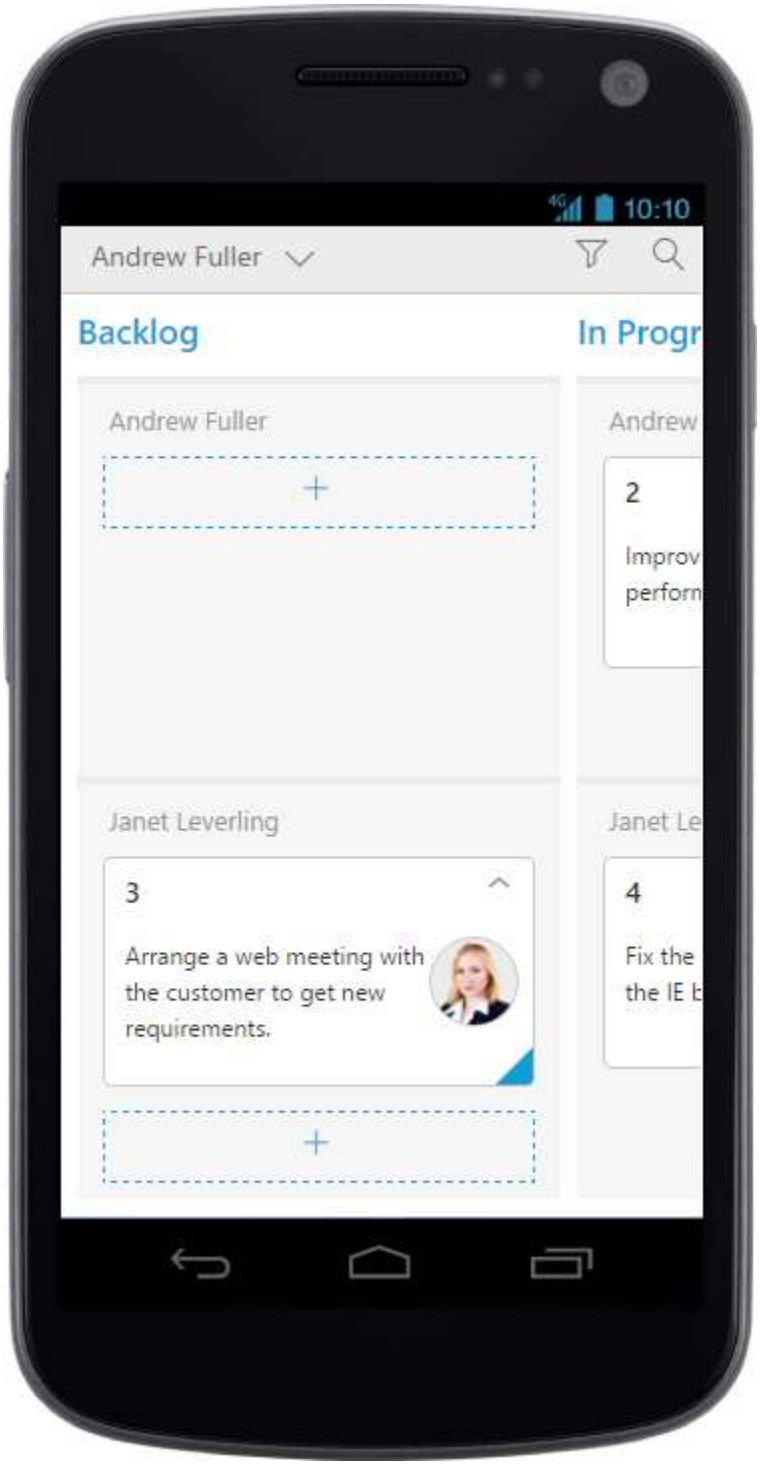
Filtering in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.

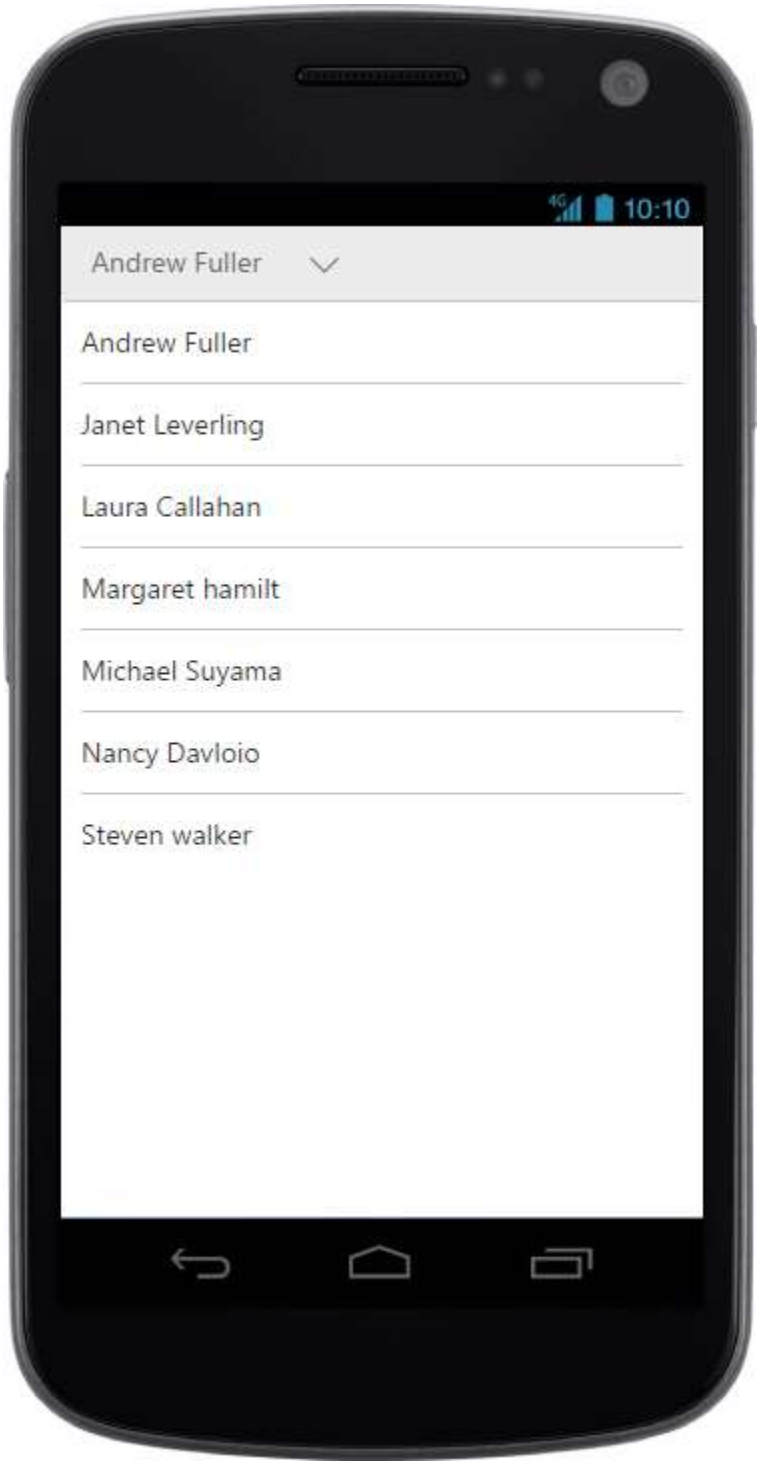


Searching in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



Copyright © 2001 - 2020 Syncfusion Inc.



Kanban with Swim-lane

Width

By default, the Kanban is adaptable to its parent container. It can adjust its width of columns based on parent container width. You can also assign width of Columns in percentage.

The following code example describes the above behavior.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.IsResponsive(true)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Width(10).Add();
col.HeaderText("In Progress").Key("InProgress").Width(10).Add();
col.HeaderText("Done").Key("Close").Width(10).Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

Note: AllowScrolling should be false while defining width in percentage.

Min Width

Min Width is used to maintain minimum width for the Kanban. If the Kanban width is less than MinWidth then the scrollbar will be displayed to maintain minimum width.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.IsResponsive(true)
.MinWidth(700)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Width(120).Add();
col.HeaderText("In Progress").Key("InProgress").Width(110).Add();
col.HeaderText("Done").Key("Close").Width(110).Add();
```

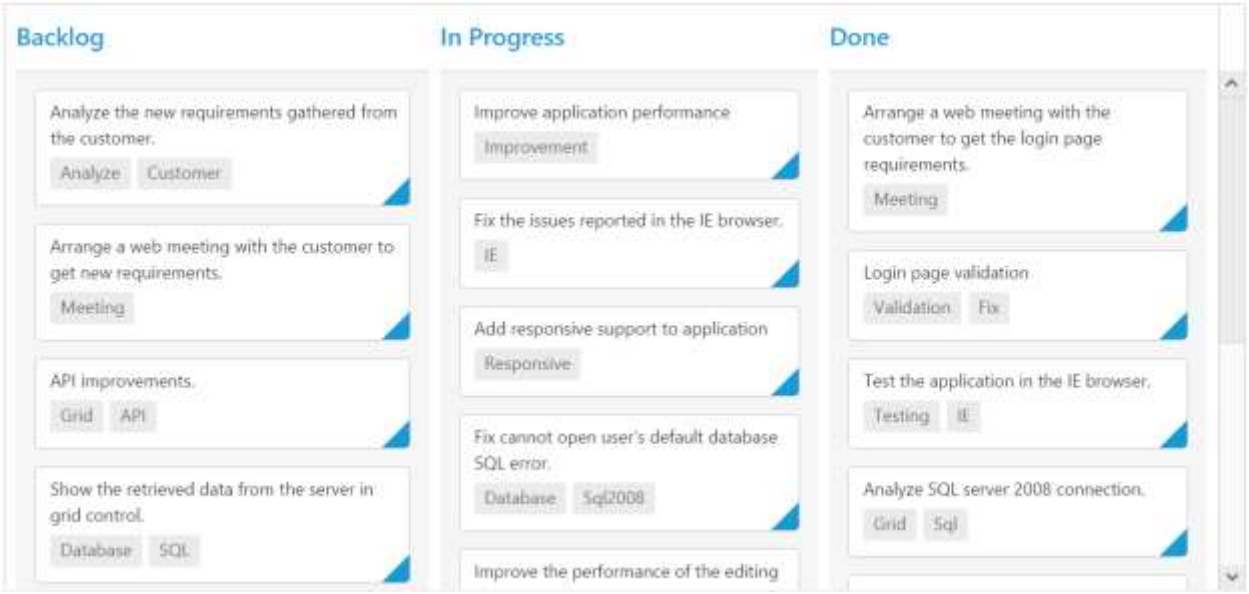
Copyright © 2001 - 2020 Syncfusion Inc.

```
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Stacked Headers

The stacked headers help you to group the logical columns in Kanban. It can be shown by setting ShowStackedHeader as true and by defining StackedHeaderRows.

Adding Stacked header columns

To stack columns in stacked header, you need to define `Column` property in `StackedHeaderColumns` with field names of visible columns.

The following code example describes the above behavior.

RAZOR

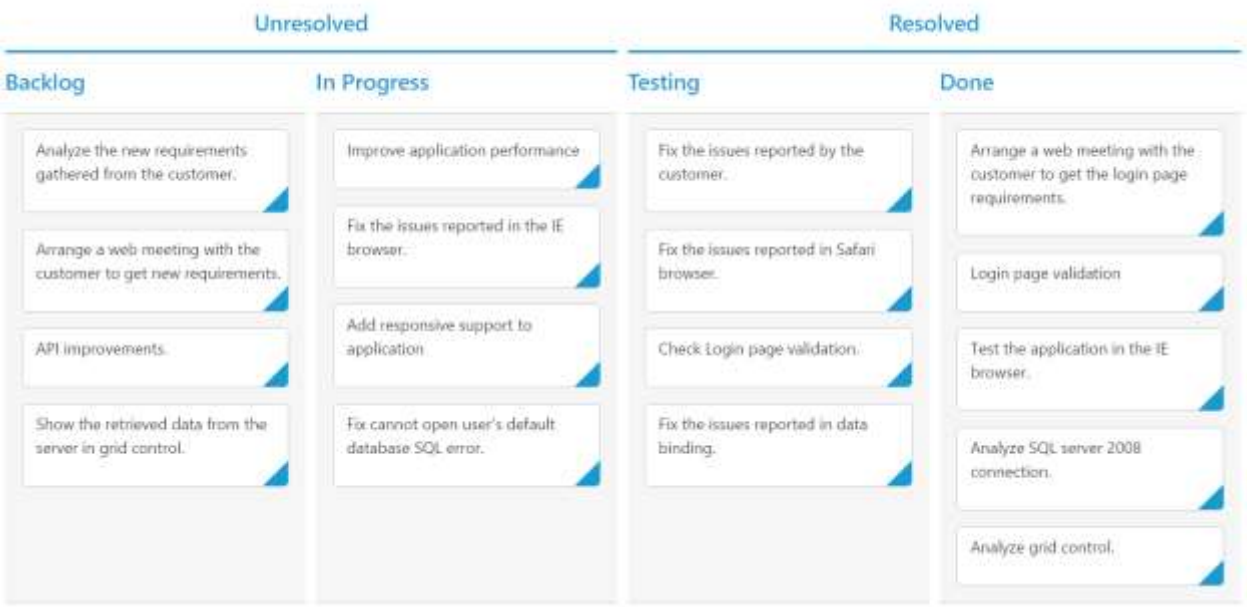
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Testing").Key("Testing").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
.StackedHeaderRows(rows =>
{
rows.StackedHeaderColumns(Columns =>
{
Columns.HeaderText("Unresolved").Column("Backlog,Validated,In
Progress").Add();
Columns.HeaderText("Resolved").Column("Testing,Done").Add();
}).Add();
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Context Menu

Context menu is used to improve user action with Kanban using popup menu. It can be shown by defining ContextMenuSettings.Enable as true. Context menu has option to add default items in ContextMenuSettings.MenuItems and customized items in ContextMenuSettings.CustomMenuItems.

Default Context Menu items

Please find the below table for default context menu items and its actions.

Section	Context menu items	Action
Header	Hide Column	Hide the current column
	Visible Columns	Show the column if already hidden
Content	Add Card	Start Add new card
Card	Edit Card	Start Edit in current card
	Delete Card	Delete the current card
	Top of Row	Move the card to Top of Row
	Bottom of Row	Move the card to Bottom of Row
	Move Up	Move the card in Up direction
	Move Down	Move the card in Down direction
	Move Left	Move the card in Left direction
	Move Right	Move the card in Right direction
	Move to Swimlane	Move the card to Swim lane which is chosen from given list

Copyright © 2001 - 2020 Syncfusion Inc.

	Print Card	Print the specific card

The following code example describes the above behavior.

RAZOR

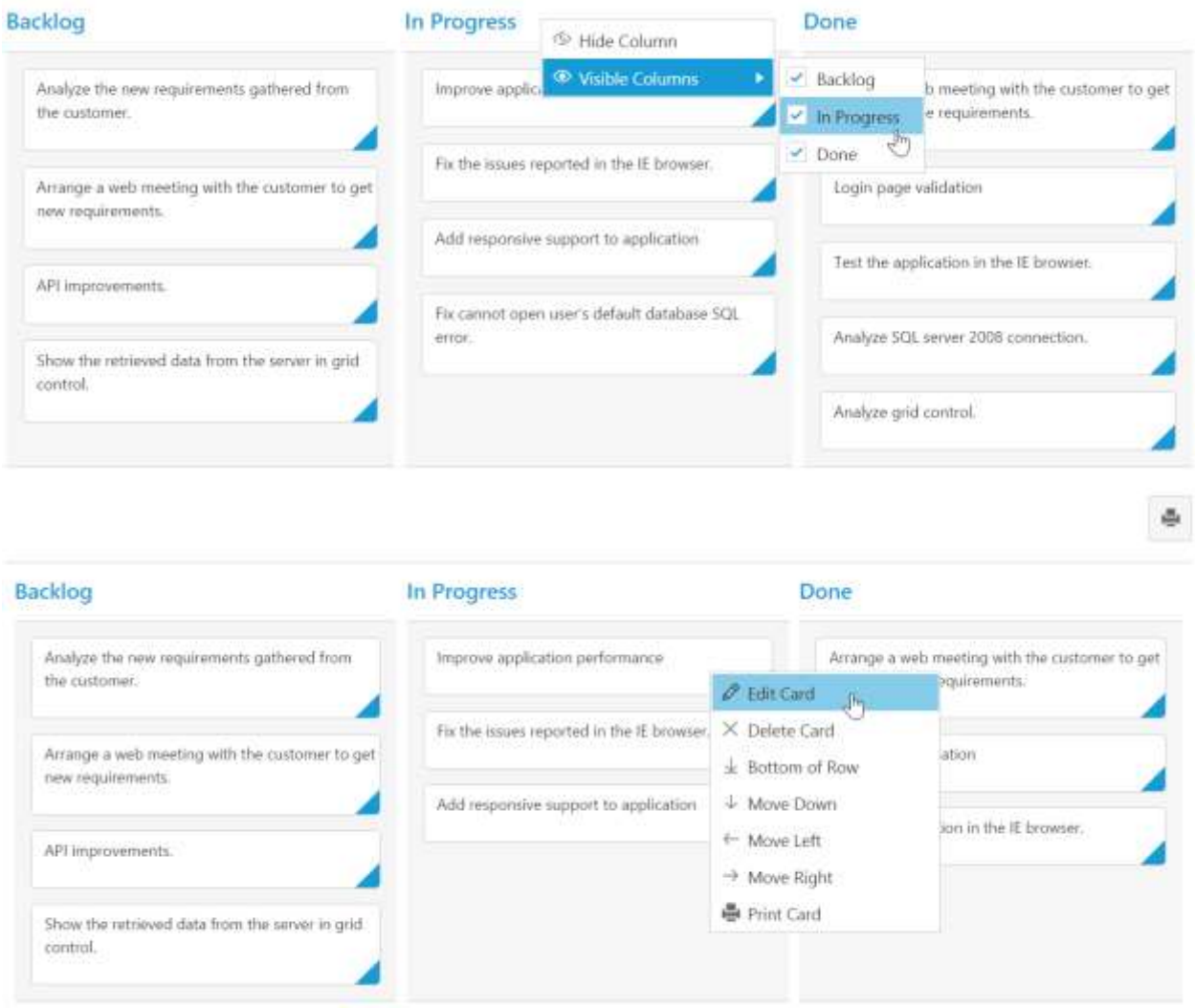
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
.ContextMenuSettings(menu => menu.Enable(true))
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Custom Context Menu

Custom context menu is used to create your own menu item and its action. To add customized context menu items, you need to use `ContextMenuSettings.CustomMenuItems` property and to bind required actions for this, use `ContextClick` event.

The following code example describes the above behavior.

RAZOR

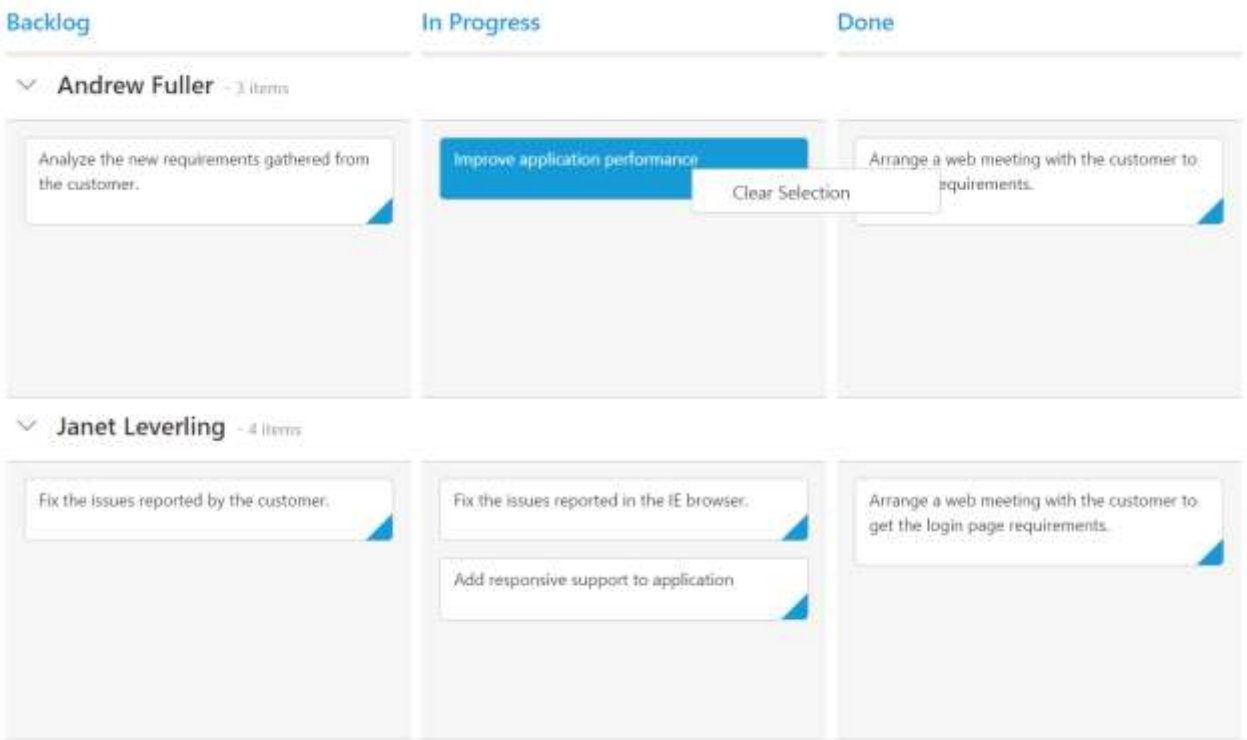
```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
```

```
.Tag("Tags")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
.ContextMenuSettings(menu => menu.Enable(true) .MenuItems(item=>
item.Reset()).CustomMenuItems(custom => custom.Text("Clear
Selection").Target(ContextTarget.All).Add()))
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Sub Context Menu

Sub context menu is used to add customized sub menu to the custom context menu item. To add a sub context menu, you need to use ContextMenuSettings.subMenu property and to bind required actions for this, use ContextClick event.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.ClientSideEvents(eve => eve.ContextClick("click"))
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
.ContextMenuSettings(menu =>
menu.Enable(true).MenuItems(item=>item.Add()).CustomMenuItems(custom =>
{
custom.Text("Clear Selection").Target(ContextTarget.All).Add();
custom.Text("Move to
Column").Target(ContextTarget.All).Template("#template").Add();
}
))
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

HTML

```
<ul id="template">
```

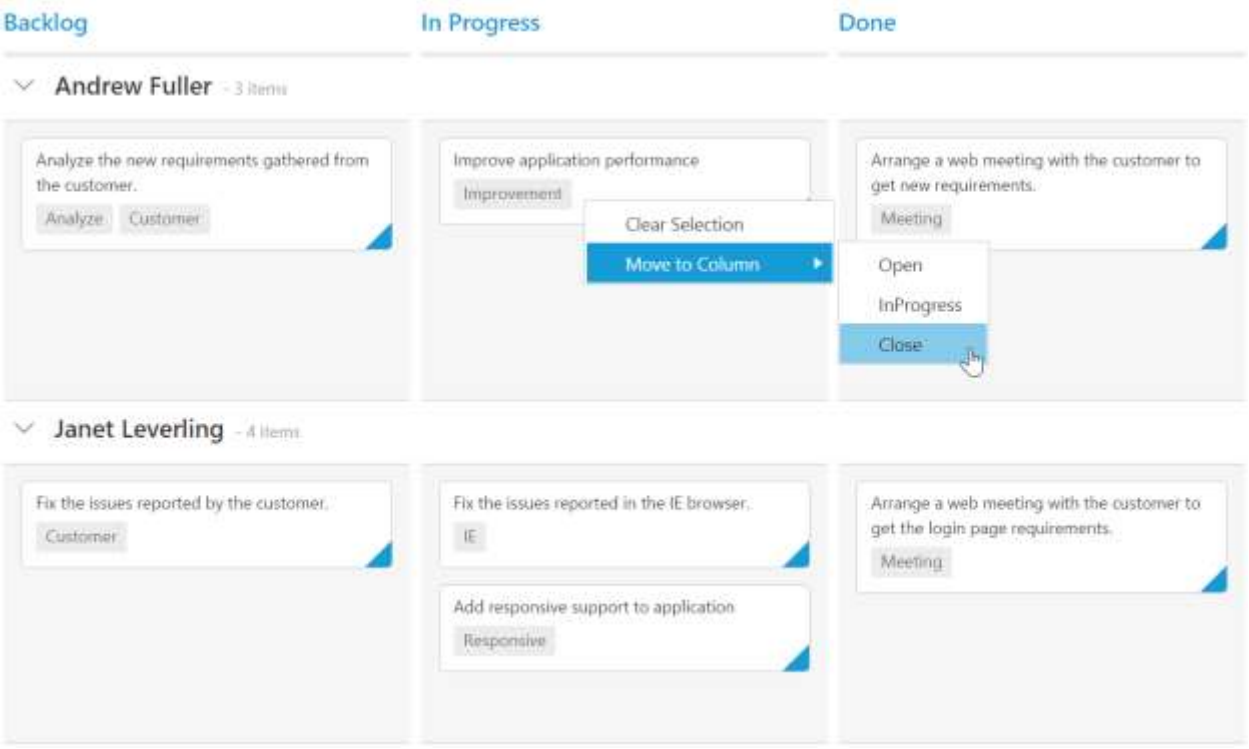
Copyright © 2001 - 2020 Syncfusion Inc.

```
<li><a>Open</a></li>
<li><a>InProgress</a></li>
<li><a>Close</a></li>
</ul>
```

JAVASCRIPT

```
function click(args)
{
if (args.text == "Clear Selection")
this.clearSelection();
}
```

The following output is displayed as a result of the above code example.



Print

Set 'AllowPrinting' as true, to enable Print icon in the Kanban toolbar. You can use 'print()' method from Kanban instance to print the Kanban.

The following code example describes the above behavior.

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Add();
col.HeaderText("Done").Key("Close").Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

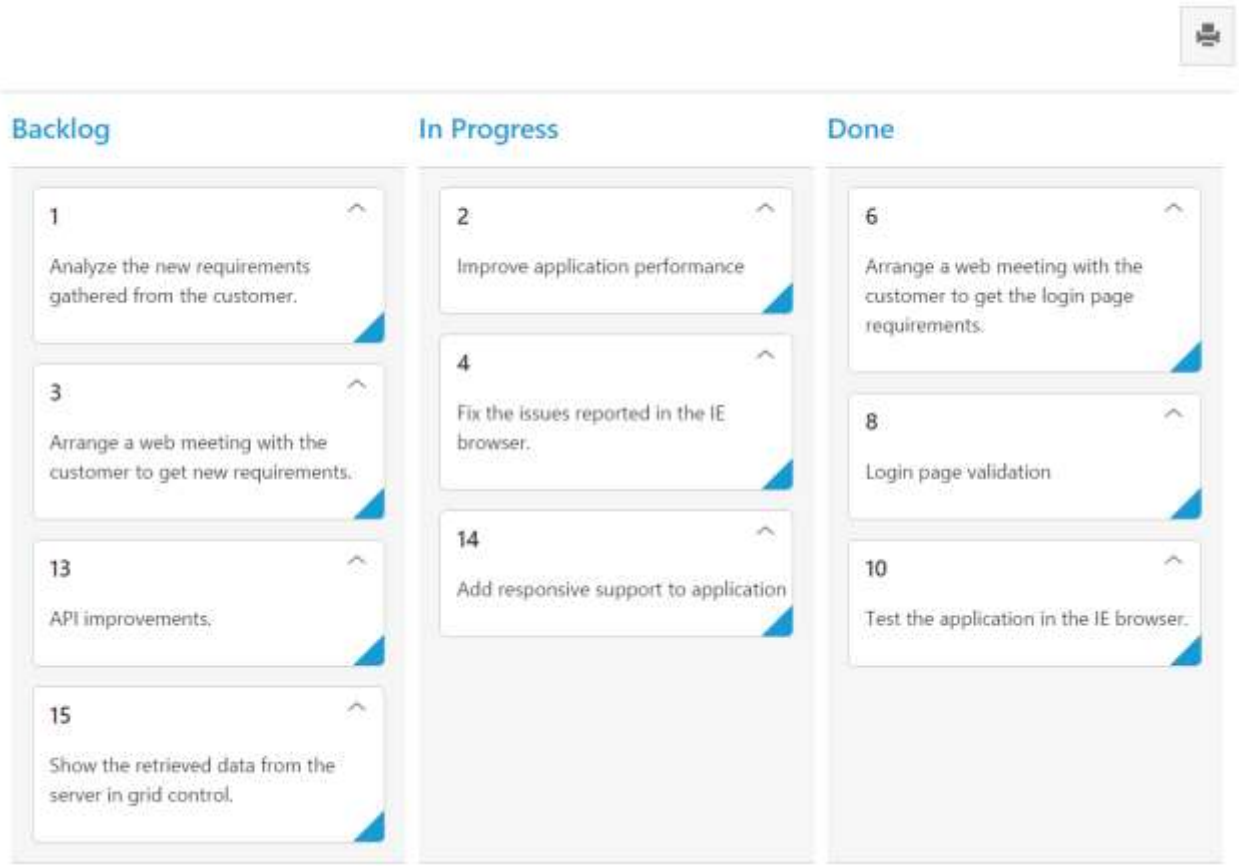
```
.KeyField("Status")
.AllowTitle(true)
.AllowPrinting(true)
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Localization

All text in Kanban can be localized using ej.Kanban.Locale object. Please find the table with list of properties and its value in locale object.

Locale key words	Text
EmptyCard	No cards to display
SaveButton	Save
CancelButton	Cancel
EditFormTitle	Details of
AddFormTitle	Add New Card
SwimlaneCaptionFormat	"- {{:count}}{{if count == 1 }} item {{else}} items {{/if}}"
FilterSettings	Filters:
Min	Min
Max	Max
FilterOfText	Of
Cards	Cards

Copyright © 2001 - 2020 Syncfusion Inc.

ItemsCount	Items Count :
Unassigned	Unassigned
AddCard	Add Card
EditCard	Edit Card
DeleteCard	Delete Card
TopofRow	Top of Row
BottomofRow	Bottom of Row
MoveUp	Move Up
MoveDown	Move Down
MoveLeft	Move Left
MoveRight	Move Right
MovetoSwimlane	Move to Swimlane
HideColumn	Hide Column
VisibleColumns	Visible Columns
PrintCard	Print Card
Search	Search

The following code example describes the above behavior.

JAVASCRIPT

```
ej.Kanban.Locale["de-DE"] = {
EmptyCard: "Keine Karten angezeigt werden",
SaveButton: "Speichern",
CancelButton: "stornieren",
EditFormTitle: "Details von ",
AddFormTitle: "Neue Karte hinzufügen",
SwimlaneCaptionFormat: "- {{:count}}{{if count == 1 }} Artikel {{else}}
Artikel {{/if}}",
FilterSettings: "Filter:",
FilterOfText: "Von",
Max: "Max.",
Min: "Min.",
Cards: "Karten",
ItemsCount:"Artikel Graf :",
Unassigned:"Nicht zugewiesen",
};
```

RAZOR

```
@(Html.EJ().Kanban("Kanban")
```

Copyright © 2001 - 2020 Syncfusion Inc.

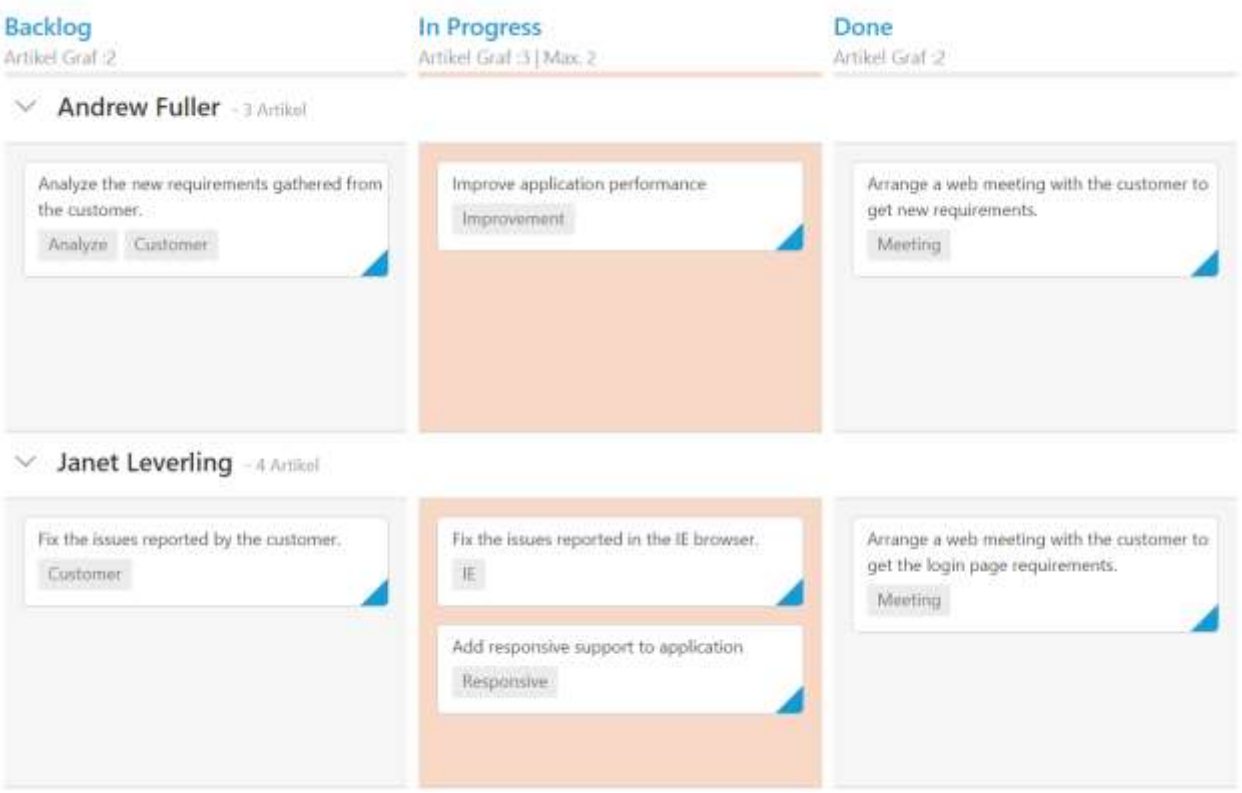
```
.DataSource((IEnumerable<object>)ViewBag.datasource)
.Locale("de-DE")
.EnableTotalCount(true)
.Columns(col =>
{
col.HeaderText("Backlog").Key("Open").Add();
col.HeaderText("In Progress").Key("InProgress").Constraints(con => {
con.Max(2); }).Add();
col.HeaderText("Done").Key("Close").Add();
})
.KeyField("Status")
.Fields(field =>
{
field.Content("Summary")
.Tag("Tags")
.SwimlaneKey("Assignee")
.PrimaryKey("Id");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Right to Left (RTL)

By default, Kanban render its text and layout from left to right. To customize Kanban's direction, you can change direction from LTR to RTL by using `EnableRTL` as true.

The following code example describes the above behavior.

JAVASCRIPT

```
ej.Kanban.Locale["ar-AE"] = {
EmptyCard: "لا بطاقات لعرض",
SaveButton: "حفظ",
CancelButton: "إلغاء",
EditFormTitle: "تفاصيل ",
AddFormTitle: "إضافة بطاقة جديدة",
SwimlaneCaptionFormat: "- {{:count}}{{if count == 1 }} بند {{else}} العناصر
{{/if}}",
FilterSettings: "مرشحات:",
FilterOfText: "من",
Max: "ماكس",
Min: "دقيقة",
Cards: " بطاقات",
ItemsCount: "عد العناصر:",
Unassigned: "غير معين",
};
```

RAZOR

```
@(Html.EJ().Kanban("Kanban")
.DataSource((IEnumerable<object>)ViewBag.datasource)
```

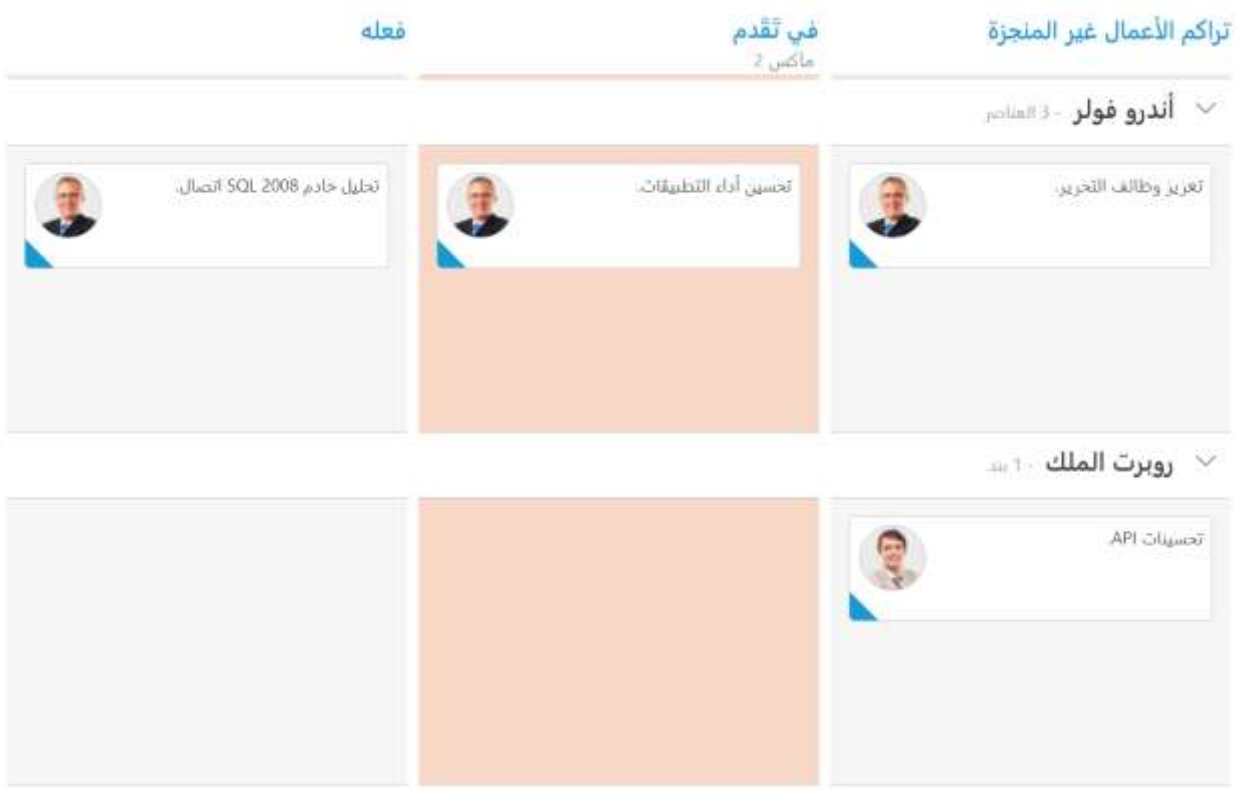
```
.Columns(col =>
{
col.HeaderText("تراكم الأعمال غير المنجزة").Key("Open").Add();
col.HeaderText("في تَقَدم").Key("InProgress").Constraints(con => { con.Max(2);
}).Add();
col.HeaderText("فعله").Key("Close").Add();
})
.AllowTitle(true)
.EnableRTL(true)
.Locale("ar-AE")
.KeyField("Status")
.Fields(field =>
{
field.Color("Type")
.Content("Summary")
.PrimaryKey("Id")
.SwimlaneKey("Assignee");
})
)
```

C#

```
namespace MVCSampleBrowser
{
public partial class KanbanController : Controller
{
//
// GET: /Kanban/
public ActionResult KanbanFeatures()
{
var DataSource = new NorthwindDataContext().Tasks.Take(30).ToList();
ViewBag.datasource = DataSource;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Styling

List of classes and its purposes

To modify Kanban appearance, you need to override default CSS of Kanban. Please find the list of CSS classes and its corresponding section in Kanban. Also you have an option to create your own custom theme for all JavaScript controls using our Theme Studio

Section	CSS class	Purpose of CSS class
Root	e-kanban	This classes are in this root element (div) of Kanban control.
	e-js	
Header	e-kanbantoolbar	This class is added at div element of Kanban toolbar.
	e-kanbanheader	This is class is added in the root element of header element. In this class, You can override thin line between header and content of Kanban.
	e-table	This class is added at 'table' of Kanban header. This CSS class makes table width as 100 %.
	e-columnheader	This class is added at 'tr' of Kanban header.
	e-headercell	This class is added in 'th' element of Kanban header. You can override background color of header and border color
	e-headercelldiv	This class is add in div which present 'th' element in header. You recommend you to use e-headercelldiv to override skeleton of header.

Copyright © 2001 - 2020 Syncfusion Inc.

Body	e-kanbancontent	This class is added at root of body content. This is to override background color of body.
	e-table	This class is added to table of content. This CSS class makes table width as 100 %.
	e-swimlanerow	This class is added to all swim lane â€˜trâ€™s in Kanban.
	e-columnrow	This class is added to all next row of swimlane â€˜trâ€™s in Kanban
	e-rowcell	This class is added to all cells in Kanban. This is to override cells appearance and styling.
	e-cardselection	This class is added to card div element of Kanban. This is override selection.
	e-hover	This class adds to card of Kanban while hover cards.

Web Accessibility

Keyboard Navigation

Supported Keyboard Interactions keys with its description are tabulated as follows

Interaction Keys	Description
Alt + j	Focus the Kanban
Insert	Insert card in Kanban
Delete	Delete card in Kanban
F2	Edit card in Kanban
Enter	Save edited or added
Esc	Cancel add or edit state
Home	Go to first card
End	Go to last card
Up arrow	Move to up card selection
Down arrow	Move to down card selection
Right arrow	Move to right card selection
Left arrow	Move to left card selection
Ctrl + UpArrow	Collapse All swim lane groups
Ctrl + DownArrow	Expand All swim lane groups
Alt + UpArrow	Collapse selected swim lane
Alt + DownArrow	Expand selected swim lane

Copyright © 2001 - 2020 Syncfusion Inc.

Alt + LeftArrow	Collapse selected column
Alt + RightArrow	Expand selected column
Shift + UpArrow	Multi Selection by Up Arrow
Shift + DownArrow	Multi Selection by Down Arrow
Shift + LeftArrow	Multi Selection by Left Arrow
Shift + RightArrow	Multi Selection by Right Arrow

How to

Render Kanban from Code behind

Kanban can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Kanban properties in the controller.

C#

```
//Namespace to get the JavaScript (Kanban) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class KanbanController: Controller
{
public ActionResult KanbanFeatures()
{
// Initialize data for Kanban
List<Tasks> Task = new List<Tasks>();
Task.Add(new Tasks(1, "Open", "Analyze the new requirements gathered from
the customer.", "Story", "Low", "Analyze,Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/1.png", 1));
Task.Add(new Tasks(2, "InProgress", "Improve application performance",
"Improvement", "Normal", "Improvement", 6, "Andrew Fuller",
"../content/images/kanban/2.png", 1));
Task.Add(new Tasks(3, "Open", "Arrange a web meeting with the customer to
get new requirements.", "Others", "Critical", "Meeting", 5.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(4, "InProgress", "Fix the issues reported in the IE
browser.", "Bug", "Release Breaker", "IE", 2.5, "Janet",
"../content/images/kanban/3.png", 2));
Task.Add(new Tasks(5, "Testing", "Fix the issues reported by the customer.",
"Bug", "Low", "Customer", 3.5, "Andrew Fuller",
"../content/images/kanban/5.png", 1));
//Initializing the kanban model
KanbanProperties kanbanObj = new KanbanProperties();
//Initializing the datasource and columns
kanbanObj.DataSource = Task;
kanbanObj.Columns.Add(new KanbanColumn() { HeaderText = "Backlog", Key = new
List<string> { "Open" } });
kanbanObj.Columns.Add(new KanbanColumn() { HeaderText = "In Progress", Key =
new List<string> { "InProgress" } });
```

```
kanbanObj.Columns.Add(new KanbanColumn() { HeaderText = "Done", Key = new
List<string> { "Close" } });
//Define the fields
kanbanObj.KeyField = "Status";
kanbanObj.Fields.Color = "Type";
kanbanObj.Fields.Content = "Summary";
kanbanObj.Fields.PrimaryKey = "Id";
//Passing kanban properties using the ViewData
ViewData["KanbanModel"] = kanbanObj;
return View();
}
}
}
```

Binding the Kanban properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Kanban("Kanban",
(Syncfusion.JavaScript.Models.KanbanProperties)ViewData["KanbanModel"]).Rend
er();
}
```

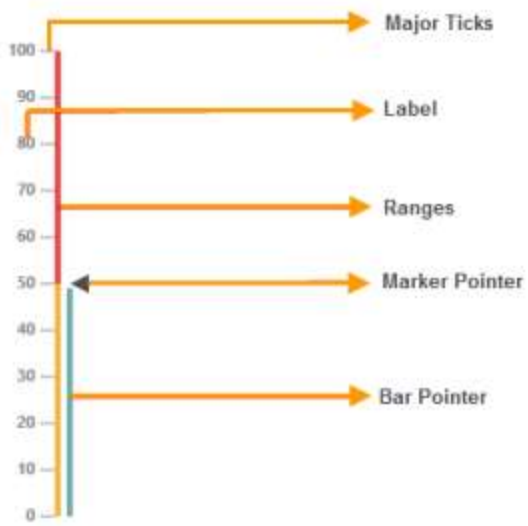
LinearGauge

Overview

The Gauge control for Essential Studio displays numerical information in the form of a scale that can be customized and oriented either vertically or horizontally. It comprises the following basic elements:

- Scales
- Pointers: bars and markers
- Ticks
- Labels
- Ranges

Copyright © 2001 - 2020 Syncfusion Inc.



Three basic scale designs available are : rectangle, rounded rectangle, and thermometer.

Key Features

- Interaction: Allows you to directly interact with pointers on a gauge.
- Indicators: Indicate the state of a gauge either active or inactive.
- Ranges: Highlight the range of values on a gauge.
- Pointers: Allows you to add multiple pointers like bar pointers and marker pointers to a gauge.
- Scale Direction: Allows you to set scales direction either horizontally or vertically.
- Animation: Supports animation effects for pointers.
- Custom Labels: Allows you to add custom label text on any part of a gauge.
- Scale Styles: Three basic styles of scales are supported: rectangle, rounded rectangle, and thermometer.

Getting Started

This section briefly explains on how to create a **Linear Gauge** control for your application.

- The ASP.NET MVC **Linear Gauge** provides support to display the **Linear Gauge** within your web page and allows you to customize it. This section encompasses the details on how to configure Linear Gauge.
- You will learn how to provide data for a **Linear Gauge** and display data in the required way. In addition, you will learn how to customize the default Linear Gauge appearance for your requirements.

Copyright © 2001 - 2020 Syncfusion Inc.



Create a Linear Gauge

ASP.NET MVC **Linear Gauge** widget basically renders with flexible APIs. You can easily create the Linear Gauge widget by the following steps.

1. First create an MVC Project and add necessary Dll's and scripts with the help of the given MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page to render **Linear Gauge**.

JS

```
@(Html.EJ().LinearGauge("thermoLinear"))
```

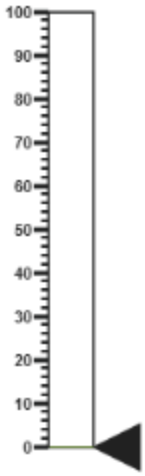
3. Add the following code example in the Controller page.

CSHARP

```
public ActionResult Default()
{
return View( );
}
```

Run the above code example to get a default **Linear Gauge** with default values as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



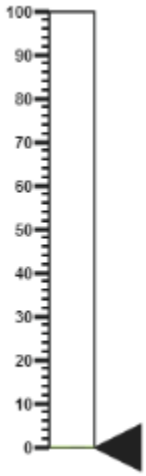
Set Height and Width values

Basic attributes of each canvas elements are height and width. You can set the height and width of the gauge using the following code example. It sets the height and width of the canvas image where the thermometer is to be rendered.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.Height(550)
.Width(500))
```

Run the above code example and you will get the following gauge as similar to default. Here height and width of the canvas are set for given values.



Set Animation option and Label Color

- You can draw the Thermometer with some Label color to display the measurement value. For example give the labelColor as “#8c8c8c”.

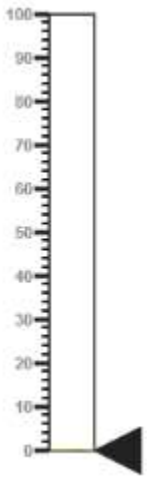
Copyright © 2001 - 2020 Syncfusion Inc.

- Set the EnableAnimation property as false to avoid animation on the pointers.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.EnableAnimation(false)
.Height(550)
.Width(500))
```

Run the above code example and you will get the following gauge as the output.



Provide Scale Values

- The scale must have the appearance of a thermometer. By giving ScaleType as Thermometer, you can render a thermometer design.
- Minimum temperature can go up to -10 and maximum temperature can rise up to 110, so you can give minimum scale value as -10 and maximum value as 110.
- Set the location values such as vertical and horizontal position of the thermometer and give the thermometer height as Length.
- You can give the Minor Interval value as 5 to get the exact temperature on the patient.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
s.Width(20)
.Type(ScaleType.Thermometer)
.BackgroundColor("transparent")
.Minimum(-10)
.Maximum(110).MinorIntervalValue(5)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Length(335)
. Border(border=>border.Width(1))
.ShowCustomLabels(true)
.Position(loc => { loc.X(50).Y(18); })
//Add the pointers customization code here
//Add the labels customization code here
//Add the ticks customization code here
//Add the Custom labels customization code here
.Add();
}))
```

Run the above code example and you will get the following gauge as the output.



Add Pointers

In **Linear Gauge** the two types of pointers available are: Marker pointer and Bar pointer.

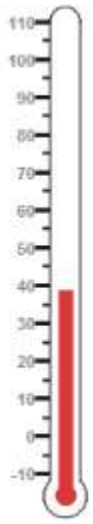
- **Marker pointer** is displayed as a pointer device that shows the actual values. But for your thermometer there is no need for the marker pointer. So you can hide the marker pointer by giving opacity as 0.
- **Bar pointer** acts as the mercury metal that shows the exact temperature of the patient. Set some of the basic properties of the Bar pointer such as Width, BarPointerDistanceFromScale, BarPointerValue and BarPointerBackgroundColor.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
//Add the pointers customization code here
.MarkerPointers(mp => { mp.MarkerOpacity(0).Add(); })
```

```
.BarPointers(bar => {
bar.Width(10)
.BarPointerDistanceFromScale(0)
.BarPointerValue(37)
.BarPointerBackgroundColor("#DB3738").Add(); })
//Add the labels customization code here
//Add the ticks customization code here
//Add the Custom labels customization code here
.Add();
}))
```

Run the above code example to get the following gauge as your output.



Add Label Customization

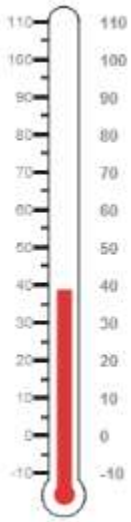
- For thermometer, you can display the label value on two sides, to get temperature in different scales. For that you can add two label values in an array.
- To display the value around the scales, labels are used. You can customize the label placement, font (including its style and family) and its distance from scale.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
//Add the pointers customization code here
//Add the labels customization code here
.Labels(label =>
{
label.Placement(LabelPlacement.Near)
.Font(font => {
```

```
font.Size("10px")
.FontFamily("Segoe UI")
.FontStyle("normal");
}).Add();
label.Placement(LabelPlacement.Far)
.DistanceFromScale(distance=>distance.X(13)).Add();
})
//Add the ticks customization code here
//Add the Custom labels customization code here
.Add();
}))
```

Run the above code example and you will get the following gauge as output.



Add Tick Details

- Tick style has two values called major interval and minor interval. You can set major ticks widthand height greater than Minor ticks. And you can give TickColor, for better visibility in light backgrounds.
- Here four tick details are used for both sides having minor and major ticks. To display the tick value add the following code example.

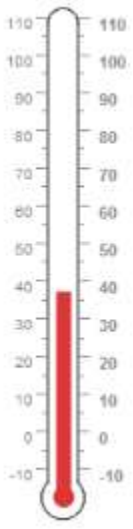
JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
//Add the pointers customization code here
//Add the labels customization code here
//Add the ticks customization code here
.Ticks(tic =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
tic.Type(TickType.MajorInterval)
.Height(8)
.Width(1)
.Color("#8c8c8c")
.DistanceFromScale(distance=>distance.Y(4)).Add();
tic.Type(TickType.MinorInterval)
.Height(4)
.Width(1)
.Color("#8c8c8c")
.DistanceFromScale(distance=>distance.Y(4)).Add();
tic.Type(TickType.MajorInterval)
.Placement(TickPlacement.Far)
.Height(8)
.Width(1)
.Color("#8c8c8c")
.DistanceFromScale(distance=>distance.Y(4)).Add();
tic.Type(TickType.MinorInterval)
.Placement(TickPlacement.Far)
.Height(4)
.Width(1)
.Color("#8c8c8c")
.DistanceFromScale(distance=>distance.Y(4)).Add();
})
//Add the Custom labels customization code here
.Add();
}))
```

Run the above code example and you will get the following gauge as output.



Add Custom Label Details

- **Custom labels** are used to specify the texts in the gauge.
- It can be customized through various properties.
- In order to show custom labels, change the showIndicators property to **true**.

Copyright © 2001 - 2020 Syncfusion Inc.

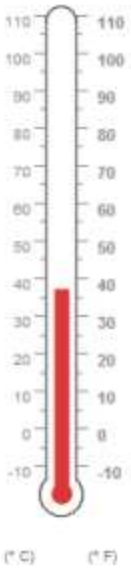
- Here you can use custom text to display three range descriptions.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
//Add the pointers customization code here
//Add the labels customization code here
//Add the ticks customization code here
//Add the Custom labels customization code here
.CustomLabels(label => {
label.Value("(° C)")
.Position(loc => loc.X(44).Y(78))
.Font(f => {
f.FontStyle("bold")
.FontFamily("Segoe UI")
.Size("12px");
})
.Color("#666666").Add();
label.Value("(° F)")
.Position(loc => loc.X(56).Y(78))
.Font(f => {
f.FontStyle("bold")
.FontFamily("Segoe UI")
.Size("12px");
})
.Color("#666666").Add();
label.Position(loc => loc.X(51).Y(90))
.Font(f => {
f.FontStyle("bold")
.FontFamily("Segoe UI")
.Size("13px");
})
.Color("#666666").Add();
})
.Add();
}))
```

Run the above code example to get the following gauge as output.

Copyright © 2001 - 2020 Syncfusion Inc.



Change Scale from Degree to Fahrenheit

Add the function that converts the temperature in degree to Fahrenheit in the label, with an index value of 1.

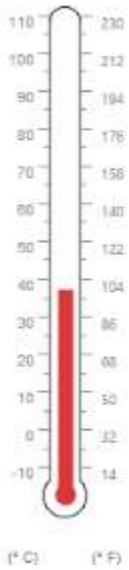
JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.ClientSideEvents(event=>event.DrawLabels("DrawLabel"))
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
s.Width(20)
.Type(ScaleTypes.Thermometer)
.BackgroundColor("transparent")
.Minimum(-10)
.Maximum(110).MinorIntervalValue(5)
.Length(335)
. Border(border=>border.Width(1))
.ShowCustomLabel(true)
.Position(loc => { loc.X(50).Y(18); }).Add();
//Add the pointers customization code here
//Add the labels customization code here
//Add the ticks customization code here
//Add the Custom labels customization code here
}))
<script>
function DrawLabel(args) {
if (args.label.index == 1) {
args.style.textValue = (args.label.value * (9 / 5)) + 32;
args.style.font = "Normal 10px Segoe UI";
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

Run the above code example and you will get the following gauge as output.



Add Custom Label for Current Value

Add the function that displays the current temperature value in the custom label.

JS

```
@(Html.EJ().LinearGauge("thermoLinear")
.LabelColor("#8c8c8c")
.ClientSideEvents(event=>event. DrawCustomLabel("DrawCustomLabel"))
.EnableAnimation(false)
.Height(550)
.Width(500)
.Scales(s =>
{
s.Width(20)
.Type(ScaleType.Thermometer)
.BackgroundColor("transparent")
.Minimum(-10)
.Maximum(110).MinorIntervalValue(5)
.Length(335)
.Border(border=>border.Width(1))
.ShowCustomLabel(true)
.Position(loc => { loc.X(50).Y(18); }).Add();
//Add the pointers customization code here
//Add the labels customization code here
//Add the ticks customization code here
//Add the Custom labels customization code here
}))
<script>
function DrawCustomLabel(args) {
if (args.customLabelIndex == 2) {
var temp = args.scaleElement.barPointers[0].value;
```

```
var value = (temp * (9 / 5)) + 32;
if (temp == -10) {
args.style.textValue = "Very Cold Weather" + "(" + value.toFixed(1) + "°
F)";
}
else if ((temp > -10 && temp < 0) || (temp > 0 && temp < 15)) {
args.style.textValue = "Cool Weather" + " (" + value.toFixed(1) + "° F)";
}
else if (temp == 0) {
args.style.textValue = "Freezing point of Water" + " (" + value.toFixed(1) +
"° F)";
}
else if (temp >= 15 && temp < 30) {
args.style.textValue = "Room Temperature" + " (" + value.toFixed(1) + "°
F)";
}
else if (temp == 30) {
args.style.textValue = "Beach Weather" + " (" + value.toFixed(1) + "° F)";
}
else if (temp == 37) {
args.style.textValue = "Body Temperature" + " (" + value.toFixed(1) + "°
F)";
}
else if (temp == 40) {
args.style.textValue = "Hot Bath Temperature" + " (" + value.toFixed(1) + "°
F)";
}
else if (temp > 40 && temp < 100) {
args.style.textValue = "Very Hot Temperature" + " (" + value.toFixed(1) + "°
F)";
}
else if (temp == 100) {
args.style.textValue = "Boiling point of Water" + " (" + value.toFixed(1) +
"° F)";
}
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Interaction and Animation

- **Linear Gauge** control contains **Interaction** feature. You can use this interaction feature to change the pointer values manually either by clicking or dragging the pointer over the Gauge. It dynamically changes the value of pointer when dragged. To Enable/Disable the user interaction you can use the `readOnly` Boolean property. The user interaction option is enabled when you set `readOnly` property as false. By default it holds the true value.
- **Linear Gauge** contains another attractive concept called **Animation**. The animation option enables the movement of the pointer from the minimum value to the current value. You can use animation option to change the pointer value dynamically. You can enable/ disable it using `enableAnimation` property. To enable animation set **enableAnimation** to 'true'.
- By default it holds the true value. You can control the speed of the pointer during animating using `animationSpeed`. It is a numerical value that holds the time in milliseconds. That is when setting value is 1000, it is considered as 1 second.

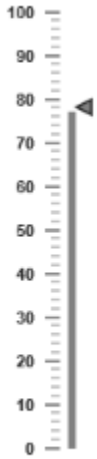
JS

```
@(Html.EJ().LinearGauge("LinearGauge1").Width(300).Height(500).Minimum(10).M
aximum(110)
.Value(78)
.EnableAnimation(true)
.AnimationSpeed(1000)
.ReadOnly(false)
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true)
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.MarkerBackgroundColor("Grey").MarkerDistanceFromScale(20).Add(); })
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(5).BarPointerDistanceFromScale(15)
.BarPointerBackgroundColor("Grey").Add();
})
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Scales

Scales are the basic functional block of the **Linear Gauge**. You can improve the appearance of scales by customizing it. The functional blocks of **Linear Gauge** are

- Marker Pointers
- Bar Pointer
- Labels
- Custom Labels
- Indicators
- Ticks
- Ranges

Copyright © 2001 - 2020 Syncfusion Inc.

Adding scale collection

Scale is the basic element of **Linear Gauge**. Scale collection is directly added to the gauge object. Refer the following code example to add scale collection in **Gauge** control.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(8).Position(position => position.X(20).Y(50))
.BackgroundColor("Grey")
.Border(border => border.Color("Grey").Width(0))
.ShowBarPointers(false).ShowMarkerPointers(true)
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance =>
distance.X(50).Y(0)).Add(); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{
mp.Width(20).Length(10).Type(MarkerType.Pentagon)
.Placement(PointerPlacement.Near)
.MarkerBackgroundColor("#FE8282")
.MarkerDistanceFromScale(20).Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(30).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(30).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Scale Customization

Colors and Border

- The **Scale** border is modified with **border** object. It has two border property, color and width are used to customize the border color of the scale and border width of the scale. Setting the background color improves the look and feel of the **Linear Gauge**. You can customize the background color of the scale using backgroundColor.
- Scales are used to enable or disable various properties such as showRanges, showIndicators, showCustomLabels, showLabels, showTicks, showBarPointers and showMarkerPointers. Enable/disable is done by setting the property into two states either "**true**" or "**false**". You can adjust the Opacity of the scale with opacity property.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(8).Position(position => position.X(20).Y(50))
.BackgroundColor("#FE8282")
.Border(border => border.Color("Red").Width(1))
.Opacity(0.5).ShadowOffset(10).Type(ScaleType.RoundedRectangle)
.ShowBarPointers(false).ShowMarkerPointers(true)
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance =>
distance.X(50).Y(0)).Add(); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{
mp.Width(20).Length(10).Type(MarkerType.Pentagon)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Placement(PointerPlacement.Near)
.MarkerBackgroundColor("#C9E1E5")
.MarkerDistanceFromScale(20).Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(30).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(30).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Appearance

- You can improve the appearance of **Linear Gauge** using various properties. You can set the interval values for the scale with major interval value and minor interval value properties and maximum and minimum value by minimum and maximum property. The width property is used to set the scale bar width.
- You can also adjust the Opacity of the scale with opacity property. The value for opacity lies between 0 and 1.**Linear Gauge** contains two scale directions, clockwise and counter clockwise. It can be set with direction property.

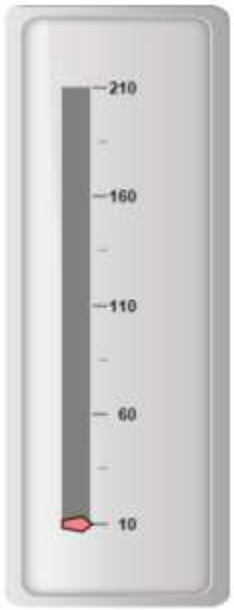
JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Adding scale collection
.Scales(scale => {
scale.Width(18).Minimum(10)
.Maximum(210)
.MinorIntervalValue(25)
.MajorIntervalValue(50)
.Direction(Directions.CounterClockwise)
.Position(position => position.X(20).Y(50))
.BackgroundColor("Grey")
.Border(border => border.Color("Grey").Width(1))
.ShowBarPointers(false).ShowMarkerPointers(true)
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance =>
distance.X(50).Y(0)).Add(); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{
mp.Width(20).Length(10).Type(MarkerType.Pentagon)
.Placement(PointerPlacement.Near)
.MarkerBackgroundColor("#FE8282")
.MarkerDistanceFromScale(20).Add();
})
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(30).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(30).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Scale Types

Scale Type is an element which decides the appearance of the gauge. **Linear Gauge** contains three scale types such as,

- Rectangle
- Rounded Rectangle
- Thermometer

Rectangle

For rectangular scale type, the scale renders with rectangular structure. Refer the following code example.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(18).Length(300).Type(ScaleType.Rectangle)
.Position(position => position.X(54).Y(50))
.BackgroundColor("#C0B08E")
.Border(border => border.Color("#C0B08E").Width(1))
.ShowMarkerPointers(false)
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0))
.Placement(TickPlacement.Far).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0)).Height(6)
.Placement(TickPlacement.Far).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Rounded Rectangle

For rounded rectangular scale type, the scale renders as rectangular structure but with constant radius rounded corner. Refer the following code example.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(8).Type(ScaleType.RoundedRectangle)
.Direction(Directions.Clockwise)
.Position(position => position.X(60).Y(50))
.BackgroundColor("#206BA4")
.Border(border => border.Color("#206BA4").Width(1))
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance => distance.X(-
20).Y(0)).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0))
.Placement(TickPlacement.Far).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0)).Height(6)
.Placement(TickPlacement.Far).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Thermometer

For thermometer scale type, the scale renders as thermometer structure with rounded bottom. Refer the following code example.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding Frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(18).Length(300).Type(ScaleType.Thermometer)
.Position(position => position.X(54).Y(50))
.BackgroundColor("#C0B08E")
.Border(border => border.Color("#C0B08E").Width(1))
.ShowMarkerPointers(false).ShowBarPointers(false)
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0))
.Placement(TickPlacement.Far).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0)).Height(6)
.Placement(TickPlacement.Far).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding multiple scales

You can set multiple scales for a single **Linear Gauge** control by using an array of scale objects. Each scale object is independent of each other. Refer the following code example to add multiple scale collection.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
//Adding Frame object
.Frame(fr=>fr.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png"))
//Adding Scale collection
.Scales(scale =>
//Adding Scale 1
{
scale.Width(8)
.Position(position => position.X(15).Y(50))
.BackgroundColor("Grey")
.Border(border => border.Color("Grey").Width(1))
.ShowMarkerPointers(true).ShowBarPointers(false)
.Labels(label => { label.DistanceFromScale(distance =>
distance.X(50).Y(0)).Add(); })
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Type(MarkerType.Pentagon)
.Placement(PointerPlacement.Near).Width(20).Length(10)
.MarkerDistanceFromScale(20).MarkerBackgroundColor("#FE8282").Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(30).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(30).Y(0)).Height(6).Add();
}).Add();
//Adding Scale 2
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
scale.Width(8).Direction(Directions.Clockwise)
.Type(ScaleType.RoundedRectangle)
.Position(position => position.X(90).Y(50))
.BackgroundColor("#206BA4")
.Border(border => border.Color("#206BA4").Width(1))
.ShowMarkerPointers(false).ShowBarPointers(false).ShowLabels(false)
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0))
.Placement(TickPlacement.Far).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(6).Color("#206BA4")
.DistanceFromScale(distance => distance.X(-27).Y(0)).Height(6)
.Placement(TickPlacement.Far).Add();
}).Add();
//Adding Scale 3
scale.Width(18).Length(300)
.Position(position => position.X(54).Y(50))
.Type(ScaleType.Thermometer)
.BackgroundColor("#C0B08E")
.Border(border => border.Color("#C0B08E").Width(1))
.ShowMarkerPointers(false).ShowBarPointers(false)
.ShowLabels(false).ShowTicks(false)
.Add();
}))
```

Execute the above code to render the following output.



Marker Pointers

Marker Pointer value points out the actual value set in the **Linear Gauge**. You can set values for various pointer attributes such as value, type, length, width, border and color in pointer collection. You can also customize the pointers to improve the appearance of gauge.

Copyright © 2001 - 2020 Syncfusion Inc.

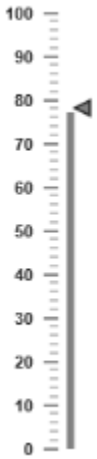
Adding marker pointer collection

You can add **Marker Pointer** collection directly to the scale object. To add pointer collection in a gauge control refer the following code example.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.Value(78)
//Adding Scale Collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true)
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(5).Length(10)
.MarkerBackgroundColor("Grey").MarkerDistanceFromScale(17).Add(); })
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(5)
.BarPointerBackgroundColor("Grey").BarPointerDistanceFromScale(15).Add();
})
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Add marker pointer value

The value property is the important element in the marker pointer collection which indicates the gauge value. Real purpose of the **Linear Gauge** is based on the pointer value. You can set the pointer value either directly during rendering the control or it can be achieved by public method.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
.Width(600).Height(150).Orientation(Orientation.Horizontal).LabelColor("Blac
k")
//Adding Frame object
.Frame(fr=>fr.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light
1.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(30).BackgroundColor("#AEC75F").Border(border =>
border.Color("#AEC75F").Width(30))
.Direction(Directions.Clockwise).Type(ScaleType.RoundedRectangle)
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(30).Length(30)
.MarkerBackgroundColor("#FE5C09").MarkerDistanceFromScale(20)
.Placement(PointerPlacement.Near).Value(67.5).Add(); })
//Adding label collection
.Labels(label => { label.Angle(90).DistanceFromScale(distance =>
distance.Y(100).X(0)).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(45).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(45).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Pointer Styles

Appearance

- Based on the value, the **pointer** points out the label value. You can set the pointer length and width using length and width property respectively. You can also adjust the opacity of the pointer using the opacity property which holds the value between 0 and 1. You can add the gradient effects to the pointer using gradient object.
- The marker pointer border is modified with the border object. It contains two border property namely color and width which are used to customize the border color of the scale and border width of the marker pointer. The background color can be customized with attribute backgroundColor.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
.Width(600).Height(150).Orientation(Orientation.Horizontal).LabelColor("Blac
k")
.EnableResize(true)
//Adding Frame object
.Frame(fr=>fr.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light
1.png"))
//Adding scale collection
.Scales(scale => {
scale.BackgroundColor("#AEC75F").Border(border =>
border.Color("#AEC75F").Width(30))
.Direction(Directions.Clockwise).Type(ScaleType.RoundedRectangle)
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(30).Length(30).MarkerOpacity(0.4)
.MarkerBackgroundColor("#FCDD34").MarkerDistanceFromScale(20)
.Placement(PointerPlacement.Near).Value(67.5).Add(); })
//Adding label collection
.Labels(label => { label.Angle(90).DistanceFromScale(distance =>
distance.Y(100).X(0)).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(-1).Y(45)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(-1).Y(45)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Positioning the pointer

- You can position the Pointer with two properties, **distanceFromScale** and **placement**. The distanceFromScale property defines the distance between the scale and pointer.
- The Placement property is used to locate the pointer with respect to scale either inside or outside the scale or along the scale. It is an enumerable data type.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false)
```

```
.Width(600).Height(150).Orientation(Orientation.Horizontal).LabelColor("Blac
k")
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light
1.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(30).BackgroundColor("#AEC75F").Border(border =>
border.Color("#AEC75F").Width(30))
.Direction(Directions.Clockwise).Type(ScaleType.RoundedRectangle)
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(30).Length(30).MarkerOpacity(0.4)
.MarkerBackgroundColor("#01A357").MarkerDistanceFromScale(60)
.Placement(PointerPlacement.Near).Value(55.5).Add(); })
//Adding label collection
.Labels(label => { label.Angle(90).DistanceFromScale(distance =>
distance.Y(100).X(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(-1).Y(45)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(-1).Y(45)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Types

It is possible to change the dimension of the marker pointer. Dimensions available for marker pointer are,

- Rectangle
- Triangle
- Ellipse
- Diamond
- Pentagon
- Circle
- Slider
- Pointer
- Wedge
- Trapezoid
- Rounded Rectangle

Copyright © 2001 - 2020 Syncfusion Inc.

Multiple Marker Pointers

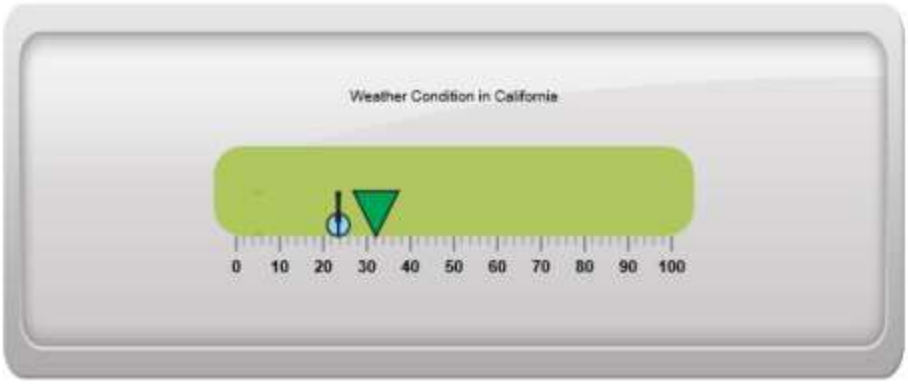
Linear Gauge can contain multiple pointers on it. You can use any combination and any number of pointers in a gauge. That is, a gauge can contain any number of marker pointer and any number of bar pointers. Refer the following code example containing multiple marker pointers.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1")
.EnableAnimation(false).LabelColor("Black").EnableResize(true)
.Width(600).Theme(Themes.FlatLight).Height(250)
.Orientation(Orientation.Horizontal).LabelColor("Black")
//Adding frame object
.Frame(fr=>fr.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light
1.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(30).BackgroundColor("#AEC75F")
.Border(border => border.Color("#AEC75F").Width(30))
.Direction(Directions.Clockwise)
.Type(ScaleType.RoundedRectangle).ShowCustomLabels(true)
//Adding marker pointer collection
.MarkerPointers(mp =>
{
//Adding marker pointer 1
mp.Width(30).Length(30)
.MarkerBackgroundColor("#01A357").MarkerDistanceFromScale(60)
.Placement(PointerPlacement.Near).Value(32.2).Add();
//Adding marker pointer 2
mp.Width(10).Length(30)
.MarkerBackgroundColor("#90DAFB").MarkerDistanceFromScale(60)
.Placement(PointerPlacement.Near).Value(23.7).Type(MarkerType.Circle).Add();
//Adding marker pointer 3
mp.Width(3).Length(30)
.MarkerBackgroundColor("#90DAFB").MarkerDistanceFromScale(60)
.Placement(PointerPlacement.Near).Value(23.7).Type(MarkerType.Star).Add();
})
//Adding label collection
.Labels(label => { label.Angle(90).DistanceFromScale(distance =>
distance.Y(100).X(0)).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(45).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(45).Y(0)).Height(6).Add();
})
//Adding custom label collection
.CustomLabels(cl => { cl.Value("Weather Condition in California")
.Position(position => position.X(52).Y(20)).Add();
})
.Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Bar Pointers

Bar Pointer value points out the actual value set in the **Linear Gauge** as marker pointer. You can set the values of the various bar pointer attributes such as value, width, border and color in bar pointer collection. You can also customize the pointers to improve the appearance of gauge.

Adding bar pointer collection

You can add Bar Pointer collection directly to the scale object. Refer the following code example.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
.Value(78)
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false)
//Adding bar pointer collection
.BarPointers(bar => {
bar.Width(5).BarPointerDistanceFromScale(15)
.BarPointerBackgroundColor("Grey").Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Linear Gauge with bar pointer

Adding bar pointer value

Bar pointer value is also important element in the **Linear Gauge** as it indicates the gauge value. Real purpose of the **Linear Gauge** is based on the pointer value. You can set the bar pointer value either directly during rendering the control or it can be achieved by public method.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale object
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false)
//Adding bar pointer collection
.BarPointers(bar => {
bar.Width(5).BarPointerDistanceFromScale(15)
.BarPointerBackgroundColor("Grey").BarPointerValue(91).Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Pointer Styles

Appearance

- Based on the value, the bar pointer points out the label value. You can set the bar pointer width using width property and you can also adjust the opacity of the pointer using opacity property that holds the value between 0 and 1. You can add the gradient effects to the pointer using gradient object.
- The marker pointer border is modified with the object border. It has two border property, color and width which are used to customize the border color of the scale and border width of the marker pointer. The background color can be customized with attribute backgroundColor.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false)
//Adding bar pointer collection
.BarPointers(bar => {
bar.Width(10).BarPointerDistanceFromScale(15).BarPointerOpacity(0.7)
.BarPointerBackgroundColor("Red").BarPointerValue(91).Add();
})
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
}))
```

Execute the above code to render the following output.



Positioning the pointer

- Bar pointer can be positioned with two properties such as **distanceFromScale** and **placement**. The distanceFromScale property defines the distance between the scale and pointer element.
- The placement property is used to locate the pointer with respect to scale either inside or outside the scale or along the scale. It is an enumerable data type.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false)
//Adding bar pointer collection
.BarPointers(bar => {
bar.Width(10).BarPointerDistanceFromScale(40)
.BarPointerBackgroundColor("#8BABFF").BarPointerValue(91).Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
}))
```

Execute the above code to render the following output.



Multiple Bar Pointers

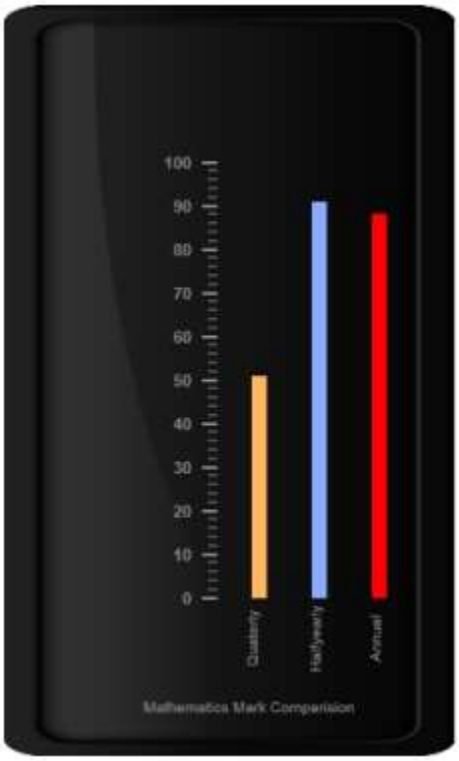
Linear Gauge can contain multiple bar pointers on it. You can use any combination and any number of pointers in a gauge. That is, a Gauge can contain any number of marker pointer and any number of bar pointers. Refer the following code example containing multiple bar pointers.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Height(500).Width(300).LabelColor("Grey")
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_dark1.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false).ShowCustomLabels(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
//Adding bar pointer 1
bar.Width(10).BarPointerDistanceFromScale(60)
.BarPointerBackgroundColor("#8BABFF").BarPointerValue(91).Add();
//Adding bar pointer 2
bar.Width(10).BarPointerDistanceFromScale(20)
.BarPointerBackgroundColor("#FDB761").BarPointerValue(51).Add();
//Adding bar pointer 3
bar.Width(10).BarPointerDistanceFromScale(100)
.BarPointerBackgroundColor("Red").BarPointerValue(88).Add();
})
//Adding tick collection
```

```
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
})
//Adding custom label collection
.CustomLabels(cl => {
cl.Value("Mathematics Mark Comparison").Position(position =>
position.X(55).Y(97)).Add();
cl.Value("Half yearly").Position(position =>
position.X(72).Y(87)).TextAngle(90).Add();
cl.Value("Quarterly").Position(position =>
position.X(56).Y(87)).TextAngle(90).Add();
cl.Value("Annual").Position(position =>
position.X(87).Y(87)).TextAngle(90).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Labels

Labels are units that are used to display the values in the scales. You can customize Labels with the properties like angle, color, font, opacity, etc.

Adding label collection

Label collection can be directly added to the scale object. Refer the following code example to add label collection in a gauge.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Value(40)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_dark1.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false).ShowCustomLabels(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(10).BarPointerDistanceFromScale(13).Add();
})
//Adding label collection
.Labels(label => { label.TextColor("White").Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Label Customization

Appearance

- The attribute **angle** is used to display the labels in the specified angles and **color** attribute is used to display the labels in specified color. You can adjust the **opacity** of the label with the property opacity and the values of it lies between 0 and 1.The includeFirstValue is a special property by enabling this property, the first value of the label is not rendered.
- Font option is also available on the labels. The basic three properties of fonts such as size, family and style can be achieved by **size**, **fontStyle** and **fontFamily**. Labels are two types such as major and minor.Major type labels are for major interval values and minor type labels are for minor interval values.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Value(40)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(true).ShowMarkerPointers(false).ShowCustomLabels(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(10).BarPointerDistanceFromScale(13).Add();
})
//Adding label collection
.Labels(label => { label.TextColor("Red").Angle(10).Opacity(0.5)
.Font(font=>font.Size("12px").FontStyle("Arial").FontFamily("Bold"))
.IncludeFirstValue(false).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(7).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Unit text and Positioning

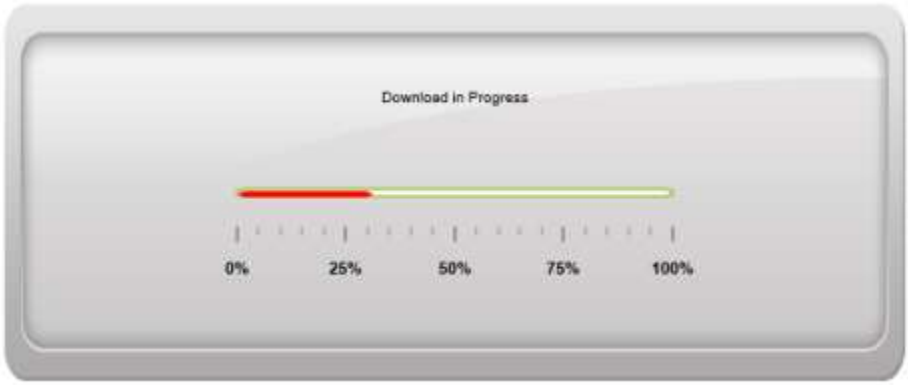
- The unitText property is used to add some text along with the labels. For example, in speedometer, you need to mention the units in kph. You can also add the unit text in front of the labels. To achieve this use the enumerable property unitTextPlacement.
- Labels can be positioned with the help of two properties such as **distanceFromScale** and **placement**. distanceFromScale property defines the distance between the scale and labels. Placement property is used to locate the labels with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Value(31).Width(600).Height(250).Theme(Themes.FlatLight).Orientation(Orient
ation.Horizontal)
.LabelColor("Black").EnableResize(true)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light1.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(5).Border(border =>
border.Color("#AEC75F").Width(2)).BackgroundColor("White")
.MajorIntervalValue(25).MinorIntervalValue(5)
.Type(ScaleType.RoundedRectangle).Direction(Directions.Clockwise)
.ShowBarPointers(true).ShowMarkerPointers(false).ShowCustomLabels(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(4).BarPointerBackgroundColor("Red").Add();
})
//Adding label collection
.Labels(label => { label.Angle(90).DistanceFromScale(dc =>
```

```
dc.X(0).Y(60)).UnitText("%").Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(25).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(25).Y(0)).Height(6).Add();
})
//Adding custom label collection
.CustomLabels(cl => { cl.Value("Download in Progress")
.Position(position => position.X(53).Y(20)).Add(); }).Add();
}))
```

Execute the above code to render the following output.



Ticks

Ticks are used to mark some values on the scale. Based on the tick's value you can set the labels on the required position.

Adding tick collection

Tick collection can be directly added to the scale object. Refer the following code example to add tick collection in a **Linear Gauge** control.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(5).Border(border =>
border.Color("Grey").Width(1)).BackgroundColor("transparent")
.Type(ScaleType.RoundedRectangle)
.ShowBarPointers(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(5).BarPointerBackgroundColor("#95C7E0").Add();
bar.Width(6).BarPointerBackgroundColor("#EDC1D7")
.BarPointerDistanceFromScale(-15).BarPointerValue(30)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.BarPointerOpacity(0.7).Add();
})
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance => distance.X(-
25).Y(0)).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(-10).Y(0)).Add();
tic.Type(TickType.MinorInterval).Width(1).Color("#8C8C8C")
.DistanceFromScale(distance => distance.X(-10).Y(0)).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Tick Customization

Appearance

- Height and width of the ticks can be applied by using the properties height and width. You can customize ticks with the properties like angle, color, etc. angle attribute is used to display the labels in the specified angles and color attribute is used to display the labels in specified color.
- Ticks are two types such as major and minor. The opacity of the labels can be adjusted with the property opacity. The opacity values lies between 0 and 1.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale object
.Scales(scale => {
scale.Width(5).Border(border =>
border.Color("Grey").Width(1)).BackgroundColor("transparent")
.Type(ScaleType.RoundedRectangle)
.ShowBarPointers(true)
//Adding bar pointer value
.BarPointers(bar =>
{
bar.Width(5).BarPointerBackgroundColor("#95C7E0").Add();
})
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance => distance.X(-
25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Height(14).Angle(10).Color("Black"
)
.DistanceFromScale(distance =>
distance.X(-10).Y(0)).Placement(TickPlacement.Near).Add();
tic.Type(TickType.MinorInterval).Width(1).Height(10)
.Opacity(0.5).Color("Black").DistanceFromScale(distance =>
distance.X(-10).Y(0)).Placement(TickPlacement.Near).Height(6).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Types

Ticks are two types such as **majorInterval** and **minorInterval**. Major type ticks are for major interval values and minor type ticks are for minor interval values.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(5).Border(border =>
border.Color("Grey").Width(1)).BackgroundColor("transparent")
.Type(ScaleType.RoundedRectangle)
.ShowBarPointers(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(5).BarPointerBackgroundColor("#95C7E0").Add();
})
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance => distance.X(-
25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Height(14).Color("Black").Add();
tic.Type(TickType.MinorInterval).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Positioning the ticks

- You can position ticks with the help of two properties such as **distanceFromScale** and **placement**. The property distanceFromScale defines the distance between the scale and ticks.
- Placement property is used to locate the ticks with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

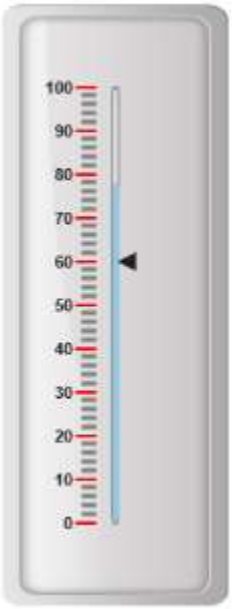
JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame collection
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(5).Border(border =>
border.Color("Grey").Width(1)).BackgroundColor("transparent")
.Type(ScaleType.RoundedRectangle)
.ShowBarPointers(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(5).BarPointerBackgroundColor("#95C7E0").Add();
})
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance => distance.X(-
25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Height(14).Color("Black")
.Placement(TickPlacement.Near).Color("Red")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.DistanceFromScale(distance=>distance.X(-10).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("Grey").Placement(TickPlacement.Near)
.DistanceFromScale(distance => distance.X(-10).Y(0)).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Ranges

Ranges are used to specify or group the scale values. You can describe the values in the pointers using ranges.

Adding range collection

Range collection can be directly added to the scale object. Refer the following code example to add range collection in a **Linear Gauge** control.

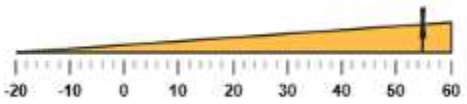
JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Height(150).Width(600).Orientation(Orientation.Horizontal)
.LabelColor("Black").IsResponsive(true)
//Adding scale collection
.Scales(scale => {
scale.Width(0).BackgroundColor("transparent").Border(border =>
border.Color("transparent").Width(0)).ShowMarkerPointers(true)
.ShowBarPointers(false).ShowCustomLabels(true).Direction(Directions.Clockwis
e)
.ShowRanges(true).Minimum(-20).Maximum(60)
//Adding label collection
.Labels(label => { label.DistanceFromScale(distance =>
distance.X(0).Y(50)).Add(); })
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.DistanceFromScale(distance => distance.X(25).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(25).Y(0)).Add();
})
//Adding marker pointer collection
.MarkerPointers(mp =>
{
mp.Width(3).Length(30).MarkerBackgroundColor("#FE5C09")
.Type(MarkerType.Star).MarkerDistanceFromScale(20)
.Placement(PointerPlacement.Near).Value(55).Add();
})
//Adding range collection
.Ranges(range=>{
range.StartValue(-20).EndValue(60).StartWidth(0)
.EndWidth(20).RangeBackgroundColor("#FEBE48")
.Placement(RangePlacement.Near).DistanceFromScale(20).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Range Customization

Appearance

The major attributes for ranges are **startValue** and **endValue**. The startValue defines the start position of the range and endValue defines the end position of the range. The startWidth and endWidth are used to specify the range width at the starting and ending position of the ranges.

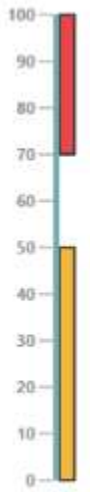
JS

```
@(Html.EJ().LinearGauge("LinearGauge1").Width(500)
.LabelColor("#8c8c8c")
//Adding scale collection
.Scales(scale => {
scale.Width(4).BackgroundColor("#10ADF5").ShowMarkerPointers(false)
.Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(false).Length(310).ShowRanges(true)
//Adding label collection
.Labels(label => {
label.Font(font => font.Size("11px").FontStyle("Bold")
.FontFamily("Segoe UI")).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(1).Color("#8c8c8c").Add();
})
//Adding range collection
.Ranges(range=>{
range.StartValue(0).EndValue(50).StartWidth(8)
.EndWidth(8).RangeBackgroundColor("#F6B53F").DistanceFromScale(5).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
range.StartValue(70).EndValue(100).StartWidth(8)
.EndWidth(8).RangeBackgroundColor("#E94649").DistanceFromScale(5).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Colors and Border

- You can customize the ranges to improve the appearance of the **Gauge**. The range border is modified with the object called **border**. It has two border property such as color and width which are used to customize the border color of the ranges and border width of the ranges.
- You can set the background color to improve the look and feel of the **Linear Gauge**. For customizing the background color of the ranges, backgroundColor is used.You can add the gradient effects to the ranges by using gradient object.

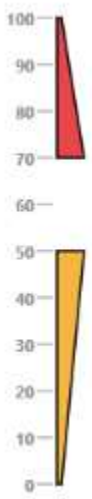
JS

```
@(Html.EJ().LinearGauge("LinearGauge1").Width(500)
.LabelColor("#8c8c8c")
//Adding scale collection
.Scales(scale => {
scale.Width(4).BackgroundColor("transparent").ShowMarkerPointers(false)
.Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(false).Length(310).ShowRanges(true)
//Adding label collection
.Labels(label => {
label.Font(font => font.Size("11px").FontStyle("Bold")
.FontFamily("Segoe UI")).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(1).Color("#8c8c8c").Add();
})
//Adding range collection
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Ranges(range=>{
range.StartValue(0).EndValue(50).StartWidth(18).Border(border =>
border.Color("Black").Width(2))
.EndWidth(3).RangeBackgroundColor("#F6B53F").DistanceFromScale(10).Add();
range.StartValue(70).EndValue(100).StartWidth(18).Border(border =>
border.Color("Black").Width(2))
.EndWidth(3).RangeBackgroundColor("#E94649").DistanceFromScale(10).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Positioning the ranges

- You can position ranges using two properties such as **distanceFromScale** and **placement**. The distanceFromScale property defines the distance between the scale and range.
- Placement property is used to locate the pointer with respect to scale either inside the scale or outside the scale or along the scale. It is an enumerable data type.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").Width(500)
.LabelColor("#8c8c8c")
//Adding scale collection
.Scales(scale => {
scale.Width(4).BackgroundColor("transparent").ShowMarkerPointers(false)
.Border(border => border.Color("transparent").Width(0))
.ShowBarPointers(false).Length(310).ShowRanges(true)
//Adding label collection
.Labels(label => {
label.Font(font => font.Size("11px").FontStyle("Bold")
.FontFamily("Segoe UI")).Add(); })
//Adding Ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(1).Color("#8c8c8c").Add();
```

```
})
//Adding range collection
.Ranges(range=>{
range.StartValue(0).EndValue(50).StartWidth(18).Border(border=>
border.Color("Black").Width(2))
.EndWidth(3).RangeBackgroundColor("#F6B53F").DistanceFromScale(10)
.Placement(RangePlacement.Near).DistanceFromScale(-30).Add();
range.StartValue(70).EndValue(100).StartWidth(18).Border(border=>
border.Color("Black").Width(2))
.EndWidth(3).RangeBackgroundColor("#E94649").DistanceFromScale(10)
.Placement(RangePlacement.Near).DistanceFromScale(-30).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Multiple Ranges

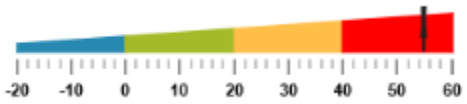
You can set multiple ranges by adding an array of range objects. Refer the following code example for multiple range functionality.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Height(150).Width(600).Orientation(Orientation.Horizontal)
.LabelColor("Black").IsResponsive(true)
//Adding scale collection
.Scales(scale => {
scale.Width(0).BackgroundColor("transparent").Border(border=>
border.Color("transparent").Width(0)).ShowMarkerPointers(true)
.ShowBarPointers(false).ShowCustomLabels(true).Direction(Directions.Clockwis
e)
.ShowRanges(true).Minimum(-20).Maximum(60)
//Adding label collection
.Labels(label => { label.DistanceFromScale(dc => dc.X(0).Y(50)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
```

```
.DistanceFromScale(distance => distance.X(-1).Y(20)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(-1).Y(20)).Add();
})
//Adding marker pointer collection
.MarkerPointers(mp =>
{
mp.Width(3).Length(30).MarkerBackgroundColor("#FE5C09")
.Type(MarkerType.Star).MarkerDistanceFromScale(20)
.Placement(PointerPlacement.Near).Value(55).Add();
})
//Adding range collection
.Ranges(range=>{
range.StartValue(-20).EndValue(0).StartWidth(5)
.EndWidth(10).RangeBackgroundColor("#2788B1")
.Border(border => border.Color("#2788B1"))
.Placement(RangePlacement.Near).DistanceFromScale(20).Add();
range.StartValue(0).EndValue(20).StartWidth(10)
.EndWidth(15).RangeBackgroundColor("#A5BA28")
.Border(border => border.Color("#A5BA28"))
.Placement(RangePlacement.Near).DistanceFromScale(20).Add();
range.StartValue(20).EndValue(40).StartWidth(15)
.EndWidth(20).RangeBackgroundColor("#FEBE48")
.Border(border => border.Color("#FEBE48"))
.Placement(RangePlacement.Near).DistanceFromScale(20).Add();
range.StartValue(40).EndValue(60).StartWidth(20)
.EndWidth(25).RangeBackgroundColor("Red")
.Border(border => border.Color("Red"))
.Placement(RangePlacement.Near).DistanceFromScale(20).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Custom labels

Custom labels are the text that can paste in any location of the **Linear Gauge**. It is used to define the purpose of the gauge.

Adding Custom label collection

Custom labels collection can be directly added to the scale object. Refer the following code to add custom labels collection in a **Linear Gauge** control.

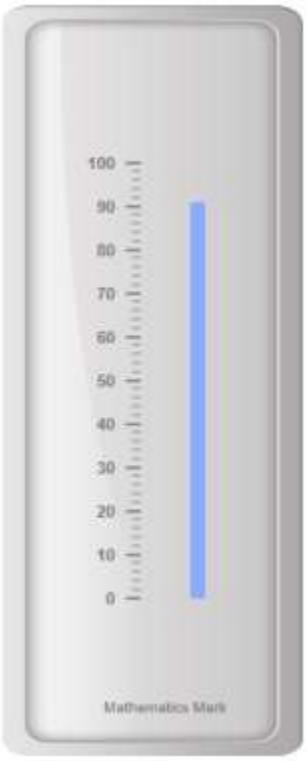
JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).LabelColor("G
rey")
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
scale.Width(0).Border(border=>border.Color("transparent").Width(0)).ShowMark
erPointers(false)
.ShowBarPointers(true).ShowCustomLabels(true)
.ShowBarPointers(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(10).BarPointerBackgroundColor("#8BABFF")
.BarPointerValue(91).BarPointerDistanceFromScale(30).Add();
})
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(dc => dc.X(-25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
})
//Adding custom label collection
.CustomLabels(cl => { cl.Value("Mathematics Mark").Position(position =>
position.X(55).Y(97)).Add(); }).Add();
}))
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Basic Customization

Appearance

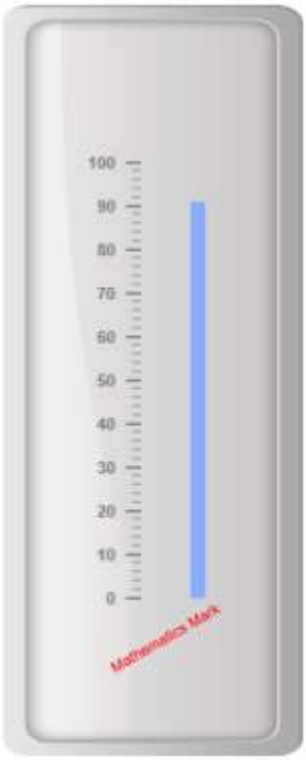
- You can customize custom labels using the properties like textAngle, color and font. The API **textAngle** is used to display the custom labels in the specified angles and **color** attribute is used to display the custom labels in specified color. You can use value attribute to set the text value in the custom labels.
- To display the custom labels, set showCustomLabels as 'true'. Font option is also available on the custom labels. The basic three properties of fonts such as size, family and style can be achieved by **size**, **fontStyle** and **fontFamily**. You can adjust the opacity of the label with the property opacity and the value of opacity lies between 0 and 1.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Height(500).Width(200).LabelColor("Grey")
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border=>border.Color("transparent").Width(0)).ShowMark
erPointers(false)
.ShowBarPointers(true).ShowCustomLabels(true)
.ShowBarPointers(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(10).BarPointerBackgroundColor("#8BABFF")
.BarPointerValue(91).BarPointerDistanceFromScale(30).Add();
})
//Adding marker pointer  collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(dc => dc.X(-25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
})
//Adding custom label collection
.CustomLabels(cl => {
cl.Value("Mathematics Mark")
.TextAngle(30)
.CustomLabelOpacity(0.5)
.Color("Red")
.Position(position => position.X(55).Y(87)).Add();
}).Add();
}))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code to render the following output.



Locating the Custom Labels

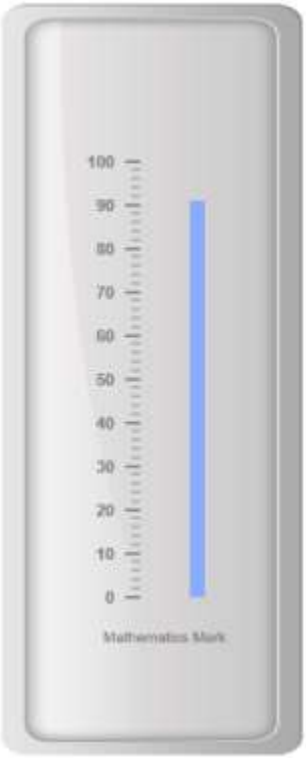
To set the location of the custom label in **Linear Gauge**, position property is used. You can position the custom labels in horizontal and vertical axis using X and Y axis respectively.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Height(500).Width(200).LabelColor("Grey")
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border=>border.Color("transparent").Width(0)).ShowMark
erPointers(false)
.ShowBarPointers(true).ShowCustomLabels(true)
.ShowBarPointers(true)
//Adding bar pointer collection
.BarPointers(bar =>
{
bar.Width(10).BarPointerBackgroundColor("#8BABFF")
.BarPointerValue(91).BarPointerDistanceFromScale(30).Add();
})
//Adding marker pointer collection
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(dc => dc.X(-25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
```

```
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
})
//Adding custom label collection
.CustomLabels(cl => {
cl.Value("Mathematics Mark")
.Position(position => position.X(55).Y(87)).Add();
}).Add();
}))
```

Execute the above code to render the following output.



Multiple Custom Labels

You can set multiple custom labels in a single **Linear Gauge** by adding an array of custom label objects. Refer the following code example for multiple custom label functionality.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false)
.Height(500).Width(200).LabelColor("Grey")
//Adding frame object
.Frame(fr=>fr.InnerWidth(8).OuterWidth(10)
.BackgroundImageUrl("../Content/images/gauge/Gauge_linear_light.png"))
//Adding scale collection
.Scales(scale => {
scale.Width(0).Border(border=>border.Color("transparent").Width(0)).ShowMark
erPointers(false)
```

```
.ShowBarPointers(true).ShowCustomLabels(true)
.ShowBarPointers(true)
//Adding bar pointers
.BarPointers(bar =>
{
bar.Width(10).BarPointerBackgroundColor("#8BABFF")
.BarPointerValue(91).BarPointerDistanceFromScale(30).Add();
})
//Adding marker pointers
.MarkerPointers(mp => { mp.Width(10).Length(10).Value(60).Add(); })
//Adding label collection
.Labels(label => { label.DistanceFromScale(dc => dc.X(-25).Y(0)).Add(); })
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(-7).Y(0)).Add();
})
//Adding custom labels
.CustomLabels(cl => {
//Adding custom label1
cl.Value("Mathematics Mark")
.Position(position => position.X(55).Y(87)).Color("Red").Add();
//Adding custom label 2
cl.Value("Marks in %")
.Position(position =>
position.X(15).Y(57)).Color("Red").TextAngle(90).Add();
}).Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Indicators

Indicators simply indicates the current status of the pointer. Indicators are in several formats such as in shape format, textual format and image format.

Setting Dimension

- You can enable indicators by setting showIndicators to 'true' in scale collection. The height and width property for the indicators are used to specify the area allocated to the indicator for the width and height respectively.
- You can use the position collection to position the indicators along **X** and **Y** axis. X specifies horizontal position in indicators whereas Y specifies vertical position in indicators. Indicators are of several types such as, dimensions like circle, rectangle, rounded rectangle, text and image. By using the type property it can be applied. For image type **imageUrl** property is used.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(frm=>frm.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png")
)
//Adding scale collection
.Scales(s =>
{
s.Width(0).Maximum(300).MinorIntervalValue(5)
.BackgroundColor("transparent").MajorIntervalValue(30)
.Border(border=>border.Color("transparent").Width(0))
.ShowIndicators(true).ShowBarPointers(false)
.Length(310)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Position(position => { position.X(51).Y(50); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{ mp.Length(10).MarkerDistanceFromScale(12)
.Width(10).MarkerBackgroundColor("Grey").Add();
})
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(7).Y(0)).Add();
})
//Adding indicator collection
.Indicators(indicator =>
{
indicator.Type(IndicatorType.Circle).Height(10)
.Width(10).Position(position => position.X(55).Y(100)).Add();
})
.Add();
}))
```

Execute the above code to render the following output.



State Ranges

State ranges are used to specify the indicator behavior in the certain region. `startValue` and `endValue` are used to set the range bound for the pointer. Whenever the pointer crosses the specified region, the indicator attributes are applied for the ranges.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
```

```
.Frame(frm=>frm.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png")
)
//Adding scale collection
.Scales(s =>
{
s.Width(0).Maximum(300).MinorIntervalValue(5)
.BackgroundColor("transparent").MajorIntervalValue(30)
.Border(border=>border.Color("transparent").Width(0))
.ShowIndicators(true).ShowBarPointers(false)
.Length(310).ShowRanges(true)
.Position(position => { position.X(51).Y(50); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{ mp.Length(10).MarkerDistanceFromScale(12)
.Width(10).MarkerBackgroundColor("Grey").Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(7).Y(0)).Add();
})
//Adding indicator collection
.Indicators(indicator =>
{
indicator.Height(10).Width(10).Type(IndicatorType.Circle)
.Position(position => position.X(55).Y(100))
//Adding state range collection
.StateRanges(sr=>{
sr.StateRangeBackgroundColor("#02A258").StateRangeStartValue(0)
.StateRangeEndValue(200).StateRangeBorderColor("#02A258")
.Add();
sr.StateRangeBackgroundColor("Grey").StateRangeStartValue(200)
.StateRangeEndValue(300).StateRangeBorderColor("Grey")
.Add();
}).Add();
})
//Adding range collection
.Ranges(range =>
{
range.StartWidth(5).EndWidth(5).StartValue(0)
.EndValue(200).RangeBackgroundColor("#94C361")
.Border(border => border.Width(1).Color("#94C361")).Add();
range.StartWidth(5).EndWidth(5).StartValue(200)
.EndValue(250).RangeBackgroundColor("#F9CF67")
.Border(border => border.Width(1).Color("#F9CF67")).Add();
range.StartWidth(5).EndWidth(5).StartValue(250)
.EndValue(300).RangeBackgroundColor("#F89B83")
.Border(border => border.Width(1).Color("#F89B83")).Add();
})
.Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Color and Appearance

The backgroundColor and borderColor sets the appearance behavior for the indicators. You can apply this only if it lies within the state ranges. Otherwise default behavior will be applied.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(frm=>frm.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png")
)
//Adding scale object
.Scales(s =>
{
s.Width(0).Maximum(300).MinorIntervalValue(5)
.BackgroundColor("transparent").MajorIntervalValue(30)
.Border(border=>border.Color("transparent").Width(0))
.ShowIndicators(true).ShowBarPointers(false)
.Length(310)
.Position(position => { position.X(51).Y(50); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{ mp.Length(10).MarkerDistanceFromScale(12)
.Width(10).MarkerBackgroundColor("Grey").Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(7).Y(0)).Add();
})
//Adding indicator collection
.Indicators(indicator =>
{
```

```
indicator.Type(IndicatorType.Circle).Height(10)
.Width(10).Position(position => position.X(55).Y(100))
.StateRanges(sr =>
{
sr.StateRangeEndValue(100)
.StateRangeStartValue(0)
.StateRangeBackgroundColor("#91B64E")
.StateRangeBorderColor("#91B64E").Add();
}).Add();
})
.Add();
}))
```

Execute the above code to render the following output.



Font options

The basic font options available for the textual type indicators in the **Linear Gauge** such as **Size**, **font style** and **font family** are achieved by the properties size, fontStyle and fontFamily.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(frm=>frm.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png")
)
//Adding scale collection
.Scales(s =>
{
s.Width(0).Maximum(300).MinorIntervalValue(5)
.BackgroundColor("transparent").MajorIntervalValue(30)
.Border(border=>border.Color("transparent").Width(0))
.ShowIndicators(true).ShowBarPointers(false)
.Length(310).ShowRanges(true)
.Position(position => { position.X(51).Y(50); })
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Adding marker pointer collection
.MarkerPointers(mp =>
{ mp.Length(10).MarkerDistanceFromScale(12)
.Width(10).MarkerBackgroundColor("Grey").Add();
})
//Adding tick collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance => distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(7).Y(0)).Add();
})
//Adding indicator collection
.Indicators(indicator =>
{
indicator.IndicatorFont(font=>
font.Size("12px").FontStyle("Bold").FontFamily("Arial"))
.Type(IndicatorType.text)
.IndicatorTextLocation(position => position.X(55).Y(100))
//Adding state range collection
.StateRanges(sr=>{
sr.StateRangeStartValue(0).StateRangeEndValue(200)
.StateRangeText("Safe").StateRangeTextColor("#94C361")
.Add();
sr.StateRangeStartValue(200).StateRangeEndValue(250)
.StateRangeText("Caution").StateRangeTextColor("#F9CF67")
.Add();
sr.StateRangeStartValue(250).StateRangeEndValue(300)
.StateRangeText("Danger").StateRangeTextColor("#F89B83")
.Add();
}).Add();
})
//Adding range collection
.Ranges(range =>
{
range.StartWidth(5).EndWidth(5).StartValue(0)
.EndValue(200).RangeBackgroundColor("#94C361")
.Border(border => border.Width(1).Color("#94C361")).Add();
range.StartWidth(5).EndWidth(5).StartValue(200)
.EndValue(250).RangeBackgroundColor("#F9CF67")
.Border(border => border.Width(1).Color("#F9CF67")).Add();
range.StartWidth(5).EndWidth(5).StartValue(250)
.EndValue(300).RangeBackgroundColor("#F89B83")
.Border(border => border.Width(1).Color("#F89B83")).Add();
})
.Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Indicator

You can set multiple indicators in a single **Linear Gauge** by adding an array of indicator objects. Refer the following code example for multiple indicator functionality.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").EnableAnimation(false).Value(78)
//Adding frame object
.Frame(frm=>frm.BackgroundImageUrl("../images/gauge/Gauge_linear_light.png")
)
//Adding scale collection
.Scales(s =>
{
s.Width(0).Maximum(300).MinorIntervalValue(5)
.BackgroundColor("transparent").MajorIntervalValue(30)
.Border(border=>border.Color("transparent").Width(0))
.ShowIndicators(true).ShowBarPointers(false)
.Length(310).ShowRanges(true)
.Position(position => { position.X(51).Y(50); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{ mp.Length(10).MarkerDistanceFromScale(12)
.Width(10).MarkerBackgroundColor("Grey").Add();
})
//Adding ticks collection
.Ticks(tic =>
{
tic.Type(TickType.MajorInterval).Width(2).Color("#8c8c8c")
.DistanceFromScale(distance =>distance.X(7).Y(0)).Add();
tic.Type(TickType.MinorInterval).Color("#8c8c8c").Width(1)
.Height(6).DistanceFromScale(distance => distance.X(7).Y(0)).Add();
})
//Adding ranges collection
.Ranges(range =>
```

```
{
range.StartWidth(5).EndWidth(5).StartValue(0)
.EndValue(200).RangeBackgroundColor("#94C361")
.Border(border => border.Width(1).Color("#94C361")).Add();
range.StartWidth(5).EndWidth(5).StartValue(200)
.EndValue(250).RangeBackgroundColor("#F9CF67")
.Border(border => border.Width(1).Color("#F9CF67")).Add();
range.StartWidth(5).EndWidth(5).StartValue(250)
.EndValue(300).RangeBackgroundColor("#F89B83")
.Border(border => border.Width(1).Color("#F89B83")).Add();
})
//Adding indicator collection
.Indicators(indicator =>
{
//Adding indicator 1
indicator.Height(10).Width(10)
.Type(IndicatorType.Circle)
.Position(position => position.X(35).Y(100))
.StateRanges(sr=>{
sr.StateRangeStartValue(0).StateRangeEndValue(200)
.StateRangeBackgroundColor("#02A258").StateRangeBorderColor("#02A258")
.Add();
sr.StateRangeStartValue(200).StateRangeEndValue(300)
.StateRangeBackgroundColor("Grey").StateRangeBorderColor("Grey")
.Add();
}).Add();
//Adding indicator 2
indicator.Height(10).Width(10)
.Type(IndicatorType.Circle)
.Position(position => position.X(75).Y(100))
.StateRanges(sr =>
{
sr.StateRangeStartValue(0).StateRangeEndValue(200)
.StateRangeBackgroundColor("Grey").StateRangeBorderColor("Grey")
.Add();
sr.StateRangeStartValue(200).StateRangeEndValue(300)
.StateRangeBackgroundColor("#02A258").StateRangeBorderColor("#02A258")
.Add();
}).Add();
})
.Add();
}))
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Exporting

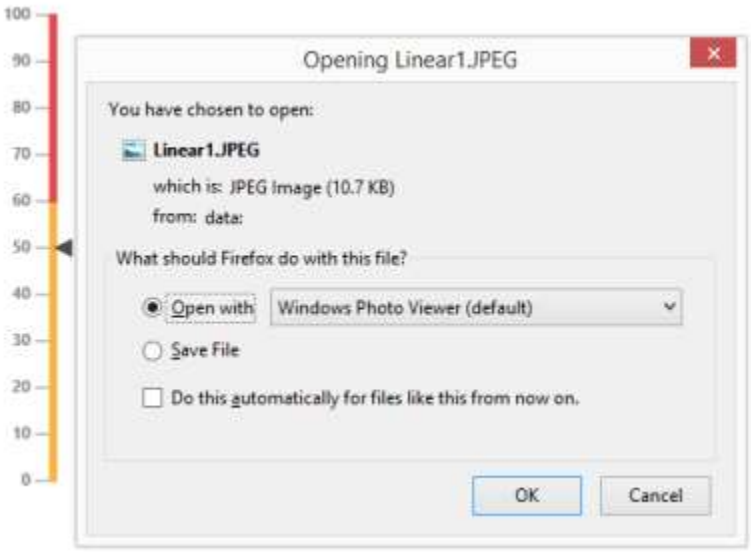
Linear Gauge has an exporting feature that converts **Gauge** control into image format and then export in client side. The method API `exportImage` is used to export the **LinearGauge**. It has two arguments such as **file name** and **file format** to specify the file name and file formats. For exporting refer the following code example.

JS

```
@(Html.EJ().LinearGauge("LinearGauge1").LabelColor("#8c8c8c").Width(500)
//Adding scale collection
.Scales(s =>
{
s.Width(4)
.BackgroundColor("transparent")
.Border(border=>border.Color("transparent").Width(0))
.ShowRanges(true)
.Length(310)
.Position(position => { position.X(51).Y(50); })
//Adding marker pointer collection
.MarkerPointers(mp =>
{ mp.Value(50).Length(10).MarkerDistanceFromScale(8)
.Width(10).MarkerBackgroundColor("#4D4D4D")
.Border(border=>border.Color("4D4D4D")).Add();
})
//Adding label collection
.Labels(label => {
label.Font(font => {
font.Size("11px").FontFamily("SegoeUI").FontStyle("bold");
}).DistanceFromScale(distance=>distance.X(-13)).Add();
})
//Adding bar pointer collection
.BarPointers(bar => {
bar.BarPointerValue(50)
.Width(4).BarPointerDistanceFromScale(8)
.BarPointerBackgroundColor("#6FAAB0").Add();
```

```
})
//Adding ticks collection
.Ticks(tic => {
tic.Type(TickType.MajorInterval)
.Width(1).Color("#8c8c8c").Add();
})
//Adding ranges collection
.Ranges(range =>
{
range.EndValue(60).StartValue(0).RangeBackgroundColor("#F6B53F")
.Border(border=>border.Color("#F6B53F"))
.StartWidth(4).EndWidth(4).Add();
range.EndValue(100).StartValue(60).RangeBackgroundColor("#E94649")
.Border(border=>border.Color("#E94649"))
.StartWidth(4).EndWidth(4).Add();
})
.Add();
}))
<script type="text/javascript">
$(function () {
$("#btnExportImage").ejButton({ width: "100px", click: "buttonClickEvent",
});
});
function buttonClickEvent() {
var FileName = $("#txtFileName").val();
var FileFormat = $("#ddlFileType").val();
$("#CoreExportLinearGauge").ejLinearGauge("exportImage", FileName,
FileFormat);
}
</script>
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

MVVM

AngularJS

Linear Gauge contains AngularJS support. It is possible to add object as well as array object in the **Linear Gauge**. The two way binding support is given to the pointer **value**, **minimum** scale value and **maximum** scale value.

Rendering the Linear gauge

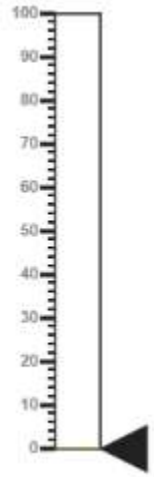
ej-LinearGauge is the control tag, where ej is tag prefix and **LinearGauge** is the control name.

Linear Gauge is rendered with the following code example.

ASPX-CS

```
<div ng-app="syncApp">
<div ng-controller="LinearGauge">
<ej-LinearGauge id="linearCore" e-readOnly="false" e-load="loadGaugeTheme"
e-enableAnimation="false" e-labelColor="#8c8c8c">
</ej-LinearGauge>
</div>
</div>
<script type="text/javascript">
<!—binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('LinearGauge', function ($scope) {
$scope.value = 0;
});
</script>
```

Execute the above code to render the following output.



Adding Scale collection

Scale is an array object and the inner tag is used for it. You can extend the Object in the array collection such as, position with hyphen in the same tag.

Example: e-position-x and e-position-y.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<!--To Render the Linear gauge-->
<ej-LinearGauge id="linearCore">
<!--Adding Scale collection to the Linear gauge-->
<e-scales>
<e-scale e-width="4" e-border-color="transparent" e-border-width="0"
e-showBarPointers="false" e-showRanges="true" e-length="310"
e-position-x="52" e-position-y="50" e-maximum="120">
</e-scale>
</e-scales>
</ej-LinearGauge>
```

Execute the above code to render the following output.



Adding Marker Pointer collection

Marker Pointer is an array object and the inner tag is used for it. You can extend the Object in the array collection such as, border with hyphen in the same tag.

Example: e-border-color.

ASPX-CS

```
<!--To Render the Linear gauge-->
<ej-LinearGauge id="linearCore">
<!--Adding Scale collection to the Linear gauge-->
<e-scales>
<e-scale>
<!--Adding marker pointer collection to the Scale collection-->
<e-markerPointers>
<e-markerPointer e-length="10" e-width="10" e-value="50"
e-backgrouundColor="#4D4D4D"
e-border-color="#4D4D4D">
</e-markerPointer>
</e-markerPointers>
</e-scale>
</e-scales>
</ej-LinearGauge>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding label collection

Label is also an array object and the inner tag is used for it. You can extend the Object in the array collection such as, font with hyphen in the same tag.

Example: e-font-size.

ASPX-CS

```
<!--To Render the Linear gauge-->
<ej-LinearGauge id="linearCore">
<!--Adding Scale collection to the Linear gauge-->
<e-scales>
<e-scale>
<!--Adding marker pointer collection to the Scale collection-->
<e-markerPointers>…</e-markerPointers>
<!--Adding label collection to the Scale collection-->
<e-labels>
<e-label  e-distanceFromScale-x="-10" e-distanceFromScale-y="0"
e-font-fontFamily="Segoe UI" e-font-fontStyle="bold"
e-font-size="11px">
</e-label>
</e-labels>
</e-scale>
</e-scales>
</ej-LinearGauge>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding Tick collection

Tick is an array object and the inner tag is used for it.

ASPX-CS

```
<!--To Render the Linear gauge-->
<ej-LinearGauge id="linearCore">
<!--Adding Scale collection to the Linear gauge-->
<e-scales>
<e-scale>
<!--Adding marker pointer collection to the Scale collection-->
<e-markerPointers>…</e-markerPointers>
<!--Adding label collection to the Scale collection-->
<e-labels>…</e-labels>
<!--Adding tick collection to the Scale collection-->
<e-ticks>
<e-tick e-type="majorinterval" e-width="2" e-color="#8c8c8c">
</e-tick>
</e-ticks>
</e-scale>
</e-scales>
</ej-LinearGauge>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding Range collection

Range is an array object and the inner tag is used for it. You can extend the Object in the array collection such as, border with hyphen in the same tag.

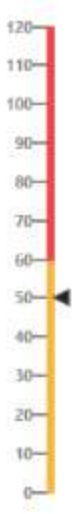
Example: e-border-color.

ASPX-CS

```
<!--To Render the Linear gauge-->
<ej-LinearGauge id="linearCore">
<!--Adding Scale collection to the Linear gauge-->
<e-scales>
<e-scale>
<!--Adding marker pointer collection to the Scale collection-->
<e-markerPointers>…</e-markerPointers>
<!--Adding label collection to the Scale collection-->
<e-labels>…</e-labels>
<!--Adding tick collection to the Scale collection-->
<e-ticks>…</e-ticks>
<!--Adding range collection to the Scale collection-->
<e-ranges>
<e-range e-startValue="0" e-endValue="60" e-startWidth="4"
e-endWidth="4" e-backgroundColor="#F6B53F"
e-border-color="#F6B53F">
</e-range>
<e-range e-startValue="60" e-endValue="120" e-startWidth="4"
e-endWidth="4" e-backgroundColor="#E94649"
e-border-color="#E94649">
</e-range>
</e-ranges>
</e-scale>
</e-scales>
</ej-LinearGauge>
```

Finally while running the above codes, the resultant gauge appears as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Two Way Binding

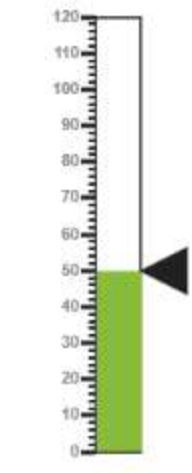
Linear Gauge support the two way binding for the property **value**, **minimum** and **maximum** as mentioned earlier. Following code example explains how to achieve the two way binding to the **Linear Gauge**.

ASPX-CS

```
<div ng-app="syncApp">
<div ng-controller="LinearGauge">
<div id="linearframe">
<ej-LinearGauge id="linearCore" e-value="nvalue" e-readOnly="false" e-
load="loadGaugeTheme" e-enableAnimation="false" e-labelColor="#8c8c8c">
</ej-LinearGauge>
</div>
<input type="text" id="txtMax" e-value="nvalue" ej-numerictextbox ng-
model="nvalue"  e-decimalplaces="2" e-showspinbutton="false"
Style="width:110px"/>
</div>
</div>
<script type="text/javascript">
<!—binding the value to the scope variables in application controller-->
angular.module('syncApp', ['ejangular'])
.controller('LinearGauge', function ($scope) {
$scope.value = 50;
});
</script>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



50

KnockoutJS

- KnockoutJS support allows you to bind the HTML elements against any of the available data models.Two types of KnockoutJS binding is supported as of AngularJS,
- one-way binding
- two-way binding
- **One way binding** refers to the process of applying observable values to all the available properties of the **Linear Gauge** control, but the changes made in it does not reflect and trigger in turn to the observable collection. This kind of binding applies to all the properties of the Linear Gauge control.
- **Two-way binding** supports both the processes – it applies the observable values to the **Linear Gauge** properties as well as the changes made in it is also reflected back and triggered within the observable collections. Only few of the schedule properties support two-way binding and they are as follows
- value
- maximum
- minimum

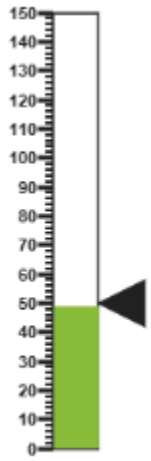
ASPX-CS

```
<div id="LinearGauge1"
data-bind="ejLinearGauge: {
value: sampleValue,
minimum: minimumValue,
maximum: maximumValue
}">
</div>
<script type="text/javascript">
$(function () {
window.viewModel = {
value: ko.observable(50),
minimum: ko.observable(0),
maximum: ko.observable(150)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
};
$(function () {
ko.applyBindings(viewModel);
});
});
</script>
```

Execute the above code to render the following output.



Events

DrawBarPointers

Triggers while the bar pointer are being drawn on the gauge, you can use DrawBarPointers event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawBarPointers("drawBarPointer")
)
function drawBarPointer(){
// Do Something
}
```

DrawCustomLabel

Triggers while the customLabel are being drawn on the gauge, you can use DrawCustomLabel event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawCustomLabel("drawCustomLabel")
)
function drawCustomLabel(){
// Do Something
}
```

DrawIndicators

Triggers while the Indicator are being drawn on the gauge, you can use DrawIndicators event.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawIndicators("drawIndicators")
)
function drawIndicators(){
// Do Something
}
```

DrawLabels

Triggers while the label are being drawn on the gauge, you can use DrawLabels event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawLabels("drawLabels")
)
function drawLabels(){
// Do Something
}
```

DrawMarkerPointers

Triggers while the marker are being drawn on the gauge, you can use DrawMarkerPointers event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawMarkerPointers("drawMarkerPointer")
)
function drawMarkerPointer(){
// Do Something
}
```

DrawRange

Triggers while the range are being drawn on the gauge, you can use DrawRange event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawRange("drawRange")
)
function drawRange(){
// Do Something
}
```

DrawTicks

Triggers while the ticks are being drawn on the gauge, you can use DrawTicks event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.DrawTicks("drawTicks")
)
function drawTicks(){
// Do Something
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

Init

Triggers when the gauge is initialized, you can use Init event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.Init("init")
)
function init(){
// Do Something
}
```

Load

Triggers while the gauge start to Load, you can use Load event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.Load("load")
)
function load(){
// Do Something
}
```

MouseClick

Triggers when the left mouse button is clicked, you can use MouseClick event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.MouseClick("MouseClick")
)
function MouseClick(){
// Do Something
}
```

MouseClickMove

Triggers when clicking and dragging the mouse pointer over the gauge pointer, you can use MouseClickMove event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.MouseClickMove("MouseClickMove")
)
function MouseClickMove(){
// Do Something
}
```

MouseClickUp

Triggers when the mouse click is released, you can use MouseClickUp event.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.MouseClickUp("MouseClickUp")
)
function MouseClickUp(){
// Do Something
}
```

RenderComplete

Triggers while the rendering of the gauge completed, you can use RenderComplete event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
.RenderComplete("RenderComplete")
)
function RenderComplete(){
// Do Something
}
```

DoubleClick

Triggers when the double click is released, you can use DoubleClick event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
//double click event
.DoubleClick("onDoubleClick")
//...
)
function onDoubleClick(sender) {
// do something
}
```

RightClick

Triggers when the right click is released, you can use RightClick event.

ASPX-CS

```
@(Html.EJ().LinearGauge("container")
//right click event
.RightClick("onRightClick")
//...
)
function onRightClick(sender) {
// do something
}
```

Methods

destroy()

Destroy the linear gauge all events bound using this._on will be unbind automatically and bring the control to pre-init state.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.destroy();
```

getBarDistanceFromScale()

To get bar distance from scale in number, you can use getBarDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getBarDistanceFromScale();
```

getBarPointerValue()

To get Bar Pointer Value in number, you can use getBarPointerValuemethod.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getBarPointerValue();
```

getBarWidth()

To get Bar Width in number, you can use getBarWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getBarWidth();
```

getCustomLabelAngle()

To get CustomLabel Angle in number, you can use getCustomLabelAngle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getCustomLabelAngle();
```

getCustomLabelValue()

To get CustomLabel Value in string, you can use getCustomLabelValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getCustomLabelValue();
```

getLabelAngle()

To get Label Angle in number, you can use getLabelAngle method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getLabelAngle();
```

getLabelPlacement()

To get LabelPlacement in number, you can use getLabelPlacement method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getLabelPlacement();
```

getLabelStyle()

To get LabelStyle in number, you can use getLabelStyle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getLabelStyle();
```

getLabelXDistanceFromScale()

To get Label XDistance From Scale in number, you can use getLabelXDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getLabelXDistanceFromScale();
```

getLabelYDistanceFromScale()

To get PointerValue in number, you can use getLabelXDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getLabelYDistanceFromScale();
```

getMajorIntervalValue()

To get Major Interval Value in number, you can use getMajorIntervalValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getMajorIntervalValue();
```

getMarkerStyle()

To get MarkerStyle in number, you can use getMarkerStyle method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getMarkerStyle();
```

getMaximumValue()

To get Maximum Value in number, you can use getMaximumValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getMaximumValue();
```

getMinimumValue()

To get PointerValue in number, you can use getMinimumValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getMinimumValue();
```

getMinorIntervalValue()

To get Minor Interval Value in number, you can use getMinorIntervalValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getMinorIntervalValue();
```

getPointerDistanceFromScale()

To get Pointer Distance From Scale in number, you can use getPointerDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getPointerDistanceFromScale();
```

getPointerHeight()

To get PointerHeight in number, you can use getPointerHeight method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getPointerHeight();
```

getPointerPlacement()

To get Pointer Placement in String, you can use getPointerPlacement method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getPointerPlacement();
```

getPointerValue()

To get PointerValue in number, you can use getPointerValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getPointerValue();
```

getPointerWidth()

To get PointerWidth in number, you can use getPointerWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getPointerWidth();
```

getRangeBorderWidth()

To get Range Border Width in number, you can use getRangeBorderWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangeBorderWidth();
```

getRangeDistanceFromScale()

To get Range Distance From Scale in number, you can use getRangeDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangeDistanceFromScale();
```

getRangeEndValue()

To get Range End Value in number, you can use getRangeEndValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangeEndValue();
```

getRangeEndWidth()

To get Range End Width in number, you can use getRangeEndWidth method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangeEndWidth();
```

getRangePosition()

To get Range Position in number, you can use getRangePosition method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangePosition();
```

getRangeStartValue()

To get Range Start Value in number, you can use getRangeStartValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangeStartValue();
```

getRangeStartWidth()

To get Range Start Width in number, you can use getRangeStartWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getRangeStartWidth();
```

getScaleBarLength()

To get ScaleBarLength in number, you can use getScaleBarLength method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getScaleBarLength();
```

getScaleBarSize()

To get Scale Bar Size in number, you can use getScaleBarSize method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getScaleBarSize();
```

getScaleBorderWidth()

To get Scale Border Width in number, you can use getScaleBorderWidth method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getScaleBorderWidth();
```

getScaleDirection()

To get Scale Direction in number, you can use getScaleDirection method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getScaleDirection();
```

getScaleLocation()

To get Scale Location in object, you can use getScaleLocation method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getScaleLocation();
```

getScaleStyle()

To get Scale Style in string, you can use getScaleStyle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getScaleStyle();
```

getTickAngle()

To get Tick Angle in number, you can use getTickAngle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getTickAngle();
```

getTickHeight()

To get Tick Height in number, you can use getTickHeight method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getTickHeight();
```

getTickPlacement()

To get getTickPlacement in number, you can use getTickPlacement method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getTickPlacement();
```

getTickStyle()

To get Tick Style in string, you can use getTickStyle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getTickStyle();
```

getTickWidth()

To get Tick Width in number, you can use getTickWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getBarDistanceFromScale();
```

getTickXDistanceFromScale()

To get get Tick XDistance From Scale in number, you can use getTickXDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getTickWidth();
```

getTickYDistanceFromScale()

To get Tick YDistance From Scale in number, you can use getTickYDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.getTickYDistanceFromScale();
```

scales()

Specifies the scales, you can use scales method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.scales();
```

setBarDistanceFromScale()

To set setBarDistanceFromScale, you can use setBarDistanceFromScale method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setBarDistanceFromScale();
```

setBarPointerValue()

To set setBarPointerValue, you can use setBarPointerValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setBarPointerValue();
```

setBarWidth()

To set setBarWidth, you can use setBarWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setBarWidth();
```

setCustomLabelAngle()

To set setCustomLabelAngle, you can use setCustomLabelAngle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setCustomLabelAngle();
```

setCustomLabelValue()

To set setCustomLabelValue, you can use setCustomLabelValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setCustomLabelValue();
```

setLabelAngle()

To set setLabelAngle, you can use setLabelAngle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setLabelAngle();
```

setLabelPlacement()

To set setLabelPlacement, you can use setLabelPlacement method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setLabelPlacement();
```

setLabelStyle()

To set setLabelStyle, you can use setLabelStyle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setLabelStyle();
```

setLabelXDistanceFromScale()

To set setLabelXDistanceFromScale, you can use setLabelXDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setLabelXDistanceFromScale();
```

setLabelYDistanceFromScale()

To set setLabelYDistanceFromScale, you can use setLabelYDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setLabelYDistanceFromScale();
```

setMajorIntervalValue()

To set setMajorIntervalValue, you can use setMajorIntervalValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setMajorIntervalValue();
```

setMarkerStyle()

To set setMarkerStyle, you can use setMarkerStyle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setMarkerStyle();
```

setMaximumValue()

To set setMaximumValue, you can use setMaximumValue method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setMaximumValue();
```

setMinimumValue()

To set setMinimumValue, you can use setMinimumValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setMinimumValue();
```

setMinorIntervalValue()

To set setMinorIntervalValue, you can use setMinorIntervalValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setMinorIntervalValue();
```

setPointerDistanceFromScale()

To set setPointerDistanceFromScale, you can use setPointerDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setPointerDistanceFromScale();
```

setPointerHeight()

To set PointerHeight, you can use setPointerHeight method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setPointerHeight();
```

setPointerPlacement()

To set setPointerPlacement, you can use setPointerPlacement method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setPointerPlacement();
```

setPointerValue()

To set PointerValue, you can use setPointerValue method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setPointerValue();
```

setPointerWidth()

To set PointerWidth, you can use setPointerWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setPointerWidth();
```

setRangeBorderWidth()

To set setRangeBorderWidth, you can use setRangeBorderWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangeBorderWidth();
```

setRangeDistanceFromScale()

To set setRangeDistanceFromScale, you can use setRangeDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangeDistanceFromScale();
```

setRangeEndValue()

To set setRangeEndValue, you can use setRangeEndValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangeEndValue();
```

setRangeEndWidth()

To set setRangeEndWidth, you can use setRangeEndWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangeEndWidth();
```

setRangePosition()

To set setRangePosition, you can use setRangePosition method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangePosition();
```

setRangeStartValue()

To set setRangeStartValue, you can use setRangeStartValue method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangeStartValue();
```

setRangeStartWidth()

To set setRangeStartWidth, you can use setRangeStartWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setRangeStartWidth();
```

setScaleBarLength()

To set setScaleBarLength, you can use setScaleBarLength method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setScaleBarLength();
```

setScaleBarSize()

To set setScaleBarSize, you can use setScaleBarSize method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setScaleBarSize();
```

setScaleBorderWidth()

To set setScaleBorderWidth, you can use setScaleBorderWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setScaleBorderWidth();
```

setScaleDirection()

To set setScaleDirection, you can use setScaleDirection method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setScaleDirection();
```

setScaleLocation()

To set setScaleLocation, you can use setScaleLocation method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setScaleLocation();
```

setScaleStyle()

To set setScaleStyle, you can use setScaleStyle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setScaleStyle();
```

setTickAngle()

To set setTickAngle, you can use setTickAngle method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickAngle();
```

setTickHeight()

To set setTickHeight, you can use setTickHeight method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickHeight();
```

setTickPlacement()

To set setTickPlacement, you can use setTickPlacement method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickPlacement();
```

setTickStyle()

To set setTickStyle, you can use setTickStyle method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickStyle();
```

setTickWidth()

To set setTickWidth, you can use setTickWidth method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickWidth();
```

setTickXDistanceFromScale()

To set setTickXDistanceFromScale, you can use setTickXDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickXDistanceFromScale();
```

setTickYDistanceFromScale()

To set setTickYDistanceFromScale, you can use setTickYDistanceFromScale method.

ASPX-CS

```
@(Html.EJ().LinearGauge("LinearGauge1"))
var linear = $("#LinearGauge1").data("ejLinearGauge");
linear.setTickYDistanceFromScale();
```

ListBox

ListBox

The ListBox control provides a list of options and allows you to choose an item from the list. It includes several other HTML elements such as images, textboxes, check box, and radio buttons and so on. It also supports data binding, template options and multi-select options.

Key Features

- ListBox Customization: Supports customization of the ListBox height and width.
- Data binding: Supports data binding with JSON data and remote data.
- Themes: Essential ASP.NET MVC controls feature 12 built-in themes and also supports custom skin options for user-defined themes.
- Reorder: Moves a list item one-step up and one-step down, so that you are able to rearrange the list items inside ListBox control
- Cascading ListBoxes: To populate contents of the second ListBox based on the selection of first ListBox.
- Grouping: The ability to group list items according to their category.

Copyright © 2001 - 2020 Syncfusion Inc.

- Multi-Selection: The ability to select multiple items from the list using Ctrl and Shift key press.
- Template: Supports using templates to display list items.
- Keyboard navigation: Supports switching focus to the control and navigation through the list items with the key press.
- RTL: Supports displaying the ListBox with list items in direction from right to left.
- LoadOnDemand: Provides LoadOnDemand support for ListBox control, to load only specific set of List Data items.

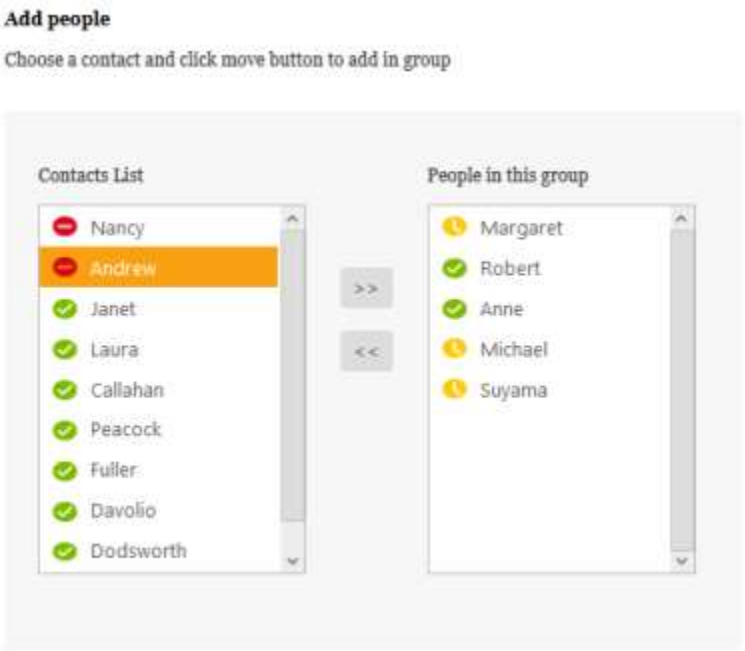
Getting Started

This section explains briefly on how to create a ListBox control in your ASP.NET MVC application.

Create your first ListBox in MVC

Here you can learn how to customize ListBox in Contact Selection tool. This allows you display the list of contacts, to select and move them to the next ListBox that has the selected items. The following example illustrates simulator of Group Creation tool like Skype messenger.

The following screenshot demonstrates the functionality of ListBox with Multi-Selection and Drag and Drop features.



Skype Group creation tool Simulator using ListBox

In the above screenshot, you can select a list item from the first ListBox widget. After you select the item, you can move the selected item to the second ListBox widget.

Create a ListBox Widget

EssentialASP.NET MVCListBox control renders with built-in features.

The following steps are used to create ListBox control.

1. You can create an MVC Project and add necessary assembly with the help of the given MVC-Getting Started Documentation.

2. Please add the below code in your layout._cshtml page to add necessary script and CSS files to render the ListBox control.

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/13.1.0.21/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<!--Scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.globalize.min.js"><
/script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.easing.1.3.min.js">
</script>
<script src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.web.all.min.js">
</script>
</head>
```

Create a CSHTML file in View page and add the following code sample to it.

3. Add the following code in your view page to render ListBox controls.

ASPX-CS

```
<div id="sample">
<h5>
<b>Add people</b></h5>
<p>
Choose a contact and click move button to add in group
</p>
<br />
<div id="control">
<div id="container1">
<p>
Contacts List</p>
@Html.EJ().ListBox("list1")
</div>
<div id="container2">
<p>
People in this group</p>
@Html.EJ().ListBox("list2")
</div>
</div>
</div>
```

4. Add the following style section for the ListBox controls alignment.

CSS

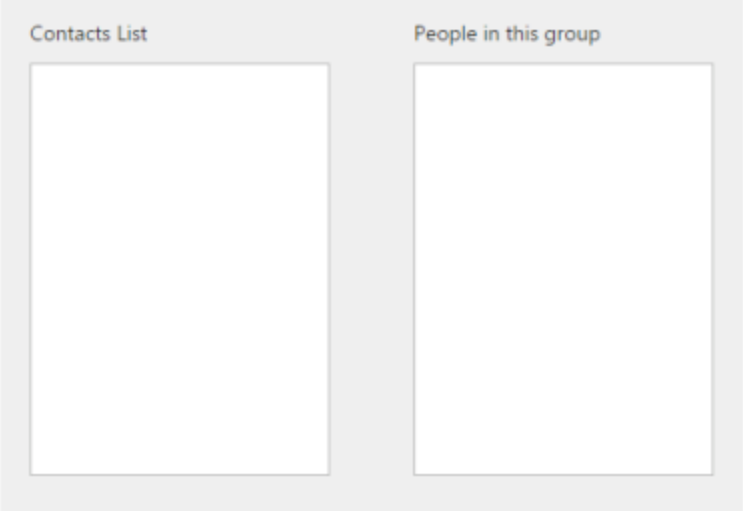
```
<style type="text/css" class="cssStyles">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
#control
{
height: 350px;
width: 500px;
padding: 25px;
background-color: #f7f7f7;
display: flex;
}
#sample
{
height: 472px;
width: 600px;
}
#list2_container
{
float: right;
}
#list1_container
{
float: left;
}
.middle-buttons
{
padding: 91px 25px 25px 25px;
}
#container1, #container2
{
width: 200px;
}
img
{
padding-right: 10px;
padding-top: 3px;
width: 18px;
height: 15px;
}
#form1
{
padding: 25px;
}
</style>
```

5. Run this code to render an empty ListBox control.

Copyright © 2001 - 2020 Syncfusion Inc.



Render ListBox with element

Configure ListBox with Items

To populate items inside ListBox, you have to add list items inside <ul> as elements. The TargetID property is used to get the list items from the target. The Id of the ul that has the list items is given as the TargetID. Include the following <ul>, <li> elements in your sample.

HTML

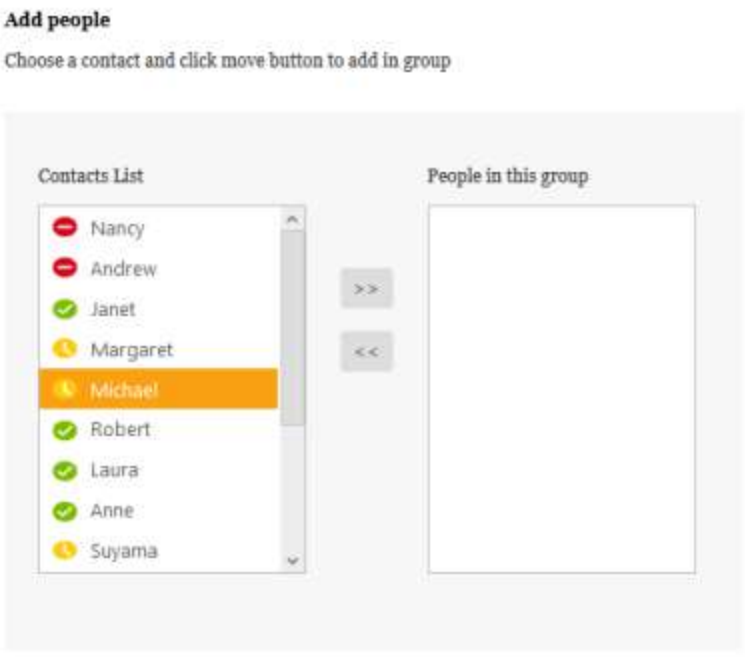
```
<div id="sample">
<h5>
<b>Add people</b></h5>
<p>
Choose a contact and click move button to add in group
</p>
<br />
<div id="control">
<div id="container1">
<p>
Contacts List</p>
@Html.EJ().ListBox("list1").TargetID("select")
<ul id="select">
<li>
<img src="~/images/listbox/busy.png" alt="Categorize" />
Nancy</li>
<li>
<img src="~/images/listbox/busy.png" alt="Done" />
Andrew</li>
<li>
<img src="~/images/listbox/avail.png" alt="Flag" />
Janet</li>
<li>
<img src="~/images/listbox/away.png" alt="Forward" />
Margaret</li>
<li>
<img src="~/images/listbox/away.png" alt="Move" />
Michael</li>
<li>
<img src="~/images/listbox/avail.png" alt="E-mail" />
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Robert</li>
<li>
<img src="~/images/listbox/avail.png" alt="Meeting" />
Laura</li>
<li>
<img src="~/images/listbox/avail.png" alt="Reply" />
Anne</li>
<li>
<img src="~/images/listbox/away.png" alt="Move" />
Suyama</li>
<li>
<img src="~/images/listbox/avail.png" alt="E-mail" />
Callahan</li>
<li>
<img src="~/images/listbox/avail.png" alt="Meeting" />
Peacock</li>
<li>
<img src="~/images/listbox/avail.png" alt="Reply" />
Fuller</li>
<li>
<img src="~/images/listbox/avail.png" alt="Reply" />
Davolio</li>
<li>
<img src="~/images/listbox/avail.png" alt="Reply" />
Dodsworth</li>
<li>
<img src="~/images/listbox/avail.png" alt="Reply" />
Louis</li>
</ul>
</div>
<div class="middle-buttons">
@Html.EJ().Button("Add").Text(">>").ShowRoundedCorner(true).ClientSideEvents
(e => e.Click("add"))
<br />
<br />
@Html.EJ().Button("Remove").Text("<<").ShowRoundedCorner(true).ClientSideEve
nts(e => e.Click("remove"))
</div>
<div id="container2">
<p>
People in this group</p>
@Html.EJ().ListBox("list2")
<ul id="selectedItems">
</ul>
</div>
</div>
</div>
```

Run the above code to render ListBox with list items rendered inside ListBox. ListBox with Contact list items is shown as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



ListBox with Contact list items

Enable Drag and Drop

You can drag an item from a ListBox and drop it in droppable element.To drag and drop a list item across controls or within a control, you have to set the AllowDrag and AllowDrop property as “true”.

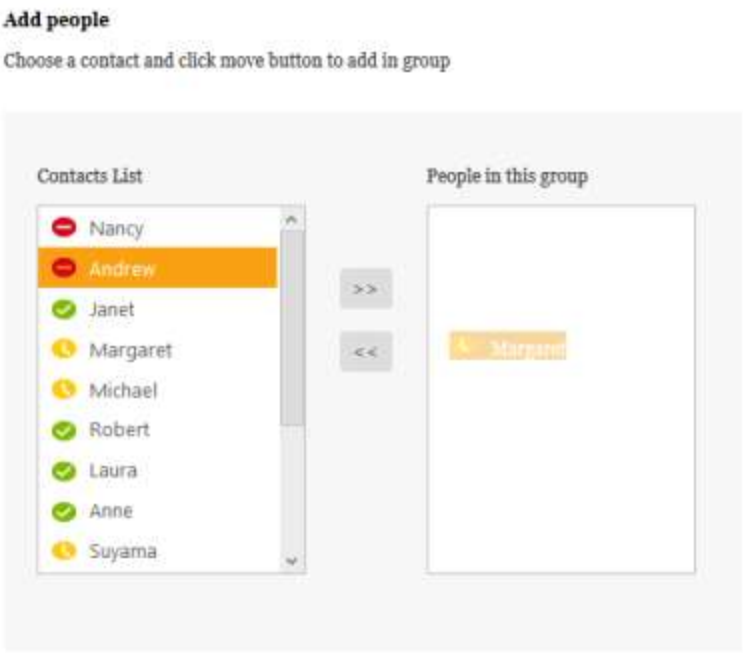
Please refer the below code snippet:

ASPX-CS

```
@Html.EJ().ListBox("list1").TargetID("select").AllowDrag(true).AllowDrop(tru
e)
@Html.EJ().ListBox("list2").AllowDrag(true).AllowDrop(true)
```

Run the above code example to render the following ListBox with Drag and drop feature. ListBox with Drag and Drop list items across control is as follows:

Copyright © 2001 - 2020 Syncfusion Inc.



ListBox with Drag and Drop list items

Enable Multiple Selection

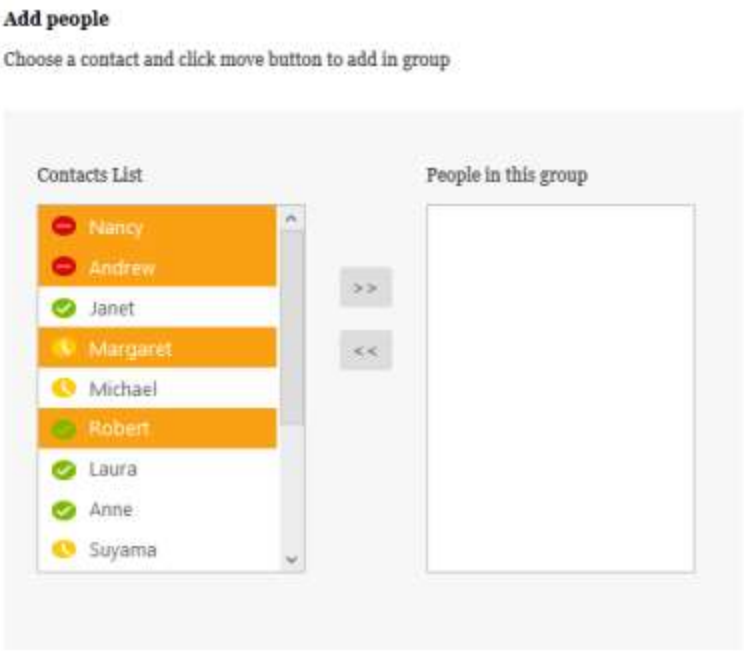
You can select multiple list items simultaneously in ListBox control, and move the multiple, selected items to the selection ListBox. To select multiple items in a ListBox, set the AllowMultiSelection property for the ListBox as “true”.

ASPX-CS

```
@Html.EJ().ListBox("list1").TargetID("select").AllowMultiSelection(true)
.AllowDragAndDrop(true)
```

Run the above code example to render the following ListBox with Multiple selection feature. ListBox control with Multiple Selection of list items is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



ListBox with Multiple Selection of list items

Add items to a Second ListBox

You have to move the selected list items to the second ListBox using AddItem(value) method and remove existing item in the first ListBox using RemoveItem() method.

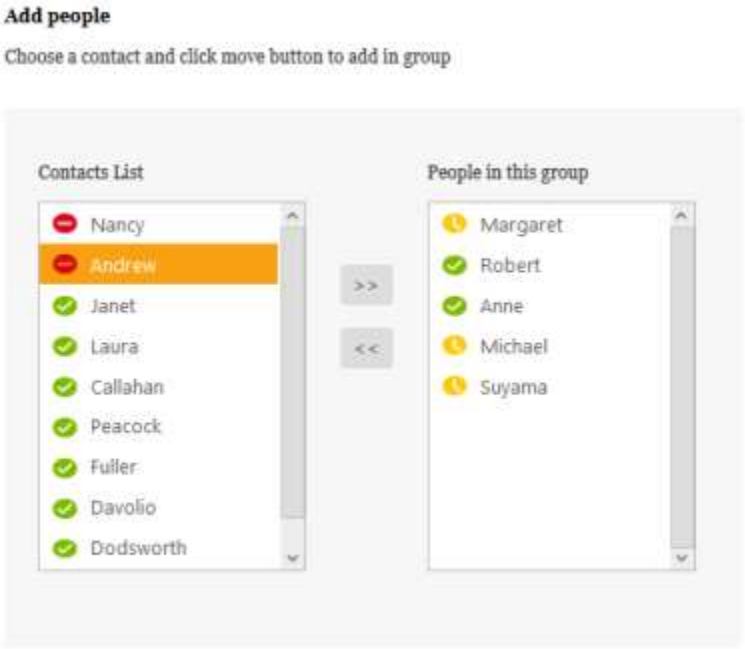
The following code sample explains how to add an item to a second ListBox.

JS

```
<script type="text/javascript">
function add(e)
{
var firstListBox = $('#list1').data("ejListBox");
var selectedItems = firstListBox.getSelectedItems();
var length = selectedItems.length;
for (i = 0; i < length; i++)
{
var value = $(selectedItems[i]).html();
selectedItems[i].remove();
var target = $('#list2').data("ejListBox");
target.addItem(value);
}
}
function remove(e)
{
var firstListBox = $('#list2').data("ejListBox");
var selectedItem = firstListBox.getSelectedItems();
var length = selectedItem.length;
for (i = 0; i < length; i++)
{
var value = $(selectedItem[i]).html();
selectedItem[i].remove();
var target = $('#list1').data("ejListBox");
```

```
target.addItem(value);
}
}
</script>
```

Run this code and you can see the output. Selected items from the first ListBox have been moved to second ListBox using AddItem() and RemoveItem() method and it is displayed in the following figure.



ListBox Selection moved to Second ListBox

Data-binding

The ListBox is populated with the node information taken from a data source. The ListBox supports binding data sources containing hierarchical data and supports Object data, Remote data, XML Data, SQL Data and LinqToSql Data, for retrieving data from a specified data source.

Data fields and configuration

The following sub-properties provides you a way to bind either the local or remote data to the ListBox control.

Property Table for ASP.NET MVC ListBox

Name	Description
Datasource	The data source contains the list of data for generating the ListBox items
Query	It specifies the query to retrieve the data from online server
ListBoxFields	It specifies the mapping fields for the data items of the ListBox
ID	It specifies the id of the tag
Text	It specifies the text content of the tag

Copyright © 2001 - 2020 Syncfusion Inc.

Value	It specifies the value of the tag
ImageUrl	Itâ€™s defines the imageURL for the image location
ImageAttributes	Itâ€™s defines the image attributes such as height, width, styles and so on
SpriteCssClass	It defines the sprite CSS for the image tag.
HtmlAttributes	It defines the HTML attributes such as id, class, styles for the item
Selected	It defines the tag value to be selected initially
ToolTipText	It specifies the Tooltip text of the tag
Category	It specifies the category of the tag
TableName	It defines the table name for tag value or display text while render with remote data

Local data

ListBox provides data binding support. Thus, you can bind the data from JSON Data source. To achieve this, map the corresponding fields with their column names.

And also, provide support to add and customize the list item by using appropriate data fields.

The following code explains you the details of data binding with ListBox.

1. Add the below code in your page to render the ListBox with local data.

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<Ski
llSet>)ViewBag.datasource).ListBoxFields(df=>
df.Text("text"))
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps



Remote data

You can bind the data for the ListBox from remote that can fetch the data from remote web service. You can pass the query string to filter the data that helps to avoid the extensive properties look up by using Query options.

The following steps explains you the details of data binding from remote.

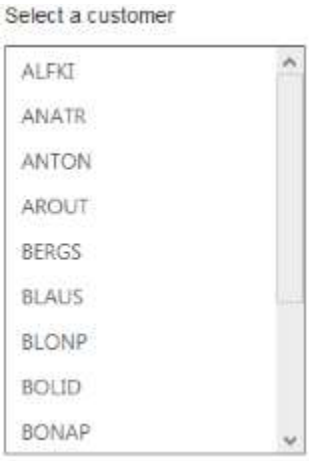
1. Add the below code in your page to render the ListBox with remote data

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div class="control">
<div class="ctrllabel">
Select a customer
</div>
@Html.EJ().ListBox("listBoxSample").Datasource(listBoxDataSource =>
listBoxDataSource.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/")).Q
uery("ej.Query().from('Customers').take(10)").ListBoxFields(f =>
f.Text("CustomerID"))
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

2. Output of the above steps.



Behavior Settings

The following are some miscellaneous properties that helps you to change the behavior of ListBox control.

Target ID

You can append a list with ListBox by using TargetId property. Define a <ul>,< li> tag that you want to display on ListBox and then set the id of parent <ul> tag to TargetId property. And its data type is string.

The following steps explains you the configuration of TargetID property in ListBox.

1. Add the below code in your page to render the ListBox

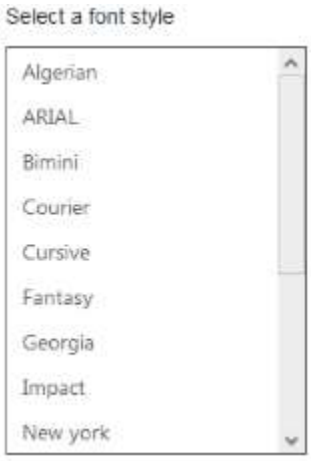
ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a font style
</h5>
@Html.EJ().ListBox("listboxSample").TargetID("targetList")
<ul id="targetList">
<li>Algerian</li>
<li>ARIAL</li>
<li>Bimini</li>
<li>Courier</li>
<li>Cursive</li>
<li>Fantasy</li>
<li>Georgia</li>
<li>Impact</li>
<li>New York</li>
<li>Sans-Serif</li>
<li>Scripts</li>
<li>Times</li>
<li>Times New Roman</li>
<li>Verdana</li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<li>Western</li>
<li>ZapfEllipt BT</li>
</ul>
</div>
```

2. Output of the above steps.



Select the value by index

ListBox widget provides you support to select an item by mentioning the index of the item. The SelectedIndex property helps you to select the particular item from the list. Its date type is number.

The following steps explains you the configuration of SelectedIndex property in ListBox.

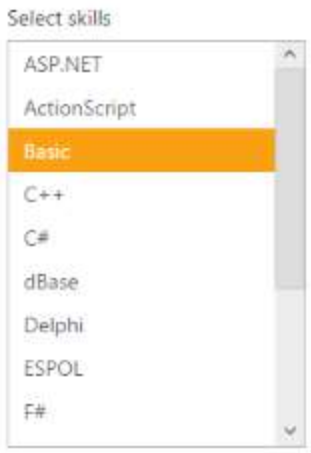
1. Add the below code in your page to render the ListBox

ASPX-CS

```
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listboxSample").Width("240").Datasource((IEnumerable<ug_
listbox.controllers.SkillSet>)ViewBag.datasource).ListBoxFields(df
=> df.Text("text")).SelectedIndex(2)
</div>
```

2. Output of the above steps.

Copyright © 2001 - 2020 Syncfusion Inc.



Enable or Disable the ListBox Widget

This features enables you to set the enable or disable options for ListBox by setting Boolean type value to Enabled property.

The following steps explains you the configuration of Enabled property in ListBox.

1. Add the below code in your view page to render the Disabled ListBox

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listboxSample").Width("240").Datasource((IEnumerable<ug_
listbox.controllers.SkillSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")).Enabled(false)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.



Persistence support

This features enables you to save current model value to browser cookies for state maintains. When you refresh the ListBox control page, it retains the model value apply from browser cookies. The date type of EnablePersistence is Boolean type.

The following steps explains you the configuration of EnablePersistence property in ListBox.

1. Add the below code in your view page to render the ListBox

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<ug_
listbox.controllers.SkillSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")) .EnablePersistence(true)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
```

```
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

Multiple Selection

Allow multiple selection

ListBox widget allows you to select multiple values from the list Items using AllowMultiSelection property. You can select multiple list items along with Control key and Shift key press. To select multiple values we need to set AllowMultiSelection value to true.

Configuring multiple selection

The following steps explain you the configuration of the AllowMultiSelection for a ListBox.

1. Add the below code in your view page to render the ListBox

ASPX-CS

```
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@{List<int> indexList = new List<int>();
indexList.Add(0);
indexList.Add(3);
}
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<Ski
llSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")).SelectedItemList(indexList).AllowMultiSelection(true)
</div>
```

2. Output for ListBox control that provides multiple selection is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple selection through index

You can select the list of items from the ListBox using selectedItemList property. Its data type is array. To achieve this, you need to set true to AllowMultiSelection property in ListBox.

The following steps explains you the configuration of SelectedItemList property in ListBox

1. Add the below code in your view page to render the ListBox with allow multiple selection enabled.

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@{List<int> indexList = new List<int>();
indexList.Add(1);
indexList.Add(4);
}
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<Ski
llSet>)ViewBag.datasource).ListBoxFields(df
=> df.Text("text")).SelectedIndices(indexList).AllowMultiSelection(true)
</div>
```

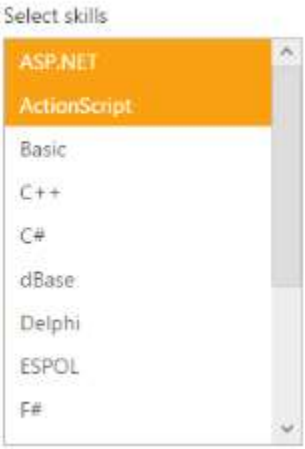
CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.



Checkbox Support

Show Checkbox

You can enable the checkbox in the ListBox with this property. The data type of ShowCheckbox value is Boolean type. It maintains multiple selection and gets the checked items on its ListBox client side events.

Defining the Checkbox support

The following steps explains you the configuration of checkbox options in ListBox.

1. Add the below code in your view page to render the ListBox with checkbox

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<ug_
listbox.controllers.SkillSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")).ShowCheckbox(true)
```

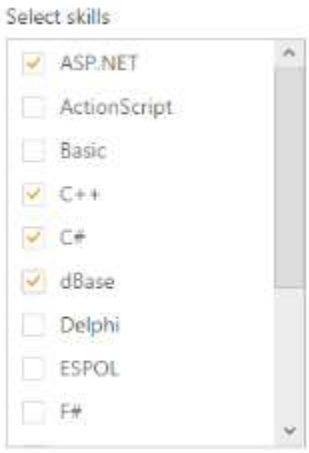
Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.



Check All

You can check all the check box in the list by using this property. The data type of CheckAll is Boolean type. To achieve this, set ShowCheckbox property as true.

Copyright © 2001 - 2020 Syncfusion Inc.

The following steps explains you the configuration of checkbox options in ListBox.

1. Add the below code in your view page to render the ListBox with all items checked initially.

ASPX-CS

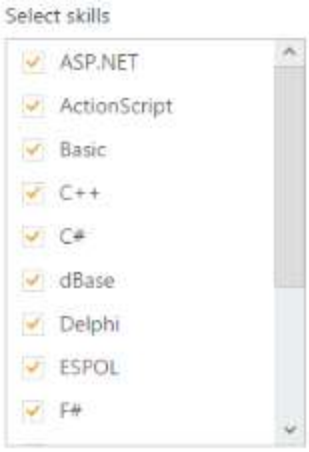
```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<ug_
listbox.controllers.SkillSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")) .ShowCheckbox(true).CheckAll(true)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.

Copyright © 2001 - 2020 Syncfusion Inc.



Uncheck All

You can uncheck all the check box in the list by using this property. The data type of UncheckAll is Boolean type. To achieve this, set ShowCheckbox property as true.

Appearance and Styling

Adjusting ListBox size

Width

ListBox widget provides you support to customize the dimensions of the ListBox. By using Width property you can set the width of the ListBox. Its data type is string.

Height

ListBox widget provides you support to customize the dimensions of the ListBox. By using Height property, you can set the height of the ListBox. Its data type is string.

Defining the ListBox size properties

The following steps explains you the configuration of Height & Width properties in ListBox.

1. Add the below code in your view page to render the ListBox with customized height and width

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Width("240").Height("302").Datasource((I
Enumerable<SkillSet>)ViewBag.datasource).ListBoxFields(df
=> df.Text("text"))
</div>
```

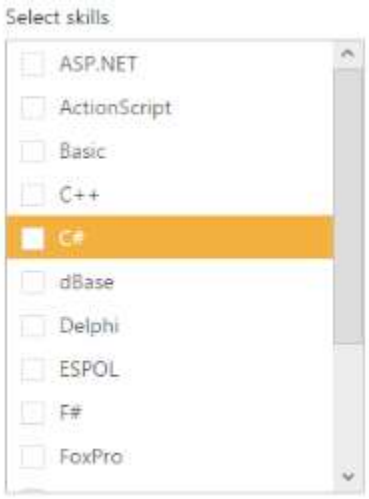
CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps



Enabling Rounded corner

ListBox widget provides you support to change the appearance of ListBox. By using showRoundedCorner you can create a rounded corner on the ListBox. Its data type is Boolean.

The following steps explains you the configuration of Rounded corner of the ListBox.

1. Add the below code in your view page to render the ListBox with rounded corners

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
```

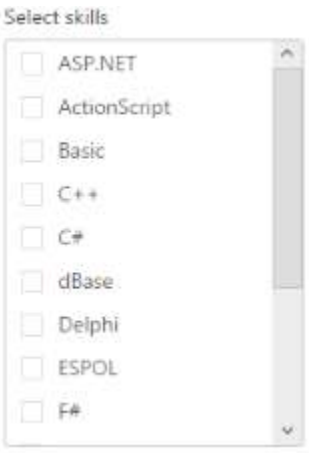
Copyright © 2001 - 2020 Syncfusion Inc.

```
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Width("240").Datasource((IEnumerable<Ski
llSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")).ShowRoundedCorner(true)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.



Copyright © 2001 - 2020 Syncfusion Inc.

Sorting

We can change ListBox items rendering order either as ascending or descending, by using "SortOrder" property. By default SortOrder will be "None". Please use code as like in below,

This is example code position of the ListBox may change if your are apply styles as per your requirement.

1. Add the below code in your view page to render the ListBox with sorting option

ASPX-CS

```
//Add the following code in View page to configure ListBox widget
<div class="contents">
<span class="txt">Select skill</span>
@Html.EJ().ListBox("skillSet").Datasource((IEnumerable<SkillSet>)ViewBag.Dat
aSource1).SortOrder(SortOrder.None).ListBoxFields(df => df.Value("skill"))
</div>
```

CSHARP

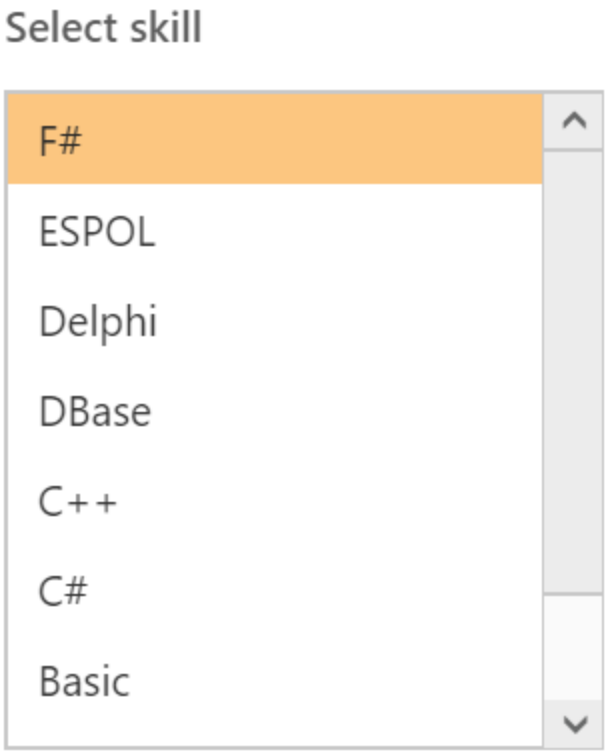
```
// Add the following code to add listbox items in the controller page
public partial class ListBoxController : Controller
{
List<SkillSet> veg1 = new List<SkillSet>();
List<GroupList> skills = new List<GroupList>();
// GET: /Sorting/
public ActionResult Sorting()
{
veg1.Add(new SkillSet { skill = "F#" });
veg1.Add(new SkillSet { skill = "ActionScript" });
veg1.Add(new SkillSet { skill = "Delphi" });
veg1.Add(new SkillSet { skill = "Basic" });
veg1.Add(new SkillSet { skill = "C++" });
veg1.Add(new SkillSet { skill = "ESPOL" });
veg1.Add(new SkillSet { skill = "C#" });
veg1.Add(new SkillSet { skill = "DBase" });
veg1.Add(new SkillSet { skill = "ASP.NET" });
ViewBag.DataSource1 = veg1;
return View();
}
}
```

CSHARP

```
public class SkillSet
{
public string skill { get; set; }
}
```

once you have included the above code your output will be shown as given,

Copyright © 2001 - 2020 Syncfusion Inc.



Theme

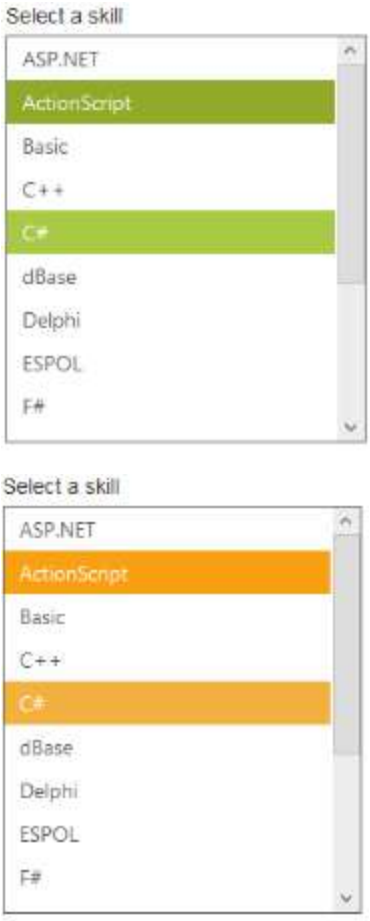
ListBox control's style and appearance can be controlled based on CSS classes. In order to apply styles to the ListBox control, you can refer to two files namely, ej.widgets.core.min.css and ej.theme.min.css. When you refer to the file ej.widgets.all.min.css, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for ListBox control namely,

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

The following screenshot illustrates the ListBox with Flat-lime and Flat-Saffron built-in themes.

Copyright © 2001 - 2020 Syncfusion Inc.



Custom class with ListBox

CSS class can be used to customize the ListBox control appearance. Define a CSS class as per your requirement and assign the class name to CssClass property. The data type of CssClass property is string.

Configuring the Custom CSS property

The following steps explains you the configuration of CssClass properties in ListBox.

1. Add the below code in your view page to render the ListBox with custom CSS class.

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Datasource((IEnumerable<ug_listbox.controllers.SkillSet>)
ViewBag.datasource).ListBoxFields(df => df.Text("text")) .CssClass("custom-class")
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

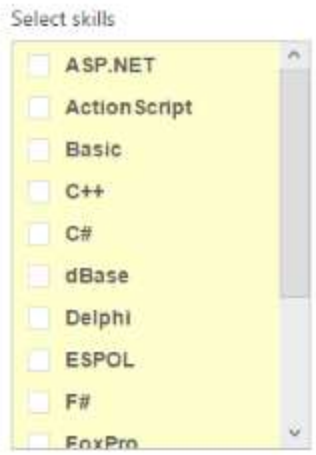
2. Configure the CSS styles to apply on ListBox.

CSS

```
<style>
.custom-class
{
background-color: #FFFFCC;
font-weight: bold;
font-family: sans-serif;
}
</style>
```

3. Output of the above steps.

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip Support

The following steps explains you the configuration of Tooltip properties in ListBox.

1. Add the below code in your page to render the ListBox with tooltip

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Datasource((IEnumerable<SkillSet>)ViewBa
g.datasource).ListBoxFields(df =>
df.Text("text").TooltipText("tooltip"))
</div>
```

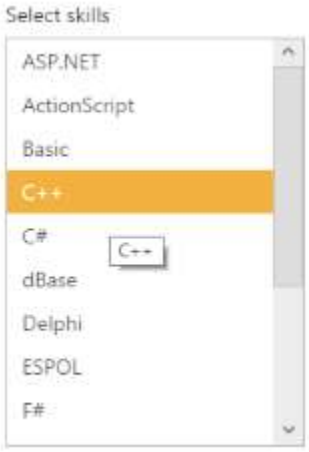
CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
public string tooltip { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET", tooltip = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript", tooltip = "ActionScript"});
skill.Add(new SkillSet { text = "Basic", tooltip = "Basic" });
skill.Add(new SkillSet { text = "C++", tooltip = "C++" });
skill.Add(new SkillSet { text = "C#", tooltip = "C#" });
skill.Add(new SkillSet { text = "dBase", tooltip = "dBase" });
skill.Add(new SkillSet { text = "Delphi", tooltip = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL", tooltip = "ESPOL" });
skill.Add(new SkillSet { text = "F#", tooltip = "F#" });
skill.Add(new SkillSet { text = "FoxPro", tooltip = "FoxPro" });
skill.Add(new SkillSet { text = "Java", tooltip = "Java" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
skill.Add(new SkillSet { text = "J#", tooltip = "J#" });
skill.Add(new SkillSet { text = "Lisp", tooltip = "Lisp" });
skill.Add(new SkillSet { text = "Logo", tooltip = "Logo" });
skill.Add(new SkillSet { text = "PHP", tooltip = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.



Cascading Support

Using cascade option, you can create a behavior of cascade between ListBox controls. For this, create a database with single field that is common between two ListBox data fields and then mention that column id in value field. With this, you can set second ListBox id in CascadeTo property in first ListBox.

In the following code example, in the first and second ListBox, "categoryId" is the common field.

The "categoryId" value of the selected item in the First Listbox that matches with "categoryId" value in the second Listbox, is retrieved and the item is loaded.

Note: In case the second ListBox is to be disabled, until the first one is selected, you can set enable property as false in second ListBox that enables automatically once the value is selected in first one.

You can add any number of cascading ListBox. For this, create a Datasource with single field value that is common between the two consecutive cascading ListBox and cascading is achieved based on that common field."

The following steps explains you the behavior of cascade ListBox.

1. Add the below code in your view page to render the ListBox with cascade option/

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div class="controlitem">
<span class="txt">
Cascading Listbox
```

```
</span>
@Html.EJ().ListBox("groupsList").Datasource((IEnumerable<Groups>)ViewBag.dat
asource).ListBoxFields(df =>
df.Value("CategoryId")).CascadeTo("subcategoryList")
</div>
<div class="controlitem">
@Html.EJ().ListBox("subcategoryList").Datasource((IEnumerable<Category>)View
Bag.datasource1).ListBoxFields(df =>
df.Value("SubCategoryId")).CascadeTo("productList").Enabled(false)
</div>
<div class="controlitem">
@Html.EJ().ListBox("productList").Datasource((IEnumerable<SubCategory>)ViewB
ag.datasource2).ListBoxFields(df =>
df.Value("ProductId")).CascadeTo("subProductList").Enabled(false)
</div>
<div class="controlitem">
@Html.EJ().ListBox("subProductList").Datasource((IEnumerable<SubProduct>)Vie
wBag.datasource3).Enabled(false)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public partial class ListBoxController : Controller
{
List<Groups> group = new List<Groups>();
List<Category> firstLevel = new List<Category>();
List<SubCategory> secondLevel = new List<SubCategory>();
List<SubProduct> thirdLevel = new List<SubProduct>();
public ActionResult Cascading()
{
group.Add(new Groups { CategoryId = "a", Text = "Clothing" });
group.Add(new Groups { CategoryId = "b", Text = "Furniture" });
ViewBag.datasource = group;
//first level child
firstLevel.Add(new Category { SubCategoryId = 11, CategoryId = "a", Text =
"Women" });
firstLevel.Add(new Category { SubCategoryId = 12, CategoryId = "b", Text =
"Home furniture" });
firstLevel.Add(new Category { SubCategoryId = 13, CategoryId = "b", Text =
"Bedding" });
ViewBag.datasource1 = firstLevel;
//second level child
secondLevel.Add(new SubCategory { ProductId = 101, SubCategoryId = 11, Text
= "men shirts" });
secondLevel.Add(new SubCategory { ProductId = 102, SubCategoryId = 11, Text
= "men pants" });
secondLevel.Add(new SubCategory { ProductId = 103, SubCategoryId = 12, Text
= "women shirts" });
secondLevel.Add(new SubCategory { ProductId = 104, SubCategoryId = 12, Text
= "women pants" });
secondLevel.Add(new SubCategory { ProductId = 105, SubCategoryId = 13, Text
= "sofa" });
secondLevel.Add(new SubCategory { ProductId = 106, SubCategoryId = 13, Text
= "chairs" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
secondLevel.Add(new SubCategory { ProductId = 106, SubCategoryId = 14, Text
= "bedsheets" });
secondLevel.Add(new SubCategory { ProductId = 108, SubCategoryId = 14, Text
= "pillows" });
ViewBag.datasource2 = secondLevel;
//third level child
thirdLevel.Add(new SubProduct { ProductId = 101, Text = "red men shirts" });
thirdLevel.Add(new SubProduct { ProductId = 101, Text = "blue men shirts"
});
thirdLevel.Add(new SubProduct { ProductId = 102, Text = "red men pants" });
thirdLevel.Add(new SubProduct { ProductId = 102, Text = "blue men pants" });
thirdLevel.Add(new SubProduct { ProductId = 103, Text = "blue women shirts"
});
thirdLevel.Add(new SubProduct { ProductId = 103, Text = "red women shirts"
});
thirdLevel.Add(new SubProduct { ProductId = 104, Text = "red women pants"
});
thirdLevel.Add(new SubProduct { ProductId = 104, Text = "blue women pants"
});
thirdLevel.Add(new SubProduct { ProductId = 105, Text = "red sofa" });
thirdLevel.Add(new SubProduct { ProductId = 105, Text = "blue sofa" });
thirdLevel.Add(new SubProduct { ProductId = 106, Text = "red chairs" });
thirdLevel.Add(new SubProduct { ProductId = 106, Text = "blue chairs" });
thirdLevel.Add(new SubProduct { ProductId = 107, Text = "red bedsheets" });
thirdLevel.Add(new SubProduct { ProductId = 107, Text = "blue bedsheets" });
thirdLevel.Add(new SubProduct { ProductId = 108, Text = "red pillows" });
thirdLevel.Add(new SubProduct { ProductId = 108, Text = "blue pillows" });
ViewBag.datasource3 = thirdLevel;
return View();
}
}
```

CSHARP

```
public class Groups
{
public string Text { get; set; }
public string CategoryId { get; set; }
public class Category
{
public string Text { get; set; }
public string CategoryId { get; set; }
public int SubCategoryId { get; set; }
}
public class SubCategory
{
public string Text { get; set; }
public int ProductId { get; set; }
public int SubCategoryId { get; set; }
}
public class SubProduct
{
public string Text { get; set; }
public int ProductId { get; set; }
}
}
```

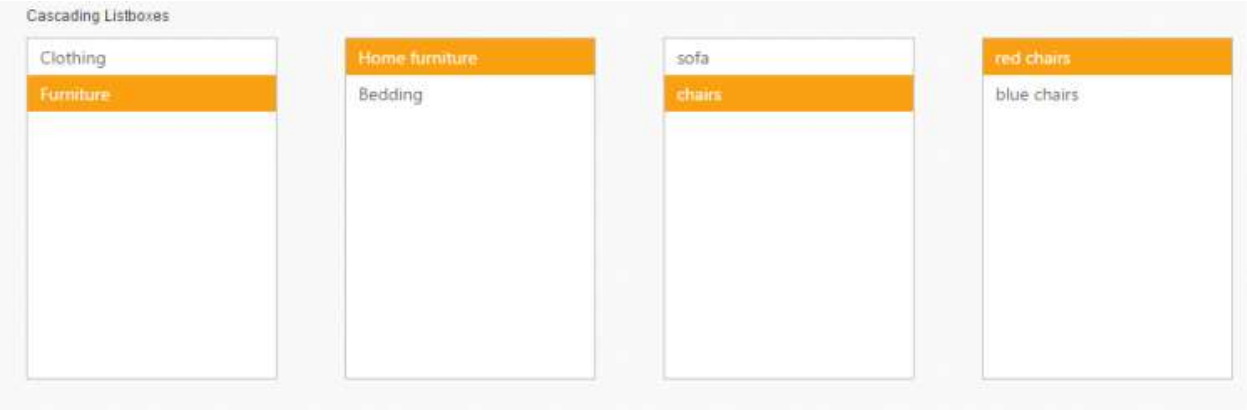
Copyright © 2001 - 2020 Syncfusion Inc.

2. Configure the styles as follows.

ASPX-CS

```
<style>
.controlitem {
margin-left: 50px;
display: inline-block;
}
</style>
```

3. Output of the above steps.



VirtualScrolling support

ListBox widget provides the VirtualScrolling support, when binding the remote data for the ListBox. It loads partially, only a set of data from remote server loaded initially, and imports data further upon loading. To enable VirtualScrolling support, set the EnableVirtualScrolling property as true. You can set ItemsCount that specifies number of items in the ListBox. You can load any number of items upon request with ItemRequest ClientSide Event.

The following steps explains you the behavior of VirtualScrolling support in ListBox.

1. Add the below code in your page to render the ListBox

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div class="control">
<h5 class="ctrllabel">
Select Customer ID
</h5>
@Html.EJ().ListBox("customerList").Datasource(listBoxDatasource =>
listBoxDatasource.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/")).Q
uery("ej.Query().from('Customers')").ListBoxFields(f =>
f.Text("CustomerID")).ItemsCount(91)
.AllowVirtualScrolling(true).ClientSideEvents(e =>
```

```
e.ActionBegin("itemRequested"))
</div>
```

JS

```
<script type="text/javascript">
function itemRequested(args) {
var target = $("#customerList").data("ejListBox");
//to load 20 items
target.model.query = ej.Query().from('Customers').range(args.start,
args.start + 20);
this.model.itemsCount = 20;
}
</script>
```

2. Output of the above steps.



Drag and Drop Support

ListBox widget provides the Drag and Drop support. A list item can be dragged from a ListBox control and can be dropped in any droppable element. To enable Drag and Drop support, set the AllowDrag and AllowDrop property as true. In control, enable the AllowDrag and AllowDrop property where you want to drop list Item.

The following steps explains you the behavior of template support with ListBox.

1. Add the below code in your page to render the ListBox

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div class="control1">
<h5 class="ctrlLabel">
Drag and drop skills
</h5>
@Html.EJ().ListBox("listBoxSample").Datasource((IEnumerable<ug_listbox.contr
ollers.SkillSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")) .AllowDrag(true).AllowDrop(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
<div class="control2">
@Html.EJ().ListBox("dragSample").AllowDrag(true).AllowDrop(true)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class SkillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<SkillSet> skill = new List<SkillSet>();
skill.Add(new SkillSet { text = "ASP.NET" });
skill.Add(new SkillSet { text = "ActionScript" });
skill.Add(new SkillSet { text = "Basic" });
skill.Add(new SkillSet { text = "C++" });
skill.Add(new SkillSet { text = "C#" });
skill.Add(new SkillSet { text = "dBase" });
skill.Add(new SkillSet { text = "Delphi" });
skill.Add(new SkillSet { text = "ESPOL" });
skill.Add(new SkillSet { text = "F#" });
skill.Add(new SkillSet { text = "FoxPro" });
skill.Add(new SkillSet { text = "Java" });
skill.Add(new SkillSet { text = "J#" });
skill.Add(new SkillSet { text = "Lisp" });
skill.Add(new SkillSet { text = "Logo" });
skill.Add(new SkillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

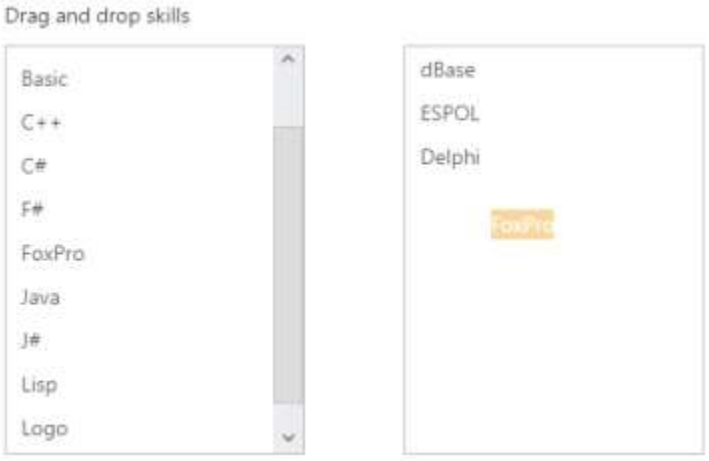
2. Add the following class in CSS.

ASPX-CS

```
<style type="text/css" class="cssStyles">
.control
{
margin-left: 20px;
}
.ctrlLabel
{
padding-bottom: 3px;
}
.control2
{
padding-left: 350px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.

3. Output of the above steps.



Template Support

ListBox widget provides the template support, when binding the data for the ListBox. For this behavior, set the common syntax /element in template property. You can add any HTML mark-up element inside the ListBox using this property.

The following steps explains you the behavior of template support with ListBox.

1. Add the below code in your view page to render the ListBox

Note: Images for this sample are available in 'installed location/images/Employee'

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div class="control">
<div class="ctrllabel">
Template support
</div>
@Html.EJ().ListBox("listBoxSample").Datasource((IEnumerable<employeespecialists>)ViewBag.datasource).Height("238")
.Template("<img class='employee-image'
src='../../Content/images/Employees/${employee-image}.png' alt='employee'
height='50px' width='50px'/>
<div class='e-name'>
${text}
</div>
<div class='design'>
${design}
</div>
<div class='cont'>
${country}
</div>")
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class EmployeeSpecialists
{
public string text { get; set; }
public string employeeImg { get; set; }
public string design { get; set; }
public string country { get; set; }
}
public ActionResult Index()
{
List<EmployeeSpecialists> employeeList = new List<EmployeeSpecialists>();
employeeList.Add(new EmployeeSpecialists { text = "Erik Linden", employeeImg
= "3", design = "Representative", country = "England" });
employeeList.Add(new EmployeeSpecialists { text = "John Linden", employeeImg
= "6", design = "Representative", country = "Norway" });
employeeList.Add(new EmployeeSpecialists { text = "Louis", employeeImg =
"7", design = "Representative", country = "Australia" });
employeeList.Add(new EmployeeSpecialists { text = "Lawrence", employeeImg =
"8", design = "Representative", country = "India" });
ViewBag.datasource = employeeList;
return View();
}
```

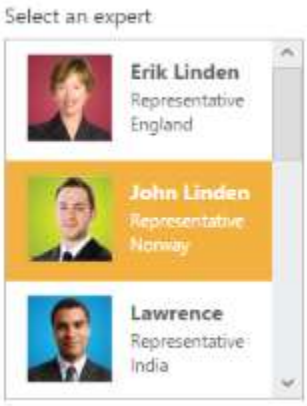
2. Customize the template in CSS.

CSS

```
<style>
.employee-image
{
margin: 0;
padding: 3px 10px 3px 3px;
border: 0 none;
width: 60px;
height: 60px;
float: left;
}
.e-name
{
font-weight: bold;
padding: 6px 3px 1px 3px;
}
.design, .cont
{
font-size: smaller;
padding: 3px 3px -1px 0px;
}
#selectExperts li
{
width: 200px;
height: 70px;
padding: 5px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.

3. Output of the above steps.



RTL Support

This feature supports to change the left-to-right alignment of the ListBox widget to right-to-left (RTL).

Defining the RTL property

The following steps explains you the configuration of enableRTL properties in ListBox.

1. Add the below code in your page to render the ListBox with right to left alignment

ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Datasource((IEnumerable<ug_listbox.contr
ollers.skillSet>)ViewBag.datasource).ListBoxFields(df =>
df.Text("text")).EnableRTL(true)
</div>
```

CSHARP

```
// Add the following code to add list items in the controller page
public class skillSet
{
public string text { get; set; }
}
public ActionResult Index()
{
List<skillSet> skill = new List<skillSet>();
skill.Add(new skillSet { text = "ASP.NET" });
skill.Add(new skillSet { text = "ActionScript" });
skill.Add(new skillSet { text = "Basic" });
skill.Add(new skillSet { text = "C++" });
skill.Add(new skillSet { text = "C#" });
skill.Add(new skillSet { text = "dBase" });
skill.Add(new skillSet { text = "Delphi" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
skill.Add(new skillSet { text = "ESPOL" });
skill.Add(new skillSet { text = "F#" });
skill.Add(new skillSet { text = "FoxPro" });
skill.Add(new skillSet { text = "Java" });
skill.Add(new skillSet { text = "J#" });
skill.Add(new skillSet { text = "Lisp" });
skill.Add(new skillSet { text = "Logo" });
skill.Add(new skillSet { text = "PHP" });
ViewBag.datasource = skill;
return View();
}
```

2. Output of the above steps.



Keyboard interaction

You can use Keyboard shortcut keys as an alternative to the mouse on using ListBox widget. ListBox Widget allows you to perform all kind of actions using keyboard shortcuts.

Keyboard shortcut keys

Shortcut Key	Description
Access keyÂ + j	Focuses into the ListBox text box
Up	Moves to previous item in the ListBox
Down	Moves to next item in the ListBox
Enter	Selects the focused item
Left	Moves to previous item in the ListBox
Right	Moves to next item in the ListBox
Home	Navigates to the starting item
End	Navigates to the end item

Copyright © 2001 - 2020 Syncfusion Inc.

Configure keyboard interaction

The following steps explains you to enable keyboard interaction for a ListBox.

1. Add the below code in your page to render the ListBox

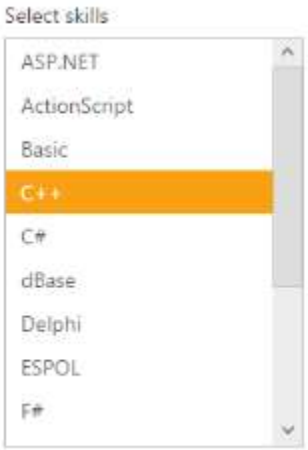
ASPX-CS

```
// Add the following code in View page to configure ListBox widget
<div id="control">
<h5 class="ctrllabel">
Select a skill
</h5>
@Html.EJ().ListBox("listBoxSample").Datasource((IEnumerable<ug_listbox.contr
ollers.skillset>)ViewBag.datasource).ListBoxFields(df
=> df.Text("text"))
</div>
```

JS

```
// Render ListBox control
<script type="text/javascript">
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
var target = $('#listBoxSample').data("ejListBox");
target.selectItemByIndex(1);
$("#listBoxSample_container").focus();
}
});
</script>
```

2. Run the sample, press Alt + J to focus in the ListBox widget that enables it and you can navigate using arrow keys.



Copyright © 2001 - 2020 Syncfusion Inc.

How to

Render ListBox from Code behind

ListBox can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of ListBox properties in the controller.

C#

```
//Namespace to get the JavaScript (Listbox) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class ListBoxController: Controller
{
public ActionResult ListBoxFeatures()
{
// initialize data
List<Bikes> bike = new List<Bikes>();
bike.Add(new Bikes { id = "bk1", text = "Aachen RTR" });
bike.Add(new Bikes { id = "bk2", text = "CBR 150-R" });
bike.Add(new Bikes { id = "bk3", text = "CBZ Xtreme" });
bike.Add(new Bikes { id = "bk4", text = "Discover" });
bike.Add(new Bikes { id = "bk5", text = "Dazzler" });
bike.Add(new Bikes { id = "bk6", text = "Flame" });
bike.Add(new Bikes { id = "bk7", text = "Faze" });
bike.Add(new Bikes { id = "bk8", text = "FZ-S" });
bike.Add(new Bikes { id = "bk9", text = "Pulsar" });
bike.Add(new Bikes { id = "bk10", text = "Shine" });
bike.Add(new Bikes { id = "bk11", text = "R15" });
bike.Add(new Bikes { id = "bk12", text = "Unicorn" });
//Initializing the Listbox model
ListBoxProperties ListObj = new ListBoxProperties();
//Initializing the datasource and other properties
ListObj.DataSource = bike;
ListObj.ListBoxFields.Text = "text";
ListObj.ListBoxFields.Value = "id";
ListObj.AllowMultiSelection = true;
//Passing Listbox properties using the ViewData
ViewData["ListModel"] = ListObj;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Binding the ListBox properties passed via



ViewData from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().ListBox("bikeList", (Syncfusion.JavaScript.Models.
ListBoxProperties)ViewData["ListModel"]).Render();
}
```

ListView

ListView

The EssentialStudio for ASP.NET MVCListView widget builds interactive list view interface. This control allows you to select an item from a list-like interface and display a set of data items in different layouts or views. Lists are used for displaying data, data navigation, result lists, and data entry.

Key Features

- AJAX Load: Loads AJAX content in the ListView content.
- Data binding: Supports Data binding with JSON data and remote data.
- Template support: The ListView supports template content.
- Group List: The ListView supports group item list.
- Check List: The ListView supports check list.

Getting Started

This section explains briefly on how to create a ListView control in your application.

Create your first ListView in MVC

The Essential StudioListView widget builds an interactive list view interface. This control allows you to select an item from a list-like interface and provides the infrastructure to display a set of data items in different layouts or views. Lists display data, data navigation, result lists, and data entry.

List view Control

Copyright © 2001 - 2020 Syncfusion Inc.

The following steps guide you to add a ListView control in a MVC application.

Create a simple ListView

1. You can create a MVC Project and add necessary Dll's and Scripts with the help of the given MVC-Getting Started Documentation.
2. You need to add the class in the Models. Define the Class with key and text field. Then create a List of that class and add the data.

C#

```
public class ListData
{
public string texts { get; set; }
}
public static class ListDataModal
{
public static List<ListData> listTempSource = new List<ListData>();
public static List<ListData> setListDataSource()
{
listTempSource.Add(new ListData { texts = "Inbox" });
listTempSource.Add(new ListData { texts = "VIP" });
listTempSource.Add(new ListData { texts = "Drafts" });
listTempSource.Add(new ListData { texts = "Sent" });
listTempSource.Add(new ListData { texts = "Junk" });
listTempSource.Add(new ListData { texts = "Trash" });
listTempSource.Add(new ListData { texts = "All mails" });
listTempSource.Add(new ListData { texts = "Mail" });
return listTempSource;
}
}
```

In the controller page, you need to pass the model class to the corresponding view.

C#

```
public ActionResult Index()
{
return View(ListDataModal.setListDataSource());
}
```

In the View page, add ListView helper and map the Local data list to corresponding DataSource and ListviewFields. You need to refer the model class at the top of the page.

ASPX-CS

```
@model List<ApplicationName.Models.ListData>
@{
@Html.EJ().ListView("Listview").ShowHeader(false).Height(206).Width(400).Dat
aSource(Model).FieldSettings(f => { f.Text("texts"); });
}
```

Run the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



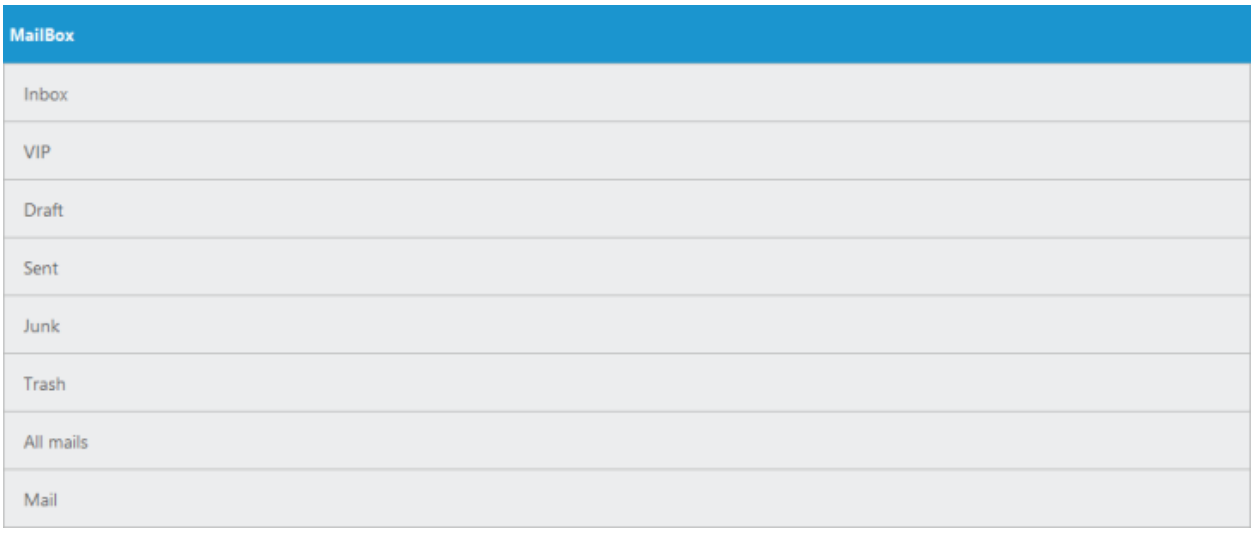
Add header

You can add a header for ListView using ShowHeader property. Refer to the following code example.

ASPX-CS

```
<!—Add Listview control with header -->
@model List<ApplicationName.Models.ListData>
@{
@Html.EJ().ListView("Listview").ShowHeader(true).HeaderTitle("Mailbox").Heig
ht(206).Width(400).DataSource(Model).FieldSettings(f => { f.Text("texts");
});
}
```

Run the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Grouped List

First Level Group List

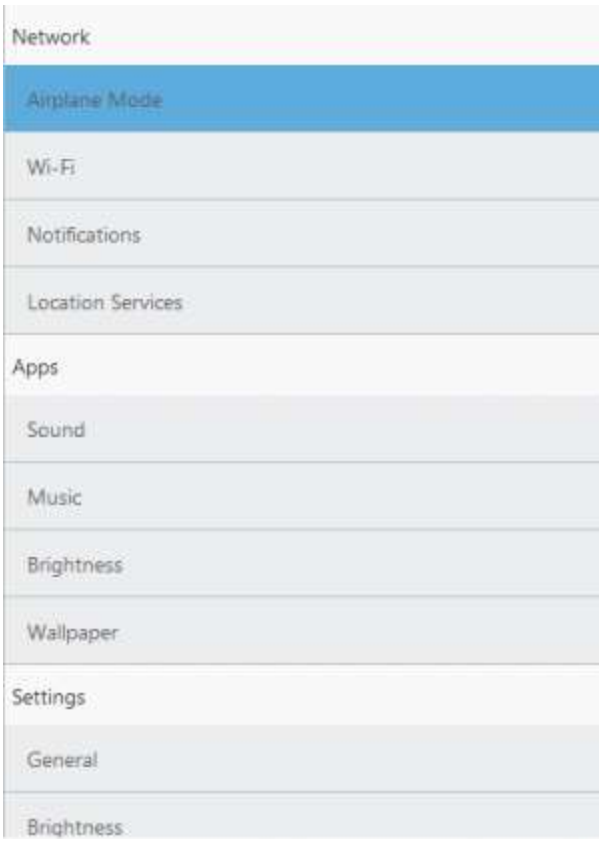
The ListView widget can make as grouped list by setting the “EnableGroupList” property as “true”. This groups the set of items listed under ul. You can identify the grouped items with the header title specified respectively.

Refer the following code example.

ASPX-CS

```
@Html.EJ().ListView("firstLevelGroupList").Width(400).EnableGroupList(true).
Groups(groupList =>
{
groupList.Add().Title("Network").Items(items =>
{
items.Add().Text("Airplane_Mode");
items.Add().Text("Wi-Fi");
items.Add().Text("Notifications");
items.Add().Text("Location_Services");
});
groupList.Add().Title("Apps").Items(items =>
{
items.Add().Text("Sound");
items.Add().Text("Brightness");
items.Add().Text("Wallpaper");
});
groupList.Add().Title("Settings").Items(items =>
{
items.Add().Text("General");
items.Add().Text("Brightness");
items.Add().Text("Wallpaper");
});
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



First Level Group List

Nested Child Group List

While selecting a list item that is grouped, you can also render another set of list items. This is achieved by defining the desired child item list within the list containing ”PrimaryKeyValue”. This PrimaryKeyValue property relates the parent child for identifying its appropriate child when clicking on the parent list item.

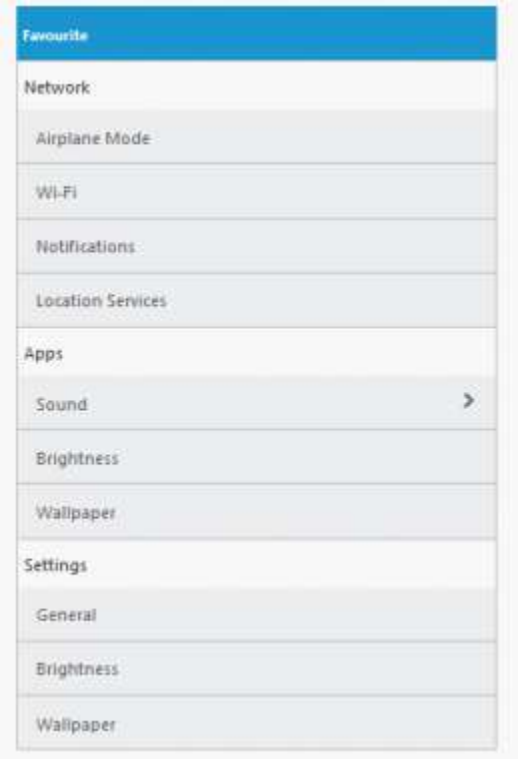
Refer the following code examples.

ASPX-CS

```
@Html.EJ().ListView("groupListSample").Width(400).ShowHeader(true).HeaderTit
le("Favorites").EnableGroupList(true).Groups(groupList =>
{
groupList.Add().Title("Network").Items(items =>
{
items.Add().Text("Airplane_Mode");
items.Add().Text("Wi-Fi");
items.Add().Text("Notifications");
items.Add().Text("Location_Services");
});
groupList.Add().Title("Apps").Items(items =>
{
items.Add().Text("Sound").PrimaryKey(1).Children(child =>
{
child.Add().Text("Ring Tone");
child.Add().Text("Message Tone");
child.Add().Text("Notification Tone");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
items.Add().Text("Brightness");
items.Add().Text("Wallpaper");
});
groupList.Add().Title("Settings").Items(items =>
{
items.Add().Text("General");
items.Add().Text("Brightness");
items.Add().Text("Wallpaper");
});
})
```



Nested Child Group List

Selection

MultiSelection

ListView has a checklist feature that is used to select multiple list items at the same time in the ListView. For this, set EnableCheckMark to “true”.

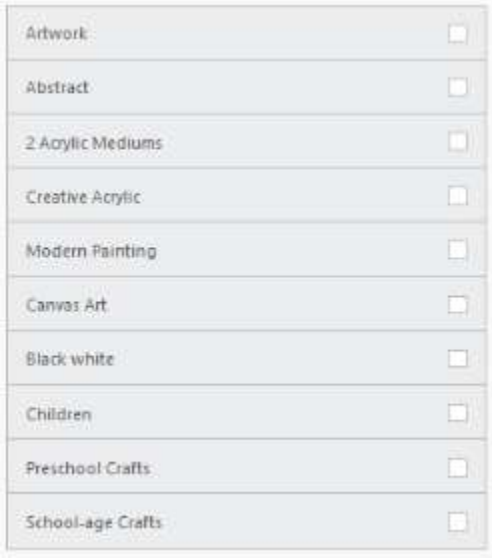
Refer the following code examples.

ASPX-CS

```
@Html.EJ().ListView("lb").EnableCheckMark(true).Width(400).Items(items =>
{
items.Add().Text("ArtWork");
items.Add().Text("Abstract");
items.Add().Text("2 Acrylic Mediums");
items.Add().Text("Creative Acrylic");
items.Add().Text("Modern Painting");
```

```
items.Add().Text("Canvas Art");
items.Add().Text("Black white");
items.Add().Text("Children");
items.Add().Text("Preschool Crafts");
items.Add().Text("School-age Crafts");
})
```

Screenshot:



Enable Check Mark

PreventSelection

When selecting a specific list item, it is highlighted with an active color. PreventSelection property is used to prevent this behavior by setting it to “true”.

Note: When the click or select action is completed, the highlight is undone automatically even when the property is set to “false”.

Refer the following code examples.

ASPX-CS

```
@Html.EJ().ListView("lb").Width(400).PreventSelection(true).Items(items =>
{
items.Add().Text("ArtWork");
items.Add().Text("Abstract");
items.Add().Text("2 Acrylic Mediums");
items.Add().Text("Creative Acrylic");
items.Add().Text("Modern Painting");
items.Add().Text("Canvas Art");
items.Add().Text("Black white");
items.Add().Text("Children");
items.Add().Text("Preschool Crafts");
items.Add().Text("School-age Crafts");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

PersistSelection

PersistSelection property is used to highlight the selected item in the ListView control even after touch end happens. By default, the active state is removed once the touch end happens.

Refer the following code examples.

ASPX-CS

```
@Html.EJ().ListView("lb").PersistSelection(true).Width(400).Items(items =>
{
items.Add().Text("ArtWork");
items.Add().Text("Abstract");
items.Add().Text("2 Acrylic Mediums");
items.Add().Text("Creative Acrylic");
items.Add().Text("Modern Painting");
items.Add().Text("Canvas Art");
items.Add().Text("Black white");
items.Add().Text("Children");
items.Add().Text("Preschool Crafts");
items.Add().Text("School-age Crafts");
})
```

Screenshot:



Persist Selection

Customize Header

In Listview, you can enable the built-in Header support. To show or hide the Header in ListView, use the ShowHeader property. By default, ListView is rendered with the Header. You can set the title for the Header by using the HeaderTitle property.

In some cases, for the purpose of navigation, you may want to show the Back button in ListView Header. To achieve this, ShowHeaderBackButton property is used. By default, ListView is not rendered with the header back button in parent page. To customize the text shown in ListViewHeaderBack button, the property HeaderBackButtonText is used.

Refer the following code example.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().ListView("lb").Width(400).ShowHeader(true).ShowHeaderBackButton(t
rue).HeaderBackButtonText("Menu").Items(items =>
{
items.Add().Text("ArtWork");
items.Add().Text("Abstract");
items.Add().Text("2 Acrylic Mediums");
items.Add().Text("Creative Acrylic");
items.Add().Text("Modern Painting");
items.Add().Text("Canvas Art");
items.Add().Text("Black white");
items.Add().Text("Children");
items.Add().Text("Preschool Crafts");
items.Add().Text("School-age Crafts");
})
```

Screenshot:



Customize Header

Data Binding

Local Data Binding

Essential Studio WebJS ListView provides support for Data Binding. Data Binding provides a simple and consistent way for applications to present and interact with data. Elements can be bounded to data from a variety of data sources. In local data binding, the data source is written inside the program. Then it is handled by the ListView control. DataSource is used to get the data source that holds the list items.

Please refer the following code examples.

Copyright © 2001 - 2020 Syncfusion Inc.

For MVC Wrapper example, you need to create model file for data-binding. Add the following model code to a CS file and save it as ListLocalData.cs.

C#

```
public class ListLocalData
{
public string text { get; set; }
}
public static class ListLocalDataModal
{
public static List<ListLocalData> listSource = new List<ListLocalData>();
public static List<ListLocalData> setListSource()
{
listSource.Add(new ListLocalData { text = "Hot Singles"});
listSource.Add(new ListLocalData { text = "Rising Artists"});
listSource.Add(new ListLocalData { text = "Live Music"});
listSource.Add(new ListLocalData { text = "Best of 2013 So Far"});
listSource.Add(new ListLocalData { text = "100 Albums - $5 Each"});
listSource.Add(new ListLocalData { text = "Hip-Hop and R&B Sale"});
listSource.Add(new ListLocalData { text = "CD Deals"});
listSource.Add(new ListLocalData { text = "Songs"});
listSource.Add(new ListLocalData { text = "Bestselling Albums"});
listSource.Add(new ListLocalData { text = "New Releases"});
listSource.Add(new ListLocalData { text = "Bestselling Songs"});
listSource.Add(new ListLocalData { text = "Rock"});
listSource.Add(new ListLocalData { text = "Gospel" });
listSource.Add(new ListLocalData { text = "Jazz"});
listSource.Add(new ListLocalData { text = "Music Trade-In"});
listSource.Add(new ListLocalData { text = "Redeem a Gift Card"});
listSource.Add(new ListLocalData { text = "Band T-Shirts"});
listSource.Add(new ListLocalData { text = "Mobile MVC"});
return listSource;
}
public static void clearSource()
{
listSource.Clear();
}
}
```

You have to modify the controller as the model is added to the sample. You can modify the controller as follows.

C#

```
public ActionResult LocalDataBinding()
{
ListLocalDataModal.clearSource();
return View(ListLocalDataModal.setListSource());
}
```

You can use the following code example to give you the exact output.

ASPX-CS

```
@model List<ListLocalData>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{
@Html.EJ().ListView("localListView").Width(400).DataSource(Model)
}
```

Screenshot:



Local Data Binding

Remote Data Binding

Essential Studio Web MVC ListView provides support for **Remote Data Binding**.

OData

OData is a standardized protocol for creating and consuming data. You can provide the OData service URL directly to the Datasource URL property.

HTML

```
@(Html.EJ().ListView("ListView1").DataSource("http://js.syncfusion.com/ejservices/Wcf/Northwind.svc/").Query("ej.Query().from('Customers').take(10)").FieldSettings(listViewFieldSettings =>
listViewFieldSettings.Text("CustomerID")))
```

Run the code to get the following output

Copyright © 2001 - 2020 Syncfusion Inc.



OData Version 4

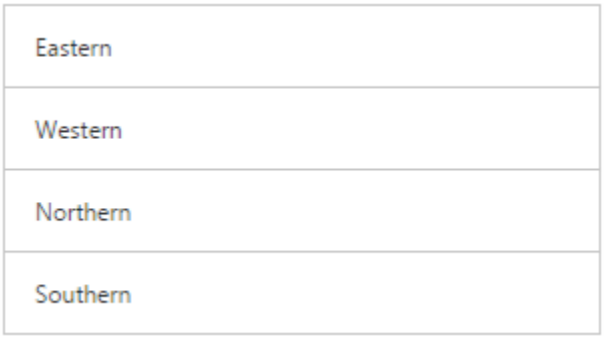
For OData Version 4 support ODataV4Adaptor should be used. By using URL property of Datasource, you can bind OData Version 4 Service link and specify Adaptor value as enum AdaptorType.ODataV4Adaptor.

For further details about OData service please refer the link.

HTML

```
@(Html.EJ().ListView("ListView1").DataSource("http://js.syncfusion.com/ejser
vices/Wcf/Northwind.svc/").Query("ej.Query().from('Customers').take(10)").Wi
dth(300).FieldSettings(listViewFieldSettings =>
listViewFieldSettings.Text("CustomerID")))
```

Run the code to get the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

URL Adaptor

URL Adaptor of **DataManager** can be used when you are required to use remote service to retrieve data. It interacts with server-side for all **DataManager** Queries and **CRUD** operations.

Now, in the following code example the data is retrieved from **Controller**.

HTML

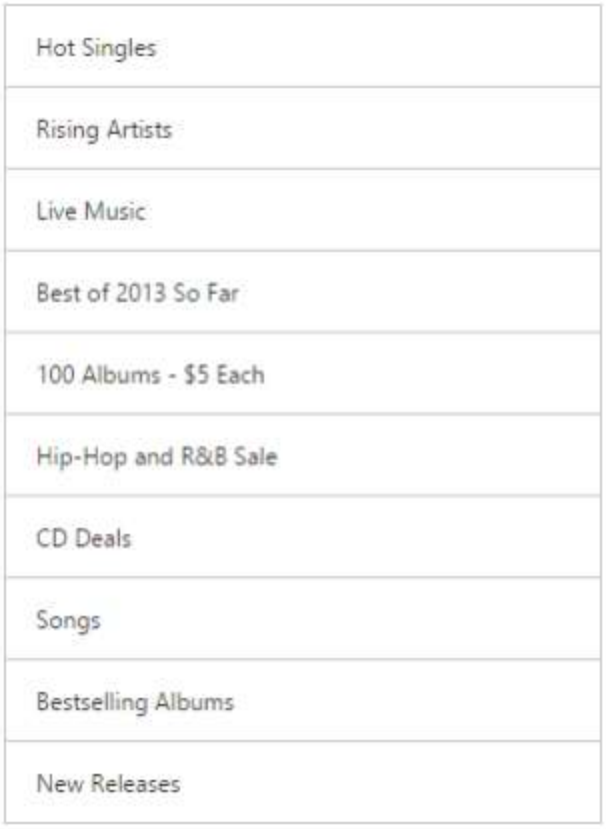
```
@(Html.EJ().ListView("ListView").DataSource(listViewDataSource =>
listViewDataSource.URL(Url.Action("DataSource",
"ListView")).Adaptor(AdaptorType.UrlAdaptor)).Query("ej.Query()").FieldSetti
ngs(listViewFieldSettings => listViewFieldSettings.Text("caption")))
```

C#

```
**Controller:**
public partial class ListviewController : Controller
{
public JsonResult DataSource()
{
IEnumerable Data = setListSource();
return Json(Data, JsonRequestBehavior.AllowGet);
}
public static List<ListLocalData> setListSource()
{
List<ListLocalData> listSource = new List<ListLocalData>();
listSource.Add(new ListLocalData { caption = "Hot Singles" });
listSource.Add(new ListLocalData { caption = "Rising Artists" });
listSource.Add(new ListLocalData { caption = "Live Music" });
listSource.Add(new ListLocalData { caption = "Best of 2013 So Far" });
listSource.Add(new ListLocalData { caption = "100 Albums - $5 Each" });
listSource.Add(new ListLocalData { caption = "Hip-Hop and R&B Sale" });
listSource.Add(new ListLocalData { caption = "CD Deals" });
listSource.Add(new ListLocalData { caption = "Songs" });
listSource.Add(new ListLocalData { caption = "Bestselling Albums" });
listSource.Add(new ListLocalData { caption = "New Releases" });
return listSource;
}
```

Run the code to get the following output

Copyright © 2001 - 2020 Syncfusion Inc.



WebAPI

WebApi Adaptor, extended from **ODataAdaptor**, of **DataManager** is used for retrieving data from WebApi service.ASP.NET Web API is a Framework for building HTTP services. You can retrieve data from ASP.NET Web API by using DataManager. Using WebApiAdaptor, you can bind WebApi service's data to ListView. The data from WebApi service must be returned as an object that has its value as data source and property "caption" with its value as dataSource's total records count.

HTML

```
@(Html.EJ().ListView("ListView").DataSource(listViewDataSource =>
listViewDataSource.URL(Url.Action("http://js.syncfusion.com/ejservices/Wcf/N
orthwind.svc/")).Adaptor(AdaptorType.WebApiAdaptor)).Query("ej.Query().from(
'Customers').take(10)").FieldSettings(listViewFieldSettings =>
listViewFieldSettings.Text("CompanyName")))
```

Run the code to get the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



FieldSettings

The FieldSettings property is used to map the DataSource field with the list item fields. In addition to the list item specific properties, the following fields are available while mapping.

FieldSettings

Properties	Definition
ParentPrimaryKey	In DB, you can relate any child item to some other item. Set here is â€˜PrimaryKeyâ€™ for the parent item. Here â€˜ParentPrimaryKeyâ€™ defines the â€˜PrimaryKeyâ€™ of some parent item to identify its parent.
Attributes	In DB, you can define your desired class name or styles for the list item through the â€˜Attributesâ€™ field.

Please refer the following code examples.

ASPX-CS

```
@model List<FieldSettingsData>
@{
@Html.EJ().ListView("localListView").Width(400).ShowHeader(true).HeaderTitle
("Music World").DataBinding(true).DataSource(Model).FieldSettings(f =>
{
f.Text("texts").PrimaryKey("primaryKeys").ParentPrimaryKey("parentPrimaryKey
s").ChildHeaderTitle("Title").ChildHeaderBackButtonText("BackIconText");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
}
```

Screenshots:



Field Settings

When you click on the parent item, it navigates to its corresponding child list item as follows.



Field Settings

Filtering

Filtering is one of the key features of ListView control. The Filtering option is added into the ListView control when the EnableFiltering property is set to “true”. This enables a simple interface to filter items from a large collection of ListView items.

Refer the following code examples.

ASPX-CS

```
@Html.EJ().ListView("lb").Width(300).EnableFiltering(true).Items(items => {
items.Add().Text("ArtWork");
items.Add().Text("Abstract");
items.Add().Text("2 Acrylic Mediums");
items.Add().Text("Creative Acrylic");
items.Add().Text("Modern Painting");
items.Add().Text("Canvas Art");
items.Add().Text("Black white");
items.Add().Text("Children");
items.Add().Text("Preschool Crafts");
items.Add().Text("School-age Crafts");
```

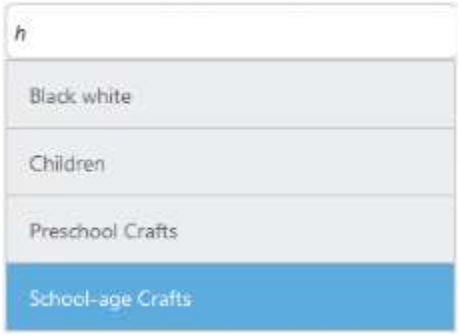
Copyright © 2001 - 2020 Syncfusion Inc.

```
})
```

Screenshot:



Enable Filtering



Enable Filtering

Dimensions

To customize the ListView dimensions, Width and Height properties are used.

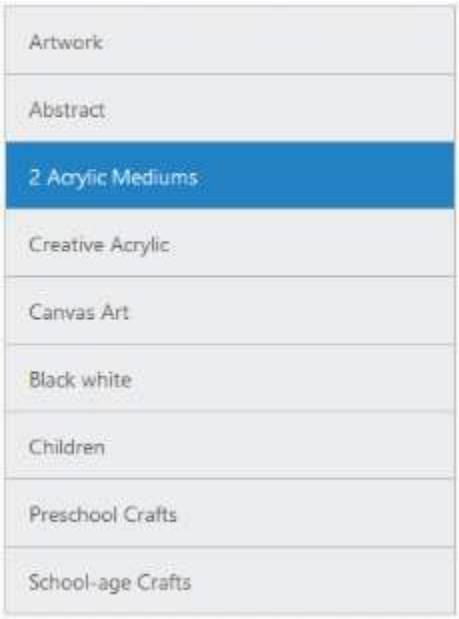
Refer to the following code examples.

ASPX-CS

```
@Html.EJMobile().ListView("lb").Height(600).Width(300).Items(items =>
{
items.Add().Text("ArtWork");
items.Add().Text("Abstract");
items.Add().Text("2 Acrylic Mediums");
items.Add().Text("Creative Acrylic");
```

```
items.Add().Text("Canvas Art");
items.Add().Text("Black white");
items.Add().Text("Children");
items.Add().Text("Preschool Crafts");
items.Add().Text("School-age Crafts");
})
```

Screenshot:



Height

Virtual Scrolling

We can load large data on demand using "AllowVirtualScrolling" property. By default, "AllowVirtualScrolling" set as boolean value of **“false”**. When it is set true, list items will be loaded on every scroll action. The number of items to be loaded per request can be specified using the “ItemRequestCount” property.By default “ItemRequestCount” value will be 5. We have provided two type of option for virtualScrolling,

Normal Mode

This mode allows you to load the list view data while scrolling i.e. each time the scroll bar is scrolled, it will send request to the server to load the data.

Continuous Mode

This mode allows you to load the list view data when the scrollbar reaches the end point. In this mode, we can specify the number of items to be loaded per request.

Please refer the following code examples.

ASPX-CS

```
@Html.EJ().ListView("localListView").Width(400).Height(300).DataSource(dataS
ource =>
dataSource.URL("http://js.syncfusion.com/ejservices/Wcf/Northwind.svc/").Cro
ssDomain(true)).Query("ej.Query().from('Customers')").FieldSettings(f =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
f.Text("CustomerID")).AllowVirtualScrolling(true).VirtualScrollMode(VirtualS
crollMode.Continuous).ItemRequestCount(8)
```

How To

Get the selected values on post back.

Single selection

Enable the property PersistSelection in order to use selection in ListView. Then use the getActiveItemText method take the selected active text through javascript and then pass it to the controller via AJAX.

HTML

```
@Html.EJ().ListView("localListView").Width(400).DataSource((IEnumerable<Cars
List>)ViewBag.datasource).FieldSettings(sf =>
sf.Text("text")).PersistSelection(true)
@Html.EJ().Button("button").ShowRoundedCorner(true).Text("Submit").ClientSid
eEvents(e => e.Click("click"))
<script>
function click(args) {
var text = $("#localListView").ejListView("getActiveItemText");
$.ajax({
type: "POST",
data: { value: text },
url: "/ListView/Features",
success: function (result) {
alert(result);
}
});
}
</script>
```

C#

```
[HttpPost]
public ActionResult Features(string value)
{
string val = value;
return Json(val, JsonRequestBehavior.AllowGet);
}
```

Multiple selection

Multiple items can be selected by using EnableCheckMark property. Hence we need to use the getCheckedItemsText method to take all the checked items and then pass it to the controller using javascript AJAX.

HTML

```
@Html.EJ().ListView("localListView").Width(400).DataSource((IEnumerable<Cars
List>)ViewBag.datasource).EnableCheckMark(true).FieldSettings(sf =>
sf.Text("text"))
@Html.EJ().Button("button").ShowRoundedCorner(true).Text("Submit").ClientSid
eEvents(e => e.Click("click"))
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function click1(args) {
var text = $("#localListView").ejListView("getCheckedItemsText");
$.ajax({
type: "POST",
data: { value: text },
url: "/ListView/Features",
success: function (result) {
alert(result);
}
});
}
</script>
```

C#

```
[HttpPost]
public ActionResult Features(object value)
{
object val = value;
return Json(val, JsonRequestBehavior.AllowGet);
}
```

Render ListView from Code behind

ListView can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side

The following code illustrates the initialization of ListView properties in the controller.

C#

```
//Namespace to get the JavaScript (ListView) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class ListViewController: Controller
{
public ActionResult ListviewFeatures()
{
//Initializing the Listbox model
ListViewProperties ListviewObj = new ListViewProperties();
//Initializing the data as items and other properties
ListViewProperties ListviewObj = new ListViewProperties();
ListviewObj.ShowHeader = false;
ListviewObj.Width = 200;
List<ListViewItem> obj = new List<ListViewItem>();
obj.Add(new ListViewItem() { Text = "Home" });
obj.Add(new ListViewItem() { Text = "Profile" });
obj.Add(new ListViewItem() { Text = "Photos" });
obj.Add(new ListViewItem() { Text = "Location" });
ListviewObj.Items = obj;
ListviewObj.EnableCheckMark = true;
//Passing Listview properties using the ViewData
ViewData["ListModel"] = ListObj;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

Binding the ListView properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().ListView("Listview",
(Syncfusion.JavaScript.Models.ListViewProperties)ViewData["ListViewModel"]).
Render();
}
```

Maps

Maps

A Map is a graphical representation of geographical data. It is used to represent the statistical data of a particular geographical area on Earth. Using pan and zoom, the maps can be navigated. Maps supports enhanced data visualization with bubbles and labels using data bound to the map.

Use Case Scenarios

Some of the Maps control use case scenarios:

- To visualize weather data.
- To visualize geographical statistics information.
- To visualize the density or availability of resources in an area.
- To visualize political information.
- To visualize the layout of a building.

Key Features

- Layers - Map is maintained through Layers and it can accommodate one or more layers.
- Map Providers Support - Syncfusion Map support map providers such as OpenStreetMap that can be added to any layers in maps.
- GeoJSON Data Input - The GeoJSON standard allows you to plot your own shapes using customized data to your maps. You can obtain some of the GeoJSON standard shape information from GEOJSON format shapes that can be used as data input for maps.
- Map Elements Customization - Map contains a set of elements, including shapes, bubbles, markers, legend, labels and data items that can be visualized with customized appearance showing additional information on the map using bound data.
- User Interaction - Options like zooming, panning and map selection enables the effective interaction on map elements.

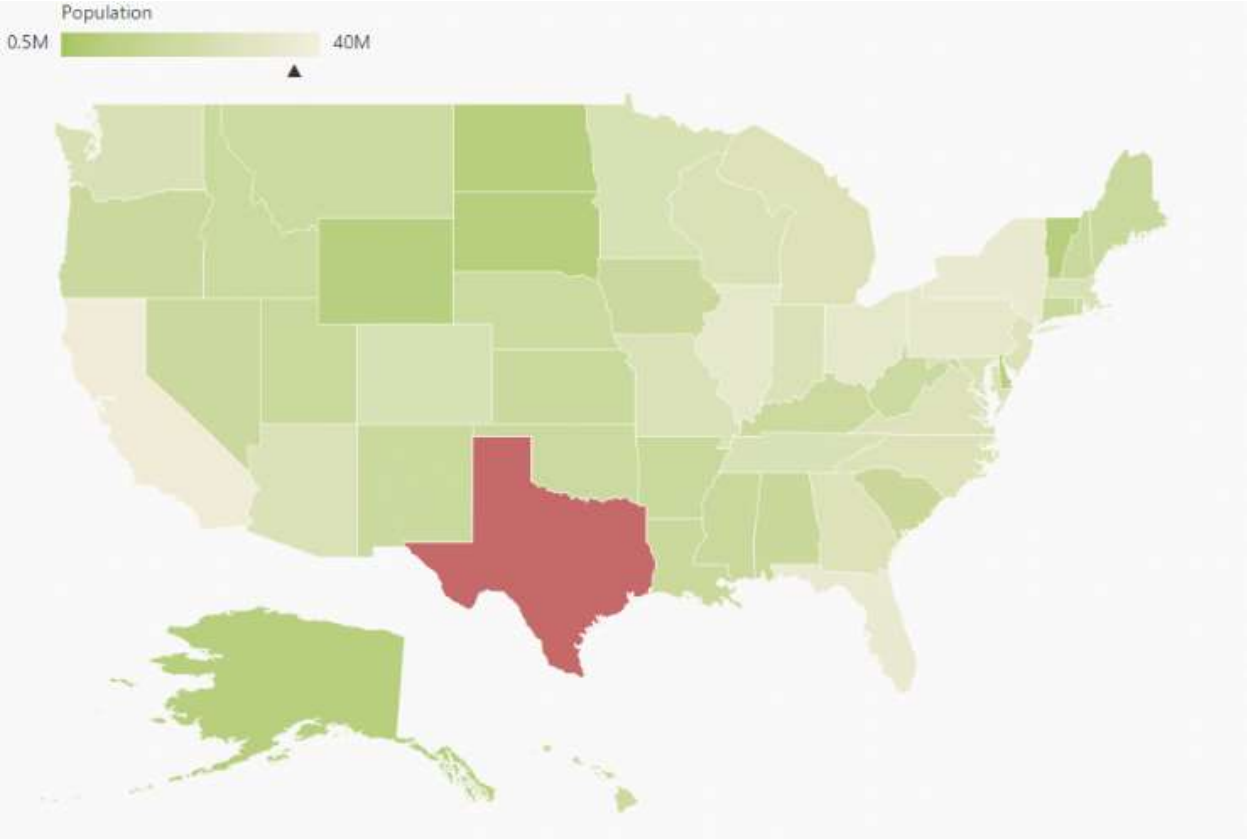
Getting Started

This section explains briefly about how to create Maps in your application with ASP.NET MVC.

Create your first Map in MVC

You can configure an Essential ASP.NET MVC Map with simple steps. In this example, you can learn how to configure USA population map with customized appearance and tooltip.

Copyright © 2001 - 2020 Syncfusion Inc.



Create a simple MVC Application

To begin, create a new ASP.NET MVC project Razor application. For creating a MVC Project and adding necessary Dll's and Scripts, refer MVC-Getting Started Documentation.

Preparing Shape Data

The Shape Data collection describing geographical shape information can be obtained from GEOJSON format shapes.

In this example, USA shape is used as shape data by utilizing the United States of America.json file in the following folder structure obtained from downloaded Maps_GeoJSON folder.

..\ Maps_GeoJSON\All Countries with States

You can store the United States of America.json file in App_Data folder as usa.json. Then read the complete contents in "usa.json" file and assign to new MapData object in "MapController.cs" in the following folder location,

~/Controller/MapController.cs

C#

```
public ActionResult Map()
{
ViewData["mapdata"] = GetUSMap();
return View();
}
public object GetUSMap()
{
```

```
string data =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
JavaScriptSerializer ser = new JavaScriptSerializer();
ser.MaxJsonLength = int.MaxValue;
return new MapData(data);
}
```

Prepare Data Source

The dataSource is populated with USA population data inside the controller relative to shape data. For better understanding, “MapController.cs” is populated with data of USA Population in “MapController.cs”.

C#

```
public ActionResult Map()
{
ViewData["mapdata"] = GetUSMap();
ViewData["datasource"] = GetUSPopulationData();
return View();
}
public object GetUSMap()
{
string data =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
JavaScriptSerializer ser = new JavaScriptSerializer();
ser.MaxJsonLength = int.MaxValue;
return new MapData(data);
}
public List<CountyPopulationData> GetUSPopulationData()
{
List<CountyPopulationData> populationData = new List<CountyPopulationData>
{
new CountyPopulationData(){ Name= "California", Population=38332521},
new CountyPopulationData(){ Name= "Texas", Population=26448193},
new CountyPopulationData(){ Name= "New York", Population=19651127},
new CountyPopulationData(){ Name= "Florida", Population=19552860},
new CountyPopulationData(){ Name= "Illinois", Population=12882135},
new CountyPopulationData(){ Name= "Pennsylvania", Population=12773801},
new CountyPopulationData(){ Name= "Ohio", Population=11570808},
new CountyPopulationData(){ Name= "Georgia", Population=9992167},
new CountyPopulationData(){ Name= "Michigan", Population=9895622},
new CountyPopulationData(){ Name= "North Carolina", Population=9848060},
new CountyPopulationData(){ Name= "New Jersey", Population=8899339},
new CountyPopulationData(){ Name= "Virginia", Population=8260405},
new CountyPopulationData(){ Name= "Washington", Population=6971406},
new CountyPopulationData(){ Name= "Massachusetts", Population=6692824},
new CountyPopulationData(){ Name= "Arizona", Population=6626624},
new CountyPopulationData(){ Name= "Indiana", Population=6570902},
new CountyPopulationData(){ Name= "Tennessee", Population=6495978},
new CountyPopulationData(){ Name= "Missouri", Population=6044171},
new CountyPopulationData(){ Name= "Maryland", Population=5928814},
new CountyPopulationData(){ Name= "Wisconsin", Population=5742713},
new CountyPopulationData(){ Name= "Minnesota", Population=5420380},
new CountyPopulationData(){ Name= "Colorado", Population=5268367},
new CountyPopulationData(){ Name= "Alabama", Population=4833722},
new CountyPopulationData(){ Name= "South Carolina", Population=4774839},
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
new CountyPopulationData(){ Name= "Louisiana", Population=4625470},
new CountyPopulationData(){ Name= "Kentucky", Population=4395295},
new CountyPopulationData(){ Name= "Oregon", Population=3930065},
new CountyPopulationData(){ Name= "Oklahoma", Population=3850568},
new CountyPopulationData(){ Name= "Puerto Rico", Population=3615086},
new CountyPopulationData(){ Name= "Connecticut", Population=3596080},
new CountyPopulationData(){ Name= "Iowa", Population=3090416},
new CountyPopulationData(){ Name= "Mississippi", Population=2991207},
new CountyPopulationData(){ Name= "Arkansas", Population=2959373},
new CountyPopulationData(){ Name= "Utah", Population=2900872},
new CountyPopulationData(){ Name= "Kansas", Population=2893957},
new CountyPopulationData(){ Name= "Nevada", Population=2790136},
new CountyPopulationData(){ Name= "New Mexico", Population=2085287},
new CountyPopulationData(){ Name= "Nebraska", Population=1868516},
new CountyPopulationData(){ Name= "West Virginia", Population=1854304},
new CountyPopulationData(){ Name= "Idaho", Population=1612136},
new CountyPopulationData(){ Name= "Hawaii", Population=1404054},
new CountyPopulationData(){ Name= "Maine", Population=1328302},
new CountyPopulationData(){ Name= "New Hampshire", Population=1323459},
new CountyPopulationData(){ Name= "Rhode Island", Population=1051511},
new CountyPopulationData(){ Name= "Montana", Population=1015165},
new CountyPopulationData(){ Name= "Delaware", Population=925749},
new CountyPopulationData(){ Name= "South Dakota", Population=844877},
new CountyPopulationData(){ Name= "Alaska", Population=735132},
new CountyPopulationData(){ Name= "North Dakota", Population=723393},
new CountyPopulationData(){ Name= "District of Columbia",
Population=646449},
new CountyPopulationData(){ Name= "Vermont", Population=626630},
new CountyPopulationData(){ Name= "Wyoming", Population=582658}
};
return populationData;
}
public class CountyPopulationData
{
public string name;
public string Name
{
get { return name; }
set { name = value; }
}
public double population;
public double Population
{
get { return population; }
set { population = value; }
}
}
```

You can refer to shape data and datasource as illustrated in the following “Map.cshtml”,

ASPX-CS

```
@{
var mapData = ViewData["mapdata"];
var dataSource = ViewData["datasource"];
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Initialize Map

- Create a tag and set the height and width to determine the map size.

ASPX-CS

```
<div style="width: 900px; height: 600px;"></div>
```

- Add the following code sample in “Map.cshtml” file, to create the Map control in the View page.

ASPX-CS

```
<div style="width:900px; height:600px;">
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData).Add();
})
)
</div>
```

- The Final Map.cshtml file appears as follows.

ASPX-CS

```
@{
var mapData = ViewData["mapdata"];
var datasource = ViewData["datasource"];
}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/ej.widgets.all.min.js"></script>
<script src="http://cdn.jsdelivr.net/jsrender/1.0pre35/jsrender.min.js"
type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.unobtrusive.min.js"
type="text/javascript"></script>
@using Syncfusion.JavaScript.DataVisualization.Models
</head>
<body>
<div style="height:600px;width:900px;">
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData).Add();
})
)
</div>
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</body>
</html>
```

The above code renders a map with default properties and shape input provided through data in layers.

Data Binding in Map

The following properties in shape layers is used for binding data in Maps control.

- DataSource
- ShapeDataPath
- ShapePropertyPath

DataSource

The DataSource property accepts collection values as input. For example, you can provide the list of objects as input.

Shape Data Path

The ShapeDataPath property is used to refer the data ID in DataSource. For example, population MapData contains data ids ‘Name’ and ‘Population’. The ShapeDataPath and ShapePropertyPath properties are related to each other (refer to ShapePropertyPath for more details).

Shape Property Path

The ShapePropertyPath property is similar to the ShapeDataPath that refers the column name in the Data property of shape layers to identify the shape. When the values of the ShapeDataPath property in

Copyright © 2001 - 2020 Syncfusion Inc.

the DataSource property and the value of ShapePropertyPath in the Data property match, then the associated object from the DataSource is bound to the corresponding shape.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("name")
.ShapePropertyPath("name")
.DataSource(datasource)
.Add();
})
)
```

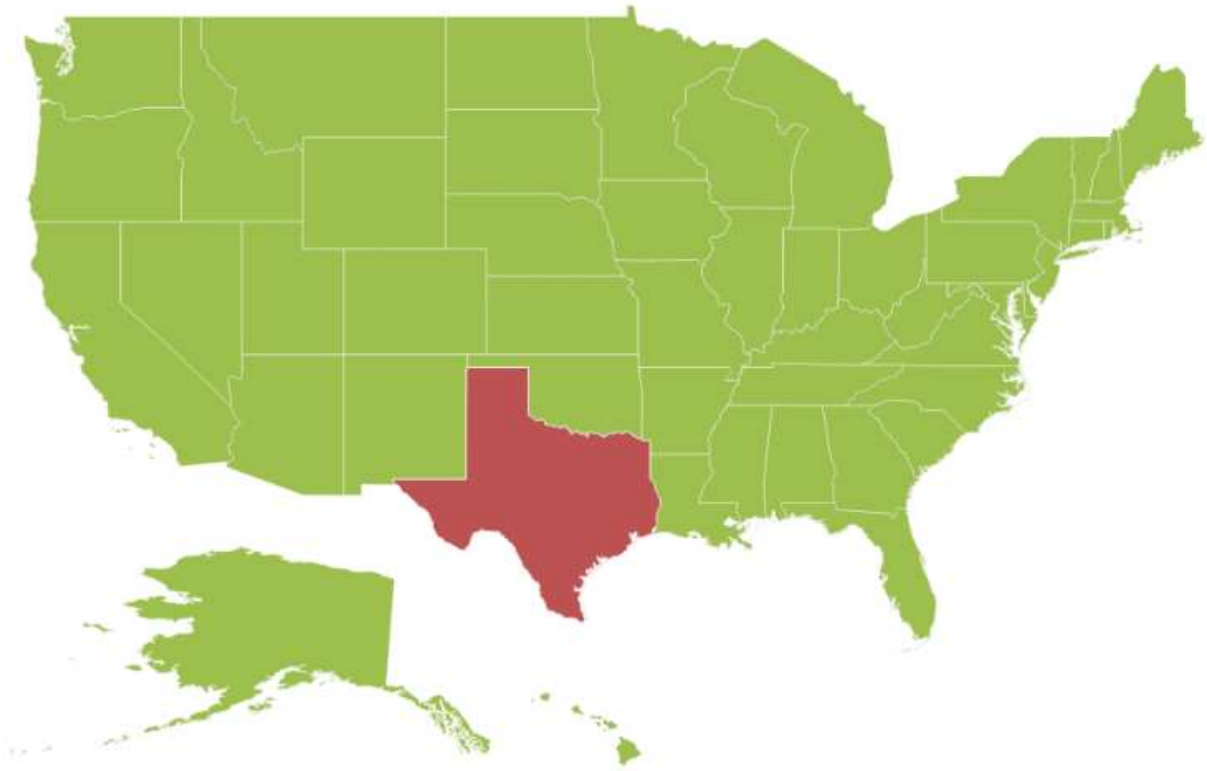
Customize Map Appearance

You can customize the shape's color by using Fill, Stroke and StrokeThickness properties in ShapeSettings.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("name")
.ShapePropertyPath("name")
.DataSource(datasource)
.EnableSelection(false)
.EnableMouseHover(true)
.ShapeSettings(ss  =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White")
.HighlightStroke ("White")
.HighlightColor("#BC5353")
.HighlightBorderWidth(1);
})
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing Map Appearance by Range

The Range color mapping is used to differentiate the shape's fill based on its underlying value and color ranges. The From and To properties defines the value ranges and the GradientColors property defines the equivalent color ranges respective to their value ranges.

Note: The EnableGradient property value should be true to apply gradient colors for maps.

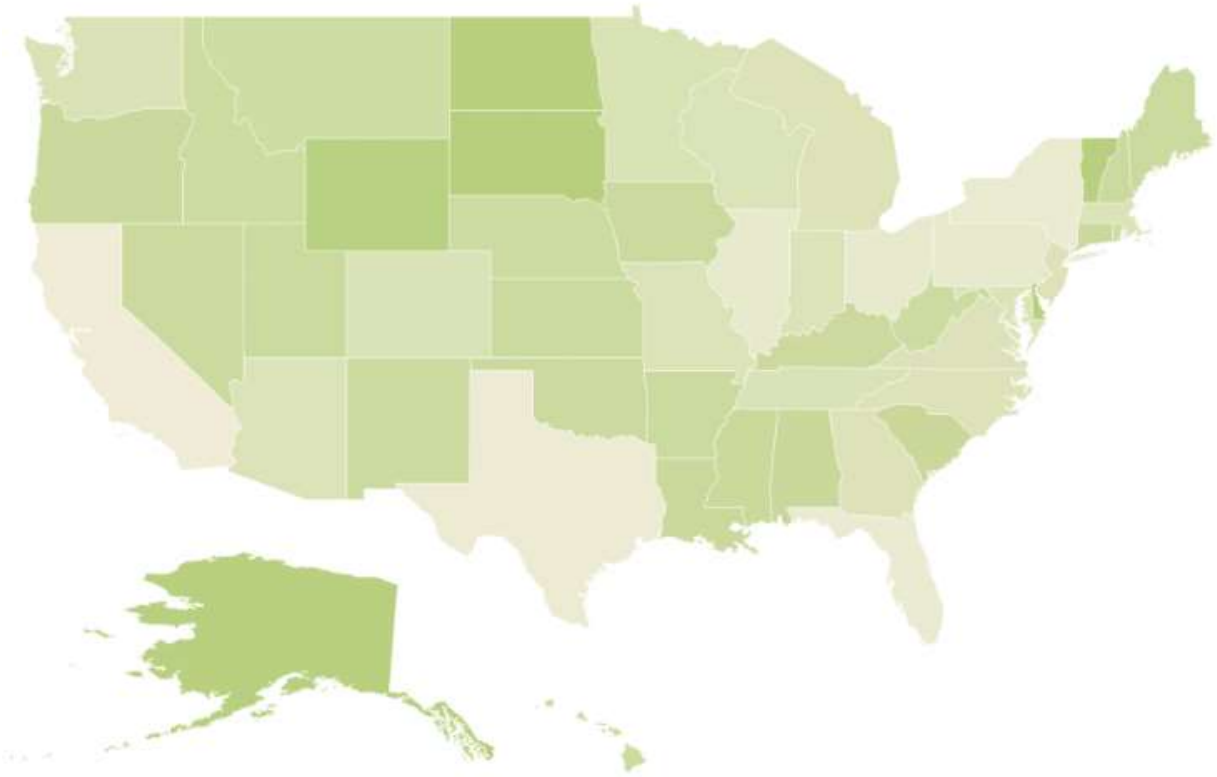
ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("name")
.ShapePropertyPath("name")
.DataSource(datasource)
.ShowMapItems(false)
.EnableSelection(false)
.EnableMouseHover(true)
.ShapeSettings(ss  =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White")
.HighlightStroke("White")
.HighlightColor("#BC5353")
.HighlightBorderWidth(1)
.ValuePath("population")
.EnableGradient(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.RangeColorMappings(cm =>
{
cm.From(500000).To(1000000).GradientColors(new List<string> { "#9CBF4E",
"#B8CE7B" }).Add();
cm.From(1000001).To(5000000).GradientColors(new List<string> { "#B8CE7B",
"#CBD89A" }).Add();
cm.From(5000001).To(10000000).GradientColors(new List<string> { "#CBD89A",
"#DEE2B9" }).Add();
cm.From(10000001).To(40000000).GradientColors(new List<string> { "#DEE2B9",
"#F1ECD8" }).Add();
});
})
.Add();
})
)
```

The following screenshot illustrates a Map with gradient color property enable.



Enable Tooltip

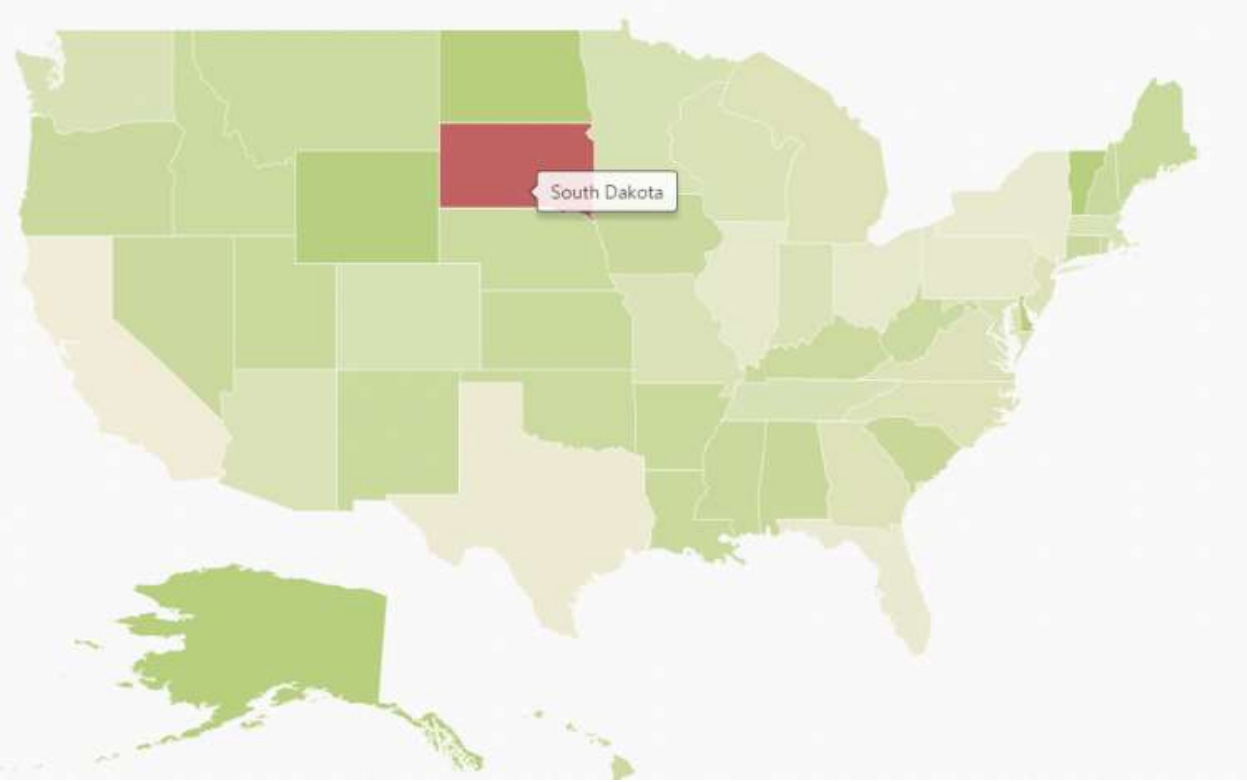
The tooltip is displayed only when ShowTooltip is set to ‘true’ in the shape layers. By default, it takes the property of the bound object that is referred in the ValuePath and displays its content on hovering the corresponding shape. The TooltipTemplate property is used for customizing the template for tooltip.

ASPX-CS

```
@(Html.EJ().Map("container")
```

```
.Layers(layer =>
{
layer.ShapeSettings(ss  =>
{
ss.ValuePath("name")
})
.ShowTooltip(true);
})
)
```

The following screenshot illustrates a map control displaying a Tooltip.



Legend

A Legend can be made visible by setting the ShowLegend property in LegendSetting.

Interactive Legend

The legends can be made interactive with an arrow mark indicating the exact range color in the legend, when the mouse hovers the corresponding shapes. You can enable this option by setting Mode property in LegendSettings value as 'Interactive'. The default value of Mode property is 'Default' to enable the normal legend.

Title

Use Title property to provide title for interactive legend.

Label

You can use LeftLabel and RightLabel property to provide left and right labels for interactive legend.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("name")
.ShapePropertyPath("name")
.DataSource(datasource)
.ShowMapItems(false)
.EnableSelection(false)
.EnableMouseHover(true)
.ShapeSettings(ss  =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White")
.HighlightStroke("White")
.HighlightColor("#BC5353")
.HighlightBorderWidth(1)
.ValuePath("population")
.EnableGradient(true)
.RangeColorMappings(cm =>
{
cm.From(500000).To(1000000).GradientColors(new List<string> { "#9CBF4E",
"#B8CE7B" }).Add();
cm.From(1000001).To(5000000).GradientColors(new List<string> { "#B8CE7B",
"#CBD89A" }).Add();
cm.From(5000001).To(10000000).GradientColors(new List<string> { "#CBD89A",
"#DEE2B9" }).Add();
cm.From(10000001).To(40000000).GradientColors(new List<string> { "#DEE2B9",
"#F1ECD8" }).Add();
});
})
.LegendSettings(ml =>
{
ml.ShowLegend(true)
.Height(15)
.Width(150)
.Position(Syncfusion.JavaScript.DataVisualization.Models.DockPosition.Toplef
t)
.Type(Syncfusion.JavaScript.DataVisualization.Models.LegendType.Layers)
.Mode(Syncfusion.JavaScript.DataVisualization.Models.LegendMode.Interactive)
.Title("Population")
.LeftLabel("0.5 M")
.RightLabel("40 M");
})
.Add();
})
)
```

DockOnMap and Dock Position

dockOnMap determines whether the legend should be placed outside or inside the map bounds.

dockPosition determines the legend placement and it is valid only when dockOnMap is true.

Copyright © 2001 - 2020 Syncfusion Inc.

Icon

To get `icon` value for legend setting. You can set `iconHeight` and `iconWidth` property for legend settings.

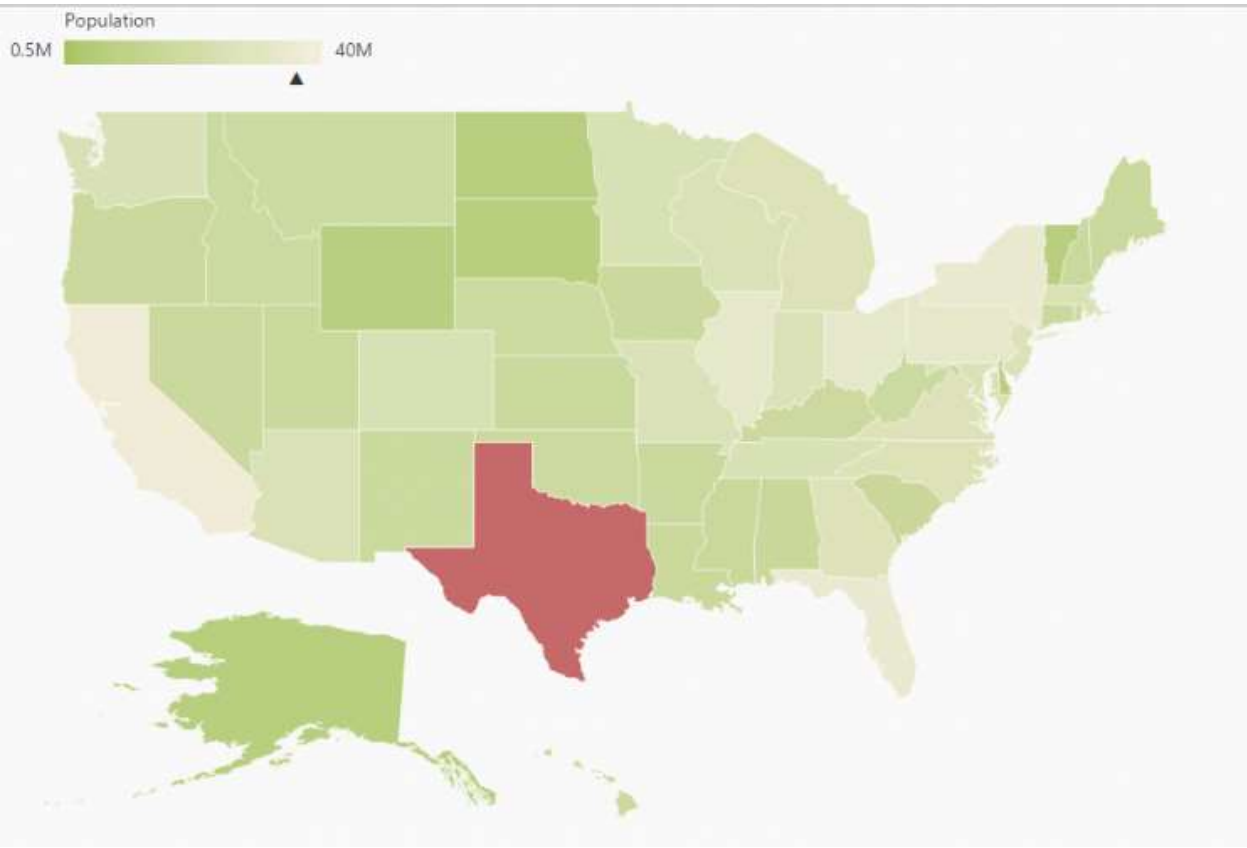
label Orientation

Set the orientation of legend labels by using `labelOrientation` property.

Show Labels

Enables or Disables the `showLabels`

The following screenshot illustrates a map displaying an interactive legend.



The complete code sample can be found here

Populate Data

In this section you can learn how to populate shape data for providing Data input to Map control and usage of DataSource property.

Shape Data

The Shape Data collection describing geographical shape information can be obtained from GEOJSON format shapes.

In this example, USA shape is used as shape data by utilizing the “United States of America.json” file in the following folder structure obtained from downloaded Maps_GeoJSON folder.

..\ Maps_GeoJSON\All Countries with States

Copyright © 2001 - 2020 Syncfusion Inc.

You can store the “United States of America.json” file in App_Data folder as “usa.json”. Then, read the complete contents in “usa.json” file and assign to new MapData object in “MapController.cs” in the following folder location,

~/Controller/MapController.cs

C#

```
public ActionResult Map()
{
ViewData["mapdata"] = GetUSMap();
return View();
}
public object GetUSMap()
{
string data =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
JavaScriptSerializer ser = new JavaScriptSerializer();
ser.MaxJsonLength = int.MaxValue;
return new MapData(data);
}
```

Data Binding

The Maps control supports data binding with the DataSource property in the shape layers.

Properties

The following properties in shape layers is be used for binding data in Maps control,

- Data Source
- shapeDataPath
- shapePropertyPath

Data Source

The DataSource property accepts the collection values as input. For example, you can provide the list of objects as input.

Shape Data Path

The ShapeDataPath property is used to refer the data ID in DataSource. For example, population MapData contains data ids ‘Name’ and ‘Population’. The ShapeDataPath and the ShapePropertyPath properties are related to each other (refer to ShapePropertyPath for more details).

Shape Property Path

The ShapePropertyPath property is similar to the ShapeDataPath that refers to the column name in the Data property of shape layers to identify the shape. When the values of the ShapeDataPath property in the DataSource property and the value of ShapePropertyPath in the Data property match, then the associated object from the DataSource is bound to the corresponding shape.

The Datasource is populated with JSON data relative to shape data and stored in JSON object. The USA population as datasource is used for better understanding.

For better understanding, “MapController.cs” is populated with data of USA Population in “MapController.cs” and you can refer both shape data and datasource as illustrated in the following “Map.cshtml”,

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public ActionResult Map()
{
ViewData["mapdata"] = GetUSMap();
ViewData["datasource"] = GetUSPopulationData();
return View();
}
public object GetUSMap()
{
string data =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
JavaScriptSerializer ser = new JavaScriptSerializer();
ser.MaxJsonLength = int.MaxValue;
return new MapData(data);
}
public List<CountyPopulationData> GetUSPopulationData()
{
List<CountyPopulationData> populationData = new List<CountyPopulationData>
{
new CountyPopulationData(){ Name= "California", Population=38332521},
new CountyPopulationData(){ Name= "Texas", Population=26448193},
new CountyPopulationData(){ Name= "New York", Population=19651127},
new CountyPopulationData(){ Name= "Florida", Population=19552860},
new CountyPopulationData(){ Name= "Illinois", Population=12882135},
new CountyPopulationData(){ Name= "Pennsylvania", Population=12773801},
new CountyPopulationData(){ Name= "Ohio", Population=11570808},
new CountyPopulationData(){ Name= "Georgia", Population=9992167},
new CountyPopulationData(){ Name= "Michigan", Population=9895622},
new CountyPopulationData(){ Name= "North Carolina", Population=9848060},
new CountyPopulationData(){ Name= "New Jersey", Population=8899339},
new CountyPopulationData(){ Name= "Virginia", Population=8260405},
new CountyPopulationData(){ Name= "Washington", Population=6971406},
new CountyPopulationData(){ Name= "Massachusetts", Population=6692824},
new CountyPopulationData(){ Name= "Arizona", Population=6626624},
new CountyPopulationData(){ Name= "Indiana", Population=6570902},
new CountyPopulationData(){ Name= "Tennessee", Population=6495978},
new CountyPopulationData(){ Name= "Missouri", Population=6044171},
new CountyPopulationData(){ Name= "Maryland", Population=5928814},
new CountyPopulationData(){ Name= "Wisconsin", Population=5742713},
new CountyPopulationData(){ Name= "Minnesota", Population=5420380},
new CountyPopulationData(){ Name= "Colorado", Population=5268367},
new CountyPopulationData(){ Name= "Alabama", Population=4833722},
new CountyPopulationData(){ Name= "South Carolina", Population=4774839},
new CountyPopulationData(){ Name= "Louisiana", Population=4625470},
new CountyPopulationData(){ Name= "Kentucky", Population=4395295},
new CountyPopulationData(){ Name= "Oregon", Population=3930065},
new CountyPopulationData(){ Name= "Oklahoma", Population=3850568},
new CountyPopulationData(){ Name= "Puerto Rico", Population=3615086},
new CountyPopulationData(){ Name= "Connecticut", Population=3596080},
new CountyPopulationData(){ Name= "Iowa", Population=3090416},
new CountyPopulationData(){ Name= "Mississippi", Population=2991207},
new CountyPopulationData(){ Name= "Arkansas", Population=2959373},
new CountyPopulationData(){ Name= "Utah", Population=2900872},
new CountyPopulationData(){ Name= "Kansas", Population=2893957},
new CountyPopulationData(){ Name= "Nevada", Population=2790136},
new CountyPopulationData(){ Name= "New Mexico", Population=2085287},
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
new CountyPopulationData(){ Name= "Nebraska", Population=1868516},
new CountyPopulationData(){ Name= "West Virginia", Population=1854304},
new CountyPopulationData(){ Name= "Idaho", Population=1612136},
new CountyPopulationData(){ Name= "Hawaii", Population=1404054},
new CountyPopulationData(){ Name= "Maine", Population=1328302},
new CountyPopulationData(){ Name= "New Hampshire", Population=1323459},
new CountyPopulationData(){ Name= "Rhode Island", Population=1051511},
new CountyPopulationData(){ Name= "Montana", Population=1015165},
new CountyPopulationData(){ Name= "Delaware", Population=925749},
new CountyPopulationData(){ Name= "South Dakota", Population=844877},
new CountyPopulationData(){ Name= "Alaska", Population=735132},
new CountyPopulationData(){ Name= "North Dakota", Population=723393},
new CountyPopulationData(){ Name= "District of Columbia",
Population=646449},
new CountyPopulationData(){ Name= "Vermont", Population=626630},
new CountyPopulationData(){ Name= "Wyoming", Population=582658}
};
return populationData;
}
public class CountyPopulationData
{
public string name;
public string Name
{
get { return name; }
set { name = value; }
}
public double population;
public double Population
{
get { return population; }
set { population = value; }
}
}
```

ASPX-CS

```
@{
var mapData = ViewData["mapdata"];
var dataSource = ViewData["datasource"];
}
```

The JSON object “populationData” is used as DataSource in the following code example.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("name")
.ShapePropertyPath("name")
.DataSource(datasource)
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Customization

Maps control supports color customization to determine the exact combination of colors for shapes displayed in Maps and tooltip support to display additional information of shape data.

Shapes Color Customization

The Map control highly supports the customization of the shape's color. The shape's color can be customized using the following ways:

- Using the Fill, Stroke and StrokeThickness properties.
- Color Mapping support.
- Color Palette support.

Shape Settings

The ShapeSettings defines the basic customization settings of shapes in the map.

The important property that makes an impact on shape colors is AutoFill. This AutoFill property is available in the shapeSettings.

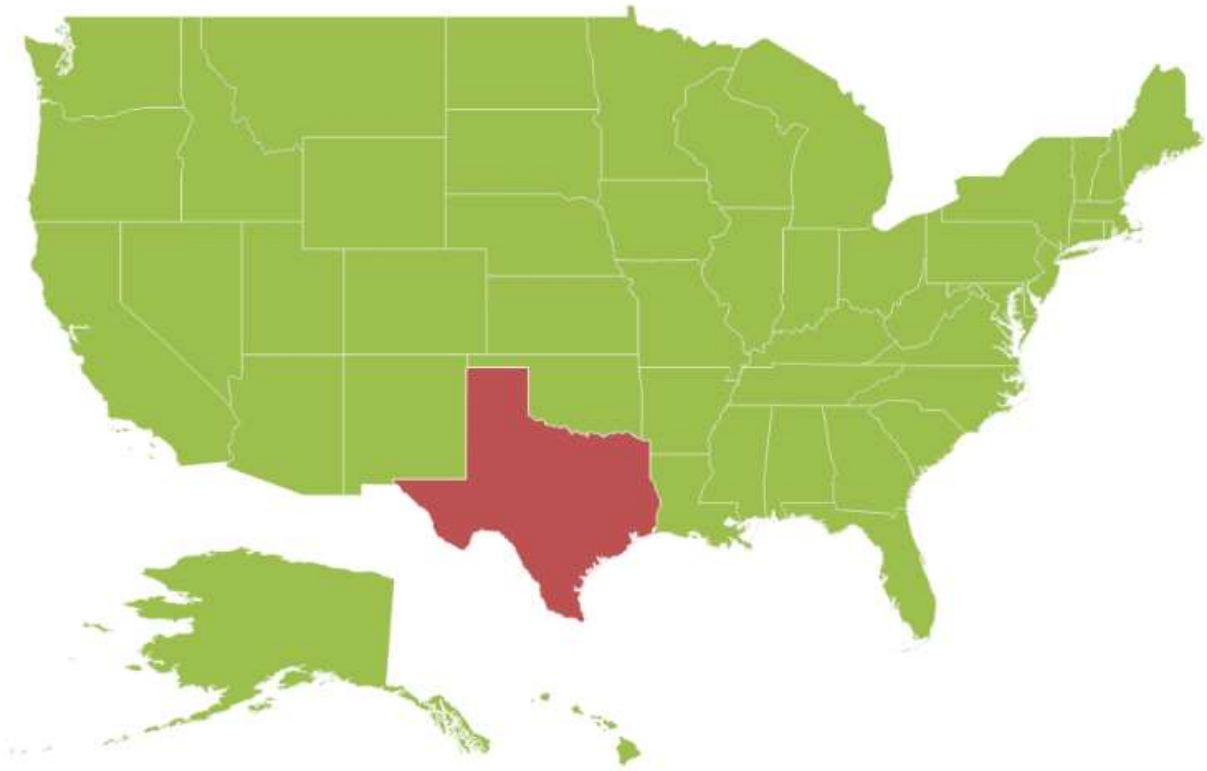
- Fill - It is used to set the fill color of the shapes in the map.
- Stroke - It is used to set the border color of the shape in the map.
- StrokeThickness - It is used to set the border thickness of the shape in the map.
- HighlightColor - It is used to set the mouse hover color for shapes in the map.
- HighlightBorderWidth - It is used to set the mouse hover border width for shapes in the map.
- SelectionColor - It is used to set the selection color for shapes in the map.

The above properties of ShapeSettings are applied only when AutoFill property value is false. By map, AutoFill property value is false.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.EnableMouseHover(true)
.ShapeSettings(ss =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White")
.HighlightStroke("White")
.HighlightColor("#BC5353")
.HighlightBorderWidth(1);
})
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Color Mapping

The Color Mapping support enables the customization of shape colors based on the underlying value of shape received from bounded data.

- ColorValuePath - It renders the field value that is to be fetched from data for each shape used for determining the shape color.
- ValuePath - It renders the field value that is to be fetched from data for each shape. This support also provides a tree map-like impact on the map UI. The various types of Color Mapping supported in maps are listed as follows.
- RangeColorMapping - It is used to differentiate the shape's fill based on its underlying value and color ranges. The properties of Range color mapping are listed in the following table.

Property Table

Property	Type	Description
From	Double	Gets or sets start value
To	Double	Gets or sets end value
Color	Color	Gets or sets the colors to be applied for specific range value containing shapes when EnableGradient property value is false.
Label	String	Gets or sets the label for legend when Mode property value is â€˜defaultâ€™.

Copyright © 2001 - 2020 Syncfusion Inc.

GradientColors	Array	Gets or sets the start point and end point gradient colors to be applied for specific range value containing shapes when EnableGradient property value is set to true.

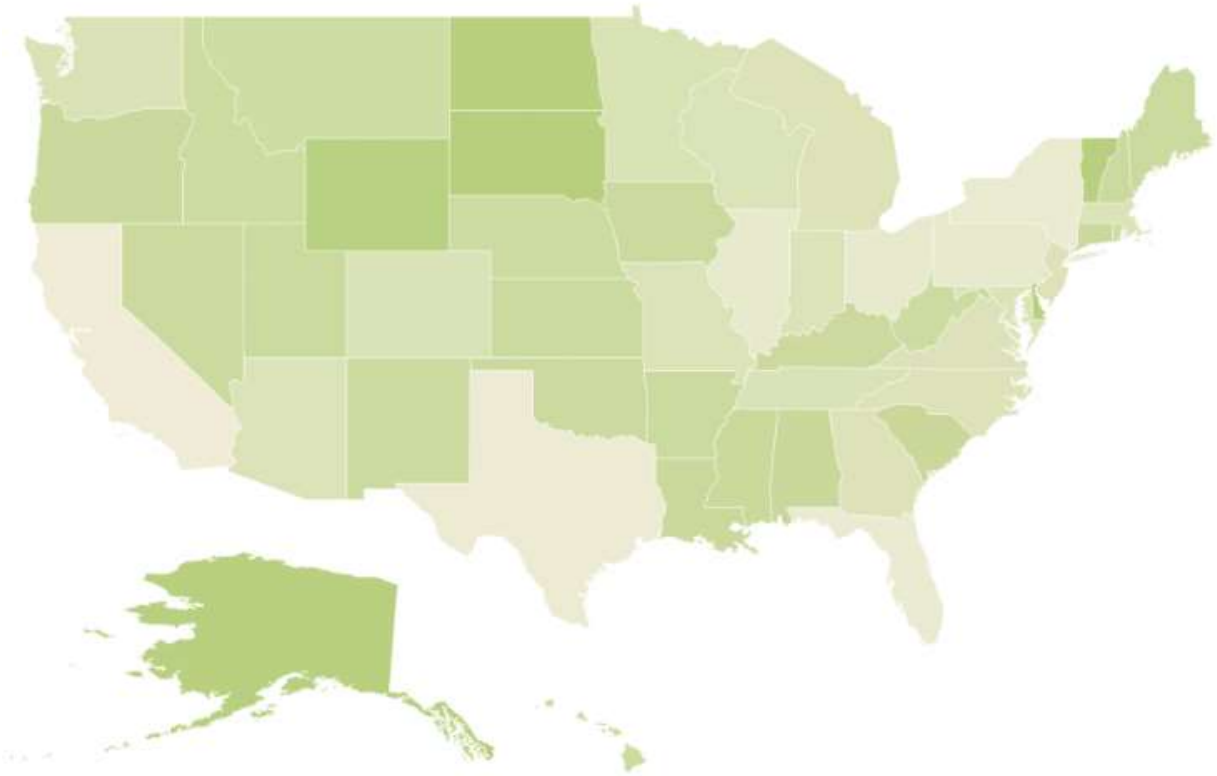
ASPX-CS

```
@(Html.EJ().Map("map")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("name")
.ShapePropertyPath("name")
.DataSource(datasource)
.ShowMapItems(false)
.ShapeSettings(ss  =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.1)
.Stroke("White")
.ValuePath("population")
.EnableGradient(true)
.RangeColorMappings(cm =>
{
cm.From(500000).To(1000000).GradientColors(new List<string> { "#9CBF4E",
"#B8CE7B" }).Add();
cm.From(1000001).To(5000000).GradientColors(new List<string> { "#B8CE7B",
"#CBD89A" }).Add();
cm.From(5000001).To(10000000).GradientColors(new List<string> { "#CBD89A",
"#DEE2B9" }).Add();
cm.From(10000001).To(40000000).GradientColors(new List<string> { "#DEE2B9",
"#F1ECD8" }).Add();
});
})
.Add();
})
)
```

When the underlying object value is 700000, then the Fill color of the corresponding shape is set between #9CBF4E and #B8CE7B.

When the underlying value is below any of the given sorted range or above the sorted range, then the Fill is set from Fill.

Copyright © 2001 - 2020 Syncfusion Inc.



- EqualColorMapping - The Equal color mapping is used to differentiate the shape's fill based on its underlying Value and Color. The properties of Equal color mapping is listed in the following table.

Property	Type	Description
Value	String	Gets or sets the value.
Color	String	Gets or sets the color for mapping.

In Equal color mapping, value property contains the values of the field set in ColorValuePath property of shape settings.

Here "MapController.cs" is populated with data of USA Election in "MapController.cs" to provide election data as input datasource.

C#

```
public ActionResult Map()
{
ViewData["datasource"] = GetElectionResults();
return View();
}
public List<ElectionData> GetElectionResults()
{
List<ElectionData> electionResults = new List<ElectionData>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{new ElectionData {State = "Alabama", Candidate = "Romney", Electors = 9 },
new ElectionData { State = "Alaska", Candidate = "Romney", Electors = 3 },
new ElectionData { State = "Arizona", Candidate = "Romney", Electors = 11 },
new ElectionData { State = "Arkansas", Candidate = "Romney", Electors = 6 },
new ElectionData { State = "California", Candidate = "Obama", Electors =
55},
new ElectionData { State = "Colorado", Candidate = "Obama", Electors = 9 },
new ElectionData { State = "Connecticut", Candidate = "Obama", Electors = 7
},
new ElectionData { State = "Delaware", Candidate = "Obama", Electors = 3 },
new ElectionData { State = "District of Columbia", Candidate = "Obama",
Electors = 3 },
new ElectionData { State = "Florida", Candidate = "Obama", Electors = 29 },
new ElectionData { State = "Georgia", Candidate = "Romney", Electors = 16 },
new ElectionData { State = "Hawaii", Candidate = "Obama", Electors = 4 },
new ElectionData { State = "Idaho", Candidate = "Romney", Electors = 4 },
new ElectionData { State = "Illinois", Candidate = "Obama", Electors = 20 },
new ElectionData { State = "Indiana", Candidate = "Romney", Electors = 11 },
new ElectionData { State = "Iowa", Candidate = "Obama", Electors = 6 },
new ElectionData { State = "Kansas", Candidate = "Romney", Electors = 6 },
new ElectionData { State = "Kentucky", Candidate = "Romney", Electors = 8 },
new ElectionData { State = "Louisiana", Candidate = "Romney", Electors = 8
},
new ElectionData { State = "Maine", Candidate = "Obama", Electors = 4 },
new ElectionData { State = "Maryland", Candidate = "Obama", Electors = 10 },
new ElectionData { State = "Massachusetts", Candidate = "Obama", Electors =
11},
new ElectionData { State = "Michigan", Candidate = "Obama", Electors = 16 },
new ElectionData { State = "Minnesota", Candidate = "Obama", Electors = 10
},
new ElectionData { State = "Mississippi", Candidate = "Romney", Electors =
6},
new ElectionData { State = "Missouri", Candidate = "Romney", Electors = 10
},
new ElectionData { State = "Montana", Candidate = "Romney", Electors = 3 },
new ElectionData { State = "Nebraska", Candidate = "Romney", Electors = 5 },
new ElectionData { State = "Nevada", Candidate = "Obama", Electors = 6 },
new ElectionData { State = "New Hampshire", Candidate = "Obama", Electors =
4},
new ElectionData { State = "New Jersey", Candidate = "Obama", Electors = 14
},
new ElectionData { State = "New Mexico", Candidate = "Obama", Electors = 5
},
new ElectionData { State = "New York", Candidate = "Obama", Electors = 29 },
new ElectionData { State = "North Carolina", Candidate = "Romney", Electors
= 15 },
new ElectionData { State = "North Dakota", Candidate = "Romney", Electors =
3},
new ElectionData { State = "Ohio", Candidate = "Obama", Electors = 18 },
new ElectionData { State = "Oklahoma", Candidate = "Romney", Electors = 7 },
new ElectionData { State = "Oregon", Candidate = "Obama", Electors = 7 },
new ElectionData { State = "Pennsylvania", Candidate = "Obama", Electors =
20},
new ElectionData { State = "Rhode Island", Candidate = "Obama", Electors =
4},
new ElectionData { State = "South Carolina", Candidate = "Romney", Electors
= 9 },
```

Copyright © 2001 - 2020 Syncfusion Inc.

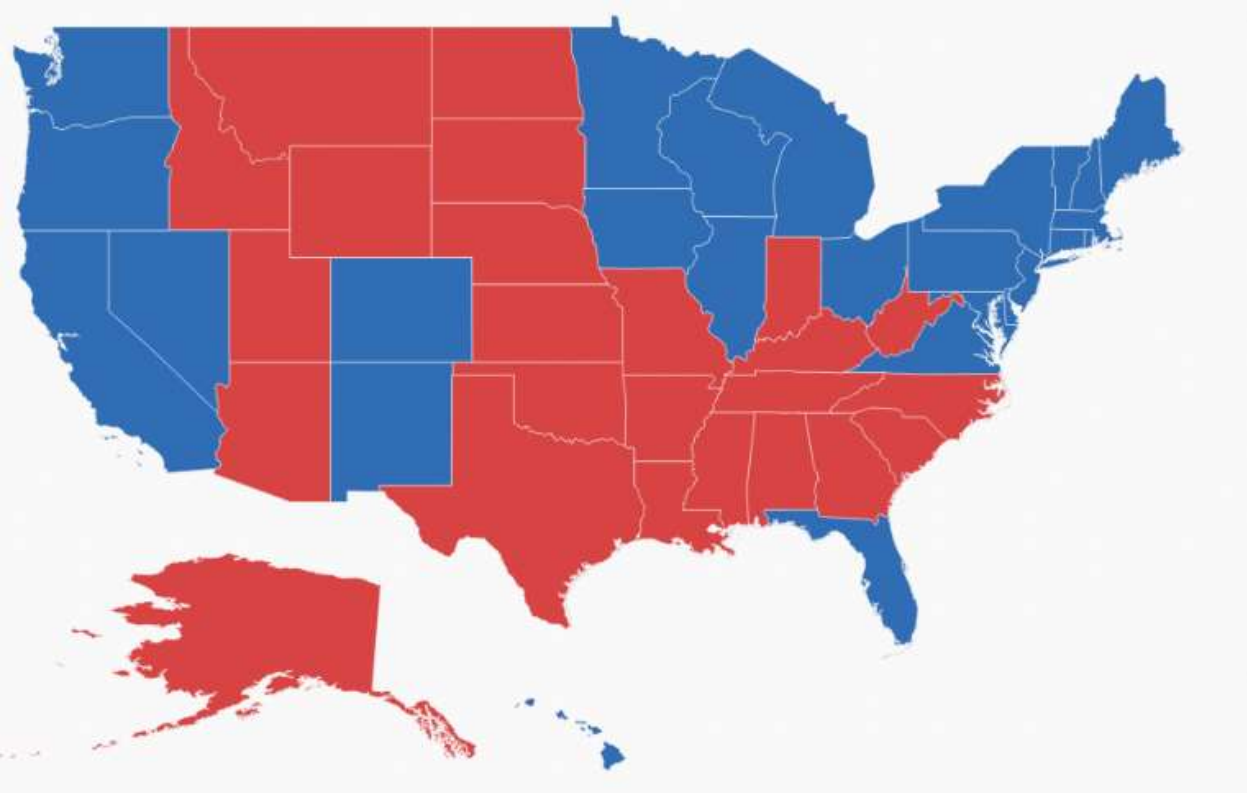
```
new ElectionData { State = "South Dakota", Candidate = "Romney", Electors =
3},
new ElectionData { State = "Tennessee", Candidate = "Romney", Electors = 11
},
new ElectionData { State = "Texas", Candidate = "Romney", Electors = 38 },
new ElectionData { State = "Utah", Candidate = "Romney", Electors = 6 },
new ElectionData { State = "Vermont", Candidate = "Obama", Electors = 3 },
new ElectionData { State = "Virginia", Candidate = "Obama", Electors = 13 },
new ElectionData { State = "Washington", Candidate = "Obama", Electors = 12
},
new ElectionData { State = "West Virginia", Candidate = "Romney", Electors =
5},
new ElectionData { State = "Wisconsin", Candidate = "Obama", Electors = 10
},
new ElectionData { State = "Wyoming", Candidate = "Romney", Electors = 3 }};
return electionResults;
}
public class ElectionData
{
private string state;
public string State
{
get { return state; }
set { state = value; }
}
public string candidate;
public string Candidate
{
get { return candidate; }
set { candidate = value; }
}
private double electors;
public double Electors
{
get { return electors; }
set { electors = value; }
}
}
```

ASPX-CS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("State")
.ShapePropertyPath("name")
.DataSource(datasource)
.ShapeSettings(shape =>
{
shape.StrokeThickness(0.5)
.AutoFill(false)
.Stroke("white")
.ValuePath("Electors")
.ColorValuePath("Candidate")
.EqualColorMappings(cm =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
cm.Value("Romney").Color("#D84444").Add();
cm.Value("Obama").Color("#316DB5").Add();
});
})
.Add();
})
)
```



Binding color from the datasource

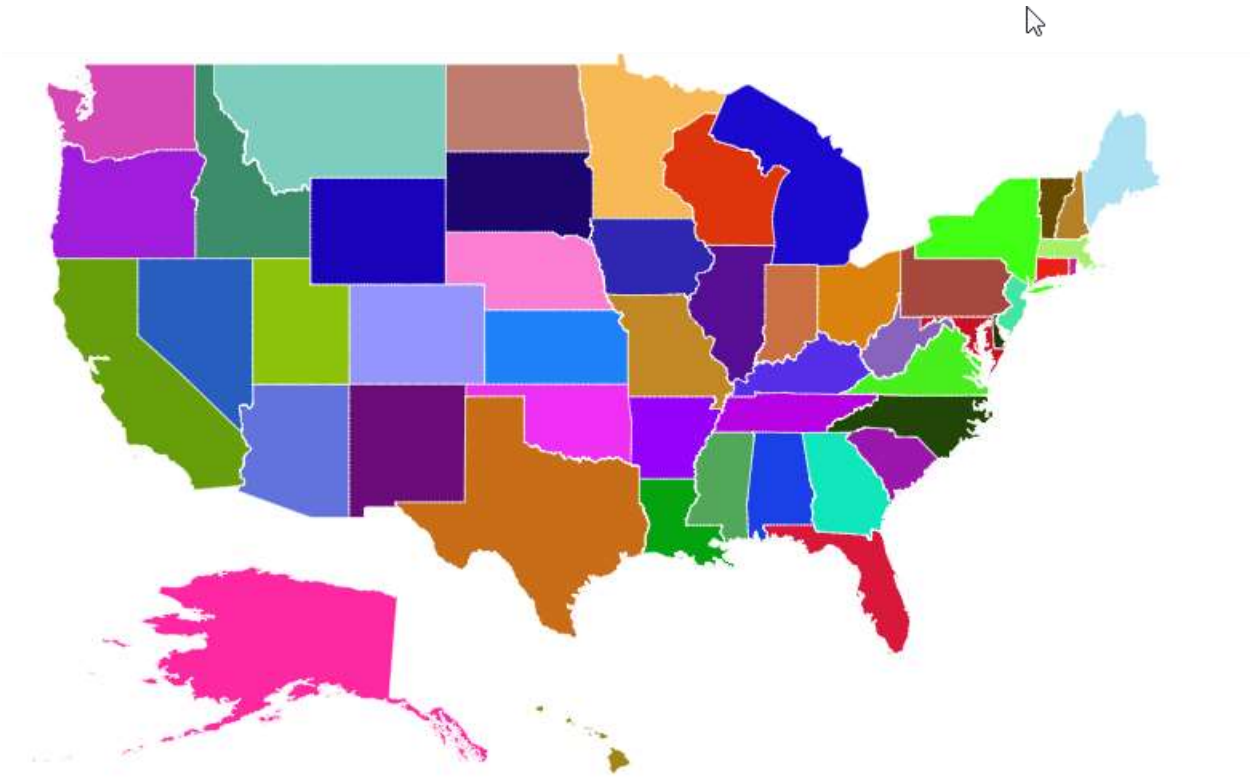
By using colorPath property, you can bind the specific field in the datasource to the map, and the shapes will be rendered based the color values available from the field you bind.

Note: While setting color for map by using colorPath property, do not set any other color mapping hence color mapping is having higher priority than colorPath.

ASPX-CS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("State")
.ShapePropertyPath("name")
.DataSource(datasource)
.ShapeSettings(shape =>
{
shape.ColorPath("ShapeFill")
})
```

```
.Add();
})
)
```



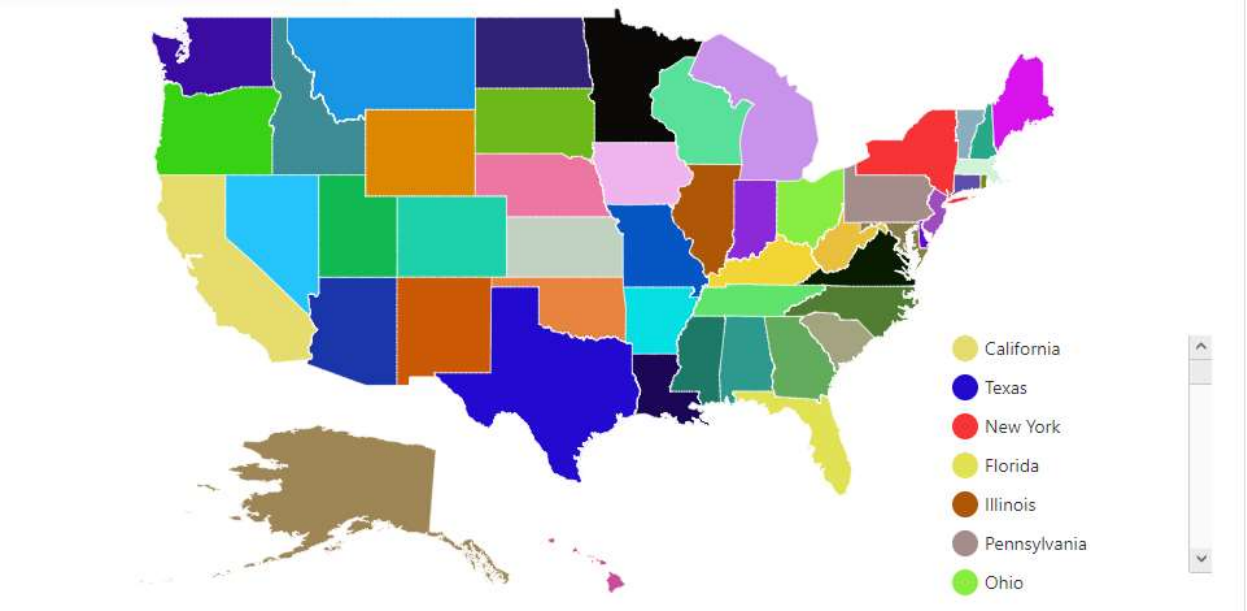
While binding color path for map, it is possible to render legend for each shape in the map. To achieve this, set the textPath for legend. textPath contains another field name in the datasource. As per the text, content in the field legend will be generated.

ASPX-CS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeDataPath("State")
.ShapePropertyPath("name")
.DataSource(datasource)
.ShapeSettings(shape =>
{
shape.ColorPath("ShapeFill")
})
.LegendSettings(legend =>
{
legend.ShowLegend(true)
.TextPath("name")
})
.Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```



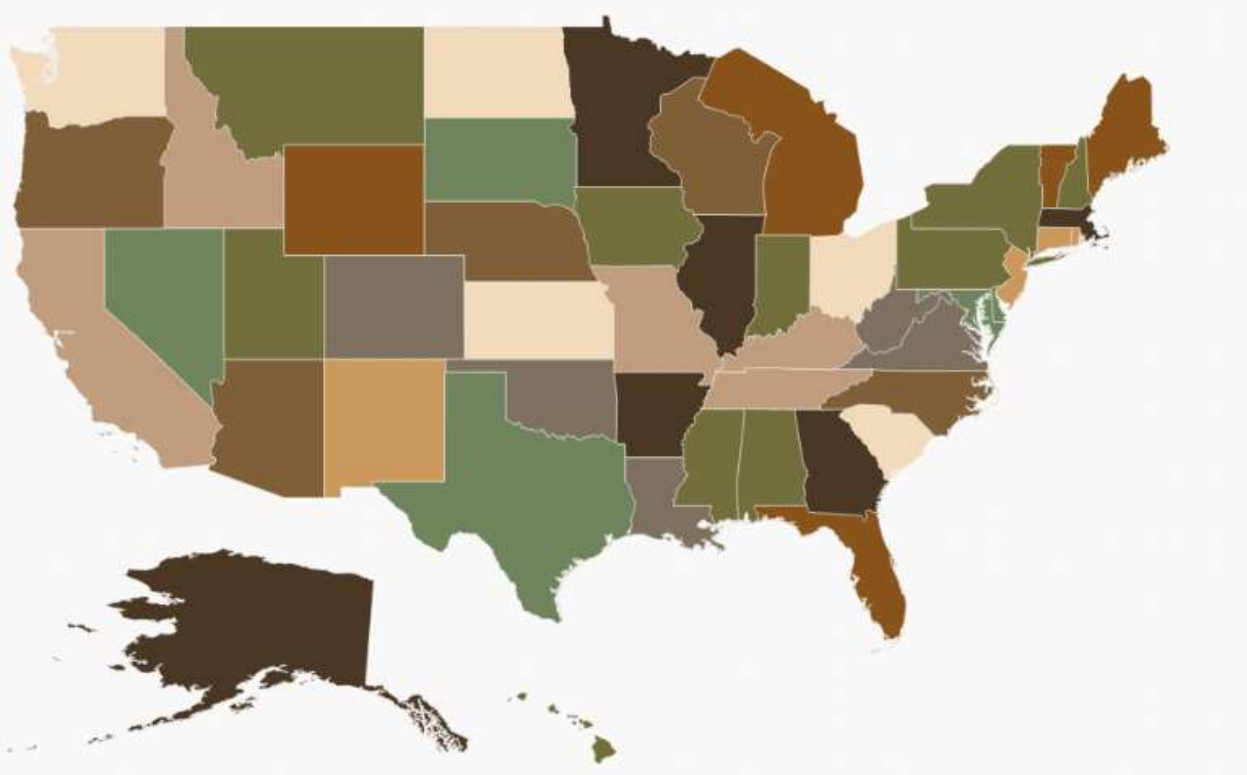
ColorPalette

AutoFill

When AutoFill property is set to true, shapes are filled with default colors from built-in palettes or custom palette.

ASPX-CS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeSettings(shape =>
{
shape.StrokeThickness(0.5)
.Stroke("white")
.AutoFill(true);
})
.Add();
})
)
```



ColorPalette

The ColorPalette property determines whether the auto fill colors are fetched from built-in color palettes or custom palette.

The ColorPalette property can be set with Palette1, Palette2, Palette3 and CustomPalette values where Palette1, Palette2 and Palette3 are built-in color palettes and default value for this property is “Palette1”.

The CustomPalette property is used to set an array of colors to be auto filled in shapes.

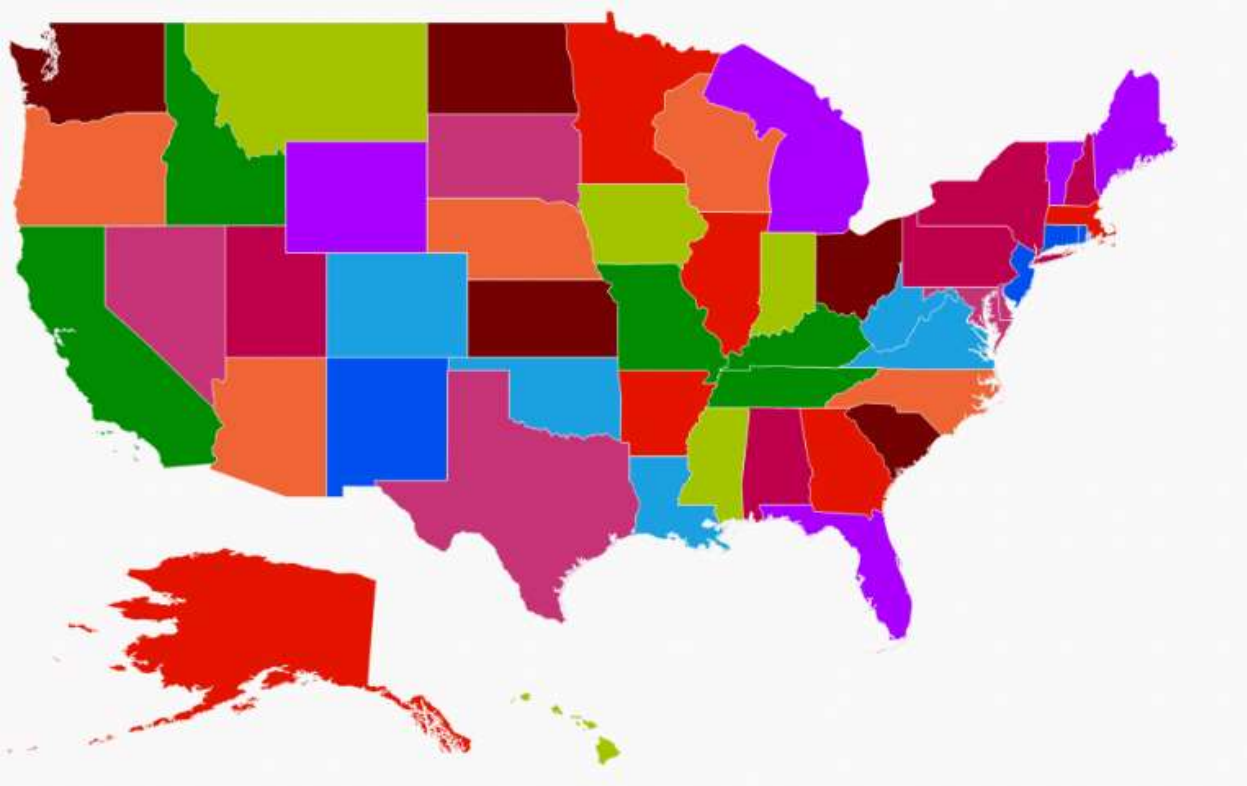
This property is enabled only when ColorPalette property value is set to “CustomPalette”.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeSettings(shape =>
{
shape.ColorPalette(ColorPalette.CustomPalette)
.AutoFill(true)
.CustomPalette(new List<string>
{
"#E51400", "#A4C400", "#730202",
"#008B00", "#EF6535",
"#1BA0E2", "#C63477", "#0050EF",
"#BF004D", "#AA00FF"
});
})
.Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```



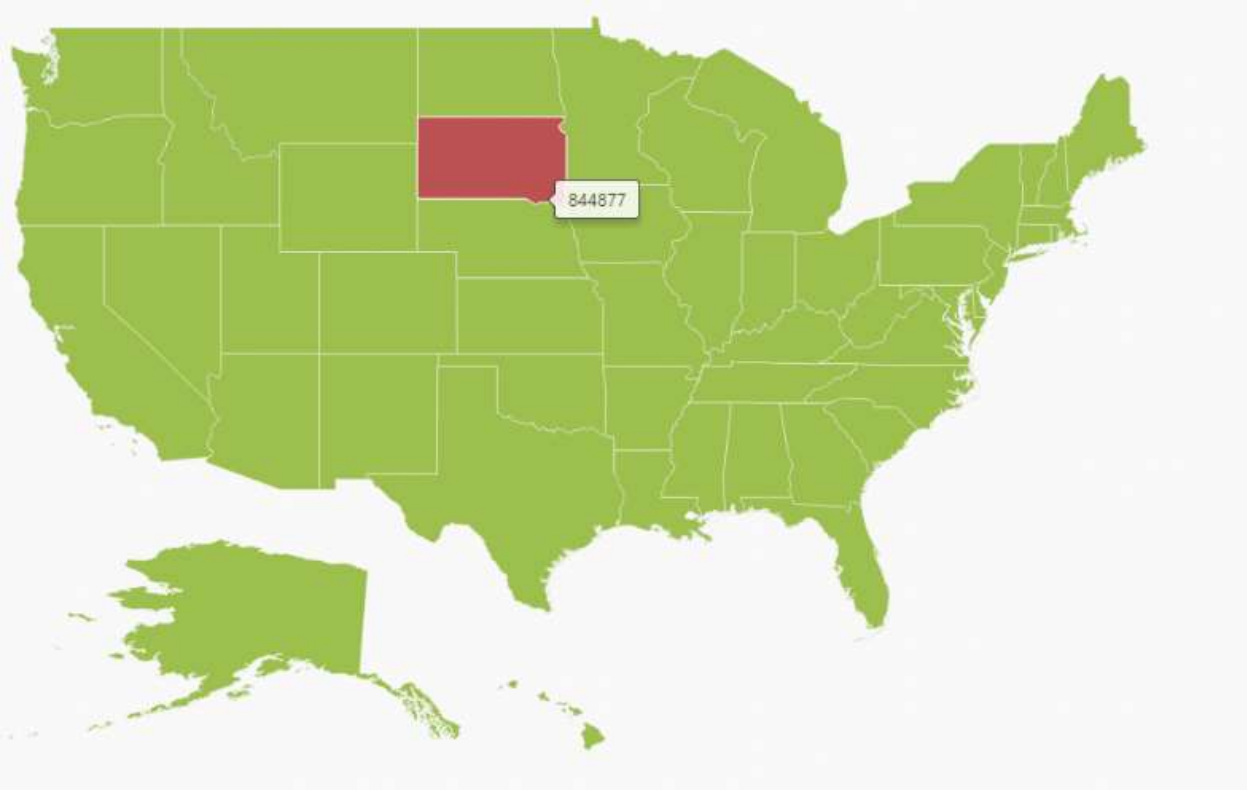
Tooltip

The tooltip is displayed only when you set ShowTooltip to “true” in the shape layers. By default, it takes the property of the bound object that is referred in the ValuePath and displays its content on hovering the corresponding shape.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeSettings(shape=>
shape.ValuePath(“name”);
})
.ShowTooltip(true)
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip Template

The ToolTipTemplate property is used for customizing the template for tooltip.

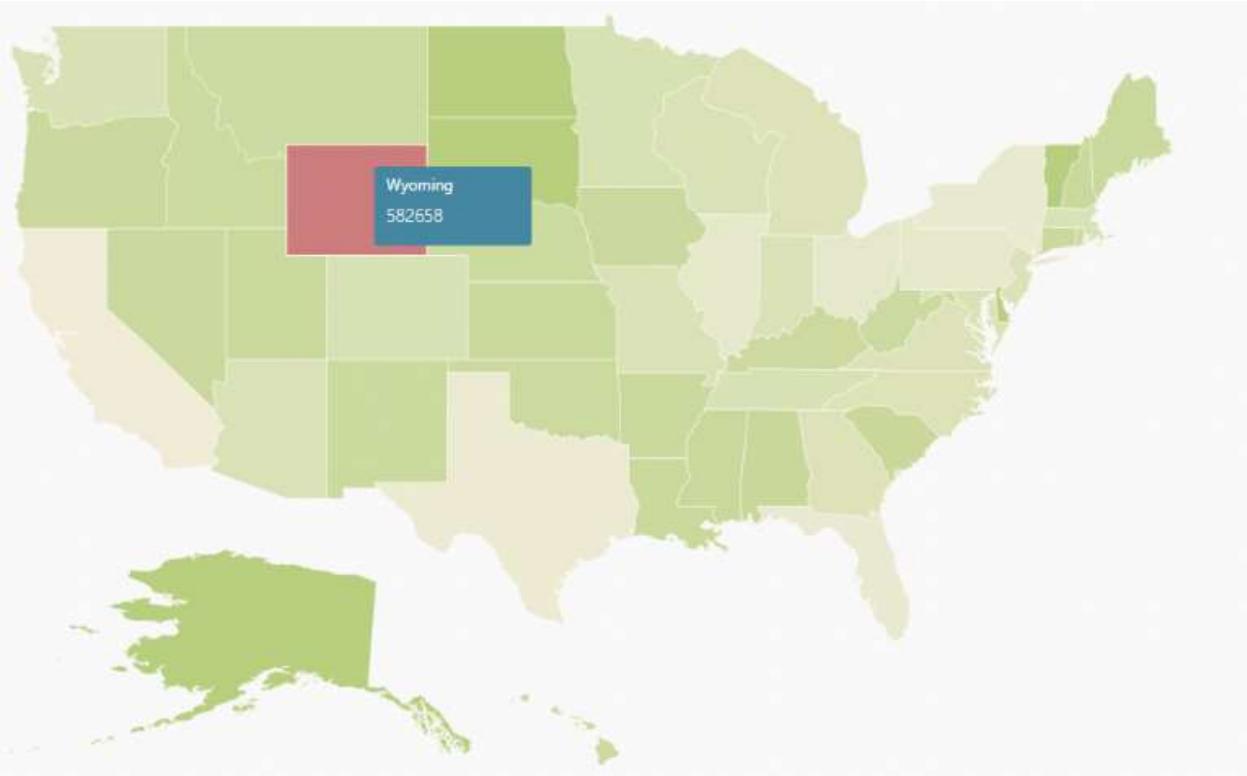
ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeSettings(shape=>
{
shape.ValuePath("name");
});
.ShowTooltip(true)
.ToolTipTemplate("'myTooltip'")
.Add();
})
)
<div  id="myTooltip" style="display: none;">
<div >
<div style="height:40px;width:78px;background:#4586a0;border-radius:3px;">
<div style="margin-top:-20px;margin-left:9px;padding-top:3px">
<label style="margin-top:-20px;font-weight:normal;font-
size:12px;color:white;font-family:Segoe UI;">{{:name}}</label>
</div>
<div style="height:10px;"></div>
<div style="margin-top:-10px;margin-left:9px;">
<label style="margin-top:-10px;font-weight:normal;font-
size:14px;color:white;font-family:segoe ui light;">{{:population}}</label>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
</div>
</div>
```

The following screenshot illustrates a map control displaying a Tooltip with template.



Customize map background

The Map background can be customized by using the background property of the Map.

ASPX-CS

```
@(Html.EJ().Map("container")
.Background("skyblue")
)
```

Base Map Index

Specifies the index of the map to determine the shape layer to be displayed, you can use baseMapIndex property and the default value is 0.

ASPX-CS

```
@(Html.EJ().Map("container")
.BaseMapIndex(0)
)
```

Center Position

Specify the centerPosition where map should be displayed

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Map("container")
.CenterPosition([38.5000, -98])
)
```

Label Settings

The labelSettings defines the basic customization settings of labels in the map.

The below properties are used for labelSettings

- enableSmartLabel - enable or disable the enableSmartLabel property.
- labelLength - set the labelLength property.
- labelPath - set the labelPath property.
- showLabels - The property specifies whether to show labels or not
- smartLabelSize - set the smartLabelSize property.

ASPX-CS

```
@(Html.EJ().Map("container")
.LabelSettings(lb=>lb.EnableSmartLabels(false).LabelLength(2).ShowLabels(fal
se).SmartLabelSize(LabelSize.Fixed))
)
```

Map Elements

Map control contains a set of map elements, including shapes, bubbles, markers, legend, labels and data items that can be visualized with customized appearance showing additional information on the map using bound data.

Markers

Markers are notes that is used to leave some message on the map.

There are two ways to set marker for map.

- Using markers and marker template
- Adding marker objects to map.

Markers

The Markers property has a list of objects that contains the data for Annotation. You can visualize these data by using MarkerTemplate property.

C#

```
public ActionResult Map()
{
// ...
ViewData["Annotationdatasource"] = GetMarkers();
return View();
}
// ...
public List<MarkerData> GetMarkers()
{
List<MarkerData> markerData = new List<MarkerData>
{
```

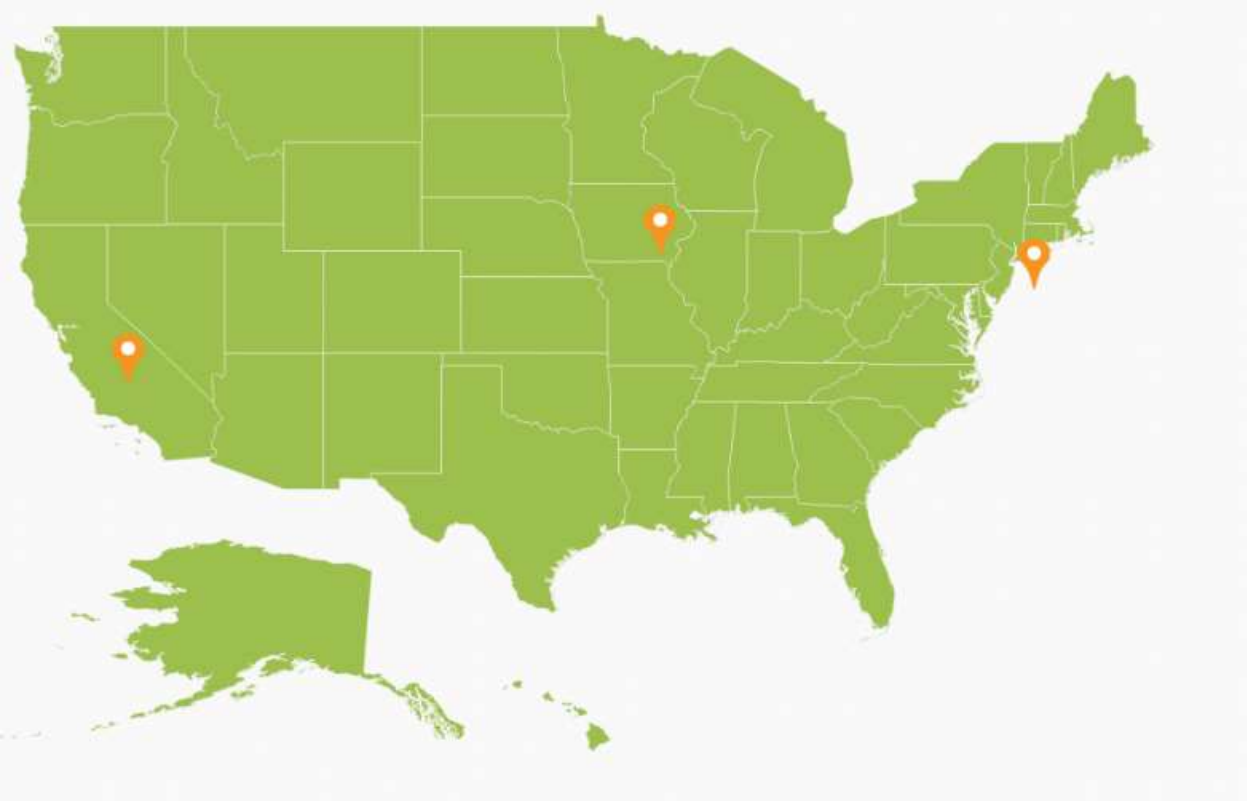
Copyright © 2001 - 2020 Syncfusion Inc.

```
new MarkerData {City = "California", Latitude = 37.0000, Longitude = -
120.0000 },
new MarkerData {City = "New York", Latitude = 40.7127, Longitude = -74.0059
},
new MarkerData {City = "Iowa", Latitude = 42, Longitude = -93}
};
return electionResults;
}
public class MarkerData
{
private double latitude;
public double Latitude
{
get { return latitude; }
set { latitude = value; }
}
public double longitude;
public double Longitude
{
get { return longitude; }
set { longitude = value; }
}
private string city;
public string City
{
get { return city; }
set { city = value; }
}
}
```

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
.Markers(markers)
.MarkerTemplate("template")
})
)
<div  id="template" style="display: none;">
<div>
<svg height="100" width="100">
<circle cx="20" cy="20" r="10" stroke="black" stroke-width="1" fill="red" />
</svg>
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Adding Marker objects to map

Without Datasource, n number of markers can be added to shape layers with Markers property. Each marker object contains the following list of properties.

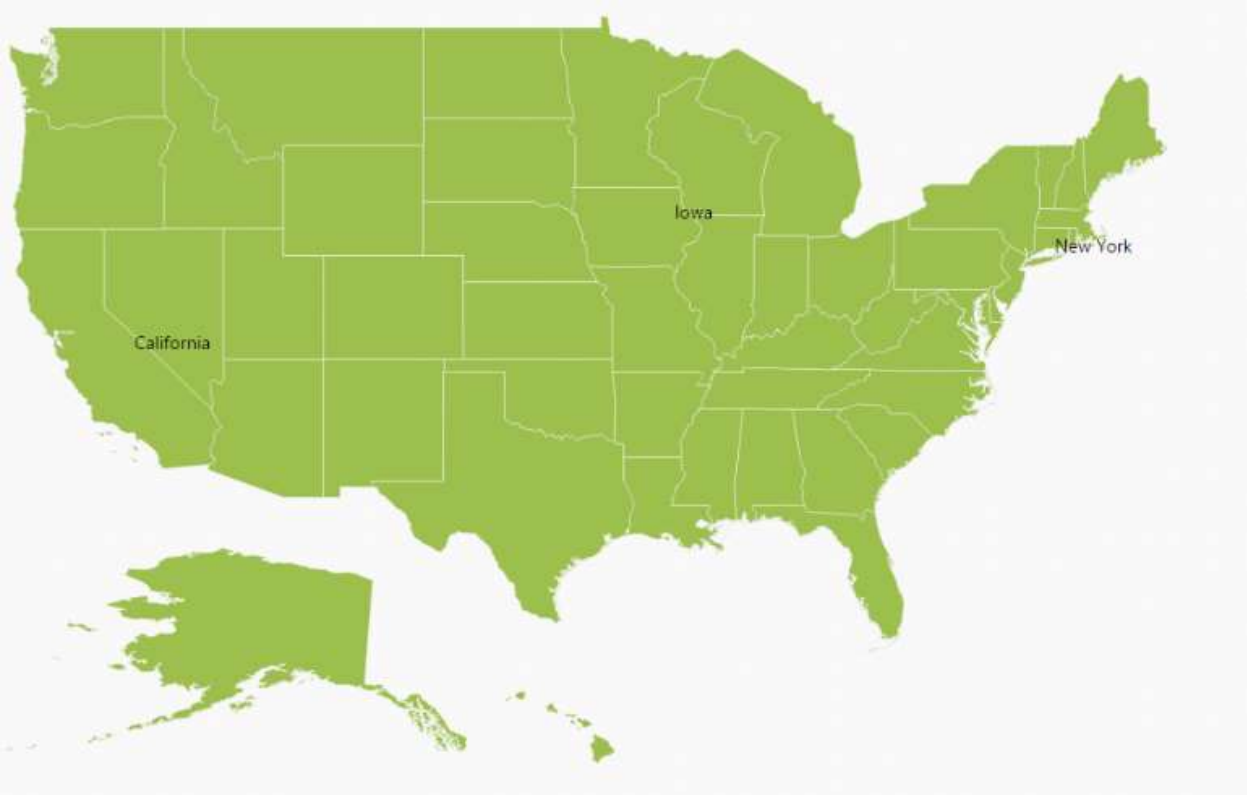
- Label - Text that displays some information about the annotation in text format.
- Latitude - Latitude point determine the Y-axis position of annotation.
- Longitude - Longitude point determine the X-axis position of annotation.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.EnableMouseHover(true)
.ShapeData(mapData)
.ShapeSettings(ss =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White")
.HighlightStroke("White")
.HighlightColor("#BC5353")
.HighlightBorderWidth(1);
})
.Markers(an =>
{
an.Label("California")
.Latitude(37)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Longitude(-120).Add();
an.Label("New York")
.Latitude(40.7127)
.Longitude(-74.0059).Add();
an.Label("Iowa")
.Latitude(42)
.Longitude(-93).Add();
})
.Add();
})
)
```



Bubbles

Bubbles in the Maps control represent the underlying data values of the map. Bubbles are scattered throughout the map shapes that contain bound values.

Bubbles are included when data binding and the BubbleSettings is set to the shape layers.

Properties

Property	Type	Description
MaxValue	String	Get or sets the maximum height and width of the bubble.
MinValue	String	Gets or sets the minimum height and width of the bubble.

ColorValuePath	String	Get or sets the field value that is to be fetched from data for each bubble used for determining the bubble color.
ValuePath	String	Gets or sets the field value that is to be fetched from data for each bubble.
ColorMappings	Collection of RangeColorMapping	Gets or sets the tree map colors.
Color	String	Gets or sets the fill color for bubbles.
ShowTooltip	Boolean	Enable or disable the tooltip for bubbles.
TooltipTemplate	String	Gets or sets the tooltip template for bubbles.

Add Bubbles to a Map

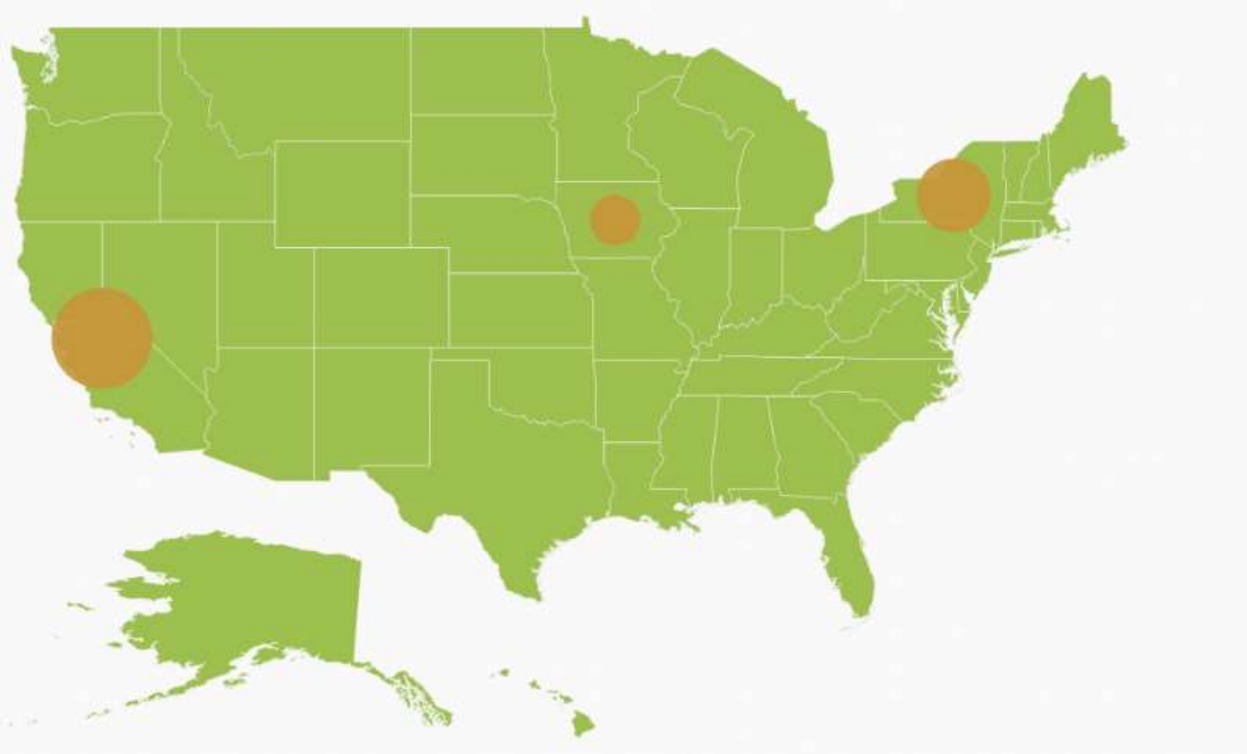
To add bubbles to a map, the bubble marker setting is added to the shape file layer. Create the Model and ViewModel as illustrated in the Data Binding topic and add the following code. Also set the `MaxValue`, `MinValue`, and `ValuePath` properties as illustrated in the following code sample.

Note: Tooltip and Color Mappings for bubble is to be set as similar to tooltip and color mappings set in layers and ShapeSettings. For more details, refer Tooltip and Color Mappings section.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.EnableMouseHover(true)
.ShapeData(mapData)
.DataSource(population_data)
.ShapeSettings(ss =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White");
})
.BubbleSettings(bs =>
{
bs.ValuePath("Population ")
.ShowBubble(true)
.MinValue(20)
.MaxValue(40)
.Color("#C99639");
})
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Add bubbles to the map

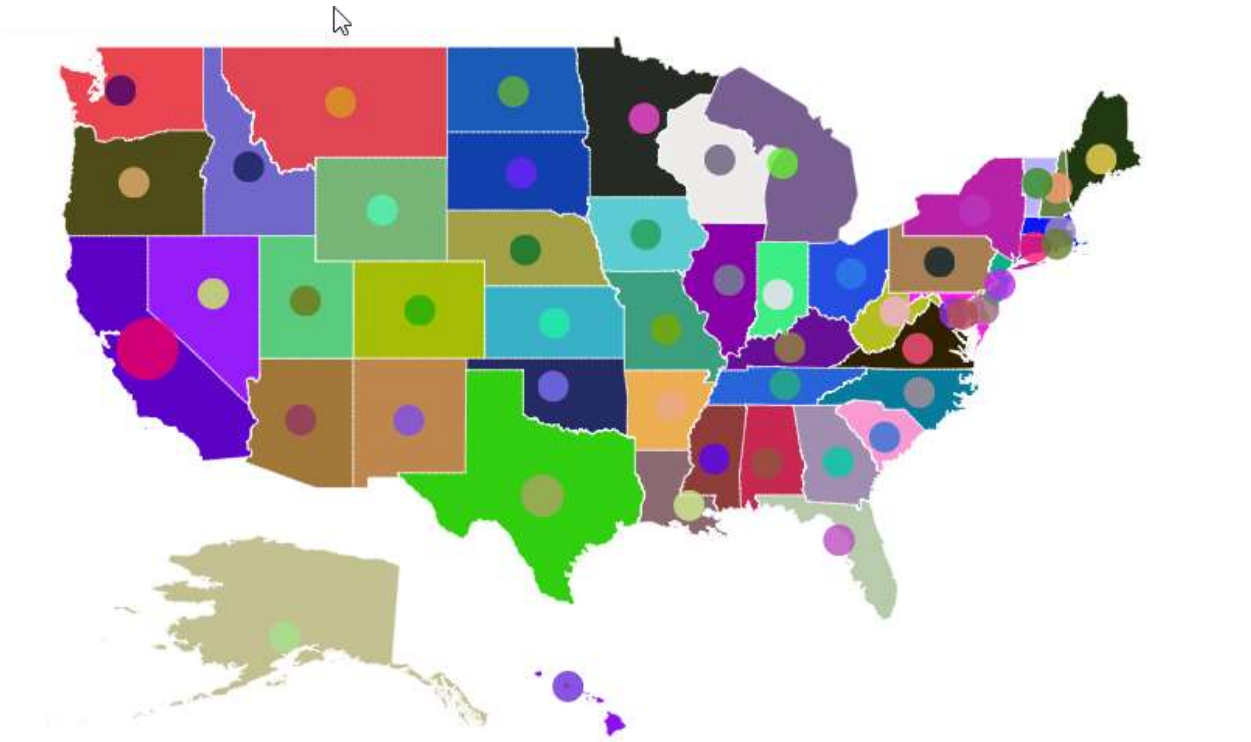
You can set color for each bubble by using `ColorPath` property in bubble settings.

Note: While settings color for bubbles by using `ColorPath`, do not set any other color mapping hence color mapping is having higher priority than `colorPath`.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.EnableMouseHover(true)
.ShapeData(mapData)
.DataSource(population_data)
.ShapeSettings(ss =>
{
ss.ColorPath("ShapeFill")
})
.BubbleSettings(bs =>
{
bs.ValuePath("Population ")
.ShowBubble(true)
.MinValue(20)
.MaxValue(40)
.ColorPath("fill");
})
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



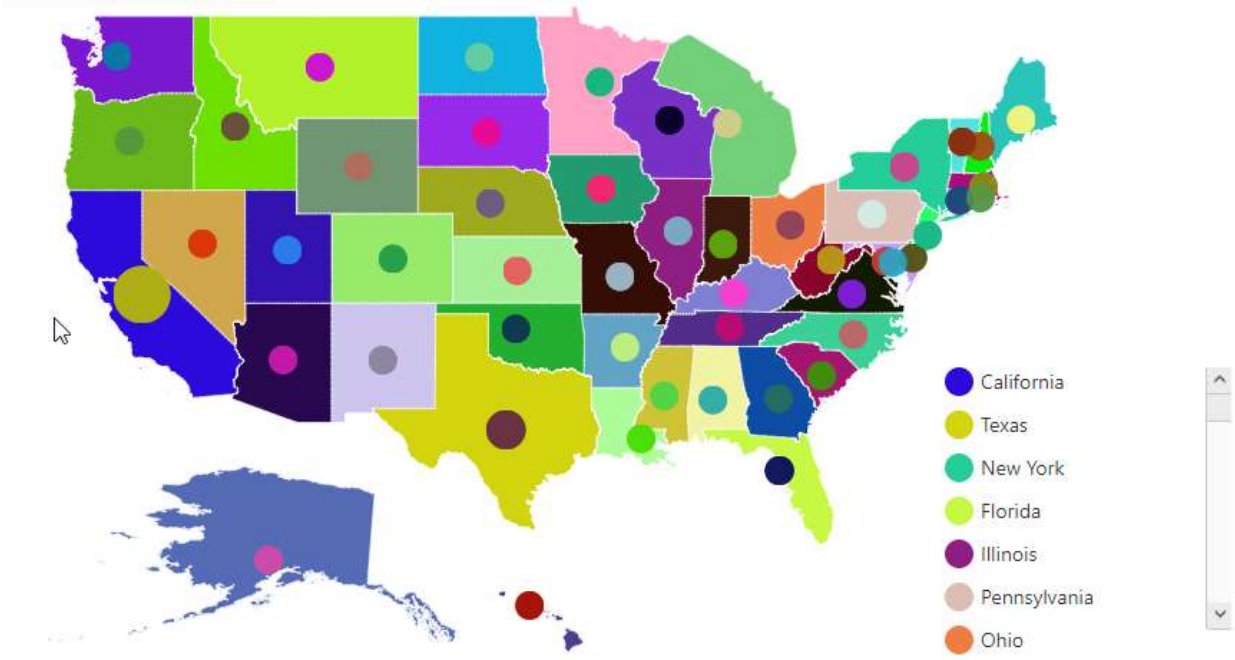
While binding colorPath for bubbles, it is possible to render the legend for each bubble. To achieve this, set the textPath for legend. textPath contains another field name in the datasource. As per the text, content in the field legend will be generated.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.EnableMouseHover(true)
.ShapeData(mapData)
.DataSource(population_data)
.ShapeSettings(ss =>
{
ss.ColorPath("ShapeFill")
})
.BubbleSettings(bs =>
{
bs.ValuePath("Population ")
.ShowBubble(true)
.MinValue(20)
.MaxValue(40)
.ColorPath("fill");
})
.LegendSettings(bs =>
{
ls.ShowLegend(true)
.TextPath("name");
})
.Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```



Legend

A legend is a key used on a map, contains swatches of symbols with descriptions. It provides valuable information for interpreting what the map is displaying you, and can be represented in various colors, shapes or other identifiers based on the data. It gives a breakdown of what each symbol represents throughout the map.

Visibility of Legend

The Legends can be made visible by setting the `ShowLegend` property of legendSettings.

Positioning of Legend

The legend can be positioned in two ways.

- Absolute Position.
- Dock Position.

Absolute Position

Based on the margin values of X and Y-axes, the Map legends can be positioned with the support of `PositionX` and `PositionY` properties available in `LegendSettings`. For positioning the legend based on margins corresponding to a map, `Position` value is set as '*None*'.

Dock Position

The map legends can be positioned in following locations within the container.

- TopLeft
- TopCenter
- TopRight
- CenterLeft

Copyright © 2001 - 2020 Syncfusion Inc.

- Center
- CenterRight
- BottomLeft
- BottomRight
- BottomCenter
- BottomRight
- None

You can set this option by using Position property in LegendSettings.

Legend Size

The map legend size can be modified using Height and Width properties in LegendSettings.

Legend for Shapes

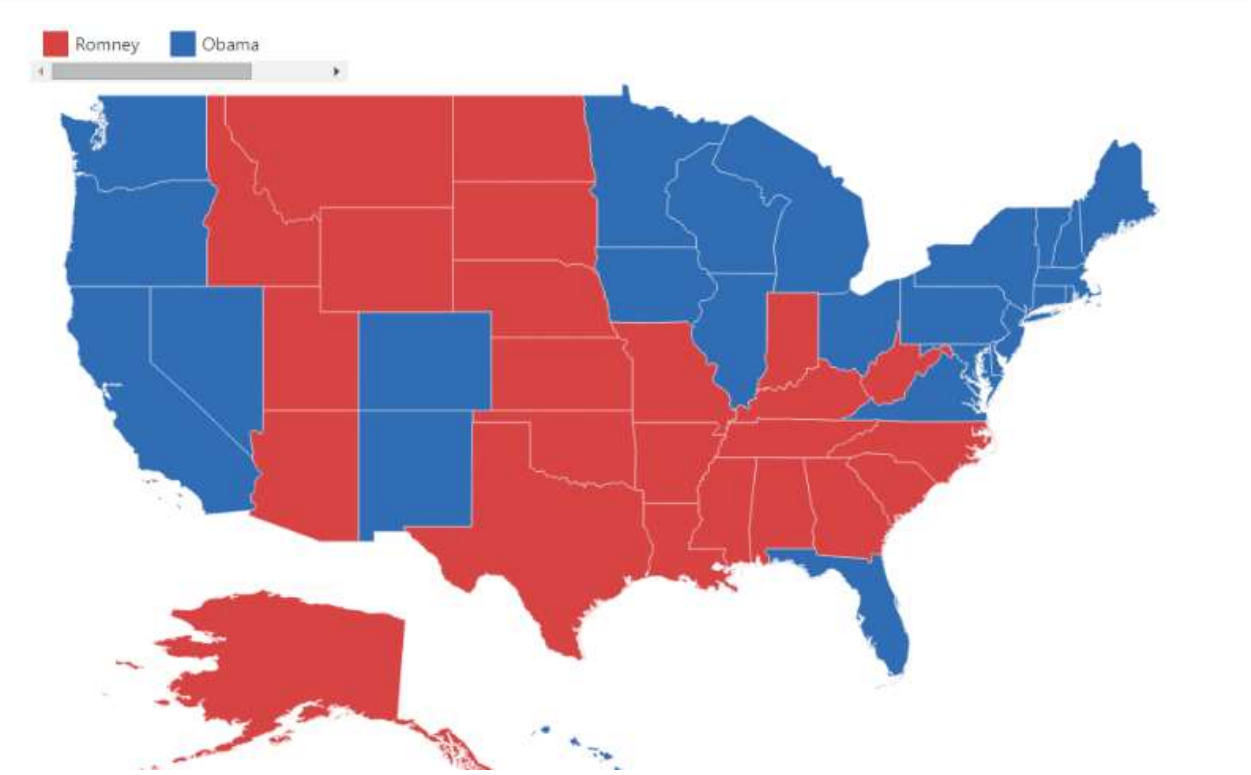
The Layer shape type legends can be generated for each color mappings in shape settings.

Note: Here, Equal Color Mapping code sample for shapeSettings with color mappings is referred.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
layer.LegendSettings(ml =>
{
ml .ShowLegend(true)
.Height(20)
.Width(60)
.PositionX(80)
.PositionY(90);
})
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Interactive Legend

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding shapes. You can enable this option by setting Mode property in LegendSettings value as “Interactive” and default value of Mode property is “Default” to enable the normal legend.

Title for Interactive Legend

You can provide the title for interactive legend by using Title property in LegendSettings.

Label for Interactive Legend

You can provide the left and right labels to interactive legend by using LeftLabel and RightLabel properties in LegendSettings.

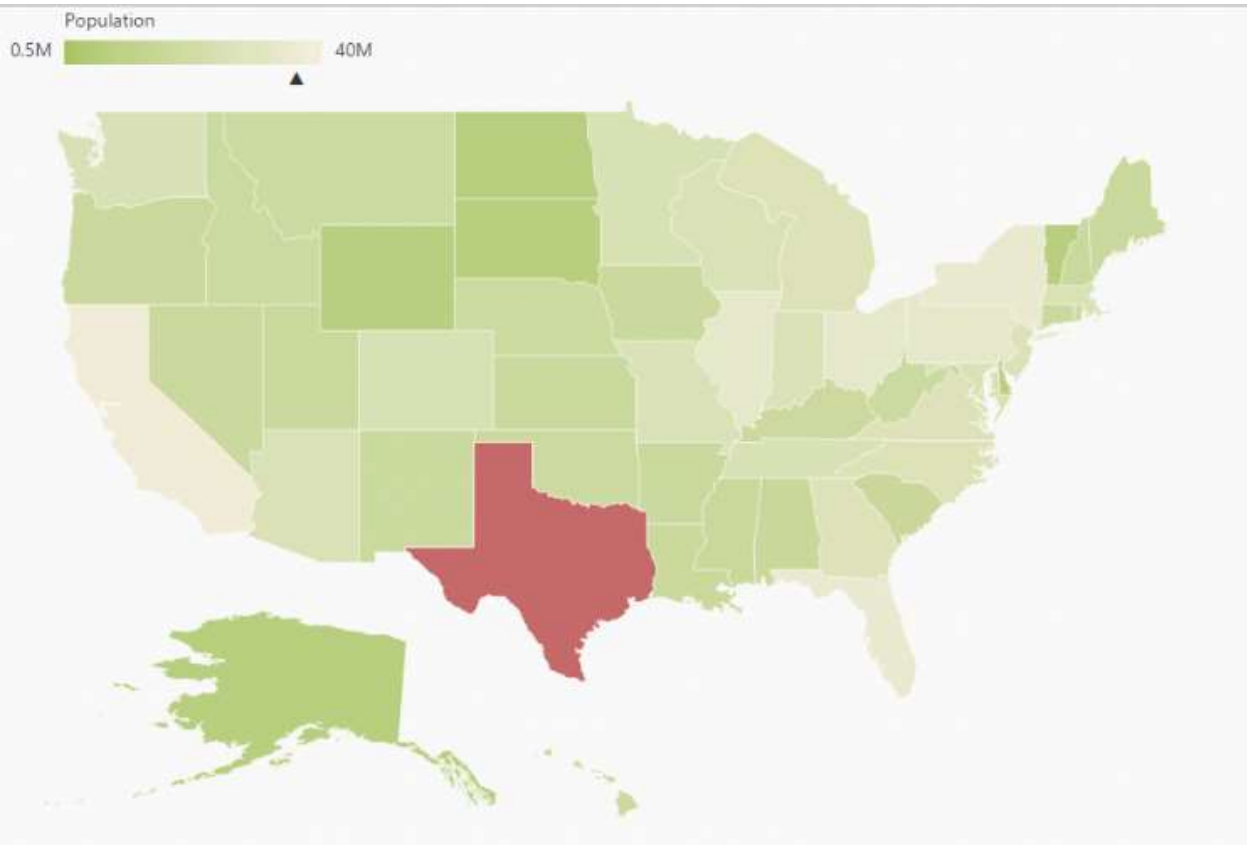
Note: Here, Range Color Mapping code snippet for shapeSettings with color mappings is referred.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
.LegendSettings(ml =>
{
ml.ShowLegend(true)
.Height(15)
.Width(150)
.Position(Syncfusion.JavaScript.DataVisualization.Models.DockPosition.Topleft)
.Mode(Syncfusion.JavaScript.DataVisualization.Models.LegendMode.Interactive)
.Title("Population")
.LeftLabel("0.5 M")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.RightLabel("40 M");
})
})
)
```



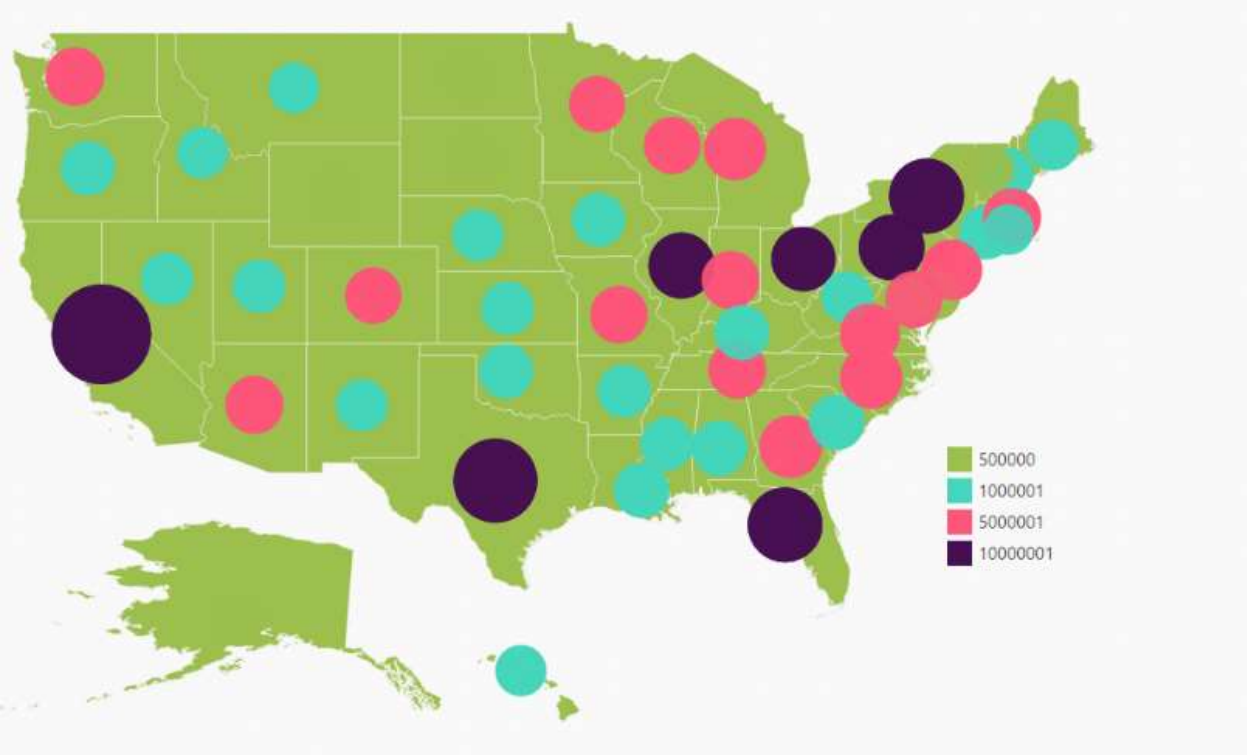
Bubble Legend

A bubble legend feature is used to provide the key (legend) for another map element bubble. You can activate the Bubble legend by setting the enum Type in LegendSettings as "*Bubble*" and this enables you to easily identify what value a particular bubble is representing.

ASPX-CS

```
@(Html.EJ().Map("container")
.Layers(layer =>
{
.LegendSettings(ml =>
{
ml.ShowLegend(true)
.Height(15)
.Width(150)
.Type(Syncfusion.JavaScript.DataVisualization.Models.LegendType.Bubbles)
.Title("Population");
})
.BubbleSettings( b1 =>
{
b1.ShowBubble(true)
.ValuePath(population)
```

```
.MinValue(20),
.MaxValue(40)
.RangeColorMappings(cm =>
{
cm.From(500000).To(1000000).GradientColors(new List<string> { "#9CBF4E",
"#B8CE7B" }).Range(10688).Add();
cm.From(1000001).To(5000000).GradientColors(new List<string> { "#B8CE7B",
"#CBD89A" }).Range(19390).Add();
cm.From(5000001).To(10000000).GradientColors(new List<string> { "#CBD89A",
"#DEE2B9" }).Range(18718).Add();
cm.From(10000001).To(40000000).GradientColors(new List<string> { "#DEE2B9",
"#F1ECD8" }).Range(30716).Add();
});
})
})
)
```



User Interaction

Options like zooming, panning and map selection enable the effective interaction on Map elements.

Map Selection

Each shape in the Map can be selected and deselected during interaction with shapes.

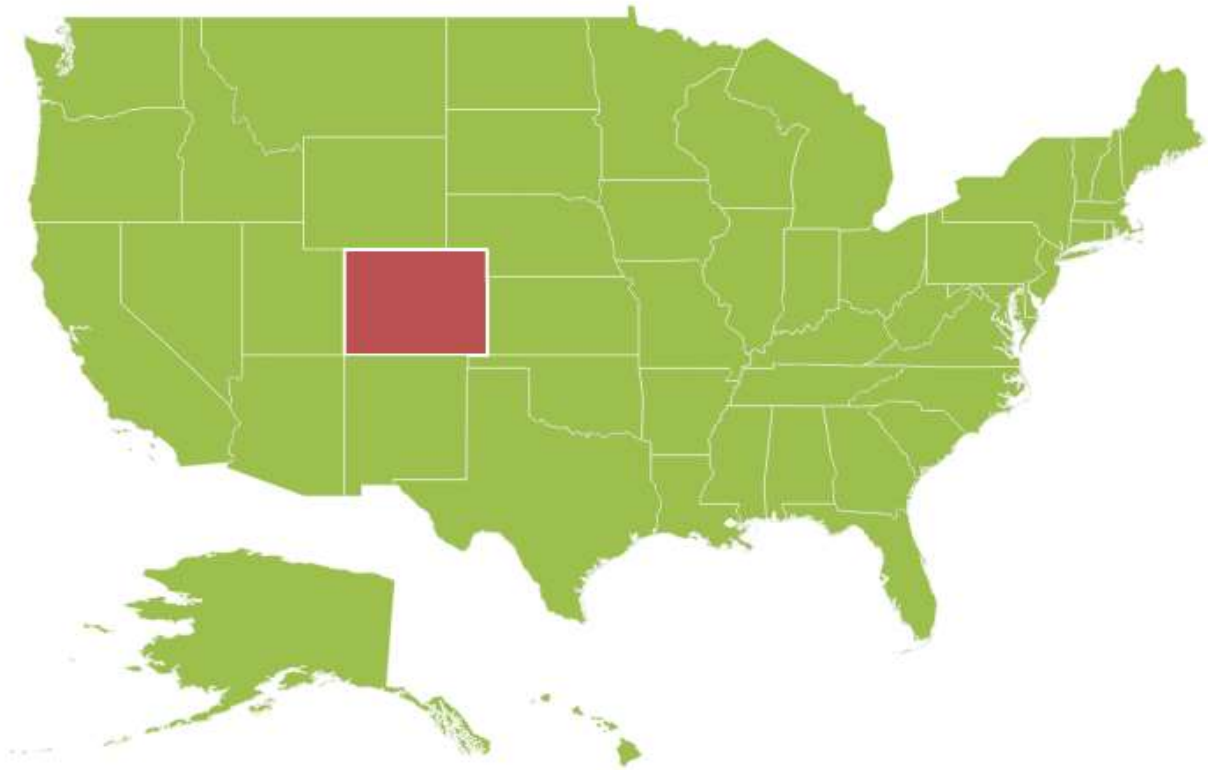
The SelectionColor property is used to get or set the selected shape color. The SelectionStroke and SelectionStrokeWidth properties are used to customize the selected shape border.

You can select the shape by tapping the shape. The single selection is enabled by the EnableSelection property of shape layer. When EnableSelection is set to false, the shapes cannot be selected.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.ShapeSettings(shape =>
{
shape.StrokeThickness(0.5)
.Fill("#9CBF4E")
.Stroke("white")
.SelectionColor("#BC5353")
.SelectionStrokeWidth(2)
.SelectionStroke("white");
})
.EnableSelection(true)
.Add();
})
)
```



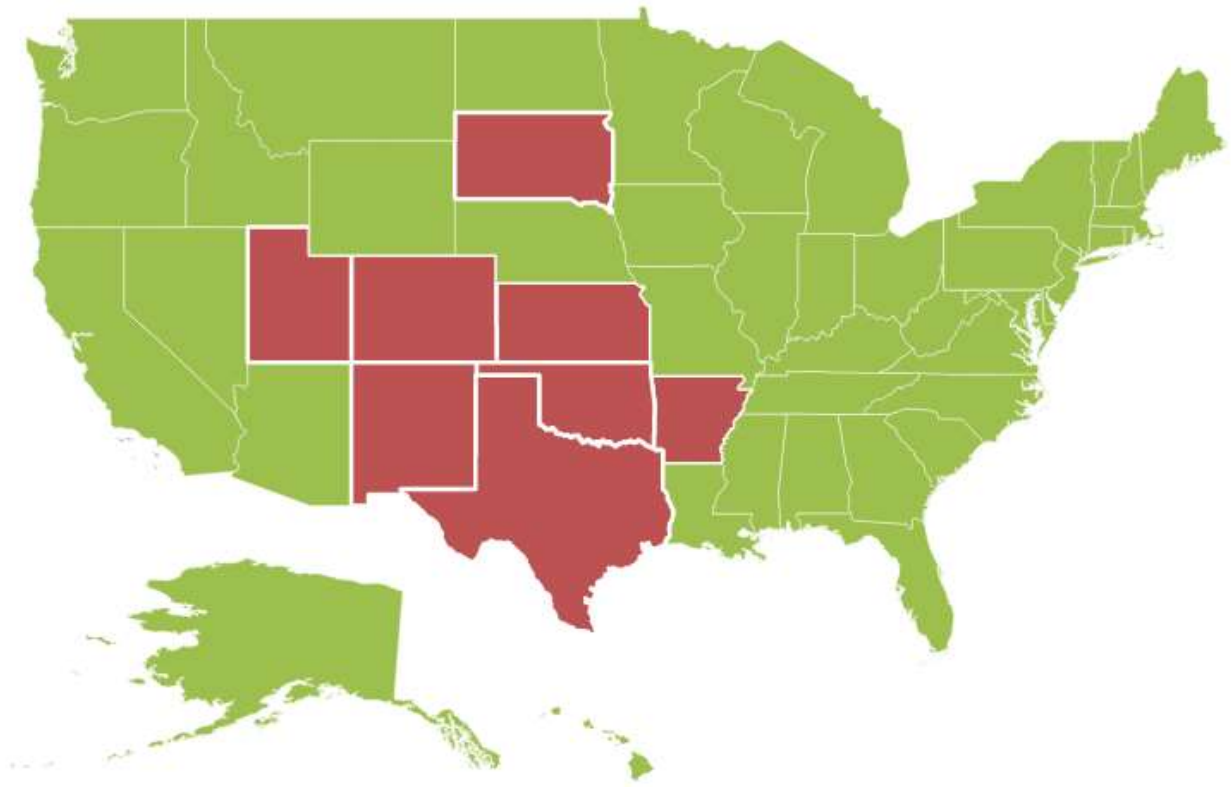
MultiSelection

This feature enables you to select multiple Map shapes on mouse taps accompanied by "Ctrl" key press. For this feature to get enabled, set the property selectionMode as "*multiple*" along with enableSelection property.

JS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
```

```
layer.ShapeData(mapData)
// ...
.EnableSelection(true)
.SelectionMode(Multiple)
// ...
.Add();
})
)
```



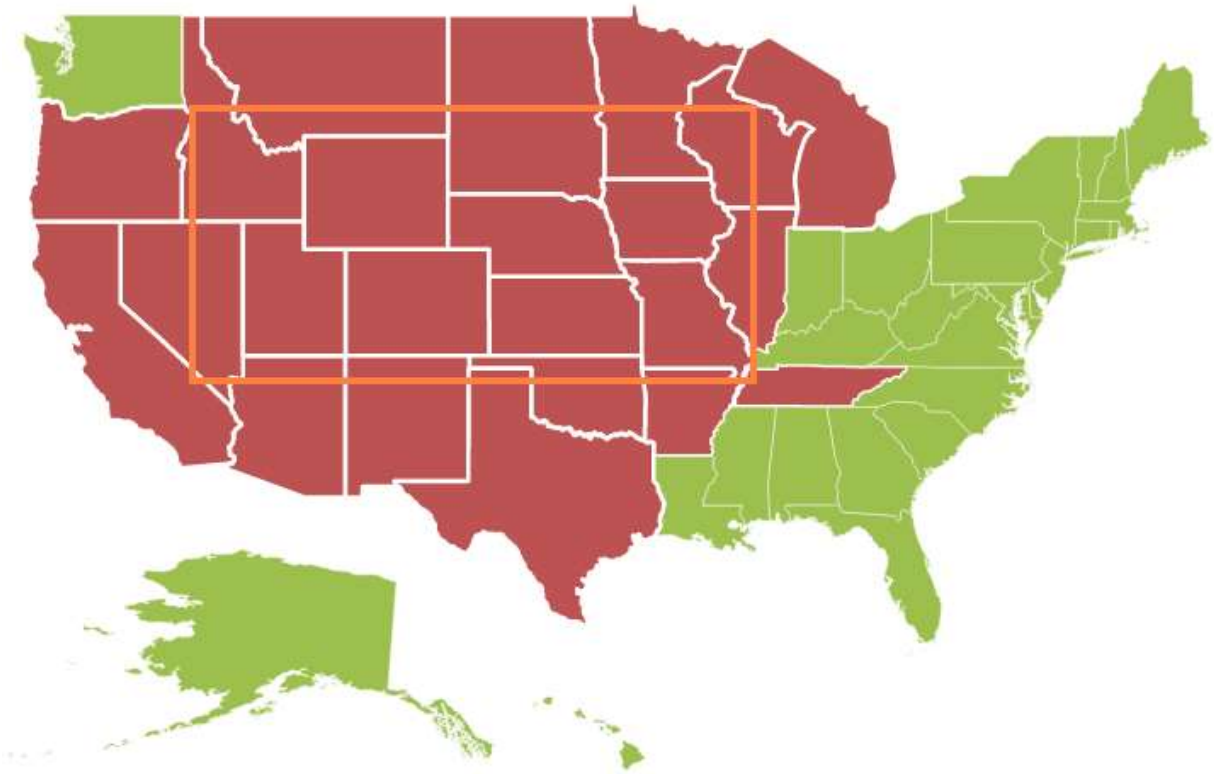
Dragging On Selection

This feature enables you to select the shapes by dragging over the shapes. While dragging over the shapes, a rectangle is generated and the shapes that come within the rectangle is selected.

You can enable this feature by setting the property DraggingOnSelection in the Layers to "*true*".

JS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer
// ...
.DraggingOnSelection(true)
// ...
.Add();
})
)
```



Zooming

The zooming feature enables you to zoom in and out of the Map to show in-depth information. It is controlled by the Level property of the Map. When the zoom level of the Map control is increased, the Map is zoomed in. When the zoom level is decreased, then the Map is zoomed out.

Properties

The following properties are related to the zooming feature of the Maps control:

- Level
- EnableZoom
- MinValue
- MaxValue

Level

The Level property determines the Map's scale size when zooming. The default value of Level is 1.

Note: The level cannot be less than 1.

EnableZoom

The EnableZoom property enables or disables the zooming feature.

MinValue

The MinValue property is used to set the minimum zoom level of the Map.

MaxValue

The MaxValue property is used to set the maximum zoom level of the Map.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.Add();
})
.ZoomSettings(zoom=>
{
zoom.EnableZoom(true)
.MinValue(1)
.MaxValue(20)
.Level(1);
})
)
```

Factor

Specifies the zoom factor for map zoom value, you can use factor property.

ASPX-CS

```
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(mapData)
.Add();
})
.ZoomSettings(zoom=>
{
zoom.EnableZoom(true)
.Factor(1)
})
)
```

Additional Options to Zoom the Map

Maps can be zoomed by using the following options also,

- Zoom method.
- Mouse scroll.
- Mouse double tap.
- Shape selection
- Position

By using Zoom method

You can zoom the Maps by using Zoom method. The Zoom method contains parameter zoom value. The Map can be zoomed or scaled based on zoom value parameter.

ASPX-CS

```
$("#map").ejMap("zoom", 2);
```

Copyright © 2001 - 2020 Syncfusion Inc.

By using mouse scroll

You can zoom the Map with mouse events by using mouse scroll. When the mouse is scrolled up, the Map is zoomed in and when the mouse is scrolled down, the Map is zoomed out.

By using mouse double tap

When the map is double-tapped by using mouse, the zoom in operation is performed.



By using Shape Selection

Map shape is zoomed to the whole map area on the shape selected. Animation can be applied for that zooming with the EnableAnimation property as true.

You can enable this feature by setting EnableZoomOnSelection property value as '*true*'.

When EnableZoomOnSelection property is set to true, then zooming of the Map control is muted on double click.

ASPX-CS

```
@(Html.EJ().Map("maps")
.ZoomSettings(zoom=>
{
zoom.EnableZoomOnSelection(true);
})
)
```

Positioning

Depending on the latitude and longitude, you can zoom the Map to the exact position. All locations are considered as latitude and longitude values and the exact location is considered as Map coordinates.

The NavigateTo is a method defined that allows you to zoom the Map control to the given location. This method contains three attributes as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

Attribute Table

Attribute	Type	Description
Latitude	Double	Latitude point of the position
Longitude	Double	Longitude point of the position
Level	Double	Zoom level of the map

JS

```
<script type="text/javascript">
function buttonClick()
{
$("#map").ejMap("navigateTo", 13, 80, 5);
}
</script>
```

Panning

The panning feature enables the Map navigation. The `EnablePan` property is used to enable or disable the panning support.

ASPX-CS

```
@(Html.EJ().Map("maps")
.EnablePan(true)
)
```

Mouse Wheel Zoom

You can zoom the Map with mouse by using mouse wheel. When the mouse is wheeled, the Map is zoomed in.

You can enable this feature by setting `EnableMouseWheelZoom` property value as '*true*'.

ASPX-CS

```
@(Html.EJ().Map("maps")
.ZoomSettings(zoom=>
{
zoom.EnableMouseWheelZoom(true);
})
)
```

Navigation Control

Navigation control is built-in with Maps control. With Navigation control, Maps can be panned in any direction and zoomed. It is possible to show or hide the NavigationControl by `EnableNavigation` property.

Copyright © 2001 - 2020 Syncfusion Inc.



ASPX-CS

```
@(Html.EJ().Map("maps")
.NavigationControl(navigation =>
{
navigation.EnableNavigation(true);
})
)
```

Zoom with Navigation Control

With Navigation control, the Maps can be zoomed. When you click the ZoomIn button, the Map is zoomed in and when you click the ZoomOut button, the Map is zoomed out.

Panning with Navigation Control

Maps can be panned with Pan buttons (TopPan button, RightPan button, BottomPan button, and LeftPan button). When you click a particular Pan button, the Map is panned on the respective directions.

Navigation Control Positions

The Navigation control can be positioned in two ways.

- Absolute Position
- Dock Position

Absolute Position

Based on the margin values of X and Y-axes, the navigation control can be positioned with the help of the X and Y properties available in AbsolutePosition. For positioning the navigation control based on margins corresponding to a Map, DockPosition value is set as '*None*'.

Dock Position

The navigation control can be positioned in following locations within the container.

Copyright © 2001 - 2020 Syncfusion Inc.

- TopLeft
- TopCenter
- TopRight
- CenterLeft
- Center
- CenterRight
- BottomLeft
- BottomRight
- BottomCenter
- BottomRight
- None

You can set this option by using the DockPosition property in NavigationControl.

Orientation

Set the orientation value for navigation control.

Name	Description
horizontal	specifies the horizontal position
vertical	specifies the vertical position

Content

Specifies the navigation control template for map, you can use content property.

ASPX-CS

```
@(Html.EJ().Map("maps")
.NavigationControl(navigation =>
{
navigation.EnableNavigation(true)
.Orientation(Orientation.Vertical)
.AbsolutePosition(new ShapePoint(5, 12))
.DockPosition(Syncfusion.JavaScript.
DataVisualization.Models.DockPosition.None);
})
)
```

Animation

Animation is enabled or disabled using enable animation property.

ASPX-CS

```
@(Html.EJ().Map("maps")
.EnableAnimation(true)
)
```

Enable Layer Change Animation

Enables or Disables the animation for layer change in map, you can use enableLayerChangeAnimation property and the default value is false.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Map("maps")
.EnableLayerChangeAnimation(true)
)
```

Responsiveness during browser resize

Map is made responsive when resizing the browser by using isResponsive property.

ASPX-CS

```
@(Html.EJ().Map("maps")
.IsResponsive(true)
)
```

Layers

Map is maintained through Layers and it can accommodate one or more layers.

Multilayer

The Multilayer support allows you to load multiple shape files in a single container, enabling maps to display more information.

Adding Multiple Layers in the Map

The shape layers is the core layer of the map. The multiple layers can be added in the shape Layers as SubShapeFileLayers within the shape Layers.

SubLayer

The subLayer is the collection of shape Layers.

In this example, World Map shape is used as shape data by utilizing the "WorldMap.json" file in the following folder structure obtained from downloaded Maps_GeoJSON folder.

..\ Maps_GeoJSON\

Here "MapController.cs" is populated with data of World Map in "MapController.cs".

~/Controller/MapController.cs

C#

```
public ActionResult Map()
{
ViewData["world_map"] = GetWorldMap();
}
public object GetWorldMap ()
{
string data =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
JavaScriptSerializer ser = new JavaScriptSerializer();
ser.MaxJsonLength = int.MaxValue;
return new MapData(data);
}
```

Refer both USA data and world map data as illustrated in the following "Map.cshtml".

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@{
var mapData = ViewData["mapdata"];
var world_map = ViewData["world_map"];
}
@(Html.EJ().Map("maps")
.Layers(layer =>
{
layer.ShapeData(world_map)
.ShapeSettings(ss =>
{
ss.Fill("#9CBF4E")
.StrokeThickness(0.5)
.Stroke("White");
})
.SubLayer(sublayer =>
{
sublayer.ShapeData(mapData)
.ShapeSettings(ss =>
{
ss.Fill("orange")
.StrokeThickness(1)
.Stroke("White");
})
.Add();
})
.Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Map Providers

Map control support map providers such as OpenStreetMap that can be added to any layers in maps.

Open Street Map

OpenStreetMap is a map of the entire world. The OpenStreetMap allows you to view, edit and use geographical data in a collaborative way from any place on the Earth.

Enable OSM

You can enable this feature by setting the LayerType property value as "OSM".

ASPX-CS

```
@(Html.EJ().Map("map")
.Layers(layer =>
{
layer.LayerType(Syncfusion.JavaScript.DataVisualization.Models.LayerType.OSM
)
.UrlTemplate("http://a.tile.openstreetmap.org/level/tileX/tileY.png”);
})
)
```

URL Template

The UrlTemplate property determines the format of tile map. You can specify the template for the tile layer.



Bing Map

Bing Map is a key feature in accessing the external geospatial imagery services for deep-zoom satellite view.

Enable Bing Maps

You can enable this feature by defining the LayerType as “Bing”.

ASPX-CS

```
@(Html.EJ().Map("map")
.Layers(layer =>
{
layer.LayerType(Syncfusion.JavaScript.DataVisualization.Models.LayerType.Bin
g)
.Key("// ...bingMapKey”).Add();
})
)
```

Key

The bing Map Key is provided as input to this key property. The Bing Map key can be obtained from http://www.microsoft.com/maps/create-a-bing-maps-key.aspx.

Copyright © 2001 - 2020 Syncfusion Inc.



AngularJS Support

AngularJS is a JavaScript framework added to a HTML page with a <script> tag. It extends HTML attributes with directives and binds data to HTML with expressions. AngularJS directives allow you to specify custom and reusable HTML tags that moderate the behavior of certain elements.

Angular binding uses directives to plug its action into the page. Directives, all prefaced with ng-, are placed in HTML attributes. To know more about Angular binding refer to: http://help.syncfusion.com/aspnetmvc/maps/angularjs-support

Apply the plugin and property assigning the Map element through the directive that starts with the letter “e-“. The following code illustrates how to bind data to the Map component through Angular support.

ASPX-CS

```
//References to be added for AngularJS support.
<script src="@Url.Content("~/Scripts/angular.min.js")"></script>
<script
src="@Url.Content("~/Scripts/ej/ej.widget.angular.min.js")"></script>
<script src="@Url.Content("~/Scripts/MapsData/USA.js")"></script>
//Initializes Map
<div ng-app="SyncApp">
<div ng-controller="MapController">
<div id="map" style="height: inherit; min-height: 356px;" ej-map e-
zoomsettings- enablezoom="enableZoom">
<div e-layers>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div e-layer e-shapedata="shapeData" e-shapesettings-fill="fill" e-
shapesettings-strokethickness="strokeThickness" e-shapesettings-
stroke="stroke">
</div>
</div>
</div>
Shape Color:  <input type="text" id="Text11" ng-model="nfill" style="width:
110px">
<div>
</div>
angular.module('SyncApp', ['ejangular'])
.controller('MapController', function ($scope)
{
$scope.enableZoom = true,
$scope.shapeData = usMap;
$scope.fill = "#9CBF4E";
$scope.strokeThickness = "0.5";
$scope.stroke = "white";
});
```



RTL

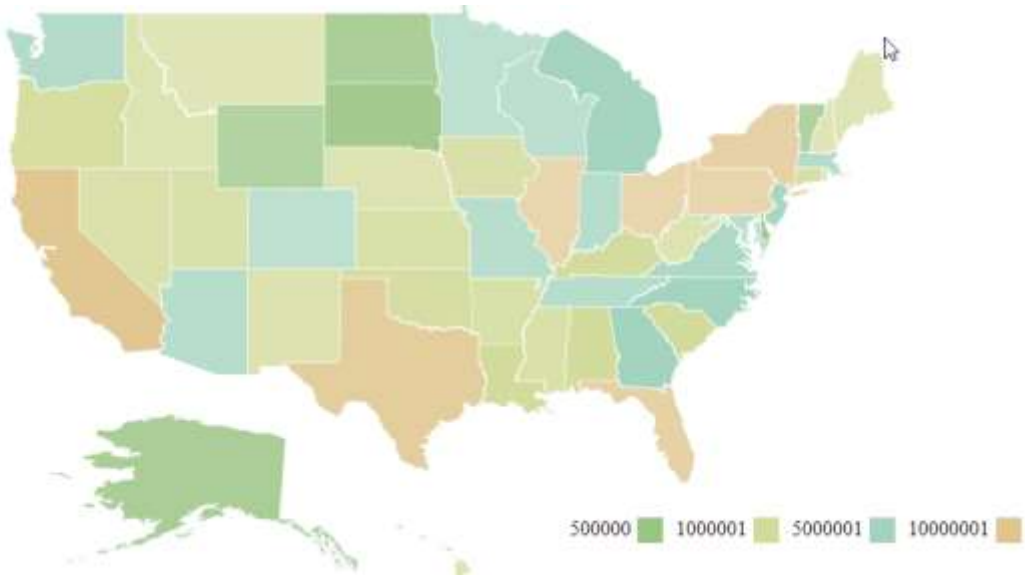
Right-to-Left or RTL describes the ability of application to handle and responds you to communicate with a right-to-left language, such as Arabic or Japanese. enableRTL property is used to change the rendering format to “Right-to-Left”. By default, legend and tooltip will be rendered as “Left-to-Right” in Maps.

ASPX-CS

```
@(Html.EJ().Map("maps")
.EnableRTL(true)
)
```

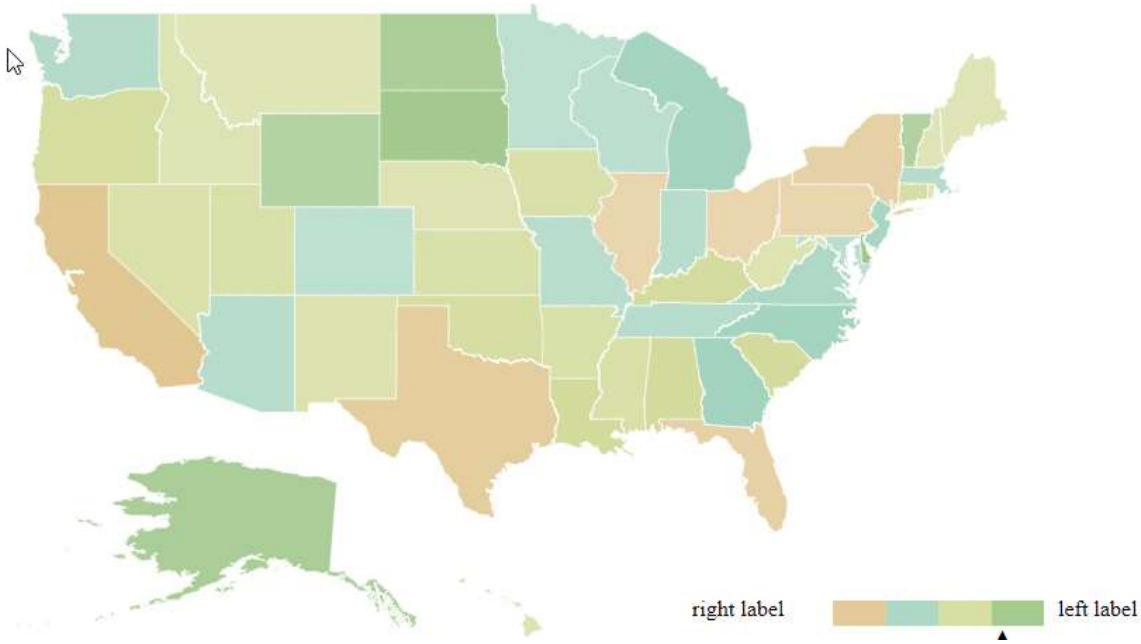
In default legend, after setting EnableRTL is set as true, legend text will be rendered first, and then legend icon will be rendered.

Copyright © 2001 - 2020 Syncfusion Inc.



Interactive legend

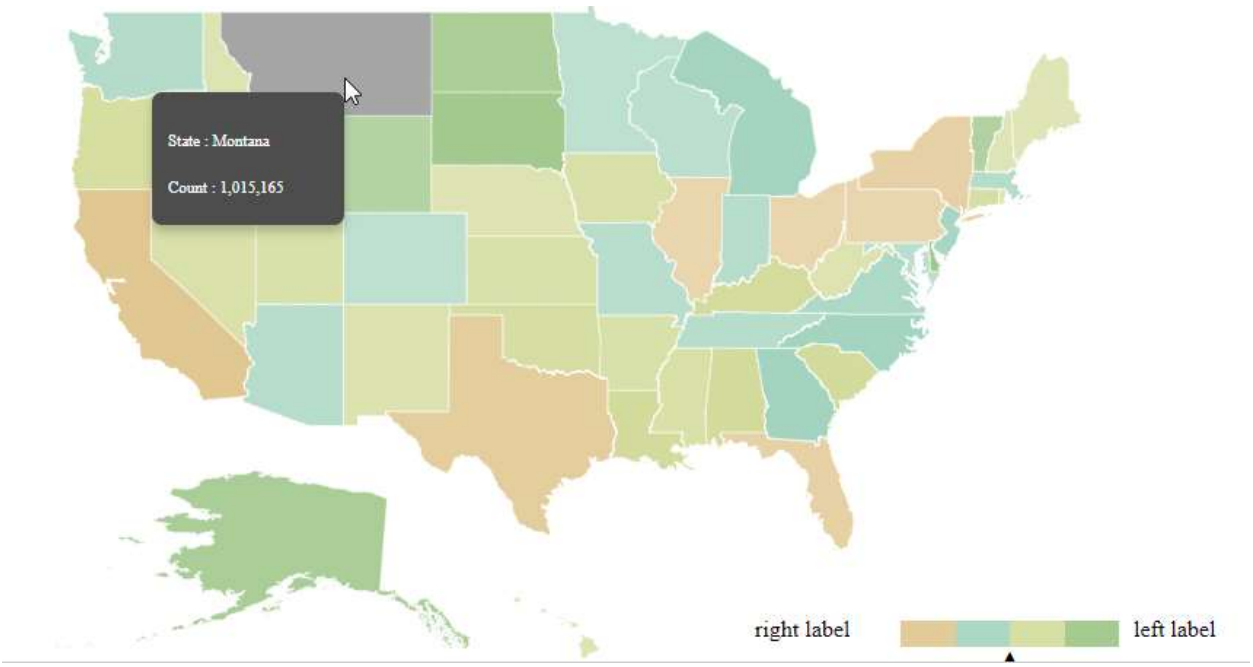
If EnableRTL is set as true, then whole interactive legend will be rendered from right-to-left, which means left label will be rendered at right side, and right label will be rendered at left side. Also, navigation arrow position will be changed to right side.



Tooltip

If EnableRTL is set as true, tooltip will be rendered at the left side of the mouse cursor.

Copyright © 2001 - 2020 Syncfusion Inc.



Methods

navigateTo(latitude, longitude, level)

Method for navigating to specific shape based on latitude, longitude and zoom level, you can use navigateTo method.

ASPX-CS

```
@(Html.EJ().Map("Map1"))
```

JS

```
var map = $("#Map1").data("ejMap");
map.navigateTo();
```

pan(direction)

Method to perform map panning, you can use pan method.

ASPX-CS

```
@(Html.EJ().Map("Map1"))
```

JS

```
var map = $("#Map1").data("ejMap");
map.pan();
```

refresh()

Method to reload the map, you can use refresh method.

ASPX-CS

```
@(Html.EJ().Map("Map1"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
var map = $("#Map1").data("ejMap");
map.refresh();
```

refreshLayers()

Method to reload the shapeLayers with updated values, you can use refreshLayers method.

ASPX-CS

```
@(Html.EJ().Map("Map1"))
```

JS

```
var map = $("#Map1").data("ejMap");
map.refreshLayers();
```

refreshNavigationControl(navigation)

Method to reload the navigation control with updated values, you can use refreshNavigationControl method.

ASPX-CS

```
@(Html.EJ().Map("Map1"))
```

JS

```
var map = $("#Map1").data("ejMap");
map.refreshNavigationControl();
```

zoom(level, isAnimate)

Method to perform map zooming, you can use zoom method.

ASPX-CS

```
@(Html.EJ().Map("Map1"))
```

JS

```
var map = $("#Map1").data("ejMap");
map.zoom();
```

Events

MarkerSelected

Triggered on selecting the map markers, you can use MarkerSelected event.

ASPX-CS

```
@(Html.EJ().Map("container")
.MarkerSelected("MarkerSelected")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
function MarkerSelected(){
// Do Something
}
```

Mouseleave

Triggers while leaving the hovered map shape, you can use Mouseleave event.

ASPX-CS

```
@(Html.EJ().Map("container")
.Mouseleave("MouseLeave")
)
```

JS

```
function MouseLeave(){
// Do Something
}
```

Mouseover

Triggers while hovering the map shape, you can use mouseover event.

ASPX-CS

```
@(Html.EJ().Map("container")
.Mouseover("MouseOver")
)
```

JS

```
function MouseOver(){
// Do Something
}
```

OnRenderComplete

Triggers once map render completed, you can use OnRenderComplete event.

ASPX-CS

```
@(Html.EJ().Map("container")
.OnRenderComplete("RenderComplete")
)
```

JS

```
function RenderComplete(){
// Do Something
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Panned

Triggers when map panning ends, you can use Panned event.

ASPX-CS

```
@(Html.EJ().Map("container")
.Panned("Panned")
)
```

JS

```
function Panned(){
// Do Something
}
```

ShapeSelected

Triggered on selecting the map shapes, you can use ShapeSelected event.

ASPX-CS

```
@(Html.EJ().Map("container")
.ShapeSelected("ShapeSelected")
)
```

JS

```
function ShapeSelected(){
// Do Something
}
```

ZoomedIn

Triggered when map is zoomed-in, you can use ZoomedIn event.

ASPX-CS

```
@(Html.EJ().Map("container")
.ZoomedIn("ZoomedIn")
)
```

JS

```
function ZoomedIn(){
// Do Something
}
```

ZoomedOut

Triggers when map is zoomed out, you can use ZoomedOut event.

ASPX-CS

```
@(Html.EJ().Map("container")
.ZoomedOut("ZoomedOut")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
function ZoomedOut(){
// Do Something
}
```

ShapeRendering

The ShapeRendering event is triggered while rendering each shape in the map.

ASPX-CS

```
@(Html.EJ().Map("container")
.ShapeRendering("shape")
)
```

JS

```
function shape(){
// Do Something
}
```

BubbleRendering

The BubbleRendering event is triggered while rendering each bubble in the map.

ASPX-CS

```
@(Html.EJ().Map("container")
.BubbleRendering("bubble")
)
```

JS

```
function bubble(){
// Do Something
}
```

LegendItemRendering

The LegendItemRendering event is triggered before rendering each legend in the map.

ASPX-CS

```
@(Html.EJ().Map("container")
.LegendItemRendering("legend")
)
```

JS

```
function legend(){
// Do Something
}
```

Click

Triggers while clicking on the layers of the map, you can use Click event.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Map("container")
.Click("click")
)
```

JS

```
function click(){
// Do Something
}
```

DoubleClick

Triggers while double clicking on the layers of the map, you can use DoubleClick event.

ASPX-CS

```
@(Html.EJ().Map("container")
.DoubleClick("doubleClick")
)
```

JS

```
function doubleClick(){
// Do Something
}
```

RightClick

Triggers while right clicking on the layers of the map, you can use RightClick event.

ASPX-CS

```
@(Html.EJ().Map("container")
.RightClick("rightClick")
)
```

JS

```
function rightClick(){
// Do Something
}
```

Load

Triggers before loading the map, you can use Load event.

ASPX-CS

```
@(Html.EJ().Map("container")
.Load("load")
)
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
function load(){
// Do Something
}
```

MaskEdit

MaskEdit

The MaskEdit control provides an easy and reliable way of collecting user input and displaying standard data in a specific format. Some common uses of the MaskEdit control are IP address editors, phone number editors, and Social Security number editors.

Key Features

- Mask: The format where the value should be entered in the text box can be set using this property.
- Input Mode: This feature allows the MaskEdit text box to act as an input text box or password text box.
- Watermark Text:The MaskEdit text box supports watermarking. A watermark is background text that appears in the text box without interfering with user text entry or the readability of the text entered. It can be used to display a ready instruction or other important information for the user. It appears automatically before the text is entered and disappears once the user begins entering text.
- Theme: Essential ASP.NET MVC controls feature 12 built-in themes: six flat themes and six with gradient effects. They also support custom skin options for user-defined themes.

Getting Started

This section explains you briefly on how to create a MaskEdit control in your ASP.NET MVC application.

Create your first MaskEdit Widget in MVC

ASP.NET MVC MaskEdit control allows you to set the type and format of the input mask that is used in the textbox and also the number of place holders. Using the following guidelines, you can create MaskEdit control for a real-time payment application.

The following screenshot illustrates the functionality of a MaskEdit. Using MaskEdit control textbox, you can enter only the assigned text format and no other formats. The input mask prevents you from entering invalid characters into the control. In this application, MobileNumber textbox has a mask value.



MaskEdit

In the above screenshot, you can type only numbers and it does not allow text format.

Create a MaskEdit control

ASP.NET MVC MaskEdit control renders built-in features like text masking, number masking and flexible APIs. You can easily create the MaskEdit control using simple HTML helper class as follows:

Copyright © 2001 - 2020 Syncfusion Inc.

1. Create an MVC Project and add necessary assemblies, styles and scripts to it. Refer MVC-Getting Started.
2. Add the following code to the corresponding view page to render MaskEdit.

ASPX-CS

```
<div class="frame">
<div class="control">
<table class="editors">
<tbody>
<tr>
<td>
<label>
Kilometers</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").Value("1000")
</td>
</tr>
<tr>
<td>
<label>
Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").Value("100")
</td>
</tr>
<tr>
<td>
<label>
Fare</label>
</td>
<td>
@Html.EJ().CurrencyTextbox("currency").Value("50")
</td>
</tr>
<tr>
<td>
<label>
Mobile No</label>
</td>
<td>
@*creating MaskEdit control*@
@Html.EJ().MaskEdit("maskedit").MaskFormat("99-999-
99999").InputMode(InputMode.Text)
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("PayBill")
</div>
</div>
</div>
```

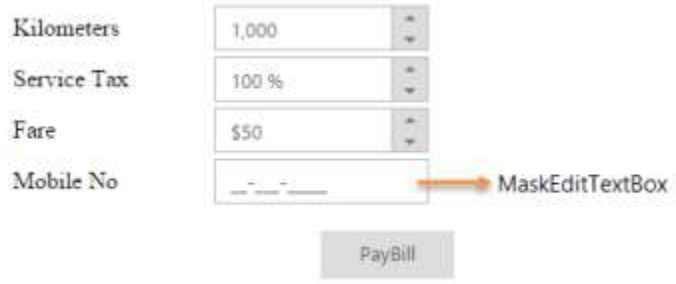
Copyright © 2001 - 2020 Syncfusion Inc.

3. Add the following styles to show MaskEdit and place it in a particular position.

CSS

```
<style>
.frame
{
width: 300px;
}
.editors
{
max-width: 400px;
}
.control .pay-bill
{
margin-left: 208px;
margin-top: 15px;
}
.editors label
{
display: block;
width: 130px;
}
.control
{
margin-top: 10px;
}
.ctrl-label
{
padding-bottom: 3px;
}
</style>
```

Execute the above code example to render the following output.



MaskEdit

Set Mask value to Mobile Number textbox

In this section, you can learn how to set mask value for MobileNumber textbox. To achieve this, set the mask value in the MaskEdit control to the desired values.

@Html.EJ().MaskEdit("maskedit").MaskFormat("99-999-99999").InputMode(InputMode.Text)

Copyright © 2001 - 2020 Syncfusion Inc.

Create a MaskEdit control for Product Key Validation

Create a MaskEdit

You can easily create the MaskEdit control using simple HTML helper class as follows.

1. Create a MVC Project and add necessary assemblies and scripts to it.

Refer MVC-Getting Started.

2. Add the following code to the corresponding view page to render MaskEdit.

ASPX-CS

```
<div class="frame">
<div class="control">
<table class="editors">
<tbody>
<tr>
<td>
<label>
Product Key</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").MaskFormat("aaaa-aaaa-aaaa-
aaaa").InputMode(InputMode.Text) @*creating MaskEdit control for product key
validation*@                          </td>
</tr>
</tbody>
</table>
<div class="product-key">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Submit")
</div>
</div>
</div>
```

3. Add the following styles to show the MaskEdit, and place it in a particular position.

CSS

```
<style>
.frame
{
width: 300px;
}
.editors
{
max-width: 400px;
}
.control .product-key
{
margin-left: 208px;
margin-top: 15px;
}
.editors label
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
display: block;
width: 130px;
}
.control
{
margin-top: 10px;
}
.control-label
{
padding-bottom: 3px;
}
</style>
```

4. Run the above code example to render the following output.



Product Key

Set Mask value to Product key textbox

You can set mask value for Product key textbox by setting the desired values to the MaskEdit control.

ASPX-CS

```
@Html.EJ().MaskEdit("maskedit").MaskFormat("aaaa-aaaa-aaaa-
aaaa").InputMode(InputMode.Text)
```

Behavior Settings

Persistence Support

MaskEditTextBox provides state maintenance support. You can maintain the previous changes made in the control after a page loads.

Configure Persistence Support

In the View page add MaskEditTextBox helper, and configure the EnablePersistence property as follows.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").EnablePersistence(true)
```

Output of MaskEditTextBox with EnablePersistence is as follows.

MaskEditTextBox at initial load

Copyright © 2001 - 2020 Syncfusion Inc.



MaskEditTextBox after changing the value and after page refresh

Enabled or Disabled

MaskEditTextBox has an option to enable or disable its element. You can set the Enabled property as “false” to enable the Textbox controls.

Configure Enabled or Disabled

In the View page add MaskEditTextBox helper, and configure the Enabled property.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").Enabled(false).MaskFormat("99-999-99999")
```

Output when Enabled is “true” and when Enabled is “false”.

Textboxes with enabled as true



Textboxes with enabled as false

Adjusting Textbox Size

MaskEditTextBox size can be modified by using the Height and Width properties. You can customize the size of Textbox by using these properties.

Configure Height and Width

MaskEditTextBox size can be modified by using the Height and Width properties.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").Width("100").Height("50")
```

Output of MaskEditTextBox after setting “Height” and “Width” is as follows.



MaskEditTextBox with height and width

Define Value

The value of MaskEditTextBox can be assigned by using the Value property. The default value for Value property is null. Specify the name attribute value for the mask edit textbox.

You can get the raw value of **MaskEdit** without literals and prompt characters by using the get_StrippedValue method.

Also you get the value of **MaskEdit** with the masked format by using the get_UnstrippedValue method.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Value

In the View page add MaskEditTextBox helper, and configure the Value property

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-99999").Value("1234567890")
```

Output of MaskEditTextBox with the value property is as follows.



MaskEditTextBox with value

Read Only Support

MaskEditTextBox supports read only option. When you enable the ReadOnly property to the control, the value cannot be changed in the MaskEditTextBox. You can set the ReadOnly property as “true” to enable this option.

Configure Read Only

In the View page add MaskEditTextBox helper, and configure the ReadOnly property.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").Value("123456").ReadOnly(true)
```

Output of MaskEditTextBox when ReadOnly is “true” is as follows. MaskEditTextBox values cannot be edited or changed.



MaskEditTextBox with readOnly

Error Visibility

The MaskEditTextBox has an option that shows the error value with red colored text. It is used to validate the Mask Edit value. You can set the ShowError property as “true” to enable this option.

Configure Error Visibility

In the View page use the corresponding Textbox helper for rendering Textbox controls.

ASPX-CS

```
@* Add the following code in your view page to render Textbox controls.*@
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").Value("123456").ShowError(true)
```

Output for MaskEditTextBox when ShowError is “true” is as follows.



Textbox with showError

Copyright © 2001 - 2020 Syncfusion Inc.



Textbox without error

Mask Edit Properties

Custom Character

The MaskEdit allows you to use the custom character option. The specified character is allowed to enter in the Mask Edit Textbox by using the CustomCharacter property.

Hide Prompt On Leave

The MaskEditTextBox provides the option to hide the prompt when you focus out from the control. The mask prompt is visible when you focus again to the control. The default value of HidePromptOnLeave is false.

Input Mode

The MaskEdit supports two type of inputs such as text and password that have been assigned by using the enum values ej.InputMode.Text and ej.InputMode.Password. The default value for InputMode is text in MaskEdit.

Mask Format

The MaskEdit provides the option to define the MaskFormat to the value. The default value for MaskFormat property is empty string.

The following steps explain the implementation of MaskEdit Properties.

In the View page use the corresponding Textbox helper for rendering MaskEdit control.

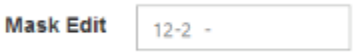
ASPX-CS

```
@* Add the following code in your view page to render Textbox controls.*@
@{Html.EJ().MaskEdit("mask")
.MaskFormat("99-999-CCCC").CustomCharacter("r").
HidePromptOnLeave(true).InputMode(InputMode.Text).Render();}
```

The output for MaskEditTextBox with its properties is as follows.



MaskEdit with MaskFormat



MaskEdit with HidePromptOnLeave



MaskEdit with prompt focus



Copyright © 2001 - 2020 Syncfusion Inc.

MaskEdit with InputMode text



MaskEdit with CustomCharacter

Appearance

Theme

The MaskEditTextBox control's style and appearance can be controlled based on CSS classes. In order to apply styles to the Textbox control, you need to refer 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. If the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Textbox control namely:

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

CSS Class

The CSS can be customized by using the CssClass in the MaskEditTextBox. You can customize the MaskEditTextBox with CssClass property to appear like your desired control.

Configure CSS Class

1. In the View page add MaskEditTextBox helper, and configure the CssClass property.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").Value("123456").CssClass("customCss")
```

2. Customize the CSS properties in custom CSS class.

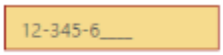
CSS

```
<style>
.customCss .e-box
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
border-color: #9d241b;
}
.customCss .e-input
{
background-color: #f6db8d;
}
.customCss .e-select
{
background-color: #ecf6ac;
border-color: #3c36e7;
}
</style>
```

The output for MaskEditTextBox after applying CssClass is as follows.



Textboxes with cssClass

Rounded Corner Support

MaskEditTextBox provides you with rounded corner support whose appearance is different from normal textbox controls.

Configure Rounded Corner Support

In the View page add MaskEditTextBox helper, and configure the ShowRoundedCorner property.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").Value("123456").ShowRoundedCorner(true)
```

Output of MaskEditTextBox when ShowRoundedCorner is “true”.



Textboxes with showRoundedCorner

Water Mark Text Support

The Textboxes provide water mark text support. You can display the initial value in the control by water mark.

Configure Water Mark Text

In the View page use the corresponding Textbox helper for rendering Textbox controls.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").WatermarkText("MaskEdit")
```

Output of MaskEditTextBox when ShowRoundedCorner is “true”.



MaskEditTextBox with watermark text

Copyright © 2001 - 2020 Syncfusion Inc.

Text Alignment Support

The MaskEditTextBox provides text alignment support that allows you to set the alignment of text in the control by using the TextAlign property.

Configure Text Alignment

In the View page use the corresponding MaskEdit helper for rendering MaskEdit control.

ASPX-CS

```
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").Value("12345677").TextAlign(TextAlign.Right)
```

The output for Textboxes when TextAlign is set to “right”.



MaskEditTextBox with textAlign

HTML Attributes Support

The MaskEditTextBox provides support for adding HTML attributes to the component. ‘HtmlAttributes’ property is used to add HTML attributes like, id, class etc.. to the components. We need to use IDictionary<string,object> to specify the HTML attributes.

Configure HTMLAttributes property

1. In the View page add MaskEditTextBox helper, and configure the HtmlAttributes property. Here we have added the Access key attribute. While pressing the “AccessKey” and “J” keys, MaskEditTextBox will gain focus.

ASPX-CS

```
@{IDictionary<string, object> maskAttribute = new Dictionary<string,
object>();
maskAttribute.Add("accesskey", "j");
}
@Html.EJ().MaskEdit("mask").MaskFormat("99-999-
99999").HtmlAttributes(maskAttribute)
```

2. Add the following JavaScript code to focus the MaskEditTextBox

JS

```
<script type="text/javascript">
$(function ()
{
$(document).on("keydown", function (e)
{
if (e.altKey && e.keyCode === 74)
{ // j- key code.
$("#mask").focus();
}
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
</script>
```

The output for MaskEditTextBox after configuring the HtmlAttributes property



MaskEditTextBox with focus

Globalization Support in MaskEdit

We have provided the globalization support in **MaskEdit** control. Our **MaskEdit** control mainly rendered based on the **maskFormat** property, so we have provided the globalization support based on the maskFormat literals. We have given this globalization support option on below maskFormat literals in **MaskEdit** control. You can change the globalization by using the locale property. The default value for **locale** property is 'en-US' in **MaskEdit** control.

Formats	Description
$	Currency symbol value will be changed based on the corresponding culture.
.	Decimal Separator value will be changed based on the corresponding culture.
,	Thousand Separator will be changed based on the corresponding culture.

To know more about EJ globalize support, please refer the below link

https://help.syncfusion.com/aspnetmvc/globalization

The following example describes the way to use localization in MaskEdit.

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

JAVASCRIPT

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/i18n/ej.culture.de-DE.min.js"></script>
```

ASPX-CS

```
<label for="mask">Mask Edit</label>
@(Html.EJ().MaskEdit("maskedit").InputMode(InputMode.Text).MaskFormat("$999,
999.99").Width("100%").Locale("de-DE"))
```

Output of MaskEdit with localization.



MaskEdit with de-DE locale



MaskEdit with default (en-US) locale

Menu

Menu

The Menu control supports displaying a Menu created from list items. The Menu is based on a hierarchy of Parent and Child elements where the Child items are rendered as sub-menu items.

Key Features

- UL/LI template: Requires UL and LI list of items as input.
- Orientation: Vertical and horizontal orientation support.
- Context menu: A Menu that is displayed wherever you right-click on the page or on the control.
- Keyboard shortcuts: Allows easy access to Menu items through shortcut keys.
- Data binding: Supports data binding Menu items as remote or local data.
- Theme: Essential ASP.NET MVC controls feature 12 built-in themes, six flat and six with gradient effects, and also supports custom skin options for user-defined themes.
- Keyboard navigation: You can interact with the Menu control using the keyboard.
- RTL support: This feature allows text in Menu items to be displayed from right to left.
- Center Menu: This feature allows you to center-align root Menu items.
- Separators and open-on-click support.

Getting Started

This section explains briefly about how to create a Menu control in your application with ASP.NET MVC

Create Syncfusion Menu in MVC

The Essential ASP.NET MVC Menu control supports displaying a Menu out of list items. The Menu is based on parent and child elements hierarchy, where the child items are rendered as the sub-menu items .The Menu control can also render with local and remote data source. The following section explains you on how to customize the Menu control for a website.

The following screenshot illustrates the appearance of Syncfusion’s website Menu.



Copyright © 2001 - 2020 Syncfusion Inc.

Syncfusion Menu

The Menu items in the above screenshot allow you to navigate through multiple menus in a page and select an item. It has a hierarchical structure with sub menus.

Create a Menu

Essential ASP.NET MVC Menu widget basically renders with built-in features like keyboard navigation, show and hides Menu items with animations and flexible API's. Refer the following guidelines to render Menu control with Local data source value.

1. You can create a MVC Project and add the necessary Dll's and scripts using the MVC-Getting Started Documentation.
2. Add the below code in your view page to add the necessary CSS and script files

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/13.1.0.21/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<!--Scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jquery.easing.1.3.min.j
s"></script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.web.all.min.js">
</script>
</head>
```

3. Add the following code example to the corresponding view page for Menu rendering.

ASPX-CS

```
@Html.EJ().Menu("SyncfusionProducts")
```

4. Execute the above code to render Empty Menu bar.

Essential ASP.NET MVC Menu without menu item

Configure Parent Menu items

Each Menu consists of a list of Menu items with list of sub level Menu item. Refer the following guidelines to initialize the root level elements of Menu control with Remote data source value. RootLevelItems data service is created to define the root level Menu items, sub items and InnerItems data services to initialize the sub level and inner sub levels and both can be referred from the following service location. In Menu Widgets mention the RootLevelItem Data Source in the Datasource property.

Copyright © 2001 - 2020 Syncfusion Inc.

Elements's properties like Id, Text, URL, and Parent Id can be defined using our menu fields and it explained briefly under the concept and features of Menu control.

http://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/

To navigate the clicked Menu item to a specific URL, define the navigation URL to each Menu item.

Initialize the Menu with data source value as follows.

ASPX-CS

```
@Html.EJ().Menu("syncfusionProducts").Items(items =>
{
items.Add().Url("#").Id("Products").Text("Products");
items.Add().Url("").Text("Support");
items.Add().Url("").Id("Purchase").Text("Purchase");
items.Add().Url("").Id("Downloads").Text("Downloads");
items.Add().Id("Resources").Url("").Text("Resources");
items.Add().Url("").Id("Company").Text("Company");
})
```

Execute the above code to display the Root level Menu items.



Essential ASP.NET MVC Menu without sub menu item

Initialize sub-level Menu items

Each Menu items consist of list of sub level Menu items. Refer the following guidelines to initialize the sub level items of Menu control. The ParentId field property maps root level Menu item to its sub level Menu item. In Menu Widgets mention the RootLevelItems and SubLevelItems Data Source in the Datasource property. The Child field property is used to define sub level Menu items of parent Menu item.

The following code example explains the initialization of first level sub menu items of Menu control.

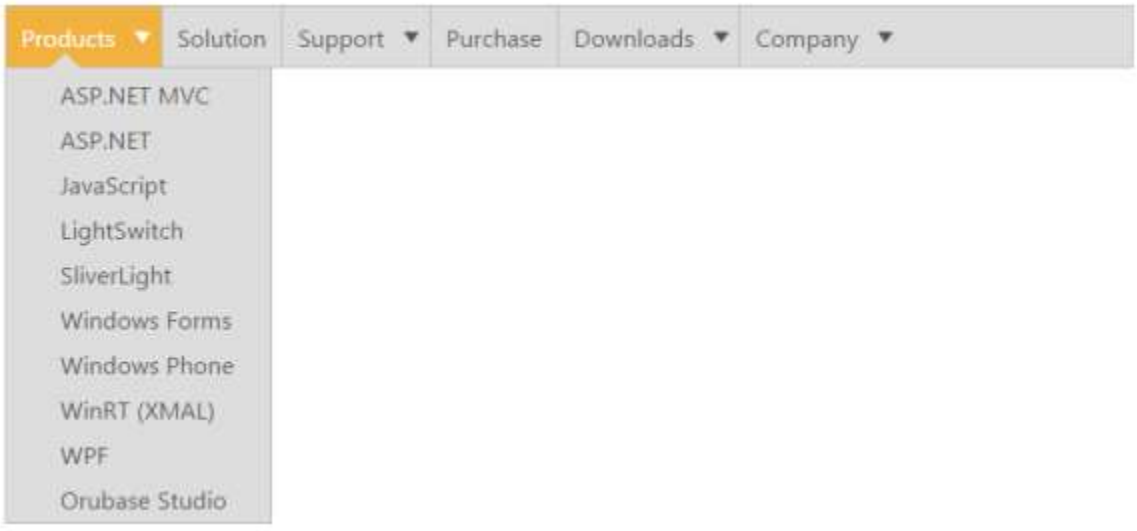
ASPX-CS

```
@Html.EJ().Menu("syncfusionProducts").Items(items =>
{
items.Add().Url("#").Id("Products").Text("Products").Children(child =>
{
child.Add().Url("").Text("ASP.NET");
child.Add().Url("").Text("ASP.NET MVC");
child.Add().Url("").Text("Mobile MVC");
child.Add().Url("").Text("Silverlight");
child.Add().Url("").Text("Windows Forms");
child.Add().Url("").Text("Windows Phone");
child.Add().Url("").Text("WinRT (XMAL)");
child.Add().Url("").Text("WPF");
child.Add().Url("").Text("Orubase Studio");
child.Add().Url("").Text("Metro Studio");
child.Add().Url("").Text("What's New");
});
items.Add().Url("").Text("Support").Children(child =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
child.Add().Url("").Text("Direct-Trac Support");
child.Add().Url("").Text("Community Forums");
child.Add().Url("").Text("Knowledge Base");
child.Add().Url("").Text("Online Documentation");
child.Add().Url("").Text("Services");
});
items.Add().Url("").Id("Purchase").Text("Purchase");
items.Add().Url("").Id("Downloads").Text("Downloads").Children(child =>
{
child.Add().Url("").Text("Evaluation");
child.Add().Url("").Text("Free E-Books");
child.Add().Url("").Text("Metro Studio");
child.Add().Url("").Text("Latest Version");
child.Add().Url("").Text("Version History");
});
items.Add().Id("Resources").Url("").Text("Resources").Children(child =>
{
child.Add().Url("").Text("Technology Resource Portal");
child.Add().Url("").Text("Case Studies");
child.Add().Url("").Text("Boucher & DataSheet");
child.Add().Url("").Text("FAQ");
});
items.Add().Url("").Id("Company").Text("Company").Children(child =>
{
child.Add().Url("").Text("About Us");
child.Add().Url("").Text("Company Blog");
child.Add().Url("").Text("Technical Blog");
child.Add().Url("").Text("Newsletter");
child.Add().Url("").Text("Partners");
child.Add().Url("").Text("Locations");
child.Add().Url("").Text("Contact Us");
child.Add().Url("").Text("Careers");
});
})
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Essential ASP.NET MVC Menu with sub menu item

Define multiple level Menu items

You can render sub menu item to multiple level in Menu control. In Menu Widgets, mention the InnerItems Data Source in the Datasource property. The Child field property is used to define the sub level Menu item of parent Menu. Specify the ParentId value to render sub level Menu item for the Menu item. Each Level Menu item have Child field to define the child level Menu item.

The following code example explains the initialization of multiple level sub menu items.

ASPX-CS

```
@Html.EJ().Menu("syncfusionProducts").Items(items =>
{
items.Add().Url("#").Id("Products").Text("Products").Children(child =>
{
child.Add().Url("").Text("ASP.NET");
child.Add().Url("").Text("ASP.NET MVC");
child.Add().Url("").Text("Mobile MVC");
child.Add().Url("").Text("Silverlight");
child.Add().Url("").Text("Windows Forms");
child.Add().Url("").Text("Windows Phone");
child.Add().Url("").Text("WinRT (XMAL)");
child.Add().Url("").Text("WPF");
child.Add().Url("").Text("Orubase Studio");
child.Add().Url("").Text("Metro Studio");
child.Add().Url("").Text("What's New").Children(child1 =>
{
child1.Add().Url("").Text("ASP.NET");
child1.Add().Url("").Text("WPF");
child1.Add().Url("").Text("Silverlight");
child1.Add().Url("").Text("Windows Forms");
child1.Add().Url("").Text("Windows Phone");
child1.Add().Url("").Text("ASP.NET MVC");
child1.Add().Url("").Text("ASP.NET");
});
});
items.Add().Url("").Text("Support").Children(child =>
{
child.Add().Url("").Text("Direct-Trac Support");
child.Add().Url("").Text("Community Forums");
child.Add().Url("").Text("Knowledge Base");
child.Add().Url("").Text("Online Documentation");
child.Add().Url("").Text("Services").Children(child1 =>
{
child1.Add().Url("").Text("Consulting");
child1.Add().Url("").Text("Training");
});
});
items.Add().Url("").Id("Purchase").Text("Purchase");
items.Add().Url("").Id("Downloads").Text("Downloads").Children(child =>
{
child.Add().Url("").Text("Evaluation");
child.Add().Url("").Text("Free E-Books");
child.Add().Url("").Text("Metro Studio");
child.Add().Url("").Text("Latest Version");
child.Add().Url("").Text("Version History");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
items.Add().Id("Resources").Url("").Text("Resources").Children(child =>
{
child.Add().Url("").Text("Technology Resource Portal").Children(child1 =>
{
child1.Add().Url("").Text("E-Books");
child1.Add().Url("").Text("White Papers");
});
child.Add().Url("").Text("Case Studies");
child.Add().Url("").Text("Boucher & DataSheet");
child.Add().Url("").Text("FAQ");
});
items.Add().Url("").Id("Company").Text("Company").Children(child =>
{
child.Add().Url("").Text("About Us").Children(child1 =>
{
child1.Add().Url("").Text("More About Us");
child1.Add().Url("").Text("Management");
child1.Add().Url("").Text("News & Events");
child1.Add().Url("").Text("Customer Quotes");
child1.Add().Url("").Text("Customer Lists");
child1.Add().Url("").Text("Case Studies");
child1.Add().Url("").Text("Awards");
child1.Add().Url("").Text("Media Kit");
});
child.Add().Url("").Text("Company Blog");
child.Add().Url("").Text("Technical Blog");
child.Add().Url("").Text("Newsletter");
child.Add().Url("").Text("Partners").Children(child1 =>
{
child1.Add().Url("").Text("Technology Partners");
child1.Add().Url("").Text("Training Partners");
child1.Add().Url("").Text("Consulting Partners");
});
child.Add().Url("").Text("Locations").Children(child1 =>
{
child1.Add().Url("").Text("RDU");
child1.Add().Url("").Text("Chennai");
});
child.Add().Url("").Text("Contact Us");
child.Add().Url("").Text("Careers");
});
})
```

Execute the above code to render the multiple sub menus in a hierarchy

Copyright © 2001 - 2020 Syncfusion Inc.



Essential ASP.NET MVC Menu with multiple level sub menu item

Orientation

It gets or sets the direction in which the Menu control renders and specifies the orientation of the normal menu. According to the orientation property the Menu control renders in horizontal or vertical.

Horizontal Menu

Horizontal orientation displays the menu items horizontally and it is the default orientation behavior of Menu control.

The following steps explains you the details on rendering the Menu control.

1. Add the below code in your view page to render the horizontal menu

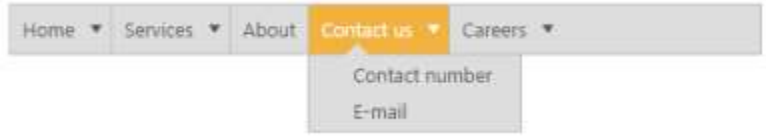
ASPX-CS

```
// In the CSHTML page, add the following code to configure Menu control.
<div class="imgframe">
@Html.EJ().Menu("menuControl").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500")
</div>
```

The following screenshot displays the output of the above code.



Horizontal Menu

Vertical Menu

You can also render Menu control in vertical direction using orientation.To set the vertical orientation of Menu control, replace the following script in the above sample code example.

1. Add the following code in your view page to render vertical menu

ASPX-CS

```
// Add the following code in the CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("menuControl").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("200").Orientation(Orientation.Vertical)
</div>
```

The following screen shot displays the output of the above code.



Vertical Menu

Data binding

Data binding enables you to synchronize the elements with different sources of data. You can bind data using two ways, Local data and remote data.

Field Members

Field is a property that includes the object type. Fields are used to bind the data source and it includes following field members to make binding easier.

List of Field members

Name	Description
DataSource	Datasource receives Essential DataManager object and JSON object.
Query	It receives query to retrieve data from the table (query is same as SQL). Example: ej.Query().from("Categories").select("CategoryID,CategoryName").take(3);
TableName	It receives table name to execute query on the corresponding table
Id	Specifies the id to menu items list
ParentId	Specifies the parent id of the table.
Text	Specifies the text of menu items list
SpriteCssClass	Specifies the sprite CSS class to â€œLIâ€• item list
LinkAttribute	Specifies the link attribute to â€œAâ€• tag in item list
ImageAttribute	Specifies the image attribute to â€œIMGâ€• tag inside items list
HtmlAttribute	Specifies the HTML attributes to â€œLIâ€• item list

Copyright © 2001 - 2020 Syncfusion Inc.

ImageUrl	Specifies the image URL to â€œIMGâ€• tag inside item list.

Local data

To define the local data to the Menu control, map the user-defined JSON data names with its appropriate dataSource column names.

1. Add the following code in your view page to render menu with local data

ASPX-CS

```
// Add the following code in your CSHTML page.
@Html.EJ().Menu("menu").MenuFields(f =>
f.Datasource((IEnumerable<Check.Controllers.CheckController.MenuJson>)Vie
wBag.datasource)
.Id("id").Text("text").ParentId("parent"))
```

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Check.Controllers
{
public class CheckController : Controller
{
public class MenuJson
{
public string text { get; set; }
public int id { get; set; }
public string parent { get; set; }
}
List<MenuJson> menu = new List<MenuJson>();
public ActionResult DataBindingJson()
{
menu.Add(new MenuJson { id = 1, text = "Group A", parent = null });
menu.Add(new MenuJson { id = 2, text = "Group B", parent = null });
menu.Add(new MenuJson { id = 3, text = "Group C", parent = null });
menu.Add(new MenuJson { id = 4, text = "Group D", parent = null });
menu.Add(new MenuJson { id = 5, text = "Group E", parent = null });
menu.Add(new MenuJson { id = 11, parent = "1", text = "Algeria" });
menu.Add(new MenuJson { id = 12, parent = "1", text = "Armenia" });
menu.Add(new MenuJson { id = 13, parent = "1", text = "Bangladesh" });
menu.Add(new MenuJson { id = 14, parent = "1", text = "Cuba" });
menu.Add(new MenuJson { id = 15, parent = "2", text = "Denmark" });
menu.Add(new MenuJson { id = 16, parent = "2", text = "Egypt" });
menu.Add(new MenuJson { id = 17, parent = "3", text = "Finland" });
menu.Add(new MenuJson { id = 18, parent = "3", text = "India" });
menu.Add(new MenuJson { id = 19, parent = "3", text = "Malaysia" });
menu.Add(new MenuJson { id = 20, parent = "4", text = "New Zealand" });
menu.Add(new MenuJson { id = 21, parent = "4", text = "Norway" });
menu.Add(new MenuJson { id = 22, parent = "4", text = "Romania" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
menu.Add(new MenuJson { id = 23, parent = "5", text = "Singapore" });
menu.Add(new MenuJson { id = 24, parent = "5", text = "Thailand" });
menu.Add(new MenuJson { id = 25, parent = "5", text = "Ukraine" });
menu.Add(new MenuJson { id = 26, parent = "11", text = "First Place" });
menu.Add(new MenuJson { id = 27, parent = "12", text = "Second Place" });
menu.Add(new MenuJson { id = 28, parent = "13", text = "Third place" });
menu.Add(new MenuJson { id = 29, parent = "14", text = "Fourth Place" });
menu.Add(new MenuJson { id = 30, parent = "15", text = "First Place" });
menu.Add(new MenuJson { id = 31, parent = "16", text = "Second Place" });
menu.Add(new MenuJson { id = 32, parent = "17", text = "Third Place" });
menu.Add(new MenuJson { id = 33, parent = "18", text = "First Place" });
menu.Add(new MenuJson { id = 34, parent = "19", text = "Second Place" });
menu.Add(new MenuJson { id = 36, parent = "21", text = "Second Place" });
menu.Add(new MenuJson { id = 37, parent = "22", text = "Third place" });
menu.Add(new MenuJson { id = 39, parent = "24", text = "First Place" });
menu.Add(new MenuJson { id = 40, parent = "25", text = "Second Place" });
ViewBag.datasource = menu;
return View();
}
}
}
```

The following screenshot displays the output of the above code.



Local data of Menu

Remote data

The Menu control also provides support for Remote data binding. Here the remote data is placed in Web service and you can render the menu items from the web service using Service URL. The data is in JSON format.

DataManager is used to manage relational data in JavaScript. DataManager uses two different classes, ej.DataManager for processing, and ej.Query for serving data. ej.DataManager communicates with data source and ej.Query generates data queries that are to be read by DataManager. To know more about DataManager and Query refer the following link location.

http://help.syncfusion.com/aspnetmvc/menu/data-binding

In the following example, http://mvc.syncfusion.com/Services/Northwnd.svc/ is used as the URL. This acts as web service that is located in the Syncfusion server. The web service with the name Northwnd.svc is used here.

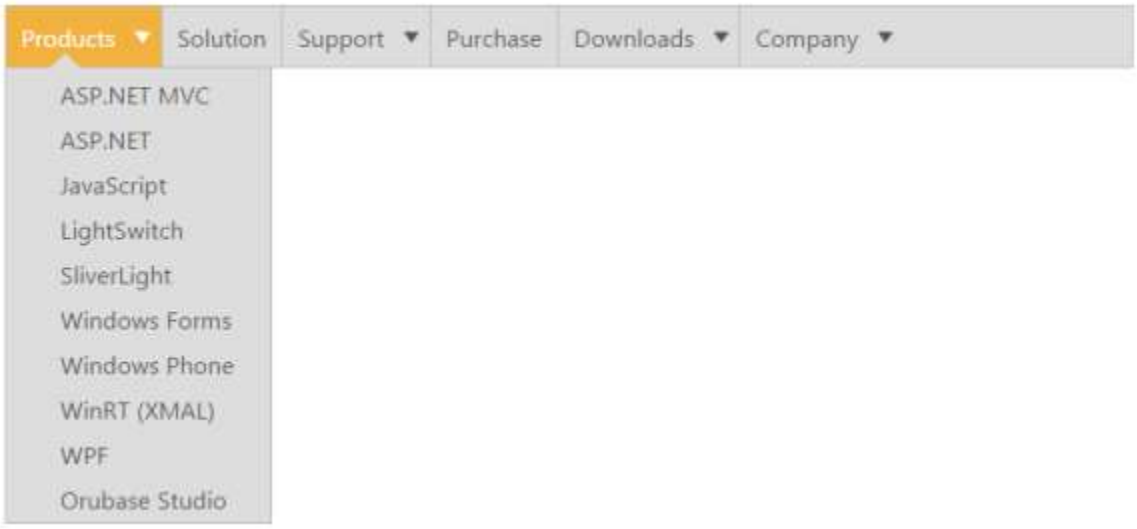
1. Add the following code in your View page to render the menu with remote data

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Menu("SyncfusionProducts").Width("600px").MenuFields(f =>
f.Datasource(d =>
d.URL("http://mvc.syncfusion.com/UGOdataServices/Northwnd.svc/")).Query("
ej.Query().from('RootLevelItems')").Id("InfoID").Text("InfoText").Child(m
enuChild =>
menuChild.Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/UGOdataServices/Northwnd.svc/")
)
.TableName("SubItems").Id("SubItemID").ParentId("InfoID").Text("SubItemte
xt").Child(subMenuChild =>
subMenuChild.Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/UGOdataServices/Northwnd.svc/")
).TableName("InnerItems").Id("InnerSubItemID").ParentId("SubItemID").Text
("InnerSubItemtext"))))
```

The following screenshot displays the output of the above code.



Remote data of Menu

Icons and navigation

Icons are the images that is displayed in the Menu control. To specify the menu with icons you can use SpriteCssClass property to display the icons.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page
@Html.EJ().Menu("menu").MenuFields(f =>
f.Datasource((IEnumerable<Check.Controllers.CheckController.icons>)ViewBa
g.datasource).Id("id").Text("text").ParentId("parent").SpriteCssClass("sp
rite"))
```

CSHARP

```
// In the controller page add the code
```

```
using System;using System.Collections.Generic;
using System.Linq;using System.Web;
using System.Web.Mvc;using Check.Models;
namespace Check.Controllers
{
public class CheckController : Controller
{
public class icons
{
public string text { get; set; }
public string sprite { get; set; }
public int id { get; set; }
public string parent { get; set; }
}
List<icons> menu = new List<icons>();
public ActionResult DataBindingJson()
{
menu.Add(new icons { id = 1, text = "Group A", parent = null });
menu.Add(new icons { id = 2, text = "Group B", parent = null });
menu.Add(new icons { id = 3, text = "Group C", parent = null });
menu.Add(new icons { id = 4, text = "Group D", parent = null });
menu.Add(new icons { id = 5, text = "Group E", parent = null });
menu.Add(new icons { id = 11, parent = "1", text = "Algeria", sprite =
"flag-dz" });
menu.Add(new icons { id = 12, parent = "1", text = "Armenia", sprite =
"flag-am" });
menu.Add(new icons { id = 13, parent = "1", text = "Bangladesh", sprite =
"flag-bangladesh" });
menu.Add(new icons { id = 14, parent = "1", text = "Cuba", sprite =
"flag-cu" });
menu.Add(new icons { id = 15, parent = "2", text = "Denmark", sprite =
"flag-denmark" });
menu.Add(new icons { id = 16, parent = "2", text = "Egypt", sprite =
"flag-eg" });
menu.Add(new icons { id = 17, parent = "3", text = "Finland", sprite =
"flag-fi" });
menu.Add(new icons { id = 18, parent = "3", text = "India", sprite =
"flag-in" });
menu.Add(new icons { id = 19, parent = "3", text = "Malaysia", sprite =
"flag-my" });
menu.Add(new icons { id = 20, parent = "4", text = "New Zealand", sprite
= "flag-new-zealand" });
menu.Add(new icons { id = 21, parent = "4", text = "Norway", sprite =
"flag-no" });
menu.Add(new icons { id = 22, parent = "4", text = "Romania", sprite =
"flag-ro" });
menu.Add(new icons { id = 23, parent = "5", text = "Singapore", sprite =
"flag-singapore" });
menu.Add(new icons { id = 24, parent = "5", text = "Thailand", sprite =
"flag-th" });
menu.Add(new icons { id = 25, parent = "5", text = "Ukraine", sprite =
"flag-ukraine" });
menu.Add(new icons { id = 26, parent = "11", text = "First Place" });
menu.Add(new icons { id = 27, parent = "12", text = "Second Place" });
menu.Add(new icons { id = 28, parent = "13", text = "Third place" });
menu.Add(new icons { id = 29, parent = "14", text = "Fourth Place" });
menu.Add(new icons { id = 30, parent = "15", text = "First Place" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
menu.Add(new icons { id = 31, parent = "16", text = "Second Place" });
menu.Add(new icons { id = 32, parent = "17", text = "Third Place" });
menu.Add(new icons { id = 33, parent = "18", text = "First Place" });
menu.Add(new icons { id = 34, parent = "19", text = "Second Place" });
menu.Add(new icons { id = 35, parent = "20", text = "First Place" });
menu.Add(new icons { id = 36, parent = "21", text = "Second Place" });
menu.Add(new icons { id = 37, parent = "22", text = "Third place" });
menu.Add(new icons { id = 38, parent = "23", text = "Third Place" });
menu.Add(new icons { id = 39, parent = "24", text = "First Place" });
menu.Add(new icons { id = 40, parent = "25", text = "Second Place" });
ViewBag.datasource = menu;
return View();
}
}
}
```

2. Add the following code in your style section.

CSS

```
<style type="text/css">
#menu
{
margin-left: 50px;
}
.e-menu li > ul > li > a
{
padding: 3px 24px 3px 35px;
}
[class^="mail-"],
[class*="mail-"]
{
background-image: url("../images/spriteimage.png");
height: 18px;
left: 2px;
top: 4px;
width: 24px;
}
.mail-dz { background-position: -68px -15px;     }
.mail-am { background-position: 91px -45px;      }
.mail-bangladesh { background-position: -98px 0;         }
.mail-cu { background-position: -607px -221px;   }
.mail-denmark { background-position: -67px -15px;     }
.mail-eg { background-position: 600px -15px;     }
.mail-fi { background-position: 12441px 12458px; }
.mail-in { background-position: -307px -103px;   }
.mail-my { background-position: 240px -102px;    }
.mail-new-zealand { background-position: -100px -45px;     }
.mail-no { background-position: -69px -45px;     }
.mail-pl { background-position: -129px -45px;    }
.mail-ca { background-position: -1345px -387px;  }
.mail-cn { background-position: -427px -42px;    }
.mail-ee { background-position: -706px -15px;    }
.mail-es { background-position: -1157px -43px    }
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays the output for the above code.



Menu with Icons

Navigation

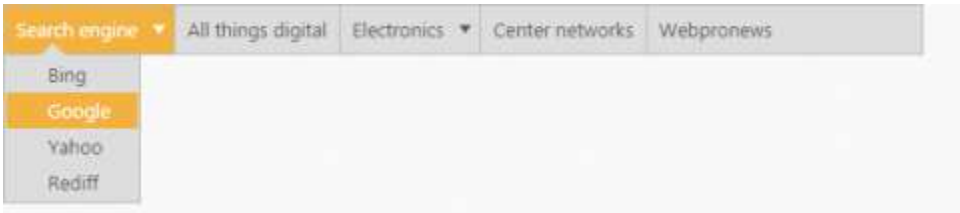
Navigation in Menu control is the default usage to navigate into the other web page. You can navigate to another page in menu item by providing link to the menu items. Navigation in Menu control can be achieved by placing “href” path to the anchor tag. Use the following code sample for navigating in Menu control.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("link").Items(items =>
{
items.Add().Url("#").Id("search").Text("Search engine").Children(child =>
{
child.Add().Url("http://www.bing.com/").Text("Bing");
child.Add().Url("https://www.google.co.in/").Text("Google");
child.Add().Url("https://in.yahoo.com/").Text("Yahoo");
child.Add().Url("http://www.rediff.com/").Text("Rediff");
});
items.Add().Url("http://allthingsd.com/").Text("All things digital");
items.Add().Url("").Text("Electronics").Children(child =>
{
child.Add().Url("http://www.engadget.com/").Text("Engadget");
child.Add().Url("http://www.electronista.com/").Text("Electronista");
child.Add().Url("http://www.gearlog.com/").Text("Gearlog");
});
items.Add().Url("http://www.centernetworks.com/").Text("Center
networks");
items.Add().Url("http://www.webpronews.com/").Text("Web pronews");
}).Width("500")
</div>
```

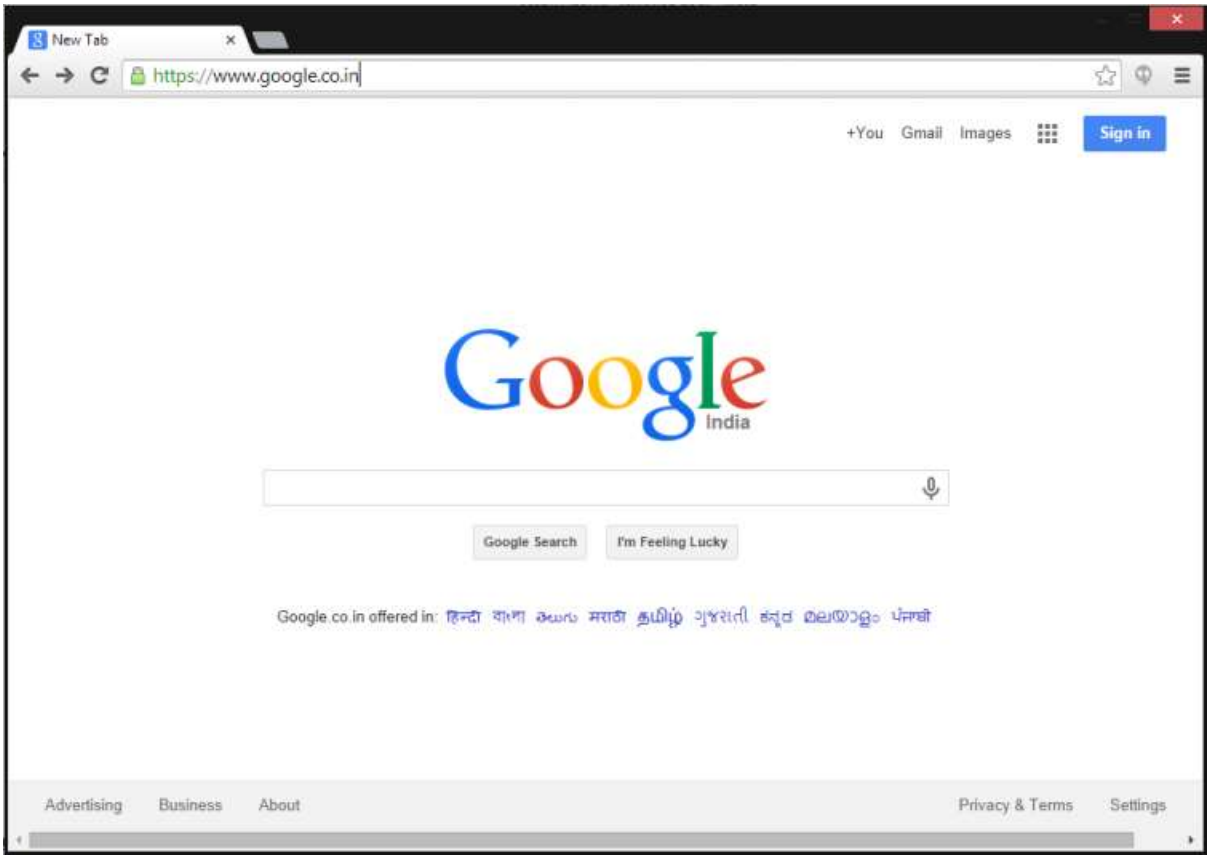
The following screenshot displays the output for the above code example.



Copyright © 2001 - 2020 Syncfusion Inc.

Navigation of Menu

When you click on “Google” that is present under “Search engine”, it navigates to the link that you specified in the sample code. Then the output is as follows.



After navigating to a menu item

Customizing the Submenu direction

You can customize the direction to open the sub menu items using SubMenuDirection property. SubMenuDirection accepts the type as string or enum and value as “Left” and “Right”.

In the following example, the Sub menus opens in the Left side of the menu.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("syncfusionProducts").Items(items =>
{
items.Add().Id("Products").Text("Products").Children(child =>
{
child.Add().Text("ASP.NET");
child.Add().Text("ASP.NET MVC");
child.Add().Text("Mobile MVC");
child.Add().Text("Silverlight");
child.Add().Text("Windows Forms");
```

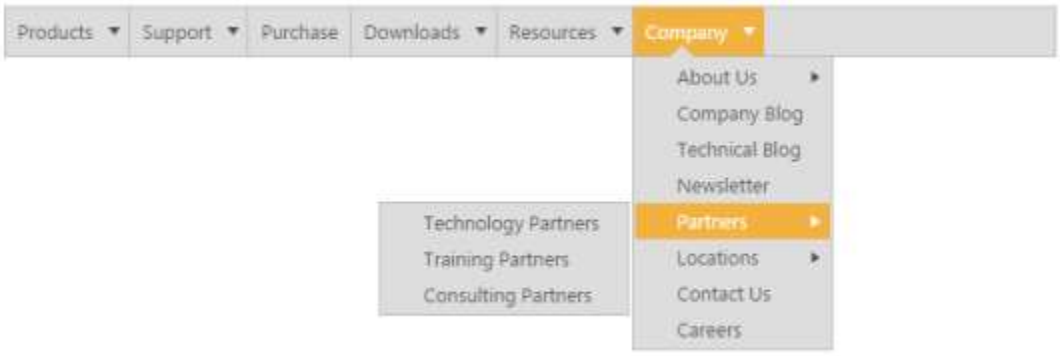
Copyright © 2001 - 2020 Syncfusion Inc.

```
child.Add().Text("Windows Phone");
child.Add().Text("WinRT (XMAL)");
child.Add().Text("WPF");
child.Add().Text("Orubase Studio");
child.Add().Text("Metro Studio");
child.Add().Text("What's New").Children(child1 =>
{
child1.Add().Text("ASP.NET");
child1.Add().Text("WPF");
child1.Add().Text("Silverlight");
child1.Add().Text("Windows Forms");
child1.Add().Text("Windows Phone");
child1.Add().Text("ASP.NET MVC");
child1.Add().Text("ASP.NET");
});
});
items.Add().Text("Support").Children(child =>
{
child.Add().Text("Direct-Trac Support");
child.Add().Text("Community Forums");
child.Add().Text("Knowledge Base");
child.Add().Text("Online Documentation");
child.Add().Text("Services").Children(child1 =>
{
child1.Add().Text("Consulting");
child1.Add().Text("Training");
});
});
items.Add().Id("Purchase").Text("Purchase");
items.Add().Id("Downloads").Text("Downloads").Children(child =>
{
child.Add().Text("Evaluation");
child.Add().Text("Free E-Books");
child.Add().Text("Metro Studio");
child.Add().Text("Latest Version");
child.Add().Text("Version History");
});
items.Add().Id("Resources").Text("Resources").Children(child =>
{
child.Add().Text("Technology Resource Portal").Children(child1 =>
{
child1.Add().Text("E-Books");
child1.Add().Text("White Papers");
});
child.Add().Text("Case Studies");
child.Add().Text("Boucher's & Data sheets");
child.Add().Text("FAQ");
});
items.Add().Id("Company").Text("Company").Children(child =>
{
child.Add().Text("About Us").Children(child1 =>
{
child1.Add().Text("More About Us");
child1.Add().Text("Management");
child1.Add().Text("News & Events");
child1.Add().Text("Customer Quotes");
child1.Add().Text("Customer Lists");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
child1.Add().Text("Case Studies");
child1.Add().Text("Awards");
child1.Add().Text("Media Kit");
});
child.Add().Text("Company Blog");
child.Add().Text("Technical Blog");
child.Add().Text("Newsletter");
child.Add().Text("Partners").Children(child1 =>
{
child1.Add().Text("Technology Partners");
child1.Add().Text("Training Partners");
child1.Add().Text("Consulting Partners");
});
child.Add().Text("Locations").Children(child1 =>
{
child1.Add().Text("RDU");
child1.Add().Text("Chennai");
});
child.Add().Text("Contact Us");
child.Add().Text("Careers");
});
}).Width("600").SubMenuDirection(Direction.Left)
</div>
```

The output for the above code example is as follows.



Customizing Submenu Direction

You can even achieve auto positioning for Context Menu. Use the following code sample for context menu in order to open the submenu items of context menu in left side.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
<div id="target" class="textarea">
HTML is written in the form of HTML elements consisting of tags enclosed
in angle
brackets (like <html> ),within the web page content. HTML tags most
commonly
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
come in pairs like and ,although some tags, known as empty elements, are
unpaired,
for example <img>. The purpose of a web browser is to read HTML documents
and compose them into visible or audible web pages. The browser does not
display
the HTML tags, but uses the tags to interpret the content of the page.
</div>
@Html.EJ().Menu("file").Items(items =>
{
items.Add().Text("Cut").Children(child =>
{
child.Add().Text("Cut Particular");
child.Add().Text("Cut Fully");
});
items.Add().Text("Copy");
items.Add().Text("Paste");
items.Add().Text("Comments");
items.Add().Text("Links");
items.Add().Text("Clear Formatting");
}).MenuType(MenuType.ContextMenu).OpenOnClick(true).ContextMenuTarget("#t
arget").SubMenuDirection(Direction.Left)
```

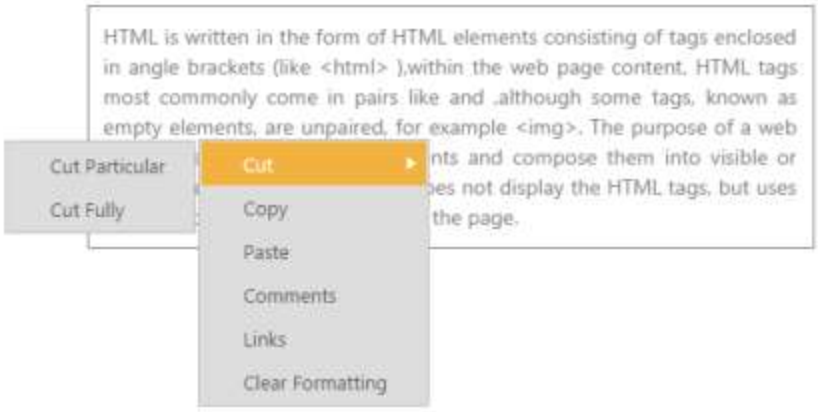
2. Add the following code in your style section.

CSS

```
<style type="text/css">
.textarea
{
border: 1px solid;
padding: 10px;
position: relative;
text-align: justify;
width: 463px;
color: gray;
margin: 0 auto;
}
</style>
```

The output for the above code example is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing Submenu direction in Context Menu

Look and feel

Essential JavaScript controls feature 12 built-in themes, six flat and gradient effects, and also supports custom skin options for user-defined themes.

12 themes support available for Menu control namely, *default-theme* flat-azure-dark *flat-lime* flat-lime-dark *flat-saffron* flat-saffron-dark *gradient-azure* gradient-azure-dark *gradient-lime* gradient-lime-dark *gradient-saffron* gradient-saffron-dark

CssClass

Menu control also customizes its appearance using user-defined CSS and custom skin options (colors and backgrounds). To apply custom themes you can use the “CssClass” property. “CssClass” property sets the root class for Menu control theme.

Using this CssClass you can override the existing styles under the theme style sheet. The theme stylesheet applies theme-specific styles like colors and backgrounds. In the following sample the value of “CssClass” property is set as “Purple-dark”. Purple-dark is added as root class to Menu control at the runtime. From this root class you can customize the Menu control theme.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").CssClass("Purple-dark")
</div>
```

2. Add the following code in your style section.

CSS

```
<style type="text/css" class="cssStyles">
.Purple-dark .e-menu,.e-menu.e-horizontal .e-list > ul {
background: pink;
}
.Purple-dark .e-menu.e-horizontal .e-list > a {
color: blue;
}
</style>
```

Following screenshot displays the output of the above code.



Look and feel of a Menu

Background Template

Menu control also provides support for template support. Normally Menu control can be created by using UL and LI tags in the preferred way. But in template supporting, you can customize the appearance of sub menu items rendering.

Initialize the Template Menu as illustrated in the following code example.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Menu("template").Items(items =>
{
items.Add().Text("Books").Children(child =>
{
child.Add().ContentTemplate(@<div class="temp temp1">
<span>BOOKS</span>
<ul>
<li><a>New Releases</a></li>
<li><a>Bestsellers</a></li>
<li><a>Upcoming</a></li>
<li><a>Box Sets</a></li>
</ul>
<ul>
<li><a>HTML Basics</a></li>
<li><a>JavaScript</a></li>
<li><a>jQuery</a></li>
<li><a>PHP Basics</a></li>
</ul>
</div>);
});
items.Add().Text("Cameras").Children(child =>
{
child.Add().ContentTemplate(@<div class="temp temp2">
<span>CAMERAS</span>
<ul>
<li><a>Point & Shoots</a></li>
<li><a>Digital SLR</a></li>
<li><a>Camcorders</a></li>
<li><a>Bestsellers</a></li>
</ul>
<ul>
<li><a>Still Camera</a></li>
<li><a>Digital Camera</a></li>
<li><a>Video Camera</a></li>
<li><a>Virtual Camera</a></li>
</ul>
</div>);
});
items.Add().Text("Movies").Children(child =>
{
child.Add().ContentTemplate(@<div class="temp temp3">
<span>MOVIES</span>
<ul>
<li><a>Genobili Actions</a></li>
<li><a>Jackie Rocks</a></li>
<li><a>Men In Blue</a></li>
<li><a>Human vs Alien</a></li>
</ul>
<ul>
<li><a>Million Dreams</a></li>
<li><a>Kung-fu</a></li>
</ul>
</div>);
});
items.Add().Text("Musics").Children(child =>
{
child.Add().ContentTemplate(@<div class="temp temp4">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<span>MUSICS</span>
<ul>
<li><a>New Releases</a></li>
<li><a>Bestsellers</a></li>
<li><a>Devotional</a></li>
<li><a>Sufi & Ghazal</a></li>
</ul>
<ul>
<li><a>Pop songs</a></li>
<li><a>Rock Music</a></li>
</ul>
</div>);
});
items.Add().Text("Gaming").Children(child =>
{
child.Add().ContentTemplate(@<div class="temp temp5">
<span>GAMING</span>
<ul>
<li><a>Upcoming</a></li>
<li><a>PC</a></li>
<li><a>PS Vista</a></li>
<li><a>PS3</a></li>
<li><a>XBox</a></li>
<li><a>Consoles</a></li>
</ul>
<ul>
<li><a>FIFA 2999</a></li>
<li><a>NBA Actions</a></li>
<li><a>Crick Champions</a></li>
<li><a>Carom legend</a></li>
</ul>
</div>);
});
})
```

2. Add the following code in your style section.

CSS

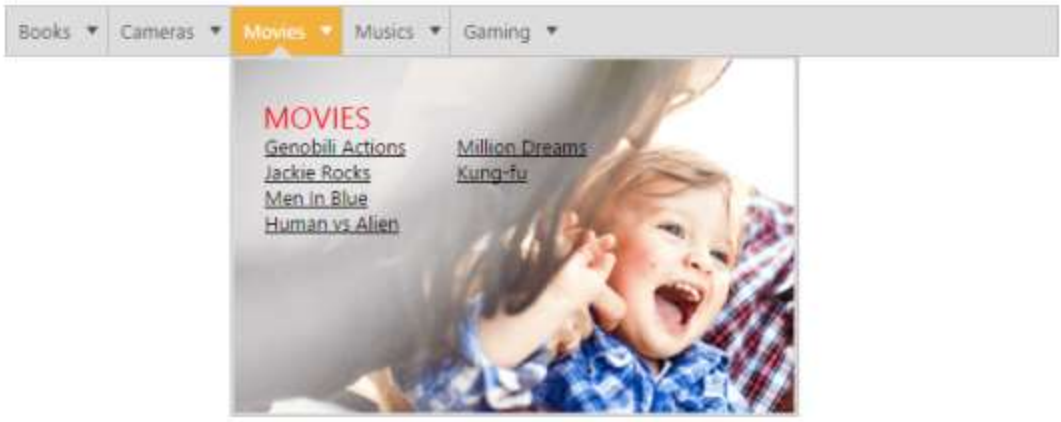
```
<style type="text/css">
.temp
{
height: 237px;
width: 375px;
font-family: segoe UI;
cursor: default;
background-size: 100% 100%;
}
.temp span
{
color: red;
float: left;
font-size: 20px;
left: 20px;
position: relative;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
top: 25px;
width: 100px;
}
.temp ul
{
float: left;
font-size: 14px;
left: -79px;
list-style-type: none;
margin: 0;
padding: 0;
position: relative;
top: 50px;
width: 128px;
}
.temp ul li
{
font-size: 13px;
}
.temp ul li a
{
text-decoration: underline;
cursor: pointer;
color: #000;
}
.temp1
{
background-image: url("1.jpg");
}
.temp2
{
background-image: url("2.jpg");
}
.temp3
{
background-image: url("3.jpg");
}
.temp4
{
background-image: url("4.jpg");
}
.e-menu.e-horizontal li > ul, .e-menu.e-horizontal li > ul > li:hover
{
background-color: #fff;
}
.e-menu.e-horizontal > li > ul:after
{
border-color: transparent transparent #fff;
}
</style>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Template

Context Menu

A context menu is a type of menu in a graphical user interface (GUI) that appears when you perform right click operation. In this Menu control you can use a context menu by specifying the type of menu as ContextMenu. A context also provides support for nested level of menu items.

Before you use the context menu, provide the target area for it.

In the following example, a context menu for the division containing text is created. In this, when you perform right click operation, the following menu appears with open, edit, etc.

1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
<div id="target" class="textarea">
HTML is written in the form of HTML elements consisting of tags enclosed
in angle
brackets (like <html> ),within the web page content. HTML tags most
commonly
come in pairs like and ,although some tags, known as empty elements, are
unpaired,
for example <img>. The purpose of a web browser is to read HTML documents
and compose them into visible or audible web pages. The browser does not
display
the HTML tags, but uses the tags to interpret the content of the page.
</div>
@Html.EJ().Menu("file").Items(items =>
{
items.Add().Text("Open").Children(child =>
{
child.Add().Text("Open with notepad");
child.Add().Text("Open with notepad++");
});
items.Add().Text("Edit");
items.Add().Text("Save");
items.Add().Text("Save as");
items.Add().Text("Print");
```

Copyright © 2001 - 2020 Syncfusion Inc.

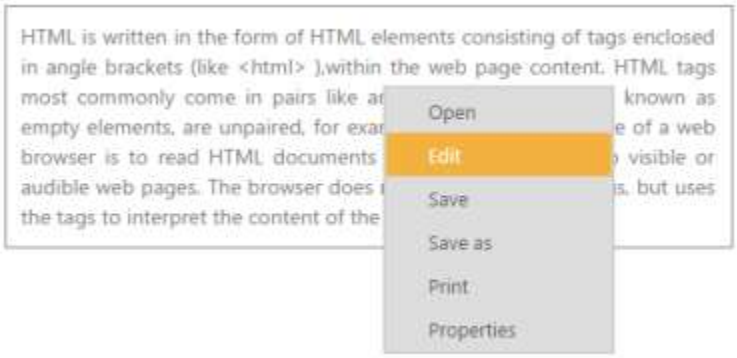
```
items.Add().Text("Properties");
}).MenuType(MenuType.ContextMenu).OpenOnClick(true).ContextMenuTarget("#t
arget")
```

2. Add the following code in your style section.

CSS

```
<style type="text/css">
.textarea
{
border: 1px solid;
padding: 10px;
position: relative;
text-align: justify;
width: 463px;
color: gray;
margin: 0 auto;
}
</style>
```

The following screen shot displays the output of the above code.



Context Menu

You can hide and show the context menu using the following methods.

HideContextMenu

Hides the context Menu control. Add the following script code in the sample in order to hide the context menu.

JS

```
<script type="text/javascript">
jQuery(function ($)
{
$("#menu").ejMenu({
width: 500,
});
//initialize the menu object
var menuObj = $("#menu").data("ejMenu");
```

```
//To enable Menu item using item id
menuObj.hideContextMenu();
});
</script>
```

ShowContextMenu

Shows the context menu control. Add the following script code in the sample in order to show the context menu.

JS

```
<script type="text/javascript">
jQuery(function ($)
{
$("#menu").ejMenu({
width: 500,
});
//initialize the menu object
var menuObj = $("#menu").data("ejMenu");
//To enable Menu item using item id
menuObj.showContextMenu();
});
</script>
```

Center Menu

You can align the Menu items to center by setting “EnableCenterAlign” property as true. “EnableCenterAlign” property accepts Boolean value. By default, its value is false. When set to true, then the root menu items is aligned in center.

1. Add the following code in your view section.

ASPX-CS

```
// Add the following code in CSHTML section.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").EnableCenterAlign(true)</div>
```

The following screenshot displays the output of the above code.



Center Menu

RTL Support

The EnableRTL option allows the Menu control to display it in the right to left direction. By default, this option is set to “false” in the Menu control.

1. The following code depicts you on how to enable the rtl property of the Menu control.

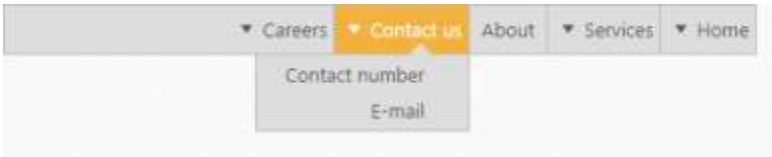
ASPX-CS

```
// The following code example depicts how to enable the rtl property of
the Menu control.
<div class="imgframe">
@Html.EJ().Menu("menuControl").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").EnableRTL(true)
</div>
```

Following screenshot displays the output for the above code.



RTL Support

Separators

Separators can be added to menu items to display a horizontal bars between menu items. Separators are similar to borders and cannot be selected.**enableSeparator** is the property that is used to display the separators in the Menu control. It accepts Boolean type value. Its default value is true.

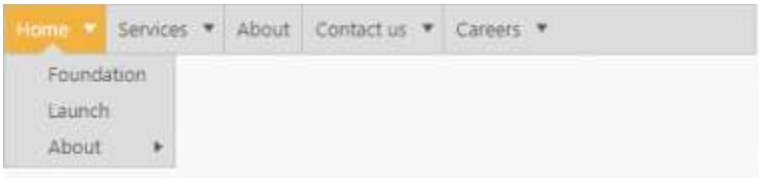
1. Add the following code in your view page.

ASPX-CS

```
// Add the following code in the CSHTML page.
<div class="imgframe">
Html.EJ().Menu("menuControl").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
```

```
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").EnableSeparator(true)
</div>
```

The following screenshot displays the output for the above code sample.



Menu with Separators

2. Add the following code in your view page to display the Menu control without separator by setting EnableSeparator as false.

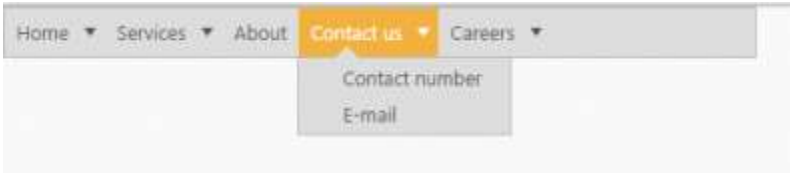
ASPX-CS

```
<div class="imgframe">
@Html.EJ().Menu("menuControl").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").EnableSeparator(false)
</div>
```

The following screenshot displays the output for the above code.



Menu without Separators

Responsive Layout

Responsive Layout is aimed at crafting sites to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors). In order to get responsive layout, you can add ej.responsive.css file in this sample. CDN link for the responsive CSS file is as follows.

http://cdn.syncfusion.com/13.1.0.21/js/web/responsive-css/ej.responsive.css

Note: Refer to the ej.responsive.css file after the ej.widgets.all.min.css file

Add the above CSS link in the code sample.

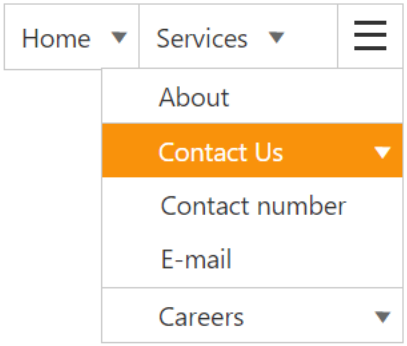
1. Add the following code in your View page.

ASPX-CS

```
// Add the following code in your CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("menuControl").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
```

```
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("200")
</div>
```

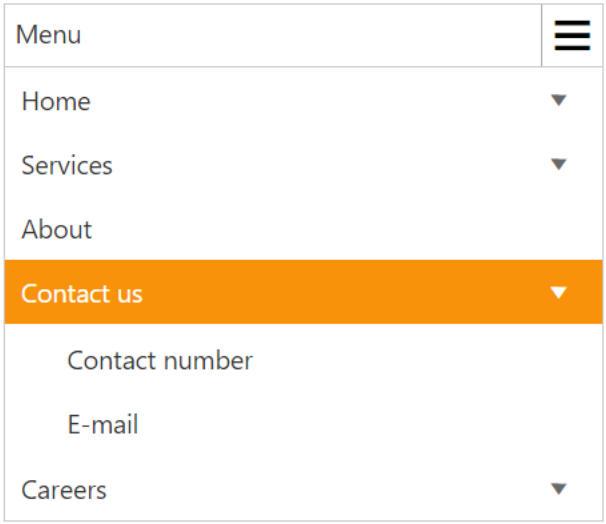
The following screenshot displays the output for the above code.



Menu with responsive layout

Responsive in Mobile or Tablet:

Menu will be displayed in mobile or Tablet as shown in the below image:



Copyright © 2001 - 2020 Syncfusion Inc.

Note: Window width below 767px is considered as Mobile or Tablet mode in our menu.

Keyboard Navigation

The Menu control also provides support for keyboard navigation. In the Menu control, it is possible to control the entire menu control items by using the provided shortcut keys.

The various keyboard shortcuts available within the Menu control are discussed in the following table,

List of keyboard shortcut keys

Keys	Usage
Esc	Closes the opened Menu control.
Enter	Selects the focused item.
Up/left/down/right arrow keys	Navigates up or previous item.
Down	Navigates down or next item.
Left	Navigates to previous group.
Right	Navigates to next group.

1. Add the following code for Keyboard navigation in your Menu control.

ASPX-CS

```
// Add the following code for Keyboard navigation in our Menu control.
<div class="imgframe">
@Html.EJ().Menu("keyboard").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

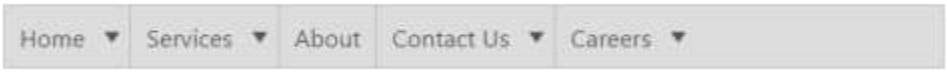
```
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500")
</div>
// Add the following code in your <script> section.
<script type="text/javascript">
jQuery(function ($)
{
//Control focus key
$(document).on("keydown", function (e)
{
if (e.altKey && e.keyCode === 74)
{
// j- key code.
$("#keyboard").focus();
}
});
});
</script>
```

2. Add the following code in your style section.

CSS

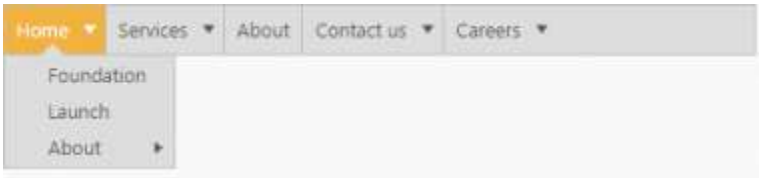
```
<style type="text/css">
#keyboard
{
margin-left: 50px;
}
</style>
```

Following screenshot displays the output of the above code.



Accessibility

When you press alt+j, the first item of the Menu control only gets focused as displayed in the following screenshot.



Keyboard Navigation

Similarly you can access the Menu control using keyboard itself.

Miscellaneous

Height

Specifies the height of the root menu. You can customize the height of the Menu control by using Height property.

1. You can specify the height of the Menu control using helper as follows.

ASPX-CS

```
// You can specify the height of the Menu control in the CSHTML page as follows.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Height("50")
</div>
```

Width

Specifies the width of the main menu. You can customize the width of the Menu control by using Width property.

Copyright © 2001 - 2020 Syncfusion Inc.

1. You can specify the width of the Menu control using helper as follows.

ASPX-CS

```
// You can specify the width of the Menu control in the CSHTML page as
follows.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("700")
</div>
```

Open on click

Specifies the sub menu items to be show or open only on click. It accepts the Boolean value. Its default value is false. If we set “OpenOnClick” property to true then the submenu items will open only on click. By default the submenu will open when we hover on menu items.

1. Add the following code in your view page to render the menu.

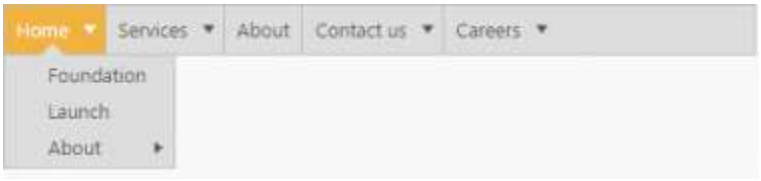
ASPX-CS

```
// Add the following code in the CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").OpenOnClick(true)
</div>
```

Output screenshot for the above code example is as follows.



Sub menu items to open on click

Animation

Animation type is used to enable or disable the Animation when hover or click on menu items. Its value type is string. It accepts two values such as “none” and “default”. Support to disable the Animation Type when hover or click on menu items is none. Support to enable the Animation Type when hover or click on menu items is default.

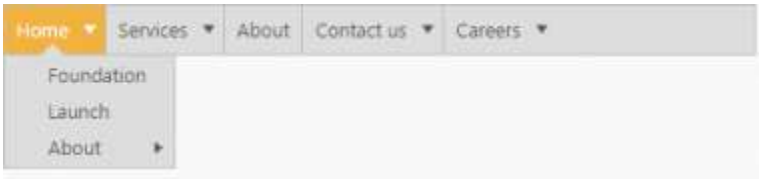
1. Add the following code in your view page to render the menu.

ASPX-CS

```
// Add the following code in the CSHTML page.
```

```
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").AnimationType(AnimationType.Default)
</div>
```

Output screenshot for the above code sample is as follows.



Animation

Title text

Specifies the title to the responsive menu. You can provide title to the Menu control by using TitleText property.

1. You can specify the title of the Menu control using helper as follows.

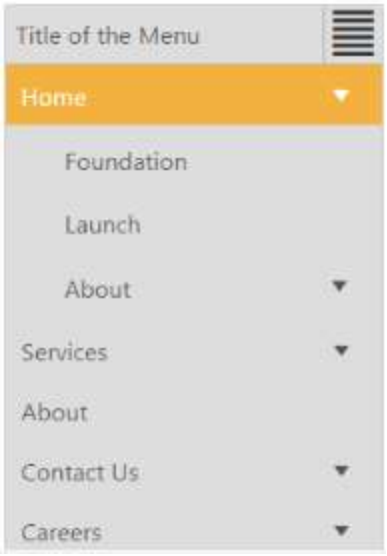
ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
// You can specify the title of the Menu control in the CSHTML page as
follows.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").TitleText("Title of the Menu")
</div>
```

The following screenshot displays the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Title text for Responsive Layout

Show root level arrows

Specifies the main menu item arrows to display only when it contains child menu items. You can use “ShowRootLevelArrows” property to display the arrows of main menu items only when it contains child menu items. This property accepts Boolean value. Its default value is true.

1. Add the following code in your view page to render the menu with root level arrows

ASPX-CS

```
// Add the following code in the CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
});
items.Add().Id("Careers").Text("Careers").Children(child =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").ShowRootLevelArrows(false)
</div>
```

The following screenshot displays the output of the above code.



Show root level arrows

Show sub level arrows

Specifies the sub menu items arrows to display only when it contains child menu items. You can use “ShowSubLevelArrows” property to show the arrows of sub menu items only when it contains child menu items. This property accepts Boolean value. Its default value is true.

1. Add the following code in your view page to render the menu with sub level arrows

ASPX-CS

```
// Add the following code in the CSHTML page.
<div class="imgframe">
@Html.EJ().Menu("menu").Items(items =>
{
items.Add().Id("Home").Text("Home").Children(child =>
{
child.Add().Text("Foundation");
child.Add().Text("Launch");
child.Add().Text("About").Children(child1 =>
{
child1.Add().Text("Company");
child1.Add().Text("Location");
});
});
items.Add().Text("Services").Children(child =>
{
child.Add().Text("Consulting");
child.Add().Text("Outsourcing");
});
items.Add().Text("About");
items.Add().Id("Contact").Text("Contact Us").Children(child =>
{
child.Add().Text("Contact number");
child.Add().Text("E-mail");
```

```
});
items.Add().Id("Careers").Text("Careers").Children(child =>
{
child.Add().Text("Position").Children(child1 =>
{
child1.Add().Text("Developer");
child1.Add().Text("Manager");
});
child.Add().Text("Apply online");
});
}).Width("500").ShowSubLevelArrows(false)
</div>
```

The following screenshot displays the output of the above code.



Show sub level arrows

How To

Render Menu from code behind

Menu can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of Menu using Menu properties from code behind.

RAZOR

```
@{
Html.EJ().Menu("Menu",
(Syncfusion.JavaScript.Models.MenuProperties)ViewData["Menu"]).Render();
}
```

C#

```
public ActionResult Index()
{
List<MenuBaseItem> obj = new List<MenuBaseItem>();
obj.Add(new MenuBaseItem() { Text="Products"});
obj.Add(new MenuBaseItem() { Text = "Support" });
obj.Add(new MenuBaseItem() { Text = "Downloads" });
obj.Add(new MenuBaseItem() { Text = "Purchase" });
obj.Add(new MenuBaseItem() { Text = "Resources" });
obj.Add(new MenuBaseItem() { Text = "Company" });
MenuProperties menu = new MenuProperties();
menu.Items = obj;
menu.Width = "150";
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
menu.Orientation = Syncfusion.JavaScript.Orientation.Vertical;
ViewData["Menu"] = menu;
return View();
}
```

NavigationDrawer

Overview

EssentialMVCNavigation Drawer is a sliding panel that displays the list of navigation options on demand. That is, by default, it is not visible but you can display it onto the left/right side of the screen by swiping or by clicking with desired target icon.

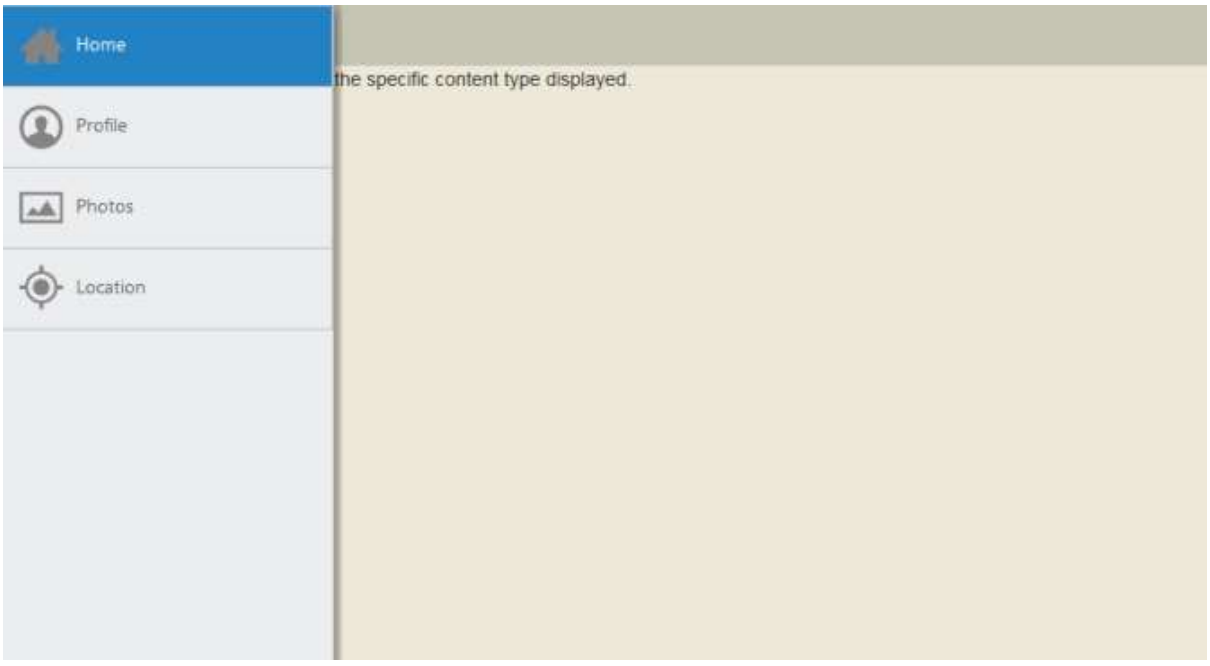
Key Features

- Direction: Specifies the rendering direction for the Navigation Drawer. That is, you can display it from right or left side of the page.
- Type: Specifies the transition type of Navigation Drawer.
- Consider SubPage: Specifies whether the Navigation Drawer to be rendered inside the closest subpage or not.
- Position: Specifies the position of the Navigation Drawer. It can be fixed or relative to the page.

Getting Started

Create your first Navigation Drawer control in MVC

In this section, you can learn how to create a simple navigation drawer.



Create Navigation Drawer Widget

The following steps guide you in adding a Navigation Drawer control for a web application that displays a list of items such as home, profile, photos and location where you can navigate to desired page by clicking on the option available in the drawer.

Copyright © 2001 - 2020 Syncfusion Inc.

You can create an MVC Project and add the necessary assemblies, styles and scripts to it. Refer to the MVC-Getting Started.

To add a Navigation Drawer control, call NavigationDrawer helper. You can display the navigational item as a list by using ListView. This is achieved by creating the ListView inside the content template. You can set the text for list items by using Text property. You can paste the following code in corresponding view page.

ASPX-CS

```
@Html.EJ().NavigationDrawer("navigationPane").Width(300).ContentTemplate(
@<div>
@Html.EJ().ListView("list").ShowHeader(false).Width(300).Items(items =>
{
items.Add().Text("Home");
items.Add().Text("Profile");
items.Add().Text("Photos");
items.Add().Text("Location");
});
</div>)
```

Create the target element as follows to display the drawer by clicking target icon.

ASPX-CS

```
<div class="navigation">
<div id="target" class="icon-target"> Drawer</div>
</div>
```

To set the target icon image from sprite and to position the target icon properly use the following code example.

CSS

```
<style>
[class*="icon-"]
{
background-image:
url("http://js.syncfusion.com/ug/web/content/drawer/sprite.png");
}
.icon-target
{
background-position: 0 -331px;
font-size: 34px;
height: 48px;
position: absolute;
text-indent: 50px;
top: -3px;
width: 48px;
z-index: 3;
}
.navigation
{
background: none repeat scroll 0 0 #C4C4B4;
color: #fff;
height: 45px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

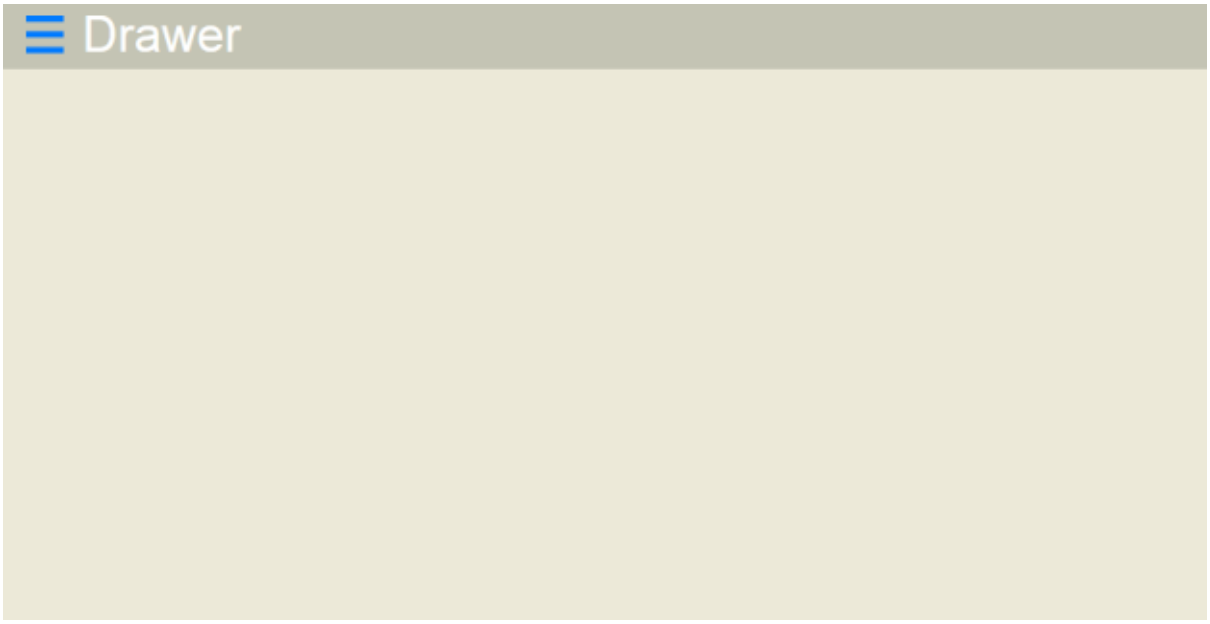
```
padding-left: 5px;
width: 100%;
}
body
{
background: none repeat scroll 0 0 #ece9d8;
}
</style>
```

Set the TargetId property as follows.

ASPX-CS

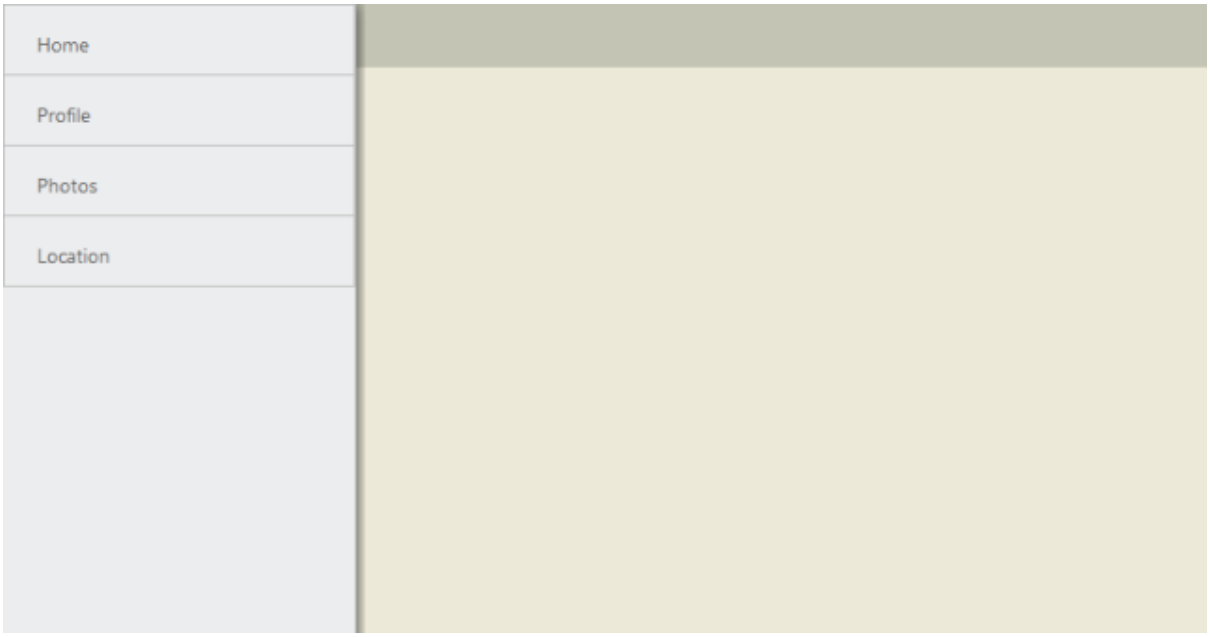
```
@Html.EJ().NavigationDrawer("navigationPane").Width(300).TargetId("target
").ContentTemplate(@<div>
@Html.EJ().ListView("list").ShowHeader(false).Width(300).Items(items =>
{
items.Add().Text("Home");
items.Add().Text("Profile");
items.Add().Text("Photos");
items.Add().Text("Location");
});
</div>)
```

Run the application to render the following output.



You can display the drawer either by clicking on the target icon or by swiping from left on the page. Refer to the following screenshot.

Copyright © 2001 - 2020 Syncfusion Inc.



You can set the images for Navigation Drawer by using the ImageClass property as follows.

ASPX-CS

```
@Html.EJ().NavigationDrawer("navigationPane").Width(300).TargetId("target
").ContentTemplate(@<div>
@Html.EJ().ListView("list").ShowHeader(false).Width(300).Items(items =>
{
items.Add().Text("Home").ImageClass("e-icon e-home");
items.Add().Text("Profile").ImageClass("e-icon e-profile");
items.Add().Text("Photos").ImageClass("e-icon e-photo");
items.Add().Text("Location").ImageClass("e-icon e-location");
});
</div>)
```

You can define the image classes specified for the list items as follows.

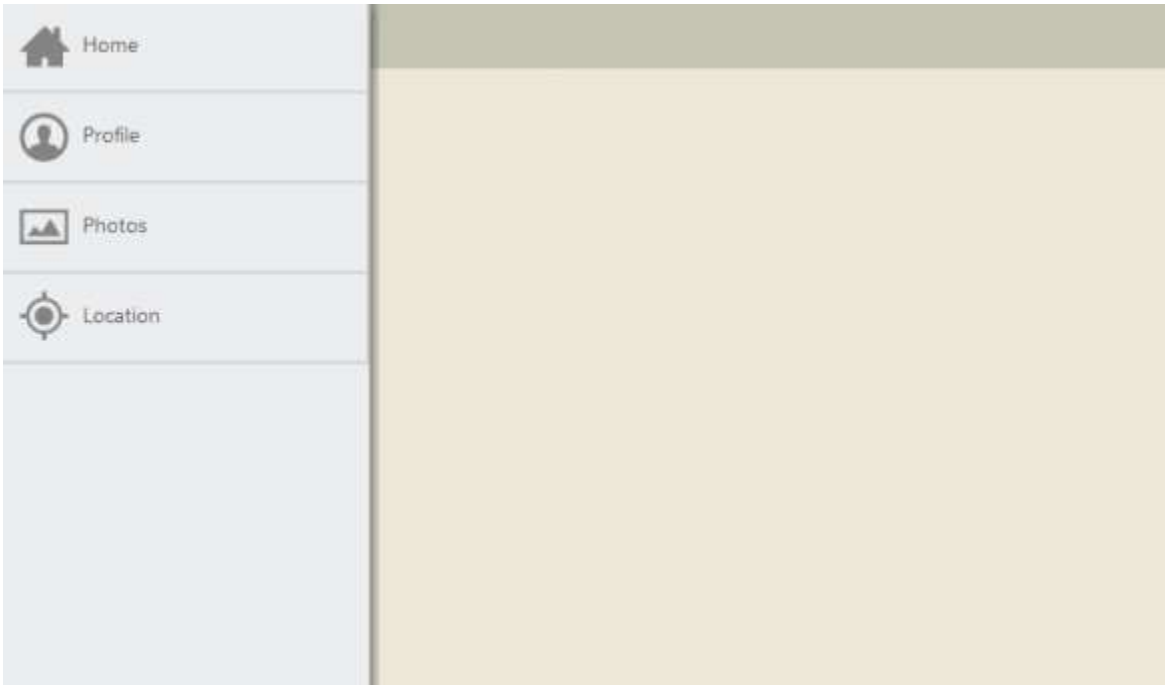
CSS

```
<style>
@font-face {
font-family: 'ej-font';
src: url('../../common-images/tools/icons.eot');
src: url('../../common-images/tools/icons.eot') format('embedded-
opentype'), url('../../common-images/tools/icons.woff')
format('woff'),url('../../common-images/tools/icons.woff')
format('woff'), url('../../common-images/tools/icons.ttf')
format('truetype'), url('../../common-images/tools/icons.svg')
format('svg');
font-weight: normal;
font-style: normal;
}
.e-home:before {
font-family: "ej-font";
content: "\e900";
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.e-profile:before {
font-family: "ej-font";
content: "\e901";
}
.e-photo:before {
font-family: "ej-font";
content: "\e903";
}
.e-location:before {
font-family: "ej-font";
content: "\e905";
}
.e-people:before {
font-family: "ej-font";
content: "\e902";
}
.e-communities:before {
font-family: "ej-font";
content: "\e904";
}
.e-home, .e-profile, .e-people, .e-photo, .e-communities, .e-location {
font-size: 24px;
color: black;
}
</style>
```

Run the application to render the following output.



Create corresponding content elements for each options in the navigation list as follows.

ASPX-CS

```
<!-- Home Page Content-->
<div id="Home">
```

```
The Home screen allows you to choose the specific content type displayed.
</div>
<!-- Profile Page Content-->
<div id="Profile" style="display: none">
The Profile page content is displayed.
</div>
<!-- Photos Page Content-->
<div id="Photos" style="display: none">
The Photos page content is displayed.
</div>
<!-- Location Page Content-->
<div id="Location" style="display: none">
The Location page content is displayed.
</div>
</div>
```

You can load the desired content for the navigation items by updating the content through MouseDown handler of ListView. You can define the handler and pass the method name with MouseDown attribute through listViewSettings. Also to know which item's content is being loaded in the page, make the list selection to persist in the drawer by setting persistSelection as true. Refer to the following code example.

ASPX-CS

```
@Html.EJ().NavigationDrawer("navigationPane").Width(300).TargetId("target
").ContentTemplate(@<div>
@Html.EJ().ListView("list").ShowHeader(false).ClientSideEvents(click =>
click.MouseDown("slideMenuClick")).PersistSelection(true).Width(300).Item
s(items =>
{
items.Add().Text("Home").ImageClass("e-icon e-home");
items.Add().Text("Profile").ImageClass("e-icon e-profile");
items.Add().Text("Photos").ImageClass("e-icon e-photo");
items.Add().Text("Location").ImageClass("e-icon e-location");
});
</div>)
```

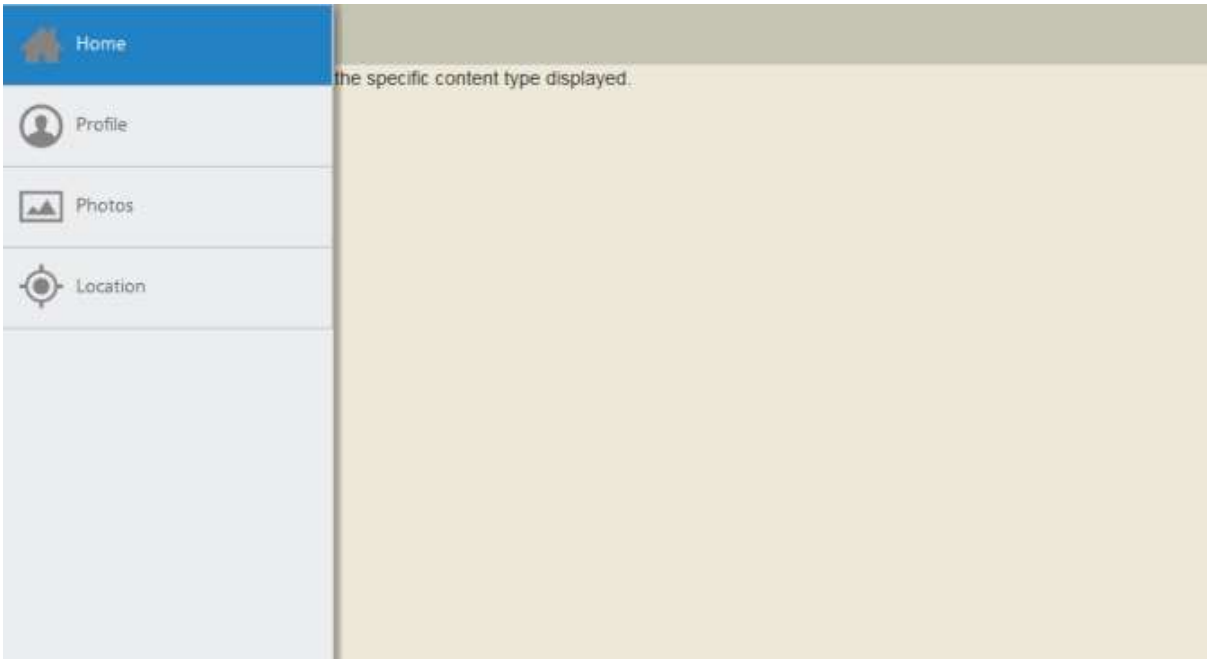
In the mouse down handler, you can hide the other content and display the respective selected item's content.

JS

```
<script type="text/javascript">
function slideMenuClick(e)
{
$('#Home, #Profile, #Photos, #Location').hide(); //Hiding all other
contents
$('#' + e.text).show(); //Displaying the content based on the text of
item selected
$("#navigationPane").ejNavigationDrawer("close");
}
</script>
```

Run the application to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Customize Direction

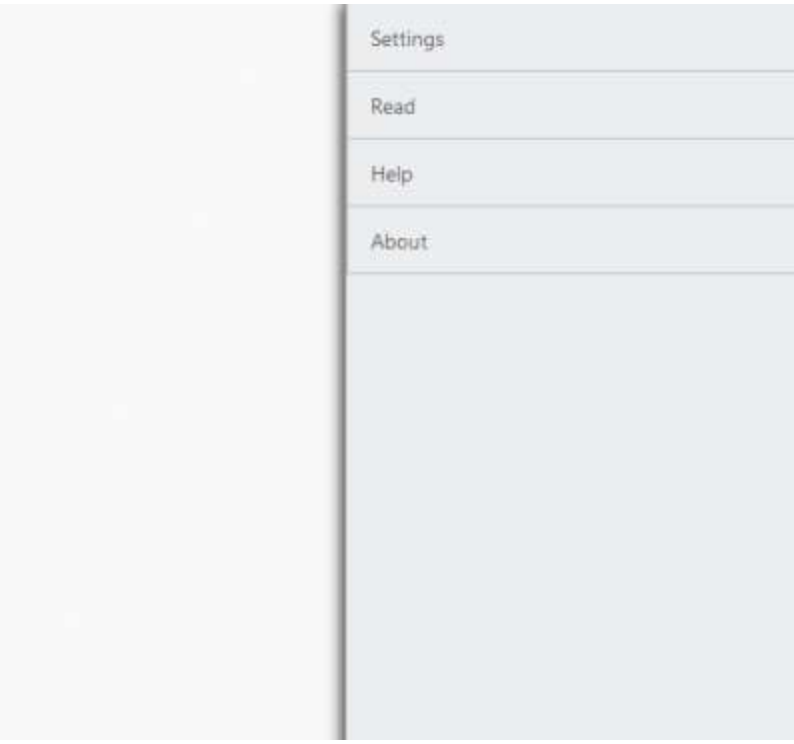
By using this property, you can set the drawer to be open from right to left direction or left to right direction. The possible direction values are Right, Left and the default value is Left. Refer to the following code example.

ASPX-CS

```
@Html.EJ().NavigationDrawer("navigationPane").Width(300).Direction(Naviga
tionDrawerDirection.Right).ContentTemplate(@<div>
@Html.EJ().ListView("list").Width(300).Items(items =>
{
items.Add().Text("Home");
items.Add().Text("Profile");
items.Add().Text("Photos");
items.Add().Text("Location");
})
</div>)
```

The following screenshot displays the output by swiping from right to left at the right side end of the screen.

Copyright © 2001 - 2020 Syncfusion Inc.



Customize Position

Position property is used to specify the position whether it is in fixed or relative to the page. When you set a normal value to position property, it appears within the container. Otherwise, it appears in the whole body .The possible position values are fixed and normal. The Default value is normal.

ASPX-CS

```
@Html.EJ().NavigationDrawer("navigationPane").Width(300).Position(Navigat
ionDrawerPosition.Fixed).ContentTemplate(@<div>
@Html.EJ().ListView("list").Width(300).Items(items =>
{
items.Add().Text("Home");
items.Add().Text("Profile");
items.Add().Text("Photos");
items.Add().Text("Location");
})
</div>)
```

The following screenshot illustrates the output by swiping from left to right at the left end of the screen.

Copyright © 2001 - 2020 Syncfusion Inc.



TargetId

This property is used to define the target Id for Navigation Drawer. The drawer opens while you click on the specified target element.

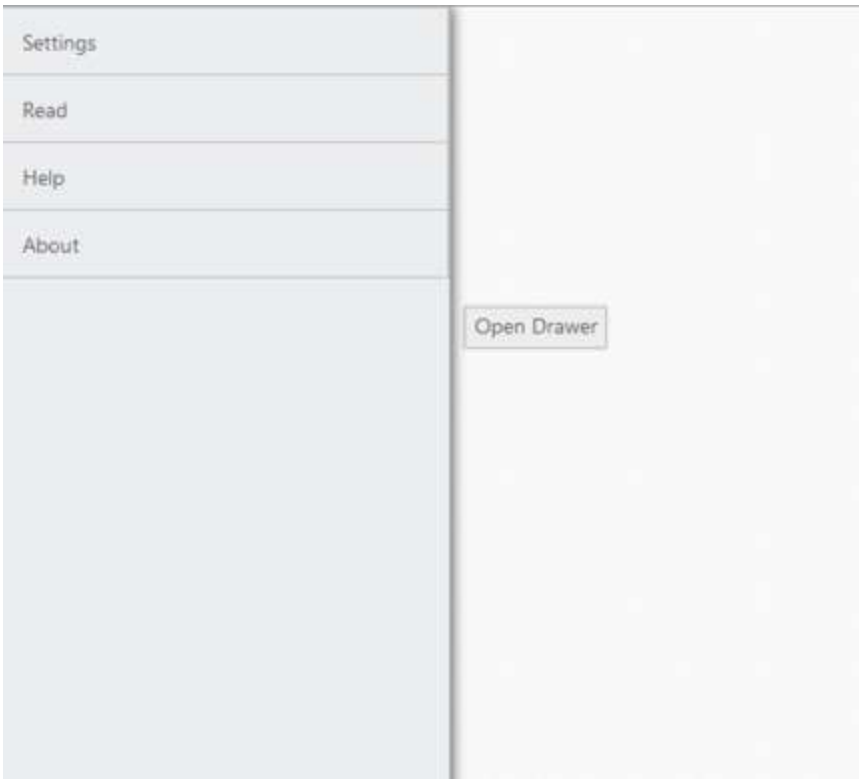
ASPX-CS

```
@{
@Html.EJ().Button("drawerTarget").Text("Open Drawer")
@Html.EJ().NavigationDrawer("navigationPane").Width(300).TargetId("drawer
Target").ContentTemplate(@<div>
@Html.EJ().ListView("list").Width(300).Items(items =>
{
items.Add().Text("Home");
items.Add().Text("Profile");
items.Add().Text("Photos");
items.Add().Text("Location");
})
</div>)
}
<style>
#drawerTarget {
top: 200px;
left: 600px;
position: absolute;
}
</style>
```

The following screenshots illustrates the output.

Copyright © 2001 - 2020 Syncfusion Inc.





Animations

You can set the transition type of the Navigation Drawer by using type property. The possible transition types are slide and overlay.

- Slide - both navigation panel and content page slides towards left/right direction to view the navigation panel items.
- Overlay - Only the navigation panel slides over the content page to view the navigation panel items. That is, part of the content page is hidden under navigation panel.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Transition slide type works only with fixed position.

The default value is Overlay.

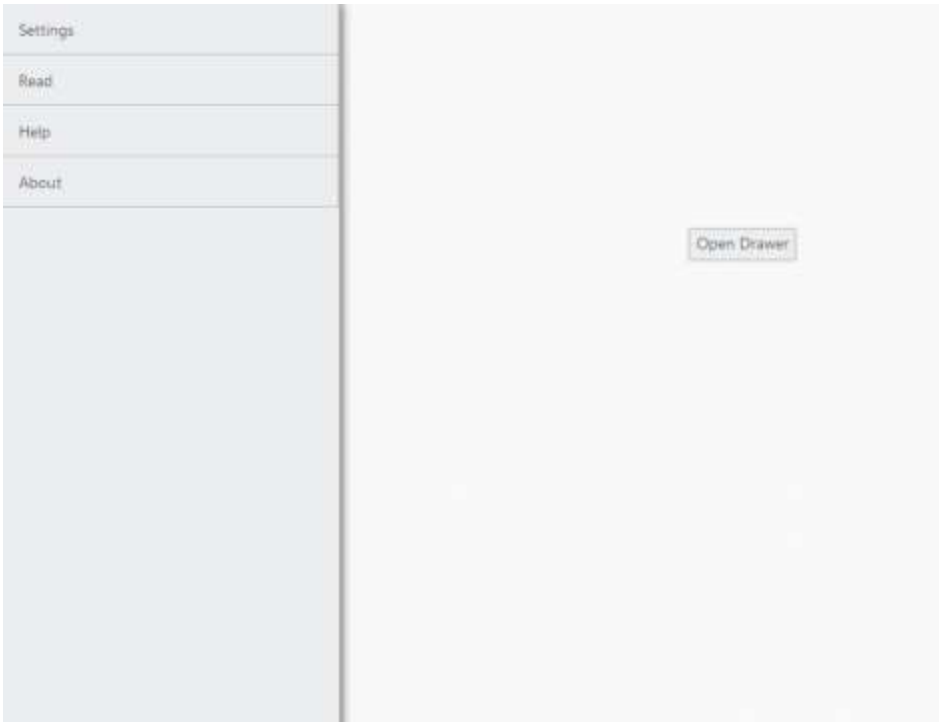
ASPX-CS

```
@{
@Html.EJ().Button("drawerTarget").Text("Open Drawer")
@Html.EJ().NavigationDrawer("navigationPane").Width(300).Type(NavigationD
rawerType.Slide).TargetId("drawerTarget").Position(NavigationDrawerPositi
on.Fixed).ContentTemplate(@<div>
@Html.EJ().ListView("list").Width(300).Items(items =>
{
items.Add().Text("Home");
items.Add().Text("Profile");
items.Add().Text("Photos");
items.Add().Text("Location");
})
</div>)
}
<style>
#drawerTarget
{
top: 200px;
left: 600px;
position: absolute;
}
</style>
<script>
$("#drawerTarget").click(function () {
$("#navigationPane").ejNavigationDrawer("open");
});
</script>
```

The following screenshot illustrates the output.

Copyright © 2001 - 2020 Syncfusion Inc.





How To

Render NavigationDrawer from Code behind

NavigationDrawer can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side

The following code illustrates the initialization of NavigationDrawer properties in the controller.

C#

```
//Namespace to get the JavaScript (NavigationDrawer) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class NavigationController: Controller
{
public ActionResult NavigationDrawer()
{
//Initializing the NavigationDrawer model
NavigationDrawerProperties NavigationObj = new
NavigationDrawerProperties();
//Initializing the NavigationDrawer content template other properties
NavigationObj.ContentTemplate = new
MvcTemplate<NavigationDrawerProperties>
{
RazorViewTemplate = (data) =>
{
return "<div id='listview'><ul><li data-ej-text='Home'></li><li data-ej-
text='Profile'></li><li data-ej-text='Photos'></li><li data-ej-
text='Location'></li></ul></div>";
}
};
NavigationObj.Width = 300;
NavigationObj.TargetId = "target";
//Passing NavigationDrawer properties using the ViewData
ViewData["NavigationModel"] = NavigationObj;
return View();
}
}
}
```

Binding the NavigationDrawer properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
<div class="navigation">
<div id="target" class="drawericon e-icon"> Drawer</div>
</div>
@{
Html.EJ().NavigationDrawer("navigationPane",
(Syncfusion.JavaScript.Models.NavigationDrawerProperties)ViewData["Naviga
tionModel"]).Render();
}
<script>
//render listview
$(function () {
$("#listview").ejListView();
});
</script>
<style>
.drawericon {
background-position: center center;
background-repeat: no-repeat;
height: 45px;
width: 32px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
background-size: 100% 100%;
padding-right: 10px;
}
.drawer icon:before {
content: "\e76b";
font-size: 24px;
height: 26px;
line-height: 24px;
}
body {
background: none repeat scroll 0 0 #ece9d8;
}
</style>
```

NumericTextBox

NumericTextBox

Essential ASP.NET MVC NumericTextBox is used to display only numeric values. It has Spin buttons to increase or decrease the values in the Text Box.

Key Features

- Min and Max Values—Specifies value range for the textbox.
- Spin Buttons—Enable you to increase or decrease the current value in the textbox.
- Step Value—Enables you to increment or decrement the current value by step value.
- Globalization— EJ MVC NumericTextBox has globalization support. It use ej.globalize.js file to globalize the number format, and parse numbers according to the culture.
- Keyboard Navigation—You can interact with NumericTextBox by using keyboard.
- RTL Support—Support for right to left alignment of textbox input.
- Decimal Values—You can configure the textbox to accept decimal values.
- Themes—NumericTextBox consist of 17 built-in themes, and also support custom skins for creating user-defined themes.

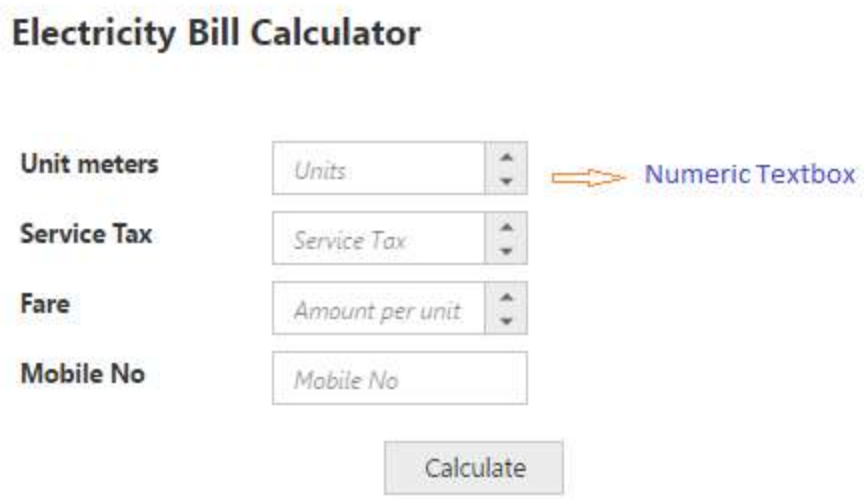
Getting Started

This section explains briefly about how to create a EJMVC NumericTextBox in ASP.NET MVC platform.

Create your first NumericTextBox in MVC

From the following steps you can learn how to create and use NumericTextBox in your application. Here we have showcased, a small Electric bill calculator application using EJMVC Editors widgets.The Essential ASP.NET MVC Editors control includes Numeric, Percentage, Currency and MaskEdit textbox controls. This will guide you to use the wide range of Editors functionalities to complete this application.

Copyright © 2001 - 2020 Syncfusion Inc.



Create Editors

ASP.NET MVC Editors renders built-in features like keyboard navigation, min and max range and flexible API's.

1. Create a MVC Project and add necessary Dll's and Scripts. Refer MVC-Getting Started.
2. Add necessary helper elements to render the Editor components.

ASPX-CS

```
<div class="editors">
<div class="element-text" style="">Electricity Bill Calculator</div>
<br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>    @* NumericTextBox creation with watermark text *@
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units")
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>     @* PercentageTextBox creation with watermark text *@
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
<td>    @* CurrencyTextBox creation with watermark text *@
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per unit")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>    @* MaskEditTextBox creation with watermark text *@
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone number")
</td>
</tr>
</tbody></table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

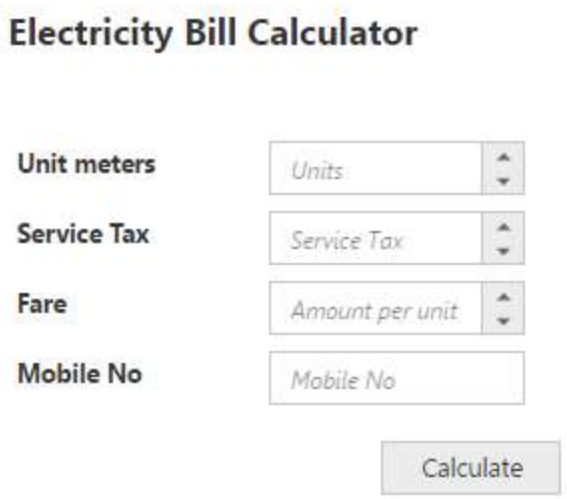
3. The following styles are added to arrange the Editors.

CSS

```
<style type="text/css" class="cssStyles">
.element-text
{
display: inline-block;
font-size: 20px;
font-weight: bolder;
height: 50px;
position: relative;
text-align: center;
top: -20px;
}
.editors
{
max-width: 400px;
border: 2px solid #DDDDDD;
}
.editors table
{
border:0px;
padding-left:50px;
}
.pay-bill
{
margin:15px 0px 10px 208px;
}
.editors label
{
display: block;
width: 130px;
}
</style>
```

4. Execute the code to render Editors as follows

Copyright © 2001 - 2020 Syncfusion Inc.



NumericTextBox with watermark text

Set MinValue, MaxValue and value in NumericTextBox

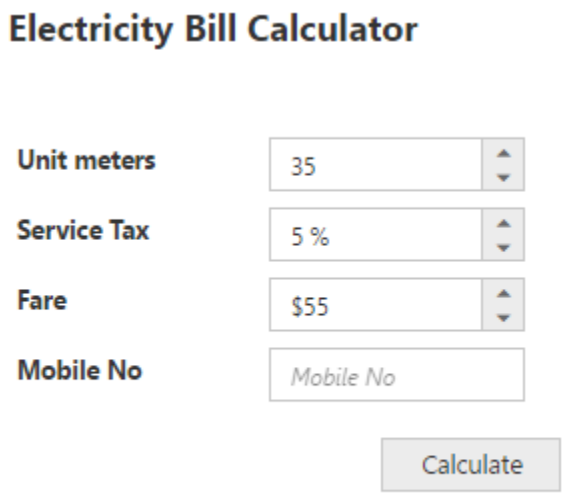
You can set the “MinValue”,“MaxValue” and “Value” in Numeric, Percentage and Currency text boxes for maintaining the range in Editors widgets. In this scenario, you have to enter the values between the default ranges and enter the phone number in the MaskEdit widget by using the ”MaskFormat” property. By using DecimalPlaces property for CurrencyTextBox you can get decimal values. The following code example illustrates how to achieve this.

ASPX-CS

```
<div class="editors">
<div class="element-text" style="">Electricity Bill Calculator</div>
<br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units").MinValue(1).M
axValue(10000).Value("70")
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax").MinValue(5).
MaxValue(100).Value("15")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
```

```
</td>
<td>
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per
unit").MinValue(0.00).MaxValue(100000.00).Value("350.00").DecimalPlaces(2
)
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone
number").MaskFormat("99-999-99999").Value("9090909090")
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

The following screenshot illustrates the output of the above code examples.



Setting the Strict Mode Option

You can set the “StrictMode” option to restrict entering values defined outside the range. The following code example illustrates how to set strict mode option.

ASPX-CS

```
<div class="editors">
<div class="element-text" style="">ElectricityBill Calculator</div><br />
<table class="editors">
<tbody>
<tr>
```

```
<td>
<label>Unit meters</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units").MinValue(1).M
axValue(10000).Value("70").EnableStrictMode(true)
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax").MinValue(5).
MaxValue(100).Value("15")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
<td>
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per
unit").MinValue(0.00).MaxValue(100000.00).Value("350.00").DecimalPlaces(2
)
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone
number").MaskFormat("99-999-99999").Value("9090909090")
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

Run the above code example and you can see that it restricts entering a value exceeding the MinValue and MaxValue range mentioned in the Numeric textbox.

Set Calculation process with NumericTextBox Widgets

You can use events to calculate the total and displays the value. You can achieve this with the help of Click event in the button widget. The calculation steps are written in the call back function of Click event button.

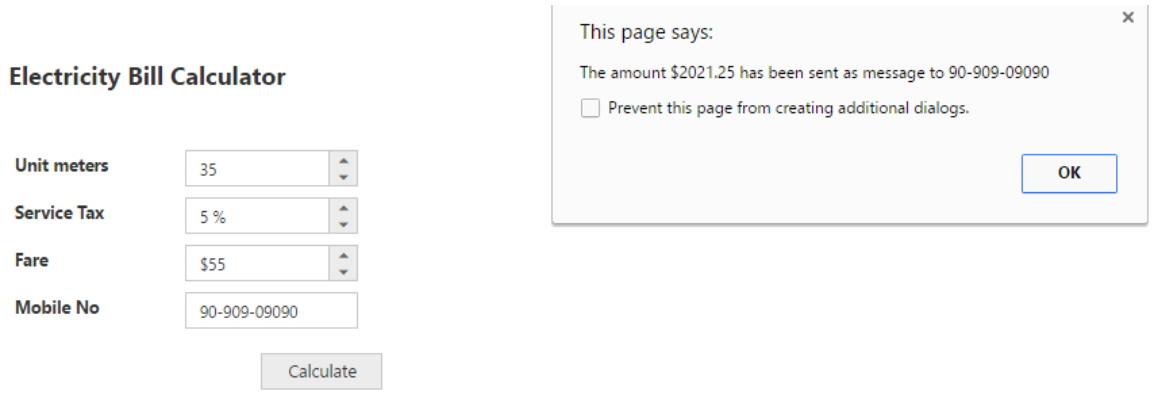
To customize the button, you can set the ContentType as TextAndImage to include the icon before the text and add the ClientSideEvents for click event.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div class="editors">
@* Refer the table format for textboxes customization *@
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate").ClientS
ideEvents(c=>c.Click("calculateBill"))
</div>
</div>
<script type="text/javascript">
function calculateBill()
{
// Declares Necessary variable creation
var unitMeter, serviceTax, amountPerKm;
unitMeter = $("#numeric").data("ejNumericTextbox");// Object of Numeric
serviceTax = $("#percent").data("ejPercentageTextbox");// Object of
Percentage
amountPerKm = $("#currency").data("ejCurrencyTextbox"); // Object of
Currency
mobileNumber = $("#maskedit").data("ejMaskEdit"); // Object of MaskEdit
// This is used to calculate the Net amount
var netAmount = unitMeter.model.value * amountPerKm.model.value;
// This is used to calculate the service tax amount
var sTax = (netAmount * serviceTax.model.value) / 100;
// This shows the calculated amount for the units
alert("The amount $" + (netAmount + sTax) + " has been sent as message to
" + mobileNumber.model.value + ".");
}
</script>
```

Run the above code sample, fill the required Textbox fields and click the Calculate button. The values are displayed and an alert message is shown. The following screen shot illustrates the final output of the Electricity bill calculator.



Behavior Settings

Decimal Places

DecimalPlaces property specifies number of values allowed after the decimal point.The default value of DecimalPlaces property is 0 i.e., by default you cannot specify decimal value in NumericTextBox. We need to add this property to allow decimal values. To set the DecimalPlaces to “-1”, that allows the decimals without any limit in NumericTextBox control.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Decimal Places

In the View page add NumericTextBox helper, and configure the DecimalPlaces property as shown below.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").DecimalPlaces(3).Value("333")
```

The output is as follows.



Persistence Support

NumericTextBox provides state maintenance support. You can maintain the previous changes made in the control after a page loads.

Configure Persistence Support

In the View page add NumericTextBox helper, and configure the EnablePersistence property as shown below.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("11").EnablePersistence(true)
```

Output of NumericTextBox with EnablePersistence is as follows.





Strict Mode Support

NumericTextBox allows you to use the strict mode option by setting the EnableStrictMode property. You can set the MinValue and MaxValue to the controls to enable strict mode functionality. Default value of this property is false. When the textbox value exceeds the MaxValue, it restricts the exceeded value and returns the MaxValue. Likewise when the textbox value goes below MinValue, it restricts the new value and returns the MinValue. When this property is enabled, it will not restrict the specified value and an error class is added to indicate wrong value is provided to the textbox.

Configure Strict Mode Support

In the View page add NumericTextBox helper, and configure the EnableStrictMode property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").MinValue(-
3).MaxValue(5).Value("10").EnableStrictMode(true)
```

Output when EnableStrictMode is “true” is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.

Enabled or Disabled

NumericTextBox has an option to enable or disable its element. You can set the Enabled property as “false” to enable the Textbox controls.

Also you can enable/disable the **NumericTextBox** by using enable and disable methods.

Configure Enabled or Disabled

In the View page add NumericTextBox helper, and configure the Enabled property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("1").Enabled(false)
```

Output when Enabled is “true” and when Enabled is “false”.





Adjusting Textbox Size

NumericTextBox size can be modified by using the Height and Width properties.

Configure Height and Width

In the View page add NumericTextBox helper, and configure the Height and Width property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Width("100").Height("50").Value("1")
```

Output of NumericTextBox after setting “Height” and “Width” is as follows.



Increment Step

The IncrementStep property is used to increase or decrease the amount of value in the Numeric textbox.

Configure Increment Step

In the View page add NumericTextBox helper, and configure the IncrementStep property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").IncrementStep(2).Value("1")
```

Output of Numeric textbox with IncrementStep is as follows.





Copyright © 2001 - 2020 Syncfusion Inc.

Define Name

When you have placed the NumericTextBox in a form, the Name property is used to send the field value at form submission. The default value of the Name property is null.

Configure Name

In the View page add NumericTextBox helper, and configure the Name property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Name("Numeric")
```

Define Value

The value of NumericTextBox can be assigned by using the Value property. The default value for Value property is null.

You can get the value of **NumericTextBox** by using getValue method.

Configure Value

In the View page add NumericTextBox helper, and configure the Value property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("12")
```

Output of NumericTextBox with the value property is as follows.



Define MaxValue and MinValue

MaxValue

The maximum limit value can be assigned to the NumericTextBox by using the MaxValue property. The default value of MaxValue property is 1.7976931348623157e+308.

MinValue

The minimum limit value can be assigned to the NumericTextBox by using the MinValue property. The default value of MinValue property is -1.7976931348623157e+308.

Configure MaxValue and MinValue

In the View page add NumericTextBox helper, and configure the MinValue and MaxValue property. .

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("2").MinValue(-1).MaxValue(2)
```

Output of NumericTextBox with MaxValue and MinValue is as follows.





Copyright © 2001 - 2020 Syncfusion Inc.

Read Only Support

NumericTextBox supports read only option. When you enable the ReadOnly property to the control, the value cannot be changed in the NumericTextBox. You can set the ReadOnly property as “true” to enable this option.

Configure Read Only

In the View page add NumericTextBox helper, and configure the ReadOnly property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("1").ReadOnly(true)
```

Output of NumericTextBox when ReadOnly is “true” is as follows. NumericTextBox values cannot be edited or changed.



Appearance

Theme

NumericTextBox control’s style and appearance can be controlled based on CSS classes. In order to apply styles you need to refer 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. If the file ej.web.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.web.all.min.css is the combination of these two.

By default, there are 17 themes support available namely:

- bootstrap
- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- high-contrast-01
- high-contrast-02
- material
- office-365

CSS Class

The CSS can be customized by using the CssClass in the NumericTextBox. You can customize the NumericTextBox with CssClass property to appear like your desired control.

Configure CSS Class

In the View page add NumericTextBox helper, and configure the CssClass property.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("1").CssClass("customCss")
```

1. Customize the CSS properties in custom CSS class.

CSS

```
<style>
.customCss .e-box {
border-color: #9d241b;
}
.customCss .e-input {
background-color: #f6db8d;
}
.customCss .e-select {
background-color: #ecf6ac;
border-color: #3c36e7;
}
</style>
```

Output of NumericTextBox after applying CssClass is as follows.



Rounded Corner Support

NumericTextBox provides you with rounded corner support whose appearance is different from normal textbox controls.

Configure Rounded Corner Support

In the View page add NumericTextBox helper, and configure the ShowRoundedCorner property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("1").ShowRoundedCorner(true)
```

Output of NumericTextBox when ShowRoundedCorner is “true”.



Spin Button Support

NumericTextBox provides you the option as to whether to display the spin button in the widget or remove it from the control by using showSpinButton property.

Configure Spin Button

In the View page add NumericTextBox helper, and configure the ShowSpinButton property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("1").ShowSpinButton(false)
```

Output of NumericTextBox when ShowSpinButton is “false”.

Copyright © 2001 - 2020 Syncfusion Inc.



Water Mark Text Support

The NumericTextBox provide water mark text support. You can display the initial value in the control by water mark.

Configure Water Mark Text

In the View page add NumericTextBox helper, and configure the ShowSpinButton property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").WatermarkText("Numeric")
```

Output of NumericTextBox after applying WaterMarkText is as follows.



Globalization Support

Globalization is language support based on the culture in **NumericTextBox** . You can achieve the **Globalization** using "**locale"** property in **NumericTextBox** .

The widget provides multi-language support using globalization. You can customize the NumericTextBox with your own language style by using this feature. You can change the globalization by using the **locale** property. The default value for **locale** property is **en-US** in NumericTextBox control.

More than 350 culture specific files are available to localize the value. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: All the culture-specific script files are available within the below specified location, once you have installed Essential Studio in your machine, therefore it is not necessary to download these files explicitly.

'(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n'
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

JAVASCRIPT

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/i18n/ej.culture.de-DE.min.js"></script>
```

You can dynamically change the language based on their culture.

Configure Localization

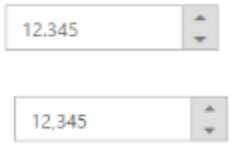
The following example describes the way to use localization in NumericTextBox.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("12345").Locale("de-DE")
```

Output of NumericTextBox with localization.



RTL Support

NumericTextBox provides RTL (Right-To-Left) support. The alignment of NumericTextBox can be changed from Left-To-Right into Right-To-Left.

Enable RTL

In the View page add NumericTextBox helper, and configure the EnableRTL property.

ASPX-CS

```
@Html.EJ().NumericTextbox("numeric").Value("11").EnableRTL(true)
```

Output of NumericTextBox when EnableRTL is “true” is as follows.



NumericTextBox with enableRTL

Keyboard Interaction

With the keyboard navigation enabled in the NumericTextBox control, it is possible to control the actions with the provided shortcut keys. Almost all the NumericTextBox functionality that are done through mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the NumericTextBox control are discussed in the following table.

Keyboard Shortcuts

Shortcut Key	Description
Access key + j	Focuses the control
Up	Increments the value
Down	Decrements the value
Tab	Focus the next element

Configuring Keyboard Navigation

The following steps explain the implementation of keyboard interaction in NumericTextBox.

In the View page add NumericTextBox helper. Set the access key property to the NumericTextBox for focusing the control while key is pressed. We need to use ‘HtmlAttributes’ property to add the ”access

Copyright © 2001 - 2020 Syncfusion Inc.

key" attribute to the Editor elements. 'HtmlAttributes' property is used to add HTML attributes like, id, class etc.. to the components. We need to use IDictionary<string,object> to specify the HTML attributes.

ASPX-CS

```
@{IDictionary<string, object> numericAttribute = new Dictionary<string,
object>();
numericAttribute.Add("accesskey", "j");
}
@Html.EJ().NumericTextbox("numeric").Name("numeric").Value("11").HtmlAttr
ibutes(numericAttribute)
```

Run the above example and press Access key + j key to focus the NumericTextBox widget. Perform provided functionality by using keyboard shortcuts.



How To

Use Normal Textboxes as Syncfusion textboxes

In an application or a web page, you may use normal textboxes along with other Syncfusion components, since there is no separate EJHelpers for textboxes.

So, you may want to make a normal textbox to look like Syncfusion textbox in order to achieve uniform look and appearance in your web page.

This can be achieved by adding **"e-textbox"** class to the HTML helpers.

By adding this class, the textbox will have standard look and appearance as other components for all the themes supported by Syncfusion.

ASPX-CS

```
@Html.TextBox("textbox", "This is a normal Textbox", new { @class = "e-
textbox"})
```

Textbox will be rendered as shown below



Normal textbox as Syncfusion textbox

Pager

Overview

The **Essential ASP.NET MVC Pager** control allows you to navigate between the pages available in a record. Pager has features like pageSizeList an option to have multiple choices of page size value of pager control and has showGoToPage text box to navigate to any required page from the current page. The Essential ASP.NET MVC Pager control also has theme support.

Key Features

- **Trendy Look:** Rich Appearance with Theme Support.
- **RTL:** Supports for Right to Left alignment.
- **Responsive Support:** Pager control will be responsive respective to the layout.
- **Localization:** Supports localization to different cultures.

Getting Started

This section briefly explains about how to create a **Pager** control in your application with ASP.NET MVC.

Adding scroller control in your MVC application

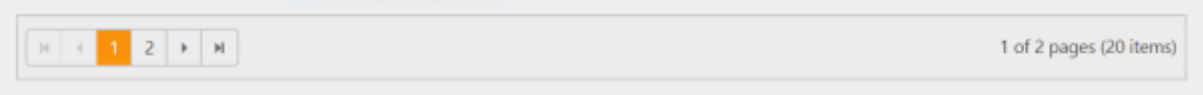
1. Create an MVC Project and add required assemblies, scripts and styles to it referring MVC-Getting-Started Documentation. 2. The Essential ASP.NET MVC Pager control based on total records count, page size and page count values. Add the following code example to the corresponding view pages to render the Pager control in your MVC application.

TotalRecordsCount: totalRecordsCount defines the total number of records which is bounded as a single data.

JAVASCRIPT

```
@Html.EJ().Pager("pager").TotalRecordsCount(20)
```

Run the above sample to get the below output.



PageSize: pageSize value defines the number of records to be displayed per page. The default value for the pageSize API is 12.

Note: The page count value changes with change in the pageSize value.

JAVASCRIPT

```
@Html.EJ().Pager("pager").TotalRecordsCount(20).PageSize(1)
```

Run the above sample to get the below output.

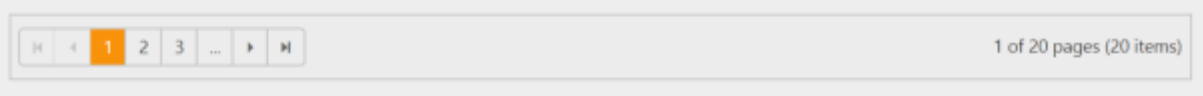


PageCount: pageCount value defines the number of pages to be displayed in the pager control for navigation. The default value for pageCount is 10 and value will be updated based on totalRecordsCount and pageSize values.

JAVASCRIPT

```
@Html.EJ().Pager("pager").TotalRecordsCount(20).PageSize(1).PageCount(3)
```

Run the above sample to get the below output.



Copyright © 2001 - 2020 Syncfusion Inc.

Behavior Settings

Page Size List

The **PageSizeList** property of the pager control allows an option to have multiple options of page size values.By defining pageSizeList, a dropdown will render in a pager with given values along with the current pageSize as selected value in dropdown.

Selected value in a dropdown will be set to pageSize API and pager will refresh based on this new pageSize.

JAVASCRIPT

```
@{List<int> pageSizeList = new List<int>();
pageSizeList.Add(1);
pageSizeList.Add(2);
pageSizeList.Add(3);
}
<div class="control">
@Html.EJ().Pager("pager").TotalRecordsCount(20).PageSize(1).PageSizeList(
pageSizeList).PageCount(3)
</div>
```



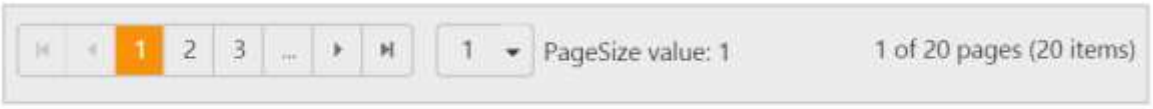
Page Size Message

The **PageSizeMessage** property allows to show a message along with the dropdown when **pageSizeList** API of pager control is defined. In the below sample, the **PageSizeMessage** of pager displays the current **pageSize** value of the pager control and will also be updated whenever the pageSize value is changed by selecting a value from the dropdown.

JAVASCRIPT

```
<div class="control">
@Html.EJ().Pager("pager").TotalRecordsCount(20).PageSize(1).PageCount(3).
PageSizeMessage("PageSize value: 1").ClientSideEvents(eve => {
eve.PageSizeSelected("pageSizeInfo");
</div>
<script>
function pageSizeInfo(e) {
var a = $("#pager").ejPager("instance");
a.option("pageSizeMessage", "PageSize value: " + e.pageSize);
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

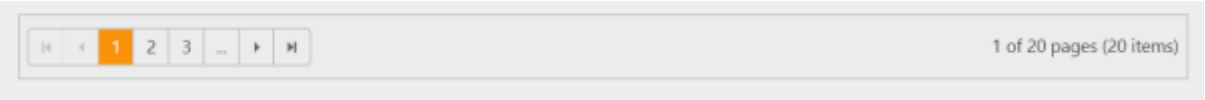


Responsive

The pager control has responsive support such that control also fit with mobile resolutions. By enabling **isResponsive** API, you can make the pager control responsive. While resizing the browser window, the inner elements in the pager control will adjust automatically to equalize the size. By default, **isResponsive** API value is false.

JAVASCRIPT

```
@Html.EJ().Pager("pager").PageCount(3).PageSize(1).CurrentPage(1).IsRespo
nsive(true).TotalRecordsCount(20)
```

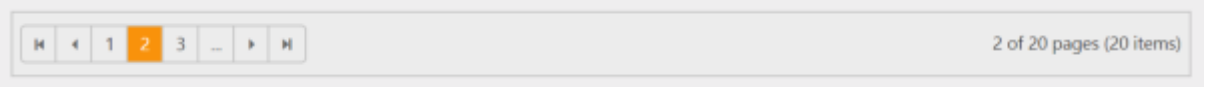


Current Page

The **CurrentPage** value of the pager determines which page to be displayed currently. The default value of the **currentPage** API is 1

JAVASCRIPT

```
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).CurrentPage(2).TotalRe
cordsCount(20)
```



Enable/Disable

You can enable or disable pager control by using **Enabled** property. Setting enabled API value as false will disable the user interaction with the pager control.

JAVASCRIPT

```
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).Enabled(false).TotalRe
cordsCount(20)
```


Customization

Show Go To Page

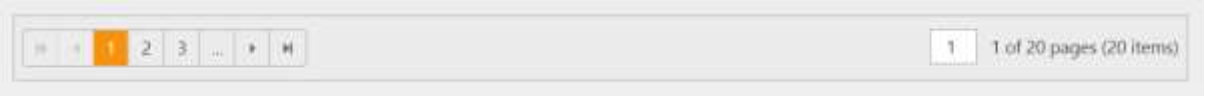
The **ShowGoToPage** property of pager renders a textbox within the pager element.

JAVASCRIPT

```
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).ShowGoToPage(true).Tot
alRecordsCount(20)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Run the above code to get the below output.



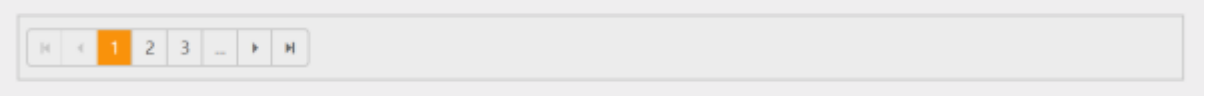
Show Page Info

The **ShowPageInfo** property of the pager control determines whether to show the page related information like total records count, total pages, current page in a pager.

JAVASCRIPT

```
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).ShowPageInfo(false).To
talRecordsCount(20)
```

Run the above code to get the below output.



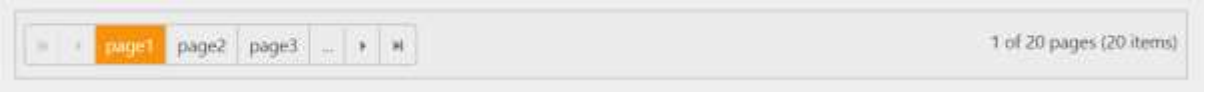
Custom Text

The **CustomText** property of pager allows to add the custom text along with numeric values in the pager numeric elements that are used for navigation in pager control.

JAVASCRIPT

```
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).CustomText("page").Tot
alRecordsCount(20)
```

Run the above code to get the below output.



External Message

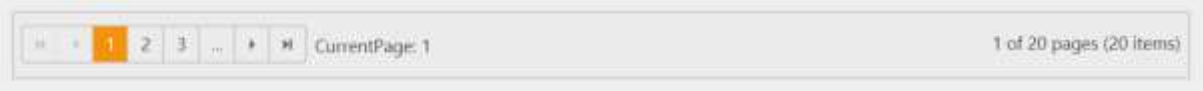
The pager control also allows to define a external message using externalMessage API to display an additional information. To use external message, we need to enable it using enableExternalMessage API. In the sample below, the externalMessage of pager is used to show the current active page of the pager control. Whenever the current page value of the pager changes, the externalMessage will be updated with the current page value.

JAVASCRIPT

```
<div class="control">
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).EnableExternalMessage(
true).ExternalMessage("CurrentPage
:1").TotalRecordsCount(20).ClientSideEvents(eve => {
eve.Change("onChange"); })
</div>
<script type="text/javascript">
function onChange(e) {
var a = $("#pager").ejPager("instance");
a.option("externalMessage", "CurrentPage: " + e.currentPage);
```

```
}
</script>
```

Run the above code to get the below output.



Custom CSS

Pager control allows you to customize the appearance using user defined CSS and custom skin options such as colors and backgrounds. To apply custom themes, we can make use of cssClass property. Using cssClass property, we can override the existing theme styles. In the following sample, value for cssClass property is set as customCss and this root class is used to customize the pager control theme.

JAVASCRIPT

```
<div class="control">
@Html.EJ().Pager("pager").PageSize(1).PageCount(3).TotalRecordsCount(20).
CssClass("customCss")
</div>
```

Defining custom class.

HTML

```
<style>
.e-pager.customCss .e-link.e-currentitem{
background:lightblue;
}
.e-pager.customCss .e-numericitem, .e-pager.customCss .e-pagermsg{
font-family:monospace;
}
.e-pager.customCss .e-pagercontainer,.e-pager.customCss .e-link ,.e-
pager.customCss .e-icon {
background-color: rgba(33,33,33,0.35);
}
.e-pager.customCss {
color:white;
background-color: rgba(33,33,33,0.30);
}
.e-pager.customCss :hover {
color:white
}
.e-pager.customCss .e-hover.e-numericitem, .e-pager.customCss .e-hover.e-
icon{
background-color:rgba(173,216,230,0.30);
}
.e-pager.customCss .e-hover.e-icon.e-disable {
background-color: rgba(33,33,33,0.35);
}
.e-pager.customCss .e-icon, .e-pager.customCss .e-numericitem{
color:white;
border-right-color: #777;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Run the above code to get the below output.



Localization

Pager has been provided with built in localization support with different culture.

Setting different culture to control

We can set different culture to **Pager** control using **Locale** API. While we change the culture of the control from default culture (English), the pager information text will be change accordingly. You can get the localized text by specifying the localized text file in your application and should be referred after ej.web.all.min.js file reference.

<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/l10n/ej.localetexts.es-ES.min.js"></script>

Also, you can set localizes text of your as shown below code snippet.

JAVASCRIPT

```
ej.Pager.Locale["es-ES"] = {
pagerInfo: "{0} de {1} páginas ({2} artículos)",
firstPageTooltip: "Ir a la primera página",
lastPageTooltip: "Ir a la última página",
nextPageTooltip: "Ir a la página siguiente",
previousPageTooltip: "Ir a la página anterior",
nextPagerTooltip: "Ir al siguiente Pager",
previousPagerTooltip: "Ir a Pager anterior"
};
ej.Pager.Locale["de-DE"] = {
pagerInfo: "{0} von {1} Seiten ({2} Beiträge)",
firstPageTooltip: "Zur ersten Seite",
lastPageTooltip: "Zur letzten Seite",
nextPageTooltip: "Zur nächsten Seite",
previousPageTooltip: "Zurück zur letzten Seite",
nextPagerTooltip: "Zum nächsten Pager",
previousPagerTooltip: "Zum vorherigen Pager"
};
```

How to?

Render Pager from Code behind

Pager can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Pager properties in the controller.

C#

```
//Namespace to get the JavaScript (Pager) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class PagerController : Controller
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
// GET: Pager
public ActionResult PagerFeatures()
{
//Initializing the Pager model
PagerProperties PagerObj = new PagerProperties();
//Initializing the datasource and other properties
PagerObj.TotalRecordsCount = 20;
PagerObj.PageSize = 1;
//Passing Pager properties using the ViewData
ViewData["PagerModel"] = PagerObj;
return View();
}
}
}
```

Binding the Pager properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Pager("searchCustomer",(Syncfusion.JavaScript.Models.PagerPrope
rties)ViewData["PagerModel"]).Render();
}
```

PivotChart

Overview

The **PivotChart** control for **Essential ASP.NET MVC** is a lightweight control that reads both **OLAP** and **Relational** datasource and visualizes it in graphical format with the ability to drill up and down.

Key Features

The key features of the **PivotChart** control is listed as follows:

- **Data source** - Supports OLAP data binding with Microsoft SQL Server Analysis Services (SSAS), XML/A data sources and Relational data sources.
- **Series** - A collection of series items that contain the actual data points is displayed on the chart.
- **Legend** - A color code that helps to differentiate between chart items. A legend also has labels beside each color to indicate that it applies to information from Series 1, Series 2, and so on.
- **Drill support** - Enables you to navigate to the inner levels in the row axis.
- **Tooltip** - Displays data point values of respective chart series on mouse hover.
- **Zooming and scrolling** – Enables you to zoom into an area of the chart so the data can be viewed with more clarity.

Getting Started

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Copyright © 2001 - 2020 Syncfusion Inc.

Creating a simple application with PivotChart and Relational datasource (Client Mode)

This section covers the information that you need to know to populate a simple PivotChart with Relational data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotChartDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in Web.config files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in Web.config files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotChart control in an MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotChart

Before initializing, empty the contents of Index.cshtml file under Views > Home folder and add the following codes.

ASPX-CS

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotChart("PivotChart1")
<style>
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```

Populate PivotChart With Data

Let us now see how to populate the PivotChart control using a sample JSON data as shown below.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").ClientSideEvents(clientSideE
vents => clientSideEvents.Load("onLoad"))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = [
{ Amount: 100, Country: "Canada", Product: "Bike" },
{ Amount: 200, Country: "Germany", Product: "Van" },
{ Amount: 300, Country: "Germany", Product: "Car" },
{ Amount: 150, Country: "United Kingdom", Product: "Bike" },
{ Amount: 200, Country: "Canada", Product: "Car" }
]
}
</script>
<style>
```

Copyright © 2001 - 2020 Syncfusion Inc.

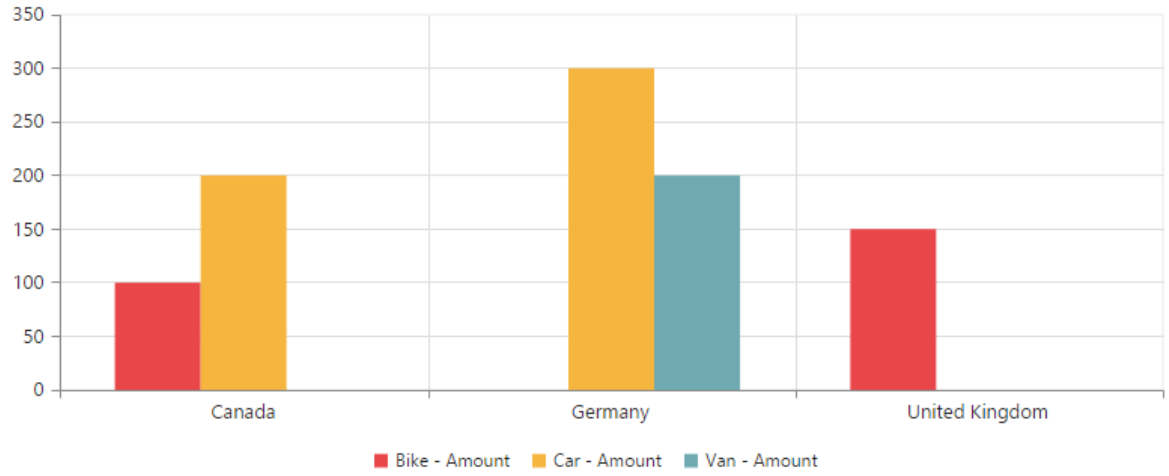
```
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```

The JSON data is set to the **“data”** property present inside the **“dataSource”** object. **“dataSource”** object allows us to set both datasource as well as the fields that needs to be displayed in the row, column, value and filter section of the PivotChart control.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").ClientSideEvents(clientSideE
vents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();}).Columns(column
s => { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Add();})).Size(size=>size.Height("460px").Widt
h("950px"))
```

The above code will generate a simple PivotChart with sales amount over products in different regions.



Apply Sorting

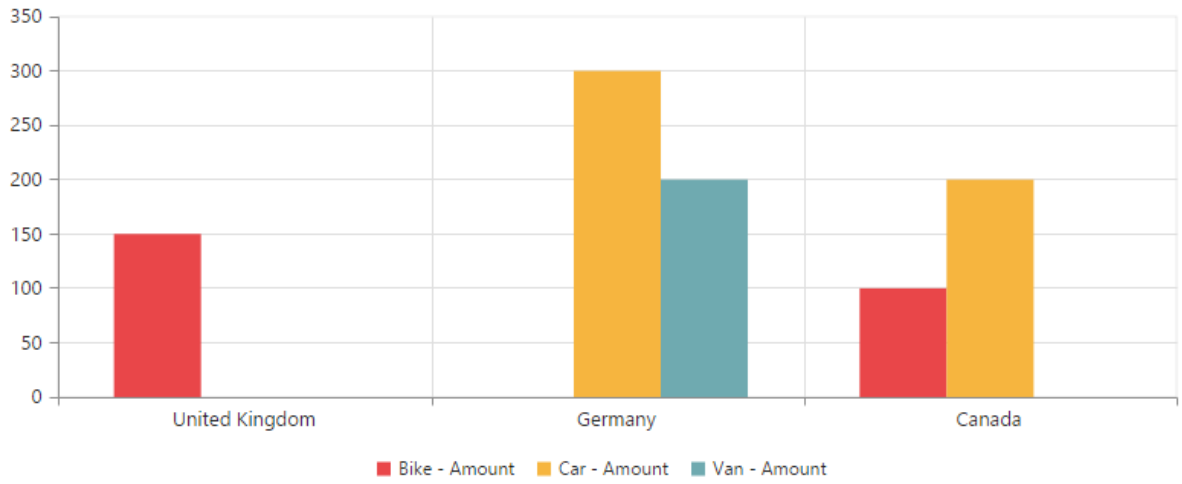
You can sort a field either to ascending or descending order using the "sortOrder" property. Sorting is applicable only for Row and Column fields.

Note: By default, fields are arranged in ascending order.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").ClientSideEvents(clientSideE
vents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource
.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").SortOrder(SortOrder.Des
cending).Add();})
```

```
.Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add(); })
.Values(values => {
values.FieldName("Amount").Add();})).Size(size=>size.Height("460px").Widt
h("950px"))
```



Apply Filtering

Filtering option allows you to specify a set of values that either need to be displayed or hidden. Also filtering option is applicable only for Row, Column and Filter areas.

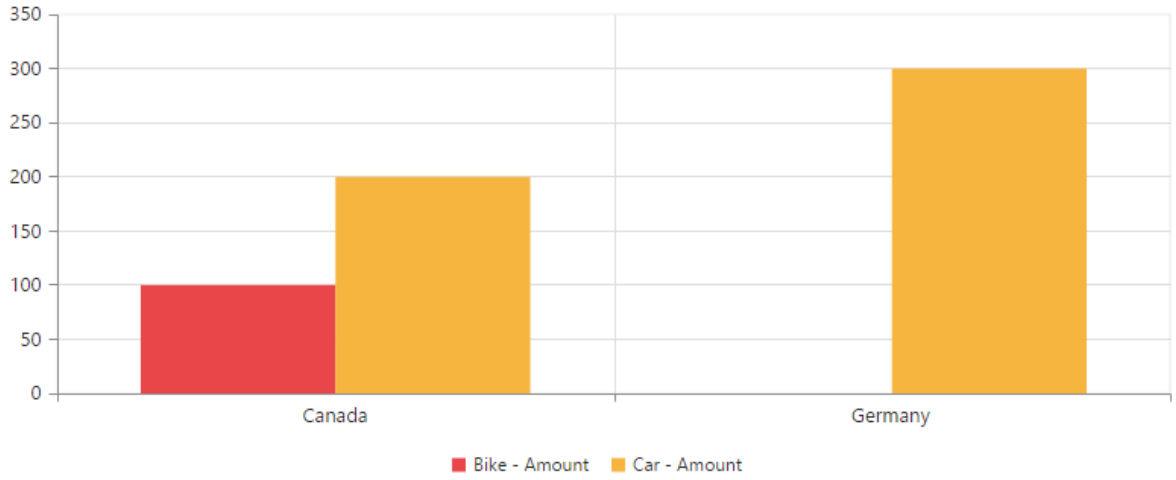
"FilterItems" object allow us to apply filtering to the fields using the following properties:

- FilterType - indicates whether the values should be included or excluded.
- Values - contains an array of values that needs to be included or excluded within the particular field.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").ClientSideEvents(clientSideE
vents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").FilterItems(filter => {
filter.FilterType(PivotFilterType.Exclude).Values(value => {
value.Add("United Kingdom"); }); }).Add(); }).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();
})).Size(size=>size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Creating a simple application with PivotChart and Relational datasource (Server Mode)

This section covers the information required to create a simple PivotChart bound to Relational datasource.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotChartDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

Copyright © 2001 - 2020 Syncfusion Inc.

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referenced assemblies in Web.config files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in Web.config files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Copyright © 2001 - 2020 Syncfusion Inc.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotChart control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
//....
//....
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

ASPX-CS

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
")).Size(size=>size.Width("950px").Height("460px"))
<style>
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```

The **"Url"** property in PivotChart control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotChart control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Relational". If WCF service is used as endpoint, the URL would look like "/RelationalService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in an existing MVC Web Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WebAPI Controller Class** and name it as **"RelationalController.cs"**, click **Add**.

Now, WebAPI controller is added to the application successfully with the file **"RelationalController.cs"**.

Note: While adding WebAPI Controller Class, name it with the suffix 'Controller' which is mandatory. For example, in this demo the controller is named as "RelationalController".

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside RelationalController.cs file.

C#

```
namespace PivotChartDemo
{
public class RelationalController: ApiController
{
}
}
```

Adding the List of Namespaces

The following are the list of namespaces to be added on top of the main class inside RelationalController.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
namespace PivotChartDemo
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public class RelationalController : ApiController
{
}
}
```

Datasource Initialization

A simple collection is provided as a datasource for the PivotChart in this demo section. This datasource is placed inside a separate class “ProductSales” in RelationalController.cs file. Refer to the following code example.

C#

```
namespace PivotChartDemo {
//....
//....
internal class ProductSales
{
public string Product
{
get;
set;
}
public string Date
{
get;
set;
}
public string Country
{
get;
set;
}
public string State
{
get;
set;
}
public int Quantity
{
get;
set;
}
public double Amount
{
get;
set;
}
public static ProductSalesCollection GetSalesData()
{
/// Geography
string[] countries = new string[] { "Australia", "Canada", "France",
"Germany", "United Kingdom", "United States" };
string[] ausStates = new string[] { "New South Wales", "Queensland",
"South Australia", "Tasmania", "Victoria" };
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string[] canadaStates = new string[] { "Alberta", "British Columbia",
"Brunswick", "Manitoba", "Ontario", "Quebec" };
string[] franceStates = new string[] { "Charente-Maritime", "Essonne",
"Garonne (Haute)", "Gers" };
string[] germanyStates = new string[] { "Bayern", "Brandenburg",
"Hamburg", "Hessen", "Nordrhein-Westfalen", "Saarland" };
string[] ukStates = new string[] { "England" };
string[] ussStates = new string[] { "New York", "North Carolina",
"Alabama", "California", "Colorado", "New Mexico", "South Carolina" };
/// Time
string[] dates = new string[] { "FY 2005", "FY 2006", "FY 2007", "FY
2008", "FY 2009" };
/// Products
string[] products = new string[] { "Bike", "Van", "Car" };
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++)
{
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product)
{
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country)
{
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString()
{
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection : List<ProductSales> { }
}
}
```

Service methods in WebAPI Controller

Define the service methods inside RelationalController class, found inside RelationalController.cs file, created while adding WebAPI Controller Class to the Application.

C#

```
namespace PivotChartDemo {
public class RelationalController: ApiController {
PivotChart pivotChart = new PivotChart();
[System.Web.Http.ActionName("InitializeChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeChart(Dictionary<string,
object> jsonResult)
{
this.BindData();
return pivotChart.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData());
}
[System.Web.Http.ActionName("DrillChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillChart(Dictionary<string, object>
jsonResult)
{
this.BindData();
return pivotChart.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["drilledSeries"].ToString());
}
private void BindData()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
this.pivotChart.PivotEngine.PivotRows.Add(new PivotItem {
FieldMappingName = "Country", FieldHeader = "Country", TotalHeader =
"Total" });
this.pivotChart.PivotEngine.PivotRows.Add(new PivotItem {
FieldMappingName = "State", FieldHeader = "State", TotalHeader = "Total"
});
this.pivotChart.PivotEngine.PivotColumns.Add(new PivotItem {
FieldMappingName = "Product", FieldHeader = "Product", TotalHeader =
"Total" });
this.pivotChart.PivotEngine.PivotCalculations.Add(new
PivotComputationInfo { CalculationName = "Amount", Description =
"Amount", FieldHeader = "Amount", FieldName = "Amount", Format = "C",
SummaryType = Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum
});
}
}
.....
..... // Datasource initialization
.....
}
```

Configure routing in WebAPIConfig Class

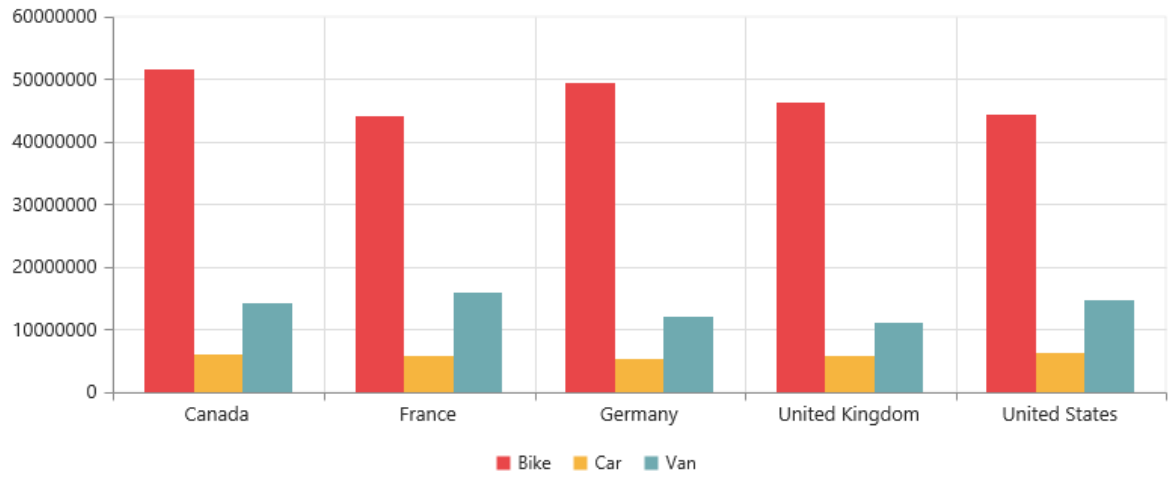
Open the WebAPIConfig.cs file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, **PivotChart** will be rendered with amount over a set of products across different customer geographic locations.

Copyright © 2001 - 2020 Syncfusion Inc.



WCF

This section demonstrates the utilization of WCF service as endpoint binding Relational datasource to a simple PivotChart. For more details on this topic, click here.

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotChart and OLAP datasource (Client Mode)

This section covers the information required to populate a simple PivotChart with OLAP data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotChartDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

Copyright © 2001 - 2020 Syncfusion Inc.

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in Web.config files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in Web.config files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

Copyright © 2001 - 2020 Syncfusion Inc.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotChart control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotChart

Before initializing, empty the contents of Index.cshtml file under Views > Home folder and add the following codes.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotChart("PivotChart1")
<style>
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```

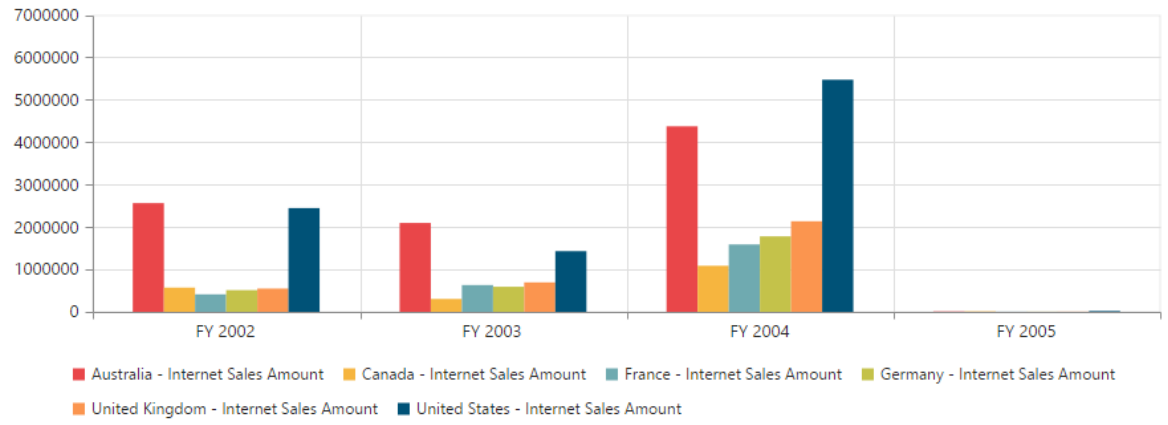
Populate PivotChart With DataSource

Initializes the OLAP datasource for PivotChart control as shown below.

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").DataSource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();}).Data("https://bi.syncfusion.com/olap/ms
mdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works")).Size(size=>size.Height("460px").Width("950px"))
```

The above code will generate a simple PivotChart with "Fiscal" field in Row, "Customer Geography" field in Column and "Internet Sales Amount" field in Value section.



Creating a simple application with PivotChart and OLAP datasource (Server Mode)

This section covers the information required to create a simple PivotChart bound to OLAP datasource from server-side.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotChartDemo"**.

Copyright © 2001 - 2020 Syncfusion Inc.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in Web.config files available inside Views folder and also at the root of the application.

XML

Copyright © 2001 - 2020 Syncfusion Inc.

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in Web.config files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotChart control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Copyright © 2001 - 2020 Syncfusion Inc.

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the **head** tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

HTML

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Olap")).Si
ze(size=>size.Height("460px").Width("950px"))
<style>
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```

The **“Url”** property in PivotChart control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotChart control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Olap". If WCF service is used as endpoint, the URL would look like "/OlapService.svc".

WebAPI

Adding a WebAPI Controller

Copyright © 2001 - 2020 Syncfusion Inc.

To add a WebAPI controller in your existing Web Application, right-click on the project in Solution Explorer and select **Add > New Item.** In the **Add New Item** window, select **WebAPI Controller Class** and name it as “OlapController.cs”, click **Add.**

Now WebAPI controller is added into your application successfully which in-turn comprise of the following file. The utilization of this file will be explained in the following sections.

- OlapController.cs

Note: While adding WebAPI Controller Class, name it with the suffix “Controller” that is mandatory. For example, in demo the controller is named as “OlapController”.

Next, remove all the existing methods such as “Get”, “Post”, “Put” and “Delete” present inside OlapController.cs file.

C#

```
namespace PivotChartDemo
{
public class OlapController : ApiController
{
}
}
```

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapController.cs file.

C#

```
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript;
namespace PivotChartDemo
{
public class OlapController : ApiController
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube and PivotChart instances are created immediately inside the main class in OlapController.cs file.

C#

```
namespace PivotChartDemo
{
public class OlapController : ApiController
{
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
PivotChart pivotChart = new PivotChart();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Other codes
}
}
```

Service methods in WebAPI Controller

Now you need to define the service methods inside OlapController class, found inside OlapController.cs file, created while adding WebAPI Controller Class to your Web Application.

C#

```
namespace PivotChartDemo
{
public class OlapController : ApiController
{
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
PivotChart pivotChart = new PivotChart();
[System.Web.Http.ActionName("InitializeChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeChart(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return pivotChart.GetJsonData(jsonResult["action"].ToString(),
DataManager);
}
[System.Web.Http.ActionName("DrillChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillChart(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Syncfusion.JavaScript.Olap.Utils.Deserialize
OlapReport(jsonResult["olapReport"].ToString()));
return pivotChart.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["drilledSeries"].ToString());
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.Name = "Default Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");
MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name = "Customer
Count" });
DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
```

```
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
return olapReport;
}
}
}
```

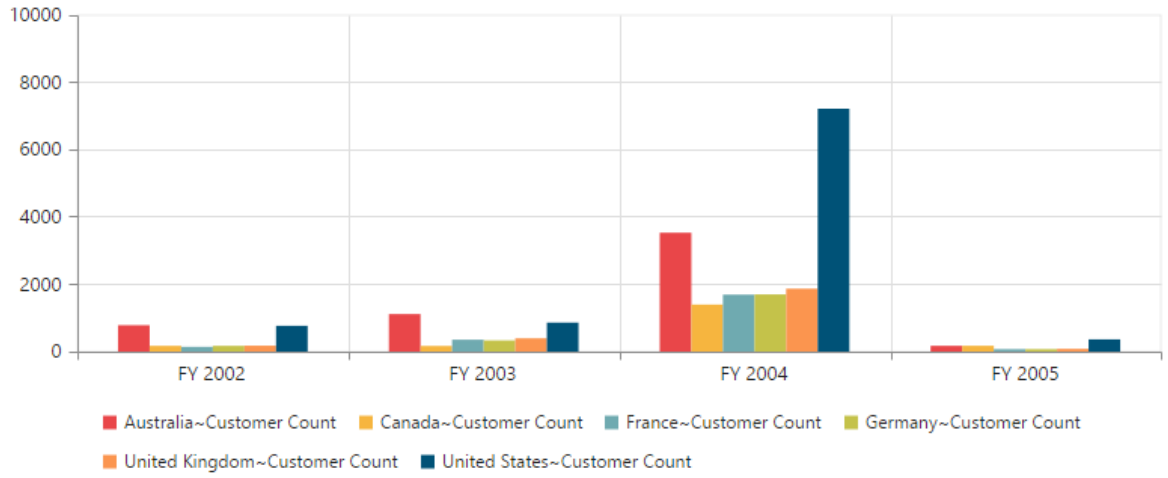
Configure routing in WebAPIConfig Class

Open the WebAPIConfig.cs file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, **PivotChart** is rendered with customer count over a period of fiscal years across different customer geographic locations.



WCF

This section demonstrates the utilization of WCF service as endpoint binding OLAP datasource to a simple PivotChart. For more details on this topic, click here.

Data Binding

Data Binding

Binding PivotChart to Collection

This section demonstrates binding of a collection to the PivotChart control as datasource. For more information on this datasource refer to the following links.

When you are using WebAPI controller, refer to the “Datasource Initialization” section under the following link.

If you are using WCF Service, refer the "Datasource Initialization" section below.

WCF

Adding a WCF Service

To add a WCF service in an existing MVC Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as RelationalService.svc, click **Add.**

Now, WCF service is added into the application successfully that comprises of the following files. The utilization of these files is explained in the immediate sections

- RelationalService.svc
- RelationalService.svc.cs
- IRelationalService.cs

Configuring WCF Service Class

Remove the “DoWork” method present inside both RelationalService.svc.cs and IRelationalService.cs files. Next, add “AspNetCompatibilityRequirements” attribute on top of main class present inside RelationalService.svc.cs and set **“RequirementsMode”** value to **“Allowed”**.

C#

```
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside RelationalService.svc.cs file.

C#

```
using System.ServiceModel.Activation;
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

Datasource Initialization

A simple collection is provided as a datasource for our PivotChart in this demo section. This datasource is placed inside a separate class named “ProductSales” in RelationalService.svc.cs file. Refer to the following code example.

C#

```
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
//....
//....
}
internal class ProductSales
{
public string Product { get; set; }
public string Date { get; set; }
public string Country { get; set; }
public string State { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public int Quantity { get; set; }
public double Amount { get; set; }
public static ProductSalesCollection GetSalesData()
{
/// Geography
string[] countries = new string[] { "Australia", "Canada", "France",
"Germany", "United Kingdom", "United States" };
string[] ausStates = new string[] { "New South Wales", "Queensland",
"South Australia", "Tasmania", "Victoria" };
string[] canadaStates = new string[] { "Alberta", "British Columbia",
"Brunswick", "Manitoba", "Ontario", "Quebec" };
string[] franceStates = new string[] { "Charente-Maritime", "Essonne",
"Garonne (Haute)", "Gers", };
string[] germanyStates = new string[] { "Bayern", "Brandenburg",
"Hamburg", "Hessen", "Nordrhein-Westfalen", "Saarland" };
string[] ukStates = new string[] { "England" };
string[] ussStates = new string[] { "New York", "North Carolina",
"Alabama", "California", "Colorado", "New Mexico", "South Carolina" };
/// Time
string[] dates = new string[] { "FY 2005", "FY 2006", "FY 2007", "FY
2008", "FY 2009" };
/// Products
string[] products = new string[] { "Bike", "Van", "Car" };
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++)
{
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product)
{
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country)
{
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
```

```
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString()
{
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection : List<ProductSales>
{
}
}
}
```

Service methods in WCF Service

First, you need to define the service methods inside IRelationalService interface, found in IRelationalService.cs file, created while adding WCF service to your Web Application.

C#

```
namespace PivotChartDemo
{
[ServiceContract]
public interface IRelationalService
{
[OperationContract]
Dictionary<string, object> InitializeChart(string action, string
currentReport, string customObject);
[OperationContract]
Dictionary<string, object> DrillChart(string action, string
drilledSeries);
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Secondly, you need to elaborate the service methods inside the main class, found in RelationalService.svc.cs file.

C#

```
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
PivotChart pivotChart = new PivotChart();
public Dictionary<string, object> InitializeChart(string action, string
currentReport, string customObject)
{
this.BindData();
return pivotChart.GetJsonData(action, ProductSales.GetSalesData());
}
public Dictionary<string, object> DrillChart(string action, string
drilledSeries)
{
this.BindData();
return pivotChart.GetJsonData(action, ProductSales.GetSalesData(),
drilledSeries);
}
private void BindData()
{
this.pivotChart.PivotEngine.PivotRows.Add(new PivotItem {
FieldMappingName = "Country", FieldHeader = "Country", TotalHeader =
"Total" });
this.pivotChart.PivotEngine.PivotRows.Add(new PivotItem {
FieldMappingName = "State", FieldHeader = "State", TotalHeader = "Total"
});
this.pivotChart.PivotEngine.PivotRows.Add(new PivotItem {
FieldMappingName = "Date", FieldHeader = "Date", TotalHeader = "Total"
});
this.pivotChart.PivotEngine.PivotColumns.Add(new PivotItem {
FieldMappingName = "Product", FieldHeader = "Product", TotalHeader =
"Total" });
this.pivotChart.PivotEngine.PivotCalculations.Add(new
PivotComputationInfo { CalculationName = "Amount", Description =
"Amount", FieldHeader = "Amount", FieldName = "Amount", Format = "C",
SummaryType = Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum
});
}
}
.....
..... // Datasource initialization
.....
}
```

Configuring Web Configuration File

Copyright © 2001 - 2020 Syncfusion Inc.

You can expose services through the properties such as binding, contract and address by using an endpoint.

1. Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IRelationalService contract and hence it is PivotChartDemo.IRelationalService.
2. Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
3. behaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

XML

```
<system.serviceModel>
……
……
<services>
<service name="PivotChartDemo.RelationalService">
<endpoint address=""
behaviorConfiguration="PivotChartDemo.RelationalServiceAspNetAjaxBehavior
"
binding="webHttpBinding" contract="PivotChartDemo.IRelationalService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotChartDemo.RelationalServiceAspNetAjaxBehavior">
<enableWebScript />
</behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Note: In this example, **“PivotChartDemo”** indicates the name and root namespace of the Web Application created in Visual Studio IDE and **“RelationalService”** indicates the name of the WCF service created.

Configuring routing file

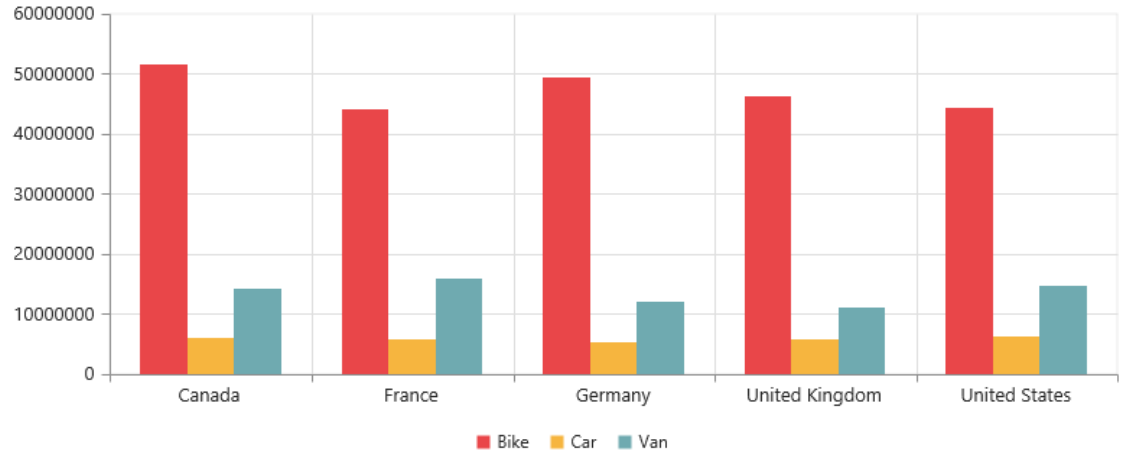
Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
}
```

Now, **PivotChart** will be rendered as shown in the below figure.



DataBinding

Binding PivotChart to Offline Cube

To connect an OLAP Cube available in local machine, set the physical path of the Cube in the connection string. The following code example illustrates the same.

C#

```
string connectionString = @"DataSource = system
drive:\OfflineCube\Adventure_Works_Ext.cub; Provider = MSOLAP;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotChart to Cube in local SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in local machine, set the server name and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = "Data source=localhost; Initial
Catalog=Adventure Works DW;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Copyright © 2001 - 2020 Syncfusion Inc.

Binding PivotChart to Cube in online SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in online server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotChart to Cube in online Mondrian Server

To connect an OLAP Cube available in Mondrian Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = @"Data Source =
https://localhost:8080/mondrian/xmla; Initial Catalog =FoodMart;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName =
Syncfusion.Olap.DataProvider.Providers.Mondrian;
```

Binding PivotChart to Cube in online ActivePivot Server

To connect an OLAP Cube available in ActivePivot Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = @"Data Source =
https://localhost:8080/cva_s/xmla; Initial Catalog = CVAS;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName=Syncfusion.Olap.DataProvider.Provid
ers.ActivePivot;
```

WCF

Adding a WCF Service

To add a WCF service in an existing web application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as **“OlapService.svc”**, click **Add**.

Now WCF service is added into your application successfully which in-turn comprise of the following files. The utilization of these files will be explained in the immediate sections.

- OlapService.svc
- OlapService.svc.cs
- IOlapService.cs

Copyright © 2001 - 2020 Syncfusion Inc.

Configuring WCF Service Class

Remove the “DoWork” method present inside both OlapService.svc.cs and IOlapService.cs files. Next, add “AspNetCompatibilityRequirements” attribute on top of main class present inside OlapService.svc.cs and set “RequirementsMode” value to “Allowed”.

C#

```
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapService.svc.cs file.

C#

```
using System.ServiceModel.Activation;
using Syncfusion.Olap.Manager;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript;
using System.Collections.Generic;
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
}
}
```

Datasource Initialization

Now the connection string to connect OLAP Cube and PivotChart instances are created immediately inside the main class in OlapService.svc.cs file.

C#

```
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
PivotChart pivotChart = new PivotChart();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
//Other codes
}
}
```

Service methods in WCF Service

First, you need to define the service methods inside IOlapService interface, found in IOlapService.cs file, created while adding WCF Service to your Web Application.

C#

```
namespace PivotChartDemo
{
[ServiceContract]
public interface IOlapService
{
[OperationContract]
Dictionary<string, object> InitializeChart(string action, string
customObject);
[OperationContract]
Dictionary<string, object> DrillChart(string action, string
drilledSeries, string olapReport, string customObject);
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Secondly, you need to elaborate the service methods inside the main class, found in OlapService.svc.cs file

C#

```
namespace PivotChartDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
PivotChart pivotChart = new PivotChart();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
public Dictionary<string, object> InitializeChart(string action, string
customObject)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return pivotChart.GetJsonData(action, DataManager);
}
public Dictionary<string, object> DrillChart(string action, string
drilledSeries, string olapReport, string customObject)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Syncfusion.JavaScript.Olap.Utils.Deserialize
OlapReport(olapReport));
return pivotChart.GetJsonData(action, DataManager, drilledSeries);
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.Name = "Default Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");
MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name = "Customer
Count" });
DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
return olapReport;
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Configuring Web Configuration File

You can expose services through the properties such as binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to **IOlapService** contract and hence it is **PivotChartDemo.IOlapService**.
- Binding: In your application, you use **webHttpBinding** to post and receive the requests and responses between the client-end and the service.
- behaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

XML

```
<system.serviceModel>
……
……
<services>
<service name="PivotChartDemo.OlapService">
<endpoint address=""
behaviorConfiguration="PivotChartDemo.OlapServiceAspNetAjaxBehavior"
binding="webHttpBinding" contract="PivotChartDemo.IOlapService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotChartDemo.OlapServiceAspNetAjaxBehavior">
<enableWebScript />
</behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

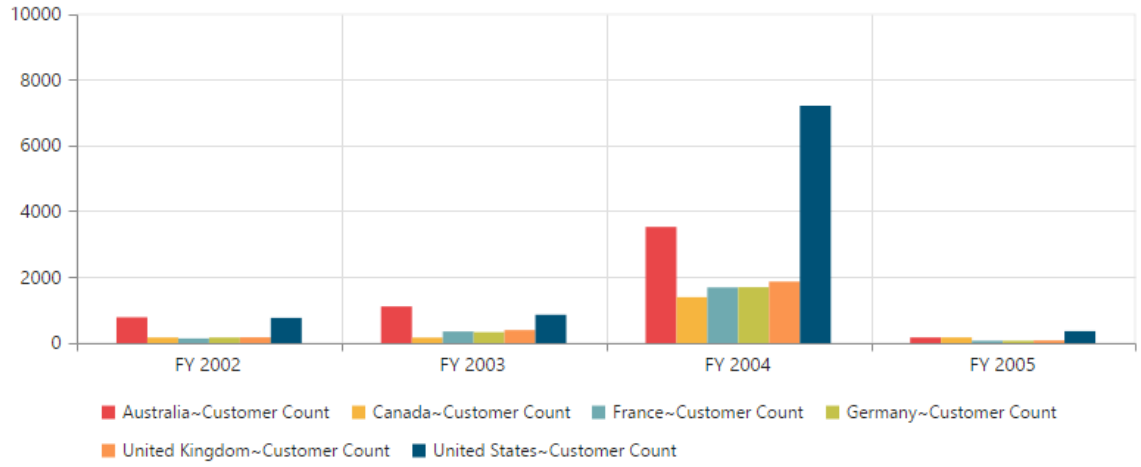
```
public static void RegisterRoutes(RouteCollection routes) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
}
```

Note: In this example, “PivotChartDemo” indicates the name and root namespace of the Web Application created in Visual Studio IDE and “OlapService” indicates the name of the WCF service created.

Now, PivotChart is rendered with Customer Count over a period of fiscal years across different customer geographic locations.



Dimensions

Set size in percentage

You can customize the PivotChart dimension by setting the width and height of the control in percentage.

ASPX-CS

```
//Set size to Chart container
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size => size.Height("80%").Width("80%"))
<style>
#PivotChart1 {
width:100%;
height:450px;
}
</style>
```

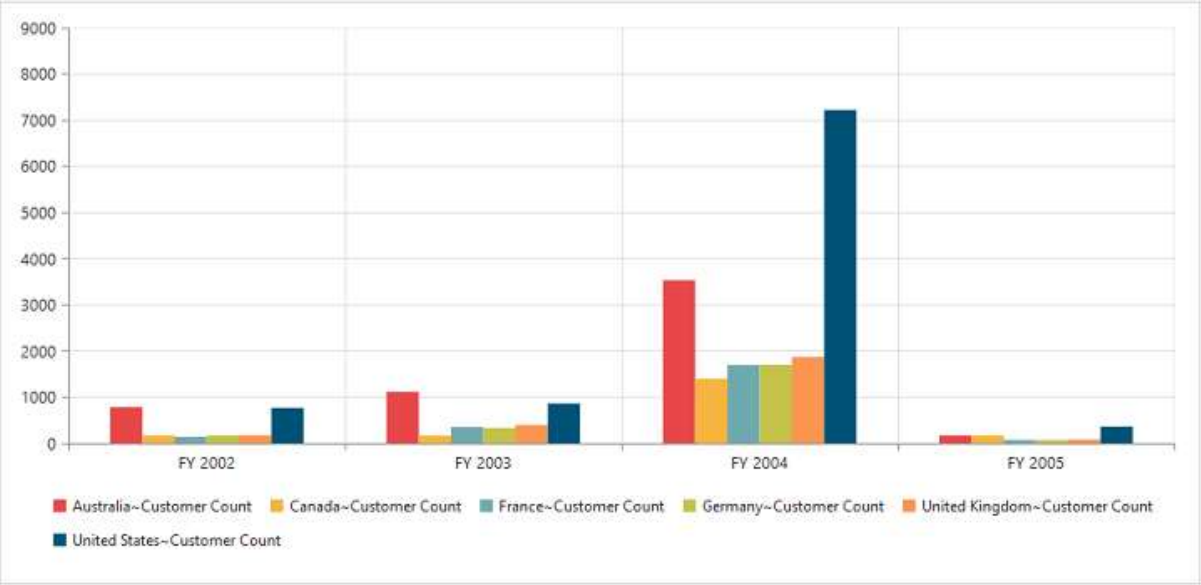
Copyright © 2001 - 2020 Syncfusion Inc.

Set size in pixels

You can customize the PivotChart dimension by setting the width and height of the control in pixels.

ASPX-CS

```
//Set size to Chart container
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size => size.Height("460px").Width("950px"))
<style>
#PivotChart1 {
width:950px;
height:460px;
}
</style>
```



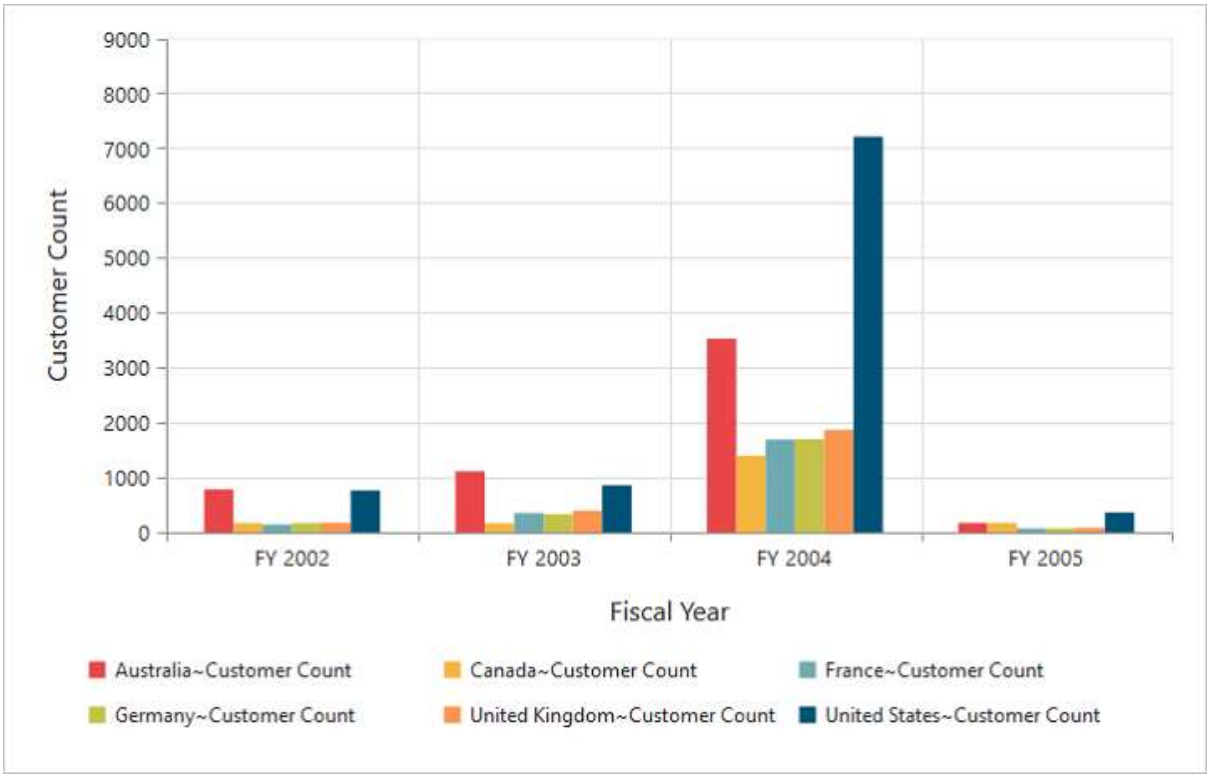
Responsive

PivotChart control supports responsive rendering based on the target device (desktop & tablet) resolution. It supports resolution upto 1024x600. You can enable responsiveness in PivotChart by setting IsResponsive property to true.

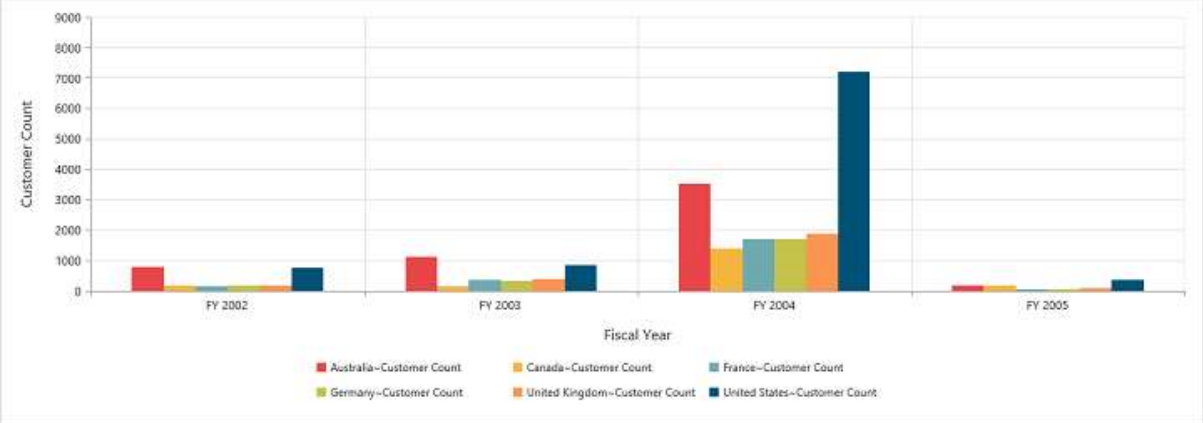
ASPX-CS

```
//Enable responsiveness to change the Chart size dynamically.
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).IsResponsive(true)
<style>
#PivotChart1 {
min-height: 275px;
min-width: 525px;
height: 460px;
width: 100%;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Normal View



ResponsiveView

Chart Types

Essential **PivotChart MVC** supports 18 different types of chart as follows:

- Column
- Stacking Column
- Bar
- Stacking Bar
- Pie
- Pyramid
- Funnel

Copyright © 2001 - 2020 Syncfusion Inc.

- Line
- Step Line
- Spline
- Area
- Step Area
- Spline Area
- Stacking Area
- Doughnut
- Scatter
- Bubble
- WaterFall

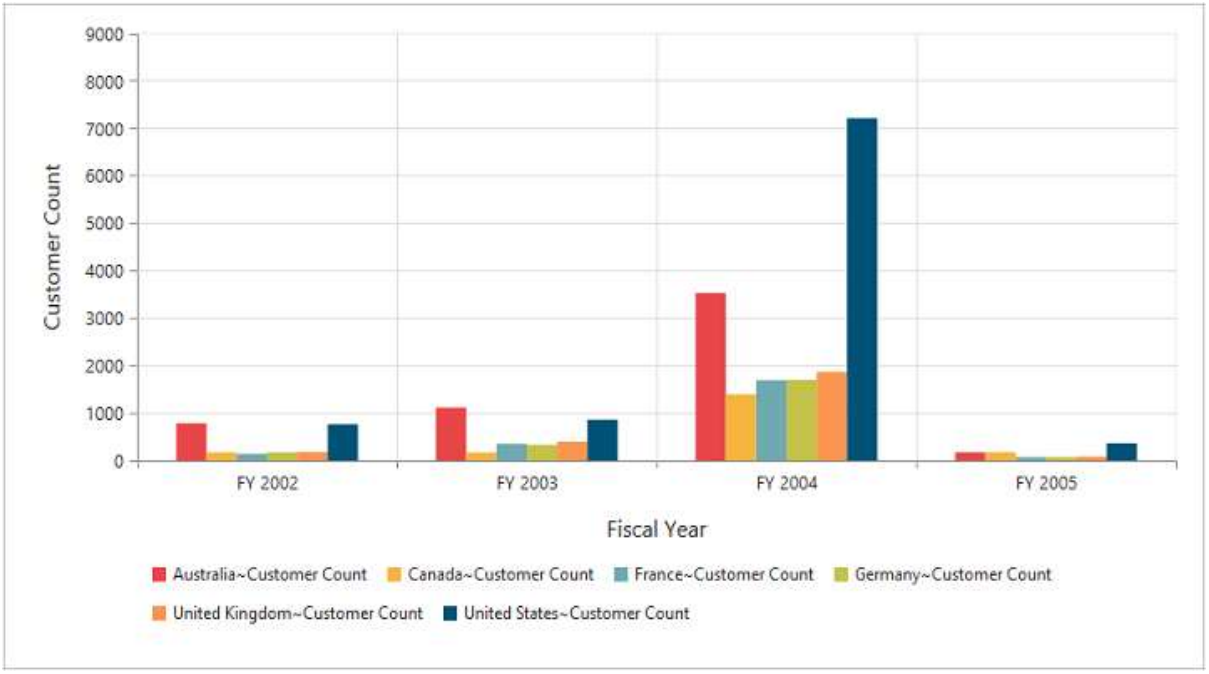
Column Chart

Column Chart is the most commonly used chart type. It uses vertical bars (called columns) to display different values of one or more items. Points from adjacent series are drawn as bars next to each other. It is used to compare the frequency, count, total or average of data in different categories. It is ideal to show the variations in the value of an item over a period of time.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Column); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays a **Column Chart**.



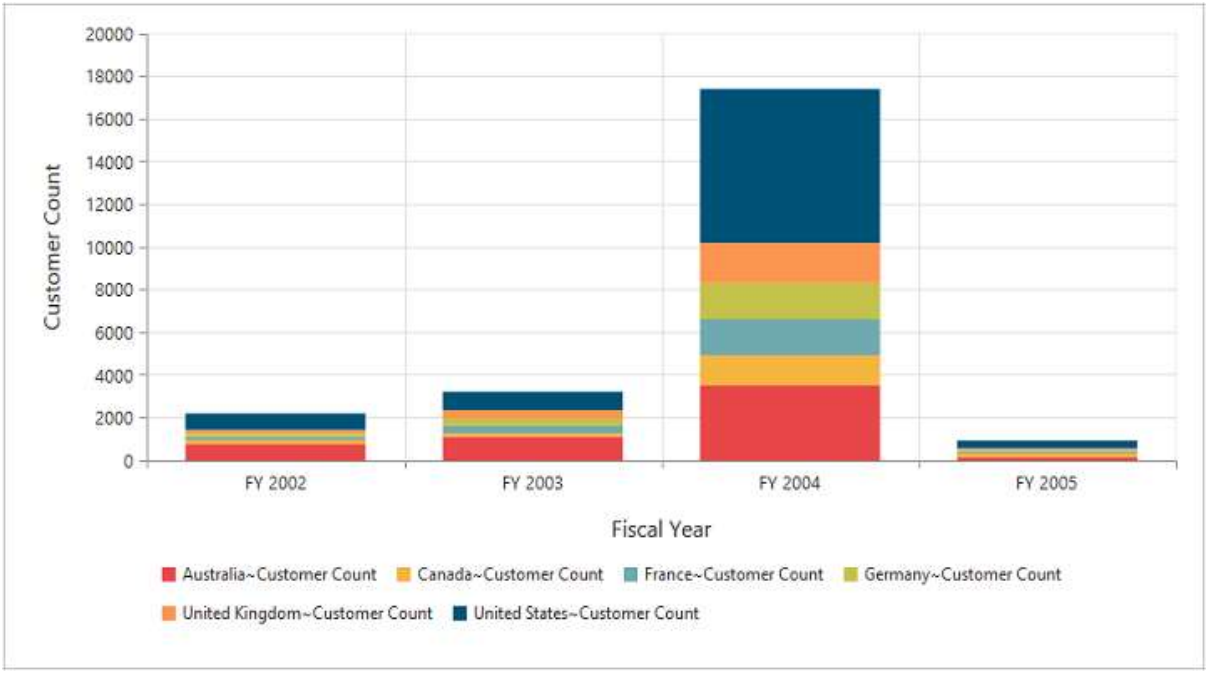
Stacking Column Chart

Stacking Column Chart is similar to column charts except the Y-values. These Y-values stack on top of each other in a specified series order. This helps to visualize the relationship of parts to whole across categories.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StackingColumn); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **stacking Column Chart**.



Bar Chart

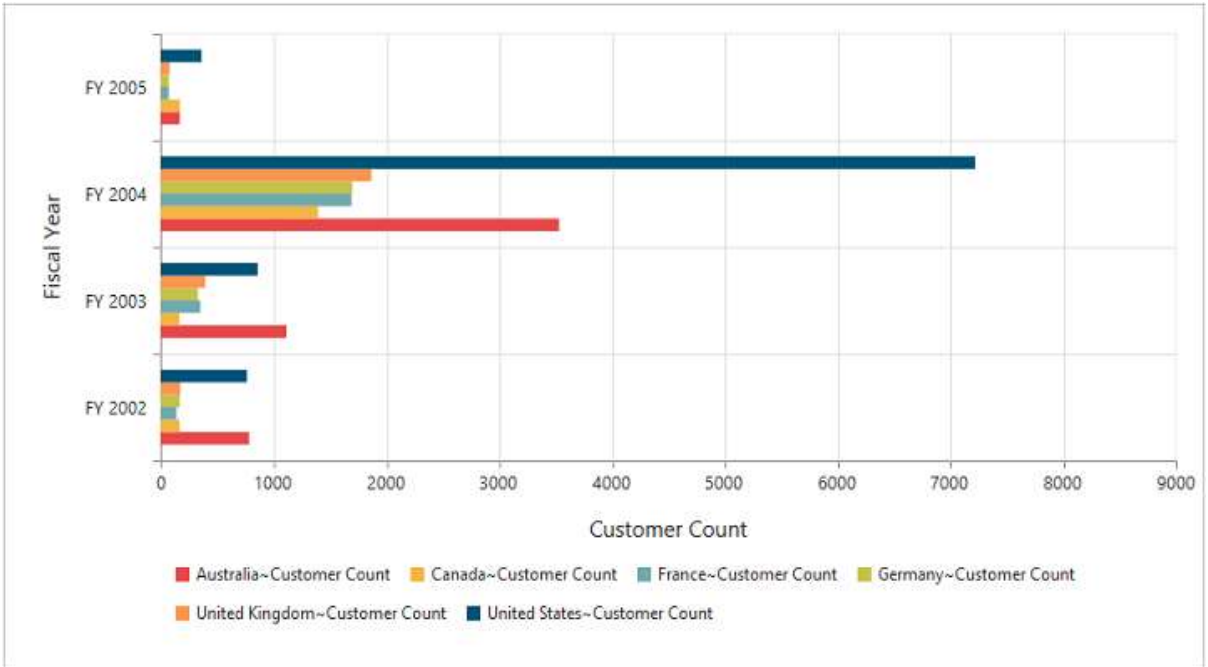
The **Bar Chart** displays horizontal bars for each point in the series and points from adjacent series. Bar charts are used to compare values across categories, for displaying the variations in the value of an item over time or for comparing the values of several items at a single point in time.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Bar); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays a **Bar Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Stacking Bar Chart

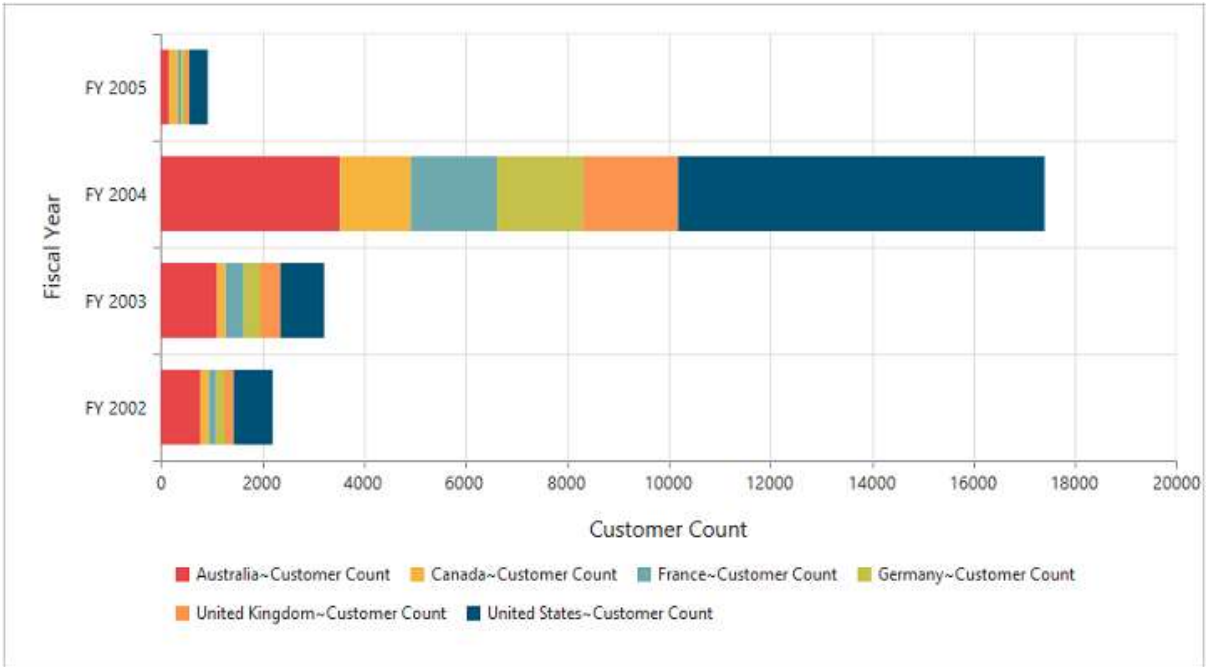
Stacking Bar Chart is a regular **bar** chart with the X-values stacked on top of each other in the specified series order.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StackingBar); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **Stacking Bar Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Pie Chart

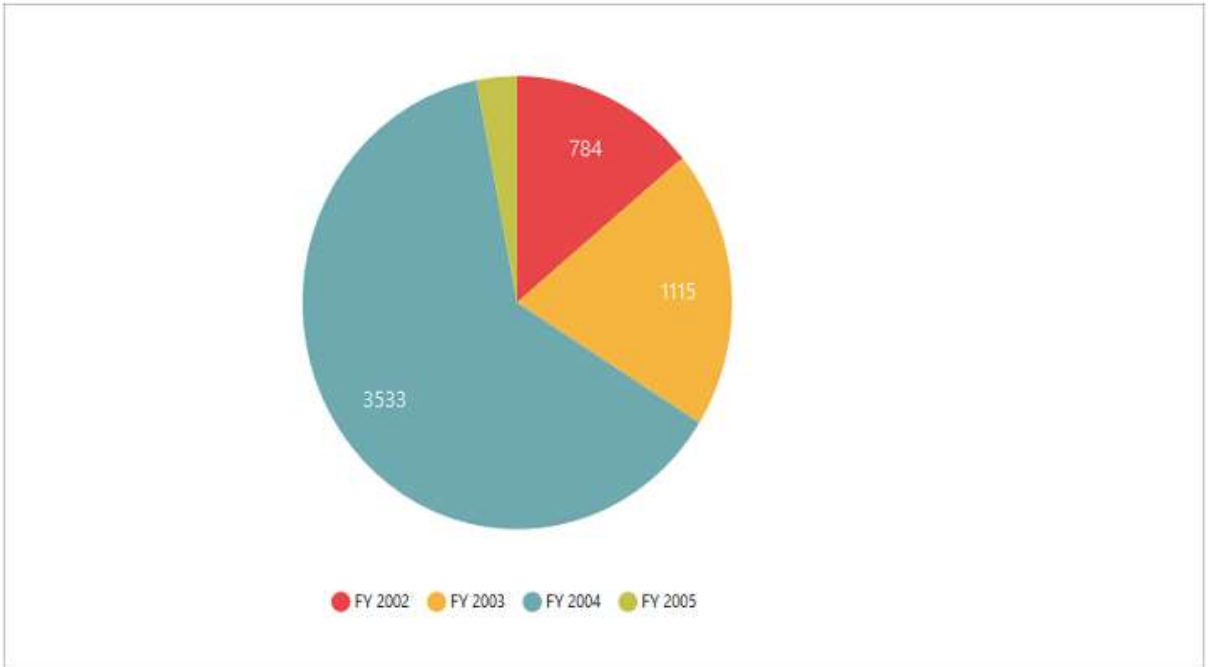
A **Pie chart** is used to summarize a set of categorical data or displaying different values of a given variable (e.g., percentage distribution). This type of chart is a circle divided into a series of segments. Each segment represents a particular category.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Pie); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays a **Pie Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Pyramid Chart

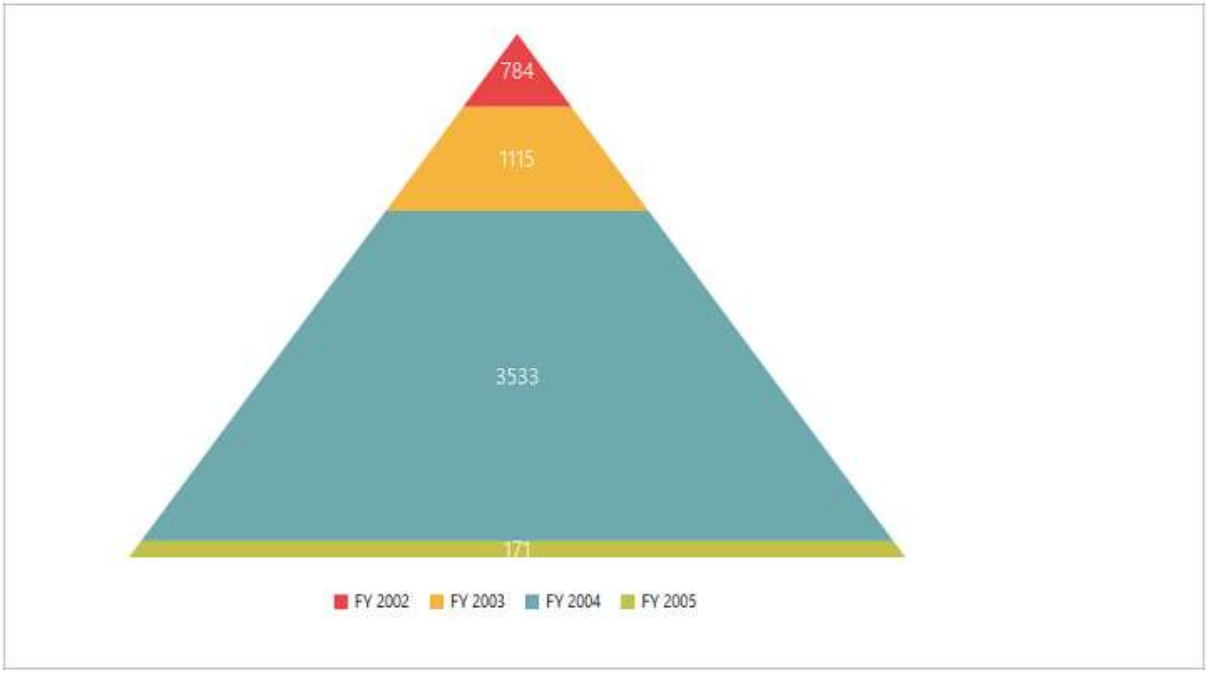
The **Pyramid Chart** type displays the data in the form of a triangle. It helps you to visualize data in a hierarchical structure without any axes.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Pyramid); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screen shot displays the **Pyramid Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Funnel Chart

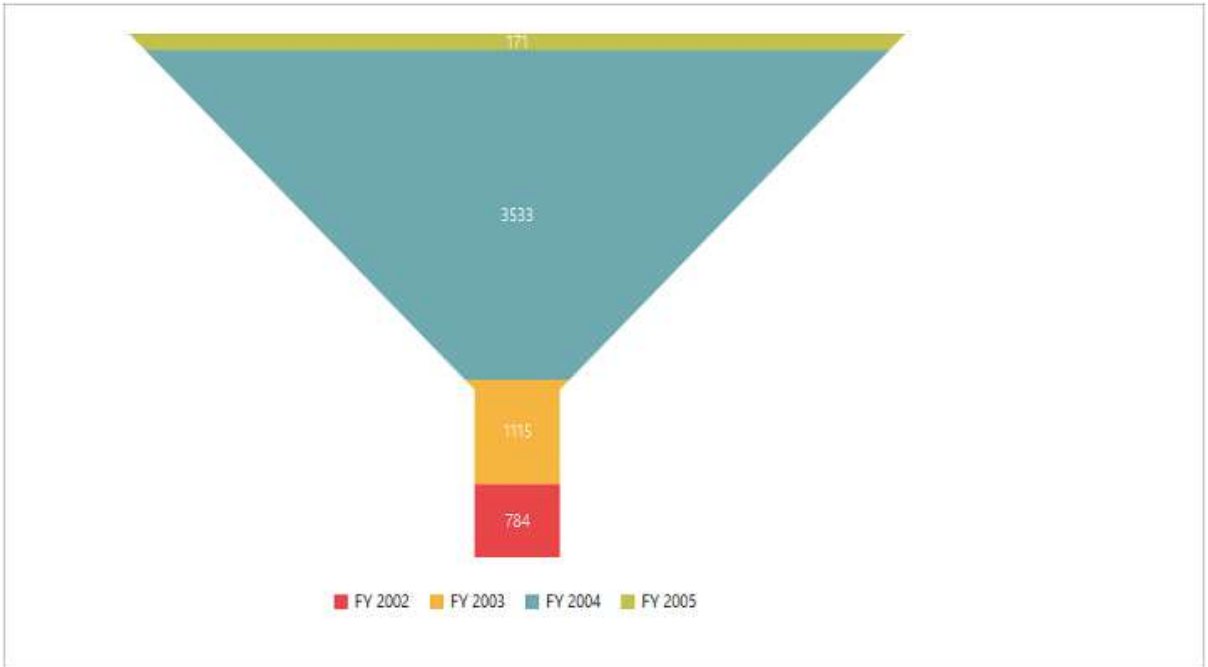
The **Funnel Chart** type displays the data in the form of an inverted triangle. It helps you to visualize data in a hierarchical structure without any axes.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Funnel); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screen shot displays the **Funnel Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Line Chart

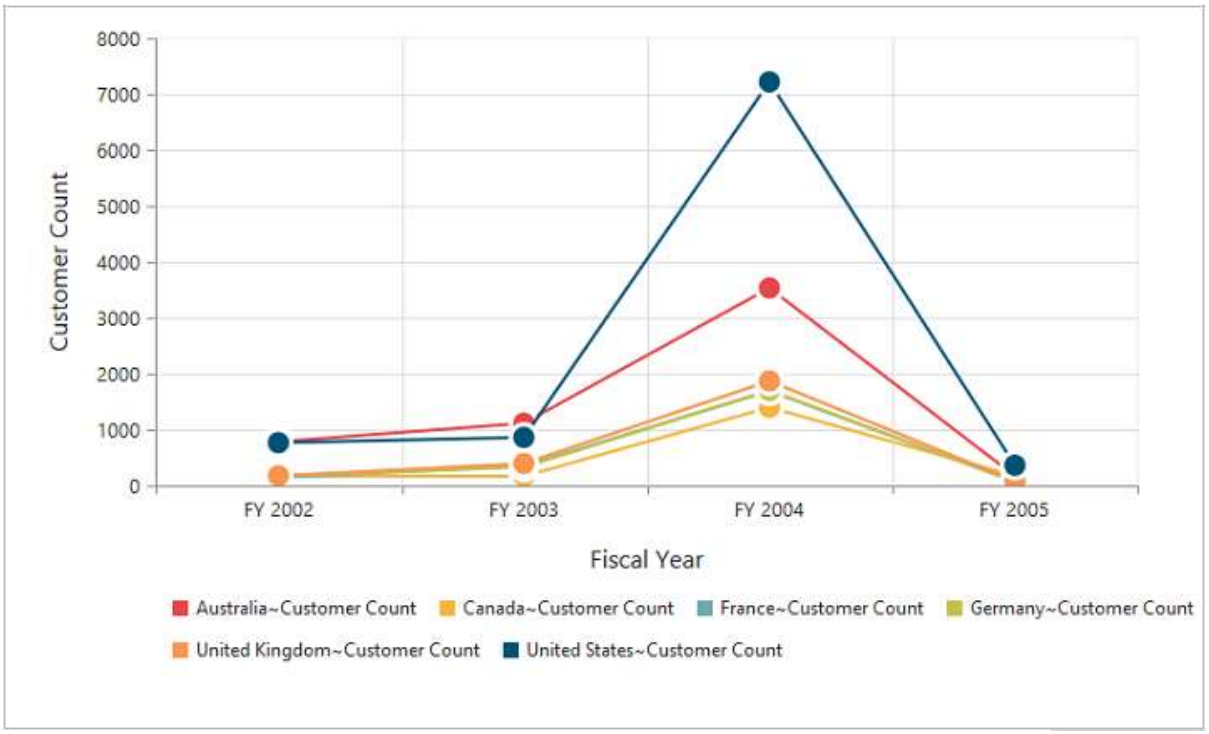
The **Line Chart** joins the data points on a plot using straight lines that show trends in data at equal intervals.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Line); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **Line Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Step Line Chart

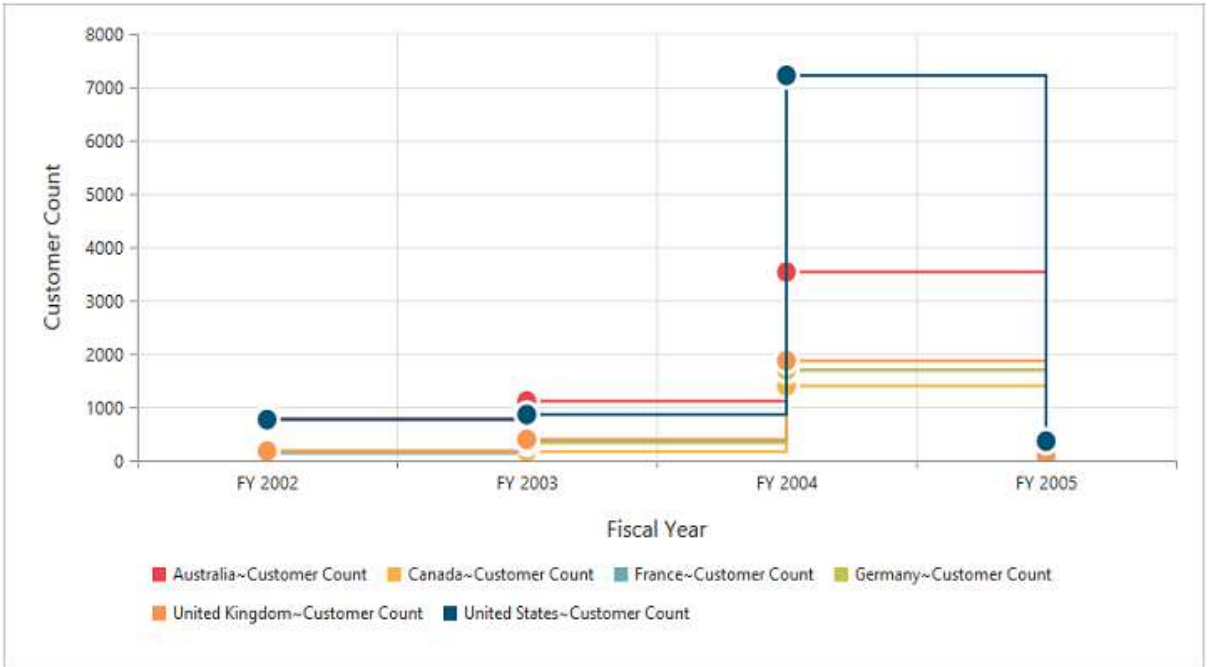
Step Line Chart uses horizontal and vertical lines to connect the data points resulting in a step like progression.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StepLine); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **Step Line Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Spline Chart

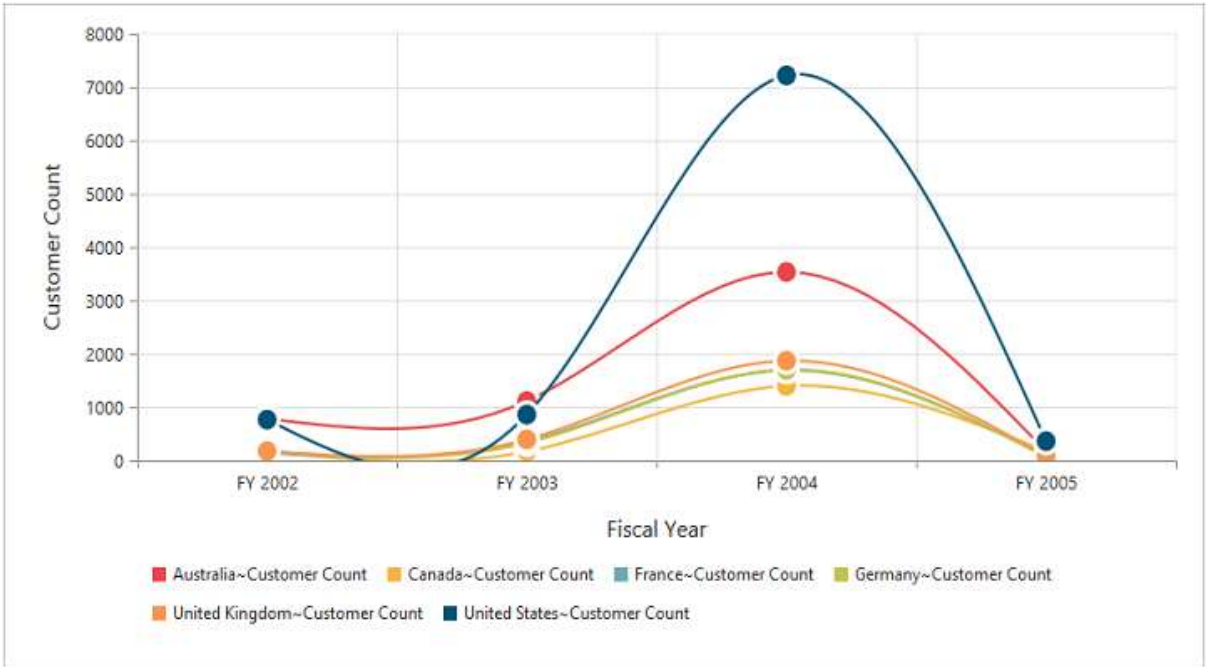
The **Spline Chart** is similar to line charts except it connects different data points using curve lines instead of straight lines.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Spline); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **Spline Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Area Chart

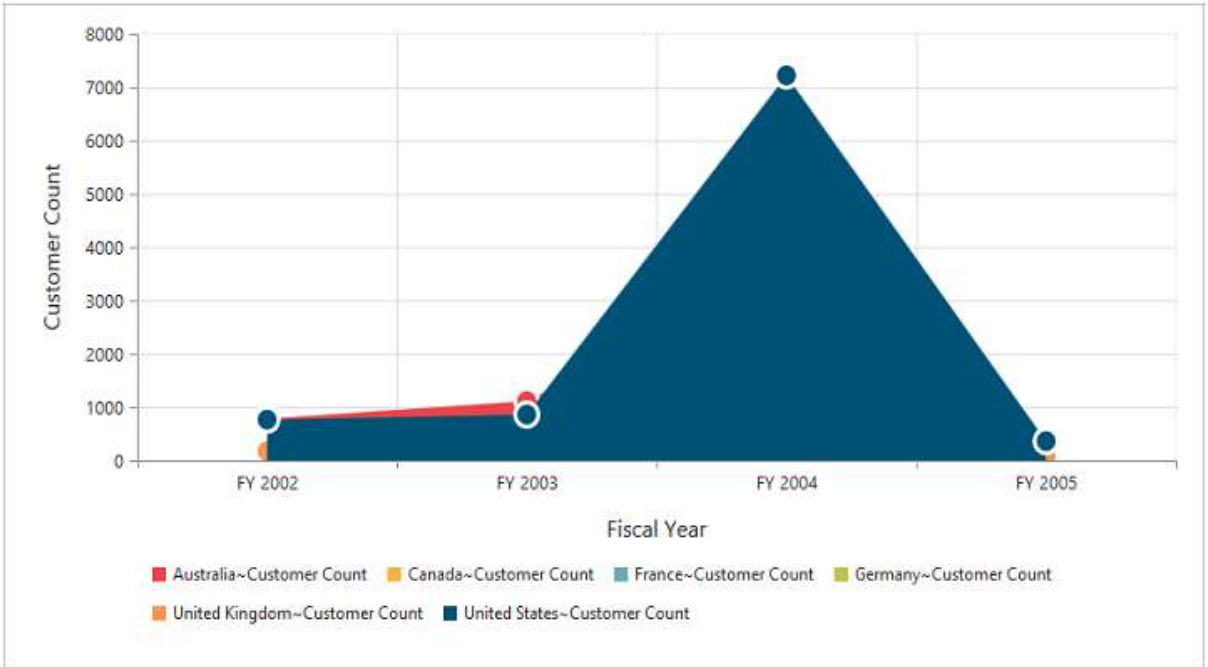
Area Chart emphasizes the degree of change of values over a period of time. Instead of rendering data as discrete bars or columns, an area chart renders it in a continuous ebb-and-flow pattern as defined against the y-axis.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Area); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **Area Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Step Area Chart

Step Area chart is similar to the regular area chart except for a straight line tracing the shortest path between the data points. The values are connected by continuous vertical and horizontal lines forming a step like progression.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StepArea); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays a **Step Area Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Spline Area Chart

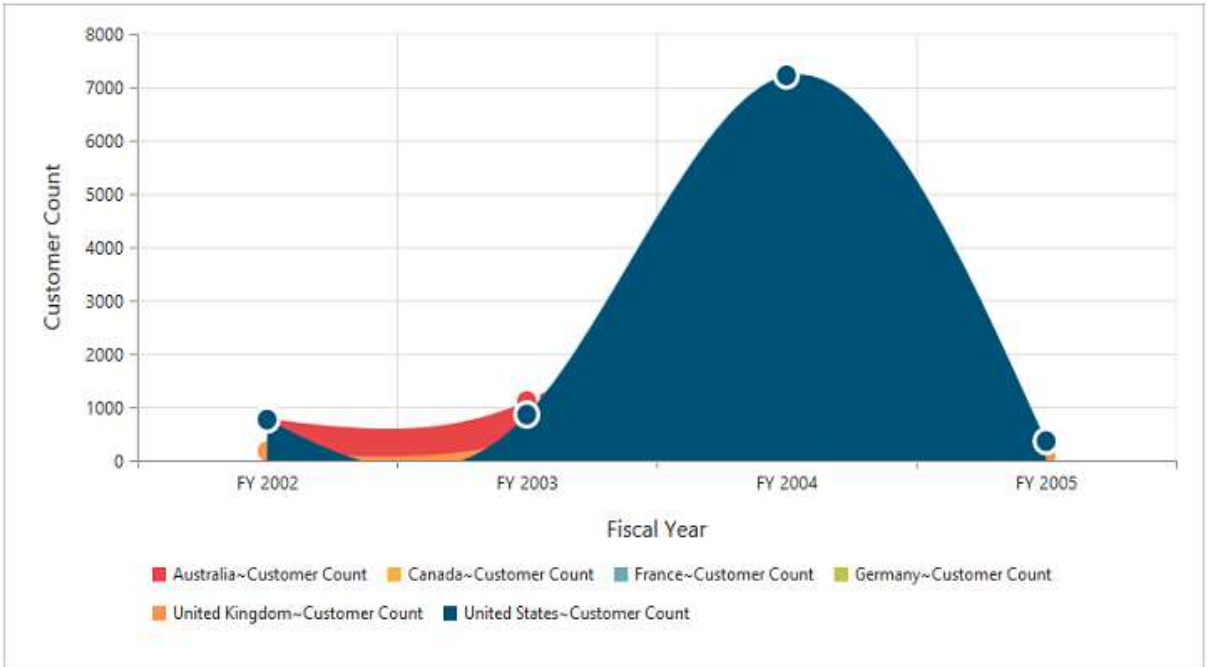
Spline Area chart is similar to Area Chart with the difference in which the data points of a series are connected. It connects each series of points by a smooth **spline curve**.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType. SplineArea); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following Screenshot displays a **Spline Area Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



Stacking Area Chart

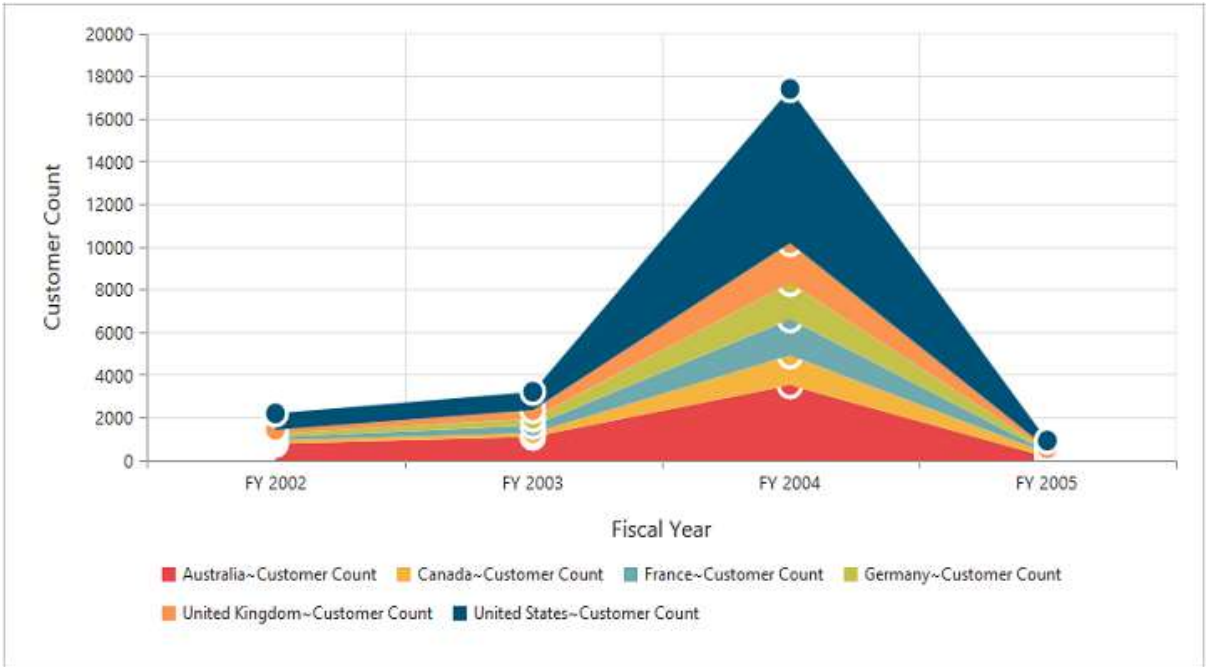
Stacking Area chart is similar to regular area chart except the “Y-values”. These “Y-values” stack on top of each other in the specified series order. This helps to visualize the relationship of parts to whole across categories.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StackingArea); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays a **Stacking Area Chart**.

Copyright © 2001 - 2020 Syncfusion Inc.



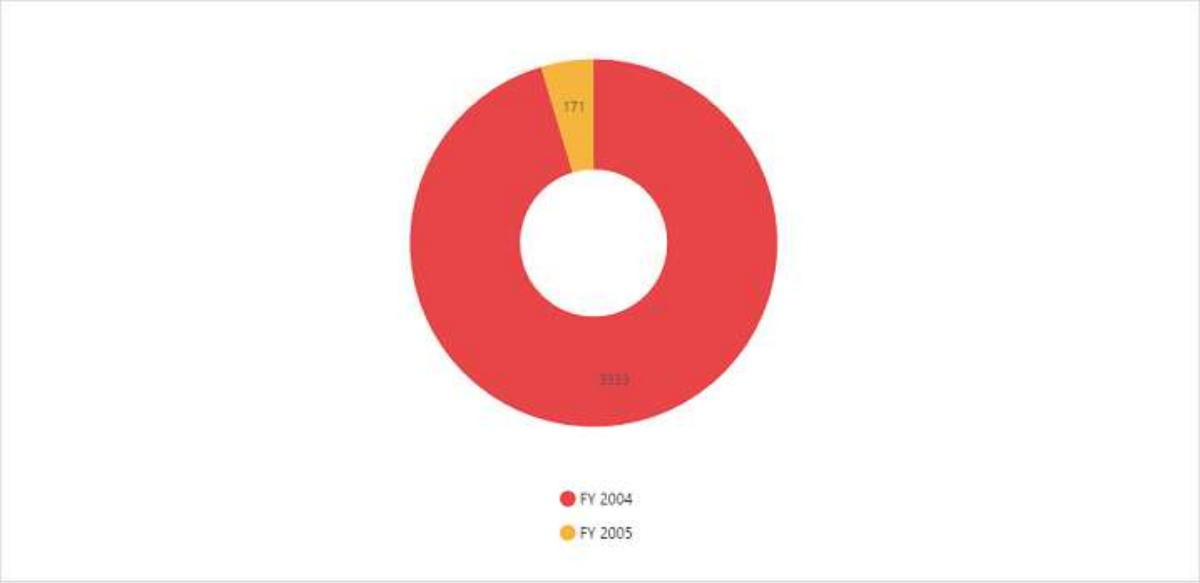
Doughnut Chart

A **Doughnut chart** is also used to summarize a set of categorical data which possesses a doughnut like structure divided into a series of segments. Each segment represents a particular category.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Doughnut); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays a **Doughnut Chart**.



Copyright © 2001 - 2020 Syncfusion Inc.

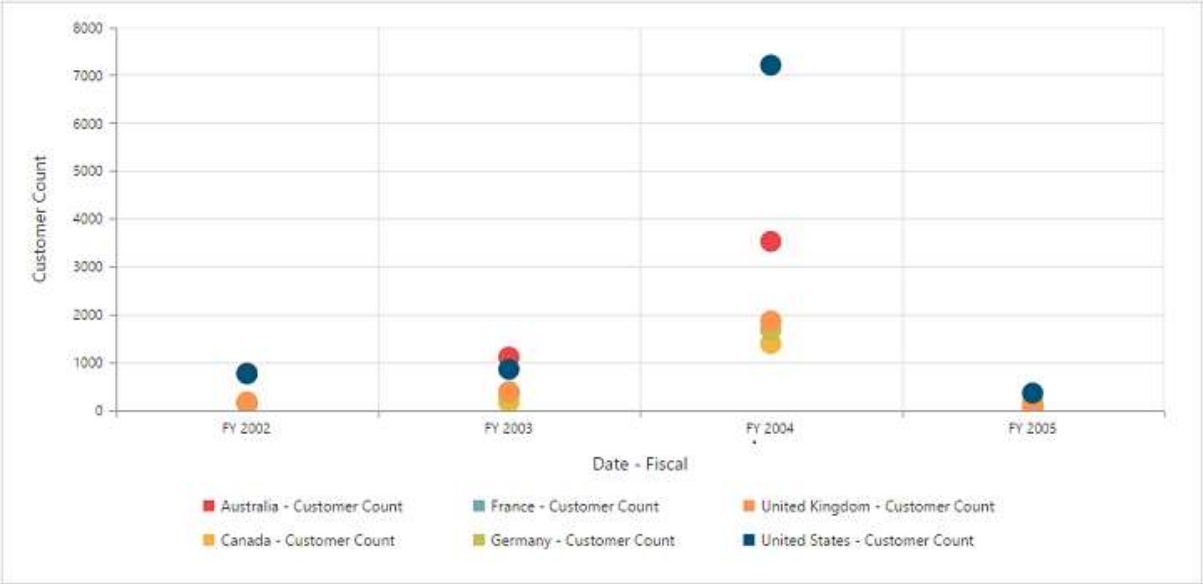
Scatter Chart

The **Scatter Chart** type displays the data as a collection of points corresponding to the associated values.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Scatter); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screen shot displays the **Scatter Chart.**



Bubble Chart

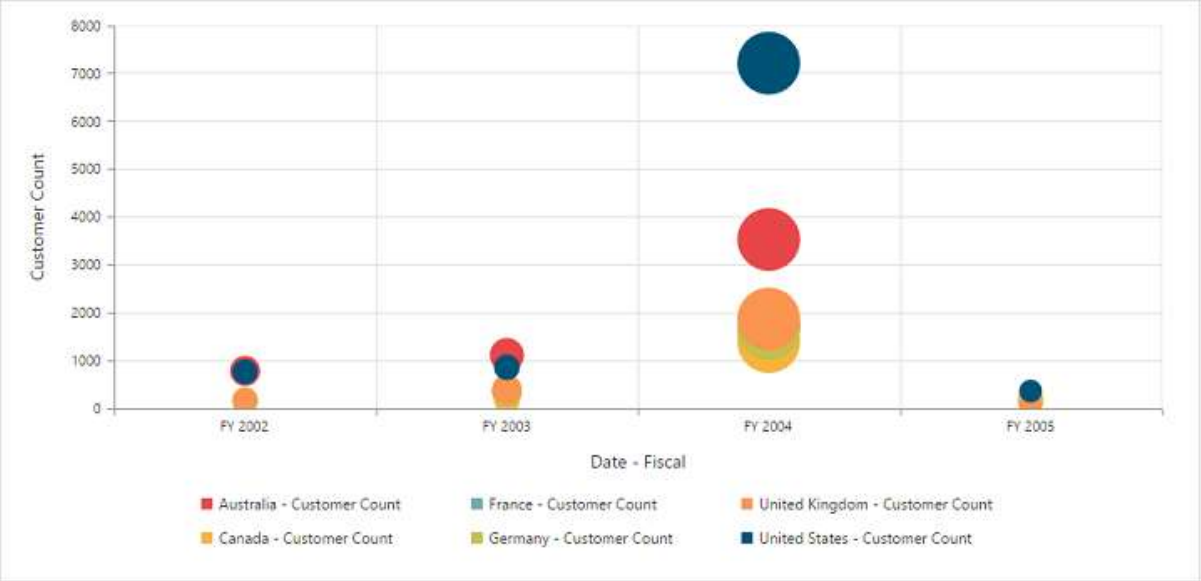
The **Bubble Chart** type displays the data as a collection of bubbles.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Bubble); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screen shot displays the **Bubble Chart.**

Copyright © 2001 - 2020 Syncfusion Inc.



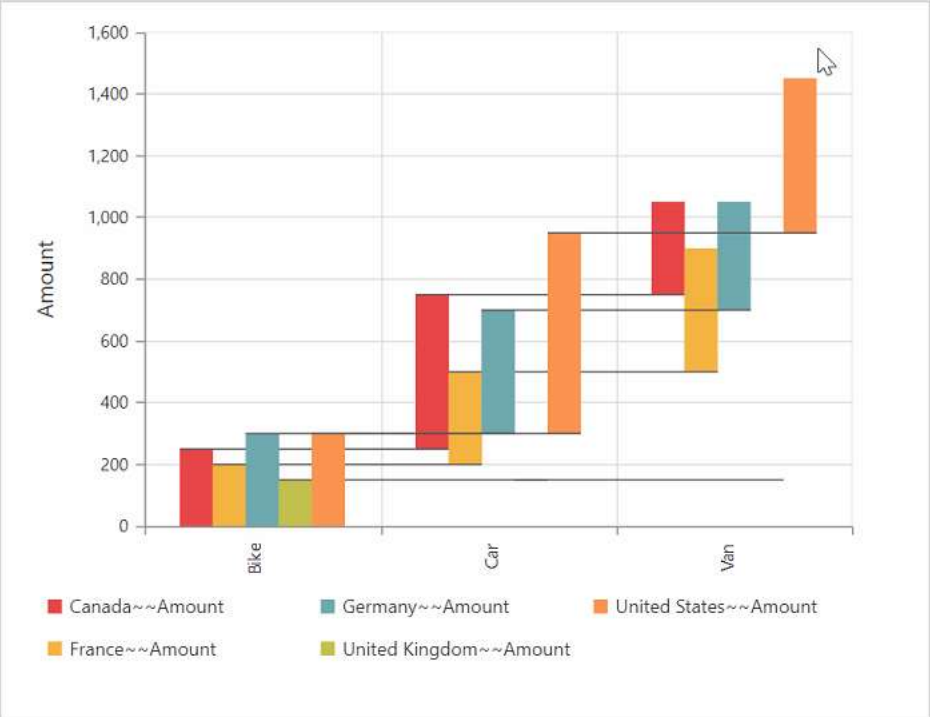
WaterFall chart

The **waterfall chart** type is used to show how an initial value is increased and decreased by a series of intermediate values, leading to a final value.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.WaterFall); }).Size(size =>
size.Height("460px").Width("950px"))
```

The following screenshot displays the **waterfall chart:**



Combination Chart

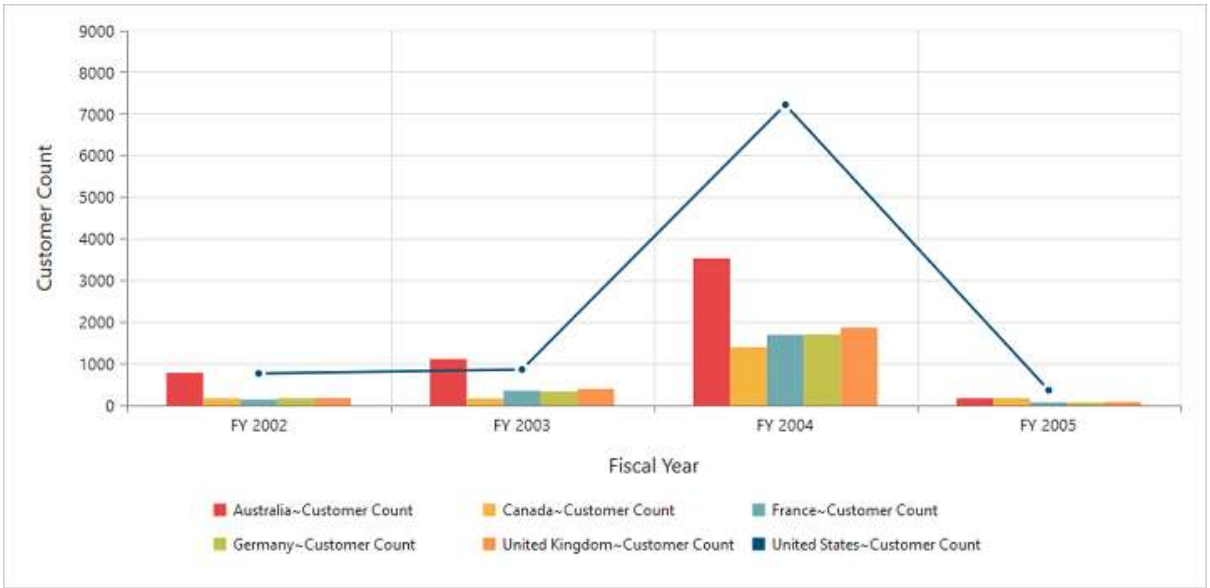
A **combination Chart** combines two or more series types in a single Chart. But there are some limitations in the combination Chart. They are:

1. Can't combine Column and Bar series.
2. Pie Chart can't be used with other series types.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Column); }).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
this.model.series[5].type = ej.PivotChart.ChartTypes.Line;
this.model.series[5].marker.visible = true;
</script>
```

The following screenshot displays a **combination Chart**.



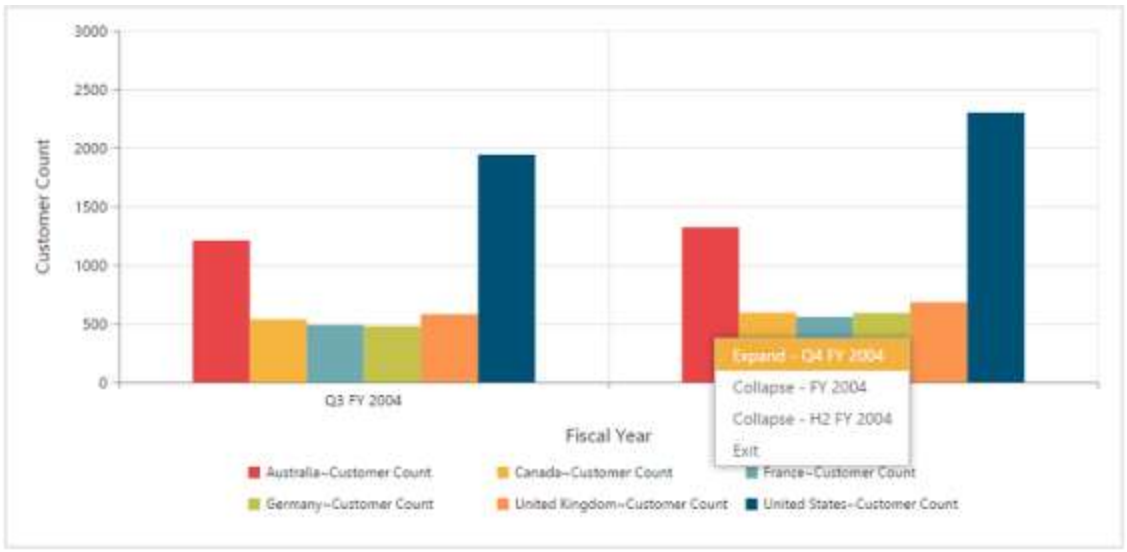
Drill Operation

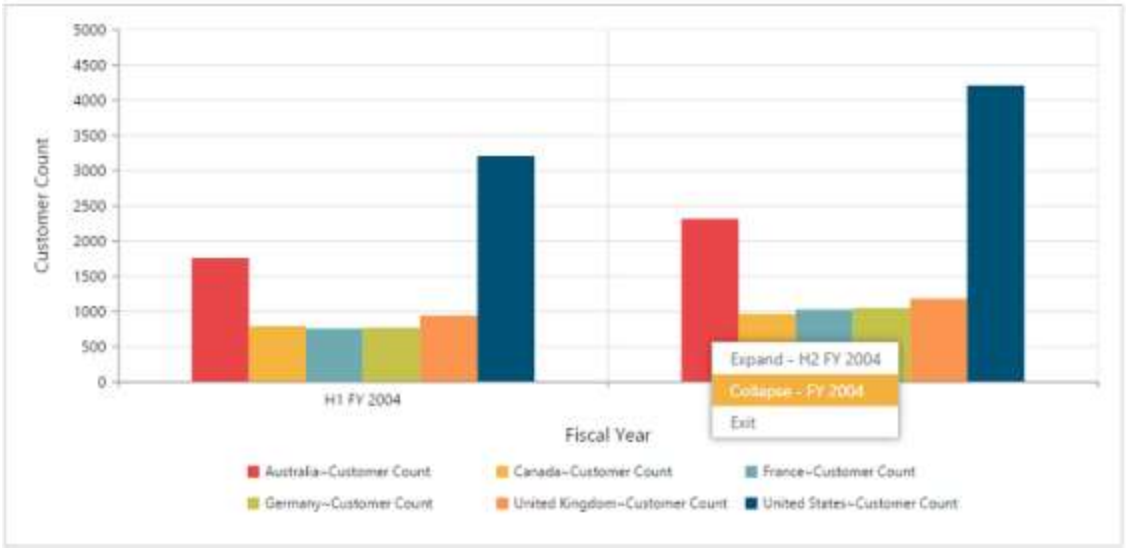
This is a basic feature of PivotChart through which the amount of information can be limited, for a better view. It allows you to drill down to access the detailed level of data or drill up to see the summarized data by using the Context Menu present in the PivotChart.

Drill up, also called roll up, navigates from more detailed data to less detailed data, by climbing up a concept hierarchy for a dimension.

Drill down, also called roll down, is the reverse of drill up. It navigates from less detailed data to more detailed data, by climbing down a concept hierarchy for a dimension.

Copyright © 2001 - 2020 Syncfusion Inc.





DrillSuccess event gets triggered when you right-click on the PivotChart and select any option available in the context menu to perform drill up or drill down operation.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(oEve =>
{oEve.DrillSuccess("DrillSuccess")});
<script type="text/javascript">
function DrillSuccess(args) {
alert("Drill Success");
}
</script>
```

Legend

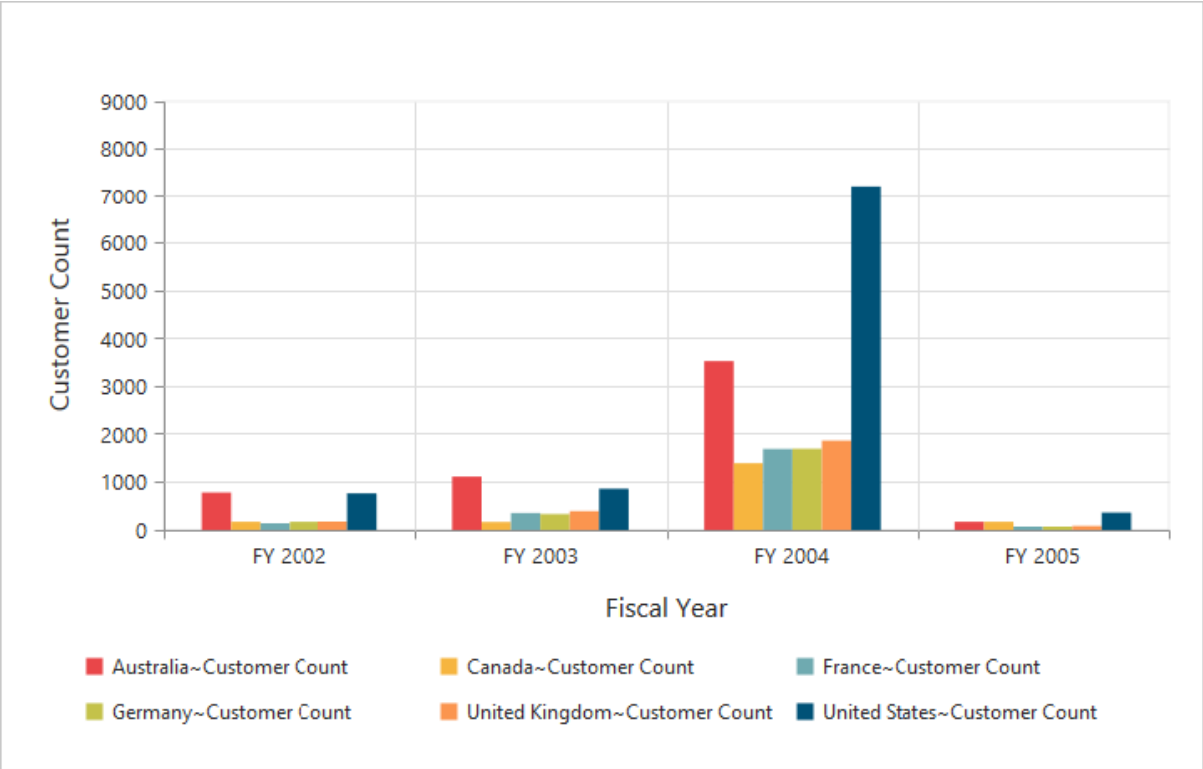
Legend Visibility

You can enable or disable legend using the Visible property inside the Legend object.

Note: By default, the legend is visible in PivotChart.

ASPX-CS

```
//Legend Visibility set as true
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true))
```



Legend Shape

You can customize the legend Shape in PivotChart control. Default value of legend shape is “Rectangle”. Following legend shapes that are supported:

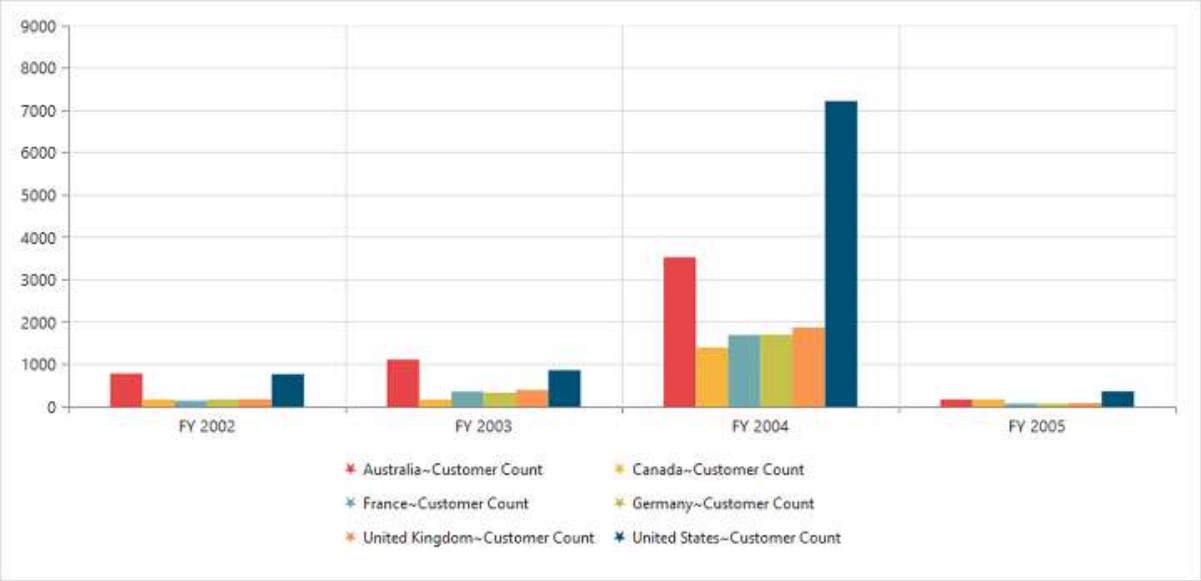
- Rectangle
- Circle
- Cross
- Diamond
- Pentagon
- Hexagon
- Star
- Ellipse
- Triangle etc.

ASPX-CS

```
//Applying legend shape
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).Shape(ChartShape.Star))
```



Legend Position

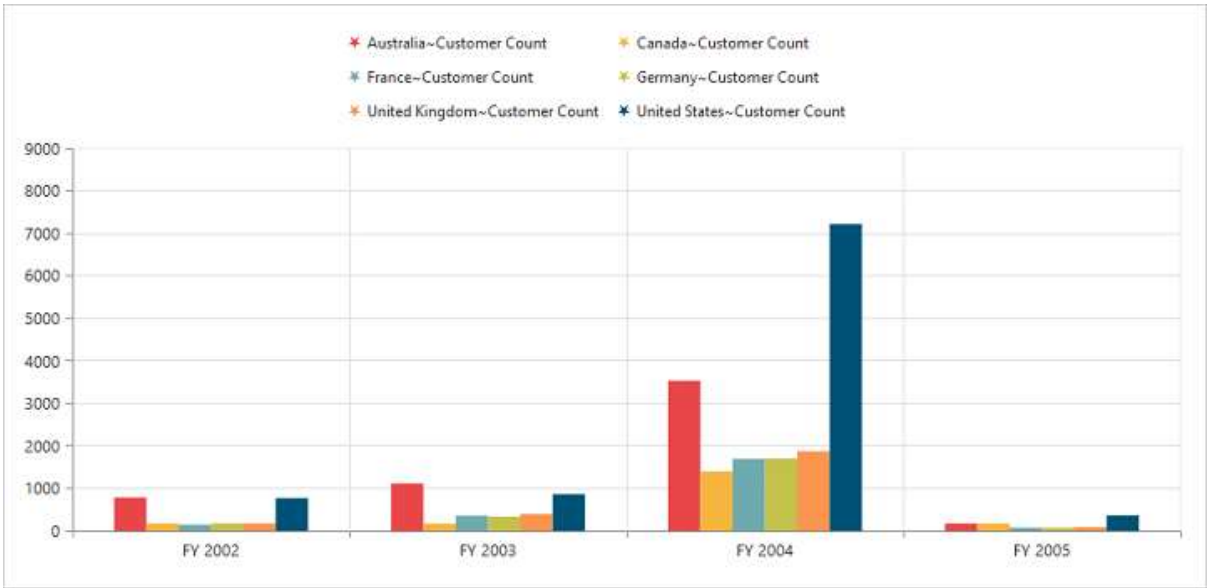
By using the Position property, you can place the legend at top, bottom, left or right of the PivotChart.

Note: Default value of legend position is "bottom" in PivotChart.

ASPX-CS

```
//Place the legend at top of the Chart
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).Shape(ChartShape.Star).Position(LegendPosition.Top))
```

Copyright © 2001 - 2020 Syncfusion Inc.

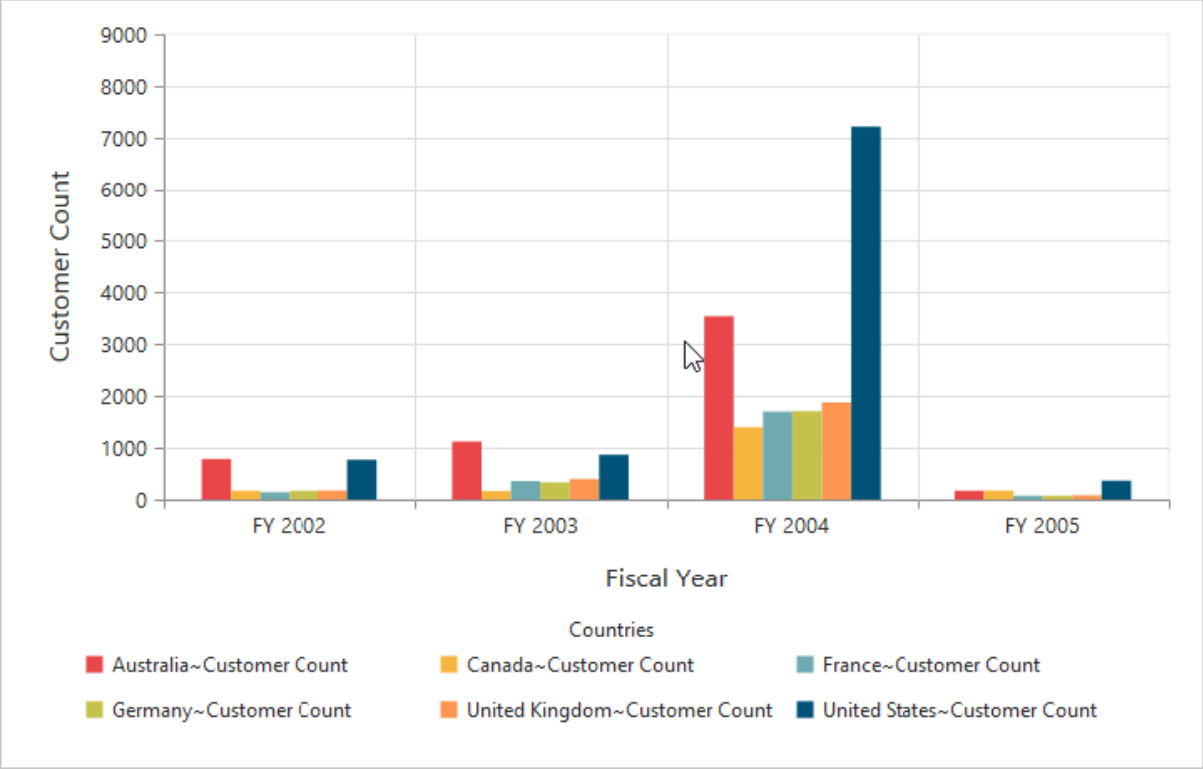


Legend Title

To add the legend title, you have to specify the title text in `title.Text` property.

ASPX-CS

```
//Add title to the Chart legend
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).Title(title => title.Text("Countries")))
```



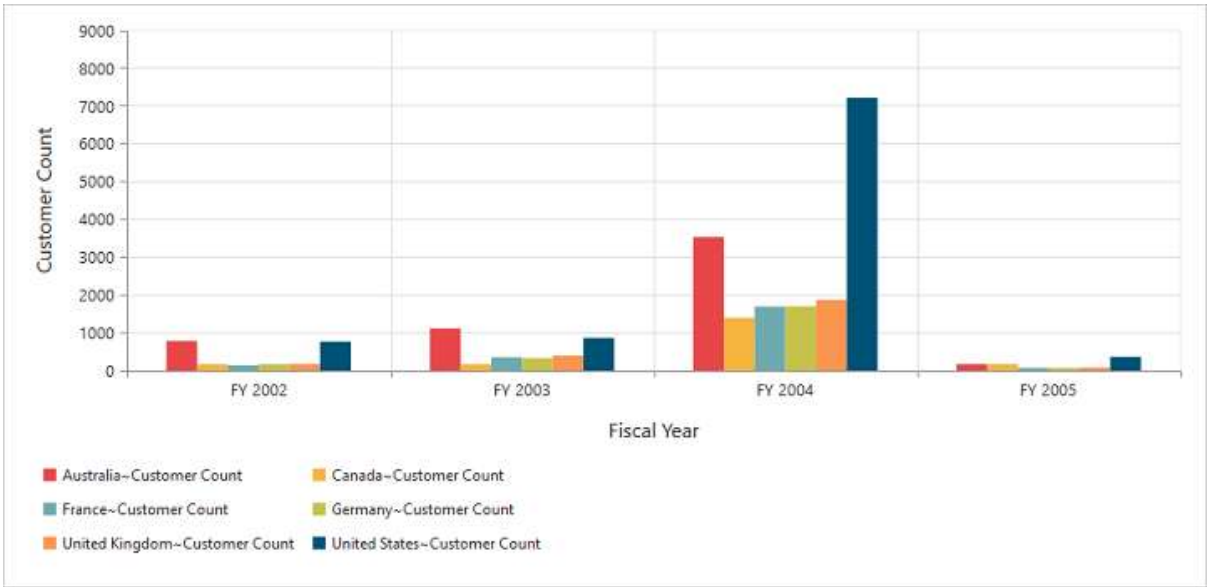
Copyright © 2001 - 2020 Syncfusion Inc.

Legend Alignment

You can align the legend to center, far and near based on its position in the Chart area using the Alignment option.

ASPX-CS

```
//Aligning the legend near to the Chart
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).Alignment(TextAlignment.Near))
```



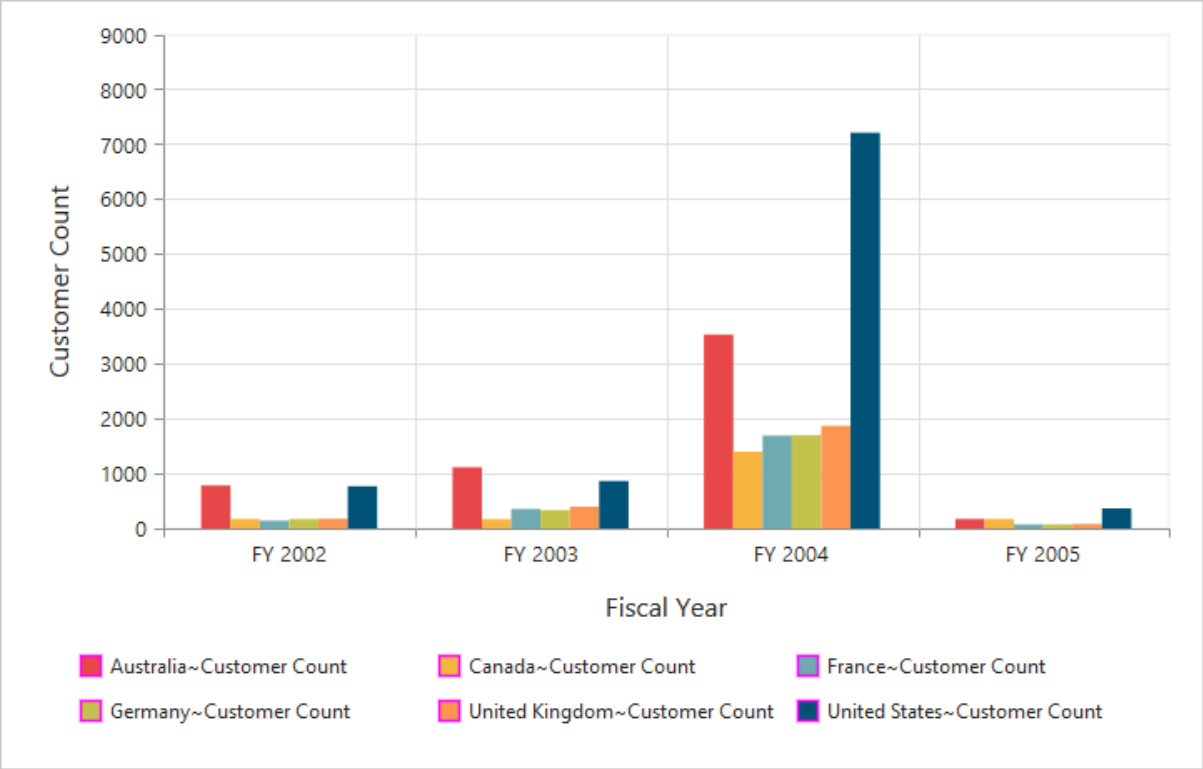
Legend Items - Size and Border

By using the legend ItemStyle.width, ItemStyle.height and ItemStyle.border properties, you can change the legend items - size and border.

ASPX-CS

```
//Changing legend items border, height and width
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).ItemStyle(ItemSize => ItemSize.Border(border =>
border.Color("magenta").Width(1.5)).Height(12).Width(12)))
```

Copyright © 2001 - 2020 Syncfusion Inc.



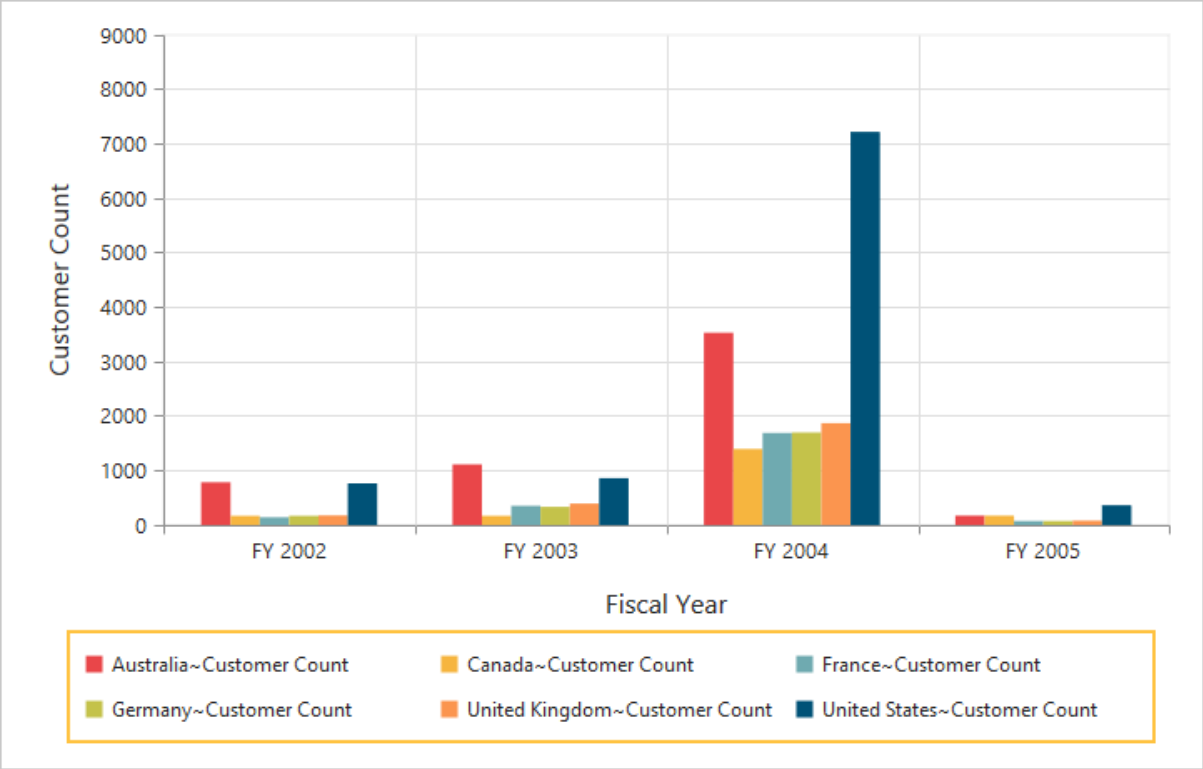
Legend Border

By using the Border option in legend, you can customize border color and width.

ASPX-CS

```
//Setting border color and width to legend
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).Border(border => border.Color("#FFC342").Width(2)))
```

Copyright © 2001 - 2020 Syncfusion Inc.



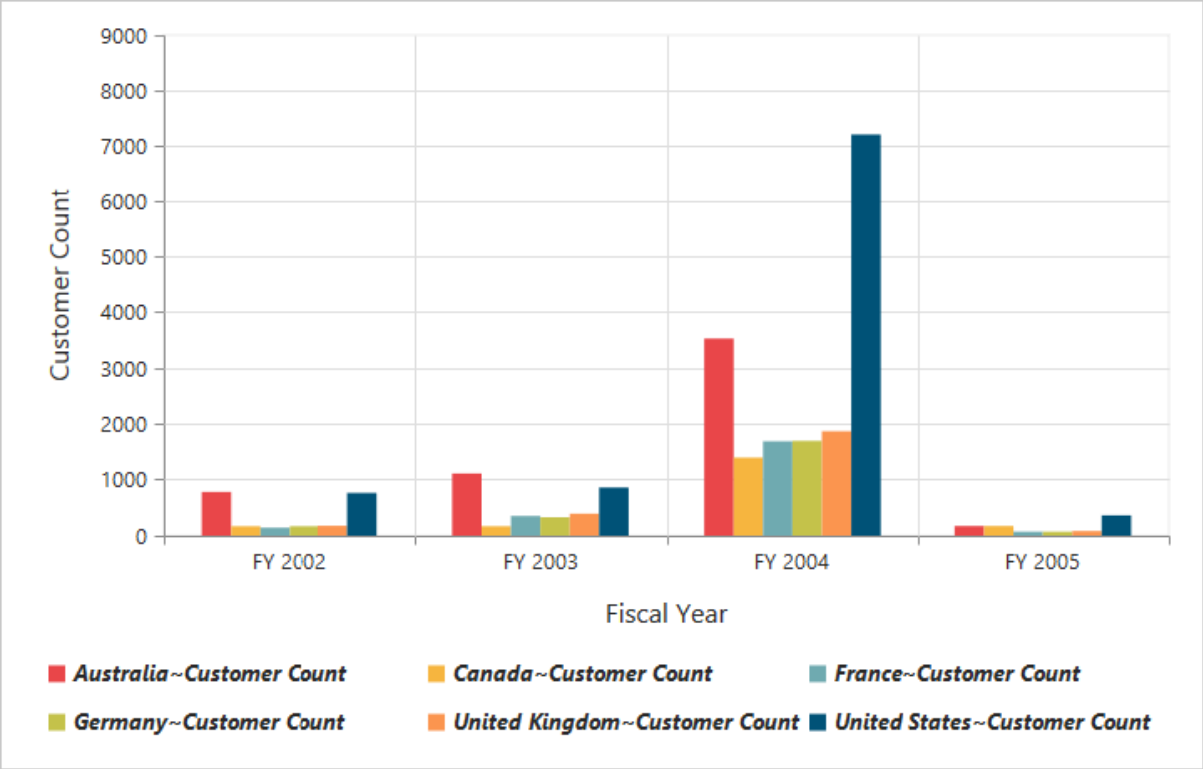
Legend Text

By using the Font option, you can customize the font family, font style, font weight and size of the legend text.

ASPX-CS

```
///Customizing the legend text
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).Legend(legend =>
legend.Visible(true).Font(font => font.FontFamily("Segoe
UI").FontWeight(ChartFontWeight.Bold).FontStyle(ChartFontStyle.Italic).Si
ze("13px")))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Axes

Label Format

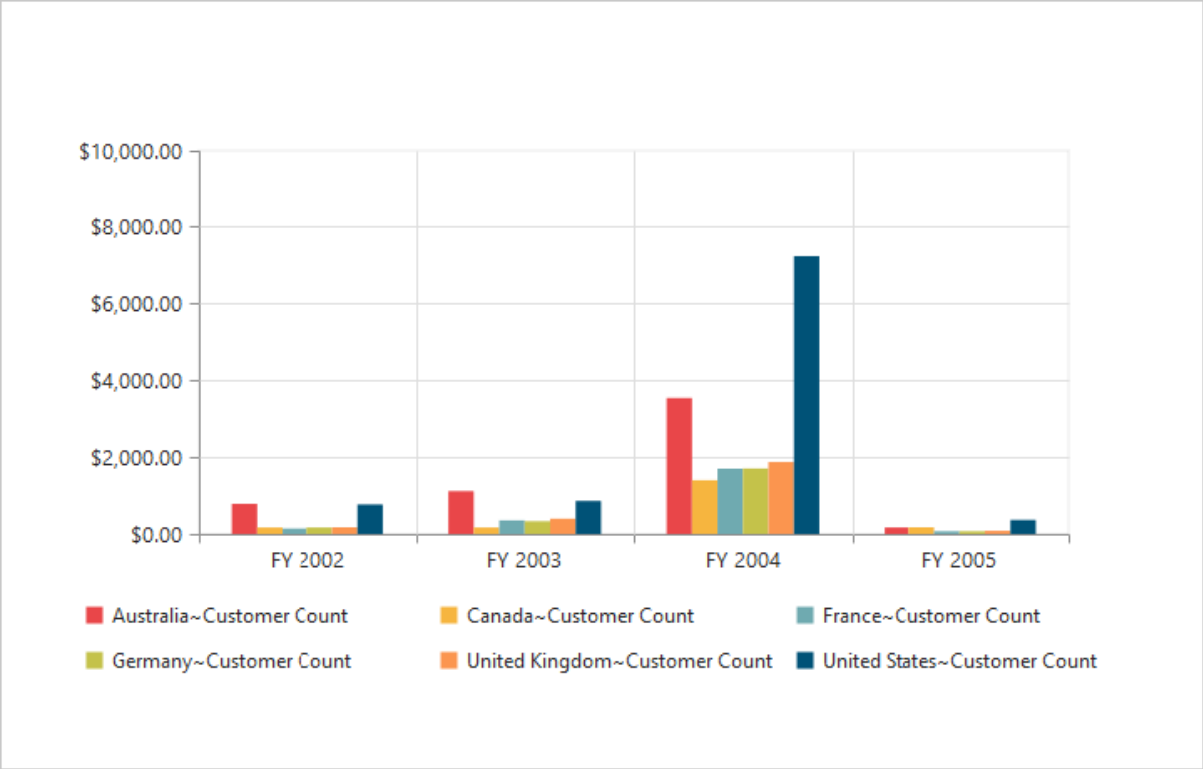
Format Numeric labels

By using the LabelFormat property, you can format the numeric labels. Numeric values can be formatted with n (number with decimal points), c (currency) and p (percentage) commands.

ASPX-CS

```
//Applying currency format to axis labels
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryYAxis(primaryY =>
primaryY.LabelFormat("c")).Url(Url.Content("/RelationalChartService.svc")
).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Following table describes the result on applying some commonly used label formats on numeric values.

Label Value	Label Format Property Value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place

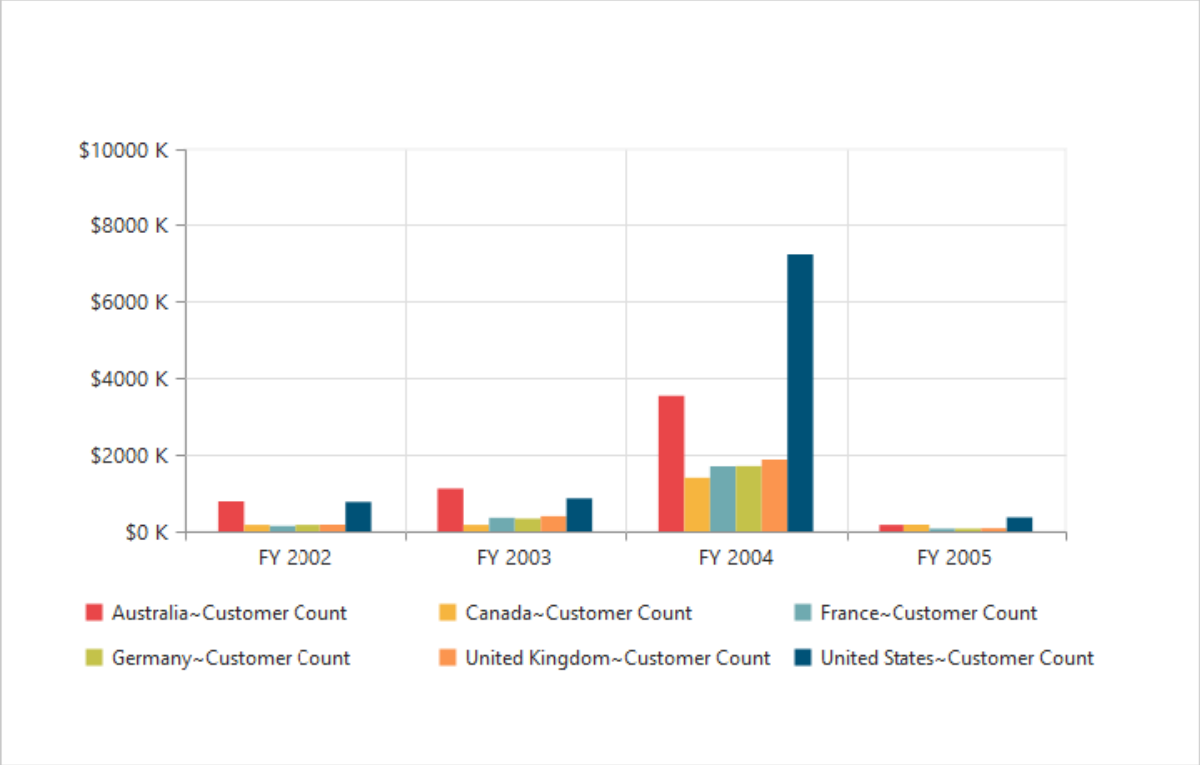
Label Format Customization

By using the `LabelFormat` property of `PrimaryYAxis`, you can add the category labels with prefix and/or suffix.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
//Adding prefix and suffix to axis labels
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryYAxis(primaryY =>
primaryY.LabelFormat("${value}K")).Url(Url.Content("/RelationalChartServi
ce.svc")).Size(size => size.Height("460px").Width("950px"))
```



Common Axis Features

Axis Visibility

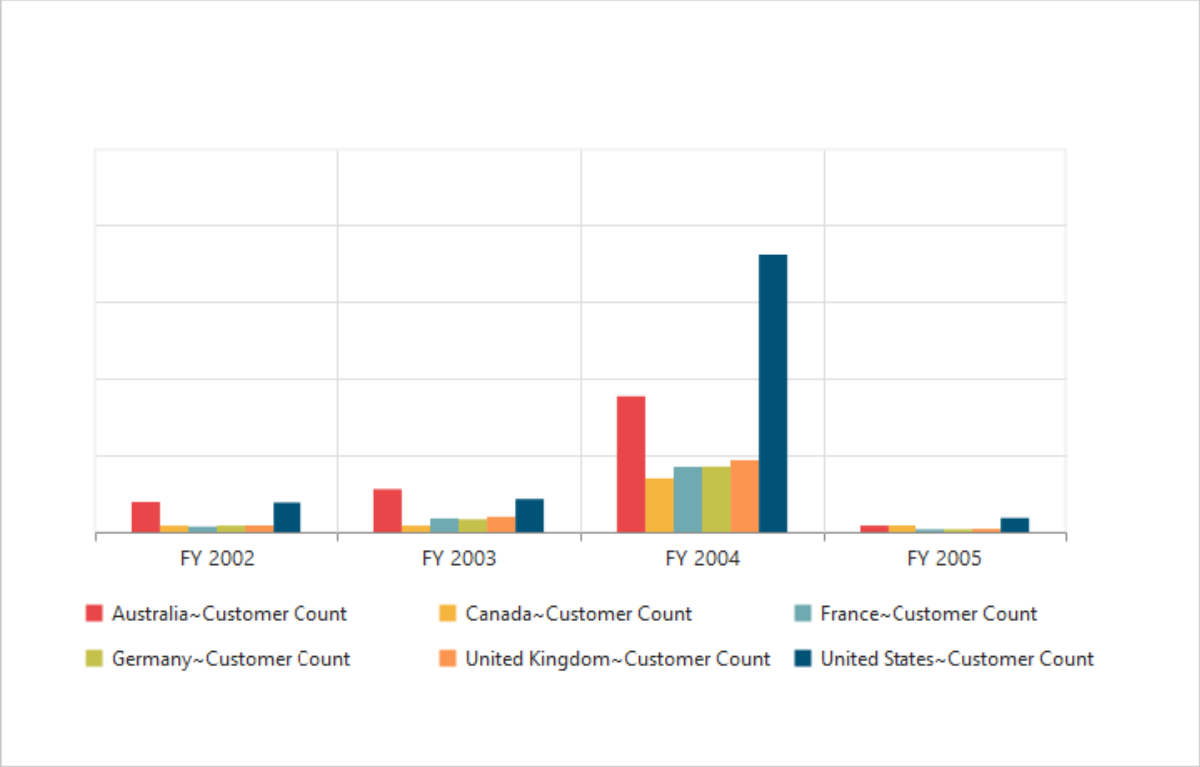
Axis visibility can be set by using the Visible property of the respective axis.

Note: By default, the value of Visible property is true in PivotChart.

ASPX-CS

```
//Disabling visibility of Y-axis
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryYAxis(primaryY =>
primaryY.Visible(false)).Url(Url.Content("/RelationalChartService.svc")).
Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



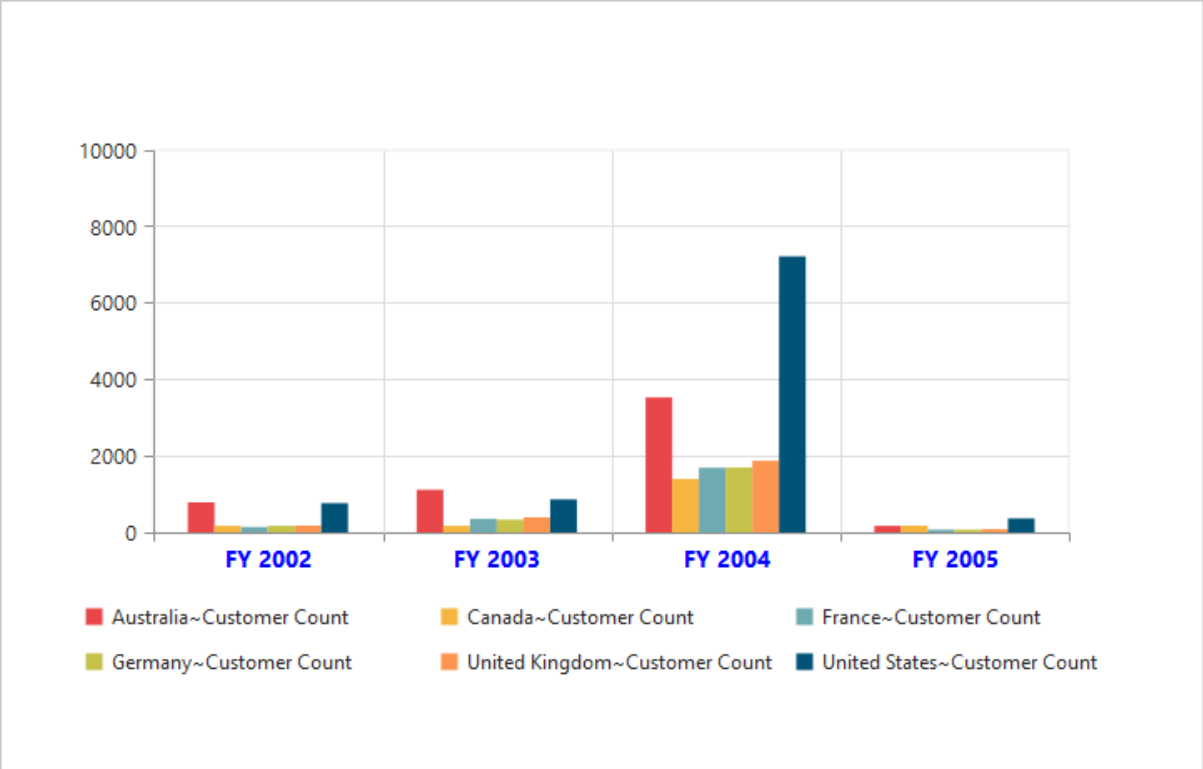
Label Customization

By using the Font property of the axis, we can customize the labels – font family, color, opacity, size and font-weight.

ASPX-CS

```
//Customizing label appearance
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.Font(font=>
font.Size("14px").FontWeight(ChartFontWeight.Bold).FontFamily("Segoe
UI").Color("blue"))).Url(Url.Content("/RelationalChartService.svc")).Size
(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



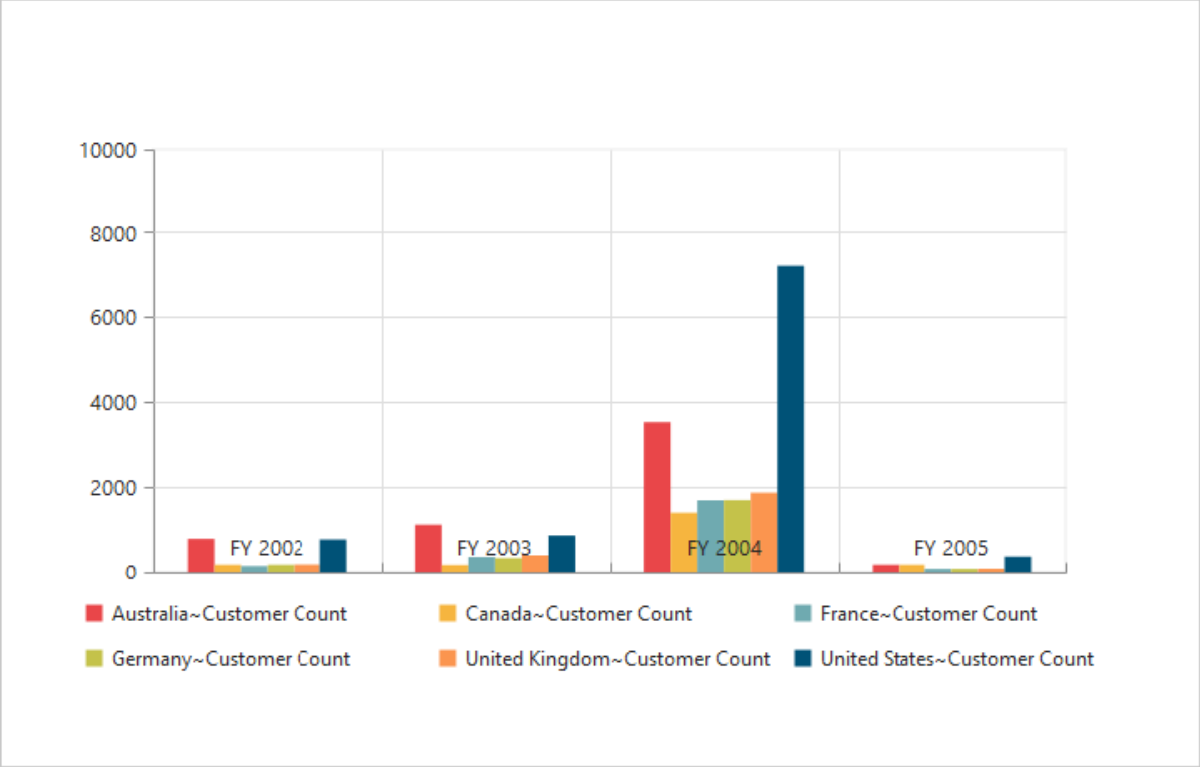
Label and Tick Positioning

Axis labels and ticks can be positioned inside or outside the Chart area by using the AxisLabelPosition and TickLinesPosition properties. The labels and ticks are positioned outside the Chart area, by default.

ASPX-CS

```
//Customizing label and tick positions
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.AxisLabelPosition(AxislabelPosition.Inside).TickLinesPosition(Ti
ckLinesPosition.Inside)).Url(Url.Content("/RelationalChartService.svc")).
Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Grid Lines Customization

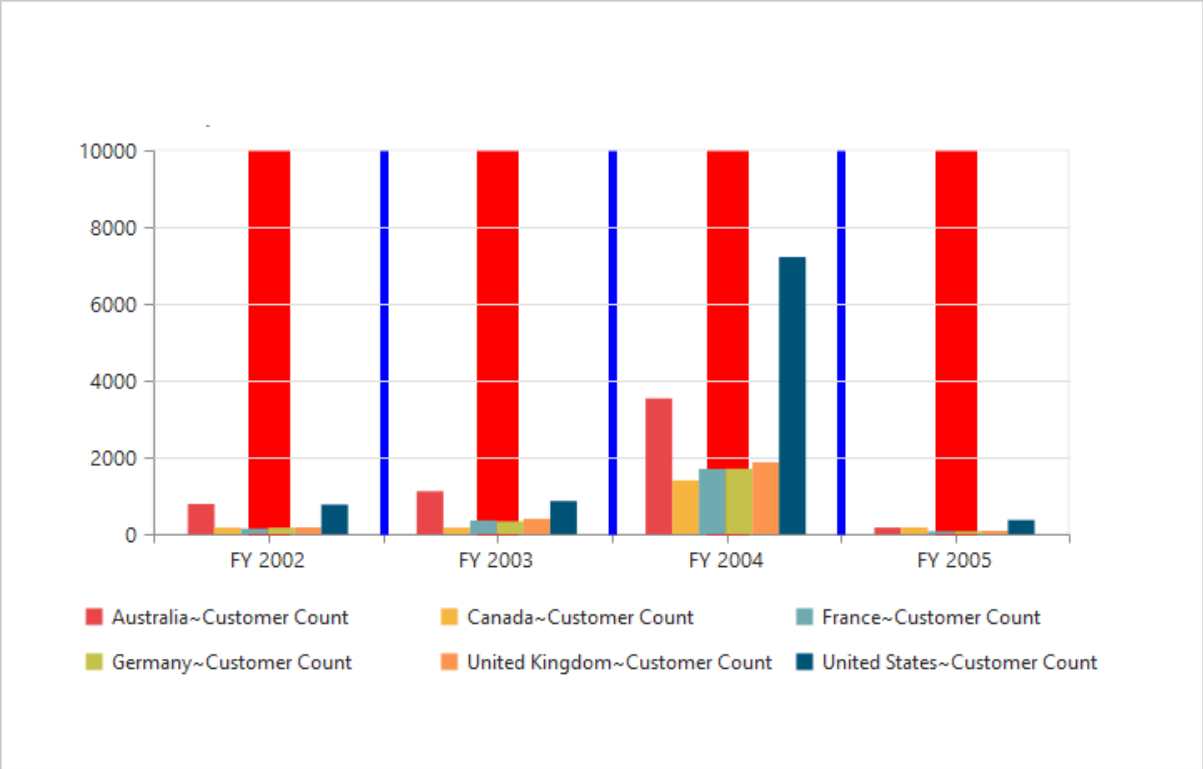
By using the MajorGridLines and MinorGridLines properties of the axis, you can customize the width, color, visibility and opacity of the grid lines.

Note: By default, the minor grid lines are not visible in PivotChart.

ASPX-CS

```
//Customizing grid lines
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.MajorGridLines(mr =>
mr.Width(5).Color("Blue").Visible(true)).MinorGridLines(mg =>
mg.Width(25).Color("Red").Visible(true)).MinorTicksPerInterval(1)).Url(Ur
l.Content("/RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tick Line Customization

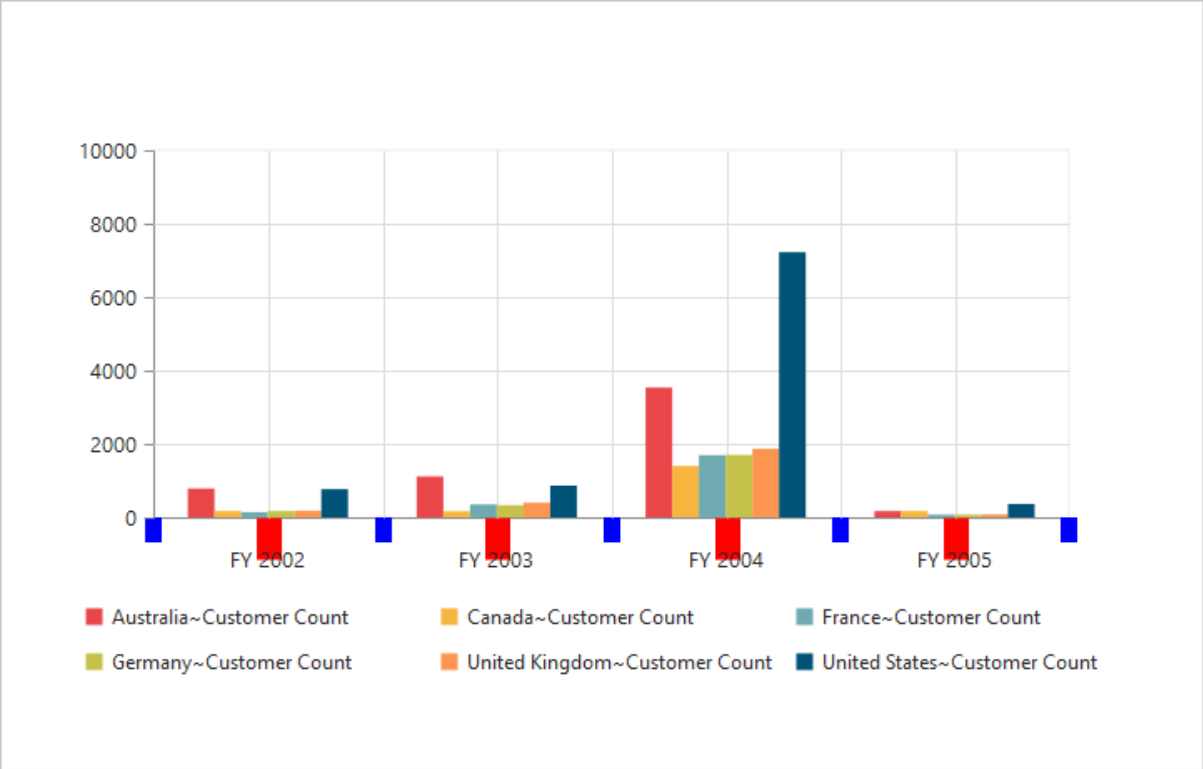
By using the MajorTickLines and MinorTickLines properties of the axis, you can customize the width, color, visibility, size and opacity of the tick lines.

Note: By default, the minor tick lines are not visible in PivotChart.

ASPX-CS

```
//Customizing tick lines
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.MajorTickLines(mr =>
mr.Width(10).Size(15).Color("Blue").Size(15).Visible(true)).MinorTickLine
s(mg =>
mg.Width(15).Size(25).Color("Red").Visible(true)).MinorTicksPerInterval(1
)).Url(Url.Content("/RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Inversing Axis

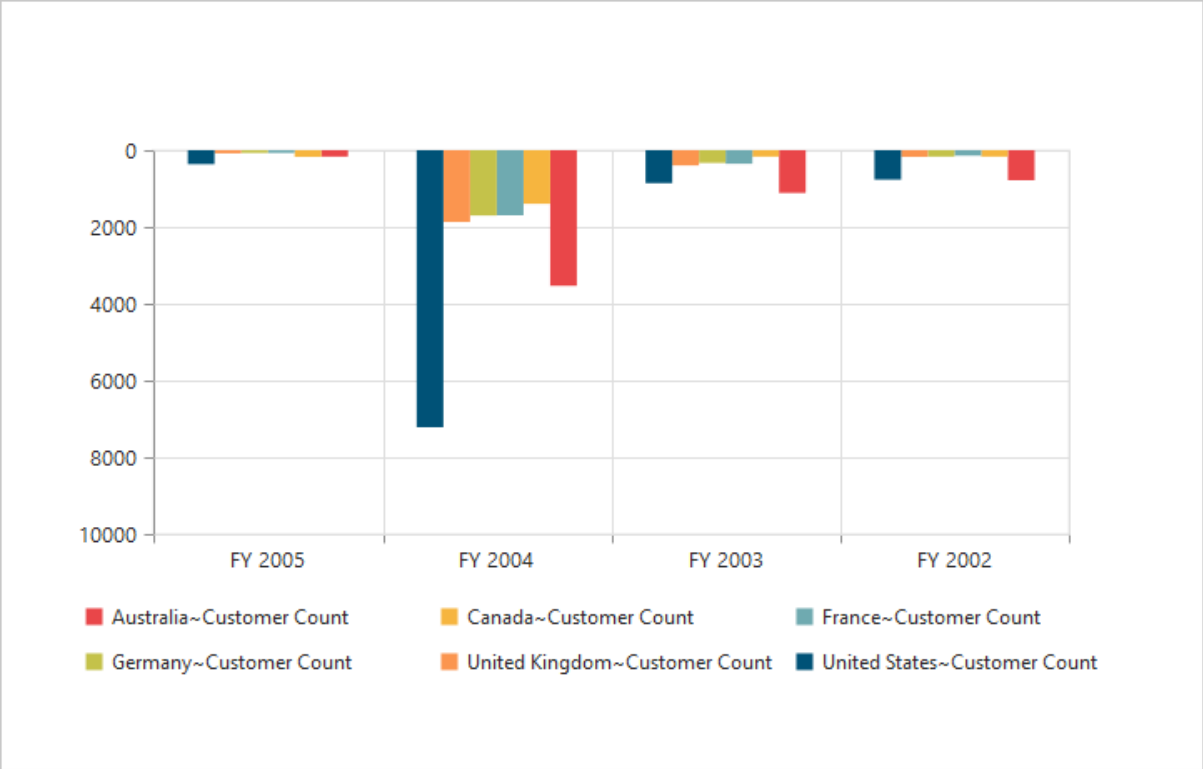
Axis can be inversed by using the `IsInversed` property of the axis.

Note: By default, the `IsInversed` property is false in PivotChart.

ASPX-CS

```
//Inversing the X and Y-axis
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.IsInversed(true)).PrimaryYAxis(primaryY =>
primaryY.IsInversed(true)).Url(Url.Content("/RelationalChartService.svc")
).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Placing Axes at Opposite Side

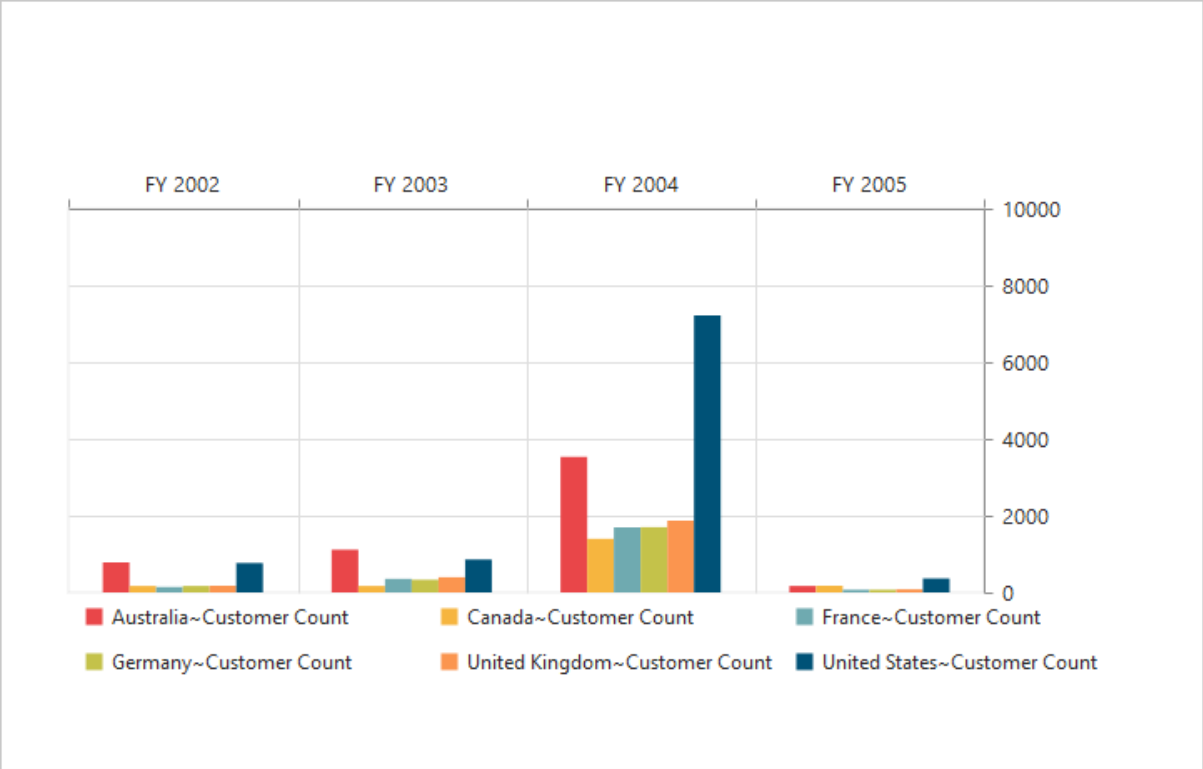
The OpposedPosition property of Chart axis can be used to place the axis at the opposite direction from its default position.

Note: By default, the OpposedPosition property is false in PivotChart.

ASPX-CS

```
//Placing axis at the opposite side of its normal position
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.OpposedPosition(true)).PrimaryYAxis(primaryY =>
primaryY.OpposedPosition(true)).Url(Url.Content("/RelationalChartService.
svc")).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Smart Axis Labels

When the axis labels overlap with each other based on the Chart dimensions and label size, you can use `LabelIntersection` property of the axis to avoid overlapping.

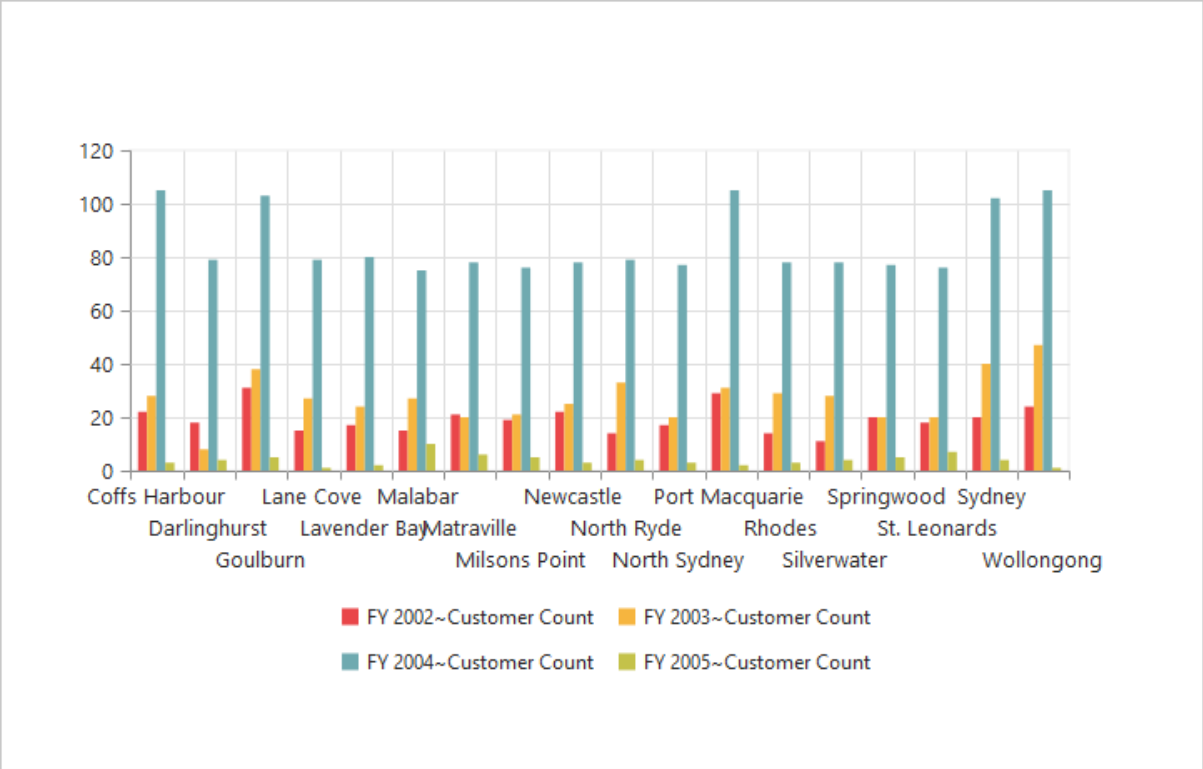
Note: By default, the `LabelIntersection` property is none in PivotChart.

- Rotate45
- Rotate90
- Trim
- MultipleRows
- Wrap
- Hide.

ASPX-CS

```
// Avoid overlapping of x-axis labels
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.
LabelIntersectAction(LabelIntersectAction.MultipleRows)).Url(Url.Content(
"/RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Title

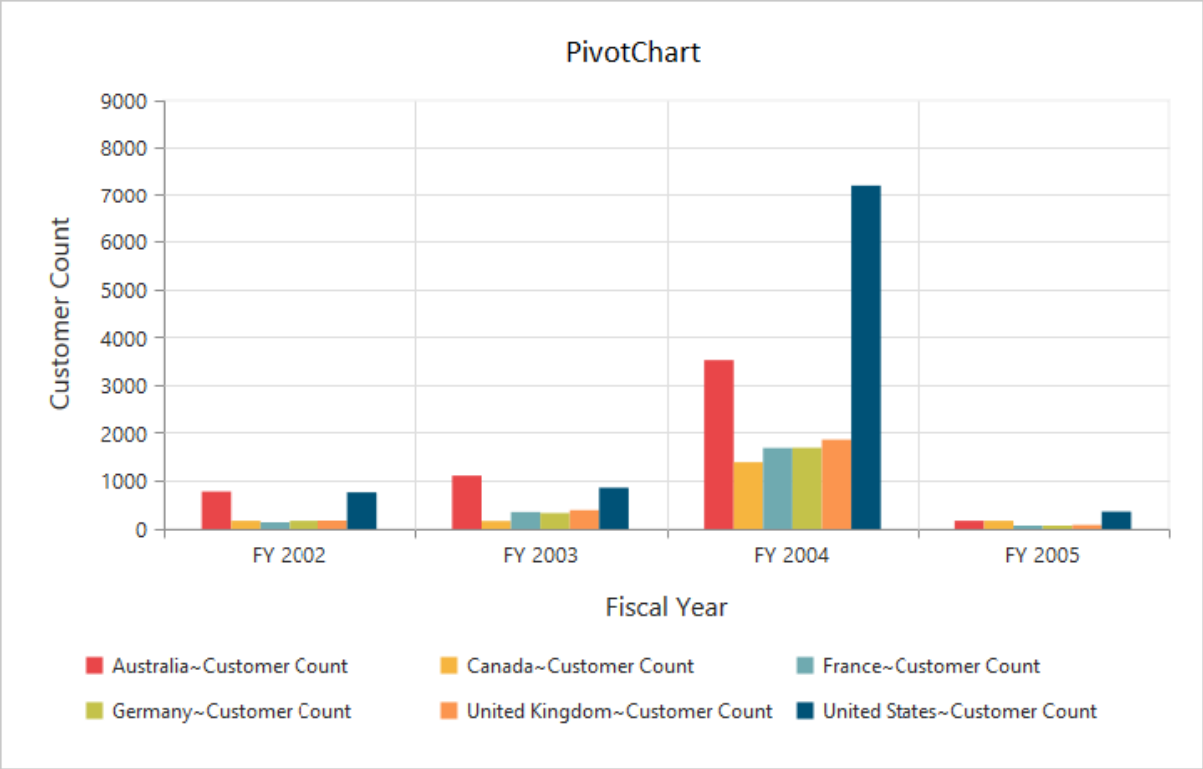
Title Text

By using the title.Text property, you can add the title text for PivotChart.

ASPX-CS

```
//Adding Chart title
@Html.EJ().Pivot().PivotChart("PivotChart1").Title(title =>
title.Text("PivotChart in
JS")).Url(Url.Content("/RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



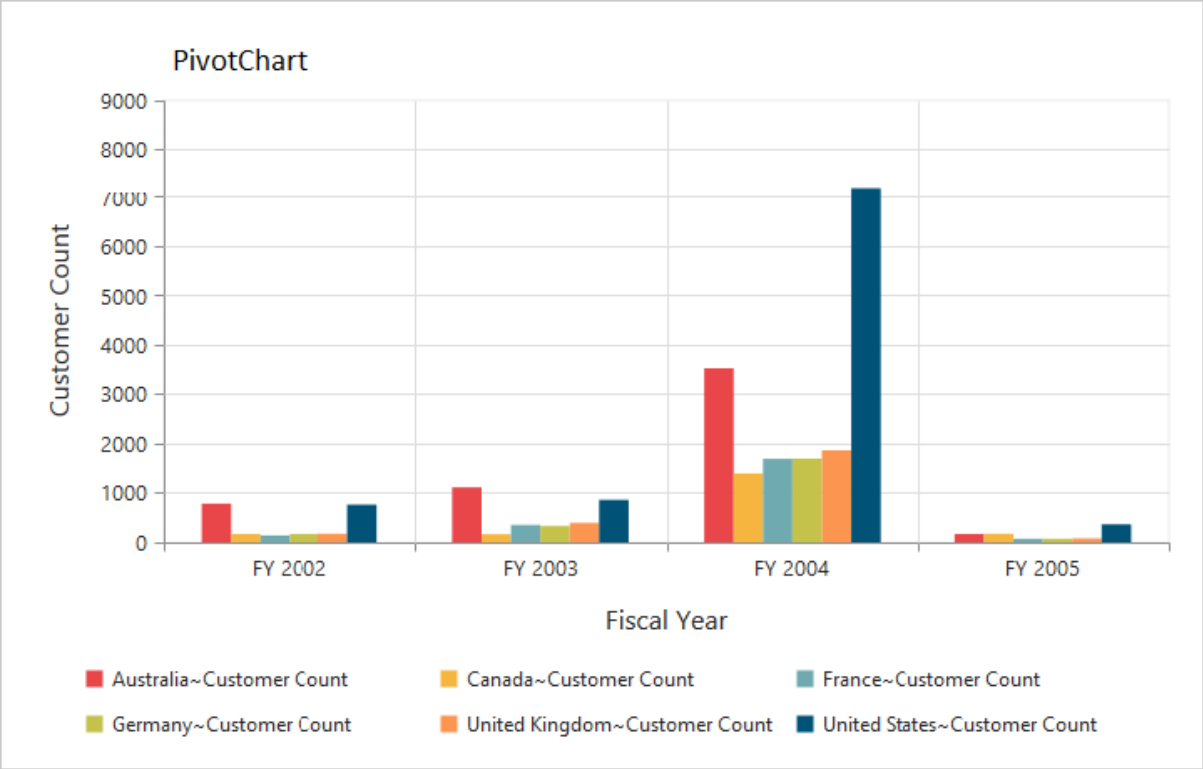
Title Alignment

By using the title.TextAlignment property, you can align the PivotChart controls title text to center, far or near.

ASPX-CS

```
//Change title text alignment
@Html.EJ().Pivot().PivotChart("PivotChart1").Title(title =>
title.Text("PivotChart in JS").TextAlignment(TextAlignment.Near)
).Url(Url.Content("/RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



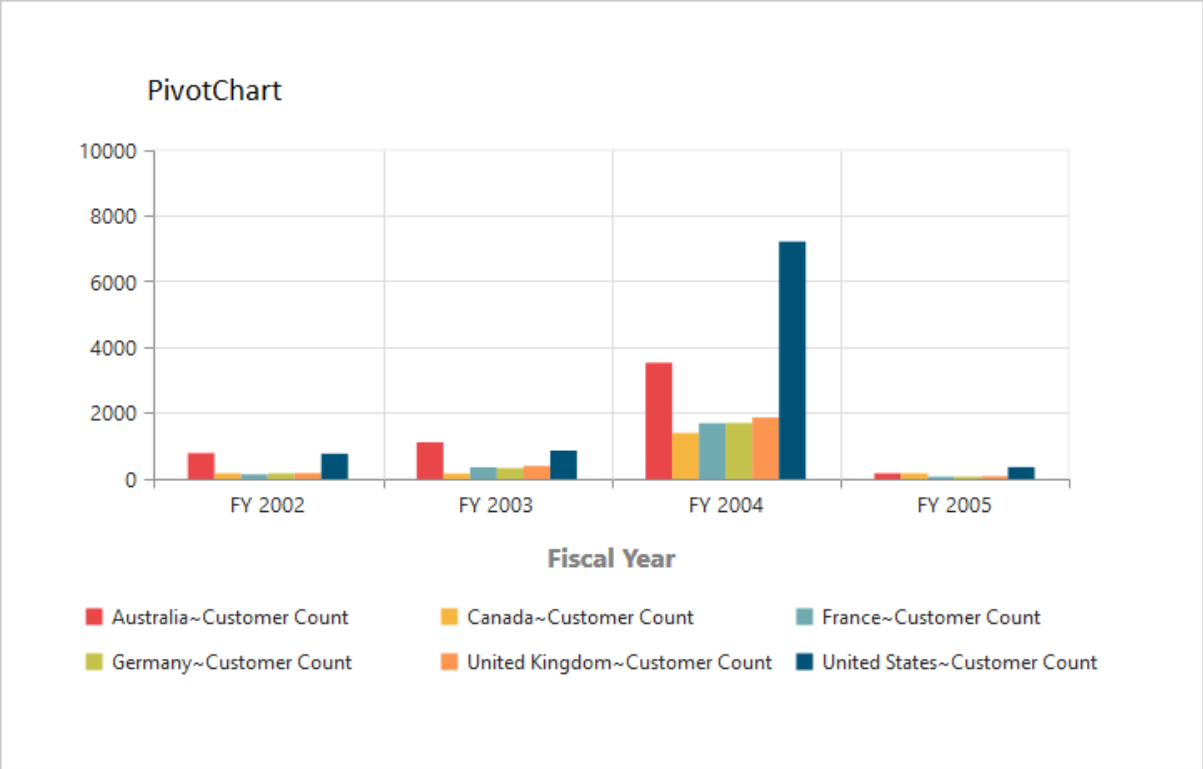
Title Customization

By using the Title property, you can add the title text for X-axis and Y-axis. Also title text can be customized by using the Text and Font properties. On setting EnableTrim to true, title text could be trimmed based on its length.

ASPX-CS

```
//Customizing axis title
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryXAxis(primaryX =>
primaryX.Title(title => title.Text("Fiscal Year").Font(font=>
font.Size("16px").FontFamily("Segoe
UI").FontWeight(ChartFontWeight.Bold).Color("grey")))).Url(Url.Content("/
RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Series

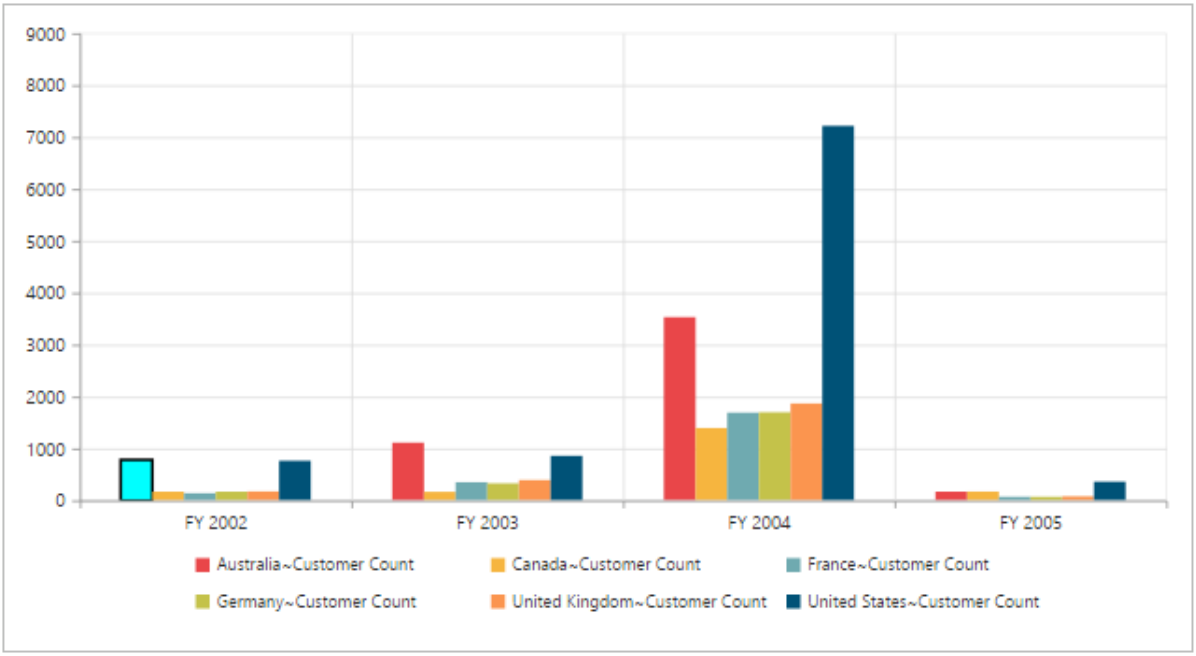
Series Point customization

By using the fill and border properties of Chart series, you can customize the PivotChart series color, border color and border width.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
this.model.series[0].points[0].fill = "aqua";
this.model.series[0].points[0].border = {
color: "black",
width: 2
};
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Axes

You can split the pivot chart and its area into multiple panes to draw multiple series with multiple axes. Additional vertical/horizontal axes in the pivot chart can be created by the axes property.

Before rendering multiple axes in the pivot chart, you should know some important properties of axes and series of the pivot chart.

Properties of axes

- **rowIndex**: Specifies the index of the row where the axis is associated.
- **columnIndex**: Specifies the index of the column where the axis is associated.
- **opposedPosition**: Specifies whether to render the axis at the opposite side of its default position.
- **labelFormat**: Provides custom formatting for axis label and supports all standard formatting types of numerical and date time values.
- **title**: You can customize the title of the axis by the following properties.
- **enableTrim**: Specifies whether to trim the axis title when it exceeds the chart area or the maximum width of the title.
- **maximumTitleWidth**: When the title exceeds the maximum width, the title gets trimmed by setting enableTrim to true.
- **visible**: Controls the visibility of the axis title.
- **name**: Unique name of the axis. To associate an axis with the series, you have to set this name to the xAxisName/yAxisName property of the series.

To know more about axes properties, Click Here

Properties of series

- **xAxisName**: Specifies the name of the X-axis that has to be associated with this series. Add an axis instance with this name to axes collection.

Copyright © 2001 - 2020 Syncfusion Inc.

- **yAxisName**: Specifies the name of the Y-axis that has to be associated with this series. Add an axis instance with this name to axes collection.

Customizing axes at row index of zero

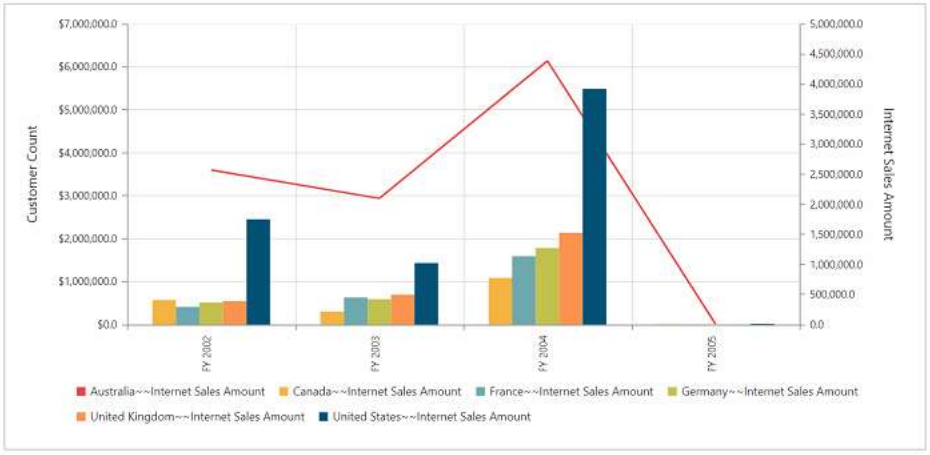
You can customize the axes in the primary pivot chart itself. To achieve this, you should give the following properties:

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.RowIndex(0).Name("yAxisConfig").Add();
}).ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad").BeforeSeriesRender("onBeforeRender"))
//....
```

JAVASCRIPT

```
<script>
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].yAxisName = "yAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
</script>
```



Customizing axes at row index one

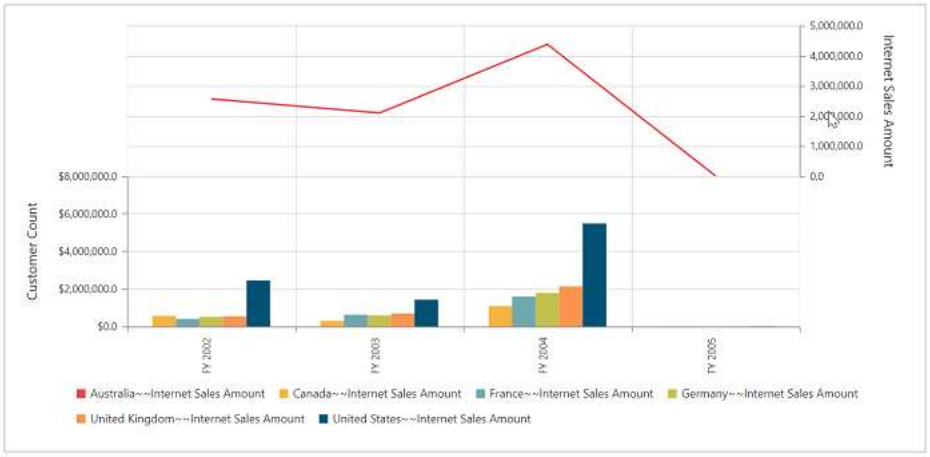
HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.RowIndex(1).Name("yAxisConfig").Add();
}).ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad").BeforeSeriesRender("onBeforeRender"))
//....
```

Copyright © 2001 - 2020 Syncfusion Inc.

JAVASCRIPT

```
<script>
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].yAxisName = "yAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
</script>
```



Customizing axes at column index of zero

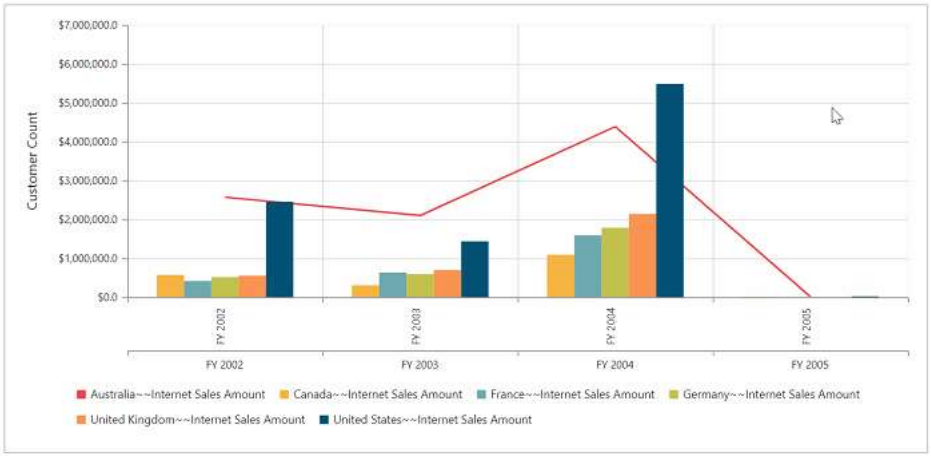
HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.ColumnIndex(0).Name("xAxisConfig").Add();
}).ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad").BeforeSeriesRender("onBeforeRender"))
//....
```

JAVASCRIPT

```
<script>
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].xAxisName = "xAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



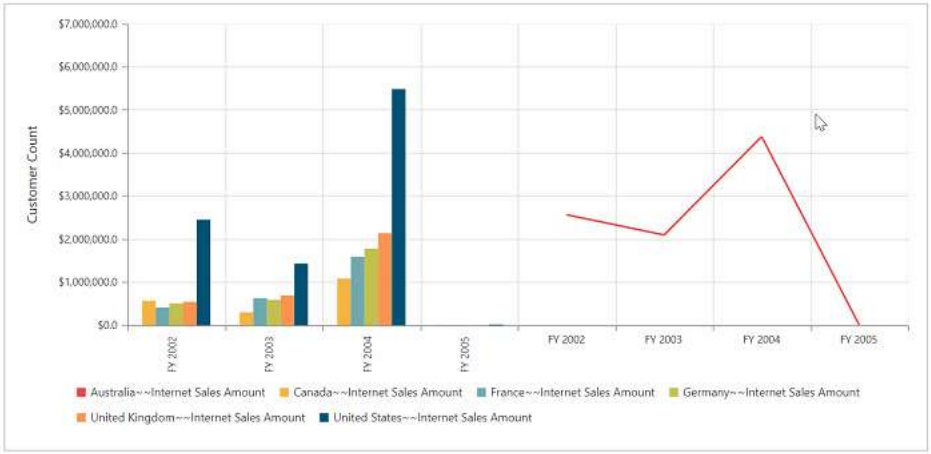
Customizing axes at column index of one

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.ColumnIndex(1).Name("xAxisConfig").Add();
}).ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad").BeforeSeriesRender("onBeforeRender"))
//....
```

JAVASCRIPT

```
<script>
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].xAxisName = "xAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
</script>
```



Copyright © 2001 - 2020 Syncfusion Inc.

Customizing series

You can customize the series in multiple axes support with the help of **beforeSeriesRender** event. You can change the series type through the **onBeforeRender** event.

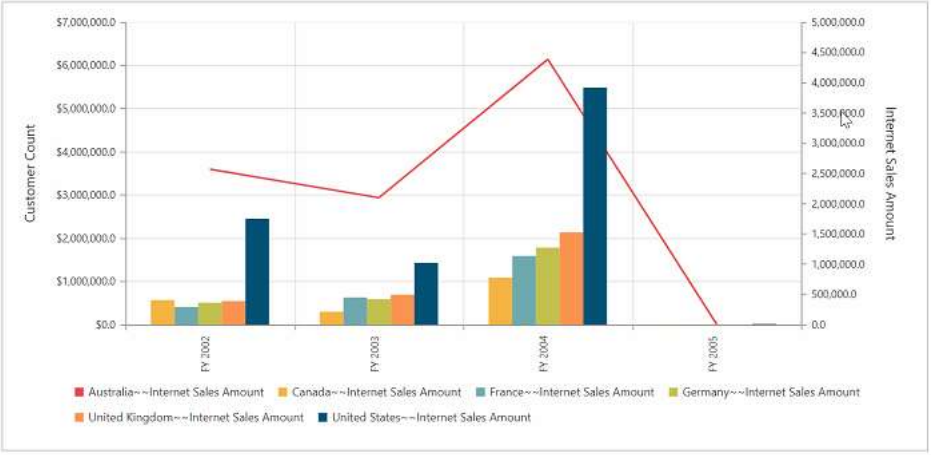
HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.Name("yAxisConfig").Add(); }).ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad").BeforeSeriesRender("onBeforeRender"))
//....
```

JAVASCRIPT

```
<script>
function onBeforeRender(args) {
for (var i = 0; i < args.series.length; i++) {
if (args.series[i].name.indexOf("Australia") != -1) {
args.series[i].yAxisName = "yAxisConfig";
args.series[i].type = "line";
}
}
return args;
}
</script>
```

Note: You have to use the same name in both name property of axes and xAxisName/yAxisName property of series in the above **beforeSeriesRender** event.



To learn more about series properties, click here.

Multiple axes support by series index

You can render the pivot chart with multiple axes by series index.

Note: This is the default behavior of multiple axes support, if you are not triggering the beforeSeriesRender event.

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.Name("y:0").Add();ax.Name("x:1").Add() })
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//....
```

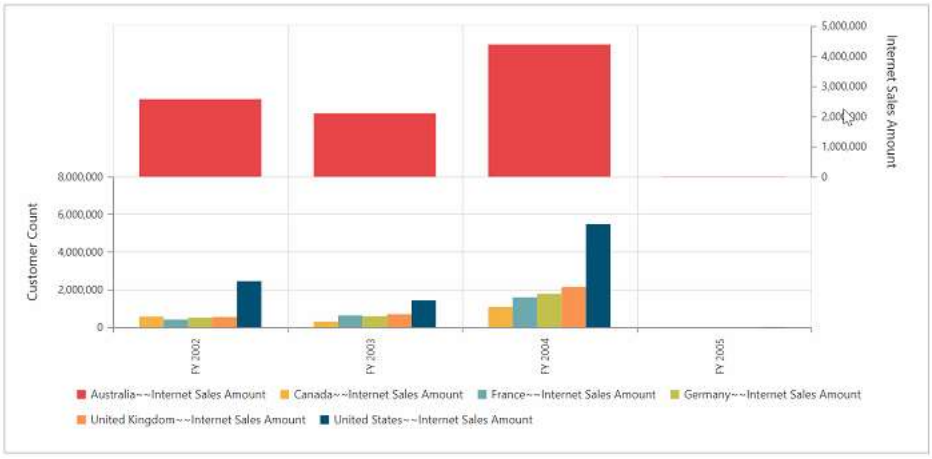
JAVASCRIPT

```
Name("y:0") => //you can create separate y axes for the series which you
want
//y indicates(yAxisName) axis in which series needs to be added.
//0 indicates the series index of pivot chart.
//you can also use pass multiple series index separated by comma(y:0,2)
Name("x:1") =>  //you can create separate x axes for the series which you
want
// x indicates(xAxisName) axis in which series needs to be added.
// 0 indicates the series index of pivot chart.
// you can also use pass multiple series index separated by comma(x:0,2)
```

For Y-axes

HTML

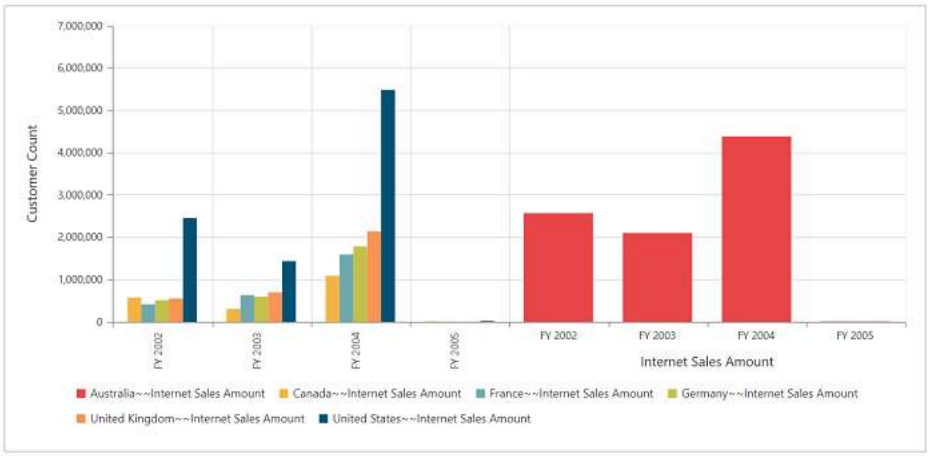
```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.Name("y:0").Add(); })
//....
```



For X-axes

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.Name("x:0").Add(); })
//....
```



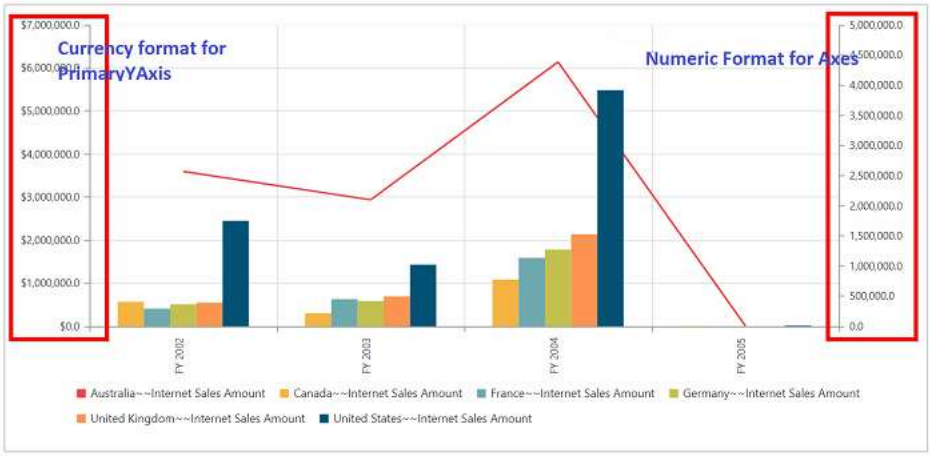
Customizing PrimaryYAxis and axes properties

labelFormat

You can customize the **labelFormat** for both PrimaryYAxis and custom axes.

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.LabelFormat("n1").Add(); }).PrimaryYAxis(primary => {
primary.LabelFormat("c1").Add(); })
//....
```

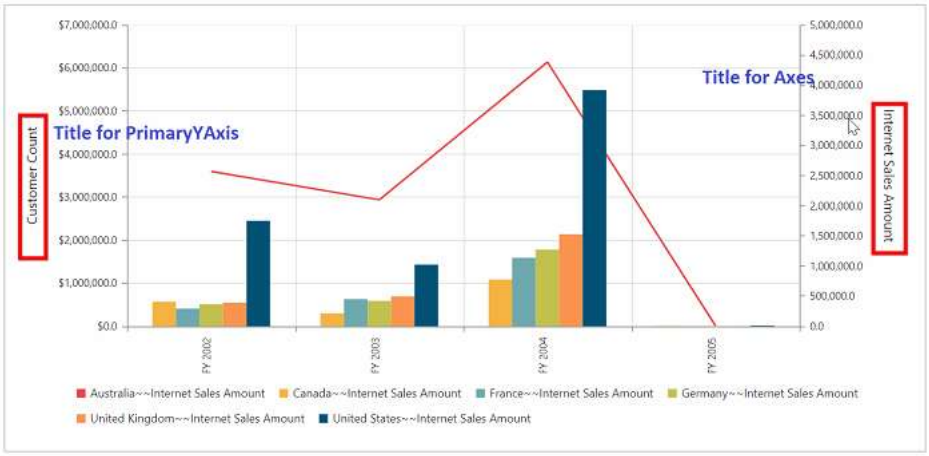


title

You can customize the title for axes by the **title** property.

HTML

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Axes(ax => {
ax.Title(title=>title.Text("Internet Sales Amount")).Add();
}).PrimaryYAxis(primary => { primary.Title(title=>title.Text("Customer
Count")).Add(); })
//....
```



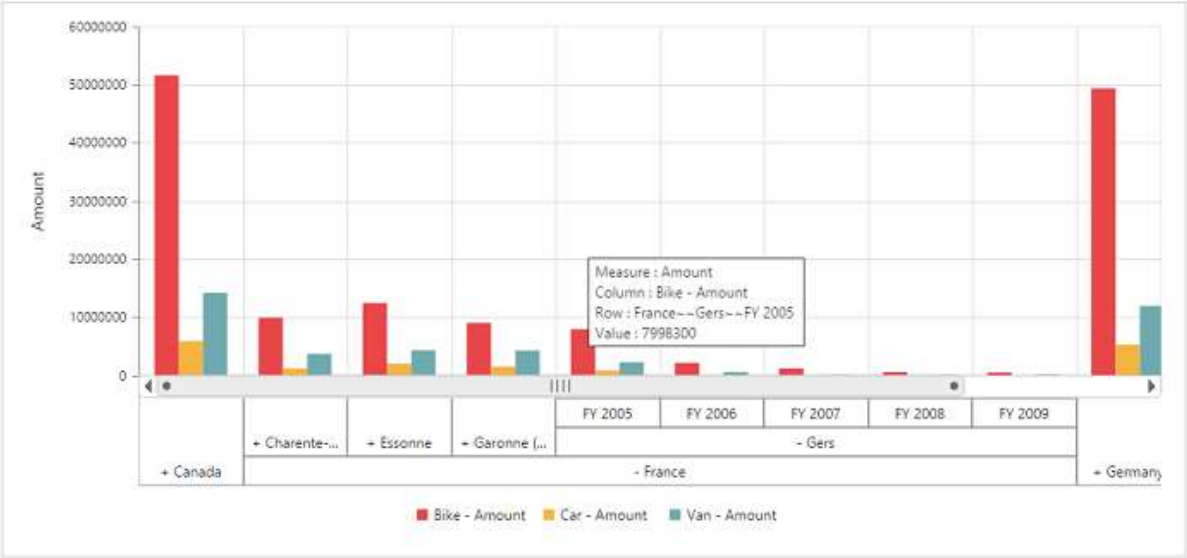
Multi-level Labels

Multi-level labels allows you to drill down to access the detailed level of data or drill up to see the summarized data by using the expander present in the OlapChart.

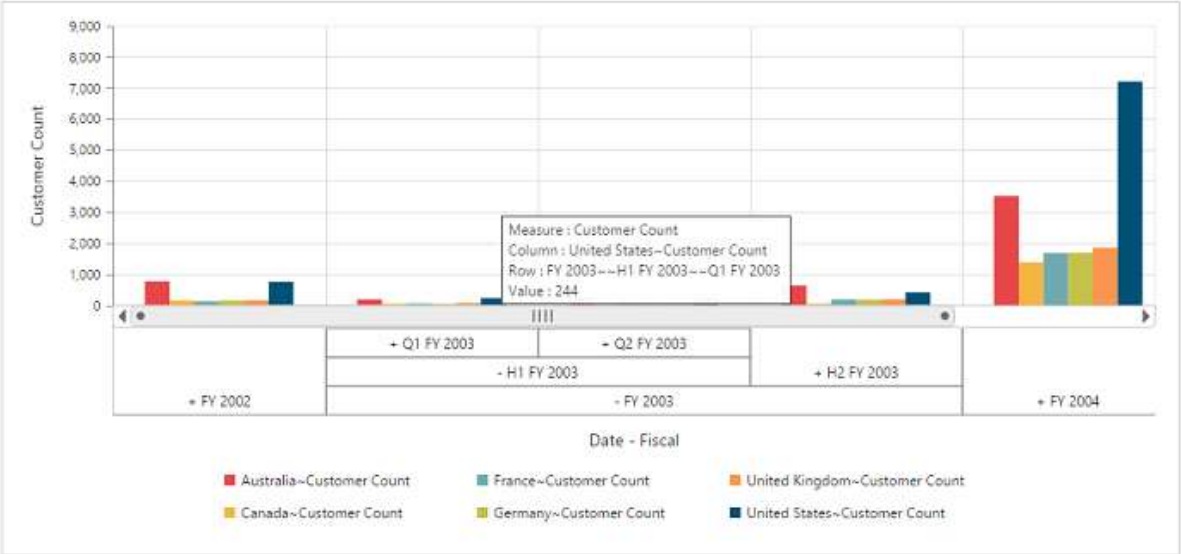
ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").EnableMultiLevelLabels(true)
```

Relational



OLAP



3D Visualization

The PivotChart control allows you to view the data in a 3D view. Following are the chart types that are supported:

- Bar
- Column
- Stacking Bar
- Stocking Column
- Pie

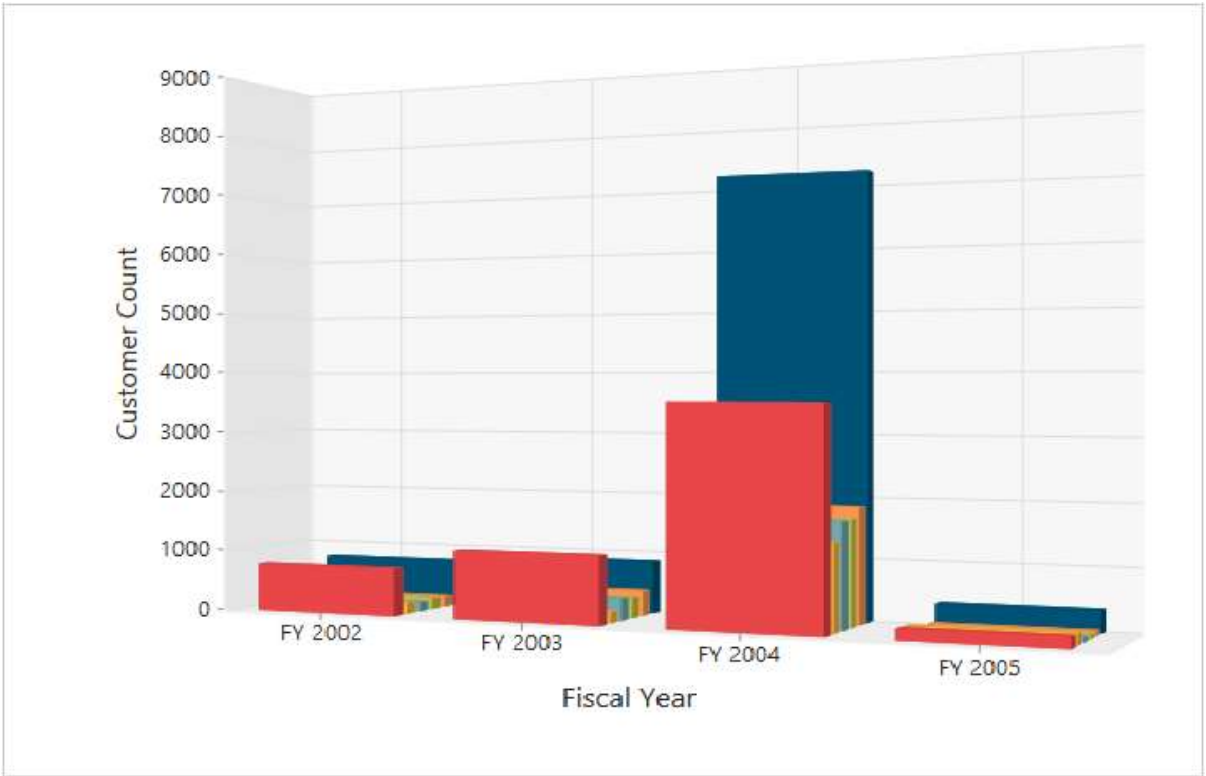
3D Column Chart

3D Column Chart is rendered by specifying the chart type as **“Column”** in the **“CommonSeriesOptions”** enumeration property as well as by setting the Enable3D property to **“true”.**

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Rotation(24).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Column); }).Size(size =>
size.Height("450px").Width("100%")).Enable3D(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



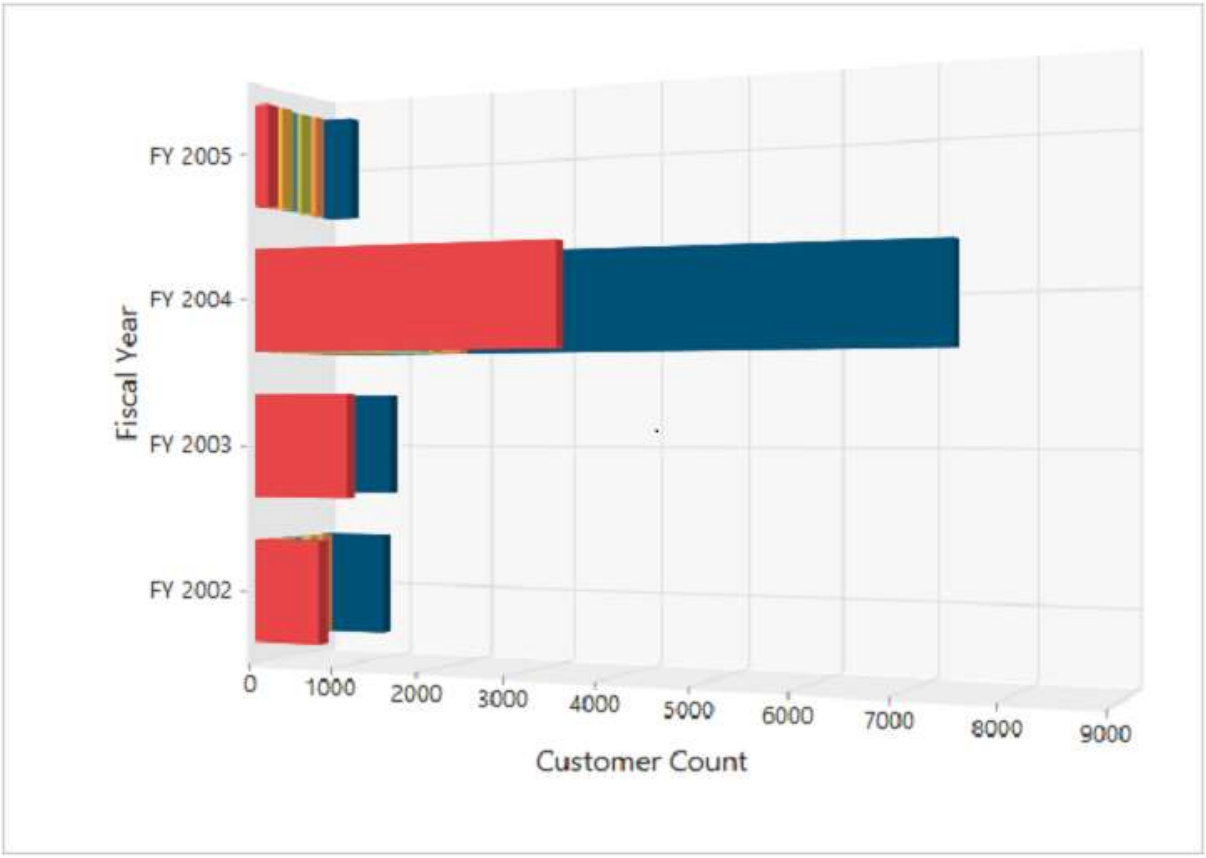
3D Bar Chart

3D Bar Chart is rendered by specifying the chart type as **"Bar"** in the **"CommonSeriesOptions"** enumeration property as well as by setting the Enable3D property to **"true".**

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Rotation(24).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Bar); }).Size(size =>
size.Height("450px").Width("100%")).Enable3D(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



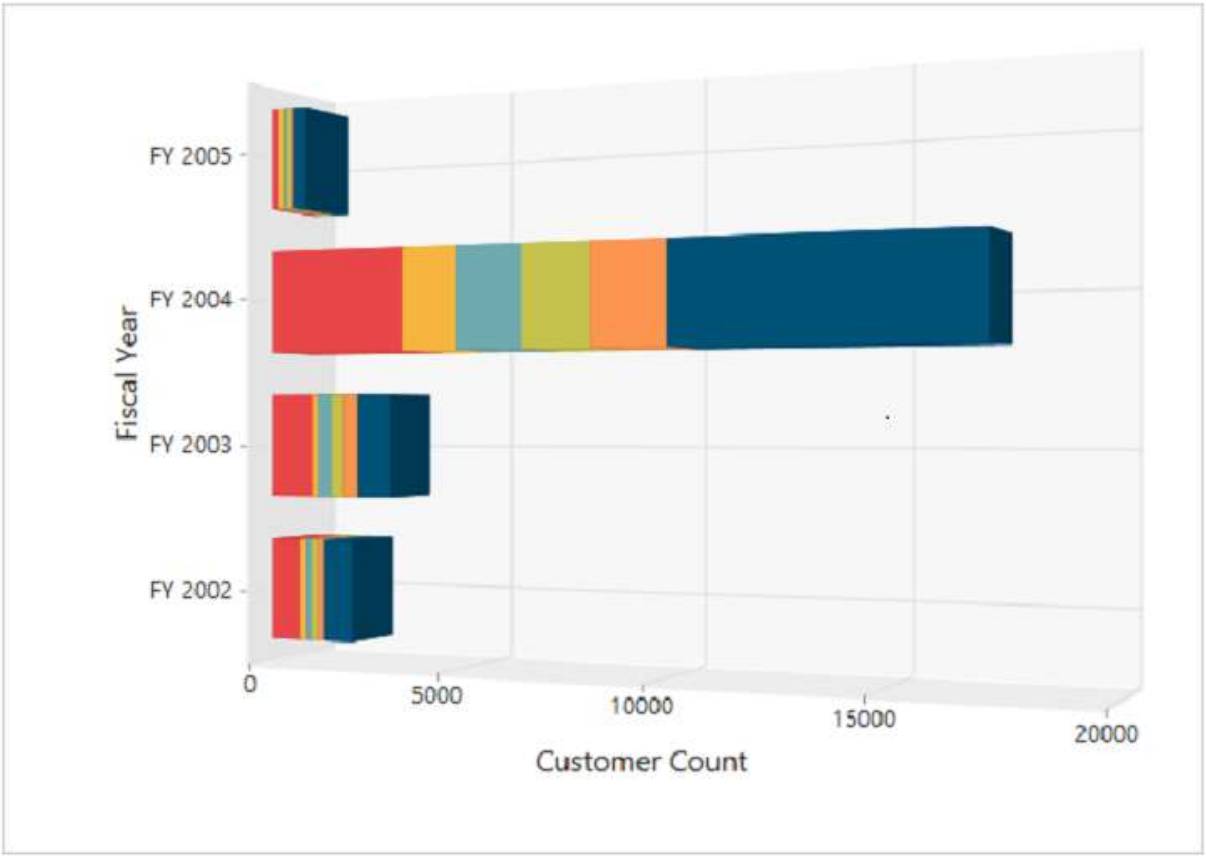
3D Stacking Bar Chart

3D Stacking Bar Chart is rendered by specifying the chart type as **"Stacking Bar"** in the **"CommonSeriesOptions"** enumeration property as well as by setting the Enable3D property to **"true".**

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Rotation(24).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StackingBar); }).Size(size =>
size.Height("450px").Width("100%")).Enable3D(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



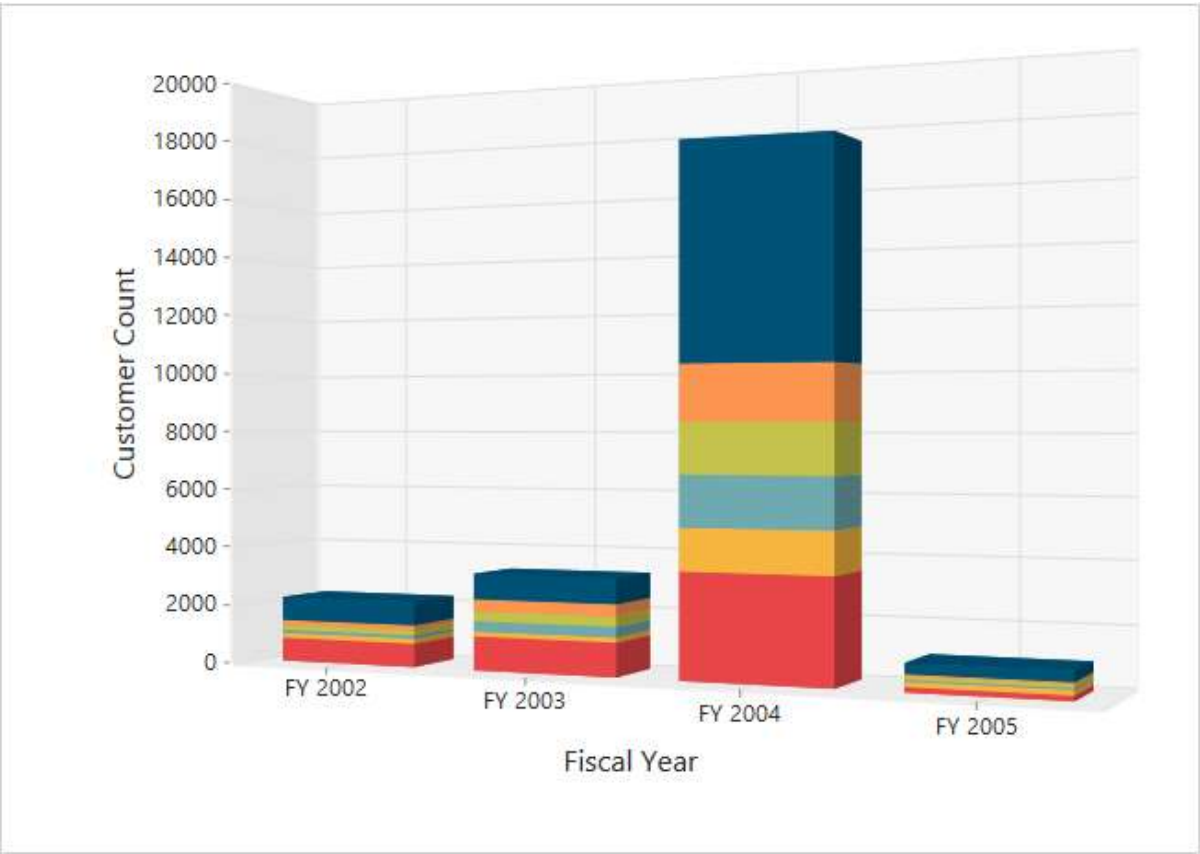
3D Stacking Column Chart

3D Stacking Column Chart is rendered by specifying the chart type as **"Stacking Column"** in the **"CommonSeriesOptions"** enumeration property as well as by setting the Enable3D property to **"true".**

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Rotation(24).CommonSeriesOptions(comm => {
comm.Type(SeriesType.StackingColumn); }).Size(size =>
size.Height("450px").Width("100%")).Enable3D(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



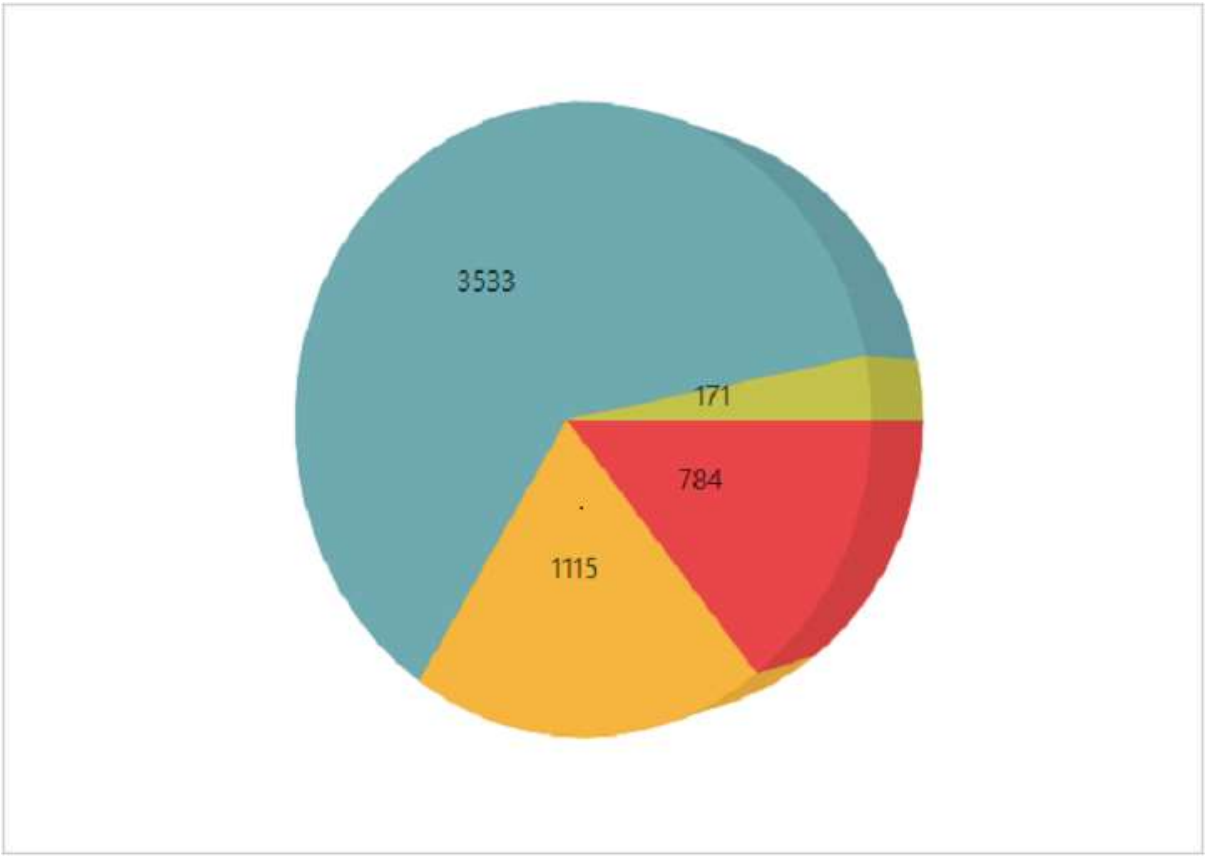
3D Pie Chart

3D Pie Chart is rendered by specifying the chart type as **“Pie”** in the **“CommonSeriesOptions”** enumeration property as well as by setting the Enable3D property to **“true”.**

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Rotation(24).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Pie); }).Size(size =>
size.Height("450px").Width("100%")).Enable3D(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



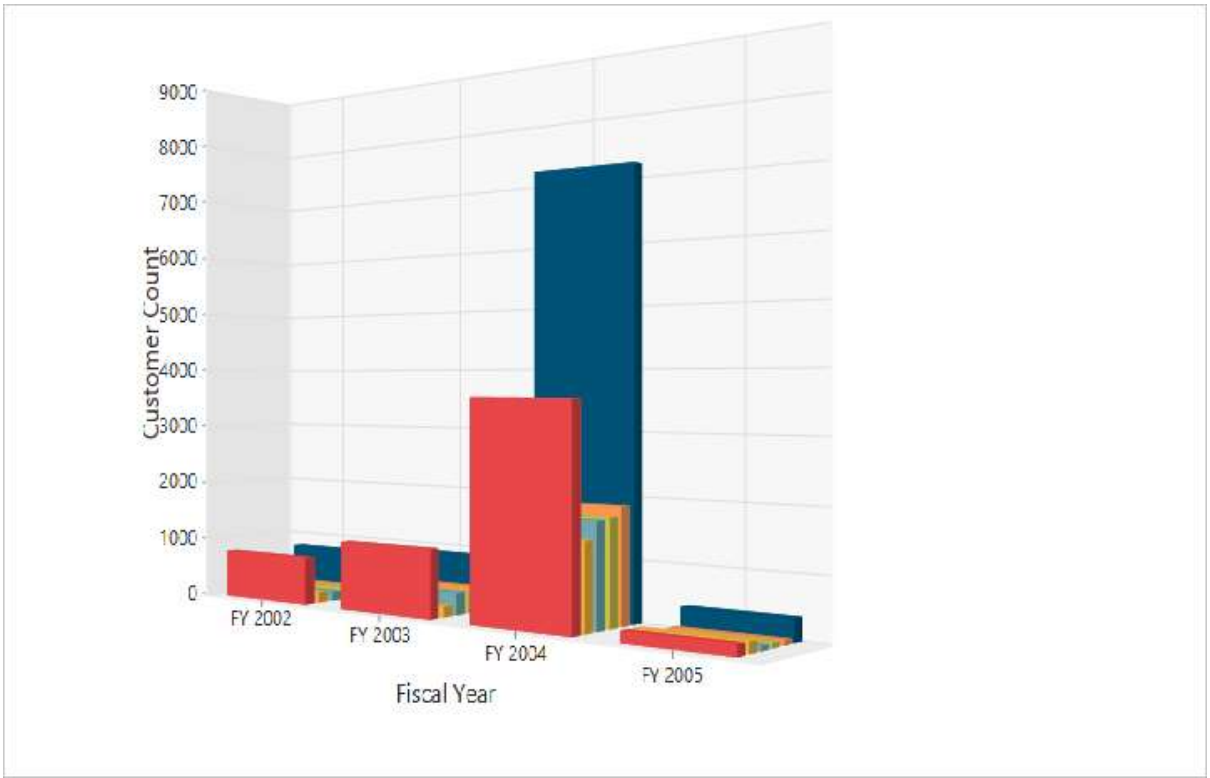
Rotating 3D Chart

We can rotate the 3D Chart towards left or right by setting an appropriate angle value to the Rotation property. The direction of the Chart display depends upon the positive or negative angle value.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Rotation(40).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Column); }).Size(size =>
size.Height("450px").Width("100%")).Enable3D(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



User Interactions

Tooltip

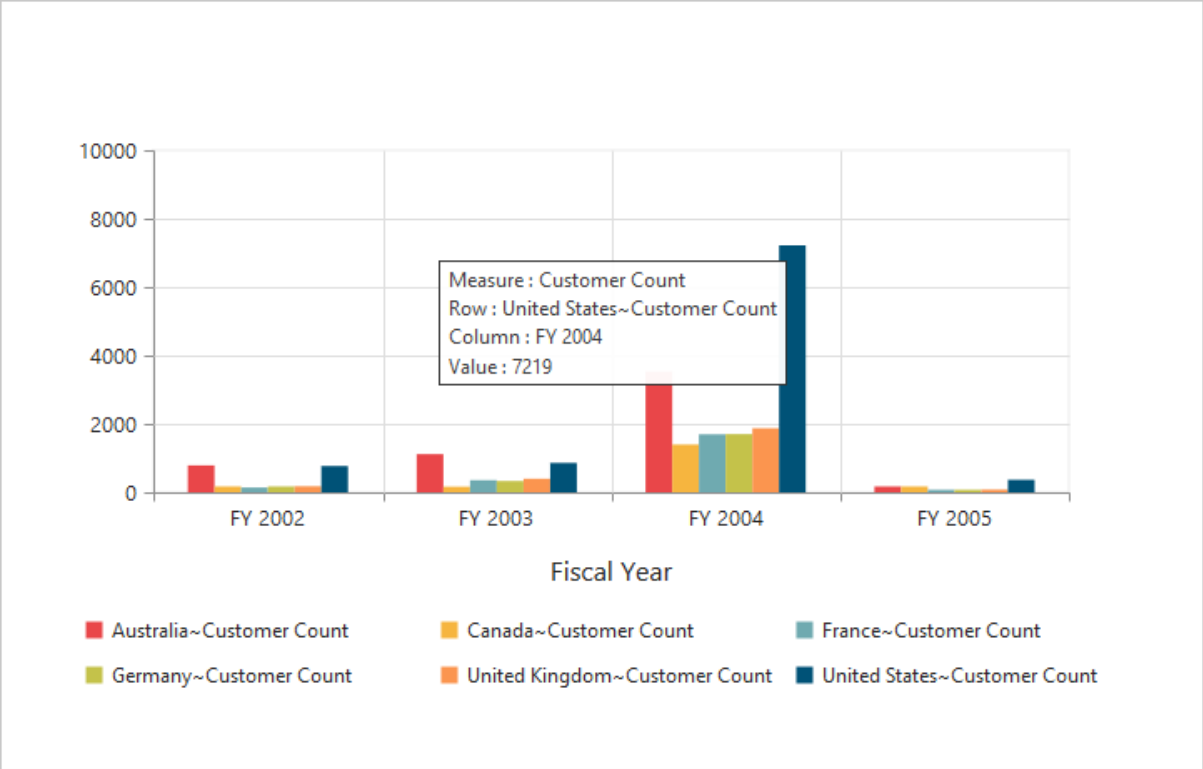
Enable Tooltip for Data Points

Tooltip for the data points can be enabled using the **“Visible”** option of the Tooltip property under **“CommonSeriesOptions”** of the PivotChart.

ASPX-CS

```
//Enabling tooltip of data point
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => { comm.Tooltip(tool => {
tool.Visible(true); })}).Size(size => size.Height("460px").Width("100%"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



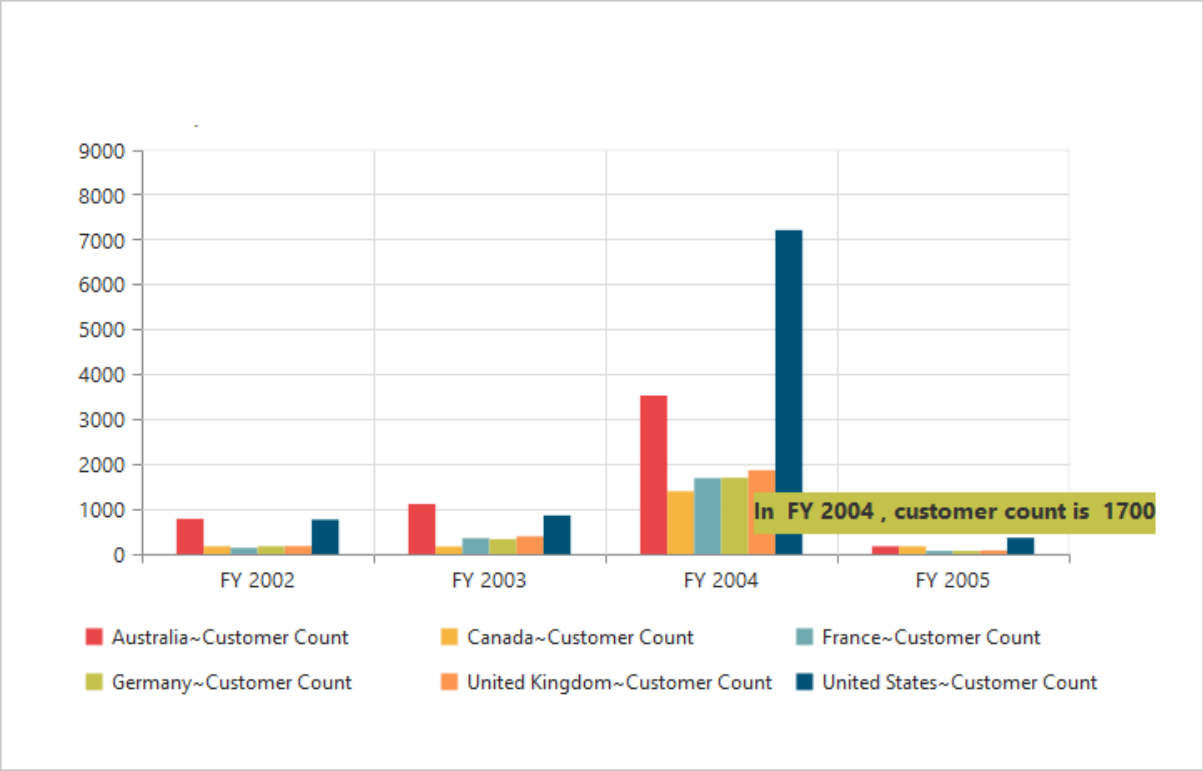
Tooltip Template

HTML elements can be displayed inside the tooltip by using the Template option. The template option takes the value of the “id” attribute from the HTML element. You can use the **#point.x#** and **#point.y#** as place holders in the HTML element to display the X and Y values of the corresponding data points.

ASPX-CS

```
<div id="Tooltip" style="display: none;">
<div id="icon"> <div id="ccicon"> </div>  </div>
<div id="value">
<div>
<label id="ccvalue"> #point.y# </label>
<label id="cc">Customer Count </label>
</div>
</div>
</div>
//Enabling tooltip of data point
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => { comm.Tooltip(tool => {
tool.Visible(true).Template("Tooltip"); });}).Size(size =>
size.Height("460px").Width("100%"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

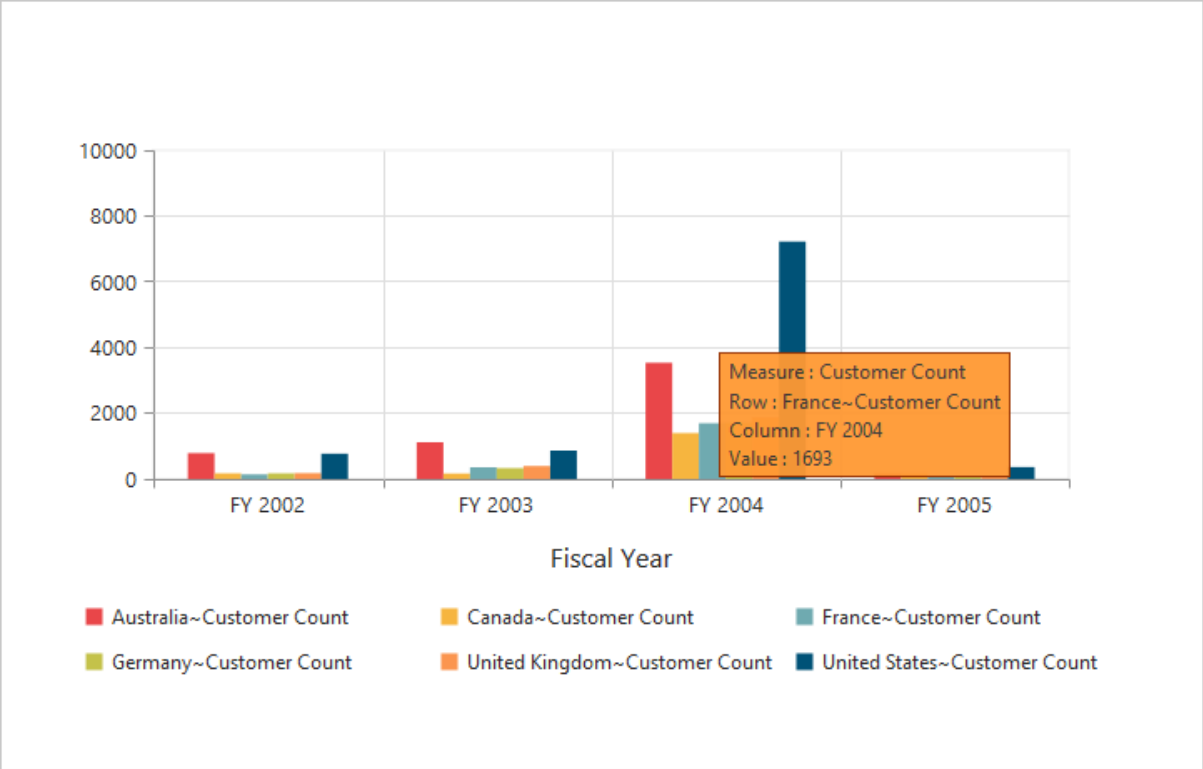


Tooltip Customization

By using Fill and Border properties of tooltip, you can customize its background color, border color and border width.

ASPX-CS

```
//Change tooltip color and border
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => { comm.Tooltip(tool => {
tool.Visible(true). Fill("#FF9933").Border(border =>
border.Width(1).Color("#993300")); });}).Size(size =>
size.Height("460px").Width("100%"))
```



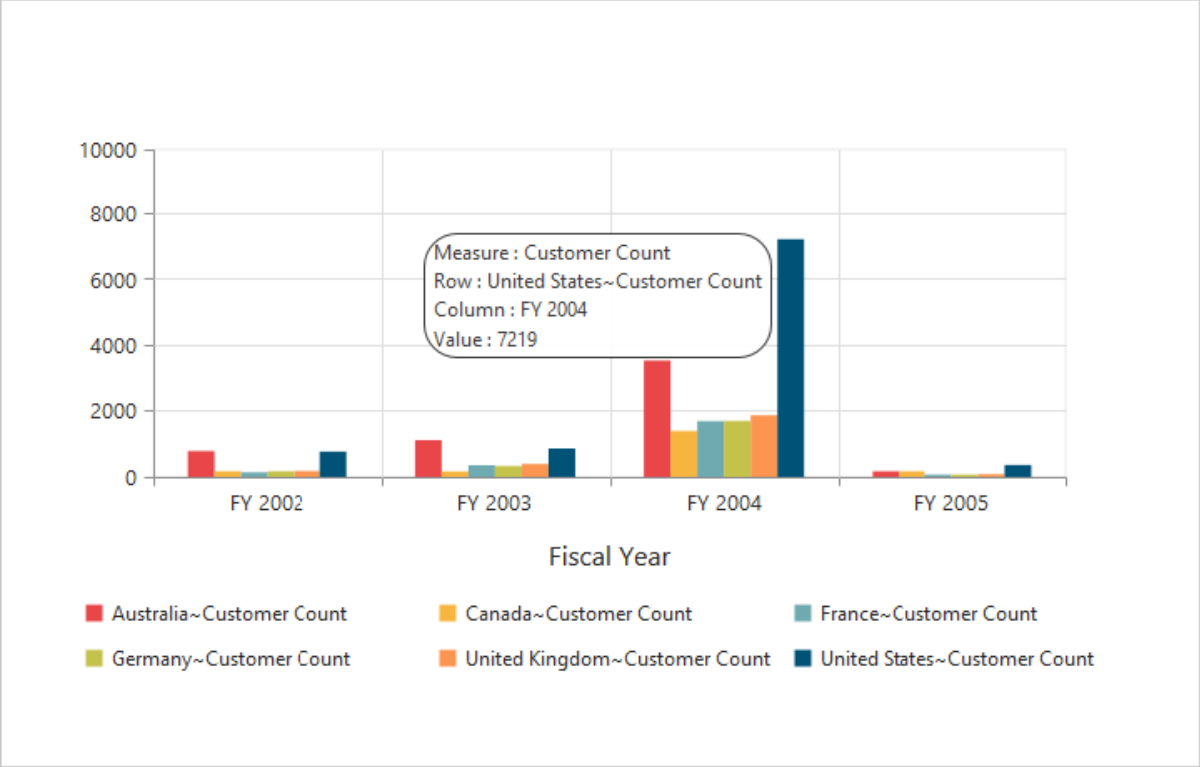
Tooltip with Rounded Corners

The tooltip properties, RX and RY are used to customize its corner radius.

ASPX-CS

```
//Customize the corner radius of the tooltip rectangle.
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => { comm.Tooltip(tool => {
tool.Visible(true).RX(50).RY(50);});}).Size(size =>
size.Height("460px").Width("100%"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Zooming and Panning

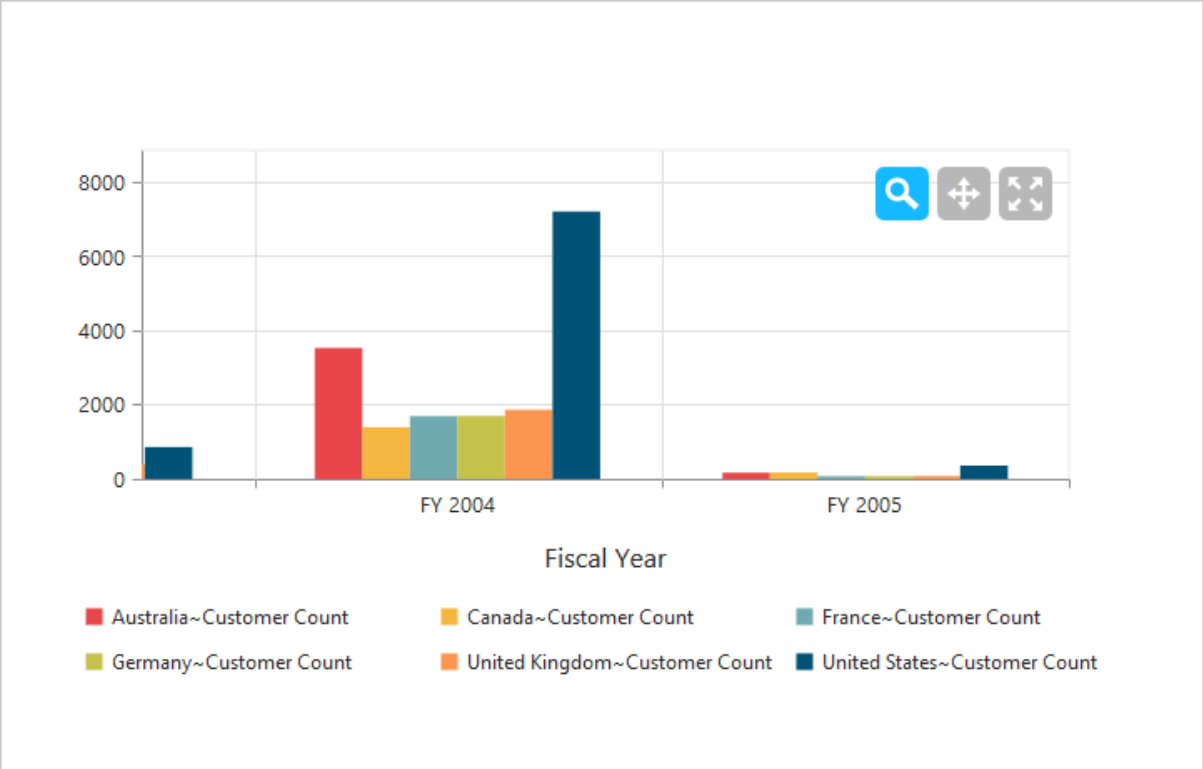
Enable Zooming

There are two ways to zoom the Chart:

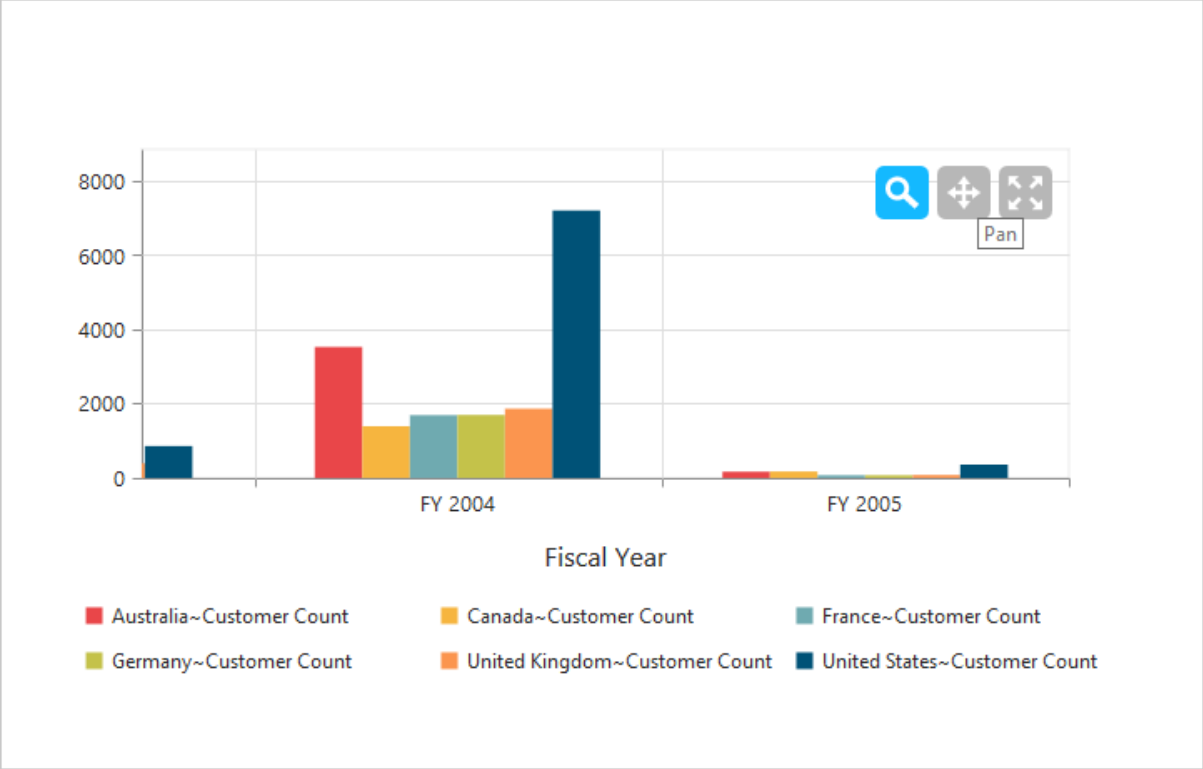
- When zooming.Enable option is set to true, you can zoom the Chart by using rubber band selection.
- When zooming.EnableMouseWheel option is set to true, you can zoom the Chart on mouse wheel scrolling.

ASPX-CS

```
//Enable zooming in chart
@Html.EJ().Pivot().PivotChart("PivotChart1").Zooming(zoom=>
zoom.Enable(true)).Url(Url.Content("/RelationalChartService.svc")).Size(s
ize => size.Height("460px").Width("950px"))
```



After zooming the Chart, a zooming toolbar will appear with options to *zoom, pan and reset*. Selecting the **"Pan"** option will allow to view the Chart and selecting the **"Reset"** option will reset the zoomed Chart.



Copyright © 2001 - 2020 Syncfusion Inc.

Types of Zooming

You can zoom the particular axis like horizontal axis or vertical axis or both axis using Type option in zooming.

Note: By default, the value for the Type option in zooming is “x,y” (indicating both axis) in PivotChart.

ASPX-CS

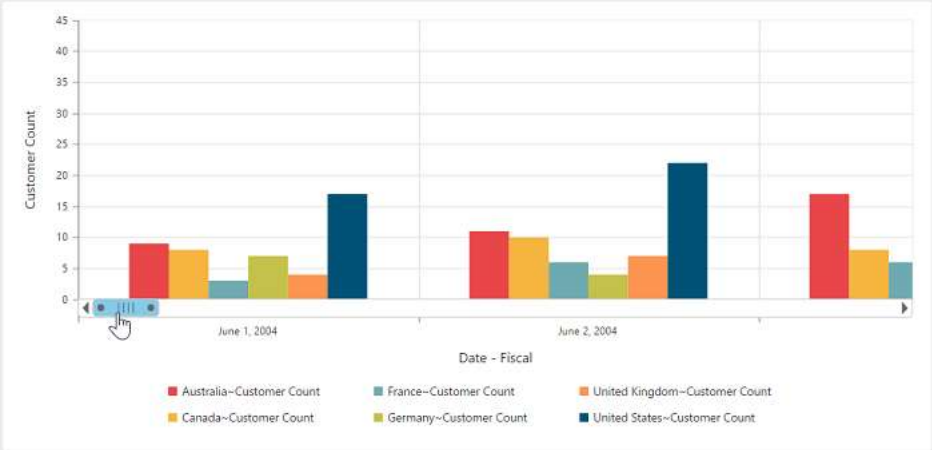
```
//Enable horizontal zooming
@Html.EJ().Pivot().PivotChart("PivotChart1").Zooming(zoom=>
zoom.Enable(true).Type("x")).Url(Url.Content("/RelationalChartService.svc
")).Size(size => size.Height("460px").Width("950px"))
```

Enable Scrollbar

- When zooming.EnableScrollbar option is set to true, the PivotChart is rendered along with the scroll bars for precise view of data. The data can be viewed by using scroll bar or by using mouse wheel scrolling.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Zooming(zoom => {
zoom.EnableScrollbar(true); })
```



Marker and Crosshair

Marker Shape Customization

In PivotChart, you can customize the marker shape with following symbols.

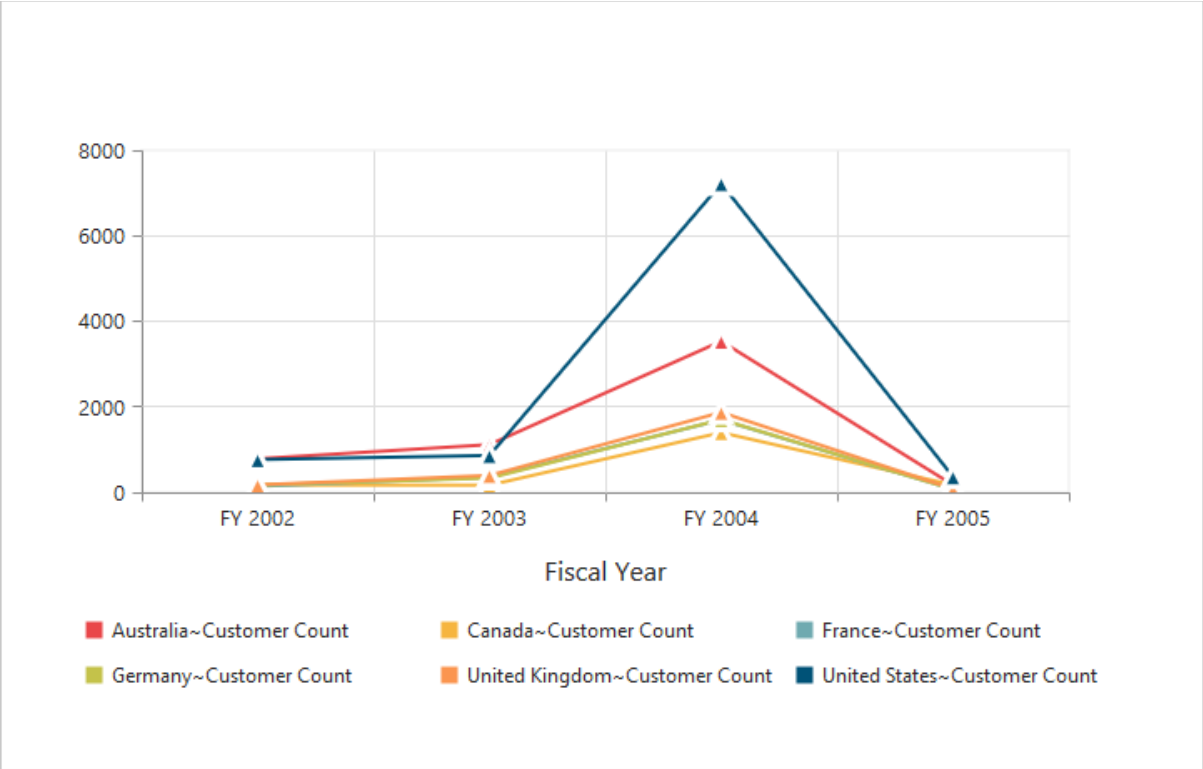
- Rectangle
- Circle
- Cross
- Diamond
- Pentagon
- Hexagon
- Star
- Ellipse

Copyright © 2001 - 2020 Syncfusion Inc.

- Triangle etc.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.Type(SeriesType.Line);}).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++)
this.model.series[seriescount].marker.shape = "Triangle";
</script>
```



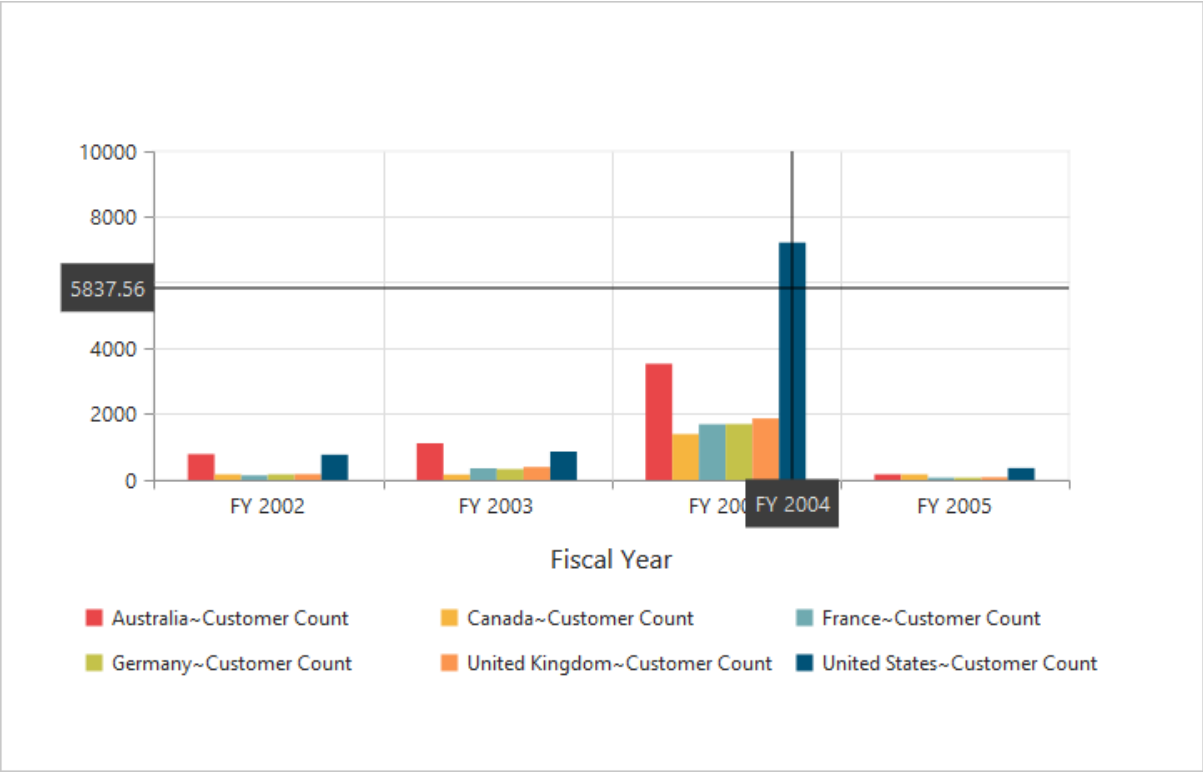
Enable Crosshair and Crosshair Label

Crosshair helps you to view the value at mouse position or touch contact point. Crosshair can be enabled by using the Visible option in CrossHair property. Crosshair label can be enabled by using the **"Visible"** option in CrosshairLabel property within its corresponding axis.

ASPX-CS

```
//Initializing Crosshair and crosshair label
@Html.EJ().Pivot().PivotChart("PivotChart1").CrossHair(crossHair =>
crossHair.Visible(true)).PrimaryXAxis(primaryX =>
primaryX.CrosshairLabel(crosshairlabel =>
crosshairlabel.Visible(true))).Url(Url.Content("/RelationalChartService.s
vc")).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



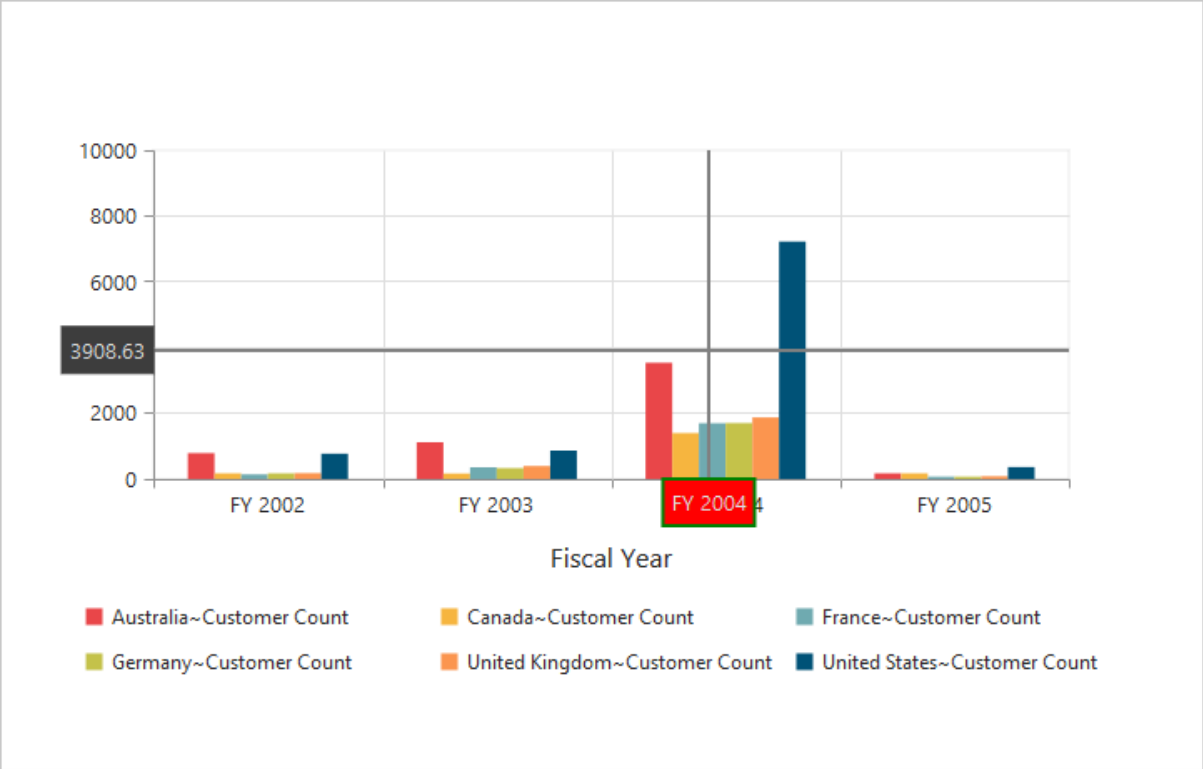
Crosshair Line and Label Customization

By using Line property of crosshair, you can customize its line color and width. Also by using Fill and Border properties of crosshair label, you can customize its background color, border color and border width.

ASPX-CS

```
//Customizing the crosshair label background color and border
@Html.EJ().Pivot().PivotChart("PivotChart1").CrossHair(crossHair =>
crossHair.Line(line =>
line.Color("gray").Width(2)).Visible(true)).PrimaryXAxis(primaryX =>
primaryX.CrosshairLabel(crosshairlabel =>
crosshairlabel.Visible(true).Fill("red").Border(border =>
border.Color("green").Width(2)))).PrimaryYAxis(primaryy =>
primaryy.CrosshairLabel(crosshairlabel =>
crosshairlabel.Visible(true))).Url(Url.Content("/RelationalChartService.s
vc")).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Trackball

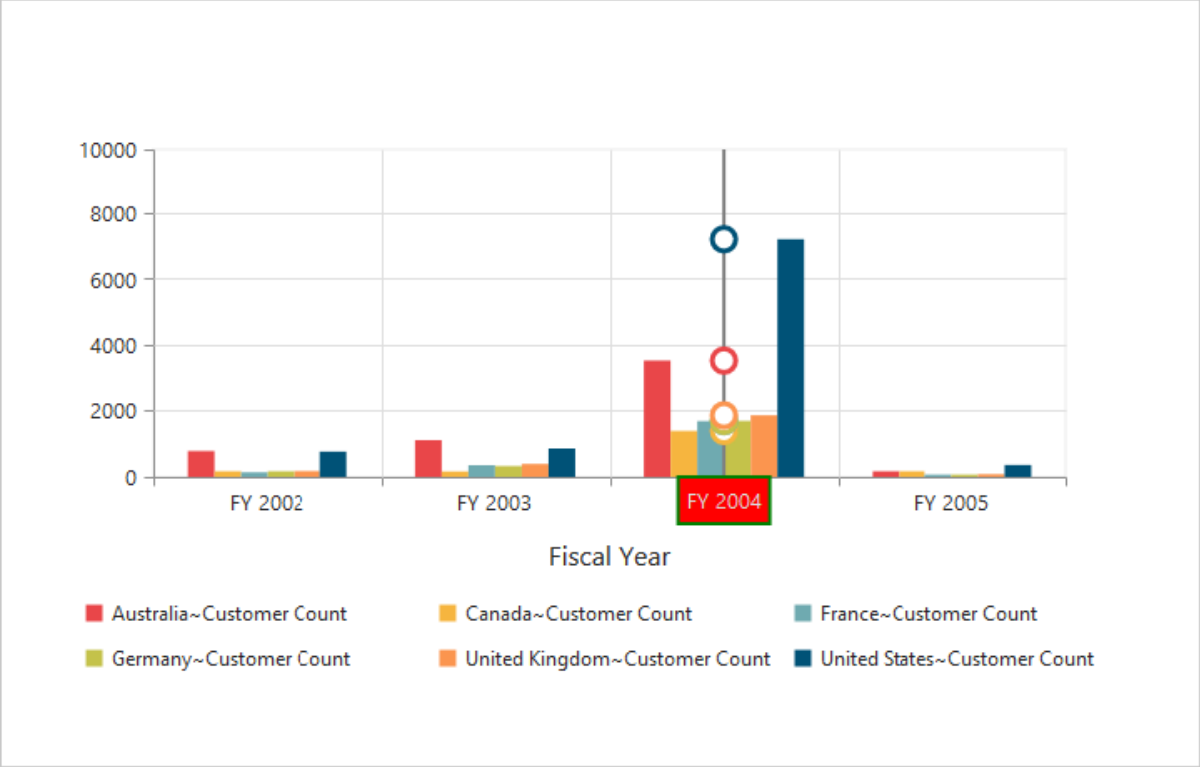
Enable trackball

Trackball can be enabled by setting both - 'Visible' option of the crosshair to true and Type option of the crosshair to **"Trackball".** The default value of type is **"Crosshair".**

ASPX-CS

```
//Change crosshair type to track ball
@Html.EJ().Pivot().PivotChart("PivotChart1").CrossHair(crossHair =>
crossHair.Visible(true).Type(CrosshairType.Trackball)).Url(Url.Content("/
RelationalChartService.svc")).Size(size =>
size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



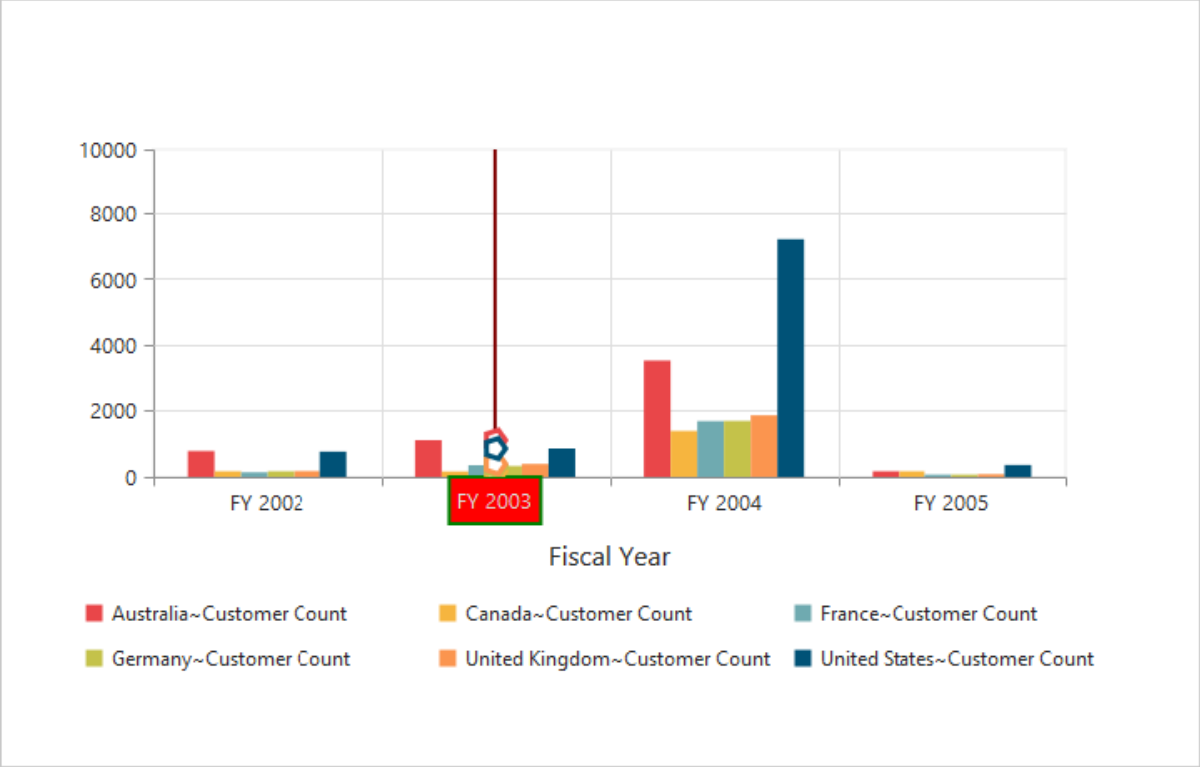
Trackball Marker and Line Customization

Shape and size of the trackball marker can be customized using the Shape and Size options of the crosshair marker. Color and width of the trackball line can be customized using the **"Line"** option in the crosshair.

ASPX-CS

```
//Customize the trackball line color and width and Marker visible and
shape
@Html.EJ().Pivot().PivotChart("PivotChart1").CrossHair(crossHair =>
crossHair.Visible(true).Type(CrosshairType.Trackball).Line(line =>
line.Color("#800000").Width(2)).Marker(marker =>
marker.Shape(ChartShape.Pentagon).Size(size =>
size.Height(9).Width(9)).Visible(true))).Url(Url.Content("/RelationalChar
tService.svc")).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Highlight

PivotChart provides highlighting support for the series and data points on mouse hover. To enable highlighting, set the **"enable"** property to true in the highlightSettings option of the series.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++)
this.model.series[seriescount].highlightSettings.enable = true
</script>
```

Highlight Mode

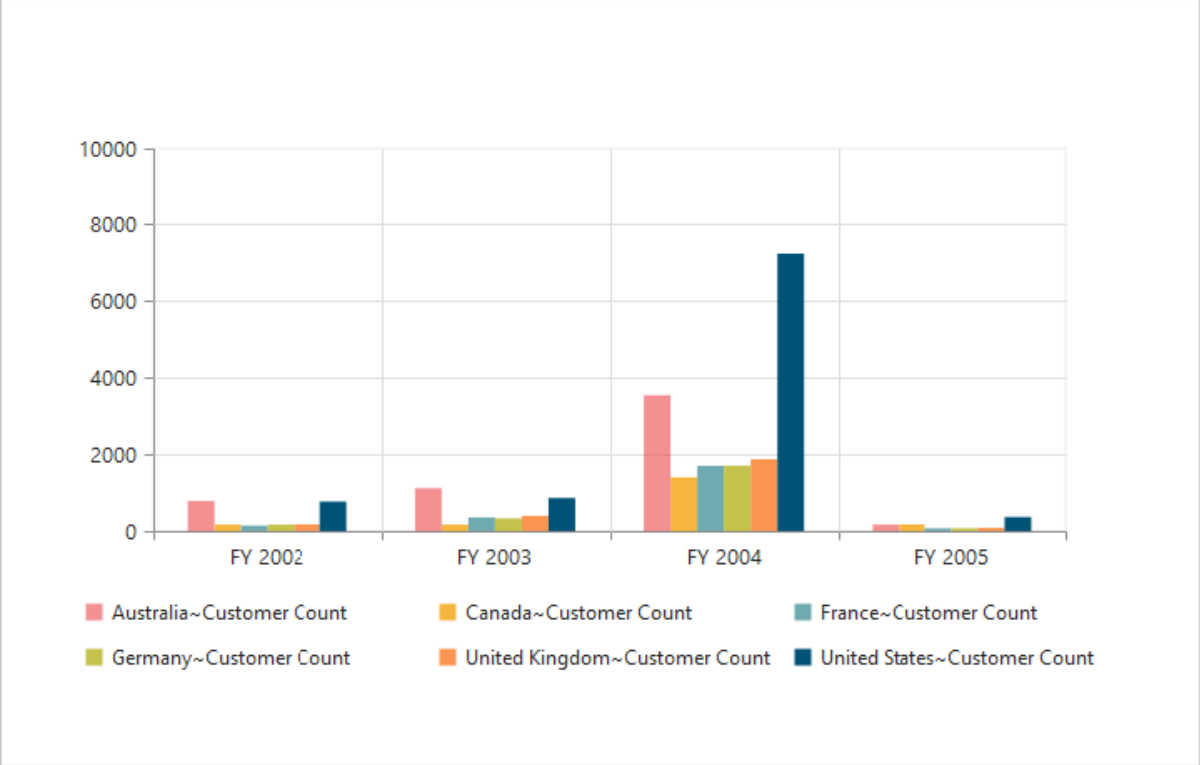
You can set three different modes for highlighting data points and series by using the mode property of the highlightSettings.

- series
- points
- cluster

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++){
this.model.series[seriescount].highlightSettings.enable = true;
this.model.series[seriescount].highlightSettings.mode = "series";
}
</script>
```



Customize the Highlight Styles

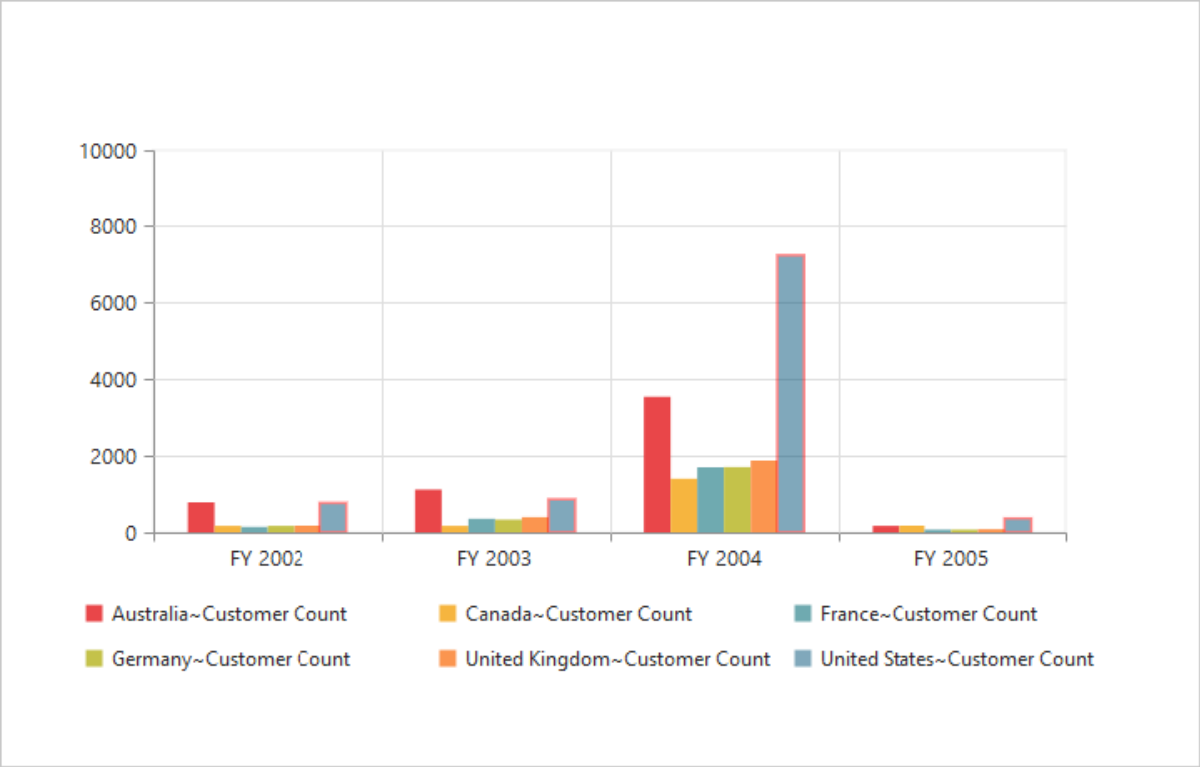
To customize the highlighted series, use border.color, border.width and opacity options in the highlightSettings property.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++){
this.model.series[seriescount].highlightSettings.enable = true;
this.model.series[seriescount].highlightSettings.opacity = "0.5";
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
this.model.series[seriescount].highlightSettings.border.width = "1.5";
this.model.series[seriescount].highlightSettings.border.color = "red";
}
</script>
```



Patterns to Highlight

PivotChart provides pattern support for highlighting the data by setting an appropriate value to the `pattern` property of the `highlightSettings`. The different types of highlight patterns are as follows.

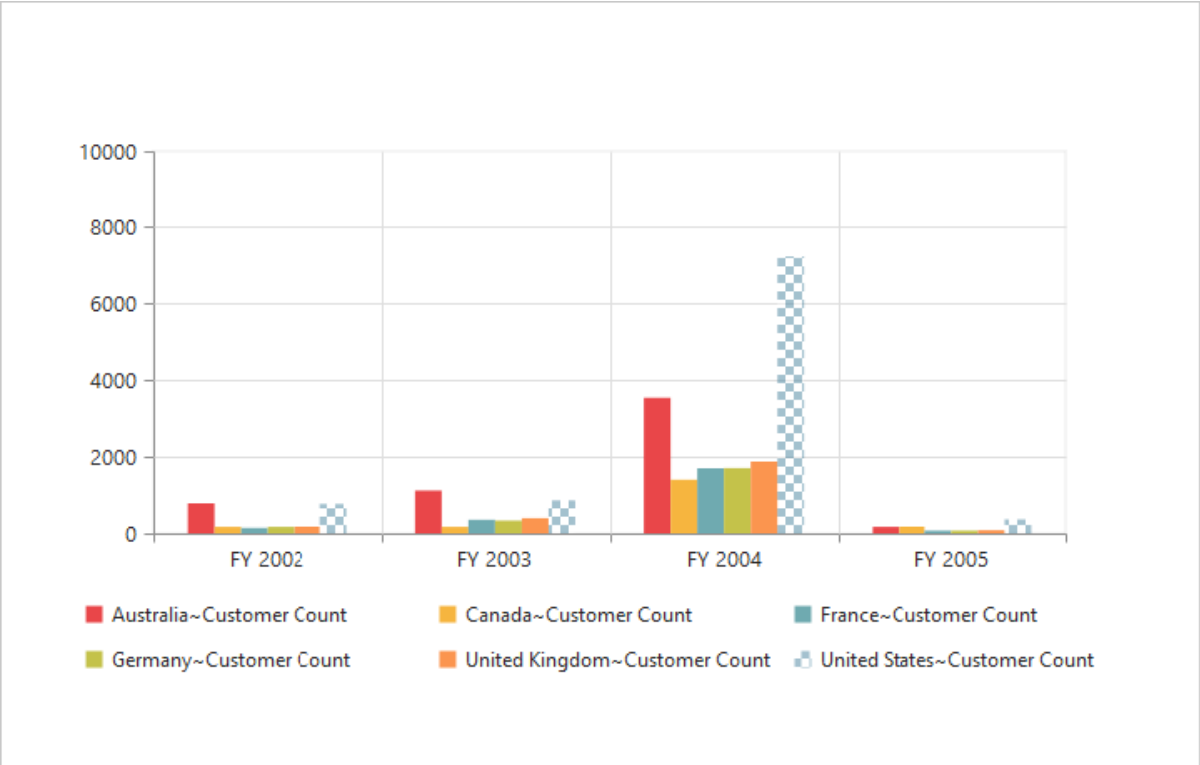
- chessboard
- crosshatch
- dots
- pacman
- grid
- turquoise
- star
- triangle
- circle
- tile
- horizontalDash
- verticalDash
- rectangle
- box
- verticalStripe
- horizontalStripe
- bubble

Copyright © 2001 - 2020 Syncfusion Inc.

- diagonalBackward
- diagonalForward

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++){
this.model.series[seriescount].highlightSettings.enable = true;
this.model.series[seriescount].highlightSettings.pattern = "chessboard";
}
</script>
```



Selection

PivotChart provides selection support for the series and data points on mouse click. To enable selection, set the **“enable”** property to true in the selectionSettings option of the series.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
```

```
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++)
this.model.series[seriescount].selectionSettings.enable = true
</script>
```

Selection Mode

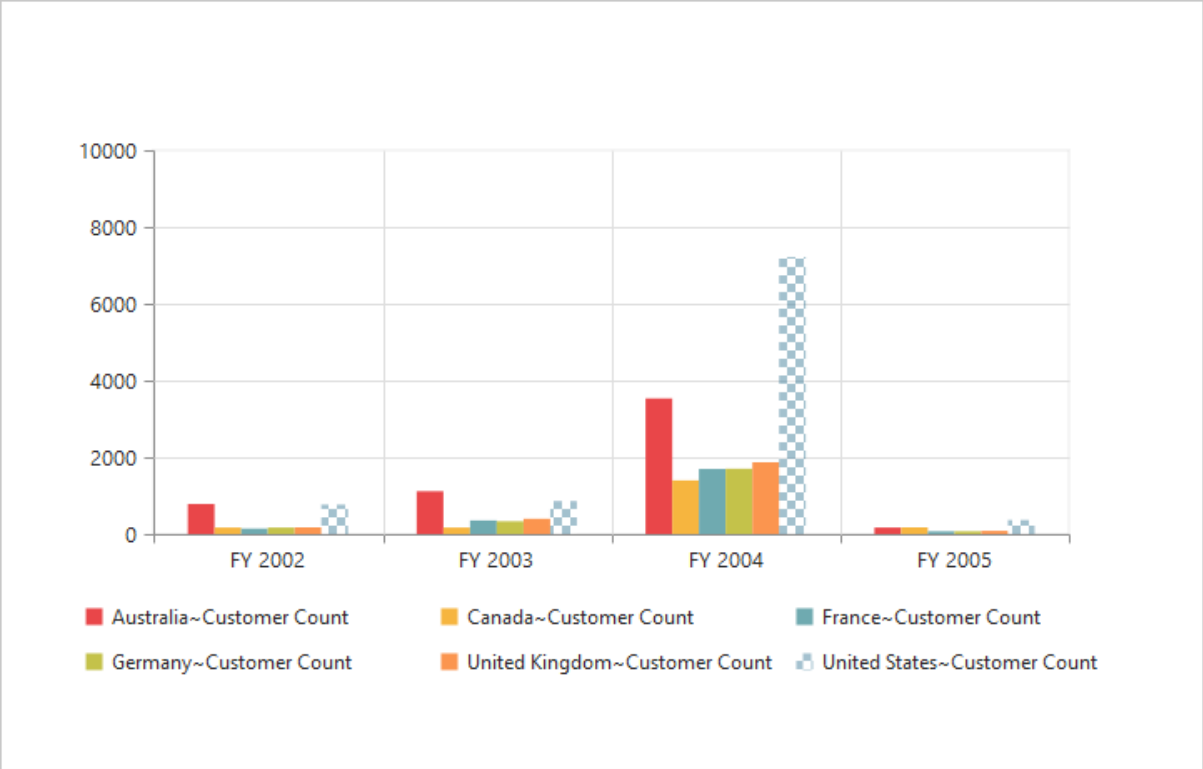
You can set three different selection mode for highlighting the data points and series by using the mode property of the selectionSettings.

- series
- points
- cluster

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++){
this.model.series[seriescount].selectionSettings.enable = true;
this.model.series[seriescount].selectionSettings.mode = "series";
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



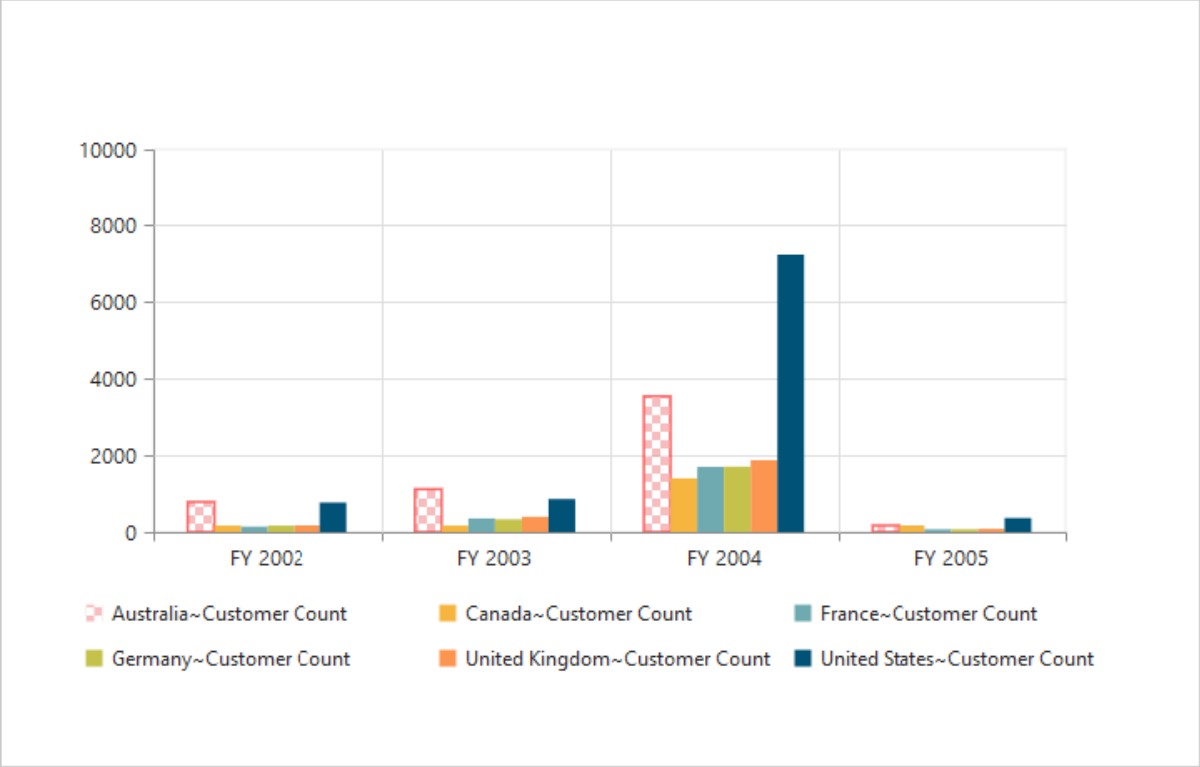
Customize the Selection Styles

To customize the selection styles, use the border.color, border.width and opacity options in the selectionSettings.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++){
this.model.series[seriescount].selectionSettings.enable = true;
this.model.series[seriescount].selectionSettings.opacity = "0.5";
this.model.series[seriescount].selectionSettings.border.width = "1.5";
this.model.series[seriescount].selectionSettings.border.color = "red";
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Patterns for Selection

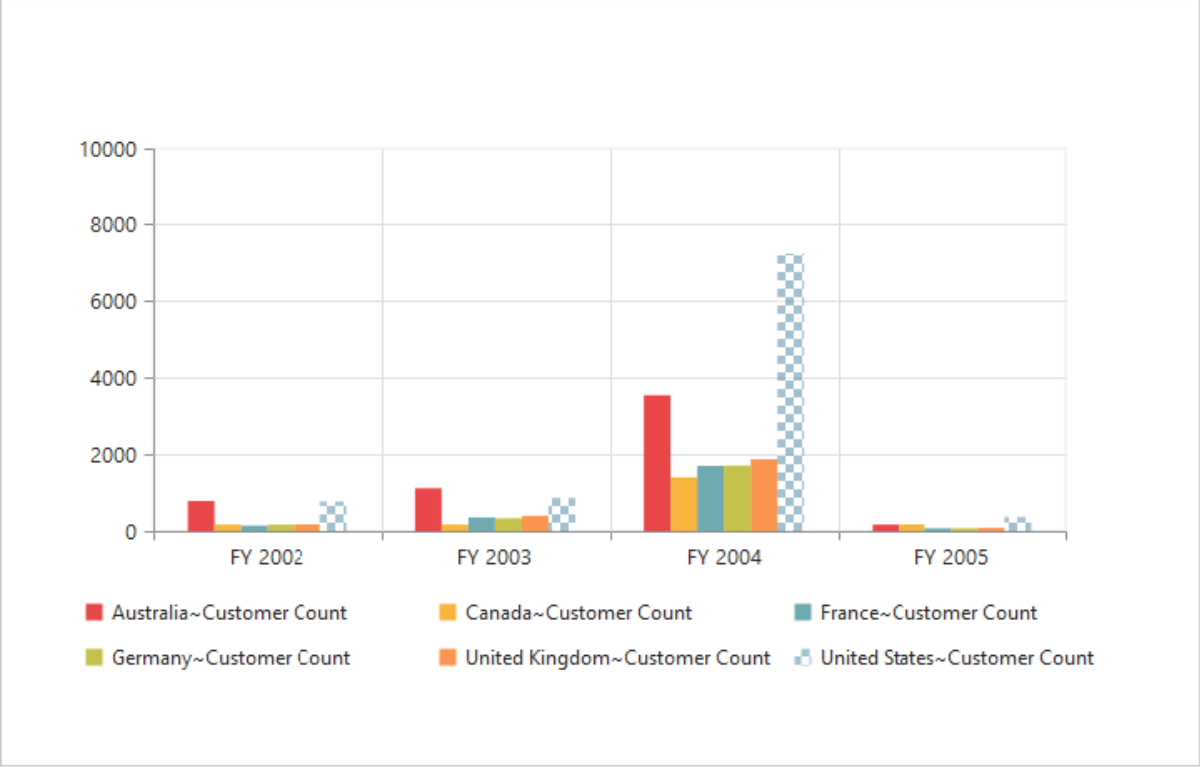
PivotChart provides pattern support for the selecting the data by setting an appropriate value to the `pattern` property of the `selectionSettings` option. The different types of selection patterns are as follows.

- chessboard
- crosshatch
- dots
- pacman
- grid
- turquoise
- star
- triangle
- circle
- tile
- horizontalDash
- verticalDash
- rectangle
- box
- verticalStripe
- horizontalStripe
- bubble
- diagonalBackward
- diagonalForward

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Size(size =>
size.Height("460px").Width("950px")).ClientSideEvents(
oEve => { oEve.SeriesRendering("onSeriesRenders"); })
<script>
function onSeriesRenders(args) {
for (var seriescount = 0; seriescount < this.model.series.length;
seriescount++){
this.model.series[seriescount].selectionSettings.enable = true;
this.model.series[seriescount].selectionSettings.pattern = "chessboard";
}
</script>
```



Exporting

The PivotChart control can be exported to the following file formats.

- Excel
- Word
- PDF
- Image

The PivotChart control can be exported by invoking **“exportPivotChart”** method, with an appropriate export option as parameter.

ASPX-CS

```
//If you want to render PivotChart in Client Mode.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotChart("PivotChart1").CommonSeriesOptions(comm =>
{ comm.Type(SeriesType.Column).Tooltip(tool => { tool.Visible(true);
}).EnableAnimation(true); }).Size(size =>
size.Height("460px").Width("100%")).PrimaryYAxis(primaryYAxis =>
primaryYAxis.Title(title => title.Text("Amount"))).Legend(legend =>
legend.Visible(true)).ClientSideEvents(oEve => { oEve.Load("load");
}).IsResponsive(true).DataSource(dataSource => dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
rows.FieldName("State").FieldCaption("State").Add(); }).Columns(columns
=> { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add(); }))
//If you want to render PivotChart in Server Mode.
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc"))..Size(size => size.Height("460px").Width("100%"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var chartObj = $('#PivotChart1').data("ejPivotChart");
//If you render PivotChart in Client Mode, set the export option like
below.
chartObj.exportPivotChart("ExcelExport","fileName");
//If you render PivotChart in Server Mode, set the export option like
below.
chartObj.exportPivotChart(ej.PivotChart.ExportOptions.Excel);;
}
</script>
```

To achieve exporting in client mode, we need to add **"Syncfusion.EJ.Export"** dependency library into the application.

When PivotChart is rendered in Client Mode, a method needs to be added in MVC controller file of the application and we need to import **"Syncfusion.EJ.Export"** namespace in the controller file.

C#

```
public void ExcelExport()
{
System.Web.Script.Serialization.JavaScriptSerializer serializer = new
System.Web.Script.Serialization.JavaScriptSerializer() { MaxJsonLength =
Int32.MaxValue };
PivotChartExcelExport pivotChartExcelExport = new
PivotChartExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartExcelExport.ExportToExcel(chartParams);
}
```

When PivotChart is rendered in Server Mode, a service method needs to be added in WCF/WebAPI for server side operations.

For WebAPI controller, the below method needs to be added.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export() {
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
htmlHelper.ExportPivotChart(args, fileName,
System.Web.HttpContext.Current.Response);
}
```

For WCF service, the below method needs to be added.

C#

```
public void Export(System.IO.Stream stream) {
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
string fileName = "Sample";
htmlHelper.ExportPivotChart(args, fileName,
System.Web.HttpContext.Current.Response);
}
```

Excel Export

User can export contents of the PivotChart to Excel document for future archival, references and analysis purposes.

Client Mode

To achieve Excel export, method name **"ExcelExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
chartObj.exportPivotChart("ExcelExport","fileName");
}
</script>
```

Following method need to be added in MVC controller file of the application.

C#

```
public void ExcelExport()
{
System.Web.Script.Serialization.JavaScriptSerializer serializer = new
System.Web.Script.Serialization.JavaScriptSerializer() { MaxJsonLength =
Int32.MaxValue };
PivotChartExcelExport pivotChartExcelExport = new
PivotChartExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartExcelExport.ExportToExcel(chartParams);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

Server Mode

To achieve Excel export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

For Excel export, **"ej.PivotChart.ExportOptions.Excel"** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart");
//Setting export option as Excel in the exportPivotChart method for
ServerMode
chartObj.exportPivotChart(ej.PivotChart.ExportOptions.Excel);
}
```

Word Export

User can export contents of the PivotChart to Word document for future archival, references and analysis purposes.

Client Mode

To achieve Word export, method name **"WordExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
chartObj.exportPivotChart("WordExport","fileName");
}
</script>
```

Following method need to be added in MVC controller file of the application.

C#

```
public void WordExport()
{
System.Web.Script.Serialization.JavaScriptSerializer serializer = new
System.Web.Script.Serialization.JavaScriptSerializer() { MaxJsonLength =
Int32.MaxValue };
PivotChartWordExport pivotChartWordExport = new PivotChartWordExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartWordExport.ExportToWord(chartParams);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Server Mode

To achieve Word export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.DocIO.Base

For Word export, **“ej.PivotChart.ExportOptions.Word”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart");
//Setting export option as Word in the exportPivotChart method
chartObj.exportPivotChart(ej.PivotChart.ExportOptions.Word);
}
```

PDF Export

User can export contents of the PivotChart to PDF document for future archival, references and analysis purposes.

Client Mode

To achieve Word export, method name **“PDFExport”** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
chartObj.exportPivotChart("PDFExport","fileName");
}
</script>
```

Following method need to be added in MVC controller file of the application.

C#

```
public void PDFExport()
{
System.Web.Script.Serialization.JavaScriptSerializer serializer = new
System.Web.Script.Serialization.JavaScriptSerializer() { MaxJsonLength =
Int32.MaxValue };
PivotChartPDFExport pivotChartPDFExport = new PivotChartPDFExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartPDFExport.ExportToPDF(chartParams);
}
```

Server Mode

To achieve PDF export, we need to add the following dependency libraries into the application.

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base

For PDF export, **“ej.PivotChart.ExportOptions.PDF”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
//Setting export option as PDF in the exportPivotChart method
chartObj.exportPivotChart(ej.PivotChart.ExportOptions.PDF);
}
```

Image Export

User can export contents of the PivotChart to image format for future archival, references and analysis purposes. We can export PivotChart to the following image formats.

- PNG
- EMF
- JPG
- GIF
- BMP

Client Mode

To achieve image export, method name **“ImageExport”** ,**”ej.PivotChart.ExportOptions.PNG”** and file name is sent as the parameter.This is similar to other image formats.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
chartObj.exportPivotChart("ImageExport","fileName",ej.PivotChart.ExportOp
tions.PNG);
}
</script>
```

Following method need to be added in MVC controller file of the application.

C#

```
public void ImageExport()
{
System.Web.Script.Serialization.JavaScriptSerializer serializer = new
System.Web.Script.Serialization.JavaScriptSerializer() { MaxJsonLength =
Int32.MaxValue };
PivotChartImageExport pivotChartImageExport = new
PivotChartImageExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
pivotChartImageExport.ExportToImage(chartParams);
}
```

Server Mode

To export PivotChart in PNG format, **"ej.PivotChart.ExportOptions.PNG"** enumeration value is sent as the parameter. This is similar to other image formats.

JS

```
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
//Setting export option as PNG in the exportPivotChart method
chartObj.exportPivotChart(ej.PivotChart.ExportOptions.PNG);
}
```

PivotChart format selection

Information: This option is applicable only for PivotChart specifically when exported to Excel document.

You can set an option to export PivotChart to an Excel document, either as image or PivotChart format itself by setting the boolean property `exportChartAsImage`, inside the `BeforeExport` event.

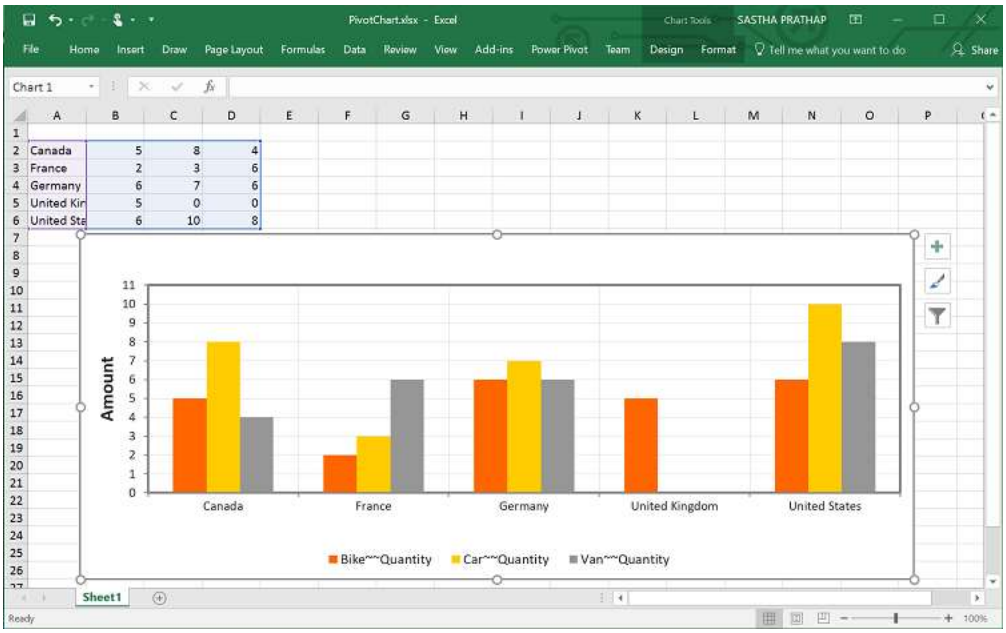
Note: By default PivotChart will be exported as image format to Excel document.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").ClientSideEvents(clientSideE
vents => clientSideEvents.BeforeExport("Exporting"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var chartObj = $('#PivotChart1').data("ejPivotChart");
chartObj.exportPivotChart("ExcelExport","fileName");
}
function Exporting(args) {
args.exportChartAsImage = false; //you can set the chart format here
}
</script>
```

The below screenshot shows the control exported to Excel document showing its own format (Pivoting Chart).

Copyright © 2001 - 2020 Syncfusion Inc.



Exporting Customization

You can add title and description to the exporting document by using title and description property obtained in the "beforeExport" event.

Note: Title and description cannot be added to image formats.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").ClientSideEvents(clientSideE
vents => clientSideEvents.BeforeExport("Exporting"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var chartObj = $('#PivotChart1').data("ejPivotChart");
//If you render PivotChart in Client Mode, set the export option like
below.
chartObj.exportPivotChart("ExcelExport","fileName");
//If you render PivotChart in Server Mode, set the export option like
below.
chartObj.exportPivotChart(ej.PivotChart.ExportOptions.Excel);;
}
function Exporting(args) {
args.title = "PivotChart";
args.description = "Visualizes both OLAP and Relational datasource in
graphical format";
}
</script>
```

You can also edit the exporting document with the use of a server side event for required exporting option.

C#

```
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using Syncfusion.EJ.Export;
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
//Following methods needs to be added in MVC controller file of the
application for JSON export
public void ExcelExport()
{
PivotChartExcelExport pivotChartExcelExport = new
PivotChartExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotChartExcelExport.ExcelExport += pivotChartExcelExport_ExcelExport;
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartExcelExport.ExportToExcel(chartParams);
}
void pivotChartExcelExport_ExcelExport(object sender,
Syncfusion.XlsIO.IWorkbook workBook)
{
//You can customize exporting document here.
}
public void WordExport()
{
PivotChartWordExport pivotChartWordExport = new PivotChartWordExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotChartWordExport.WordExport += pivotChartWordExport_WordExport;
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartWordExport.ExportToWord(chartParams);
}
void pivotChartWordExport_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
public void PDFExport()
{
PivotChartPDFExport pivotChartPDFExport = new PivotChartPDFExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotChartPDFExport.AddPDFHeaderFooter +=
pivotChartPDFExport_AddPDFHeaderFooter;
pivotChartPDFExport.PDFExport += pivotChartPDFExport_PDFExport;
Dictionary<string, string> chartParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotChartPDFExport.ExportToPDF(chartParams);
}
void pivotChartPDFExport_PDFExport(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can customize exporting document here.
}
void pivotChartPDFExport_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
//You can add header/footer information to the PDF document.
}
//Following service method needs to be added in WCF/WebAPI for
PivotEngine.
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
htmlHelper.ExcelExport += htmlHelper_ExcelExport;
htmlHelper.WordExport += htmlHelper_WordExport;
htmlHelper.AddPDFHeaderFooter += htmlHelper_AddPDFHeaderFooter;
htmlHelper.PDFExport += htmlHelper_PDFExport;
htmlHelper.ExportPivotChart(args, fileName,
System.Web.HttpContext.Current.Response);
}
void htmlHelper_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void htmlHelper_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void htmlHelper_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
void htmlHelper_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
```

The name of the document can be customized as per the users requirement.

For Client mode, we need to send file name as parameter to the **“exportPivotChart”** method along with service URL.

JAVASCRIPT

```
function exportBtnClick(args)
{
var chartObj = $('#PivotChart1').data("ejPivotChart ");
chartObj.exportPivotChart("https://js.syncfusion.com/ejservices/api/Pivot
Chart/Olap/ExcelExport", "fileName");
}
```

For Server mode, the exporting document name is provided in the WebAPI controller as found in the below code snippet.

Copyright © 2001 - 2020 Syncfusion Inc.

C#

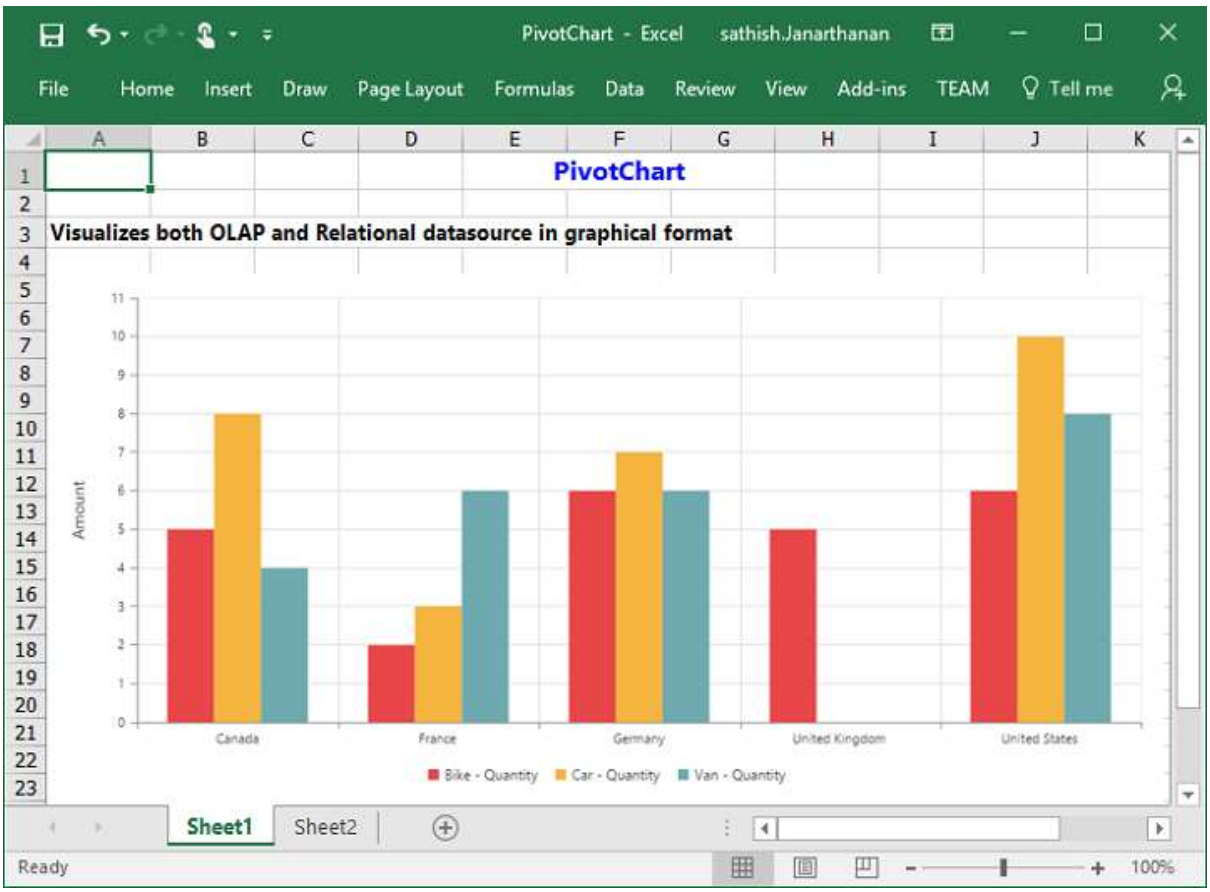
```
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export() {
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "File name is customized here";
htmlHelper.ExportPivotChart(args, fileName,
System.Web.HttpContext.Current.Response);
}
```

For customizing name in WCF Service, below code snippet is used.

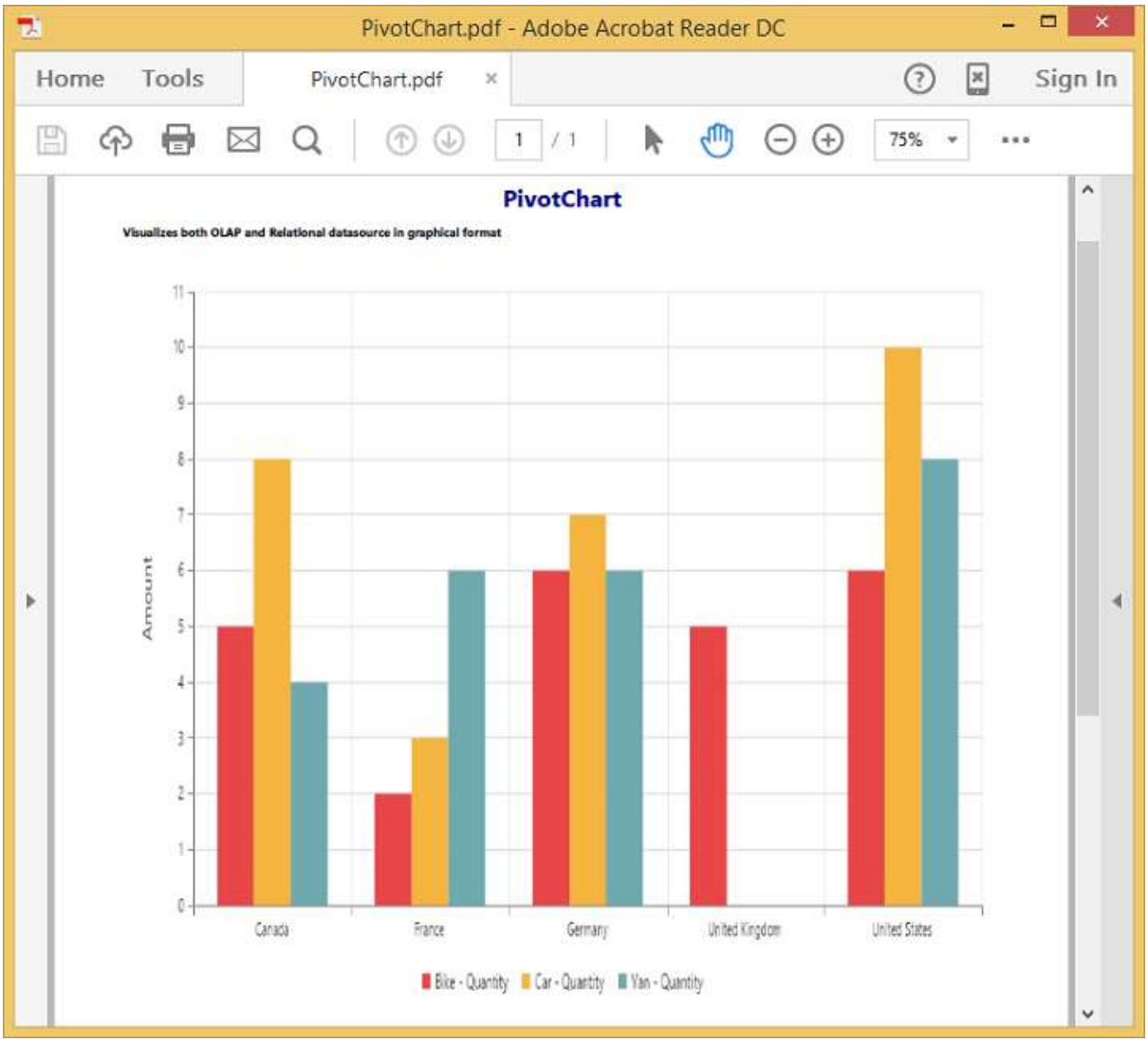
C#

```
public void Export(System.IO.Stream stream) {
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
string fileName = " File name is customized here ";
htmlHelper.ExportPivotChart(args, fileName,
System.Web.HttpContext.Current.Response);
}
```

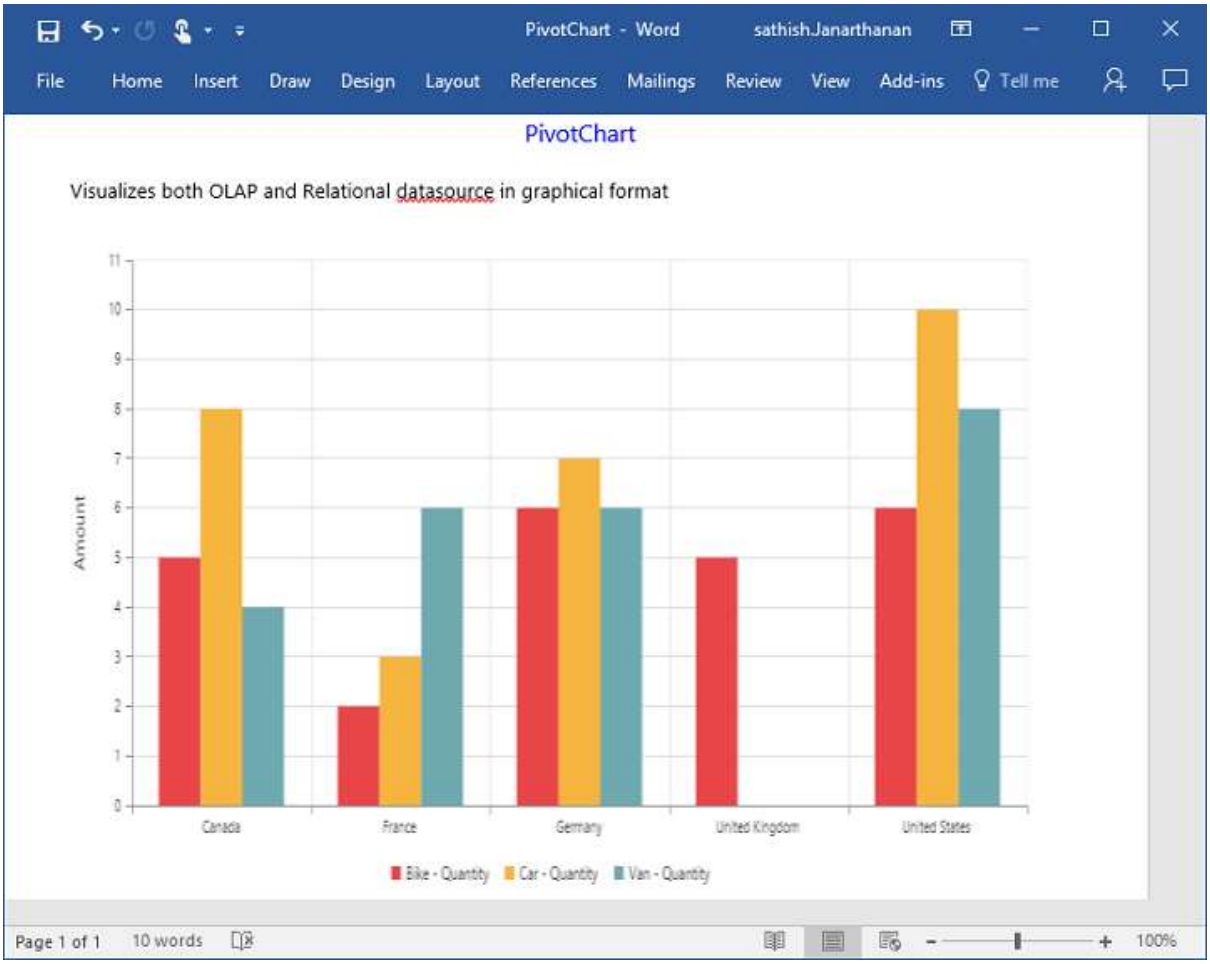
The below screenshot shows the PivotChart control exported to Excel document.



The below screenshot shows the PivotChart control exported to PDF document.



The below screenshot shows the PivotChart control exported to Word document.



The below screenshot shows the PivotChart control exported to PNG format.



Appearance

Built-in Themes

Following are the built-in themes available in the PivotChart.

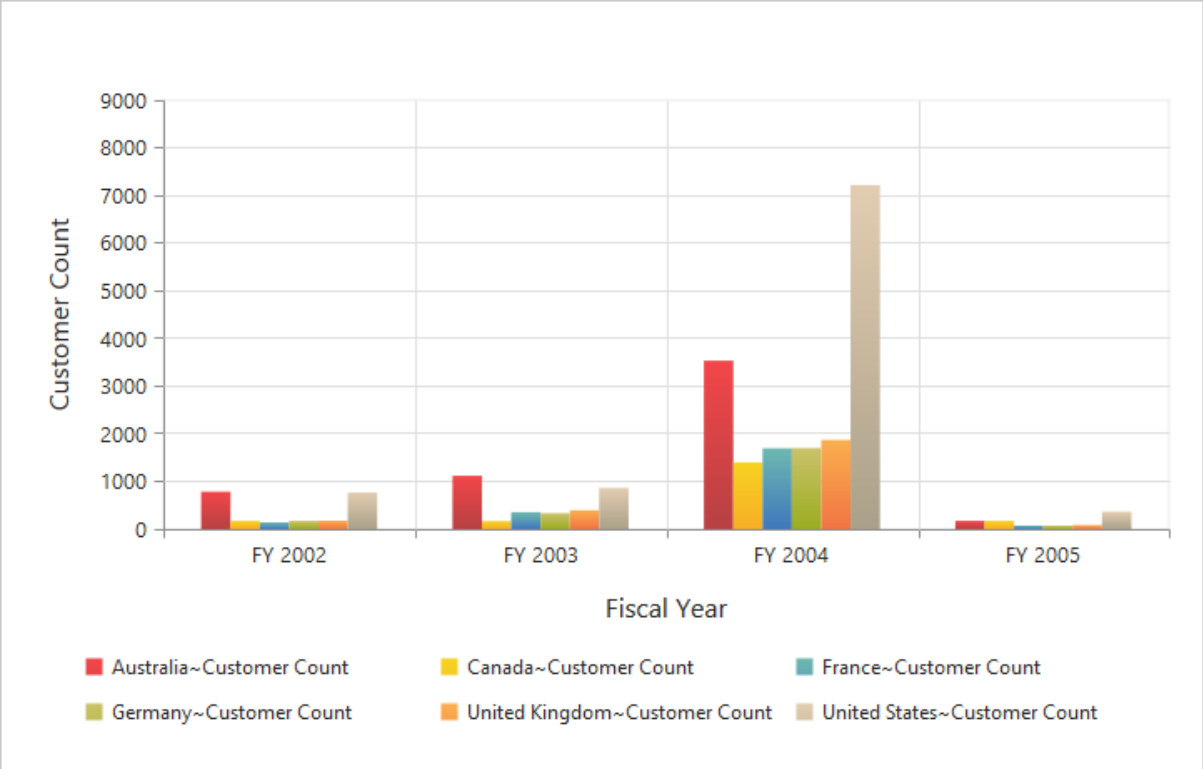
- flatlight
- gradientlight
- gradientdark
- azure
- azuredark
- lime
- limedark
- saffron
- saffrondark
- gradientlight
- gradientdark
- highcontrast01
- highcontrast02
- material
- office365
- bootstrap

By using the Theme property, you can set the desired theme in PivotChart. By default, **“Flat Light”** theme is applied to PivotChart.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Using gradient theme
@Html.EJ().Pivot().PivotChart("PivotChart1").Theme(Syncfusion.JavaScript.
DataVisualization.ChartTheme.GradientLight).Url(Url.Content("/RelationalC
hartService.svc")).Size(size => size.Height("460px").Width("950px"))
```



PivotChart - Area Customization

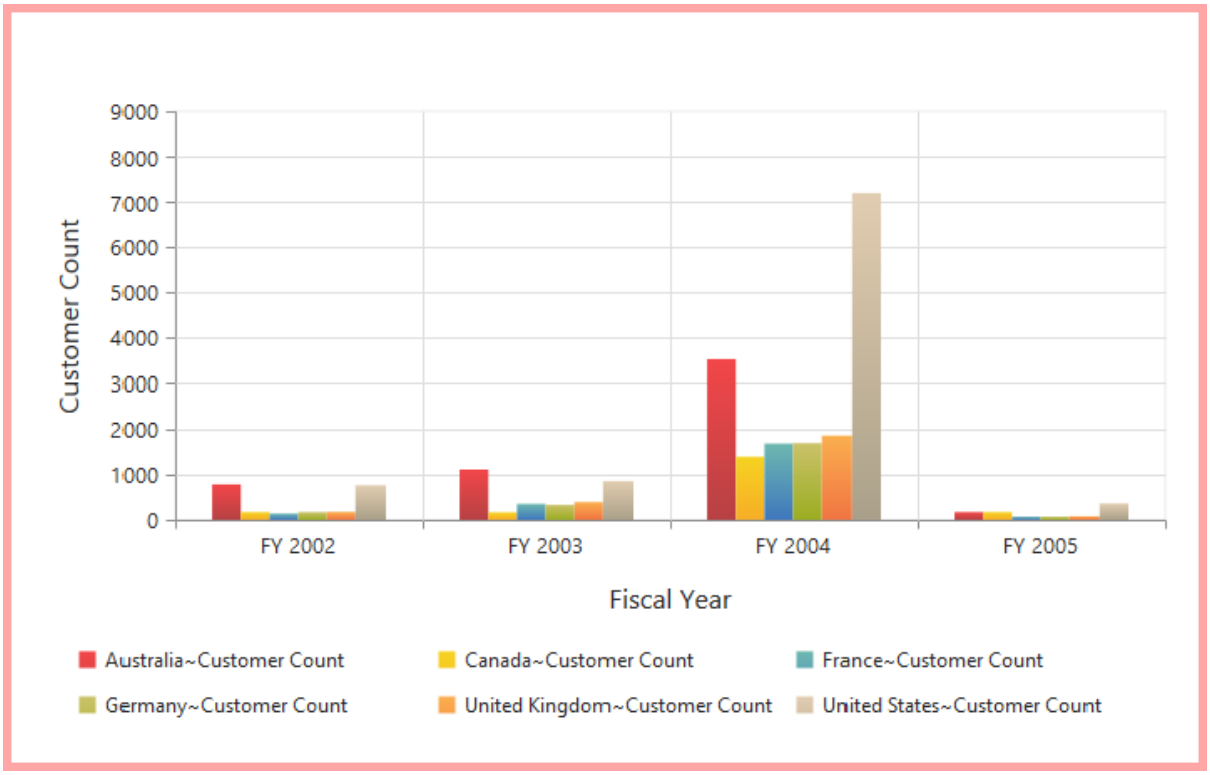
Border Customization

To customize the PivotChart border, use Border property in PivotChart.

ASPX-CS

```
//Customize the chart border and opacity
@Html.EJ().Pivot().PivotChart("PivotChart1").Border(border =>
border.Color("#FF0000").Width(5)).Url(Url.Content("/RelationalChartServic
e.svc")).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



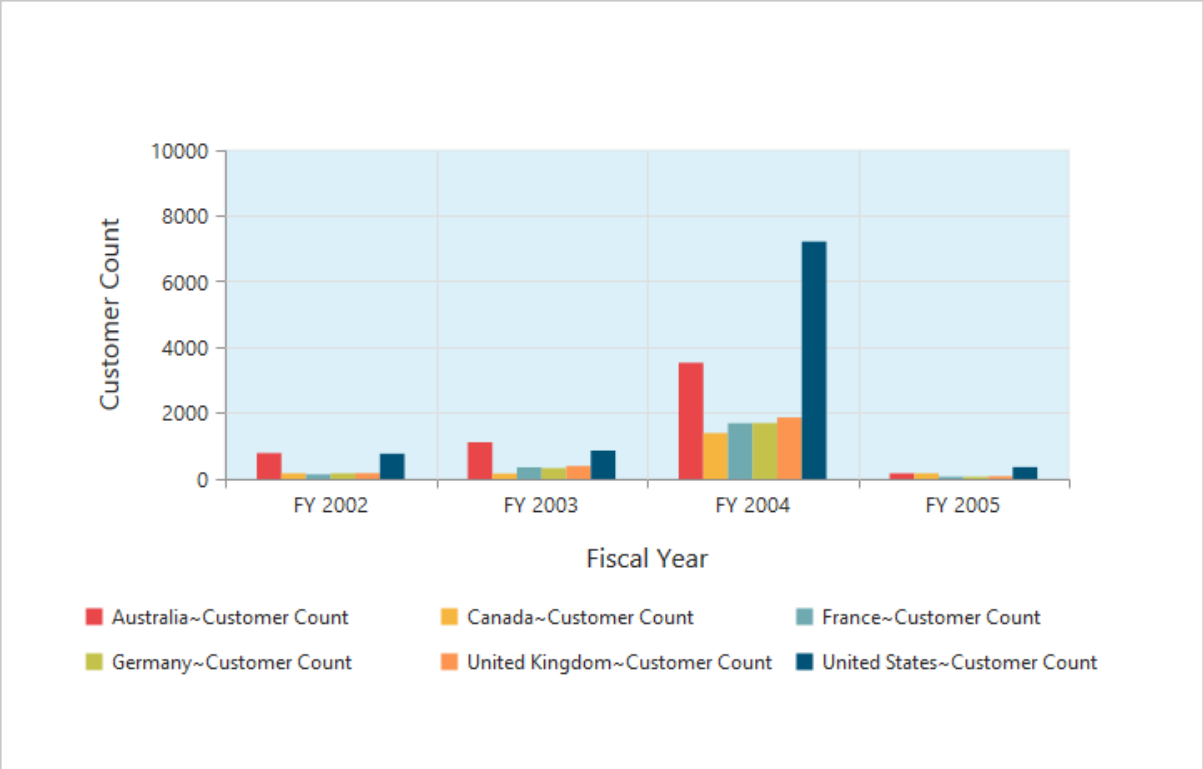
Background Customization

The PivotChart control background can be customized by using the Background property in the Chart area.

ASPX-CS

```
//Setting background for Chart area
@Html.EJ().Pivot().PivotChart("PivotChart1").ChartArea(area =>
area.Background("skyblue")).Url(Url.Content("/RelationalChartService.svc"
)).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



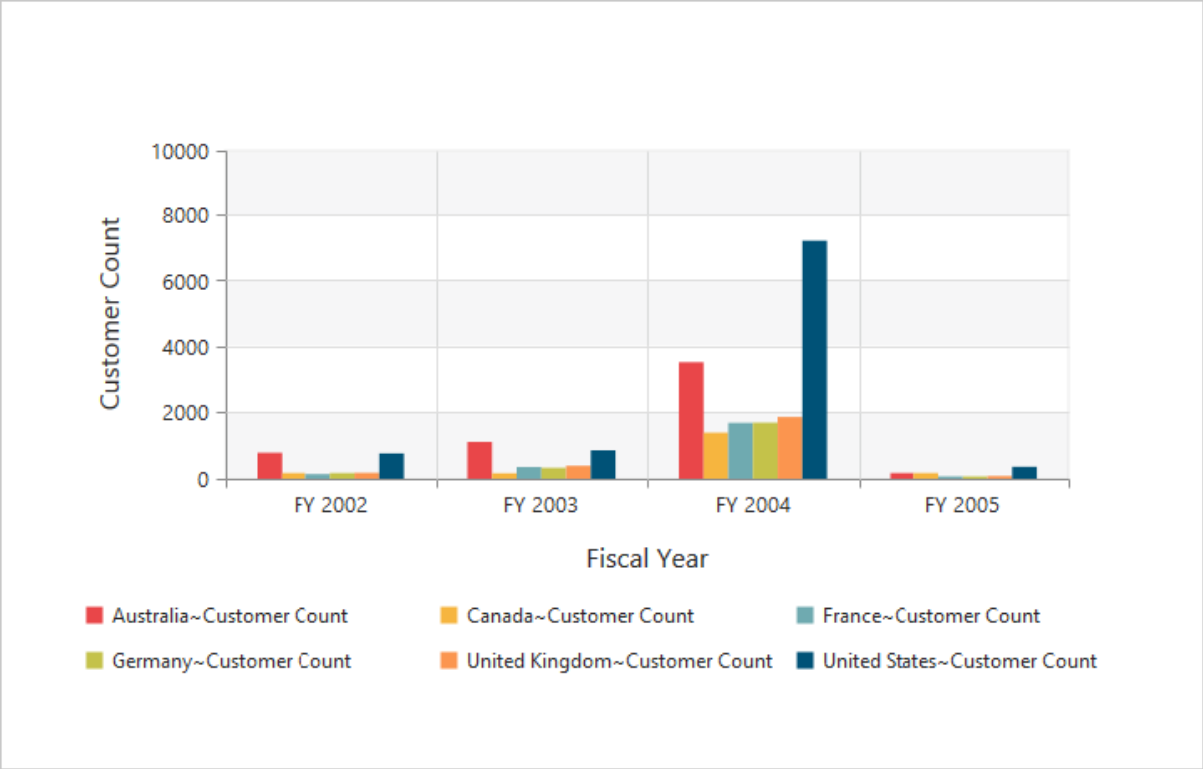
Grid Bands Customization

By using the AlternateGridBand property of the axis, you can provide different color for grid rows and columns formed by the grid lines in the Chart area. The properties Odd and Even are used to customize the grid bands at odd and even positions respectively.

ASPX-CS

```
//Customizing horizontal grid bands at even position
@Html.EJ().Pivot().PivotChart("PivotChart1").PrimaryYAxis(primaryYAxis =>
primaryYAxis.AlternateGridBand(gridband => gridband.Even(even =>
even.Fill("#A7A9AB").Opacity(0.1)))).Url(Url.Content("/RelationalChartSer
vice.svc")).Size(size => size.Height("460px").Width("950px"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Animation

You can enable animation by using the EnableAnimation property under CommonSeriesOptions of the PivotChart control. This animates the Chart series on two occasions - when the Chart is loaded for the first time and when you change the series type by using the Type property.

ASPX-CS

```
//Enabling animation in series
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).CommonSeriesOptions(comm => {
comm.EnableAnimation(true); }).Size(size =>
size.Height("460px").Width("100%"))
```

Localization

Localization in PivotChart

We can localize the PivotChart controls text with a collection of localized strings using **ej.PivotChart.Locale** for different cultures. By default, the PivotChart control is localized in **“en-US”.**

Following code example illustrates on how to localize PivotChart based on **“French”** culture.

ASPX-CS

```
@Html.EJ().Pivot().PivotChart("PivotChart1").Url(Url.Content("/Relational
ChartService.svc")).Locale("fr-FR").Size(size =>
size.Height("320px").Width("100%"))
<script type="text/javascript">
ej.PivotChart.Locale["fr-FR"] = {
Measure: "Mesure",
Row: "Rangée",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Column: "Colonne",
Value: "Valeur",
Expand: "Développer",
Collapse: "Effondrement",
Exit: "Quitter"
};
</script>
```

Following table localizes the in-built keywords to **“French”** culture for PivotChart.

Keywords	Values
Measure	“Mesure”
Row	"Rangée "
Column	"Colonne”
Value	"Valeur "
Expand	"Développer "
Collapse	"Effondrement "
Exit	“Quitter "

Localization and Globalization of Cube Info (Client Mode)

Content displayed within the PivotChart control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

- To get localized data from OLAP Cube, we need to set **“Locale Identifier”** in the connection string to a specific culture in the **“Data”** property present inside **“DataSource”**.
- To bind the globalized content in PivotChart control, we need to set **“Locale”** property to a specific culture and want to refer specific culture file in the sample.

Note: Culture files are present under **“[installed drive]:\Users\[user name]\AppData\Local\Syncfusion\EssentialStudio\X.X.X.X\MVC\Samples\web\Scripts\cultures”.**

ASPX-CS

```
//1036 refers to “fr-FR” culture.
@Html.EJ().Pivot().PivotChart("PivotChart1").Locale("fr-
FR").CommonSeriesOptions(comm => {
comm.Type(SeriesType.Column).Tooltip(tool => { tool.Visible(true);
}).EnableAnimation(true); }).Size(size =>
size.Height("350px").Width("100%")).PrimaryXAxis(primaryXAxis =>
primaryXAxis.Title(title => title.Text("Date -
Fiscal")).LabelRotation(0)).PrimaryYAxis(primaryYAxis =>
primaryYAxis.Title(title => title.Text("Internet Sales
Amount"))).Legend(legend =>
legend.Visible(true).RowCount(2)).ClientSideEvents(
oEve => { oEve.Load("loadTheme");
}).IsResponsive(true).DataSource(dataSource =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();})
.Data("https://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1036;").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works"))
```

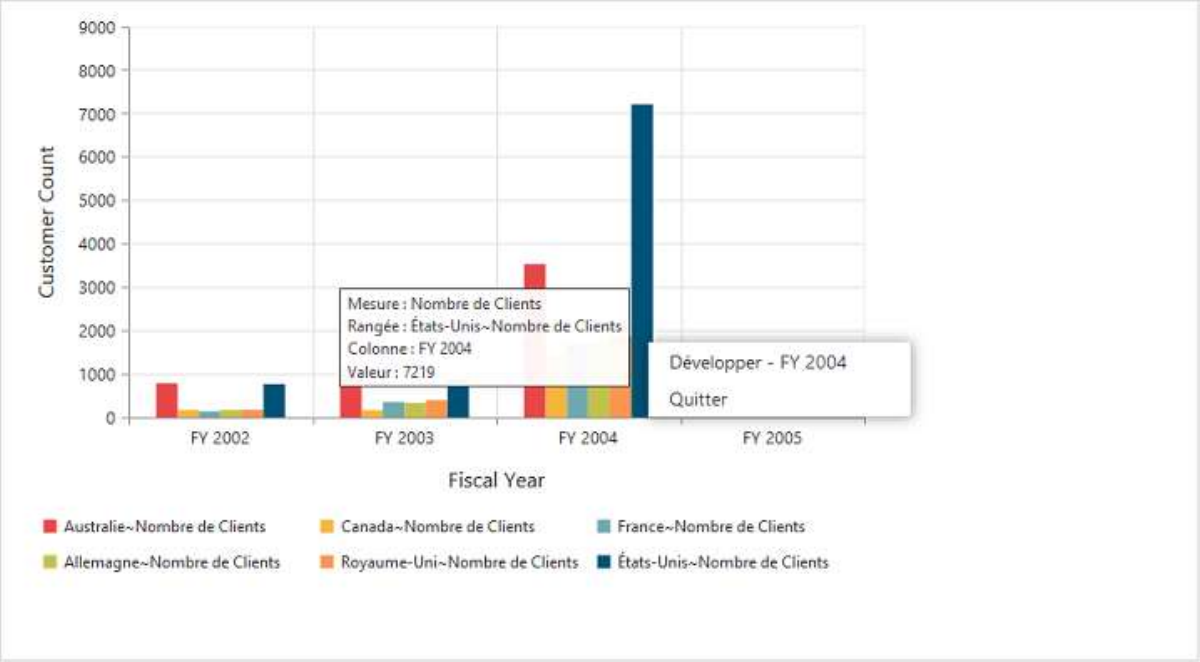
Localization and Globalization of Cube Info

Content displayed within the PivotChart control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

- To get the localized string based on different cultures, from OLAP Cube, we need to set **"Locale Identifier"** in the connection string to a specific culture.
- To bind the globalized content in PivotGrid control, we need to set **"Culture"** and **"OverrideDefaultFormatStrings"** properties in OlapDataManager class to a specific culture.

C#

```
//1036 refers to "fr-FR" culture.
string connectionString = "Data Source=localhost; Initial
Catalog=Adventure Works DW; Locale Identifier=1036;";
DataManager = new OlapDataManager(connectionString);
DataManager.Culture = new System.Globalization.CultureInfo(1036);
DataManager.OverrideDefaultFormatStrings = true;
```



How to

Configure pivot chart through the model properties

There comes a time when you want to set pivot chart properties other than using PivotChartPropertiesBuilder i.e., in the Controller side. At that time, you can pass PivotChartProperties model instance as a parameter to the pivot chart helper method.

We can set properties to pivot chart control in server-side using the PivotChartProperties class and it is used in the view page with the help of pivot chart helper overload.

In addition to the id parameter of pivot chart helper method, we can also pass PivotChartProperties model as another parameter to the pivot chart helper method.

Relational

The following code example explains how to render the pivot chart control in relational mode.

RAZOR

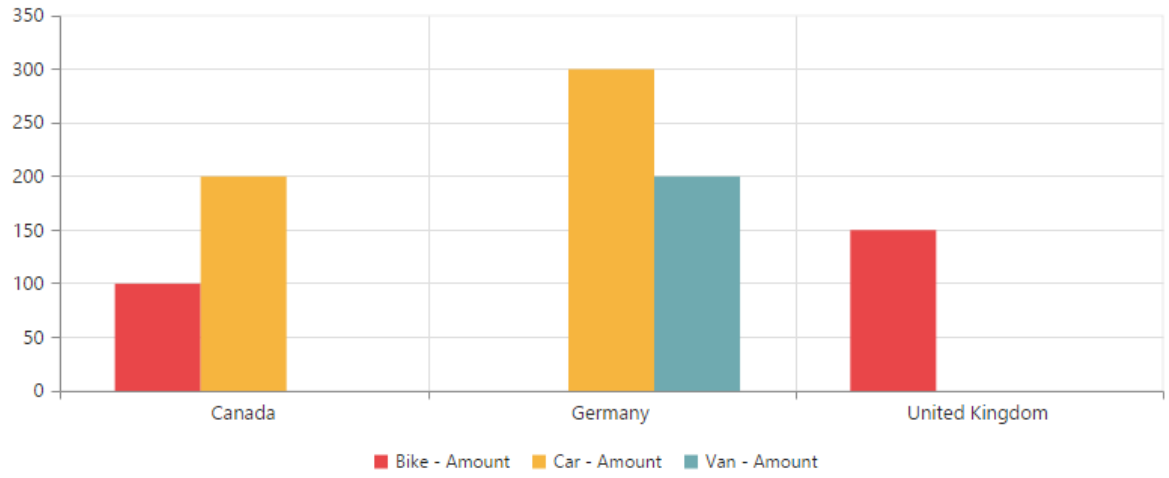
```
@model Syncfusion.JavaScript.Models.PivotChartProperties
@(Html.EJ().Pivot().PivotChart("PivotChart", Model))
<script type="text/javascript">
function load(args) {
args.model.dataSource.data = pivot_dataset;
}
</script>
```

C#

```
namespace PivotChart
{
public class PivotChartController: Controller
{
public ActionResult PivotChartFeatures()
{
Syncfusion.JavaScript.Models.PivotChartProperties pivotchart = new
Syncfusion.JavaScript.Models.PivotChartProperties();
PivotDataSource pivotchartDS = new PivotDataSource();
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "Country", FieldCaption = "Country"
});
rows.Add(new Field() { FieldName = "State", FieldCaption = "State" });
pivotchartDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "Product", FieldCaption = "Product"
});
pivotchartDS.Columns = cols;
pivotchart.Load = "load";
List<Field> values = new List<Field>();
values.Add(new Field() { FieldName = "Amount" });
pivotchartDS.Values = values;
pivotchart.DataSource = pivotchartDS;
return View(pivotchart);
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

As a result of the previous code example, the pivot chart will be displayed as shown below:



OLAP

The following code example explains how to render the pivot chart control in OLAP mode.

RAZOR

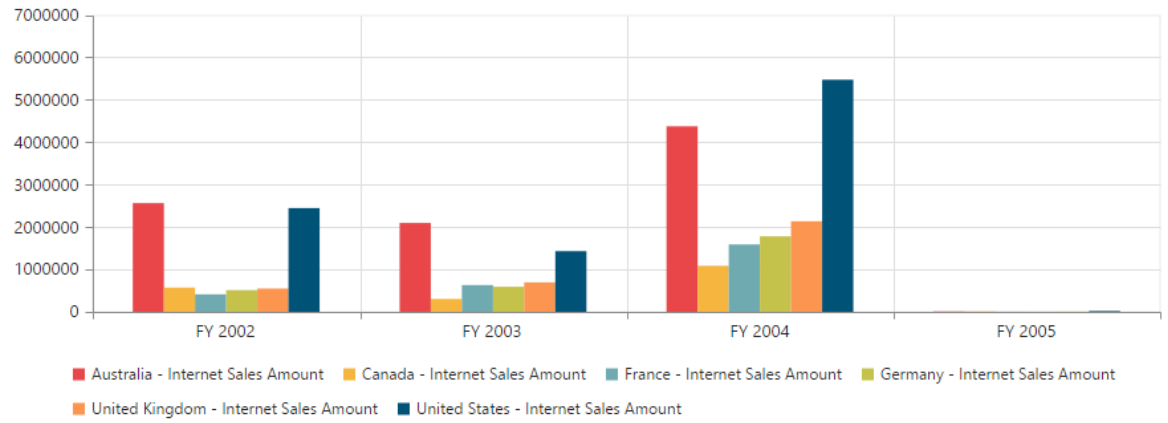
```
@model Syncfusion.JavaScript.Models.PivotChartProperties
@(Html.EJ().Pivot().PivotChart("PivotChart", Model))
```

C#

```
namespace PivotChart
{
public class PivotChartController: Controller
{
public ActionResult PivotChartFeatures()
{
Syncfusion.JavaScript.Models.PivotChartProperties pivotchart = new
Syncfusion.JavaScript.Models.PivotChartProperties();
PivotDataSource pivotchartDS = new PivotDataSource();
pivotchartDS.Data = "https://bi.syncfusion.com/olap/msmdpump.dll";
pivotchartDS.Cube = "Adventure Works";
pivotchartDS.Catalog = "Adventure Works DW 2008 SE";
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "[Date].[Fiscal]" });
pivotchartDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "[Customer].[Customer Geography]" });
pivotchartDS.Rows = cols;
List<MeasuresItems> measures = new List<MeasuresItems>();
measures.Add(new MeasuresItems() { FieldName = "[Measures].[Internet
Sales Amount]" });
List<Field> values = new List<Field>();
values.Add(new Field() { Measures = measures, Axis =
Syncfusion.JavaScript.AxisName.Column });
pivotchartDS.Values = values;
pivotchart.DataSource = pivotchartDS;
return View(pivotchart);
```

```
}
}
}
```

As a result of the previous code example, the pivot chart will be displayed as shown below:



PivotClient

PivotClient

The **PivotClient** control for ASP.NET MVC provides support to bind both OLAP and Relational data sources. You can visualize the results in graphical and tabular formats (chart and grid).The **PivotClient** control is also used to create and edit a report on-the-fly that is stored for later use.

Key Features

The key features of the **PivotClient** control is as follows:

- **Data Source:** Binds the **PivotClient** control with Relational data source and with Microsoft Analysis Services databases and **XML/A** data sources.
- **Axis Element Builder:** Allows building an **OLAP/Pivot** report by placing elements in the different axes of the **PivotClient**. There are four axes that are supported namely column, row, slicer and values.
- **Member Editor:** Lists out the members of the selected field for Relational data source and members of the selected hierarchy in a tree-view-like structure for OLAP data source.
- **Toolbar:** The toolbar of the **PivotClient** component provide options for exporting and manipulating the **OLAP/Pivot** report such as add report, new report, remove report, etc.,
- **Cube Dimension Browser:** A tree-view-like structure that organizes the dimensions, measures and named sets from the selected OLAP cube into independent logical groups.
- **Field List:** List which explores the fields available in Relational data source connected.
- **Visualization Controls:** For a diagrammatic representation of multidimensional data.

Getting Started

Getting Started

Important

Copyright © 2001 - 2020 Syncfusion Inc.

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotClient and Relational datasource (Client Mode)

This section covers the information that you need to know to populate a simple PivotClient with Relational data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotClientDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.Export
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in Web.config files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in Web.config files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS References

The scripts and style sheets that are mandatorily required to render PivotClient control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js
4. jsrender.min.js

Scripts and style sheets are referred under the <head> tag in _Layout.cshtml file which is found inside Views > Shared folder.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotClient

Before initializing, empty the contents of Index.cshtml file under Views > Home folder and add the following codes.

HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("Relational
Browser")
```

Populate PivotClient With Data

Let us now see how to populate the PivotClient control using a sample JSON data as shown below.

HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("Relational
Browser").ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad"))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = [
{ Amount: 100, Country: "Canada", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Alberta" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "British Columbia" },
{ Amount: 300, Country: "Canada", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Brunswick" },
{ Amount: 150, Country: "Canada", Date: "FY 2008", Product: "Bike",
Quantity: 3, State: "Manitoba" },
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 4, State: "Ontario" },
{ Amount: 100, Country: "Canada", Date: "FY 2007", Product: "Van",
Quantity: 1, State: "Quebec" },
{ Amount: 200, Country: "France", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Charente-Maritime" },
{ Amount: 250, Country: "France", Date: "FY 2006", Product: "Van",
Quantity: 4, State: "Essonne" },
{ Amount: 300, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Garonne (Haute)" },
{ Amount: 150, Country: "France", Date: "FY 2008", Product: "Van",
Quantity: 2, State: "Gers" },
{ Amount: 200, Country: "Germany", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "Bayern" },
{ Amount: 250, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Brandenburg" },
{ Amount: 150, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 4, State: "Hamburg" },
{ Amount: 200, Country: "Germany", Date: "FY 2008", Product: "Bike",
Quantity: 4, State: "Hessen" },
{ Amount: 150, Country: "Germany", Date: "FY 2007", Product: "Van",
Quantity: 3, State: "Nordrhein-Westfalen" },
{ Amount: 100, Country: "Germany", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Saarland" },
{ Amount: 150, Country: "United Kingdom", Date: "FY 2008", Product:
"Bike", Quantity: 5, State: "England" },
{ Amount: 250, Country: "United States", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Alabama" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product: "Van",
Quantity: 4, State: "California" },
{ Amount: 100, Country: "United States", Date: "FY 2006", Product:
"Bike", Quantity: 2, State: "Colorado" },
{ Amount: 150, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "New Mexico" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product:
"Bike", Quantity: 4, State: "New York" },
{ Amount: 250, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "North Carolina" },
{ Amount: 300, Country: "United States", Date: "FY 2007", Product: "Van",
Quantity: 4, State: "South Carolina" }
]
}
</script>
```

The JSON data is set to the **"data"** property present inside the **"dataSource"** object. **"dataSource"** object allows us to set both datasource as well as the fields that needs to be displayed in the row, column, value and filter section of the PivotClient control.

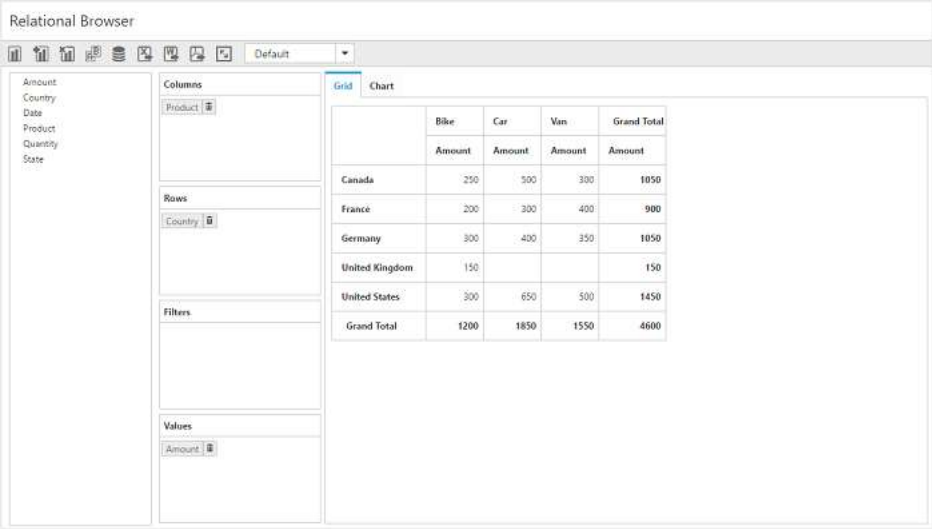
HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("Relational
Browser").ClientSideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();}).Columns(column
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
s => { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();}))
```

Now, PivotClient is rendered with PivotChart and PivotGrid with "Country" field in Row, "Product" field in Column and "Amount" field in Value section.



Creating a simple application with PivotClient and Relational datasource (Server Mode)

This section covers the information required to create a simple PivotClient bound to Relational datasource.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotClientDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referenced assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Linq.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Olap.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Compression.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.PivotAnalysis.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
```

```
<add assembly="Syncfusion.Pdf.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.XlsIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.DocIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in Web.config files available inside Views folder and also at the root of the application.

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotClient control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js
4. jsrender.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

HTML

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("Relational
Browser").Url(Url.Content("/Relational"))
```

The **“Url”** property in PivotClient control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotClient control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Relational". If WCF service is used as endpoint, the URL would look like "/RelationalService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in an existing MVC Web Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WebAPI Controller Class** and name it **“RelationalController.cs”**, click **Add**.

Now, WebAPI controller is added to the application successfully with the file **“RelationalController.cs”**.

Note: While adding WebAPI Controller Class, name it with the suffix ‘Controller’ which is mandatory. For example, in this demo the controller is named as “RelationalController”.

Copyright © 2001 - 2020 Syncfusion Inc.

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside RelationalController.cs file.

C#

```
namespace PivotClientDemo
{
public class RelationalController: ApiController
{
}
}
```

Adding the List of Namespaces

The following are the list of namespaces to be added on top of the main class inside RelationalController.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlServerCe;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Web;
using System.Web.Http;
using System.Web.Script.Serialization;
namespace PivotClientDemo
{
public class RelationalController : ApiController
{
}
}
```

Datasource Initialization

A simple collection is provided as a datasource for the PivotClient in this demo section. This datasource is placed inside a separate class “ProductSales” in RelationalController.cs file. Refer to the following code example.

C#

```
namespace PivotClientDemo {
.....
.....
internal class ProductSales {
public string Product {
get;
set;
}
public string Date {
get;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
set;
}
public string Country {
get;
set;
}
public string State {
get;
set;
}
public int Quantity {
get;
set;
}
public double Amount {
get;
set;
}
public static ProductSalesCollection GetSalesData() {
/// Geography
string[] countries = new string[] {
"Australia",
"Canada",
"France",
"Germany",
"United Kingdom",
"United States"
};
string[] ausStates = new string[] {
"New South Wales",
"Queensland",
"South Australia",
"Tasmania",
"Victoria"
};
string[] canadaStates = new string[] {
"Alberta",
"British Columbia",
"Brunswick",
"Manitoba",
"Ontario",
"Quebec"
};
string[] franceStates = new string[] {
"Charente-Maritime",
"Essonne",
"Garonne (Haute)",
"Gers",
};
string[] germanyStates = new string[] {
"Bayern",
"Brandenburg",
"Hamburg",
"Hessen",
"Nordrhein-Westfalen",
"Saarland"
};
```

```
string[] ukStates = new string[] {
"England"
};
string[] ussStates = new string[] {
"New York",
"North Carolina",
"Alabama",
"California",
"Colorado",
"New Mexico",
"South Carolina"
};
/// Time
string[] dates = new string[] {
"FY 2005",
"FY 2006",
"FY 2007",
"FY 2008",
"FY 2009"
};
/// Products
string[] products = new string[] {
"Bike",
"Van",
"Car"
};
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++) {
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product) {
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country) {
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString() {
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection: List < ProductSales > {}
}
}
```

Service methods in WebAPI Controller

Define the service methods inside RelationalController class, found inside RelationalController.cs file, created while adding WebAPI Controller Class to the Application.

C#

```
namespace PivotClientDemo {
public class RelationalController: ApiController
{
PivotClient pivotClient = new PivotClient();
PivotChart pivotChart = new PivotChart();
PivotGrid pivotGrid = new PivotGrid();
JavaScriptSerializer serializer = new JavaScriptSerializer();
string conStringforDB = "";//Enter appropriate connection string to
connect database for saving and loading operation of reports.
[System.Web.Http.ActionName("InitializeClient")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeClient(Dictionary<string,
object> jsonResult)
{
this.BindData();
```

```
return pivotClient.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), null);
}
[System.Web.Http.ActionName("FetchMembers")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> FetchMembers(Dictionary<string, object>
jsonResult)
{
pivotClient.PopulateData(jsonResult["currentReport"].ToString());
var dict = pivotClient.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["headerTag"].ToString());
return dict;
}
[System.Web.Http.ActionName("DrillChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillChart(Dictionary<string, object>
jsonResult)
{
this.BindData();
this.pivotChart.PivotEngine.PivotRows =
this.pivotClient.PivotReport.PivotRows;
this.pivotChart.PivotEngine.PivotColumns =
this.pivotClient.PivotReport.PivotColumns;
this.pivotChart.PivotEngine.PivotCalculations =
this.pivotClient.PivotReport.PivotCalculations;
return pivotChart.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["drilledSeries"].ToString());
}
[System.Web.Http.ActionName("Filtering")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> Filtering(Dictionary<string, object>
jsonResult)
{
pivotClient.PopulateData(jsonResult["currentReport"].ToString());
var dict = pivotClient.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["filterParams"].ToString());
return dict;
}
[System.Web.Http.ActionName("NodeDropped")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> NodeDropped(Dictionary<string, object>
jsonResult)
{
Dictionary<string, object> dict =
pivotClient.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["args"].ToString());
return dict;
}
[System.Web.Http.ActionName("ToolbarOperations")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> ToolbarOperations(Dictionary<string,
object> jsonResult)
{
Dictionary<string, object> dict =
pivotClient.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["args"].ToString());
return dict;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
[System.Web.Http.ActionName("SaveReportToDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> SaveReportToDB(Dictionary<string,
object> jsonResult)
{
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty;
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["reportName"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(jsonResult["clientReports"].ToString()).ToArray())
;
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Save");
return dictionary;
}
[System.Web.Http.ActionName("RemoveReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> RemoveReportFromDB(Dictionary<string,
object> jsonResult)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["reportName"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
cmd1 = new SqlCeCommand("DELETE FROM ReportsTable WHERE ReportName LIKE
'%" + reportName + "%'", con);
}
}
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Remove");
return dictionary;
}
[System.Web.Http.ActionName("RenameReportInDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> RenameReportInDB(Dictionary<string,
object> jsonResult)
{
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty, renameReport = string.Empty;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["selectedReport"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
renameReport = jsonResult["renameReport"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
cmd1 = new SqlCeCommand("update ReportsTable set ReportName=@RenameReport
where ReportName like '%" + reportName + "%'", con);
}
}
cmd1.Parameters.Add("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Rename");
return dictionary;
}
[System.Web.Http.ActionName("FetchReportListFromDB")]
[System.Web.Http.HttpPost]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public Dictionary<string, object>
FetchReportListFromDB(Dictionary<string, object> jsonResult)
{
string reportNames = string.Empty, currentRptName = string.Empty,
operationalMode = jsonResult["operationalMode"].ToString(), analysisMode
= jsonResult["analysisMode"].ToString();
foreach (System.Data.DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string);
if (currentRptName.IndexOf("##" + operationalMode + "#>>#" +
analysisMode) >= 0)
{
currentRptName = currentRptName.Replace("##" + operationalMode + "#>>#" +
analysisMode, "");
reportNames = reportNames == "" ? currentRptName : reportNames + "__" +
currentRptName;
}
}
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("ReportNameList", reportNames);
dictionary.Add("action", jsonResult["action"].ToString());
return dictionary;
}
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
{
PivotReport report = new PivotReport();
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
string currentRptName = string.Empty;
foreach (DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string).Replace("##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower(), "");
if (currentRptName.Equals(jsonResult["reportName"].ToString()))
{
byte[] reportString = new byte[2 * 1024];
reportString = (row.ItemArray[1] as byte[]);
if (analysisMode.ToLower() == "pivot" && operationalMode.ToLower() ==
"servermode")
dictionary = pivotClient.GetJsonData("LoadReport",
ProductSales.GetSalesData(), Encoding.UTF8.GetString(reportString));
else
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
private void BindData()
{
this.pivotClient.PivotReport.PivotRows.Add(new PivotItem {
FieldMappingName = "Country", FieldHeader = "Country", TotalHeader =
"Total" });
this.pivotClient.PivotReport.PivotColumns.Add(new PivotItem {
FieldMappingName = "Product", FieldHeader = "Product", TotalHeader =
"Total" });
this.pivotClient.PivotReport.PivotCalculations.Add(new
PivotComputationInfo { CalculationName = "Amount", Description =
"Amount", FieldHeader = "Amount", FieldName = "Amount", Format = "C",
SummaryType = Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum
});
}
}
.....
..... // Initialize the datasource
.....
}
```

Configure routing in WebAPIConfig Class

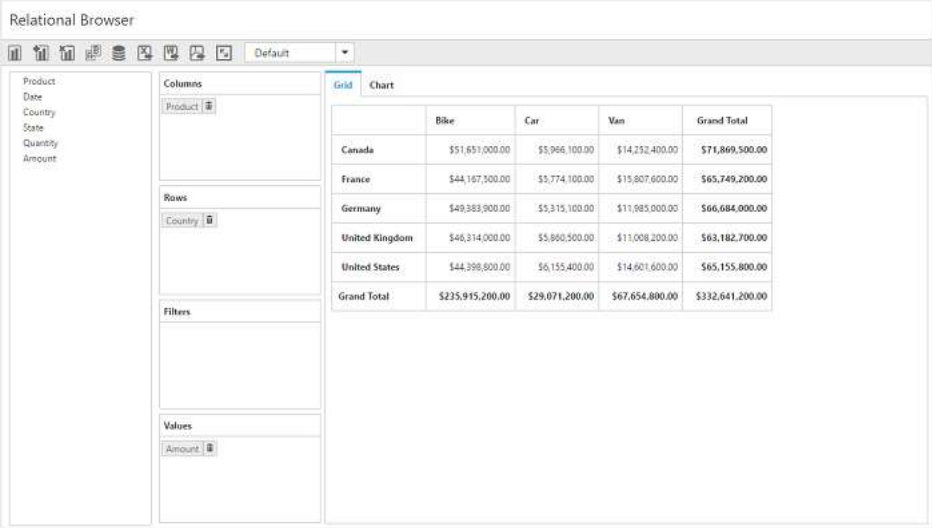
Open the "WebApiConfig.cs" file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, **PivotClient** is rendered with PivotChart and PivotGird with "Country" field in Row, "Product" field in Column and "Amount" field in Value section.

Copyright © 2001 - 2020 Syncfusion Inc.



WCF

This section demonstrates the utilization of WCF service as endpoint binding Relational datasource to a simple PivotClient. For more details on this topic, click here.

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotClient and OLAP datasource (Client Mode)

This section covers the information that you need to know to populate a simple PivotClient with OLAP data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotClientDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.Export
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

Copyright © 2001 - 2020 Syncfusion Inc.

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

Copyright © 2001 - 2020 Syncfusion Inc.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS References

The scripts and style sheets that are mandatorily required to render PivotClient control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js
4. jsrender.min.js

Scripts and style sheets are referred under the <head> tag in _Layout.cshtml file which is found inside Views > Shared folder.

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Initialize PivotClient

Before initializing, empty the contents of Index.cshtml file under Views > Home folder and add the following codes.

HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("OLAP Browser")
```

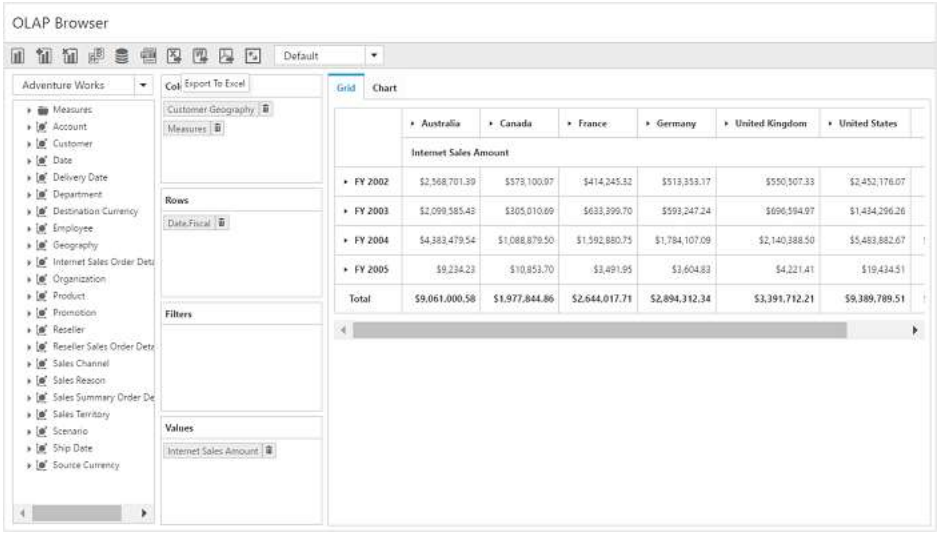
Populate PivotClient With DataSource

Initializes the OLAP datasource for PivotClient control as shown below.

HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("OLAP
Browser").DataSource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();}).Data("https://bi.syncfusion.com/olap/ms
mdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works"))
```

Now, PivotClient is rendered with PivotChart and PivotGrid with "Customer Geography" field in Column, "Fiscal" field in Row and "Internet Sales Amount" field in Value section.



Creating a simple application with PivotClient and OLAP datasource (Server Mode)

This section covers the information required to create a simple PivotClient bound to OLAP datasource.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotClientDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referenced assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

Copyright © 2001 - 2020 Syncfusion Inc.

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Linq.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Olap.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Compression.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.PivotAnalysis.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Pdf.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.XlsIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.DocIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotClient control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js
4. jsrender.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the **head** tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

HTML

```
@using Syncfusion.JavaScript;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div>
@Html.EJ().Pivot().PivotClient("PivotClient1").Title("OLAP
Browser").Url(Url.Content("/Olap"))
</div>
```

The **"Url"** property in PivotClient control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotClient control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Olap". If WCF service is used as endpoint, the URL would look like "/OlapService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in an existing Web Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WebAPI Controller Class** and name it as OlapController.cs, click Add.

Now, WebAPI controller is added to the application successfully with the file **"OlapController.cs"**.

Note: While adding WebAPI Controller Class, name it with the suffix 'Controller' that is mandatory. For example, in this demo the controller is named as **"OlapController"**.

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside OlapController.cs file.

C#

```
namespace PivotClientDemo
{
public class OlapController: ApiController
{
}
}
```

Adding the List of Namespaces

The following are the list of namespaces to be added on top of the main class inside OlapController.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlServerCe;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Http;
using System.Web.Script.Serialization;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace PivotClientDemo
{
public class OlapController: ApiController
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube, PivotClient and JavaScriptSerializer instances are created immediately inside the main class in OlapController.cs file.

C#

```
namespace PivotClientDemo
{
public class OlapController: ApiController
{
PivotClient pivotClientHelper = new PivotClient();
PivotTreeMap treeMapHelper = new PivotTreeMap();
PivotChart chartHelper = new PivotChart();
JavaScriptSerializer serializer = new JavaScriptSerializer();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
string conStringforDB = "";//Enter appropriate connection string to
connect database for saving and loading operation of reports
//Other codes
}
}
```

Service methods in WebAPI Controller

Define the service methods inside "OlapController" class, found inside OlapController.cs file, created while adding WebAPI Controller Class to the Application.

C#

```
namespace PivotClientDemo {
public class OlapController: ApiController
{
PivotClient pivotClientHelper = new PivotClient();
PivotTreeMap treeMapHelper = new PivotTreeMap();
PivotChart chartHelper = new PivotChart();
JavaScriptSerializer serializer = new JavaScriptSerializer();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
string conStringforDB = "";//Enter appropriate connection string to
connect database for saving and loading operation of reports
[System.Web.Http.ActionName("InitializeClient")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeClient(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = null;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["clientParams"].ToString());
}
[System.Web.Http.ActionName("InitializeGrid")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeGrid(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return jsonResult.ContainsKey("gridLayout") ?
pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["gridLayout"].ToString()) :
pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager);
}
[System.Web.Http.ActionName("InitializeChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeChart(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return chartHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager);
}
[System.Web.Http.ActionName("InitializeTreeMap")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeTreeMap(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return treeMapHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager);
}
[System.Web.Http.ActionName("DrillChart")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillChart(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return chartHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["drilledSeries"].ToString());
}
[System.Web.Http.ActionName("DrillTreeMap")]
[System.Web.Http.HttpPost]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public Dictionary<string, object> DrillTreeMap(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return treeMapHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["drillInfo"].ToString());
}
[System.Web.Http.ActionName("FilterElement")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> FilterElement(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["clientParams"].ToString());
}
[System.Web.Http.ActionName("RemoveSplitButton")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> RemoveSplitButton(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["clientParams"].ToString());
}
[System.Web.Http.ActionName("FetchMemberTreeNodes")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> FetchMemberTreeNodes(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["dimensionName"].ToString());
}
[System.Web.Http.ActionName("DrillGrid")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillGrid(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["cellPosition"].ToString(),
jsonResult["headerInfo"].ToString(), jsonResult["layout"].ToString());
}
[System.Web.Http.ActionName("NodeDropped")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> NodeDropped(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["dropType"].ToString(),
jsonResult["nodeInfo"].ToString());
}
[System.Web.Http.ActionName("CubeChanged")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> CubeChanged(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (jsonResult["olapReport"].ToString() != "")
DataManager.SetCurrentReport(Syncfusion.JavaScript.Olap.Utils.Deserialize
OlapReport(jsonResult["olapReport"].ToString()));
if (jsonResult["clientReports"].ToString() != "")
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
dynamic customData =
serializer.Deserialize<dynamic>(jsonResult["customObject"].ToString());
if (customData is Dictionary<string, object> &&
customData.ContainsKey("isPaging"))
{
DataManager.CurrentReport.EnablePaging = true;
DataManager.CurrentReport.PagerOptions.SeriesPageSize = 5;
DataManager.CurrentReport.PagerOptions.CategoricalPageSize = 3;
}
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["cubeName"].ToString(),
jsonResult["clientParams"].ToString());
}
[System.Web.Http.ActionName("MeasureGroupChanged")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> MeasureGroupChanged(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

```
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["measureGroupName"].ToString());
}
[System.Web.Http.ActionName("ToolbarOperations")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> ToolbarOperations(Dictionary<string,
object> jsonResult)
{
var toolbarOperation = "";
dynamic customData = "";
OlapDataManager DataManager = new OlapDataManager(connectionString);
if ((jsonResult["olapReport"]) != null &&
!string.IsNullOrEmpty(jsonResult["olapReport"].ToString()))
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
if ((jsonResult["clientReports"]) != null &&
!string.IsNullOrEmpty(jsonResult["clientReports"].ToString()))
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
toolbarOperation = jsonResult["toolbarOperation"] == null ? null :
jsonResult["toolbarOperation"].ToString();
if (jsonResult.ContainsKey("customObject") &&
(jsonResult["customObject"]) != null &&
!string.IsNullOrEmpty(jsonResult["customObject"].ToString()))
customData =
serializer.Deserialize<dynamic>(jsonResult["customObject"].ToString());
if ((customData is Dictionary<string, object> &&
customData.ContainsKey("isPaging") && (toolbarOperation == "Add Report"
|| toolbarOperation == "New Report")))
{
DataManager.CurrentReport.EnablePaging = true;
DataManager.CurrentReport.PagerOptions.SeriesPageSize = 5;
DataManager.CurrentReport.PagerOptions.CategoricalPageSize = 3;
}
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, toolbarOperation, jsonResult["clientInfo"].ToString());
}
[System.Web.Http.ActionName("MemberExpanded")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> MemberExpanded(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (!string.IsNullOrEmpty(jsonResult["olapReport"].ToString()))
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
if (!string.IsNullOrEmpty(jsonResult["clientReports"].ToString()))
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, Convert.ToBoolean(jsonResult["checkedStatus"].ToString()),
jsonResult["parentNode"].ToString(), jsonResult["tag"].ToString(),
jsonResult["dimensionName"].ToString(),
jsonResult["cubeName"].ToString());
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
[System.Web.Http.ActionName("UpdateReport")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> UpdateReport(Dictionary<string, object>
jsonResult)
{
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
jsonResult["clientParams"].ToString(),
jsonResult["olapReport"].ToString(),
jsonResult["clientReports"].ToString());
}
[System.Web.Http.ActionName("SaveReportToDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> SaveReportToDB(Dictionary<string,
object> jsonResult)
{
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty;
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["reportName"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(jsonResult["clientReports"].ToString()).ToArray())
;
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Save");
return dictionary;
}
[System.Web.Http.ActionName("RemoveReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> RemoveReportFromDB(Dictionary<string,
object> jsonResult)
{
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["reportName"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
cmd1 = new SqlCeCommand("DELETE FROM ReportsTable WHERE ReportName LIKE
'%" + reportName + "%'", con);
}
}
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Remove");
return dictionary;
}
[System.Web.Http.ActionName("RenameReportInDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> RenameReportInDB(Dictionary<string,
object> jsonResult)
{
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty, renameReport = string.Empty;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["selectedReport"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
renameReport = jsonResult["renameReport"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
cmd1 = new SqlCeCommand("update ReportsTable set ReportName=@RenameReport
where ReportName like '%" + reportName + "%'", con);
}
}
cmd1.Parameters.Add("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Rename");
return dictionary;
}
[System.Web.Http.ActionName("FetchReportListFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object>
FetchReportListFromDB(Dictionary<string, object> jsonResult)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string reportNames = string.Empty, currentRptName = string.Empty,
operationalMode = jsonResult["operationalMode"].ToString(), analysisMode
= jsonResult["analysisMode"].ToString();
foreach (System.Data.DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string);
if (currentRptName.IndexOf("##" + operationalMode + "#>>#" +
analysisMode) >= 0)
{
currentRptName = currentRptName.Replace("##" + operationalMode + "#>>#" +
analysisMode, "");
reportNames = reportNames == "" ? currentRptName : reportNames + "__" +
currentRptName;
}
}
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("ReportNameList", reportNames);
dictionary.Add("action", jsonResult["action"].ToString());
return dictionary;
}
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
{
PivotReport report = new PivotReport();
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
string currentRptName = string.Empty;
foreach (DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string).Replace("##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower(), "");
if (currentRptName.Equals(jsonResult["reportName"].ToString()))
{
byte[] reportByte = new byte[2 * 1024];
reportByte = (row.ItemArray[1] as byte[]);
if (operationalMode.ToLower() == "servermode" && analysisMode == "olap")
{
var repCol = Encoding.UTF8.GetString(reportByte);
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (repCol.IndexOf("<?xml version") == 0)
{
var reportString = "";
reportString =
Syncfusion.JavaScript.Olap.Utils.CompressData(row.ItemArray[1] as
byte[]);
DataManager.Reports =
pivotClientHelper.DeserializedReports(reportString);
DataManager.SetCurrentReport(DataManager.Reports[0]);
return pivotClientHelper.GetJsonData("toolbarOperation", DataManager,
"Load Report", jsonResult["reportName"].ToString());
}
else
{
dynamic customData = serializer.Deserialize<dynamic>(repCol.ToString());
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataManager.Reports =
pivotClientHelper.DeserializedReports(customData[customData[customData.Le
ngth - 1]["cubeIndex"]]["Reports"]);
DataManager.SetCurrentReport(DataManager.Reports[customData[customData[cu
stomData.Length - 1]["cubeIndex"]]["ReportIndex"]]);
dictionary = pivotClientHelper.GetJsonData("toolbarOperation",
DataManager, "Load Report", jsonResult["reportName"].ToString());
dictionary.Add("Collection", repCol);
}
}
}
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
pivotClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
[System.Web.Http.ActionName("GetMDXQuery")]
[System.Web.Http.HttpPost]
public string GetMDXQuery(Dictionary<string, object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
return DataManager.GetMDXQuery();
}
[System.Web.Http.ActionName("ToggleAxis")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> ToggleAxis(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
DataManager.Reports =
pivotClientHelper.DeserializedReports(jsonResult["clientReports"].ToStrin
g());
DataManager.ToggleAxis(DataManager.CurrentReport);
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["clientReports"].ToString());
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
[System.Web.Http.ActionName("Paging")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> Paging(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.SetPaging(jsonResult["curren
tReport"].ToString(), jsonResult["pagingInfo"].ToString()));
return pivotClientHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["layout"].ToString());
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport() { Name = "Default Report" };
olapReport.CurrentCubeName = "Adventure Works";
MeasureElements measureElement = new MeasureElements();
measureElement.Elements.Add(new MeasureElement { UniqueName =
"[Measures].[Customer Count]" });
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
olapReport.SeriesElements.Add(dimensionElementRow);
olapReport.CategoricalElements.Add(measureElement);
return olapReport;
}
}
}
```

Configure routing in WebAPIConfig Class

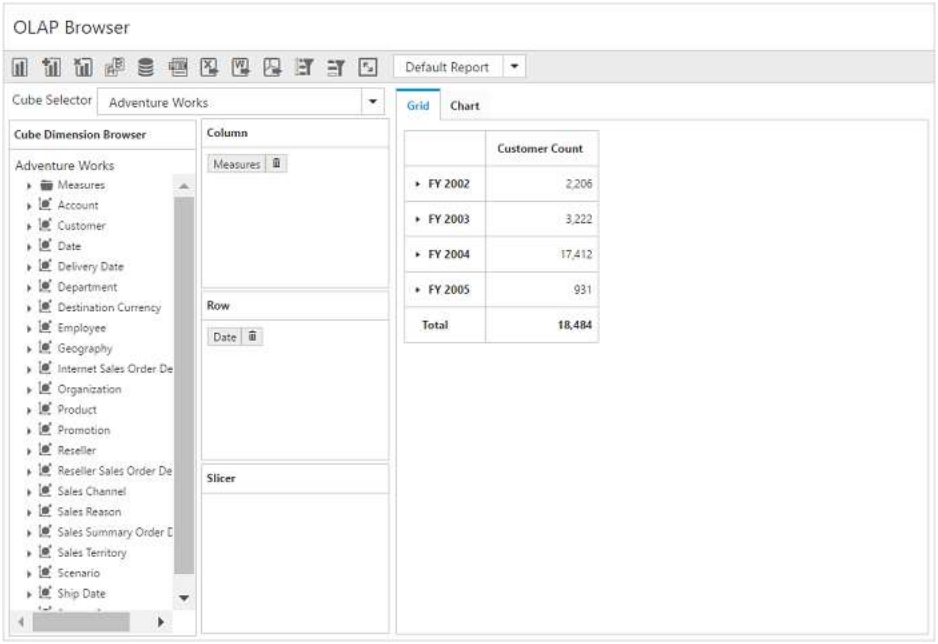
Open the "WebAPIConfig.cs" file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, **PivotClient** is rendered with PivotChart and PivotGrid showing Customer Count over a period of fiscal years.

Copyright © 2001 - 2020 Syncfusion Inc.



WCF

This section demonstrates the utilization of WCF service as endpoint binding OLAP datasource to a simple PivotClient. For more details on this topic, click here.

Data Binding

Data Binding

Binding PivotClient to Collection

This section demonstrates binding of a collection to the PivotClient control as datasource. For more information on this datasource refer the following links.

If you are using WebAPI controller, refer the "Datasource Initialization" section under the following link.

If you are using WCF Service, refer the "Datasource Initialization" section below.

WCF

Adding a WCF Service

To add a WCF service in an existing MVC Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as RelationalService.svc, click **Add.**

Now, WCF service is added into the application successfully that comprises of the following files. The utilization of these files is explained in the immediate sections

- RelationalService.svc
- RelationalService.svc.cs
- IRelationalService.cs

Configuring WCF Service Class

Copyright © 2001 - 2020 Syncfusion Inc.

Remove the **"DoWork"** method present inside both RelationalService.svc.cs and IRelationalService.cs files. Next, add **"AspNetCompatibilityRequirements"** attribute on top of main class present inside RelationalService.svc.cs and set **"RequirementsMode"** value to **"Allowed"**.

C#

```
namespace PivotClientDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside RelationalService.svc.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
using System;
using System.Collections.Generic;
using System.Data;
using System.ServiceModel.Activation;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using System.Data.SqlServerCe;
using Syncfusion.Olap.Manager;
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Web;
using System.Web.Http;
using System.Web.Script.Serialization;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
namespace PivotClientDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

Datasource Initialization

A simple collection is provided as a datasource for our PivotClient in this demo section. This datasource is placed inside a separate class named “ProductSales” in RelationalService.svc.cs file. Refer to the following code example.

C#

```
namespace PivotClientDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
……
……
}
internal class ProductSales
{
public string Product { get; set; }
public string Date { get; set; }
public string Country { get; set; }
public string State { get; set; }
public int Quantity { get; set; }
public double Amount { get; set; }
public static ProductSalesCollection GetSalesData()
{
/// Geography
string[] countries = new string[] { "Australia", "Canada", "France",
"Germany", "United Kingdom", "United States" };
string[] ausStates = new string[] { "New South Wales", "Queensland",
"South Australia", "Tasmania", "Victoria" };
string[] canadaStates = new string[] { "Alberta", "British Columbia",
"Brunswick", "Manitoba", "Ontario", "Quebec" };
string[] franceStates = new string[] { "Charente-Maritime", "Essonne",
"Garonne (Haute)", "Gers", };
string[] germanyStates = new string[] { "Bayern", "Brandenburg",
"Hamburg", "Hessen", "Nordrhein-Westfalen", "Saarland" };
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string[] ukStates = new string[] { "England" };
string[] ussStates = new string[] { "New York", "North Carolina",
"Alabama", "California", "Colorado", "New Mexico", "South Carolina" };
/// Time
string[] dates = new string[] { "FY 2005", "FY 2006", "FY 2007", "FY
2008", "FY 2009" };
/// Products
string[] products = new string[] { "Bike", "Van", "Car" };
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++)
{
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product)
{
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country)
{
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString()
{
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection : List<ProductSales>
{
}
}
}
```

Service methods in WCF Service

First, declare the service methods inside **IRelationalService** interface, found in IRelationalService.cs file created while adding WCF Service to the Application.

C#

```
namespace PivotClientDemo {
[ServiceContract]
public interface IRelationalService
{
[OperationContract]
Dictionary<string, object> InitializeClient(string action);
[OperationContract]
Dictionary<string, object> FetchMembers(string action, string
currentReport, string customObject, string headerTag);
[OperationContract]
Dictionary<string, object> DrillChart(string action, string
drilledSeries);
[OperationContract]
Dictionary<string, object> Filtering(string action, string filterParams,
string currentReport, string customObject);
[OperationContract]
Dictionary<string, object> NodeDropped(string action, string args);
[OperationContract]
Dictionary<string, object> ToolbarOperations(string action, string args);
[OperationContract]
Dictionary<string, object> SaveReportToDB(string reportName, string
operationalMode, string analysisMode, string olapReport, string
clientReports);
[OperationContract]
Dictionary<string, object> RemoveReportFromDB(string reportName, string
operationalMode, string analysisMode);
[OperationContract]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Dictionary<string, object> RenameReportInDB(string selectedReport, string
renameReport, string operationalMode, string analysisMode);
[OperationContract]
Dictionary<string, object> LoadReportFromDB(string reportName, string
operationalMode, string analysisMode, string olapReport, string
clientReports);
[OperationContract]
Dictionary<string, object> FetchReportListFromDB(string action, string
operationalMode, string analysisMode);
[OperationContract]
void Export(System.IO.Stream stream);
}
}
```

Then, elaborate the service methods inside the main class, found in RelationalService.svc.cs file.

C#

```
namespace PivotClientDemo {
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService: IRelationalService
{
PivotClient pivotClient = new PivotClient();
PivotChart pivotChart = new PivotChart();
PivotGrid pivotGrid = new PivotGrid();
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
JavaScriptSerializer serializer = new JavaScriptSerializer();
public Dictionary<string, object> InitializeClient(string action)
{
this.BindData();
return pivotClient.GetJsonData(action, ProductSales.GetSalesData(),
null);
}
public Dictionary<string, object> FetchMembers(string action, string
currentReport, string customObject, string headerTag)
{
pivotClient.PopulateData(currentReport);
return pivotClient.GetJsonData(action, ProductSales.GetSalesData(),
headerTag);
}
public Dictionary<string, object> DrillChart(string action, string
drilledSeries)
{
this.BindData();
this.pivotChart.PivotEngine.PivotRows =
this.pivotClient.PivotReport.PivotRows;
this.pivotChart.PivotEngine.PivotColumns =
this.pivotClient.PivotReport.PivotColumns;
this.pivotChart.PivotEngine.PivotCalculations =
this.pivotClient.PivotReport.PivotCalculations;
return pivotChart.GetJsonData(action, ProductSales.GetSalesData(),
drilledSeries);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public Dictionary<string, object> Filtering(string action, string
filterParams, string currentReport, string customObject)
{
pivotClient.PopulateData(currentReport);
return pivotClient.GetJsonData(action, ProductSales.GetSalesData(),
filterParams);
}
public Dictionary<string, object> NodeDropped(string action, string args)
{
return pivotClient.GetJsonData(action, ProductSales.GetSalesData(),
args);
}
public Dictionary<string, object> ToolbarOperations(string action, string
args)
{
return pivotClient.GetJsonData(action, ProductSales.GetSalesData(),
args);
}
public Dictionary<string, object> SaveReportToDB(string reportName,
string operationalMode, string analysisMode, string olapReport, string
clientReports)
{
reportName = reportName + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(clientReports).ToArray());
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Save");
return dictionary;
}
public Dictionary<string, object> RemoveReportFromDB(string reportName,
string operationalMode, string analysisMode)
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
```

```
con.Open();
reportName = reportName + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
cmd1 = new SqlCeCommand("DELETE FROM ReportsTable WHERE ReportName LIKE
'%" + reportName + "%'", con);
}
}
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Remove");
return dictionary;
}
public Dictionary<string, object> RenameReportInDB(string selectedReport,
string renameReport, string operationalMode, string analysisMode)
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
selectedReport = selectedReport + "##" + operationalMode.ToLower() +
"#>>#" + analysisMode.ToLower();
renameReport = renameReport + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(selectedReport))
{
cmd1 = new SqlCeCommand("update ReportsTable set ReportName=@RenameReport
where ReportName like '%" + selectedReport + "%'", con);
}
}
cmd1.Parameters.Add("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Rename");
return dictionary;
}
public Dictionary<string, object> FetchReportListFromDB(string action,
string operationalMode, string analysisMode)
{
string reportNames = string.Empty;
string currentRptName = string.Empty;
foreach (System.Data.DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string);
if (currentRptName.IndexOf("##" + operationalMode + "#>>#" +
analysisMode) >= 0)
{
currentRptName = currentRptName.Replace("##" + operationalMode + "#>>#" +
analysisMode, "");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
reportNames = reportNames == "" ? currentRptName : reportNames + "__" +
currentRptName;
}
}
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("ReportNameList", reportNames);
dictionary.Add("action", action);
return dictionary;
}
public Dictionary<string, object> LoadReportFromDB(string reportName,
string operationalMode, string analysisMode, string olapReport, string
clientReports)
{
PivotReport report = new PivotReport();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
string currentRptName = string.Empty;
foreach (DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string).Replace("##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower(), "");
if (currentRptName.Equals(reportName))
{
if (operationalMode.ToLower() == "servermode" && analysisMode == "olap")
{
var reportString = "";
OlapDataManager DataManager = new OlapDataManager();
reportString = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
DataManager.Reports = pivotClient.DeserializedReports(reportString);
DataManager.SetCurrentReport(DataManager.Reports[0]);
return pivotClient.GetJsonData("toolbarOperation", DataManager, "Load
Report", reportName);
}
else
{
byte[] reportString = new byte[2 * 1024];
reportString = (row.ItemArray[1] as byte[]);
if (analysisMode.ToLower() == "pivot" && operationalMode.ToLower() ==
"servermode")
dictionary = pivotClient.GetJsonData("LoadReport",
ProductSales.GetSalesData(), Encoding.UTF8.GetString(reportString));
else
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
}
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
private void BindData()
{
this.pivotClient.PivotReport.PivotRows.Add(new PivotItem {
FieldMappingName = "Country", FieldHeader = "Country", TotalHeader =
"Total" });
this.pivotClient.PivotReport.PivotColumns.Add(new PivotItem {
FieldMappingName = "Product", FieldHeader = "Product", TotalHeader =
"Total" });
this.pivotClient.PivotReport.PivotCalculations.Add(new
PivotComputationInfo { CalculationName = "Amount", Description =
"Amount", FieldHeader = "Amount", FieldName = "Amount", Format = "C",
SummaryType = Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum
});
}
}
.....
..... // Initialize the datasource
.....
}
```

Configuring Web Configuration File

The services could be exposed through the properties, binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IRelationalService contract and hence it is PivotClientDemo.IRelationalService.
- Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
- BehaviorConfiguration: This property contains the name of the behavior to be used in the endpoint

The endpointBehaviors are illustrated as follows

XML

```
<system.serviceModel>
……
……
<services>
```

```
<service name="PivotClientDemo.RelationalService">
<endpoint address=""
behaviorConfiguration="PivotClientDemo.RelationalServiceAspNetAjaxBehavio
r" binding="webHttpBinding" contract="PivotClientDemo.IRelationalService"
/>
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotClientDemo.RelationalServiceAspNetAjaxBehavior">
<enableWebScript />
</behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Note: In this example, **"PivotClientDemo"** indicates the name and root namespace of the Web Application created in Visual Studio IDE and "RelationalService" indicates the name of the WCF service created.

Configuring routing file

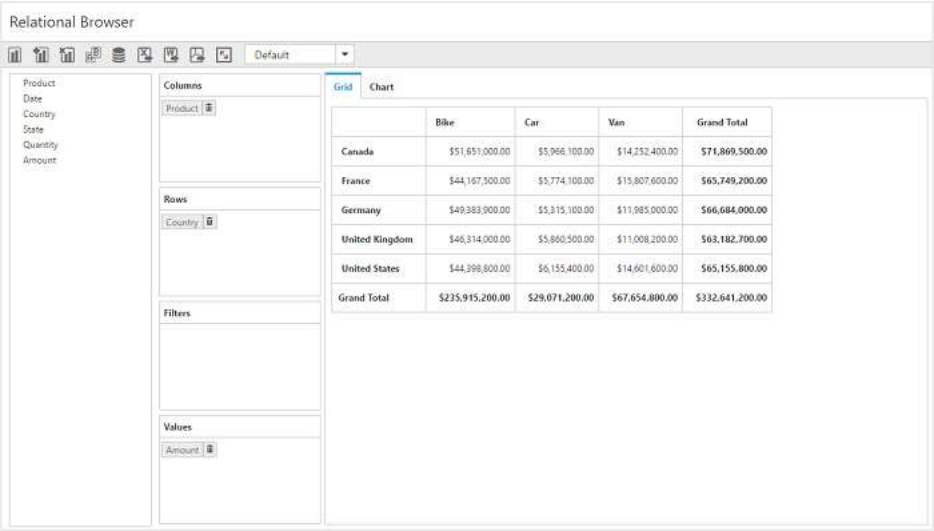
Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
//NOTE: In the following highlighted lines _wcf_ is just the folder name
inside which service files(*.svc) are present.
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new {
controller = "Home", action = "Index", id = UrlParameter.Optional
},
namespaces: new [] {
"PivotClientDemo.Controllers"
}
);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Now, **PivotClient** is rendered with PivotChart and PivotGrid with "Country" field in Row, "Product" field in Column and "Amount" field in Value section.



Data Binding

Binding PivotClient to Offline Cube

To connect an OLAP Cube available in local machine, set the physical path of the Cube in the connection string. The following code example illustrates the same.

C#

```
string connectionString = @"DataSource= system
drive:\OfflineCube\Adventure_Works_Ext.cub; Provider = MSOLAP;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotClient to Cube in local SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in local machine, set the server name and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = "Data source=localhost; Initial
Catalog=Adventure Works DW;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotClient to Cube in online SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in online server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
static string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotClient to Cube in online Mondrian Server

To connect an OLAP Cube available in Mondrian Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = @"Data Source =
https://localhost:8080/mondrian/xmla; Initial Catalog =FoodMart;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName =
Syncfusion.Olap.DataProvider.Providers.Mondrian;
```

Binding PivotClient to Cube in online ActivePivot Server

To connect an OLAP Cube available in ActivePivot Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString=@"Data Source =
https://localhost:8080/cva_s/xmla; Initial Catalog = CVAS;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName=Syncfusion.Olap.DataProvider.Provid
ers.ActivePivot;
```

WCF

Adding a WCF Service

To add a WCF service in an existing MVC web application, right-click on project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as **OlapService.svc**, click Add.

Now, WCF service is added into your application successfully and it comprises of the following files. The utilization of these files is explained in the immediate sections.

- OlapService.svc
- OlapService.svc.cs
- IOlapService.svc.cs

Configuring WCF Service Class

Remove the **“DoWork”** method present inside both OlapService.svc.cs and IOlapService.cs files. Next, add **“AspNetCompatibilityRequirements”** attribute on top of main class present inside OlapService.svc.cs file and set **“RequirementsMode”** value to **“Allowed”**.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace PivotClientDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapService.svc.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.ServiceModel.Activation;
using System.Data.SqlServerCe;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using System.IO;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Http;
using System.Web.Script.Serialization;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
namespace PivotClientDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube, PivotClient and "JavaScriptSerializer" instances are created immediately inside the main class in OlapService.svc.cs file.

C#

```
namespace PivotClientDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
PivotClient pivotClientHelper = new PivotClient();
PivotTreeMap treemapHelper = new PivotTreeMap();
PivotChart chartHelper = new PivotChart();
JavaScriptSerializer serializer = new JavaScriptSerializer();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
string conStringforDB = "";//Enter appropriate connection string to
connect database for saving and loading operation of reports
//Other codes
}
}
```

Service methods in WCF Service

First, define the service methods inside **IOlapService** interface, found in IOlapService.cs file, created while adding WCF Service to your Web Application.

C#

```
namespace PivotClientDemo {
[ServiceContract]
public interface IOlapService
{
[OperationContract]
Dictionary<string, object> InitializeClient(string action, string
customObject, string clientParams);
```

```
[OperationContract]
Dictionary<string, object> FetchMemberTreeNodes(string action, string
dimensionName, string olapReport);
[OperationContract]
Dictionary<string, object> InitializeChart(string action, string
currentReport, string customObject);
[OperationContract]
Dictionary<string, object> InitializeTreeMap(string action, string
currentReport, string customObject);
[OperationContract]
Dictionary<string, object> DrillChart(string action, string
drilledSeries, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> DrillTreeMap(string action, string drillInfo,
string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> InitializeGrid(string action, string
currentReport, string gridLayout, string customObject);
[OperationContract]
Dictionary<string, object> DrillGrid(string action, string cellPosition,
string currentReport, string clientReports, string headerInfo, string
layout);
[OperationContract]
Dictionary<string, object> FilterElement(string action, string
clientParams, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> RemoveSplitButton(string action, string
clientParams, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> NodeDropped(string action, string dropType,
string nodeInfo, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> CubeChanged(string action, string cubeName,
string clientParams,string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> MeasureGroupChanged(string action, string
measureGroupName);
[OperationContract]
Dictionary<string, object> ToolbarOperations(string action, string
toolbarOperation, string clientInfo, string olapReport, string
clientReports);
[OperationContract]
Dictionary<string, object> UpdateReport(string action, string
clientParams, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> SaveReportToDB(string reportName, string
operationalMode, string analysisMode, string olapReport, string
clientReports);
[OperationContract]
Dictionary<string, object> RemoveReportFromDB(string reportName, string
operationalMode, string analysisMode);
[OperationContract]
Dictionary<string, object> RenameReportInDB(string selectedReport, string
renameReport, string operationalMode, string analysisMode);
[OperationContract]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Dictionary<string, object> LoadReportFromDB(string reportName, string
operationalMode, string analysisMode, string olapReport, string
clientReports);
[OperationContract]
Dictionary<string, object> FetchReportListFromDB(string action, string
operationalMode, string analysisMode);
[OperationContract]
Dictionary<string, object> MemberExpanded(string action, bool
checkedStatus, string parentNode, string tag, string dimensionName,
string cubeName, string olapReport, string clientReports);
[OperationContract]
void Export(System.IO.Stream stream);
[OperationContract]
string GetMDXQuery(string olapReport);
[OperationContract]
Dictionary<string, object> ToggleAxis(string action, string
currentReport, string clientReports);
}
}
```

Secondly, elaborate the service methods inside the main class, found in OlapService.svc.cs file

C#

```
namespace PivotClientDemo {
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
PivotClient pivotClientHelper = new PivotClient();
PivotTreeMap treemapHelper = new PivotTreeMap();
PivotChart chartHelper = new PivotChart();
JavaScriptSerializer serializer = new JavaScriptSerializer();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
string conStringforDB = "";//Enter appropriate connection string to
connect database for saving and loading operation of reports
public Dictionary<string, object> InitializeClient(string action, string
customObject, string clientParams)
{
OlapDataManager DataManager = null;
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return pivotClientHelper.GetJsonData(action, DataManager, clientParams);
}
public Dictionary<string, object> InitializeGrid(string action, string
currentReport, string gridLayout, string customObject)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return pivotClientHelper.GetJsonData(action, DataManager, gridLayout);
}
public Dictionary<string, object> InitializeChart(string action, string
currentReport, string customObject)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return chartHelper.GetJsonData(action, DataManager);
}
public Dictionary<string, object> InitializeTreeMap(string action, string
currentReport, string customObject)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return treemapHelper.GetJsonData(action, DataManager);
}
public Dictionary<string, object> DrillChart(string action, string
drilledSeries, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return chartHelper.GetJsonData(action, DataManager, drilledSeries);
}
public Dictionary<string, object> DrillTreeMap(string action, string
drillInfo, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return treemapHelper.GetJsonData(action, DataManager, drillInfo);
}
public Dictionary<string, object> FilterElement(string action, string
clientParams, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager, clientParams);
}
public Dictionary<string, object> RemoveSplitButton(string action, string
clientParams, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager, clientParams);
}
public Dictionary<string, object> FetchMemberTreeNodes(string action,
string dimensionName, string olapReport)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
return pivotClientHelper.GetJsonData(action, DataManager, dimensionName);
}
public Dictionary<string, object> DrillGrid(string action, string
cellPosition, string currentReport,string clientReports, string
headerInfo, string layout)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager, cellPosition,
headerInfo, layout);
}
public Dictionary<string, object> NodeDropped(string action, string
dropType, string nodeInfo, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager, dropType,
nodeInfo);
}
public Dictionary<string, object> CubeChanged(string action, string
cubeName, string clientParams, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (olapReport != "")
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
if (clientReports != "")
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager, cubeName,
clientParams);
}
public Dictionary<string, object> MeasureGroupChanged(string action,
string measureGroupName)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
return pivotClientHelper.GetJsonData(action, DataManager,
measureGroupName);
}
public Dictionary<string, object> ToolbarOperations(string action, string
toolbarOperation, string clientInfo, string olapReport, string
clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (!string.IsNullOrEmpty(olapReport))
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
if (!string.IsNullOrEmpty(clientReports))
```

```
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager,
toolbarOperation, clientInfo);
}
public Dictionary<string, object> MemberExpanded(string action, bool
checkedStatus, string parentNode, string tag, string dimensionName,
string cubeName, string olapReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (!string.IsNullOrEmpty(olapReport))
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
if (!string.IsNullOrEmpty(clientReports))
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
return pivotClientHelper.GetJsonData(action, DataManager, checkedStatus,
parentNode, tag, dimensionName, cubeName);
}
public Dictionary<string, object> UpdateReport(string action, string
clientParams, string olapReport, string clientReports)
{
return pivotClientHelper.GetJsonData(action, clientParams, olapReport,
clientReports);
}
public Dictionary<string, object> SaveReportToDB(string reportName,
string operationalMode, string analysisMode, string olapReport, string
clientReports)
{
reportName = reportName + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(clientReports).ToArray());
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Save");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return dictionary;
}
public Dictionary<string, object> RemoveReportFromDB(string reportName,
string operationalMode, string analysisMode)
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = reportName + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
cmd1 = new SqlCeCommand("DELETE FROM ReportsTable WHERE ReportName LIKE
'%" + reportName + "%'", con);
}
}
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Remove");
return dictionary;
}
public Dictionary<string, object> RenameReportInDB(string selectedReport,
string renameReport, string operationalMode, string analysisMode)
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
selectedReport = selectedReport + "##" + operationalMode.ToLower() +
"#>>#" + analysisMode.ToLower();
renameReport = renameReport + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(selectedReport))
{
cmd1 = new SqlCeCommand("update ReportsTable set ReportName=@RenameReport
where ReportName like '%" + selectedReport + "%'", con);
}
}
cmd1.Parameters.Add("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Rename");
return dictionary;
}
public Dictionary<string, object> FetchReportListFromDB(string action,
string operationalMode, string analysisMode)
{
string reportNames = string.Empty;
string currentRptName = string.Empty;
foreach (System.Data.DataRow row in GetDataTable().Rows)
```

```
{
currentRptName = (row.ItemArray[0] as string);
if (currentRptName.IndexOf("##" + operationalMode + "#>>#" +
analysisMode) >= 0)
{
currentRptName = currentRptName.Replace("##" + operationalMode + "#>>#" +
analysisMode, "");
reportNames = reportNames == "" ? currentRptName : reportNames + "__" +
currentRptName;
}
}
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("ReportNameList", reportNames);
dictionary.Add("action", action);
return dictionary;
}
public Dictionary<string, object> LoadReportFromDB(string reportName,
string operationalMode, string analysisMode, string olapReport, string
clientReports)
{
PivotReport report = new PivotReport();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
string currentRptName = string.Empty;
foreach (DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string).Replace("##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower(), "");
if (currentRptName.Equals(reportName))
{
byte[] reportByte = new byte[2 * 1024];
reportByte = (row.ItemArray[1] as byte[]);
if (operationalMode.ToLower() == "servermode" && analysisMode == "olap")
{
var repCol = Encoding.UTF8.GetString(reportByte);
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (repCol.IndexOf("<?xml version") == 0)
{
var reportString = "";
reportString =
Syncfusion.JavaScript.Olap.Utils.CompressData(row.ItemArray[1] as
byte[]);
DataManager.Reports =
pivotClientHelper.DeserializedReports(reportString);
DataManager.SetCurrentReport(DataManager.Reports[0]);
return pivotClientHelper.GetJsonData("toolbarOperation", DataManager,
"Load Report", reportName);
}
else
{
dynamic customData = serializer.Deserialize<dynamic>(repCol.ToString());
DataManager.Reports =
pivotClientHelper.DeserializedReports(customData[customData[customData.Le
ngth - 1]["cubeIndex"]]["Reports"]);
DataManager.SetCurrentReport(DataManager.Reports[customData[customData[cu
stomData.Length - 1]["cubeIndex"]]["ReportIndex"]]);
dictionary = pivotClientHelper.GetJsonData("toolbarOperation",
DataManager, "Load Report", reportName);
```

```
dictionary.Add("Collection", repCol);
}
}
}
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
public void Export(Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
pivotClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
public string GetMDXQuery(string olapReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
return DataManager.GetMDXQuery();
}
public Dictionary<string, object> ToggleAxis(string action, string
currentReport, string clientReports)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
DataManager.Reports =
pivotClientHelper.DeserializedReports(clientReports);
DataManager.ToggleAxis(DataManager.CurrentReport);
return pivotClientHelper.GetJsonData(action, DataManager, clientReports);
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport() { Name = "Default Report" };
olapReport.CurrentCubeName = "Adventure Works";
MeasureElements measureElement = new MeasureElements();
measureElement.Elements.Add(new MeasureElement { UniqueName =
"[Measures].[Customer Count]" });
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
olapReport.SeriesElements.Add(dimensionElementRow);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
olapReport.CategoricalElements.Add(measureElement);
return olapReport;
}
}
}
```

Configuring Web Configuration File

You can expose services through the properties, binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IOlapService contract and hence it is PivotClientDemo.IOlapService.
- Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
- behaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

XML

```
<system.serviceModel>
……
……
<services>
<service name="PivotClientDemo.OlapService">
<endpoint address=""
behaviorConfiguration="PivotClientDemo.OlapServiceAspNetAjaxBehavior"
binding="webHttpBinding" contract="PivotClientDemo.IOlapService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotClientDemo.OlapServiceAspNetAjaxBehavior">
<enableWebScript /> </behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

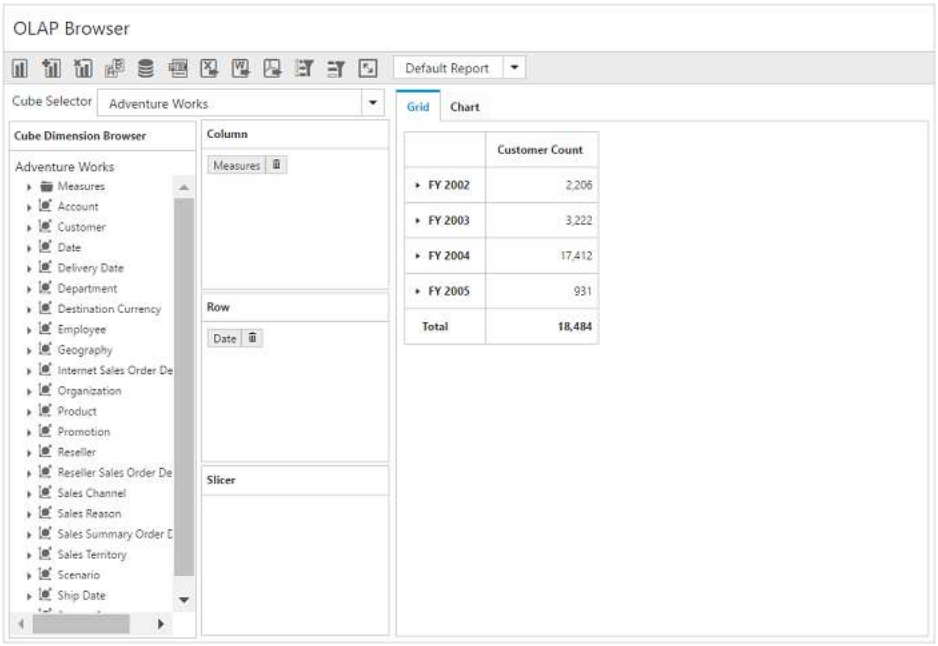
Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
//NOTE: In the following highlighted lines _wcf_ is just the folder name
inside which service files(*.svc) are present.
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new {
controller = "Home", action = "Index", id = UrlParameter.Optional
},
namespaces: new [] {
"PivotClientDemo.Controllers"
}
);
}
```

Note: In this example, **"PivotClientDemo"** indicates the name and root namespace of the Web Application created in Visual Studio IDE and "OlapService" indicates the name of the created WCF service.

Now, **PivotClient** is rendered with PivotChart and PivotGrid showing Customer Count over a period of fiscal years.



PivotClient: Elements

Cube Selector

Cube Selector allows to select any one of the cubes that are available in connected database. This can be achieved with a drop-down list displaying the list of cube names. On selecting a cube from the drop-down list the corresponding cube elements get loaded.

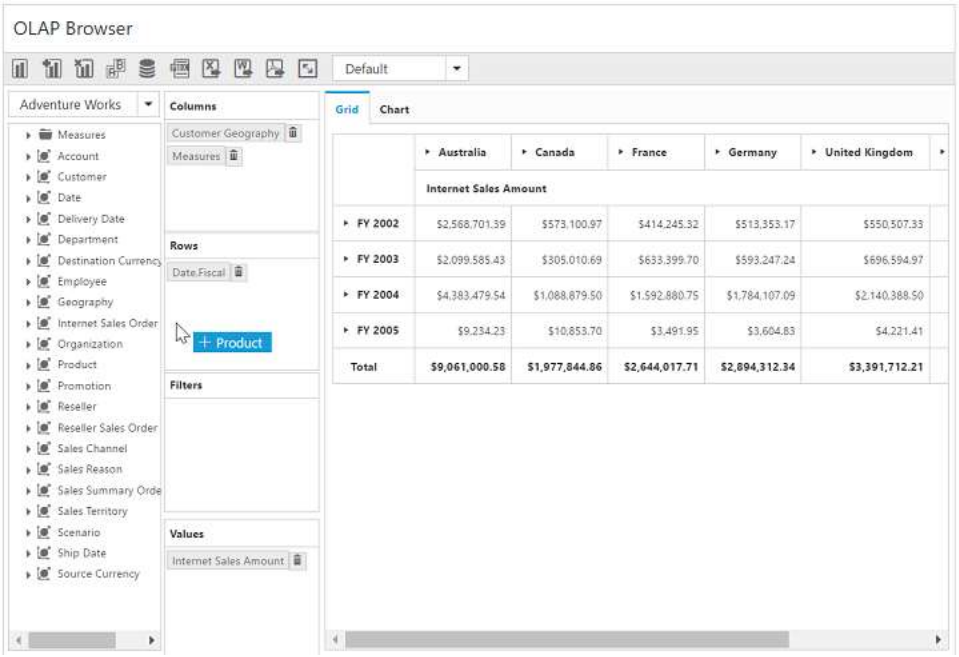


Cube Dimension Browser

Cube Dimension Browser is a treeview like structure that organizes the cube elements such as dimensions, hierarchies, measures, etc., from the selected cube into independent logical groups.

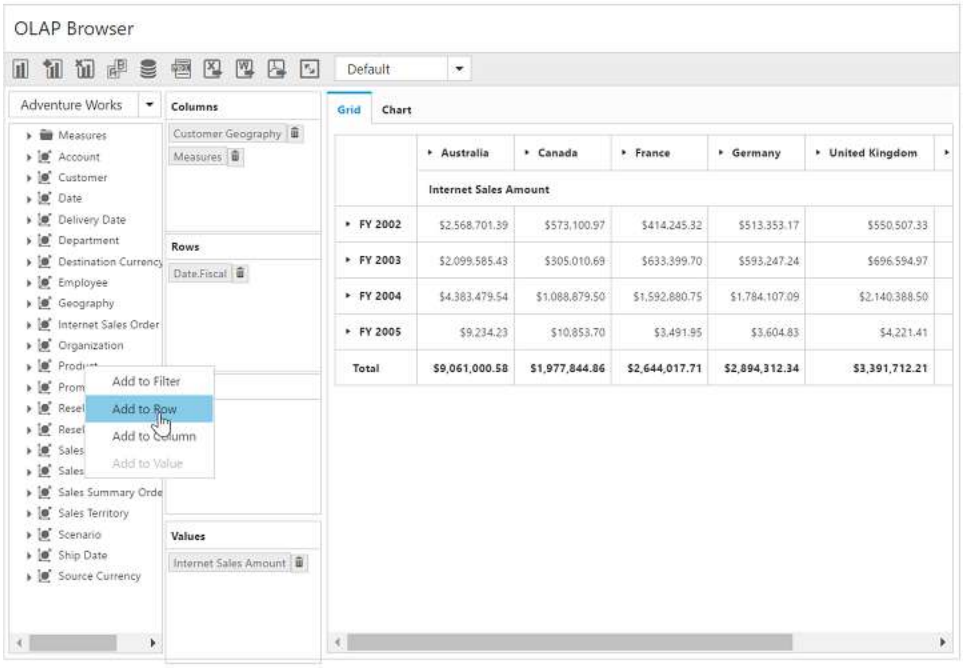
Drag and Drop

You can alter the report on fly through drag-and-drop operation.



Context Menu

You can also alter the report by using context menu.



Types of Node in Cube Dimension Browser

- **Display Folder** - Folder that contains a set of similar elements.
- **Measure** - Quantity available for analysis.
- **Dimension** - A name given to the parts of the cube that categorize the data.
- **Attribute Hierarchy** - Level of attributes down the hierarchy.
- **User-defined Hierarchy** - Members of a dimension in a hierarchical structure.
- **Level** - Denotes a specific level in the category.
- **Named Set** - A collection of tuples and members, which can be defined and saved as a part of the cube definition.

Attribute Hierarchy

Attribute hierarchy contains the following levels:

- A leaf level that contains each distinct attribute member, with each member of the leaf level also known as a leaf member.
- Intermediate levels if the attribute hierarchy is a parent-child hierarchy.
- An optional (All) level (IsAggregatable=True) containing the aggregated value of the attribute hierarchy's leaf members, with the member of the (All) level also known as the (All) member.

User-Defined Hierarchy

User-defined hierarchy organizes the members of a dimension into hierarchical structure and provides navigation paths in a cube. For example, take a dimension table that supports three attributes such as Year, Quarter and Month. The Year, Quarter and Month attributes are used to construct a user-defined hierarchy, named Calendar, in the time dimension which relates all the levels.

Copyright © 2001 - 2020 Syncfusion Inc.

Differentiating User-defined hierarchy and Attribute hierarchy

- User-defined hierarchy may contain more than one level whereas attribute hierarchy contains only one level.
- User-defined hierarchy provides the navigation path between the levels taken from attribute hierarchies of the same dimension.
- In our component, visually these two types are differed by the folder icons mentioned in the immediate topic.

Symbolic representation of the nodes inside Cube Dimension Browser

Icon	Name	Node type	Is Draggable
![Folder icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/Folder icon.png)	Display Folder	Display Folder	False
![Measure icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/Measure icon.png)	Measure	Measure	True
![Dimension icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/Dimension icon.png)	Dimension	Dimension	True
![User defined hierarchy icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/user defined hierarchy.png)	User Defined Hierarchy	Hierarchy	True
![Attribute hierarchy icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/attribute hierarchy.png)	Attribute Hierarchy	Hierarchy	True
![First level icon in ASP NET MVC pivot client control](OLAP-Client-Elements*images/first level.png)* *![Second level icon in ASP NET MVC pivot client control](OLAP-Client-Elements*images/second level.png) ![Third level icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/third level.png)	Levels (in order)	Level Element	True
![NamedSet icon in ASP NET MVC pivot client control](OLAP-Client-Elements_images/names set element.png)	Named Set	Named Set	True

Copyright © 2001 - 2020 Syncfusion Inc.

Axis Element Builder

Axis Element Builder allows you to build the elements in the axes of PivotClient. It supports three axes namely Categorical, Series and Slicer. Based on the construction of axes, PivotGrid and PivotChart will display the resultant data.

Categorical (Column)

The categorical axis defines one or more elements that are displayed along the Chart's y-axis as labels and in the columns of the Grid. If more than one dimension is on the categorical axis, the Chart/Grid will stack each dimension. The order in which the dimensions are stacked is based on the order that they appear on the categorical axis.

Series (Row)

The series axis defines one or more dimensions that are displayed along the Chart's x-axis as labels and in the rows of the Grid. If more than one dimension is on the series axis, the Chart/Grid will stack each dimension. The order in which the dimensions are stacked is based on the order that they appear on the series axis.

Slicer

The slicer axis filters the multidimensional data displayed in the Chart/Grid. It lets you analyze any member of a dimension, in-depth. For the slicer to display the member's data, that member must not be present on both categorical axis and series axis.

![Axis element builder in ASP NET MVC pivot client control](OLAP-Client-Elements_images/Axis Element Builder.png)

Split Button

Split Button highlights the elements in the Axis Element Builder. It holds measures, dimensions, named set. When a dimension/hierarchy/level/named set node is dragged and dropped from Cube Dimensional Browser into Axis Element Builder, a Split Button is created along with displaying the corresponding dimension/named set caption.

When a measure node is dragged and dropped, the Axis Element Builder will create a Split Button for holding the Measures. The next time when a measure is added, it maintains the same single Split Button to hold the entire measure collection.

Split Buttons can be removed with the help of remove button which is available on its side.

![Measure split button in JavScript pivot client control](OLAP-Client-Elements_images/Measure split button.png)



Addition Elements to an Axis

The measure, dimension, hierarchy, level and named set elements can be dragged from the Cube Dimension Browser and dropped into the Axis Element Builder at desired position using the drag-and-drop operation. Also the measure, dimension, hierarchy, level and named set elements can be moved from one axis to another by dragging an appropriate Split Button and dropping them at desired position.

![Adding elements to an axis in ASP NET MVC pivot client control](OLAP-Client-Elements_images/add element.png)

Remove Elements from an Axis

In order to remove measures, dimension, hierarchy, level and named set element from the Axis Element Builder, click the remove button available next to the Split Button.

Copyright © 2001 - 2020 Syncfusion Inc.

![Removing elements from an axis in ASP NET MVC pivot client control](OLAP-Client-Elements_images/remove element.png)

Rearrange Elements in an Axis

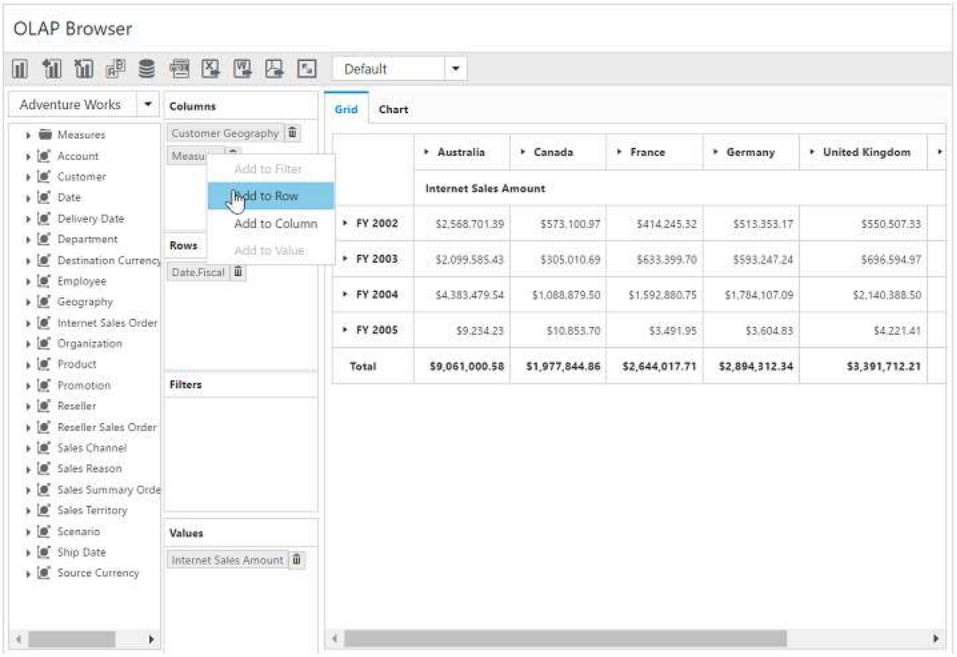
Rearranging can be done by dragging an element and dropping at desired position.

![Rearranging elements in an axis of ASP NET MVC pivot client control](OLAP-Client-Elements_images/rearragne element.png)

![Rearranged elements in ASP NET MVC pivot client control](OLAP-Client-Elements_images/rearrange 1.png)

Context Menu

You can alter the report by using context menu.



Elements Editor

Measure Editor

Measure Editor is a dialog that displays the collection of measures in the current report.

![Measure editor in ASP NET MVC pivot client control](OLAP-Client-Elements_images/measure editor.png)

To remove a measure, click the remove icon next to the corresponding measure. To avoid removing the current selection, click Cancel.

![Removing measure from ASP NET MVC pivot client control](OLAP-Client-Elements_images/remove measure editor.png)

Member Editor

Member Editor is a dialog which displays the members of the current hierarchy in a treeview structure.

![Member editor in ASP NET MVC pivot client control](OLAP-Client-Elements_images/member editor.png)

Copyright © 2001 - 2020 Syncfusion Inc.

The “Check All” and “Uncheck All” option in the dialog lets the user to select and unselect entire nodes for filtering. For more details, about filtering, click here.

Toolbar



The options available in the toolbar are,

- **New Report** - Creates a new reports list, clearing the existing report collection in order to provide a new platform for new deployment based on the existing cube elements.
- **Add Report** - Adds a new report to the existing list of reports.
- **Remove Report** - Removes the current report from the report list. It could not be possible to remove if the report list contains only one report in it.
- **Rename Report** - The rename option lets the user to change the name of the current report.
- **Report Manipulation in DB** - which contains the options to manipulate the reports in the Database.
- **MDX Query** – This option displays the MDX query executed to retrieve the data at that instant.
- **Defer Update** – This option lets the user to update the widget on-demand rather than updating it for every interaction.
- **Export to Excel** - This option is used to export the Grid and Chart into an Excel worksheet.
- **Export to Word** – This option is used to export the Grid and Chart into a Word document.
- **Export to PDF** – This option is used to export the Grid and Chart into a PDF document.
- **Full Screen View** – Displays the active control (Chart/Grid) in full/maximized view.
- **Filter/Sort Column** - Filters/Sorts the data in the OlapReport with respect to Column.
- **Filter/Sort Row** - Filters/Sorts the data in the OlapReport with respect to Row.
- **Toggle Axis** – Interchanges the items between Categorical and Series axes.
- **Chart types** – Allows user to change the Chart type dynamically.
- **Report List** - It holds all the reports of the current session of the PivotClient control and displays in a dropdown list. By default, from the dropdown list a report can be selected.

Following are the options available for database report manipulation,



- **Save** - The save option stores the report collection by only one name in a database.
- **Save As** - The save as option stores the report collection by more than one name in a database.
- **Load** - The load option picks the saved report collection from the database and loads it by clearing the existing collection of reports.
- **Remove** - Removes the selected report from the saved report list in a database.
- **Rename** - The rename option lets the user to change the name of the selected report from the saved report list in a database.

Report Manipulation

New Report

New Report option helps the user to create a new report collection with a single report along with clear the existing report collection. On clicking the New Report icon in toolbar, the New Report dialog opens as shown below.

Copyright © 2001 - 2020 Syncfusion Inc.

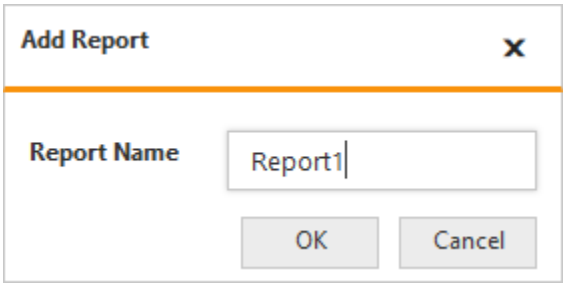
![New report in ASP NET MVC pivot client control](OLAP-Client-Elements_images/add new report.png)

By giving the required name and clicking OK, the report collection gets loaded with only one empty report with entered name. On clicking Cancel button, report creation gets canceled.

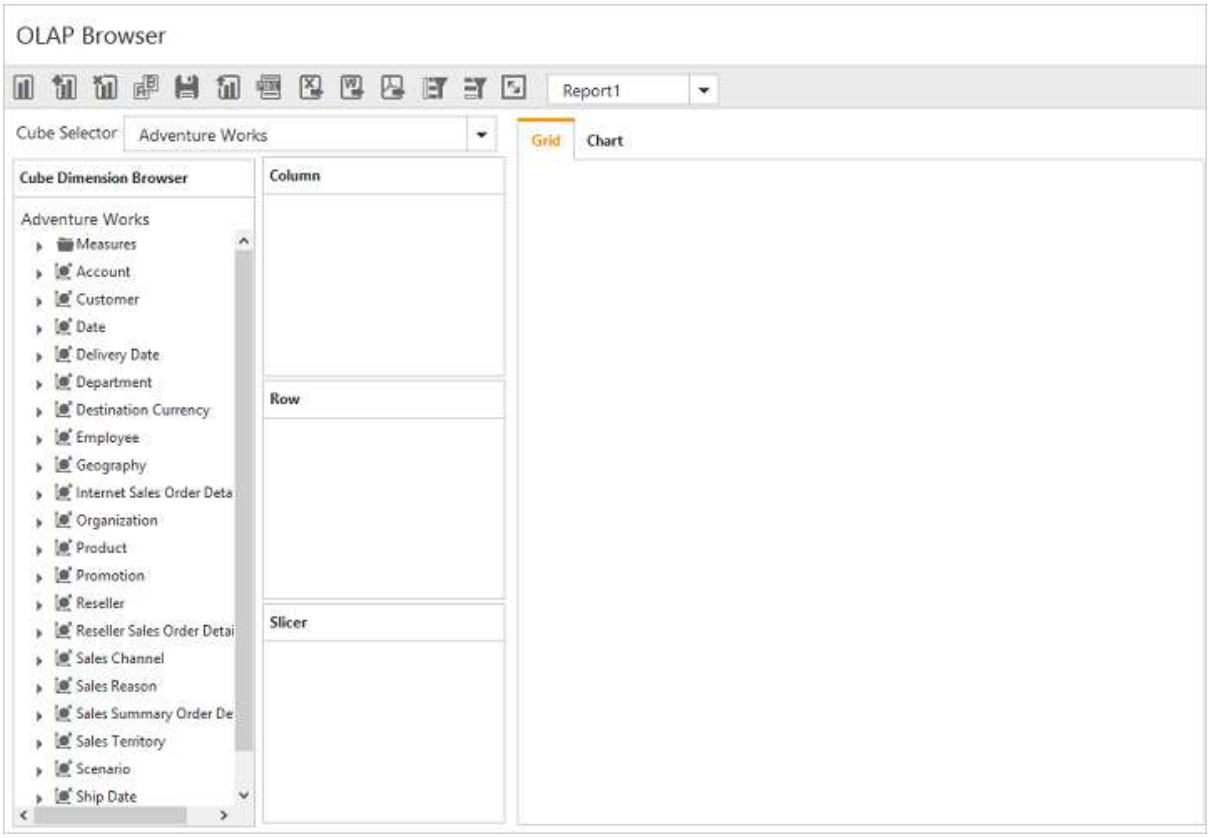
![New report in ASP NET MVC pivot client control](OLAP-Client-Elements_images/add new report1.png)

Add Report

Add Report option lets the user to add a report to the existing report collection. On clicking the Add Report icon in toolbar, the Add Report dialog opens prompting for a name for the report to be added as shown below.



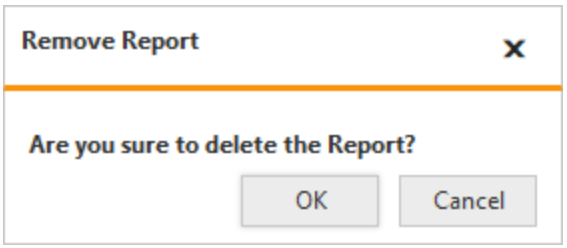
On clicking OK button after giving appropriate name in the respective column, a report with the entered name is added to the collection. On clicking Cancel button, report creation gets canceled.



Remove Report

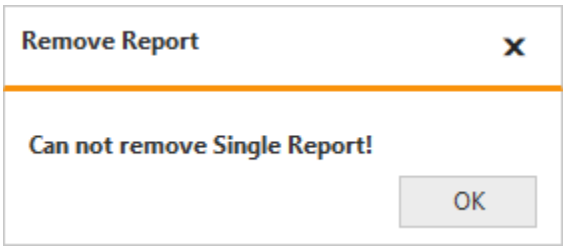
Remove Report option removes the current/active report from the report collection. On clicking the icon at toolbar, a dialog will open prompting for user's confirmation on removal of report as shown below.

Copyright © 2001 - 2020 Syncfusion Inc.



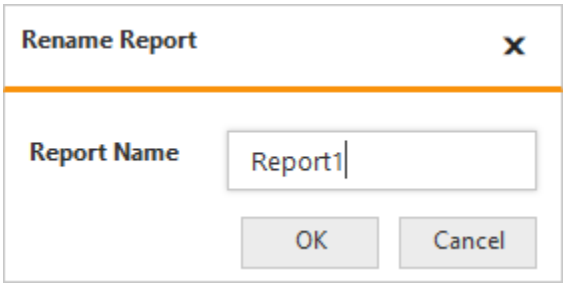
On clicking OK button, the report gets removed from the collection. On clicking Cancel button, the operation gets aborted.

This option works only if the report collection has more than one report. If not, the below dialog opens on clicking the Remove Report toolbar icon which displays a message that single report cannot be removed.



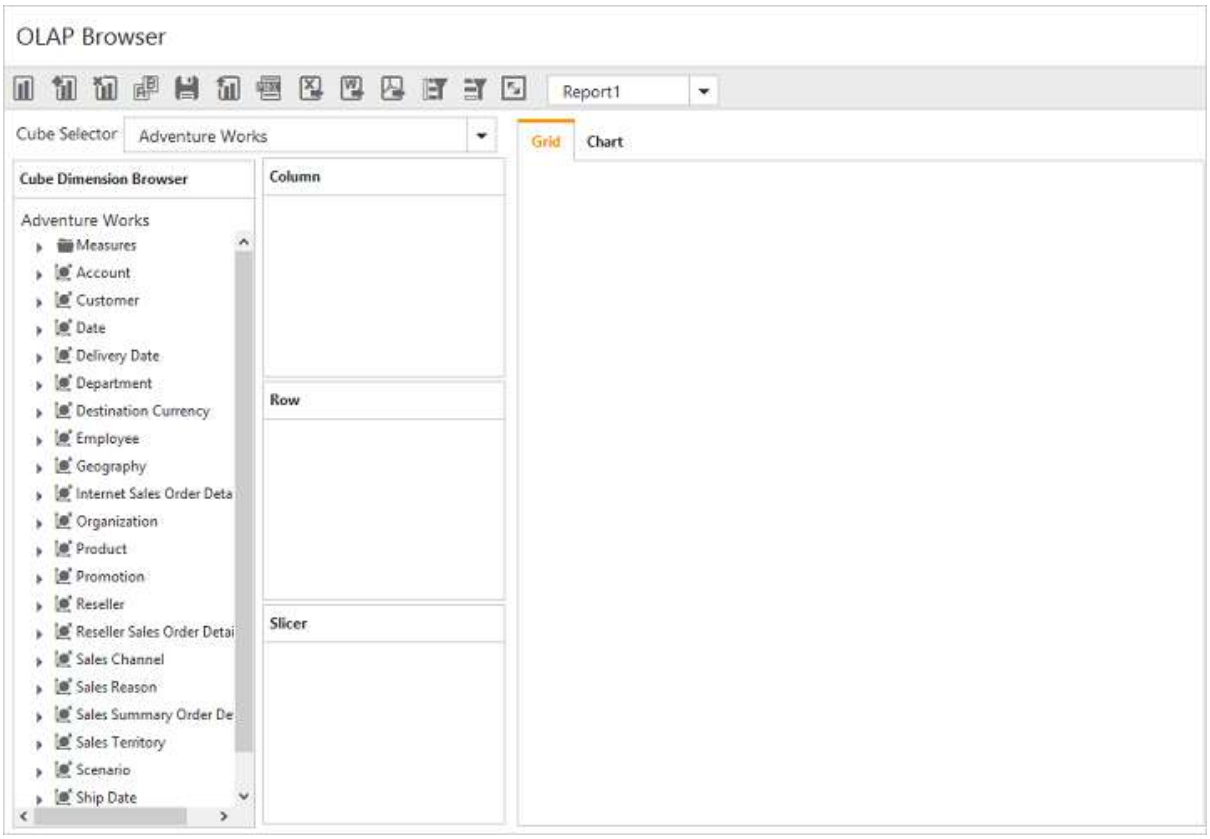
Rename Report

Rename Report option lets the user to rename the current/active report. On clicking the corresponding icon at toolbar, the below dialog opens prompting the new name.



After giving a name and clicking OK, the active report gets refreshed with the new name. On clicking Cancel button, the operation gets canceled.

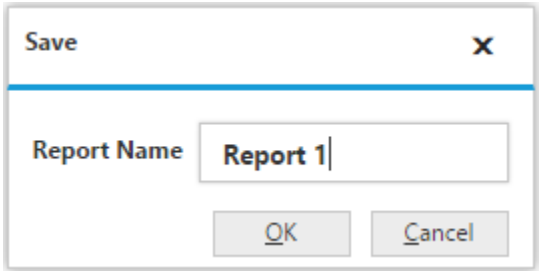
Copyright © 2001 - 2020 Syncfusion Inc.



Save

Save option saves the entire report collection by only one name to a database. The database is connected to the application through a connection string provided in the service.

On clicking the Save icon, the Save dialog is opened prompting for a name with which the report collection needs to be stored.



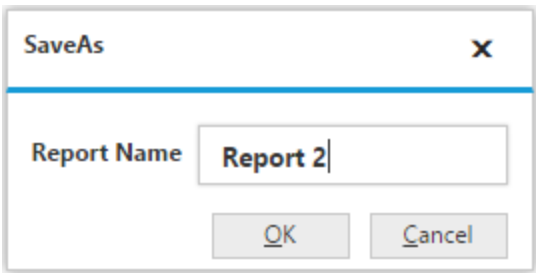
After providing appropriate name and clicking OK, the report collection will get saved to the connected database. Clicking Cancel button, withdraws the saving operation.

Save As

Save As option saves the entire report collection by two or more name to a database. The database is connected to the application through a connection string provided in the service.

On clicking the Save As icon, the Save As dialog is opened prompting for a name with which the report collection needs to be stored.

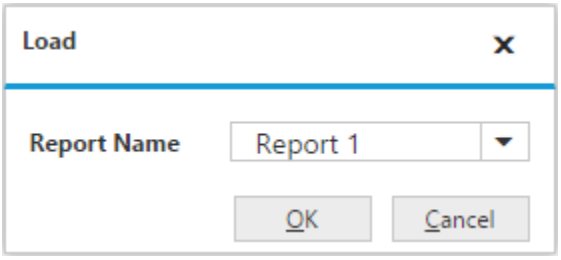
Copyright © 2001 - 2020 Syncfusion Inc.



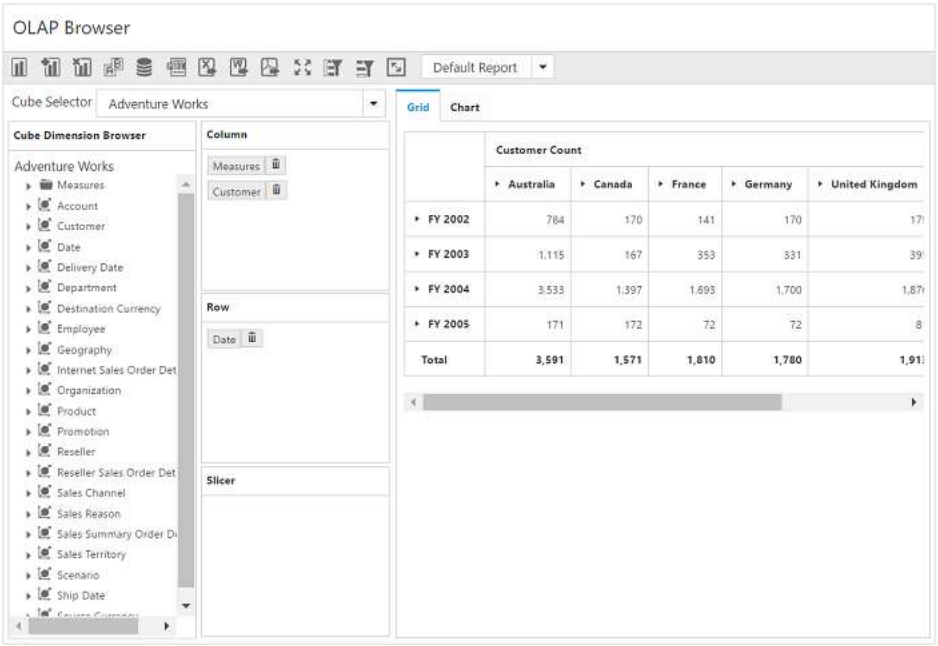
After providing appropriate name and clicking OK, the report collection will get saved to the connected database. Clicking Cancel button, withdraws the saving operation.

Load

Load option loads a saved report collection from the database via service. Clicking the respective icon at toolbar, the Load dialog opens.



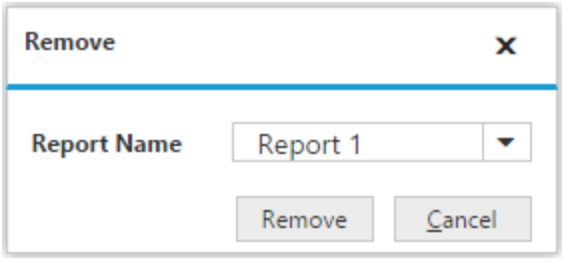
Any one of the report from the report collection could be selected from the drop-down list and OK button is clicked for loading the selected report collection. Cancel button is used for to abort the report loading operation.



Remove

Remove option removes the saved report from the report collection. On clicking the icon at toolbar, a dialog will open prompting for user's confirmation on removal of report as shown below.

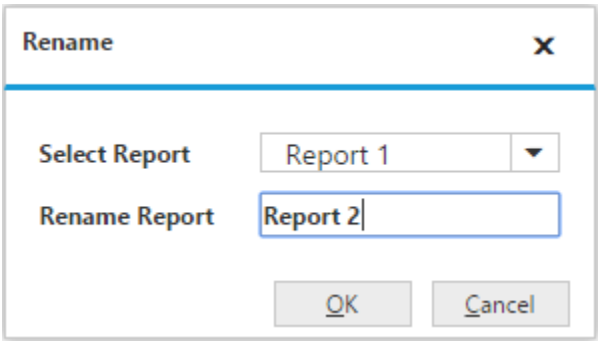
Copyright © 2001 - 2020 Syncfusion Inc.



On clicking OK button, the report gets removed from the collection. On clicking Cancel button, the operation gets aborted.

Rename

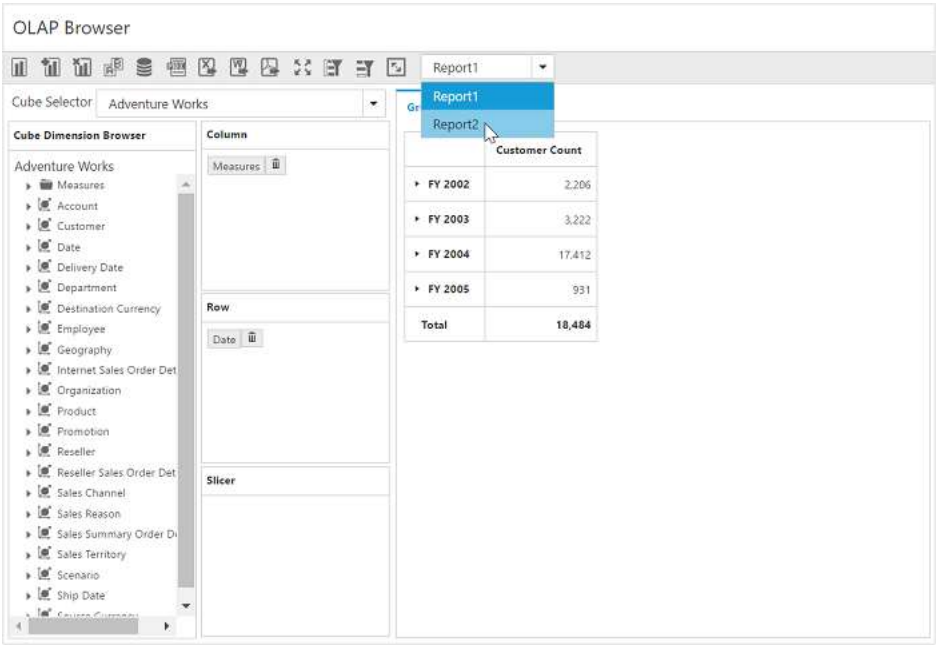
Rename option lets the user to rename the saved report. On clicking the corresponding icon at toolbar, the below dialog opens prompting the new name.



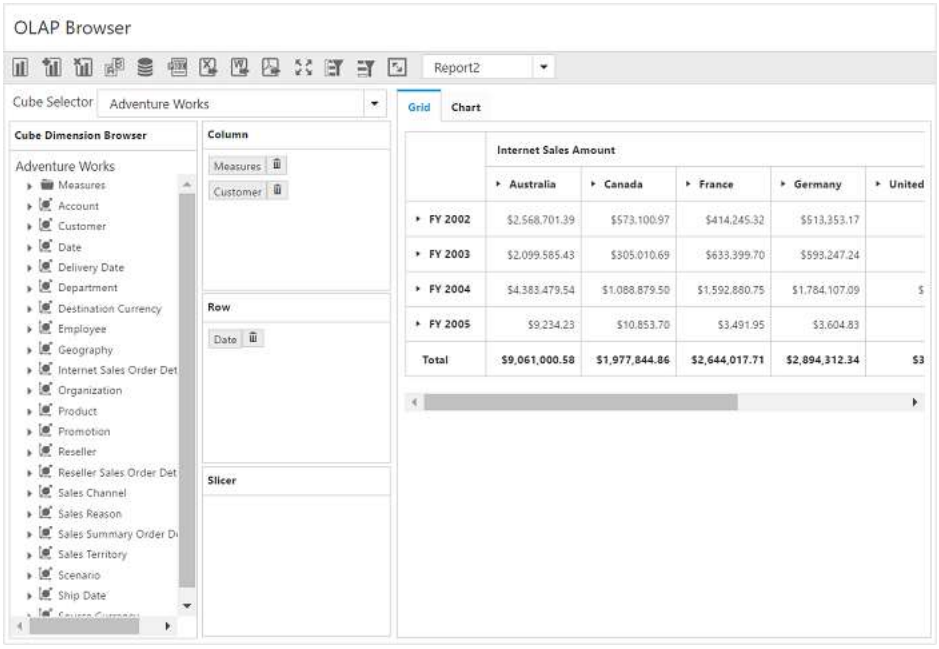
After giving a name and clicking OK, the active report gets refreshed with the new name. On clicking Cancel button, the operation gets canceled.

Report List

Report List drop-down contains the names of all the reports in the report collection.



On selecting the required report from the report list, the selected report will be set as active report and gets loaded.



PivotGrid and PivotChart

The PivotGrid and PivotChart controls will be rendered with respect to the operations done at axis element builder.

Relational

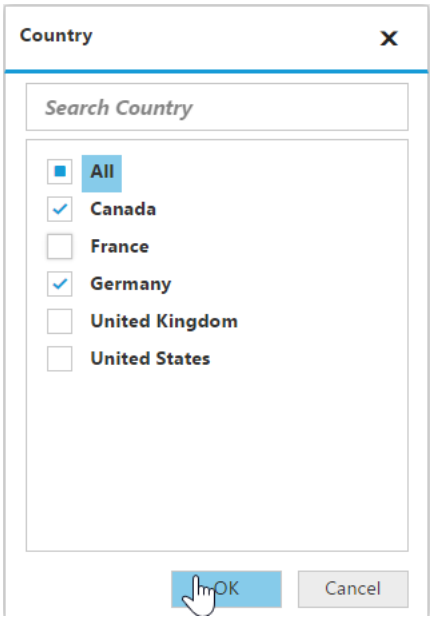
Data Exploration

Filtering

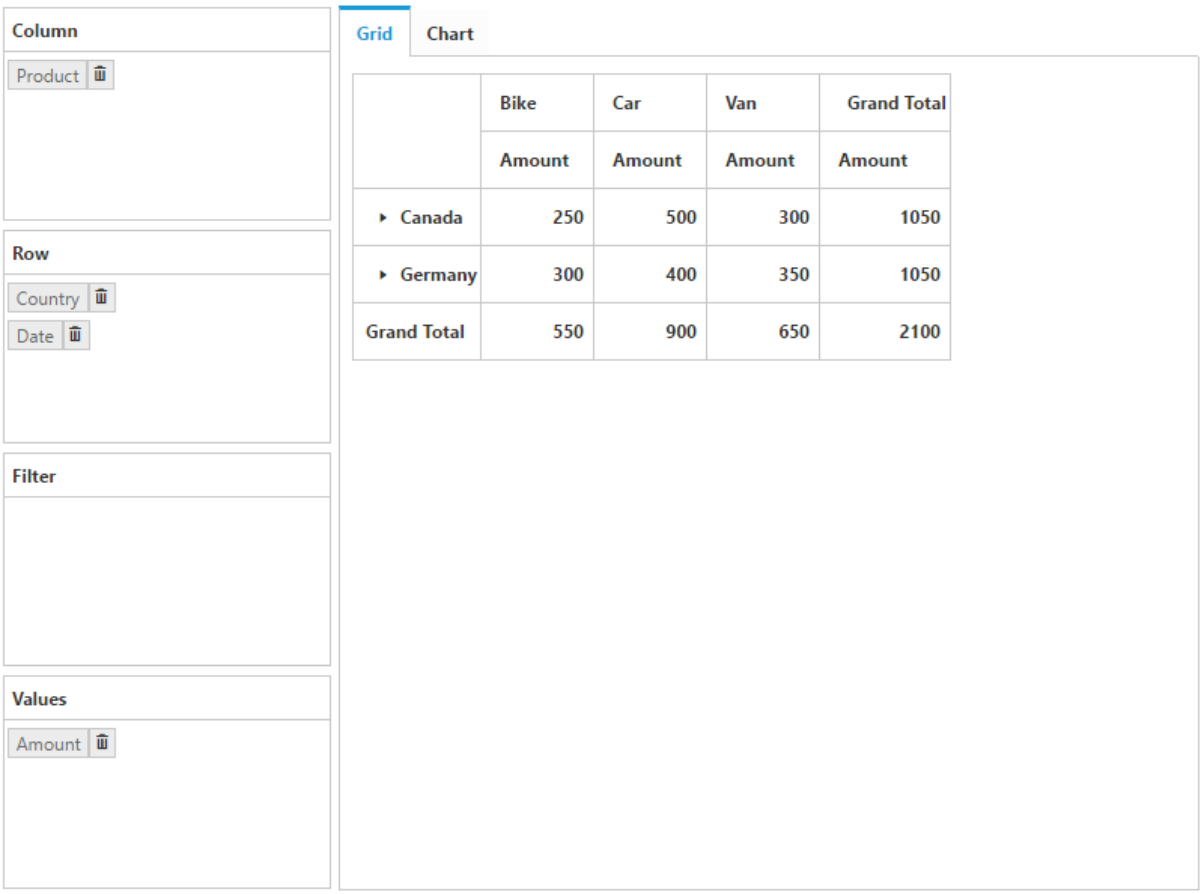
Filtering by Member

By clicking Split Button of a field, Member Editor dialog opens through which members are filtered by checking and unchecking the check boxes corresponding to the members.

Copyright © 2001 - 2020 Syncfusion Inc.



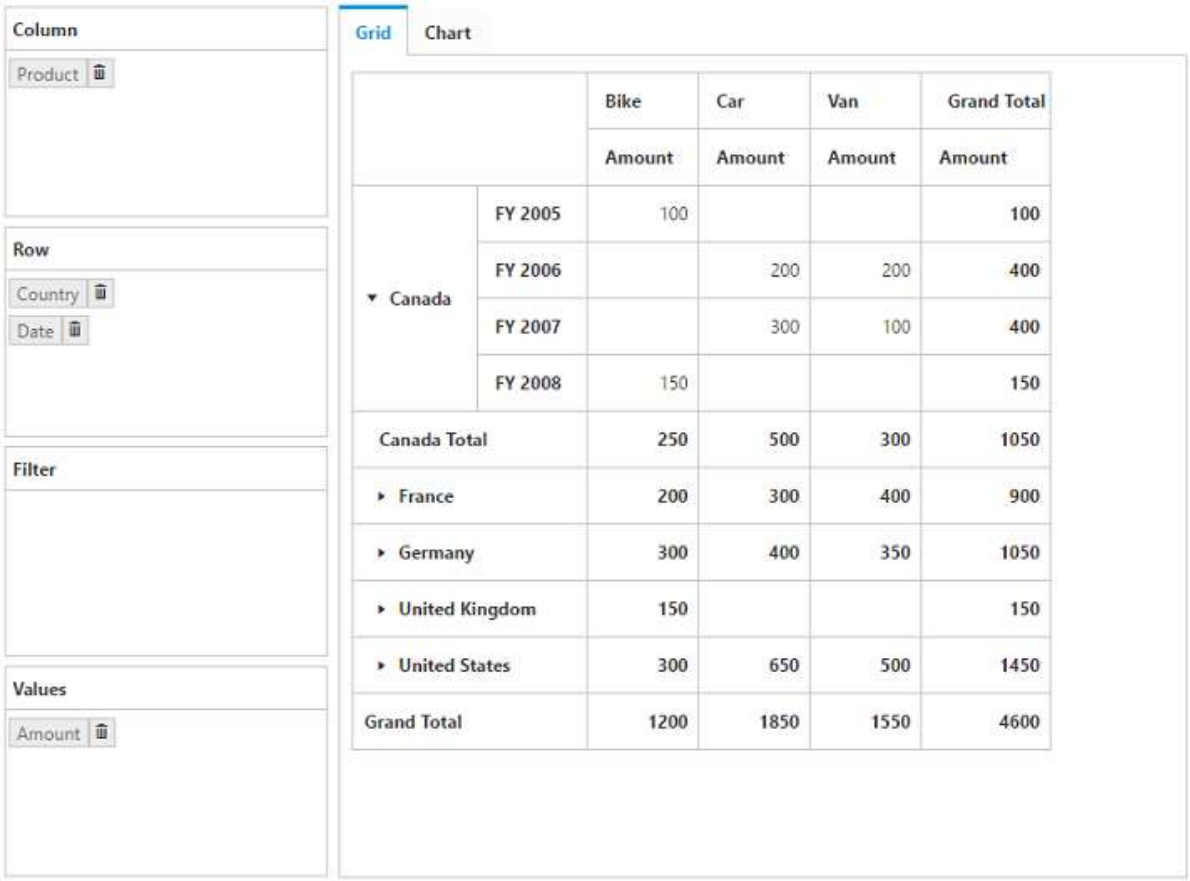
On clicking the "OK" button, based on the selected members in the Member Editor dialog, PivotReport gets updated and refreshes the PivotGrid and PivotChart controls. The "Cancel" button is used to cancel the selection.



The above filter illustrates that the members 'Canada' and 'Germany' are alone included in the Grid and Chart controls.

Grouping

The data can be grouped when more than one field element is added to Categorical or Series in Axis Element Builder. Based on the order of addition, data is grouped and the report is updated. In the following example, the **Date** dimension values get grouped, with respect to **Country** dimension values. Likewise multiple field members can be grouped by dragging the elements from pivot field list to Axis Element Builder.

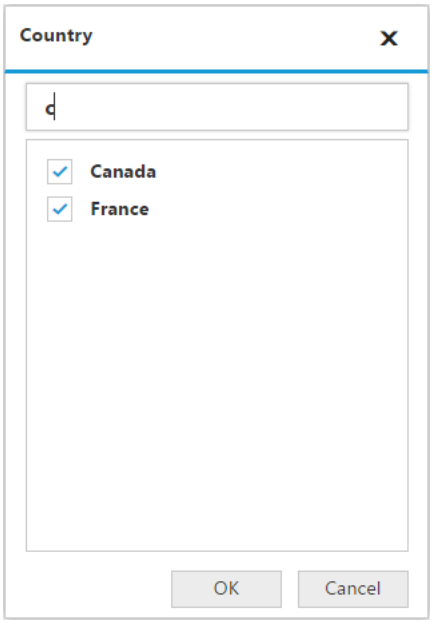


		Bike	Car	Van	Grand Total
		Amount	Amount	Amount	Amount
▼ Canada	FY 2005	100			100
	FY 2006		200	200	400
	FY 2007		300	100	400
	FY 2008	150			150
Canada Total		250	500	300	1050
▸ France		200	300	400	900
▸ Germany		300	400	350	1050
▸ United Kingdom		150			150
▸ United States		300	650	500	1450
Grand Total		1200	1850	1550	4600

Searching

Members can be searched and displayed from the members list inside the Member Editor dialog.

Copyright © 2001 - 2020 Syncfusion Inc.



Exporting

Chart and Grid in the PivotClient widget can be exported to Excel, Word and PDF documents by clicking the respective toolbar icons.



Exporting feature provides an option that allows you to export either PivotChart or PivotGrid or both with the use of the property ClientExportMode.

The property ClientExportMode takes any one of the following value:

- **ChartAndGrid** – Exports both PivotChart and PivotGrid controls. This is the default mode.
- **ChartOnly** – Exports PivotChart control alone.
- **GridOnly** – Exports PivotGrid control alone.

JOSN Export

Information: By default, exporting is done with the use of JSON Records maintained in client-side for both client and server modes.

In order to make use of exporting with client side JSON data. The control can be exported by invoking “BeforeExport” event, with an appropriate export option as parameter.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents =>
clientSideEvents.Load("onLoad").BeforeExport("Export")).DataSource(dataSo
urce => dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
values.FieldName("Amount").Format("currency").Add();
})).ClientExportMode(ClientExportMode.ChartAndGrid)
<script type="text/javascript">
function Export(args) {
args.url = "ExportPivotClient";
}
</script>
```

When PivotClient is exported with JSON mode, a method needs to be added in MVC controller file of the application.

C#

```
public void ExportPivotClient()
{
JavaScriptSerializer serializer = new JavaScriptSerializer() {
MaxJsonLength = Int32.MaxValue };
PivotClientExport pivotClient = new PivotClientExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotClient.ExportPivotClient(string.Empty, args,
System.Web.HttpContext.Current.Response);
}
```

Customize the export document name

The document name to be exported could be customized. Following code sample illustrates the same.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents =>
clientSideEvents.Load("onLoad").BeforeExport("Export")).DataSource(dataSo
urce => dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).ClientExportMode(ClientExportMode.ChartAndGrid)
<script type="text/javascript">
function Export(args) {
args.url = "ExportPivotClient";
args.fileName=" File name is customized here ";
}
</script>
```

PivotEngine Export

Information: This feature is applicable only at server mode operation.

In order to perform exporting with the use of PivotEngine available in server-side, the 'exportMode' property obtained in the “BeforeExport” event is set to "ej.PivotClient.ExportMode.PivotEngine" as shown below.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/Relation
alClient")).ClientExportMode(ClientExportMode.ChartAndGrid).ClientSideEve
nts(oEve => { oEve.BeforeExport("Export"); })
<script type="text/javascript">
function Export(args) {
args.exportMode = ej.PivotClient.ExportMode.PivotEngine;
}
</script>
```

For WebAPI controller, the below method needs to be added to perform exporting with PivotEngine.

C#

```
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
```

For WCF service, the below service method needs to be added to perform exporting with PivotEngine.

C#

```
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
```

File format selection

Information: This option is applicable only for PivotClient excel export.

You can set the option for exporting excel sheet as either *.xls* or *.xlsx* format before export the document by using 'fileFormat' property obtained in the “BeforeExport” event.

Note: By default excel document can be exported as ".xls" format using PivotEngine export.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(oEve => {
oEve.BeforeExport("Exporting"); })
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function Exporting(args) {
args.exportMode = ej.PivotClient.ExportMode.PivotEngine;
args.fileFormat = ".xlsx"; //you can set the excel sheet format here
}
</script>
```

Customize the export document name

The document name could be customized inside the method in WebAPI Controller. Following code sample illustrates the same.

C#

```
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
string fileName = " File name is customized here ";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
```

For customizing name in WCF Service, below code snippet is used.

C#

```
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
string fileName = " File name is customized here ";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
```

PivotChart - Exporting Format

Information: This option is applicable only for PivotChart in PivotClient specifically when exported to Excel document.

You can set an option to export PivotChart to an Excel document, either as image or PivotChart format itself by setting the boolean property `exportChartAsImage`, inside the `BeforeExport` event.

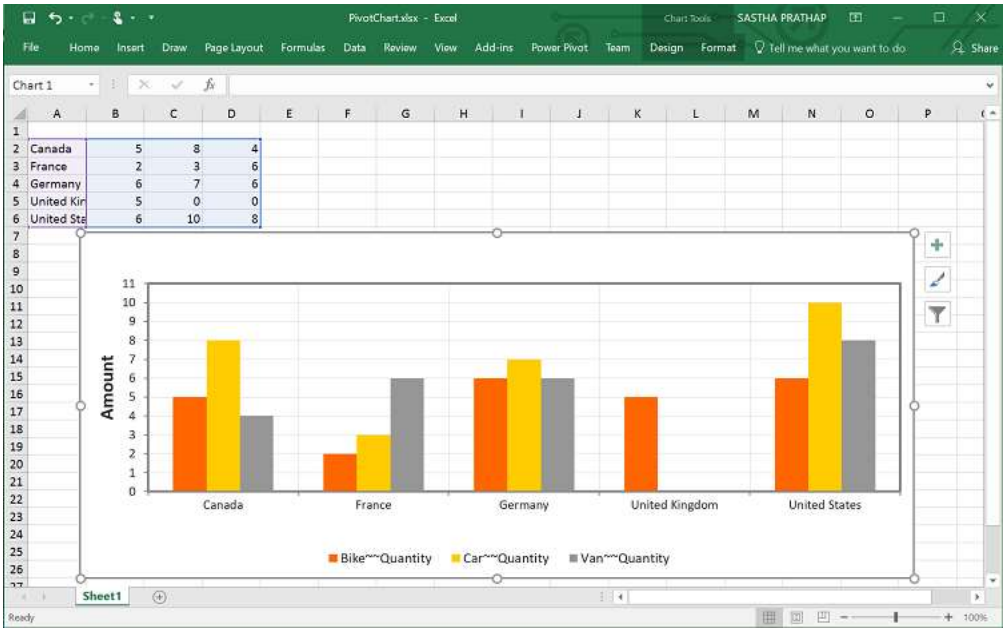
Note: By default PivotChart will be exported as image format to Excel document.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/Relation
alClient")).ClientExportMode(ClientExportMode.ChartOnly).ClientSideEvents
(oEve => { oEve.BeforeExport("Export"); })
<script type="text/javascript">
function Export(args) {
args.exportChartAsImage = false; //you can set the chart format here
}
</script>
```

The below screenshot shows the control exported to Excel document showing its own format (Pivoting Chart).



Exporting Customization

You can add title and description to the exporting document by using the title and description properties respectively obtained in the BeforeExport event. Similarly, you can enable or disable styling on the exported document by using the exportWithStyle property.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(oEve => {
oEve.BeforeExport("Exporting"); })
<script type="text/javascript">
function Exporting(args) {
//ClientMode export
args.url = "ExportPivotClient";
//PivotEngine Export
args.exportMode = ej.PivotClient.ExportMode.PivotEngine;
args.title = "PivotClient";
args.description = "Visualizes both OLAP and Relational datasource in
tabular and graphical formats";
args.exportWithStyle = true;   // by default it sets as true. It improves
performance on exporting huge data when it sets as false.
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

You can also edit the exporting document with the use of a server side event for required exporting option.

C#

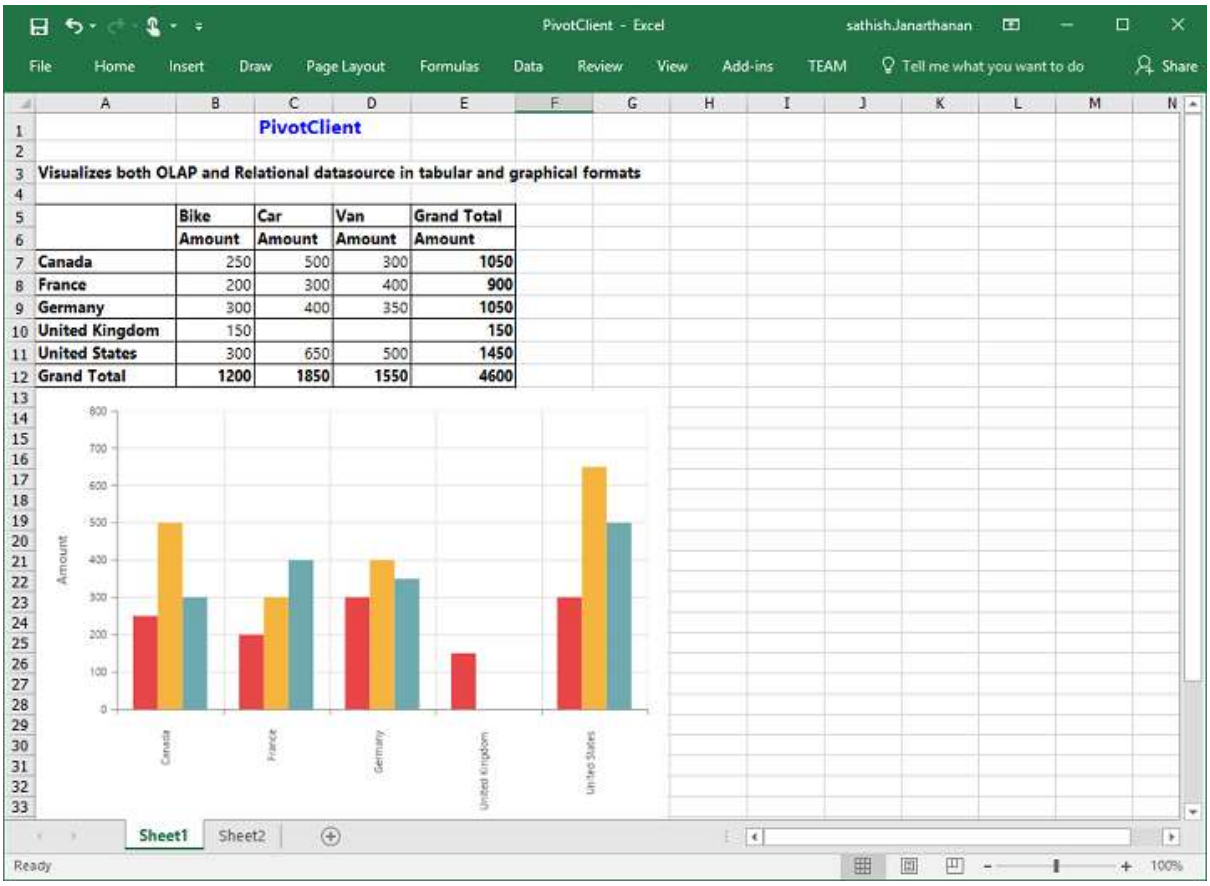
```
//...
using Syncfusion.EJ.Export;
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
//Following methods needs to be added in MVC controller file of the
application for JSON Export.
public void ExportPivotClient()
{
JavaScriptSerializer serializer = new JavaScriptSerializer() {
MaxJsonLength = Int32.MaxValue };
PivotClientExport pivotClient = new PivotClientExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotClient.ExcelExport += pivotClient_ExcelExport;
pivotClient.WordExport += pivotClient_WordExport;
pivotClient.AddPDFHeaderFooter += pivotClient_AddPDFHeaderFooter;
pivotClient.PDFExport += pivotClient_PDFExport;
pivotClient.ExportPivotClient(string.Empty, args,
System.Web.HttpContext.Current.Response);
}
void pivotClient_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void pivotClient_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void pivotClient_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
void pivotClient_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
//Following service method needs to be added in WebAPI controller for
PivotEngine Export.
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
```

Copyright © 2001 - 2020 Syncfusion Inc.

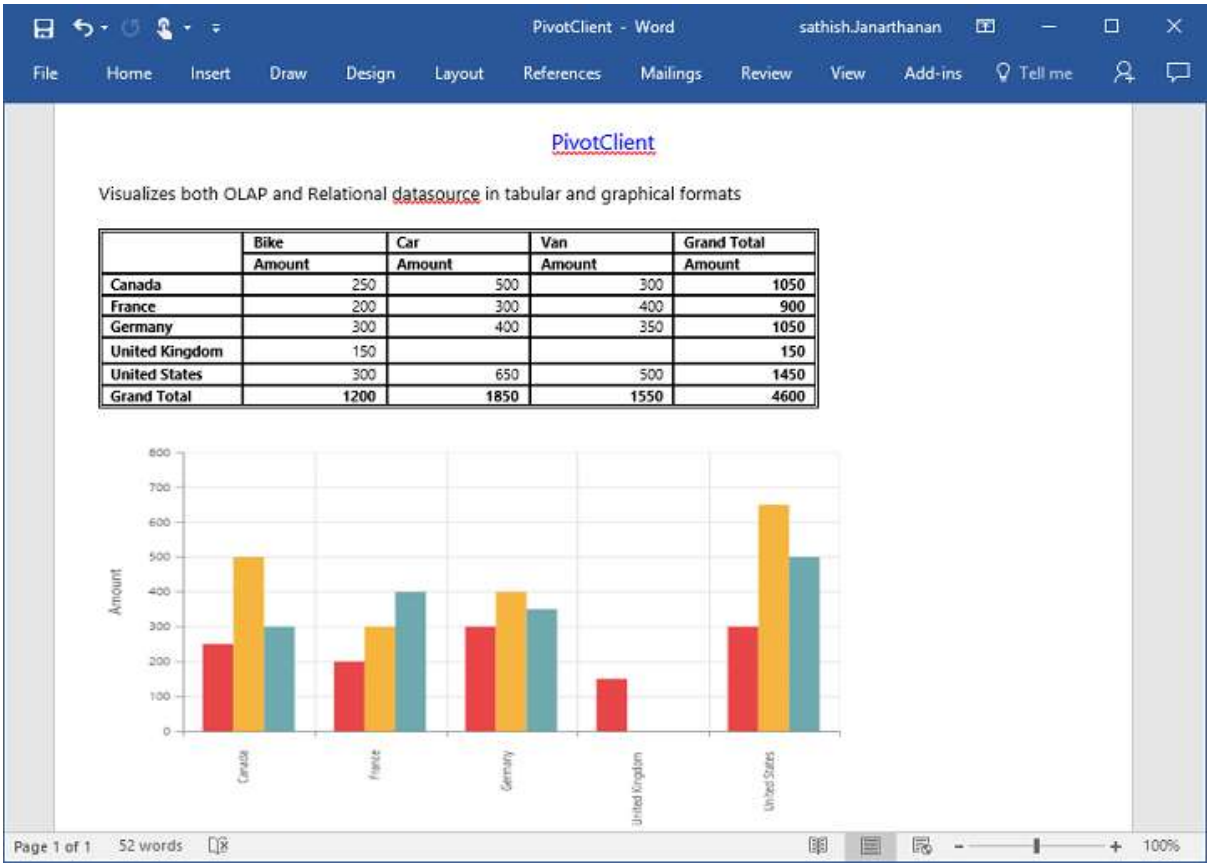
```
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
pivotClient.PopulateData(gridParams["currentReport"]);
pivotClient.ExcelExport += pivotClient_ExcelExport;
pivotClient.WordExport += pivotClient_WordExport;
pivotClient.AddPDFHeaderFooter += pivotClient_AddPDFHeaderFooter;
pivotClient.PDFExport += pivotClient_PDFExport;
string fileName = "Sample";
pivotClient.ExportPivotClient(ProductSales.GetSalesData(), args,
fileName, System.Web.HttpContext.Current.Response);
}
void pivotClient_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void pivotClient_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void pivotClient_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
void pivotClient_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
```

The below screenshot shows the PivotGrid and PivotChart controls exported to Excel document.

Copyright © 2001 - 2020 Syncfusion Inc.

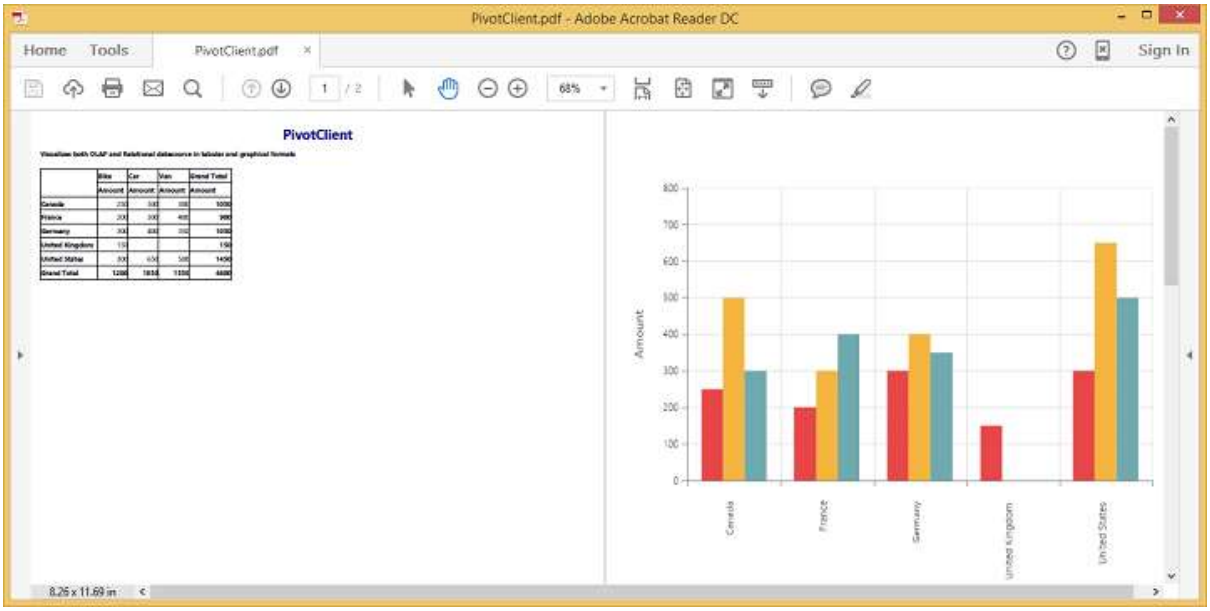


The below screenshot shows the PivotGrid and PivotChart controls exported to Word document.



	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
Canada	250	500	300	1050
France	200	300	400	900
Germany	300	400	350	1050
United Kingdom	150			150
United States	300	650	500	1450
Grand Total	1200	1850	1550	4600

The below screenshot shows the PivotGrid and PivotChart controls exported to PDF document.



Value Sorting

Information: This feature is applicable only for the relational datasource.

PivotClient provides support for value sorting that allows you to sort columns and rows based on value fields.

Copyright © 2001 - 2020 Syncfusion Inc.

The headers of the column to be sorted are given in the HeaderText property under ValueSortSettings in field wise order separated by a string. The string which is used to separate the headers is given in the HeaderDelimiters property.

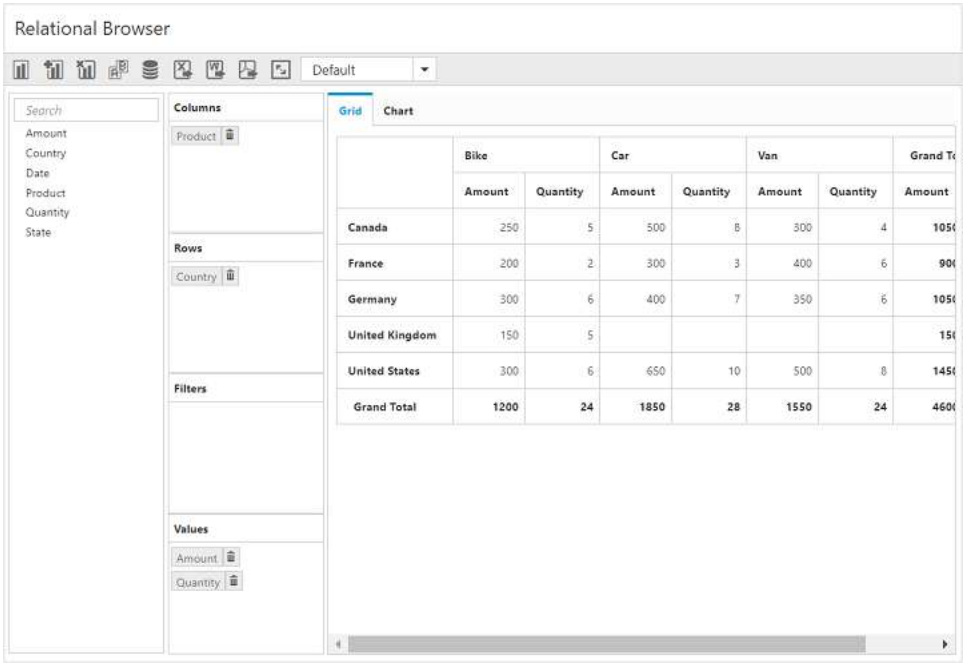
Also, you can sort the column by clicking the column header. By clicking the same header once again, the sorting direction will be reversed. The sorting operation is performed by the using the sortOrder property.

The following code snippet shows how to sort values in descending order.

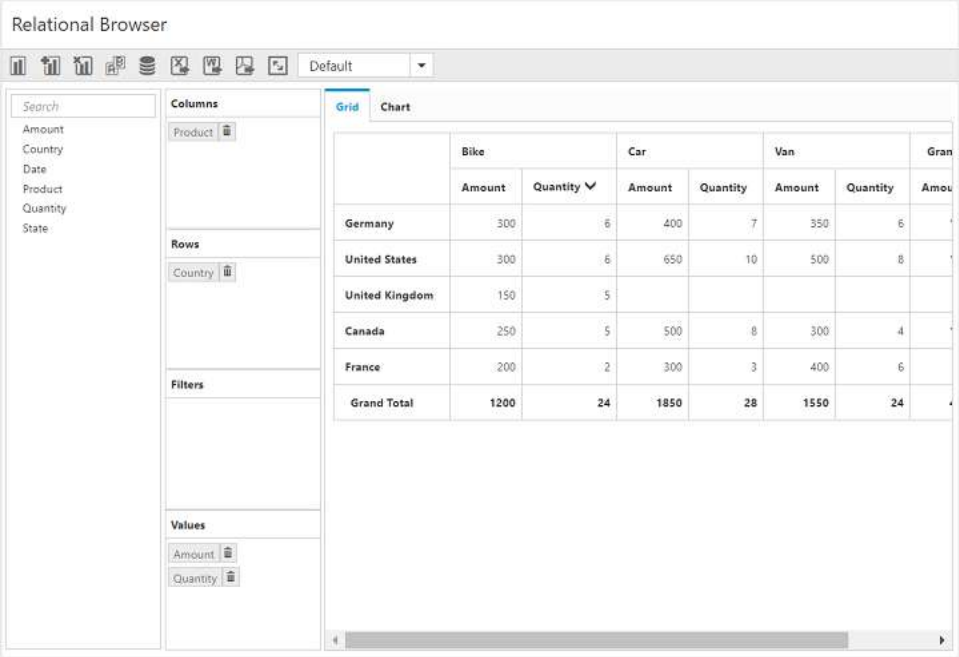
JS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();
values.FieldName("Quantity").Add();
})).ValueSortSettings(valuesortsettings=>valuesortsettings.HeaderText("Bi
ke##Quantity").HeaderDelimiters("##").SortOrder(SortOrder.Descending))
```

The below screenshot shows PivotClient before applying value sorting.



The below screenshot shows PivotClient after applying value sorting.



Summary Types

Information: This feature is applicable only for the relational datasource.

Allows you to perform calculations over a group of values using summaryType option. This property includes several types of aggregations like sum, average, minimum, maximum etc. You can see the totals of PivotClient control with any one of the below mentioned summary types.

Also, summary type options are added in the context menu of pivot buttons, in values section, to change the resultant summary value dynamically.

Client Mode

Client mode supports the following summary types. By default, sum is applied.

- Sum
- Average
- Count
- Min
- Max

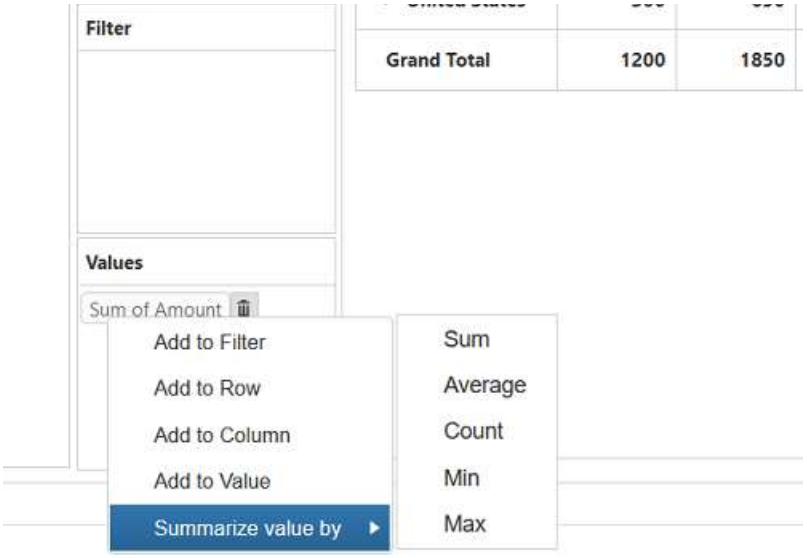
Refer the following code sample to implement summary type through code behind.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").SummaryType(PivotAnalysisSummaryType.Average).
Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
values.FieldName("Quantity").FieldCaption("Quantity").SummaryType(PivotAn
alysisSummaryType.TotalSum).Add(); }))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = pivot_dataset; // Datasource
}
</script>
```



Server Mode

Server mode supports the following summary types. By default, count is applied.

- Sum
- Average
- Count
- Min
- Max
- DoubleTotalSum
- DoubleAverage
- DoubleMaximum
- DoubleMinimum
- DoubleStandardDeviation
- DoubleVariance
- DecimalTotalSum
- IntTotalSum
- DisplayIfDiscreteValuesEqual
- Count numbers
- Standard deviation
- Standard deviation of population
- Variance
- Variance of population

Copyright © 2001 - 2020 Syncfusion Inc.

Through code behind

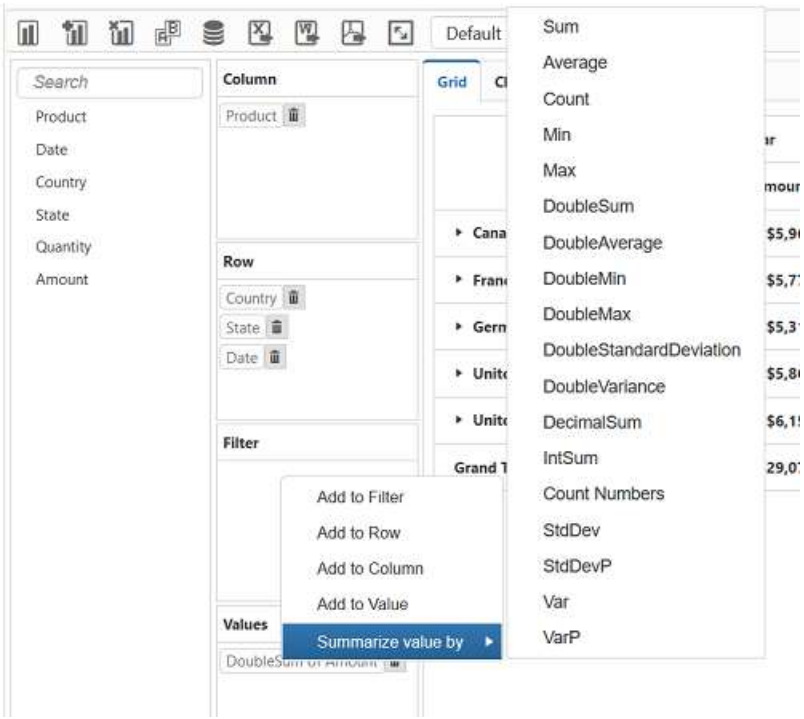
Refer the following code sample to implement summary type through code behind.

C#

```
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
//...
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
return pivotSetting;
}
```

Through UI interaction

You can change summary types at run-time through built-in context menu for respective value field. Once the summary type is selected, the corresponding pivot item will be updated with the respective summary type.



To achieve this behavior at run-time, add the following `ChangeSummaryType` method in WebAPI/WCF service.

WebAPI

C#

```
[System.Web.Http.ActionName("ChangeSummaryType")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> ChangeSummaryType(Dictionary<string,
object> jsonResult)
{
```

```
pivotClient.PopulateData(jsonResult["currentReport"].ToString());
var dictionary = pivotClient.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["summaryType"].ToString());
return dictionary;
}
```

WCF

C#

```
public Dictionary<string, object> ChangeSummaryType(string action, string
currentReport, string summaryType)
{
pivotClient.PopulateData(currentReport);
return pivotClient.GetJsonData(action, ProductSales.GetSalesData(),
summaryType);
}
```

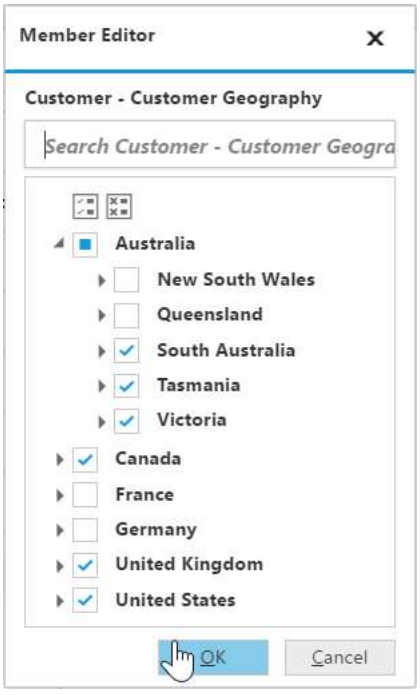
OLAP

Data Exploration

Filtering

Filtering by Member

After clicking Split button of a dimension, Member Editor dialog opens through which members are filtered by checking and unchecking the check boxes corresponding to the members. On clicking the "OK" button, based on the selected members in the Member Editor dialog, OlapReport gets updated and refreshes the PivotGrid and PivotChart controls. The "Cancel" button is used to cancel the selection.



The above filter illustrates that the members France and Germany, along with New South Wales and Queensland are excluded from the Grid and Chart controls.

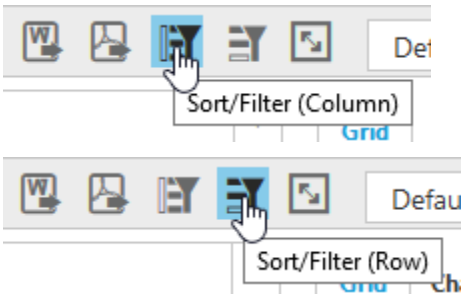
Filtering by Value

Information: This feature is applicable only for OLAP data source bound from server-side.

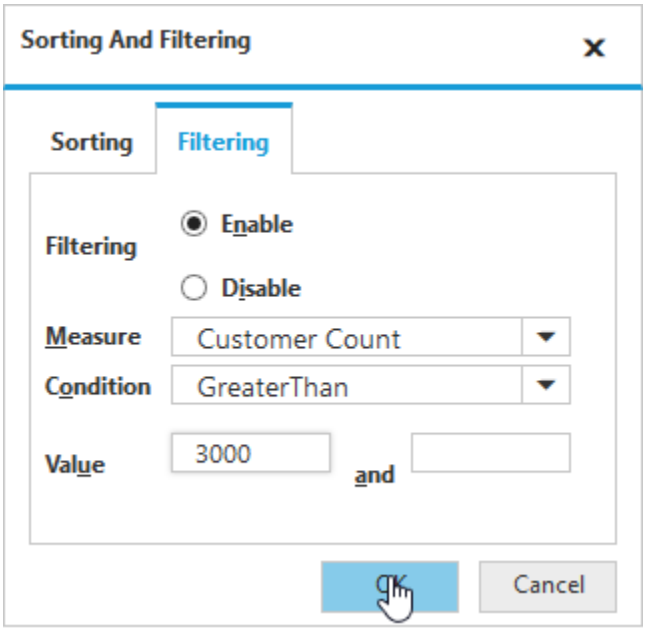
The Filtering tab in the Sorting and Filtering dialog box of PivotClient provides the options to apply custom filters on the multi-dimensional data. It enables the user to filter the rows and columns of the selected Measure.

- **Column Filter** - Filters the columns related to the specified measure according to the condition applied for filtering.
- **Row Filter** - Filters the rows related to the specified measure according to the condition applied for filtering.

Sorting and Filtering dialog box for rows and columns are opened by clicking the corresponding icon in the toolbar.



The following screenshot displays the Filtering tab in Sorting and Filtering dialog box.



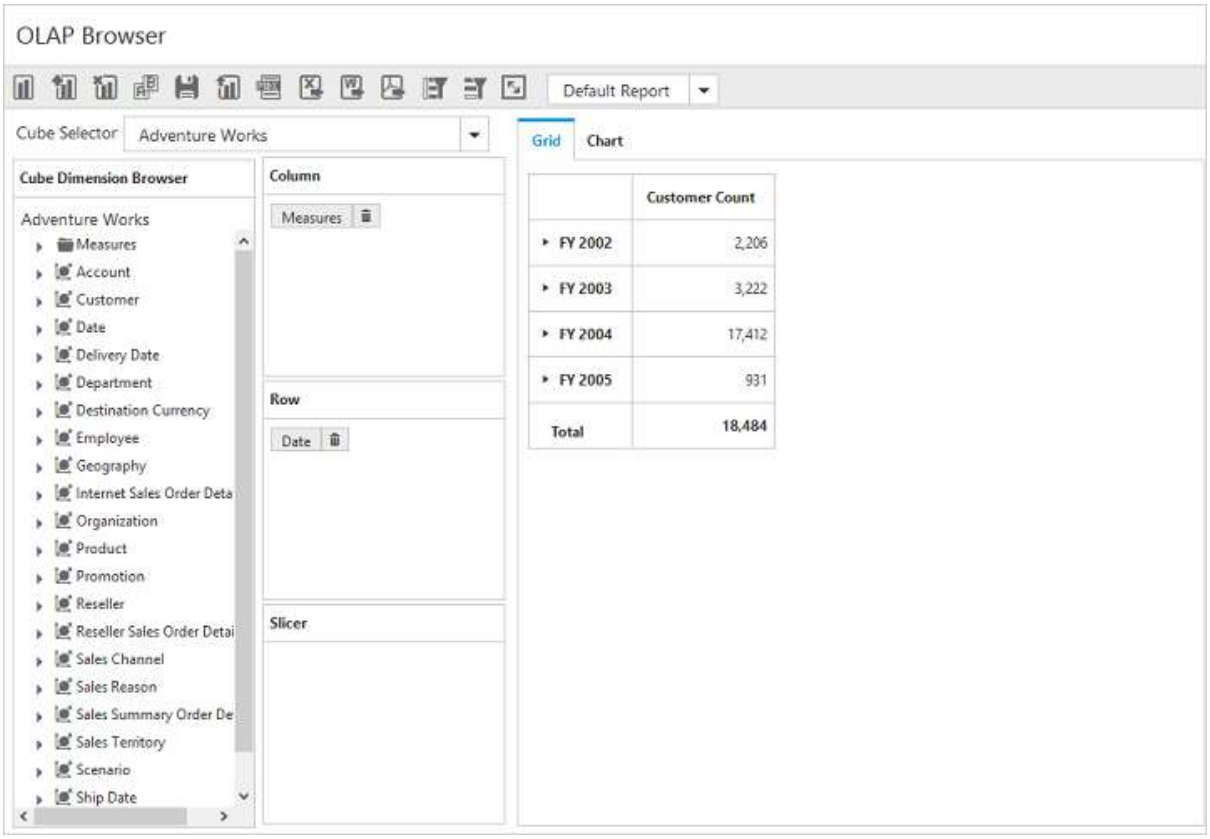
The options in the Filtering tab are as follows:

- **Enabling Filtering** – Enables/Disables the Filtering option.
- **Measure** – The measure for filtering is selected from the collection of measures in drop down list.
- **Condition** – Condition with which the filtering is applied.

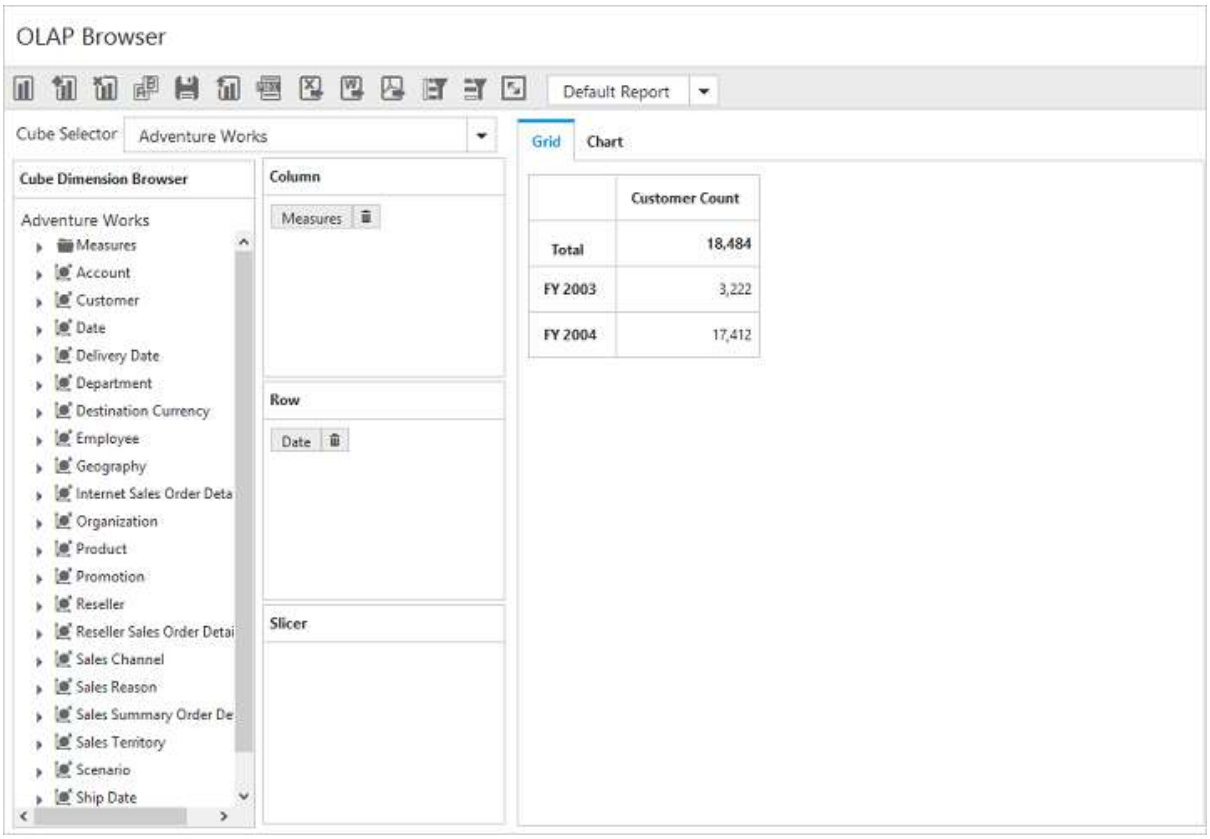
Copyright © 2001 - 2020 Syncfusion Inc.

- **Value** – Value to compare with each data according to the condition.

The following screenshot displays data before Filtering.



The following screenshot displays the data after Filtering.

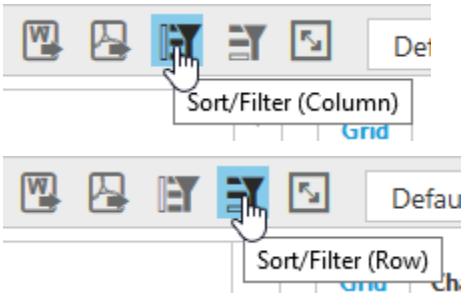


Sorting

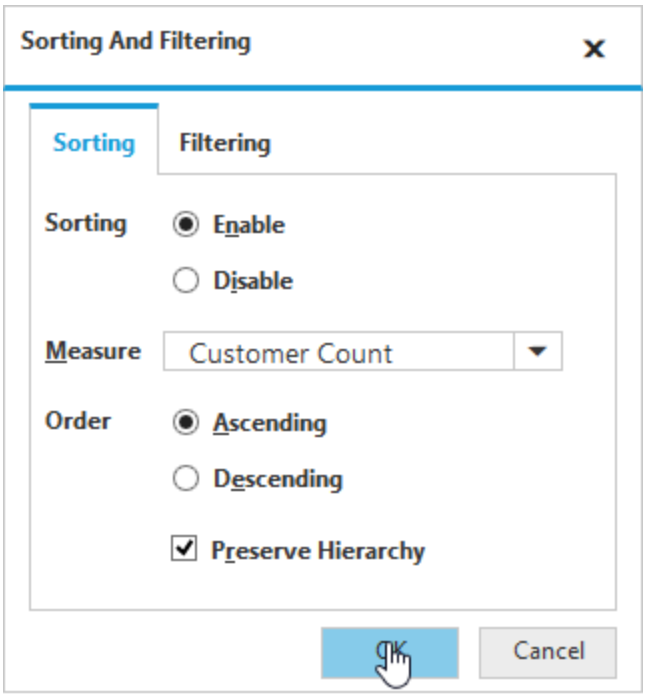
The Sorting tab in the Sorting And Filtering dialog box provides the option to sort the results by rows/columns, either in ascending or descending order.

- **Column sorting** - Sorts the columns based on the summary values of each column.
- **Row sorting** - Sorts the rows based on the summary values of each row.

Sorting And Filtering dialog box for rows and columns are opened by clicking the corresponding icon in the toolbar.



The following screenshot displays the Sorting And Filtering dialog box.

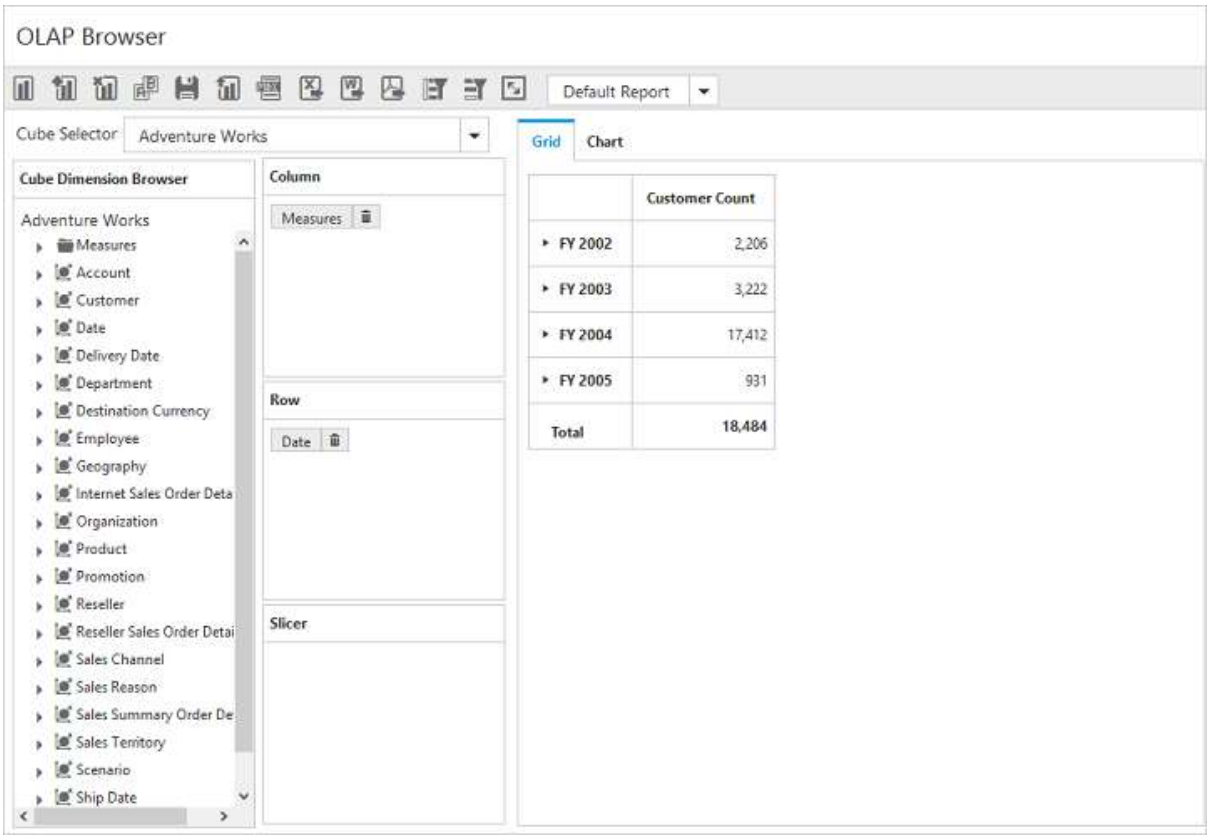


The options in the Sorting tab are as follows:

- **Measure** – The measure for sorting is selected from the collection of measures in drop-down list.
- **Order** – Specifies the sorting order.
- **Preserve Hierarchy** – Sorts the records without changing the hierarchy order.

The following screenshot displays the data after applying sorting in ascending order for rows.

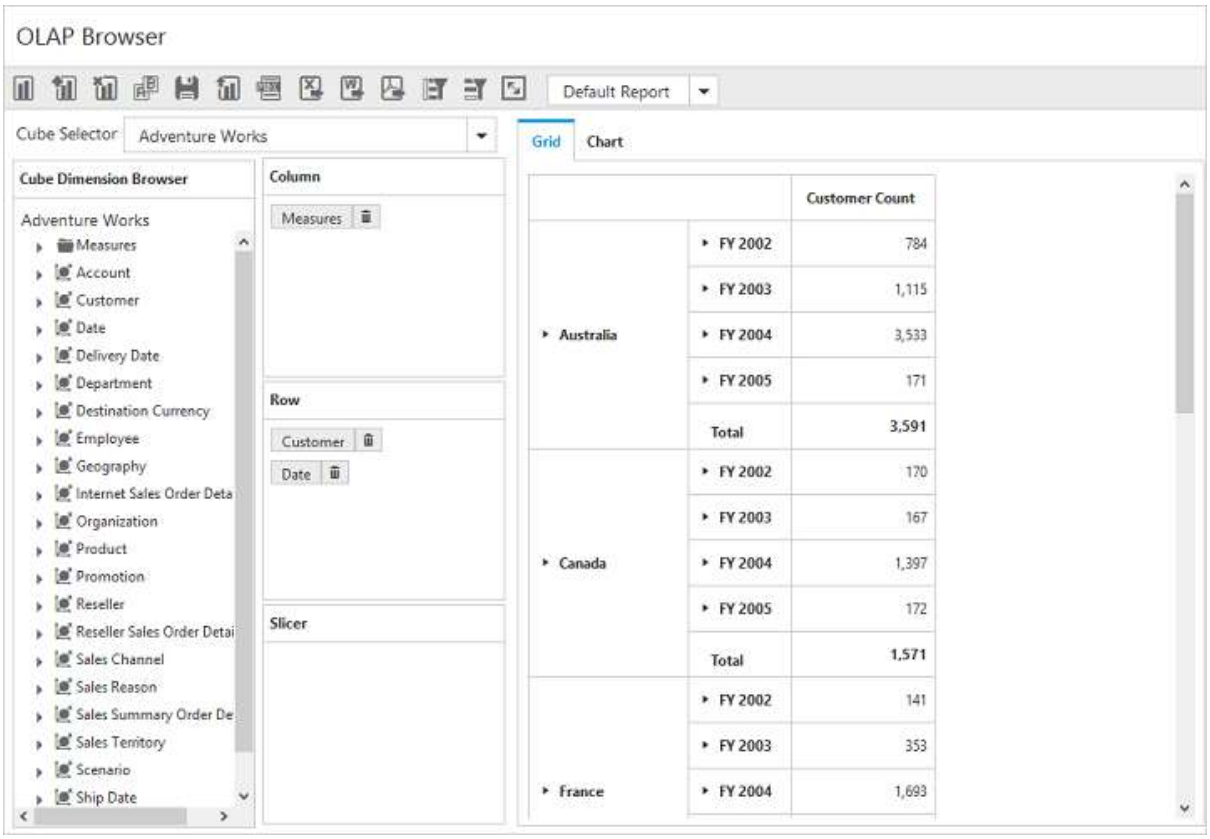
Copyright © 2001 - 2020 Syncfusion Inc.



Grouping

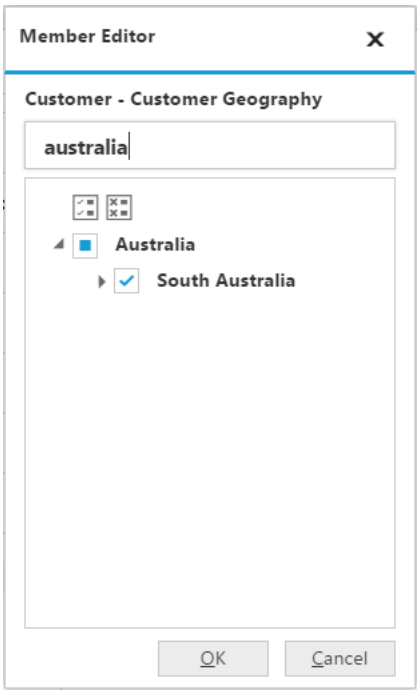
The data can be grouped when more than one dimension element is added to column or row in Axis Element Builder. Based on the order of addition, data is grouped and the report is updated. In the following example, members of the **Date** dimension get grouped, with respect to the members of **Customer** dimension. Likewise, multiple dimension members can be grouped by dragging the elements from Cube Dimension Browser to Axis Element Builder.

Copyright © 2001 - 2020 Syncfusion Inc.



Searching

Members can be searched and displayed from the members list inside the Member Editor dialog.



Exporting

PivotChart and PivotGrid in the PivotClient widget can be exported to Excel, Word and PDF documents by clicking the respective toolbar icons.



Exporting feature provides an option that allows you to export either PivotChart or PivotGrid or both with the use of the property ClientExportMode.

The property ClientExportMode takes any one of the following value:

- **ChartAndGrid** – Exports both PivotChart and PivotGrid controls. This is the default mode.
- **ChartOnly** – Exports PivotChart control alone.
- **GridOnly** – Exports PivotGrid control alone.

JSON Export

Note: By default, exporting is done with the use of JSON Records maintained in client-side for both client and server modes

In order to perform exporting with the use of a custom service method, the service containing the exporting method is hosted and its link is given in url as shown below. Without giving any value to the 'url' property it takes our default exporting service link.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).ClientExportMode(ClientExportMode.ChartAndGrid).ClientSideEvents(oE
ve => { oEve.BeforeExport("Export"); })
<script type="text/javascript">
function Export(args) {
args.url = "ExportPivotClient";
}
</script>
```

When PivotClient is exported with JSON export mode, a method needs to be added in MVC controller file of the application.

C#

```
public void ExportPivotClient()
{
JavaScriptSerializer serializer = new JavaScriptSerializer() {
MaxJsonLength = Int32.MaxValue };
PivotClientExport pivotClient = new PivotClientExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotClient.ExportPivotClient(string.Empty, args,
System.Web.HttpContext.Current.Response);
}
```

Customize the export document name

The name of the document to be exported could be customized. Following code sample illustrates the same.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapServ
ice")).ClientExportMode(ClientExportMode.ChartAndGrid).ClientSideEvents(o
Eve => { oEve.BeforeExport("Export"); })
<script type="text/javascript">
function Export(args) {
args.url = "ExportPivotClient";
args.fileName=" File name is customized here ";
}
</script>
```

PivotEngine Export

Information: This feature is applicable only at server mode operation.

In order to perform exporting with the use of PivotEngine available in server-side, the 'exportMode' property obtained in the “BeforeExport” event is set to "ej.PivotClient.ExportMode.PivotEngine" as shown below.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).ClientExportMode(ClientExportMode.ChartAndGrid).ClientSideEvents(oE
ve => { oEve.BeforeExport("Export"); })
<script type="text/javascript">
function Export(args) {
args.exportMode = ej.PivotClient.ExportMode.PivotEngine;
}
</script>
```

For WebAPI controller, the below method needs to be added to perform exporting with PivotEngine.

C#

```
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
olapClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

For WCF service, the below service method needs to be added to perform exporting with PivotEngine.

C#

```
public void Export(Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string fileName = "Sample";
olapClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

File format selection

Information: This option is applicable only for PivotClient when exporting to Excel document.

You can set the option for exporting the control to Excel document either in *.xls* or *.xlsx* format, using fileFormat property inside the BeforeExport event.

Note: By default excel document will be exported to ".xls" format using PivotEngine export.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(oEve => {
oEve.BeforeExport("Exporting"); })
<script type="text/javascript">
function Exporting(args) {
args.exportMode = ej.PivotClient.ExportMode.PivotEngine;
args.fileFormat = ".xlsx"; //you can set the excel sheet format here
}
</script>
```

Customize the export document name

The document name could be customized inside the method in WebAPI Controller. Following code sample illustrates the same.

C#

```
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = " File name is customized here ";
olapClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

For customizing name in WCF Service, below code snippet is used.

C#

```
public void Export(Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName =  " File name is customized here ";
olapClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

PivotChart - Exporting Format

Information: This option is applicable only for PivotChart in PivotClient specifically when exported to Excel document.

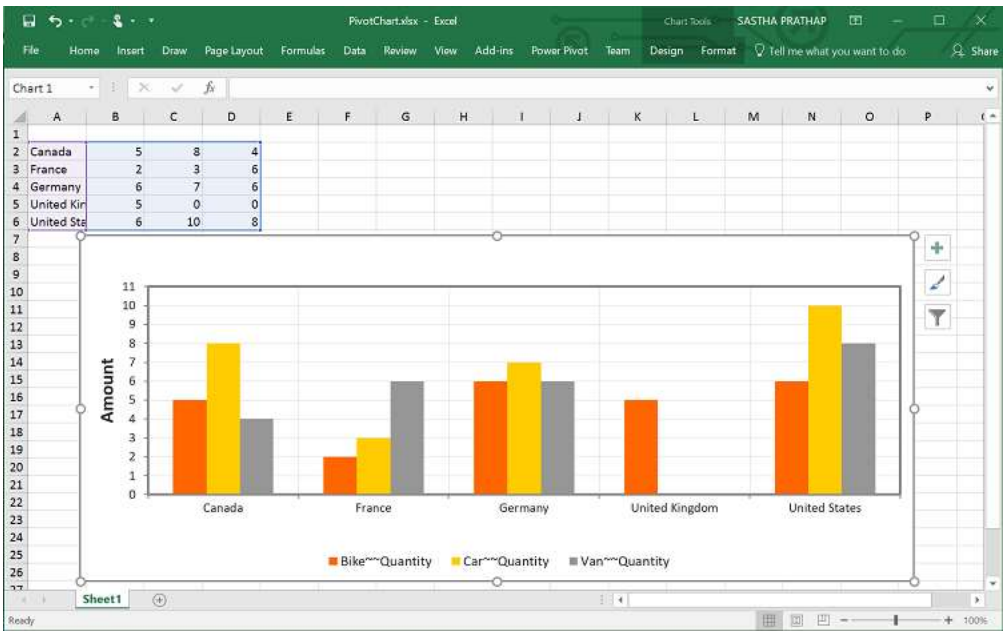
You can set an option to export PivotChart to an Excel document, either as image or PivotChart format itself by setting the Boolean property exportChartAsImage, inside the BeforeExport event.

Note: By default PivotChart will be exported as image format to Excel document.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).ClientExportMode(ClientExportMode.ChartOnly).ClientSideEvents(oEve
=> { oEve.BeforeExport("Export"); })
<script type="text/javascript">
function Export(args) {
args.exportChartAsImage = false; //You can set the chart format here
}
</script>
```

The below screenshot shows the control exported to Excel document showing its own format (Pivoting Chart).



Exporting Customization

You can add title and description to the exporting document by using the title and description properties respectively obtained in the BeforeExport event. Similarly, you can enable or disable styling on the exported document by using the exportWithStyle property.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(oEve => {
oEve.BeforeExport("Exporting"); })
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function Exporting(args) {
//ClientMode export
args.url = "ExportPivotClient";
//PivotEngine Export
args.exportMode = ej.PivotClient.ExportMode.PivotEngine;
args.title = "PivotClient";
args.description = "Visualizes both OLAP and Relational datasource in
tabular and graphical formats";
args.exportWithStyle = true;   // by default it sets as true. It improves
performance on exporting huge data when it sets as false.
}
</script>
```

You can also edit the exporting document with the use of a server side event for required exporting option.

C#

```
//...
using Syncfusion.EJ.Export;
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
//Following methods needs to be added in MVC controller file of the
application for JSON Export.
public void ExportPivotClient()
{
JavaScriptSerializer serializer = new JavaScriptSerializer() {
MaxJsonLength = Int32.MaxValue };
PivotClientExport pivotClient = new PivotClientExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
pivotClient.ExcelExport += pivotClient_ExcelExport;
pivotClient.WordExport += pivotClient_WordExport;
pivotClient.AddPDFHeaderFooter += pivotClient_AddPDFHeaderFooter;
pivotClient.PDFExport += pivotClient_PDFExport;
pivotClient.ExportPivotClient(string.Empty, args,
System.Web.HttpContext.Current.Response);
}
void pivotClient_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void pivotClient_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void pivotClient_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
void pivotClient_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
//Following service method needs to be added in WebAPI controller for
PivotEngine Export.
[System.Web.Http.ActionName("ExportOlapClient")]
[System.Web.Http.HttpPost]
public void ExportOlapClient()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
olapClientHelper.ExcelExport += olapClientHelper_ExcelExport;
olapClientHelper.WordExport += olapClientHelper_WordExport;
olapClientHelper.AddPDFHeaderFooter +=
olapClientHelper_AddPDFHeaderFooter;
olapClientHelper.PDFExport += olapClientHelper_PDFExport;
string fileName = "Sample";
olapClientHelper.ExportPivotClient(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
void olapClientHelper_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void olapClientHelper_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void olapClientHelper_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
void olapClientHelper_ExcelExport(object sender,
Syncfusion.XlsIO.IWorkbook workBook)
{
//You can customize exporting document here.
}
```

Exporting complete data on Paging

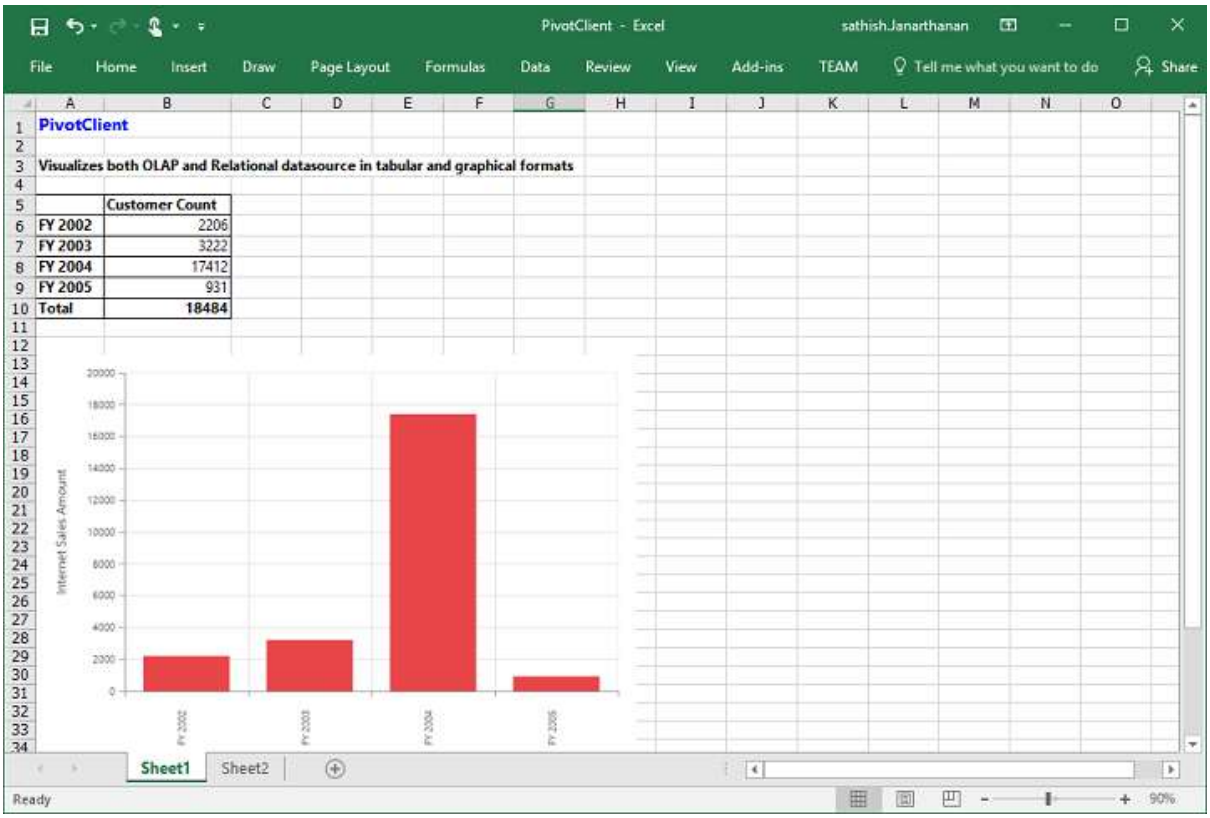
When paging is enabled, you can export the complete data by enabling the `EnableCompleteDataExport` property. It is supported in both types of JSON and PivotEngine export and it is applicable for all kinds of exporting formats available in PivotClient.

ASPX-CS

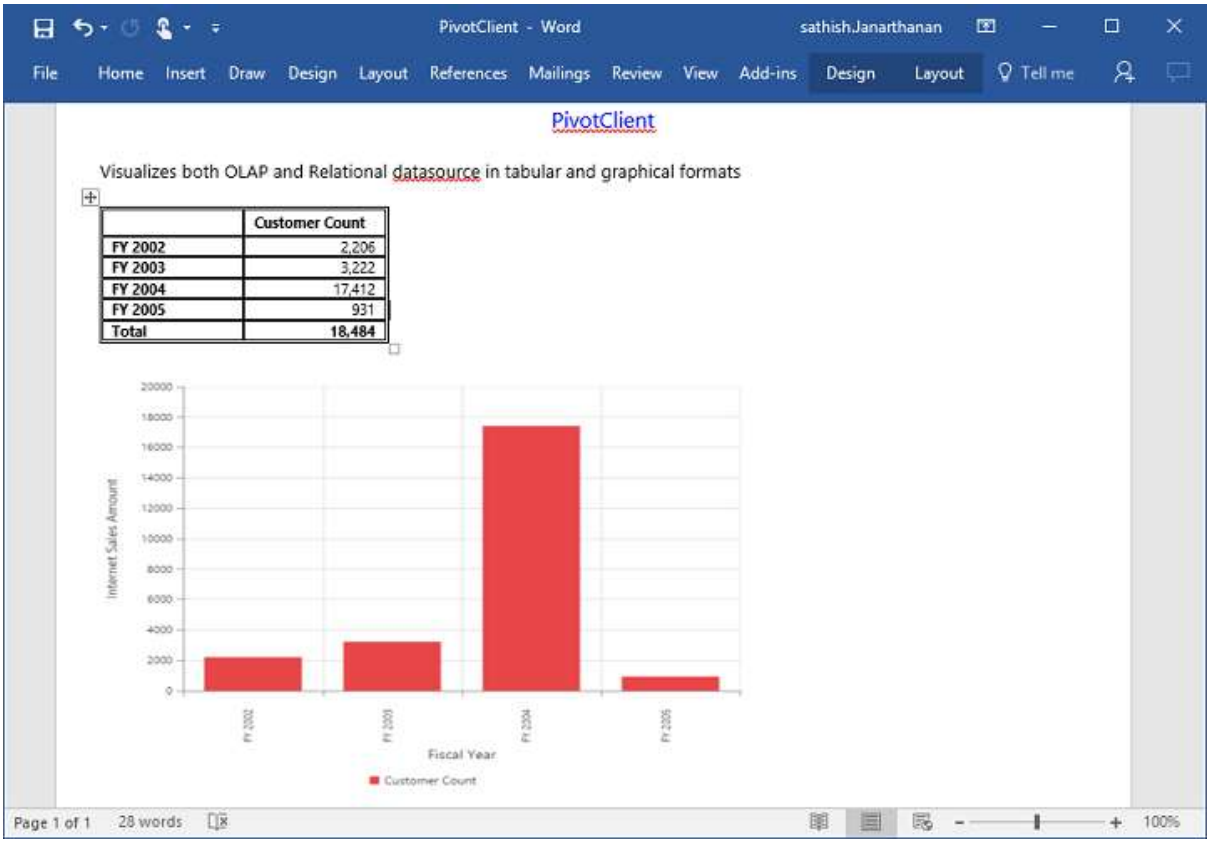
```
@Html.EJ().Pivot().PivotClient("PivotClient1").EnableCompleteDataExport(t
rue)
```

The below screenshot shows the PivotGrid and PivotChart controls exported to Excel document.

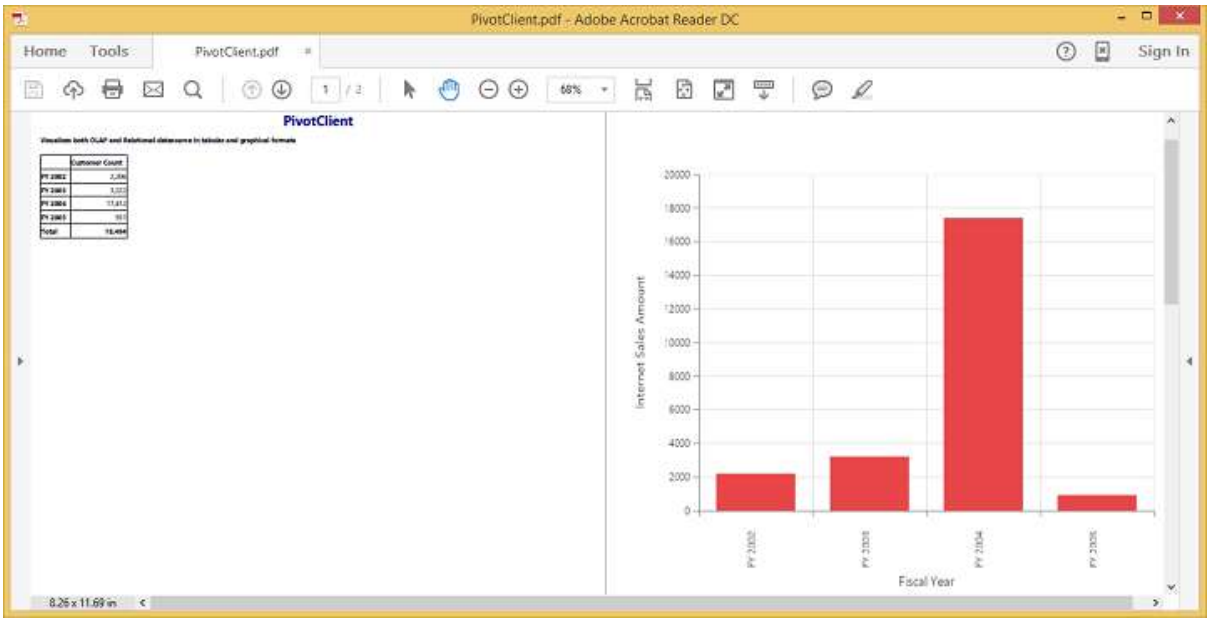
Copyright © 2001 - 2020 Syncfusion Inc.



The below screenshot shows the PivotGrid and PivotChart controls exported to Word document.



The below screenshot shows the PivotGrid and PivotChart controls exported to PDF document.



Defer Update

Information: This feature is applicable only for OLAP data source bound from server-side.

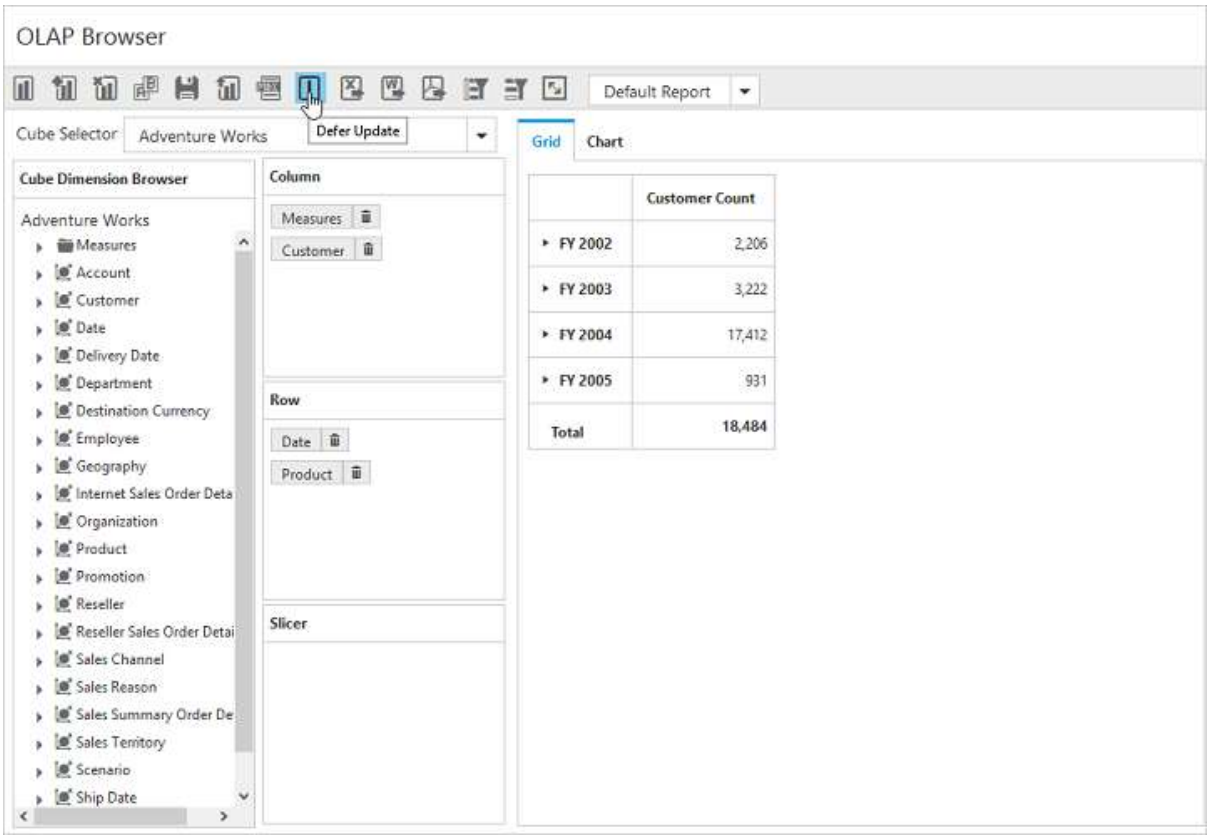
Defer Update support allows the user to refresh the control on-demand and not during every user interaction. To enable this functionality, set the EnableDeferUpdate property to true. By default, the value is set to false.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).Title("Olap Browser").EnableDeferUpdate(true)
```

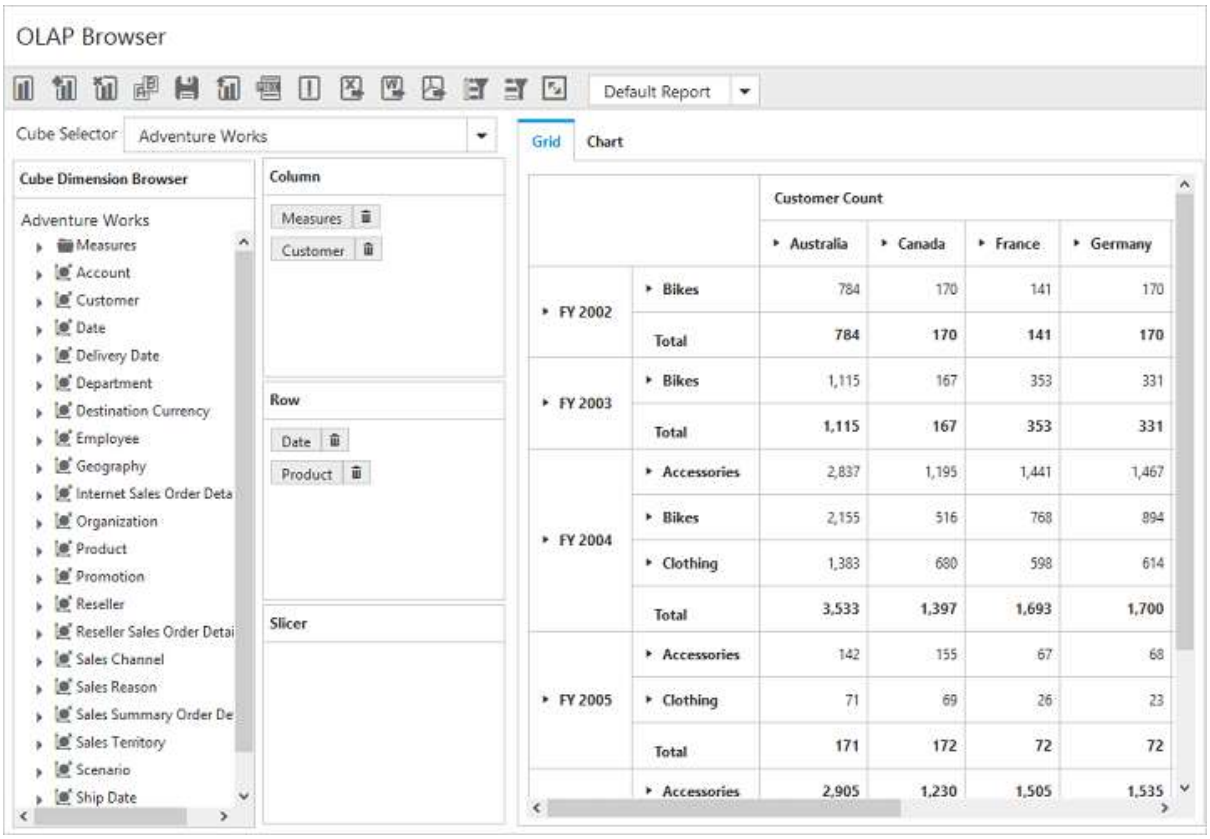
After enabling this property, an icon for Defer Update will appear inside the toolbar.

Copyright © 2001 - 2020 Syncfusion Inc.



On clicking the icon, after making the necessary UI interactions, the PivotGrid and PivotChart controls will be updated according to the OlapReport available at that instant.

Copyright © 2001 - 2020 Syncfusion Inc.



KPI

Key Performance Indicators (KPIs) are business metric that help to figure out the progress of an enterprise in meeting its business goals.

The different indicators available in KPI are:

- KPI Value: A physical measure or a calculated measure.
- KPI Goal: Defines the target for the measure.
- KPI Status: Evaluates the current status of the value compared to the goal.
- KPI Trend: Evaluate the current trend of the value compared to the goal.

The **“KpiElements”** class in OLAP Base library holds the KPI name and when its object is added to an OlapReport, you can view the resultant information in PivotClient.

To enable KPI option set the property `EnableKPI` to `true`.

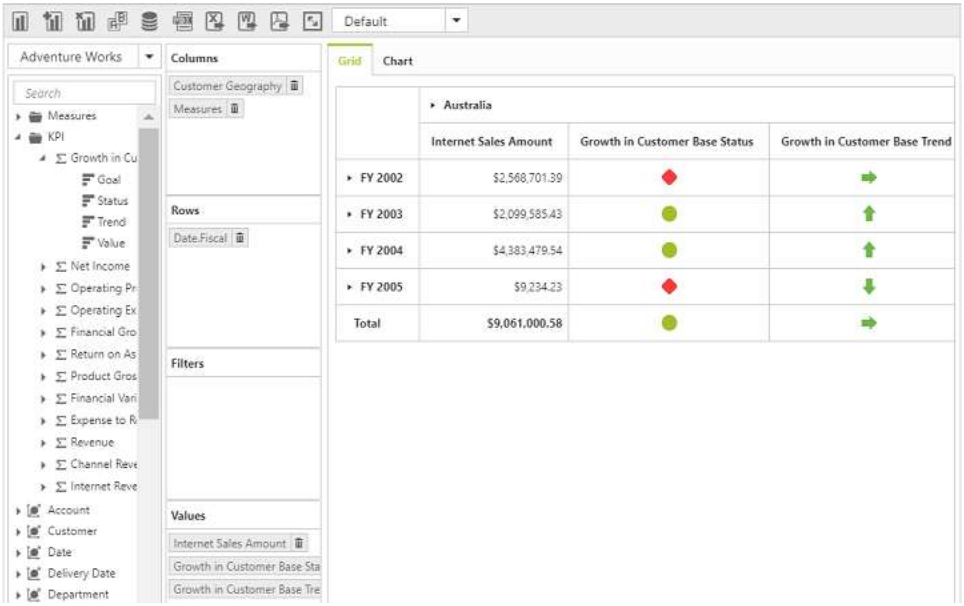
Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Date].[Fiscal]").Add();
}).Columns(columns => { columns.FieldName("[Product].[Product
Categories]").Add(); }).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
measures.FieldName("[Measures].[Growth in Customer Base Trend]").Add();
measures.FieldName("[Measures].[Growth in Customer Base Status]").Add();
}).Axis(AxisName.Column).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Data("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works")).EnableKPI(true)
```



Server Mode

Client side property declaration

ASPX-CS

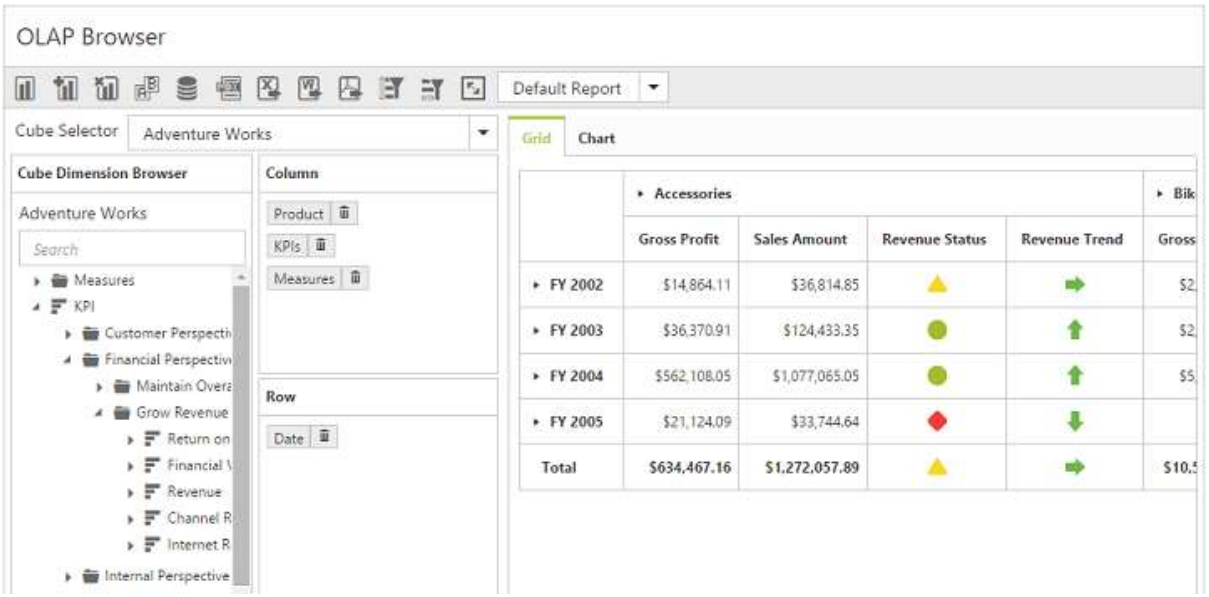
```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClient")).Title("OLAP Browser").EnableKPI(true)
```

Framing OlapReport with KPI Elements

C#

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "KPI Report";
olapReport.CurrentCubeName = "Adventure Works";
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement {
Name = "Gross Profit"
});
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
KpiElements kpiElement = new KpiElements();
kpiElement.Elements.Add(new KpiElement {
Name = "Revenue", ShowKPIStatus = true, ShowKPIGoal = false, ShowKPITrend = true, ShowKPIValue = true
});
olapReport.CategoricalElements.Add(kpiElement);
olapReport.CategoricalElements.Add(measureElementColumn);
olapReport.SeriesElements.Add(dimensionElementRow);
```

Copyright © 2001 - 2020 Syncfusion Inc.



Measure Groups

Information: This feature is applicable only for OLAP data source bound from server-side.

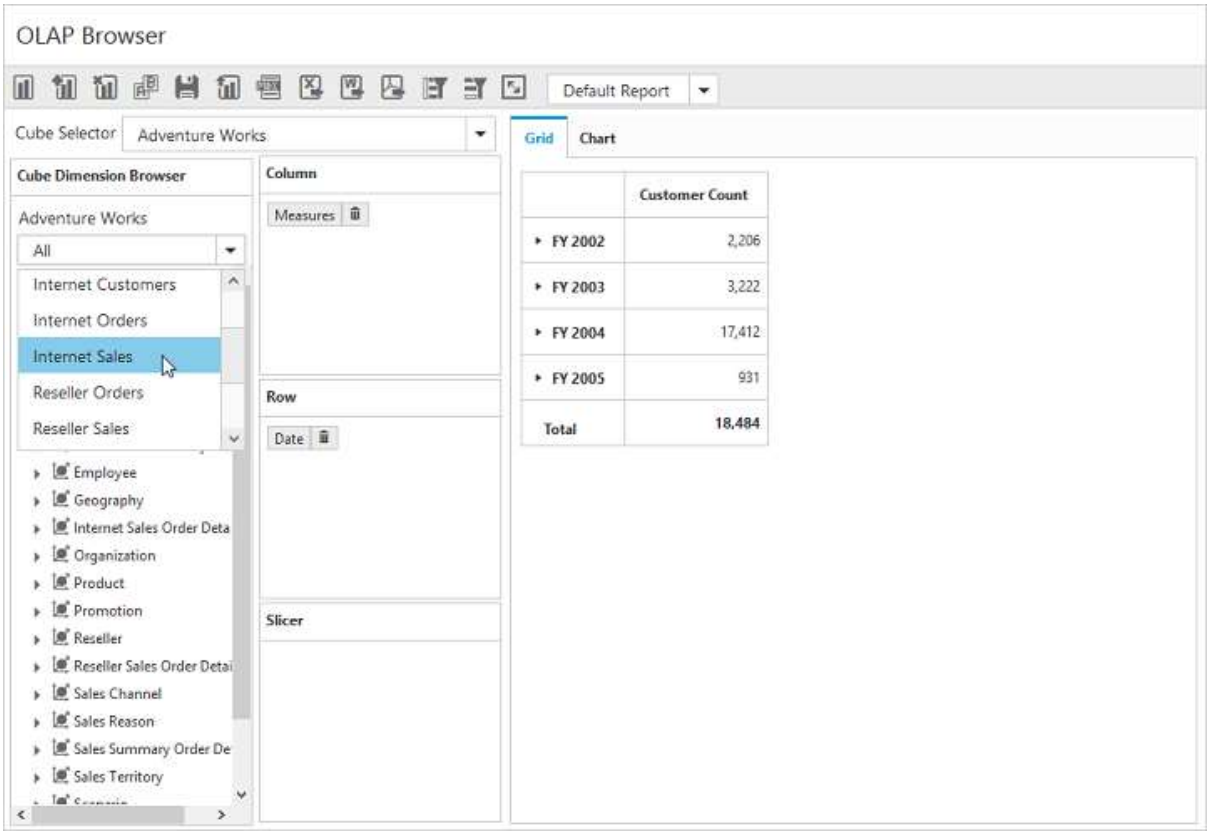
In Cube Dimension Browser, treeview can be viewed in a filtered manner through the Measures option. This feature allows you to view the list of measure groups and dimensions associated with the selected measure group from the cube. For enabling this, the EnableMeasureGroups property is set to true. By default, its value is set to false.

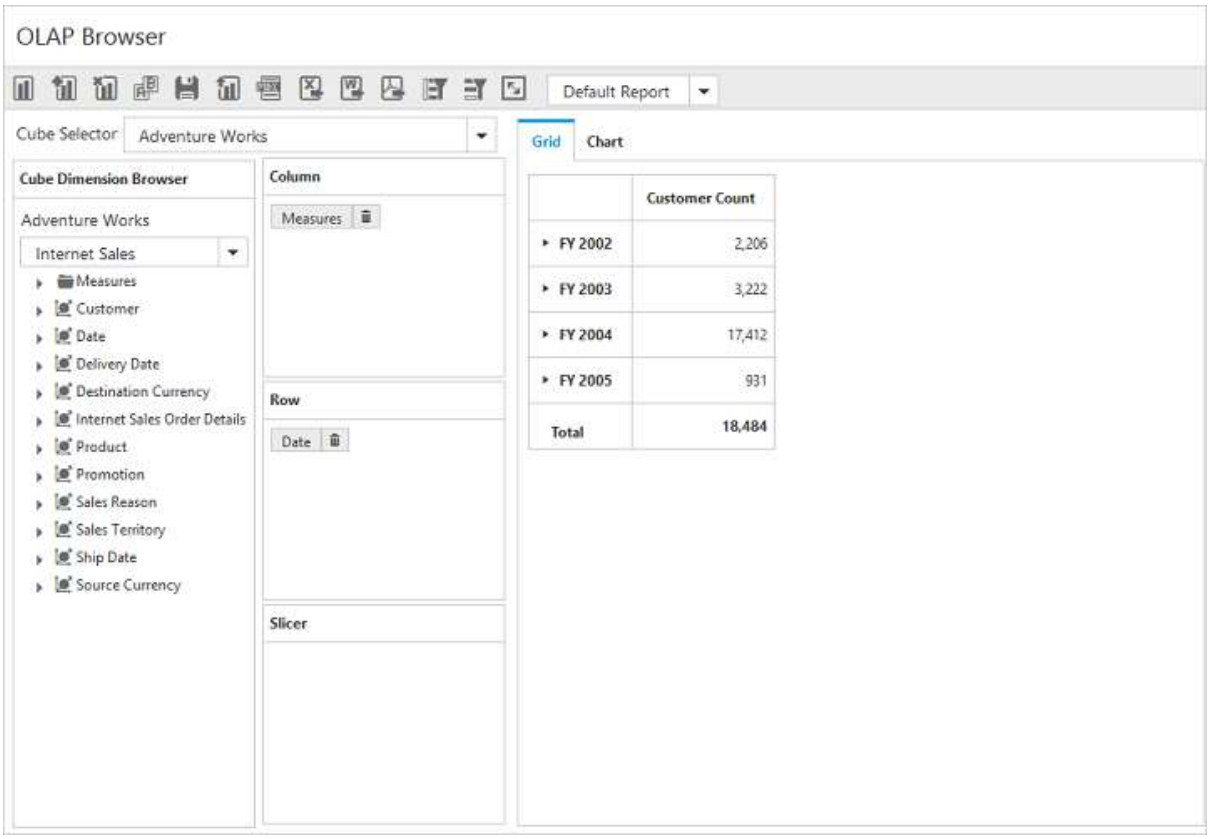
ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).Title("OLAP Browser").EnableMeasureGroups(true)
```

On selecting a measure group from the drop-down list, the Cube Dimension Browser treeview displays the related measures and dimensions as follows.

Copyright © 2001 - 2020 Syncfusion Inc.


OLAP Browser
Default Report
Cube Selector
Adventure Works
Cube Dimension Browser
Adventure Works
All
Internet Customers
Internet Orders
Internet Sales
Reseller Orders
Reseller Sales
Employee
Geography
Internet Sales Order Deta
Organization
Product
Promotion
Reseller
Reseller Sales Order Detai
Sales Channel
Sales Reason
Sales Summary Order De
Sales Territory
Column
Measures
Row
Date
Slicer
Grid
Chart
Customer Count
FY 2002 2,206
FY 2003 3,222
FY 2004 17,412
FY 2005 931
Total 18,484


OLAP Browser
Default Report
Cube Selector
Adventure Works
Cube Dimension Browser
Adventure Works
Internet Sales
Measures
Customer
Date
Delivery Date
Destination Currency
Internet Sales Order Details
Product
Promotion
Sales Reason
Sales Territory
Ship Date
Source Currency
Column
Measures
Row
Date
Slicer
Grid
Chart
Customer Count
FY 2002 2,206
FY 2003 3,222
FY 2004 17,412
FY 2005 931
Total 18,484

Paging

Information: This feature is applicable only for OLAP data source.

Paging helps to improve the rendering performance of the PivotClient control by dividing large amount of data into sections and displaying one section at a time.

Using Pager

You can enable Pager in PivotClient by setting the EnablePaging property to true. You can provide the page size and current page details for each axis through PagerOptions property.

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.PagerOptions(pagerOptions => {
pagerOptions.CategoricalPageSize(3).SeriesPageSize(3).CategoricalCurrentP
age(1).SeriesCurrentPage(1); })).EnablePaging(true)
```

Server Mode

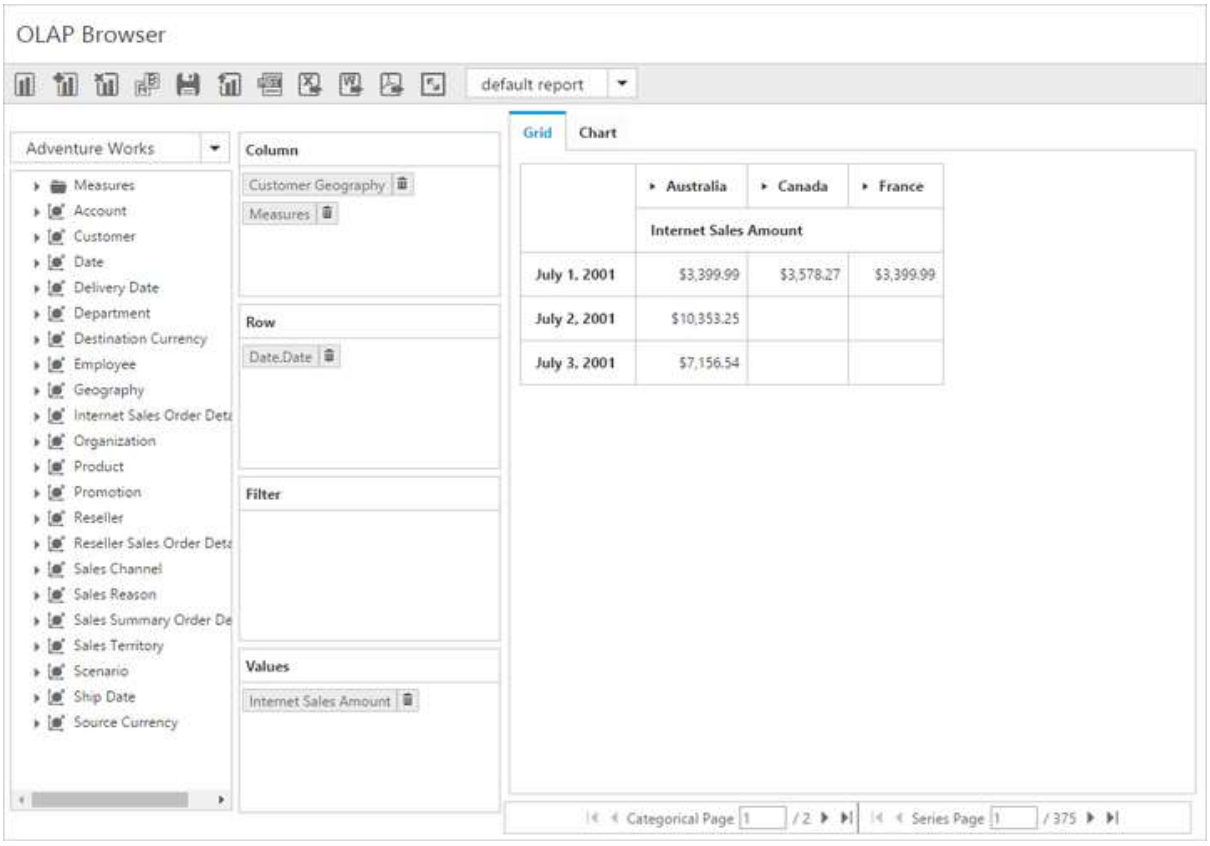
ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").EnablePaging(True).Url("/O
lapService")
```

Following are the navigation option available in Pager.

- Move First - Navigates to the first page.
- Move Last - Navigates to the last page.
- Move Previous - Navigates to the previous page from the current page.
- Move Next - Navigates to the next page from the current page.
- Numeric Box - Navigates to the desired page by entering an appropriate page number in numeric value.

Copyright © 2001 - 2020 Syncfusion Inc.



Using Virtual Scrolling

Virtual Scrolling is a technique that allows user to view the PivotClient information page by page with the use of vertical and horizontal scrollbar. You can enable Virtual Scrolling option in PivotClient by setting the enableVirtualScrolling property to true. You can provide the page size and current page details for each axis through PagerOptions property.

Client Mode

ASPX-CS

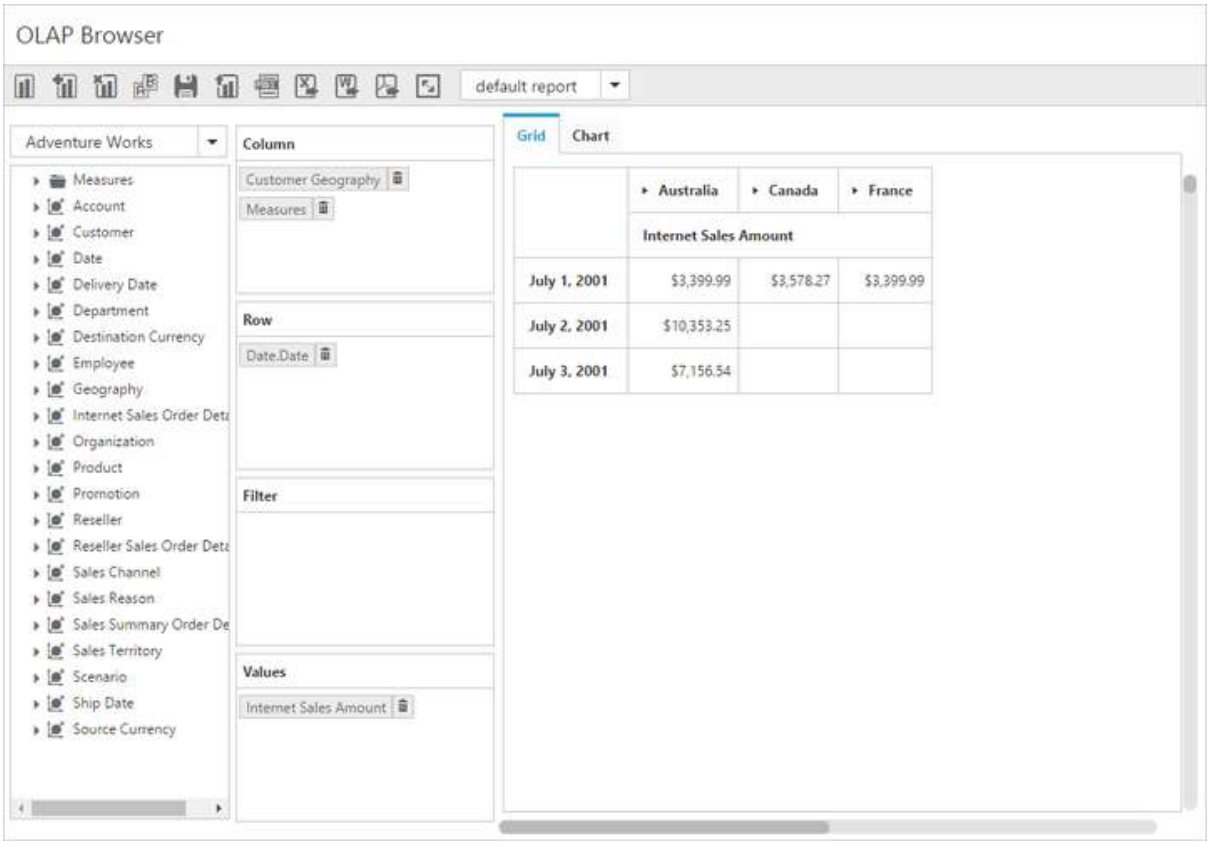
```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.PagerOptions(pagerOptions => {
pagerOptions.CategoricalPageSize(3).SeriesPageSize(3).CategoricalCurrentP
age(1).SeriesCurrentPage(1); })).enableVirtualScrolling(true)
```

Server Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").EnableVirtualScrolling(Tru
e).Url("/OlapService")
```

Copyright © 2001 - 2020 Syncfusion Inc.



Page Settings

The properties associated to paging are:

- EnablePaging – This property is used to enable/disable paging in PivotClient control.
- PagerOptions.CategoricalPageSize - Specifies the number of categorical columns to be displayed within a page of the PivotClient control.
- PagerOptions.SeriesPageSize - Specifies the number of series rows to be displayed within a page of the PivotClient control.
- PagerOptions.CategoricalCurrentPage - Sets the current page of the categorical axis in PivotClient control.
- PagerOptions.SeriesCurrentPage - Sets the current page of the series axis in PivotClient control.

For client mode, the page setting for categorical and series axes are mandatorily needed to be set in data source property by using the following properties.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.PagerOptions(pagerOptions => {
pagerOptions.CategoricalPageSize(3).SeriesPageSize(3).CategoricalCurrentP
age(1).SeriesCurrentPage(1); })).EnablePaging(true)
```

For server mode, the page settings for categorical and series axes are done only through OlapReport object, created inside WebAPI or WCF file.

C#

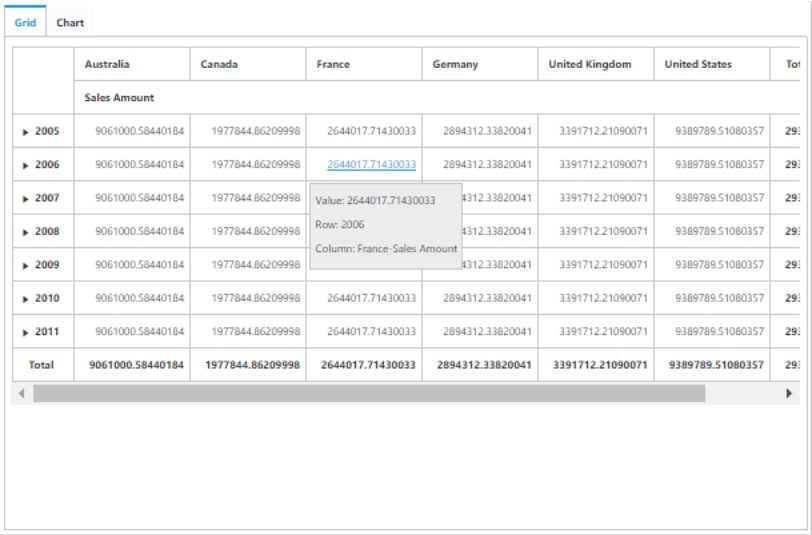
Copyright © 2001 - 2020 Syncfusion Inc.

```
OlapReport olapReport = new OlapReport();
olapReport.CurrentCubeName = "Adventure Works";
olapReport.EnablePaging = true;
olapReport.PagerOptions.SeriesPageSize = 4;
olapReport.PagerOptions.CategoricalPageSize = 5;
```

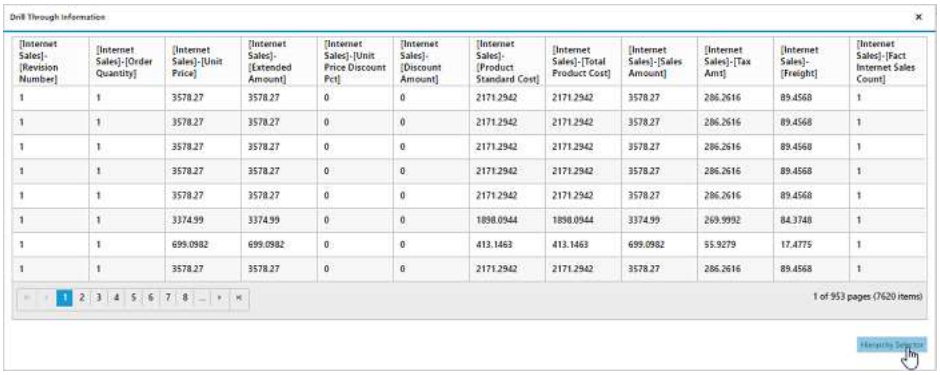
Drill Through

Drill-through retrieves the raw items that are used to create a specified cell. To enable drill-through support, set EnableDrillThrough property to true. Raw items are obtained through the drillThrough event, using which user can bind them to an external widget for precise view.

Note: Drill-through is supported in PivotGrid only when we configure and enable drill-through action at the Cube.

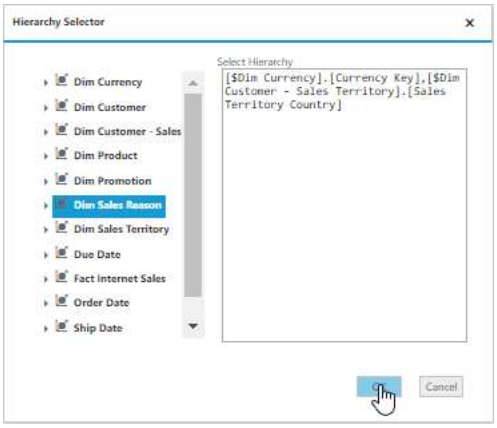


On clicking any value cell, the "Drill Through Information" dialog will be opened. It consists of Grid with data which are associated with the measure values of the clicked value cell. In this example, the measure behind the respective cell is “Sales Amount” and the values of the dimensions which are associated with this measure are alone displayed in the Grid.



On clicking the "Hierarchy Selector" button which is displayed below the Grid, the "Hierarchy Selector" dialog will be opened. It consists of the dimensions which are associated with the measure of clicked value cell. In this example, the measure behind the respective cell is “Sales Amount” and the dimensions associated with this measure are alone displayed in the dialog.

Copyright © 2001 - 2020 Syncfusion Inc.



By dragging and dropping the respective hierarchies and finally clicking “OK” button, Drill through MDX query will be framed and executed internally and provides back the raw items through "DrillThrough" event. In this example, we have bound the raw items obtained to our ejGrid widget. Please refer the code sample and screen-shot below.

OLAP

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(.....).EnableDr
illThrough(true).ClientSideEvents(oEve => {
oEve.DrillThrough("drilledData"); })
<script type="text/javascript">
function drilledData(args) {
$(".e-dialog, .e-clientDialog, .e-tableDlg").remove();
gridData = JSON.parse(args.data);
var dialogContent = ej.buildTag("div#" + this._id + "_tableDlg.e-
tableDlg", $("<div id=\"Grid1\"></div>"))[0].outerHTML;
var dialogFooter = ej.buildTag("div", ej.buildTag("button#btnOK.e-
dialogBtnOK", "Hierarchy Selector")[0].outerHTML, { "float": "right",
"margin": "-5px 0 6px" })[0].outerHTML
ejDialog = ej.buildTag("div#clientDialog.e-clientDialog", dialogContent +
dialogFooter, { "opacity": "1" }).attr("title", "Drill Through
Information")[0].outerHTML;
$(ejDialog).appendTo("#" + this._id);
$("#btnOK").ejButton().css({ margin: "30px 0 20px 0" });
$("#Grid1").ejGrid({
dataSource: gridData,
allowPaging: true,
allowTextWrap: true,
pageSettings: { pageSize: 8 }
});
this.element.find(".e-clientDialog").ejDialog({ width: "70%", content:
"#" + this._id, enableResize: false, close:
ej.proxy(ej.Pivot.closePreventPanel, this) });
var pivotClient = $("#" + this._id).data("ejPivotClient");
$("#btnOK").click(function () {
ej.Pivot.openHierarchySelector(pivotClient);
});
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Server Mode

HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).EnableDrillThrough(true).ClientSideEvents(oEve => {
oEve.DrillThrough("drilledData"); })
<script type="text/javascript">
function drilledData(args) {
$(".e-dialog, .e-clientDialog, .e-tableDlg").remove();
gridData = ej.isNullOrUndefined(args.data.d) ?
JSON.parse(args.data.DrillDataTable) : JSON.parse(args.data.d[1].Value);
var dialogContent = ej.buildTag("div#" + this._id + "_tableDlg.e-
tableDlg", $("<div id=\"Grid1\"></div>"))[0].outerHTML;
var dialogFooter = ej.buildTag("div", ej.buildTag("button#btnOK.e-
dialogBtnOK", "Hierarchy Selector")[0].outerHTML, { "float": "right",
"margin": "-5px 0 6px" })[0].outerHTML
ejDialog = ej.buildTag("div#clientDialog.e-clientDialog", dialogContent +
dialogFooter, { "opacity": "1" }).attr("title", "Drill Through
Information")[0].outerHTML;
$(ejDialog).appendTo("#" + this._id);
$("#btnOK").ejButton().css({ margin: "30px 0 20px 0" });
$("#Grid1").ejGrid({
dataSource: gridData,
allowPaging: true,
allowTextWrap: true,
pageSettings: { pageSize: 8 }
});
this.element.find(".e-clientDialog").ejDialog({ width: "70%", content:
"#" + this._id, enableResize: false, close:
ej.proxy(ej.Pivot.closePreventPanel, this) });
var pivotClient = this;
$("#btnOK").click(function () {
$(".e-dialog, .e-clientDialog, .e-tableDlg").remove();
if (pivotClient.model.operationalMode ==
ej.PivotGrid.OperationalMode.ServerMode) {
pivotClient._waitingPopup.show()
pivotClient.doAjaxPost("POST", pivotClient.model.url + "/" +
pivotClient.model.serviceMethodSettings.drillThroughHierarchies,
JSON.stringify({ "currentReport": pivotClient.currentReport, "layout":
pivotClient.model.layout, "cellPos": "", "customObject":
JSON.stringify(pivotClient.model.customObject) }), function (args) {
ej.Pivot.openHierarchySelector(pivotClient, args);
})
}
});
}
</script>
```

When PivotGrid is rendered in server mode, below service methods need to be added in WCF/WebAPI for drill through operation.

For WebAPI controller, the below methods need to be added.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

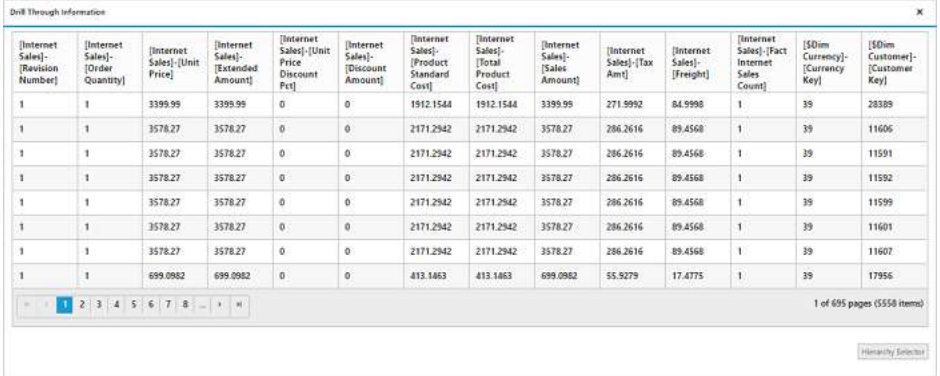
```
[System.Web.Http.ActionName("DrillThroughHierarchies")]
[System.Web.Http.HttpPost]
public string DrillThroughHierarchies(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return pivotClientHelper.DrillthroughHierarchies(DataManager,
jsonResult["layout"].ToString(), jsonResult["cellPos"].ToString());
}
[System.Web.Http.ActionName("DrillThroughDataTable")]
[System.Web.Http.HttpPost]
public Dictionary<string, object>
DrillThroughDataTable(Dictionary<string, object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return pivotClientHelper.DrillthroughDataTable(DataManager,
jsonResult["layout"].ToString(), jsonResult["cellPos"].ToString(),
jsonResult["selector"].ToString());
}
```

For WCF service, the below methods need to be added.

C#

```
public string DrillThroughHierarchies(string currentReport, string
layout, string cellPos)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return pivotClientHelper.DrillthroughHierarchies(DataManager, layout,
cellPos);
}
public Dictionary<string, object> DrillThroughDataTable(string
currentReport, string layout, string cellPos, string selector)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return pivotClientHelper.DrillthroughDataTable(DataManager, layout,
cellPos, selector);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Relational

To enable drill-through support, set the enableDrillThrough property to true. Raw items are obtained through the drillThrough event.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(.....).EnableDr
illThrough(true).ClientSideEvents(oEve => {
oEve.DrillThrough("drilledData").Load("onLoad"); })
```

HTML

```
<script type="text/javascript">
function onLoad(args) {
this.model.analysisMode = "pivot";
}
function drilledData(args) {
gridData = args.selectedData;
var dialogContent = ej.buildTag("div#Grid", { })[0].outerHTML;
ejDialog = ej.buildTag("div#clientDialog.e-clientDialog", dialogContent,
{ "opacity": "1" }).attr("title", "Drill Through
Information")[0].outerHTML;
$(ejDialog).appendTo("#" + this._id);
this.element.find(".e-clientDialog").ejDialog({ width: "70%", content:
"#" + this._id, enableResize: false, close:
ej.proxy(ej.Pivot.closePreventPanel, this) });
$("#Grid").ejGrid({
dataSource: gridData,
allowScrolling: true,
scrollSettings: { width: "85%" },
allowPaging: true,
allowResizing: true,
allowResizeToFit: true,
pageSettings: {pageSize: 8}
});
}
</script>
![DrillThrough data in ASP NET MVC relational pivot
client](DrillThrough_images/relational_drilledData.png)
```

Calculated Members

Information: This feature is applicable only for Server Mode bound with OLAP data source.

Copyright © 2001 - 2020 Syncfusion Inc.

Calculated members are the customized dimension members or measures that are defined based on the cube data. Values for calculated members are computed at run-time.

The two types of calculated members are as follows:

Calculated Measure – Calculated measure created from a measure element.

Calculated Dimension – Calculated member created from a dimension element.

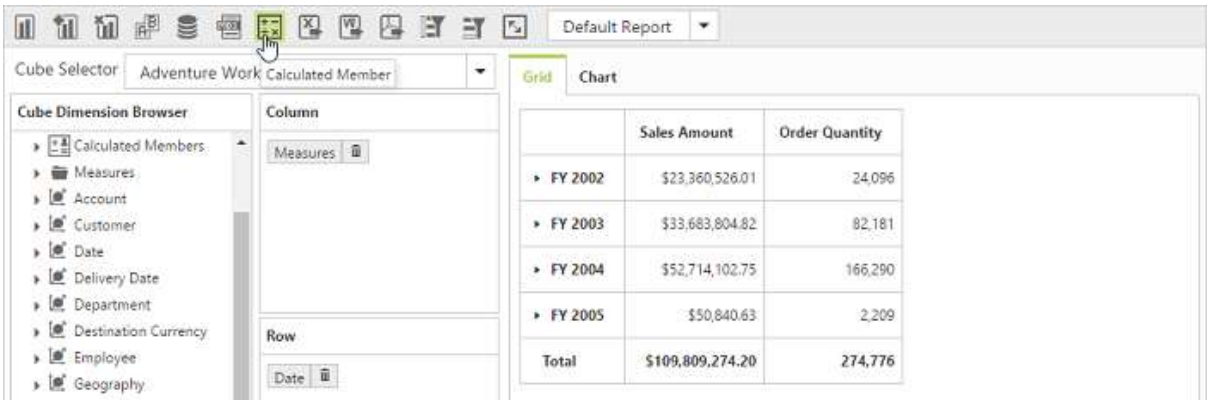
PivotClient provides support to insert new calculated members based on the existing OLAP field members either through UI dialog or code behind.

Through UI

To show the icon in toolbar panel, we need to set the property `EnableCalculatedMember` as `true` under `ToolbarIconSettings` object.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("~/api/OlapClient")).ToolbarIconSettings(toolbaricon => {
toolbaricon.EnableCalculatedMember(true); })
```



To insert a new calculated member, open the Calculated Member dialog by clicking the icon available in toolbar panel. We can define following options inside the dialog.

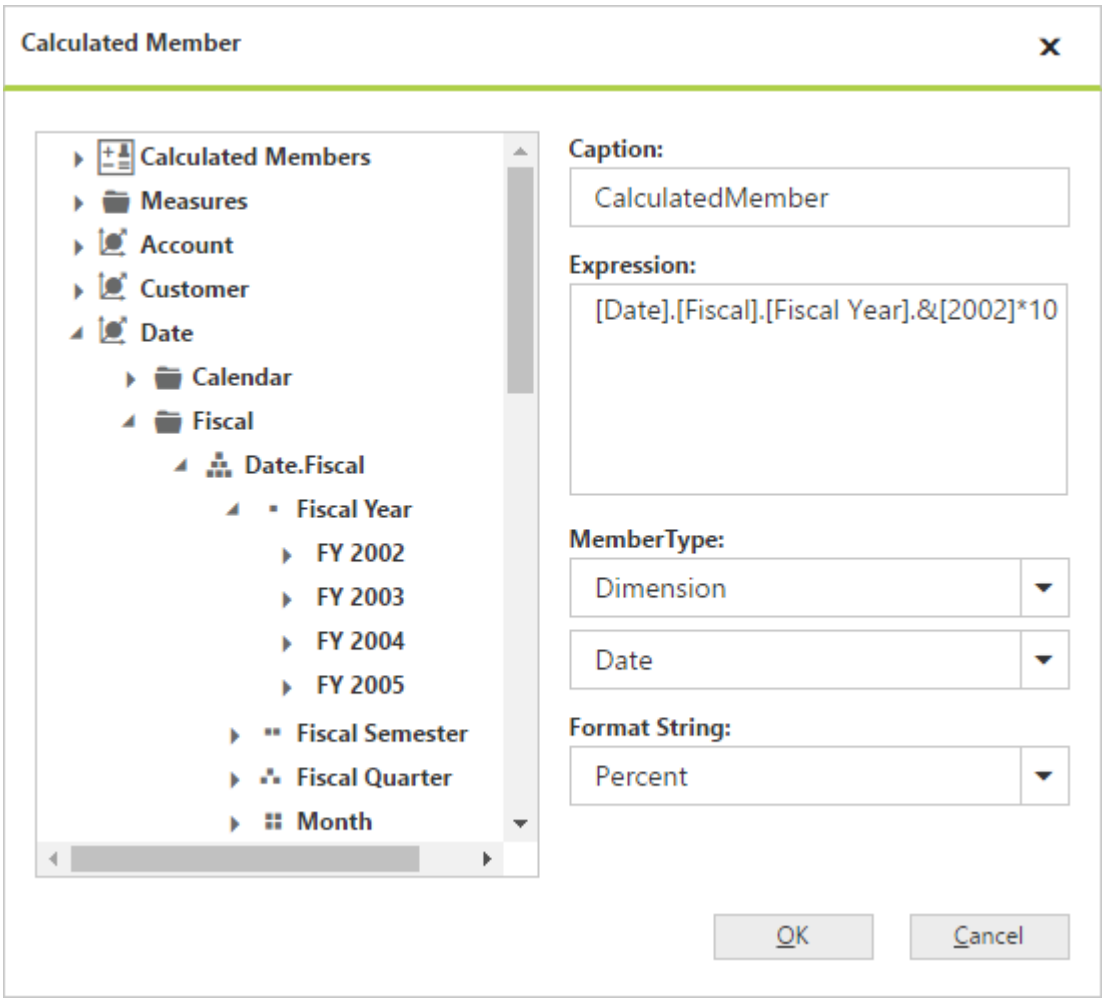
Caption - To set the name for the calculated member.

Expression - To set the formula for the calculated member where you can insert required members through Cube Dimension Browser by a simple drag and drop option with required operators to make formula.

Member Type - To specify the hierarchy of the member set in expression.

Format String - To set the format for the calculated member.

Copyright © 2001 - 2020 Syncfusion Inc.



Through Code Behind

For code-behind, we can create the calculated members by defining formula in OlapReport with the following definitions.

Name – Name of the calculated member.

Expression – Expression to form the calculated member.

Measure/Dimension Element – You should add a measure or dimension element from which the calculated member should be created.

Following steps will explain how to create and add a calculated member in an OlapReport:

1. Create the dimension measure or element from which the calculated member has to be created.
2. Create the calculated member by giving the name and expression.
3. Add the element created in Step 1 to the calculated member.
4. Once the calculated member is defined, add that member to the calculated members collection in the OlapReport.
5. Add the newly created calculated members in categorical or series axis of an OlapReport.

Following code snippet will describe the creation and addition of a calculated members in OlapReport:

Copyright © 2001 - 2020 Syncfusion Inc.

Calculated Measure

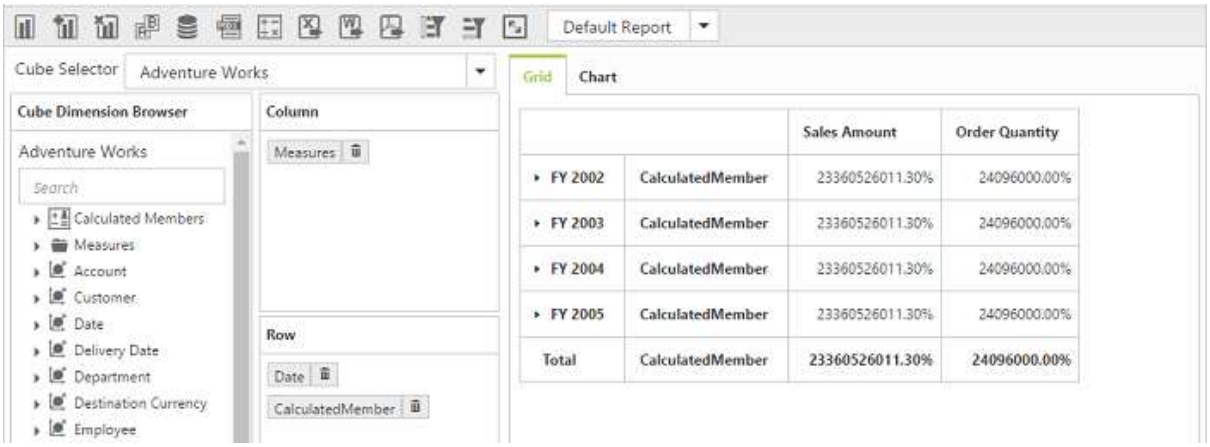
C#

```
MeasureElement measureElement = new MeasureElement();
measureElement.Name = "Order Quantity";
CalculatedMember calculatedMeasure = new CalculatedMember();
calculatedMeasure.Name = "Oder on Discount";
calculatedMeasure.Expression = "[Measures].[Order Quantity] +
([Measures].[Order Quantity] * 0.10)";
calculatedMeasure.AddElement(measureElement);
olapReport.CalculatedMembers.Add(calculatedMeasure);
olapReport.CategoricalElements.Add(calculatedMeasure);
```

Calculated Dimension

C#

```
DimensionElement internalDimension = new DimensionElement();
internalDimension.Name = "Product";
internalDimension.AddLevel("Product Categories", "Category");
// Calculated Dimension
CalculatedMember calculatedDimension = new CalculatedMember();
calculatedDimension.Name = "Bikes & Components";
calculatedDimension.Expression = "([Product].[Product
Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components] )";
calculatedDimension.AddElement(internalDimension);
olapReport.CalculatedMembers.Add(calculatedDimension);
olapReport.CategoricalElements.Add(calculatedDimension);
```



Common

Save and Load Report

Save and load report allows you to save the current report collection of PivotClient and render the control later on loading the same.

We can save and load the report in two ways.

- Database
- Local Storage

Copyright © 2001 - 2020 Syncfusion Inc.

Save Report to Database

We can store the report collection of PivotClient to database, by using the SaveReport event in PivotClient.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1”).ClientSideEvents(events =>
events.SaveReport("saveReportSettings"))
<script>
function saveReportSettings(args) {
if(args.saveReportSettings)
return args.saveReportSetting.url = "../wcf/OlapClientService.svc";
}
</script>
```

Service method needs to be added in WCF/WebAPI for storing PivotClient report collection in database.

For WebAPI controller, the below method needs to be added.

C#

```
[System.Web.Http.ActionName("SaveReportToDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> SaveReportToDB(Dictionary<string,
object> jsonResult)
{
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString(), reportName =
string.Empty;
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
reportName = jsonResult["reportName"].ToString() + "##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(jsonResult["clientReports"].ToString()).ToArray())
;
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
dictionary.Add("CurrentAction", "Save");
return dictionary;
}
```

For WCF controller, the below method needs to be added.

C#

```
public Dictionary<string, object> SaveReportToDB(string reportName,
string operationalMode, string analysisMode, string olapReport, string
clientReports)
{
reportName = reportName + "##" + operationalMode.ToLower() + "#>>#" +
analysisMode.ToLower();
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(clientReports).ToArray());
cmd1.ExecuteNonQuery();
con.Close();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("CurrentAction", "Save");
return dictionary;
}
```

Save Report to Local Storage

We can store the report collection of PivotClient to local storage, by setting the EnableLocalStorage property to true and by defining the SaveReport event of PivotClient.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1”).ClientSideEvents(events =>
events.SaveReport("saveReportSettings")).EnableLocalStorage(true)
<script>
function saveReportSettings(args) {
var reportCollection = [];
if ((localStorage.pivotClientRPTCollection != "" &&
!ej.isNullOrUndefined(localStorage.pivotClientRPTCollection))) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
reportCollection = JSON.parse(localStorage.pivotClientRPTCollection);
}
if(args.saveReportSettings){
reportCollection.push(({ reportName: args.saveReportSetting.reportName,
reportCol: args.saveReportSetting.reportCollection }));
localStorage.pivotClientRPTCollection =
JSON.stringify(reportCollection);;
}
}
</script>
```

Load Report from Database

We can load the stored report collection of PivotClient from database, by using the FetchReport and LoadReport events in PivotClient.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1”).ClientSideEvents(events =>
events.LoadReport("reportSettings").FetchReport(”reportSettings”))
<script>
function reportSettings(args) {
if (args.fetchReportSetting)
return args.fetchReportSetting.url = "../wcf/OlapClientService.svc";
else if (args.loadReportSetting)
return args.loadReportSetting.url = "../wcf/OlapClientService.svc";
}
</script>
```

Service methods need to be added in WCF/WebAPI for storing PivotClient report collection in database.

For WebAPI controller, the below methods need to be added.

C#

```
[System.Web.Http.ActionName("FetchReportListFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object>
FetchReportListFromDB(Dictionary<string, object> jsonResult)
{
string reportNames = string.Empty, currentRptName = string.Empty,
operationalMode = jsonResult["operationalMode"].ToString(), analysisMode
= jsonResult["analysisMode"].ToString();
foreach (System.Data.DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string);
if (currentRptName.IndexOf("##" + operationalMode + "#>>#" +
analysisMode) >= 0)
{
currentRptName = currentRptName.Replace("##" + operationalMode + "#>>#" +
analysisMode, "");
reportNames = reportNames == "" ? currentRptName : reportNames + "__" +
currentRptName;
}
}
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("ReportNameList", reportNames);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
dictionary.Add("action", jsonResult["action"].ToString());
return dictionary;
}
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
{
PivotReport report = new PivotReport();
string operationalMode = jsonResult["operationalMode"].ToString(),
analysisMode = jsonResult["analysisMode"].ToString();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
string currentRptName = string.Empty;
foreach (DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string).Replace("##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower(), "");
if (currentRptName.Equals(jsonResult["reportName"].ToString()))
{
byte[] reportByte = new byte[2 * 1024];
reportByte = (row.ItemArray[1] as byte[]);
if (operationalMode.ToLower() == "servermode" && analysisMode == "olap")
{
var repCol = Encoding.UTF8.GetString(reportByte);
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (repCol.IndexOf("<?xml version") == 0)
{
var reportString = "";
reportString =
Syncfusion.JavaScript.Olap.Utils.CompressData(row.ItemArray[1] as
byte[]);
DataManager.Reports =
pivotClientHelper.DeserializedReports(reportString);
DataManager.SetCurrentReport(DataManager.Reports[0]);
return pivotClientHelper.GetJsonData("toolbarOperation", DataManager,
"Load Report", jsonResult["reportName"].ToString());
}
else
{
dynamic customData = serializer.Deserialize<dynamic>(repCol.ToString());
DataManager.Reports =
pivotClientHelper.DeserializedReports(customData[customData[customData.Le
ngth - 1]["cubeIndex"]]["Reports"]);
DataManager.SetCurrentReport(DataManager.Reports[customData[customData[cu
stomData.Length - 1]["cubeIndex"]]["ReportIndex"]]);
dictionary = pivotClientHelper.GetJsonData("toolbarOperation",
DataManager, "Load Report", jsonResult["reportName"].ToString());
dictionary.Add("Collection", repCol);
}
}
else
{
if (analysisMode.ToLower() == "pivot" && operationalMode.ToLower() ==
"servermode")
dictionary = pivotClientHelper.GetJsonData("LoadReport",
ProductSales.GetSalesData(), Encoding.UTF8.GetString(reportByte));
else
```

```
dictionary.Add("report", Encoding.UTF8.GetString(reportByte));
break;
}
}
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

For WCF controller, the below method needs to be added.

C#

```
public Dictionary<string, object> FetchReportListFromDB(string action,
string operationalMode, string analysisMode)
{
string reportNames = string.Empty;
string currentRptName = string.Empty;
foreach (System.Data.DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string);
if (currentRptName.IndexOf("##" + operationalMode + "#>>#" +
analysisMode) >= 0)
{
currentRptName = currentRptName.Replace("##" + operationalMode + "#>>#" +
analysisMode, "");
reportNames = reportNames == "" ? currentRptName : reportNames + "__" +
currentRptName;
}
}
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("ReportNameList", reportNames);
dictionary.Add("action", action);
return dictionary;
}
public Dictionary<string, object> LoadReportFromDB(string
reportName,string operationalMode,string analysisMode, string olapReport,
string clientReports)
{
PivotReport report = new PivotReport();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
string currentRptName = string.Empty;
foreach (DataRow row in GetDataTable().Rows)
{
currentRptName = (row.ItemArray[0] as string).Replace("##" +
operationalMode.ToLower() + "#>>#" + analysisMode.ToLower(), "");
if (currentRptName.Equals(reportName))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
byte[] reportByte = new byte[2 * 1024];
reportByte = (row.ItemArray[1] as byte[]);
if (operationalMode.ToLower() == "servermode" && analysisMode == "olap")
{
var repCol = Encoding.UTF8.GetString(reportByte);
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (repCol.IndexOf("<?xml version") == 0)
{
var reportString = "";
reportString =
Syncfusion.JavaScript.Olap.Utils.CompressData(row.ItemArray[1] as
byte[]);
DataManager.Reports =
pivotClientHelper.DeserializedReports(reportString);
DataManager.SetCurrentReport(DataManager.Reports[0]);
return pivotClientHelper.GetJsonData("toolbarOperation", DataManager,
"Load Report", reportName);
}
else
{
dynamic customData = serializer.Deserialize<dynamic>(repCol.ToString());
DataManager.Reports =
pivotClientHelper.DeserializedReports(customData[customData[customData.Le
ngth - 1]["cubeIndex"]]["Reports"]);
DataManager.SetCurrentReport(DataManager.Reports[customData[customData[cu
stomData.Length - 1]["cubeIndex"]]["ReportIndex"]]);
dictionary = pivotClientHelper.GetJsonData("toolbarOperation",
DataManager, "Load Report", reportName);
dictionary.Add("Collection", repCol);
}
}
else
{
if (analysisMode.ToLower() == "pivot" && operationalMode.ToLower() ==
"servermode")
dictionary = pivotClientHelper.GetJsonData("LoadReport",
ProductSales.GetSalesData(), Encoding.UTF8.GetString(reportByte));
else
dictionary.Add("report", Encoding.UTF8.GetString(reportByte));
break;
}
}
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Load Report from Local Storage

We can load the stored report collection of PivotClient from local storage, by setting the EnableLocalStorage property to true and using the LoadReport and FetchReport events in PivotClient.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1”).ClientSideEvents(events =>
events.LoadReport("reportSettings").FetchReport(”reportSettings”)).Enable
LocalStorage(true)
<script>
function reportSettings(args) {
var reportCollection = [];
if ((localStorage.pivotClientRPTCollection != "" &&
!ej.isNullOrUndefined(localStorage.pivotClientRPTCollection))) {
reportCollection = JSON.parse(localStorage.pivotClientRPTCollection);
}
if (args.fetchReportSetting)
args.fetchReportSetting.reportList = $.map(reportCollection, function
(item, index) { return item.reportName; }).join("__");
else if (args.loadReportSetting)
args.loadReportSetting.reportCollection = $.map(reportCollection,
function (item, index) { if (item.reportName ==
args.loadReportSetting.selectedReport) return item.reportCol; });
}
</script>
```

Layout Customization

Size

Allows you to render PivotClient in different sizes. You can set height and width under size property.

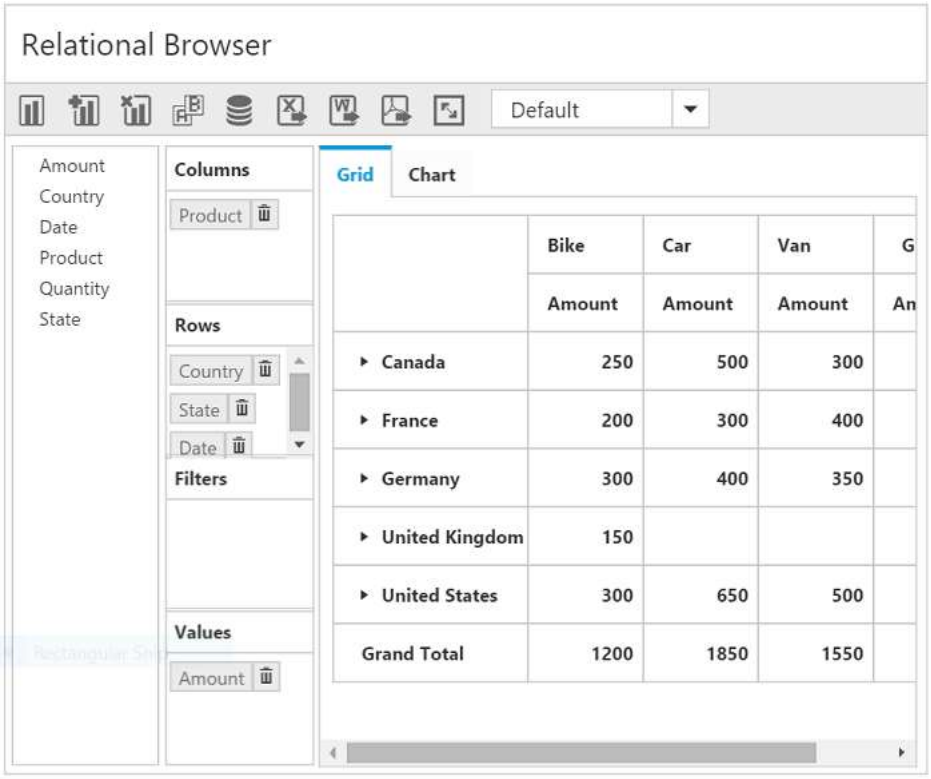
Set size in Pixels

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add(); })).size(siz => {
siz.Height("685px"); siz.Width("1000px"); })
```

PivotClient with decreased size from default size.

Copyright © 2001 - 2020 Syncfusion Inc.



Set size in percentage

You can set the PivotClient size in percentage also.

Note: Size of the parent container should be set in Pixels.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add(); })).size(siz => {
siz.Height("80%"); siz.Width("50%"); })
```

Control Placement

Tab View

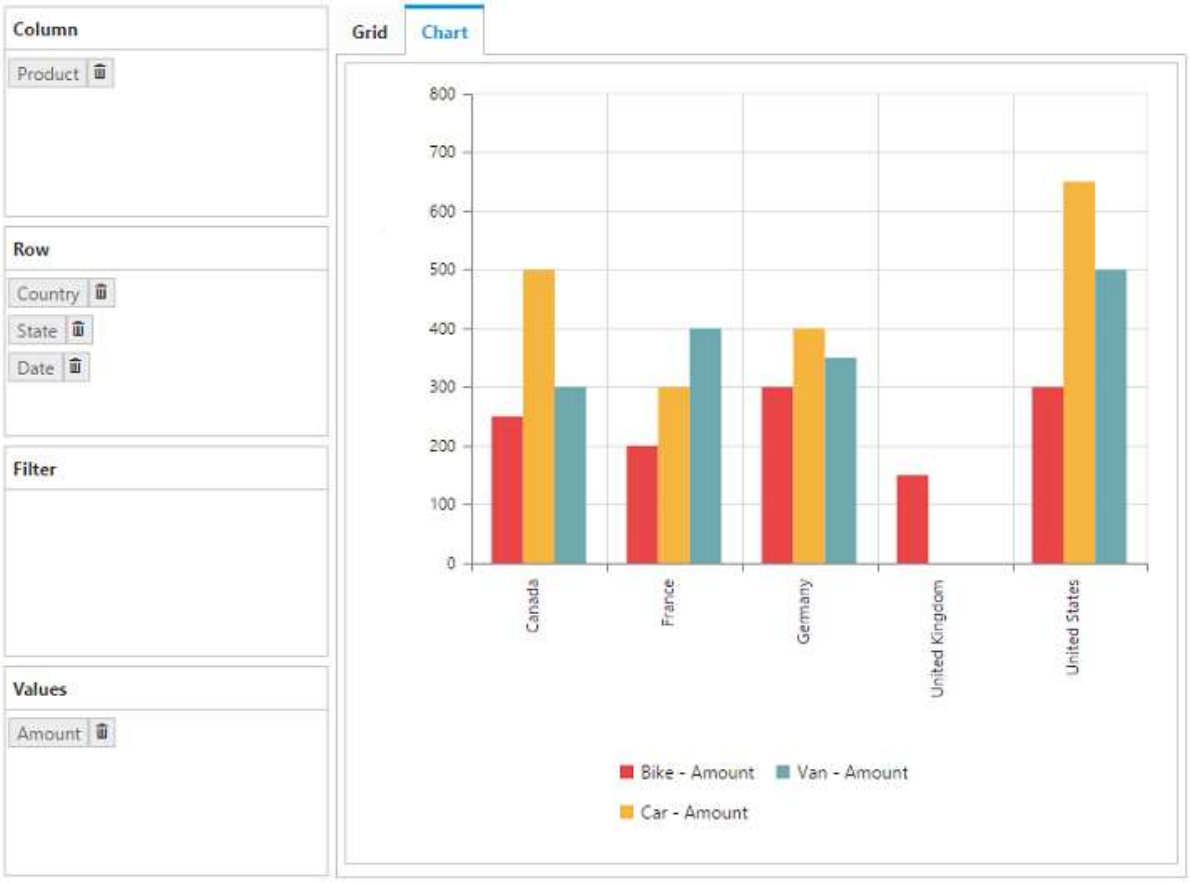
In Tab view representation, both Grid and Chart will be displayed in separate tabs. This could be set by using the ControlPlacement property under the DisplaySettings option. By default, **Tab** value is set.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.ControlPlacement(
PivotClientControlPlacement.Tab))
```



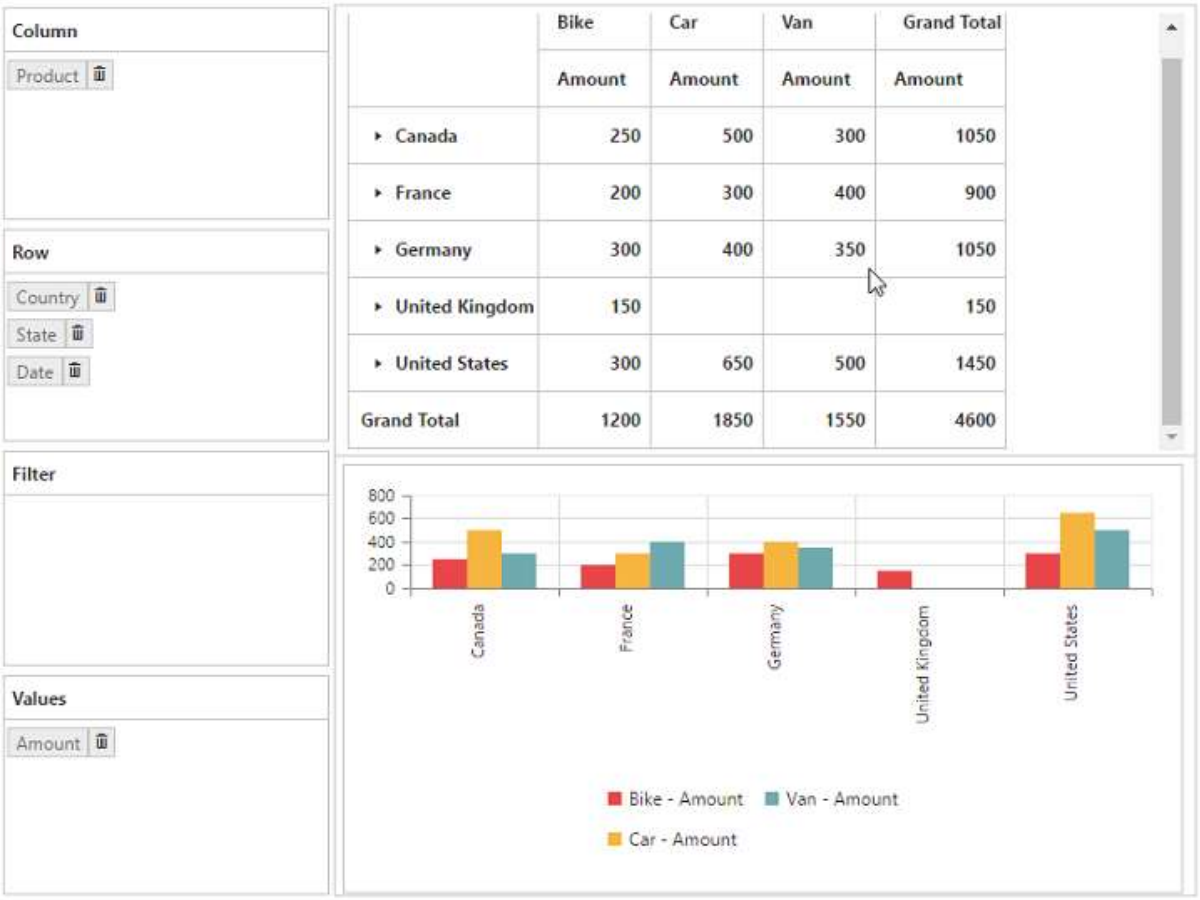
Tile View

In Tile View representation, both Grid and Chart will be displayed one above the other, in the same layout. Tile view can be set by using the ControlPlacement property under the DisplaySettings option.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.ControlPlacement(Piv
otClientControlPlacement.Tile))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Default View

Grid View

To display Grid control by default, set DefaultView property under DisplaySettings option to **Grid**, which is the default value of the property.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.DefaultView(
PivotClientDefaultView.Grid))
```

Copyright © 2001 - 2020 Syncfusion Inc.

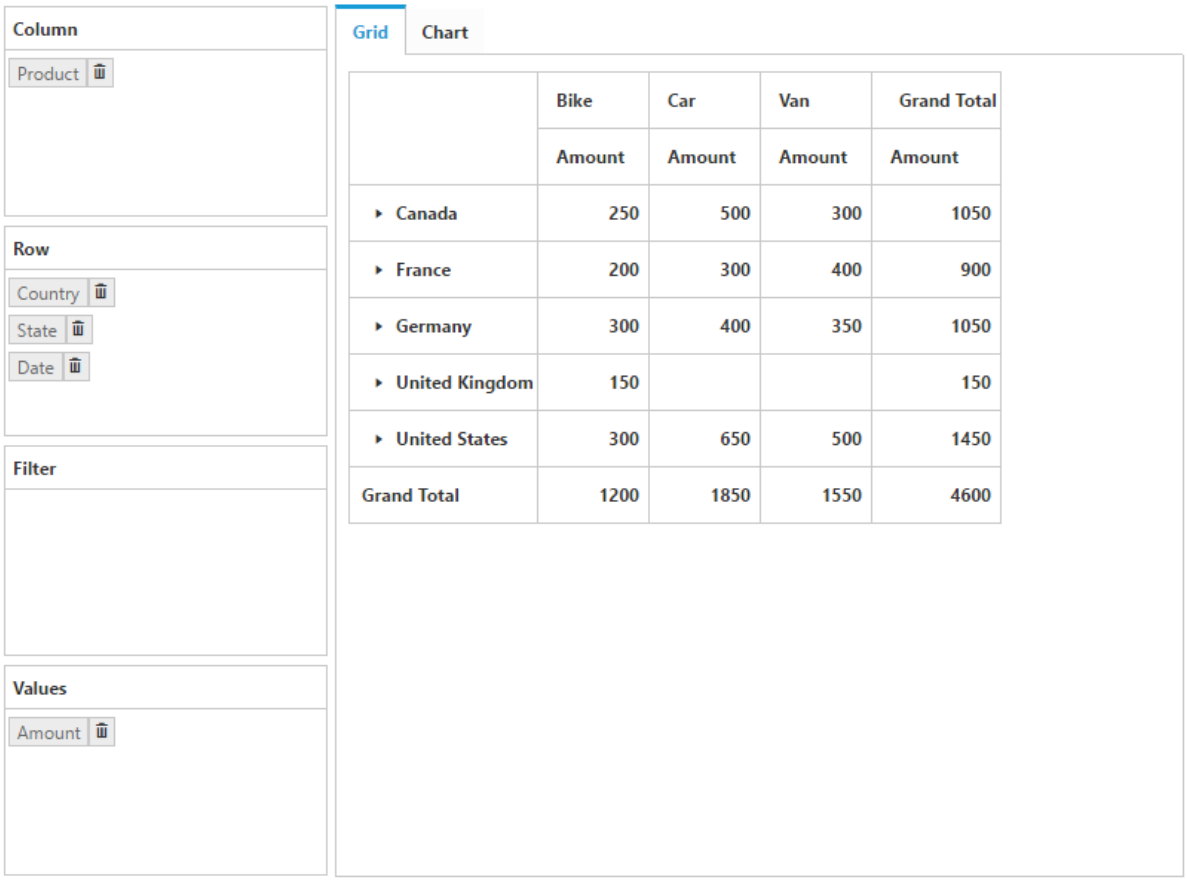


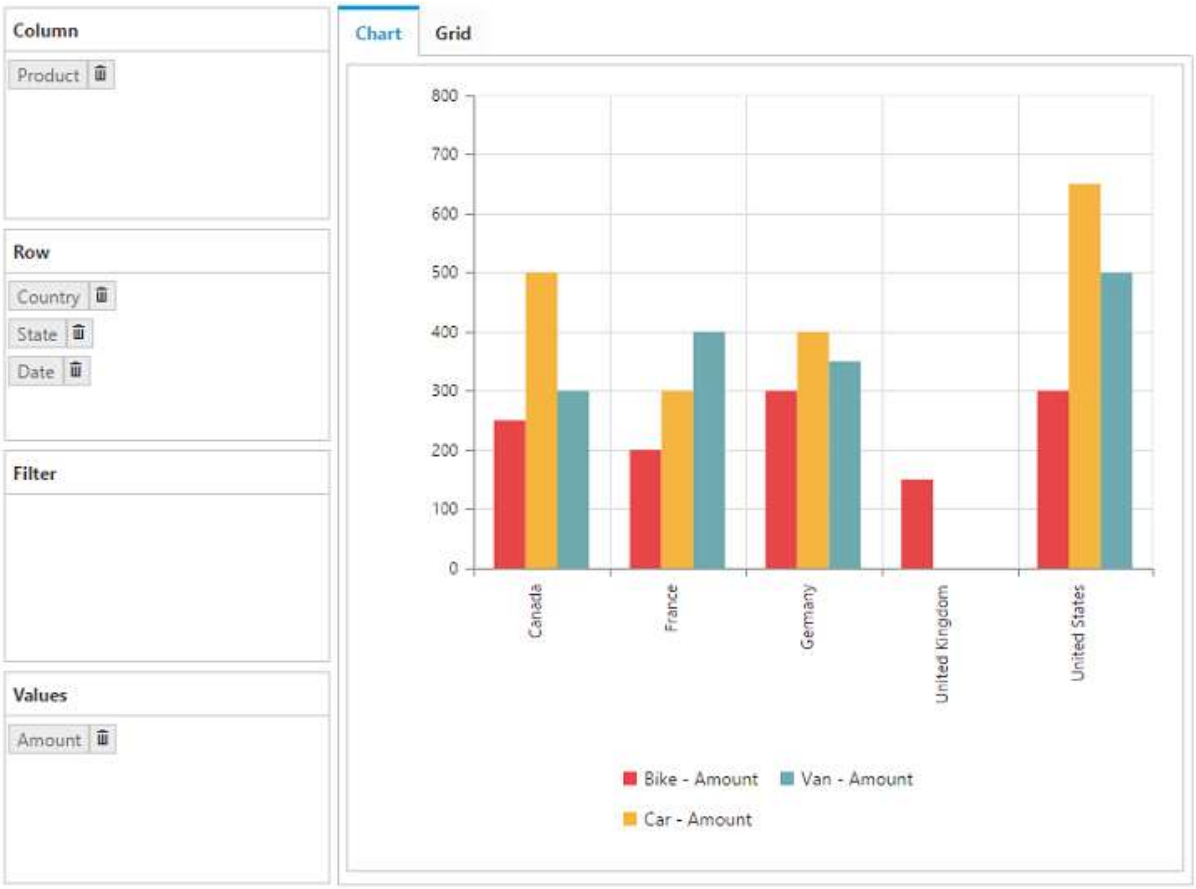
Chart View

To display Chart control by default, set the DefaultView property to **Chart**.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.DefaultView(
PivotClientDefaultView.Chart))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Display Mode

Grid Only

By the Mode property under DisplaySettings option to **GridOnly**, PivotGrid component alone will get rendered and PivotChart will not be rendered.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.Mode(
PivotClientDisplayMode.GridOnly))
```

Copyright © 2001 - 2020 Syncfusion Inc.

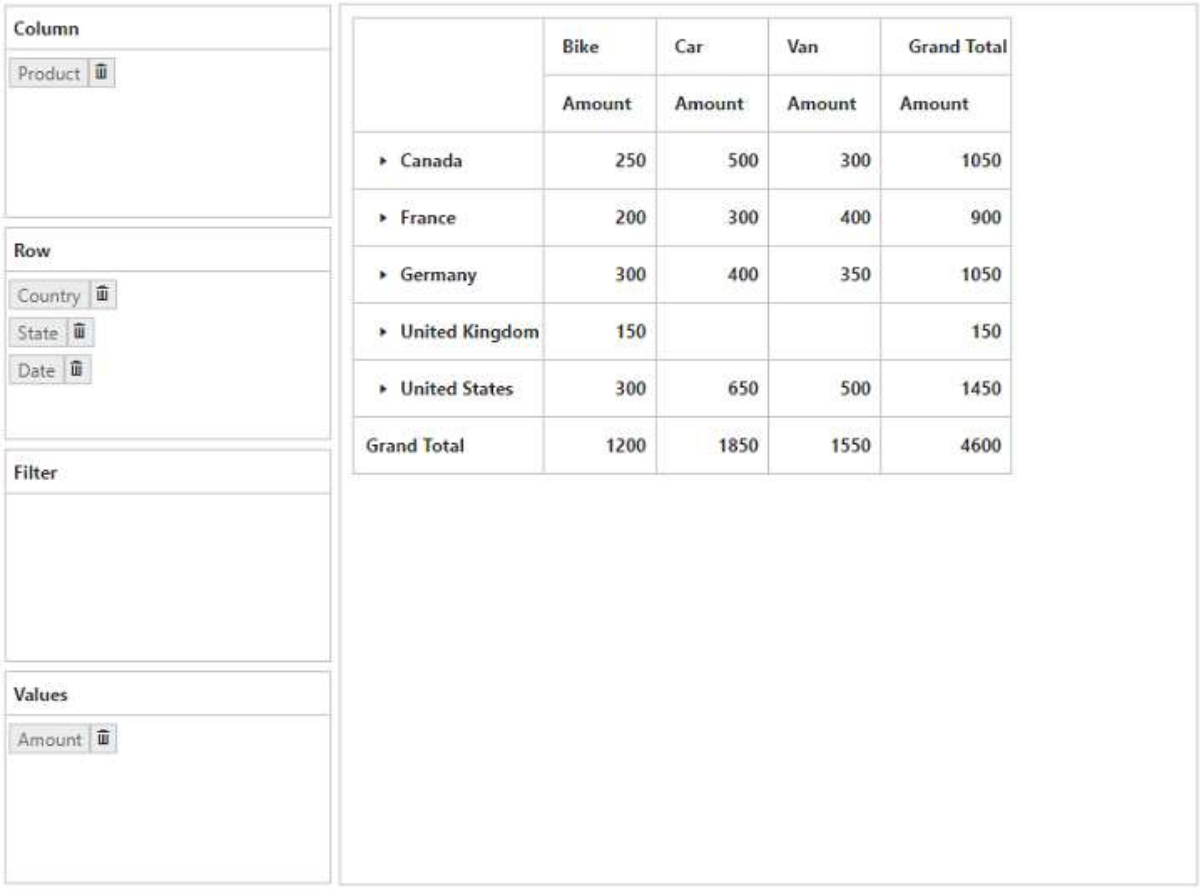


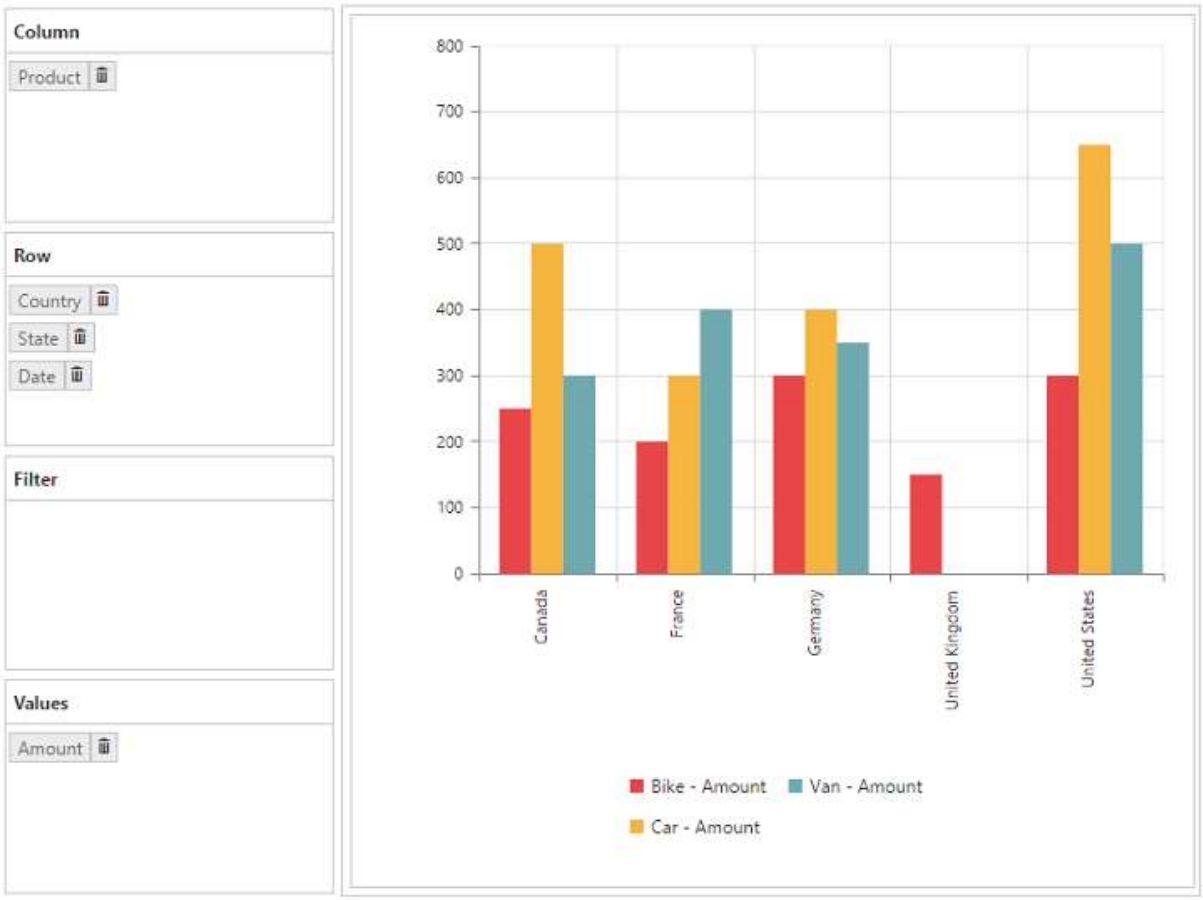
Chart Only

By the Mode property under DisplaySettings option to **ChartOnly**, PivotChart component alone will get rendered and PivotGrid will not be rendered.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.Mode(
PivotClientDisplayMode.ChartOnly))
```

Copyright © 2001 - 2020 Syncfusion Inc.



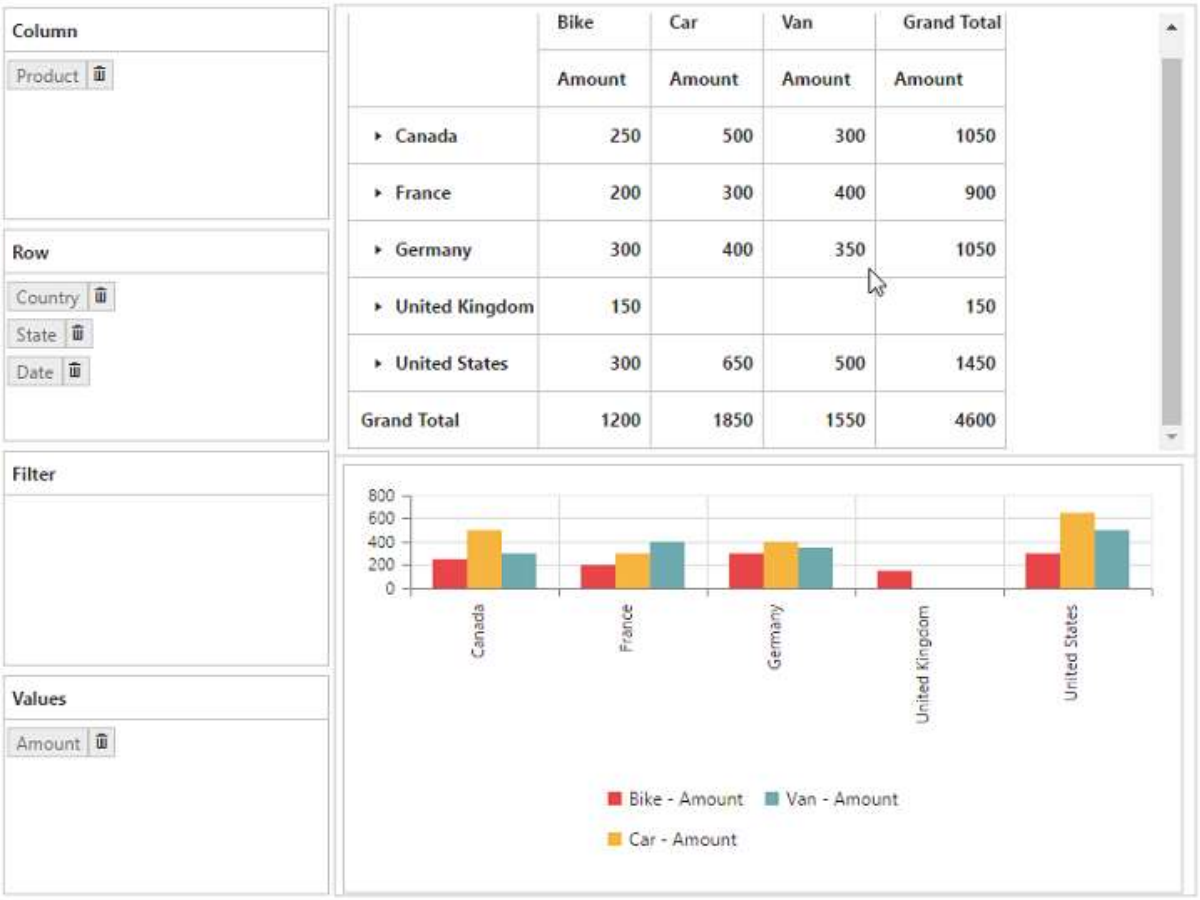
Both Chart and Grid

By the Mode property under DisplaySettings option to **ChartAndGrid**, data is displayed in both Grid and Chart. This is the default value of the property.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.Mode(
PivotClientDisplayMode.ChartAndGrid))
```

Copyright © 2001 - 2020 Syncfusion Inc.



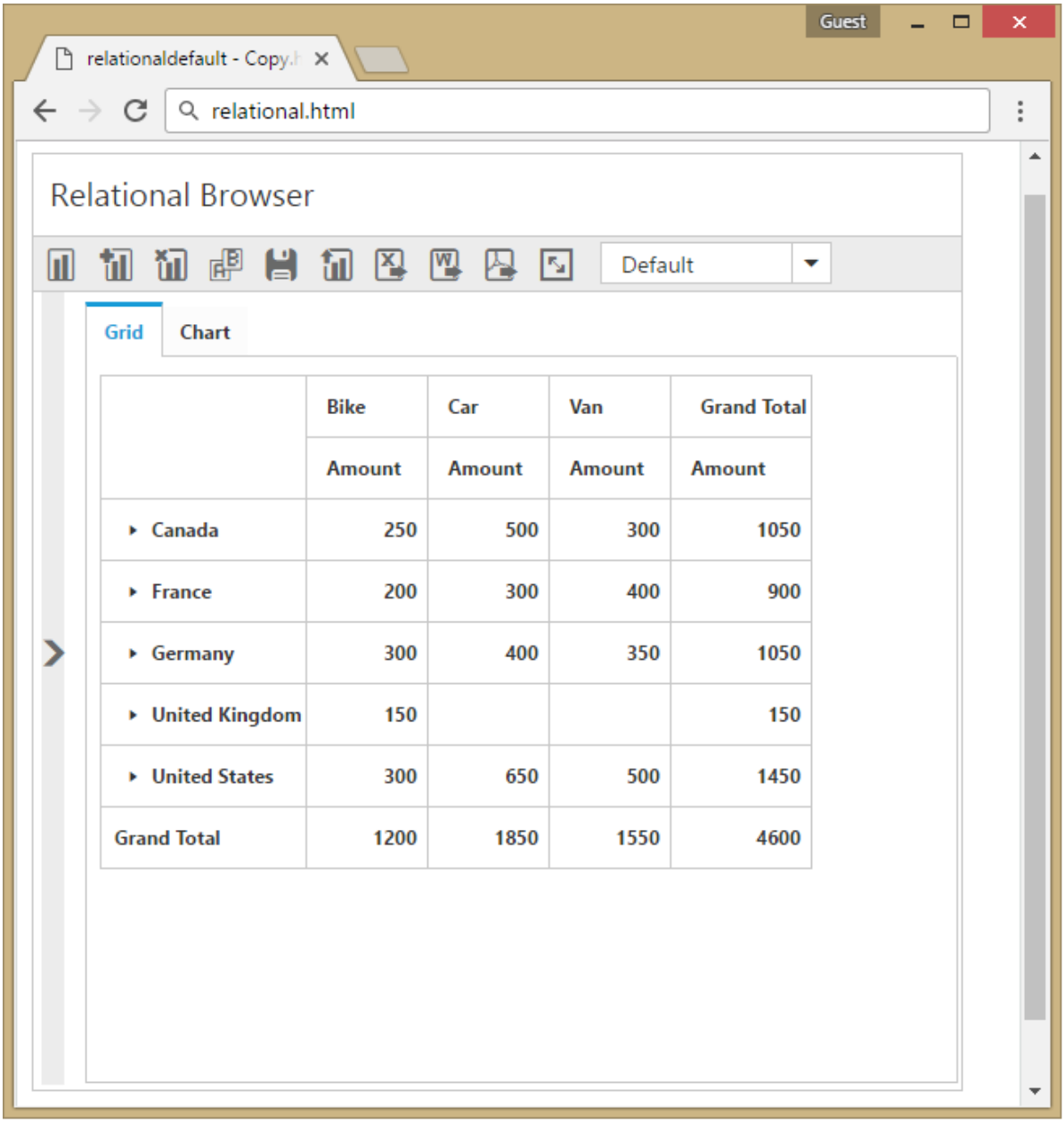
Toggle Panel

Toggle panel option lets the user to toggle the visibility of Axis Element Builder and Cube Dimension Browser panels in PivotClient with a use of a button. The button could be added to the control by enabling the EnableTogglePanel property under DisplaySettings option. This property is disabled by default.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.EnableTogglePanel(tr
ue))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Collapse Toggle Panel By Default

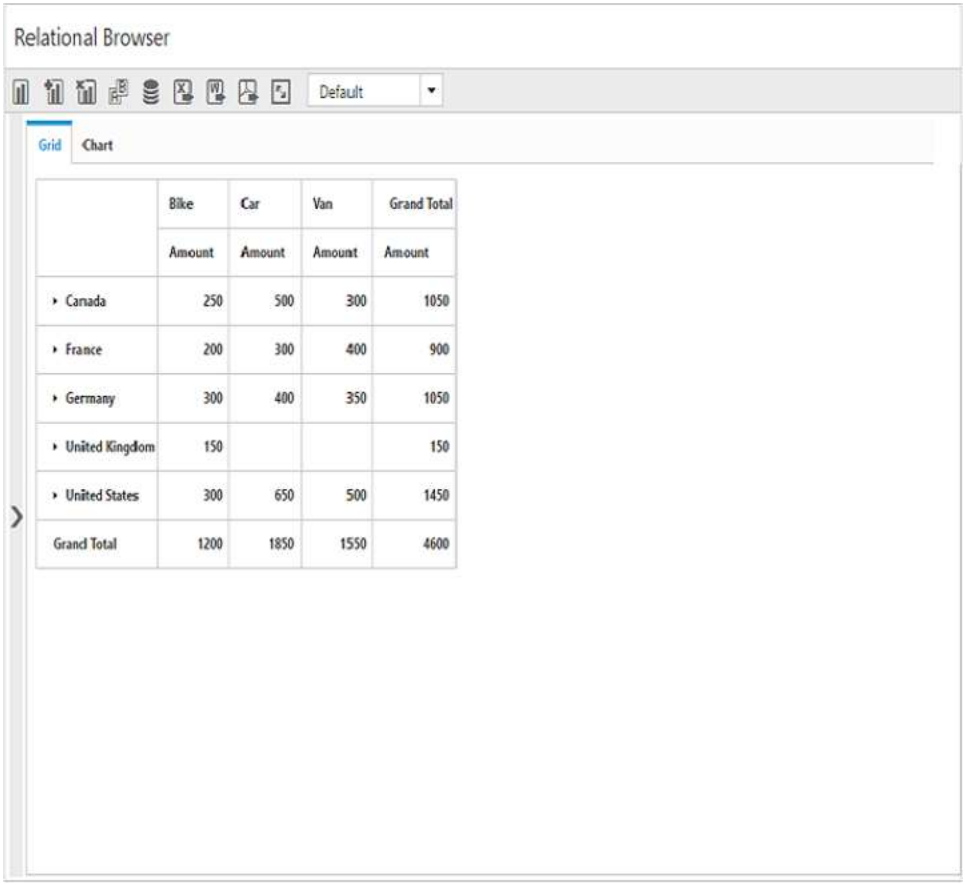
Allows the user to hide “Cube Browser” and “Axis Element Builder” panels while initiating the widget. You can enable this option in PivotClient by setting the `CollapseCubeBrowserByDefault` property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
values.FieldName("Amount").Format("currency").Add();
})).CollapseCubeBrowserByDefault(true)
```



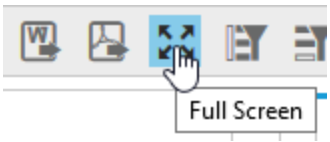
Maximized/Full Screen View

Full screen view helps to visualize the PivotGrid and PivotChart controls inside PivotClient precisely according to the browser window size. By selecting full screen icon in the toolbar, the control which is in the view gets maximized. Drilldown action can also be performed in both PivotGrid and PivotChart in the maximized view. This option is enabled by setting the EnableFullScreen property under DisplaySettings option to true. The value is false by default.

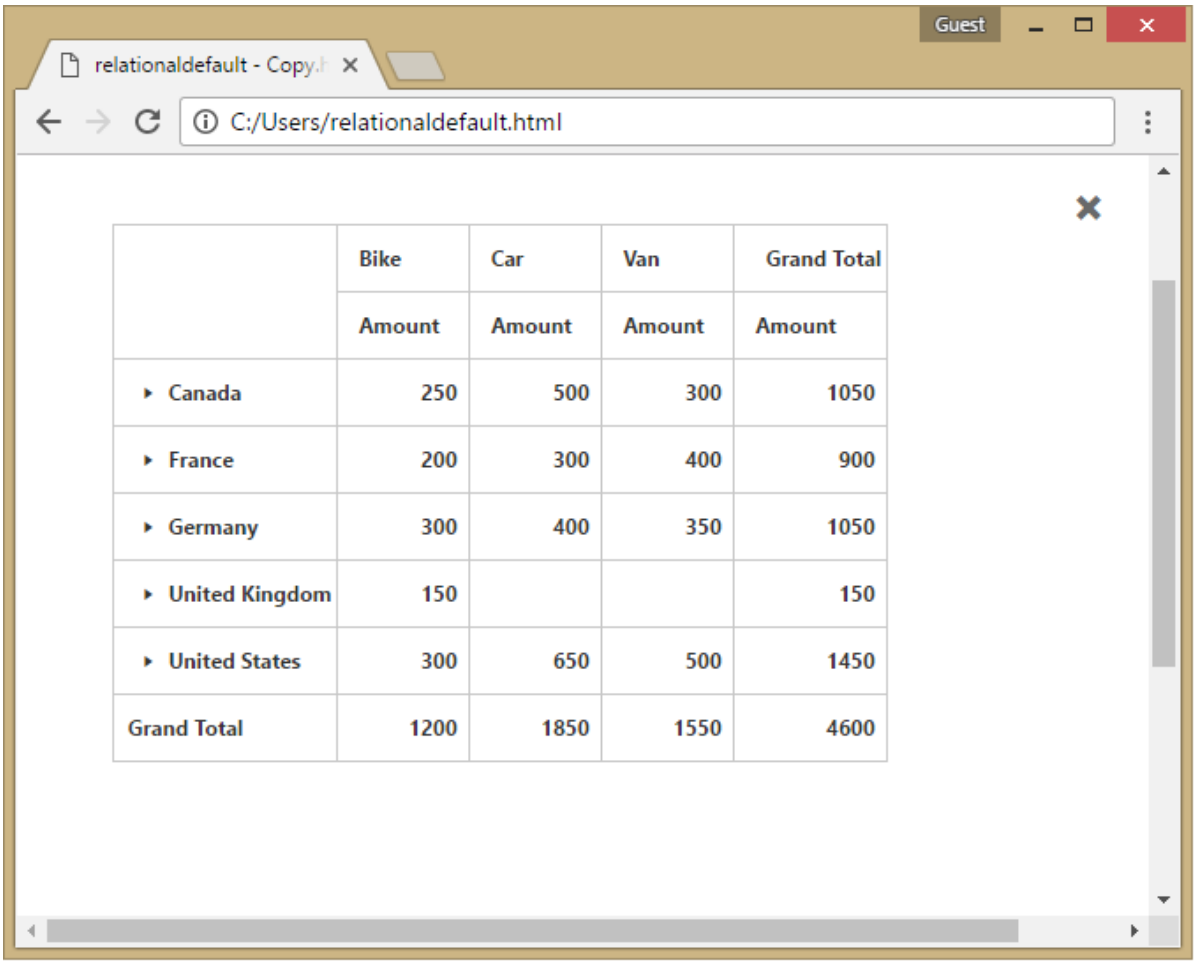
ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).DisplaySettings(displaySettings=>displaySettings.EnableFullScreen(tru
e))
```

Copyright © 2001 - 2020 Syncfusion Inc.



The following screenshot shows the maximized view of PivotGrid.



	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
▸ Canada	250	500	300	1050
▸ France	200	300	400	900
▸ Germany	300	400	350	1050
▸ United Kingdom	150			150
▸ United States	300	650	500	1450
Grand Total	1200	1850	1550	4600

Chart Types

While loading the PivotClient initially, the PivotChart widget can be rendered in any one of the available chart types using the ChartType property.

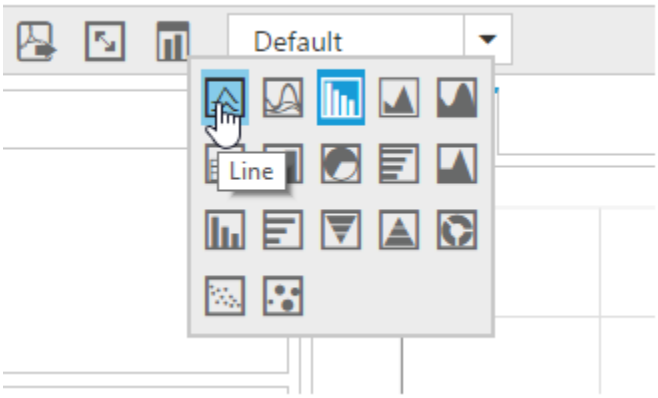
ASPX-CS

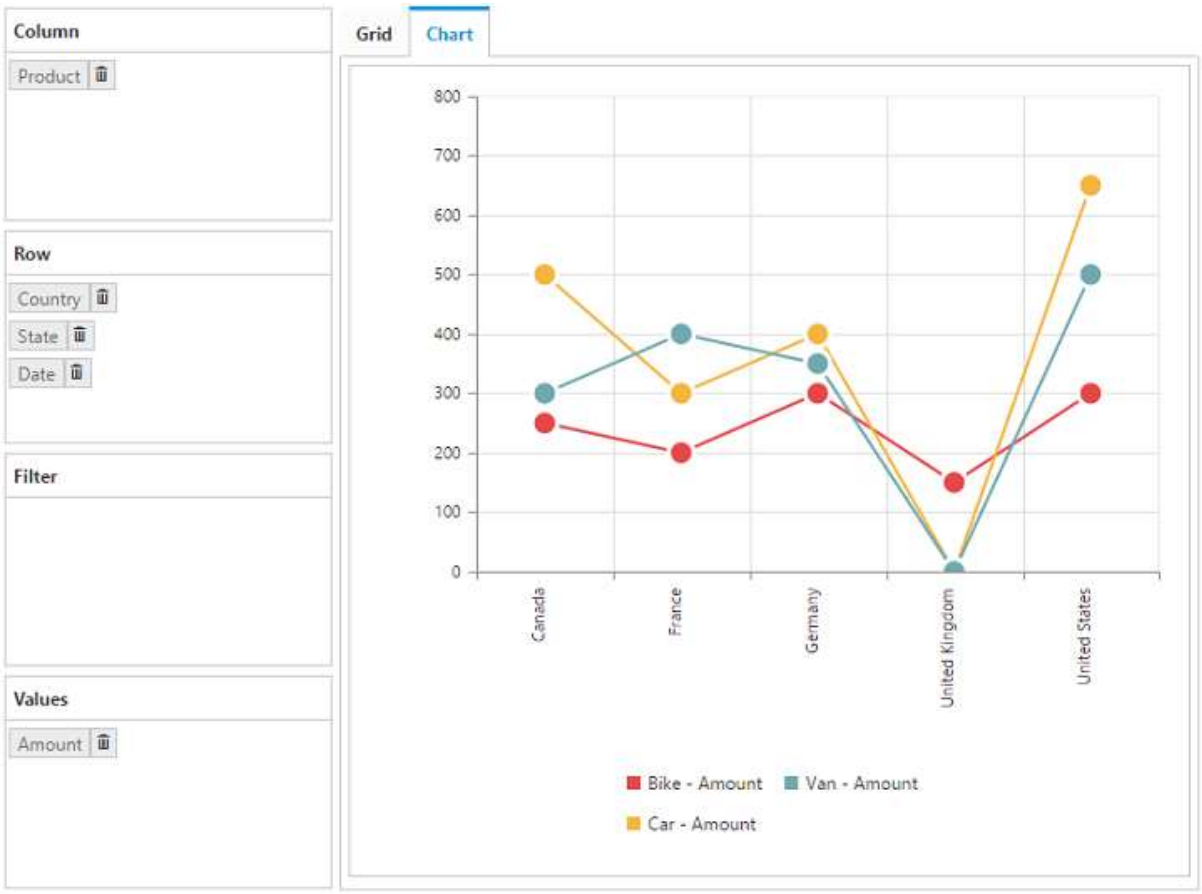
```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).ChartType(PivotChartType.Column)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The ChartType property takes Column Chart by default. The types available are Column, Stacking Column, Bar, Stacking Bar, Line, Spline, Step Line, Area, Spline Area, Step Area, Stacking Area, Pie, Funnel and Pyramid.

The Chart Type can also be changed dynamically through the toolbar icon.





PivotTreeMap

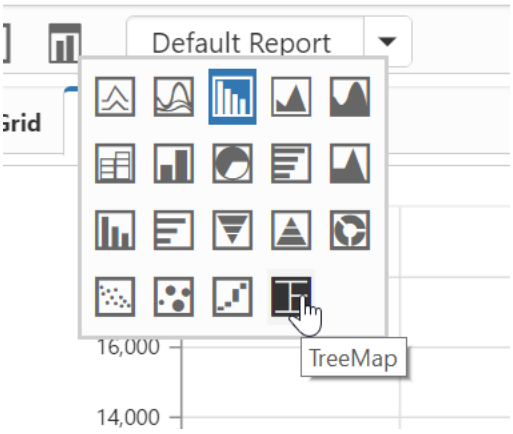
Information: This feature is applicable only for OLAP data source bound from server-side.

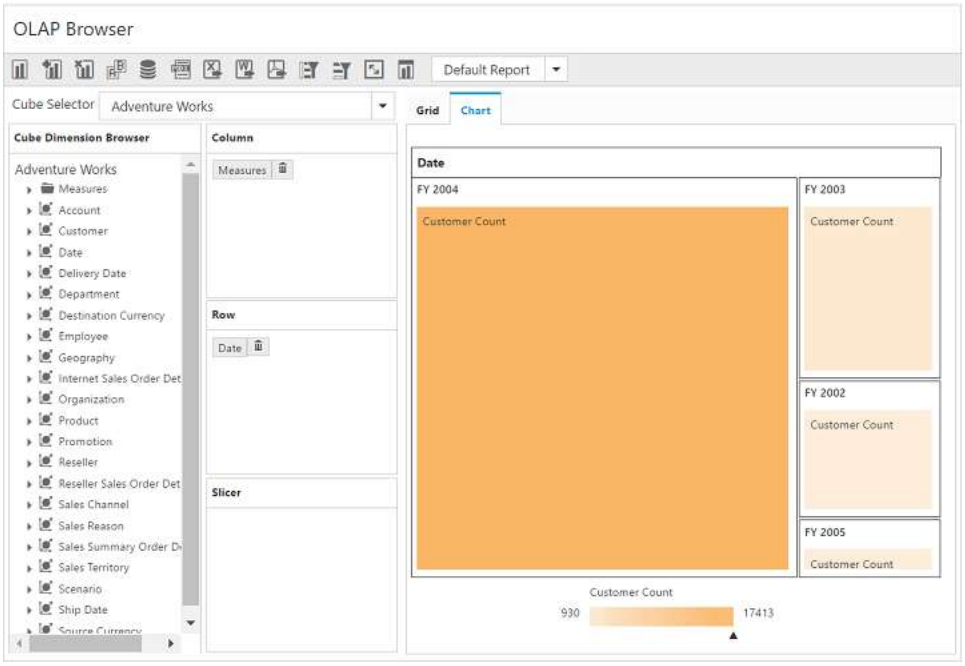
You can include the PivotTreeMap component as one of the chart types by setting EnablePivotTreeMap property to true.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).EnablePivotTreeMap(true)
```





Report Toolbar

You can customize the display of toolbar by enabling/disabling the visibility of each of the icons. This can be achieved by setting the properties under ToolbarIconSettings option to false. The values are true by default.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1”).ToolbarIconSettings(toolba
ricon => {
toolbaricon.EnableAddReport(false).EnableNewReport(false).EnableRemoveRep
ort(false); })
```



The following screenshot shows after disabling the toolbar icons.



Advanced Filtering and Sorting

It allows to filter and sort the field members in PivotClient.

Client Mode

In client mode, you can enable Advanced Filtering and Sorting option in PivotClient by setting the EnableAdvancedFilter property under DataSource to true.

HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.EnableAdvancedFilter(true))
```

Server Mode

In server mode, you can enable the Advanced Filtering and Sorting option in PivotClient by setting the EnableAdvancedFilter property to true.

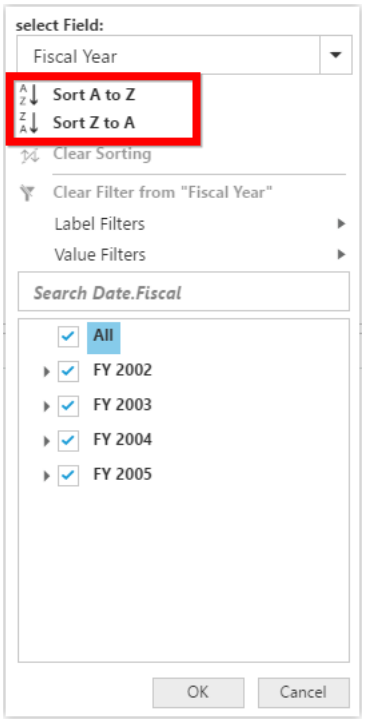
HTML

```
@Html.EJ().Pivot().PivotClient("PivotClient1").EnableAdvancedFilter(true)
```

Sorting

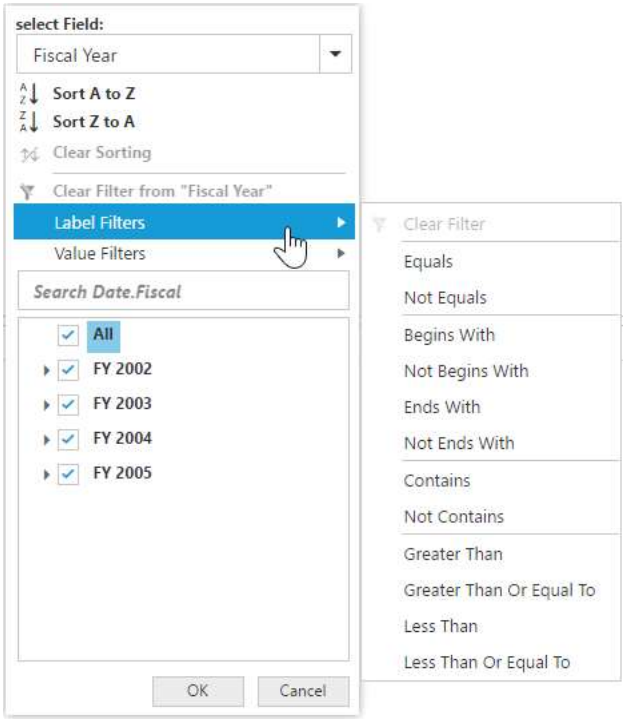
Sorting provides an option to sort the members of a field either in ascending or descending order.

Copyright © 2001 - 2020 Syncfusion Inc.



Label Filtering

Label filtering provides an option to filter the members of a field purely based on their caption.

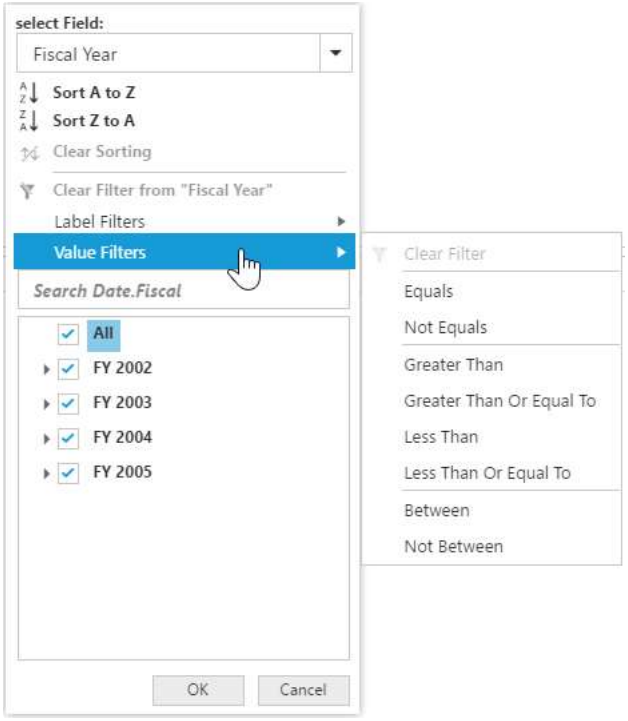


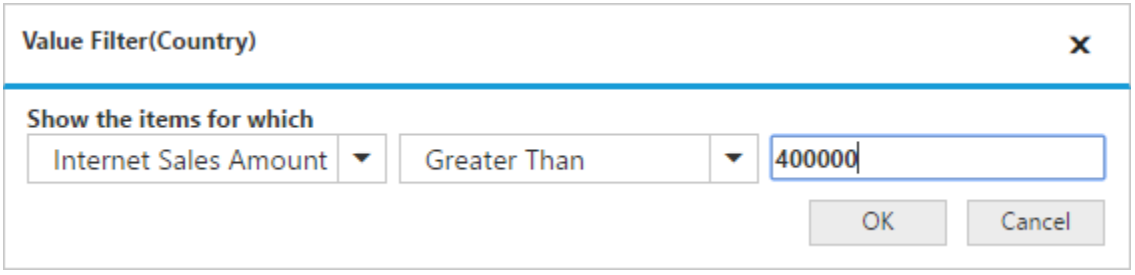
Copyright © 2001 - 2020 Syncfusion Inc.



Value Filtering

Value filtering provides an option to filter members based on the total values of the appropriate measure between the members of the level.

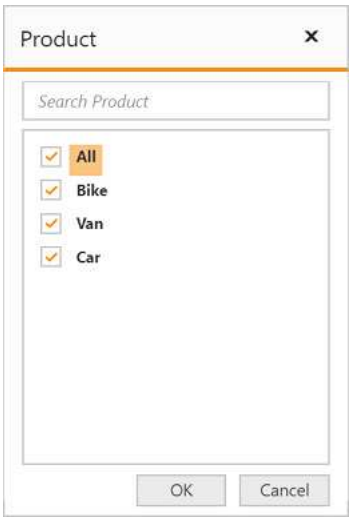




Member Editor

Member editor is a dialog that displays the members of the current field in a tree view structure, which can be opened by clicking the pivot button available in the axis elements. It helps to search, filter, and sort the field members available in the pivot client control.

Copyright © 2001 - 2020 Syncfusion Inc.



Member editor - Paging

Member editor paging helps to improve the rendering performance of the dialog by dividing large amount of data into sections and displaying them.

You can enable member editor paging and set member editor page size in PivotClient control by setting the EnableMemberEditorPaging and MemberEditorPageSize properties.

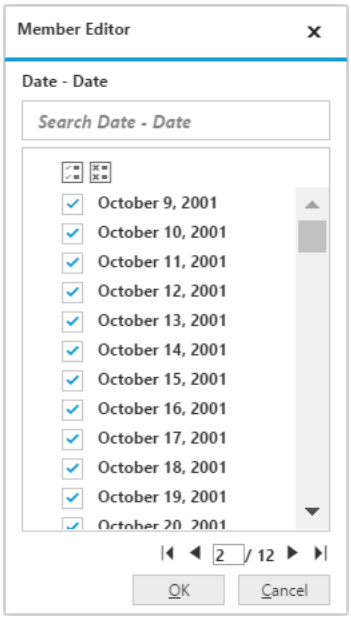
ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").EnableMemberEditorPaging(t
rue).MemberEditorPageSize(100)
```

Following are the navigation option available in Member Editor Pager.

- Move First - Navigates to the first page.
- Move Previous - Navigates to the previous page from the current page.
- Move Next - Navigates to the next page from the current page.
- Move Last - Navigates to the last page.
- Numeric Box - Navigates to the desired page by entering an appropriate page number in numeric value.

Copyright © 2001 - 2020 Syncfusion Inc.



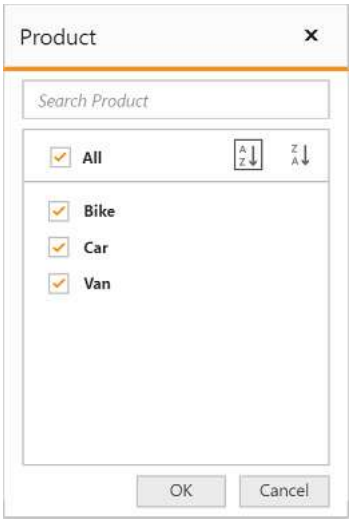
Member editor - Sorting

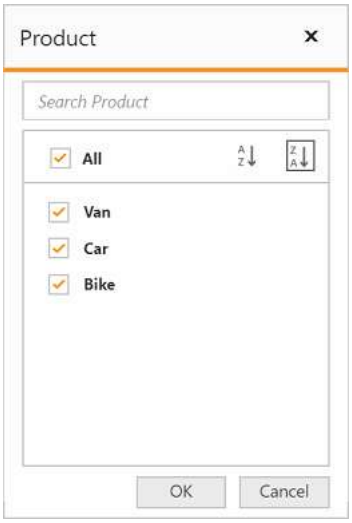
The sorting support in member editor helps you to sort the field members either in ascending or descending order.

You can enable the member editor sorting in the pivot grid control by setting the [EnableMemberEditorSorting] property.

ASPX-CS

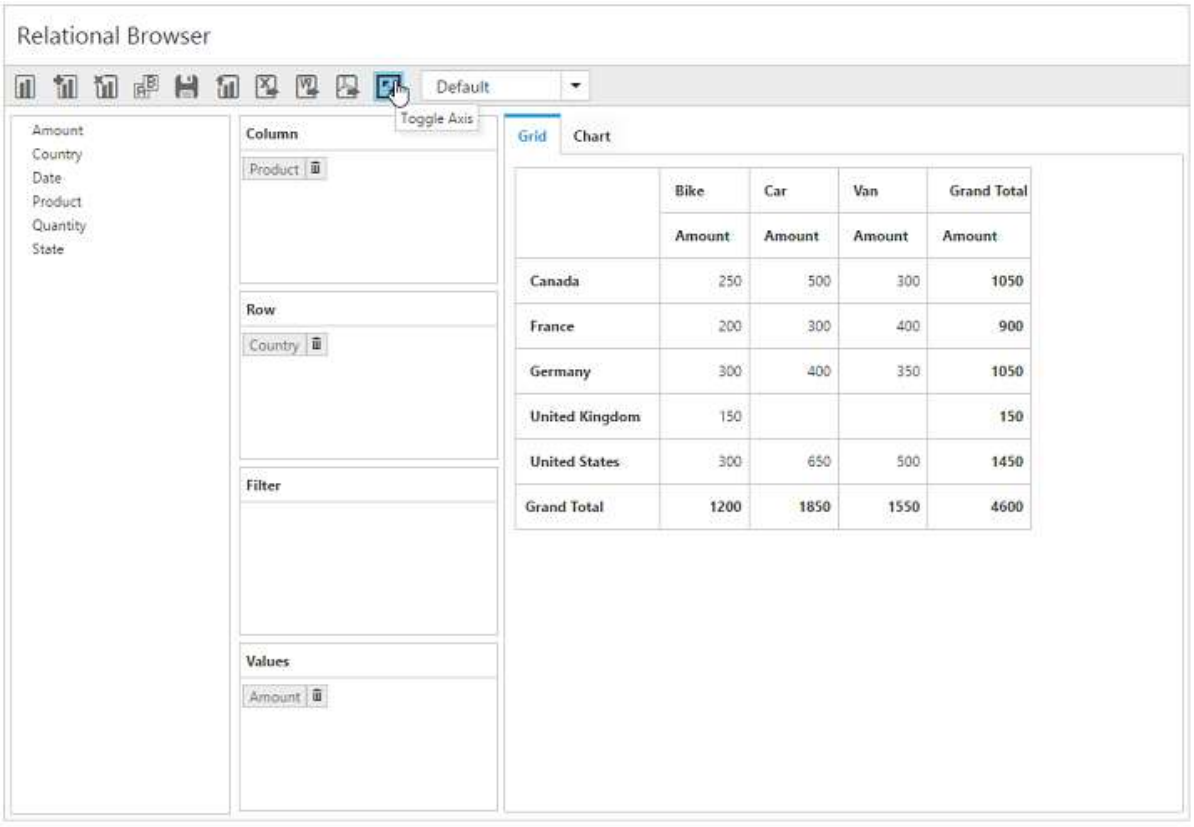
```
@Html.EJ().Pivot().PivotClient("PivotClient1").EnableMemberEditorSorting(
true)
```



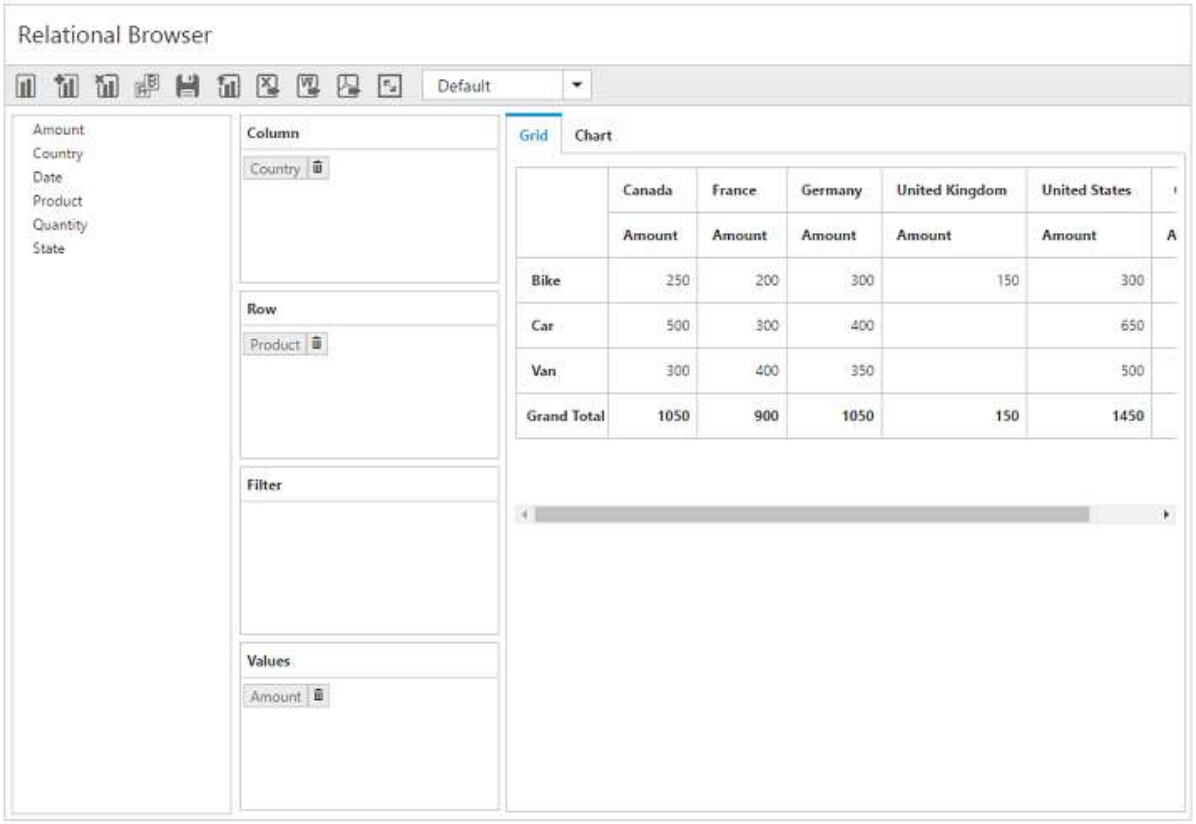


Toggle Axis

Toggle Axis support swaps all the axis elements between column and row axes, thereby refreshing PivotChart and PivotGrid view. This can be achieved with the use of a toolbar icon.



Copyright © 2001 - 2020 Syncfusion Inc.



Localization and Globalization

Localization in PivotClient control

We can localize the PivotClient control texts with a collection of localized strings using **“ej.PivotClient.Locale”** for different cultures.

Note: By default, the PivotClient control is localized in **“en-US”.**

Following code example illustrates on how to localize PivotClient based on **“French”** culture.

ASPX-CS

```
<script type="text/javascript">
ej.PivotClient.Locale["fr-FR"] = {
DeferUpdate: "Différer Mise à jour",
MDXQuery: "de requêtes MDX",
Column: "Colonne",
Row: "Rangée",
Slicer: "Tranche",
CubeSelector: "Sélecteur de Cube",
ReportName: "Nom du rapport",
NewReport: "Nouveau rapport",
CubeDimensionBrowser: "Cube navigateur dimnesion",
AddReport: "Ajouter un rapport",
RemoveReport: "Retirer rapport",
//...
}
ej.PivotGrid.Locale["fr-FR"] = {
ToolTipRow: "Rangée",
ToolTipColumn: "Colonne",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ToolTipValue: "Valeur"
}
ej.PivotChart.Locale["fr-FR"] = {
Measure: "Mesure",
Row: "Rangée",
Column: "Colonne",
Value: "Valeur",
Expand: "Développer",
Collapse: "Effondrement",
Exit: "Quitter"
}
ej.PivotSchemaDesigner.Locale["fr-FR"] = {
AddToFilter: "Ajouter au filtre",
AddToRow: "Ajouter à la rangée",
AddToColumn: "Ajouter à la colonne",
AddToValues: "Ajouter à la valeur",
PivotTableFieldList: "Liste des champs PivotTable",
//...
}
</script>
@Html.EJ().Pivot().PivotClient("PivotClient1").Url(Url.Content("/OlapClie
nt")).Locale("fr-FR")
```

Following table localizes the in-built keywords to **French** culture for PivotClient.

Keywords	Values
DeferUpdate	DiffÃ©rer Mise Ã jour
MDXQuery	de requÃªtes MDX
Column	Colonne
Row	RangÃ©e
Slicer	Tranche
CubeSelector	SÃ©lecteur de Cube
ReportName	Nom du rapport
NewReport	Nouveau rapport
CubeDimensionBrowser	Cube navigateur dimnesion
AddReport	Ajouter un rapport
RemoveReport	Retirer rapport
CannotRemoveSingleReport	Vous ne pouvez pas supprimer Rapport unique
AreYouSureToDeleteTheReport	Etes-vous sÃ»r de vouloir supprimer le rapport
RenameReport	Renommer rapport
SaveReport	Enregistrer le rapport

Copyright © 2001 - 2020 Syncfusion Inc.

LoadReport	Rapport de charge
ToggleAxis	Basculer Axis
ExportToExcel	Exporter vers Excel
ExportToWord	Exporter vers Word
ExportToPdf	Exporter vers PDF
FullScreen	Plein Ã©cran
Grid	Grille
Chart	Graphiq
OK	Bien
Cancel	Annuler
MeasureEditor	Mesurer Ã©diteur
MemberEditor	Sous la direction de membres
Measures	Mesures
SortOrFilterColumn	Tri/filtrage (colonne)
SortOrFilterRow	Tri/filtrage (ligne)
SortingAndFiltering	Trier et filtrer
Sorting	Tri
Measure	Mesurer
Order	Ordre
Filtering	Filtrage
Condition	Condition
Value	Valeur
PreserveHierarchy	PrÃ©server hiÃ©rarchie
Ascending	Croissant
Descending	Descendant
Enable	Permettre
Disable	DÃ©sactiver
And	et
Line	ligne

Copyright © 2001 - 2020 Syncfusion Inc.

Spline	spline
Column	colonne
Area	zone
SplineArea	spline zone
StepLine	Ã©tape ligne
StepArea	Ã©tape zone
Pie	tarte
Bar	bar
StackingArea	Stacking zone
StackingColumn	Colonne d'empilage
StackingBar	Stacking bar
Pyramid	pyramide
Funnel	entonnoir
ChartTypes	Types de graphiques
Sort	Trier
SelectField	sÃ©lectionnez Champ
LabelFilterLabel	Afficher les Ã©lÃ©ments pour lesquels l'Ã©tiquette
ValueFilterLabel	Voir les articles pour lesquels
LabelFilters	Filtres d'Ã©tiquetage
BeginsWith	Commence par
NotBeginsWith	Non Commence par
EndsWith	Se termine par
NotEndsWith	Non Se termine par
Contains	Contient
NotContains	Ne contient pas
ValueFilters	Filtres de valeur
ClearFilter	Effacer le filtre
Equals	Ã‰quivaut Ã
Top10	"Top Count"

Copyright © 2001 - 2020 Syncfusion Inc.

EqualTo	Ã‰gal Ã
NotEquals	pas equals
GreaterThan	Plus grand que
GreaterThanOrEqualTo	Plus grand ou Ã©gal Ã
LessThan	Moins que
LessThanOrEqualTo	InfÃ©rieur ou Ã©gal Ã
Between	Entre
NotBetween	Entre pas
DoesNotEndsWith	Ne se termine par
DoesNotContains	Ne contient
DoesNotEquals	N'est pas Ã©gaux
IsGreaterThan	Est supÃ©rieure Ã
IsGreaterThanOrEqualTo	Est supÃ©rieure ou Ã©gale Ã
IsLessThan	Est infÃ©rieure Ã
IsLessThanOrEqualTo	Est infÃ©rieure ou Ã©gale Ã
IsLessThanOrEqualTo	Est infÃ©rieure ou Ã©gale Ã
Search	Recherchez
SetReportNameAlertMsg	Veuillez dÃ©finir le nom du rapport
SetRecordNameAlertMsg	Veuillez dÃ©finir nom d'enregistrement
Remove	DÃ©poser
Rename	Renommer
DBReport	La manipulation en DB Rapport
SelectReport	SÃ©lectionnez Rapport
SelectRecord	SÃ©lectionnez Enreg
SaveAs	Enregistrer sous
Save	Sauvegarder
Save	Sauvegarder
Load	Charger
RenameRecord	L'enregistrement de nom

Copyright © 2001 - 2020 Syncfusion Inc.

RemoveRecord	Effacer l'enregistrement
RecordName	Nom d'enregistrement
SelectRecordAlertMsg	Veuillez sÃ©lectionner un enregistrement valide
FilterInvalidAlertMsg	OpÃ©ration non valide
FilterEndValueAlertMsg	Veuillez dÃ©finir une valeur de fin
FilterStartValueAlertMsg	Veuillez dÃ©finir une valeur de dÃ©part
FilterConditionAlertMsg	Veuillez dÃ©finir une condition valide
FilterMeasureSelectionAlertMsg	Veuillez sÃ©lectionner une mesure valide
FilterSortElementAlertMsg	Ã‰lÃ©ment non trouvÃ© dans l'axe de la colonne. Veuillez ajouter un Ã©lÃ©ment dans l'axe de la colonne pour trier/filtrage
FilterSortrowMeasureAlertMsg	Veuillez ajouter Ã la mesure de l'axe de rang
FilterSortcolMeasureAlertMsg	Veuillez ajouter Ã la mesure de l'axe de la colonne
FilterSortColumnAlertMsg	On ne trouve pas dans la dimension de l'axe de la colonne. Veuillez ajouter la dimension de l'axe de l'Ã©lÃ©ment dans la colonne de tri et de filtrage
FilterSortRowAlertMsg	On ne trouve pas dans la dimension de l'axe de rang. Veuillez ajouter la dimension Ã©lÃ©ment de la ligne de tri axe/filtrage
MDXAlertMsg	Veuillez ajouter une mesure, dimension ou de hiÃ©rarchie dans un axe appropriÃ© pour afficher la requÃªte MDX
Alert	Alert
TreeMap	Carte d'arbres
Bubble	Bubble
Scatter	dispersion
Doughnut	Donut
Close	Fermer
ReportList	Liste des rapports
AddToSlicer	Ajouter Ã Slicer
AddToRow	Ajouter Ã la rangÃ©e
AddToColumn	Ajouter Ã la colonne
AddToColumn	Ajouter Ã la colonne

Copyright © 2001 - 2020 Syncfusion Inc.

Localization and Globalization of Cube Info (Client Mode)

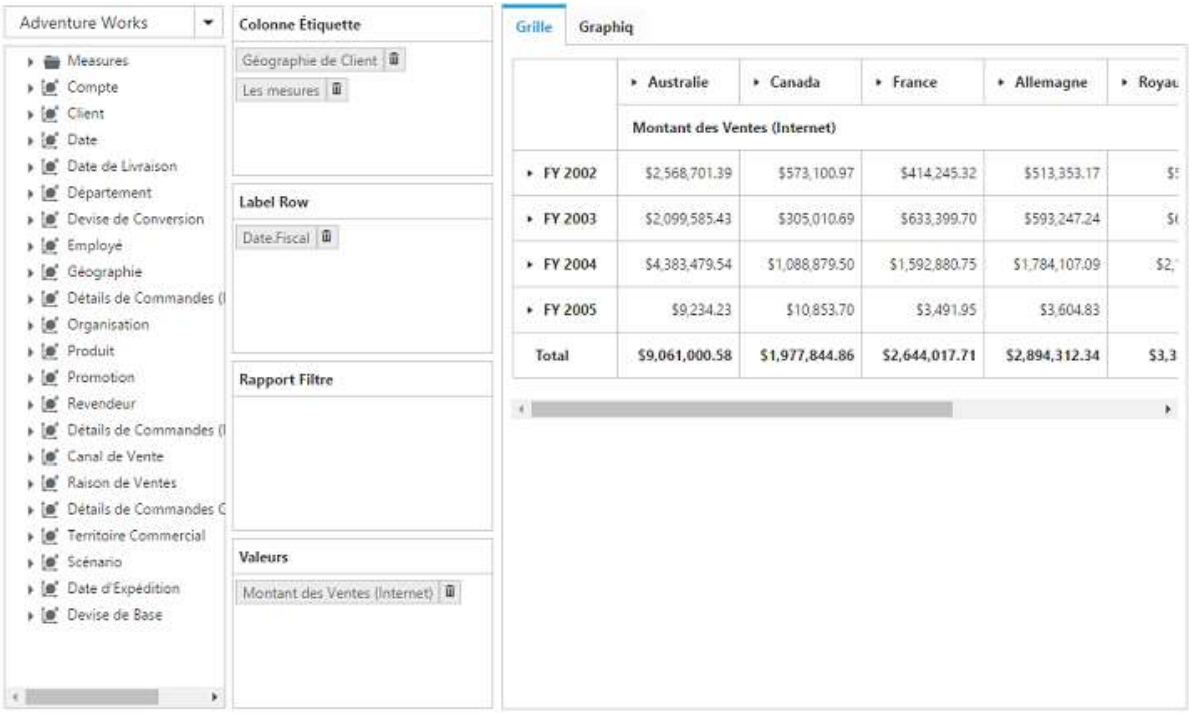
Content displayed within the PivotClient control are obtained from the OLAP cube. Following are the steps to get the localized and globalized cube content.

- To get localized data from OLAP cube, we need to set **"Locale Identifier"** in the connection string to a specific culture in the **"Data"** property present inside **"DataSource"**.
- To bind the globalized content in PivotClient control, we need to set **"Locale"** property to a specific culture and the specific culture file is referred in the sample.

Note: Culture files are present under **"[installed drive]:\Users\[user name]\AppData\Local\Syncfusion\EssentialStudio\X.X.X.X\MVC\Samples\web\Scripts\cultures".**

ASPX-CS

```
//1036 refers to "fr-FR" culture.
@Html.EJ().Pivot().PivotClient("PivotClient1").DataSource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();})
.Data("https://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1036;").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works")).Locale("fr-FR")
```



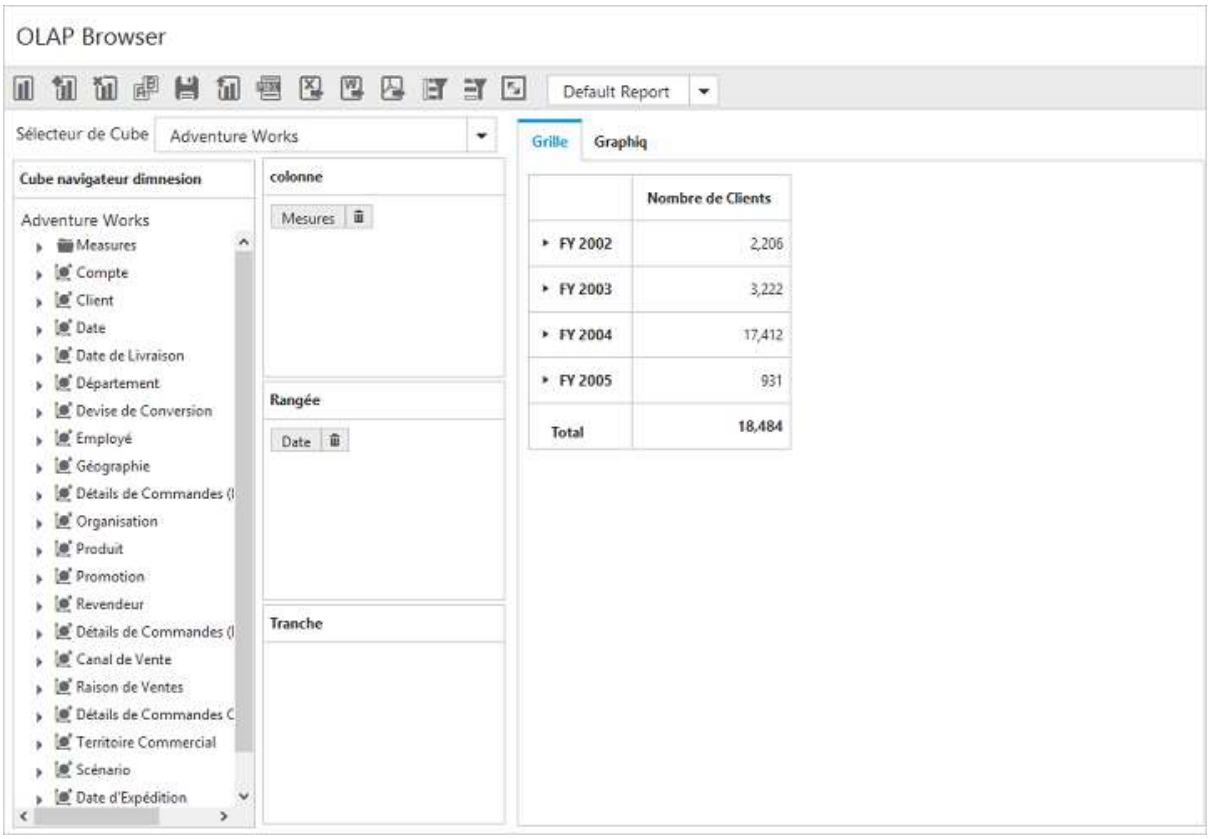
Localization and Globalization of Cube Info (Server Mode)

Content displayed within the PivotClient control are obtained from the OLAP cube. Following are the steps to get the localized and globalized Cube content.

- To get the localized string based on different cultures, from OLAP cube, you need to set **"Locale Identifier"** in the connection string to a specific culture.
- To bind the globalized content in PivotClient control, you need to set **"Culture"** and OverrideDefaultFormatStrings properties in OlapDataManager class to a specific culture.

C#

```
//1036 refers to "fr-FR" culture.
string connectionString = "Data Source=localhost; Initial
Catalog=Adventure Works DW; Locale Identifier=1036;";
DataManager = new OlapDataManager(connectionString);
DataManager.Culture = new System.Globalization.CultureInfo(1036);
DataManager.OverrideDefaultFormatStrings = true;
```



Localization and Globalization of Relational Info (Client Mode)

Content displayed within the PivotClient control are obtained from the Relational datasource. Following are the steps to get localized as well as globalized content.

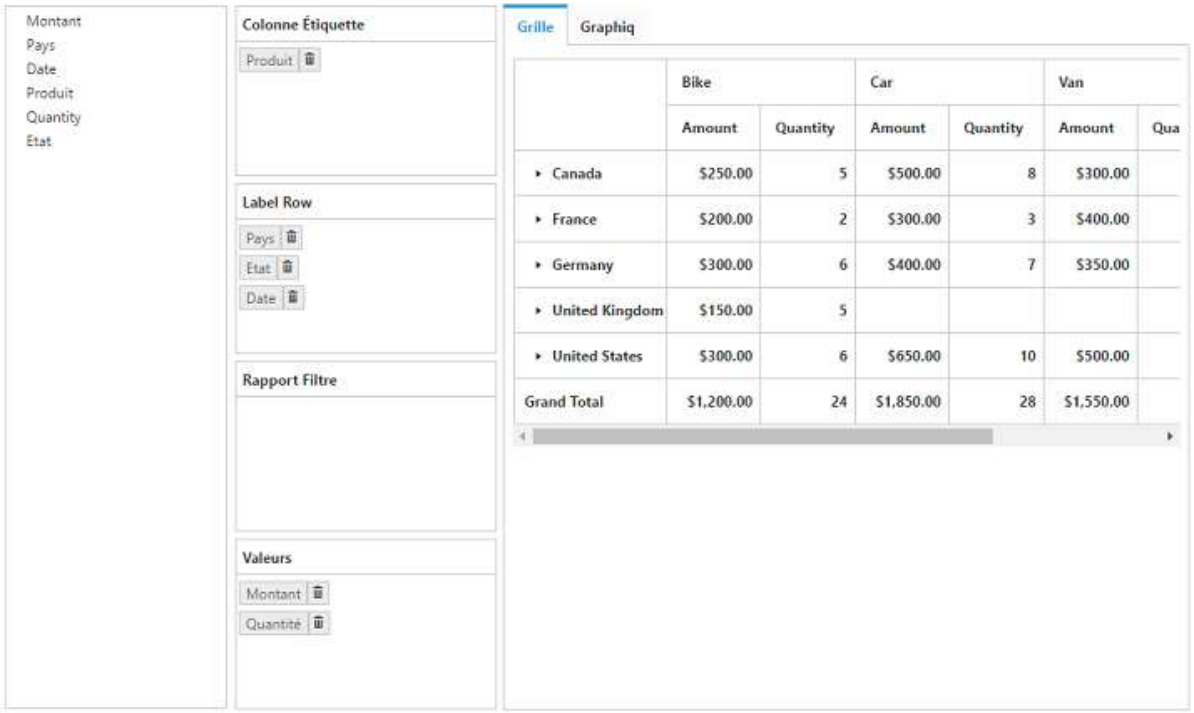
- To get the localized content, the Relational datasource must have localized headers in them which will be directly applied to PivotClient.
- To globalize the values appeared in PivotClient, we need to set the **"Format"** and **"Locale"** property accordingly. Also the specific culture file is referred in the sample.

Note: Culture files are present under **"[installed drive]:\Users\[user name]\AppData\Local\Syncfusion\EssentialStudio\X.X.X.X\MVC\Samples\web\Scripts\cultures".**

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add(); }))
```



Localization and Globalization of Relational Info (Server Mode)

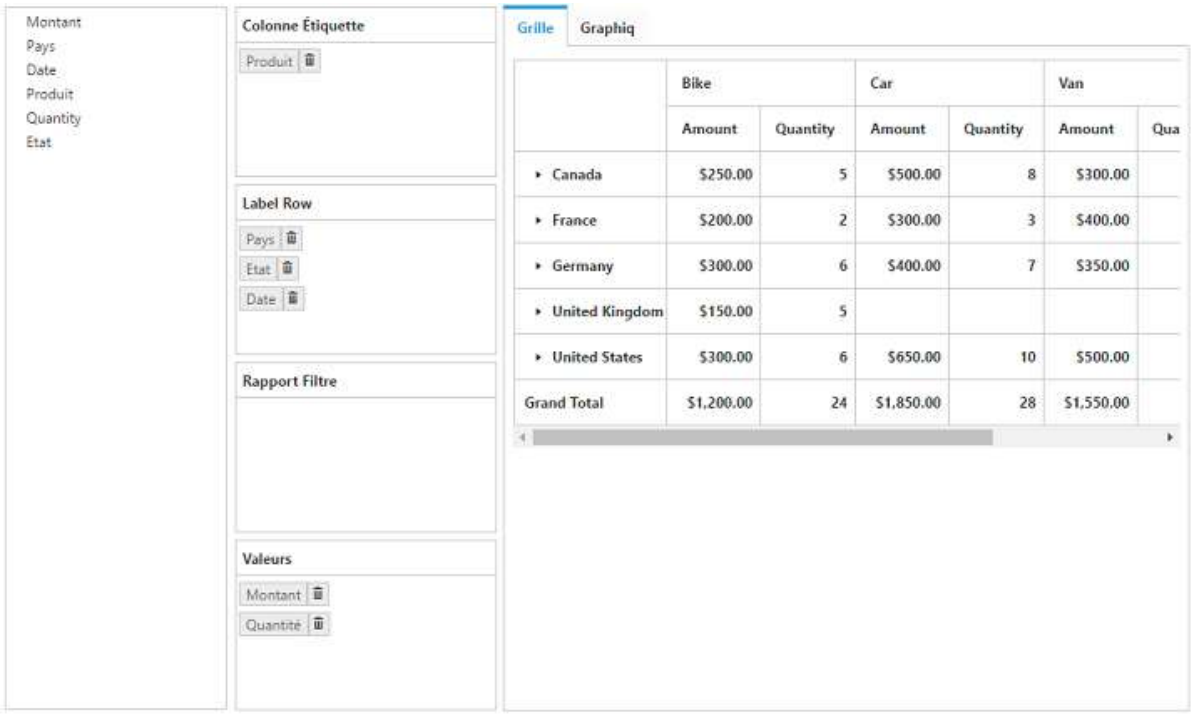
Content displayed within the PivotClient control are obtained from the Relational datasource. Following are the steps to get localized as well as globalized content.

- To get the localized content, the Relational datasource must have localized headers in them which will be directly applied to PivotClient.
- **"Format"** settings in "PivotComputationInfo" class would globalize the values appeared in PivotClient.

C#

```
PivotReport pivotSetting = new PivotReport();
//...
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum
});
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.



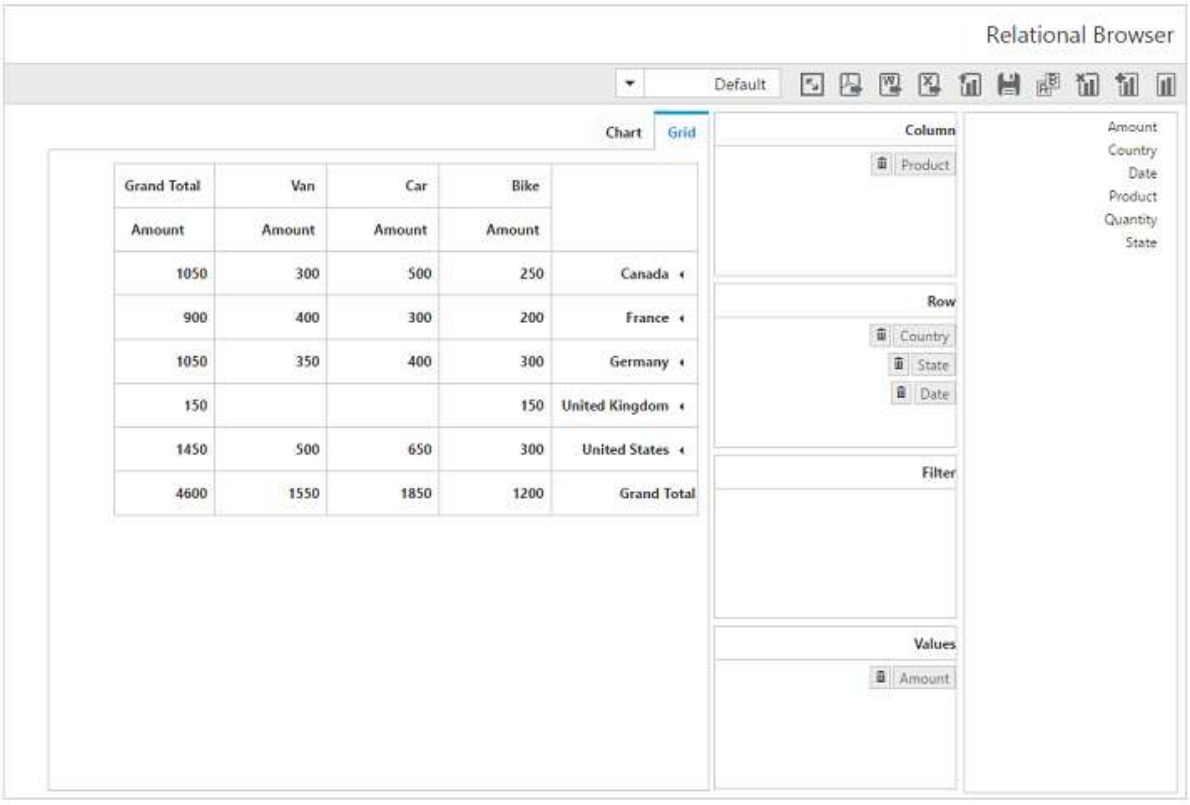
RTL

You can render our PivotClient control from Right to Left by setting EnableRTL property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add(); })).EnableRTL(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



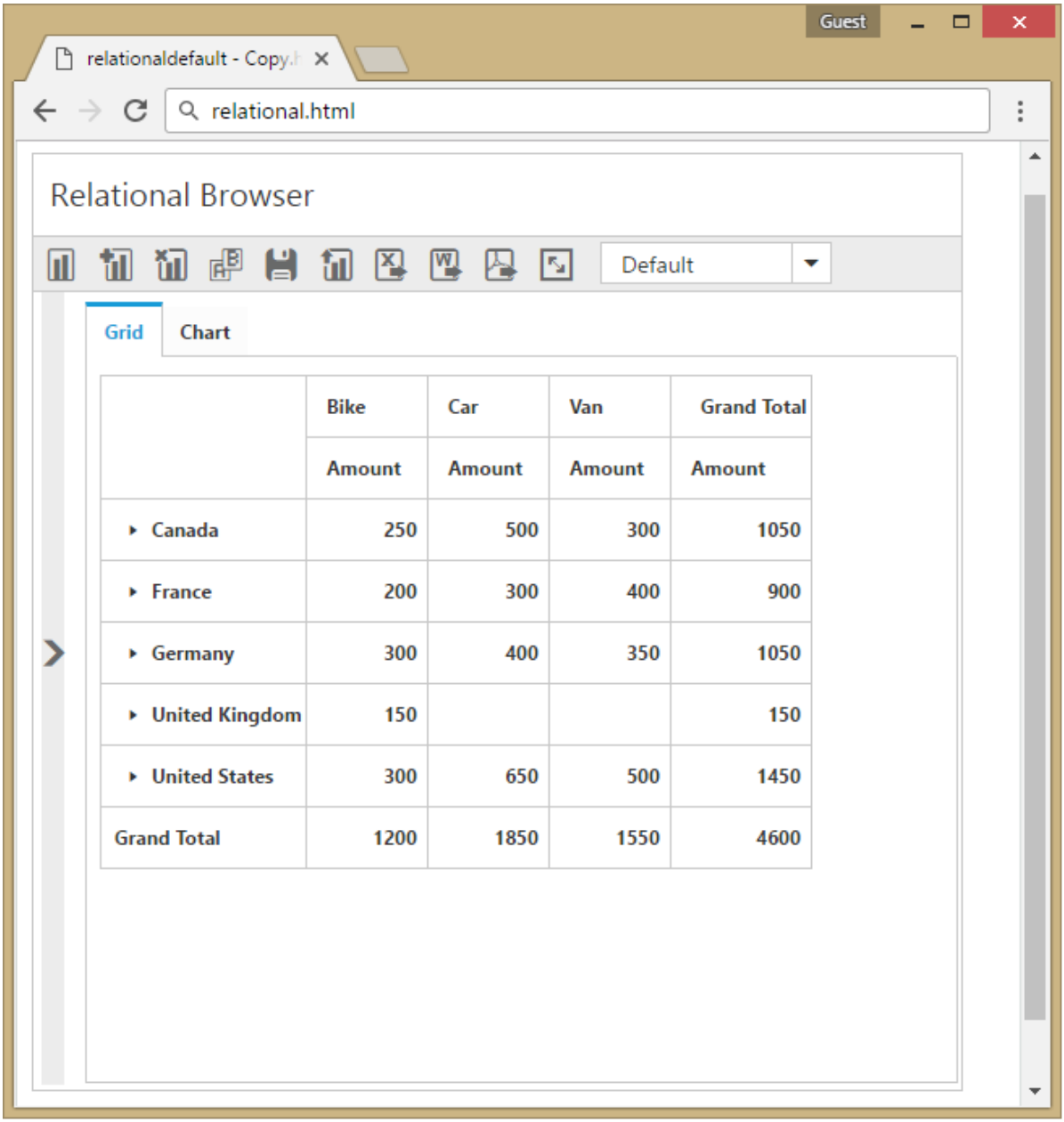
Responsive Layout

PivotClient widget supports responsive rendering based on the target device (desktop and tablet) resolution. It supports resolution up to 1024x600. You can enable responsiveness in PivotClient by setting IsResponsive property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).IsResponsive(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Splitter

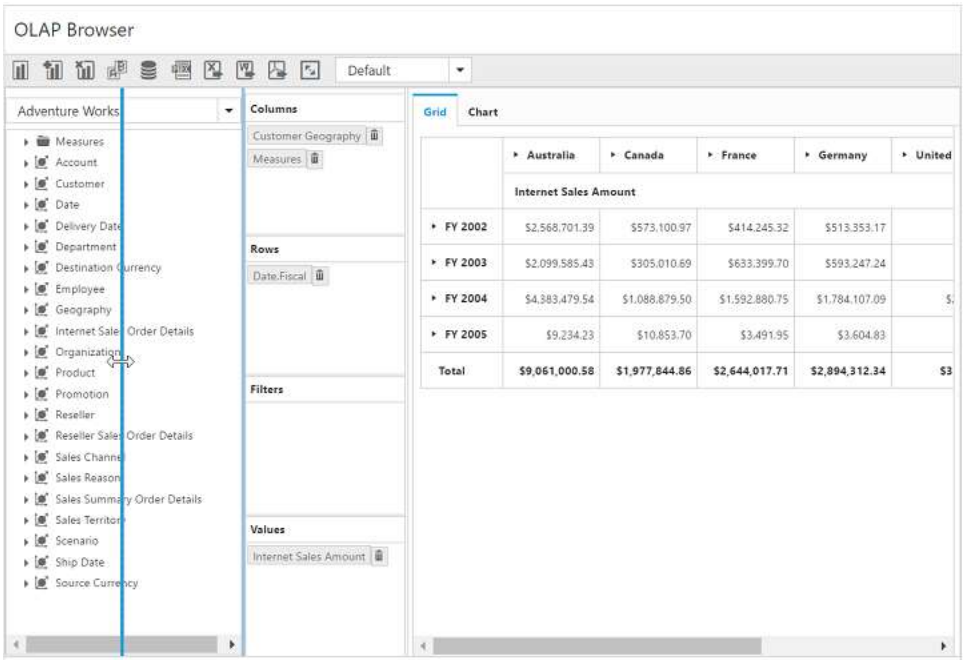
Information: This feature is not applicable for OLAP datasource bound from server-side.

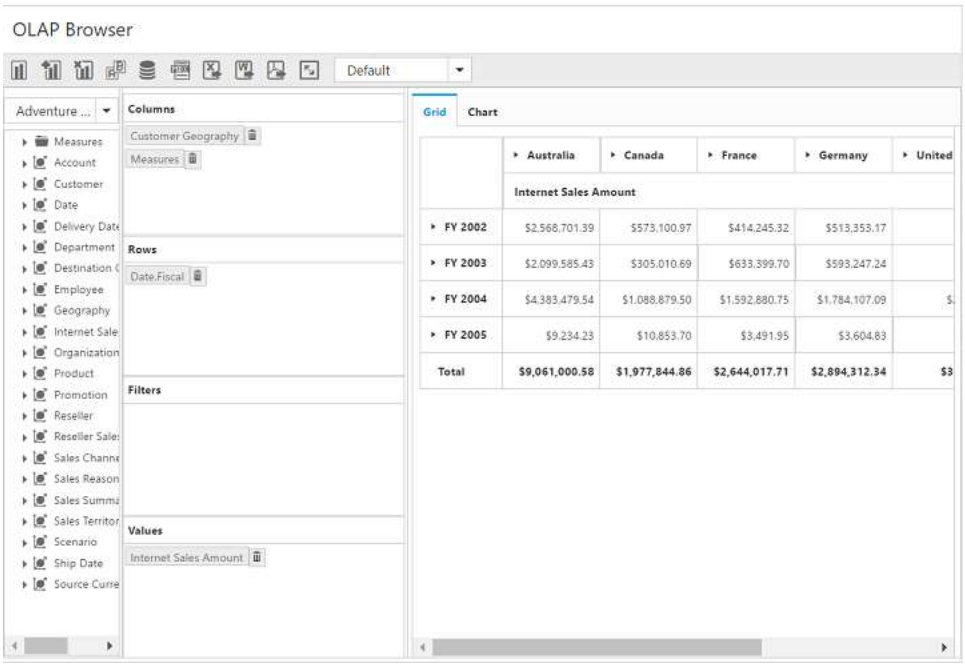
You can resize Cube Dimension Browser and Axis Element Builder by setting EnableSplitter property to true.This property is disabled by default.

ASPX-CS

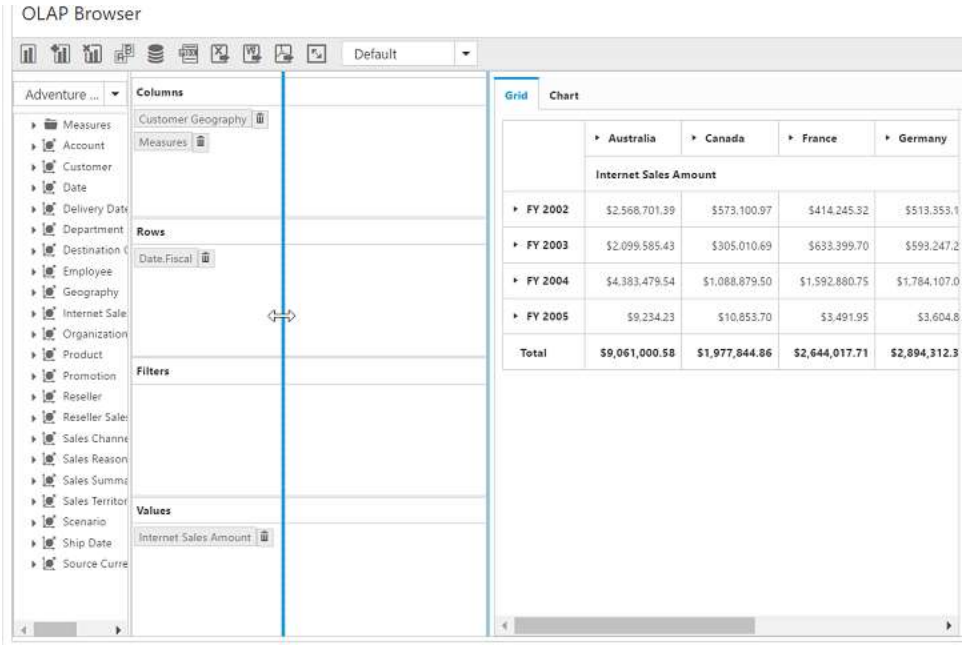
```
@Html.EJ().Pivot().PivotClient("PivotClient1").ClientSideEvents(clientSid
eEvents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
```

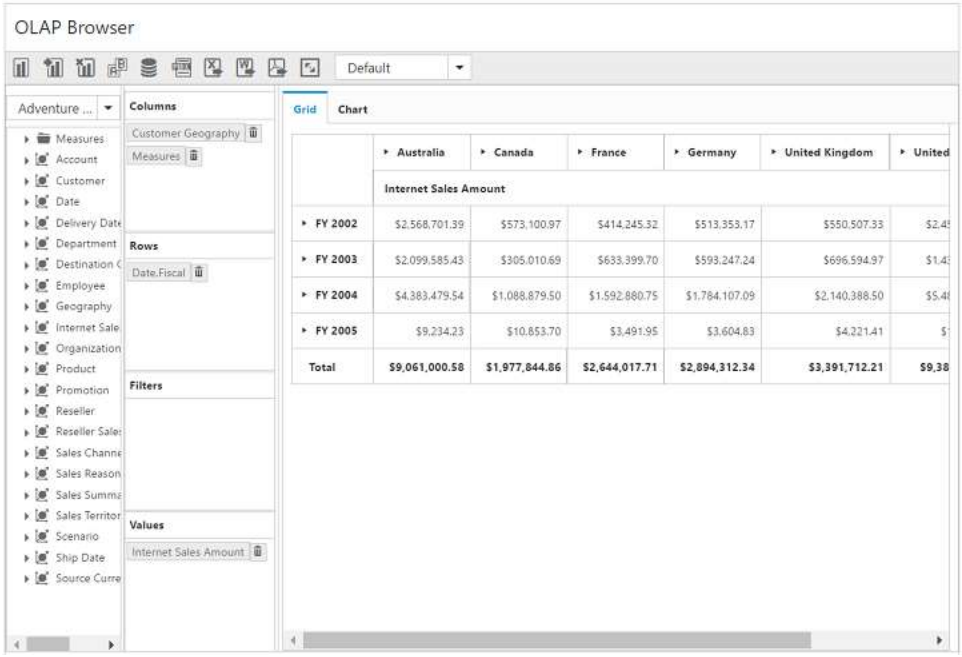
```
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
})).EnableSplitter(true)
```





Copyright © 2001 - 2020 Syncfusion Inc.





How to

Configure pivot client through the model properties

There comes a time when you want to set pivot client properties other than using `PivotClientPropertiesBuilder` i.e., in the Controller side. At that time, you can pass `PivotClientProperties` model instance as a parameter to the pivot client helper method.

We can set properties to pivot client control in server-side using the `PivotClientProperties` class and it is used in the view page with the help of pivot client helper overload.

In addition to the `id` parameter of pivot client helper method, we can also pass `PivotClientProperties` model as another parameter to the pivot client helper method.

Copyright © 2001 - 2020 Syncfusion Inc.

Relational

The following code example explains how to render the pivot client control in relational mode.

RAZOR

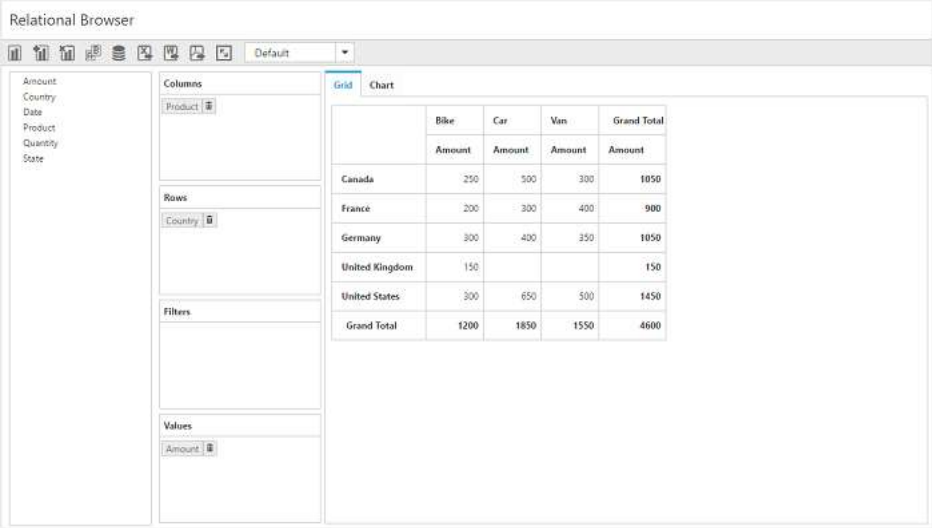
```
@model Syncfusion.JavaScript.Models.PivotClientProperties
@(Html.EJ().Pivot().PivotClient("PivotClient", Model))
<script type="text/javascript">
function load(args) {
args.model.dataSource.data = pivot_dataset;
}
</script>
```

C#

```
namespace PivotClient
{
public class PivotClientController: Controller
{
public ActionResult PivotClientFeatures()
{
Syncfusion.JavaScript.Models.PivotClientProperties pivotclient = new
Syncfusion.JavaScript.Models.PivotClientProperties();
PivotDataSource pivotclientDS = new PivotDataSource();
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "Country", FieldCaption = "Country"
});
rows.Add(new Field() { FieldName = "State", FieldCaption = "State" });
pivotclientDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "Product", FieldCaption = "Product"
});
pivotclientDS.Columns = cols;
pivotclient.Load = "load";
List<Field> values = new List<Field>();
values.Add(new Field() { FieldName = "Amount" });
pivotclientDS.Values = values;
pivotclient.DataSource = pivotclientDS;
return View(pivotclient);
}
}
}
```

As a result of the previous code example, the pivot client will be displayed as shown below:

Copyright © 2001 - 2020 Syncfusion Inc.



OLAP

The following code example explains how to render the pivot client control in OLAP mode.

RAZOR

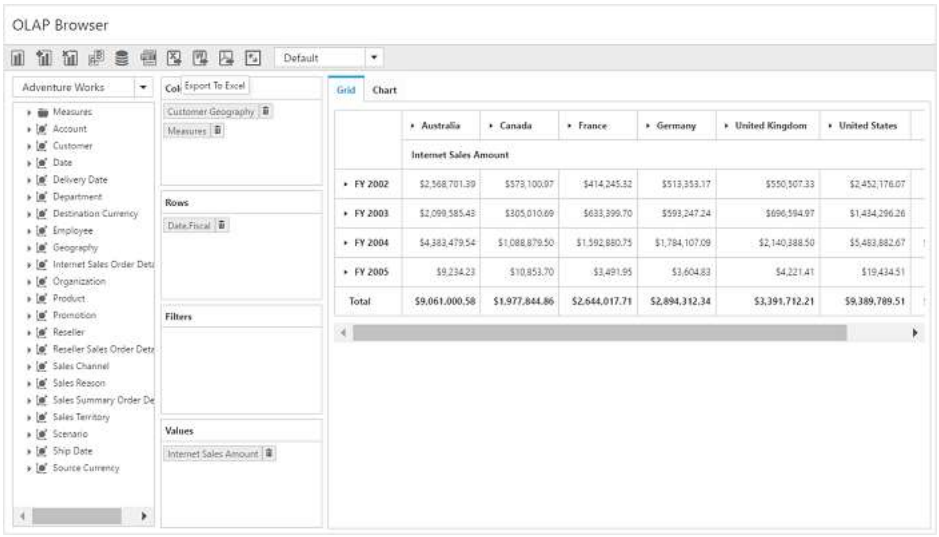
```
@model Syncfusion.JavaScript.Models.PivotClientProperties
@(Html.EJ().Pivot().PivotClient("PivotClient", Model))
```

C#

```
namespace PivotClient
{
public class PivotClientController: Controller
{
public ActionResult PivotClientFeatures()
{
Syncfusion.JavaScript.Models.PivotClientProperties pivotclient = new
Syncfusion.JavaScript.Models.PivotClientProperties();
PivotDataSource pivotclientDS = new PivotDataSource();
pivotclientDS.Data = "https://bi.syncfusion.com/olap/msmdpump.dll";
pivotclientDS.Cube = "Adventure Works";
pivotclientDS.Catalog = "Adventure Works DW 2008 SE";
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "[Date].[Fiscal]" });
pivotclientDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "[Customer].[Customer Geography]" });
pivotclientDS.Columns = cols;
List<MeasuresItems> measures = new List<MeasuresItems>();
measures.Add(new MeasuresItems() { FieldName = "[Measures].[Internet
Sales Amount]" });
List<Field> values = new List<Field>();
values.Add(new Field() { Measures = measures, Axis =
Syncfusion.JavaScript.AxisName.Column });
pivotclientDS.Values = values;
pivotclient.DataSource = pivotclientDS;
return View(pivotclient);
}
```

```
}
}
```

As a result of the previous code example, the pivot client will be displayed as shown below:



PivotGauge

Overview

The **PivotGauge** control for **Essential ASP.NET MVC** reads both **OLAP** and **Relational** datasource.

Key Features

The key features of the **PivotGauge** control is as follows:

- **Data source:** Supports OLAP data binding with Microsoft Analysis Service database, **XML/A** data sources and Relational data sources.
- **Tooltip:** Displays the value and goal information in the tooltip.
- **Multiple gauges and Layouts:** Support to customize the layout while rendering multiple controls.
- **Frame types:** Built-in frame types provide a rich appearance of control.
- **Indicators:** Displays the active/inactive state of PivotGauge.
- **Ranges:** Highlighting the range of values in scale.
- **Pointers:** Points the actual value and goal information.
- **Header labels:** Support to show or hide header labels and indicators.

Getting Started

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotGauge and Relational datasource (Client Mode)

This section covers the information required to populate a simple PivotGauge with Relational data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGaugeDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotGauge widget in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate widget related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotGauge

Before initializing, empty the contents of "Index.cshtml" file under Views > Home folder and add the following codes.

ASPX-CS

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotGauge("PivotGauge1")
```

Populate PivotGauge With Data

Let us now see how to populate the PivotGauge control using a sample JSON data as shown below.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").ClientSideEvents(clientSideE
vents => clientSideEvents.Load("onLoad"))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = [
{ Amount: 100, Country: "Canada", Product: "Bike" },
{ Amount: 200, Country: "Germany", Product: "Van" },
{ Amount: 300, Country: "Germany", Product: "Car" },
{ Amount: 150, Country: "United Kingdom", Product: "Bike" },
{ Amount: 200, Country: "Canada", Product: "Car" }
]
}
</script>
```

The JSON data is set to the **“Data”** property present inside the **“DataSource”** object. The **“DataSource”** object allows us to set the raw data input and the fields in the rows, columns, values and filters.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").ClientSideEvents(clientSideE
vents => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();}).Columns(column
s => { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();})).Scales(scale
=>
{
scale.ShowRanges(true).Radius(150).ShowScaleBar(true).Size(1).Border(bor
=> bor.Width(0.5)).ShowIndicators(true).ShowLabels(true).
Pointers(pointer =>
{
pointer.ShowBackNeedle(true).BackNeedleLength(20).Length(120).Width(7).Ad
d();
pointer.Type(PointerType.Marker).DistanceFromScale(5).Placement(PointerPl
acement.Center).BackgroundColor("#29A4D9").Length(25).Width(15).MarkerTyp
e(MarkerType.Diamond).Add();
}).
Ticks(ticks =>
{
ticks.Type(CircularTickTypes.Major).DistanceFromScale(2).Height(16).Width
(1).Color("#8c8c8c").Add();
ticks.Type(CircularTickTypes.Minor).Height(6).Width(1).DistanceFromScale(
2).Color("#8c8c8c").Add();
})
.Labels(labels => { labels.Color("#8c8c8c").Add(); })
.Ranges(ranges =>
{
ranges.DistanceFromScale(-5).BackgroundColor("#fc0606").Border(bor =>
bor.Color("#fc0606")).Add();
ranges.DistanceFromScale(-5).Add();
})
.CustomLabels(customLabel =>
{
customLabel.Position(location => location.X(180).Y(290)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(320)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(150)).Font(font =>
font.Size("12px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
}).Add();
})
```

The above code will generate a simple PivotGauge as shown below.

Copyright © 2001 - 2020 Syncfusion Inc.



Creating a simple application with PivotGauge and Relational datasource (Server Mode)

This section covers the information required to create a simple PivotGauge bound to Relational datasource from server-side.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGaugeDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotGauge widget in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate widget related scripts.

ASPX-CS

```
<body>
//....
//....
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

ASPX-CS

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Url(Url.Content("/Relational
")).BackgroundColor("transparent").EnableTooltip(true).Scales(scale =>
{
scale.ShowRanges(true).Radius(150).ShowScaleBar(true).Size(1).Border(bor
=> bor.Width(0.5)).ShowIndicators(true).ShowLabels(true).
Pointers(pointer =>
{
pointer.ShowBackNeedle(true).BackNeedleLength(20).Length(120).Width(7).Ad
d();
pointer.Type(PointerType.Marker).DistanceFromScale(5).Placement(PointerPl
acement.Center).BackgroundColor("#29A4D9").Length(25).Width(15).MarkerTyp
e(MarkerType.Diamond).Add();
}).
Ticks(ticks =>
{
ticks.Type(CircularTickTypes.Major).DistanceFromScale(2).Height(16).Width
(1).Color("#8c8c8c").Add();
ticks.Type(CircularTickTypes.Minor).Height(6).Width(1).DistanceFromScale(
2).Color("#8c8c8c").Add();
})
.Labels(labels => { labels.Color("#8c8c8c").Add(); })
.Ranges(ranges =>
{
ranges.DistanceFromScale(-5).BackgroundColor("#fc0606").Border(bor =>
bor.Color("#fc0606")).Add();
ranges.DistanceFromScale(-5).Add();
})
.CustomLabels(customLabel =>
{
customLabel.Position(location => location.X(180).Y(290)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(320)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(150)).Font(font =>
font.Size("12px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
}).Add();
}).ClientSideEvents(oCli => { oCli.RenderSuccess("loadPivotGaugeTheme");
}).IsResponsive(true).LabelFormatSettings(labelFormatSettings=>
labelFormatSettings.DecimalPlaces(2))
```

The **"Url"** property in PivotGauge widget points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotGauge widget as endpoint are WCF and WebAPI.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Relational". If WCF service is used as endpoint, the URL would look like "/RelationalService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in an existing MVC Web Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WebAPI Controller Class** and name it as **"RelationalController.cs"**, click **Add**.

Now, WebAPI controller is added to the application successfully with the file **"RelationalController.cs"**.

Note: While adding WebAPI Controller Class, name it with the suffix "Controller" that is mandatory. For example, in demo the controller is named as "RelationalController".

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside RelationalController.cs file.

C#

```
namespace PivotGaugeDemo
{
public class RelationalController: ApiController
{
}
}
```

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside RelationalController.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
namespace PivotGaugeDemo
{
public class RelationalController : ApiController
{
}
}
```

Datasource Initialization

A simple collection is provided as a datasource for the PivotGauge in this demo section. This datasource is placed inside a separate class "ProductSales" in RelationalController.cs file. Refer to the following code example.

C#

```
namespace PivotGaugeDemo
{
//....
//....
internal class ProductSales
{
```

```
public string Product { get; set; }
public string Date { get; set; }
public string Country { get; set; }
public string State { get; set; }
public int Quantity { get; set; }
public double Amount { get; set; }
public static ProductSalesCollection GetSalesData()
{
/// Geography
string[] countries = new string[] { "Australia", "Canada", "France",
"Germany", "United Kingdom", "United States" };
string[] ausStates = new string[] { "New South Wales", "Queensland",
"South Australia", "Tasmania", "Victoria" };
string[] canadaStates = new string[] { "Alberta", "British Columbia",
"Brunswick", "Manitoba", "Ontario", "Quebec" };
string[] franceStates = new string[] { "Charente-Maritime", "Essonne",
"Garonne (Haute)", "Gers", };
string[] germanyStates = new string[] { "Bayern", "Brandenburg",
"Hamburg", "Hessen", "Nordrhein-Westfalen", "Saarland" };
string[] ukStates = new string[] { "England" };
string[] ussStates = new string[] { "New York", "North Carolina",
"Alabama", "California", "Colorado", "New Mexico", "South Carolina" };
/// Time
string[] dates = new string[] { "FY 2005", "FY 2006", "FY 2007", "FY
2008", "FY 2009" };
/// Products
string[] products = new string[] { "Bike", "Van", "Car" };
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++)
{
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product)
{
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country)
{
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
```

```
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString()
{
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection : List<ProductSales>
{
}
}
}
```

Service methods in WebAPI Controller

Now you need to define the service methods inside RelationalController class, found inside RelationalController.cs file, created while adding WebAPI Controller Class to your Web Application.

C#

```
namespace PivotGaugeDemo
{
public class RelationalController : ApiController
{
PivotGauge pivotGauge = new PivotGauge();
[HttpPost]
[ActionName("InitializeGauge")]
public Dictionary<string, object> InitializeGauge(Dictionary<string,
object> jsonResult)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
pivotGauge.PivotReport = BindDefaultData();
return pivotGauge.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData());
}
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Date",
FieldHeader = "Date", TotalHeader = "Total" });
pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName =
"Product", FieldHeader = "Product", TotalHeader = "Total" });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
return pivotSetting;
}
}
}
```

Configure routing in WebAPIConfig Class

Open the "WebAPIConfig.cs" file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, **PivotGauge** will be rendered as shown in the below figure.

Copyright © 2001 - 2020 Syncfusion Inc.



WCF

This section demonstrates the utilization of WCF service as endpoint binding Relational datasource to a simple PivotGauge. For more details on this topic, click here.

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotGauge and OLAP datasource (Client Mode)

This section covers the information required to populate a simple PivotGauge with OLAP data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGaugeDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotGauge widget in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate widget related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Initialize PivotGauge

Before initializing, empty the contents of "Index.cshtml" file under Views > Home folder and add the following codes.

ASPX-CS

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotGauge("PivotGauge1")
```

Populate PivotGauge With DataSource

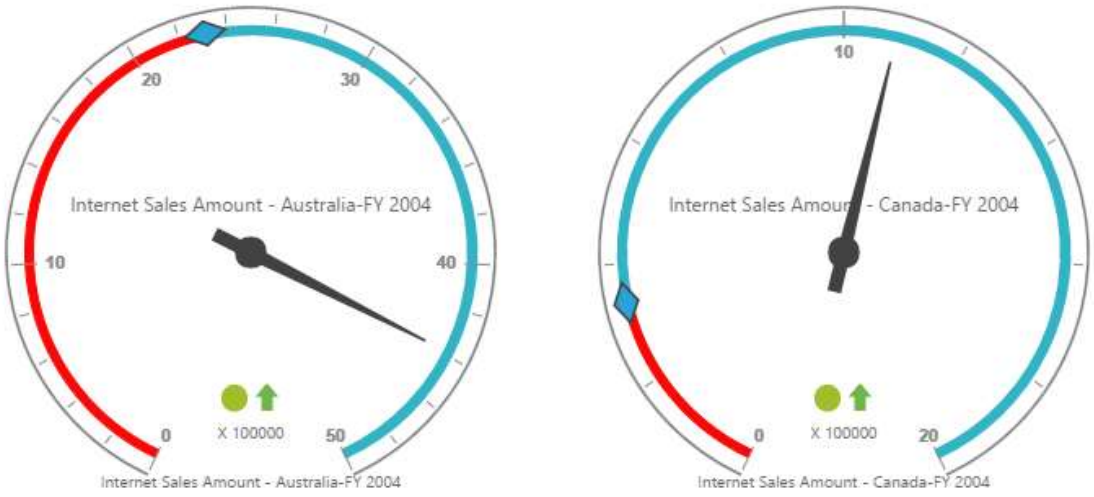
Initializes the OLAP datasource for PivotGauge widget as shown below.

HTML

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Date].[Fiscal]").Add();
}).Columns(columns => { columns.FieldName("[Customer].[Customer
Geography]").Add(); }).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
measures.FieldName("[Measures].[Internet Revenue Status]").Add();
measures.FieldName("[Measures].[Internet Revenue Trend]").Add();
measures.FieldName("[Measures].[Internet Revenue Goal]").Add();
}).Axis(AxisName.Column).Add(); })
.Data("http://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works")).Scales(scale =>
{
scale.ShowRanges(true).Radius(150).ShowScaleBar(true).Size(1).Border(bor
=> bor.Width(0.5)).ShowIndicators(true).ShowLabels(true).
Pointers(pointer =>
{
pointer.ShowBackNeedle(true).BackNeedleLength(20).Length(120).Width(7).Ad
d();
pointer.Type(PointerType.Marker).DistanceFromScale(5).Placement(PointerPl
acement.Center).BackgroundColor("#29A4D9").Length(25).Width(15).MarkerTyp
e(MarkerType.Diamond).Add();
}).
Ticks(ticks =>
{
ticks.Type(CircularTickTypes.Major).DistanceFromScale(2).Height(16).Width
(1).Color("#8c8c8c").Add();
ticks.Type(CircularTickTypes.Minor).Height(6).Width(1).DistanceFromScale(
2).Color("#8c8c8c").Add();
})
.Labels(labels => { labels.Color("#8c8c8c").Add(); })
.Ranges(ranges =>
{
ranges.DistanceFromScale(-5).BackgroundColor("#fc0606").Border(bor =>
bor.Color("#fc0606")).Add();
ranges.DistanceFromScale(-5).Add();
})
.CustomLabels(customLabel =>
{
customLabel.Position(location => location.X(180).Y(290)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
```

```
customLabel.Position(location => location.X(180).Y(320)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(150)).Font(font =>
font.Size("12px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
}).Add();
})
```

The above code will generate a simple PivotGauge as shown in below figure.



Creating a simple application with PivotGauge and OLAP datasource (Server Mode)

This section covers the information required to create a simple PivotGauge bound to OLAP datasource.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGaugeDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in Web.config files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in Web.config file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotGauge widget in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the **head** tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

ASPX-CS

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the `RenderBody()` function call in **_Layout.cshtml** file in-order to generate widget related scripts.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

ASPX-CS

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Url(Url.Content("/Olap")).Ba
ckgroundColor("transparent").EnableTooltip(true).Scales(scale =>
{
scale.ShowRanges(true).Radius(150).ShowScaleBar(true).Size(1).Border(bor
=> bor.Width(0.5)).ShowIndicators(true).ShowLabels(true).
Pointers(pointer =>
{
pointer.ShowBackNeedle(true).BackNeedleLength(20).Length(120).Width(7).Ad
d();
pointer.Type(PointerType.Marker).DistanceFromScale(5).Placement(PointerPl
acement.Center).BackgroundColor("#29A4D9").Length(25).Width(15).MarkerTyp
e(MarkerType.Diamond).Add();
}).
Ticks(ticks =>
{
ticks.Type(CircularTickTypes.Major).DistanceFromScale(2).Height(16).Width
(1).Color("#8c8c8c").Add();
ticks.Type(CircularTickTypes.Minor).Height(6).Width(1).DistanceFromScale(
2).Color("#8c8c8c").Add();
})
.Labels(labels => { labels.Color("#8c8c8c").Add(); })
.Ranges(ranges =>
{
ranges.DistanceFromScale(-5).BackgroundColor("#fc0606").Border(bor =>
bor.Color("#fc0606")).Add();
ranges.DistanceFromScale(-5).Add();
})
.CustomLabels(customLabel =>
{
customLabel.Position(location => location.X(180).Y(290)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(320)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
customLabel.Position(location => location.X(180).Y(150)).Font(font =>
font.Size("12px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("#666666").Add();
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).ClientSideEvents(oCli => { oCli.RenderSuccess("loadPivotGaugeTheme");
}).IsResponsive(true).LabelFormatSettings(labelFormatSettings=>
labelFormatSettings.DecimalPlaces(2))
```

The **"Url"** property in PivotGauge widget points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotGauge widget as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Olap". If WCF service is used as endpoint, the URL would look like "/OlapService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in your existing Web Application, right-click on the project in Solution Explorer and select **Add > New Item.** In the **Add New Item** window, select **WebAPI Controller Class** and name it as "OlapController.cs", click **Add.**

Now WebAPI controller is added into your application successfully which in-turn comprise of the following file. The utilization of this file will be explained in the following sections.

- OlapController.cs

Note: While adding WebAPI Controller Class, name it with the suffix "Controller" that is mandatory. For example, in demo the controller is named as "OlapController".

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside OlapController.cs file.

C#

```
namespace PivotGaugeDemo
{
public class OlapController : ApiController
{
}
}
```

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapController.cs file.

C#

```
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript;
namespace PivotGaugeDemo
{
public class OlapController : ApiController
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube and PivotGauge instances are created immediately inside the main class in OlapController.cs file.

C#

```
namespace PivotGaugeDemo
{
public class OlapController : ApiController
{
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
PivotGauge pivotGauge = new PivotGauge();
//Other codes
}
}
```

Service methods in WebAPI Controller

Now you need to define the service methods inside OlapController class, found inside OlapController.cs file, created while adding WebAPI Controller Class to your Web Application.

C#

```
namespace PivotGaugeDemo
{
public class OlapController : ApiController
{
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
PivotGauge pivotGauge = new PivotGauge();
[System.Web.Http.ActionName("InitializeGauge")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeGauge(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return pivotGauge.GetJsonData(jsonResult["action"].ToString(),
DataManager);
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.Name = "Default Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");
MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name = "Customer
Count" });
DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
```

```
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
return olapReport;
}
}
}
```

Configure routing in WebAPIConfig Class

Open the WebAPIConfig.cs file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, PivotGauge will be rendered with the provided data as shown below.





WCF

This section demonstrates the utilization of WCF service as endpoint binding OLAP datasource to a simple PivotGauge. For more details on this topic, click here.

Data Binding

DataBinding

Binding PivotGauge to Collection

This section demonstrates binding of a collection to the PivotGauge control as datasource. For more information on this datasource refer to the following links.

When you are using WebAPI controller, refer to the “Datasource Initialization” section under the following link.

If you are using WCF Service, refer the "Datasource Initialization" section below.

WCF

Adding a WCF Service

To add a WCF service in an existing application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WCF Service** and name it as **“RelationalService.svc”** and click **Add**.

Now WCF service is added into your application successfully which in-turn comprise of the following files. The utilization of these files will be explained in the immediate sections.

- RelationalService.svc
- RelationalService.svc.cs
- IRelationalService.cs

Configuring WCF Service Class

Remove the “DoWork” method present inside both RelationalService.svc.cs and IRelationalService.cs files. Next, add “AspNetCompatibilityRequirements” attribute on top of main class present inside RelationalService.svc.cs and set **“RequirementsMode”** value to **“Allowed”**.

C#

```
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.EJ
- Syncfusion.EJ.MVC
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside RelationalService.svc.cs file.

C#

```
using System.ServiceModel.Activation;
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

Datasource Initialization

A simple collection is provided as a datasource for our PivotGauge in this demo section. This datasource is placed inside a separate class named “ProductSales” in RelationalService.svc.cs file. Refer to the following code example.

C#

```
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
//....
//....
}
internal class ProductSales
{
public string Product { get; set; }
public string Date { get; set; }
public string Country { get; set; }
public string State { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public int Quantity { get; set; }
public double Amount { get; set; }
public static ProductSalesCollection GetSalesData()
{
/// Geography
string[] countries = new string[] { "Australia", "Canada", "France",
"Germany", "United Kingdom", "United States" };
string[] ausStates = new string[] { "New South Wales", "Queensland",
"South Australia", "Tasmania", "Victoria" };
string[] canadaStates = new string[] { "Alberta", "British Columbia",
"Brunswick", "Manitoba", "Ontario", "Quebec" };
string[] franceStates = new string[] { "Charente-Maritime", "Essonne",
"Garonne (Haute)", "Gers", };
string[] germanyStates = new string[] { "Bayern", "Brandenburg",
"Hamburg", "Hessen", "Nordrhein-Westfalen", "Saarland" };
string[] ukStates = new string[] { "England" };
string[] ussStates = new string[] { "New York", "North Carolina",
"Alabama", "California", "Colorado", "New Mexico", "South Carolina" };
/// Time
string[] dates = new string[] { "FY 2005", "FY 2006", "FY 2007", "FY
2008", "FY 2009" };
/// Products
string[] products = new string[] { "Bike", "Van", "Car" };
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++)
{
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product)
{
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country)
{
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
```

```
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString()
{
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection : List<ProductSales>
{
}
}
}
```

Service methods in WCF Service

First, you need to define the service methods inside IRelationalService interface, found in IRelationalService.cs file, created while adding WCF service to your application.

C#

```
namespace PivotGaugeDemo
{
[ServiceContract]
public interface IRelationalService
{
[OperationContract]
Dictionary<string, object> InitializeGauge(string action, string
customObject);
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Secondly, you need to elaborate the service methods inside the main class, found in RelationalService.svc.cs file.

C#

```
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
PivotGauge pivotGauge = new PivotGauge();
public Dictionary<string, object> InitializeGauge(string action, string
customObject)
{
pivotGauge.PivotReport = this.BindDefaultData();
return pivotGauge.GetJsonData(action, ProductSales.GetSalesData());
}
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Date",
FieldHeader = "Date", TotalHeader = "Total" });
pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName =
"Product", FieldHeader = "Product", TotalHeader = "Total" });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
return pivotSetting;
}
}
.....
..... // Datasource initialization
.....
}
```

Configuring Web Configuration File

You can expose services through the properties such as binding, contract and address by using an endpoint.

1. Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IRelationalService contract and hence it is PivotGaugeDemo.IRelationalService.
2. Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
3. behaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

XML

```
<system.serviceModel>
……
```

```
……
<services>
<service name="PivotGaugeDemo.RelationalService">
<endpoint address=""
behaviorConfiguration="PivotGaugeDemo.RelationalServiceAspNetAjaxBehavior
"
binding="webHttpBinding" contract="PivotGaugeDemo.IRelationalService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotGaugeDemo.RelationalServiceAspNetAjaxBehavior">
<enableWebScript />
</behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Note: In this example, **"PivotGaugeDemo"** indicates the name and root namespace of the application created in Visual Studio IDE and **"RelationalService"** indicates the name of the WCF service created.

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
}
```

Now, **PivotGauge** will be rendered as shown in the below figure.

Copyright © 2001 - 2020 Syncfusion Inc.



DataBinding

Binding PivotGauge to Offline Cube

To connect an OLAP Cube available in local machine, set the physical path of the Cube in the connection string. The following code example illustrates the same.

C#

```
string connectionString = @"DataSource = system
drive:\OfflineCube\Adventure_Works_Ext.cub; Provider = MSOLAP;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotGauge to Cube in local SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in local machine, set the server name and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = "Data source=localhost; Initial
Catalog=Adventure Works DW;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotGauge to Cube in online SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in online server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotGauge to Cube in online Mondrian Server

To connect an OLAP Cube available in Mondrian Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = @"Data Source =
http://localhost:8080/mondrian/xmla; Initial Catalog =FoodMart;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName =
Syncfusion.Olap.DataProvider.Providers.Mondrian;
```

Binding PivotGauge to Cube in online ActivePivot Server

To connect an OLAP Cube available in ActivePivot Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = @"Data Source =
http://localhost:8080/cva_s/xmla; Initial Catalog = CVAS;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName=Syncfusion.Olap.DataProvider.Provid
ers.ActivePivot;
```

WCF

Adding a WCF Service

To add a WCF service in an existing web application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as **“OlapService.svc”**, click **Add**.

Now WCF service is added into your application successfully which in-turn comprise of the following files. The utilization of these files will be explained in the immediate sections.

- OlapService.svc
- OlapService.svc.cs
- IOlapService.cs

Configuring WCF Service Class

Remove the “DoWork” method present inside both OlapService.svc.cs and IOlapService.cs files. Next, add “AspNetCompatibilityRequirements” attribute on top of main class present inside OlapService.svc.cs and set “RequirementsMode” value to “Allowed”.

C#

```
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.EJ
- Syncfusion.EJ.MVC
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapService.svc.cs file.

C#

```
using System.ServiceModel.Activation;
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript;
using System.Collections.Generic;
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Datasource Initialization

Now the connection string to connect OLAP Cube and PivotGauge instances are created immediately inside the main class in OlapService.svc.cs file.

C#

```
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
PivotGauge pivotGauge = new PivotGauge();
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
//Other codes
}
}
```

Service methods in WCF Service

First, you need to define the service methods inside IOlapService interface, found in IOlapService.cs file, created while adding WCF Service to your Web Application.

C#

```
namespace PivotGaugeDemo
{
[ServiceContract]
public interface IOlapService
{
[OperationContract]
Dictionary<string, object> InitializeGauge(string action,string
customObject);
}
}
```

Secondly, you need to elaborate the service methods inside the main class, found in OlapService.svc.cs file

C#

```
namespace PivotGaugeDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
PivotGauge pivotGauge = new PivotGauge();
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
public Dictionary<string, object> InitializeGauge(string action,string
customObject)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return pivotGauge.GetJsonData(action, DataManager);
}
private OlapReport CreateOlapReport()
{
OlapReport report = new OlapReport();
report.CurrentCubeName = "Adventure Works";
//Specifying the KPI name
KpiElements kpiElement = new KpiElements();
kpiElement.Elements.Add(new KpiElement { Name = "Internet Revenue",
ShowKPIGoal = true, ShowKPIStatus = true, ShowKPIValue = true,
ShowKPITrend = true });
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the dimension name
dimensionElementColumn.Name = "Customer";
//Adding the level with the hierarchy Name
dimensionElementColumn.AddLevel("Customer Geography", "Country");
//Specifying the measure name
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Internet
Sales Amount" });
DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the dimension name
dimensionElementRow.Name = "Date";
//Adding the level with the hierarchy Name
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
dimensionElementRow.Hierarchy.LevelElements["Fiscal Year"].Add("FY
2004");
dimensionElementRow.Hierarchy.LevelElements["Fiscal
Year"].IncludeAvailableMembers = true;
report.CategoricalElements.Add(dimensionElementColumn);
report.CategoricalElements.Add(kpiElement);
report.CategoricalElements.Add(measureElementColumn);
report.SeriesElements.Add(dimensionElementRow);
return report;
}
}
}
```

Configuring Web Configuration File

You can expose services through the properties such as binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to **IOlapService** contract and hence it is **PivotGaugeDemo.IOlapService**.
- Binding: In your application, you use **webHttpBinding** to post and receive the requests and responses between the client-end and the service.
- behaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

XML

```
<system.serviceModel>
……
……
<services>
<service name="PivotGaugeDemo.OlapService">
<endpoint address=""
behaviorConfiguration="PivotGaugeDemo.OlapServiceAspNetAjaxBehavior"
binding="webHttpBinding" contract="PivotGaugeDemo.IOlapService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotGaugeDemo.OlapServiceAspNetAjaxBehavior">
<enableWebScript />
</behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Note: In this example, “PivotGaugeDemo” indicates the name and root namespace of the Web Application created in Visual Studio IDE and “OlapService” indicates the name of the WCF service created.

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
}
```

Now, PivotGauge is rendered with customer count over across different customer geographic locations.

Copyright © 2001 - 2020 Syncfusion Inc.





Frame Type

Full Circle

Full Circle frame lets the PivotGauge display in circular shape. Frame type can be set using the FrameType property. By default, the frame type is "FullCircle".

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Frame(frame=> {
frame.FrameType(Frame.FullCircle) })
```



Copyright © 2001 - 2020 Syncfusion Inc.

Half Circle

Half Circle frame lets the PivotGauge to display in semi-circular shape. For this, frame type needs to be set as "HalfCircle" within the FrameType property and need to set StartAngle and SweepAngle for the PivotGauge in the Scales property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Frame(frame=>frame.FrameType
(Frame.HalfCircle).HalfCircleFrameStartAngle(180).HalfCircleFrameEndAngle
(360)).Scales(scale =>{ scale.StartAngle(180).SweepAngle(180).Add();})
```



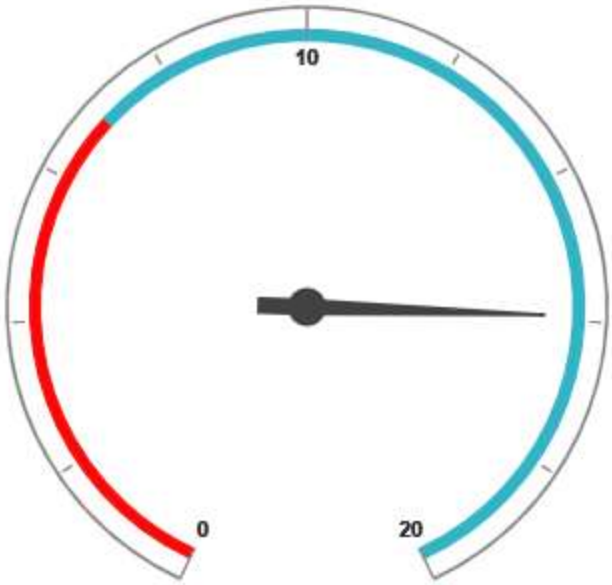
Scale

Adding Scale

Scale can be added within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.ShowScaleBar(true).Radius(150).Add(); })
```



Scale Customization

Pointer Cap

Pointer Cap is a circular shape element which is located at the center of the PivotGauge. It can be customized with the property PointerCap inside the scales option. Following are the properties used to customize its appearance.

- **Radius** – sets the radius of the pointer cap.
- **BorderColor** – sets the color of the pointer cap border.
- **BorderWidth** – sets the width of the pointer cap border.
- **BackgroundColor** – sets the background color of the pointer cap.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.ShowScaleBar(true).PointerCap(pointerCap =>
pointerCap.Radius(5).BorderWidth(2).BorderColor("green").BackgroundColor(
"yellow")).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Appearance

The appearance of the scale can be customized through the following properties.

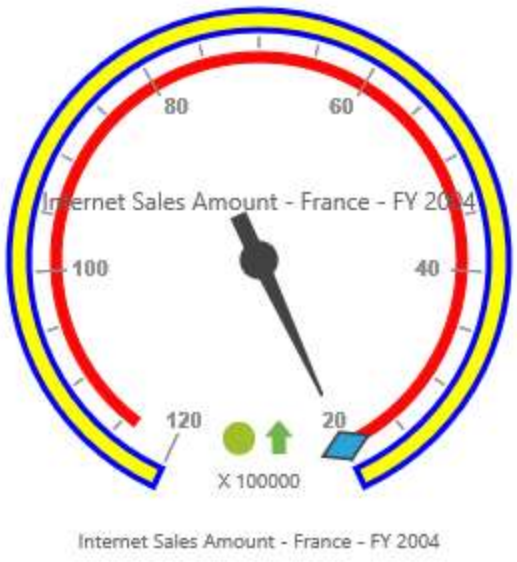
- **Radius** – sets the radius of the scale.
- **BackgroundColor** – sets the background color of the scale.
- **Border** – sets the border of the scale with color and width properties.
- **Size** – sets the size of the scale.
- **Minimum** – sets the least value of the scale.
- **Maximum** – sets the highest value of the scale.
- **MajorIntervalValue** – sets the interval between minor ticks in the scale.
- **MinorIntervalValue** – sets the interval between major ticks in the scale.
- **Direction** – sets the direction of the scale. By default it takes “Clockwise” direction.

The ShowIndicators, ShowTicks, ShowRanges, ShowPointers, and ShowScaleBar properties are used to enable/disable the indicators, ticks, ranges, pointers and scale bar respectively. By default, ShowTicks and ShowPointers are set to true, and other properties are set as false.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.ShowScaleBar(true).Radius(120).BackgroundColor("yellow").Border(bo
rder =>
border.Color("blue").Width(3)).Size(10).Minimum(20).Maximum(120).MajorInt
ervalValue(20).MinorIntervalValue(5).Direction(Directions.CounterClockwis
e).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Pointers

Pointer Types

PivotGauge pointers has two types such as,

- Needle
- Marker

Needle type pointers are the default pointers which is always located at the center of the Gauge. Following shapes that are supported for the needle pointers are:

- Rectangle
- Triangle
- Trapezoid
- Arrow
- Image.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
pointers.Type(PointerType.Needle).NeedleType(NeedleType.Trapezoid).Add();
}).Add(); })
```

Copyright © 2001 - 2020 Syncfusion Inc.



For marker pointer, the available shapes are Rectangle, Triangle, Ellipse, Diamond, Pentagon, Circle, Slider, Pointer, Wedge, Trapezoid, RoundedRectangle and Image.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
pointers.Type(PointerType.Marker).MarkerType(MarkerType.Diamond).Add();
}).Add(); })
```



Copyright © 2001 - 2020 Syncfusion Inc.

Adding Pointer Collection

Pointer collection can be directly added to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
pointers.Type(PointerType.Needle).NeedleType(NeedleType.Triangle).Add();
pointers.Type(PointerType.Marker).MarkerType(MarkerType.Diamond).Add();
}).Add(); })
```



Appearance Customization

The appearance of the pointer can be customized through the following properties.

- **Border** – sets the "Color" and "Width" of the pointer border.
- **BackgroundColor** – sets the background color of the pointer.
- **Length** – sets the length of the pointer.
- **Width** – sets the width of the pointer.
- **Opacity** – sets the opacity of the pointer. By default, the value is 1.
- **Type** – sets the type of the pointer. By default, the type is "Needle".

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
//For customizing needle pointer
pointers.Border(border =>
border.Color("green").Width(2)).BackgroundColor("yellow").Length(120).Wid
th(7).Opacity(0.6).Type(PointerType.Needle).NeedleType(NeedleType.Triangl
e).Add();
//For customizing marker pointer
```

```
pointers.Border(border =>
border.Color("green").Width(2)).BackgroundColor("yellow").Length(25).Widt
h(15).Opacity(0.8).Type(PointerType.Marker).MarkerType(MarkerType.Diamond
).Add();
}).Add();
})
```



Pointer Position

Pointer can be positioned with the help of below two properties.

- **DistanceFromScale** - defines the distance between scale and pointer. By default, the value is 0.
- **Placement** - defines the location of the pointer. By default, the value is "Center".

Note: Both the properties can be applied only if the pointer type is set to “Marker”. Needle pointer type appears only at the center of the control, which is its default position.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
pointers.Type(PointerType.Marker).DistanceFromScale(2).Placement(PointerP
lacement.Far).Add(); }).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Pointer Image

It is possible to replace the pointers with image. To view the pointers as image, we need to set the appropriate location in the ImageUrl property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
//For replacing needle pointer with image
pointers.Type(PointerType.Needle).NeedleType(NeedleType.Image).ImageUrl("
../image.png").Add();
//For replacing marker pointer with image
pointers.Type(PointerType.Marker).MarkerType(MarkerType.Image).ImageUrl("
../image.png").Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Pointer Value Text

To display the current value of the pointers in PivotGauge control, **"PointerValueText"** option inside pointers is used. Following are the properties used to enable and customize the pointer value text.

- **ShowValue** – enables the pointer value text by setting the property to "true". By default, its value is "true".
- **Distance** – sets the distance between pointer and text.
- **Color** – sets the color of the text.
- **Opacity** – sets the opacity of the text. By default, its value is 1.
- **Angle** – sets the rotation angle of the text. By default, its value is 0.
- **Font** – sets the font size, font style and font family of the text.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Pointers(pointers => {
//For needle type
pointers.Type(PointerType.Needle).PointerValueText(pointerValueText =>
pointerValueText.ShowValue(true).Distance(10).Color("red").Opacity(70).An
gle(20).Font(font =>
font.FontFamily("Arial").FontStyle("Normal").Size("15px"))).Add();
//For marker type
pointers.Type(PointerType.Marker).PointerValueText(pointerValueText =>
pointerValueText.ShowValue(true).Distance(40).Color("red").Opacity(70).An
gle(-40).Font(font =>
font.FontFamily("Arial").FontStyle("Normal").Size("15px"))).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Ticks

Adding Tick Collection

Tick collection can be directly added to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Ticks(ticks => { ticks.Type(CircularTickTypes.Major).Add();
}).Add();
})
```

Tick Customization

The appearance of the tick can be customized through the following properties.

- **Type** – indicates whether ticks are for major or minor intervals. By default, the type is "Major".
- **Height** – sets the height of the ticks.
- **Width** – sets the width of the ticks.
- **Angle** – rotates the ticks to a specified angle. By default, the angle value is 0.
- **Color** – displays the ticks in specified color.
- **DistanceFromScale** – sets the distance between scale and ticks. By default, the values is 0.
- **Placement** – positions the ticks with respect to the scale. By default, the value is set to "Far".

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Ticks(ticks => {
ticks.Type(CircularTickTypes.Major).Height(15).Width(4).Angle(0).Color("g
reen").DistanceFromScale(2).Placement(TickPlacement.Near).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Ranges

Adding Range Collection

Range collection can be directly added to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.ShowRanges(true).Ranges(ranges => {
ranges.DistanceFromScale(10).Add(); }).Add();
})
```

Appearance Customization

The appearance of the range can be customized through the following properties.

- StartValue – defines the start position of the range.
- EndValue – defines the end position of the range.
- StartWidth – sets the width at starting position of the range.
- EndWidth – sets the width at ending position of the range.
- BackgroundColor – sets the background color of the range.
- Border – sets the height and width of the border of the range.
- Placement – sets the position of the range.
- DistanceFromScale – sets the distance between the range and scale.

Positioning the range could be set either through Placement or DistanceFromScale property.

Note: By default, placement takes the value “Near”, whereas other enumeration values available are “Far” and “Center”.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
scales.ShowRanges(true).Ranges(ranges => {
ranges.StartValue(20).EndValue(50).StartWidth(2).EndWidth(6).Border(borde
r =>
border.Color("red").Width(2)).BackgroundColor("yellow").DistanceFromScale
(20).Add();
ranges.StartValue(50).EndValue(100).StartWidth(2).EndWidth(7).Border(bord
er =>
border.Color("green").Width(2)).BackgroundColor("blue").Placement(RangePl
acement.Near).Add();
}).Add();
})
```



Note: On setting both the position properties - "DistanceFromScale" and "Placement" for a range, the value set in "DistanceFromScale" is given preference.

Multiple Ranges

Multiple ranges can be added in Ranges to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.ShowRanges(true).Ranges(ranges => {
ranges.StartValue(0).EndValue(10).BackgroundColor("green").DistanceFromSc
ale(-5).Add();
ranges.StartValue(10).EndValue(30).BackgroundColor("yellow").DistanceFrom
Scale(-5).Add();
ranges.StartValue(30).EndValue(50).BackgroundColor("red").DistanceFromSca
le(-5).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Layout

Row-wise Layout

Gauges can be arranged in specified number of rows by using the RowsCount property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").RowsCount(2)
```



Column-wise Layout

Gauges can be arranged in specified number of columns by using the ColumnsCount property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").ColumnsCount(2)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Labels

Adding Label Collection

Label collection can be directly added to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Labels(labels => { labels.Angle(20).Add(); }).Add();
})
```

Appearance Customization

The appearance of the Label can be customized through the following properties.

- **Angle** – used to display labels in a rotated manner. By default, the value is 0.
- **Color** – displays the label in specified color
- **Opacity** – sets the opacity of the label. By default, the value is 1.
- **Type** – indicates the label for major intervals or minor intervals. By default, it takes major intervals.
- **IncludeFirstValue** – includes the initial value based on user requirement. By default, the value is “true”.
- **Font** – sets the font size, font style and font family of the label.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Labels(labels =>
{
//for Major labels
labels.Type(GaugeTypes.Major).Color("#1AFF01").Opacity(80).IncludeFirstVa
lue(false).Font(font =>
font.Size("15px").FontFamily("Arial").FontStyle("Bold")).Add();
//for Minor labels
labels.Type(GaugeTypes.Minor).Color("#FF103F").Opacity(80).IncludeFirstVa
lue(true).Font(font =>
font.Size("10px").FontFamily("Arial").FontStyle("Normal")).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Unit Text

The UnitText property is used to add some text along with the labels. Normally, we indicate the unit/measurement of the numeric value through unit text. Using the UnitTextPosition property, the text can be positioned either in front or back.

Note: By default, text appears at the back.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.Labels(labels => {
//for Major labels
labels.Type(GaugeTypes.Minor).UnitText("$").UnitTextPosition(UnitTextPlac
ement.Front).Add();
//for Minor labels
labels.Type(GaugeTypes.Minor).UnitText("$").UnitTextPosition(UnitTextPlac
ement.Front).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Custom labels

Adding Custom Label Collection

You can apply custom custom label Collection by using `CustomLabels` which can be directly added to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.CustomLabels(customLabels => { customLabels.Position(position =>
position.X(180).Y(290)).Add(); }).Add();
})
```

Appearance Customization

The appearance of the custom labels can be customized through the following properties.

- **Position** – used to set the position of the labels.
- **Font** – sets the font size, font style and font family of the label text.
- **Color** – sets the color of the label text.
- **TextAngle** – rotates the label to a specified angle. By default, the value is 0.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.CustomLabels(customLabels =>
{
customLabels.Position(position => position.X(180).Y(320)).Font(font =>
font.Size("12px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("blue").TextAngle(20).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple Custom Labels

Multiple ranges can be added in CustomLabels to the scales option within the PivotGauge control.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Scales(scales => {
scales.CustomLabels(customLabels =>
{
customLabels.Position(position => position.X(180).Y(150)).Font(font =>
font.Size("12px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("red").Add();
customLabels.Position(position => position.X(180).Y(320)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("green").Add();
customLabels.Position(position => position.X(180).Y(290)).Font(font =>
font.Size("10px").FontFamily("Segoe
UI").FontStyle("Normal")).Color("blue").Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Label Format

You can customize the format of labels displayed in the PivotGauge control using LabelFormatSettings property.

Following are the formats that can be applied to labels in PivotGauge:

- NumberFormats - Allows the user to change the number format of the label values in PivotGauge.
- DecimalPlaces - Allows you to set the number of digits displayed after decimal point.
- PrefixText - Allows you to add a text at the beginning of the label.
- SuffixText - Allows you to add text at the end of the label.

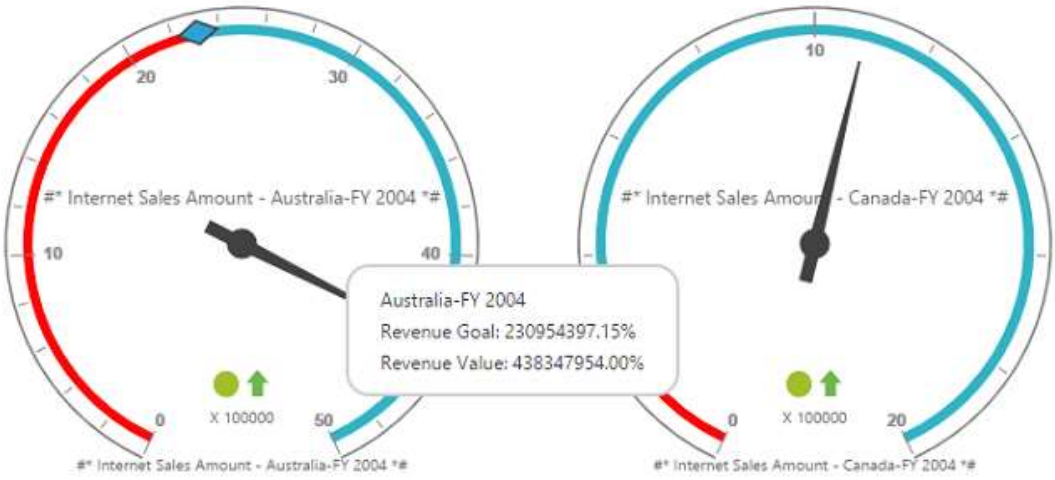
Number format for the label values can be set to any of the following type:

- Default
- Currency
- Percentage
- Fraction
- Scientific
- Text
- Notation

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").LabelFormatSettings(labelFor
matSettings=>
labelFormatSettings.NumberFormats(NumberFormat.Percentage).DecimalPlaces(
2).PrefixText("#*").SuffixText("*#"))
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip

Tooltip can be enabled by using the `EnableTooltip` property.

Note: By default, this property is set to "false".

ASPX-CS

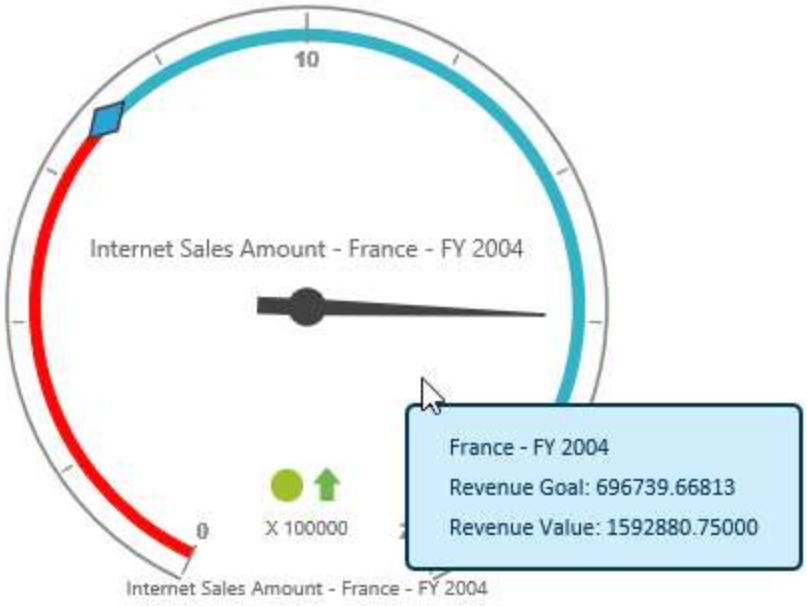
```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").EnableTooltip(true)
```

Tooltip appearance can be customized by overriding its CSS class.

CSS

```
.e-pivotgauge-tooltip {
background-color: #D2E9FE!important;
border: 2px solid #01465C!important;
color: #01232E!important;
border-radius: 5px!important;
margin-top: 20px;
text-align: left;
font: 12 px Segoe UI;
line-height: 20px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Animation

The animation option makes the pointer to rotate from minimum value to actual value with animation effects. The animation can be enabled/disabled by using the EnableAnimation property. Speed of the pointer can be controlled by using the AnimationSpeed property. Time is represented in milliseconds.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").EnableAnimation(true).Animat
ionSpeed(1000)
```

Localization

Localization in PivotGauge Control

We can localize the PivotGauge control texts with a collection of localized strings using **"ej.PivotGauge.Locale"** for different cultures.

Note: By default, the PivotGauge control is localized in **"en-US"**.

Following code example illustrates on how to localize PivotGauge based on **"French"** culture.

ASPX-CS

```
<script>
ej.PivotGauge.Locale["fr-FR"] = {
RevenueGoal: "Objectif de chiffre d'affaires",
RevenueValue: "Valeur du chiffre d'affaires"
}
</script>
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Locale("fr-FR")
```

Following table localizes the in-built keywords to **"French"** culture for PivotGauge.

Copyright © 2001 - 2020 Syncfusion Inc.

Keywords	Values
RevenueGoal	"Objectif de chiffre d'affaires"
RevenueValue	"Valeur du chiffre d'affaires "

Localization and Globalization of Cube Info

Content displayed within the PivotGauge control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

To get the localized string based on different cultures, from OLAP Cube, we need to set **"Locale Identifier"** in the connection string to a specific culture. The attribute is set for PivotGauge in Client Mode as shown below

ASPX-CS

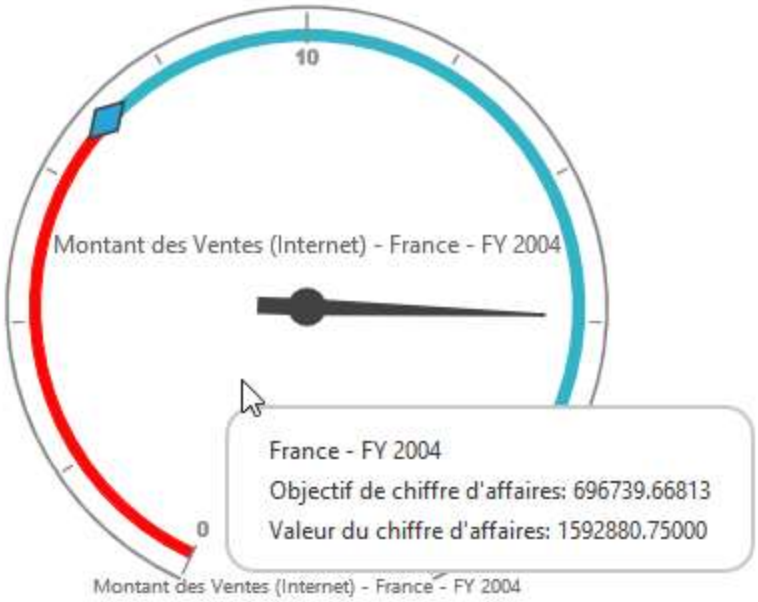
```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").Locale("fr-
FR").DataSource(dataSource =>
dataSource.Data("http://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1036;"))
```

For Server Mode, we need to set **"Culture"** and **"OverrideDefaultFormatStrings"** properties in "OlapDataManager" class to a specific culture along with setting **"Locale Identifier"** in connection string.

C#

```
//1036 refers to "fr-FR" culture.
string connectionString = "Data Source=localhost; Initial
Catalog=Adventure Works DW; Locale Identifier=1036;";
DataManager = new OlapDataManager(connectionString);
DataManager.Culture = new System.Globalization.CultureInfo(1036);
DataManager.OverrideDefaultFormatStrings = true;
```

Copyright © 2001 - 2020 Syncfusion Inc.

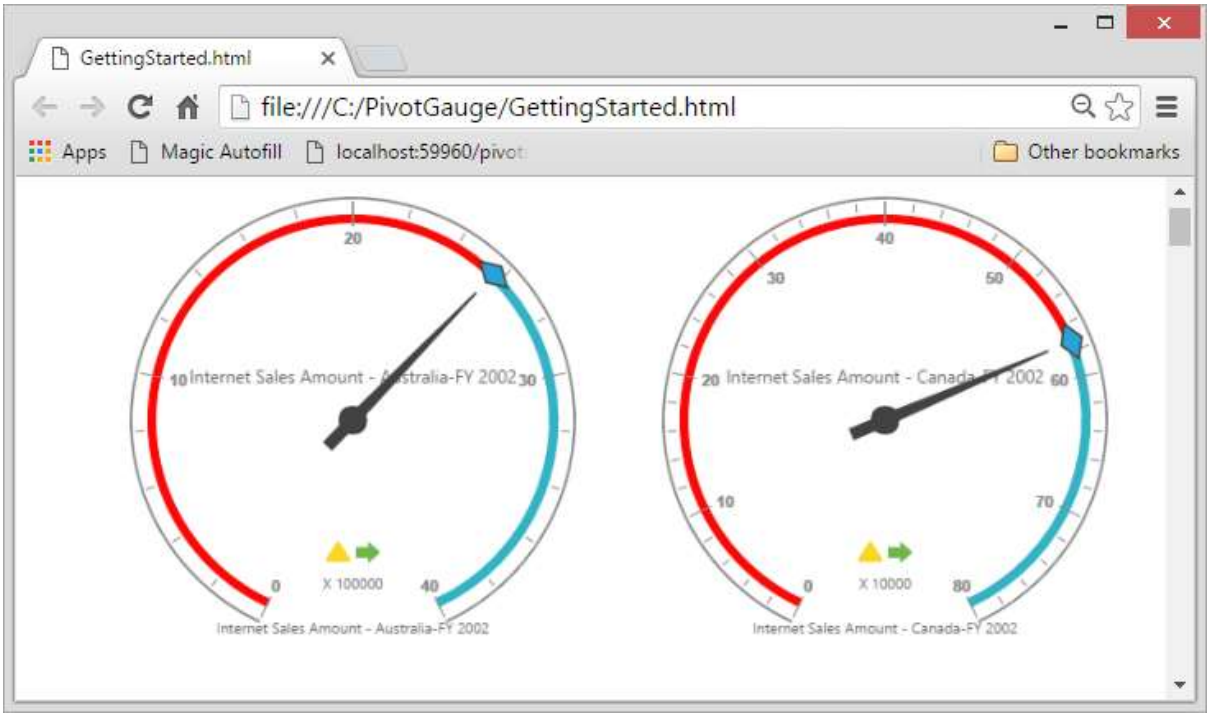


Responsive Layout

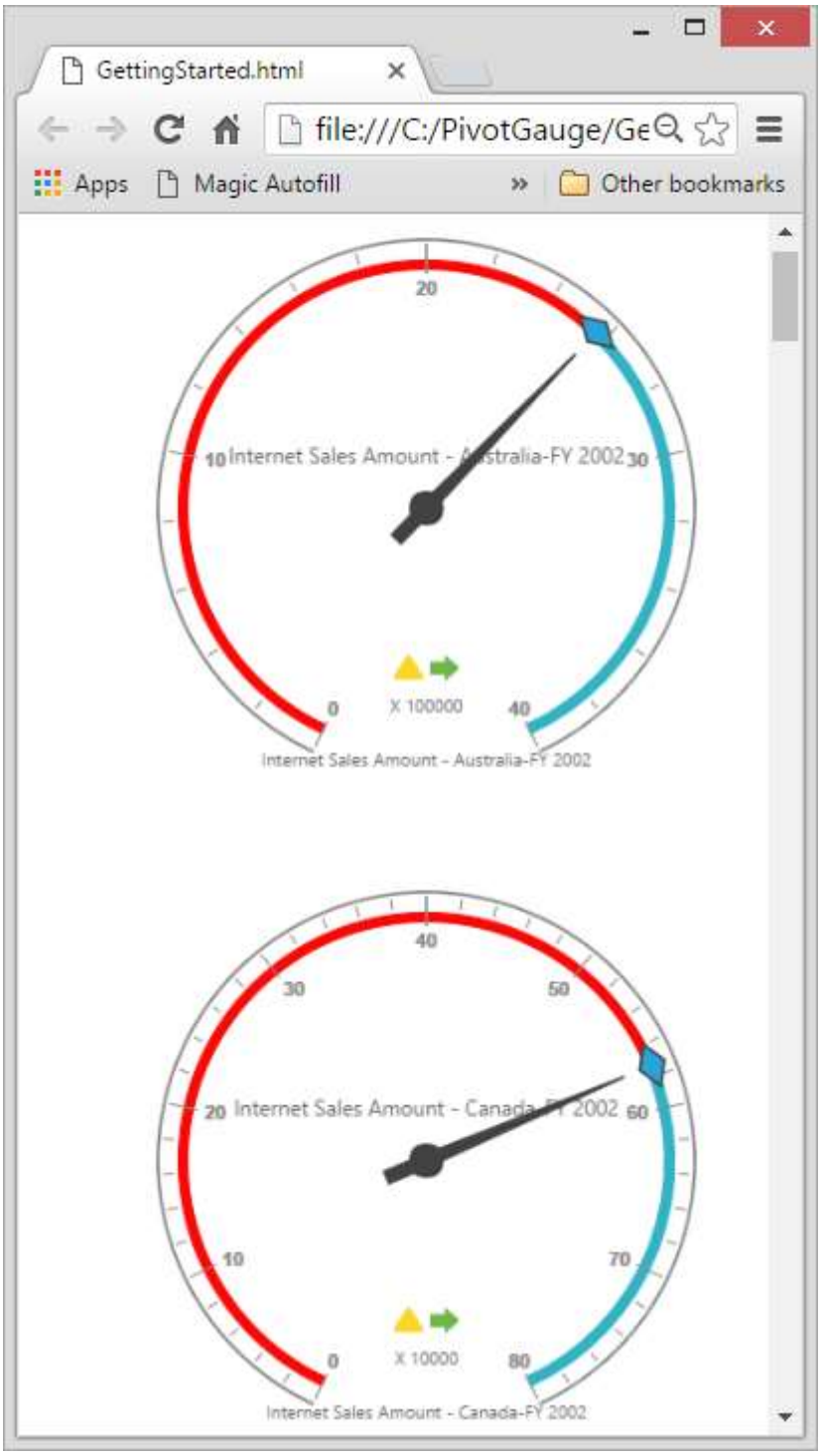
PivotGauge control supports responsive rendering based on the target device (desktop and tablet) resolution. It supports resolution upto 1024x600. You can enable responsiveness in PivotGauge by setting `IsResponsive` property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotGauge("PivotGauge1").IsResponsive(true)
```



Normal View



Responsive View

How to

Configure pivot gauge through the model properties

There comes a time when you want to set pivot gauge properties other than using PivotGaugePropertiesBuilder i.e., in the Controller side. At that time, you can pass PivotGaugeProperties model instance as a parameter to the pivot gauge helper method.

We can set properties to pivot gauge control in server-side using the PivotGaugeProperties class and it is used in the view page with the help of pivot gauge helper overload.

In addition to the id parameter of pivot gauge helper method, we can also pass PivotGaugeProperties model as another parameter to the pivot gauge helper method.

Relational

The following code example explains how to render the pivot gauge control in relational mode.

RAZOR

```
@model Syncfusion.JavaScript.Models.PivotGaugeProperties
@(Html.EJ().Pivot().PivotGauge("PivotGauge", Model))
<script type="text/javascript">
function load(args) {
args.model.dataSource.data = pivot_dataset;
}
</script>
```

C#

```
namespace PivotGauge
{
public class PivotGaugeController: Controller
{
public ActionResult PivotGaugeFeatures()
{
Syncfusion.JavaScript.Models.PivotGaugeProperties pivotgauge = new
Syncfusion.JavaScript.Models.PivotGaugeProperties();
PivotDataSource pivotgaugeDS = new PivotDataSource();
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "Country", FieldCaption = "Country"
});
rows.Add(new Field() { FieldName = "State", FieldCaption = "State" });
pivotgaugeDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "Product", FieldCaption = "Product"
});
pivotgaugeDS.Columns = cols;
pivotgauge.Load = "load";
List<Field> values = new List<Field>();
values.Add(new Field() { FieldName = "Amount" });
pivotgaugeDS.Values = values;
pivotgauge.DataSource = pivotgaugeDS;
return View(pivotgauge);
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

As a result of the previous code example, the pivot gauge will be displayed as shown below:



OLAP

The following code example explains how to render the pivot gauge control in OLAP mode.

RAZOR

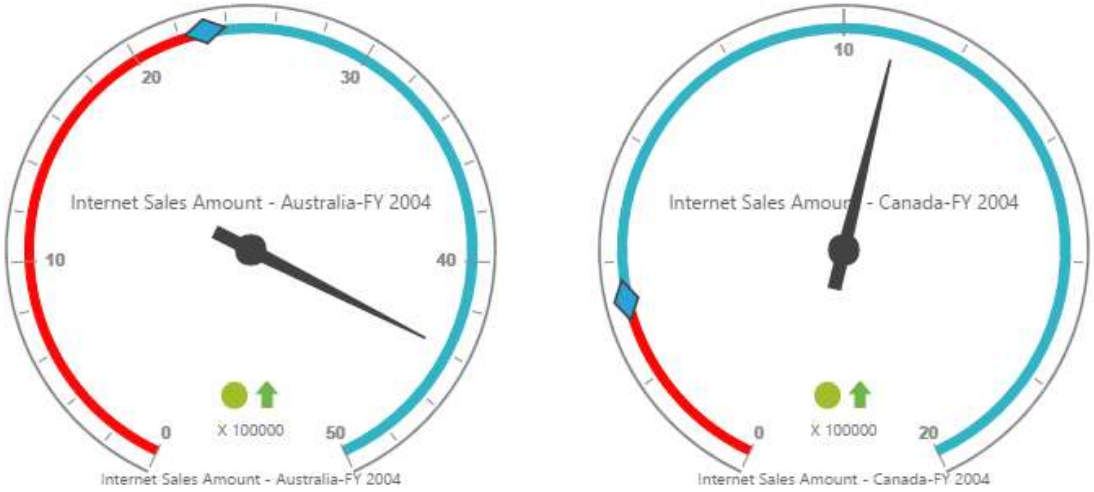
```
@model Syncfusion.JavaScript.Models.PivotGaugeProperties
@(Html.EJ().Pivot().PivotGauge("PivotGauge", Model))
```

C#

```
namespace PivotGauge
{
public class PivotGaugeController: Controller
{
public ActionResult PivotGaugeFeatures()
{
Syncfusion.JavaScript.Models.PivotGaugeProperties pivotgauge = new
Syncfusion.JavaScript.Models.PivotGaugeProperties();
PivotDataSource pivotgaugeDS = new PivotDataSource();
pivotgaugeDS.Data = "https://bi.syncfusion.com/olap/msmdpump.dll";
pivotgaugeDS.Cube = "Adventure Works";
pivotgaugeDS.Catalog = "Adventure Works DW 2008 SE";
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "[Date].[Fiscal]" });
pivotgaugeDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "[Customer].[Customer Geography]" });
pivotgaugeDS.Rows = cols;
List<MeasuresItems> measures = new List<MeasuresItems>();
```

```
measures.Add(new MeasuresItems() { FieldName = "[Measures].[Internet
Sales Amount]" });
List<Field> values = new List<Field>();
values.Add(new Field() { Measures = measures, Axis =
Syncfusion.JavaScript.AxisName.Column });
pivotgaugeDS.Values = values;
pivotgauge.DataSource = pivotgaugeDS;
return View(pivotgauge);
}
}
}
```

As a result of the previous code example, the pivot gauge will be displayed as shown below:



PivotTreeMap

Overview

The PivotTreeMap control lets the user to visualize OLAP data in the form of nested nodes in hierarchical order with the ability to drill up and down.

Key Features

- **Data Source** - Binds the PivotTreeMap control with the Microsoft SQL Server Analysis Services (SSAS) and XML/A data sources.
- **Drill Support** - Enables you to navigate through the inner levels of a hierarchy elements.
- **Color Mapping** - Allows user to differentiate leaf nodes using various color codes either based on their value or members.
- **Legend** - Differentiates the color code based on value range (from minimum to maximum).

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotTreeMap and OLAP datasource (Client Mode)

This section covers the information that you need to know to populate a simple PivotTreeMap with OLAP data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Empty Web Application** by using Visual Studio IDE and name the project as **"PivotTreeMapDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
```

```
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS References

The scripts and style sheets that are mandatorily required to render PivotTreeMap control in a MVC Web Application which are highlighted below in an appropriate order.

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js
4. jsrender.min.js

Scripts and style sheets are referred under the <head> tag in _Layout.cshtml file which is found inside Views > Shared folder.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotTreeMap

Before initializing, empty the contents of "Index.cshtml" file under Views > Home folder and add the following codes.

HTML

```
@using Syncfusion.JavaScript;
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1")
```

Populate PivotTreeMap with DataSource

Initializes the OLAP datasource for PivotTreeMap control as shown below.

HTML

```
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1").IsResponsive(true).DataS
ource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();})
.Data("http://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works"))
<!--Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
```

The above code will generate a simple PivotTreeMap with "Internet Sales Amount" over a period of fiscal years across different customer geographic locations.



Creating a simple application with PivotTreeMap and OLAP datasource (Server Mode)

This section covers the information required to create a simple PivotTreeMap bound to OLAP datasource.

Note: We will be illustrating this section by creating a simple Web Application through Visual Studio IDE since PivotTreeMap can also act as a server-side control with .NET dependency. ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Empty Web Application** by using Visual Studio IDE and name the project as **"PivotTreeMapDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.EJ
- Syncfusion.EJ.MVC
- Syncfusion.EJ.Pivot

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, 14.2 version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referenced assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Linq.Base, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<add assembly="Syncfusion.Olap.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Compression.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.PivotAnalysis.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render a PivotTreeMap control in a Web Application are mentioned in an appropriate order.

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js
4. jsrender.min.js

Click here to know more about script and style sheets available online (CDN Link).

Copyright © 2001 - 2020 Syncfusion Inc.

Scripts and style sheets are referred under the **head** tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/js/assets/external/jsrender.min.js"
type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

HTML

```
@using Syncfusion.JavaScript;
<div>
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1").Url(Url.Content("/Olap")
)
</div>
<!--Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The **"Url"** property in PivotTreeMap control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotTreeMap control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Olap". If WCF service is used as endpoint, the URL would look like "/OlapService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in your existing Web Application, right-click on the project in Solution Explorer and select **Add > New Item.** In the **Add New Item** window, select **WebAPI Controller Class** and name it as "OlapController.cs", click **Add.**

Now WebAPI controller is added into your application successfully which in-turn comprise of the following file. The utilization of this file will be explained in the following sections.

- OlapController.cs

Note: While adding WebAPI Controller Class, name it with the suffix "Controller" that is mandatory. For example, in demo the controller is named as "OlapController".

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside OlapController.cs file.

C#

```
namespace PivotTreeMapDemo
{
public class OlapController : ApiController
{
}
}
```

Adding the List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapController.cs file.

C#

```
using System.Web;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using System.Web.Script.Serialization;
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
namespace PivotTreeMapDemo
{
public class OlapController : ApiController
{
```

```
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube and PivotTreeMap instances are created immediately inside the main class in OlapController.cs file.

C#

```
namespace PivotTreeMapDemo
{
public class OlapController : ApiController
{
PivotTreeMap htmlHelper = new PivotTreeMap();
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
//Other codes
}
}
```

Service methods in WebAPI Controller

Now you need to define the service methods inside "OlapController" class, found inside OlapController.cs file, created while adding WebAPI Controller Class to your Web Application.

C#

```
namespace PivotTreeMapDemo
{
public class OlapController : ApiController
{
PivotTreeMap htmlHelper = new PivotTreeMap();
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
[System.Web.Http.ActionName("InitializeTreeMap")]
[System.Web.Http.HttpPost]
public Dictionary< string, object > InitializeTreeMap(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = null;
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager);
}
[System.Web.Http.ActionName("DrillTreeMap")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillTreeMap(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager = new OlapDataManager(connectionString);
```

```
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["olapReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["drillInfo"].ToString());
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.Name = "Default Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Date";
dimensionElementColumn.AddLevel("Fiscal", "Fiscal Year");
MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name = "Customer
Count" });
DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Customer";
dimensionElementRow.AddLevel("Customer Geography", "Country");
DimensionElement dimensionElementRow1 = new DimensionElement();
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
return olapReport;
}
}
}
```

Configure routing in WebAPIConfig Class

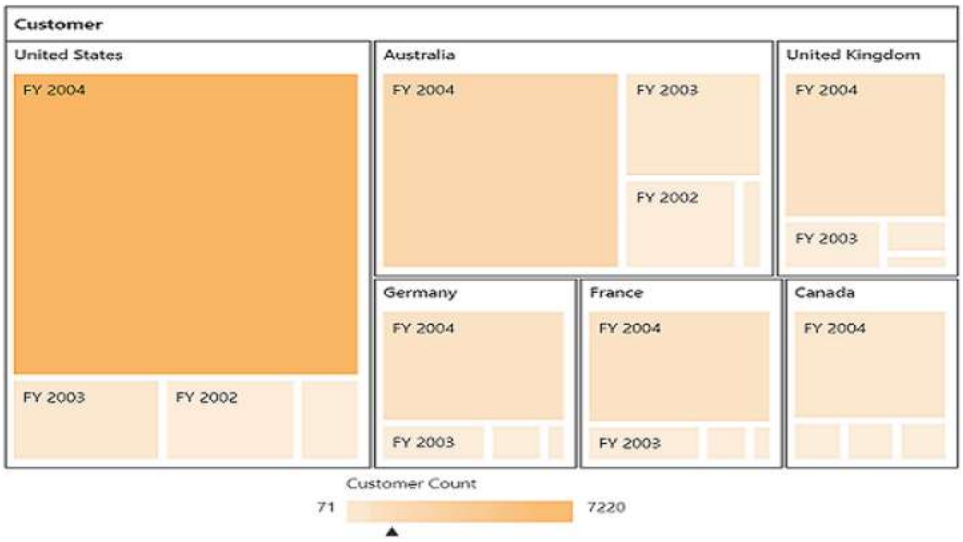
Open the WebAPIConfig.cs file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, PivotTreeMap is rendered with customer count over different customer geographic locations across a period of fiscal years.

Copyright © 2001 - 2020 Syncfusion Inc.



WCF

This section demonstrates the utilization of WCF service as endpoint binding OLAP datasource to a simple PivotTreeMap. For more details on this topic, click here.

DataBinding

Binding PivotTreeMap to Offline Cube

To connect an OLAP Cube available in local machine, physical path of the Cube needs to be set in the connection string. Below code sample illustrates the same.

C#

```
string connectionString = @"DataSource = system
drive:\OfflineCube\Adventure_Works_Ext.cub; Provider = MSOLAP;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotTreeMap to Cube in local SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in local machine, server name and database name needs to be set in the connection string. If you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. Below code sample illustrates the same.

C#

```
string connectionString = "Data source=localhost; Initial
Catalog=Adventure Works DW;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotTreeMap to Cube in online SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in online server through **XML/A**, host server link and database name needs to be set in the connection string. If you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. Below code sample illustrates the same.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotTreeMap to Cube in online Mondrian Server

To connect an OLAP Cube available in Mondrian Server through **XML/A**, host server link and database name needs to be set in the connection string. If you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. Below code sample illustrates the same.

C#

```
string connectionString = @"Data Source =
http://localhost:8080/mondrian/xmla; Initial Catalog =FoodMart;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName =
Syncfusion.Olap.DataProvider.Providers.Mondrian;
```

Binding PivotTreeMap to Cube in online ActivePivot Server

To connect an OLAP Cube available in ActivePivot Server through **XML/A**, host server link and database name needs to be set in the connection string. If you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. Below code sample illustrates the same.

C#

```
string connectionString = @"Data Source =
http://localhost:8080/cva_s/xmla; Initial Catalog = CVAS;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName=Syncfusion.Olap.DataProvider.Provid
ers.ActivePivot;
```

WCF

Adding a WCF Service

To add a WCF service in an existing Web application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as OlapService.svc, click **Add**.

Now WCF service is added into your application successfully which in-turn comprise of the following files. The utilization of these files will be explained in the immediate sections.

- OlapService.svc
- OlapService.svc.cs
- IOlapService.cs

Configuring WCF Service Class

Remove the **“DoWork”** method present inside both OlapService.svc.cs and IOlapService.cs files. Next, add **“AspNetCompatibilityRequirements”** attribute on top of main class present inside OlapService.svc.cs and set **“RequirementsMode”** value to **“Allowed”**.

C#

```
namespace PivotTreeMapDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- Syncfusion.EJ
- Syncfusion.EJ.MVC
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapService.svc.cs file.

C#

```
using System.Web;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Net;
using System.Web.Script.Serialization;
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
using System.ServiceModel.Activation;
namespace PivotTreeMaptDemo
{
```

```
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
}
}
```

Datasource Initialization

Now the connection string to connect OLAP Cube, PivotTreeMap, and "JavaScriptSerializer" instances are created immediately inside the main class in OlapService.svc.cs file.

C#

```
namespace PivotTreeMaptDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
PivotTreeMap htmlHelper = new PivotTreeMap();
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
JavaScriptSerializer serializer = new JavaScriptSerializer();
//Other codes
}
}
```

Service methods in WCF Service

First, declare the service methods inside **IOlapService** interface, found in IOlapService.cs file, created while adding WCF Service to the Application.

C#

```
namespace PivotTreeMaptDemo
{
[ServiceContract]
public interface IOlapService
{
[OperationContract]
Dictionary< string, object > InitializeTreeMap(string action, string
currentReport, string customObject);
[OperationContract]
Dictionary<string, object> DrillTreeMap(string action, string drillInfo,
string olapReport, string customObject);
}
}
```

Then, elaborate the service methods inside the main class, found in OlapService.svc.cs file.

C#

```
namespace PivotTreeMaptDemo
{
```

```
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService : IOlapService
{
PivotTreeMap htmlHelper = new PivotTreeMap();
string connectionString = "Data
Source=http://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
JavaScriptSerializer serializer = new JavaScriptSerializer();
//This method provides the required information from the server side when
initializing the PivotTreeMap.
public Dictionary< string, object > InitializeTreeMap(string action,
string currentReport, string customObject)
{
OlapDataManager DataManager = null;
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return htmlHelper.GetJsonData(action, DataManager);
}
public Dictionary<string, object> DrillTreeMap(string action, string
drillInfo, string olapReport, string customObject)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(olapRe
port));
return htmlHelper.GetJsonData(action, DataManager, drillInfo);
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.Name = "Default Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
//Specifying the Name for the Dimension Element
dimensionElementColumn.Name = "Date";
dimensionElementColumn.AddLevel("Fiscal", "Fiscal Year");
MeasureElements measureElementColumn = new MeasureElements();
//Specifying the Name for the Measure Element
measureElementColumn.Elements.Add(new MeasureElement { Name = "Customer
Count" });
DimensionElement dimensionElementRow = new DimensionElement();
//Specifying the Dimension Name
dimensionElementRow.Name = "Customer";
dimensionElementRow.AddLevel("Customer Geography", "Country");
///Adding Row Members
olapReport.SeriesElements.Add(dimensionElementRow);
///Adding Column Members
olapReport.CategoricalElements.Add(dimensionElementColumn);
///Adding Measure Element
olapReport.CategoricalElements.Add(measureElementColumn);
return olapReport;
}
}
}
```

Configuring Web Configuration File

Copyright © 2001 - 2020 Syncfusion Inc.

The services could be exposed through the properties, binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IOlapService contract and hence it is PivotTreeMapDemo.IOlapService.
- Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
- BehaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

XML

```
<system.serviceModel>
......
......
<services>
<service name="PivotTreeMapDemo.OlapService">
<endpoint address=""
behaviorConfiguration="PivotTreeMapDemo.OlapServiceAspNetAjaxBehavior"
binding="webHttpBinding" contract="PivotTreeMapDemo.IOlapService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotTreeMapDemo.OlapServiceAspNetAjaxBehavior">
<enableWebScript />
</behavior>
</endpointBehaviors>
</behaviors>
......
......
</system.serviceModel>
```

Note: In this example, **"PivotTreeMapDemo"** indicates the name and root namespace of the Application created in Visual Studio IDE and **"OlapService"** indicates the name of the WCF service created.

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
);
}
```

The above code will generate a simple PivotTreeMap showing Customer Count over different customer geographic locations across a period of fiscal years.

{% include image.html url="/js/PivotTreeMap/OLAP-Connectivity_images/olapwebapi.png" %}

Drill Operation

Drill down exposes the hierarchy by clicking on a node and this results in allowing the TreeMap to move to the next level or sub level. We can return back to the normal TreeMap view by clicking on the node header at the top.

Named Sets

Named Sets is a multidimensional expression (MDX) that returns a set of dimension members, which can be created by combining cube data, arithmetic operators, numbers, and functions.

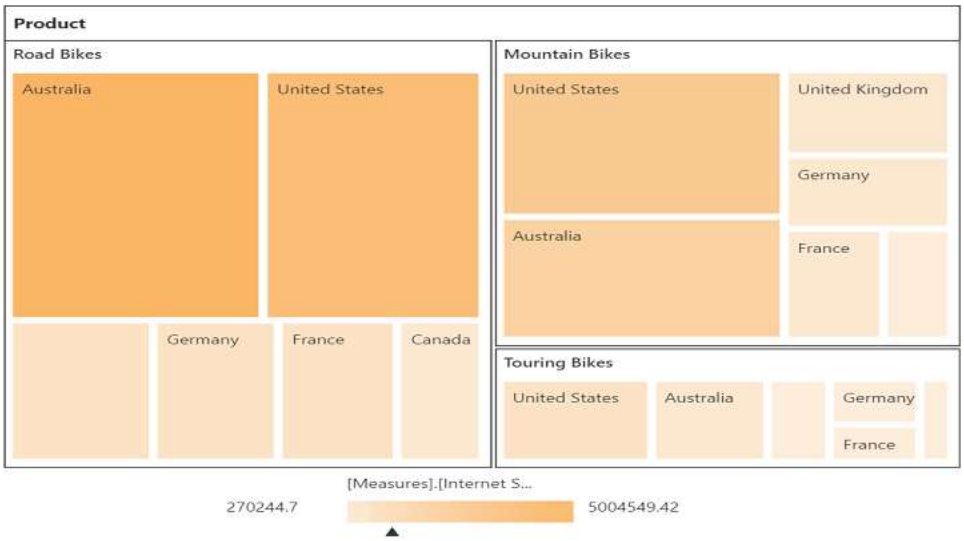
Client Mode

You can bind the Named Sets in PivotTreeMap by setting it's unique name in the `FieldName` property either in row or column axis and `IsNamedSets` Boolean property to "true".

JS

```
<!--Create a tag which acts as a container for PivotTreeMap-->
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Core Product
Group]").IsNamedSets(true).Add(); }).Columns(columns => {
columns.FieldName("[Customer].[Customer Geography]").Add();
}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add(); })
.Data("http://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works"))
<! --Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
```

```
</script>
```



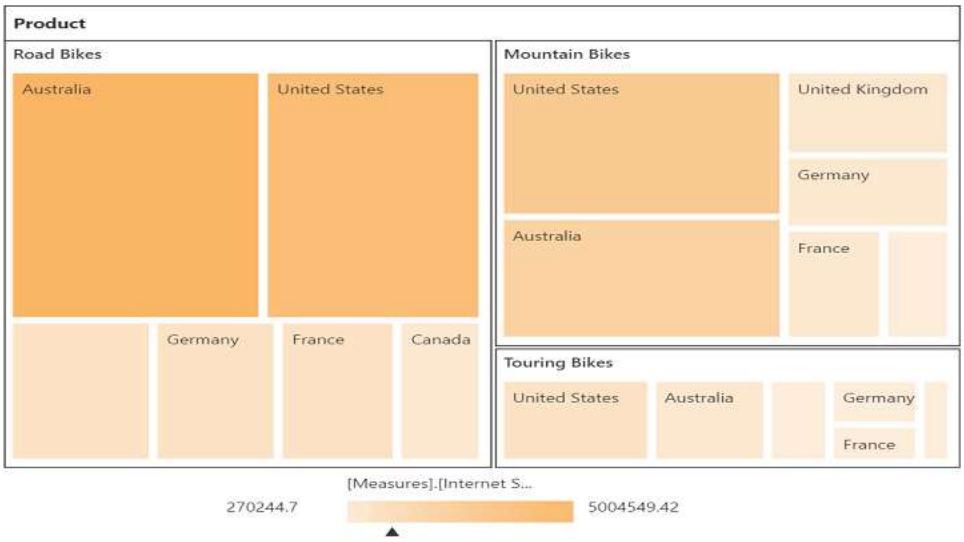
Server Mode

You can add Named Sets in the PivotTreeMap by using NamedSetElement class in the OlapReport.

C#

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementColumn = new DimensionElement();
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country ");
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement {
Name = "Internet Sales Amount"
});
NamedSetElement dimensionElementRow = new NamedSetElement();
dimensionElementRow.Name = "Core Product Group";
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(measureElementColumn);
olapReport.SeriesElements.Add(dimensionElementRow);
```

Copyright © 2001 - 2020 Syncfusion Inc.



Color Mapping

You can customize the colors of the leaf nodes of PivotTreeMap using the color mapping support.

Color mapping is categorized into two different types such as,

- Normal
- Range

You can color all the leaf nodes with the different colors by setting the color value of the rangeColorMapping property.

Normal Color Mapping

You can customize the nodes based on number of leaf items using different color ranges. You can also define the color value range using from and to properties.

The following code sample explains how to customize PivotTreeMap appearance using **Normal** mode (i.e., differentiate color based on number of the leaf items in each header).

JS

```
<! --Create a tag which acts as a container for PivotTreeMap-->
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Date].[Fiscal]").Add();
}).Columns(columns => { columns.FieldName("[Customer].[Customer
Geography]").Add(); }).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Customer Count]").Add();
}).Axis(AxisName.Column).Add();
}).Data("http://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works")).ClientSideEvents(oEve => {
oEve.RenderSuccess("onTypeChange"); })
<script type="text/javascript">
function onTypeChange(args) {
treemapTarget = $('#PivotTreeMap1TreeMapContainer').data("ejTreeMap");
treemapTarget.model.colorValuePath = "";
treemapTarget.model.enableGradient = false;
treemapTarget.model.showLegend = false;
treemapTarget.model.legendSettings.leftLabel = "";
```

```
treemapTarget.model.legendSettings.rightLabel = "";
treemapTarget.model.rangeColorMapping = [];
treemapTarget.model.colorValuePath = "Index";
treemapTarget.model.rangeColorMapping.push(
{ color: "#9de24f", from: "0", to: "0" },
{ color: "#a2e2fe", from: "1", to: "1" },
{ color: "#ffff66", from: "2", to: "2" },
{ color: "#FF0040", from: "3", to: "3" },
{ color: "#f6b53f", from: "4", to: "4" },
{ color: "#6FAAB0", from: "5", to: "5" },
{ color: "#C4C24A", from: "6", to: "6" }
)
treemapTarget.refresh();
}
</script>
<! --Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
```



Range Color Mapping

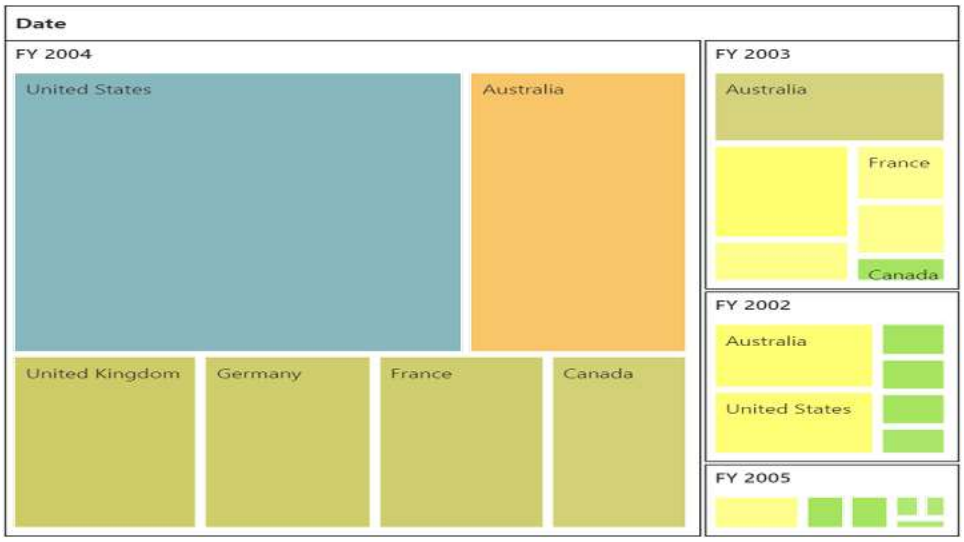
You can customize the nodes based on its value and color ranges using **Range** color. You can also define the color value range using from and to properties.

The following code sample explains how to customize PivotTreeMap appearance using **Range** mode (i.e., differentiate color for leaf items based on values).

JS

```
<script type="text/javascript">
function onTypeChange(args) {
```

```
treemapTarget = $('#PivotTreeMap1TreeMapContainer').data("ejTreeMap");
treemapTarget.model.colorValuePath = "";
treemapTarget.model.enableGradient = false;
treemapTarget.model.showLegend = false;
treemapTarget.model.legendSettings.leftLabel = "";
treemapTarget.model.legendSettings.rightLabel = "";
treemapTarget.model.rangeColorMapping = [];
treemapTarget.model.colorValuePath = "Value";
treemapTarget.model.rangeColorMapping.push(
{ color: "#a2e2fe", from: "0", to: "10" },
{ color: "#9de24f", from: "11", to: "250" },
{ color: "#ffff66", from: "251", to: "1000" },
{ color: "#C4C24A", from: "1001", to: "3000" },
{ color: "#f6b53f", from: "3001", to: "5000" },
{ color: "#6FAAB0", from: "5001", to: "10000" },
{ color: "#FF0040", from: "10001", to: "20000" }
)
treemapTarget.refresh();
}
</script>
```



Legend

Legend Visibility

The legend shows the value range differences and color occurrence in the respective leaf node while you hover it with the cursor.

Note: By default, the legend is visible in PivotTreeMap.

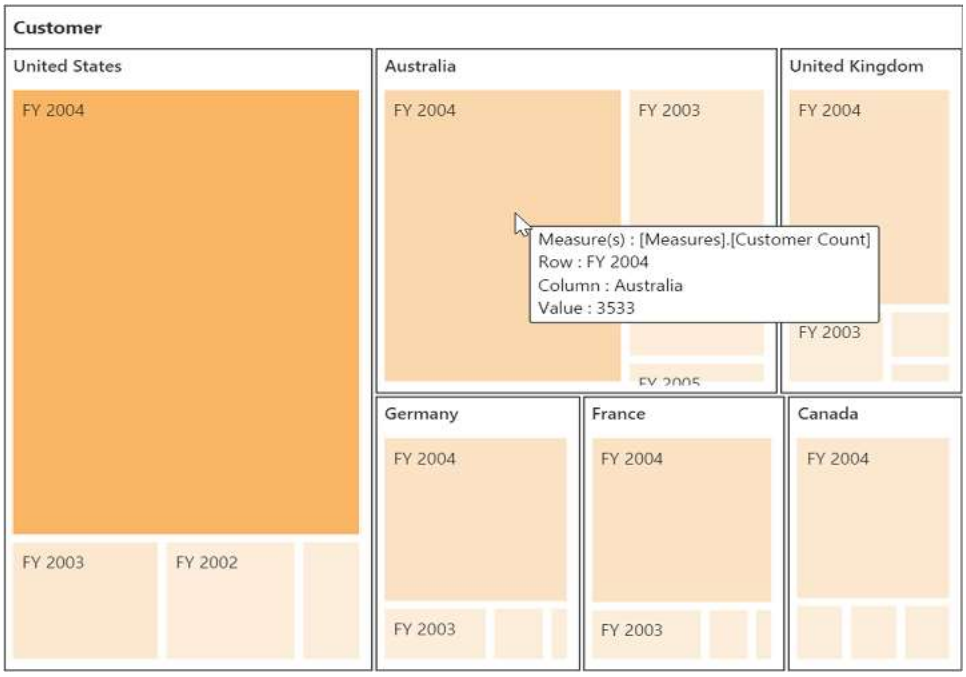
Copyright © 2001 - 2020 Syncfusion Inc.



You can disable the legend by setting the property **showLegend** as **false**. The following code example shows how to disable the legend.

JS

```
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Date].[Fiscal]").Add();
}).Columns(columns => { columns.FieldName("[Customer].[Customer
Geography]").Add(); }).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Customer Count]").Add();
}).Axis(AxisName.Column).Add();
}).Data("http://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works")).ClientSideEvents(oEve => {
oEve.RenderSuccess("showLegend"); })
<script type="text/javascript">
function showLegend(args) {
pivotTreeMap = $("#PivotTreeMap1TreeMapContainer").data("ejTreeMap");
pivotTreeMap.model.showLegend = false;
pivotTreeMap.refresh();
}
</script>
<! --Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
```



Localization

Localization and Globalization of Cube Info (Client Mode)

Content displayed within the PivotTreeMap control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

- To get localized data from OLAP Cube, we need to set **"Locale Identifier"** in the connection string to a specific culture in the **"data"** property present inside **"DataSource"**.

ASPX-CS

```
//1036 refers to "fr-FR" culture.
<!--Create a tag which acts as a container for PivotTreeMap-->
@Html.EJ().Pivot().PivotTreeMap("PivotTreeMap1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Customer].[Customer
Geography]").Add(); }).Columns(columns => {
columns.FieldName("[Date].[Fiscal]").Add(); }).Values(values => {
values.Measures(measures => { measures.FieldName("[Measures].[Customer
Count]").Add(); }).Axis(AxisName.Column).Add(); })
.Data("http://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1036;").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works")).IsResponsive(true)
<! --Tooltip labels can be localized here-->
<script id="tooltipTemplate" type="application/jsrender">
<div style="background:White; color:black; font-size:12px; font-
weight:normal; border: 1px solid #4D4D4D; white-space: nowrap;border-
radius: 2px; margin-right: 25px; min-width: 110px;padding-right: 5px;
padding-left: 5px; padding-bottom: 2px ;width: auto; height: auto;">
<div>Measure(s) : {{:~Measures(#data)}}</div><div>Row :
{{:~Row(#data)}}</div><div>Column : {{:~Column(#data)}}</div><div>Value :
{{:~Value(#data)}}</div>
</div>
</script>
```

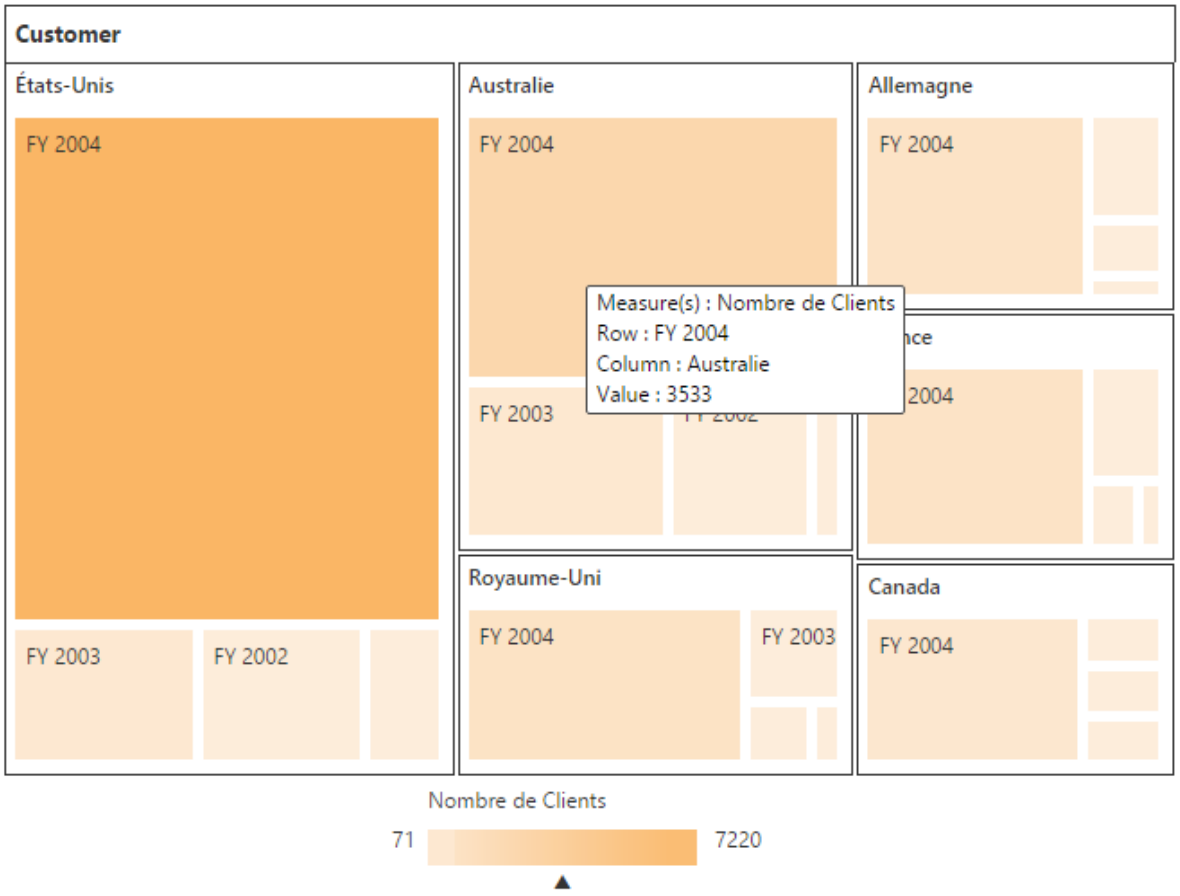
Localization and Globalization of Cube Info (Server Mode)

Content displayed within the PivotTreeMap control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

- To get the localized string based on different cultures, from OLAP Cube, we need to set **“Locale Identifier”** in the connection string to a specific culture.
- To bind the globalized content in PivotTreeMap control, we need to set **“Culture”** and **“OverrideDefaultFormatStrings”** properties in "OlapDataManager" class to a specific culture.

C#

```
//1036 refers to “fr-FR” culture.
string connectionString = "Data Source=localhost; Initial
Catalog=Adventure Works DW; Locale Identifier=1036;";
DataManager = new OlapDataManager(connectionString);
DataManager.Culture = new System.Globalization.CultureInfo(1036);
DataManager.OverrideDefaultFormatStrings = true;
```



How to

Configure pivot tree map through the model properties

There comes a time when you want to set pivot tree map properties other than using PivotTreeMapPropertiesBuilder i.e., in the Controller side. At that time, you can pass PivotTreeMapProperties model instance as a parameter to the pivot tree map helper method.

We can set properties to pivot tree map control in server-side using the PivotTreeMapProperties class and it is used in the view page with the help of pivot tree map helper overload.

In addition to the id parameter of pivot tree map helper method, we can also pass PivotTreeMapProperties model as another parameter to the pivot tree map helper method.

The following code example explains how to render the pivot tree map control in relational mode.

The following code example which will explain to render the pivot tree map control.

RAZOR

```
@model Syncfusion.JavaScript.Models.PivotTreeMapProperties
@(Html.EJ().Pivot().PivotTreeMap("PivotTreeMap", Model))
```

C#

```
namespace PivotTreeMap
{
public class PivotTreeMapController: Controller
{
public ActionResult PivotTreeMapFeatures()
{
Syncfusion.JavaScript.Models.PivotTreeMapProperties pivottreemap = new
Syncfusion.JavaScript.Models.PivotTreeMapProperties();
PivotDataSource pivottreemapDS = new PivotDataSource();
pivottreemapDS.Data = "https://bi.syncfusion.com/olap/msmdpump.dll";
pivottreemapDS.Cube = "Adventure Works";
pivottreemapDS.Catalog = "Adventure Works DW 2008 SE";
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "[Date].[Fiscal]" });
pivottreemapDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "[Customer].[Customer Geography]" });
pivottreemapDS.Columns = cols;
List<MeasuresItems> measures = new List<MeasuresItems>();
measures.Add(new MeasuresItems() { FieldName = "[Measures].[Internet
Sales Amount]" });
List<Field> values = new List<Field>();
values.Add(new Field() { Measures = measures, Axis =
Syncfusion.JavaScript.AxisName.Column });
pivottreemapDS.Values = values;
pivottreemap.DataSource = pivottreemapDS;
return View(pivottreemap);
}
}
}
```

As a result of the previous code example, the pivot tree map will be displayed as shown below:

Copyright © 2001 - 2020 Syncfusion Inc.



PdfViewer

Overview

The PDF viewer for ASP .NET MVC is a visualization component for viewing and printing the PDF documents in web pages. It is powered by HTML5 or JavaScript and provides various customizations.

Key Features

- Uploads the PDF document into the server as byte array or stream and displays the same in the client.
- Views password protected or encrypted PDF documents.
- Allows customization of the default toolbar and creation of your own toolbar.
- Provides magnification tools and view modes for better viewing experience.
- Supports built-in themes for enhancing appearance.
- Supports responsive rendering while resizing the control/window.
- Supports text selection and search.
- Supports text markup annotations.
- Includes handwritten signatures in the PDF document.
- Prints the PDF document with the changes made.
- Downloads the PDF document with the changes made.
- Supports localization to adopt to the machines culture and language.
- Compatible with all the modern browsers that provides HTML5/CSS3/JavaScript support.

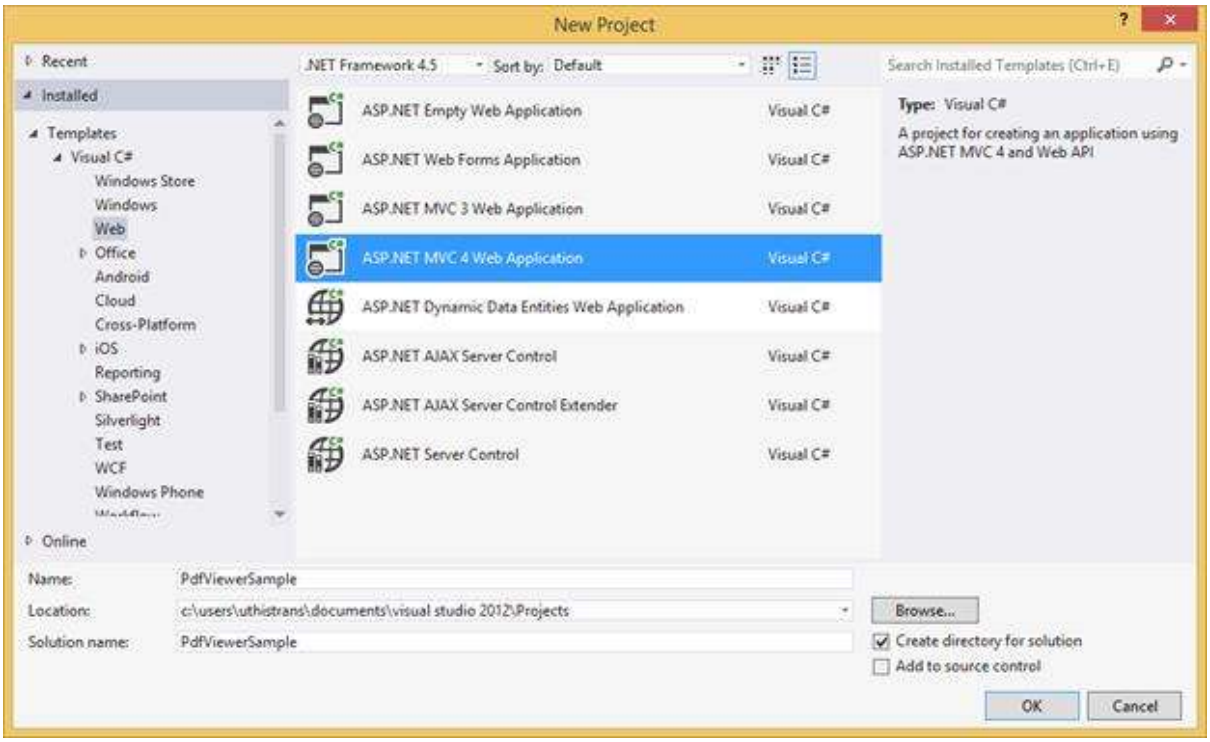
Getting Started

This section explains how to add and use a PDF viewer control in your web application with ASP.NET MVC.

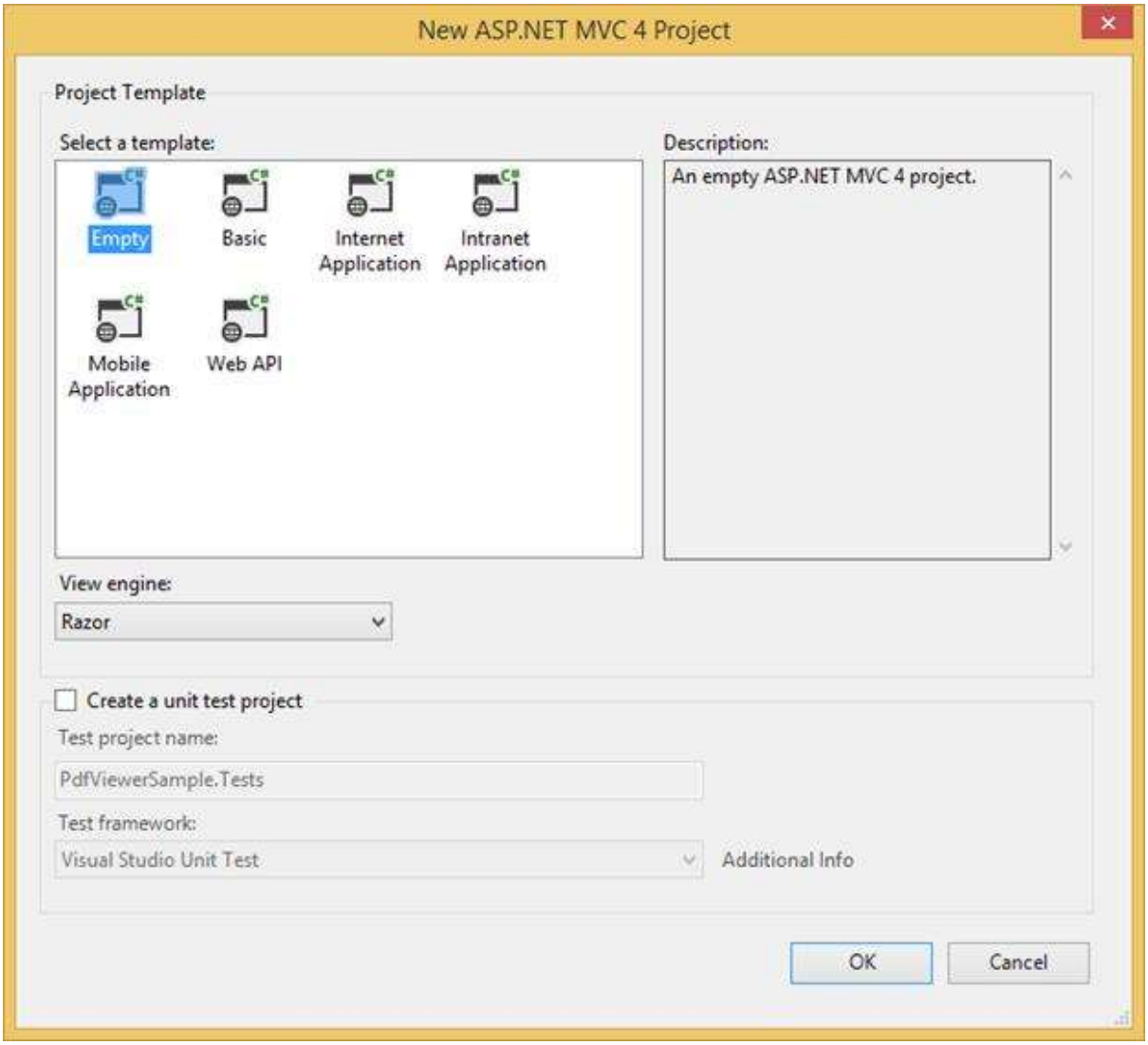
Create your first PDF viewer application in ASP.NET MVC

Open Visual Studio and create a new project by clicking New Project. Select the Web category and ASP.NET MVC4 Web Application template, and then click OK. The following screenshot displays the Project Creation Wizard in Visual Studio 2012:

Copyright © 2001 - 2020 Syncfusion Inc.

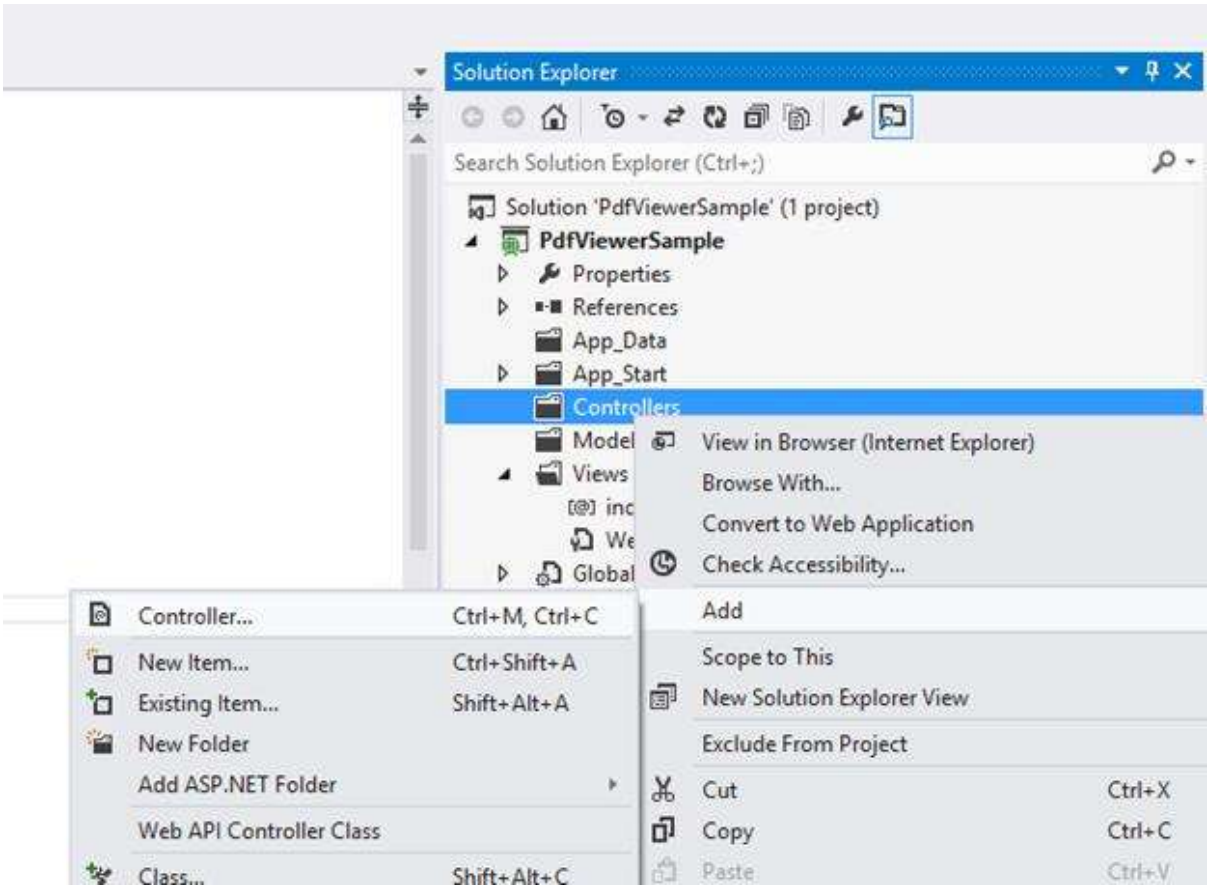


The following screenshot displays how to select the project template with razor view engine:

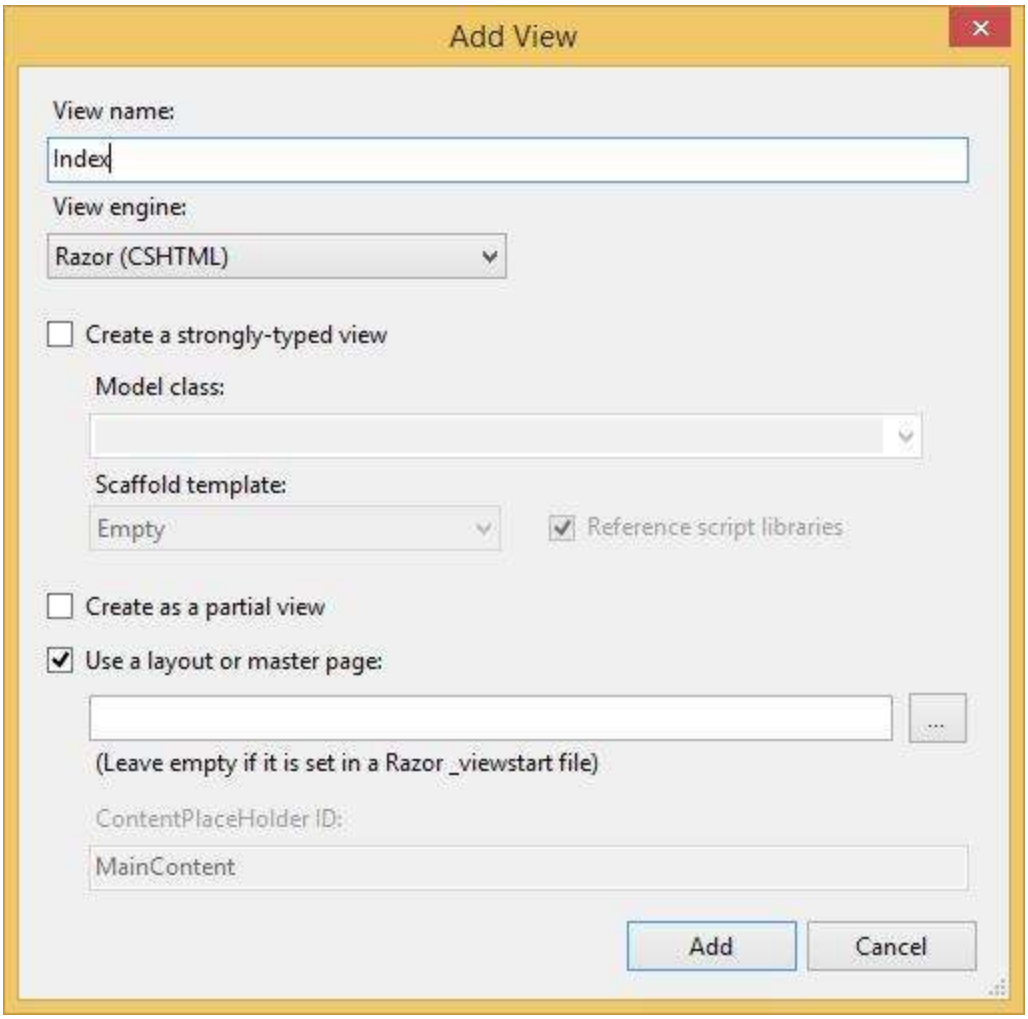


Add Controller and View page

Add new MVC controller in the Controllers folder and name it as **PdfSampleController.cs**



Create a new folder **PdfSample** in views and add new view **Index.cshtml** in it.



Modify RouteConfig.cs

Modify the routing to map to the **PdfSample** controller as follows.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
namespace PdfViewerSample
{
public class RouteConfig
{
public static void RegisterRoutes(RouteCollection routes)
{
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "PdfSample", action = "Index", id =
UrlParameter.Optional }
```

```
);
}
}
}
```

Modify WebApiConfig.cs

Modify the routing to map to the API controller of **PdfViewerSample** project as follows.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;
namespace PdfViewerSample
{
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "api/{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
}
```

Configuring Global.asax.cs

The WebApiConfig.cs and RouteConfig.cs must be configured in the Application_Start() method of Global.asax.cs

C#

```
protected void Application_Start()
{
AreaRegistration.RegisterAllAreas();
WebApiConfig.Register(GlobalConfiguration.Configuration);
FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
RouteConfig.RegisterRoutes(RouteTable.Routes);
BundleConfig.RegisterBundles(BundleTable.Bundles);
}
```

Add References, Scripts, Styles

Add References

Add reference to the following assemblies from NuGet package:

- System.Web.Http
- System. Web.Http.WebHost
- System.Net.Http.WebRequest
- System.Net.Http.Formatting

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The System.Web.Routing and System.Net.Http assemblies are also required, which are referred by default when creating the project.

Add the following references that are necessary for the PDF viewer control and set the Copy Local property to **True**

- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base
- Syncfusion.EJ
- Syncfusion.EJ.PdfViewer
- Syncfusion.EJ.MVC

Note: Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Please refer to this link to know about registering Syncfusion license key in your ASP.NET MVC application to use our components.

Add Scripts and Styles

Find the scripts for the PDF viewer control and style sheets in any of the following locations:

Local Disk: Click here to know more about script and style sheets installed in local machine.

CDN Link: Click here to know more about script and style sheets available online.

NuGet Package: Click here to know more about script and style sheets available in NuGet package.

Note: For getting started, you can use the ej.web.all.min.js file, which encapsulates all the ej widgets and frameworks in a single file. In production, it is highly recommended to use custom script generator to create custom script file with required widgets and its dependencies to reduce the size of the script file.

Add the script files and CSS files in the <title> tag of View page.

HTML

```
<link href="https://cdn.syncfusion.com/16.1.0.24/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://cdn.syncfusion.com/js/assets/external/jquery-
3.1.1.min.js"></script>
<script
src="https://cdn.syncfusion.com/16.1.0.24/js/web/ej.web.all.min.js"></scr
ipt>
```

Note: If the *Layout.cshtml is available, refer to the script files and CSS files in the* Layout.cshtml.

Modify Web.config

To render the PDF viewer in unobtrusive mode, refer to the **ej.unobtrusive.js** script file. Otherwise set **UnobtrusiveJavaScriptEnabled** to false in Web.config file. Add the following namespace in Web.config of Views.

XML

```
<add namespace="Syncfusion.EJ"/>
<add namespace="Syncfusion.MVC.EJ"/>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Add Control in View page

Initializing the PDF viewer control

1.Setting ServiceUrl and PdfService

To initialize the PDF viewer control, use the ServiceUr property that is necessary to access the service from which the PDF document is loaded and processed for the control. The PdfService property can also be used to specify the location of the supporting PDF service.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").PdfService("Local"))
```

2.Provide Web API action methods

The Web API action methods must be available in the server (Web API controller) to load and process the PDF documents. The available API methods in the controller are Load, FileUpload, Download, and Unload. However, we can modify the name of the server action methods as used in the controller using the ServerActionSettings property in the PDF viewer control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ServerActionSettings(s=>s.Load("LoadAction").FileUpload("FileUploadAction").Download("DownloadAction")))
```

3.Load PDF documents from client side

The PDF documents can be loaded in the PDF viewer control using the load() method in the client side. The path of the PDF document and the base64 string of the document can be used to load the PDF document.

Note: If the name of the PDF document is only passed as parameter in the load() method, the PDF document must be available in the folder which is specified in the Load action method in the controller.

JAVASCRIPT

```
var pdfviewer=$("#pdfviewer").data("ejPdfViewer");
pdfviewer.load("HTTP Succinctly");
```

When the PDF document is loaded in the PDF viewer control, the documentLoad event will be triggered. We can define the event method using the DocumentLoaded property of the control.

HTML

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.DocumentLoaded("documentLoaded")))
<script type="text/javascript">
function documentLoaded(args) {
alert("The document" +args.fileName + "is ready to view");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

4.Load the documents during control initialization

The PDF document can also be loaded during the PDF viewer control initialization using the DocumentPath property. The path of the PDF document and the base64 string of the document can be used to load the PDF document.

Note: If the name of the PDF document is only set in the DocumentPath property, the PDF document must be available in the folder which is specified in the Load action method in the controller

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").DocumentPath("HTTP Succinctly"))
```

5.Unload the documents from PDF viewer control

The PDF document loaded in the PDF viewer control can be unloaded using the unload() method in the client side.

JAVASCRIPT

```
var pdfviewer=$("#pdfviewer").data("ejPdfViewer");
pdfviewer.unload();
```

When the PDF document is unloaded from the PDF viewer control, the documentUnload event will be triggered. You can define the event method using the DocumentUnloaded property of the control.

HTML

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.DocumentUnloaded("documentUnloaded")))
<script type="text/javascript">
function documentUnloaded(args) {
alert("The document is unloaded from the PDF viewer control");
}
</script>
```

6.Events available in the PDF viewer control

Destroy

When the PDF viewer control is destroyed, the Destroy event will be triggered. You can define the event method using the Destroy property of the control.

HTML

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.Destroy("destroyed")))
<script type="text/javascript">
function destroyed (args) {
alert("The PDF viewer control is destroyed.");
}
</script>
```

AjaxRequestFailure

Copyright © 2001 - 2020 Syncfusion Inc.

When the AJAX requests from the client to the API controller is failed, ajaxRequestFailure event will be triggered. You can define the event method using the AjaxRequestFailure property of the control.

HTML

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.AjaxRequestFailure("ajaxFailureRequest")))
<script type="text/javascript">
function ajaxFailureRequest (args) {
alert("The AJAX request to the server is failed.");
}
</script>
```

HyperlinkClicked

When the hyperlinks available in the PDF document is clicked, the hyperlinkClick event will be triggered. You can define the event method using the HyperlinkClicked property of the control

HTML

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.HyperlinkClicked("hyperlinkClick")))
<script type="text/javascript">
function hyperlinkClick (args) {
alert("The hyperlink in the PDF document is clicked.");
}
</script>
```

Updating the PDF Viewer size based on the parent element

updateViewerSize()

The PDF Viewer is resized based on the parent element size by using the updateViewerSize() method.

Note: PDF Viewer will be resized automatically when the browser window is resized, and if the isResponsive property is set to true.

JAVASCRIPT

```
var pdfviewer=$("#pdfviewer").data("ejPdfViewer");
pdfviewer.updateViewerSize();
```

Encrypting and decrypting the JSON data sent from server

setJSONData(jsonData)

In server-side

In the Web API controller, users can add and encrypt their data along with the PDF document page content obtained from ProcessPdf() and send it to the client-side for rendering in PDF Viewer.

Note: Data should be returned only as the serialized string (JsonConvert.SerializeObject)

C#

```
string data = "Encrypted Data";
//Create the JSON object
var json = new Object();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Add additional/encrypted data to the JSON data
json = new { pdfData = helper.ProcessPdf(jsonResult), additionalData =
data, };
//Serialize the JSON data
string output = JsonConvert.SerializeObject(json);
//Return to the client
return output;
```

In client-side

In the client-side ajaxRequestSuccess event, you can get the encrypted responseData as argument. It must be decrypted and the original JSON data should be send to the setJSONData() for rendering the PDF document in PDF Viewer.

JAVASCRIPT

```
$(function () {
$("#viewer").ejPdfViewer({ serviceUrl: '../api/PdfViewer',
ajaxRequestSuccess: " onAjaxRequestSuccess " });
});
function onAjaxRequestSuccess(args) {
//Create PDF Viewer object
var pdfviewerObj = $("#viewer").data("ejPdfViewer");
//Parse JSON data to get the original data
var jsData = JSON.parse(args.responseData);
//Send the decrypted data for further rendering
pdfviewerObj.setJSONData(jsData["pdfData"]);
}
```

Note: Users can use their own encryption and decryption algorithm. The JSON data format should be formed as mentioned here

Displaying PDF document using Remote service

Add the following code snippet to Index.cshtml. Here, the PDF viewer uses hosted service in the remote machine to process the PDF.

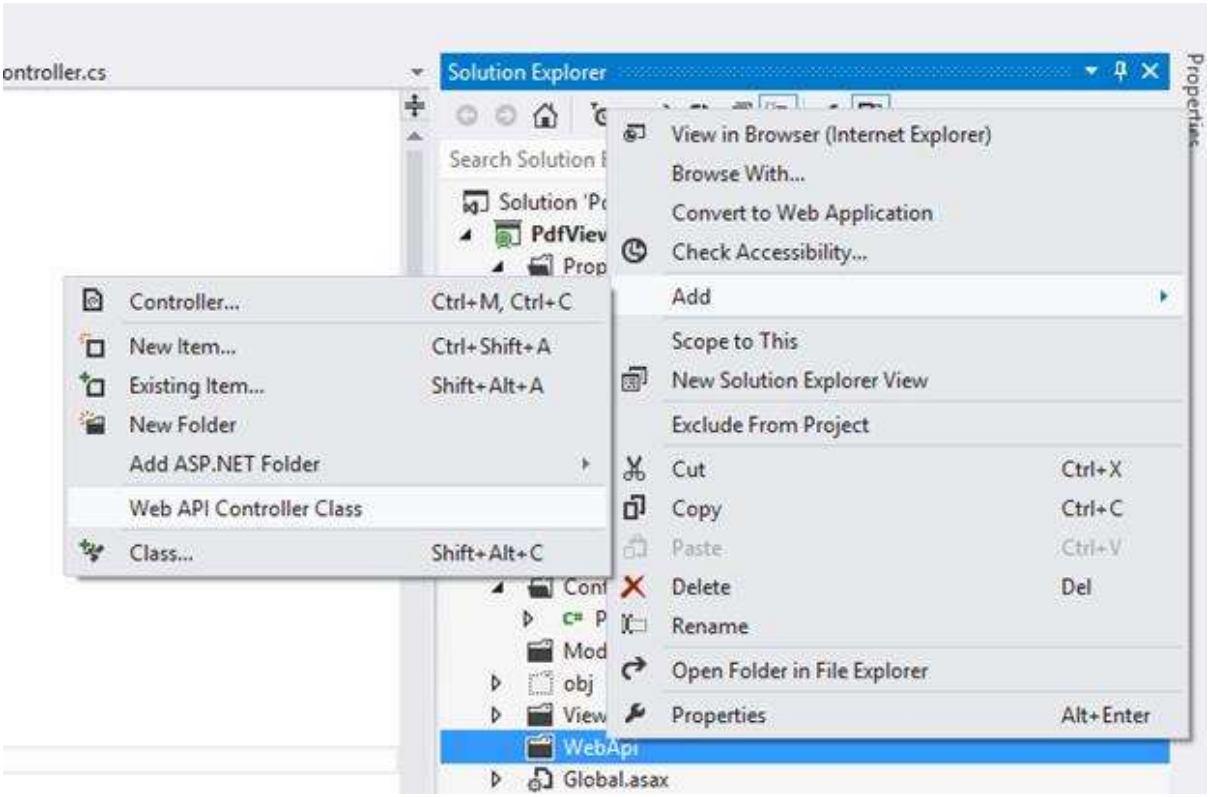
HTML

```
@using Syncfusion.JavaScript
@using Syncfusion.MVC.EJ
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<title>PDF viewer</title>
<link href="https://cdn.syncfusion.com/16.1.0.24/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://cdn.syncfusion.com/js/assets/external/jquery-
3.1.1.min.js"></script>
<script
src="https://cdn.syncfusion.com/16.1.0.24/js/web/ej.web.all.min.js"></scr
ipt>
</head>
<body>
<div>
<div style="width:100%;height:780px;">
```

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/
ejServices/api/PdfViewer"))
</div>
</div>
@(Html.EJ().ScriptManager())
</body>
</html>
```

Displaying PDF document using Web API

Add new folder **WebApi** in the solution and add new Web API Controller Class. Name it as **PdfViewerController** and click OK





Replace the below code in the PdfViewerController.cs

C#

```
using Newtonsoft.Json;
using Syncfusion.EJ.PdfViewer;
using System.Collections.Generic;
using System.IO;
using System.Web;
```

```
using System.Web.Http;
namespace PdfViewerSample.WebApi
{
public class PdfViewerController : ApiController
{
//Post action for processing the PDF documents.
public object Load(Dictionary<string, string> jsonResult)
{
PdfViewerHelper helper = new PdfViewerHelper();
if (jsonResult.ContainsKey("isInitialLoading"))
helper.Load(HttpContext.Current.Server.MapPath("~/App_Data/HTTP
Succinctly.pdf"));
return JsonConvert.SerializeObject(helper.ProcessPdf(jsonResult));
}
//Post action for processing the PDF documents when uploading to the
ejPdfviewer widget.
public object FileUpload(Dictionary<string, string> jsonResult)
{
PdfViewerHelper helper = new PdfViewerHelper();
if (jsonResult.ContainsKey("uploadedFile"))
{
var fileUrl = jsonResult["uploadedFile"];
byte[] byteArray = Convert.FromBase64String(fileUrl);
MemoryStream stream = new MemoryStream(byteArray);
helper.Load(stream);
}
return JsonConvert.SerializeObject(helper.ProcessPdf(jsonResult));
}
//Post action for downloading the PDF documents from the ejPdfviewer
widget.
public object Download(Dictionary<string, string> jsonResult)
{
PdfViewerHelper helper = new PdfViewerHelper();
return helper.GetDocumentData(jsonResult);
}
//Post action for unloading and disposing the PDF document resources in
server side from the ejPdfviewer widget.
public void Unload()
{
PdfViewerHelper helper = new PdfViewerHelper();
helper.UnLoad();
}
}
}
```

Note: Add the PDF document to be viewed in App_Data folder of the project.

Add the following code snippet to Index.cshtml. Here, the PDF viewer uses the Web API to process the PDF.

HTML

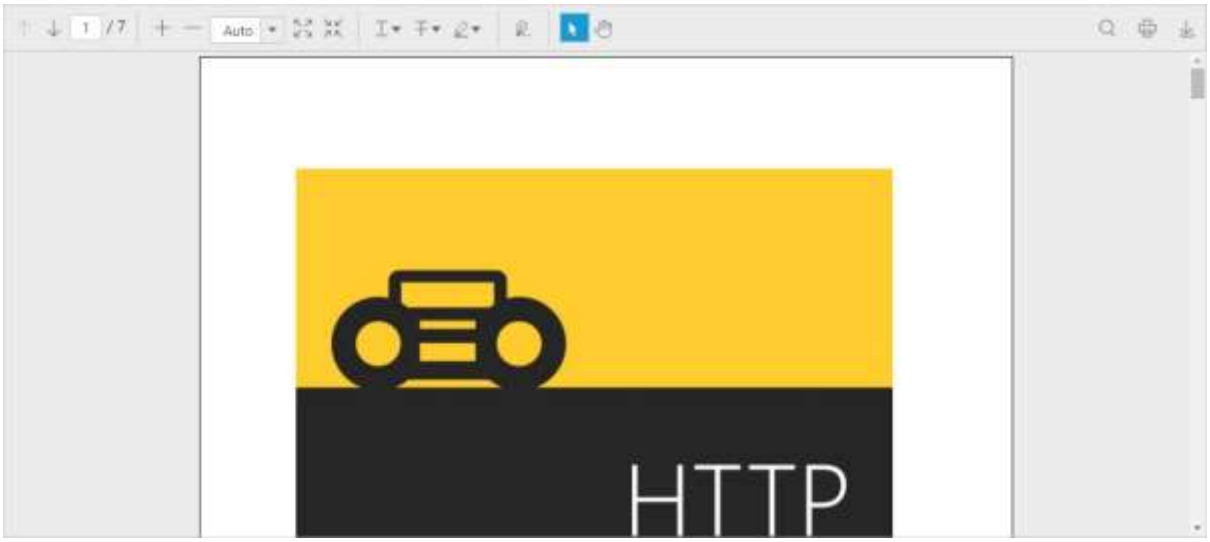
```
@using Syncfusion.JavaScript
@using Syncfusion.MVC.EJ
<!DOCTYPE html>
<html>
```

```
<head>
<meta name="viewport" content="width=device-width" />
<title>PDF viewer</title>
<link href="https://cdn.syncfusion.com/16.1.0.24/js/web/flat-
azure/ej.web.all.min.css" rel="stylesheet" />
<script src="https://cdn.syncfusion.com/js/assets/external/jquery-
3.1.1.min.js"></script>
<script
src="https://cdn.syncfusion.com/16.1.0.24/js/web/ej.web.all.min.js"></scr
ipt>
</head>
<body>
<div>
<div style="width:100%;height:780px;">
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolu
te("~/api/PdfViewer")))
</div>
</div>
@(Html.EJ().ScriptManager())
</body>
</html>
```

Sample:

http://www.syncfusion.com/downloads/support/directtrac/general/ze/PdfViewer_GettingStarted-1550599436

Run the sample and you will see the PDF viewer control as in the below screenshot.



Toolbar Customization

Customizing default toolbar

The default toolbar is grouped into the following set of tools.

MagnificationTools	Contains ZoomIn, ZoomOut, Zoom, FitPage and FitWidth tools.
NavigationTools	Contains GoToNext and GoToLast tools.

Copyright © 2001 - 2020 Syncfusion Inc.

PrintTools	Contains print tool.
DownloadTool	Contains download tool.
TextSearchTool	Contains text search tool.
TextMarkupAnnotationTools	Contains text markup annotation tools.
SignatureTool	Contains signature tool.
SelectionTool	Contains selection tool.

The PDF viewer control has an option to show or hide these grouped items in the default toolbar. You can hide or display any of these tools by using the toolbarSettings property. The following code snippet describes how to show the magnification tools in the widget.

JAVASCRIPT

```
<div style="width:100%;height:780px;">
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolu
te("~/api/PdfViewer"))
.ToolbarSettings(t=>t.Items(Syncfusion.JavaScript.PdfViewerEnums.ToolbarI
tems.MagnificationTools)))
</div>
```

The PDF viewer control also has an option to show or hide the complete default toolbar. You can achieve this by using the showToolbar(bool) method. The following code snippet describes how to hide the default toolbar in the widget.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showToolbar(false);
</script>
```

Magnification tools

The magnification tools of the PDF viewer contain ZoomIn, ZoomOut, Zoom, FitPage, and FitWidth tools in the default toolbar. The PDF viewer control also has an option to show or hide the magnification tools in the default toolbar. You can achieve this by using the showMagnificationTools(bool) method. The following code snippet describes how to hide the magnification tools in the default toolbar in the widget.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showMagnificationTools(false);
</script>
```

APIs available for Magnification tools

fitToPage()

Copyright © 2001 - 2020 Syncfusion Inc.

You can fit the page size of the PDF document loaded in the PDF viewer control using the fitToPage() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").fitToPage();
</script>
```

fitToWidth()

You can fit the page width of the PDF document loaded in the PDF viewer control using the fitToWidth() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").fitToWidth();
</script>
```

zoomIn()

You can zoom the PDF document page loaded in the PDF viewer control using zoomIn() method. The maximum zoom percentage is 400%.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").zoomIn();
</script>
```

zoomOut()

You can zoom out the PDF document page loaded in the PDF viewer control using the zoomOut() method. The minimum zoom percentage is 50%.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").zoomOut();
</script>
```

zoomTo(zoomValue)

The PDF document page loaded in the PDF viewer control can be zoomed to a specific value using the zoomTo(zoomValue) method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").zoomTo(200);
</script>
```

zoomPercentage

Copyright © 2001 - 2020 Syncfusion Inc.

The zoomPercentage property of the PDF viewer control returns the current zoom value of the PDF document.

JAVASCRIPT

```
<script>
var pdfviewer = $("pdfviewer").data("ejPdfViewer");
var zoomValue = pdfviewer. zoomPercentage;
</script>
```

Navigation Tools

The navigation tools of the PDF viewer contain GoToNext, GoToPrevious, and current page number tools in the default toolbar. The PDF viewer control also has an option to show or hide the navigation tools in the default toolbar. You can achieve this by using the showPageNavigationTools(bool) method. The following code snippet describes how to hide the navigation tools in the default toolbar in the widget.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showPageNavigationTools(false);
</script>
```

APIs available for Navigation Tools

goToPreviousPage()

The previous page of the PDF document from the current page can be navigated in the PDF viewer using goToPreviousPage() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").goToPreviousPage();
</script>
```

goToNextPage()

The next page of the PDF document from the current page can be navigated in the PDF viewer using the goToNextPage() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").goToNextPage();
</script>
```

goToFirstPage()

The first page of the PDF document can be navigated in the PDF viewer using the goToFirstPage() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").goToFirstPage();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

goToLastPage()

The last page of the PDF document can be navigated in the PDF viewer using the goToLastPage() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").goToLastPage();
</script>
```

goToPage(pageNumber)

The specific page in the PDF document can be navigated using the goToPage(pageNumber) method. If the page is not available for the given pageNumber, the PDF viewer retains the existing page in view.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").goToPage(4);
</script>
```

Text Markup Annotation Tools

The text markup annotation tools of the PDF viewer contain strikeout, highlight, and underline tools in the default toolbar. The PDF viewer control also has an option to show or hide the text markup annotation tools in the default toolbar. You can achieve this by using the showTextMarkupAnnotationTools(bool) method. The following code snippet describes how to hide the text markup annotation tools in the default toolbar in the widget.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showTextMarkupAnnotationTools(false);
</script>
```

Print Tool

The print tool of the PDF viewer contains an option to print the PDF document. When the print button is clicked, the changes made in the PDF document will be printed along with the document using the browser's default printer settings. The PDF viewer control provides the option to show or hide the print tool in the default toolbar. You can achieve this by using the showPrintTools(bool) method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showPrintTools(false);
</script>
```

The printing of the PDF document can be achieved in the client side by calling the print() method.

JAVASCRIPT

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script>
$("#pdfviewer").data("ejPdfViewer").print();
</script>
```

Download Tool

The download tool of the PDF viewer contains an option to download the PDF document. When the download button is clicked, the changes made in the PDF document will be saved in a copy of the original PDF document and it will be downloaded in the browser. The PDF viewer control provides the option to show or hide the download tool in the default toolbar. You can achieve this by using the showDownloadTool(bool) method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showDownloadTool(false);
</script>
```

The PDF document can be downloaded in the client side by calling the download() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").download();
</script>
```

Signature tool

The signature tool of the PDF viewer contains an option to include handwritten signatures in the PDF document. The PDF viewer control provides the option to show or hide the signature tool in the default toolbar. You can achieve this by using the showSignatureTool(bool) method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showSignatureTool(false);
</script>
```

Selection tool

The selection tool of the PDF viewer control contains options for selection and panning interaction modes. The PDF viewer control provides the option to show or hide the selection tool in the default toolbar. You can achieve this by using the showSelectionTool(bool) method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showSelectionTool(false);
</script>
```

Text Search tool

The text search tool of the PDF viewer control is used to initiate and close the text search process in the PDF viewer control. The PDF viewer control provides the option to show or hide the text search tool in the default toolbar. You can achieve this by using the showTextSearchTool(bool) method.

Copyright © 2001 - 2020 Syncfusion Inc.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").showTextSearchTool(false);
</script>
```

APIs available for Text Search tool

searchText(targetText)

searchText(targetText) method allows us to search the target text in the PDF document and highlights the occurrences in the pages.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").searchText("name");
</script>
```

searchNext()

The next occurrences of the searched text from the current occurrence can be searched using the searchNext() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").searchNext();
</script>
```

SearchPrevious()

The previous occurrence of the searched text from the current occurrence can be searched using the searchPrevious() method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").searchPrevious();
</script>
```

matchCase(enableMatchCase)

The target text in the PDF document can be searched with its casing using the matchCase(enableMatchCase) method.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").matchCase(true);
</script>
```

cancelSearchText()

The text search can be canceled and the highlighted occurrences from the PDF viewer can be removed using the cancelSearchText() method.

Copyright © 2001 - 2020 Syncfusion Inc.

JAVASCRIPT

```
<script>
$("#pdfviewer").data("ejPdfViewer").cancelSearchText();
</script>
```

Events in the PDF viewer control

BeforePrint

When the print option is clicked, the beforePrint event will be triggered before printing the PDF document from the PDF viewer control. We can define the event method using the BeforePrint property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/
ejServices/api/PdfViewer").ClientSideEvents(e=>e.BeforePrint("beforePrint
")))
<script type="text/javascript">
function beforePrint(args) {
alert("The PDF document is made ready for printing.");
}
</script>
```

AfterPrint

After the printing process is completed, the afterPrint event will be triggered. We can define the event method using the AfterPrint property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/
ejServices/api/PdfViewer").ClientSideEvents(e=>e.AfterPrint("afterPrint")
))
<script type="text/javascript">
function afterPrint (args) {
alert("Printing completed successfully");
}
</script>
```

ZoomChanged

When the zoom value of the PDF document is changed using the magnification tools, the zoomChange event will be triggered. We can define the event method using the ZoomChanged property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/
ejServices/api/PdfViewer").ClientSideEvents(e=>e.ZoomChanged("zoomChanged
")))
<script type="text/javascript">
function zoomChanged(args) {
alert("The magnification changes from " + args.previousZoomPercentage + "
to" + args.currentZoomPercentage);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
```

PageClicked

When a page of the PDF document is clicked, the pageClick event will be triggered. We can define the event method using the PageClicked property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.PageClicked("pageClicked")))
<script type="text/javascript">
function pageClicked (args) {
alert("The page is clicked");
}
</script>
```

Adding Custom toolbar

The toolbar can be customized by hiding the existing toolbar.

The following code snippet shows how to create a customer toolbar by using the client side APIs

HTML

```
<body>
<div>
<div id="toolbarDiv" style="height:38px;width:100%;border:solid black 1px" class="toolbarclass " role="toolbar">
<div style="float:left; padding:5px;">
<div class="e-pdfviewer-icon e-pdfviewer-gotoprevious" id="viewer-previous" onclick="callClientSideMethod('previous')"></div>
<div class="e-pdfviewer-icon e-pdfviewer-gotonext" id="viewer-next" onclick="callClientSideMethod('next')"></div>
<input id="currentPage" class="e-pdfviewer-pagenumber e-pdfviewer-elementalignments" type="text" onkeypress="updatePageNumber(event)" />
<span id="totalPageCount" class="e-pdfviewer-labelpageno"></span>
<div class="e-pdfviewer-icon e-pdfviewer-zoomin" id="viewer-zoomin" onclick="callClientSideMethod('zoomin')"></div>
<div class="e-pdfviewer-icon e-pdfviewer-zoomout" id="viewer-zoomout" onclick="callClientSideMethod('zoomout')"></div>
<div class="e-pdfviewer-icon e-pdfviewer-fitwidth" id="viewer-fitwidth" onclick="callClientSideMethod('fitwidth')"></div>
<div class="e-pdfviewer-icon e-pdfviewer-fitpage" id="viewer-fitpage" onclick="callClientSideMethod('fitpage')"></div>
</div>
<div style="float:right; padding:5px;">
<div class="e-pdfviewer-icon e-pdfviewer-print" id="viewer-print" onclick="callClientSideMethod('print')"></div>
<div class="e-pdfviewer-icon e-pdfviewer-download" id="viewer-download" onclick="callClientSideMethod('download')"></div>
</div>
</div>
<div style="width:100%;height:780px;margin-top:3px">
```

```
@Html.EJ().PdfViewer("PdfViewer1").ServiceUrl(VirtualPathUtility.ToAbsolu
te("~/api/PdfViewer")).ClientSideEvents(e =>
e.DocumentLoaded("onDocumentLoad").PageChanged("pageChange"))
</div>
</div>
@(Html.EJ().ScriptManager())
<script type="text/javascript">
var currentPageNumber = 1;
$(function () {
var pdfviewer = $('#PdfViewer1').data('ejPdfViewer');
document.getElementById('currentPage').value = 1;
document.getElementById('totalPageCount').innerHTML = "/" +
pdfviewer.pageCount;
pdfviewer.showToolbar(false);
});
function onDocumentLoad() {
var _ejPdfViewer = $('#PdfViewer1').data('ejPdfViewer');
document.getElementById('totalPageCount').innerHTML = "/" +
_ejPdfViewer.pageCount;
}
function pageChange(args) {
currentPageNumber = args.currentPageNumber;
document.getElementById('currentPage').value = args.currentPageNumber;
}
function updatePageNumber(event) {
var _ejPdfViewer = $('#PdfViewer1').data("ejPdfViewer");
currentPageNumber = document.getElementById('currentPage').value;
if (event.which == 13) {
if (currentPageNumber > 0 && currentPageNumber <= _ejPdfViewer.pageCount)
{
_ejPdfViewer.goToPage(currentPageNumber);
} else {
currentPageNumber = _ejPdfViewer.currentPageNumber;
}
}
}
function callClientSideMethod(apiName) {
var _ejPdfViewer = $('#PdfViewer1').data("ejPdfViewer");
switch (apiName) {
case 'print':
_ejPdfViewer.print();
break;
case 'zoomin':
_ejPdfViewer.zoomIn();
break;
case 'zoomout':
_ejPdfViewer.zoomOut();
break;
case 'fitpage':
_ejPdfViewer.fitToPage();
break;
case 'fitwidth':
_ejPdfViewer.fitToWidth();
break;
case 'previous':
_ejPdfViewer.goToPreviousPage();
break;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
case 'next':
_ejPdfViewer.goToNextPage();
break;
case 'download':
_ejPdfViewer.download();
break;
}
}
</script>
<style>
#totalPageCount {
float: none !important;
}
</style>
</body>
```

Note: Ensure the icon images available in the referred location for the custom toolbar.

Run the sample. You can view the PDF viewer with custom toolbar.

Sample:

http://www.syncfusion.com/downloads/support/directtrac/general/ze/PdfViewer_CustomToolbar-656684563

Localization

The text labels provided in the PDF viewer control can be localized using the collection of localized strings using ej.PdfViewer.Locale for different cultures. By default, the PDF viewer control is localized in "**en-US**" culture.

The different locale value for the PDF viewer control can be specified using the locale property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolu
te("~/api/PdfViewer")).Locale("de-DE"))
```

The following table shows the default values for the components in the PDF viewer control in 'en-US' culture:

Component	Function	Keywords	Values
toolbar	first	headerText	First
		contentText	Go to the first page of the PDF document.
	previous	headerText	Previous
		contentText	Go to the previous page of the PDF document.
	next	headerText	Next

Copyright © 2001 - 2020 Syncfusion Inc.

		contentText	Go to the next page of the PDF document.
	last	headerText	Last
		contentText	Go to the last page of the PDF document.
	zoomIn	headerText	Zoom-In
		contentText	Zoom in to the PDF document.
	zoomOut	headerText	Zoom-Out
		contentText	Zoom out of the PDF document.
	pageIndex	headerText	Page Number
		contentText	Current page number in view.
	zoom	headerText	Zoom
		contentText	Zoom in or out of the PDF document.
	fitToWidth	headerText	Fit to Width
		contentText	Fit the PDF page to the width of the container.
	fitToPage	headerText	Fit to Page
		contentText	Fit the PDF page to the container.
	print	headerText	Print
		contentText	Print the PDF document.
	download	headerText	Download
		contentText	Download the PDF document.
	search	headerText	Search Text
		contentText	Search text in the PDF pages.
	highlight	headerText	Highlight Text
		contentText	Highlight text in the PDF pages.

Copyright © 2001 - 2020 Syncfusion Inc.

	strikeout	headerText	Strikethrough Text
		contentText	Strikethrough text in the PDF pages.
	underline	headerText	Underline Text
		contentText	Underline text in the PDF pages.
	signature	headerText	Signature
		contentText	Add or create the hand-written signature.
	select	headerText	Selection
		contentText	Selection tool for text.
	scroll	headerText	Panning
		contentText	Click to pan around the document.
contextMenu	copy	contentText	Copy
	googleSearch	contentText	Search Google
	openPopup	contentText	Open Pop-Up Note
	Delete	contentText	Delete
	properties	contentText	Properties
	Find	contentText	Find
	matchCase	contentText	Match Case
	auto	contentText	Auto
propertyWindow	underlineProperties	contentText	Underline Properties
	strikeOutProperties	contentText	StrikeOut Properties
	highlightProperties	contentText	Highlight Properties
	signatureProperties	contentText	Signature Properties
	appearance	contentText	Appearance
	general	contentText	General
	color	contentText	Color
	opacity	contentText	Opacity
	author	contentText	Author

Copyright © 2001 - 2020 Syncfusion Inc.

	subject	contentText	Subject
	modified	contentText	Modified
	ok	contentText	OK
	cancel	contentText	Cancel
	locked	contentText	Locked
signatureWindow	Signature	contentText	Add Signature
	Add	contentText	Add
	clear	contentText	Clear

The following code snippet illustrates you to change the localization to German culture "**de-DE**"

HTML

```
@using Syncfusion.JavaScript
@using Syncfusion.MVC.EJ
<div>
<div style="width:100%;height:780px;">
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolu
te("~/api/PdfViewer")).Locale("de-DE"))
</div>
</div>
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$(function () {
ej.PdfViewer.Locale["de-DE"] = {
toolbar: {
first: {
headerText: 'Erste',
contentText: 'Gehen Sie auf die erste Seite des PDF-Dokument.'
},
previous: {
headerText: 'Zurück',
contentText: 'Gehen Sie auf die vorherige Seite des PDF-Dokuments.'
},
next: {
headerText: 'Nächster',
contentText: 'Gehen Sie auf die nächste Seite des PDF-Dokuments.'
},
last: {
headerText: 'Letzte',
contentText: 'Gehen Sie auf die letzte Seite des PDF-Dokuments.'
},
zoomIn: {
headerText: 'Hineinzoomen',
contentText: 'Vergrößern Sie das PDF-Dokument.'
},
zoomOut: {
headerText: 'Rauszoomen',
contentText: 'Zoom aus dem PDF-Dokument.'
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
},
pageIndex: {
headerText: 'Seitennummer',
contentText: 'Aktuelle Seitenzahl, um zu sehen.'
},
zoom: {
headerText: 'Zoom',
contentText: 'Vergrößern oder Verkleinern auf dem PDF-Dokument.'
},
fitToWidth: {
headerText: 'An Breite anpassen',
contentText: 'Montieren Sie die PDF-Seite an die Breite des Behälters.'
},
fitToPage: {
headerText: 'An Seite anpassen',
contentText: 'Montieren Sie die PDF-Seite in den Behälter.'
},
print: {
headerText: ' Drucken',
contentText: ' Drucken Sie das PDF-Dokument.'
},
download: {
headerText: ' Herunterladen',
contentText: ' Laden Sie das PDF-Dokument'
},
search: {
headerText: 'Suche Text',
contentText: 'Suchen Sie Text in den PDF-Seiten.'
},
highlight: {
headerText: 'hervorheben Text',
contentText: 'Markieren Sie Text in den PDF-Seiten.',
},
strikeout: {
headerText: 'StrikeOut Text',
contentText: 'Strike Out Text in den PDF-Seiten.',
},
underline: {
headerText: 'unterstreichen Text',
contentText: 'Unterstreichen Sie den Text in den PDF-Seiten.',
},
signature: {
headerText: 'Unterschrift',
contentText: 'Hinzufügen oder Erstellen der handschriftlichen
Unterschrift.',
},
select: {
headerText: 'Auswahl',
contentText: 'Auswahlwerkzeug für Text.',
},
scroll: {
headerText: 'Schwenken',
contentText: 'Klicken Sie, um das Dokument zu verschieben',
}
},
contextMenu: {
copy: {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
contentText: 'Kopieren',
},
googleSearch: {
contentText: 'Google suchen',
},
find: {
contentText: 'Finden:',
},
matchCase: {
contentText: 'MatchCase',
},
auto: {
contentText: 'Auto',
},
openPopup: {
contentText: 'Open Pop-Up Note',
},
delete: {
contentText: 'Löschen',
},
properties: {
contentText: 'Eigenschaften....',
},
},
propertyWindow: {
underlineProperties: {
contentText: "Unterstreichen"
},
strikeOutProperties: {
contentText: "StrikeOutProperties"
},
highlightProperties: {
contentText: "Markieren Sie die Eigenschaften"
},
signatureProperties: {
contentText: "Unterschrift Eigenschaften"
},
appearance: {
contentText: "das Auftreten"
},
general: {
contentText: "General"
},
color: {
contentText: "Farbe:"
},
opacity: {
contentText: "Opazität:"
},
author: {
contentText: "Autor:"
},
subject: {
contentText: "Fach:"
},
modified: {
contentText: "geändert:"
```

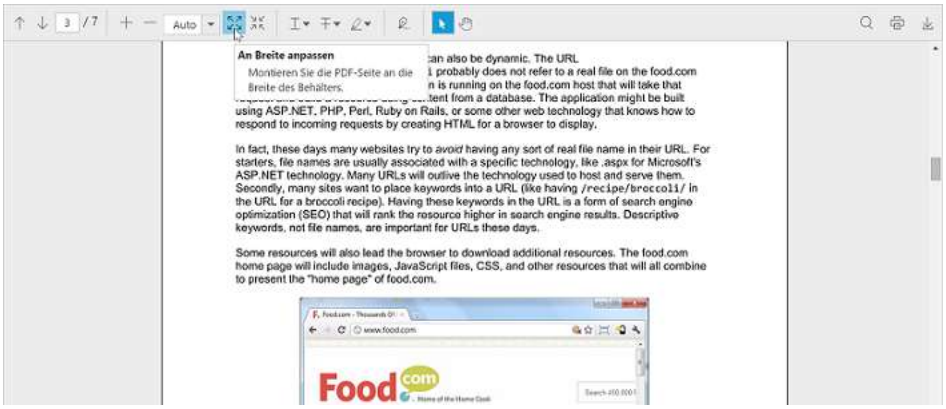
Copyright © 2001 - 2020 Syncfusion Inc.

```
},
ok: {
contentText: "OK"
},
cancel: {
contentText: "stornieren"
},
locked: {
contentText: "eingesperrt"
}
},
signatureWindow: {
signature: {
contentText: "Signatur hinzufügen"
},
add: {
contentText: "Hinzufügen"
},
clear: {
contentText: "Klar"
},
}
};
});
</script>
```

Sample

http://www.syncfusion.com/downloads/support/directtrac/general/ze/PdfViewer_Localization-1273554386

The following screenshot shows the PDF viewer with tooltip in German language.



Text Markup Annotation

The PDF viewer control supports adding text markup annotations in the PDF documents. The control also renders the existing text markup annotations from the PDF document when the document is loaded in it.

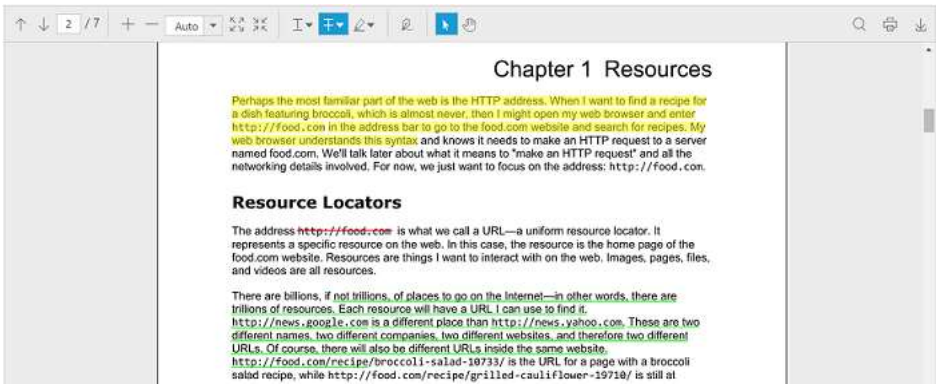
The text markup annotation tools in the default toolbar can be enabled or disabled using the toolbarSettings property.

The following code snippet describes how to show only the text markup annotation tools in the control.

JAVASCRIPT

```
(Html.EJ().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer")).ToolbarSettings(t =>
t.Items(Syncfusion.JavaScript.PdfViewerEnums.ToolbarItems.TextMarkupAnnotationTools)))
```

The following screenshot shows the PDF viewer with the PDF documents containing text markup annotations:



Enable or Disable the text markup annotations

The adding and modifying of the text markup annotations can be enabled or disabled using the EnableTextMarkupAnnotations property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").EnableTextMarkupAnnotations(true))
```

The adding and modifying of the highlight annotations can be enabled or disabled using the EnableHighlightAnnotation property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").EnableHighlightAnnotation(true))
```

The adding and modifying of the strikethrough annotations can be enabled or disabled using the EnableStrikethroughAnnotation property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").EnableStrikethroughAnnotation(true))
```

The adding and modifying of the underline annotations can be enabled or disabled using the EnableUnderlineAnnotation property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").EnableUnderlineAnnotation(true))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Providing settings for text markup annotations

You can get or set the settings of the highlight annotations using the highlightSettings property. The highlightSettings property is used to set the color, opacity, author, subject, modifiedDate and isLocked properties of the highlight annotations.

JAVASCRIPT

```
$(function () {
$("#pdfviewer").ejPdfViewer({ highlightSettings: { color: "#ffff00",
author: "Guest", opacity: 0.5, subject: "highlight", modifiedDate: "2017-
03-27 12:00:51", isLocked: true} });
});
```

You can get or set the settings of the strikethrough annotations using the strikethroughSettings property. The strikethroughSettings property is used to set the color, opacity, author, subject, modifiedDate, and isLocked properties of the strikethrough annotations.

JAVASCRIPT

```
$(function () {
$("#pdfviewer").ejPdfViewer({ strikethroughSettings: { color: "#ff0000",
author: "Guest", opacity: 0.5, subject: "strikethrough", modifiedDate:
"2017-03-27 12:00:51", isLocked: true
} });
});
```

You can get or set the settings of the underline annotations using the underlineSettings property. The underlineSettings property is used to set the color, opacity, author, subject, modifiedDate, and isLocked properties of the underline annotations.

JAVASCRIPT

```
$(function () {
$("#pdfviewer").ejPdfViewer({ underlineSettings: { color: "#00ff00",
author: "Guest", opacity: 0.5, subject: "underline", modifiedDate: "2017-
03-27 12:00:51", isLocked: true
} });
});
```

Adding text markup annotations

The text markup annotations are added to the PDF document using the text markup annotation tools provided in the toolbar of the PDF viewer control.

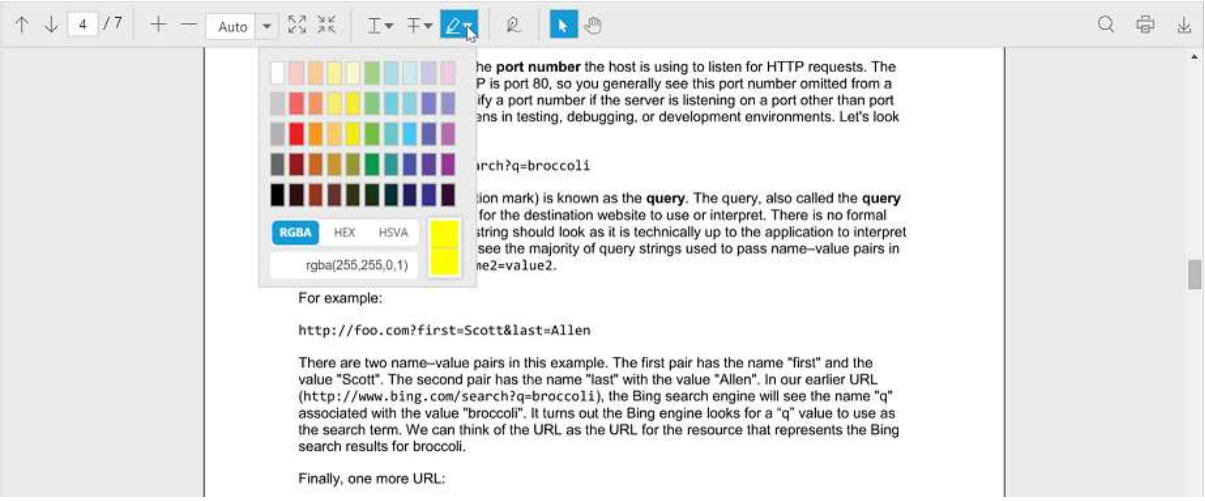
The annotations can be added to the PDF document from the client side using the annotationType property and addAnnotation method.

JAVASCRIPT

```
var pdfviewerObj = $("#pdfviewer").data("ejPdfViewer");
pdfviewerObj.addAnnotation(ej.PdfViewer.AnnotationType.Underline);
```

Copyright © 2001 - 2020 Syncfusion Inc.

The colorpicker control is provided in the text markup annotation tools to select the desired color for the text markup annotation to be added in the document.



When the text markup annotation is added in the PDF document, the annotationAdd event will be triggered in the control. The event method can be defined using the AnnotationAdd property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.AnnotationAdd("annotationAdded")))
function annotationAdded(args) {
alert("A text markup annotation is added in the PDF document");
}
```

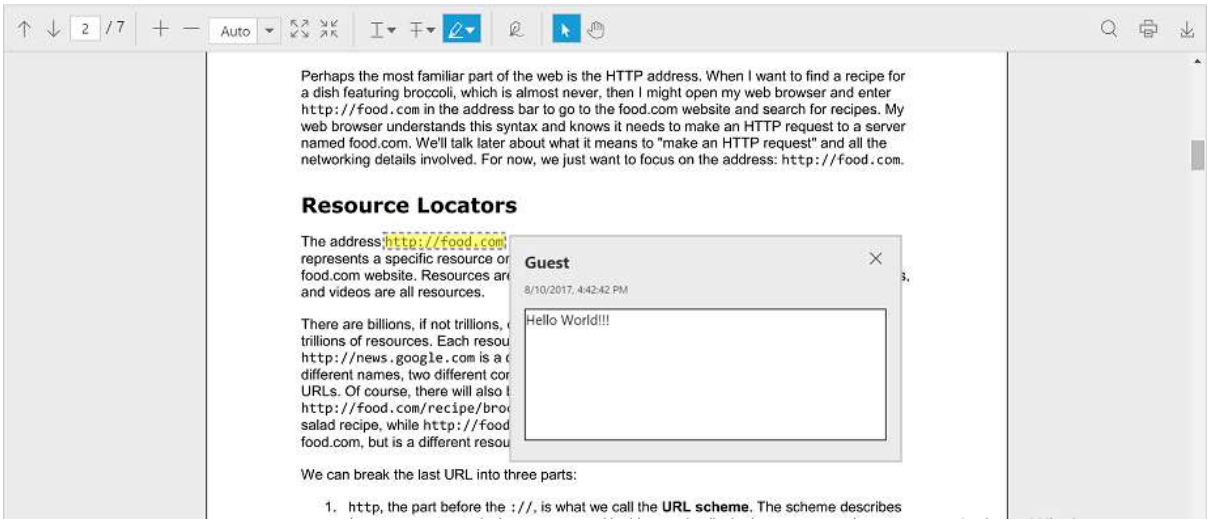
Adding notes to the text markup annotation

The notes can be added to the text markup annotations using the context menu in the PDF viewer control.

Right click the text markup annotation in the PDF document, the context menu will be opened in the PDF viewer control. You can select the option **Open Pop-up Note** to add the note to the annotation.

The following screenshot shows that the notes have been added to a text markup annotation in the PDF document:

Copyright © 2001 - 2020 Syncfusion Inc.



Editing the text markup annotation

The properties of the text markup annotations in the PDF document can be edited using the properties window provided in the PDF viewer control. The color, opacity, author, and subject of the text markup annotation can be modified. The properties of the text markup annotation can also be locked using the locked checkbox in the properties window.

Note: The isLocked property of the text markup annotation will lock the properties of the selected annotation. The property of the text markup annotation can be modified once the isLocked property is disabled.

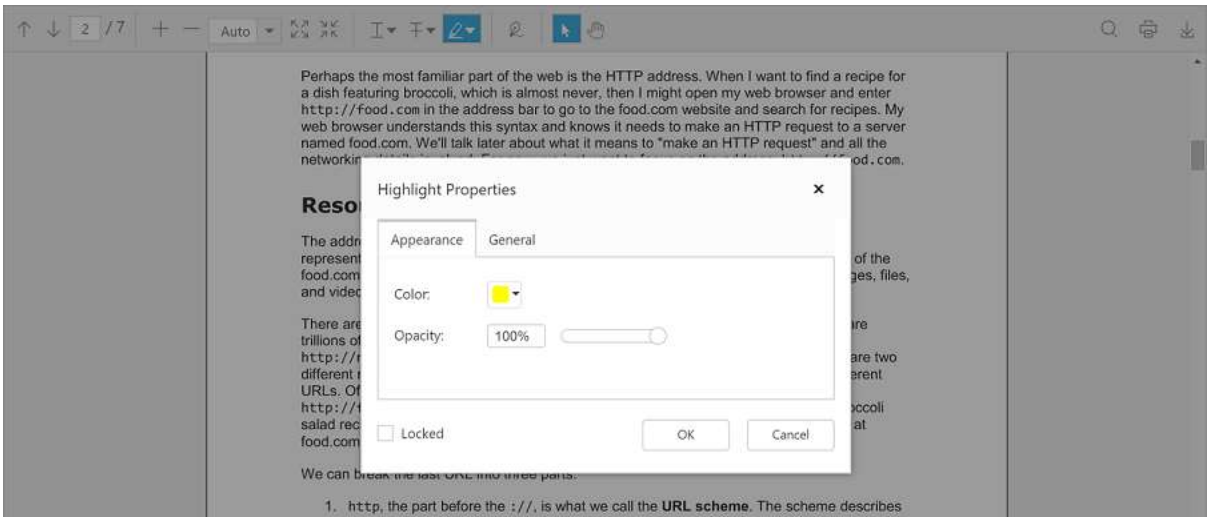
When a property of the text markup annotation is changed in the PDF viewer control, annotationPropertiesChange event will be triggered in the control. The event method can be defined using the AnnotationPropertiesChange property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.AnnotationPropertiesChange("annotationPropertiesChanged")))
function annotationPropertiesChanged(args) {
alert("The properties of a text markup annotation is modified in the PDF viewer");
}
```

The following screenshot shows the properties window used for editing the properties of a text markup annotation:

Copyright © 2001 - 2020 Syncfusion Inc.



The annotationDelete event will be triggered in the control when the text markup annotation in the PDF document is deleted. The event method can be defined using the AnnotationDelete property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.AnnotationDelete("annotationDeleted")))
function annotationDeleted (args) {
alert("A text markup annotation is deleted in the PDF document");
}
```

You can undo and redo the changes made to text markup annotations included in the PDF documents using the undo() and redo() methods.

JAVASCRIPT

```
function undoChanges() {
$("#pdfviewer").data("ejPdfViewer").undo();
}
function redoChanges() {
$("#pdfviewer").data("ejPdfViewer").redo();
}
```

Saving the text markup annotation

When the download tool is selected in the toolbar, the text markup annotations will be saved in the PDF document. This action will not affect the original document.

Printing the text markup annotation

When the print tool is selected in the toolbar, the PDF document will be printed along with the text markup annotations added to the pages. This action will not affect the original document.

Handwritten Signature

The PDF viewer supports adding handwritten signatures into the PDF document. The handwritten signature reduces the paper work of reviewing the content and verifies it digitally.

Copyright © 2001 - 2020 Syncfusion Inc.

The PDF viewer control has an option in the toolbarSettings property to enable or disable the signature button in the default toolbar.

The following code snippet describes how to disable the signature tool in the widget.

JAVASCRIPT

```
<div style="width:100%;height:780px;">
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolu
te("~/api/PdfViewer"))
.ToolbarSettings(t=>t.Items(Syncfusion.JavaScript.PdfViewerEnums.ToolbarI
tems.SignatureTool)))
</div>
```

Enable or Disable handwritten signature

The adding of handwritten signature feature in the PDF document can be enabled or disabled using the EnableSignature property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/
ejServices/api/PdfViewer").EnableSignature(true))
```

Providing settings for hand written signature

You can get or set the settings of the handwritten signature using the signatureSettings property. The signatureSettings property is used to set the color and opacity of the hand written signature.

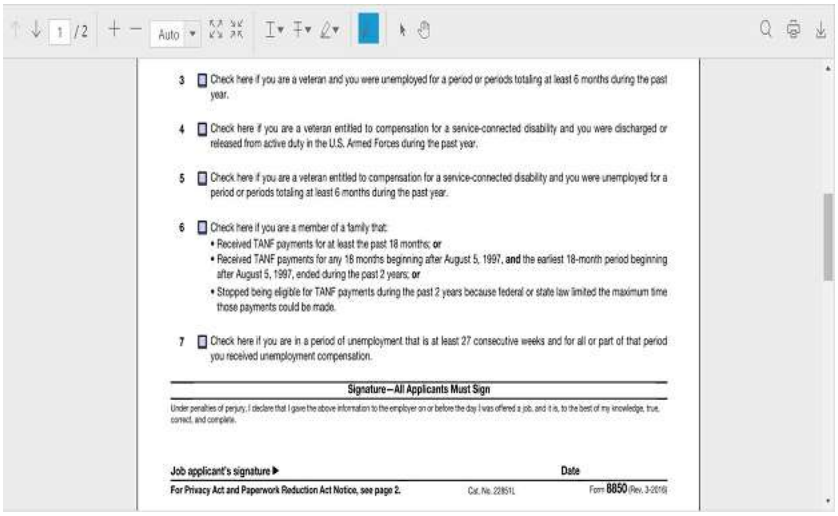
JAVASCRIPT

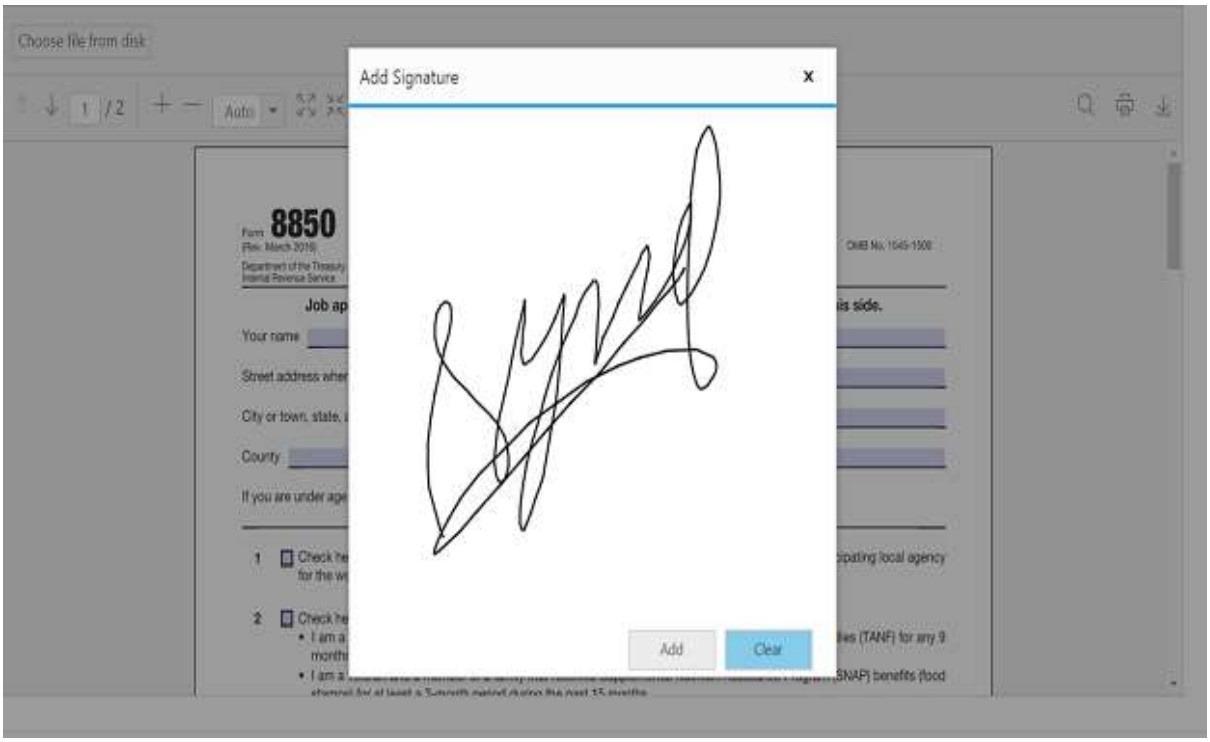
```
<script type="text/javascript">
$(function() {
$("#pdfviewer").ejPdfViewer({signatureSettings: { color: "#ff0000",
opacity: "0.5" } })
});
</script>
```

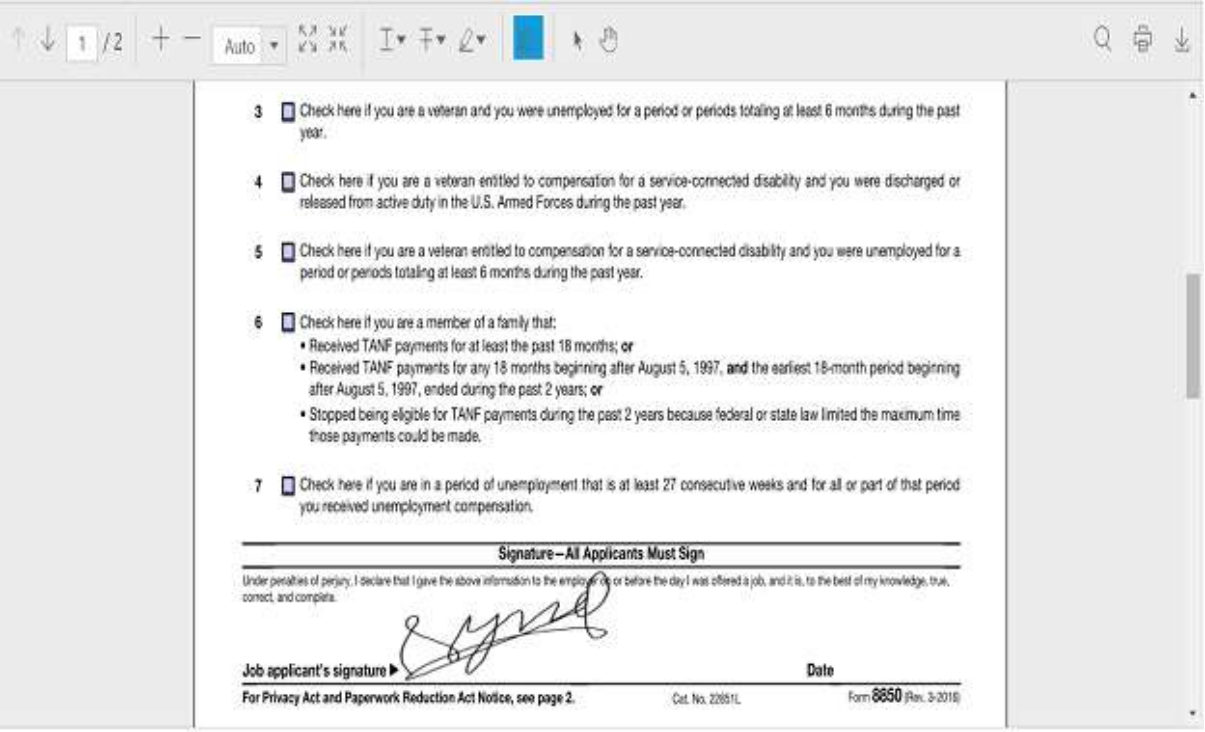
Adding signature in the PDF document

The handwritten signature can be added by drawing the signature content in the signature panel and clicking the button labeled ADD. The following screenshots are the pictorial representation of this.

Copyright © 2001 - 2020 Syncfusion Inc.


3 Check here if you are a veteran and you were unemployed for a period or periods totaling at least 6 months during the past year.
4 Check here if you are a veteran entitled to compensation for a service-connected disability and you were discharged or released from active duty in the U.S. Armed Forces during the past year.
5 Check here if you are a veteran entitled to compensation for a service-connected disability and you were unemployed for a period or periods totaling at least 6 months during the past year.
6 Check here if you are a member of a family that:
• Received TANF payments for at least the past 18 months; or
• Received TANF payments for any 18 months beginning after August 5, 1997, and the earliest 18-month period beginning after August 5, 1997, ended during the past 2 years; or
• Stopped being eligible for TANF payments during the past 2 years because federal or state law limited the maximum time those payments could be made.
7 Check here if you are in a period of unemployment that is at least 27 consecutive weeks and for all or part of that period you received unemployment compensation.
Signature—All Applicants Must Sign
Under penalties of perjury, I declare that I gave the above information to the employer on or before the day I was offered a job, and it is, to the best of my knowledge, true, correct, and complete.
Job applicant's signature ▶
Date
For Privacy Act and Paperwork Reduction Act Notice, see page 2.
Cat. No. 22851L
Form 8850 (Rev. 3-2019)


Choose file from disk
Add Signature
Add
Clear
Form 8850
(Rev. March 2019)
Department of the Treasury
Internal Revenue Service
OMB No. 1545-1500
Your name
County
1
2



When the handwritten signature is added in the PDF document, the signatureAdd event will be triggered in the control. The event method can be defined using the SignatureAdd property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.SignatureAdd("signatureAdded")))
<script type="text/javascript">
function signatureAdded (args) {
alert("A hand written signature is added in the PDF document");
}
</script>
```

Move, Resize and Delete the Handwritten Signature

The handwritten signature content can be moved to the specified location within the page bounds by using touch gestures, arrow keys, and mouse. We can also resize the handwritten signature content by maintaining the aspect ratio.

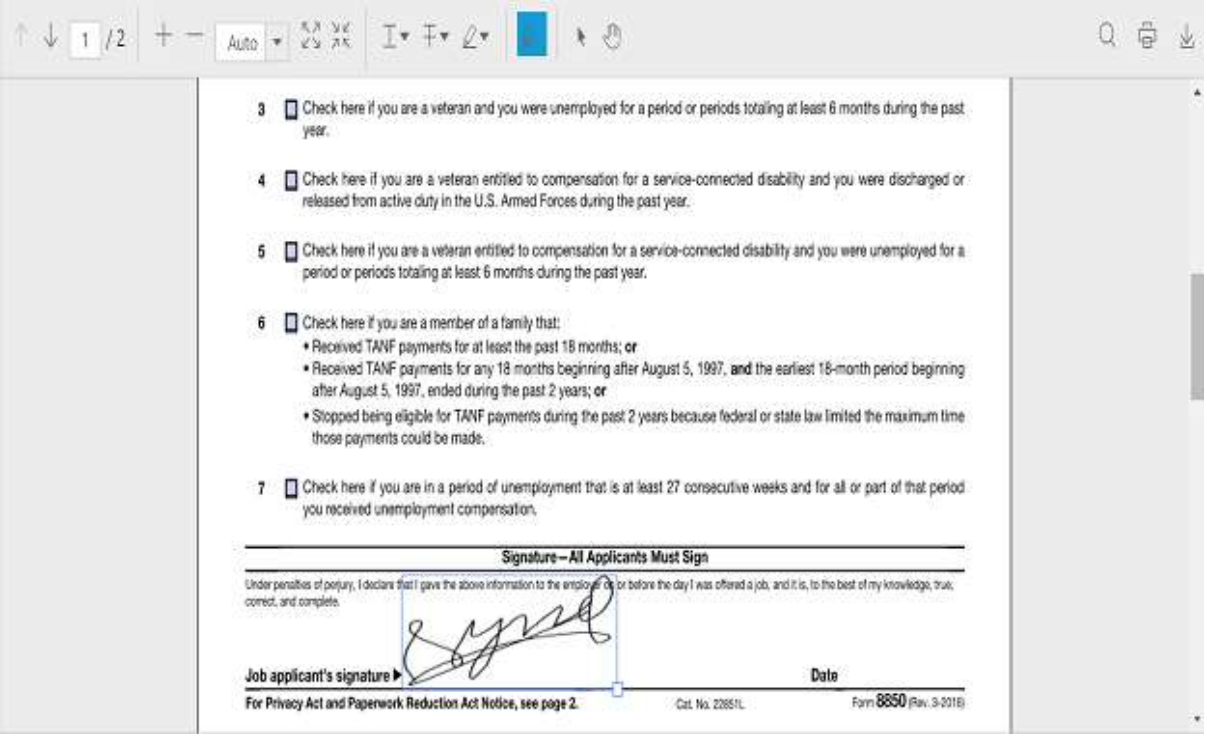
When the handwritten signature is resized in the PDF viewer control, the signatureResize event will be triggered in the control. The event method can be defined using the SignatureResize property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.SignatureResize("signatureResize")))
<script type="text/javascript">
function signatureResize (args) {
```

```
alert("A hand written signature is resized in the PDF viewer control");
}
</script>
```

The selected handwritten signature content can be deleted using the “Delete” option in the context menu or delete key. The following screenshots are the pictorial representation of this.



When the handwritten signature is deleted from the PDF document, the signatureDelete event will be triggered in the control. The event method can be defined using the SignatureDelete property of the control.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.SignatureDelete("signatureDelete")))
<script type="text/javascript">
function signatureDelete (args) {
alert("A hand written signature is removed from the PDF document");
}
</script>
```

You can undo and redo the changes made to the handwritten signatures included in the PDF document using the undo() and redo() methods.

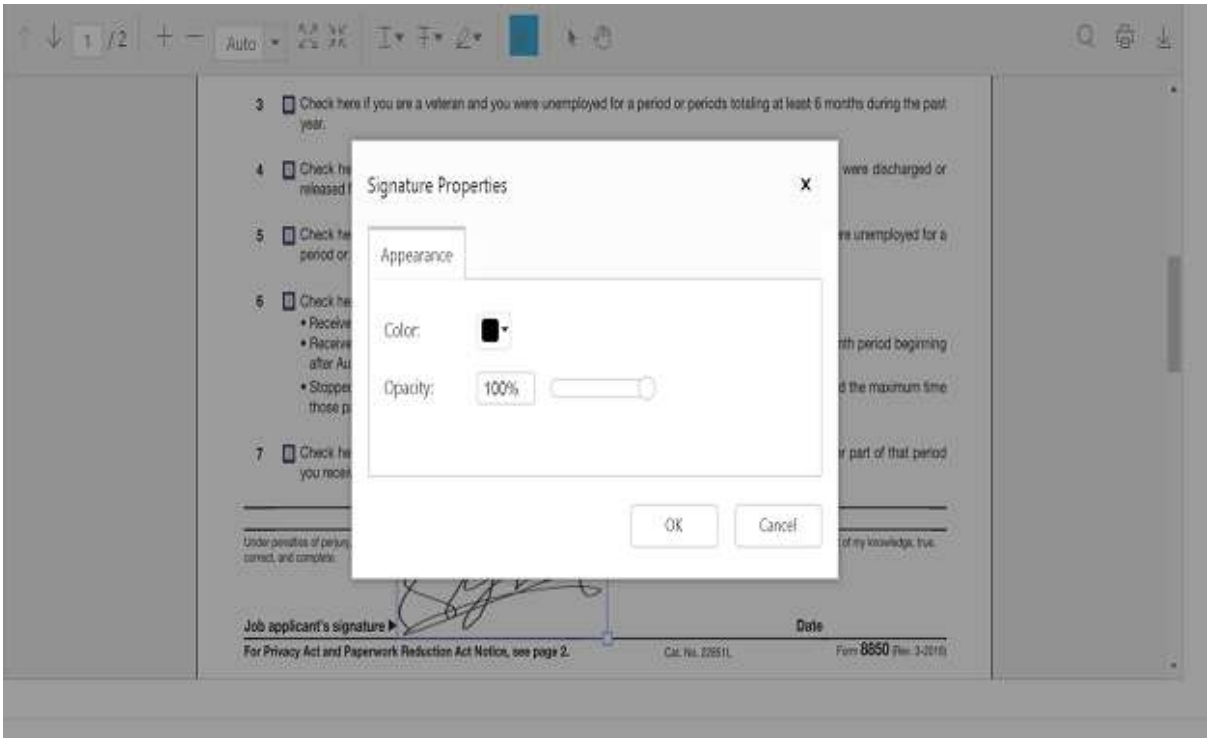
JAVASCRIPT

```
<script type="text/javascript">
function undoChanges() {
$("#pdfviewer").data("ejPdfViewer").undo();
}
```

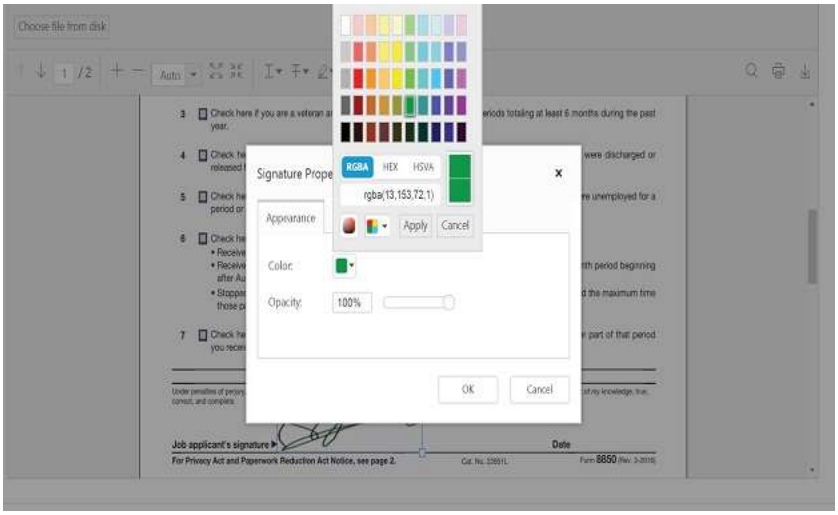
```
function redoChanges() {
$("#pdfviewer").data("ejPdfViewer").redo();
}
</script>
```

Changing properties of handwritten signature

The properties that is opacity and color of a handwritten signature can be modified by color palate and opacity slider, which is available in the “Properties” option in context menu. The following screenshots are the pictorial representation of this.

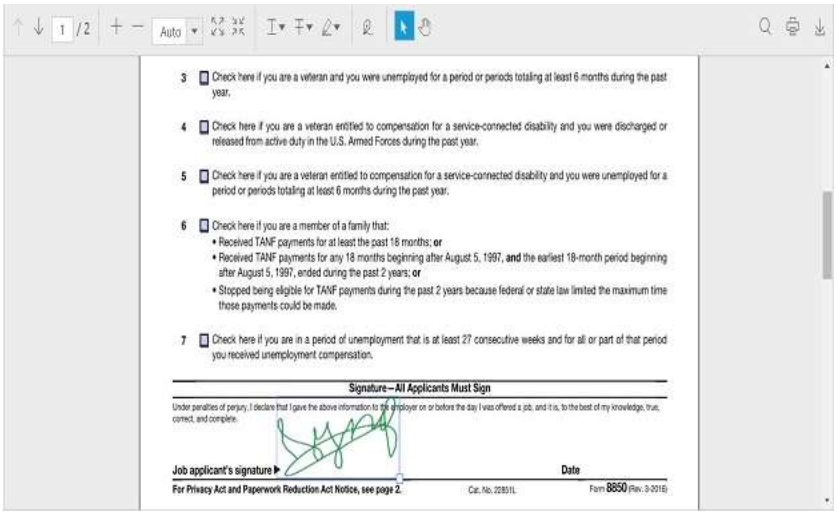


Select the desired color from the color palette and then click OK button.

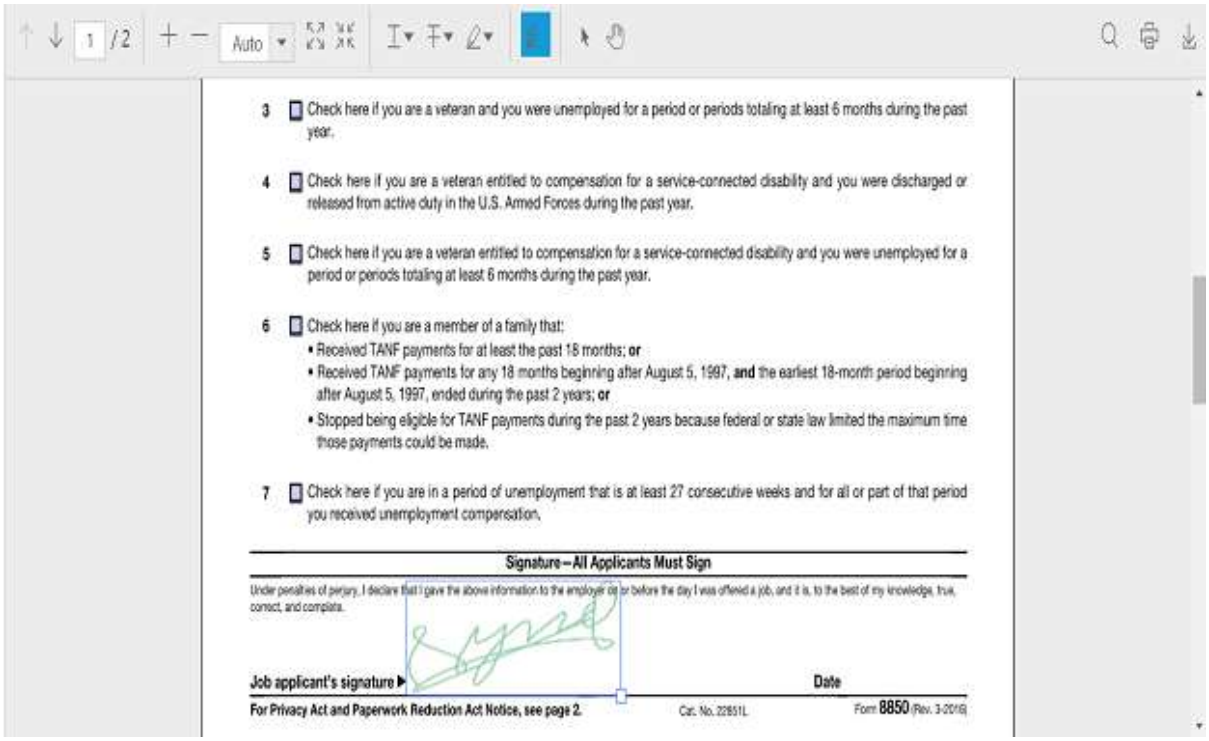


The selected color will be updated on the signature.

Copyright © 2001 - 2020 Syncfusion Inc.



You can also change the opacity of the added signature in the “properties” option.



When a property of the handwritten signature is changed in the PDF viewer control, the signaturePropertiesChange event will be triggered in the control. The event method can be defined using the SignaturePropertiesChange property of the control.

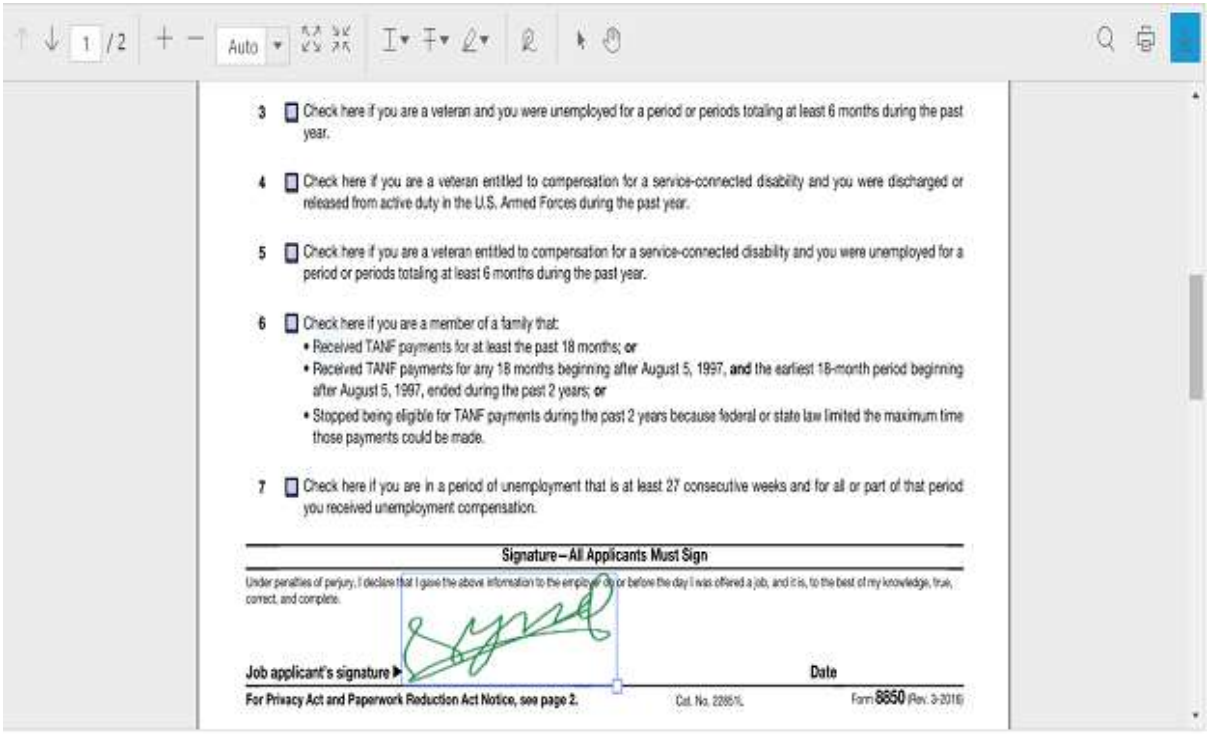
JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").ClientSideEvents(e=>e.SignaturePropertiesChange ("signaturePropertiesChanged")))
<script type="text/javascript">
function signaturePropertiesChanged(args) {
alert("A property of the hand written signature is changed in the PDF viewer control");
```

```
}
</script>
```

Saving the Signature

The added signature can be saved to the PDF document and can be downloaded by clicking the download button in the toolbar. This action will not affect the original document.



Printing the Signature

When the print button is clicked in the toolbar, the PDF document will be printed along with the signature added to the pages. This action will not affect the original document.

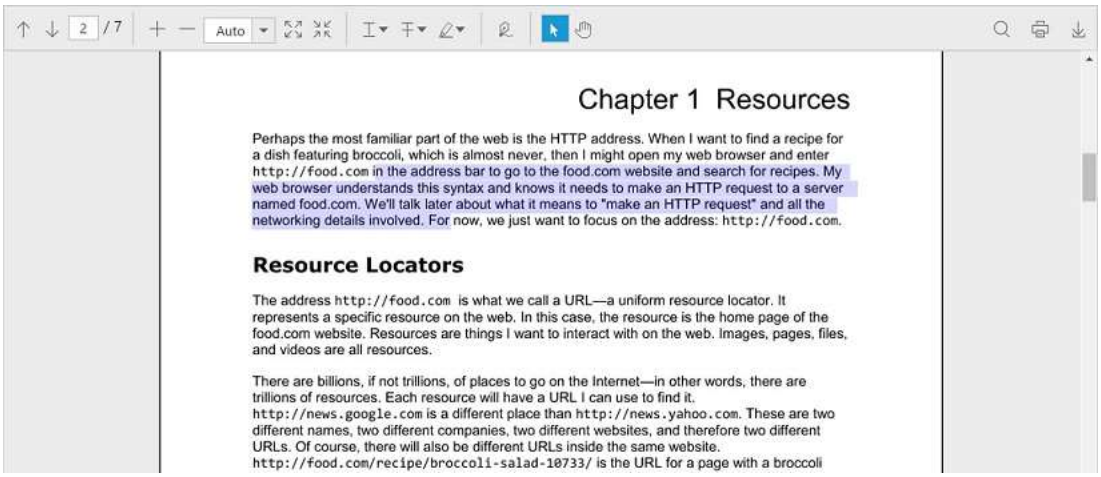
Interaction Mode

The PDF viewer control provides interaction modes for easy interaction with the PDF documents in the control. Selection mode and Panning mode are the two interaction modes.

Selection Mode

In this mode, the text selection can be performed in the PDF document loaded in the PDF viewer control. The panning and scrolling of the pages by touch cannot be performed in this mode.

Copyright © 2001 - 2020 Syncfusion Inc.



Enable or disable the Text selection

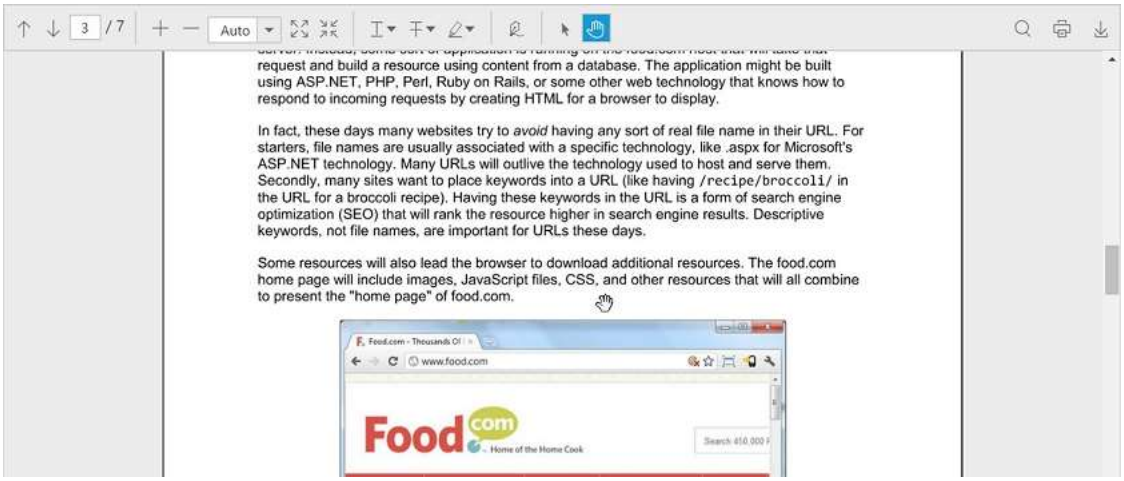
The text selection in the PDF viewer control can be enabled or disabled using the EnableTextSelection property.

JAVASCRIPT

```
@(Html.EJ().PdfViewer("pdfviewer").ServiceUrl("https://js.syncfusion.com/ejServices/api/PdfViewer").EnableTextSelection(true))
```

Panning Mode

In this mode, the panning and scrolling of the pages by touch can be performed in the PDF document loaded in the PDF viewer control, but the text selection cannot be performed.



Appearance

You can apply themes to the PDF viewer to match the themes of your application.

The following are the available built-in themes:

- Flat-azure
- Flat-lime
- Flat-lime-dark
- Flat-saffron

Copyright © 2001 - 2020 Syncfusion Inc.

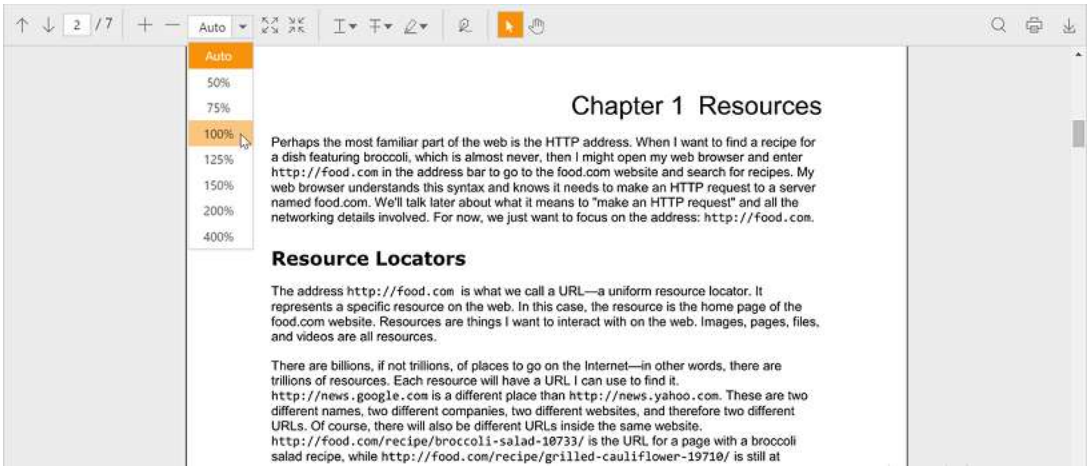
- Flat-saffron-dark
- Gradient-azure
- Gradient-azure-dark
- Gradient-lime
- Gradient-lime-dark
- Gradient-saffron
- Gradient-saffron-dark
- Bootstrap
- High-contrast-01
- High-contrast-02
- Material
- Office-365

By default, ‘flat-azure’ theme is applied to PDF viewer. You can change the theme by changing the stylesheet reference. Applying ‘gradient-saffron’ theme to PDF viewer is illustrated as follows.

JAVASCRIPT

```
<!—style sheet for 'gradient-saffron' theme.-->
<link href="https://cdn.syncfusion.com/16.1.0.24/js/web/gradient-saffron/ej.web.all.min.css" rel="stylesheet" />
<script src="https://cdn.syncfusion.com/js/assets/external/jquery-3.1.1.min.js"></script>
<script src="https://cdn.syncfusion.com/16.1.0.24/js/web/ej.web.all.min.js"></script>
```

The following screenshot shows the PDF viewer in ‘gradient-saffron’ theme:



Supported and Unsupported Elements

The supported and un-supported elements of the PDF viewer control are:

Features	Support
Text	Yes
Graphical Elements(lines, curves etc.,)	Yes

Fonts	
Standard Fonts	Yes
TrueType Fonts*	Yes
Embedded Fonts(Type 0, Type 1,Type 3)	Yes
Image Formats	
JPEG	Yes
PNG	Yes
TIFF	Yes
Inline images	Yes
Image masking Techniques	
Soft Mask	Yes
Image Mask	Yes
Color space	
RGB	Yes
CMYK	Yes
Gray	Yes
ICC	Yes
Indexed	Yes
CalRGB*	Yes
CalGray*	Yes
Lab*	Yes
Separation*	Yes
DeviceN*	Yes
Pattern*	Yes
Compression Filters	
DCTDecode (Image, Content)	Yes
CCITTFaxDecode (Image)	Yes
FlateDecode (Image, Content)	Yes
LZWDecode (Content,Image)	Yes

Copyright © 2001 - 2020 Syncfusion Inc.

ASCII85Decode (Content, Image)	Yes
ASCIIHexDecode (Image)	Yes
JBIG2Decode (Image)	Yes
RunLengthDecode(Image)	Yes
JPXDecode(Image)	Yes
Interactive Features	
Actions	No
Annotations	Yes
Hyperlink	Yes
Attachments	No
Bookmarks	No
Form	Yes
Page Navigation	Yes
Zoom	Yes
Selection	Yes
Search	Yes
Encrypted Documents	Yes
Edit	No
Conformance	Yes
Print	Yes
Download	Yes
Export to image	No

Supported only for shapes and text*

Supports only text markup annotations

When the document contains text that uses TrueType System Fonts to be rendered, the content in the PDF document is rendered using the font installed in the system or machine in the PDF Viewer control. If the required font is not installed in the system, the contents would be rendered using **Arial** font.*

Copyright © 2001 - 2020 Syncfusion Inc.

PercentageTextBox

PercentageTextBox

Essential ASP.NET MVC PercentageTextBox is used to display only percentage values. It has Spin buttons to increase or decrease the values in the Text Box.

Key Features

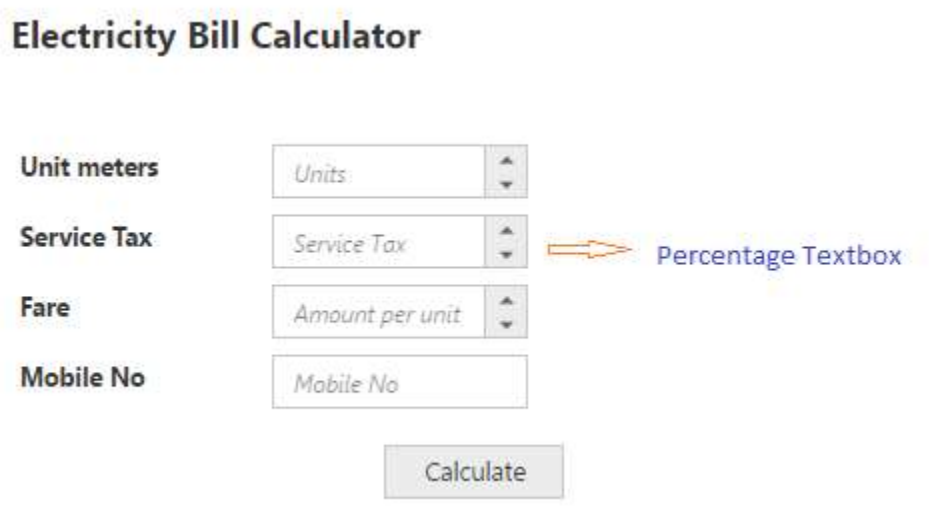
- Min and Max Values—Specifies value range for the textbox.
- Spin Buttons—Enable you to increase or decrease the current value in the textbox.
- Step Value—Enables you to increment or decrement the current value by step value.
- Globalization— EJ MVC PercentageTextBox has globalization support. It use ej.globalize.js to globalize the number format, and parse numbers according to the culture.
- Keyboard Navigation—You can interact with PercentageTextBox by using keyboard.
- RTL Support—Support for right to left alignment of textbox input.
- Decimal Values—You can configure the textbox to accept decimal values.
- Themes—PercentageTextBox consist of 17 built-in themes , and also support custom skins for creating user-defined themes.

Getting Started

This section explains briefly about how to create a EJMVC PercentageTextBox in ASP.NET MVC platform.

Create your first PercentageTextBox in MVC

From the following steps you can learn how to create and use PercentageTextBox in your application. Here we have showcased, a small Electric bill calculator application using EJMVC Editors widgets.The Essential ASP.NET MVC Editors control includes Numeric, Percentage, Currency and MaskEdit textbox controls. This guide helps you to use the wide range of Editors functionalities to complete this application.



Create Editors

ASP.NET MVC Editors renders built-in features like keyboard navigation, min and max range and flexible API's.

Copyright © 2001 - 2020 Syncfusion Inc.

1. Create a MVC Project and add necessary Dll's and Scripts. Refer MVC-Getting Started.
2. Add necessary helper elements to render the Editor components.

ASPX-CS

```
<div class="editors">
<div class="element-text" style="">Electricity Bill Calculator</div>
<br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>    @* NumericTextBox creation with watermark text *@
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units")
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>     @* PercentageTextBox creation with watermark text *@
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
<td>    @* CurrencyTextBox creation with watermark text *@
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per unit")
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>    @* MaskEditTextBox creation with watermark text *@
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone number")
</td>
</tr>
</tbody></table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

3. The following styles are added to arrange the Editors.

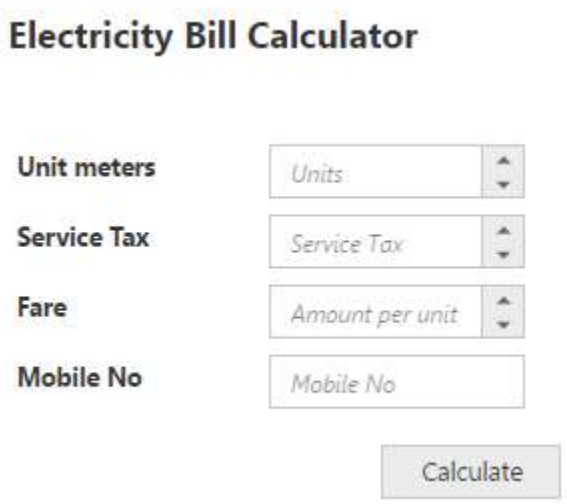
CSS

```
<style type="text/css" class="cssStyles">
.element-text
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
display: inline-block;
font-size: 20px;
font-weight: bolder;
height: 50px;
position: relative;
text-align: center;
top: -20px;
}
.editors
{
max-width: 400px;
border: 2px solid #DDDDDD;
}
.editors table
{
border:0px;
padding-left:50px;
}
.pay-bill
{
margin:15px 0px 10px 208px;
}
.editors label
{
display: block;
width: 130px;
}
</style>
```

4. Execute the code to render Editors as follows



PercentageTextBox with watermark text

Set MinValue, MaxValue and value in PercentageTextBox

You can set the “MinValue”,“MaxValue” and “Value” in Numeric, Percentage and Currency text boxes for maintaining the range in Editors widgets. In this scenario, you have to enter the values between the default ranges and enter the phone number in the MaskEdit widget by using the ”MaskFormat” property. By using DecimalPlaces property for CurrencyTextBox you can get decimal values. The following code example illustrates how to achieve this.

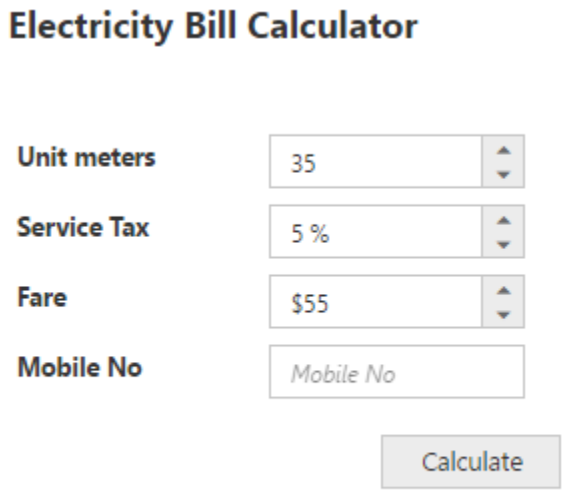
ASPX-CS

```
<div class="editors">
<div class="element-text" style="">Electricity Bill Calculator</div>
<br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units").MinValue(1).M
axValue(10000).Value("70")
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax").MinValue(5).
MaxValue(100).Value("15")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
<td>
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per
unit").MinValue(0.00).MaxValue(100000.00).Value("350.00").DecimalPlaces(2
)
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone
number").MaskFormat("99-999-99999").Value("9090909090")
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
```

The following screenshot illustrates the output of the above code examples.



Setting the Strict Mode Option

You can set the “StrictMode” option to restrict entering values defined outside the range. The following code example illustrates how to set strict mode option.

ASPX-CS

```
<div class="editors">
<div class="element-text" style="">ElectricityBill Calculator</div><br />
<table class="editors">
<tbody>
<tr>
<td>
<label>Unit meters</label>
</td>
<td>
@Html.EJ().NumericTextbox("numeric").WatermarkText("Units").MinValue(1).M
axValue(10000).Value("70").EnableStrictMode(true)
</td>
</tr>
<tr>
<td>
<label>Service Tax</label>
</td>
<td>
@Html.EJ().PercentageTextbox("percent").WatermarkText("Tax").MinValue(5).
MaxValue(100).Value("15")
</td>
</tr>
<tr>
<td>
<label>Fare</label>
</td>
```

```
<td>
@Html.EJ().CurrencyTextbox("currency").WatermarkText("Amount per
unit").MinValue(0.00).MaxValue(100000.00).Value("350.00").DecimalPlaces(2
)
</td>
</tr>
<tr>
<td>
<label>Mobile No</label>
</td>
<td>
@Html.EJ().MaskEdit("maskedit").WatermarkText("Phone
number").MaskFormat("99-999-99999").Value("9090909090")
</td>
</tr>
</tbody>
</table>
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate")
</div>
</div>
```

Run the above code example and you can see that it restricts entering a value exceeding the MinValue and MaxValue range mentioned in the Numeric textbox.

Set Calculation process with PercentageTextBox Widgets

You can use events to calculate the total and displays the value. You can achieve this with the help of Click event in the button widget. The calculation steps are written in the call back function of Click event button.

To customize the button, you can set the ContentType as TextAndImage to include the icon before the text and add the ClientSideEvents for click event.

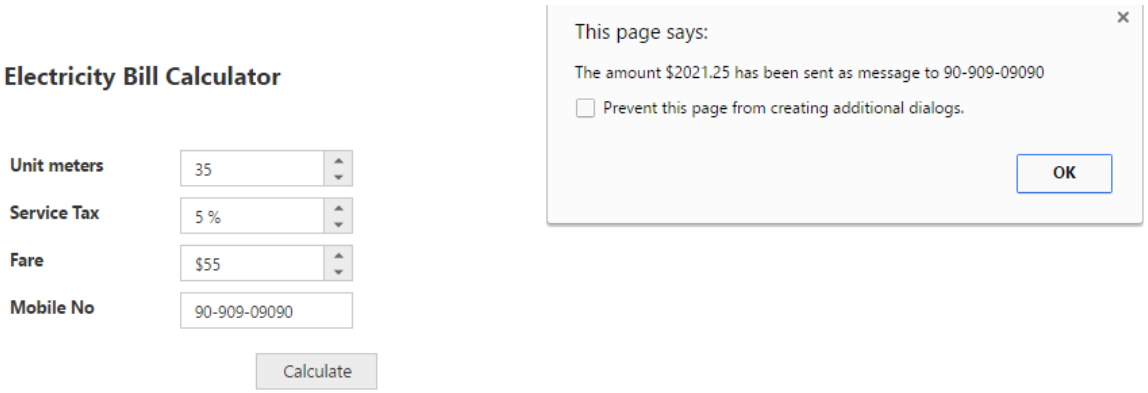
ASPX-CS

```
<div class="editors">
@* Please refer the table format for textboxes customization *@
<div class="pay-bill">
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Calculate").ClientS
ideEvents(c=>c.Click("calculateBill"))
</div>
</div>
<script type="text/javascript">
function calculateBill() {
// Declares Necessary variable creation
var unitMeter, serviceTax, amountPerKm;
unitMeter = $("#numeric").data("ejNumericTextbox");// Object of Numeric
serviceTax = $("#percent").data("ejPercentageTextbox");// Object of
Percentage
amountPerKm = $("#currency").data("ejCurrencyTextbox"); // Object of
Currency
mobileNumber = $("#maskedit").data("ejMaskEdit"); // Object of MaskEdit
// This is used to calculate the Net amount
var netAmount = unitMeter.model.value * amountPerKm.model.value;
// This is used to calculate the service tax amount
var sTax = (netAmount * serviceTax.model.value) / 100;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// This shows the calculated amount for the units
alert("The amount $" + (netAmount + sTax) + " has been sent as message to
" + mobileNumber.model.value + ".");
}
</script>
```

Run the above code sample, fill the required Textbox fields and click the Calculate button. The values are displayed and an alert message is shown. The following screen shot illustrates the final output of the Electricity bill calculator.



- Behavior Settings
- Decimal Places
- Persistence Support
- Strict Mode Support
- Enabled or Disabled
- Adjusting Textbox Size
- Increment Step
- Define Name
- Define Value
- Define maxValue and minValue
- Read Only Support
- Error Visibility
- Appearance
- Theme
- CSS Class
- Rounded Corner Support
- Spin Button Support
- Water Mark Text Support
- Text Alignment Support
- Localization Support
- RTL Support
- Keyboard Interaction

Copyright © 2001 - 2020 Syncfusion Inc.

Behavior Settings

Decimal Places

DecimalPlaces property specifies number of values allowed after the decimal point.The default value of DecimalPlaces property is 0 i.e., by default you cannot specify decimal value in PercentageTextBox. We need to add this property to allow decimal values. To set the DecimalPlaces to “-1”, that allows the decimals without any limit in PercentageTextBox control.

Configure Decimal Places

In the View page add PercentageTextBox helper, and configure the DecimalPlaces property as follows.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").DecimalPlaces(2).Value("22")
```

The output is as follows.



Persistence Support

PercentageTextBox provides state maintenance support. You can maintain the previous changes made in the control after a page loads.

Configure Persistence Support

In the View page add PercentageTextBox helper, and configure the EnablePersistence property as follows.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").Value("22").EnablePersistence(tru
e)
```

Output of PercentageTextBox with EnablePersistence is as follows.





Strict Mode Support

PercentageTextBox allows you to use the strict mode option by setting the EnableStrictMode property. You can set the MinValue and MaxValue to the controls to enable strict mode functionality. Default value of this property is false. When the textbox value exceeds the MaxValue, it restricts the exceeded value and returns the MaxValue. Likewise when the textbox value goes below MinValue, it restricts the new value and returns the MinValue. When this property is enabled, it does not restrict the specified value and an error class is added to indicate wrong value is provided to the textbox.

Configure Strict Mode Support

In the View page add PercentageTextBox helper, and configure the EnableStrictMode property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").MinValue(-
5).MaxValue(3).EnableStrictMode(true).Value(10)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Output when EnableStrictMode is “true” is as follows.



Enabled or Disabled

PercentageTextBox has an option to enable or disable its element. You can set the Enabled property as “false” to enable the Textbox controls.

Also you can enable/disable the **PercentageTextBox** by using enable and disable methods.

Configure Enabled or Disabled

In the View page add PercentageTextBox helper, and configure the Enabled property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").Value("2").Enabled(false)
```

Output when Enabled is “true” and when Enabled is “false”.





Adjusting Textbox Size

PercentageTextBox size can be modified by using the Height and Width properties.

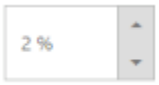
Configure Height and Width

In the View page add PercentageTextBox helper, and configure the Height and Width property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").Value("2").Width("100").Height("5
0")
```

Output of PercentageTextBox after setting “Height” and “Width” is as follows.



Increment Step

The IncrementStep property is used to increase or decrease the amount of value in the Percentage textbox.

Configure Increment Step

In the View page add PercentageTextBox helper, and configure the IncrementStep property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").IncrementStep(3).Value("1")
```

Output of Percentage textbox with IncrementStep is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.





Define Name

When you have placed the PercentageTextBox in a form, the Name property is used to send the field value at form submission. The default value of the Name property is null.

Configure Name

In the View page add PercentageTextBox helper, and configure the Name property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").Name("Percentage")
```

Define Value

The value of PercentageTextBox can be assigned by using the Value property. The default value for Value property is null.

You can get the value of **PercentageTextBox** by using getValue method.

Configure Value

In the View page add PercentageTextBox helper, and configure the Value property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").Value("21")
```

Output of PercentageTextBox with the value property is as follows.



Define MaxValue and MinValue

MaxValue

The maximum limit value can be assigned to the PercentageTextBox by using the MaxValue property. The default value of MaxValue property is 1.7976931348623157e+308.

MinValue

The minimum limit value can be assigned to the PercentageTextBox by using the MinValue property. The default value of MinValue property is -1.7976931348623157e+308.

Configure MaxValue and MinValue

In the View page add PercentageTextBox helper, and configure the MinValue and MaxValue property. .

ASPX-CS

```
@Html.EJ().PercentageTextbox("percent").Value("3").MinValue(-
2).MaxValue(3)
```

Output of PercentageTextBox with MaxValue and MinValue is as follows.



Copyright © 2001 - 2020 Syncfusion Inc.



Read Only Support

PercentageTextBox supports read only option. When you enable the ReadOnly property to the control, the value cannot be changed in the PercentageTextBox. You can set the ReadOnly property as “true” to enable this option.

Configure Read Only

In the View page add PercentageTextBox helper, and configure the ReadOnly property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").Value("2").ReadOnly(true)
```

Output of PercentageTextBox when ReadOnly is “true” is as follows. PercentageTextBox values cannot be edited or changed.



Appearance

Theme

PercentageTextBox control’s style and appearance can be controlled based on CSS classes. In order to apply styles you need to refer 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. If the file ej.web.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.web.all.min.css is the combination of these two.

By default, there are 17 themes support available namely:

- bootstrap
- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- high-contrast-01
- high-contrast-02
- material
- office-365

CSS Class

The CSS can be customized by using the CssClass in the PercentageTextBox. You can customize the PercentageTextBox with CssClass property to appear like your desired control.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure CSS Class

In the View page add PercentageTextBox helper, and configure the CssClass property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").Value("2").CssClass("customCss
")
```

Customize the CSS properties in custom CSS class.

CSS

```
<style>
.customCss .e-box
{
border-color: #9d241b;
}
.customCss .e-input
{
background-color: #f6db8d;
}
.customCss .e-select
{
background-color: #ecf6ac;
border-color: #3c36e7;
}
</style>
```

Output of PercentageTextBox after applying CssClass is as follows.



Rounded Corner Support

PercentageTextBox provides you with rounded corner support whose appearance is different from normal textbox controls.

Configure Rounded Corner Support

In the View page add PercentageTextBox helper, and configure the ShowRoundedCorner property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").Value("2").ShowRoundedCorner(t
rue)
```

Output of PercentageTextBox when ShowRoundedCorner is “true”.



Spin Button Support

PercentageTextBox provides you the option as to whether to display the spin button in the widget or remove it from the control by using showSpinButton property.

Configure Spin Button

In the View page add PercentageTextBox helper, and configure the ShowSpinButton property.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").Value("2").ShowSpinButton(fals
e)
```

Output of PercentageTextBox when ShowSpinButton is “false”.



Water Mark Text Support

The PercentageTextBox provide water mark text support. You can display the initial value in the control by water mark.

Configure Water Mark Text

In the View page add PercentageTextBox helper, and configure the ShowSpinButton property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").WatermarkText("Percentage")
```

Output of PercentageTextBox after applying WaterMarkText is as follows.



Globalization Support

Globalization is language support based on the culture in **Textboxes**. You can achieve the **Globalization** using “**locale”** property in **PercentageTextBox** .

The **PercentageTextBox** widget provides multi-language support using globalization. You can customize the **PercentageTextBox** with your own language style by using this feature. You can change the globalization by using the **locale** property. The default value for **locale** property is **en-US** in **PercentageTextBox** control.

More than 350 culture specific files are available to localize the value. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: All the culture-specific script files are available within the below specified location, once you have installed Essential Studio in your machine, therefore it is not necessary to download these files explicitly.

'(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n'
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

Refer the below German culture file in head section of HTML page after the reference of **ej.web.all.min.js** file.

JAVASCRIPT

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion }}/js/i18n/ej.culture.de-DE.min.js"></script>
```

You can dynamically change the language based on their culture.

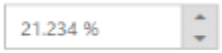
Configure Localization

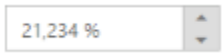
The following example describes the way to use localization in PercentageTextBox.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").Value("21234").Locale("de-DE")
```

Output of PercentageTextBox with localization.





RTL Support

PercentageTextBox provides RTL (Right-To-Left) support. The alignment of PercentageTextBox can be changed from Left-To-Right into Right-To-Left.

Enable RTL

In the View page add PercentageTextBox helper, and configure the EnableRTL property.

ASPX-CS

```
@Html.EJ().PercentageTextbox("percentage").Value("22").EnableRTL(true)
```

Output of PercentageTextBox when EnableRTL is “true” is as follows.



Keyboard Interaction

With the keyboard navigation enabled in the PercentageTextBox control, it is possible to control the actions with the provided shortcut keys. Almost all the PercentageTextBox functionality that are done through mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the PercentageTextBox control are discussed in the following table.

Shortcut Key	Description
Access key + j	Focuses the control
Up	Increments the value
Down	Decrements the value
Tab	Focus the next element

Copyright © 2001 - 2020 Syncfusion Inc.

Configuring Keyboard Navigation

The following steps explain the implementation of keyboard interaction in PercentageTextBox.

In the View page add PercentageTextBox helper. Set the access key property to the PercentageTextBox for focusing the control while key is pressed. We need to use 'HtmlAttributes' property to add the "access key" attribute to the Editor elements. 'HtmlAttributes' property is used to add HTML attributes like, id, class etc.. to the components. We need to use IDictionary<string,object> to specify the HTML attributes.

ASPX-CS

```
@{IDictionary<string, object> percentAttribute = new Dictionary<string,
object>();
percentAttribute.Add("accesskey", "k");
}
@Html.EJ().PercentageTextbox("percentage").Name("percent").Value("22").Ht
mlAttributes(percentAttribute)
```

Run the above example and press Access key + j key to focus the PercentageTextBox widget. Perform provided functionality by using keyboard shortcuts.



PivotGrid

Overview

PivotGrid control is an easily configurable, presentation-quality business control that reads both OLAP and Relational datasource, allows to create multi-dimensional views for analysis and satisfying business user needs.

Some of the key feature available in PivotGrid are listed below:

1. Data Binding: Supports both OLAP and Relational datasource data binding in the same environment.
2. PivotTable Field List: Lists the entire datasource bound to PivotGrid and allows UI interaction such as filtering and drag drop operation of any measure, dimension, hierarchy, level and field at runtime.
3. Grouping Bar: Supports UI interaction at runtime through sorting, filtering and drag drop operation of the existing field items bound through report.
4. Grid Layout: Four different layouts such as normal, top summary, no summary and excel-like layout are supported.
5. Export: PivotGrid can be exported to Word, PDF, CSV and Excel documents.
6. PivotGrid control specific features such as hyperlink, cell selection, conditional formatting, RTL, globalization and localization are supported as well.

Getting Started

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotGrid and Relational datasource (Client Mode)

This section covers the information that you need to know to populate a simple PivotGrid with Relational data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGridDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
```

```
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Exprot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS References

The scripts and style sheets that are mandatorily required to render PivotGrid control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Scripts and style sheets are referred under the head tag in _Layout.cshtml file which is found inside Views > Shared folder.

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotGrid

Before initializing, empty the contents of "Index.cshtml" file under Views > Home folder and add the following codes.

HTML

```
@using Syncfusion.JavaScript;
<div>
@Html.EJ().Pivot().PivotGrid("PivotGrid1")
</div>
```

Populate PivotGrid With Data

Let us now see how to populate the PivotGrid control using a sample JSON data as shown below.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad"))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = [
{ Amount: 100, Country: "Canada", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Alberta" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "British Columbia" },
{ Amount: 300, Country: "Canada", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Brunswick" },
{ Amount: 150, Country: "Canada", Date: "FY 2008", Product: "Bike",
Quantity: 3, State: "Manitoba" },
{ Amount: 200, Country: "Canada", Date: "FY 2006", Product: "Car",
Quantity: 4, State: "Ontario" },
{ Amount: 100, Country: "Canada", Date: "FY 2007", Product: "Van",
Quantity: 1, State: "Quebec" },
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ Amount: 200, Country: "France", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Charente-Maritime" },
{ Amount: 250, Country: "France", Date: "FY 2006", Product: "Van",
Quantity: 4, State: "Essonne" },
{ Amount: 300, Country: "France", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Garonne (Haute)" },
{ Amount: 150, Country: "France", Date: "FY 2008", Product: "Van",
Quantity: 2, State: "Gers" },
{ Amount: 200, Country: "Germany", Date: "FY 2006", Product: "Van",
Quantity: 3, State: "Bayern" },
{ Amount: 250, Country: "Germany", Date: "FY 2007", Product: "Car",
Quantity: 3, State: "Brandenburg" },
{ Amount: 150, Country: "Germany", Date: "FY 2008", Product: "Car",
Quantity: 4, State: "Hamburg" },
{ Amount: 200, Country: "Germany", Date: "FY 2008", Product: "Bike",
Quantity: 4, State: "Hessen" },
{ Amount: 150, Country: "Germany", Date: "FY 2007", Product: "Van",
Quantity: 3, State: "Nordrhein-Westfalen" },
{ Amount: 100, Country: "Germany", Date: "FY 2005", Product: "Bike",
Quantity: 2, State: "Saarland" },
{ Amount: 150, Country: "United Kingdom", Date: "FY 2008", Product:
"Bike", Quantity: 5, State: "England" },
{ Amount: 250, Country: "United States", Date: "FY 2007", Product: "Car",
Quantity: 4, State: "Alabama" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product: "Van",
Quantity: 4, State: "California" },
{ Amount: 100, Country: "United States", Date: "FY 2006", Product:
"Bike", Quantity: 2, State: "Colorado" },
{ Amount: 150, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "New Mexico" },
{ Amount: 200, Country: "United States", Date: "FY 2005", Product:
"Bike", Quantity: 4, State: "New York" },
{ Amount: 250, Country: "United States", Date: "FY 2008", Product: "Car",
Quantity: 3, State: "North Carolina" },
{ Amount: 300, Country: "United States", Date: "FY 2007", Product: "Van",
Quantity: 4, State: "South Carolina" }
];
}
</script>
```

The JSON data is set to the **“data”** property present inside the **“DataSource”** object. **“DataSource”** object allows us to set both datasource as well as the fields that needs to be displayed in the row, column, value and filter section of the PivotGrid control.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();}).Columns(column
s => { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();}))
```

The above code will generate a simple PivotGrid with "Country" field in Row, "Product" field in Column, and "Amount" field in Value section.

Copyright © 2001 - 2020 Syncfusion Inc.

	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
Canada	250	500	300	1050
France	200	300	400	900
Germany	300	400	350	1050
United Kingdom	150			150
United States	300	650	500	1450
Grand Total	1200	1850	1550	4600

Apply Sorting

You can sort a field either to ascending or descending order using the "sortOrder" property. Sorting is applicable only for Row and Column fields. By default, fields are arranged in ascending order.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows =>
{rows.FieldName("Country").FieldCaption("Country").SortOrder(SortOrder.As
cending).Add();}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();}))
```

	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
United States	300	650	500	1450
United Kingdom	150			150
Germany	300	400	350	1050
France	200	300	400	900
Canada	250	500	300	1050
Grand Total	1200	1850	1550	4600

Copyright © 2001 - 2020 Syncfusion Inc.

Sort Row/Column by Date

You can sort a field either in ascending or descending order according for date type by using the **SortOrder** property. Sorting is applicable only for Row and Column fields. By default, fields are arranged in ascending order.

Note: To apply sorting by date, you need to specify the Format and FormatString in the field.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows =>
{rows.FieldName("Date").FieldCaption("Date").Format("date").FormatString(
"dd-MM-yyyy").SortOrder(SortOrder.Descending).Add();}).Columns(columns =>
{
columns.FieldName("Day").FieldCaption("Day").Format("date").FormatString(
"ddd").SortOrder(SortOrder.Ascending).Add(); }).Values(values => {
values.FieldName("Amount").Add();}))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = [
{ Amount: 100, Date: "5-1-2017", Day: "Wednesday" },
{ Amount: 200, Date: "1-2-2017", Day: "Sunday" },
{ Amount: 300, Date: "1-1-2018", Day: "Thursday" },
{ Amount: 150, Date: "5-1-2018", Day: "Wednesday" },
{ Amount: 200, Date: "1-2-2017", Day: "Thursday" },
{ Amount: 100, Date: "1-1-2018", Day: "Sunday" },
{ Amount: 200, Date: "5-1-2017", Day: "Wednesday" },
{ Amount: 250, Date: "1-2-2017", Day: "Sunday" }
//....
];
}
</script>
```

	Sunday	Wednesday	Thursday	Grand Total
	Amount	Amount	Amount	Amount
5-1-2018		150		150
1-1-2018	100		300	400
1-2-2017	450		200	650
5-1-2017		300		300
Grand Total	550	450	500	1500

Apply Filtering

Filtering option allows you to specify a set of values that either need to be displayed or hidden. Also filtering option is applicable only for Row, Column and Filter areas.

“FilterItems” object allow us to apply filtering to the fields using the following properties:

Copyright © 2001 - 2020 Syncfusion Inc.

- FilterType - indicates whether the values should be included or excluded.
- Values - specify an array of values that needs to be included or excluded within the particular field.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").FilterItems(filter => {
filter.FilterType(PivotFilterType.Exclude).Values(value => {
value.Add("United Kingdom"); }); }).Add(); }).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add(); }))
```

	Bike	Car	Grand Total
	Amount	Amount	Amount
Canada	250	500	**750**
France	200	300	**500**
Germany	300	400	**700**
United States	300	650	**950**
Grand Total	**1050**	**1850**	**2900**

Apply Summary Types

Allows us to specify the required summary type that PivotGrid should use in its summary cells. **"Sum"** is the default summary type. Following are the summary types that are supported:

- Sum
- Average
- Count
- Min
- Max

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").SummaryType(PivotSummaryType.Average).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
values.FieldName("Quantity").FieldCaption("Quantity").SummaryType(PivotSu
mmaryType.TotalSum).Add(); }))
```

	Bike		Car		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	125	5	250	8	187.5	13
France	200	2	300	3	250	5
Germany	150	6	200	7	175	13
United States	150	6	212.5	10	181.25	16
Grand Total	156.25	19	240.63	28	198.44	47

Creating a simple application with PivotGrid and Relational datasource (Server Mode)

This section covers the information required to create a simple PivotGrid bound to Relational datasource.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGridDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referenced assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.5">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Linq.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Olap.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Pdf.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.XlsIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.DocIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotGrid control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the <head> tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

HTML

```
@using Syncfusion.JavaScript;
<div>
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/Relational")
)
</div>
```

The **“Url”** property in PivotGrid control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotGrid control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Relational". If WCF service is used as endpoint, the URL would look like "/RelationalService.svc".

WebAPI

Adding a WebAPI Controller

To add a WebAPI controller in an existing MVC Web Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WebAPI Controller Class** and name it as **“RelationalController.cs”**, click **Add**.

Now, WebAPI controller is added to the application successfully with the file **“RelationalController.cs”**.

Note: While adding WebAPI Controller Class, name it with the suffix ‘Controller’ which is mandatory. For example, in this demo the controller is named as “RelationalController”.

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside RelationalController.cs file.

C#

```
namespace PivotGridDemo
{
public class RelationalController: ApiController
{
}
}
```

Adding the List of Namespaces

Copyright © 2001 - 2020 Syncfusion Inc.

The following are the list of namespaces to be added on top of the main class inside RelationalController.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlServerCe;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Http;
using System.Web.Script.Serialization;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
namespace PivotGridDemo
{
public class RelationalController : ApiController
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube, PivotGrid and JavaScriptSerializer instances are created immediately inside the main class in **RelationalController.cs** file.

C#

```
namespace PivotGridDemo {
.....
.....
internal class ProductSales {
public string Product {
get;
set;
}
public string Date {
get;
set;
}
public string Country {
get;
set;
}
public string State {
get;
set;
}
public int Quantity {
get;
set;
}
public double Amount {
get;
```

```
set;
}
public static ProductSalesCollection GetSalesData() {
/// Geography
string[] countries = new string[] {
"Australia",
"Canada",
"France",
"Germany",
"United Kingdom",
"United States"
};
string[] ausStates = new string[] {
"New South Wales",
"Queensland",
"South Australia",
"Tasmania",
"Victoria"
};
string[] canadaStates = new string[] {
"Alberta",
"British Columbia",
"Brunswick",
"Manitoba",
"Ontario",
"Quebec"
};
string[] franceStates = new string[] {
"Charente-Maritime",
"Essonne",
"Garonne (Haute)",
"Gers",
};
string[] germanyStates = new string[] {
"Bayern",
"Brandenburg",
"Hamburg",
"Hessen",
"Nordrhein-Westfalen",
"Saarland"
};
string[] ukStates = new string[] {
"England"
};
string[] ussStates = new string[] {
"New York",
"North Carolina",
"Alabama",
"California",
"Colorado",
"New Mexico",
"South Carolina"
};
/// Time
string[] dates = new string[] {
"FY 2005",
"FY 2006",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
"FY 2007",
"FY 2008",
"FY 2009"
};
/// Products
string[] products = new string[] {
"Bike",
"Van",
"Car"
};
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++) {
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product) {
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country) {
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
case "France":
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString() {
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection: List < ProductSales > {}
}
}
```

Service methods in WebAPI Controller

Define the service methods inside RelationalController class, found inside RelationalController.cs file, created while adding WebAPI Controller Class to the Application.

C#

```
namespace PivotGridDemo
{
public class RelationalController: ApiController
{
PivotGrid htmlHelper = new PivotGrid();
JavaScriptSerializer serializer = new JavaScriptSerializer();
Dictionary<string, object> dict = new Dictionary<string, object>();
static int cultureIDInfoval = 1033;
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
[System.Web.Http.ActionName("InitializeGrid")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeGrid(Dictionary<string,
object> jsonResult)
{
htmlHelper.PivotReport = BindDefaultData();
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData());
return dict;
}
[System.Web.Http.ActionName("FetchMembers")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> FetchMembers(Dictionary<string, object>
jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["headerTag"].ToString(),
jsonResult["sortedHeaders"].ToString());
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return dict;
}
[System.Web.Http.ActionName("Filtering")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> Filtering(Dictionary<string, object>
jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["filterParams"].ToString(),
jsonResult["sortedHeaders"].ToString());
return dict;
}
[System.Web.Http.ActionName("NodeStateModified")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> NodeStateModified(Dictionary<string,
object> jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["headerTag"].ToString(),
jsonResult["dropAxis"].ToString(), jsonResult["filterParams"].ToString(),
jsonResult["sortedHeaders"].ToString());
return dict;
}
[System.Web.Http.ActionName("NodeDropped")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> NodeDropped(Dictionary<string, object>
jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["dropAxis"].ToString(),
jsonResult["headerTag"].ToString(),
jsonResult.ContainsKey("filterParams") ?
jsonResult["filterParams"].ToString() : null,
jsonResult["sortedHeaders"].ToString());
return dict;
}
[System.Web.Http.ActionName("Sorting")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> Sorting(Dictionary<string, object>
jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["sortedHeaders"].ToString());
return dict;
}
[System.Web.Http.ActionName("CalculatedField")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> CalculatedField(Dictionary<string,
object> jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), null, jsonResult["headerTag"].ToString());
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return dict;
}
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
System.Web.HttpContext.Current.Response);
}
[System.Web.Http.ActionName("SaveReport")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> SaveReport(Dictionary<string, object>
jsonResult)
{
string mode = jsonResult["operationalMode"].ToString();
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", jsonResult["reportName"].ToString());
if (mode == "serverMode")
cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(jsonResult["clientReports"].ToString()).T
oArray());
else if (mode == "clientMode")
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(jsonResult["clientReports"].ToString()).ToArray())
;
cmd1.ExecuteNonQuery();
con.Close();
return null;
}
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
byte[] reportString = new byte[2 * 1024];
PivotReport report = new PivotReport();
var reports = "";
string mode = jsonResult["operationalMode"].ToString();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && jsonResult.ContainsKey("clientReports"))
{
reports = jsonResult["clientReports"].ToString();
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
if (mode == "clientMode")
{
reportString = (row.ItemArray[1] as byte[]);
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
report = htmlHelper.DeserializedReports(reports);
htmlHelper.PivotReport = report;
dictionary = htmlHelper.GetJsonData("loadOperation",
ProductSales.GetSalesData(), "Load Report",
jsonResult["reportName"].ToString());
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
[System.Web.Http.ActionName("DeferUpdate")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DeferUpdate(Dictionary<string, object>
jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), null, null, null,
jsonResult["sortedHeaders"].ToString(),
jsonResult["filterParams"].ToString());
return dict;
}
[System.Web.Http.ActionName("CellEditing")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> CellEditing(Dictionary<string, object>
jsonResult)
{
htmlHelper.PopulateData(jsonResult["currentReport"].ToString());
dict = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
ProductSales.GetSalesData(), jsonResult["index"].ToString(),
jsonResult["summaryValues"].ToString(),
jsonResult["valueHeaders"].ToString());
return dict;
}
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Product",
FieldHeader = "Product", TotalHeader = "Total" });
pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName =
"Country", FieldHeader = "Country", TotalHeader = "Total" });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
return pivotSetting;
}
}
.....
..... // Initialize the datasource
.....
}
```

Configure routing in WebAPIConfig Class

Open the "WebAPIConfig.cs" file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Now, **PivotGrid** will be rendered with Sales Amount over a set of products across different customer geographic locations.

	Canada	France	Germany	United Kingdom	United States	Grand Total
	Amount	Amount	Amount	Amount	Amount	Amount
Bike	$51,651,000.00	$44,167,500.00	$49,383,900.00	$46,314,000.00	$44,398,800.00	**$235,915,200.00**
Car	$5,966,100.00	$5,774,100.00	$5,315,100.00	$5,860,500.00	$6,155,400.00	**$29,071,200.00**
Van	$14,252,400.00	$15,807,600.00	$11,985,000.00	$11,008,200.00	$14,601,600.00	**$67,654,800.00**
Grand Total	**$71,869,500.00**	**$65,749,200.00**	**$66,684,000.00**	**$63,182,700.00**	**$65,155,800.00**	**$332,641,200.00**

WCF

This section demonstrates the utilization of WCF service as endpoint binding Relational datasource to a simple PivotGrid. For more details on this topic, click here.

Getting Started

Important

Starting with v16.2.0.x, if you refer to Syncfusion assemblies from trial setup or from the NuGet feed, include a license key in your projects. Refer to this link to learn about registering Syncfusion license key in your ASP.NET Core application to use our components.

Creating a simple application with PivotGrid and OLAP datasource (Client Mode)

This section covers the information that you need to know to populate a simple PivotGrid with OLAP data completely on the client-side.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGridDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select Add Reference. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages

Copyright © 2001 - 2020 Syncfusion Inc.

MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referred assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS References

The scripts and style sheets that are mandatorily required to render PivotGrid control in a MVC Web Application are mentioned in an appropriate order below:

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Scripts and style sheets are referred under the head tag in _Layout.cshtml file which is found inside Views > Shared folder.

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

ASPX-CS

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Initialize PivotGrid

Before initializing, empty the contents of "Index.cshtml" file under Views > Home folder and add the following codes.

HTML

```
@using Syncfusion.JavaScript;
<div>
@Html.EJ().Pivot().PivotGrid("PivotGrid1")
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Populate PivotGrid With DataSource

Initializes the OLAP datasource for PivotGrid control as shown below.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();}).Data("https://bi.syncfusion.com/olap/ms
mdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works"))
```

The above code will generate a simple PivotGrid with "Fiscal" field in Row, "Customer Geography" field in Column and "Internet Sales Amount" field in Value section.

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	**$7,072,084.24**
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	**$5,762,134.30**
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	**$16,473,618.05**
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	**$50,840.63**
Total	**$9,061,000.58**	**$1,977,844.86**	**$2,644,017.71**	**$2,894,312.34**	**$3,391,712.21**	**$9,389,789.51**	**$29,358,677.22**

Creating a simple application with PivotGrid and OLAP datasource (Server Mode)

This section covers the information required to create a simple PivotGrid bound to OLAP datasource.

Note: ASP.NET MVC Web Application will contain a service that transfers data to server-side, processes and returns back to client-side for control rendering and re-rendering. The service utilized for communication could be either WCF or WebAPI based on user requirement.

Project Initialization

Create a new **ASP.NET MVC Web Application** using Visual Studio IDE and name the project as **"PivotGridDemo"**.

Select the View engine as **'Razor'** and Project template as **'Internet Application'** and finally click **OK** button to create an application.

Now add the following dependency libraries as references into your MVC Web Application. In order to add them to your application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries will be found.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then

Copyright © 2001 - 2020 Syncfusion Inc.

the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

The version of Syncfusion libraries based on .NET Framework and MVC version are classified below. For example, version is illustrated as,

MVC Version	MVC Version of Syncfusion assembly	Base Version of Syncfusion assembly	System.Web.Mvc	System.Web.WebPages
MVC3	{{ site.mvc3releaseversion }}	{{ site.35esreleaseversion }}	3.0	1.0
MVC4	{{ site.mvc4releaseversion }}	{{ site.40esreleaseversion }}	4.0	2.0
MVC5	{{ site.mvc5releaseversion }}	{{ site.45esreleaseversion }}	5.0	3.0

Register the referenced assemblies in "Web.config" files available inside Views folder and also at the root of the application.

XML

```
<compilation debug="true" targetFramework="4.0">
<assemblies>
……
……
<add assembly="Syncfusion.EJ, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Pivot, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Export, Version= {{ site.40esreleaseversion }}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<add assembly="Syncfusion.EJ.Mvc, Version= {{ site.40esreleaseversion }},
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Linq.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Olap.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Compression.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.PivotAnalysis.Base, Version= {{
site.40esreleaseversion }}, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.Pdf.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.XlsIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.DocIO.Base, Version= {{ site.40esreleaseversion
}}, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Register the required namespaces in "Web.config" files available inside Views folder and also at the root of the application

XML

```
<namespaces>
……
……
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.JavaScript" />
</namespaces>
```

Note: Registering assemblies and namespaces earlier helps to include the control in view page with the help of intellisense.

Set the **UnobtrusiveJavaScriptEnabled** property to false under **appSettings** tag in "Web.config" file at the root folder.

XML

```
<configuration>
……
……
<appSettings>
……
……
<add key="UnobtrusiveJavaScriptEnabled" value="false" />
</appSettings>
</configuration>
```

Scripts and CSS Initialization

The scripts and style sheets that are mandatorily required to render PivotGrid control in a MVC Web Application are mentioned in an appropriate order below:

Copyright © 2001 - 2020 Syncfusion Inc.

1. ej.web.all.min.css
2. jQuery-3.0.0.min.js
3. ej.web.all.min.js

Click here here to know more about scripts and style sheets available online (CDN Link).

Scripts and style sheets are referred under the **head** tag in **_Layout.cshtml** file which is found inside **Views > Shared folder.**

HTML

```
<head>
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" type="text/css"
/>
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
3.0.0.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
</head>
```

The script manager is initialized immediately after the RenderBody() function call in **_Layout.cshtml** file in-order to generate control related scripts.

HTML

```
<body>
……
……
@RenderBody()
@(Html.EJ().ScriptManager())
</body>
```

Control Initialization

Before initializing, empty the contents of **Index.cshtml** file under **Views > Home** folder and add the following codes. Register the namespaces at the top of the page and then add the control.

HTML

```
@using Syncfusion.JavaScript;
<div>
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/Olap"))
</div>
```

The **“Url”** property in PivotGrid control points the service endpoint, where data are processed and fetched in the form of JSON. The services used in PivotGrid control as endpoint are WCF and WebAPI.

Note: The above "Index.cshtml" contains WebAPI URL, which is "/Olap". If WCF service is used as endpoint, the URL would look like "/OlapService.svc".

WebAPI

Adding a WebAPI Controller

Copyright © 2001 - 2020 Syncfusion Inc.

To add a WebAPI controller in an existing Web Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select **WebAPI Controller Class** and name it as OlapController.cs, click Add.

Now, WebAPI controller is added to the application successfully with the file **"OlapController.cs"**.

Note: While adding WebAPI Controller Class, name it with the suffix 'Controller' that is mandatory. For example, in this demo the controller is named as **"OlapController"**.

Next, remove all the existing methods such as "Get", "Post", "Put" and "Delete" present inside OlapController.cs file.

C#

```
namespace PivotGridDemo
{
public class OlapController: ApiController
{
}
}
```

Adding the List of Namespaces

The following are the list of namespaces to be added on top of the main class inside OlapController.cs file.

C#

```
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Linq;
using System.Web.Http;
using System.Web.Script.Serialization;
using Syncfusion.JavaScript.Olap;
using System.Web;
using System.Data.SqlServerCe;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
using System.Data;
using System.Text;
namespace PivotGridDemo
{
public class OlapController: ApiController
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube, PivotGrid and JavaScriptSerializer instances are created immediately inside the main class in OlapController.cs file.

C#

```
namespace PivotGridDemo
```

```
{
public class OlapController: ApiController
{
Syncfusion.JavaScript.PivotGrid htmlHelper = new
Syncfusion.JavaScript.PivotGrid();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
JavaScriptSerializer serializer = new JavaScriptSerializer();
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
//Other codes
}
}
```

Service methods in WebAPI Controller

Define the service methods inside OlapController class, found inside OlapController.cs file, created while adding WebAPI Controller Class to the Application.

C#

```
namespace PivotGridDemo
{
public class OlapController: ApiController
{
Syncfusion.JavaScript.PivotGrid htmlHelper = new
Syncfusion.JavaScript.PivotGrid();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
static int cultureIDInfoval = 1033;
JavaScriptSerializer serializer = new JavaScriptSerializer();
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
[System.Web.Http.ActionName("InitializeGrid")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeOlapGrid(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = null;
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
DataManager.OverrideDefaultFormatStrings = true;
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult.ContainsKey("gridLayout") ?
jsonResult["gridLayout"].ToString() : null,
Convert.ToBoolean(jsonResult["enablePivotFieldList"].ToString()));
}
[System.Web.Http.ActionName("DrillGrid")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DrillGrid(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager = new OlapDataManager(connectionString);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
connectionString, DataManager, jsonResult["cellPosition"].ToString(),
jsonResult["headerInfo"].ToString(), jsonResult["layout"].ToString());
}
[System.Web.Http.ActionName("NodeDropped")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> NodeDropped(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["dropType"].ToString(),
jsonResult["nodeInfo"].ToString(), jsonResult.ContainsKey("filterParams")
? jsonResult["filterParams"].ToString() : null,
jsonResult["gridLayout"].ToString(), true);
}
[System.Web.Http.ActionName("Filtering")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> Filtering(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
connectionString, DataManager, null,
jsonResult["filterParams"].ToString(),
jsonResult["gridLayout"].ToString());
}
[System.Web.Http.ActionName("FetchMembers")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> FetchMembers(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, null, jsonResult["headerTag"].ToString());
}
[System.Web.Http.ActionName("RemoveButton")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> RemoveButton(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
connectionString, DataManager, null, jsonResult["headerInfo"].ToString(),
jsonResult["gridLayout"].ToString());
}
[System.Web.Http.ActionName("MemberExpanded")]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
[System.Web.Http.HttpPost]
public Dictionary<string, object> MemberExpanded(Dictionary<string,
object> jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (!string.IsNullOrEmpty(jsonResult["currentReport"].ToString()))
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, Convert.ToBoolean(jsonResult["checkedStatus"].ToString()),
jsonResult["parentNode"].ToString(), jsonResult["tag"].ToString(),
jsonResult["cubeName"].ToString());
}
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
[System.Web.Http.ActionName("SaveReport")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> SaveReport(Dictionary<string, object>
jsonResult)
{
string mode = jsonResult["operationalMode"].ToString();
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", jsonResult["reportName"].ToString());
if (mode == "clientMode")
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(jsonResult["clientReports"].ToString()).ToArray())
;
else if (mode == "serverMode")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(jsonResult["clientReports"].ToString()).T
oArray());
cmd1.ExecuteNonQuery();
con.Close();
return null;
}
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
{
string mode = jsonResult["operationalMode"].ToString();
byte[] reportString = new byte[4 * 1024];
var reports = "";
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && jsonResult.ContainsKey("clientReports"))
{
reports = jsonResult["clientReports"].ToString();
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
if (mode == "clientMode")
{
reportString = row.ItemArray[1] as byte[];
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
dynamic customData =
serializer.Deserialize<dynamic>(jsonResult["customObject"].ToString());
var cultureIDInfo = new System.Globalization.CultureInfo(("en-US")).LCID;
if (customData is Dictionary<string, object> &&
customData.ContainsKey("Language"))
{
cultureIDInfo = new
System.Globalization.CultureInfo((customData["Language"])).LCID;
}
connectionString = connectionString.Replace("" + cultureIDInfoval + "",
"" + cultureIDInfo + "");
cultureIDInfoval = cultureIDInfo;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DataManager.Culture = new
System.Globalization.CultureInfo((cultureIDInfo));
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(report
s));
DataManager.OverrideDefaultFormatStrings = true;
dictionary = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["gridLayout"].ToString(),
Convert.ToBoolean(jsonResult["enablePivotFieldList"].ToString()));
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
[System.Web.Http.ActionName("DeferUpdate")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> DeferUpdate(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(jsonResult["curr
entReport"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, null, jsonResult["filterParams"].ToString());
}
[System.Web.Http.ActionName("Paging")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> Paging(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(htmlHelper.SetPaging(jsonResult["currentRepo
rt"].ToString(), jsonResult["pagingInfo"].ToString()));
return htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["layout"].ToString());
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.CurrentCubeName = "Adventure Works";
MeasureElements measureElement = new MeasureElements();
measureElement.Elements.Add(new MeasureElement { UniqueName =
"[Measures].[Internet Sales Amount]" });
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
DimensionElement dimensionElementColumn = new DimensionElement();
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");
olapReport.SeriesElements.Add(dimensionElementRow);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(measureElement);
return olapReport;
}
}
}
```

Configure routing in WebAPIConfig Class

Open the "WebAPIConfig.cs" file found in **App_Start** folder. Then routing could be configured as shown in the following code example.

C#

```
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
```

Now, **PivotGrid** is rendered with Customer Count over a period of fiscal years across different customer geographic locations.

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	**$7,072,084.24**
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	**$5,762,134.30**
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	**$16,473,618.05**
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	**$50,840.63**
Total	**$9,061,000.58**	**$1,977,844.86**	**$2,644,017.71**	**$2,894,312.34**	**$3,391,712.21**	**$9,389,789.51**	**$29,358,677.22**

WCF

This section demonstrates the utilization of WCF service as endpoint binding OLAP datasource to a simple PivotGrid. For more details on this topic, click here.

Data Binding

Data Binding

Binding PivotGrid to Collection

This section demonstrates binding of a collection to the PivotGrid control as datasource. For more information on this datasource refer the following links.

If you are using WebAPI controller, refer the "Datasource Initialization" section under the following link.

If you are using WCF Service, refer the "Datasource Initialization" section below.

Copyright © 2001 - 2020 Syncfusion Inc.

WCF

Adding a WCF Service

To add a WCF service in an existing MVC Application, right-click on the project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as RelationalService.svc, click **Add.**

Now, WCF service is added into the application successfully that comprises of the following files. The utilization of these files is explained in the immediate sections

- RelationalService.svc
- RelationalService.svc.cs
- IRelationalService.cs

Configuring WCF Service Class

Remove the **"DoWork"** method present inside both RelationalService.svc.cs and IRelationalService.cs files. Next, add **"AspNetCompatibilityRequirements"** attribute on top of main class present inside the RelationalService.svc.cs file and set **"RequirementsMode"** value to **"Allowed"**.

C#

```
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Note: If you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside RelationalService.svc.cs file.

C#

```
using Syncfusion.JavaScript;
using Syncfusion.PivotAnalysis.Base;
using System;
using System.Collections.Generic;
using System.Data;
using System.ServiceModel.Activation;
using System.Data.SqlServerCe;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.Http;
using System.Web.Script.Serialization;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
}
}
```

Datasource Initialization

Now the connection string to connect OLAP Cube, PivotGrid and JavaScriptSerializer instances are created immediately inside the main class in RelationalService.svc.cs file.

C#

```
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService : IRelationalService
{
……
……
}
internal class ProductSales
{
public string Product { get; set; }
public string Date { get; set; }
public string Country { get; set; }
public string State { get; set; }
public int Quantity { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public double Amount { get; set; }
public static ProductSalesCollection GetSalesData()
{
/// Geography
string[] countries = new string[] { "Australia", "Canada", "France",
"Germany", "United Kingdom", "United States" };
string[] ausStates = new string[] { "New South Wales", "Queensland",
"South Australia", "Tasmania", "Victoria" };
string[] canadaStates = new string[] { "Alberta", "British Columbia",
"Brunswick", "Manitoba", "Ontario", "Quebec" };
string[] franceStates = new string[] { "Charente-Maritime", "Essonne",
"Garonne (Haute)", "Gers", };
string[] germanyStates = new string[] { "Bayern", "Brandenburg",
"Hamburg", "Hessen", "Nordrhein-Westfalen", "Saarland" };
string[] ukStates = new string[] { "England" };
string[] ussStates = new string[] { "New York", "North Carolina",
"Alabama", "California", "Colorado", "New Mexico", "South Carolina" };
/// Time
string[] dates = new string[] { "FY 2005", "FY 2006", "FY 2007", "FY
2008", "FY 2009" };
/// Products
string[] products = new string[] { "Bike", "Van", "Car" };
Random r = new Random(123345345);
int numberOfRecords = 2000;
ProductSalesCollection listOfProductSales = new ProductSalesCollection();
for (int i = 0; i < numberOfRecords; i++)
{
ProductSales sales = new ProductSales();
sales.Country = countries[r.Next(1, countries.GetLength(0))];
sales.Quantity = r.Next(1, 12);
/// 1 percent discount for 1 quantity
double discount = (30000 * sales.Quantity) *
(double.Parse(sales.Quantity.ToString()) / 100);
sales.Amount = (30000 * sales.Quantity) - discount;
sales.Date = dates[r.Next(r.Next(dates.GetLength(0) + 1))];
sales.Product = products[r.Next(r.Next(products.GetLength(0) + 1))];
switch (sales.Product)
{
case "Car":
{
sales.Date = "FY 2005";
break;
}
}
switch (sales.Country)
{
case "Australia":
{
sales.State = ausStates[r.Next(ausStates.GetLength(0))];
break;
}
case "Canada":
{
sales.State = canadaStates[r.Next(canadaStates.GetLength(0))];
break;
}
case "France":
```

```
{
sales.State = franceStates[r.Next(franceStates.GetLength(0))];
break;
}
case "Germany":
{
sales.State = germanyStates[r.Next(germanyStates.GetLength(0))];
break;
}
case "United Kingdom":
{
sales.State = ukStates[r.Next(ukStates.GetLength(0))];
break;
}
case "United States":
{
sales.State = ussStates[r.Next(ussStates.GetLength(0))];
break;
}
}
listOfProductSales.Add(sales);
}
return listOfProductSales;
}
public override string ToString()
{
return string.Format("{0}-{1}-{2}", this.Country, this.State,
this.Product);
}
public class ProductSalesCollection : List<ProductSales>
{
}
}
}
```

Service methods in WCF Service

First, declare the service methods inside **IRelationalService** interface, found in IRelationalService.cs file created while adding WCF Service to the Application.

C#

```
namespace PivotGridDemo
{
[ServiceContract]
public interface IRelationalService
{
[OperationContract]
Dictionary<string, object> InitializeGrid(string action);
[OperationContract]
Dictionary<string, object> FetchMembers(string action, string headerTag,
string sortedHeaders, string currentReport);
[OperationContract]
Dictionary<string, object> Filtering(string action, string filterParams,
string sortedHeaders, string currentReport);
[OperationContract]
```

```
Dictionary<string, object> NodeStateModified(string action, string
headerTag, string dropAxis, string sortedHeaders, string filterParams,
string currentReport);
[OperationContract]
Dictionary<string, object> NodeDropped(string action, string dropAxis,
string headerTag, string sortedHeaders, string filterParams, string
currentReport);
[OperationContract]
Dictionary<string, object> Sorting(string action, string sortedHeaders,
string currentReport);
[OperationContract]
void Export(System.IO.Stream stream);
[OperationContract]
Dictionary<string, object> DeferUpdate(string action, string
filterParams, string sortedHeaders, string currentReport);
[OperationContract]
Dictionary<string, object> CalculatedField(string action, string
headerTag, string currentReport);
[OperationContract]
Dictionary<string, object> CellEditing(string action, string index,
string valueHeaders, string summaryValues, string currentReport);
[OperationContract]
Dictionary<string, object> SaveReport(string reportName, string
operationalMode, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> LoadReportFromDB(string reportName, string
operationalMode, string olapReport, string clientReports);
}
}
```

Then, elaborate the service methods inside the main class, found in RelationalService.svc.cs file.

C#

```
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class RelationalService: IRelationalService
{
PivotGrid htmlHelper = new PivotGrid();
JavaScriptSerializer serializer = new JavaScriptSerializer();
Dictionary<string, object> dict = new Dictionary<string, object>();
static int cultureIDInfoval = 1033;
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
public Dictionary<string, object> InitializeGrid(string action)
{
htmlHelper.PivotReport = BindDefaultData();
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData());
return dict;
}
```

```
public Dictionary<string, object> FetchMembers(string action, string
headerTag, string sortedHeaders, string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(),
headerTag, sortedHeaders);
return dict;
}
public Dictionary<string, object> Filtering(string action, string
filterParams, string sortedHeaders, string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(),
filterParams, sortedHeaders);
return dict;
}
public Dictionary<string, object> NodeStateModified(string action, string
headerTag, string dropAxis, string sortedHeaders, string filterParams,
string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(),
headerTag, dropAxis, filterParams, sortedHeaders);
return dict;
}
public Dictionary<string, object> NodeDropped(string action, string
dropAxis, string headerTag, string sortedHeaders, string filterParams,
string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(),
dropAxis, headerTag, filterParams, sortedHeaders);
return dict;
}
public Dictionary<string, object> Sorting(string action, string
sortedHeaders, string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(),
sortedHeaders);
return dict;
}
public Dictionary<string, object> CalculatedField(string action, string
headerTag, string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(), null,
headerTag);
return dict;
}
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
System.Web.HttpContext.Current.Response);
}
public Dictionary<string, object> SaveReport(string reportName, string
operationalMode, string olapReport, string clientReports)
{
string mode = operationalMode;
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
//cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(clientReports).ToArray());
if (mode == "serverMode")
cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(clientReports).ToArray());
else if (mode == "clientMode")
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(clientReports).ToArray());
cmd1.ExecuteNonQuery();
con.Close();
return null;
}
public Dictionary<string, object> LoadReportFromDB(string reportName,
string operationalMode, string olapReport, string clientReports)
{
byte[] reportString = new byte[2 * 1024];
PivotReport report = new PivotReport();
var reports = "";
string mode = operationalMode;
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && !string.IsNullOrEmpty(clientReports))
{
reports = clientReports;
}
else
{
```

```
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
if (mode == "clientMode")
{
reportString = (row.ItemArray[1] as byte[]);
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
report = htmlHelper.DeserializedReports(reports);
htmlHelper.PivotReport = report;
dictionary = htmlHelper.GetJsonData("loadOperation",
ProductSales.GetSalesData(), "Load Report", reportName);
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
public Dictionary<string, object> DeferUpdate(string action, string
filterParams, string sortedHeaders, string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(), null,
null, null, sortedHeaders, filterParams);
return dict;
}
public Dictionary<string, object> CellEditing(string action, string
index, string valueHeaders, string summaryValues, string currentReport)
{
htmlHelper.PopulateData(currentReport);
dict = htmlHelper.GetJsonData(action, ProductSales.GetSalesData(), index,
summaryValues, valueHeaders);
return dict;
}
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Product",
FieldHeader = "Product", TotalHeader = "Total" });
pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName =
"Country", FieldHeader = "Country", TotalHeader = "Total" });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
return pivotSetting;
}
}
.....
..... // Initialize the datasource
.....
}
```

Configuring Web Configuration File

The services could be exposed through the properties, binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IRelationalService contract and hence it is PivotGridDemo.IRelationalService.
- Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
- BehaviorConfiguration: This property contains the name of the behavior to be used in the endpoint

The endpointBehaviors are illustrated as follows

XML

```
<system.serviceModel>
……
……
<services>
<service name="PivotGridDemo.RelationalService">
<endpoint address=""
behaviorConfiguration="PivotGridDemo.RelationalServiceAspNetAjaxBehavior"
binding="webHttpBinding" contract="PivotGridDemo.IRelationalService" />
</service>
</services>
</system.serviceModel>
```

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotGridDemo.RelationalServiceAspNetAjaxBehavior">
<enableWebScript />
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Note: In this example, **"PivotGridDemo"** indicates the name and root namespace of the Web Application created in Visual Studio IDE and "RelationalService" indicates the name of the WCF service created.

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
//NOTE: In the following highlighted lines _wcf_ is just the folder name
inside which service files(*.svc) are present.
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new {
controller = "Home", action = "Index", id = UrlParameter.Optional
},
namespaces: new [] {
"PivotGridDemo.Controllers"
}
);
}
```

Now, **PivotGrid** will be rendered with Sales Amount over a set of products across different customer geographic locations.

	Canada	France	Germany	United Kingdom	United States	Grand Total
	Amount	Amount	Amount	Amount	Amount	Amount
Bike	$51,651,000.00	$44,167,500.00	$49,383,900.00	$46,314,000.00	$44,398,800.00	**$235,915,200.00**
Car	$5,966,100.00	$5,774,100.00	$5,315,100.00	$5,860,500.00	$6,155,400.00	**$29,071,200.00**
Van	$14,252,400.00	$15,807,600.00	$11,985,000.00	$11,008,200.00	$14,601,600.00	**$67,654,800.00**
Grand Total	**$71,869,500.00**	**$65,749,200.00**	**$66,684,000.00**	**$63,182,700.00**	**$65,155,800.00**	**$332,641,200.00**

Copyright © 2001 - 2020 Syncfusion Inc.

Data Binding

Binding PivotGrid to Offline Cube

To connect an OLAP Cube available in local machine, set the physical path of the Cube in the connection string. The following code example illustrates the same.

C#

```
string connectionString = @"DataSource= system
drive:\OfflineCube\Adventure_Works_Ext.cub; Provider = MSOLAP;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotGrid to Cube in local SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in local machine, set the server name and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = "Data source=localhost; Initial
Catalog=Adventure Works DW;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotGrid to Cube in online SQL Server

To connect an OLAP Cube available in SQL Server Analysis Service in online server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
static string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
```

Binding PivotGrid to Cube in online Mondrian Server

To connect an OLAP Cube available in Mondrian Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString = @"Data Source =
https://localhost:8080/mondrian/xmla; Initial Catalog =FoodMart;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName =
Syncfusion.Olap.DataProvider.Providers.Mondrian;
```

Copyright © 2001 - 2020 Syncfusion Inc.

Binding PivotGrid to Cube in online ActivePivot Server

To connect an OLAP Cube available in ActivePivot Server through **XML/A**, set the host server link and database name in the connection string. When you have any credentials to connect your Cube, then set the “User ID” and “Password” attributes accordingly. The following code example illustrates the same.

C#

```
string connectionString=@"Data Source =
https://localhost:8080/cva_s/xmla; Initial Catalog = CVAS;";
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.DataProvider.ProviderName=Syncfusion.Olap.DataProvider.Provid
ers.ActivePivot;
```

WCF

Adding a WCF Service

To add a WCF service in an existing MVC web application, right-click on project in Solution Explorer and select **Add > New Item**. In the **Add New Item** window, select WCF Service and name it as **OlapService.svc**, click Add.

Now, WCF service is added into your application successfully and it comprises of the following files. The utilization of these files is explained in the immediate sections.

- OlapService.svc
- OlapService.svc.cs
- IOlapService.svc.cs

Configuring WCF Service Class

Remove the **“DoWork”** method present inside both OlapService.svc.cs and IOlapService.cs files. Next, add **“AspNetCompatibilityRequirements”** attribute on top of main class present inside OlapService.svc.cs file and set **“RequirementsMode”** value to **“Allowed”**.

C#

```
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
}
}
```

List of Dependency Libraries

Next you need to add the below mentioned dependency libraries into your Web Application. These libraries could be found in GAC (Global Assembly Cache) as well.

To add them to your Web Application, right-click on **References** in Solution Explorer and select **Add Reference**. Now in the **Reference Manager** dialog, under **Assemblies > Extension**, the following Syncfusion libraries are found.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: If you have installed any version of SQL Server Analysis Service (SSAS) or Microsoft ADOMD.NET utility, then the location of Microsoft.AnalysisServices.AdomdClient library is [system drive:\Program Files (x86)\Microsoft.NET\ADOMD.NET]. And if you have installed any version of Essential Studio, then the location of Syncfusion libraries is [system drive:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\Assemblies].

- Microsoft.AnalysisServices.AdomdClient
- Syncfusion.Compression.Base
- Syncfusion.Linq.Base
- Syncfusion.Olap.Base
- Syncfusion.PivotAnalysis.Base
- System.Data.SqlServerCe (Version: 4.0.0.0)
- Syncfusion.XlsIO.Base
- Syncfusion.Pdf.Base
- Syncfusion.DocIO.Base
- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.EJ.Pivot
- Syncfusion.EJ.MVC

List of Namespaces

Following are the list of namespaces to be added on top of the main class inside OlapService.svc.cs file.

C#

```
using System;
using System.Web;
using System.Collections.Generic;
using System.Linq;
using System.ServiceModel;
using System.Text;
using System.ServiceModel.Activation;
using System.Web.Script.Serialization;
using Syncfusion.Olap.Manager;
using Syncfusion.Olap.Reports;
using Syncfusion.JavaScript.Olap;
using System.Data.SqlServerCe;
using System.Data;
using OLAPUTILS = Syncfusion.JavaScript.Olap;
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
}
}
```

Datasource Initialization

Now, the connection string to connect OLAP Cube, PivotGrid and JavaScriptSerializer instances are created immediately inside the main class in OlapService.svc.cs file.

C#

```
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
Syncfusion.JavaScript.PivotGrid htmlHelper = new
Syncfusion.JavaScript.PivotGrid();
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
JavaScriptSerializer serializer = new JavaScriptSerializer();
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
public Dictionary<string, object> InitializeGrid(string action, string
gridLayout, bool enablePivotFieldList, object customObject)
public Dictionary<string, object> InitializeGrid(string action, string
gridLayout, bool enablePivotFieldList, object customObject)
//Other codes
}
}
```

Service methods in WCF Service

First, define the service methods inside **IOlapService** interface, found in IOlapService.cs file, created while adding WCF Service to your Web Application.

C#

```
namespace PivotGridDemo
{
[ServiceContract]
public interface IOlapService
{
[OperationContract]
Dictionary<string, object> InitializeGrid(string action, string
gridLayout, bool enablePivotFieldList, object customObject);
[OperationContract]
Dictionary<string, object> DrillGrid(string action, string cellPosition,
string currentReport, string headerInfo, string layout, object
customObject);
[OperationContract]
Dictionary<string, object> Paging(string action, string pagingInfo,
string currentReport, string gridLayout, object customObject);
[OperationContract]
Dictionary<string, object> NodeDropped(string action, string dropType,
string nodeInfo, string filterParams, string gridLayout, string
currentReport);
[OperationContract]
Dictionary<string, object> RemoveButton(string action, string headerInfo,
string gridLayout, string currentReport);
[OperationContract]
Dictionary<string, object> FetchMembers(string action, string headerTag,
string currentReport);
[OperationContract]
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Dictionary<string, object> Filtering(string action, string filterParams,
string gridLayout, string currentReport);
[OperationContract]
Dictionary<string, object> MemberExpanded(string action, bool
checkedStatus, string parentNode, string tag, string cubeName, string
currentReport);
[OperationContract]
void Export(System.IO.Stream stream);
[OperationContract]
Dictionary<string, object> DeferUpdate(string action, string
filterParams, string currentReport);
[OperationContract]
Dictionary<string, object> SaveReport(string reportName, string
operationalMode, string olapReport, string clientReports);
[OperationContract]
Dictionary<string, object> LoadReportFromDB(string action, string
gridLayout, bool enablePivotFieldList, object customObject, string
reportName, string operationalMode, string olapReport, string
clientReports);
}
}
```

Secondly, elaborate the service methods inside the main class, found in OlapService.svc.cs file.

C#

```
namespace PivotGridDemo
{
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
public class OlapService: IOlapService
{
Syncfusion.JavaScript.PivotGrid htmlHelper = new
Syncfusion.JavaScript.PivotGrid();
public static int cultureIDInfoval = 1033;
string connectionString = "Data
Source=https://bi.syncfusion.com/olap/msmdpump.dll; Initial
Catalog=Adventure Works DW 2008 SE;";
JavaScriptSerializer serializer = new JavaScriptSerializer();
string conStringforDB = ""; //Enter appropriate connection string to
connect database for saving and loading operation of reports
public Dictionary<string, object> InitializeGrid(string action, string
gridLayout, bool enablePivotFieldList, object customObject)
public Dictionary<string, object> InitializeGrid(string action, string
gridLayout, bool enablePivotFieldList, object customObject)
{
OlapDataManager DataManager = null;
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(CreateOlapReport());
return htmlHelper.GetJsonData(action, DataManager, gridLayout,
enablePivotFieldList);
}
public Dictionary<string, object> DrillGrid(string action, string
cellPosition, string currentReport, string headerInfo, string layout,
object customObject)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, connectionString, DataManager,
cellPosition, headerInfo, layout);
}
public Dictionary<string, object> NodeDropped(string action, string
dropType, string nodeInfo, string filterParams, string gridLayout, string
currentReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, DataManager, dropType, nodeInfo,
filterParams, gridLayout, true);
}
public Dictionary<string, object> Filtering(string action, string
filterParams, string gridLayout, string currentReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, connectionString, DataManager,
null, filterParams, gridLayout);
}
public Dictionary<string, object> FetchMembers(string action, string
headerTag, string currentReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, DataManager, null, headerTag);
}
public Dictionary<string, object> Paging(string action, string
pagingInfo, string currentReport, string gridLayout, object customObject)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(htmlHelper.SetPaging(currentReport,
pagingInfo));
return htmlHelper.GetJsonData(action, DataManager, gridLayout);
}
public Dictionary<string, object> RemoveButton(string action, string
headerInfo, string gridLayout, string currentReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, connectionString, DataManager,
null, headerInfo, gridLayout);
}
public Dictionary<string, object> MemberExpanded(string action, bool
checkedStatus, string parentNode, string tag, string cubeName, string
currentReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
if (!string.IsNullOrEmpty(currentReport))
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, DataManager, checkedStatus,
parentNode, tag, cubeName);
}
public void Export(System.IO.Stream stream)
```

```
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5); ;
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
public Dictionary<string, object> SaveReport(string reportName, string
operationalMode, string olapReport, string clientReports)
{
string mode = operationalMode;
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
//cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(clientReports).ToArray());
if (mode == "serverMode")
cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(clientReports).ToArray());
else if (mode == "clientMode")
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(clientReports).ToArray());
cmd1.ExecuteNonQuery();
con.Close();
return null;
}
public Dictionary<string, object> LoadReportFromDB(string action, string
layout, bool enablePivotFieldList, object customObject, string
reportName, string operationalMode, string olapReport, string
clientReports)
{
string mode = operationalMode;
var reports = "";
byte[] reportString = new byte[4 * 1024];
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && !string.IsNullOrEmpty(clientReports))
{
```

```
reports = clientReports;
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
if (mode == "clientMode")
{
reportString = row.ItemArray[1] as byte[];
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
dynamic customData =
serializer.Deserialize<dynamic>(customObject.ToString());
var cultureIDInfo = new System.Globalization.CultureInfo(("en-US")).LCID;
if (customData is Dictionary<string, object> &&
customData.ContainsKey("Language"))
{
cultureIDInfo = new
System.Globalization.CultureInfo((customData["Language"])).LCID;
}
connectionString = connectionString.Replace("" + cultureIDInfoval + "",
"" + cultureIDInfo + "");
cultureIDInfoval = cultureIDInfo;
DataManager.Culture = new
System.Globalization.CultureInfo((cultureIDInfo));
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(report
s));
DataManager.OverrideDefaultFormatStrings = true;
dictionary = htmlHelper.GetJsonData(action, DataManager, layout,
enablePivotFieldList);
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString =
conStringforDB };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public Dictionary<string, object> DeferUpdate(string action, string
filterParams, string currentReport)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Utils.DeserializeOlapReport(currentReport));
return htmlHelper.GetJsonData(action, DataManager, null, filterParams);
}
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.CurrentCubeName = "Adventure Works";
MeasureElements measureElement = new MeasureElements();
measureElement.Elements.Add(new MeasureElement { UniqueName =
"[Measures].[Internet Sales Amount]" });
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
DimensionElement dimensionElementColumn = new DimensionElement();
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");
olapReport.SeriesElements.Add(dimensionElementRow);
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(measureElement);
return olapReport;
}
}
}
```

Configuring Web Configuration File

You can expose services through the properties, binding, contract and address by using an endpoint.

- Contract: This property indicates that the contract of the endpoint is exposing. Here you are referring to IOlapService contract and hence it is PivotGridDemo.IOlapService.
- Binding: In your application, you use webHttpBinding to post and receive the requests and responses between the client-end and the service.
- behaviorConfiguration: This property contains the name of the behavior to be used in the endpoint.

The endpointBehaviors are illustrated as follows.

XML

```
<system.serviceModel>
……
……
<services>
<service name="PivotGridDemo.OlapService">
<endpoint address=""
behaviorConfiguration="PivotGridDemo.OlapServiceAspNetAjaxBehavior"
binding="webHttpBinding" contract="PivotGridDemo.IOlapService" />
</service>
</services>
</system.serviceModel>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The endpointBehaviors contain all the behaviors for an endpoint. You can link each endpoint to the respective behavior only by using this name property.

XML

```
<system.serviceModel>
<behaviors>
<endpointBehaviors>
<behavior name="PivotGridDemo.OlapServiceAspNetAjaxBehavior">
<enableWebScript /> </behavior>
</endpointBehaviors>
</behaviors>
……
……
</system.serviceModel>
```

Configuring routing file

Routing configuration needs to be done in RouteConfig.cs file found under **AppStart** folder. The configuration would allow picking appropriate WCF service without any issues.

C#

```
public static void RegisterRoutes(RouteCollection routes) {
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
//NOTE: In the following highlighted lines _wcf_ is just the folder name
inside which service files(*.svc) are present.
routes.IgnoreRoute("{resource}.svc/{*pathInfo}");
routes.IgnoreRoute("{resource}.svc");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new {
controller = "Home", action = "Index", id = UrlParameter.Optional
},
namespaces: new [] {
"PivotGridDemo.Controllers"
}
);
}
```

Note: In this example, **"PivotGridDemo"** indicates the name and root namespace of the Web Application created in Visual Studio IDE and "OlapService" indicates the name of the WCF service created.

Now, **PivotGrid** will be rendered with Internet Sales Amount over a period of fiscal years across different customer geographic locations.

Copyright © 2001 - 2020 Syncfusion Inc.

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	**$7,072,084.24**
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	**$5,762,134.30**
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	**$16,473,618.05**
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	**$50,840.63**
Total	**$9,061,000.58**	**$1,977,844.86**	**$2,644,017.71**	**$2,894,312.34**	**$3,391,712.21**	**$9,389,789.51**	**$29,358,677.22**

Relational

Grouping Bar

Initialization

Grouping Bar allows user to dynamically alter the report by filter, sort and remove operations in the PivotGrid control. Based on the Relational datasource and report bound to the PivotGrid control, Grouping Bar will be automatically populated. You can enable Grouping Bar option in PivotGrid by setting the `EnableGroupingBar` property to true.

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableGroupingBar(true).Client
SideEvents(clientSideEvents =>
clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
rows.FieldName("State").FieldCaption("State").Add(); }).Columns(columns
=> { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();
values.FieldName("Quantity").Add(); }).Filters(filters => {
filters.FieldName("Date").FieldCaption("Date").Add(); }))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = pivot_dataset;//Array of data
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Date

Amount | Quantity | Product

Country / State		Bike		Car		Van		Grand Total	
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	Alberta	100	2					100	2
	British Columbia					200	3	200	3
	Brunswick			300	4			300	4
	Manitoba	150	3					150	3
	Ontario			200	4			200	4
	Quebec					100	1	100	1
Canada Total		250	5	500	8	300	4	1050	17

Server Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/RelationalService")).EnableGroupingBar(true)
```

Drag field here

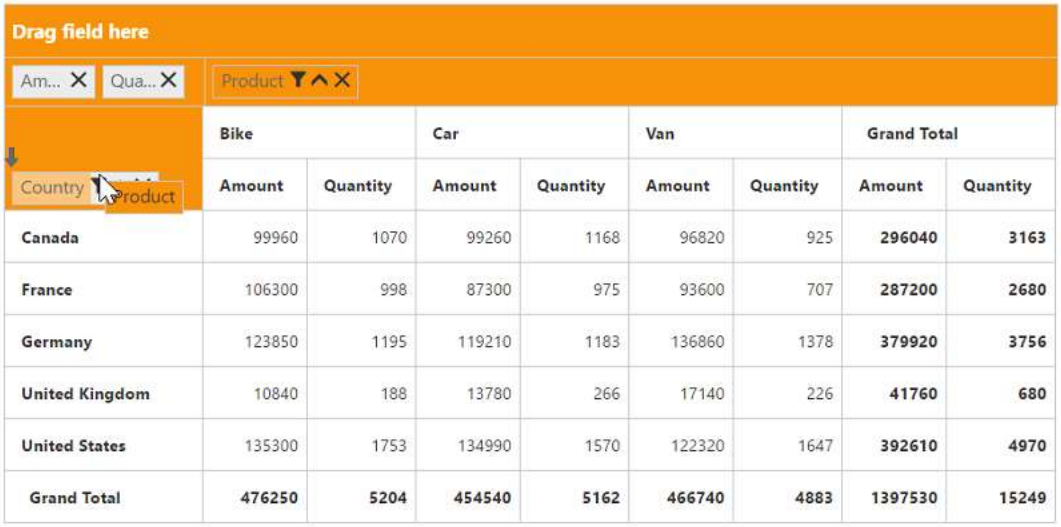
Amount | Product | Country

Date / State		Bike			Bike Total
		Canada	France	United States	
		Amount	Amount	Amount	Amount
FY 2005	Alabama		$78,491,400.00		$78,491,400.00
	Alberta	$27,077,100.00			$27,077,100.00
	British Columbia	$26,943,600.00			$26,943,600.00
	Brunswick	$28,211,400.00			$28,211,400.00
	New York			$75,191,700.00	$75,191,700.00
FY 2005 Total		$82,232,100.00	$78,491,400.00	$75,191,700.00	$235,915,200.00

Drag and Drop

You can alter the report on fly through drag-and-drop operation.

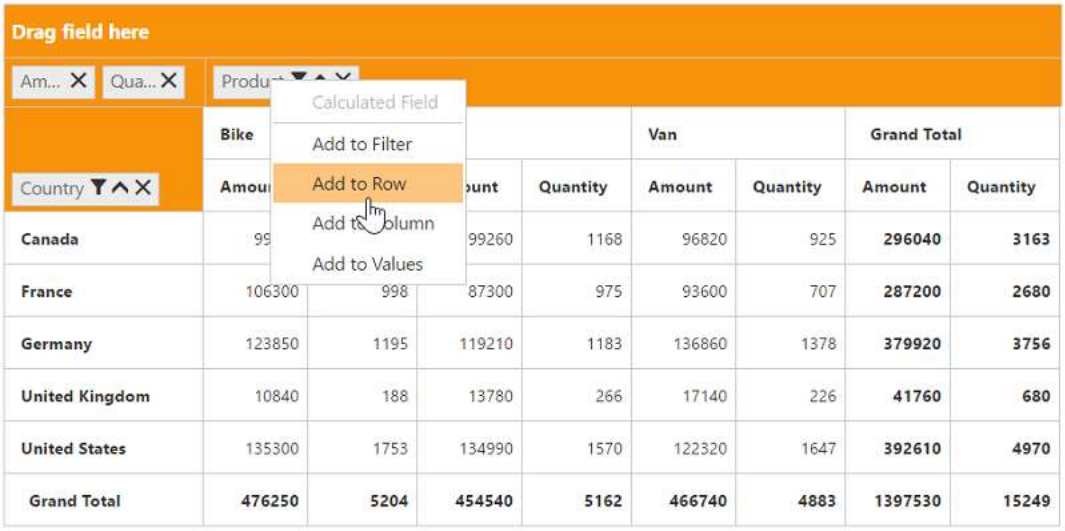
Copyright © 2001 - 2020 Syncfusion Inc.



	Bike		Car		Van		Grand Total	
Country	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	99960	1070	99260	1168	96820	925	**296040**	**3163**
France	106300	998	87300	975	93600	707	**287200**	**2680**
Germany	123850	1195	119210	1183	136860	1378	**379920**	**3756**
United Kingdom	10840	188	13780	266	17140	226	**41760**	**680**
United States	135300	1753	134990	1570	122320	1647	**392610**	**4970**
Grand Total	**476250**	**5204**	**454540**	**5162**	**466740**	**4883**	**1397530**	**15249**

Context Menu

You can alter the report by using context menu.

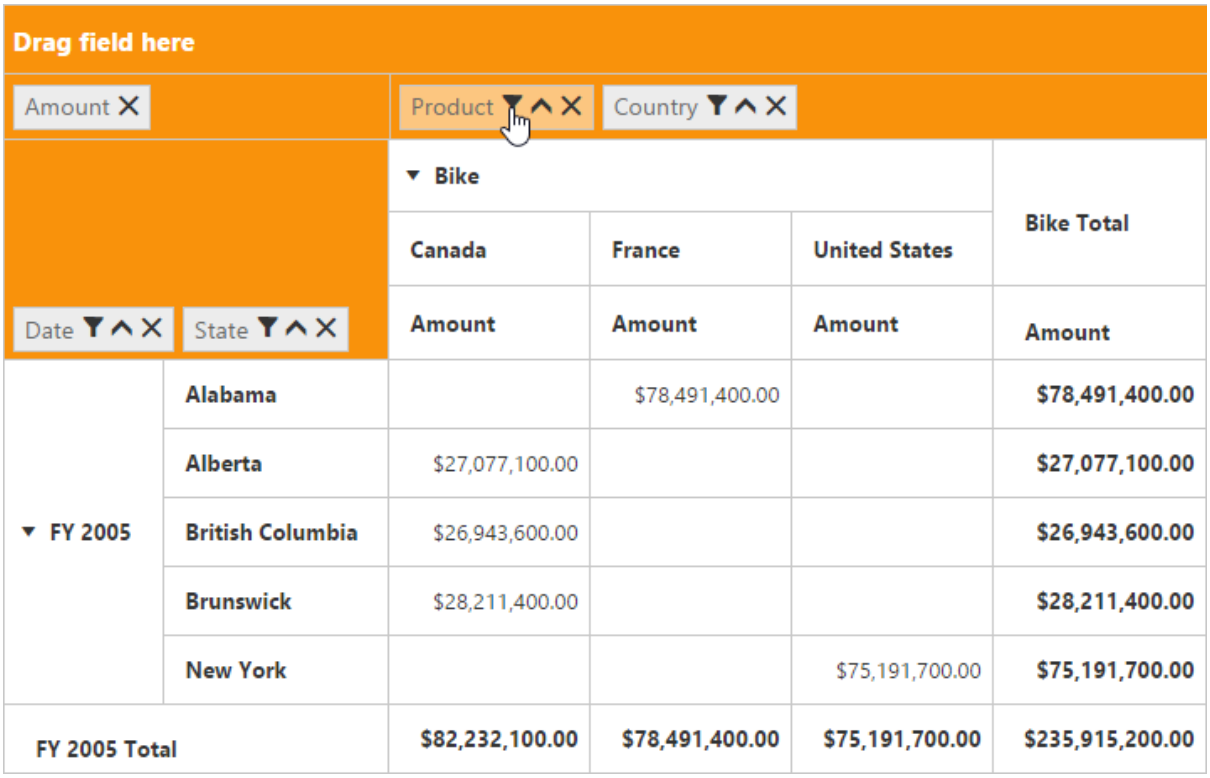


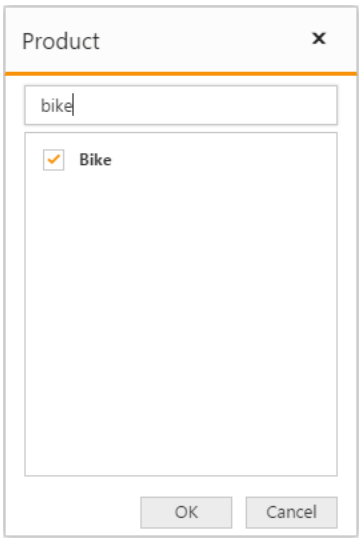
	Bike				Van		Grand Total	
Country	Amou		unt	Quantity	Amount	Quantity	Amount	Quantity
Canada	99		99260	1168	96820	925	**296040**	**3163**
France	106300	998	87300	975	93600	707	**287200**	**2680**
Germany	123850	1195	119210	1183	136860	1378	**379920**	**3756**
United Kingdom	10840	188	13780	266	17140	226	**41760**	**680**
United States	135300	1753	134990	1570	122320	1647	**392610**	**4970**
Grand Total	**476250**	**5204**	**454540**	**5162**	**466740**	**4883**	**1397530**	**15249**

Searching Values

Search option available in Grouping Bar allows you to search a specific value that needs to be filtered from the list of values inside the filter pop-up window.

Copyright © 2001 - 2020 Syncfusion Inc.

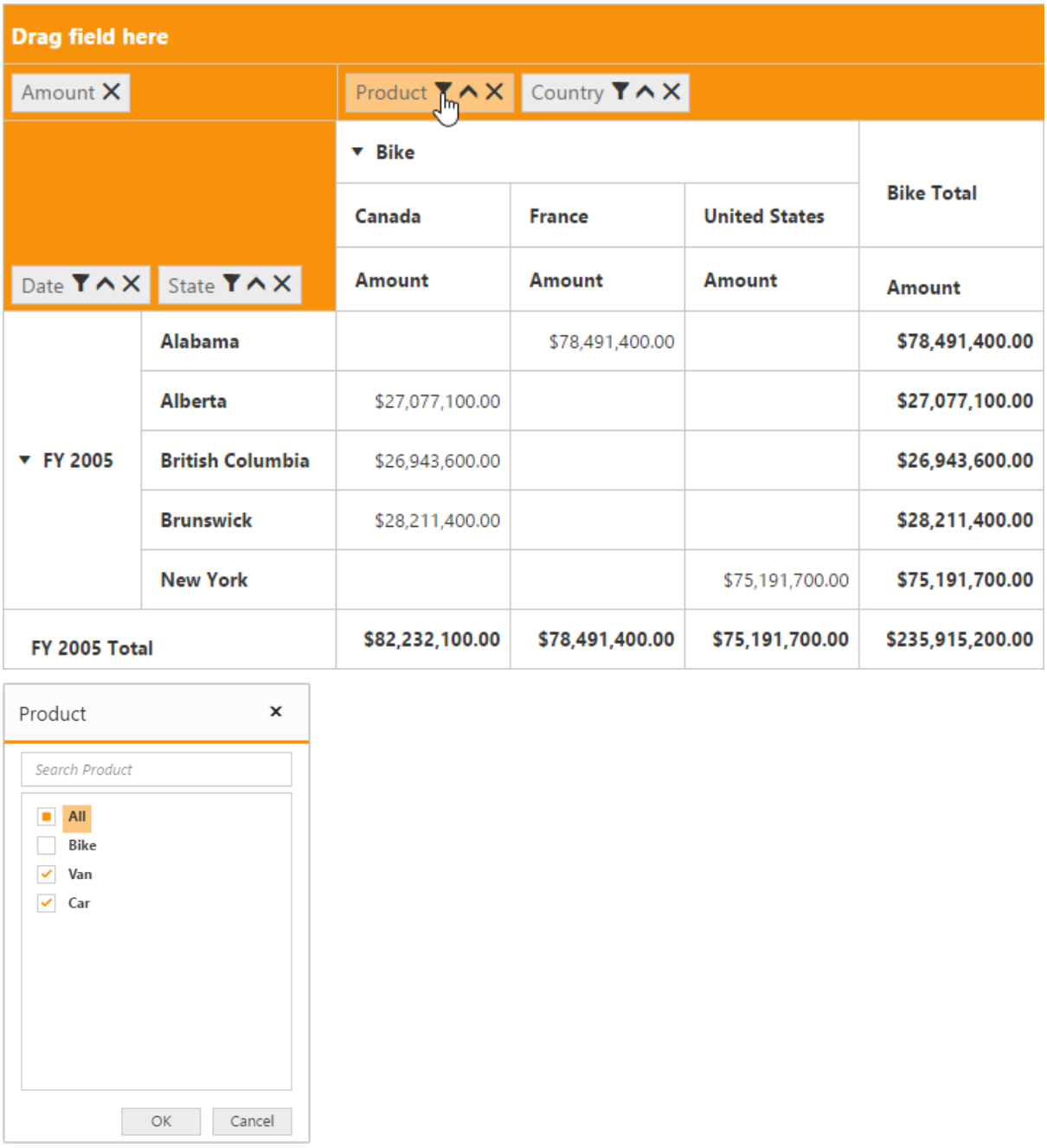




Filtering Values

Filtering option available in Grouping Bar allows you to select a specific set of values that needs to be displayed in the PivotGrid control. At least one value needed to be in checked state while filtering otherwise “Ok” button will be disabled.

Copyright © 2001 - 2020 Syncfusion Inc.



Sorting Values

Sorting option available in Grouping Bar allows you to arrange headers either in ascending or descending order. Sorting option is applicable for fields available only in Row and Column region. By default, headers are sorted in ascending order. Regarding sorting indicator, up arrow denotes ascending order and down arrow denotes descending order.

Copyright © 2001 - 2020 Syncfusion Inc.

Drag field here					
Amount ✕		Product ▼ ^ ✕	Country ▼ ^ ✕		
		▼ Bike			Bike Total
Date ▼ ^ ✕	State ▼ ^ ✕	Canada	France	United States	
		Amount	Amount	Amount	Amount
▼ FY 2005	Alabama		$78,491,400.00		$78,491,400.00
	Alberta	$27,077,100.00			$27,077,100.00
	British Columbia	$26,943,600.00			$26,943,600.00
	Brunswick	$28,211,400.00			$28,211,400.00
	New York			$75,191,700.00	$75,191,700.00
FY 2005 Total		$82,232,100.00	$78,491,400.00	$75,191,700.00	$235,915,200.00

Drag field here					
Amount ✕		Product ▼ ^ ✕	Country ▼ v ✕		
		▼ Bike			Bike Total
Date ▼ ^ ✕	State ▼ ^ ✕	United States	France	Canada	
		Amount	Amount	Amount	Amount
▼ FY 2005	Alabama		$78,491,400.00		$78,491,400.00
	Alberta			$27,077,100.00	$27,077,100.00
	British Columbia			$26,943,600.00	$26,943,600.00
	Brunswick			$28,211,400.00	$28,211,400.00
	New York	$75,191,700.00			$75,191,700.00
FY 2005 Total		$75,191,700.00	$78,491,400.00	$82,232,100.00	$235,915,200.00

Removing Field

Remove option available in the Grouping Bar allows you to completely remove a specific field from the PivotGrid control. Remove operation can be either achieved by clicking remove icon available inside each field or by dragging and dropping field out of Grouping Bar region.

Copyright © 2001 - 2020 Syncfusion Inc.

Drag field here

Amount ✕		Product ▼ ^ ✕	Country ▼ ^ ✕		
		▼ Bike			Bike Total
Date ▼ ^ ✕	State ▼ ^ ✕	Canada	France	United States	
		Amount	Amount	Amount	Amount
▼ FY 2005	Alabama		$78,491,400.00		$78,491,400.00
	Alberta	$27,077,100.00			$27,077,100.00
	British Columbia	$26,943,600.00			$26,943,600.00
	Brunswick	$28,211,400.00			$28,211,400.00
	New York			$75,191,700.00	$75,191,700.00
FY 2005 Total		$82,232,100.00	$78,491,400.00	$75,191,700.00	$235,915,200.00

Drag field here

Amount ✕		Country ▼ ^ ✕			
		Canada	France	United States	Grand Total
Date ▼ ^ ✕	State ▼ ^ ✕	Amount	Amount	Amount	Amount
▼ FY 2005	Alabama		$109,102,500.00		$109,102,500.00
	Alberta	$37,862,700.00			$37,862,700.00
	British Columbia	$38,304,000.00			$38,304,000.00
	Brunswick	$39,497,700.00			$39,497,700.00
	New York			$107,874,300.00	$107,874,300.00
FY 2005 Total		$115,664,400.00	$109,102,500.00	$107,874,300.00	$332,641,200.00
Grand Total		$115,664,400.00	$109,102,500.00	$107,874,300.00	$332,641,200.00

PivotTable Field List

Initialization

Field List, also known as Pivot Schema Designer, allows user to add, rearrange, filter and remove fields to show data in PivotGrid exactly the way they want.

Copyright © 2001 - 2020 Syncfusion Inc.

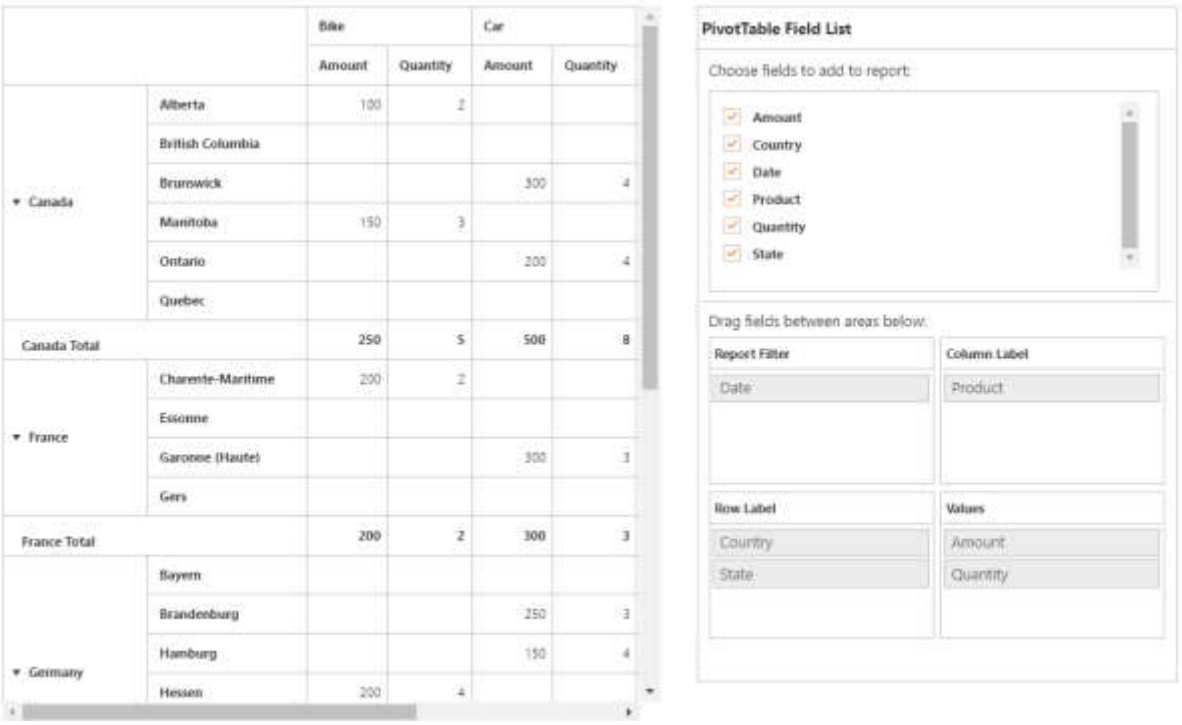
Based on the datasource, Relational, bound to the PivotGrid control, PivotTable Field List will be automatically populated with Cube Information or Field Names. PivotTable Field List provides an Excel like appearance and behavior.

In-order to initialize PivotTable Field List, first you need to define a “div” tag with an appropriate “id” attribute which acts as a container for the control. Then you need to initialize the PivotTable Field List by using the **“PivotSchemaDesigner”** method.

Client Mode

ASPX-CS

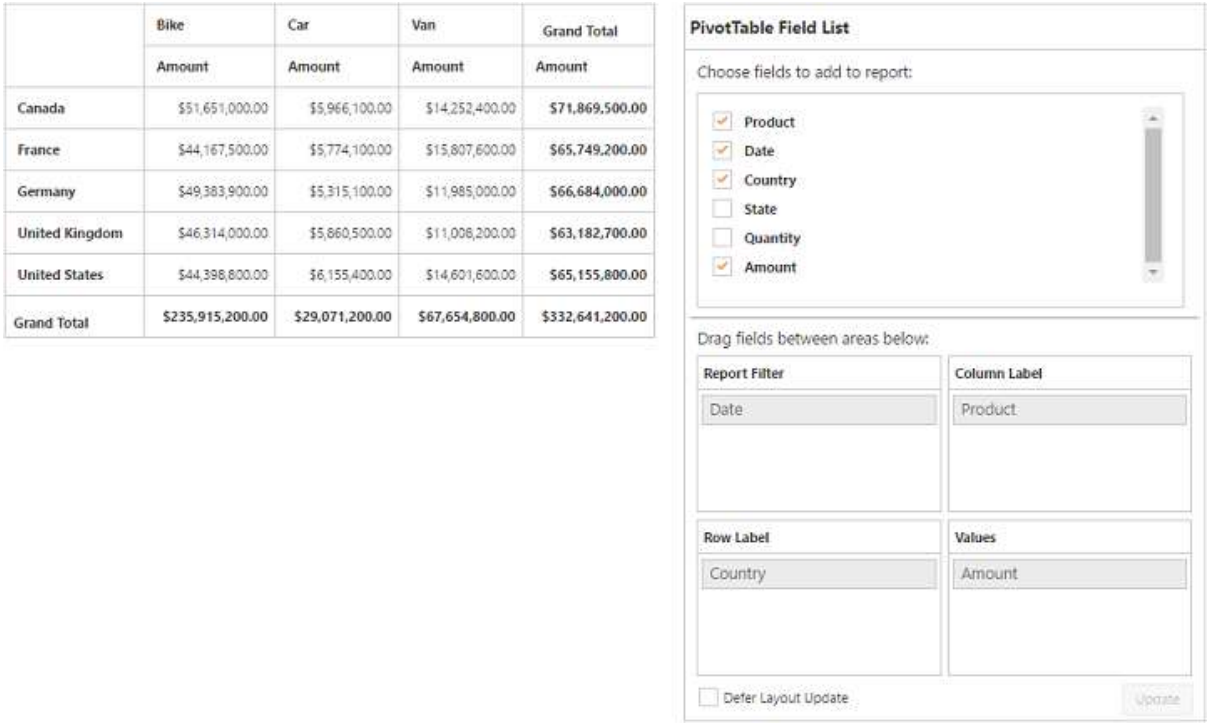
```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts =>
clientSideEvents.Load("onLoad").RenderSuccess("loadSchemaDesigner")).Data
Source(dataSource => dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
rows.FieldName("State").FieldCaption("State").Add(); }).Columns(columns
=> { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();
values.FieldName("Quantity").Add(); }))
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner")
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = pivot_dataset;//Array of data
}
function loadSchemaDesigner(args) {
$("#PivotSchemaDesigner").ejPivotSchemaDesigner({ pivotControl: this,
layout: ej.PivotSchemaDesigner.Layouts.Excel });
args.model.renderSuccess = null;
}
</script>
```



Server Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/RelationalSe
rvice")).ClientSideEvents(events =>
events.AfterServiceInvoke("OnAfterServiceInvoke"))
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner").Layout(Pivo
tSchemaDesignerLayout.Excel)
<script type="text/javascript">
OnAfterServiceInvoke = function(evt) {
if (evt.action == "initialize") {
var PivotSchemaDesigner =
$("#PivotSchemaDesigner").data('ejPivotSchemaDesigner');
if (PivotSchemaDesigner.model.pivotControl == null) {
PivotSchemaDesigner.model.pivotControl = this;
PivotSchemaDesigner.model.enableWrapper = true;
PivotSchemaDesigner.model.layout = "excel";
PivotSchemaDesigner._load();
}
}
}
</script>
```



	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
Canada	$51,651,000.00	$5,966,100.00	$14,252,400.00	$71,869,500.00
France	$44,167,500.00	$5,774,100.00	$15,807,600.00	$65,749,200.00
Germany	$49,383,900.00	$5,315,100.00	$11,985,000.00	$66,684,000.00
United Kingdom	$46,314,000.00	$5,860,500.00	$11,008,200.00	$63,182,700.00
United States	$44,398,800.00	$6,155,400.00	$14,601,600.00	$65,155,800.00
Grand Total	$235,915,200.00	$29,071,200.00	$67,654,800.00	$332,641,200.00

Layout

The top portion of the layout shows Field or Cube items in a categorized way. They can be dynamically added into the report either by drag and drop option or through simple check box selection.

On item(s) selection they will be placed in Row section by default except numeric based item(s) or measures, which will alone be placed in the Value section by default.

The bottom portion of the layout is segregated as below.

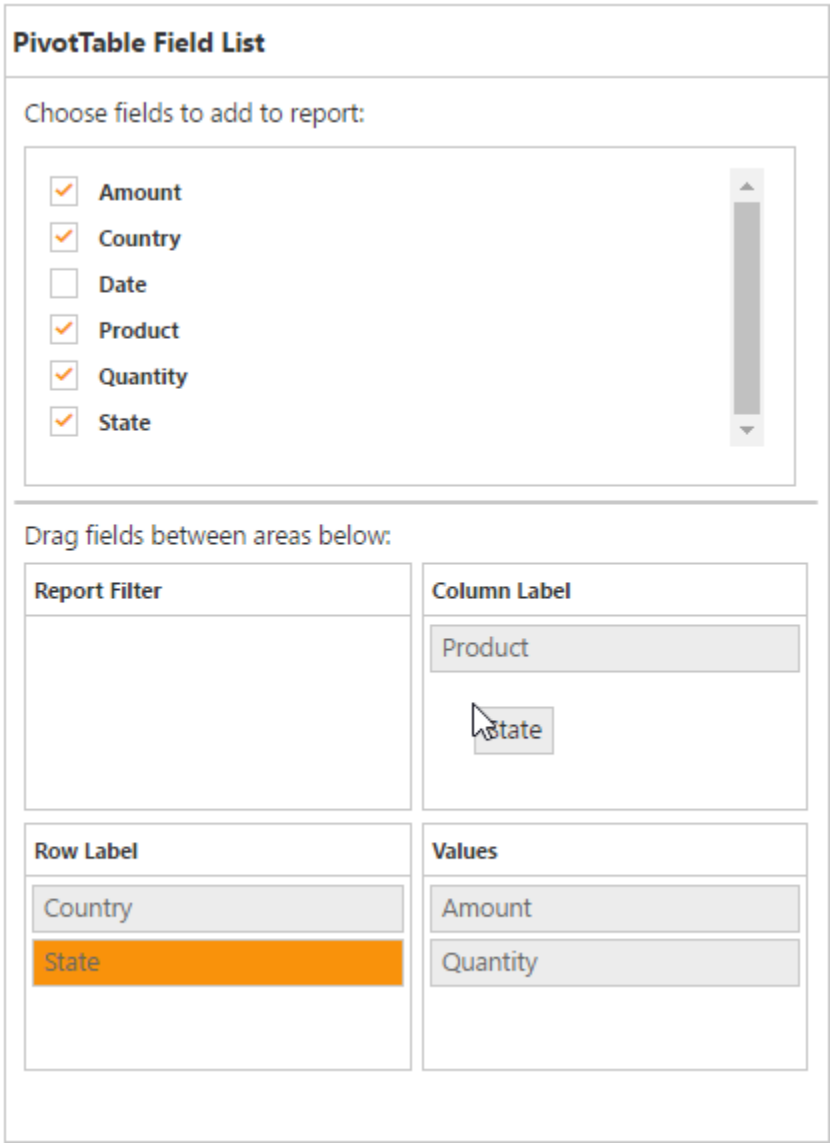
Copyright © 2001 - 2020 Syncfusion Inc.

- Report Filter: Exclusively designed to filter an item(s) placed in this particular position of the layout.
- Value Section: The value label usually displays the numeric value item(s) present in the report.
- Column Section: It is used to display item(s) as column header and values in the PivotGrid control.
- Row Section: It is used to display item(s) as row header and values in the PivotGrid control.

UI Interactions

By Drag and Drop

You can alter the report on fly through drag-and-drop operation. You can drag any item from Field List and drop into column, row, value or filter section available at the bottom of the Field List.



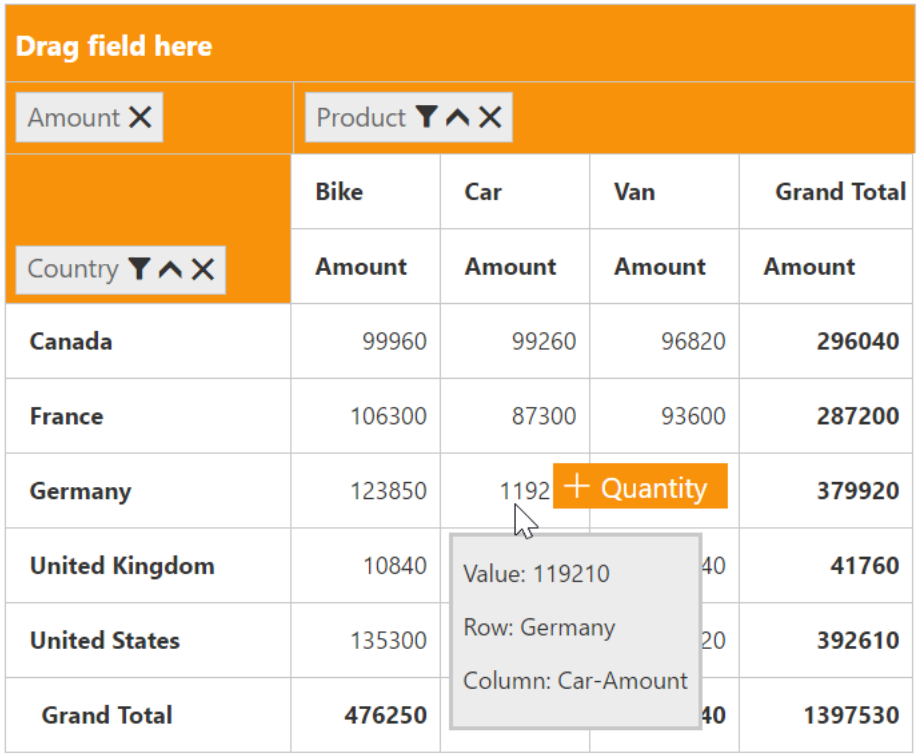
By Drag and Drop to Grid Headers

You can also drag and drop elements from field list to grid headers.

Drag field here				
Amount ✕	Product ▼ ^ ✕			
Country ▼ ^ ✕	Bike	Car + Date	Van	Grand Total
	Amount	Amount	Amount	Amount
Canada	99960	99260	96820	296040
France	106300	87300	93600	287200
Germany	123850	119210	136860	379920
United Kingdom	10840	13780	17140	41760
United States	135300	134990	122320	392610
Grand Total	476250	454540	466740	1397530

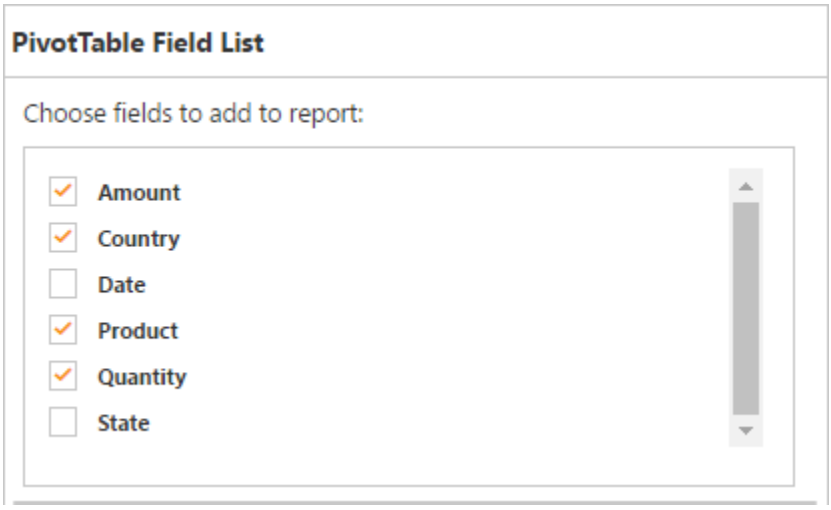
Drag field here				
Amount ✕	Product ▼ ^ ✕			
Country ▼ ^ ✕	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
Canada	99960	99260	96820	296040
France	106300	87300	93600	287200
Germany + Date	123850	119210	136860	379920
United Kingdom	10840	13780	17140	41760
United States	135300	134990	122320	392610
Grand Total	476250	454540	466740	1397530

Copyright © 2001 - 2020 Syncfusion Inc.



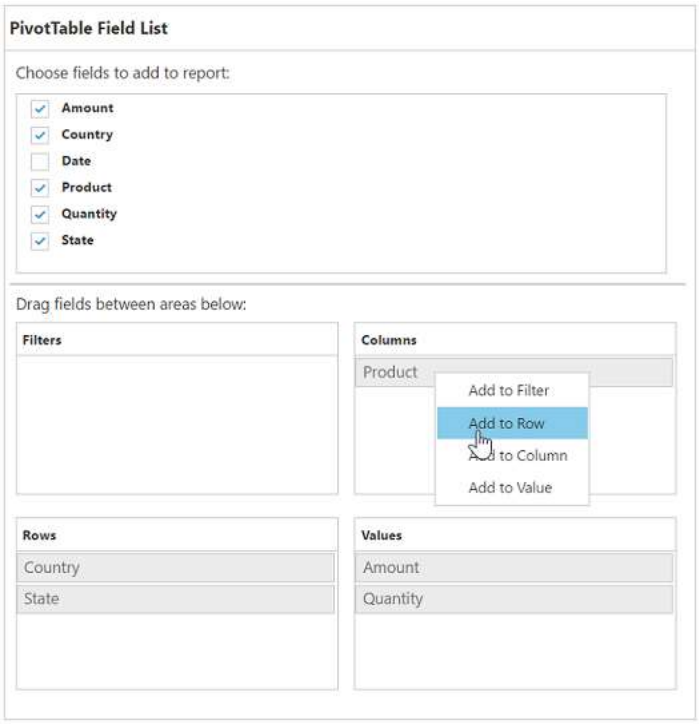
By Treeview Selection

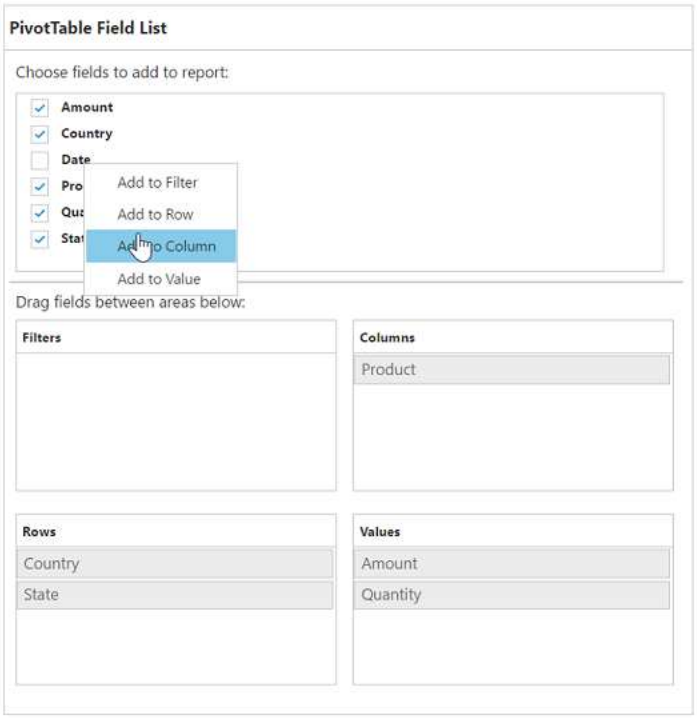
You can also alter the report on fly through check and uncheck option as an alternate. By default, fields will be added to the Row Axis when checked.



By Context Menu

You can also alter the report by using context menu.

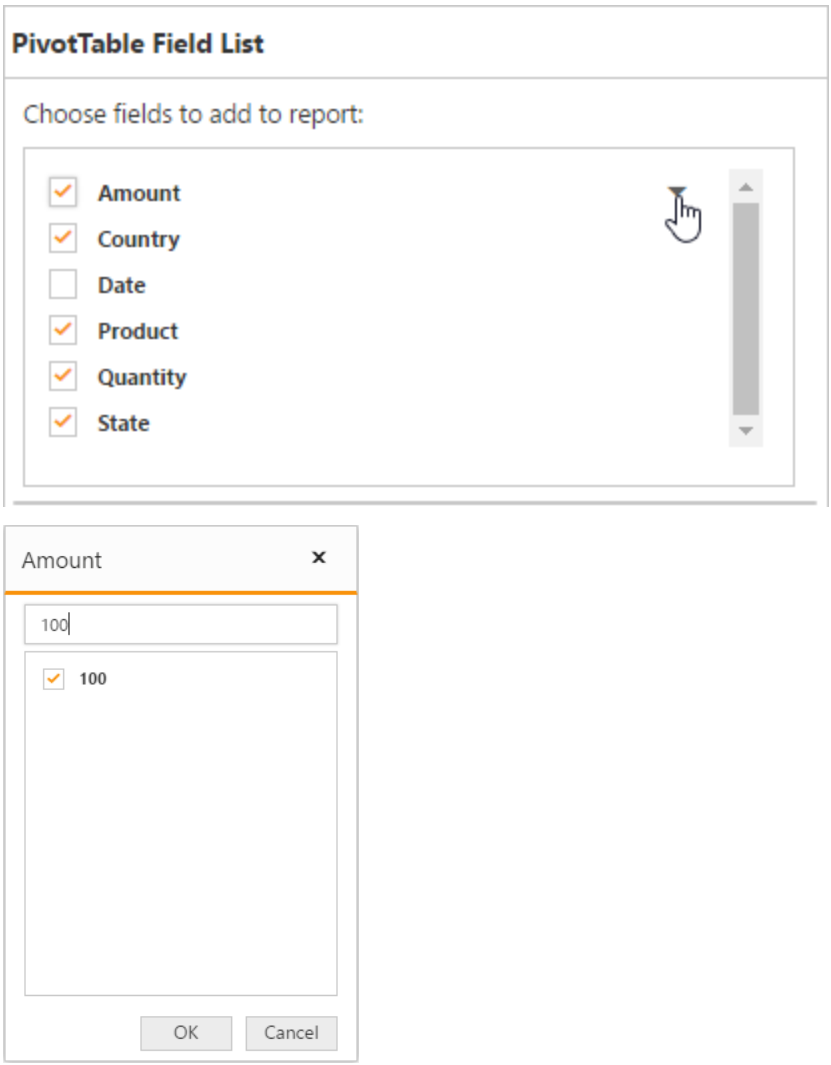




Searching Values

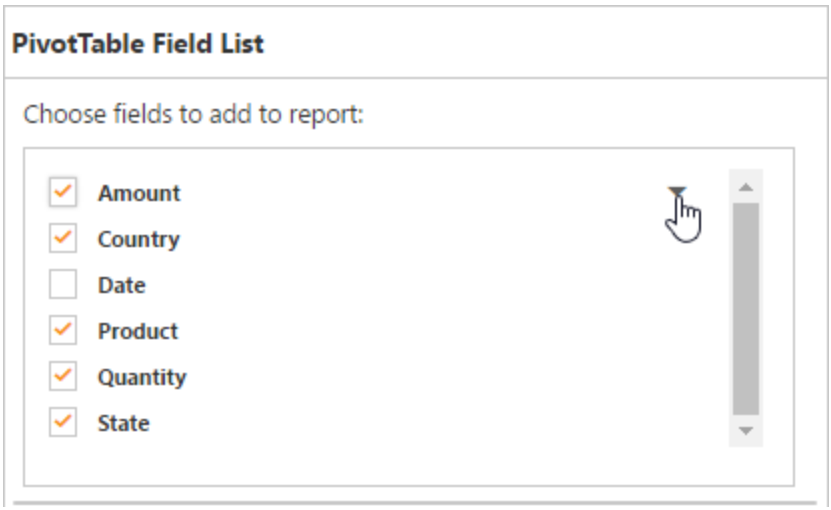
Search option available in Field List allows you to search a specific value that needs to be filtered from the list of values inside the filter pop-up window.

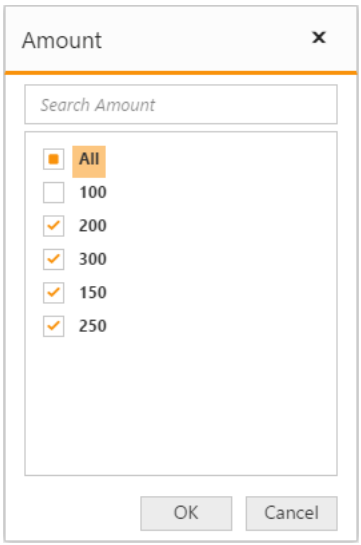
Copyright © 2001 - 2020 Syncfusion Inc.



Filtering

Values can be filtered by checking/unchecking the check box besides them, inside the filter pop-up window. At least one value needed to be checked state while filtering otherwise “Ok” button will be disabled.





Format Strings

Information: This feature is applicable only for Relational data source at Client Mode.

Formatted numeric values like n (number with decimal points), c (currency or accounting) and p (percentage) can be displayed with desired decimal places using FormatString option.

Note: This is applicable only when the Format option is set.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
rows.FieldName("State").FieldCaption("State").Add(); }).Columns(columns
=> { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add().FormatString("{0:c0}"
); }))
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
▸ Canada	$250	$500	$300	$1,050
▸ France	$200	$300	$400	$900
▸ Germany	$300	$400	$350	$1,050
▸ United Kingdom	$150			$150
▸ United States	$300	$650	$500	$1,450
Grand Total	$1,200	$1,850	$1,550	$4,600

The following table describes the result of applying some commonly used format strings on numeric values.

Label Value	Label Format property value	Result	Description
1000	n1	1000.0	The Number is rounded to 1 decimal place
1000	n2	1000.00	The Number is rounded to 2 decimal place
1000	n3	1000.000	The Number is rounded to 3 decimal place
0.01	p1	1.0%	The Number is converted to percentage with 1 decimal place
0.01	p2	1.00%	The Number is converted to percentage with 2 decimal place
0.01	p3	1.000%	The Number is converted to percentage with 3 decimal place
1000	c1	$1,000.0	The Currency symbol is appended to number and number is rounded to 1 decimal place
1000	c2	$1,000.00	The Currency symbol is appended to number and number is rounded to 2 decimal place
1.0	d2	01	The number is rounded with 2 whole number place
1.0	d3	001	The number is rounded with 3 whole number place

The following table describes some common results on applying standard format sets to date format using FormatString option.

Format Pattern	Description	Result
d	The day of the month between 1 and 31. Â	"1" Â to "31"

Copyright © 2001 - 2020 Syncfusion Inc.

dd	The day of the month with leading zero if required.	"01" to "31"
ddd	Abbreviated day name.	"Mon" to "Sun"
dddd	The full day name	"Monday" to "Sunday"
M	The month of the year between 1 - 12	"1" to "12"
MM	The month of the year with leading zero if required	"01" to "12"
MMM	Abbreviated month name	"Jan" to "Dec"
MMMM	The full month name	"January" to "December"
yy	The year as a two-digit number	"99" or "08"
yyyy	The full four digit year	"1999" or "2008"
MM/dd/yyyy	The Date is displayed in "month/date/year" format	"04/09/1900"
d M, y	The Date is displayed in "date month, year" format	"7 9, 0"
d MM, y	The Date is displayed in "date month, year" format	"7 09, 0"
dddd, d MMMM, yy	The Date is displayed in "day, date month, year" format	"Tuesday, 7 September, 00"
yyyy-MM-dd	The Date is displayed in "year-month-date"(UTC) format	"1900-09-06"

Summary Type

Information: This feature is applicable only for Relational datasource only at Client Mode.

Allow us to specify the required summary type that PivotGrid should use in its summary cells. “Sum” is the default summary type. Following summary types that are supported:

- Sum
- Average
- Count
- Min

Copyright © 2001 - 2020 Syncfusion Inc.

- Max

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").SummaryType(PivotAnalysisSummaryType.Average).
Add();
values.FieldName("Quantity").FieldCaption("Quantity").SummaryType(PivotAn
alysisSummaryType.TotalSum).Add(); }))
<script type="text/javascript">
function onLoad(args) {
args.model.dataSource.data = pivot_dataset; // Datasource
}
</script>
```

	Bike		Car		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	125	5	250	8	187.5	13
France	200	2	300	3	250	5
Germany	150	6	200	7	175	13
United States	150	6	212.5	10	181.25	16
Grand Total	156.25	19	240.63	28	198.44	47

State persistence

Expand and collapse state of row and column headers are maintained during every UI interaction.

Value Sorting

Information: This feature is applicable for Relational datasource.

Value Sorting allows to sort columns and rows based on value fields.

The headers of the column to be sorted is given in the 'HeaderText' property under 'ValueSortSettings' in field wise order separated by a string. The string which is used to separate the headers is given in the property 'HeaderDelimiters'.

Also you can sort column by clicking the column header. On clicking the same header once again will reverse the sorting direction.

JS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add();
values.FieldName("Quantity").Add();
})).ValueSortSettings(valuesortsettings=>valuesortsettings.HeaderText("Bi
ke##Quantity").HeaderDelimiters("##").SortOrder(SortOrder.Descending))
```

	Bike		Car		Van		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	99960	1070	99260	1168	96820	925	296040	3163
France	106300	998	87300	975	93600	707	287200	2680
Germany	123850	1195	119210	1183	136860	1378	379920	3756
United Kingdom	10840	188	13780	266	17140	226	41760	680
United States	135300	1753	134990	1570	122320	1647	392610	4970
Grand Total	476250	5204	454540	5162	466740	4883	1397530	15249

	Bike		Car		Van		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
United States	135300	1753	134990	1570	122320	1647	392610	4970
Germany	123850	1195	119210	1183	136860	1378	379920	3756
Canada	99960	1070	99260	1168	96820	925	296040	3163
France	106300	998	87300	975	93600	707	287200	2680
United Kingdom	10840	188	13780	266	17140	226	41760	680
Grand Total	476250	5204	454540	5162	466740	4883	1397530	15249

Calculated Field

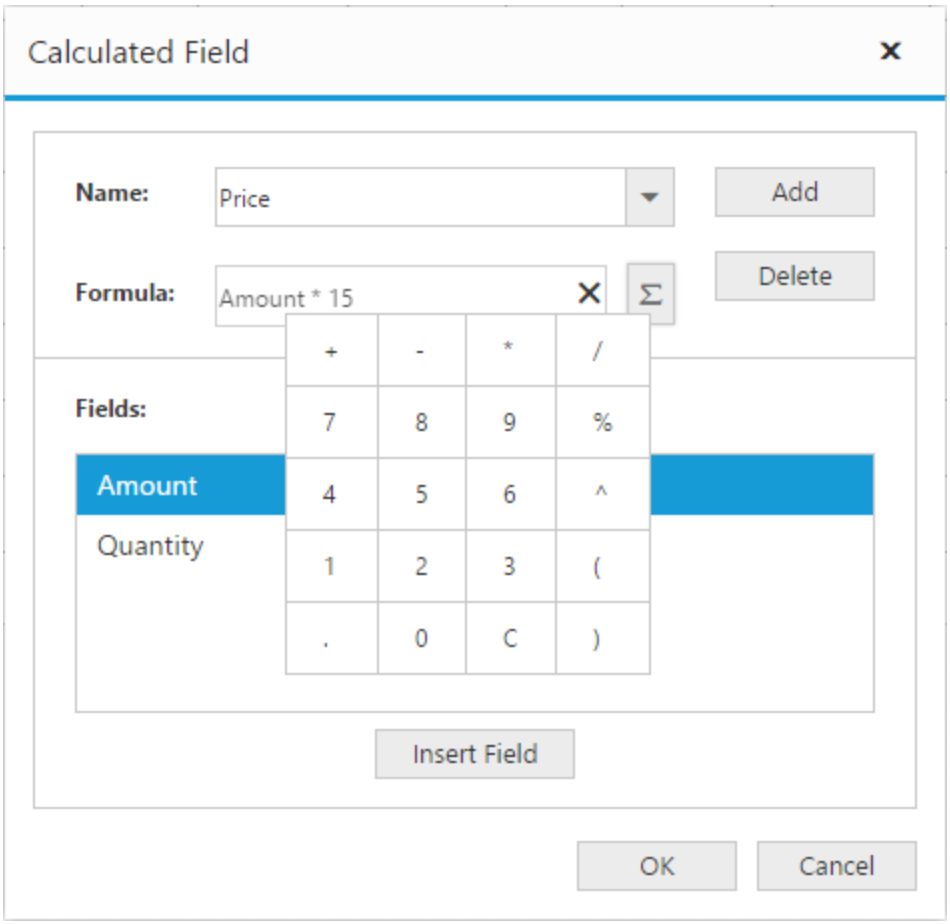
Note: This feature is applicable only for Relational data source.

PivotGrid provides support to insert a new calculated field based on the existing Pivot Fields either through Calculated Field dialog or code behind.

Through UI

To insert a new calculated Field, open the Calculated Field dialog using the Grouping Bar context menu. We can define "Name" for the new Calculated Field and "Formula" can be entered by inserting required fields through Fields section. For inserting numbers and operators, you can use formula pop-up as shown in the below screen-shot.

Copyright © 2001 - 2020 Syncfusion Inc.



Click **Add** for adding the respective Calculated Field and **OK** to populate the PivotGrid control.

Through Code-behind

For client mode, Calculated Field can be created at code-behind by defining formula based on the existing Pivot Fields in the PivotGrid. To indicate a field as a calculated field we need to set `IsCalculatedField` property to true and `Formula` property to set the expression.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableGroupingBar(true).DataSo
urce(dataSource => dataSource.Values(values => {
values.FieldName("Amount").Add();
values.FieldName("Price").FieldCaption("Price").IsCalculatedField(true).F
ormula("Amount * 15").Add(); }))
```

For server mode, we need to set **CalculationType** property to **CalculationType.Formula** and **Formula** property to set the expression for the PivotComputation item in the PivotReport.

C#

```
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Country",
FieldHeader = "Country", TotalHeader = "Total" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName =
"Product", FieldHeader = "Product", TotalHeader = "Total" });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo  {
CalculationName = "Price",
FieldHeader = "Price",
FieldName = "Price",
CalculationType = CalculationType.Formula,
Formula = "Amount * 10 + 12"
});
return pivotSetting;
}
```

Drag field here

Amount ✕ Price ✕ Product ▼ ∧ ✕

Country ▼ ∧ ✕	Bike		Car		Van		Grand Total	
	Amount	Price	Amount	Price	Amount	Price	Amount	Price
Canada	250	3750	500	7500	300	4500	1050	15750
France	200	3000	300	4500	400	6000	900	13500
Germany	300	4500	400	6000	350	5250	1050	15750
United Kingdom	150	2250					150	2250
United States	300	4500	650	9750	500	7500	1450	21750
Grand Total	1200	18000	1850	27750	1550	23250	4600	69000

Cell Editing

Information: This feature is applicable only for Relational data source.

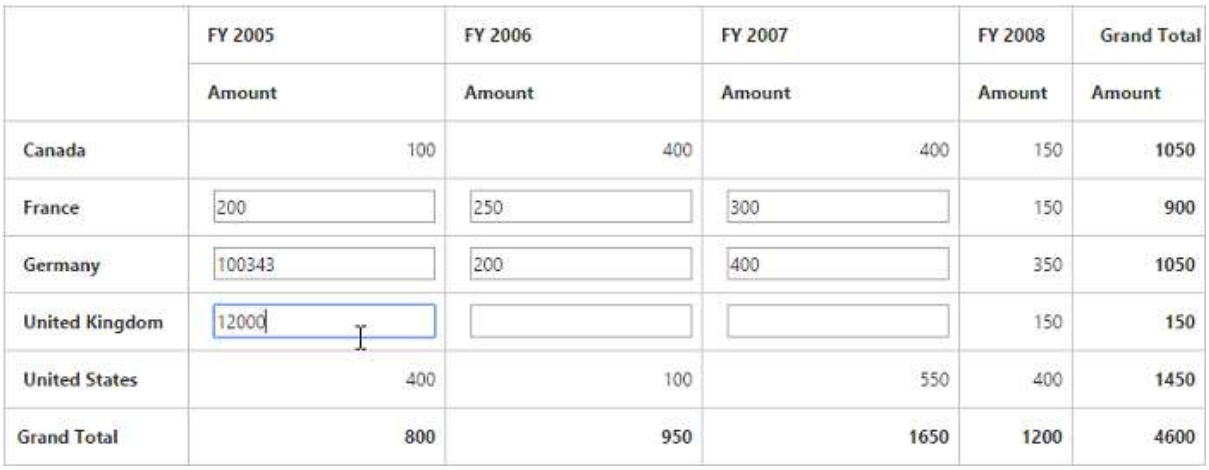
Cell editing allows you to edit and alter the values in PivotGrid. The summary values will be recreated based on the edited values. By selecting multiple cells (like in cell selection feature), you can edit multiple cells at the same time.

You can enable cell editing option in PivotGrid by setting the EnableCellEditing property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableCellEditing(false).DataS
ource(.....)
```

Copyright © 2001 - 2020 Syncfusion Inc.



	FY 2005	FY 2006	FY 2007	FY 2008	Grand Total
	Amount	Amount	Amount	Amount	Amount
Canada	100	400	400	150	1050
France	200	250	300	150	900
Germany	100343	200	400	350	1050
United Kingdom	12000			150	150
United States	400	100	550	400	1450
Grand Total	800	950	1650	1200	4600

Defer Update

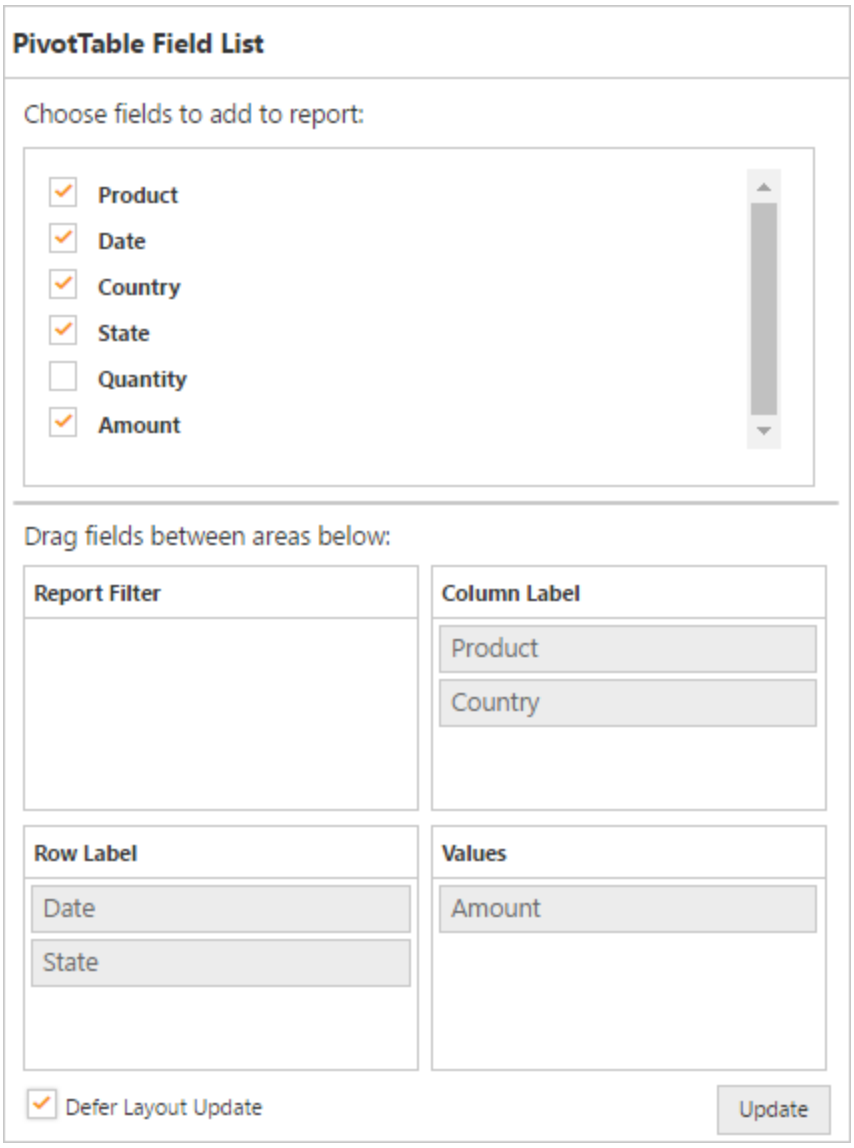
Information: This feature is applicable for Relational datasource only at Server Mode.

Defer Update support allows you to refresh the control only on-demand and not during every UI interaction.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/RelationalSe
rvice")).ClientSideEvents(events =>
events.AfterServiceInvoke("OnAfterServiceInvoke"))
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner").Layout(Pivo
tSchemaDesignerLayout.Excel)
<script type="text/javascript">
OnAfterServiceInvoke = function(evt) {
if (evt.action == "initialize") {
var PivotSchemaDesigner =
$("#PivotSchemaDesigner").data('ejPivotSchemaDesigner');
if (PivotSchemaDesigner.model.pivotControl == null) {
PivotSchemaDesigner.model.pivotControl = this;
PivotSchemaDesigner.model.enableWrapper = true;
PivotSchemaDesigner.model.layout = "excel";
PivotSchemaDesigner._load();
}
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Sub Total Hiding

Note: This feature is applicable only for Relational data source.

You can hide the **Sub Total** for respective fields in rows and columns by setting the ShowSubTotal property to false.

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Date").FieldCaption("Date").ShowSubTotal(false).Add();
rows.FieldName("Product").FieldCaption("Product").Add();
}).Columns(columns => {
columns.FieldName("Country").FieldCaption("Country").Add();
columns.FieldName("State").FieldCaption("State").Add(); }).Values(values
=> { values.FieldName("Amount").Add();
values.FieldName("Quantity").Add(); }))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Server Mode

C#

```
private PivotReport BindDefaultData()
{
PivotReport pivotSetting = new PivotReport();
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Date",
FieldHeader = "Date", TotalHeader = "Total", ShowSubTotal = false });
pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Product",
FieldHeader = "Product", TotalHeader = "Total", ShowSubTotal = true });
pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName =
"Country", FieldHeader = "Country", TotalHeader = "Total", ShowSubTotal =
false });
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "C", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });
return pivotSetting;
}
```

		Canada		France		Germany		United Kingdom		United States		Grand Total	
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
▼ FY 2005	Bike	100	2	200	2	100	2			200	4	600	10
	Van									200	4	200	4
▼ FY 2006	Bike									100	2	100	2
	Car	200	4									200	4
	Van	200	3	250	4	200	3					650	10
▼ FY 2007	Car	300	4	300	3	250	3			250	4	1100	14
	Van	100	1			150	3			300	4	550	8
▼ FY 2008	Bike	150	3			200	4	150	5			500	12
	Car					150	4			400	6	550	10
	Van			150	2							150	2
Grand Total		1050	17	900	11	1050	19	150	5	1450	24	4600	76

Exporting

The PivotGrid control can be exported to the following file formats.

- Excel
- Word
- PDF
- CSV

The PivotGrid control can be exported by invoking **“exportPivotGrid”** method, with an appropriate export option as parameter.

JSON Export

Information: By default JSON export mode will be applied for server and client mode.

ASPX-CS

```
//If you want to render PivotGrid in Client Mode.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => { values.FieldName("Amount").Add(); }))
//If you want to render PivotGrid in Server Mode.
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("../Relational
Service"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("ExcelExport","fileName");
}
</script>
```

To achieve exporting, we need to add **“Syncfusion.EJ.Export”** dependency library into the application.

When PivotGrid is rendered, a method needs to be added in MVC controller file of the application and we need to import **“Syncfusion.EJ.Export”** namespace in the controller file.

C#

```
//...
using Syncfusion.EJ.Export;
public void ExcelExport()
{
PivotGridExcelExport pGrid = new PivotGridExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToExcel(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Excel Export

User can export the contents of PivotGrid to an Excel document for future archival, references and analysis purposes.

To achieve Excel export, the controller method name **“ExcelExport”** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("ExcelExport","fileName");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void ExcelExport()
{
PivotGridExcelExport pGrid = new PivotGridExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToExcel(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Word Export

User can export the contents of PivotGrid to a Word document for future archival, references and analysis purposes.

To achieve Word export, controller method name **"WordExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("WordExport","fileName");
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void WordExport()
{
PivotGridWordExport pGrid = new PivotGridWordExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToWord(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

PDF Export

User can export the contents of PivotGrid to a PDF document for future archival, references and analysis purposes.

To achieve Word export, controller method name **"PDFExport"** and file name is sent as the parameter.

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("PDFExport","fileName");
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void PDFExport()
{
PivotGridPDFExport pGrid = new PivotGridPDFExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToPDF(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

CSV Export

User can export the contents of PivotGrid to a CSV document for future archival, references and analysis purposes.

To achieve CSV export, controller method name **"CSVExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("CSVExport","fileName");
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void CSVExport()
{
PivotGridCSVExport pGrid = new PivotGridCSVExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToCSV(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Customize the export document name

For customizing file name, we need to send file name as parameter to the **“exportPivotGrid”** method along with method name.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("ExcelExport","fileName");
}
```

PivotEngine Export

Information: This feature is applicable only at server mode operation.

In order to perform exporting with the use of PivotEngine available in server-side, the 'exportMode' property obtained in the “BeforeExport” event is set to the value "ej.PivotGrid.ExportMode.PivotEngine" as shown below.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("../Relational
Service")).ClientSideEvents(clientSideEvents =>
clientSideEvents.BeforeExport("Export"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
function Export(args) {
args.exportMode = ej.PivotGrid.ExportMode.PivotEngine;
}
</script>
```

When PivotGrid is rendered in server mode, a service method needs to be added in WCF/WebAPI for server side operations.

For WebAPI controller, the below method needs to be added.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string fileName = "Sample";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
HttpContext.Current.Response);
}
```

For WCF service, the below method needs to be added.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
System.Web.HttpContext.Current.Response);
}
```

Excel Export

User can export the contents of PivotGrid to an Excel document for future archival, references and analysis purposes.

To achieve Excel export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

For Excel export, **“ej.PivotGrid.ExportOptions.Excel”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as Excel in the exportPivotGrid method for
ServerMode
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
```

Word Export

User can export the contents of PivotGrid to a Word document for future archival, references and analysis purposes.

To achieve Word export, we need to add the following dependency libraries into the application.

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.Compression.Base
- Syncfusion.DocIO.Base

For Word export, **“ej.PivotGrid.ExportOptions.Word”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as Word in the exportPivotGrid method
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Word);
}
```

PDF Export

User can export the contents of PivotGrid to a PDF document for future archival, references and analysis purposes.

To achieve PDF export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base

For PDF export, **“ej.PivotGrid.ExportOptions.PDF”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as PDF in the exportPivotGrid method
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.PDF);
}
```

CSV Export

User can export contents of PivotGrid to a CSV document for future archival, references and analysis purposes.

For CSV export, **“ej.PivotGrid.ExportOptions.CSV”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as CSV in the exportPivotGrid method
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.CSV);
}
```

File format selection

Information: This option is applicable only for PivotGrid when exporting to Excel document.

You can set the option for exporting the control to Excel document either in *.xls* or *.xlsx* format, using `fileFormat` property inside the `BeforeExport` event.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: By default excel document will be exported to ".xls" format using PivotEngine export.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("../Relational
Service")).ClientSideEvents(clientSideEvents =>
clientSideEvents.BeforeExport("Export"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
function Export(args) {
args.exportMode = ej.PivotGrid.ExportMode.PivotEngine;
args.fileFormat = ".xlsx"; //you can set the excel sheet format here
}
</script>
```

Customize the export document name

For customizing name in WebAPI controller, below code snippet is used.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
string fileName = " File name is customized here ";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
HttpContext.Current.Response);
}
```

For customizing name in WCF Service, below code snippet is used.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
string fileName = " File name is customized here ";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
System.Web.HttpContext.Current.Response);
}
```

Exporting Customization

You can add title and description to the exporting document by using the title and description properties respectively obtained in the BeforeExport event. Similarly, you can enable or disable styling on the exported document by using the exportWithStyle property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.BeforeExport("Exporting"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//JSON Export
pGridObj.exportPivotGrid("ExcelExport","fileName");
//PivotEngine Export
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
function Exporting(args) {
args.title = "PivotGrid";
args.description = "Displays both OLAP and Relational datasource in
tabular format";
args.exportWithStyle = true;   // by default it sets as true. It improves
performance on exporting huge data when it sets as false.
}
</script>
```

You can also edit the exporting document with the use of a server-side event for required exporting option.

C#

```
//...
using Syncfusion.EJ.Export;
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
//Following methods needs to be added in MVC controller file of the
application for JSON export
public void ExcelExport()
{
PivotGridExcelExport pGrid = new PivotGridExcelExport();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExcelExport += pGrid_ExcelExport;
pGrid.ExportToExcel(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
public void PDFExport()
{
PivotGridPDFExport pGrid = new PivotGridPDFExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.AddPDFHeaderFooter += pGrid_AddPDFHeaderFooter;
pGrid.PDFExport += pGrid_PDFExport;
pGrid.ExportToPDF(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_PDFExport(object sender, Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can customize exporting document here.
}
void pGrid_AddPDFHeaderFooter(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can add header/footer information to the PDF document.
}
public void CSVExport()
{
PivotGridCSVExport pGrid = new PivotGridCSVExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.CSVExport += pGrid_CSVExport;
pGrid.ExportToCSV(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_CSVExport(object sender, string csvString)
{
//You can customize exporting document here.
}
public void WordExport()
{
PivotGridWordExport pGrid = new PivotGridWordExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.WordExport += pGrid_WordExport;
pGrid.ExportToWord(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
void pGrid_WordExport(object sender, Syncfusion.DocIO.DLS.WordDocument
document)
{
//You can customize exporting document here.
}
//Following service methods needs to be added in WEBAPI controller file
for PivotEngine export
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
Dictionary<string, string> gridParams =
serializer.Deserialize<Dictionary<string, string>>(args);
htmlHelper.PopulateData(gridParams["currentReport"]);
htmlHelper.ExcelExport += htmlHelper_ExcelExport;
htmlHelper.WordExport += htmlHelper_WordExport;
htmlHelper.AddPDFHeaderFooter += htmlHelper_AddPDFHeaderFooter;
htmlHelper.PDFExport += htmlHelper_PDFExport;
htmlHelper.CSVExport += htmlHelper_CSVExport;
string fileName = "Sample";
htmlHelper.ExportPivotGrid(ProductSales.GetSalesData(), args, fileName,
System.Web.HttpContext.Current.Response);
}
void htmlHelper_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
void htmlHelper_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
void htmlHelper_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void htmlHelper_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void htmlHelper_CSVExport(object sender, string csvString)
{
//You can customize exporting document here.
}
```

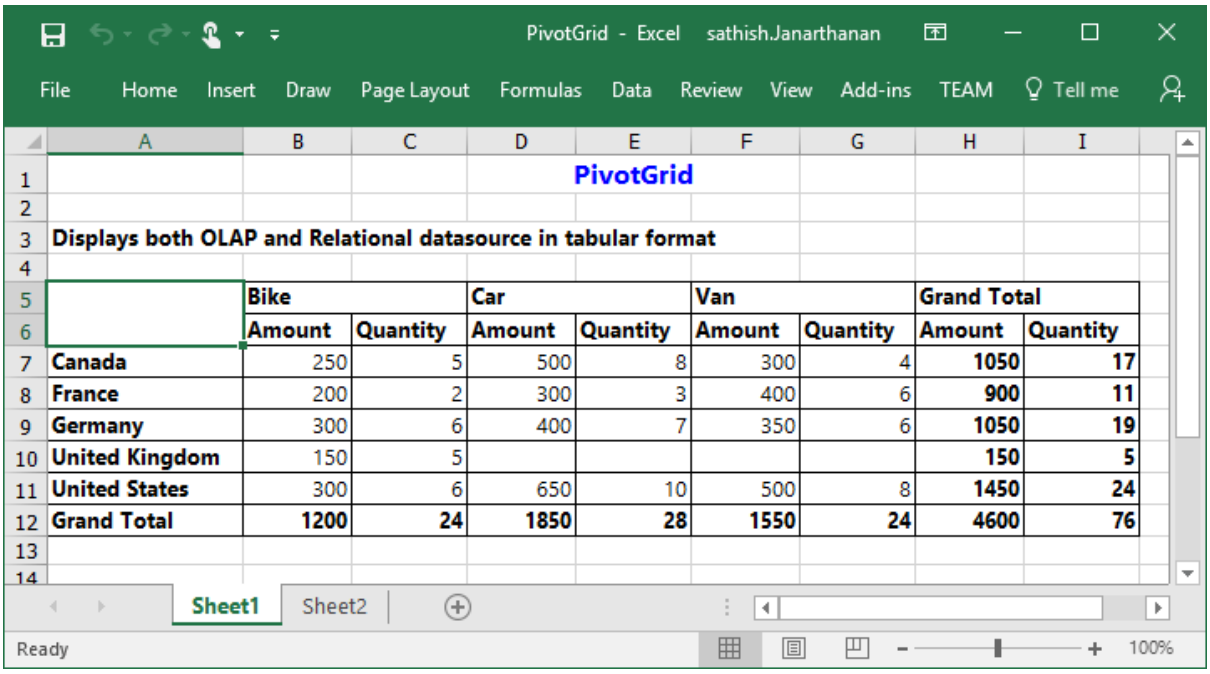
Exporting complete data on Paging

When paging is enabled, you can export the complete data by enabling the `EnableCompleteDataExport` property. It is supported in both types of JSON and PivotEngine export and it is applicable for all kinds of exporting formats available in PivotGrid.

ASPX-CS

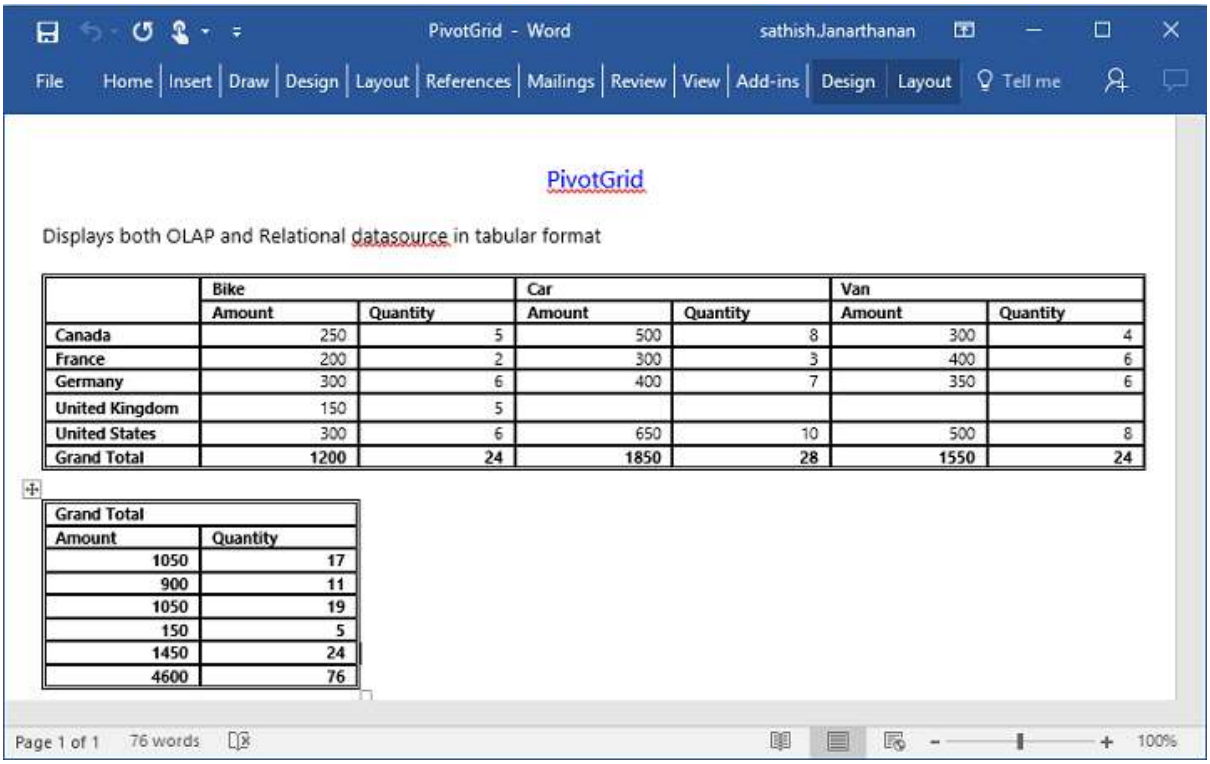
```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableCompleteDataExport(true)
```

The below screenshot shows the PivotGrid control exported to Excel document.



	Bike		Car		Van		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	250	5	500	8	300	4	1050	17
France	200	2	300	3	400	6	900	11
Germany	300	6	400	7	350	6	1050	19
United Kingdom	150	5					150	5
United States	300	6	650	10	500	8	1450	24
Grand Total	1200	24	1850	28	1550	24	4600	76

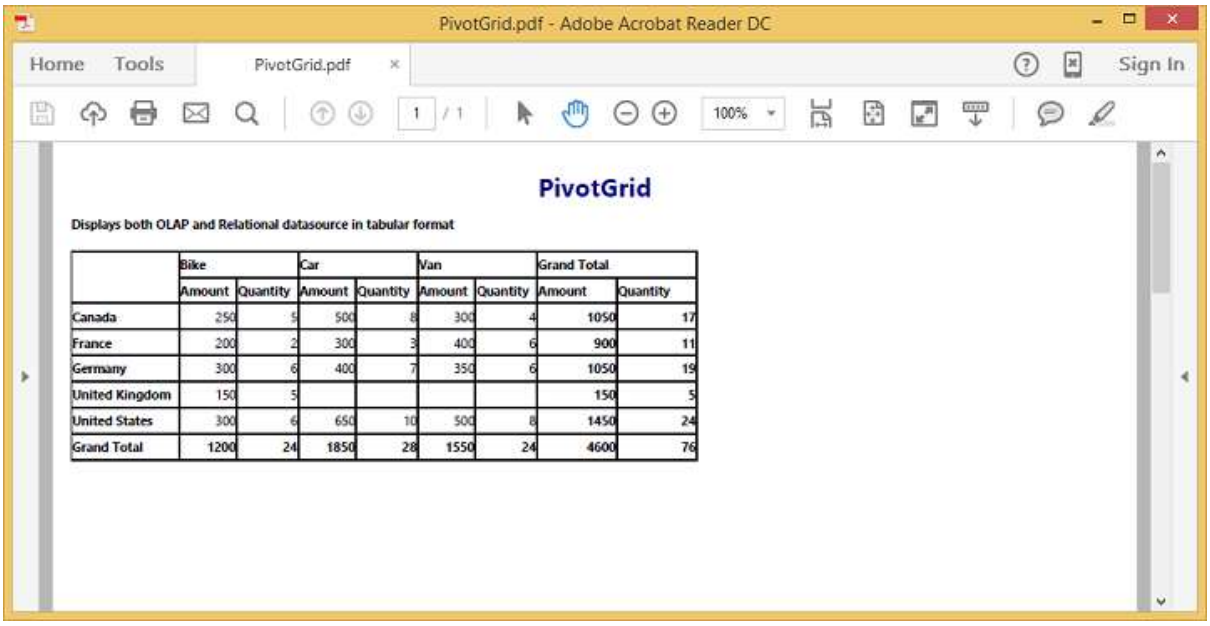
The below screenshot shows the PivotGrid control exported to Word document.



	Bike		Car		Van	
	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	250	5	500	8	300	4
France	200	2	300	3	400	6
Germany	300	6	400	7	350	6
United Kingdom	150	5				
United States	300	6	650	10	500	8
Grand Total	1200	24	1850	28	1550	24

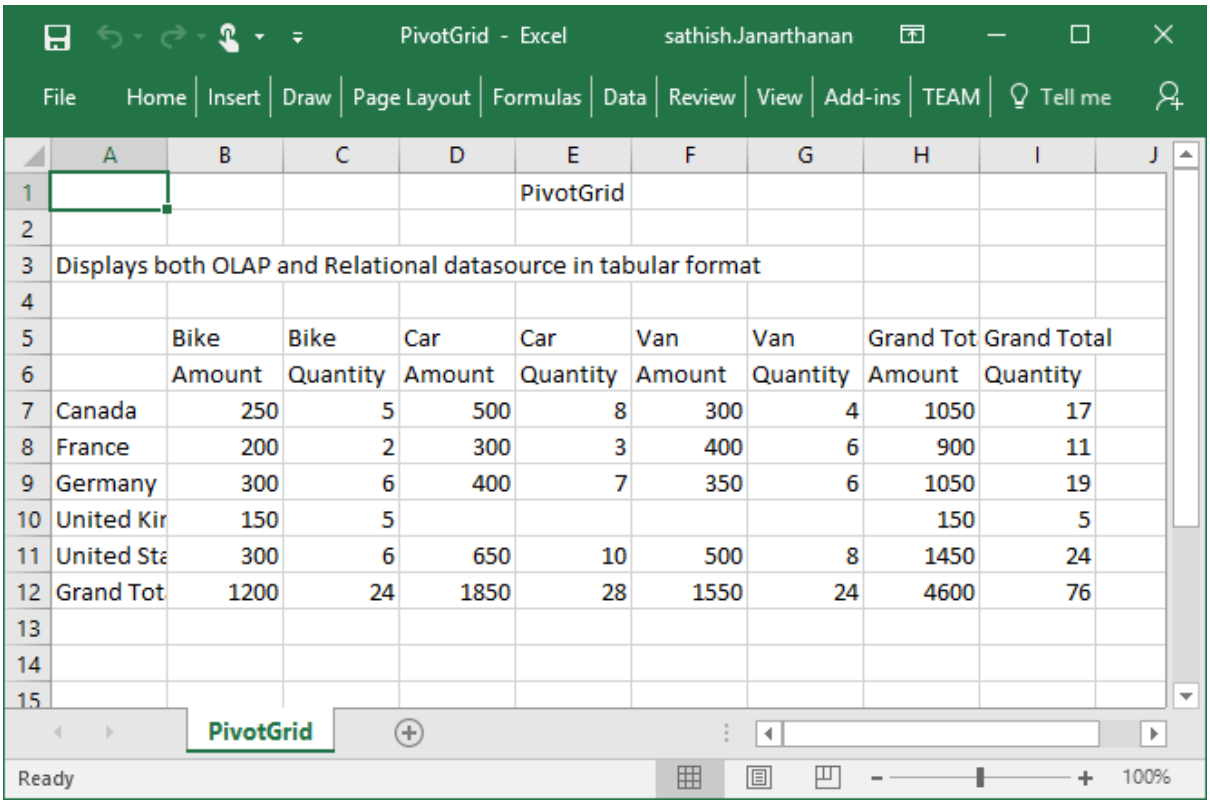
Grand Total	
Amount	Quantity
1050	17
900	11
1050	19
150	5
1450	24
4600	76

The below screenshot shows the PivotGrid control exported to PDF document.



	Bike		Car		Van		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	250	5	500	8	300	4	1050	17
France	200	2	300	3	400	6	900	11
Germany	300	6	400	7	350	6	1050	19
United Kingdom	150	5					150	5
United States	300	6	650	10	500	8	1450	24
Grand Total	1200	24	1850	28	1550	24	4600	76

The below screenshot shows the PivotGrid control exported to CSV document.



		Bike	Bike	Car	Car	Van	Van	Grand Tot	Grand Total
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
7	Canada	250	5	500	8	300	4	1050	17
8	France	200	2	300	3	400	6	900	11
9	Germany	300	6	400	7	350	6	1050	19
10	United Kir	150	5					150	5
11	United Sta	300	6	650	10	500	8	1450	24
12	Grand Tot	1200	24	1850	28	1550	24	4600	76

Note: Grand total and sub-totals can be hidden while exporting the document.

Collapse by default

Information: This feature is applicable only for Relational datasource.

Allows you to collapse all members displayed in the grid. You can enable collapsing all members by default in the PivotGrid by setting the EnableCollapseByDefault(/api/js/ejpivotgrid#members: enablecollapsebydefault) property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableCollapseByDefault(true)
```

	Bike	Car	Van	Grand Total
	Quantity	Quantity	Quantity	Quantity
▸ Canada	5	8	4	17
▸ France	2	3	6	11
▸ Germany	6	7	6	19
▸ United Kingdom	5			5
▸ United States	6	10	8	24
Grand Total	24	28	24	76

Group by date

Information: This feature is applicable for Relational datasource alone (in Client Mode).

Allows the user to categorize the date type field and showcase them based on year, quarter, month and day formats.

ASPX-CS

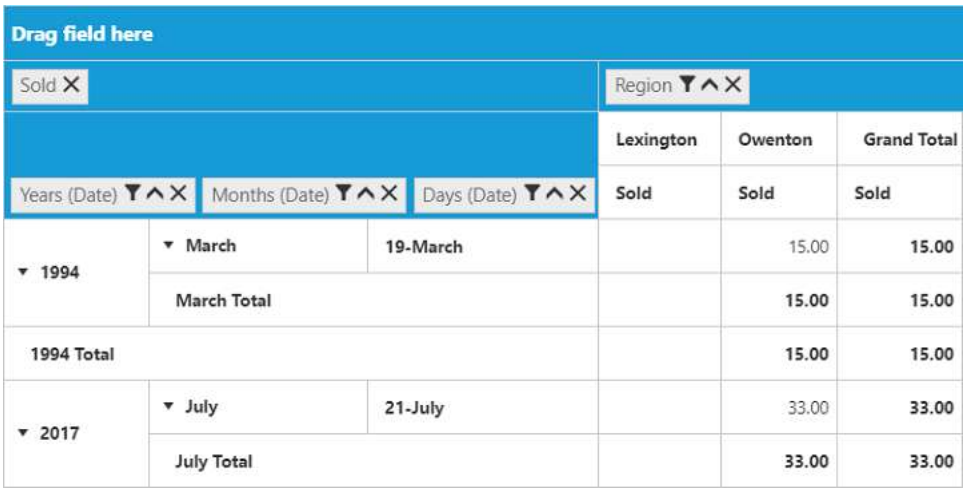
```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
rows.FieldName("State").FieldCaption("State").Add(); }).Columns(columns
=> {
columns.FieldName("Date").FieldCaption("Date").Format("date").FormatStrin
g("yyyy-MMM-ddd").Delimiter("-").GroupByDate(groupByDate => {
groupByDate.Interval(groupinterval => { groupinterval.Add("yyyy");
groupinterval.Add("MMM-dd"); }); }).Add(); }).Values(values => {
values.FieldName("Product").Add(); values.FieldName("Quantity").Add();
}))
```

The properties associated with group by date option is,

- Format - To set the data type format.
- FormatString - Specifies the structure of the date format.

Copyright © 2001 - 2020 Syncfusion Inc.

- Delimiter - Specifies the separator of the date values.
- GroupByDate.Interval - Specifies the pattern in which date type to be displayed.



OLAP

KPI

Key Performance Indicators (KPI) are business metric that help to figure out the progress of an enterprise in meeting its business goals.

The different indicators available in KPI are:

- KPI Value: A physical measure or a calculated measure.
- KPI Goal: Defines the target for the measure.
- KPI Status: Evaluates the current status of the value compared to the goal.
- KPI Trend: Evaluate the current trend of the value compared to the goal.

The **“KpiElements”** class in OLAP Base library holds the KPI name and when its object is added to an OlapReport, you can view the resultant information in PivotGrid.

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Date].[Fiscal]").Add();
}).Columns(columns => { columns.FieldName("[Product].[Product
Categories]").Add(); }).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
measures.FieldName("[Measures].[Growth in Customer Base Trend]").Add();
}).Axis(AxisName.Column).Add();
}).Data("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Accessories		Bikes	
	Internet Sales Amount	Growth in Customer Base Trend	Internet Sales Amount	Growth in Customer Base Trend
Australia	$138,690.63		$8,852,050.00	
Canada	$103,377.85		$1,821,302.39	
France	$63,406.78		$2,553,575.71	
Germany	$62,232.59		$2,808,514.35	
United Kingdom	$76,630.04		$3,282,842.66	
United States	$256,422.07		$8,999,859.53	
Total	$700,759.96		$28,318,144.65	

Server Mode

C#

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Sample Report";
olapReport.CurrentCubeName = "Adventure Works";
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement { Name = "Gross
Profit" });
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
KpiElements kpiElement = new KpiElements();
kpiElement.Elements.Add(new KpiElement { Name = "Revenue", ShowKPIStatus
= true, ShowKPIGoal = false, ShowKPITrend = true, ShowKPIValue = true });
olapReport.CategoricalElements.Add(kpiElement);
olapReport.CategoricalElements.Add(measureElementColumn);
olapReport.SeriesElements.Add(dimensionElementRow);
```

	Sales Amount	Revenue Status	Revenue Trend	Gross Profit
FY 2002	$23,360,526.01			$2,833,920.51
FY 2003	$33,683,804.82			$3,547,898.07
FY 2004	$52,714,102.75			$6,141,181.94
FY 2005	$50,840.63			$28,365.72
Total	$109,809,274.20			$12,551,366.25

Named Set

Named set is a multidimensional expression (MDX) that returns a set of dimension members, which can be created by combining cube data, arithmetic operators, numbers and functions.

Client Mode

You can bind the Named Sets in PivotGrid by setting it's unique name in the `FieldName` property either in row or column axis and `IsNamedSets` Boolean property to "true".

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("[Date].[Fiscal]").Add();
}).Columns(columns => { columns.FieldName("[Core Product
Group]").IsNamedSets(true).Add(); }).Values(values => {
values.Measures(measures => { measures.FieldName("[Measures].[Internet
Sales Amount]").Add(); }).Axis(AxisName.Column).Add();
}).Data("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works"))
```

	Accessories	Bikes				Clothing
		Mountain Bikes	Road Bikes	Touring Bikes	Total	
	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount	Internet Sales Amount
▸ FY 2002		$1,341,121.04	$5,730,963.20		$7,072,084.24	
▸ FY 2003		$2,214,410.01	$3,547,724.28		$5,762,134.30	
▸ FY 2004	$667,015.32	$6,397,228.51	$5,241,896.55	$3,844,801.05	$15,483,926.11	$322,676.62
▸ FY 2005	$33,744.64					$17,095.99
Total	$700,759.96	$9,952,759.56	$14,520,584.04	$3,844,801.05	$28,318,144.65	$339,772.61

Server Mode

You can add Named Sets in the PivotGrid by using "NamedSetElement" Class in the OlapReport.

C#

```
OlapReport olapReport = new OlapReport();
olapReport.Name = "Customer Report";
olapReport.CurrentCubeName = "Adventure Works";
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
MeasureElements measureElementColumn = new MeasureElements();
measureElementColumn.Elements.Add(new MeasureElement {
Name = "Internet Sales Amount"
});
NamedSetElement dimensionElementColumn = new NamedSetElement();
dimensionElementColumn.Name = "Core Product Group";
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(measureElementColumn);
olapReport.SeriesElements.Add(dimensionElementRow);
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Accessories	Bikes			Clothing
		Mountain Bikes	Road Bikes	Touring Bikes	
	Internet Sales Amount				
▸ FY 2002		$1,341,121.04	$5,730,963.20		
▸ FY 2003		$2,214,410.01	$3,547,724.28		
▸ FY 2004	$667,015.32	$6,397,228.51	$5,241,896.55	$3,844,801.05	$322,676.62
▸ FY 2005	$33,744.64				$17,095.99

Grouping Bar

Initialization

Grouping Bar allows user to dynamically alter the report by filter and remove operations in the PivotGrid control. Based on the OLAP datasource and report bound to the PivotGrid control, Grouping Bar will be automatically populated. You can enable Grouping Bar option in PivotGrid by setting the `EnableGroupingBar` property to true.

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableGroupingBar(true).DataSo
urce(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();}).Data("https://bi.syncfusion.com/olap/ms
mdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works"))
```

Drag field here

Internet Sales A... | Customer Geography | Measures

Date.Fiscal	▸ Australia	▸ Canada	▸ France	▸ Germany	▸ United Kingdom	▸ United States	Total
	Internet Sales Amount						
▸ FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
▸ FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
▸ FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
▸ FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

Copyright © 2001 - 2020 Syncfusion Inc.

Server Mode

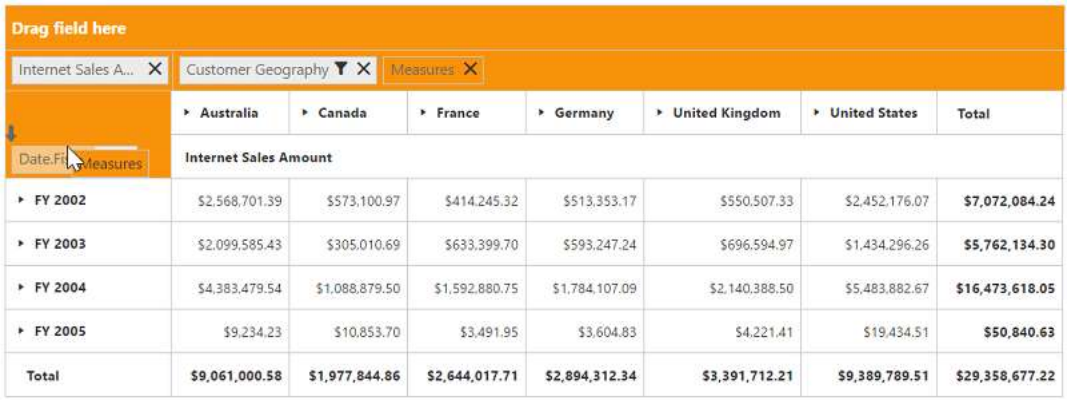
ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/OLAPService")).EnableGroupingBar(true)
```



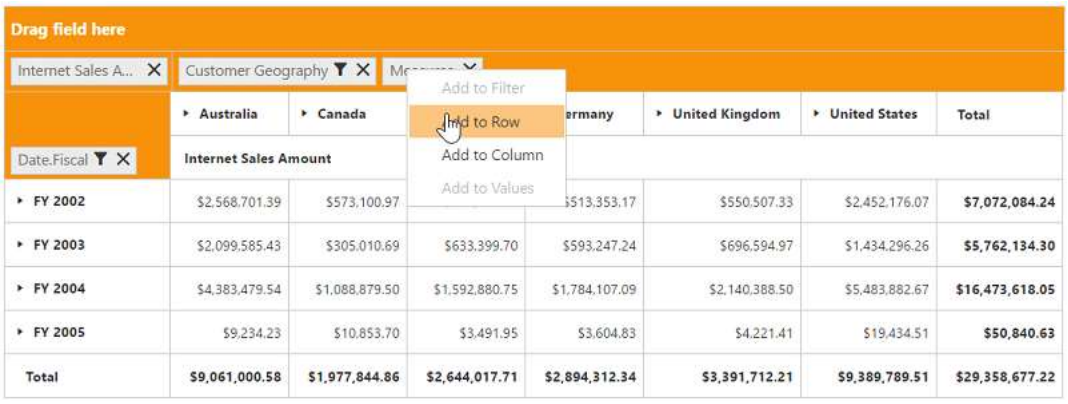
Drag and Drop

You can alter the report on fly through drag-and-drop operation.



Context Menu

You can also alter the report by using context menu.

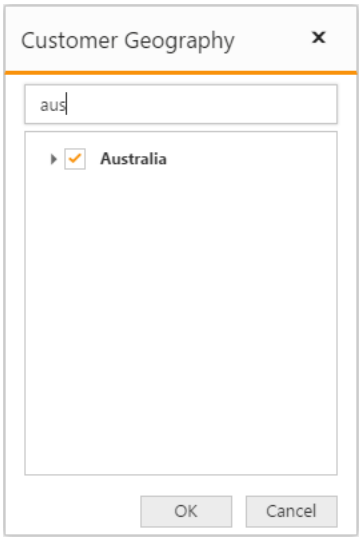


Copyright © 2001 - 2020 Syncfusion Inc.

Searching Values

Search option available in Grouping Bar allows you to search a specific value that needs to be filtered from the list of values inside the filter pop-up window.

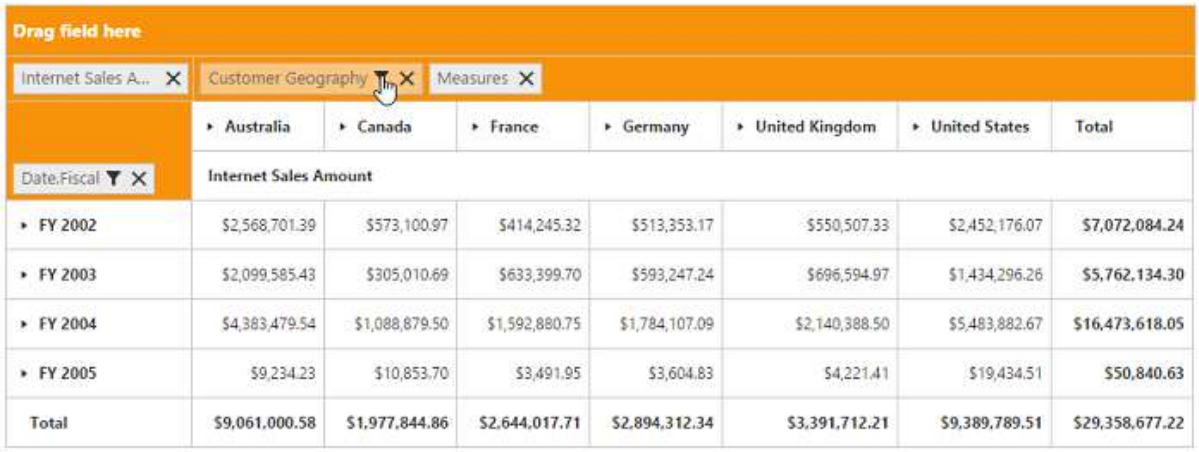


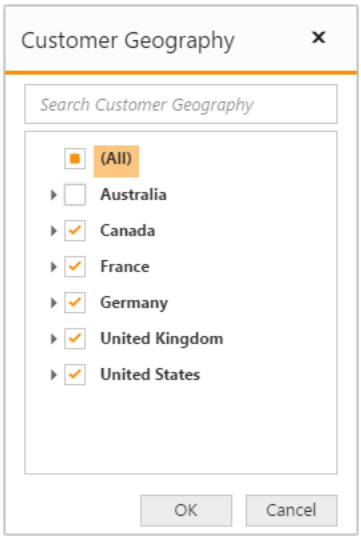


Filtering Values

Filtering option available in Grouping Bar allows you to select a specific set of values that needs to be displayed in the PivotGrid control. At least one value needed to be in checked state while filtering otherwise “Ok” button will be disabled.

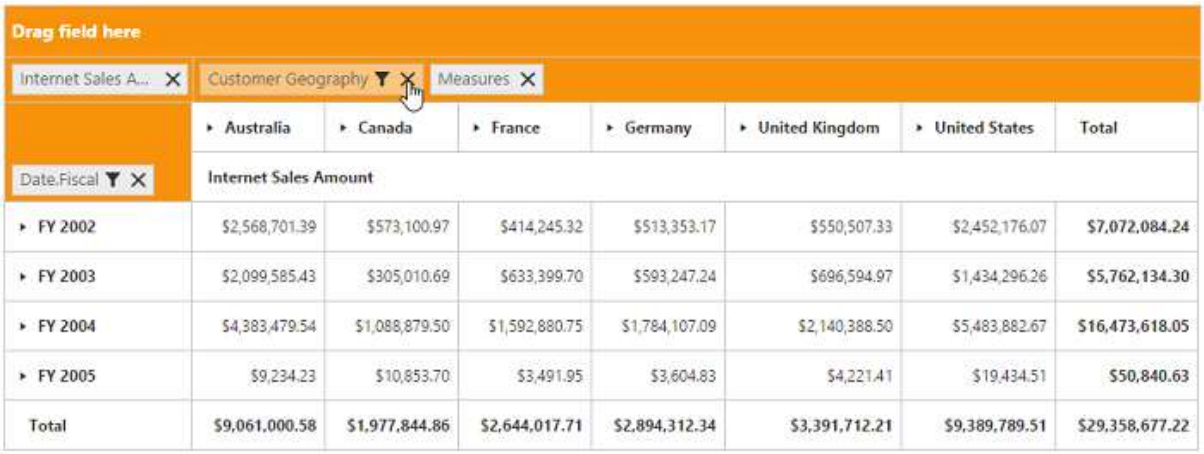
Copyright © 2001 - 2020 Syncfusion Inc.



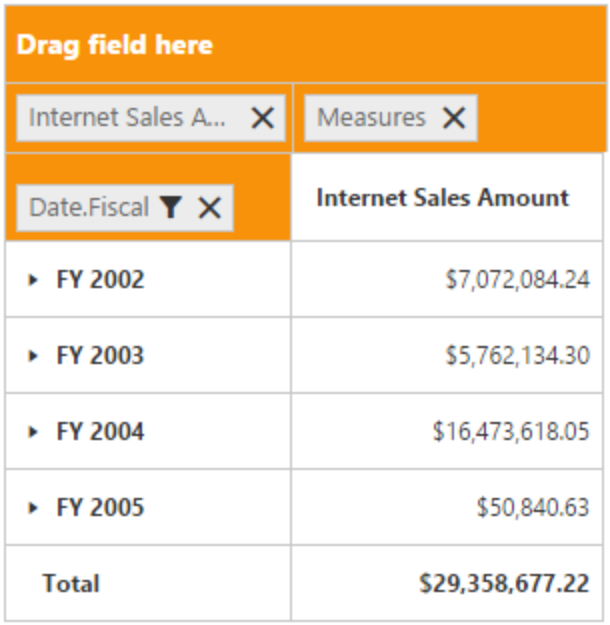


Removing Field

Remove option available in the Grouping Bar allows you to completely remove a specific field from the PivotGrid control. Remove operation can be either achieved by clicking remove icon available inside each field or by dragging and dropping field out of Grouping Bar region.



Copyright © 2001 - 2020 Syncfusion Inc.



PivotTable Field List

Initialization

Field List, also known as Pivot Schema Designer, allows user to add, rearrange, filter and remove fields to show data in PivotGrid exactly the way they want.

Based on the datasource, OLAP, bound to the PivotGrid control, PivotTable Field List will be automatically populated with Cube Information or Field Names. PivotTable Field List provides an Excel like appearance and behavior.

In-order to initialize PivotTable Field List, first you need to define a “div” tag with an appropriate “id” attribute which acts as a container for the control. Then you need to initialize the PivotTable Field List by using the **“PivotSchemaDesigner”** method.

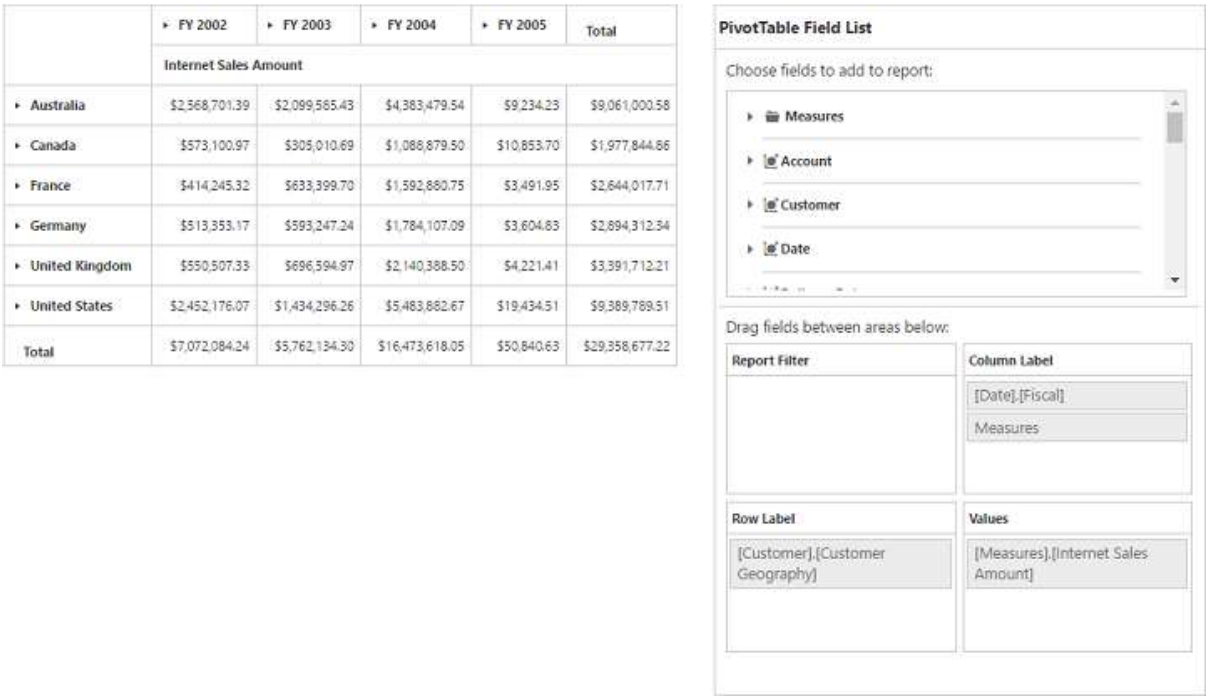
Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts =>
clientSideEvents.RenderSuccess("loadSchemaDesigner")).DataSource(dataSour
ce => dataSource.Values(values => {
values.Axis(AxisName.Column).Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add(); }).Add();
}).Rows(rows => { rows.FieldName("[Customer].[Customer
Geography]").Add();
}).Columns(columns=>{columns.FieldName("[Date].[Fiscal]").Add();}).Data("
https://bi.syncfusion.com/olap/msmdpump.dll").Cube("Adventure
Works").Catalog("Adventure Works DW 2008 SE"))
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner")
<script type="text/javascript">
function loadSchemaDesigner(args) {
var PivotSchemaDesigner =
$("#PivotSchemaDesigner").data('ejPivotSchemaDesigner');
if (PivotSchemaDesigner.model.pivotControl == null) {
PivotSchemaDesigner.model.pivotControl = this;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
PivotSchemaDesigner.model.enableWrapper = true;
PivotSchemaDesigner.model.layout = "excel";
PivotSchemaDesigner._load();
}
args.model.renderSuccess = null;
}
</script>
```



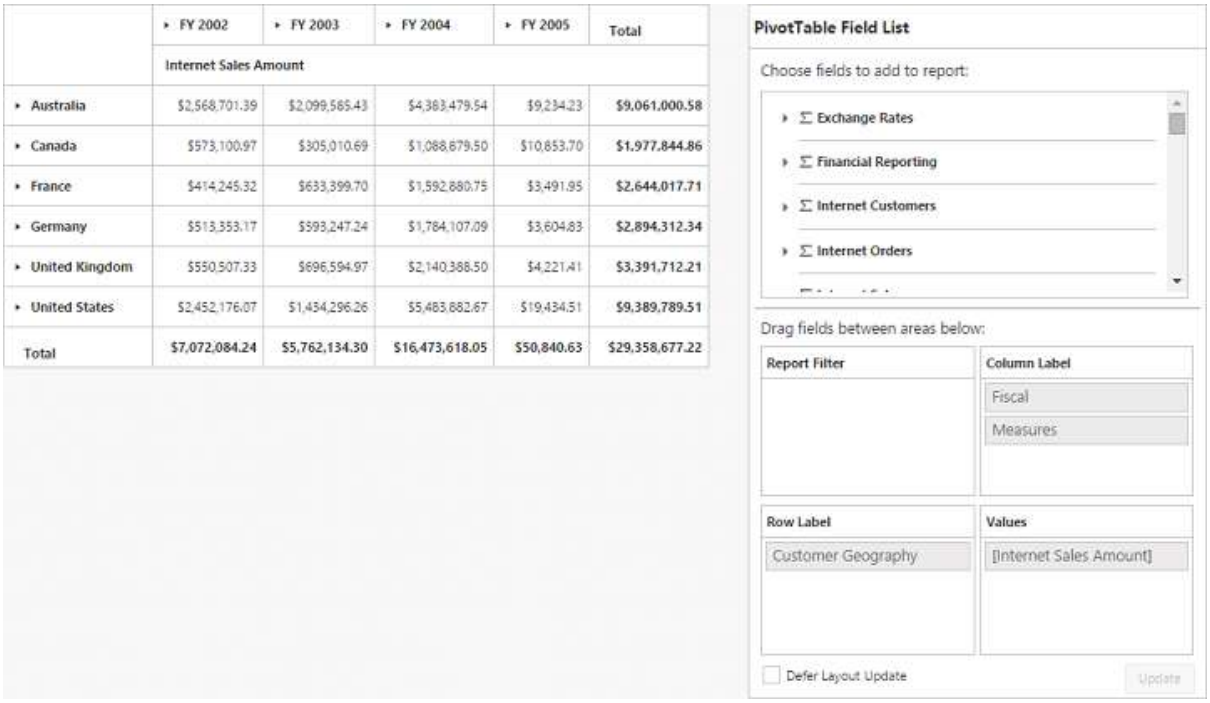
	FY 2002	FY 2003	FY 2004	FY 2005	Total
	Internet Sales Amount				
Australia	$2,568,701.39	$2,099,585.43	$4,383,479.54	$9,234.23	$9,061,000.58
Canada	$573,100.97	$305,010.69	$1,088,879.50	$10,853.70	$1,977,844.86
France	$414,245.32	$633,399.70	$1,592,880.75	$3,491.95	$2,644,017.71
Germany	$513,353.17	$593,247.24	$1,784,107.09	$3,604.83	$2,894,312.34
United Kingdom	$550,507.33	$696,594.97	$2,140,388.50	$4,221.41	$3,391,712.21
United States	$2,452,176.07	$1,434,296.26	$5,483,882.67	$19,434.51	$9,389,789.51
Total	$7,072,084.24	$5,762,134.30	$16,473,618.05	$50,840.63	$29,358,677.22

Server Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("~/OLAPService
")).ClientSideEvents(events =>
events.AfterServiceInvoke("OnAfterServiceInvoke"))
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner").Layout(Pivo
tSchemaDesignerLayout.Excel)
<script type="text/javascript">
OnAfterServiceInvoke = function(evt) {
if (evt.action == "initialize") {
var PivotSchemaDesigner =
$("#PivotSchemaDesigner").data('ejPivotSchemaDesigner');
if (PivotSchemaDesigner.model.pivotControl == null) {
PivotSchemaDesigner.model.pivotControl = this;
PivotSchemaDesigner.model.enableWrapper = true;
PivotSchemaDesigner.model.layout = "excel";
PivotSchemaDesigner._load();
}
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



	FY 2002	FY 2003	FY 2004	FY 2005	Total
	Internet Sales Amount				
Australia	$2,568,701.39	$2,099,565.43	$4,383,479.54	$9,234.23	$9,061,000.58
Canada	$573,100.97	$305,010.69	$1,088,679.50	$10,853.70	$1,977,844.86
France	$414,245.32	$633,399.70	$1,592,880.75	$3,491.95	$2,644,017.71
Germany	$513,353.17	$593,247.24	$1,784,107.09	$3,604.83	$2,894,312.34
United Kingdom	$550,507.33	$696,594.97	$2,140,388.50	$4,221.41	$3,391,712.21
United States	$2,452,176.07	$1,434,296.26	$5,483,882.67	$19,434.51	$9,389,789.51
Total	$7,072,084.24	$5,762,134.30	$16,473,618.05	$50,840.63	$29,358,677.22

Layout

The top portion of the layout shows Field or Cube items in a categorized way. They can be dynamically added into the report either by drag and drop option or through simple check box selection.

On item(s) selection they will be placed in Row section by default except numeric based item(s) or measures, which will alone be placed in the Value section by default.

The bottom portion of the layout is segregated as below.

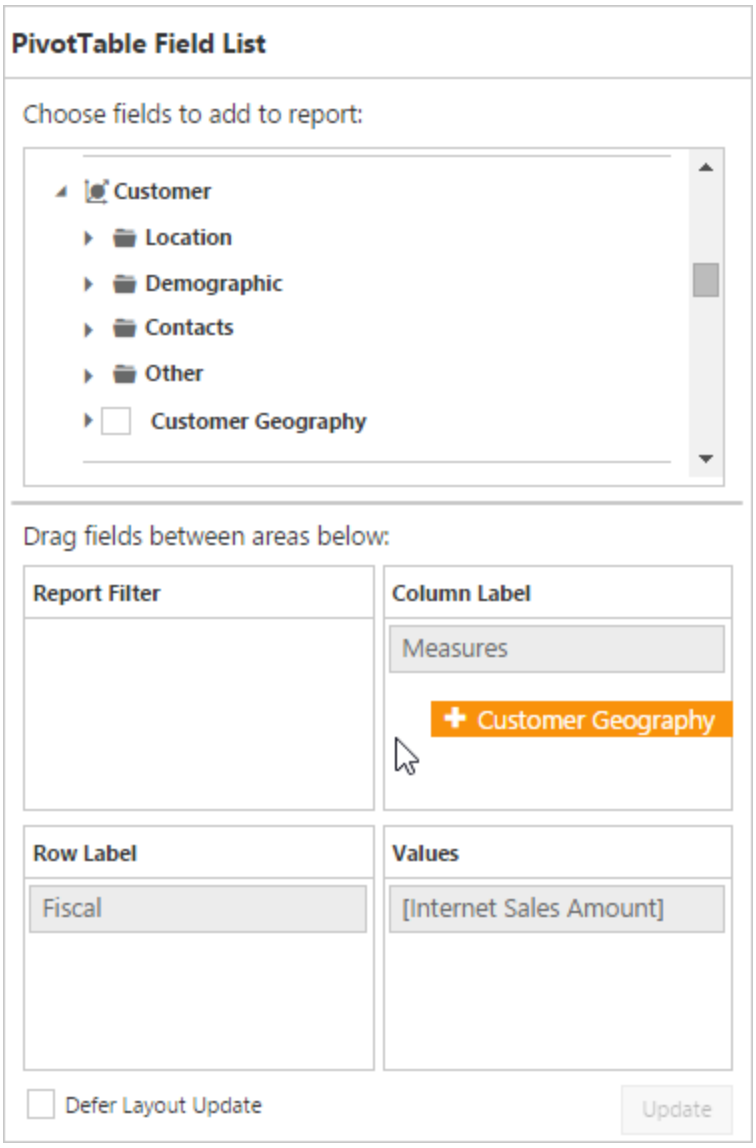
- Report Filter: Exclusively designed to filter an item(s) placed in this particular position of the layout.
- Value Section: The value label usually displays the numeric value item(s) present in the report.
- Column Section: It is used to display item(s) as column header and values in the PivotGrid control.
- Row Section: It is used to display item(s) as row header and values in the PivotGrid control.

UI Interactions

By Drag and Drop

You can alter the report on fly through drag-and-drop operation. You can drag any item from Field List and drop into column, row, value or filter section available at the bottom of the Field List.

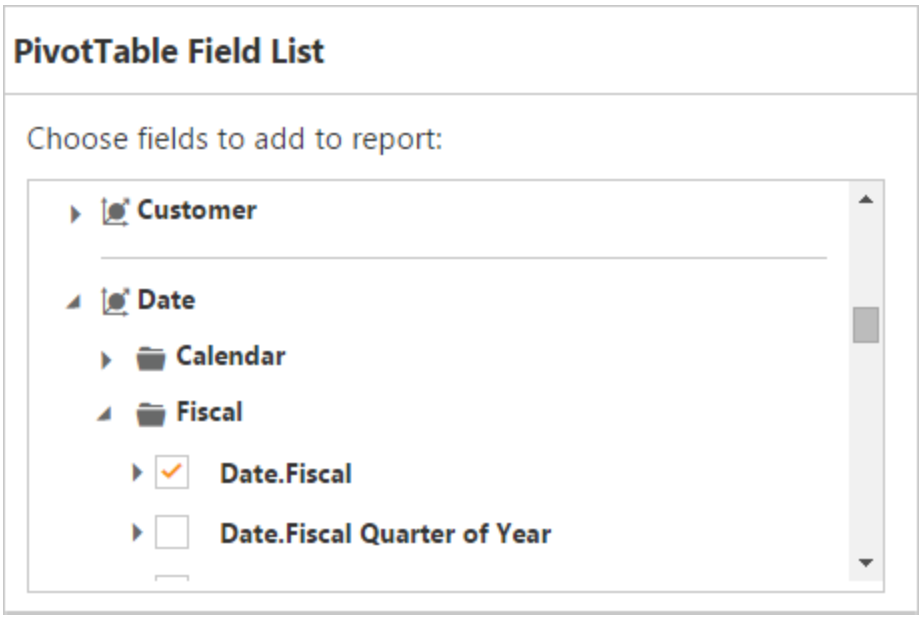
Copyright © 2001 - 2020 Syncfusion Inc.



By Treeview Selection

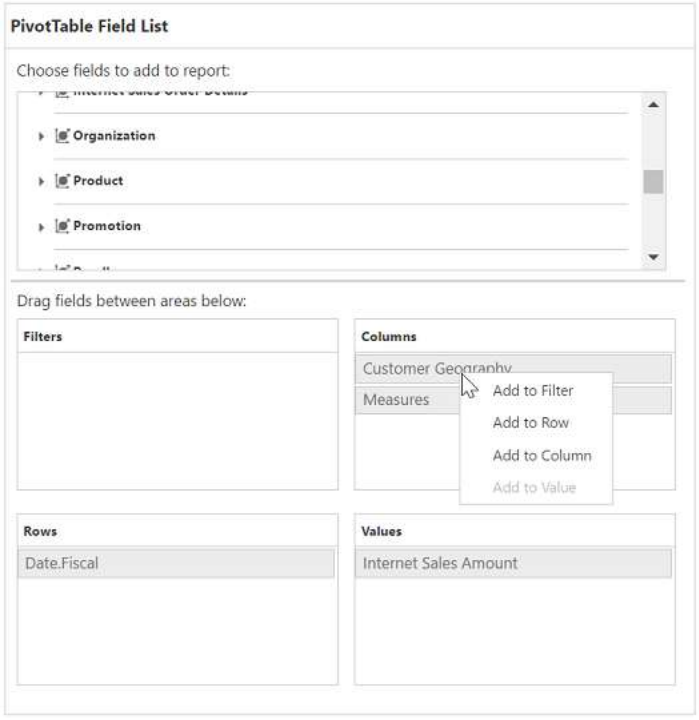
You can also alter the report on fly through check and uncheck option as an alternate. By default, fields will be added to the Row Label when checked.

Copyright © 2001 - 2020 Syncfusion Inc.

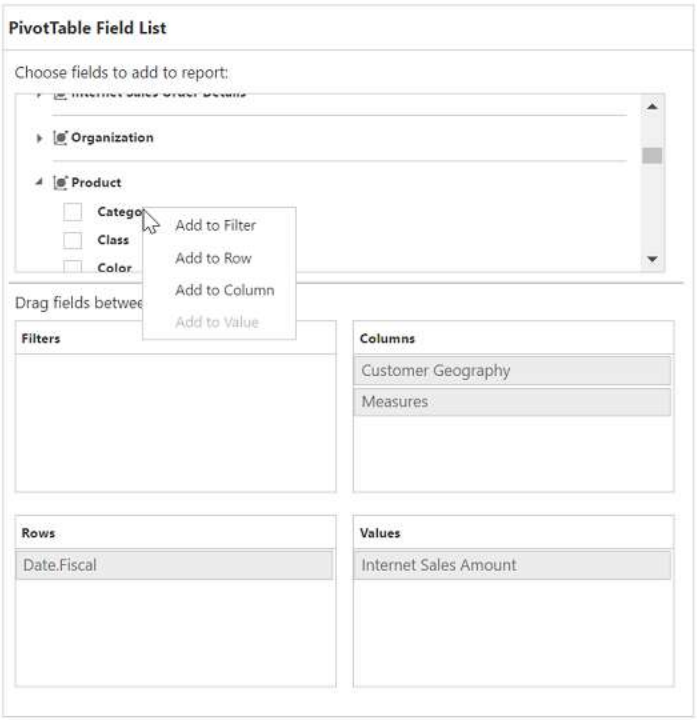


By Context Menu

You can also alter the report by using context menu.

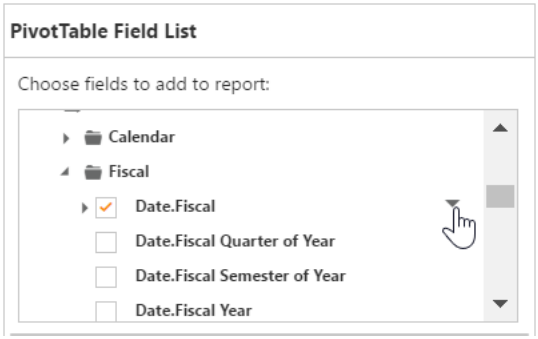


Copyright © 2001 - 2020 Syncfusion Inc.



Searching Values

Search option available in Field List allows you to search a specific value that needs to be filtered from the list of values inside the filter pop-up window.

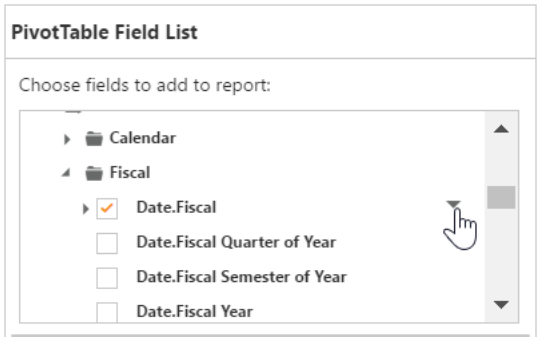


Copyright © 2001 - 2020 Syncfusion Inc.



Filtering

Values can be filtered by checking/unchecking the check box besides them, inside the filter pop-up window. At least one value needed to be checked state while filtering otherwise “Ok” button will be disabled.





Write-back

Information: This feature is applicable only for OLAP datasource only at Server Mode.

Copyright © 2001 - 2020 Syncfusion Inc.

We can now edit the values in PivotGrid and update a write enabled Cube at the back-end (SSAS) dynamically at runtime.

Note: Write-back is only supported for measures that use the **SUM** aggregation.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/OLAPService"
)).EnableCellEditing(true)
```

C#

```
public Dictionary < string, object > WriteBack(string action, string
value, string rowUniqueName, string columnUniqueName, string
currentReport) {
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(Syncfusion.JavaScript.Olap.Utils.Deserialize
OlapReport(currentReport));
return htmlHelper.GetJsonData(action, DataManager, value, rowUniqueName,
columnUniqueName);
}
```

	January	February	March	April	May
	Sales Amount Quota				
2005	1407017.22969963	1274053.85346841	1410559.62348303	1365055.33601641	1410557.18055029
2006	1410559.62348321	1000.00000004808	2000	3000	4000
2007	1410559.62348154	1274494.41762611	5000	7000	8000
2008	1410557.18007034	1319553.49148253	6000	3000	5600.5678
2009	1410557.18007034	1274051.64694865	1410557.18055029	1365055.33601641	1410557.18055029
2010	1410557.18007034	1274051.64694865	1410557.18055029	1365055.33601641	1410557.18055029
Total	**8459808.0168754**	**6417205.0564744**	**8472265.92945609**	**8190478.79206778**	**8463343.08326104**

Defer Update

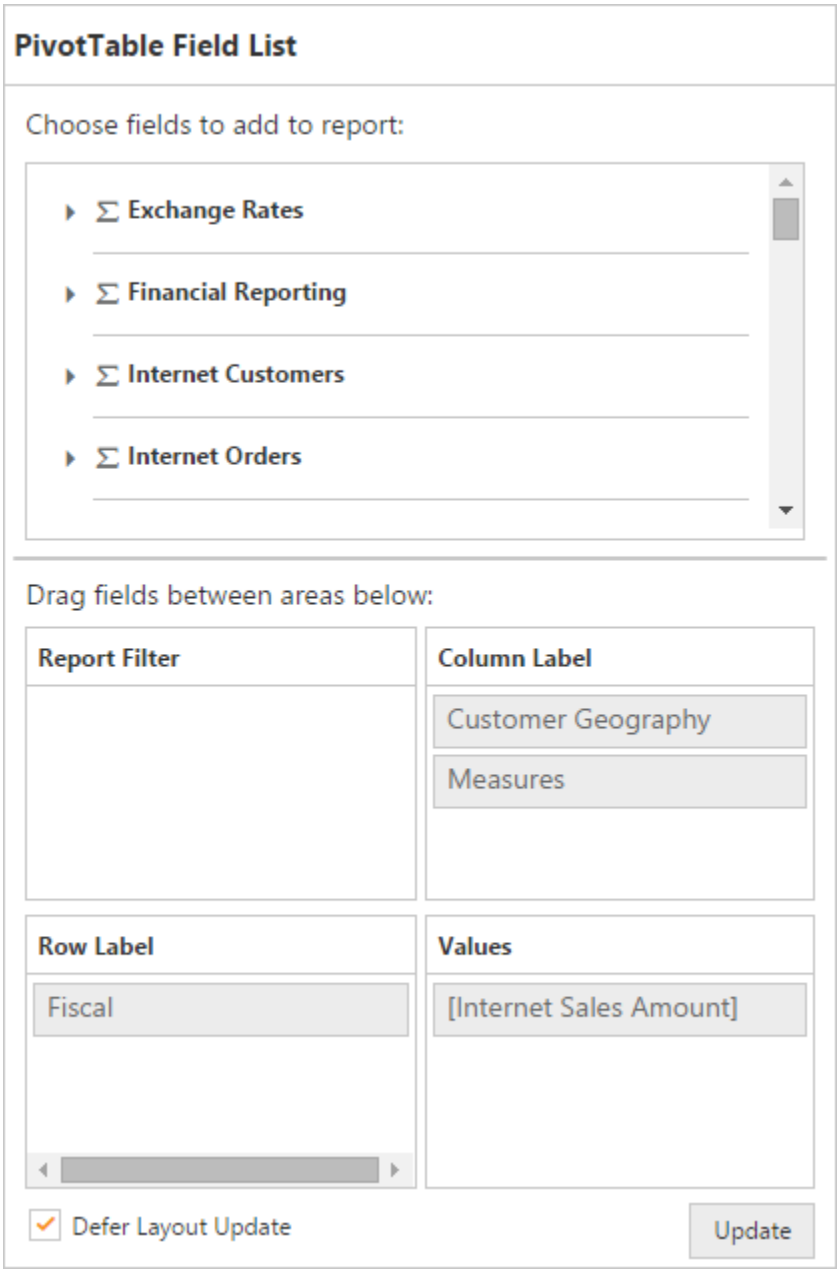
Information: This feature is applicable for OLAP datasource only at Server Mode.

Defer Update support allows you to refresh the control only on-demand and not during every UI interaction.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/OLAPService"
)).ClientSideEvents(events =>
events.AfterServiceInvoke("OnAfterServiceInvoke"))
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner").Layout(Pivo
tSchemaDesignerLayout.Excel)
<script type="text/javascript">
OnAfterServiceInvoke = function(evt) {
if (evt.action == "initialize") {
```

```
var PivotSchemaDesigner =
$("#PivotSchemaDesigner").data('ejPivotSchemaDesigner');
if (PivotSchemaDesigner.model.pivotControl == null) {
PivotSchemaDesigner.model.pivotControl = this;
PivotSchemaDesigner.model.enableWrapper = true;
PivotSchemaDesigner.model.layout = "excel";
PivotSchemaDesigner._load();
}
}
}
</script>
```



Exporting

The PivotGrid control can be exported to the following file formats.

- Excel
- Word
- PDF
- CSV

The PivotGrid control can be exported by invoking **"exportPivotGrid"** method, with an appropriate export option as parameter.

JSON Export

Information: By default JSON export mode will be applied for server and client mode.

ASPX-CS

```
//For PivotGrid in Client Mode.
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();})
.Data("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure
Works DW 2008 SE").Cube("Adventure Works"))
//For PivotGrid in Server Mode.
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("../OLAPServic
e"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("ExcelExport","fileName");
}
</script>
```

To achieve exporting, we need to add **"Syncfusion.EJ.Export"** dependency library into the application.

When PivotGrid is rendered, a method needs to be added in MVC controller file of the application and we need to import **"Syncfusion.EJ.Export"** namespace in the controller file.

C#

```
//...
using Syncfusion.EJ.Export;
public void ExcelExport()
{
PivotGridExcelExport pGrid = new PivotGridExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToExcel(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Excel Export

User can export the contents of PivotGrid to an Excel document for future archival, references and analysis purposes.

To achieve Excel export, the controller method name **"ExcelExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("ExcelExport","fileName");
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void ExcelExport()
{
PivotGridExcelExport pGrid = new PivotGridExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToExcel(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Word Export

User can export the contents of PivotGrid to a Word document for future archival, references and analysis purposes.

To achieve Word export, controller method name **"WordExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("WordExport","fileName");
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using Syncfusion.EJ.Export;
public void WordExport()
{
PivotGridWordExport pGrid = new PivotGridWordExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToWord(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

PDF Export

User can export contents of the PivotGrid to PDF document for future archival, references and analysis purposes.

To achieve Word export, controller method name **"PDFExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("PDFExport","fileName");
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void PDFExport()
{
PivotGridPDFExport pGrid = new PivotGridPDFExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToPDF(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

CSV Export

User can export the contents of PivotGrid to a CSV document for future archival, references and analysis purposes.

To achieve CSV export, controller method name **"CSVExport"** and file name is sent as the parameter.

JS

```
<script type="text/javascript">
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("CSVExport","fileName");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
```

Following method needs to be added in MVC controller file of the application.

C#

```
//...
using Syncfusion.EJ.Export;
public void CSVExport()
{
PivotGridCSVExport pGrid = new PivotGridCSVExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExportToCSV(fileName, args,
System.Web.HttpContext.Current.Response);
}
```

Customize the export document name

For customizing file name, we need to send file name as parameter to the **"exportPivotGrid"** method along with method name.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid("ExcelExport","fileName");
}
```

PivotEngine Export

Information: This feature is applicable only at server mode operation.

In order to perform exporting with the use of PivotEngine available in server-side, the 'exportMode' property obtained in the "BeforeExport" event is set to the value "ej.PivotGrid.ExportMode.PivotEngine" as shown below.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("../OLAPServic
e")).ClientSideEvents(clientSideEvents =>
clientSideEvents.BeforeExport("Export"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);;
}
function Export(args) {
args.exportMode = ej.PivotGrid.ExportMode.PivotEngine;
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

A service method needs to be added in WCF/WebAPI for server side operations.

For WebAPI controller, the below method needs to be added.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

For WCF service, the below method needs to be added.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
public void Export(System.IO.Stream stream)
{
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

Excel Export

User can export the contents of PivotGrid to an Excel document for future archival, references and analysis purposes.

To achieve Excel export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

For Excel export, **“ej.PivotGrid.ExportOptions.Excel”** enumeration value is sent as the parameter.

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as Excel in the exportPivotGrid method for
ServerMode
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
```

Word Export

User can export the contents of PivotGrid to a Word document for future archival, references and analysis purposes.

To achieve Word export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.DocIO.Base

For Word export, **“ej.PivotGrid.ExportOptions.Word”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as Word in the exportPivotGrid method
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Word);
}
```

PDF Export

User can export the contents of PivotGrid to a PDF document for future archival, references and analysis purposes.

To achieve PDF export, we need to add the following dependency libraries into the application.

- Syncfusion.Compression.Base
- Syncfusion.Pdf.Base

For PDF export, **“ej.PivotGrid.ExportOptions.PDF”** enumeration value is sent as the parameter.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as PDF in the exportPivotGrid method
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.PDF);
}
```

CSV Export

User can export the contents of PivotGrid to a CSV document for future archival, references and analysis purposes.

For CSV export, **“ej.PivotGrid.ExportOptions.CSV”** enumeration value is sent as the parameter.

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
function exportBtnClick(args)
{
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//Setting export option as CSV in the exportPivotGrid method
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.CSV);
}
```

File format selection

Information: This option is applicable only for PivotGrid when exporting to Excel document.

You can set the option for exporting the control to Excel document either in *.xls* or *.xlsx* format, using `fileFormat` property inside the `BeforeExport` event.

Note: By default excel document will be exported to ".xls" format using PivotEngine export.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("../Relational
Service")).ClientSideEvents(clientSideEvents =>
clientSideEvents.BeforeExport("Export"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
function Export(args) {
args.exportMode = ej.PivotGrid.ExportMode.PivotEngine;
args.fileFormat = ".xlsx"; //you can set the excel sheet format here
}
</script>
```

Customize the export document name

For customizing name in WebAPI controller, below code snippet is used.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export() {
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = " File name is customized here ";
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

For customizing name in WCF Service, below code snippet is used.

C#

```
//...
using Syncfusion.Compression.Base;
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
public void Export(System.IO.Stream stream) {
System.IO.StreamReader sReader = new System.IO.StreamReader(stream);
string args =
System.Web.HttpContext.Current.Server.UrlDecode(sReader.ReadToEnd()).Remo
ve(0, 5);;
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = " File name is customized here ";
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
```

Exporting Customization

You can add title and description to the exporting document by using the title and description properties respectively obtained in the BeforeExport event. Similarly, you can enable or disable styling on the exported document by using the exportWithStyle property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.BeforeExport("Exporting"))
@Html.EJ().Button("Button1").ClientSideEvents(clientSideEvents => {
clientSideEvents.Click("exportBtnClick"); }).Text("Export")
<script type="text/javascript">
function exportBtnClick(args) {
var pGridObj = $('#PivotGrid1').data("ejPivotGrid");
//JSON Export
pGridObj.exportPivotGrid("ExcelExport","fileName");
//PivotEngine Export
pGridObj.exportPivotGrid(ej.PivotGrid.ExportOptions.Excel);
}
function Exporting(args) {
args.title = "PivotGrid";
args.description = "Displays both OLAP and Relational datasource in
tabular format";
args.exportWithStyle = true;   // by default it sets as true. It improves
performance on exporting huge data when it sets as false.
}
</script>
```

You can also edit the exporting document with the use of a server side event for required exporting option.

C#

```
//...
using Syncfusion.EJ.Export;
using Syncfusion.Compression.Base;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
using Syncfusion.XlsIO;
using Syncfusion.DocIO.Base;
using Syncfusion.Pdf.Base;
//Following methods needs to be added in MVC controller file of the
application for JSON export
public void ExcelExport()
{
PivotGridExcelExport pGrid = new PivotGridExcelExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.ExcelExport += pGrid_ExcelExport;
pGrid.ExportToExcel(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
public void PDFExport()
{
PivotGridPDFExport pGrid = new PivotGridPDFExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.AddPDFHeaderFooter += pGrid_AddPDFHeaderFooter;
pGrid.PDFExport += pGrid_PDFExport;
pGrid.ExportToPDF(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_PDFExport(object sender, Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can customize exporting document here.
}
void pGrid_AddPDFHeaderFooter(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can add header/footer information to the PDF document.
}
public void CSVExport()
{
PivotGridCSVExport pGrid = new PivotGridCSVExport();
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.CSVExport += pGrid_CSVExport;
pGrid.ExportToCSV(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_CSVExport(object sender, string csvString)
{
//You can customize exporting document here.
}
public void WordExport()
{
PivotGridWordExport pGrid = new PivotGridWordExport();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string args =
System.Web.HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = "Sample";
pGrid.WordExport += pGrid_WordExport;
pGrid.ExportToWord(fileName, args,
System.Web.HttpContext.Current.Response);
}
void pGrid_WordExport(object sender, Syncfusion.DocIO.DLS.WordDocument
document)
{
//You can customize exporting document here.
}
//Following service methods needs to be added in WEBAPI controller file
for PivotEngine export
[System.Web.Http.ActionName("Export")]
[System.Web.Http.HttpPost]
public void Export()
{
string args = HttpContext.Current.Request.Form.GetValues(0)[0];
OlapDataManager DataManager = new OlapDataManager(connectionString);
string fileName = "Sample";
htmlHelper.ExcelExport += htmlHelper_ExcelExport;
htmlHelper.WordExport += htmlHelper_WordExport;
htmlHelper.AddPDFHeaderFooter += htmlHelper_AddPDFHeaderFooter;
htmlHelper.PDFExport += htmlHelper_PDFExport;
htmlHelper.CSVExport += htmlHelper_CSVExport;
htmlHelper.ExportPivotGrid(DataManager, args, fileName,
System.Web.HttpContext.Current.Response);
}
void htmlHelper_ExcelExport(object sender, Syncfusion.XlsIO.IWorkbook
workBook)
{
//You can customize exporting document here.
}
void htmlHelper_WordExport(object sender,
Syncfusion.DocIO.DLS.WordDocument document)
{
//You can customize exporting document here.
}
void htmlHelper_AddPDFHeaderFooter(object sender,
Syncfusion.Pdf.PdfDocument pdfDoc)
{
//You can add header/footer information to the PDF document.
}
void htmlHelper_PDFExport(object sender, Syncfusion.Pdf.PdfDocument
pdfDoc)
{
//You can customize exporting document here.
}
void htmlHelper_CSVExport(object sender, string csvString)
{
//You can customize exporting document here.
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

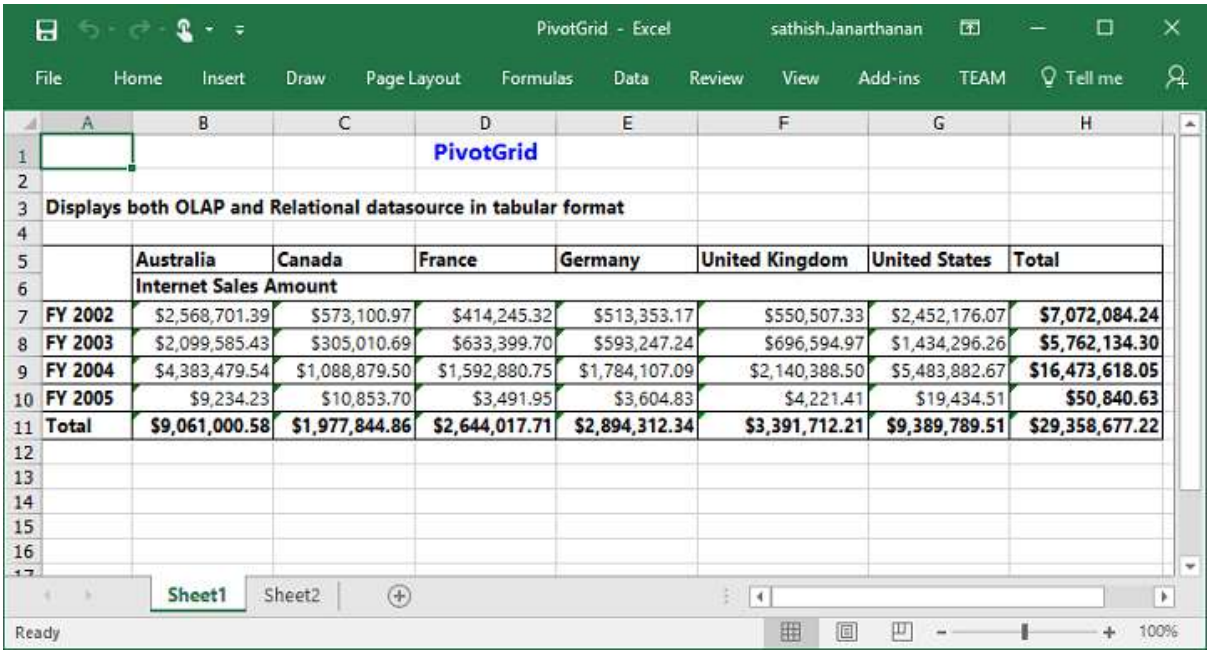
Exporting complete data on Paging

When paging is enabled, you can export the complete data by enabling the EnableCompleteDataExport property. It is supported in both types of JSON and PivotEngine export and it is applicable for all kinds of exporting formats available in PivotGrid.

ASPX-CS

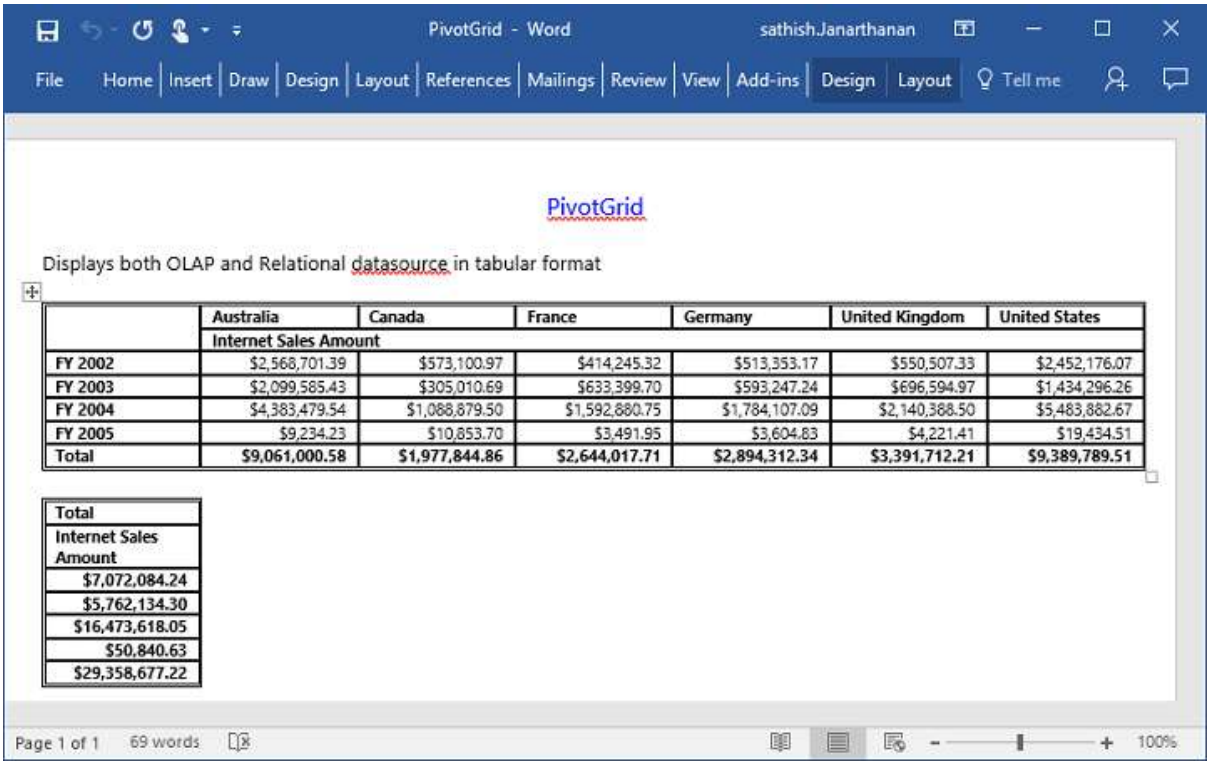
```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableCompleteDataExport(true)
```

The below screenshot shows the PivotGrid control exported to Excel document.



	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

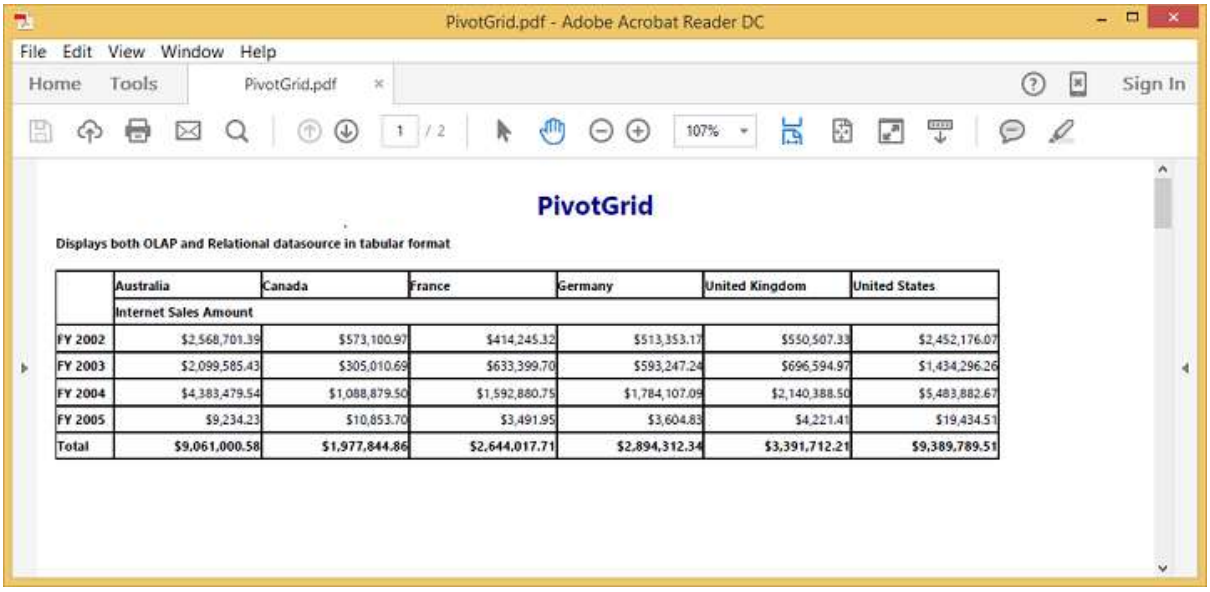
The below screenshot shows the PivotGrid control exported to Word document.



	Australia	Canada	France	Germany	United Kingdom	United States
	Internet Sales Amount					
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51

Total
Internet Sales Amount
$7,072,084.24
$5,762,134.30
$16,473,618.05
$50,840.63
$29,358,677.22

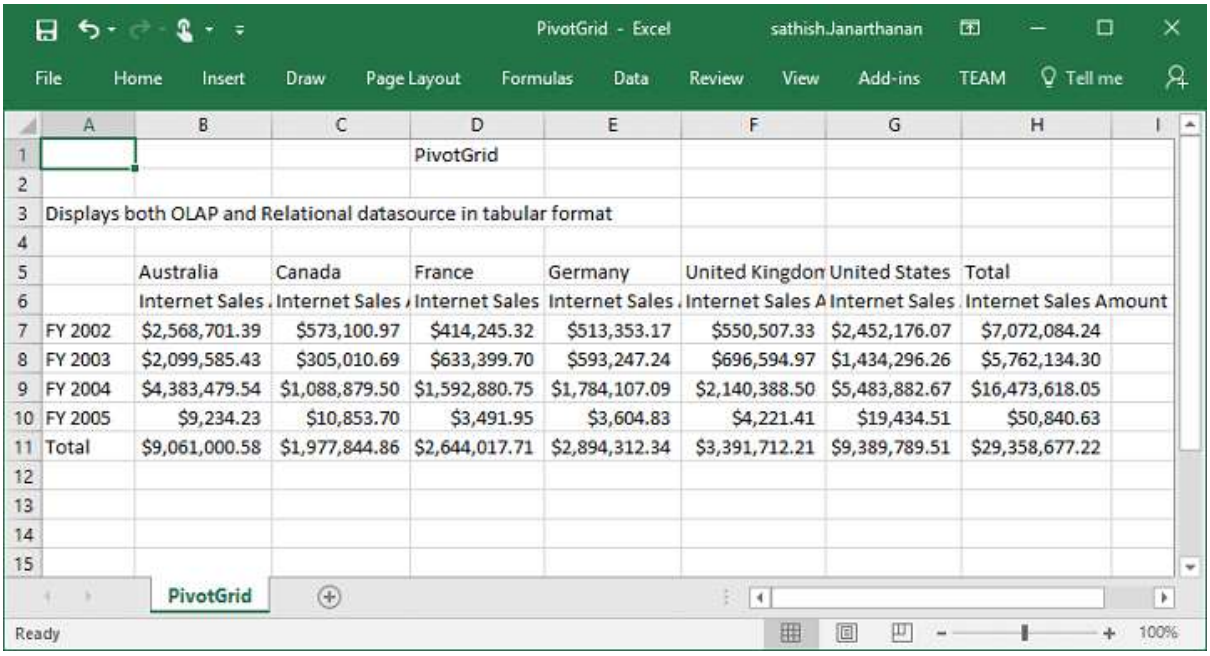
The below screenshot shows the PivotGrid control exported to PDF document.



	Australia	Canada	France	Germany	United Kingdom	United States
	Internet Sales Amount					
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51

The below screenshot shows the PivotGrid control exported to CSV document.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: Grand total can be hidden while exporting the document.

Common

PivotGrid: Elements

Hyperlink

The PivotGrid control supports hyperlink option to link data for each individual cell. Hyperlink can be enabled separately for row header, column header, value and summary cells. Following are the respective properties:

- **EnableColumnHeaderHyperlink** - Enables hyperlink for column headers.
- **EnableRowHeaderHyperlink** - Enables hyperlink for row headers.
- **EnableSummaryCellHyperlink** - Enables hyperlink for summary cell.
- **EnableValueCellHyperlink** - Enables hyperlink for value cell.

Also hyperlink option provides separate events for row header, column header, value and summary cells as mentioned below.

- **ColumnHeaderHyperlinkClick** - Returns column header information through event on hyperlink click.
- **RowHeaderHyperlinkClick** - Returns row header information through event on hyperlink click.
- **SummaryCellHyperlinkClick** - Returns summary cell information through event on hyperlink click.
- **ValueCellHyperlinkClick** - Returns value cell information through event on hyperlink click.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(events =>
events.ValueCellHyperlinkClick("CellClickEvent").RowHeaderHyperlinkClick(
"CellClickEvent").ColumnHeaderHyperlinkClick("CellClickEvent").SummaryCel
lHyperlinkClick("CellClickEvent")).HyperlinkSettings(hypLink =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
hypLink.EnableColumnHeaderHyperlink(true).EnableRowHeaderHyperlink(true).
EnableValueCellHyperlink(true).EnableSummaryCellHyperlink(true))
<script type="text/javascript">
CellClickEvent = function(evt) {
alert("Cell Click event is fired");
}
</script>
```



	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,582,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

Selection

You can select a particular range of value cells from PivotGrid and manipulate/display them. Cell selection is applicable only for value cells and you can enable this functionality by setting EnableCellSelection property to true.

The **"CellSelection"** event would be triggered as soon as the selection process is over, that is, when the mouse left click is released. The event argument contains a collection of JSON records and header values, which contains information about the selected cells.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableCellSelection(true).Clie
ntSideEvents(events => events.CellSelection("valueCellClick"))
<script type="text/javascript">
valueCellClick = function(evt) {
// The event lets you to perform required operation with the selected set
of cells. The details of the selected range can be obtained in the
parameter of the event.
cellvalue = evt.JSONRecords;
rowheaders = evt.rowHeader;
colheaders = evt.columnHeader;
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

Cell Context

Cell context allows user to perform any custom operation on cell right-click. For example, you can create and display context menu on cell right-click.

Cell context is enabled by setting the EnableCellContext property to true. The **"cellContext"** event would be raised as soon as right-click is done providing cell information through event argument.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableCellContext(true).Client
SideEvents(events=>events.CellContext("cell_RightClick"))
<script type="text/javascript">
cell_RightClick = function(evt) {
//Write your Cell Context code here
}
</script>
```

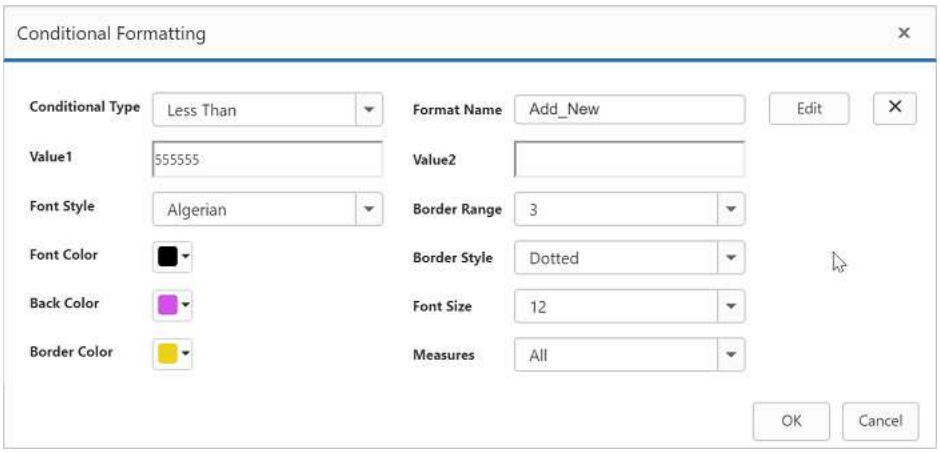
Conditional Formatting

Conditional formatting allows user to highlight particular cells with certain color, font-style, font-family etc. based on the condition they have met. It is enabled by setting EnableConditionalFormatting property to true and the formatting dialog is launched when **"createConditionalDialog"** method is invoked.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableConditionalFormatting(tr
ue)
@Html.EJ().Button("Button1").Size(ButtonSize.Normal).Text("Conditional").
ShowRoundedCorner(true).ClientSideEvents(events =>
events.Click("btnClick"))
<script type="text/javascript">
function btnClick(e) {
var pivotGridObj = $('#PivotGrid1').data("ejPivotGrid");
if (pivotGridObj.model.enableConditionalFormatting) {
pivotGridObj.createConditionalDialog();
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.93	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

Export

We can export the PivotGrid with highlighted particular cells along with its formatting styles.

LIMITATIONS FOR WORD:

The following border styles are not supported

- Solid
- Groove
- Ridge

LIMITATIONS FOR PDF:

Border styles are not applicable.

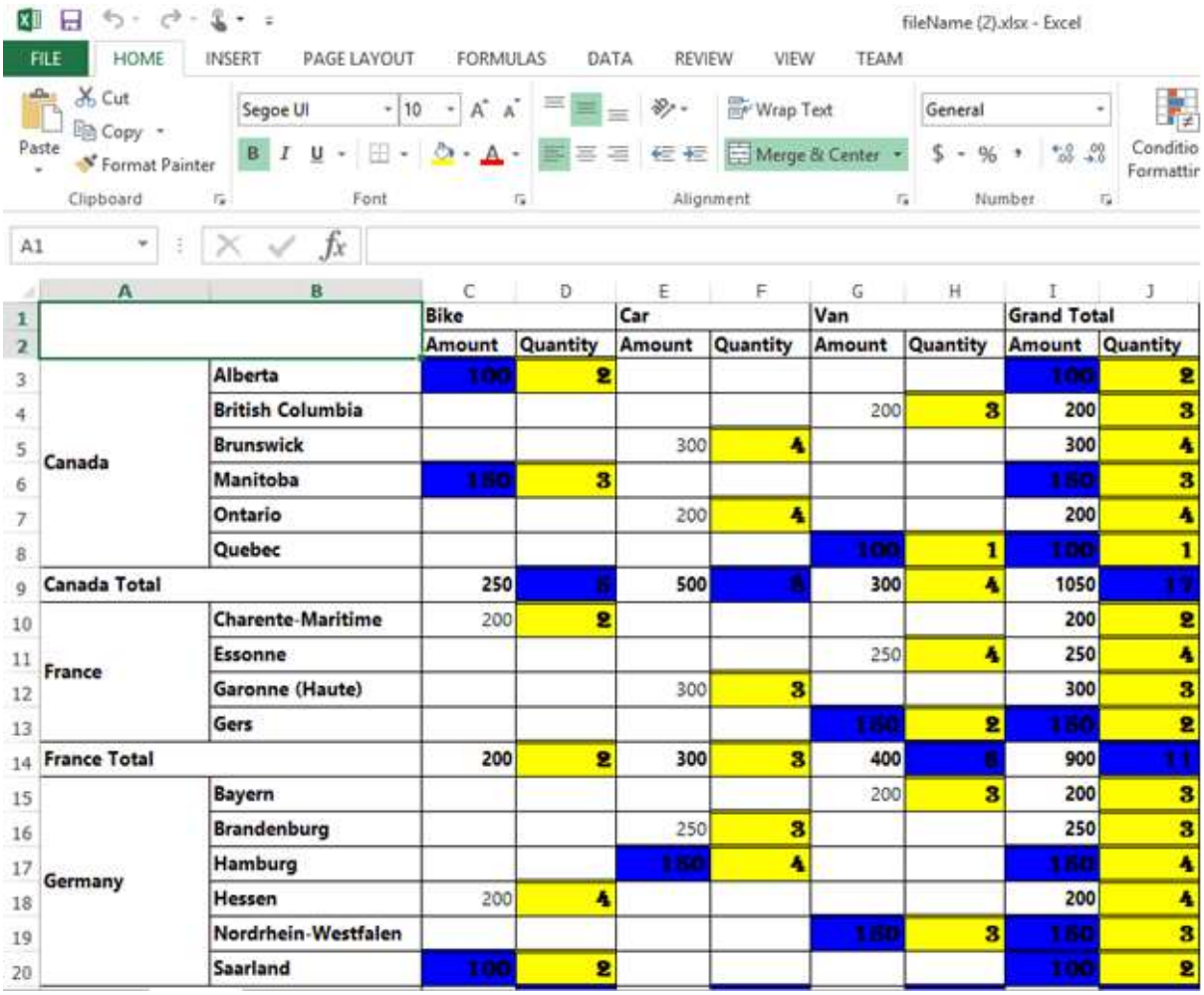
LIMITATIONS FOR EXCEL:

The following border styles are alone supported

- Dashed
- Dotted
- Double

Also border size is not supported.

Copyright © 2001 - 2020 Syncfusion Inc.



		Bike		Car		Van		Grand Total	
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
Canada	Alberta	100	2					100	2
	British Columbia					200	3	200	3
	Brunswick			300	4			300	4
	Manitoba	150	3					150	3
	Ontario			200	4			200	4
	Quebec					100	1	100	1
Canada Total		250	5	500	8	300	4	1050	17
France	Charente-Maritime	200	2					200	2
	Essonne					250	4	250	4
	Garonne (Haute)			300	3			300	3
	Gers					150	2	150	2
France Total		200	2	300	3	400	6	900	11
Germany	Bayern					200	3	200	3
	Brandenburg			250	3			250	3
	Hamburg			150	4			150	4
	Hessen	200	4					200	4
	Nordrhein-Westfalen					150	3	150	3
	Saarland	100	2					100	2

Number format

Information: This feature is applicable only for all modes.

Allows us to specify the required number format that PivotGrid should use in its values by setting the Format option. Following number formats that are supported:

- number
- decimal
- currency
- percentage
- date
- time
- scientific
- accounting
- fraction
- string

Relational

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
rows.FieldName("State").FieldCaption("State").Add(); }).Columns(columns
=> { columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add();
values.FieldName("Quantity").Format("decimal").Add(); }))
```

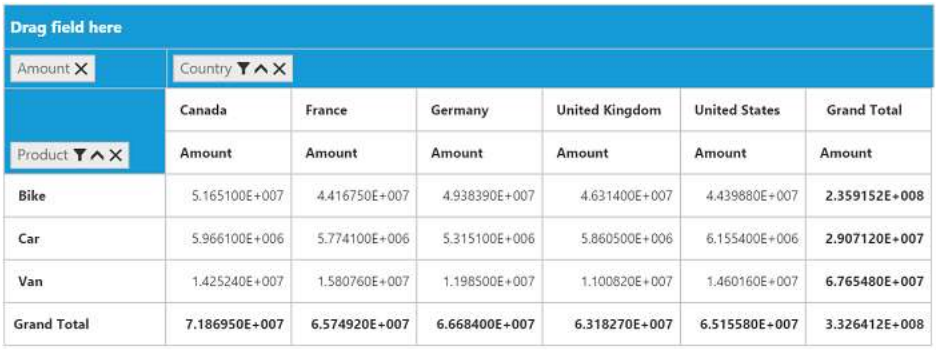
	Bike		Car		Van		Grand Total	
	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
▸ Canada	$250.00	5.00	$500.00	8.00	$300.00	4.00	$1,050.00	17.00
▸ France	$200.00	2.00	$300.00	3.00	$400.00	6.00	$900.00	11.00
▸ Germany	$300.00	6.00	$400.00	7.00	$350.00	6.00	$1,050.00	19.00
▸ United Kingdom	$150.00	5.00					$150.00	5.00
▸ United States	$300.00	6.00	$650.00	10.00	$500.00	8.00	$1,450.00	24.00
Grand Total	$1,200.00	24.00	$1,850.00	28.00	$1,550.00	24.00	$4,600.00	76.00

Server Mode

You can set Number Format through the Format property. You should specify the format to the property as per the MS standard notation.

```
private PivotReport BindDefaultData()

{

PivotReport pivotSetting = new PivotReport();

pivotSetting.PivotRows.Add(new PivotItem { FieldMappingName = "Product", FieldHeader = "Product", TotalHeader = "Total" });

pivotSetting.PivotColumns.Add(new PivotItem { FieldMappingName = "Country", FieldHeader = "Country", TotalHeader = "Total" });

//specify the format here

pivotSetting.PivotCalculations.Add(new PivotComputationInfo { CalculationName = "Amount", Description = "Amount", FieldHeader = "Amount", FieldName = "Amount", Format = "E", SummaryType = Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum });

return pivotSetting;

}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Drag field here

Amount	Country					
	Canada	France	Germany	United Kingdom	United States	Grand Total
Product	Amount	Amount	Amount	Amount	Amount	Amount
Bike	5.165100E+007	4.416750E+007	4.938390E+007	4.631400E+007	4.439880E+007	**2.359152E+008**
Car	5.966100E+006	5.774100E+006	5.315100E+006	5.860500E+006	6.155400E+006	**2.907120E+007**
Van	1.425240E+007	1.580760E+007	1.198500E+007	1.100820E+007	1.460160E+007	**6.765480E+007**
Grand Total	**7.186950E+007**	**6.574920E+007**	**6.668400E+007**	**6.318270E+007**	**6.515580E+007**	**3.326412E+008**

OLAP

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableGroupingBar(true).DataSo
urce(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales
Amount]").Format("percent").Add();
}).Axis(AxisName.Column).Add();}).Data("https://bi.syncfusion.com/olap/ms
mdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works"))
```



	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	256,870,139 %	57,310,097 %	41,424,532 %	51,335,317 %	55,050,733 %	245,217,607 %	**707,208,424 %**
FY 2003	209,958,543 %	30,501,069 %	63,339,970 %	59,324,724 %	69,659,497 %	143,429,626 %	**576,213,430 %**
FY 2004	438,347,954 %	108,887,950 %	159,288,075 %	178,410,709 %	214,038,850 %	548,388,267 %	**1,647,361,805 %**
FY 2005	923,423 %	1,085,370 %	349,195 %	360,483 %	422,141 %	1,943,451 %	**5,084,063 %**
Total	**906,100,058 %**	**197,784,486 %**	**264,401,771 %**	**289,431,234 %**	**339,171,221 %**	**938,978,951 %**	**2,935,867,722 %**

Server Mode

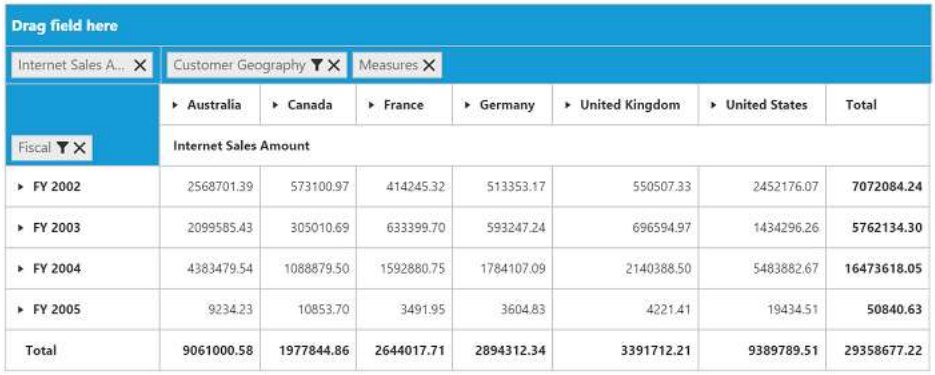
OLAP server mode supports the following Number Formats in addition to the above mentioned formats.

- General
- RoundTrip
- FixedPoint
- HexaDecimal

Note: You can set the number format through the Format property.

```
private OlapReport CreateOlapReport()
{
OlapReport olapReport = new OlapReport();
olapReport.CurrentCubeName = "Adventure Works";
```

```
MeasureElements measureElement = new MeasureElements();
measureElement.Elements.Add(new MeasureElement { UniqueName = "[Measures].[Internet Sales Amount]", Format = NumberFormat.FixedPoint }); //Specify the format here
DimensionElement dimensionElementRow = new DimensionElement();
dimensionElementRow.Name = "Date";
dimensionElementRow.AddLevel("Fiscal", "Fiscal Year");
DimensionElement dimensionElementColumn = new DimensionElement();
dimensionElementColumn.Name = "Customer";
dimensionElementColumn.AddLevel("Customer Geography", "Country");
olapReport.SeriesElements.Add(dimensionElementRow);
olapReport.CategoricalElements.Add(dimensionElementColumn);
olapReport.CategoricalElements.Add(measureElement);
return olapReport;
}
```



Save and Load Report

Allows you to save the current report of PivotGrid and render the control with the saved report later. We can save and load the report in two ways:

- Database
- Local Storage

Save Report to Database

By using current report name, storage option and url, we can save the current report of PivotGrid to database.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....)
@Html.EJ().Button("Button1").Size(ButtonSize.Normal).Text("Save").ShowRoundedCorner(true).ClientSideEvents(events => events.Click("saveReport"))
<script>
function saveReport(){
url = "../RelationalService",
```

```
name = "report",
storage = "db",
pGridObj.saveReport(name, storage, url);
}
</script>
```

A Service method needs to be added in WCF/WebAPI for storing current report of PivotGrid to database.

For WebAPI controller, the below method needs to be added.

C#

```
[System.Web.Http.ActionName("SaveReport")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> SaveReport(Dictionary<string, object>
jsonResult)
{
string mode = jsonResult["operationalMode"].ToString();
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString = "Enter
appropriate connection string to connect with database" };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", jsonResult["reportName"].ToString());
if (mode == "clientMode")
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(jsonResult["clientReports"].ToString()).ToArray())
;
else if (mode == "serverMode")
cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(jsonResult["clientReports"].ToString()).T
oArray());
cmd1.ExecuteNonQuery();
con.Close();
return null;
}
```

For WCF service, the below method needs to be added.

C#

```
public Dictionary<string, object> SaveReport(string reportName, string
operationalMode, string olapReport, string clientReports)
{
string mode = operationalMode;
bool isDuplicate = true;
SqlCeConnection con = new SqlCeConnection() { ConnectionString = "Enter
appropriate connection string to connect with database" };
con.Open();
SqlCeCommand cmd1 = null;
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
isDuplicate = false;
cmd1 = new SqlCeCommand("update ReportsTable set Report=@Reports where
ReportName like @ReportName", con);
}
}
if (isDuplicate)
{
cmd1 = new SqlCeCommand("insert into ReportsTable
Values(@ReportName,@Reports)", con);
}
cmd1.Parameters.Add("@ReportName", reportName);
//cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(clientReports).ToArray());
if (mode == "serverMode")
cmd1.Parameters.Add("@Reports",
OLAPUTILS.Utils.GetReportStream(clientReports).ToArray());
else if (mode == "clientMode")
cmd1.Parameters.Add("@Reports",
Encoding.UTF8.GetBytes(clientReports).ToArray());
cmd1.ExecuteNonQuery();
con.Close();
return null;
}
```

Save Report to Local Storage

To save the current report of PivotGrid to local storage, we need to call the SaveReport method in PivotGrid.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).ClientSideEv
ents(events => events.SaveReport("saveToLocal"))
@Html.EJ().Button("Button1").Size(ButtonSize.Normal).Text("save To
Local").ShowRoundedCorner(true).ClientSideEvents(events =>
events.Click("saveLocal"))
<script>
function saveLocal(){
url = "",
name = "report",
storage = "local",
pGridObj.saveReport(name, storage, url);
}
function saveToLocal(args){
```

```
localStorage.setItem("report", JSON.stringify(args.report));
}
</script>
```

Load Report from Database

By using the stored report name, database name and url, we can load the saved report of PivotGrid from database.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....)
@Html.EJ().Button("Button1").Size(ButtonSize.Normal).Text("Load
DB").ShowRoundedCorner(true).ClientSideEvents(events =>
events.Click("loadDB"))
<script>
function loadDB(){
url = "../RelationalService",
name = "report",
storage = "db",
pGridObj.loadReport(name, storage, url);
}
</script>
```

Service methods need to be added in WCF/WebAPI for loading the stored report in database.

Relational

For WebAPI controller, the below methods need to be added.

C#

```
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
{
byte[] reportString = new byte[2 * 1024];
PivotReport report = new PivotReport();
var reports = "";
string mode = jsonResult["operationalMode"].ToString();
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && jsonResult.ContainsKey("clientReports"))
{
reports = jsonResult["clientReports"].ToString();
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
if (mode == "clientMode")
{
reportString = (row.ItemArray[1] as byte[]);
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
report = htmlHelper.DeserializedReports(reports);
htmlHelper.PivotReport = report;
dictionary = htmlHelper.GetJsonData("loadOperation",
ProductSales.GetSalesData(), "Load Report",
jsonResult["reportName"].ToString());
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString = "Enter
appropriate connection string to connect with database" };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

For WCF service, the below methods need to be added.

C#

```
public Dictionary<string, object> LoadReportFromDB(string reportName,
string operationalMode, string olapReport, string clientReports)
{
byte[] reportString = new byte[2 * 1024];
PivotReport report = new PivotReport();
var reports = "";
string mode = operationalMode;
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && !string.IsNullOrEmpty(clientReports))
{
reports = clientReports;
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
if (mode == "clientMode")
{
reportString = (row.ItemArray[1] as byte[]);
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
report = htmlHelper.DeserializedReports(reports);
htmlHelper.PivotReport = report;
dictionary = htmlHelper.GetJsonData("loadOperation",
ProductSales.GetSalesData(), "Load Report", reportName);
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString = "Enter
appropriate connection string to connect with database"  };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

OLAP

For WebAPI controller, the below methods need to be added.

C#

```
[System.Web.Http.ActionName("LoadReportFromDB")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> LoadReportFromDB(Dictionary<string,
object> jsonResult)
{
string mode = jsonResult["operationalMode"].ToString();
byte[] reportString = new byte[4 * 1024];
var reports = "";
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && jsonResult.ContainsKey("clientReports"))
{
reports = jsonResult["clientReports"].ToString();
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as
string).Equals(jsonResult["reportName"].ToString()))
{
if (mode == "clientMode")
```

```
{
reportString = row.ItemArray[1] as byte[];
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
dynamic customData =
serializer.Deserialize<dynamic>(jsonResult["customObject"].ToString());
var cultureIDInfo = new System.Globalization.CultureInfo(("en-US")).LCID;
if (customData is Dictionary<string, object> &&
customData.ContainsKey("Language"))
{
cultureIDInfo = new
System.Globalization.CultureInfo((customData["Language"])).LCID;
}
connectionString = connectionString.Replace("" + cultureIDInfoval + "",
"" + cultureIDInfo + "");
cultureIDInfoval = cultureIDInfo;
DataManager.Culture = new
System.Globalization.CultureInfo((cultureIDInfo));
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(report
s));
DataManager.OverrideDefaultFormatStrings = true;
dictionary = htmlHelper.GetJsonData(jsonResult["action"].ToString(),
DataManager, jsonResult["gridLayout"].ToString(),
Convert.ToBoolean(jsonResult["enablePivotFieldList"].ToString()));
}
return dictionary;
}
private DataTable GetDataTable()
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString = "Enter
appropriate connection string to connect with database"  };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

For WCF service, the below methods need to be added.

C#

```
public Dictionary<string, object> LoadReportFromDB(string action, string
layout, bool enablePivotFieldList, object customObject, string
```

```
reportName, string operationalMode, string olapReport, string
clientReports)
{
string mode = operationalMode;
var reports = "";
byte[] reportString = new byte[4 * 1024];
Dictionary<string, object> dictionary = new Dictionary<string, object>();
if (mode == "serverMode" && !string.IsNullOrEmpty(clientReports))
{
reports = clientReports;
}
else
{
foreach (DataRow row in GetDataTable().Rows)
{
if ((row.ItemArray[0] as string).Equals(reportName))
{
if (mode == "clientMode")
{
reportString = row.ItemArray[1] as byte[];
dictionary.Add("report", Encoding.UTF8.GetString(reportString));
break;
}
else if (mode == "serverMode")
{
reports = OLAPUTILS.Utils.CompressData(row.ItemArray[1] as byte[]);
break;
}
}
}
}
if (reports != "")
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
dynamic customData =
serializer.Deserialize<dynamic>(customObject.ToString());
var cultureIDInfo = new System.Globalization.CultureInfo(("en-US")).LCID;
if (customData is Dictionary<string, object> &&
customData.ContainsKey("Language"))
{
cultureIDInfo = new
System.Globalization.CultureInfo((customData["Language"])).LCID;
}
connectionString = connectionString.Replace("" + cultureIDInfoval + "",
"" + cultureIDInfo + "");
cultureIDInfoval = cultureIDInfo;
DataManager.Culture = new
System.Globalization.CultureInfo((cultureIDInfo));
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(report
s));
DataManager.OverrideDefaultFormatStrings = true;
dictionary = htmlHelper.GetJsonData(action, DataManager, layout,
enablePivotFieldList);
}
return dictionary;
}
private DataTable GetDataTable()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
SqlCeConnection con = new SqlCeConnection() { ConnectionString = "Enter
appropriate connection string to connect with database" };
con.Open();
DataSet dSet = new DataSet();
new SqlCeDataAdapter("Select * from ReportsTable", con).Fill(dSet);
con.Close();
return dSet.Tables[0];
}
```

Load Report from Local Storage

To load the stored report of PivotGrid from local storage, we need to call the LoadReport method in PivotGrid.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).ClientSideEv
ents(events => events.LoadReport("loadFromLocal"))
@Html.EJ().Button("Button1").Size(ButtonSize.Normal).Text("Load").ShowRou
ndedCorner(true).ClientSideEvents(events => events.Click("loadReport"))
<script>
function loadReport(){
url = "",
name = "report",
storage = "local",
pGridObj.loadReport(name, storage, url);
}
function loadFromLocal(args){
args.targetControl.model.dataSource =
JSON.parse(localStorage.getItem("report"));
}
</script>
```

Save and Load XML Report

Information: This feature is applicable only for the relational datasource at Server Mode alone.

This support allows you to save the current report of PivotGrid as XML and render the control with the saved XML report later.

Note: The following dependency libraries should be added to your web application.

- Syncfusion.PivotAnalysis.Wpf
- Syncfusion.Grid.Wpf

Save XML Report

By using the 'saveXMLReport' method, you can save the current report of PivotGrid as XML file.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("~/api/Relatio
nalGrid"))
@Html.EJ().Button("Button1").Size(ButtonSize.Normal).Text("Save").ShowRou
ndedCorner(true).ClientSideEvents(events => events.Click("saveReport"))
<script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function saveReport() {
pGridObj = $('#PivotGrid1').data("ejPivotGrid");
url = "../api/RelationalGrid/XmlExport";
pGridObj.saveXMLReport("XML", url); // YOu can specify XML file name here
}
</script>
```

The service method should be added in WebAPI for exporting the current report as XML.

For **WebAPI controller**, the following method should be added.

Note: The following namespaces should be added on the top of the main class in WebAPI controller file.

- using System.IO;
- using System.Xml;
- using System.Xml.Serialization;
- using Syncfusion.Windows.Controls.PivotGrid;

C#

```
PivotGrid htmlHelper = new PivotGrid();
[System.Web.Http.ActionName("XmlExport")]
[System.Web.Http.HttpPost]
public void XmlExport()
{
string clientReport = HttpContext.Current.Request.Form.GetValues(0)[0];
string fileName = HttpContext.Current.Request.Form.GetValues(1)[0];
htmlHelper.ExportToXml(clientReport, fileName,
HttpContext.Current.Response);
}
```

Load XML Report

You can load the XML report in PivotGrid by using the following methods.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("~/api/Relatio
nalGrid"))
```

The service methods should be added in WebAPI for loading the stored XML report.

For **WebAPI controller**, the following methods should be added.

C#

```
PivotGrid htmlHelper = new PivotGrid();
JavaScriptSerializer serializer = new JavaScriptSerializer();
Dictionary<string, bool> clientSideAPI = new Dictionary<string, bool>();
Syncfusion.Windows.Controls.PivotGrid.PivotGridSerializer xmlSerializer =
new Syncfusion.Windows.Controls.PivotGrid.PivotGridSerializer();
string filePath = HttpContext.Current.Server.MapPath(".").Split(new
string[] { "\\api" }, StringSplitOptions.None)[0] + "\\pivot.xml";
[System.Web.Http.ActionName("XmlImport")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> XmlImport()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
Dictionary<string, object> dictionary = new Dictionary<string, object>();
StreamReader stream = null;
stream = new StreamReader(@"C:\\Users\\Desktop\\file.xml"); //here you
can specify xml file path
string xmlString = stream.ReadToEnd();
XmlDocument doc = new XmlDocument();
doc.LoadXml(xmlString);
if (doc.DocumentElement.Name.ToString() == "PivotGridSerializer")
{
XmlSerializer GridSerializer = new
XmlSerializer(typeof(PivotGridSerializer), new Type[] {
typeof(DoubleTotalSummary), typeof(CountSummary),
typeof(IntTotalSummary), typeof(DoubleAverageSummary),
typeof(DoubleMaxSummary), typeof(DoubleMinSummary),
typeof(DoubleStDevSummary), typeof(DoubleVarianceSummary),
typeof(DecimalTotalSummary) });
PivotGridSerializer PivotGridControl = new PivotGridSerializer();
PivotReport pivotReport = new PivotReport();
using (TextReader reader = new StringReader(xmlString))
{
if (doc.DocumentElement.Name.ToString() == "PivotGridSerializer")
PivotGridControl =
(PivotGridSerializer)GridSerializer.Deserialize(reader);
htmlHelper.PivotReport = this.UpdateXMLReport(PivotGridControl,
pivotReport, doc.DocumentElement.Name.ToString());
}
}
else if (doc.DocumentElement.Name.ToString() == "PivotReport")
{
htmlHelper.ImportFromXml(xmlString);
}
var ser = serializer.Serialize(clientSideAPI);
dictionary = htmlHelper.GetJsonData("xmlImport",
ProductSales.GetSalesData(), ser, null);
return dictionary;
}
private PivotReport UpdateXMLReport(PivotGridSerializer PivotGridControl,
PivotReport pivotReport, string documentName)
{
PivotReport report = new PivotReport();
if (documentName == "PivotGridSerializer")
{
foreach (PivotItem item in PivotGridControl.PivotRows)
report.PivotRows.Add(item);
foreach (PivotItem item in PivotGridControl.PivotColumns)
report.PivotColumns.Add(item);
foreach (PivotComputationInfo item in PivotGridControl.PivotCalculations)
report.PivotCalculations.Add(item);
foreach (FilterExpression item in PivotGridControl.Filters)
{
item.DimensionName = (string.IsNullOrEmpty(item.DimensionName)) ?
item.DimensionHeader : item.DimensionName;
report.Filters.Add(item);
}
foreach (var item1 in report.PivotColumns)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var sortItem = PivotGridControl.GridSortCollection.Where(i =>
i.FieldMappingName == item1.FieldMappingName).FirstOrDefault();
if (sortItem != null && sortItem.IsDescSort)
{
item1.Comparer = new Syncfusion.JavaScript.ReverseOrderComparer();
}
}
foreach (var item1 in report.PivotRows)
{
var sortItem = PivotGridControl.GridSortCollection.Where(i =>
i.FieldMappingName == item1.FieldMappingName).FirstOrDefault();
if (sortItem != null && sortItem.IsDescSort)
{
item1.Comparer = new Syncfusion.JavaScript.ReverseOrderComparer();
}
}
clientSideAPI.Add("ShowGroupingBar", PivotGridControl.ShowGroupingBar);
clientSideAPI.Add("ShowGrandTotals", PivotGridControl.ShowGrandTotals);
clientSideAPI.Add("AllowResizeColumns",
PivotGridControl.AllowResizeColumns);
clientSideAPI.Add("ResizePivotGridToFit",
PivotGridControl.ResizePivotGridToFit);
clientSideAPI.Add("AllowSelection", PivotGridControl.AllowSelection);
clientSideAPI.Add("DeferLayoutUpdate",
PivotGridControl.DeferLayoutUpdate);
clientSideAPI.Add("FreezeHeaders", PivotGridControl.FreezeHeaders);
clientSideAPI.Add("ShowFieldList", PivotGridControl.ShowFieldList);
clientSideAPI.Add("ShowSubTotals", PivotGridControl.ShowSubTotals);
}
return report;
}
[System.Web.Http.ActionName("InitializeGrid")]
[System.Web.Http.HttpPost]
public Dictionary<string, object> InitializeGrid(Dictionary<string,
object> jsonResult)
{
StreamReader stream = null;
stream = new StreamReader(filePath); //here you can specify xml file path
string xmlString = stream.ReadToEnd();
XmlDocument doc = new XmlDocument();
doc.LoadXml(xmlString);
htmlHelper.ImportFromXml(xmlString);
var ser = serializer.Serialize(clientSideAPI);
dict = htmlHelper.GetJsonData("xmlImport", ProductSales.GetSalesData(),
ser, null);
return dict;
}
```

Grid Layout

Normal Layout

A layout in which summary cells are positioned at the bottom of each parent member and their child members appear next to them. Normal layout is the default layout in PivotGrid control. The enumeration property `Layout` needs to be set to **“PivotGridLayout.Normal”** in-order to view PivotGrid in normal layout.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Layout(PivotGridLayout.Normal)
.Url(Url.Content("/OLAPService"))
```

		Australia	Canada	France	Germany	United Kingdom	United States	Total
		Internet Sales Amount						
FY 2002		$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	H1 FY 2003	$894,630.70	$195,331.22	$281,268.39	$245,662.66	$332,670.05	$775,069.93	$2,724,632.94
	H2 FY 2003	$1,204,954.73	$109,679.47	$352,131.31	$347,584.58	$363,924.93	$659,226.34	$3,037,501.36
	Total	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004		$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005		$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total		$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

No Summaries Layout

Information: This feature is applicable only for OLAP datasource.

A layout in which summary cells are completely hidden and the child members appear next to their parent member. The enumeration property Layout needs to be set to **"PivotGridLayout.NoSummaries"** in-order to view PivotGrid without summaries.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Layout(PivotGridLayout.NoSumma
ries).Url(Url.Content("/OLAPService"))
```

		Australia	Canada	France	Germany	United Kingdom	United States
		Internet Sales Amount					
FY 2002		$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07
FY 2003	H1 FY 2003	$894,630.70	$195,331.22	$281,268.39	$245,662.66	$332,670.05	$775,069.93
	H2 FY 2003	$1,204,954.73	$109,679.47	$352,131.31	$347,584.58	$363,924.93	$659,226.34
FY 2004		$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67
FY 2005		$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51

Excel-like Layout

A layout in which summary cells are positioned besides each parent member and their child members appear next to them. The enumeration property Layout needs to be set to **"PivotGridLayout.ExcelLikeLayout"** in-order to view PivotGrid in excel-like layout.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Layout(PivotGridLayout.ExcelLi
keLayout).Url(Url.Content("/OLAPService"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	**$7,072,084.24**
FY 2003	**$2,099,585.43**	**$305,010.69**	**$633,399.70**	**$593,247.24**	**$696,594.97**	**$1,434,296.26**	$5,762,134.30
H1 FY 2003	$894,630.70	$195,331.22	$281,268.39	$245,662.66	$332,670.05	$775,069.93	**$2,724,632.94**
H2 FY 2003	$1,204,954.73	$109,679.47	$352,131.31	$347,584.58	$363,924.93	$659,226.34	**$3,037,501.36**
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	**$16,473,618.05**
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	**$50,840.63**
Grand Total	**$9,061,000.58**	**$1,977,844.86**	**$2,644,017.71**	**$2,894,312.34**	**$3,391,712.21**	**$9,389,789.51**	**$29,358,677.22**

Top Summary Layout

Information: This feature is applicable only for OLAP datasource only at Server Mode.

A layout in which summary cells are positioned at the top of each parent member and their child members appear next to them. The enumeration property Layout needs to be set to **"PivotGridLayout.NormalTopSummary"** in-order to view PivotGrid in top summaries layout.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Layout(PivotGridLayout.NormalTopSummary).Url(Url.Content("~/OLAPService"))
```

		Total	Australia	Canada	France	Germany	United Kingdom	United States
		Internet Sales Amount						
Total		$29,358,677.22	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51
FY 2002		$7,072,084.24	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07
FY 2003	Total	$5,762,134.30	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26
	H1 FY 2003	$2,724,632.94	$894,630.70	$195,331.22	$281,268.39	$245,662.66	$332,670.05	$775,069.93
	H2 FY 2003	$3,037,501.36	$1,204,954.73	$109,679.47	$352,131.31	$347,584.58	$363,924.93	$659,226.34
FY 2004		$16,473,618.05	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67
FY 2005		$50,840.63	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51

Grand Total Hiding

Grand Total Hiding can be classified into three categories.

- Row Grand Total Hiding
- Column Grand Total Hiding
- Both

Row Grand Total Hiding

You can hide the **Grand Total** in row alone by setting the property EnableRowGrandTotal to false.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableRowGrandTotal(false).Dat
aSource(.....)
```

		Canada		France		Germany		United Kingdom		United States		Grand Total	
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
▼ FY 2005	Bike	100	2	200	2	100	2			200	4	600	10
	Van									200	4	200	4
FY 2005 Total		100	2	200	2	100	2			400	8	800	14
▼ FY 2006	Bike									100	2	100	2
	Car	200	4									200	4
	Van	200	3	250	4	200	3					650	10
FY 2006 Total		400	7	250	4	200	3			100	2	950	16
▼ FY 2007	Car	300	4	300	3	250	3			250	4	1100	14
	Van	100	1			150	3			300	4	550	8
FY 2007 Total		400	5	300	3	400	6			550	8	1650	22
▼ FY 2008	Bike	150	3			200	4	150	5			500	12
	Car					150	4			400	6	550	10
	Van			150	2							150	2
FY 2008 Total		150	3	150	2	350	8	150	5	400	6	1200	24

Column Grand Total Hiding

You can hide the **Grand Total** in column alone by setting the property EnableColumnGrandTotal to false.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableColumnGrandTotal(false).
DataSource(.....)
```

		Canada		France		Germany		United Kingdom		United States	
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
▼ FY 2005	Bike	100	2	200	2	100	2			200	4
	Van									200	4
FY 2005 Total		100	2	200	2	100	2			400	8
▼ FY 2006	Bike									100	2
	Car	200	4								
	Van	200	3	250	4	200	3				
FY 2006 Total		400	7	250	4	200	3			100	2
▼ FY 2007	Car	300	4	300	3	250	3			250	4
	Van	100	1			150	3			300	4
FY 2007 Total		400	5	300	3	400	6			550	8
▼ FY 2008	Bike	150	3			200	4	150	5		
	Car					150	4			400	6
	Van			150	2						
FY 2008 Total		150	3	150	2	350	8	150	5	400	6
Grand Total		1050	17	900	11	1050	19	150	5	1450	24

Copyright © 2001 - 2020 Syncfusion Inc.

Both

You can hide the **Grand Total** in both row and column by setting the property EnableGrandTotal to false.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableGrandTotal(false).DataSo
urce(.....)
```

		Canada		France		Germany		United Kingdom		United States	
		Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity	Amount	Quantity
▼ FY 2005	Bike	100	2	200	2	100	2			200	4
	Van									200	4
FY 2005 Total		100	2	200	2	100	2			400	8
▼ FY 2006	Bike									100	2
	Car	200	4								
	Van	200	3	250	4	200	3				
FY 2006 Total		400	7	250	4	200	3			100	2
▼ FY 2007	Car	300	4	300	3	250	3			250	4
	Van	100	1			150	3			300	4
FY 2007 Total		400	5	300	3	400	6			550	8
▼ FY 2008	Bike	150	3			200	4	150	5		
	Car					150	4			400	6
	Van			150	2						
FY 2008 Total		150	3	150	2	350	8	150	5	400	6

Advanced Filtering and Sorting

It allows to filter and sort the field members in PivotGrid.

Client Mode

In client mode, you can enable Advanced Filtering and Sorting option in PivotGrid by setting the EnableAdvancedFilter property under DataSource to true.

HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.EnableAdvancedFilter(true))
```

Server Mode

In server mode, you can enable the Advanced Filtering and Sorting option in PivotGrid by setting the EnableAdvancedFilter property to true.

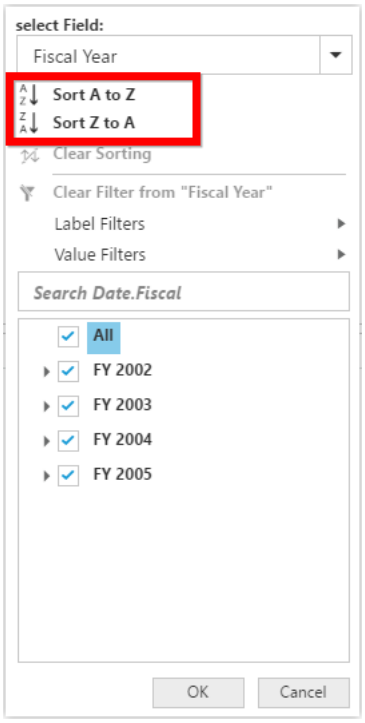
HTML

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableAdvancedFilter(true)
```

Sorting

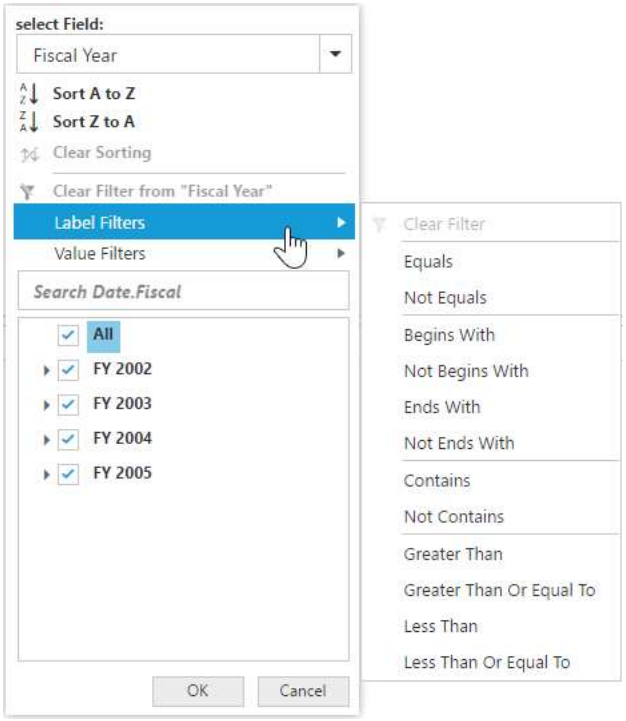
Sorting provides an option to sort the members of a field either in ascending or descending order.

Copyright © 2001 - 2020 Syncfusion Inc.



Label Filtering

Label filtering provides an option to filter the members of a field purely based on their caption.

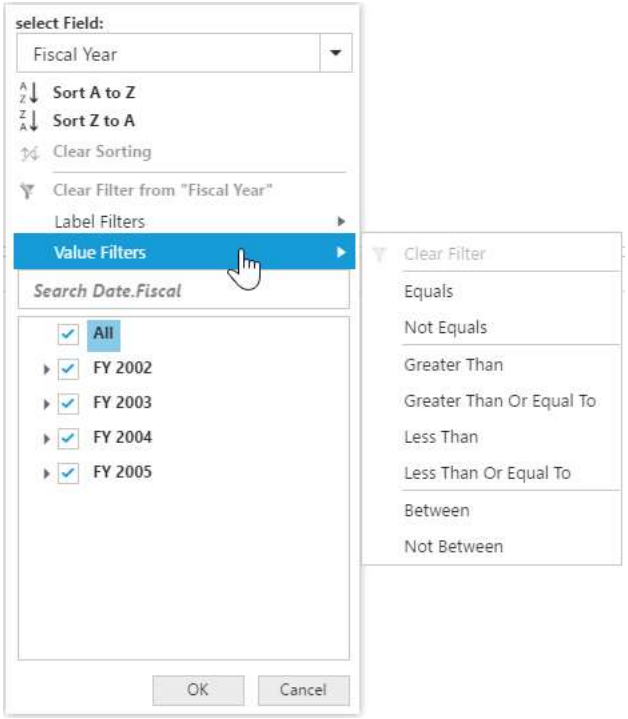


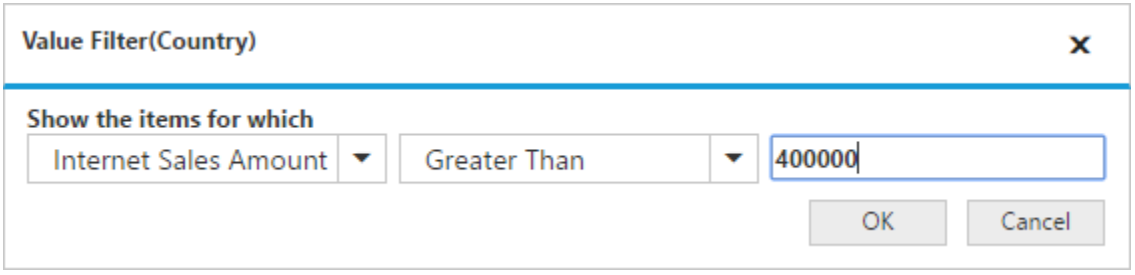
Copyright © 2001 - 2020 Syncfusion Inc.



Value Filtering

Value filtering provides an option to filter members based on the total values of the appropriate measure between the members of the level.





Paging

Pager

Paging helps to improve the rendering performance of the PivotGrid control by dividing large amount of data into sections and displaying one section at a time. You can enable Paging option in PivotGrid by setting the `EnablePaging` property to true. You can provide the page size and current page details for each axis in `PagerOptions` property.

Copyright © 2001 - 2020 Syncfusion Inc.

In-order to initialize a **Pager**, first you need to define a **div** tag with an appropriate **id** attribute which acts as a container for the widget. Then you need to initialize the widget using **ejPivotPager** method.

Inside the **ejPivotPager** method, the enumeration property mode needs to be set to **ej.PivotPager.Mode.Both** in-order to display both categorical pager and series pager. The other enumerations such as **ej.PivotPager.Mode.Categorical** and **ej.PivotPager.Mode.Series** will display only categorical pager and series pager respectively.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.PagerOptions(pagerOptions => {
pagerOptions.CategoricalPageSize(3).SeriesPageSize(3).CategoricalCurrentP
age(1).SeriesCurrentPage(1); })).EnablePaging(true)
@Html.EJ().Pivot().PivotPager("Pager1").Mode(PivotPagerMode.Both).TargetC
ontrolID("PivotGrid1")
```

Following are the navigation options available in Pager.

- Move First - Navigates to the first page.
- Move Last - Navigates to the last page.
- Move Previous - Navigates to the previous page from the current page.
- Move Next - Navigates to the next page from the current page.
- Numeric Box - Navigates to the desired page by entering an appropriate page number in numeric value.

	Aaron A. Allen	Aaron A. Hayes	Aaron A. Zhang	Aaron Alexander	Aaron B. Adams
	Internet Sales Amount				
▸ Australia					
▸ Canada	$3,399.99				
▸ France					
▸ Germany					

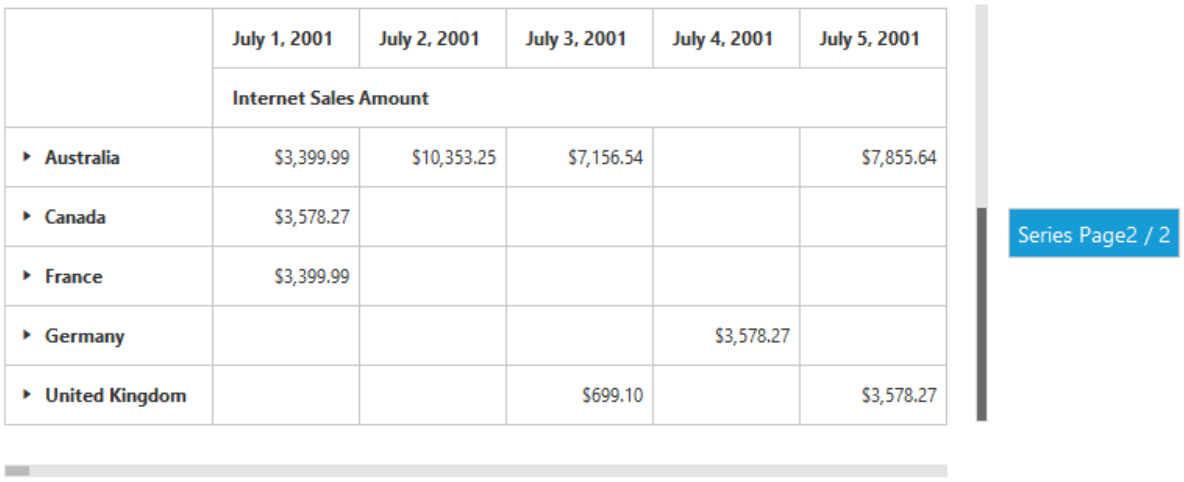


Virtual Scrolling

Virtual Scrolling is a technique that allows user to view the PivotGrid information page by page with the use of vertical and horizontal scrollbar. You can enable Virtual Scrolling option in PivotGrid by setting the EnableVirtualScrolling property to true. You can provide the page size and current page details for each axis in PagerOptions property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.PagerOptions(pagerOptions => {
pagerOptions.CategoricalPageSize(3).SeriesPageSize(3).CategoricalCurrentP
age(1).SeriesCurrentPage(1); })).EnableVirtualScrolling(true)
```



Page Settings

The properties associated to paging are:

- EnablePaging – This property is used to enable/disable paging in PivotClient control.
- PagerOptions.CategoricalPageSize - Specifies the number of categorical columns to be displayed within a page of the PivotClient control.
- PagerOptions.SeriesPageSize - Specifies the number of series rows to be displayed within a page of the PivotClient control.
- PagerOptions.CategoricalCurrentPage - Sets the current page of the categorical axis in PivotClient control.
- PagerOptions.SeriesCurrentPage - Sets the current page of the series axis in PivotClient control.

For client mode, page setting for categorical and series axes are mandatorily needed to be set in data source property by using the following properties.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.PagerOptions(pagerOptions => {
pagerOptions.CategoricalPageSize(3).SeriesPageSize(3).CategoricalCurrentP
age(1).SeriesCurrentPage(1); })).EnablePaging(true)
```

For server mode, the page settings for categorical and series axes are done only through report object, created inside WebAPI or WCF file.

For Relational data source

C#

```
PivotReport pivotReport = new PivotReport();
pivotReport.PagerOptions.SeriesPageSize = 4;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
pivotReport.PagerOptions.CategoricalPageSize = 5;
pivotReport.PagerOptions.SeriesCurrentPage = 1;
pivotReport.PagerOptions.CategoricalCurrentPage = 1;
```

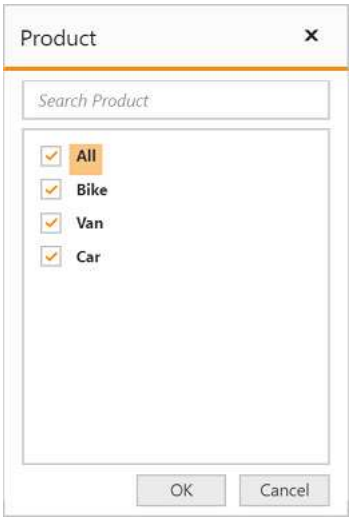
For OLAP data source

C#

```
OlapReport olapReport = new OlapReport();
olapReport.CurrentCubeName = "Adventure Works";
olapReport.EnablePaging = true;
olapReport.PagerOptions.SeriesPageSize = 4;
olapReport.PagerOptions.CategoricalPageSize = 5;
olapReport.PagerOptions.CategoricalCurrentPage = 1;
olapReport.PagerOptions.SeriesCurrentPage = 1;
```

Member Editor

Member editor is a dialog that displays the members of the current field in a tree view structure, which can be opened by clicking the filter icon available in the grouping bar. It helps to search, filter, and sort the field members available in the pivot grid control.



Member editor - Paging

Member editor paging helps to improve the rendering performance of the dialog by dividing large amount of data into sections and displaying them.

You can enable member editor paging and set member editor page size in PivotGrid control by setting the EnableMemberEditorPaging and MemberEditorPageSize properties.

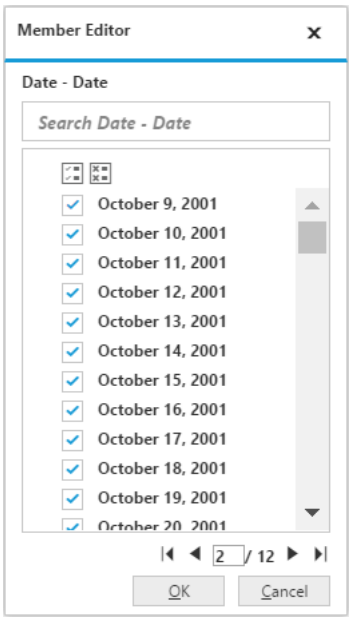
ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableMemberEditorPaging(true)
.MemberEditorPageSize(100)
```

Following are the navigation option available in Member Editor Pager.

- Move First - Navigates to the first page.

- Move Previous - Navigates to the previous page from the current page.
- Move Next - Navigates to the next page from the current page.
- Move Last - Navigates to the last page.
- Numeric Box - Navigates to the desired page by entering an appropriate page number in numeric value.



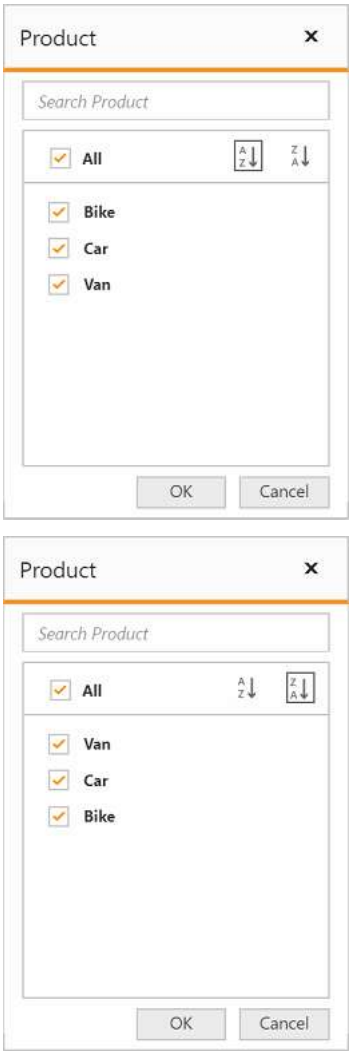
Member editor - Sorting

The sorting support in member editor helps you to sort the field members either in ascending or descending order.

You can enable the member editor sorting in the pivot grid control by setting the EnableMemberEditorSorting property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableMemberEditorSorting(true
)
```



Frozen Header

Allows you to freeze the header of the Grid so that it will be always visible when scrolling the content with a large number of rows or columns providing a precise view.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).FrozenHeader
Settings(frohead => frohead.EnableFrozenHeaders(true))
```

Copyright © 2001 - 2020 Syncfusion Inc.



We can also freeze the row/column headers individually by setting the below properties.

ASPX-CS

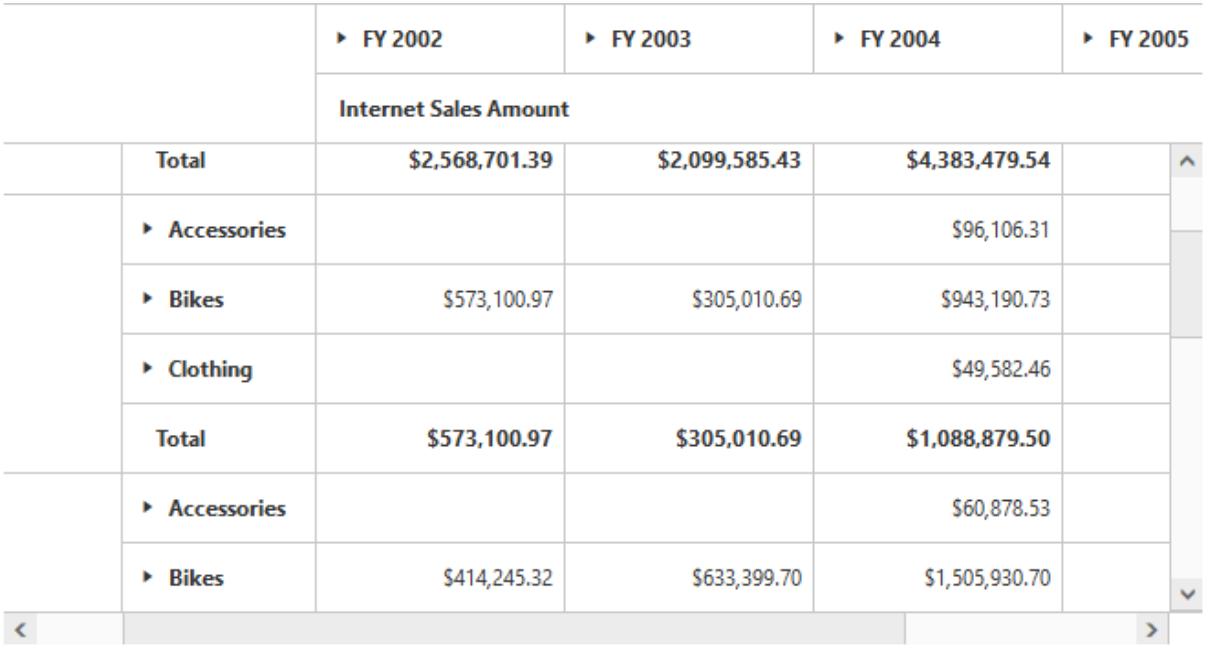
```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).FrozenHeader
Settings(frohead => frohead.EnableFrozenRowHeaders(true))  //To Freeze
the Row headers
```



ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).FrozenHeader
Settings(frohead => frohead.EnableFrozenColumnHeaders(true))  //To Freeze
the Column headers
```

Copyright © 2001 - 2020 Syncfusion Inc.

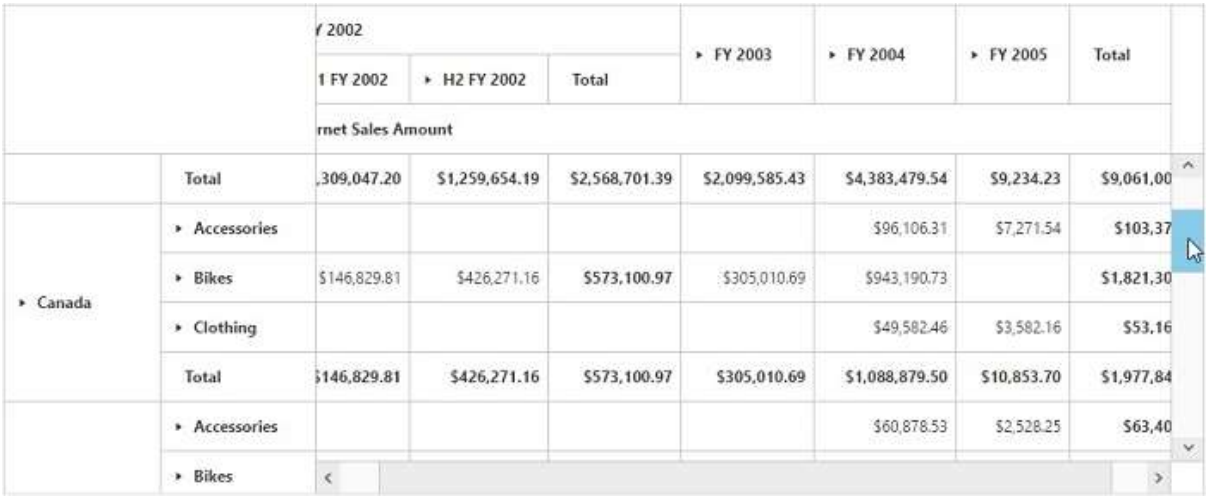


		FY 2002	FY 2003	FY 2004	FY 2005
		Internet Sales Amount			
	Total	$2,568,701.39	$2,099,585.43	$4,383,479.54	
	Accessories			$96,106.31	
	Bikes	$573,100.97	$305,010.69	$943,190.73	
	Clothing			$49,582.46	
	Total	$573,100.97	$305,010.69	$1,088,879.50	
	Accessories			$60,878.53	
	Bikes	$414,245.32	$633,399.70	$1,505,930.70	

We can also set the size of the scroller (horizontal and vertical) in PivotGrid by using below property.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).FrozenHeader
Settings(frohead => frohead.ScrollerSize(18))
```



		Y 2002			FY 2003	FY 2004	FY 2005	Total
		1 FY 2002	H2 FY 2002	Total				
		rnet Sales Amount						
	Total	,309,047.20	$1,259,654.19	$2,568,701.39	$2,099,585.43	$4,383,479.54	$9,234.23	$9,061,00
Canada	Accessories					$96,106.31	$7,271.54	$103,37
	Bikes	$146,829.81	$426,271.16	$573,100.97	$305,010.69	$943,190.73		$1,821,30
	Clothing					$49,582.46	$3,582.16	$53,16
	Total	$146,829.81	$426,271.16	$573,100.97	$305,010.69	$1,088,879.50	$10,853.70	$1,977,84
	Accessories					$60,878.53	$2,528.25	$63,40
	Bikes							

Drill Through

Drill-through retrieves the raw items that are used to create a specific cell. To enable drill-through support, set EnableDrillThrough property to true. Raw items are obtained through the [DrillThrough] event, using which user can bind them to an external widget for precise view.

OLAP

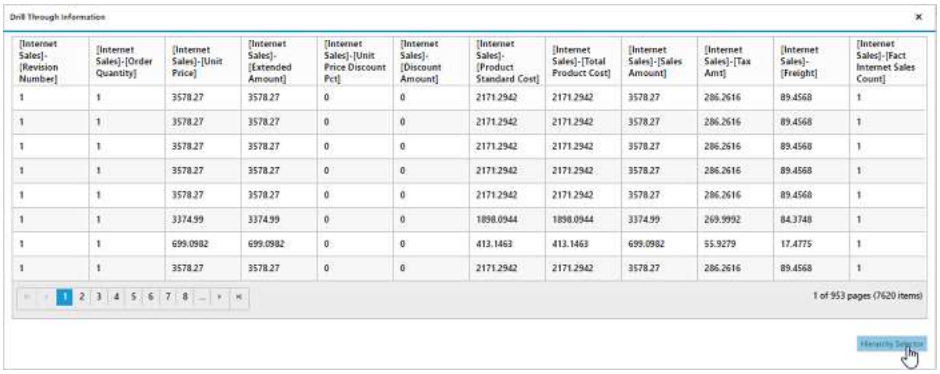
Note: Drill-through is supported in PivotGrid only when we configure and enable drill-through action at the Cube.

Copyright © 2001 - 2020 Syncfusion Inc.

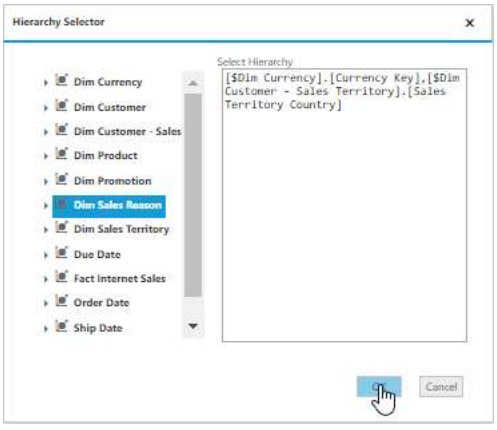
	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Sales Amount						
▸ 2005	9061000.58440184	1977844.86209998	2644017.71430033	2894312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
▸ 2006	9061000.58440184	1977844.86209998	2644017.71430033	2894312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
▸ 2007	9061000.58440184	1977844.86209998		4312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
▸ 2008	9061000.58440184	1977844.86209998		4312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
▸ 2009	9061000.58440184	1977844.86209998		4312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
▸ 2010	9061000.58440184	1977844.86209998	2644017.71430033	2894312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
▸ 2011	9061000.58440184	1977844.86209998	2644017.71430033	2894312.33820041	3391712.21090071	9389789.51080357	29358677.2207068
Total	9061000.58440184	1977844.86209998	2644017.71430033	2894312.33820041	3391712.21090071	9389789.51080357	29358677.2207068

Value: 2644017.71430033
Row: 2006
Column: France-Sales Amount

On clicking any value cell, the "Drill Through Information" dialog will be opened. It consists of a Grid with data which are associated with the measure values of clicked value cell. In this example, the measure behind the respective cell is “Sales Amount” and the values of the dimensions which are associated with this measure are alone displayed in the Grid.



On clicking the "Hierarchy Selector" button which is displayed below the Grid, the Hierarchy Selector dialog will be opened. It consists of the dimensions which are associated with the measure of the clicked value cell. In this example, the measure behind the respective cell is “Sales Amount” and the dimensions associated with this measure are alone displayed in the dialog.



By dragging and dropping the respective hierarchies and finally clicking “OK”, drill through MDX query will be framed and executed internally and provides back the raw items through "DrillThrough" event. In

Copyright © 2001 - 2020 Syncfusion Inc.

this example, we have bound the raw items obtained to our ejGrid widget. Please refer the code sample and screen-shot below.

Client Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).EnableDrillT
hrough(true).ClientSideEvents(events =>
events.DrillThrough("drilledData"))
<script type="text/javascript">
function drilledData(args) {
$(".e-dialog, .e-clientDialog, .e-tableDlg").remove();
gridData = JSON.parse(args.data);
var dialogContent = ej.buildTag("div#" + this._id + "_tableDlg.tableDlg",
$("<div id=\"Grid1\"></div>"))[0].outerHTML;
var dialogFooter = ej.buildTag("div", ej.buildTag("button#btnOK.e-
dialogBtnOK", "Hierarchy Selector")[0].outerHTML, { "float": "right",
"margin": "-5px 0 6px" })[0].outerHTML
ejDialog = ej.buildTag("div#clientDialog.e-clientDialog", dialogContent +
dialogFooter, { "opacity": "1" }).attr("title", "Drill Through
Information")[0].outerHTML;
$(ejDialog).appendTo("#" + this._id);
$("#btnOK").ejButton().css({ margin: "30px 0 20px 0" });
$("#Grid1").ejGrid({
dataSource: gridData,
allowPaging: true,
allowTextWrap: true,
pageSettings: { pageSize: 8 }
});
this.element.find(".e-clientDialog").ejDialog({ width: "70%", content:
"#" + this._id, enableResize: false, close:
ej.proxy(ej.Pivot.closePreventPanel, this) });
var pivotGrid = $("#" + this._id).data("ejPivotGrid");
$("#btnOK").click(function () {
ej.Pivot.openHierarchySelector(pivotGrid);
});
}
</script>
```

Server Mode

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/OLAPService"
)).EnableDrillThrough(true).ClientSideEvents(events =>
events.DrillThrough("drilledData"))
<script type="text/javascript">
function drilledData(args) {
$(".e-dialog, .e-clientDialog, .e-tableDlg").remove();
gridData = ej.isNullOrUndefined(args.data.d) ?
JSON.parse(args.data.DrillDataTable) : JSON.parse(args.data.d[1].Value);
var dialogContent = ej.buildTag("div#" + this._id + "_tableDlg.e-
tableDlg", $("<div id=\"Grid1\"></div>"))[0].outerHTML;
var dialogFooter = ej.buildTag("div", ej.buildTag("button#btnOK.e-
dialogBtnOK", "Hierarchy Selector")[0].outerHTML, { "float": "right",
"margin": "-5px 0 6px" })[0].outerHTML
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ejDialog = ej.buildTag("div#clientDialog.e-clientDialog", dialogContent +
dialogFooter, { "opacity": "1" }).attr("title", "Drill Through
Information")[0].outerHTML;
$(ejDialog).appendTo("#" + this._id);
$("#btnOK").ejButton().css({ margin: "30px 0 20px 0" });
$("#Grid1").ejGrid({
dataSource: gridData,
allowPaging: true,
allowTextWrap: true,
pageSettings: { pageSize: 8 }
});
this.element.find(".e-clientDialog").ejDialog({ width: "70%", content:
"#" + this._id, enableResize: false, close:
ej.proxy(ej.Pivot.closePreventPanel, this) });
var pivotGrid = this;
$("#btnOK").click(function () {
$(".e-dialog, .e-clientDialog, .tableDlg").remove();
if (pivotGrid.model.operationalMode ==
ej.PivotGrid.OperationalMode.ServerMode) {
pivotGrid._waitingPopup.show()
pivotGrid.doAjaxPost("POST", pivotGrid.model.url + "/" +
pivotGrid.model.serviceMethodSettings.drillThroughHierarchies,
JSON.stringify({ "currentReport":
JSON.parse(pivotGrid.getOlapReport()).Report, "layout":
pivotGrid.model.layout, "cellPos": "", "customObject":
JSON.stringify(pivotGrid.model.customObject) }), function (args) {
ej.Pivot.openHierarchySelector(pivotGrid, args);
})
}
});
}
</script>
```

When PivotGrid is rendered in server mode, below service methods need to be added in WCF/WebAPI for drill through operation.

For WebAPI controller, the below methods need to be added.

C#

```
[System.Web.Http.ActionName("DrillThroughHierarchies")]
[System.Web.Http.HttpPost]
public string DrillThroughHierarchies(Dictionary<string, object>
jsonResult)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return htmlHelper.DrillthroughHierarchies(DataManager,
jsonResult["layout"].ToString(), jsonResult["cellPos"].ToString());
}
[System.Web.Http.ActionName("DrillThroughDataTable")]
[System.Web.Http.HttpPost]
public Dictionary<string, object>
DrillThroughDataTable(Dictionary<string, object> jsonResult)
{
```

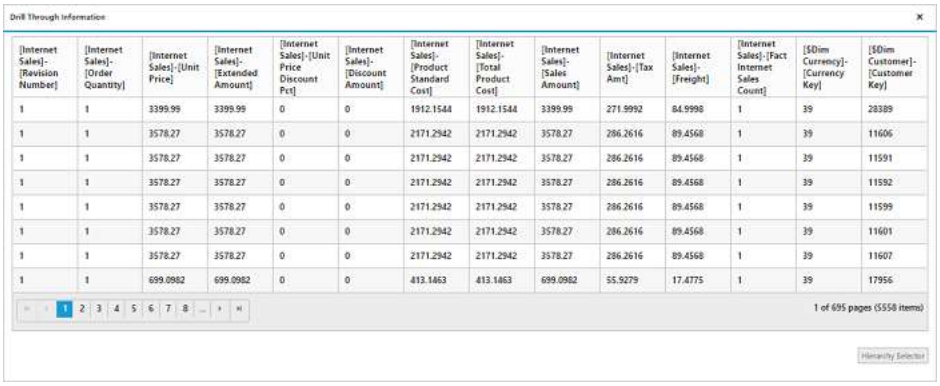
Copyright © 2001 - 2020 Syncfusion Inc.

```
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(jsonRe
sult["currentReport"].ToString()));
return htmlHelper.DrillthroughDataTable(DataManager,
jsonResult["layout"].ToString(), jsonResult["cellPos"].ToString(),
jsonResult["selector"].ToString());
}
```

For WCF service, the below methods need to be added.

C#

```
public string DrillThroughHierarchies(string currentReport, string
layout, string cellPos)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return htmlHelper.DrillthroughHierarchies(DataManager, layout, cellPos);
}
public Dictionary<string, object> DrillThroughDataTable(string
currentReport, string layout, string cellPos, string selector)
{
OlapDataManager DataManager = new OlapDataManager(connectionString);
DataManager.SetCurrentReport(OLAPUTILS.Utils.DeserializeOlapReport(curren
tReport));
return htmlHelper.DrillthroughDataTable(DataManager, layout, cellPos,
selector);
}
```



Relational

To enable drill-through support, set EnableDrillThrough property to true. Raw items are obtained through the DrillThrough event.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(.....).EnableDrillT
hrough(true).ClientSideEvents(events => events.DrillThrough("drillData"))
<script type="text/javascript">
function drillData(args) {
gridData = args.selectedData;
var dialogContent = ej.buildTag("div#Grid", { height: "50px"
})[0].outerHTML;
```

```
ejDialog = ej.buildTag("div#clientDialog.e-clientDialog", dialogContent,
{ "opacity": "1", "overflow": "auto" }).attr("title", "Drill Through
Information")[0].outerHTML;
$(ejDialog).appendTo("#" + this._id);
this.element.find(".e-clientDialog").ejDialog({ width: "70%", height:
"100%", content: "#" + this._id, enableResize: false, close:
ej.proxy(ej.Pivot.closePreventPanel, this) });
$("#Grid").ejGrid({
dataSource: gridData,
});
}
</script>
```



Resizing Column

Allows the user to change the column width by holding and dragging the column border using the mouse pointer.

Column Width Based on Size

The property EnableColumnResizing adjusts the width of each column based on size of the widget.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableColumnResizing(true).Dat
aSource(.....)
```

Column Width Based on Text

The ResizeColumnsToFit property automatically adjusts the width of each column based on the maximum content length available in the respective column.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ResizeColumnsToFit(true).DataS
ource(.....)
![Column resizing in ASP NET MVC pivot grid control](Column-
Resizing_images/columnresizing.png)
```

Styling

By default, PivotGrid supports **17** different themes.

- default-theme
- flat-azure-dark

Copyright © 2001 - 2020 Syncfusion Inc.

- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap-theme
- high-contrast-01
- high-contrast-02
- material
- office-365

You can also customize the appearance of the following component manually.

- PivotGrid
- Grouping Bar
- PivotTable Field List

To change the appearance of PivotGrid, apply the below customized CSS.

CSS

```
.e-pivotgrid table {
color: #565656;
background-color: White;
}
.e-pivotgrid th,
.e-pivotgrid td {
border: solid 1px;
border-color: #c4c4c4;
}
.e-pivotgrid .value {
background-color: White;
}
.e-pivotgrid .summary {
background-color: aqua !important;
color: #565656;
}
.e-pivotgrid .colheader, .e-pivotgrid .rowheader {
color: #5c5c5c;
background: white;
background-repeat: repeat;
}
.e-pivotgrid .colheader:hover, .e-pivotgrid .rowheader:hover {
color: white;
background: #91aa29;
background-repeat: repeat;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

To change the appearance of Grouping Bar, apply the below customized CSS.

CSS

```
.e-pivotgrid .summary {
background-color: aqua !important;
color: #565656;
}
.e-pivotgrid .e-grpRow, .e-pivotgrid .values, .e-pivotgrid .columns, .e-
pivotgrid .e-drag {
background: green;
}
.e-pivotgrid .e-pivotButton .e-btn.e-select, .e-pivotgrid .e-rows
.pivotButton, .e-pivotgrid .values .e-pivotButton, .e-pivotgrid .columns
.e-pivotButton, .e-pivotgrid .e-drag .e-pivotButton {
background: blanchedalmond;
}
.e-pivotgrid .colheader, .e-pivotgrid .rowheader {
color: #5c5c5c;
background: white;
background-repeat: repeat;
}
.e-pivotgrid .colheader:hover, .e-pivotgrid .rowheader:hover {
color: white;
background: #91aa29;
background-repeat: repeat;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

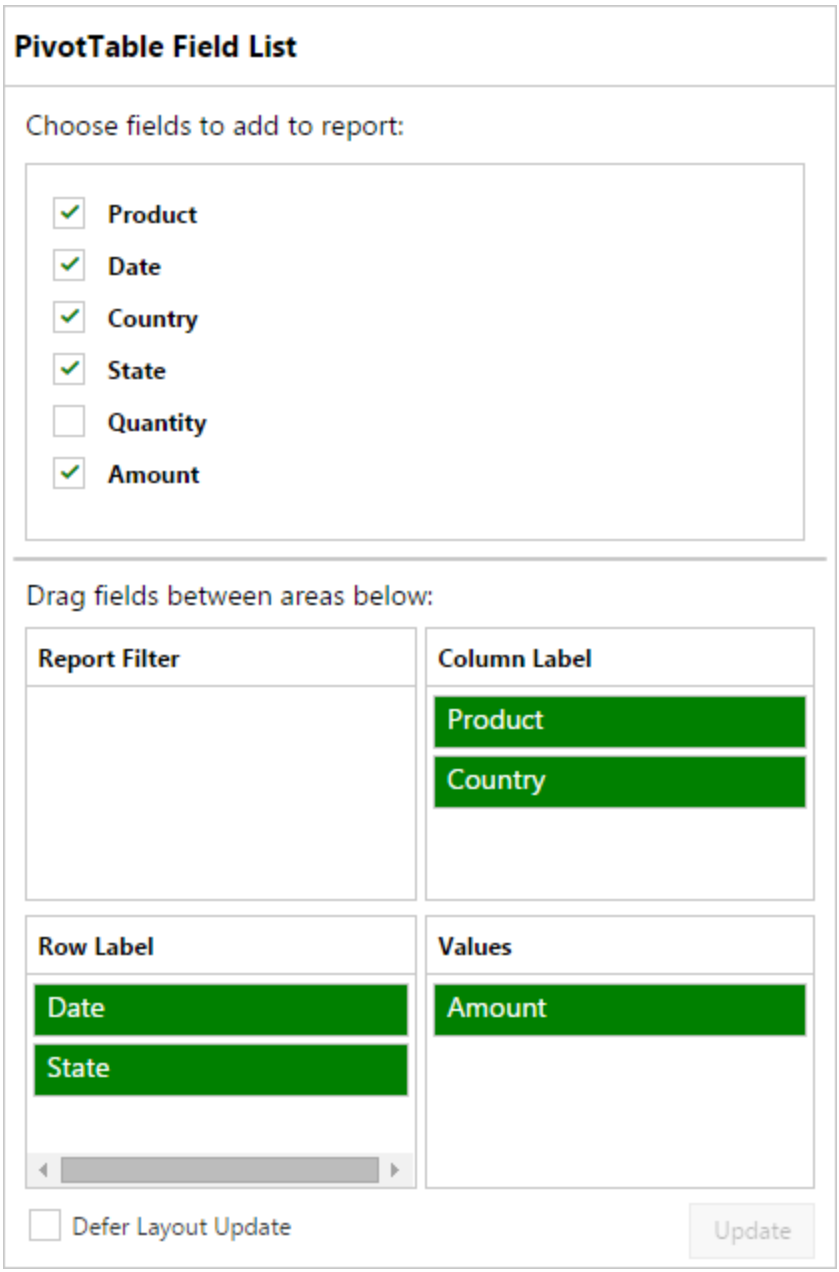


To change the appearance of PivotTable Field List, apply the below customized CSS.

CSS

```
.e-pivotschemadesigner, .e-pivotschemadesigner .e-fieldTable {
background-color: white;
color: black;
}
.e-pivotschemadesigner .e-treeview .e-active {
background: white;
}
.e-pivotschemadesigner .e-chkbox-wrap .e-chk-image.e-stop, .e-chkbox-wrap
.e-chk-image.e-checkmark {
color:green
}
.e-pivotschemadesigner .subheadText, .e-pivotschemadesigner .centerHead,
.e-pivotschemadesigner .e-schemaFieldTree.e-treeview .e-text{
color: black;
}
.e-pivotschemadesigner .e-pivotButton .e-pvtBtn {
background: green;
color: white;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



You can also customize the appearance of the Pivot component using our Theme Studio utility. To know more about Theme Studio click here.

Localization and Globalization

Localization in PivotGrid

You can localize the PivotGrid controls text with a collection of localized strings using **"ej.PivotGrid.Locale"** for different cultures. By default, the PivotGrid control is localized in **"en-US"** culture.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Locale("fr-
FR").ClientSideEvents(events =>
events.AfterServiceInvoke("OnAfterServiceInvoke") ...)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner1").Layout(Piv
otSchemaDesignerLayout.Excel)
@Html.EJ().Pivot().PivotPager("Pager1").Mode(PivotPagerMode.Both).Locale(
"fr-FR").TargetControlID("PivotGrid1")
< script type = "text/javascript" >
function OnAfterServiceInvoke(args) {
if (args.action == "initialize")
$("#PivotSchemaDesigner1").ejPivotSchemaDesigner({
pivotControl: this,
layout: ej.PivotSchemaDesigner.Layouts.Excel,
locale: "fr-FR"
});
}
ej.PivotSchemaDesigner.Locale["fr-FR"] = {
ClearFilter: "Effacer le filtre",
SelectField:"sélectionnez Champ",
Measures: "Mesures",
Warning: "Avertissement",
....
....
};
ej.PivotGrid.Locale["fr-FR"] = {
Sort:"Tri",
SelectField: "sélectionnez Champ",
LabelFilterLabel:"Afficher les éléments pour lesquels l'étiquette",
ValueFilterLabel:"Afficher les éléments pour lesquels",
...
...
};
ej.PivotPager.Locale["fr-FR"] = {
SeriesPage: "Série Page",
CategoricalPage: "Catégorique Page"
};
< /script>
```

The following table lists the default keywords in French culture for PivotGrid.

Keyword	Values
Sort	Tri
SelectField	sÃ©lectionnez Champ
LabelFilterLabel	Afficher les Ã©lÃ©ments pour lesquels l'Ã©tiquette
ValueFilterLabel	Afficher les Ã©lÃ©ments pour lesquels
LabelFilters	Filtres d'Ã©tiquetage
BeginsWith	Commence par
NotBeginsWith	Non Commence pa
EndsWith	se termine par
NotEndsWith	Non termine avec

Copyright © 2001 - 2020 Syncfusion Inc.

Contains	Contient
NotContains	Ne contient pas
ValueFilters	Filtres de valeur
ClearFilter	Clear Filter
Equals	Equals
NotEquals	Not Equals
GreaterThan	SupÃ©rieur
GreaterThanOrEqualTo	supÃ©rieur ou Ã©gal Ã
LessThan	Less Than
LessThanOrEqualTo	Moins ou Ã©gal Ã
Between	Entre
NotBetween	Non Entre
AddToFilter	Ajouter Ã filtre
AddToRow	Ajouter Ã la rangÃ©e
AddToColumn	Ajouter Ã la colonne
AddToValues	Ajouter aux valeurs
Warning	Avertissement
Error	Error
GroupingBarAlertMsg	Le champ que vous dÃ©placez ne peut pas Ãªtre placÃ© dans cette zone du rapport
Measures	Mesures
Expand	DÃ©velopper
Collapse	RÃ©duire
ToolTipRow	Row
ToolTipColumn	Colonne
ToolTipValue	Value
NoValue	Pas de valeu
SeriesPage	Series Page
CategoricalPage	CatÃ©gorique page
DragFieldHere	champ de glisser ic

Copyright © 2001 - 2020 Syncfusion Inc.

ColumnArea	Drop colonne ici
RowArea	Drop ligne ic
ValueArea	LÃ¢che valeurs ici
Close	Fermer
OK	OK
Cancel	Annuler
Remove	Supprimer
Goal	Goal
Status	Status
Trend	Trend
Value	value
ConditionalFormattingErrorMsg	La valeur donnÃ©e ne correspond pas
ConditionalFormattingConformMsg	Etes-vous sÃ»r de vouloir supprimer le format sÃ©lectionnÃ©?
EnterOperand1	Entrez OpÃ©rande1
EnterOperand2	Entrez OpÃ©rande2
ConditionalFormatting	Mise en forme conditionnelle
Condition	Type conditionnel
Value1	Value1
Value2	Value2
Editcondition	Modifier Condition
AddNew	Ajouter
Format	Format
Backcolor	Back Color
Borderrange	Border Range
Borderstyle	Border Style
Fontsize	Font Size
Fontstyle	aille de la police
Bordercolor	Couleur Bordure
AliceBlue	AliceBlue

Copyright © 2001 - 2020 Syncfusion Inc.

Black	Noir
Blue	Bleu
Brown	Brown
Gold	Gold
Green	Green
Lime	Lime
Maroon	Bordeaux
Orange	Orange
Pink	Pink
Red	rouge
Violet	Violet
White	Blanc
Yellow	Jaune
Solid	Solid
Dashed	pointillÃ©e
Dotted	Dotted
Double	Double
Groove	Groove
Inset	Encart
Outset	Outset
Ridge	Ridge
None	Aucun
Algerian	AlgÃ©rie
Arial	Arial
BodoniMT	Bodoni MT
BritannicBold	Britannic Bold
Cambria	Cambria
Calibri	Calibri
CourierNew	Courier New

Copyright © 2001 - 2020 Syncfusion Inc.

DejaVuSans	DejaVu Sans
Forte	Forte
Gerogia	Gerogia
Impact	Impact
SegoeUI	Segoe UI
Tahoma	Tahoma
TimesNewRoman	Times New Roman
Verdana	Verdana
CubeDimensionBrowser	Navigateur Dimension Cube
SelectHierarchy	SÃ©lectionnez HiÃ©rarchie
CalculatedField	Champ calculÃ©
Name	nom
Add:	Ajouter:
Formula	Formule:
Delete	Supprimer
Fields	Champs
CalculatedFieldNameNotFound	PrÃ©nom CalculatedField est introuvable
InsertField	InsÃ©rer un champ
EmptyField	S'il vous plaÃ®t entrez le nom de champ calculÃ© ou la formule
NotValid	formule donnÃ©e est pas valide
NotPresent	champ Valeur utilisÃ©e dans toute la formule de champ calculÃ© est pas prÃ©sent dans le PivotGrid
Confirm	champ calculÃ© avec le mÃªme nom existe dÃ©jÃ en raison de vouloir remplacer.?
CalcValue	Champ calculÃ© peut Ãªtre insÃ©rÃ© que dans le champ de la zone de valeur
Total	Total
GrandTotal	GrandTotal
DoesNotBeginsWith	N'a pas commence par
DoesNotEndsWith	Ne se termine par

Copyright © 2001 - 2020 Syncfusion Inc.

DoesNotContains	Ne contient
DoesNotEquals	N'est pas Ã©gaux
IsGreaterThan	Est supÃ©rieure Ã
IsGreaterThanOrEqualTo	Est supÃ©rieure ou Ã©gale Ã
IsLessThan	Est infÃ©rieure Ã
IsLessThanOrEqualTo	Est infÃ©rieure ou Ã©gale Ã
NumberFormatting	Formatage des chiffres
FrozenHeaders	En-tÃªtes congelÃ©
CellSelection	SÃ©lection de cellules
CellContext	Contexte cellulaire
ColumnResize	Redimensionner la colonne
ExcelLikeLayout	Comme la mise en page d'Excel
FrozenHeader	En-tÃªte congelÃ©e
AdvancedFiltering	Filtrage avancÃ©
Amount	QuantitÃ©
Quantity	Quantity
Measures	Mesures visant
NumberFormats	Les formats de nombre
Exporting	L'exportation
FileName	Nom de fichier
ToolTip	ExtrÃ©mitÃ© de l'outil
RTL	RTL
CollapseByDefault	Par dÃ©faut l'effondrement
EnableDisablePaging	Enalbe / DÃ©sactiver la pagination
PagingOptions	Options d'appel
CategoricalPageSize	Taille de page catÃ©gorique
SeriesPageSize	Taille de page sÃ©rie
HyperLink	Hyper Link
CellEditing	Montage de cellules

Copyright © 2001 - 2020 Syncfusion Inc.

GroupingBar	Bar de groupement
SummaryCustomization	Personnalisation Sommaire
SummaryTypes	Types Sommaire
SummaryType	Type de statistique
EnableRowHeaderHyperlink	Activer l'en-tÃªte de ligne HyperLink
EnableColumnHeaderHyperlink	Activer l'en-tÃªte de colonne HyperLink
EnableValueCellHyperlink	Activer la cellule Valeur HyperLink
EnableSummaryCellHyperlink	Activer la cellule de synthÃ¨se HyperLink
HideGrandTotal	Masquer Grand Total
HideSubTotal	Masquer Sous-total
Both	Les deux
Sum	Somme
Average	La moyenne
Count	Count
Min	Min
Max	Max
Excel	Excel
Word	Mot
PDF	PDF
CSV	CSV
MultipleItems	Plusieurs Ã©lÃ©ments
All	Tous les
Search	Recherchez

The following table lists the default keywords in French culture for PivotTable Field List.

Keywords	Values
PivotTableFieldList	Liste de champs de tableau croisé dynamique
ChooseFieldsToAddToReport	Choisissez champs à ajouter à Signaler"
DragFieldBetweenAreasBelow	Faites glisser terrain entre les zones ci-dessous
ReportFilter	Rapport Filtre

Copyright © 2001 - 2020 Syncfusion Inc.

ColumnLabel	colonne Étiquette
RowLabel	Label Row
Values	Valeurs
ClearFilter	Effacer le filtre
SelectField	sélectionnez Champ
Measures	Mesures
Warning	Avertissement
AlertMsg	Le champ que vous déplacez ne peut pas être placé dans cette zone du rapport
Goal	Goal
Status	Status
Trend	Trend
AddToFilter	Ajouter à filtrer
AddToRow	Ajouter à la rangée
AddToColumn	Ajouter à la colonne
AddToValues	Ajouter à la valeur
DeferLayoutUpdate	Différer Mise à jour
Update	Mettre à jour
OK	OK
Cancel	Annuler
Close	Fermer
DoesNotBeginsWith	N'a pas commence par
DoesNotEndsWith	Ne se termine par
DoesNotContains	Ne contient
DoesNotEquals	N'est pas égaux
IsGreaterThan	Est supérieure à
IsGreaterThanOrEqualTo	Est supérieure ou égale à
IsLessThan	Est inférieure à
IsLessThanOrEqualTo	Est inférieure ou égale à
Sort	Trier

Copyright © 2001 - 2020 Syncfusion Inc.

LabelFilterLabel	Afficher les éléments pour lesquels l'étiquette
ValueFilterLabel	Afficher les éléments pour lesquels
LabelFilters	Les filtres de l'étiquette
BeginsWith	Commence par
NotBeginsWith	Commence pas avec
EndsWith	Se termine par
NotEndsWith	Pas une fin avec
Contains	Contient
NotContains	Contient pas
ValueFilters	Les filtres de valeur
ClearFilter	Clear Filter
Equals	Est égal à
NotEquals	Pas égaux
GreaterThan	Plus de
GreaterThanOrEqualTo	Supérieure ou égale à
LessThan	Moins de
LessThanOrEqualTo	Inférieure ou égale à
Between	Entre
NotBetween	Pas entre
Search	Recherchez

The following table lists the default keywords in French culture for Pivot Pager.

Keywords	Values
SeriesPage	SÃ©rie Page
CategoricalPage	CatÃ©gorique Page
Error	Error
OK	OK
Close	Fermer
PageCountErrorMsg	Entrez le numÃ©ro de page valide

Copyright © 2001 - 2020 Syncfusion Inc.

Localization and Globalization of Cube Info (Client Mode)

Content displayed within the PivotGrid control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

- To get localized data from OLAP Cube, we need to set **"Locale Identifier"** in the connection string to a specific culture in the **"Data"** property present inside **"DataSource"**.
- To bind the globalized content in PivotGrid control, we need to set **"Locale"** property to a specific culture and want to refer specific culture file in the sample.

Note: Culture files are present under **"[installed drive]:\Users\[user name]\AppData\Local\Syncfusion\EssentialStudio\X.X.X.X\MVC\Samples\web\Scripts\cultures".**

ASPX-CS

```
//1036 refers to "fr-FR" culture.
@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource =>
dataSource.Rows(rows=>{rows.FieldName("[Date].[Fiscal]").Add();}).Columns
(columns=>{columns.FieldName("[Customer].[Customer
Geography]").Add();}).Values(values => { values.Measures(measures => {
measures.FieldName("[Measures].[Internet Sales Amount]").Add();
}).Axis(AxisName.Column).Add();})
.Data("https://bi.syncfusion.com/olap/msmdpump.dll;Locale
Identifier=1036;"").Catalog("Adventure Works DW 2008 SE").Cube("Adventure
Works"))
```

Localization and Globalization of Cube Info (Server Mode)

Content displayed within the PivotGrid control are obtained from the OLAP Cube. So following are the steps that needs to be done to get the localized and globalized Cube content.

- To get the localized string based on different cultures, from OLAP Cube, you need to set **"Locale Identifier"** in the connection string to a specific culture.
- To bind the globalized content in PivotGrid control, you need to set **"Culture"** and OverrideDefaultFormatStrings properties in OlapDataManager class to a specific culture.

C#

```
//1036 refers to "fr-FR" culture.
string connectionString = "Data Source=localhost; Initial
Catalog=Adventure Works DW; Locale Identifier=1036;";
DataManager = new OlapDataManager(connectionString);
DataManager.Culture = new System.Globalization.CultureInfo(1036);
DataManager.OverrideDefaultFormatStrings = true;
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Australie	Canada	France	Allemagne	Royaume-Uni	États-Unis	Total
	Montant des Ventes (Internet)						
FY 2002	2 568 701,39 €	573 100,97 €	414 245,32 €	513 353,17 €	550 507,33 €	2 452 176,07 €	7 072 084,24 €
FY 2003	2 099 585,43 €	305 010,69 €	633 399,70 €	593 247,24 €	696 594,97 €	1 434 296,26 €	5 762 134,30 €
FY 2004	4 383 479,54 €	1 068 879,50 €	1 592 860,75 €	1 784 107,09 €	2 140 388,50 €	5 483 882,67 €	16 473 618,05 €
FY 2005	9 234,23 €	10 853,70 €	3 491,95 €	3 604,83 €	4 221,41 €	19 434,51 €	50 840,63 €
Total	9 061 000,58 €	1 977 844,86 €	2 644 017,71 €	2 894 312,34 €	3 391 712,21 €	9 389 789,51 €	29 358 677,22 €

Localization and Globalization of Relational Info (Client Mode)

Content displayed within the PivotGrid control are obtained from the Relational datasource. So following are the steps that needs to done to get localized as well as globalized content.

- To get the localized content, the Relational datasource must have localized headers in them which will be directly applied to PivotGrid.
- To globalize the values appear in PivotGrid , we need to set **“Format”** and **“Locale”** property to a specific culture and want to refer specific culture file in the sample.

Note: Culture files are present under **“[installed drive]:\Users\[user name]\AppData\Local\Syncfusion\EssentialStudio\X.X.X.X\MVC\Samples\web\Scripts\cultures”.**

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEve
nts => clientSideEvents.Load("onLoad")).DataSource(dataSource =>
dataSource.Rows(rows => {
rows.FieldName("Country").FieldCaption("Country").Add();
}).Columns(columns => {
columns.FieldName("Product").FieldCaption("Product").Add();
}).Values(values => {
values.FieldName("Amount").Format("currency").Add(); }))
```

Localization and Globalization of Relational Info (Server Mode)

Content displayed within the PivotGrid control are obtained from the Relational datasource. So following are the steps that needs to done to get localized as well as globalized content.

- To get the localized content, the Relational datasource must have localized headers in them which will be directly applied to PivotGrid.
- **“Format”** settings in PivotComputationInfo class would globalize the values appear in PivotGrid.

C#

```
PivotReport pivotSetting = new PivotReport();
//...
pivotSetting.PivotCalculations.Add(new PivotComputationInfo {
CalculationName = "Amount", Description = "Amount", FieldHeader =
"Amount", FieldName = "Amount", Format = "E", SummaryType =
Syncfusion.PivotAnalysis.Base.SummaryType.DoubleTotalSum
});
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
FY 2005	1.329543E+008	2.907120E+007	3.706140E+007	1.990869E+008
FY 2006	5.239890E+007		1.289550E+007	6.529440E+007
FY 2007	2.920320E+007		8.480100E+006	3.768330E+007
FY 2008	1.529940E+007		5.793300E+006	2.109270E+007
FY 2009	6.059400E+006		3.424500E+006	9.483900E+006
Grand Total	2.359152E+008	2.907120E+007	6.765480E+007	3.326412E+008

RTL

You can render our PivotGrid control from Right to Left by setting EnableRTL property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/OLAPService"
)).EnableRTL(true)
```

Total	United States	United Kingdom	Germany	France	Canada	Australia	
						Internet Sales Amount	
$7,072,084.24	$2,452,176.07	$550,507.33	$513,353.17	$414,245.32	$573,100.97	$2,568,701.39	FY 2002
$5,762,134.30	$1,434,296.26	$696,594.97	$593,247.24	$633,399.70	$305,010.69	$2,099,585.43	FY 2003
$16,473,618.05	$5,483,882.67	$2,140,388.50	$1,784,107.09	$1,592,880.75	$1,088,879.50	$4,383,479.54	FY 2004
$50,840.63	$19,434.51	$4,221.41	$3,604.83	$3,491.95	$10,853.70	$9,234.23	FY 2005
$29,358,677.22	$9,389,789.51	$3,391,712.21	$2,894,312.34	$2,644,017.71	$1,977,844.86	$9,061,000.58	Total

Responsive

PivotGrid and PivotTable Field list control supports responsive rendering based on the target device (desktop and tablet) resolution. It supports resolution upto 1024x600. You can enable responsiveness in PivotGrid by setting IsResponsive property to true.

On resizing the browser, the PivotTable Field list will get collapse and an icon will appear on the left-hand side of the browser. User can toggle its view and perform UI interaction.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").Url(Url.Content("/OLAPService"
)).IsResponsive(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

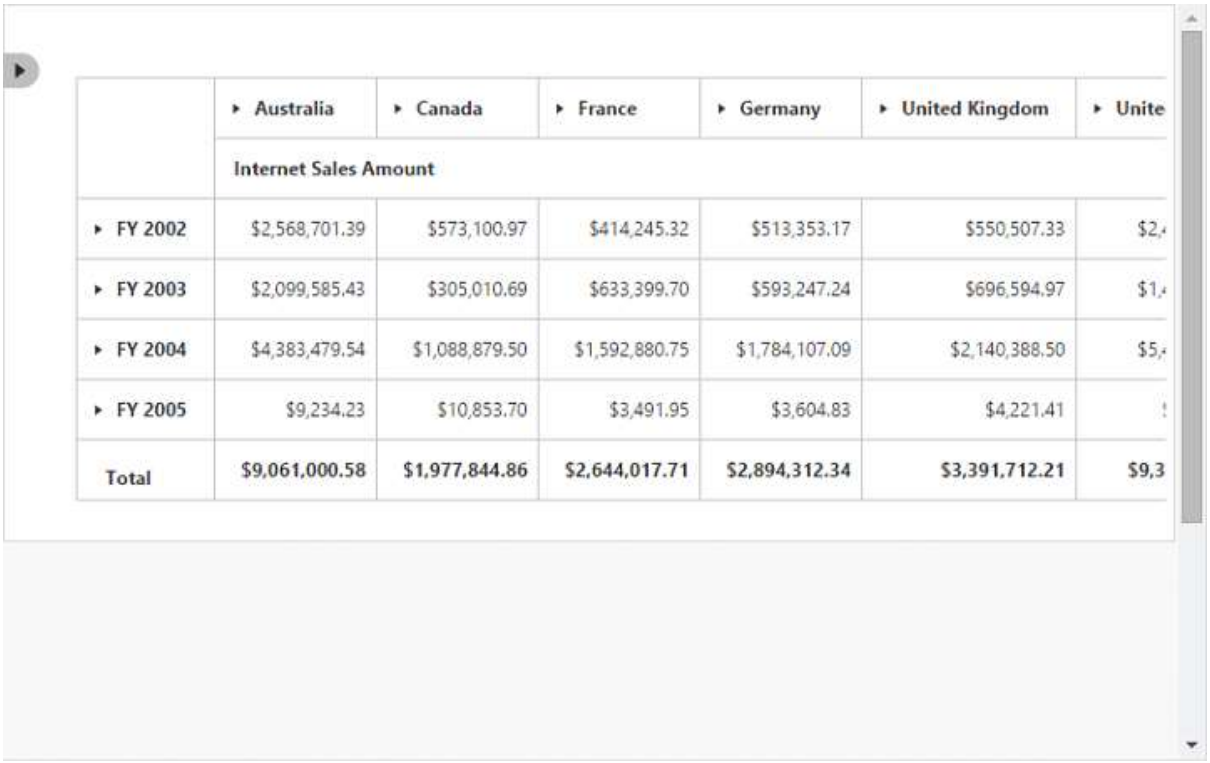
Normal PivotGrid



	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

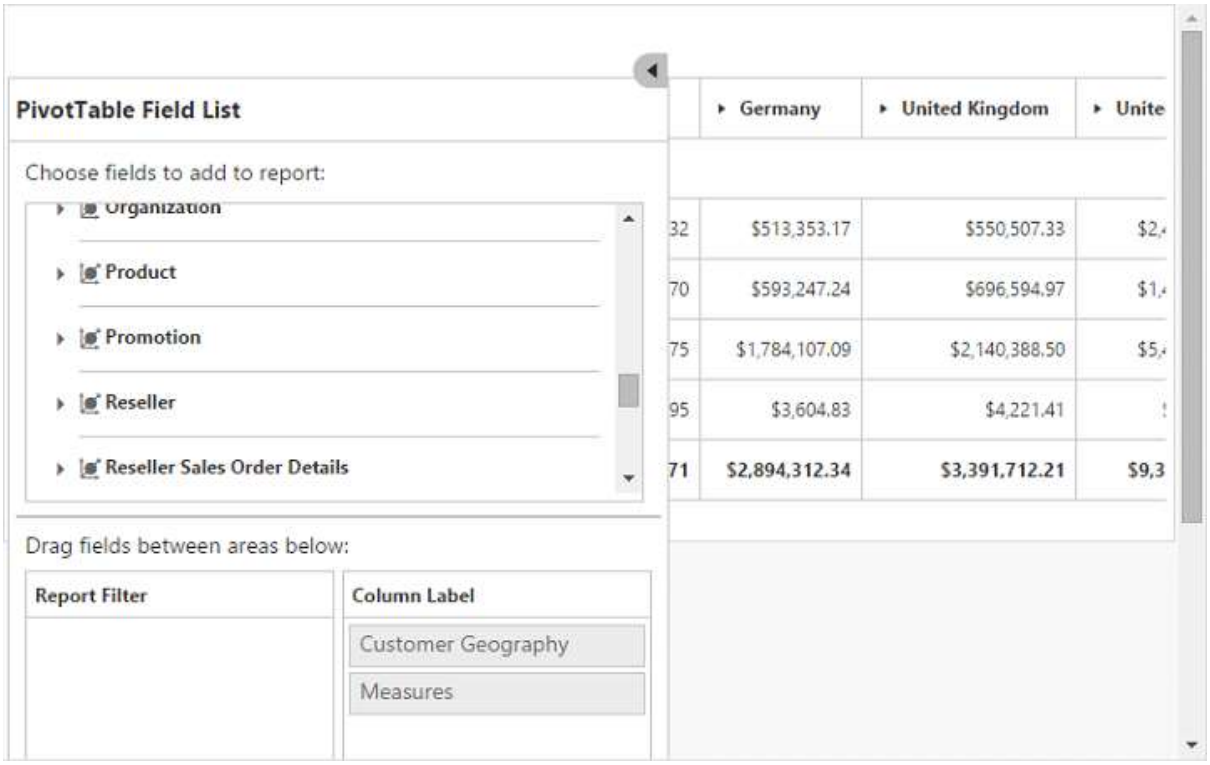
Responsive PivotGrid

Copyright © 2001 - 2020 Syncfusion Inc.



	Australia	Canada	France	Germany	United Kingdom	Unite
	Internet Sales Amount					
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,4
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,4
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,4
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,3

Responsive PivotTable Field List - Collapsed



Responsive PivotTable Field List - Expanded

ToolTip

Allows you to display the details of the cell on hovering value cells. By default, tooltip is enabled. You can disable tooltip in PivotGrid by setting the EnableToolTip property to false.

ASPX-CS

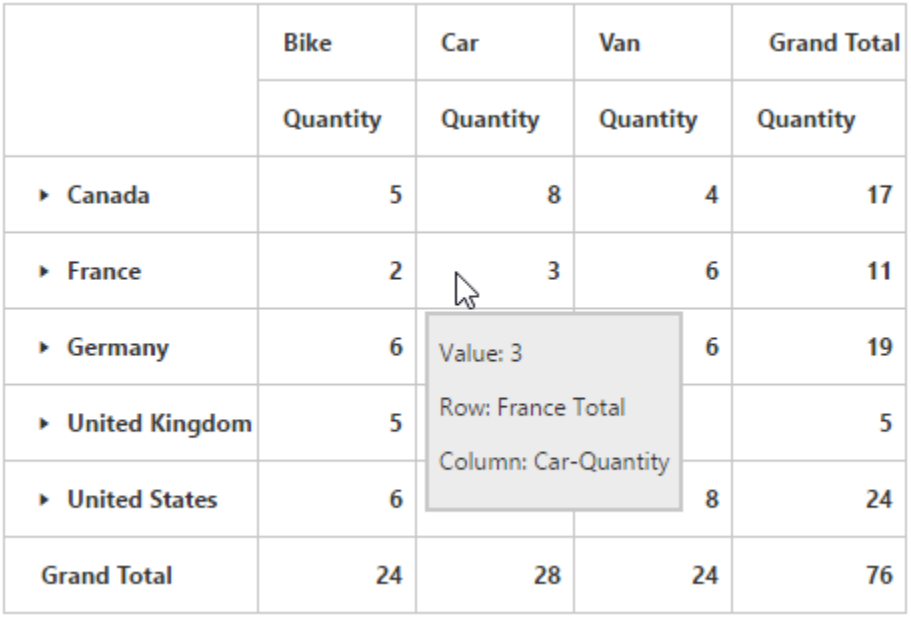
```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableToolTip(false)
```

ToolTip Animation

The PivotGrid provides option to animate tooltip displayed in the grid. The animation enhances the appearance of tooltip by displaying it slowly. You can enable animation in tooltip by setting EnableToolTipAnimation property to true.

ASPX-CS

```
@Html.EJ().Pivot().PivotGrid("PivotGrid1").EnableToolTipAnimation(true)
```



How to

Configure pivot grid through the model properties

There comes a time when you want to set pivot grid properties other than using PivotGridPropertiesBuilder i.e., in the Controller side. At that time, you can pass PivotGridProperties model instance as a parameter to the pivot grid helper method.

We can set properties to pivot grid control in server-side using the PivotGridProperties class and it is used in the view page with the help of pivot grid helper overload.

In addition to the id parameter of pivot grid helper method, we can also pass PivotGridProperties model as another parameter to the pivot grid helper method.

Relational

The following code example explains how to render the pivot grid control in relational mode.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@model Syncfusion.JavaScript.Models.PivotGridProperties
@(Html.EJ().Pivot().PivotGrid("PivotGrid", Model))
<script type="text/javascript">
function load(args) {
args.model.dataSource.data = pivot_dataset;
}
</script>
```

C#

```
namespace PivotGrid
{
public class PivotGridController: Controller
{
public ActionResult PivotGridFeatures()
{
Syncfusion.JavaScript.Models.PivotGridProperties pivotgrid = new
Syncfusion.JavaScript.Models.PivotGridProperties();
PivotDataSource pivotgridDS = new PivotDataSource();
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "Country", FieldCaption = "Country"
});
rows.Add(new Field() { FieldName = "State", FieldCaption = "State" });
pivotgridDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "Product", FieldCaption = "Product"
});
pivotgridDS.Columns = cols;
pivotgrid.Load = "load";
List<Field> values = new List<Field>();
values.Add(new Field() { FieldName = "Amount" });
pivotgridDS.Values = values;
pivotgrid.DataSource = pivotgridDS;
return View(pivotgrid);
}
}
}
```

As a result of the previous code example, the pivot grid will be displayed as shown below:

Copyright © 2001 - 2020 Syncfusion Inc.

	Bike	Car	Van	Grand Total
	Amount	Amount	Amount	Amount
Canada	250	500	300	1050
France	200	300	400	900
Germany	300	400	350	1050
United Kingdom	150			150
United States	300	650	500	1450
Grand Total	1200	1850	1550	4600

OLAP

The following code example explains how to render the pivot grid control in OLAP mode.

RAZOR

```
@model Syncfusion.JavaScript.Models.PivotGridProperties
@(Html.EJ().Pivot().PivotGrid("PivotGrid", Model))
```

C#

```
namespace PivotGrid
{
public class PivotGridController: Controller
{
public ActionResult PivotGridFeatures()
{
Syncfusion.JavaScript.Models.PivotGridProperties pivotgrid = new
Syncfusion.JavaScript.Models.PivotGridProperties();
PivotDataSource pivotgridDS = new PivotDataSource();
pivotgridDS.Data = "https://bi.syncfusion.com/olap/msmdpump.dll";
pivotgridDS.Cube = "Adventure Works";
pivotgridDS.Catalog = "Adventure Works DW 2008 SE";
List<Field> rows = new List<Field>();
rows.Add(new Field() { FieldName = "[Date].[Fiscal]" });
pivotgridDS.Rows = rows;
List<Field> cols = new List<Field>();
cols.Add(new Field() { FieldName = "[Customer].[Customer Geography]" });
pivotgridDS.Columns = cols;
List<MeasuresItems> measures = new List<MeasuresItems>();
measures.Add(new MeasuresItems() { FieldName = "[Measures].[Internet
Sales Amount]" });
List<Field> values = new List<Field>();
values.Add(new Field() { Measures = measures, Axis =
Syncfusion.JavaScript.AxisName.Column });
pivotgridDS.Values = values;
pivotgrid.DataSource = pivotgridDS;
```

```
return View(pivotgrid);
}
}
}
```

As a result of the previous code example, the pivot grid will be displayed as shown below:

	Australia	Canada	France	Germany	United Kingdom	United States	Total
	Internet Sales Amount						
FY 2002	$2,568,701.39	$573,100.97	$414,245.32	$513,353.17	$550,507.33	$2,452,176.07	$7,072,084.24
FY 2003	$2,099,585.43	$305,010.69	$633,399.70	$593,247.24	$696,594.97	$1,434,296.26	$5,762,134.30
FY 2004	$4,383,479.54	$1,088,879.50	$1,592,880.75	$1,784,107.09	$2,140,388.50	$5,483,882.67	$16,473,618.05
FY 2005	$9,234.23	$10,853.70	$3,491.95	$3,604.83	$4,221.41	$19,434.51	$50,840.63
Total	$9,061,000.58	$1,977,844.86	$2,644,017.71	$2,894,312.34	$3,391,712.21	$9,389,789.51	$29,358,677.22

ProgressBar

Progress Bar

Essential ProgressBar control is a graphical control element used to visualize the changing status of an extended operation. It is available in ASP.NET MVC product. ProgressBar provides an interactive way to display the progression of the task. You can configure the item size, orientation and the display text on the ProgressBar control.

Key Features

Some important features of ProgressBar are as follows:

- Value support— you can set a numeric value for the ProgressBar status.
- Percentage support— you can set a percentage value for the control.
- CustomText— ProgressBar control allows you to set the text value that is to be displayed inside the control.
- StateMaintenance— the progress value is maintained even after the page is refreshed to resume the current progress of the operation.
- Range – to set the minimum and maximum values of the ProgressBar control
- Theme—ProgressBar control consists of 12 built-in themes (6 – metro and 6 – gradient effects), and also supports custom skins to set user-defined themes.

Getting Started

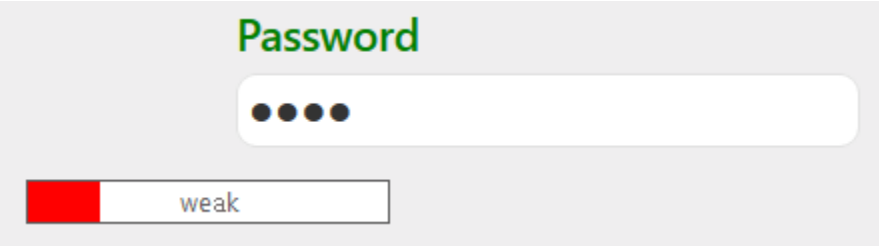
This section briefly describes how to create a Progress Bar control using ASP.NET MVC and learn its features.

Create your first Progress Bar in MVC

ASP.NET MVCProgress Bar control provides support to display a Progress Bar that allows you to change the process of the ProgressBar animations and flexible APIs. Using the following guidelines, you can create the ProgressBar to validate the Password strength.

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot illustrates the functionality of a Progress Bar and displays the final result of the Password Strength Validation for your password using Progress Bar.



Progress Bar

Create a Textbox and Progress Bar

ASP.NET MVC Progress Bar control indicates the current progress of an operation like uploading a document through a simple interface. You can easily create the Progress bar control using simple HTML helpers as follows.

1. Create an MVC Project and add necessary assemblies, scripts and styles to it.

Refer MVC-Getting Started.

2. Add the following code to the corresponding view page to render Progress Bar.

ASPX-CS

```
<div class="start" >
<label for="start">Password</label>
<input type="password" id="password"/>
@Html.EJ().ProgressBar("progressBar").Value(20).Height("20px").Width("180
px")
</div>
```

3. Add the following styles to show the Progress Bar and Textbox.

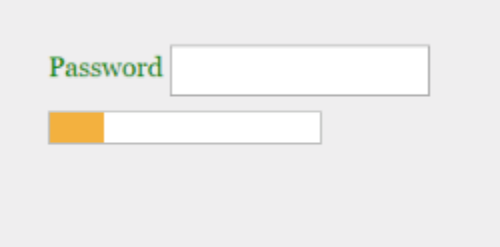
CSS

```
<style>
.start
{
margin-left: 105px;
color: green;
font-size: 18px;
}
.control
{
margin-bottom: 5px;
margin-left: 230px;
}
#progressBar
{
margin-top: 10px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</style>
```

4. Execute the above code to render the following output.



Find the Strength of the Password

In this scenario, the advancement of the Progress Bar is changed according to the length and special characters present in the text of the password field. This is achieved by binding the change in the properties of your control and by checking the length of the password field.

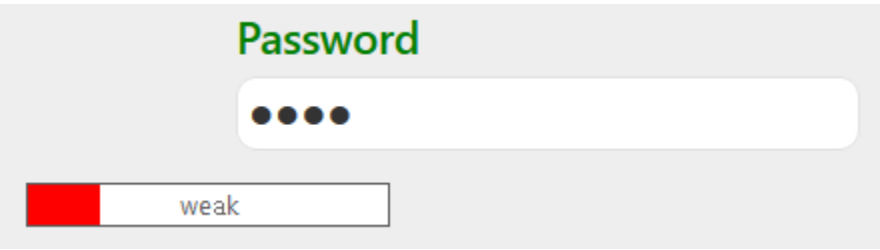
JS

```
<script>
var progressObj, k = 10, i = 0;
$(document).keydown(function() {
i = $("#password").val().length;
if (i < 4)
{
progress2();
$('.e-progress').css({ background: 'red' });
}
else if (i > 4 && i < 7)
{
progress1();
$('.e-progress').css({ background: 'yellow' });
}
else if (i > 7)
{
var password = $("#password").val();
if (/^[a-zA-Z0-9- ]*$/.test(password) == false)
{
progress();
$('.e-progress').css({ background: 'green' });
}
}
});
$(function ()
{
progressObj = $("#progressBar").data("ejProgressBar");
});
function progress()
{
progressObj.option("text", " very strong");
progressObj.option("percentage", k + 90);
}
function progress1()
```

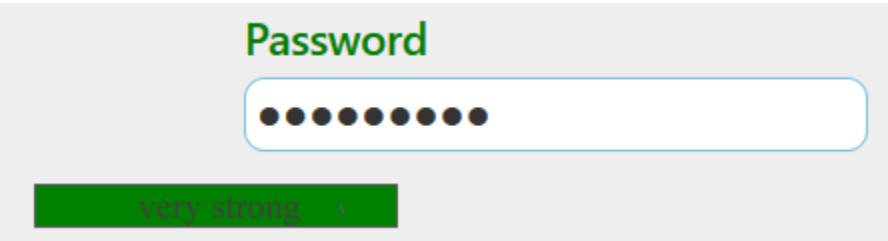
```
{
progressObj.option("text", "strong");
progressObj.option("percentage", k + 50);
}
function progress2()
{
progressObj.option("percentage", k + 10);
progressObj.option("text", "weak");
}
</script>
```

- The progress() function changes the text inside the Progress Bar to Very Strong and percentage to 100, and it is invoked when the length of the text is more than 7 and the text contains a symbol in it. Then it shows the Progress Bar in violet color.
- The progress1() function changes the text inside the Progress Bar to Strong and percentage to 60, and it is invoked when the length of the text is more than 4. Then it shows the Progress Bar in green color
- The progress 2() function changes the text inside the Progress Bar to Weak and percentage to 30, and it is invoked when the length of the text is less than 4. Then it shows the ProgressBar in red color.

The following screenshot displays a Progress Bar control for the above scenario.







Define value

Value

The Value for the ProgressBar is set by using ‘Value’ property. The Value should be between the minimum (min) and the maximum (max) values (number) of the ProgressBar. By default, the MinValue is 0 and the MaxValue is 100 in ProgressBar, and the ‘Value’ is set to 0(number).

Copyright © 2001 - 2020 Syncfusion Inc.

The following steps explain you on how to set the Value for the ProgressBar widget.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render the ProgressBar control with customized value.
@Html.EJ().ProgressBar("progressbar").MinValue(40).MaxValue(80).Value(60)
.Height("20").Width("500")
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getValue()});
});
</script>
```

The following screenshot displays the output for the above code.



Percentage

The ProgressBar value is set in Percentage by using the 'Percentage' property. The value should be between the min and max values (number) of the ProgressBar. By default, the MinValue is 0 and the MaxValue is 100 in ProgressBar, and percentage is set to 0 (number).

The following steps explain you on how to set the value in Percentage for the ProgressBar widget.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render the ProgressBar control with customized percentage.
@Html.EJ().ProgressBar("progressbar").MinValue(40).MaxValue(80).Percentag
e(60).Height("20").Width("500")
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getPercentage() + " %"});
});
</script>
```

The following screenshot displays the output.

Copyright © 2001 - 2020 Syncfusion Inc.



Percentage in Progress Bar

Setting Range

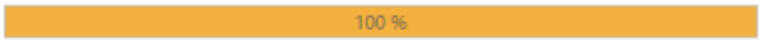
The range of the ProgressBar is set by using minimum and maximum values. The Minimum value specifies the value where the ProgressBar shows the process to have started. The Maximum value specifies the value where the process is completed. You can set the range by using the 'MinValue' and 'MaxValue' property.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render the ProgressBar control with customized range.
@Html.EJ().ProgressBar("progressbar").MinValue(40).MaxValue(80).Value(80)
.Height("20").Width("500")
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getPercentage() + " %" });
});
</script>
```

The following screenshot displays the output.



Appearance and Styling

Adjusting ProgressBar size

ProgressBar widget provides the ability to change or adjust the ProgressBar size. The 'Height' and 'Width' properties in the ProgressBar widget allows you to set the maximum height and maximum width for the ProgressBar. The value set to this property is string or Number type.

The following steps explain you on how to adjust the ProgressBar size.

1. In the VIEW page, add a helper element to render the ProgressBar widget

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render the ProgressBar control with customized size.
@Html.EJ().ProgressBar("progressbar").Value(40).Height("40").Width("400")
```

```
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getValue() + " %"});
});
</script>
```

The following screenshot displays the output.



Custom text

Custom text is displayed when the ProgressBar shows different levels of progress in the ProgressBar. Support for Custom Text to mention the percentage or any other message inside the ProgressBar is possible.

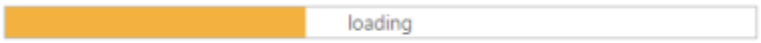
The following steps explain the configuration of the Custom Text for the ProgressBar widget.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render the ProgressBar control with customized text.
@Html.EJ().ProgressBar("progressbar").Text("loading").Value(40).Height("2
0").Width("500")
```

The following screenshot displays the output.



Theme

The ProgressBar widget style and appearance are controlled based on CSS classes. In order to apply Theme to the ProgressBar widget, you can refer two files, namely, ej.widgets.core.min.css and ej.theme.min.css. When the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these both.

By default, there are 12 themes' support available for the ProgressBar widget namely,

- Default-theme
- Flat-azure-dark
- Fat-lime
- Flat-lime-dark

Copyright © 2001 - 2020 Syncfusion Inc.

- Flat-saffron
- Flat-saffron-dark
- Gradient-azure
- Gradient-azure-dark
- Gradient-lime
- Gradient-lime-dark
- Gradient-saffron
- Gradient-saffron-dark

CSS class

To apply custom styles to the ProgressBar widget, you can specify the CssClass property. The specified CSS name is added in the root of the ProgressBar widget.

The following code example is used to render the ProgressBar widget with customized style.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render the ProgressBar control with customized style.
@Html.EJ().ProgressBar("progressbar").Value(70).Height("20").Width("500")
.CssClass("custom")
```

JS

```
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getValue() + " %"});
});
```

2. Add the following styles to render the ProgressBar with customized style.

CSS

```
<style type="text/css">
.custom .e-progress
{
background-color:gray;
}
</style>
```

The following screenshot displays the output.



Copyright © 2001 - 2020 Syncfusion Inc.

Enabling the ProgressBar

The ProgressBar is enabled by using the 'Enabled' Property. When this property is set to 'false', it disables the ProgressBar widget. By default, 'Enabled' property is set to 'true' in the ProgressBar widget.

The following steps explain how to disable the ProgressBar widget when 'Enabled' property is set to 'false'.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
disable the ProgressBar control.
@Html.EJ().ProgressBar("progressbar").Value(70).Height("20").Width("500")
.Enabled(false)
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getValue() + " %"});
});
</script>
```

The following screenshot displays the output for the above code.

Disabled Progress Bar

State Maintenance

Save current model value to cookies for State Maintenance. While refreshing the ProgressBar widget, page retains the model value applied from browser cookies. By default, the 'EnablePersistence' property is set to 'false' in the ProgressBar.

The following steps explain the State Maintenance in the ProgressBar control.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

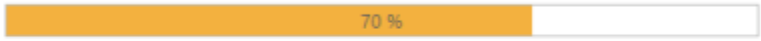
ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
enable the state maintenance in the ProgressBar control.
@Html.EJ().ProgressBar("progressbar").Value(70).Height("20").Width("500")
.EnablePersistence(true)
<script>
var progress;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getValue() + " %"});
});
</script>
```

The following screenshot displays the output.



RTL Support

Right-to-left starts from the right of the page and continues to the left. By default, this option is set to 'false' in the ProgressBar control. The EnableRTL option allows the ProgressBar control to display it in the right to left direction.

The following steps explain how to enable the RTL property of the ProgressBar control.

1. In the VIEW page, add a helper element to render the ProgressBar widget.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render ProgressBar control to display it in the right-to-left direction.
@Html.EJ().ProgressBar("progressbar").Value(70).Height("20").Width("500")
.EnableRTL(true)
<script>
var progress;
$(document).ready(function ()
{
progress = $("#progressbar").data("ejProgressBar");
progress.setModel({ text: progress.getValue() + " %"});
});
</script>
```

The following screenshot displays the output.



How To

Render ProgressBar from code behind

ProgressBar can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

Copyright © 2001 - 2020 Syncfusion Inc.

The following code illustrates the rendering of ProgressBar using ProgressBar properties from code behind.

RAZOR

```
@{Html.EJ().ProgressBar("progress", Model).Render();}
```

C#

```
public ActionResult Index()
{
ProgressBarProperties Obj = new ProgressBarProperties();
Obj.Text = "Loading...";
Obj.Value = 60;
return View(Obj);
}
```

RadialMenu

RadialMenu

Essential ASP.NET MVC Radial Menu control is a context that represents the menu items are arranged in a circular order with a centric button element in it. By default, only the center button is visible. The **Radial Menu** displays the root level menu item with rotational animation effects on clicking the center menu button. You can close it either by clicking anywhere in the document or by clicking the center button where the root level items are displayed.

Key Features

- **Nested Menu**: Supports to render the multiple level of sub menu items.
- **Image Customization**: Enables to customize images for all levels of menu item.
- **Item Customization**: Supports to customize Radial Menu items with badges and slider settings.
- **Dimension**: Allows to customize Radial Menu radius and position.

Getting Started

Creating your first RadialMenu in MVC application

1. Create an MVC Project and add necessary assemblies, scripts and CSS files given in MVC-Getting Started Documentation.
2. Add RadialMenu control using the helper from EJ namespace.

HTML

```
@Html.EJ().RadialMenu("RadialMenu1")
```

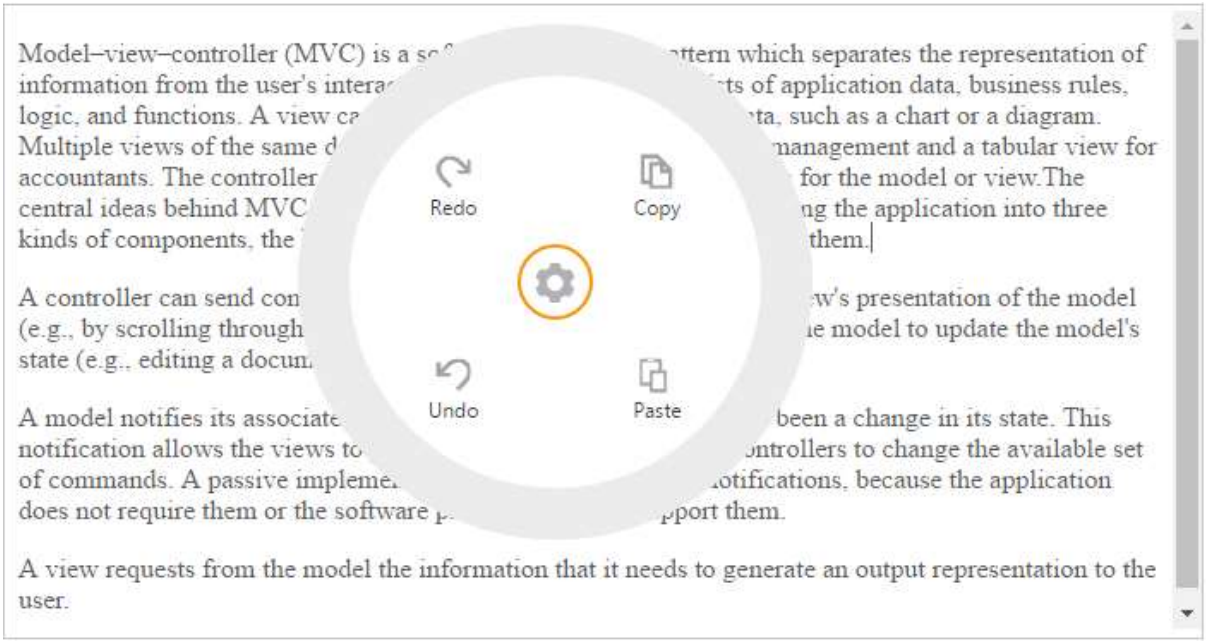
3. Execute the code and get a empty RadialMenu control as below

Copyright © 2001 - 2020 Syncfusion Inc.



Create a Simple RadialMenu with RTE content

RadialMenu can be rendered using ul li element or div element.



Create a simple Radial Menu

The following steps guide you to add a **Radial Menu** control.

Create a MVC Project and add the necessary DLL and scripts with the help of the given MVC Getting Started documentation.

Add the following code snippet in corresponding View page to render the RadialMenu.

You can set the images for each item by giving the image **url** with the **ImageURL** attribute in the RadialMenuItems and text with **Text** attribute. Refer the following code example.Initialize **Radial Menu** control and set its target content as follows.

RAZOR

```
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").TargetElementId("radialtarget1").Items(items
=>
{
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/copy.png")).Text("Copy");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/paste.png")).Text("Paste");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/redo.png")).Text("Redo");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/undo.png")).Text("Undo");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
```

Refer to the following code example to add target content to the **Radial Menu**.

HTML

```
<div id="contentDiv">
<div id="radialtarget1" class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<textarea id="rteSample1" rows="10" cols="70" style="height: 440px">
<p>Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it.
The model consists of application data, business rules, logic, and
functions. A view can be any output representation of data, such as a
chart or a diagram.
Multiple views of the same data are possible, such as a bar chart for
management and a tabular view for accountants.
The controller mediates input, converting it to commands for the model or
view.The central ideas behind MVC are code reusable and in addition to
dividing the application into three kinds of components, the MVC design
defines the interactions between them.</p>
<p>A controller can send commands to its associated view to change the
view's presentation of the model (e.g., by scrolling through a document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).</p>
<p>A model notifies its associated views and controllers when there has
been a change in its state. This notification allows the views to produce
updated output, and the controllers to change the available set of
commands. A passive implementation of MVC omits these notifications,
because the application does not require them or the software platform
does not support them.</p>
<p>A view requests from the model the information that it needs to
generate an output representation to the user.</p>
</textarea>
@{
Html.EJ().RTE("rteSample1").Width("100%").MinWidth("10px").ShowContextMen
u(false).ClientSideEvents(clientSideEvent =>
clientSideEvent.Create("rteCreate").Change("rteChange").Select("radialSho
w")).ShowToolbar(false).Render();
}
</div>
</div>
</div>
</div>
```

Add the following styles in your code.

CSS

```
<style type="text/css" class="cssStyles">
.e-radialmenu .e-radial
{
background-image:
url("mvc.syncfusion.com/demos/web/Images/RadialMenu/settings.png");
```

Copyright © 2001 - 2020 Syncfusion Inc.

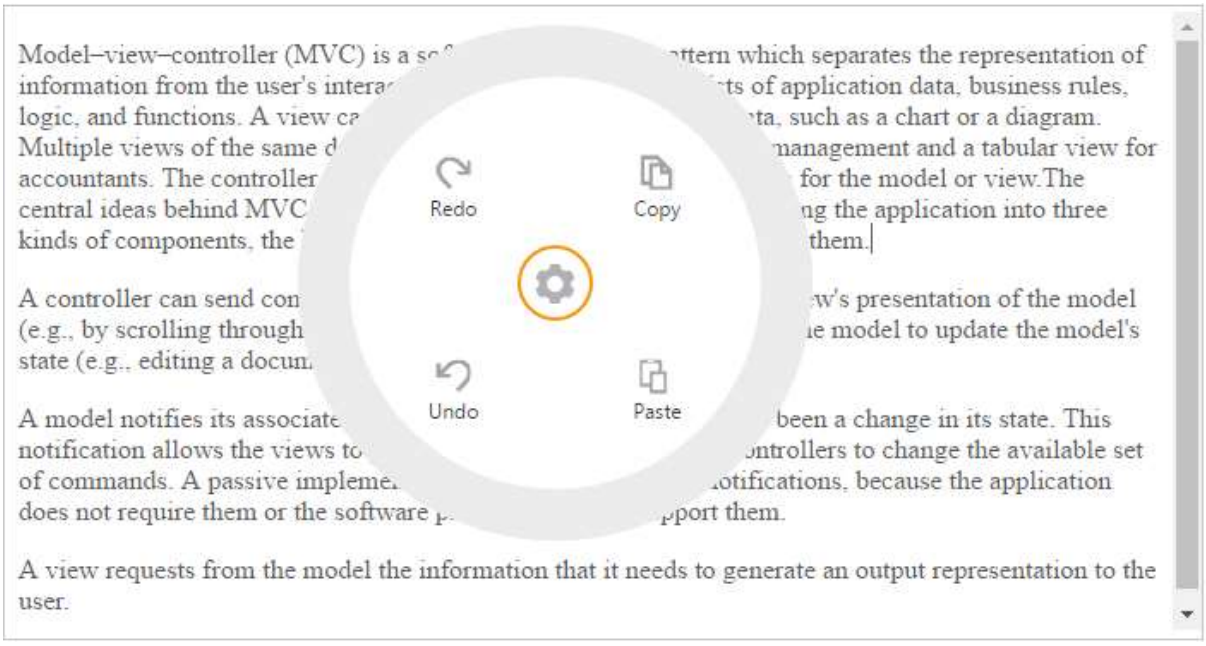
```
}
#rteSample1
{
width: 100%;
}
</style>
```

You can display the **Radial Menu** by performing desired action on the target content while selecting the text inside the target. Therefore, call the **radialShow** method in the select action of the content. Refer the following code example and add it to the script.

JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialRarget1"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

Run the above code and select any text inside the target. The settings icon is displayed. Click that icon to render the following output.



RadialMenu Dependencies

The external script dependencies of the RadialMenu component are,

- jQuery 1.7.1 and later versions.

And the internal script dependencies of the RadialMenu widget are:

FileÂ	Description / Usage
ej.core.min.js	Must be referred always before using all the JS controls.
ej.data.min.js	Used to handle data operation and should be used while binding data to JS controls.
ej.touch.min.js	For providing touch support
ej.radialmenu.min.js	This is main source file specific for rendering RadialMenu

Dimension

You can customize **Radial Menu** dimension by using **Radius** and **Position** properties.

Radius

You can customize the **Radial Menu** size by using the **Radius** property. By default, the **Radial Menu** Radius is set as 150px. Refer to the following code example.

RAZOR

```
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").TargetElementId("radialtarget1").Items(items
=>
{
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/font.png")).Text("Bold");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/f1.png")).Text("Italic");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/align.png")).Text("Align");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/sort.png")).Text("Sort");
}).Radius(150).Render();
```

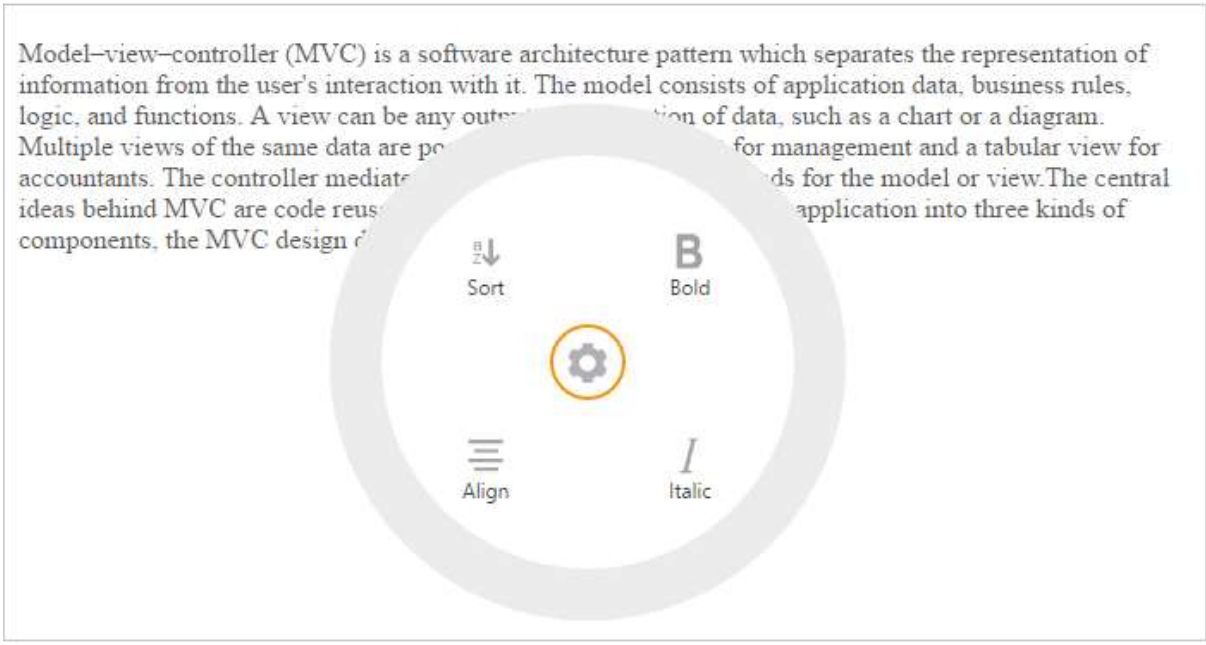
Add the following script in your code.

JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialRarget2"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot illustrates the **Radial Menu** while clicking on the settings icon.



Position

To display the **Radial Menu** in the web page in a specific area, we can use the **Position** property. By default, the **Radial Menu** Position is set as null.

Refer to the following code example.

RAZOR

```
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").TargetElementId("radialtarget1").Items(items
=>
{
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/copy.png")).Text("Copy");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/paste.png")).Text("Paste");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/redo.png")).Text("Redo");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/undo.png")).Text("Undo");
}).Radius(150).Render();
```

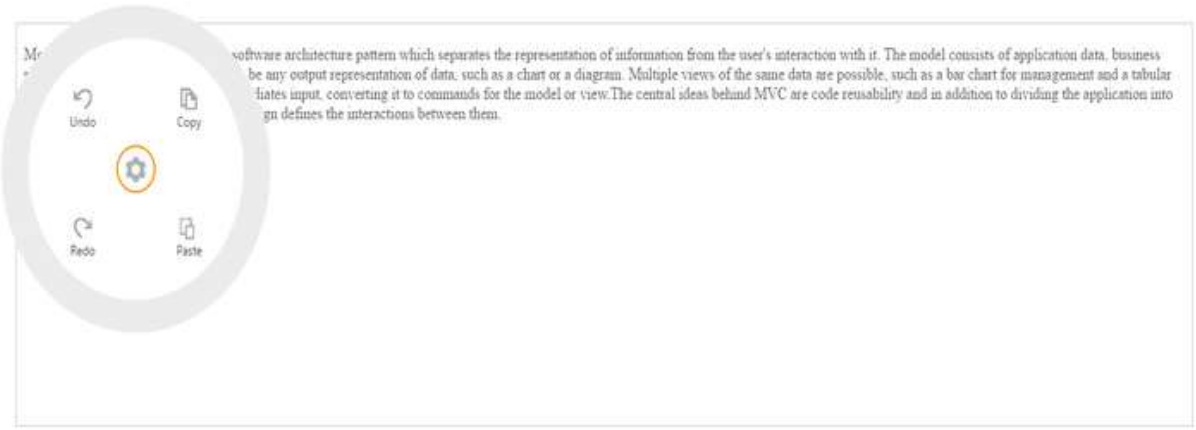
Add the following script in your code.

JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialTarget2"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
```

```
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

The following screenshot illustrates the output while selecting the text in the page.



Item Customization

You can customize individual **Radial Menu** items by using the items properties.

Adding image and text to RadialMenu items

The **ImageUrl** property specifies the URL of the image for the items. **Text** attribute is used to specify the item text. Refer to the following code example.

You can add the page content in RTE control by referring this section.

RAZOR

```
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").TargetElementId("radialTarget1").Items(items
=>
{
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/copy.png")).Text("Copy");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/paste.png")).Text("Paste");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/redo.png")).Text("Redo");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/undo.png")).Text("Undo");
})
```

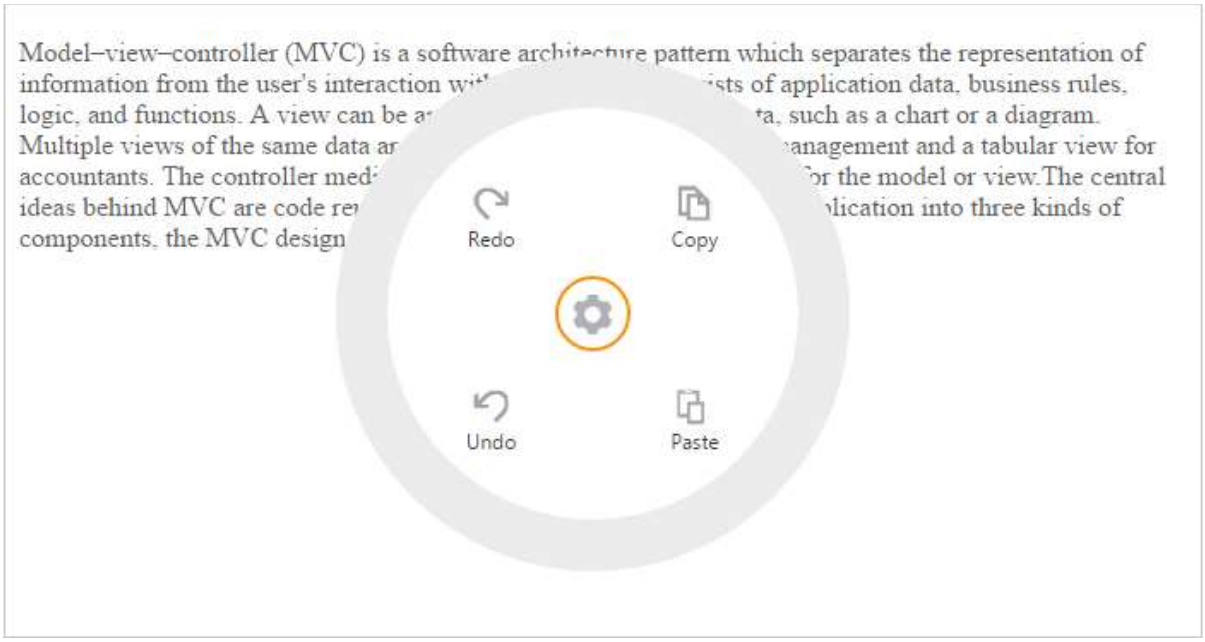
You can display the **Radial Menu** by performing desired action on the target content while selecting the text inside the target. Therefore, call the **radialShow** method in the select action of the content. Refer the following code example and add it to the script.

JAVASCRIPT

Copyright © 2001 - 2020 Syncfusion Inc.

```
function radialShow(e) {
var target = $("#radialtarget2"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

The following screenshot illustrates the output.



Adding events to Radial Menu items

You can specify the click event to corresponding image/text of **Radial Menu** items for performing some specific action. You need to handle the click event in script manually for each item click.

You can add the page content in RTE control by referring to this section

RAZOR

```
@{
Html.EJ().RadialMenu("nestedRadialMenu").ImageClass("imageClass").BackIma
geClass("backImageClass").TargetElementId("radialtarget2").Items(items =>
{
items.Add().Text("Copy").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/copy.png")).Click("Bold");
items.Add().Text("Font").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/font.png"));
items.Add().Text("List").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/list.png")).Children(children =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/list.png")).Text("List");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/l5.png")).Text("List");
});
items.Add().Text("Bold").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/font.png")).Click("bold").Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/f1.png")).Text("Italic").Click("italic");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/font.png")).Text("Bold").Click("bold");
});
items.Add().Text("Paste).ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/paste.png"));
items.Add().Enabled(false).ImageURL(Url.Content("mvc.syncfusion.com/demos
/web/Images/RadialMenu/undo.png")).Children(children =>
{
children.Add().Enabled(false).ImageURL(Url.Content("mvc.syncfusion.com/de
mos/web/Images/RadialMenu/undo.png"));
children.Add().Enabled(false).ImageURL(Url.Content("mvc.syncfusion.com/de
mos/web/Images/RadialMenu/redo.png"));
});
items.Add().Text("Alignment").ImageURL(Url.Content("mvc.syncfusion.com/de
mos/web/Images/RadialMenu/align.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/a1.png")).Text("Left");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/a2.png")).Text("Right");
});
}).Render();
}
```

Add the following script in your code.

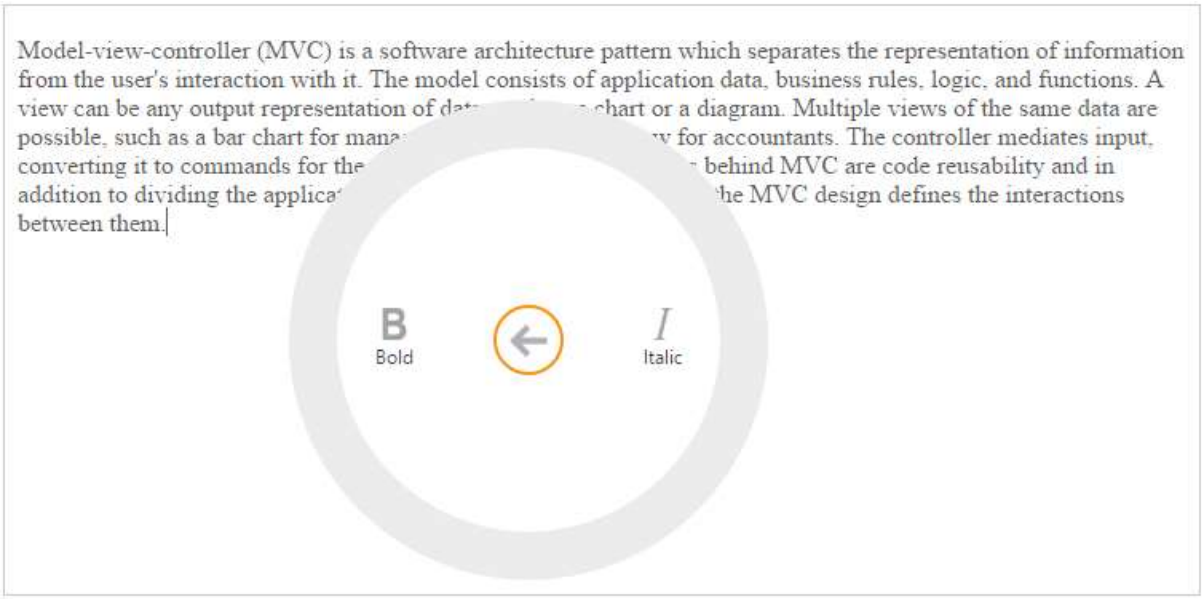
JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialtarget2"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
function italic(e) {
rteObj.executeCommand("italic");
}
function bold(e) {
rteObj.executeCommand("bold");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
function copy(e) {
rteObj.executeCommand("copy");
}
```

The following screenshot illustrates the output.



Badge Customization in Radial Menu Items

You can specify set badges for the items by using badge settings in radial menu. You can set value for badge and enable badge with default value.

The **Enabled** property to enable or disable badges. **Value** attribute is to set the badge value.

RAZOR

```
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").TargetElementId("radialRarget1").Items(items
=>
{
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/copy.png")).Text("Copy").Badge(badge =>
badge.Enabled(true).Value(2));
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/paste.png")).Text("Paste");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/redo.png")).Text("Redo");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/undo.png")).Text("Undo");
})
```

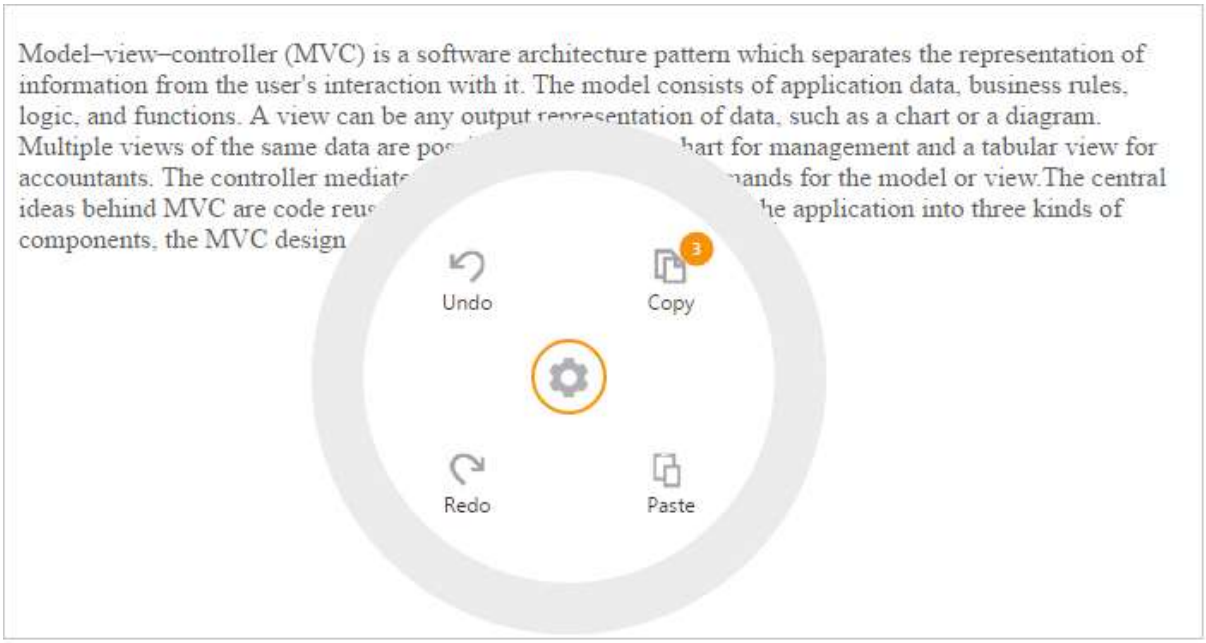
Add the following script in your code.

JAVASCRIPT

```
function radialShow(e) {
```

```
var target = $("#radialtarget2"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

The following screenshot illustrates the output.



Slider Customization in Radial Menu Items

You can customize the Radial Menu with slider settings.The **data-ej-type** property specifies the type of radial menu item, where you can set the type for RadialMenu item.

You can customize the **Radial Menu** slider settings by using the **Ticks**, **StrokeWidth** and **LabelSpace** properties. The **Ticks** property specifies the slider ticks for radial menu item.**StrokeWidth** attribute is used to specify the slider's stroke width value.**LabelSpace** attribute is used to specify the value of slider label space.

Refer to the following code example.

You can add SliderSettings in radial menu items by referring this section.

RAZOR

```
@{
Html.EJ().RadialMenu("nestedRadialMenu").ImageClass("imageClass").BackIma
geClass("backImageClass").TargetElementId("radialTarget2").Items(items =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
items.Add().Text("Copy").ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/copy.png")).Click("Bold");
items.Add().Text("Font").ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/font.png").Type(MenuItemType.Slider).SliderSettings(slider => slider.StrokeWidth(1).Ticks(new List<double> { 0, 2, 4, 6, 8, 10, 12, 14 })));
items.Add().Text("List").ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/list.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/list.png")).Text("List");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/l5.png")).Text("List");
});
items.Add().Text("Bold").ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/font.png")).Click("bold").Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/f1.png")).Text("Italic").Click("italic");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/font.png")).Text("Bold").Click("bold");
});
items.Add().Text("Paste).ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/paste.png"));
items.Add().Enabled(false).ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/undo.png")).Children(children =>
{
children.Add().Enabled(false).ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/undo.png"));
children.Add().Enabled(false).ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/redo.png"));
});
items.Add().Text("Alignment").ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/align.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/a1.png")).Text("Left");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/RadialMenu/a2.png")).Text("Right");
});
}).Render();
}
```

Add the following script in your code.

JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialtarget2"), radialRadius = 150, radialDiameter = 2 * radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX = target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() - radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

The following screenshot illustrates the output.

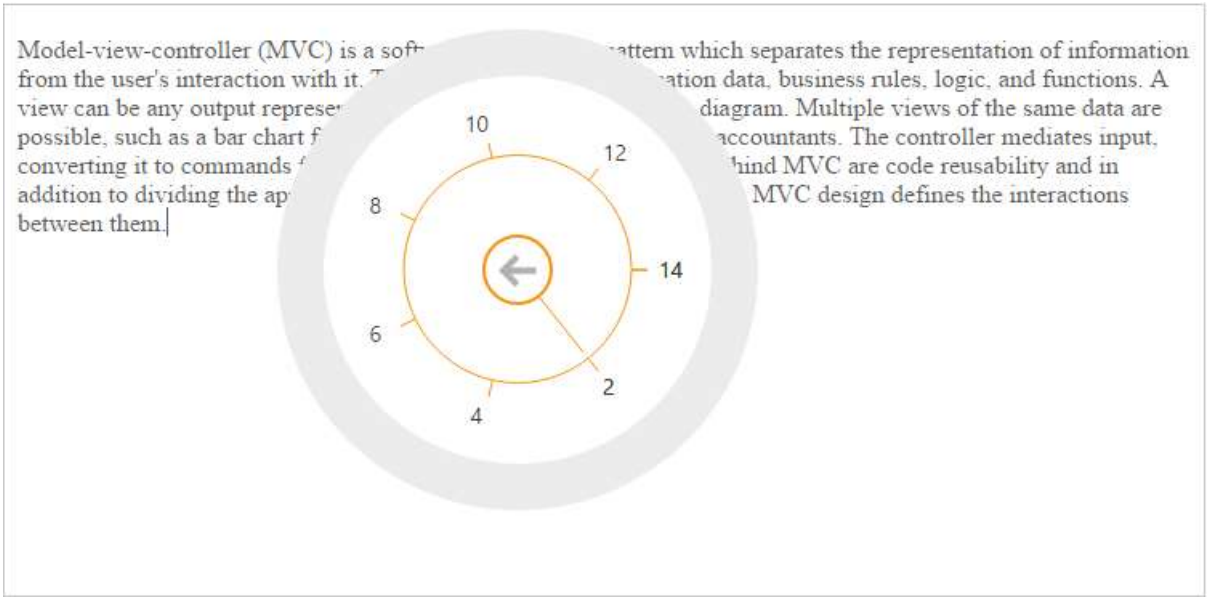


Image Customization

You can customize the **Radial Menu's** Center and Back images by using the **ImageClass** and **BackImageClass** properties. Every menu item can be added with image using **url** property. By using this **ImageClass** attribute, you can customize the **Radial Menu** center image.

Sub-Items are also supported in the **Radial Menu**. To navigate Sub-Items, click the arrows in the outer ring and it displays the corresponding sub-items group. Clicking the center button when a sub-items group is shown, displays the items on the previous level. Nested **Radial Menu** has the second level back button. In this case, you can use the **BackImageClass** attribute to change your second level back button. **BackImageClass** is used to customize the **nestedRadialmenu** back image. Refer to the following code example.

You can add the page content with text-area by referring to this section.

RAZOR

```
@{
Html.EJ().RadialMenu("nestedRadialMenu").ImageClass("image-
class").BackImageClass("back-
image").TargetElementId("radialTarget2").Items(items =>
{
items.Add().Text("Copy").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/copy.png"));
items.Add().Text("Font").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/font.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/f1.png")).Text("Italic");
```

```
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/font.png")).Text("Bold");
});
items.Add().Text("Table").ImageURL(Url.Content("mvc.syncfusion.com/demos/
web/Images/RadialMenu/table.png"));
items.Add().Text("List").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/list.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/l1.png")).Text("List");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/l2.png")).Text("List");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/l3.png")).Text("List");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/l4.png")).Text("List");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/l5.png")).Text("List");
});
items.Add().Text("Paste").ImageURL(Url.Content("mvc.syncfusion.com/demos/
web/Images/RadialMenu/paste.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/c1.png")).Text("Paste");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/c2.png")).Text("Paste");
});
items.Add().Text("Sort").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/sort.png")).Children(children =>
{
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/s1.png")).Text("Sort");
children.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/
RadialMenu/s2.png")).Text("Sort");
});
items.Add().Text("Alignment").ImageURL(Url.Content("mvc.syncfusion.com/de
mos/web/Images/RadialMenu/align.png")).Children(children =>
{
children.Add().Text("Left").ImageURL(Url.Content("mvc.syncfusion.com/demo
s/web/Images/RadialMenu/a1.png")).Click("left");
children.Add().Text("Right").ImageURL(Url.Content("mvc.syncfusion.com/dem
os/web/Images/RadialMenu/a2.png")).Click("right");
});
items.Add().Text("Draw").ImageURL(Url.Content("mvc.syncfusion.com/demos/w
eb/Images/RadialMenu/draw.png"));
}).Render();
}
```

Add the following script in your code.

JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialTarget2"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
```

Copyright © 2001 - 2020 Syncfusion Inc.

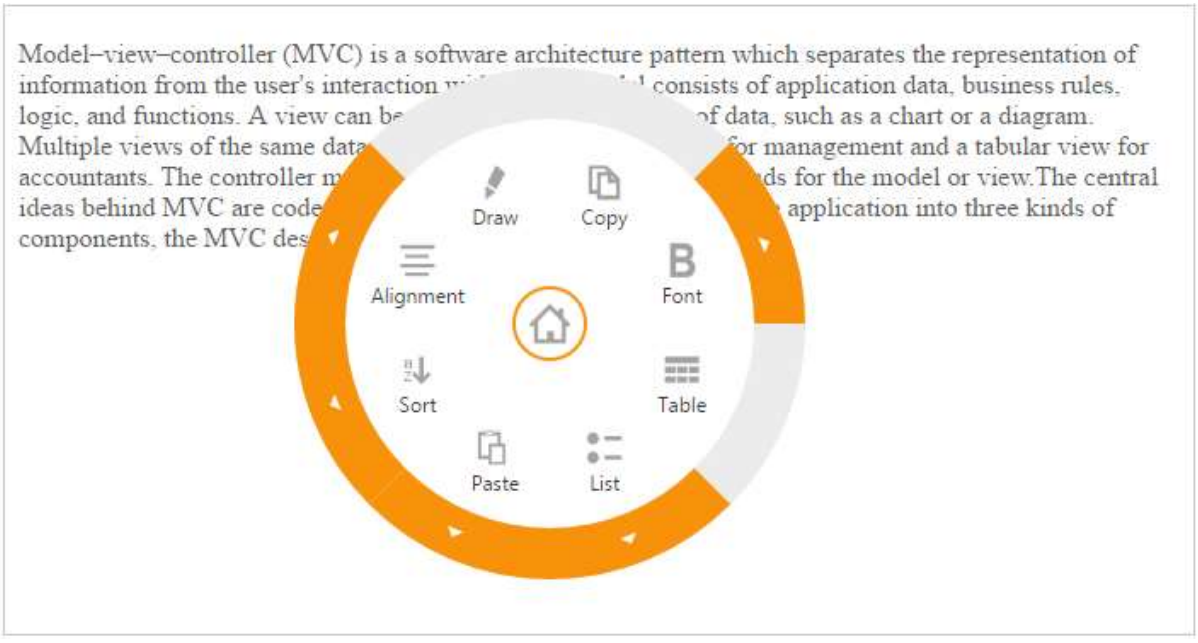
```
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

Add the following styles in your code.

CSS

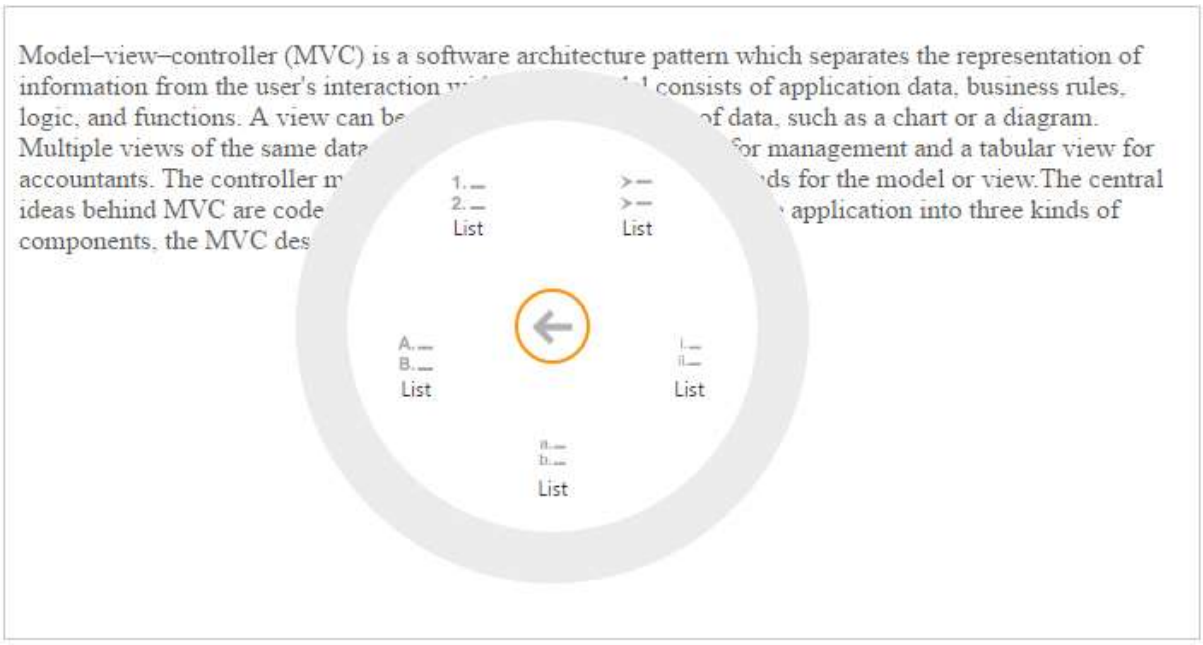
```
<style type="text/css" class="cssStyles">
.e-radialmenu .image-class
{
background-image:
url("mvc.syncfusion.com/demos/web/Images/RadialMenu/main.png");
}
.e-radialmenu .back-image
{
background-image:
url("mvc.syncfusion.com/demos/web/Images/RadialMenu/Back_button.png");
}
</style>
```

The following screenshot illustrates the output.



Radial Menu - Image Customization – Main menu

When you click the arrow, it navigates to the child item as illustrated in the following screenshot.



Radial Menu- Image Customization – Child menu

Appearance and Styling

You can customize RadialMenu control style and the appearance by using available themes or cssClass property.

Theme

In order to apply styles to the RadialMenu control, refer these 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.web.all.min.css file, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.web.all.min.css is the combination of these two.

By default, there are 13 theme support available for RadialMenu component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap

CSS Class

RadialMenu component's appearance can only be customized using its CSS classes. Define CSS class, as per requirement and assign the class name to cssClass property.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure RadialMenu using CSS class

Define CSS class to customize the RadialMenu control.

CSS

```
<style type="text/css">
/* Customize the radialmenu */
.e-radialmenu .e-default, .e-radialmenu .e-outerdefault.customCss
{
fill:#f00;
}
</style>
```

In the HTML page, define the li items for RadialMenu component.

RAZOR

```
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").CssClass("customCss").TargetElementId("radialtarget1").Items(ite
ms
=>
{
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/copy.png")).Text("Copy");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/paste.png")).Text("Paste");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/redo.png")).Text("Redo");
items.Add().ImageURL(Url.Content("mvc.syncfusion.com/demos/web/Images/Rad
ialMenu/undo.png")).Text("Undo");
})
```

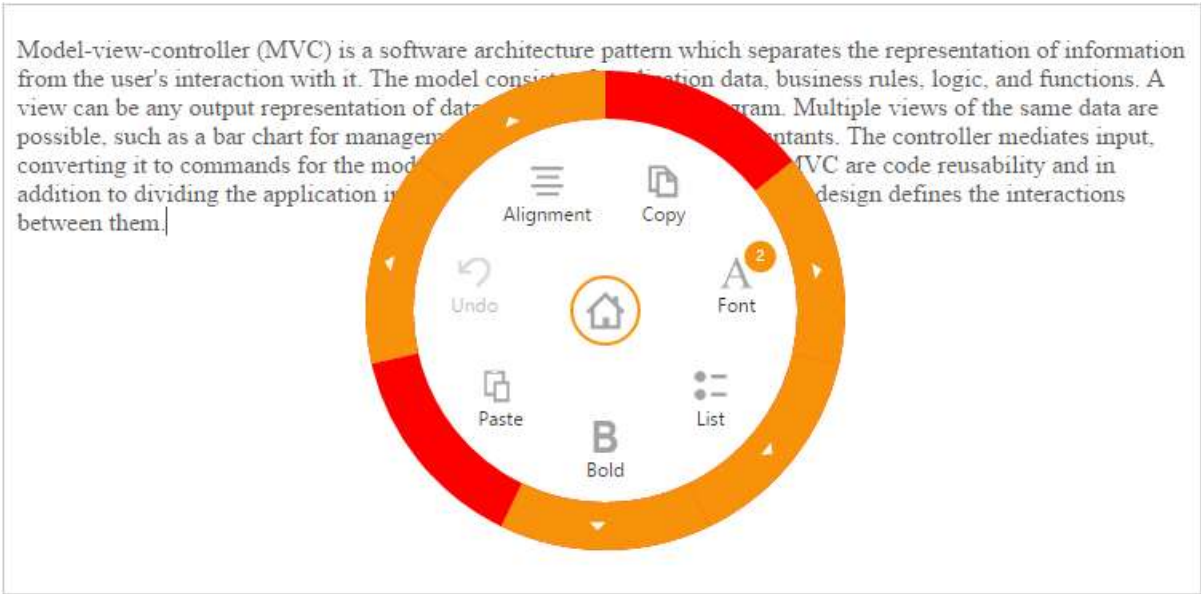
Customize **Radial Menu** control with cssClass in script as follows.

JAVASCRIPT

```
function radialShow(e) {
var target = $("#radialtarget1"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = target.offset().top + e.event.clientY, iframeX =
target.offset().left + e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
}
```

Output of RadialMenu configured based on CSS class is as follow,

Copyright © 2001 - 2020 Syncfusion Inc.



Template Support

Template support for RadialMenu items will allow you to use any type of \<svg\> Permitted tags inside our template. Here for example, using this template support you can use the SVG icons in Radial Menu instead of image tags. To use SVG icons in RadialMenu, you need to use PrependTo property.

Add SVG to item Icon

Using SVG icon will optimize the icons quality and to reduce space occupation by normal images and SVG images are Scalable and Zoomable. Define the text element for SVG with x and y position and code for rendering the font icons. Assign the SVG element ID to PrependTo property.

Please refer the following code to render the SVG icon in RadialMenu,

RAZOR

```
<div id="contentDiv">
<div id="radialTarget1" class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<textarea id="rteSample1" rows="10" cols="70" style="height: 440px">
<p>Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it.
The model consists of application data, business rules, logic, and
functions. A view can be any output representation of data, such as a
chart or a diagram.
Multiple views of the same data are possible, such as a bar chart for
management and a tabular view for accountants.
The controller mediates input, converting it to commands for the model or
view.The central ideas behind MVC are code reusable and in addition to
dividing the application into three kinds of components, the MVC design
defines the interactions between them.</p>
<p>A controller can send commands to its associated view to change the
view's presentation of the model (e.g., by scrolling through a document).
It can also send commands to the model to update the model's state (e.g.,
editing a document).</p>
```

```
<p>A model notifies its associated views and controllers when there has
been a change in its state. This notification allows the views to produce
updated output, and the controllers to change the available set of
commands. A passive implementation of MVC omits these notifications,
because the application does not require them or the software platform
does not support them.</p>
<p>A view requests from the model the information that it needs to
generate an output representation to the user.</p>
</textarea>
@{
Html.EJ().RTE("rteSample1").Width("100%").MinWidth("10px").ShowContextMen
u(false).ClientSideEvents(clientSideEvent =>
clientSideEvent.Create("rteCreate").Change("rteChange").Select("radialSho
w")).ShowToolbar(false).Render();
}
</div>
</div>
</div>
@Html.EJ().RadialMenu("defaultRadialMenu").ImageClass("e-
radial").TargetElementId("radialTarget1").Items(items
=>
{
items.Add().PrependTo("#template1").Text("Bold").Click("bold");
items.Add().PrependTo("#template2").Text("Italic").Click("italic");
items.Add().PrependTo("#template3").Text("Redo").Click("redo").Enabled(fa
lse);
items.Add().PrependTo("#template4").Text("Undo").Click("undo").Enabled(fa
lse);
})
</div>
<svg id="template1" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="210" y="100">&#xe636;</text>
</svg>
<svg id="template2" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="210" y="218">&#xe635;</text>
</svg>
<svg id="template3" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="90" y="215">&#xe606;</text>
</svg>
<svg id="template4" style="display: none"
xmlns="http://www.w3.org/2000/svg">
<text x="93" y="100">&#xe607;</text>
</svg>
```

Add the following code in your script section.

JAVASCRIPT

```
<script type="text/javascript">
var rteObj, rteElement = $("#rteSample1"), radialElement =
$('#defaultRadialMenu'), action = 0, forRedo = 0;
$(function () {
if (!(ej.browserInfo().name == "msie" && ej.browserInfo().version < 9)) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#radialTarget1").parent().css("position", "relative");
}
else {
$("#contentDiv").html("Radial Menu is only supported from Internet
Explorer Versioned 9 and above.").css({ "font-size": "20px", "color":
"red" });
}
$(window).resize(function(){
if(ej.isMobile() && ej.isPortrait())
$('#defaultRadialMenu').css({ "left": 25 })
});
});
function rteCreate(e) {
rteObj = this;
}
function radialShow(e) {
var target = $("#radialTarget1"), radialRadius = 150, radialDiameter = 2
* radialRadius,
// To get Iframe positions
iframeY = e.event.clientY, iframeX = e.event.clientX,
// To set Radial Menu position within target
x = iframeX > target.width() - radialRadius ? target.width() -
radialDiameter : (iframeX > radialRadius ? iframeX - radialRadius : 0),
y = iframeY > target.height() - radialRadius ? target.height() -
radialDiameter : (iframeY > radialRadius ? iframeY - radialRadius : 0);
radialElement.ejRadialMenu("setPosition", x, y);
$('iframe').contents().find('body').blur();
}
function rteChange(e) {
radialElement.ejRadialMenu("enableItem", "Undo");
}
function bold(e) {
rteObj.executeCommand("bold");
data = rteObj._getSelectedHtmlString() ? true : false;
if (data) action += 1;
forRedo = action;
}
function italic(e) {
rteObj.executeCommand("italic");
data = rteObj._getSelectedHtmlString() ? true : false;
if (data) action += 1;
forRedo = action;
}
function undo(e) {
rteObj.executeCommand("undo");
action -= 1;
if (action == 0)
radialElement.ejRadialMenu("disableItem", "Undo");
radialElement.ejRadialMenu("enableItem", "Redo");
}
function redo(e) {
rteObj.executeCommand("redo");
action += 1;
if (forRedo == action)
radialElement.ejRadialMenu("disableItem", "Redo");
radialElement.ejRadialMenu("enableItem", "Undo");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

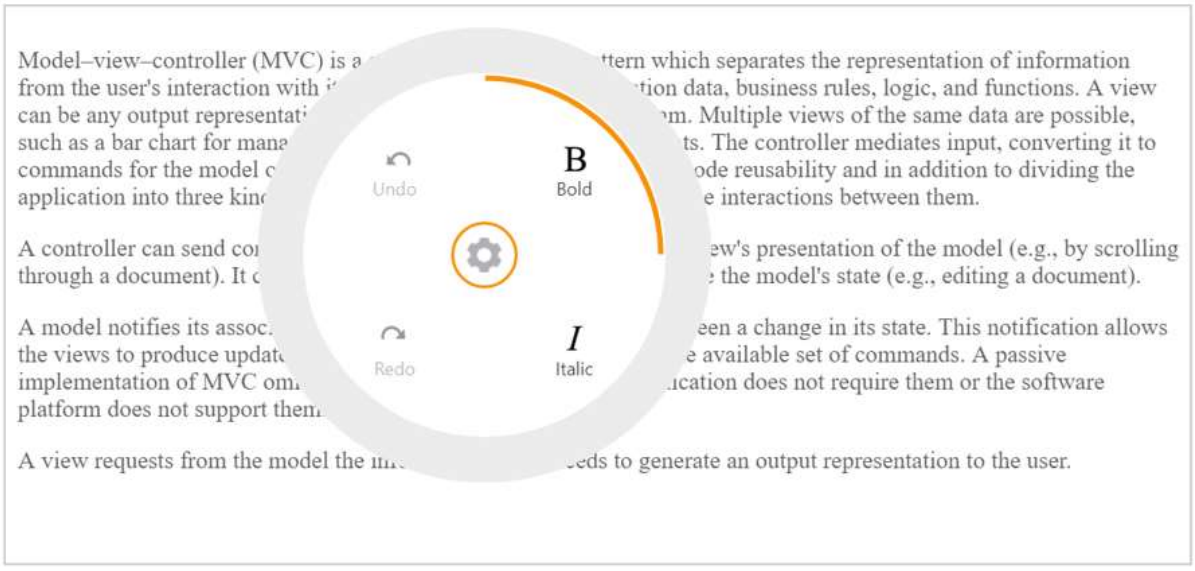
Add the following code in your style section,

CSS

```
<style type="text/css" class="cssStyles">
.e-radialmenu .e-radial {
background-image: url("../Images/RadialMenu/settings.png");
}
#rteSample1 {
width: 100%;
}
font-face {
font-family: 'ej-webfont';
font-weight: normal;
font-style: normal;
font-size: 20px;
}
.e-radialmenu .e-abs.e-radialshow, .e-radialmenu .e-abs.e-scaleshow {
/* use !important to prevent issues with browser extensions that change
fonts */
font-family: 'ej-webfont' !important;
speak: none;
font-size: 20px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
/* Better Font Rendering =========== */
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
</style>
```

The following screenshot illustrates the output,

Copyright © 2001 - 2020 Syncfusion Inc.



Note: This is the example sample for SVG icon support for Radial Menu.Like wise you can add any SVG element to it, but you need to customize and position the element individually.

How to

Render RadialMenu from Code behind

RadialMenu can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of RadialMenu properties in the controller.

C#

```
//Namespace to get the JavaScript (RadialMenu) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class RadialmenuController: Controller
{
public ActionResult RadialmenuFeatures()
{
//Initializing the RadialMenu model
RadialMenuProperties RadialmenuObj = new RadialMenuProperties();
//Initializing the items and other properties
List<RadialMenuBaseItem> obj = new List<RadialMenuBaseItem> ();
obj.Add(new RadialMenuBaseItem(){ Text = "Copy" , ImageURL =
"http://mvc.syncfusion.com/demos/web/Images/RadialMenu/copy.png"});
obj.Add(new RadialMenuBaseItem() { Text = "Paste", ImageURL =
"http://mvc.syncfusion.com/demos/web/Images/RadialMenu/paste.png" });
obj.Add(new RadialMenuBaseItem() { Text = "Undo", ImageURL =
"http://mvc.syncfusion.com/demos/web/Images/RadialMenu/undo.png" });
obj.Add(new RadialMenuBaseItem() { Text = "Redo", ImageURL =
"http://mvc.syncfusion.com/demos/web/Images/RadialMenu/redo.png" });
RadialmenuObj.Items = obj;
RadialmenuObj.ImageClass = "e-radial";
RadialmenuObj.TargetElementId = "radialtarget1";
//Passing Radialmenu properties using the ViewData
ViewData["RadialMenuModel"] = RadialmenuObj;
```

```
return View();
}
}
```

Binding the RadialMenu properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
<div id="radialtarget11" class="content-container-fluid">
</div>
@{
Html.EJ().RadialMenu("RadialMenu1",
(Syncfusion.JavaScript.Models.RadialMenuProperties)ViewData["RadialMenuMo
del"]).Render();
}
<style type="text/css" class="cssStyles">
.e-radialmenu .e-radial {
background-image:
url("http://mvc.syncfusion.com/demos/web/Images/RadialMenu/settings.png")
;
}
.e-radialmenu.e-displaynone {
display: block;
}
</style>
```

RadialSlider

Overview

The RadialSlider control that provides an optimized interface for selecting a numeric value using touch interface.This allows the user to select a value or range of values in a circular motion.

Key Features

- Angle Support: Provides start and end angle level view of RadialSlider.
- Animation: Offers animation effect for the RadialSlider handle.
- Image Customization: Supports customize the images of inner circle in Radial Slider.
- Dimension: Allows to change the radius and Stroke width of RadialSlider .
- Accuracy: Provides way to select accurate numeric value and customize the display values

Getting Started

Creating your first RadialSlider in MVC application

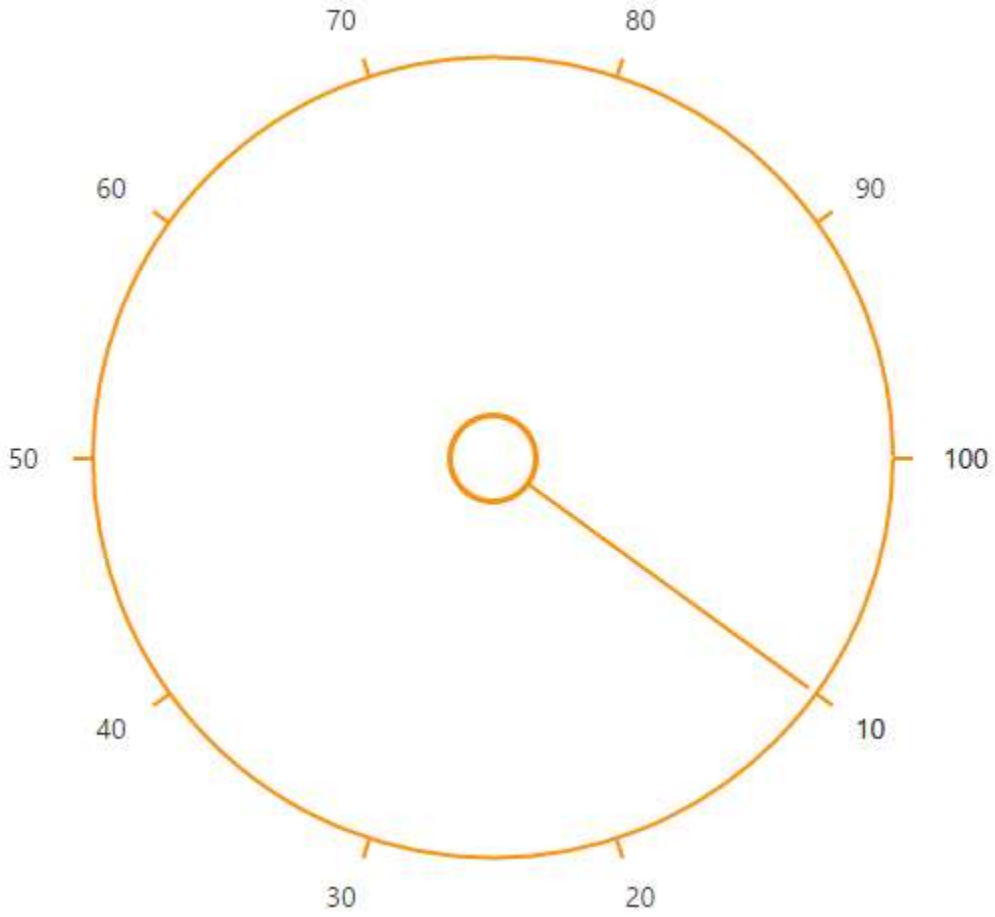
1. Create an MVC Project and add necessary assemblies, scripts and CSS files given in MVC-Getting Started Documentation.
2. Add RadialSlider control using the helper from EJ namespace.

HTML

```
@Html.EJ().RadialSlider("RadialSlider1")
```

Copyright © 2001 - 2020 Syncfusion Inc.

3. Execute the code and get a Default RadialSlider control as below



RadialSlider Dependencies

The external script dependencies of the RadialSlider component are,

- jQuery 1.7.1 and later versions.

And the internal script dependencies of the RadialSlider widget are:

FileÂ	Description / Usage
ej.core.min.js	Must be referred always before using all the JS controls.
ej.touch.min.js	For providing touch support
ej.radialslider.min.js	This is main source file specific for rendering RadialSlider

Dimension

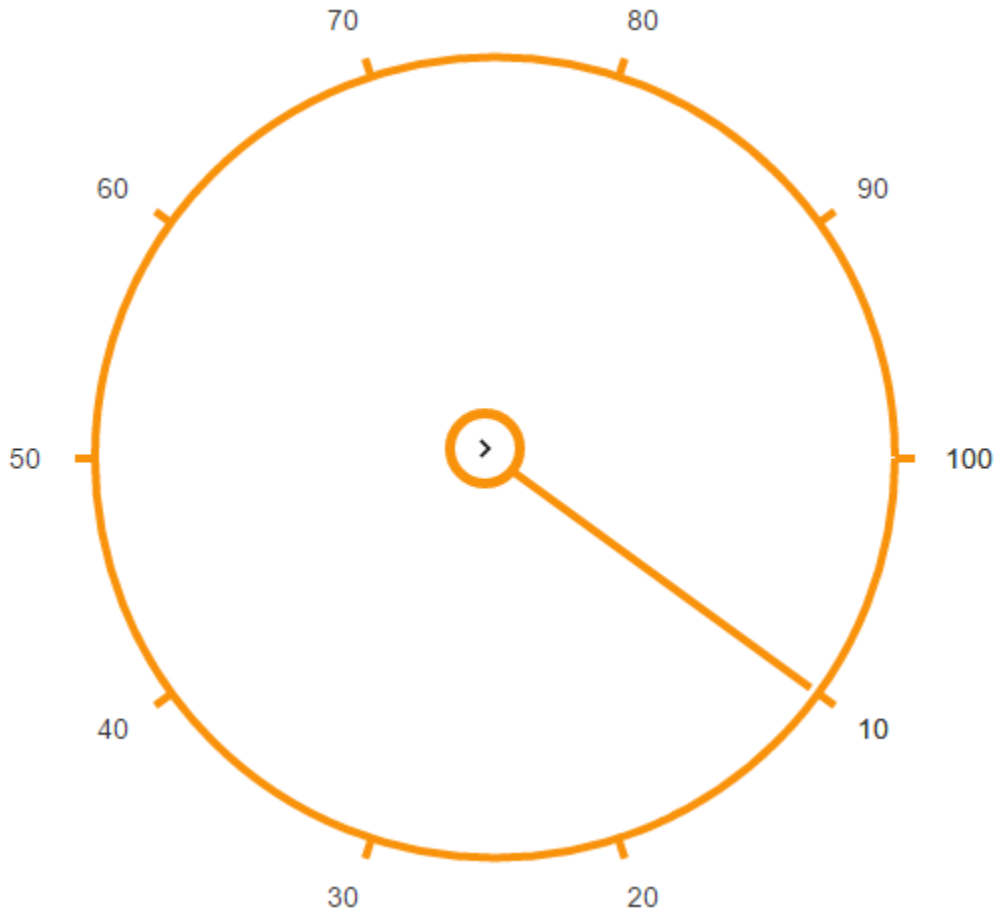
Stroke Width

Radial Slider StrokeWidth property specifies the width of the outline . By default, the **Radial slider** StrokeWidth is set as 2. Refer to the following code example.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").StrokeWidth(4).InnerCircleImageUrl("../
images/radialslider/chevron-right.png")
```

The following screenshot illustrates the output of the above code.



Setting radius

The **RadialSlider** property **Radius** indicates the radius of the RadialSlider's circle ant its allow to change radius value. By default, the **Radial Slider** Radius is set to 200. Refer to the following code example.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").Radius(250)
```

Behavior settings

The following are some properties that enables you to an option to enhance the behavior of **RadialSlider** control.

Show/hide RadialSlider on initial rendering

The **RadialSlider** property **AutoOpen** allow to enable or disable RadialSlider visibility. By default,Value of the property is set as **true** and type of the property is **Boolean**.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").AutoOpen(false)
```

Value Accuracy

The **RadialSlider** property **EnableRoundOff** allow to show the rounded off value when user changes it. By default, value of the property is set as **true**. For accurate values you can set this as false.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").EnableRoundOff(true)
```

Display inline

The **RadialSlider** property **Inline** is used to show the control values inline of the slider. By default, value of the property is set as **false**.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").Inline(true)
```

Modifying Label Space

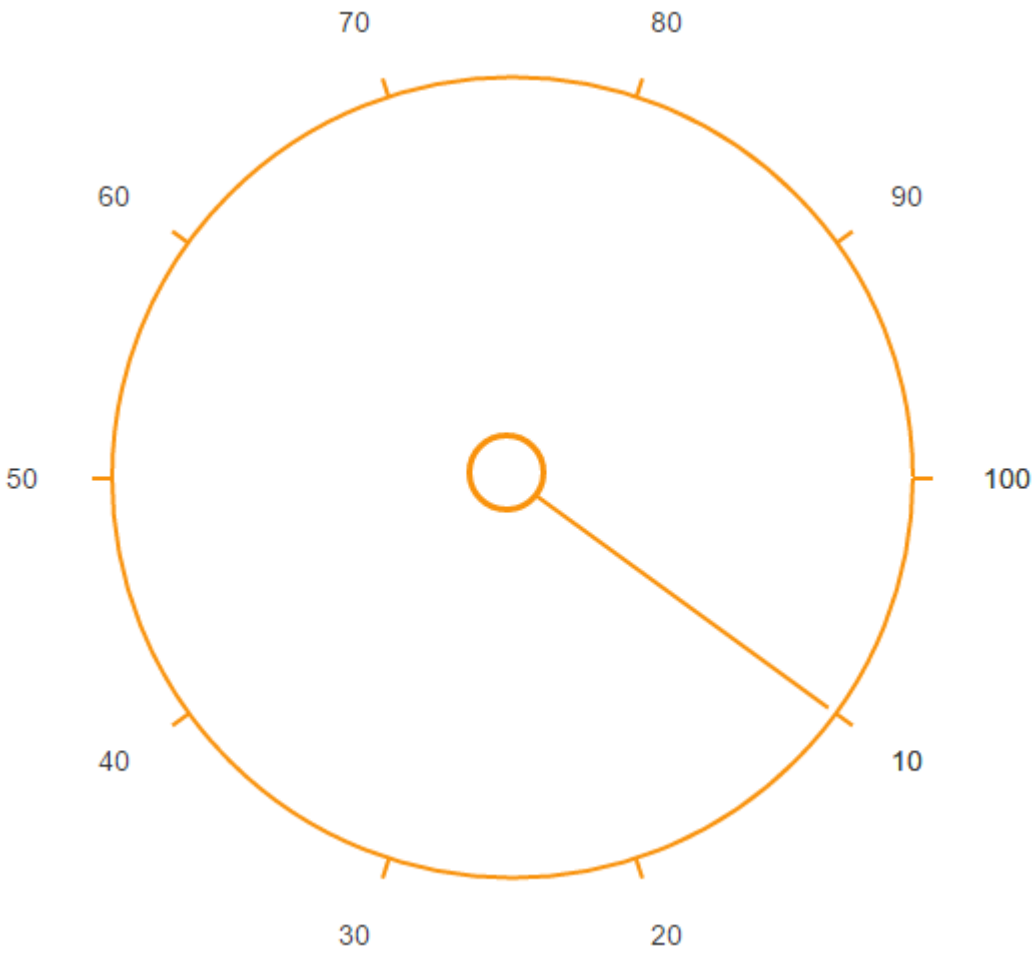
The **RadialSlider** property **LabelSpace** allow to define the space of label in it. By default, the **RadialSlider** labelSpace is set as 30. Refer to the following code example.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").LableSpace(40)
```

The following screenshot shows the output for the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Show/Hide inner circle

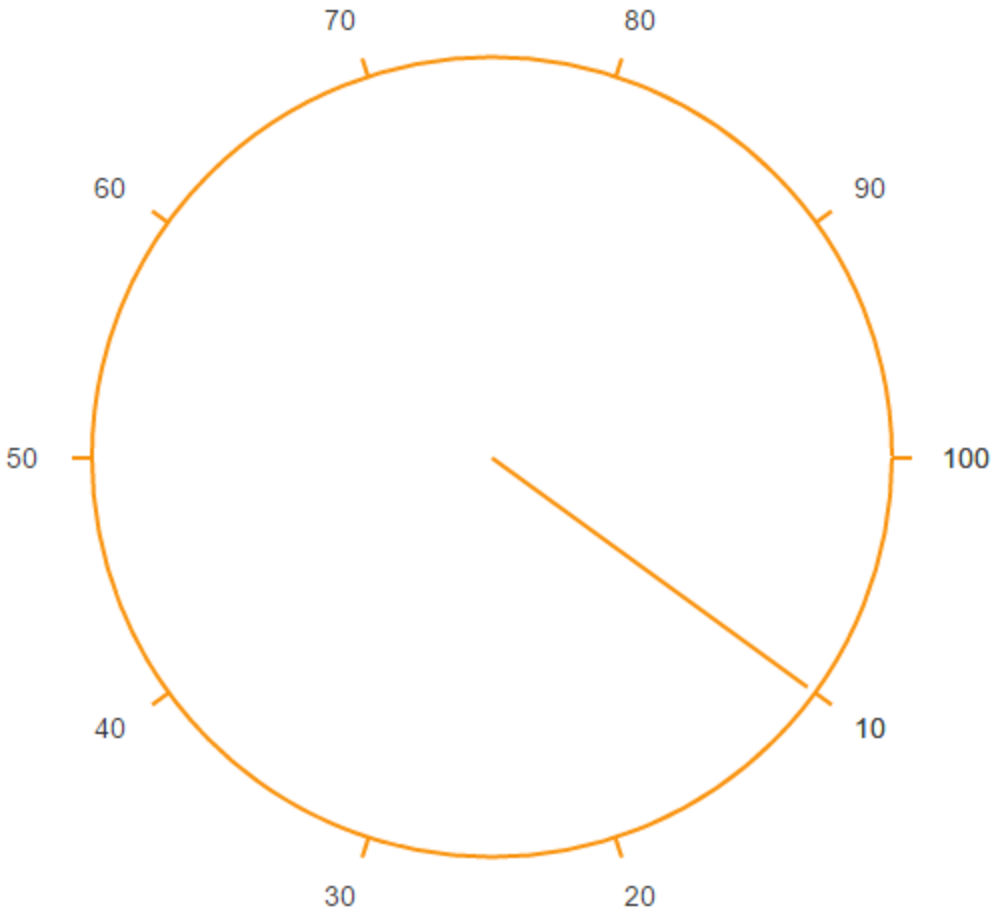
The **RadialSlider** property **ShowInnerCircle** allow to show or hide the inner circle of the **RadialSlider**.By default,Value of the property is set as **true** and type of the property is **Boolean**.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").ShowInnerCircle(false).InnerCircleImage
Url("../images/radialslider/chevron-right.png")
```

The following screenshot shows the output for the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Values customization

The **RadialSlider** property **Ticks** allow to set an array of numeric value which will be displayed in it. By Default **RadialSlider** ticks value is a set to an array of length 11 as following,

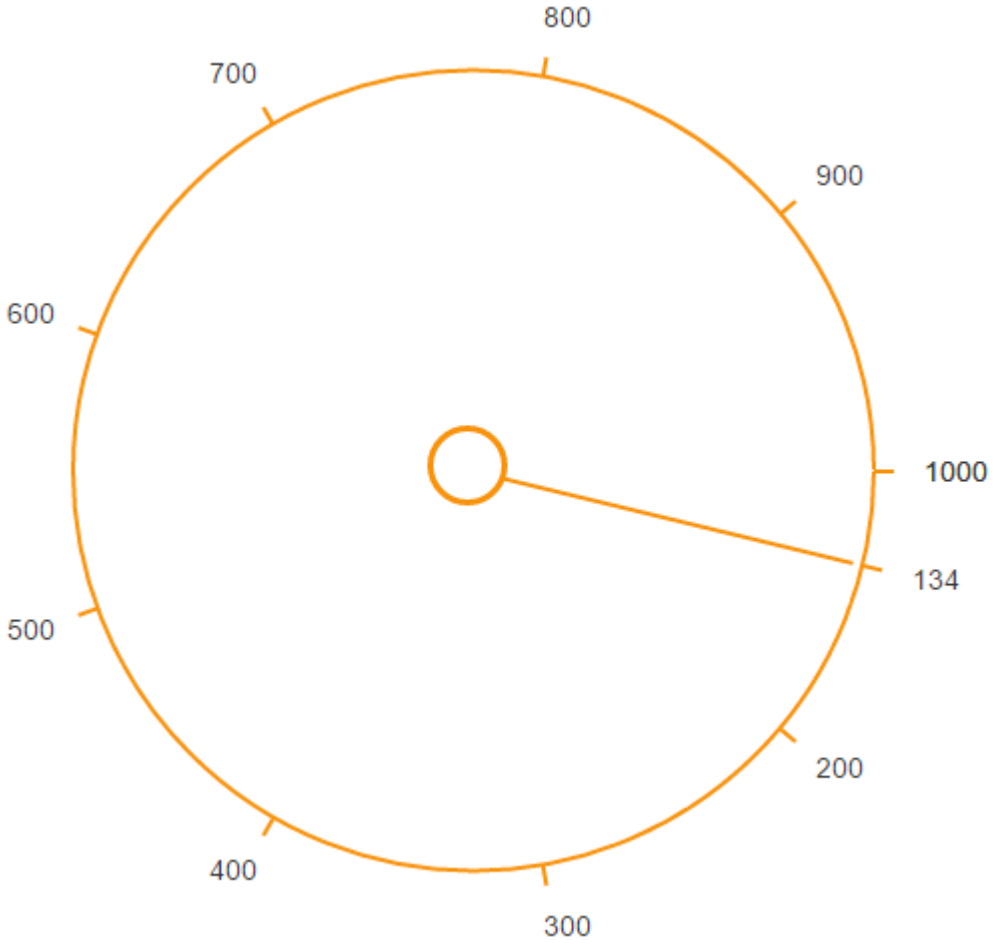
Ticks = new List<double>{ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 }. You can refer the below code for reference.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").Ticks(new List<double>{ 100, 200, 300,
400, 500, 600, 700, 800, 900, 1000 })
```

The following screenshot shows the output for the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Animation Effect

The animation effect for the **RadialSlider** can be enabled or disabled using the **EnableAnimation** property.By default, it is set as **true**, so that the handle movement will be animated.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").EnableAnimation(true).InnerCircleImageU
rl("../images/radialslider/chevron-right.png")
```

Display Angle Settings

Start Angle

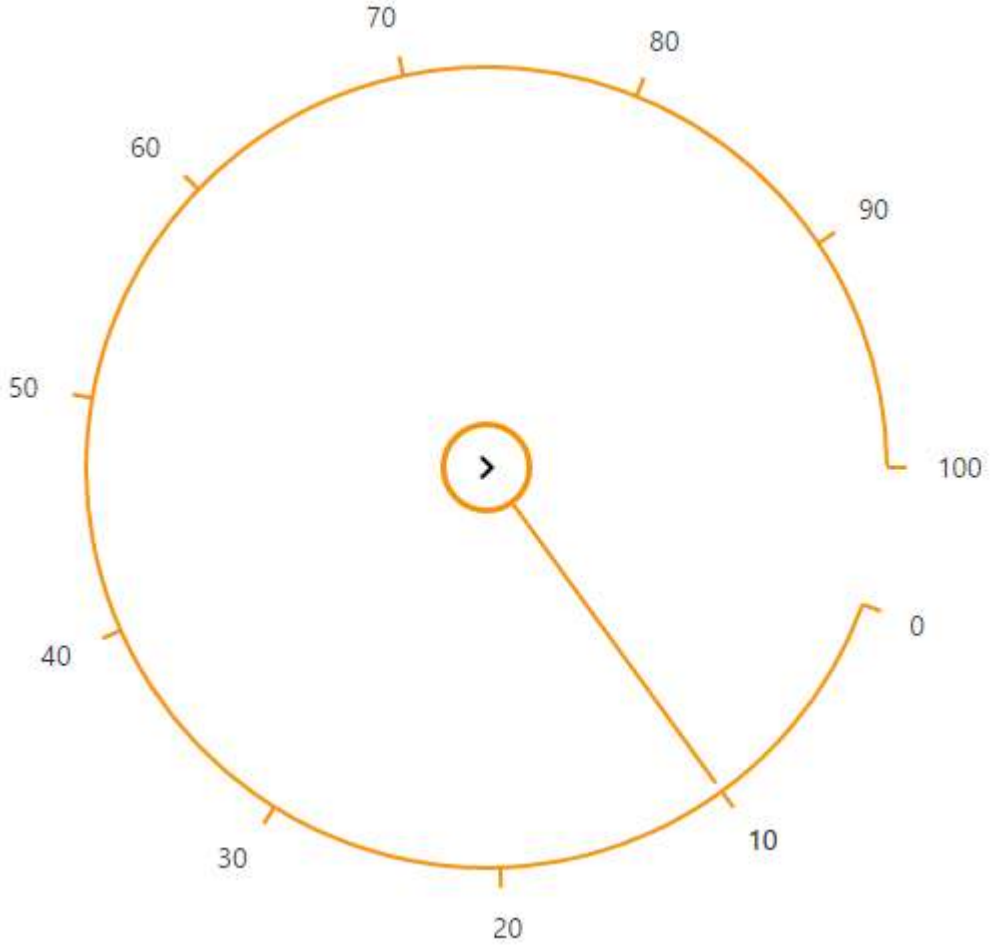
The **RadialSlider** property **StartAngle** allows you to change the startAngle level of the **RadialSlider**. By default, the **Radial Slider** StartAngle is set as 0. Refer to the following code example.

ASPX-CS

```
@Html.EJ().RadialSlider("radialSlider").InnerCircleImageUrl("../images/ra
dialslider/chevron-right.png").StartAngle(20)
```

The following screenshot illustrates the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



End Angle

The **RadialSlider** property **EndAngle** allows you to change the endAngle level of the **RadialSlider**. By default, the **Radial Slider** endAngle is set as 360. Refer to the following code example.

ASPX-CS

```
@Html.EJ().RadialSlider("radialSlider").InnerCircleImageUrl("../images/ra
dialslider/chevron-right.png").EndAngle(300)
```

The following screenshot illustrates the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.

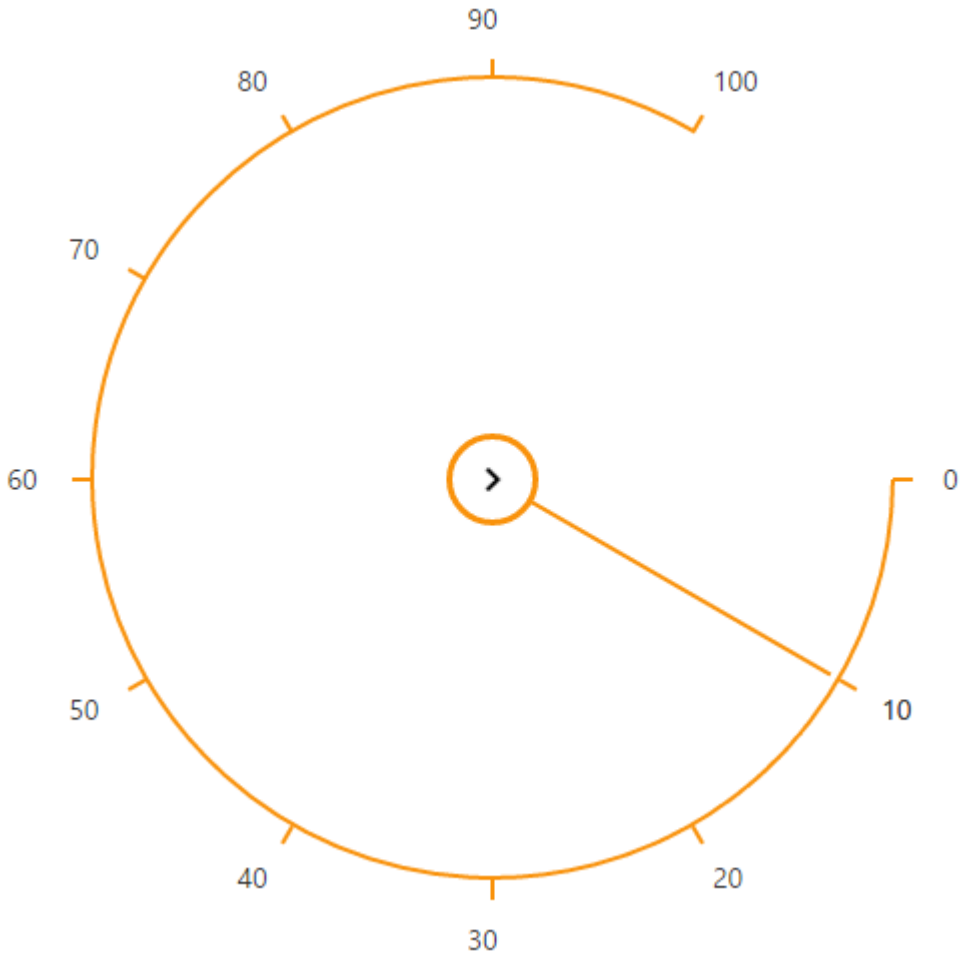


Image customization

Customizing inner circle

Using Class

The **RadialSlider** property **InnerCircleImageClass** allow to set image for the inner circle of the **RadialSlider**. By default, the **Radial Slider** innerCircleImageClass is set as "null". Refer to the following code example.

ASPX-CS

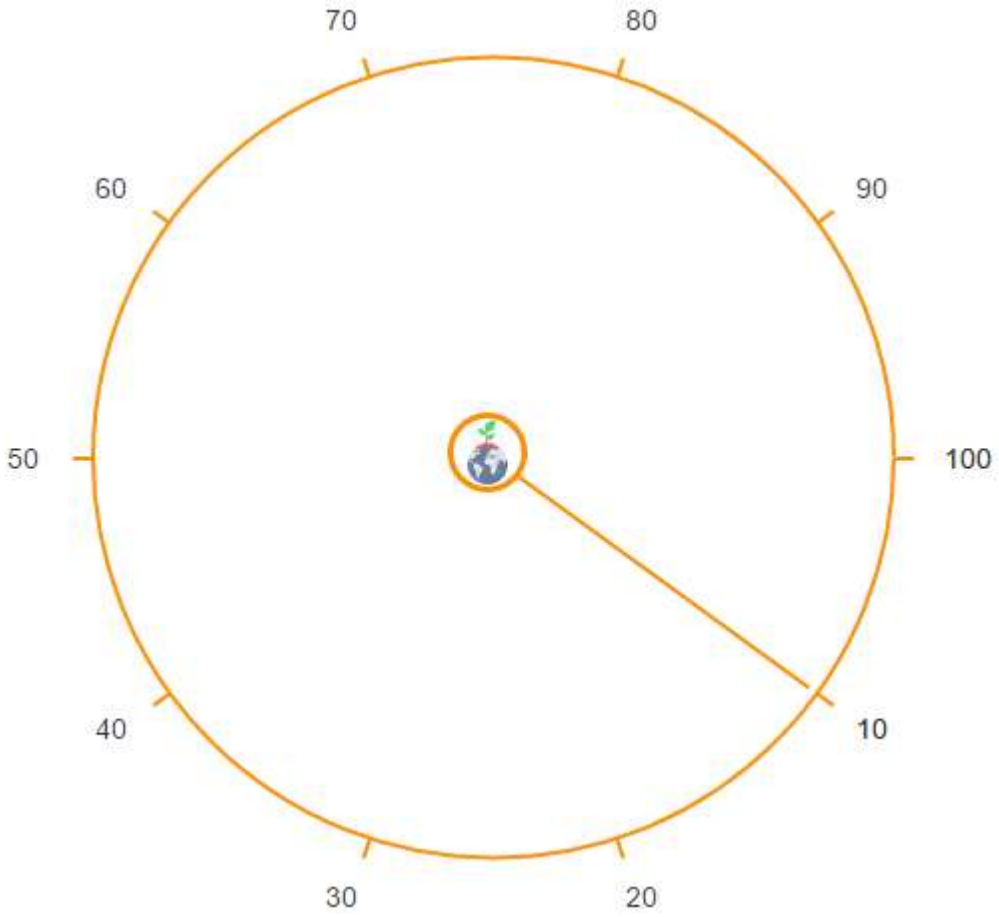
```
@Html.EJ().RadialSlider("slider").InnerCircleImageClass("custom-circle")
```

CSS

```
<style>
.e-custom-circle {
background-image: url("../images/radialslider/diagram.png");
}
</style>
```

The following screenshot illustrates the output of above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Using image URL

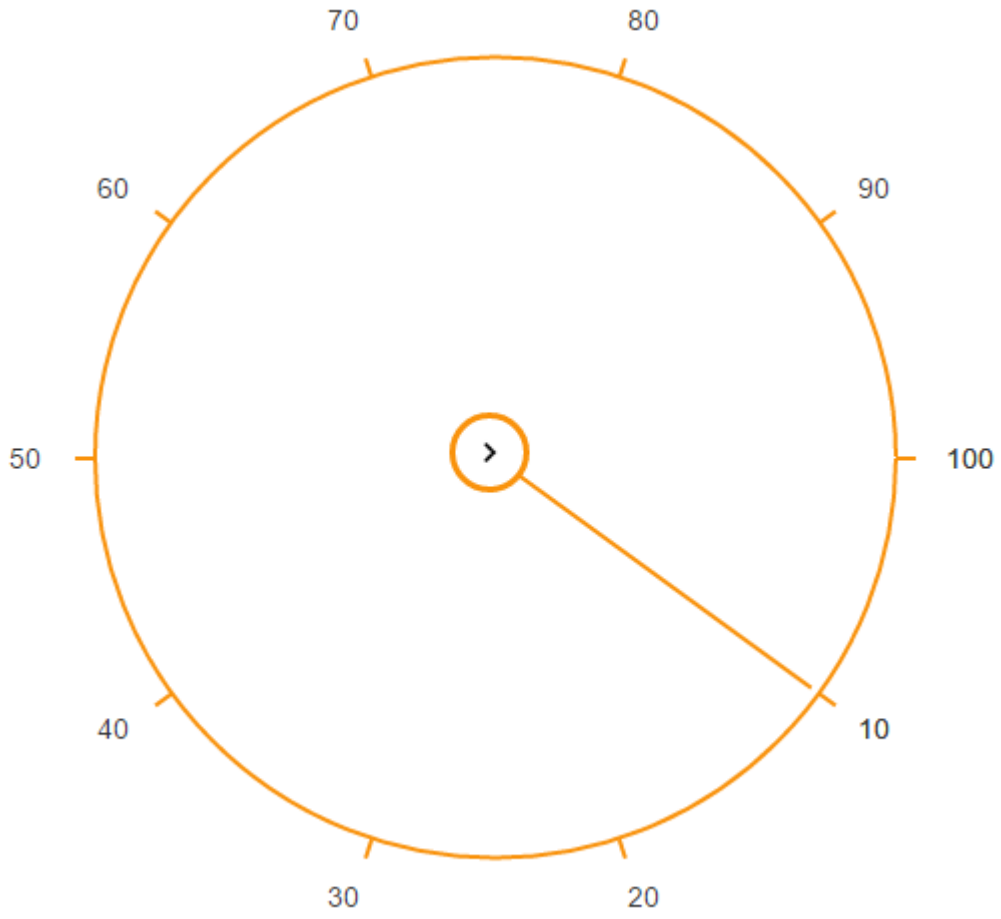
The **RadialSlider** property **InnerCircleImageUrl** allow to set URL image to the inner circle of the **RadialSlider**. By default, the **Radial Slider** InnerCircleImageUrl is set as "null". Refer to the following code example.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").InnerCircleImageUrl("../images/radialsl
ider/chevron-right.png")
```

The following screenshot illustrates the output of above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Appearance and Styling

You can customize RadialSlider control style and appearance by using available themes or cssClass property.

Theme

In order to apply styles to the RadialSlider control, refer to 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.web.all.min.css file, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.web.all.min.css is the combination of these two.

By default, there are 13 theme support available for RadialSlider component, namely:

- flat-azure
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark

Copyright © 2001 - 2020 Syncfusion Inc.

- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap

CssClass

RadialSlider control also allows you to customize its appearance using user-defined CSS. To apply custom themes you can use **CssClass** property, which sets the root class for **RadialSlider** theme.

Using this **CssClass** you can override the existing styles under the theme style sheet. In the following example, the value of **CssClass** property is set as "**Purple-dark**". **Purple-dark** is added as root class to **RadialSlider** control at the runtime. From this root class you can customize the **RadialSlider** appearance.

Add the following code example to the corresponding CSHTML page to render the RadialSlider with the customized style.

ASPX-CS

```
@Html.EJ().RadialSlider("slider").CssClass("Purple-dark")
```

Using below style sheet,exiting theme style sheet file has been over-ridden using root class "**Purple-dark**".

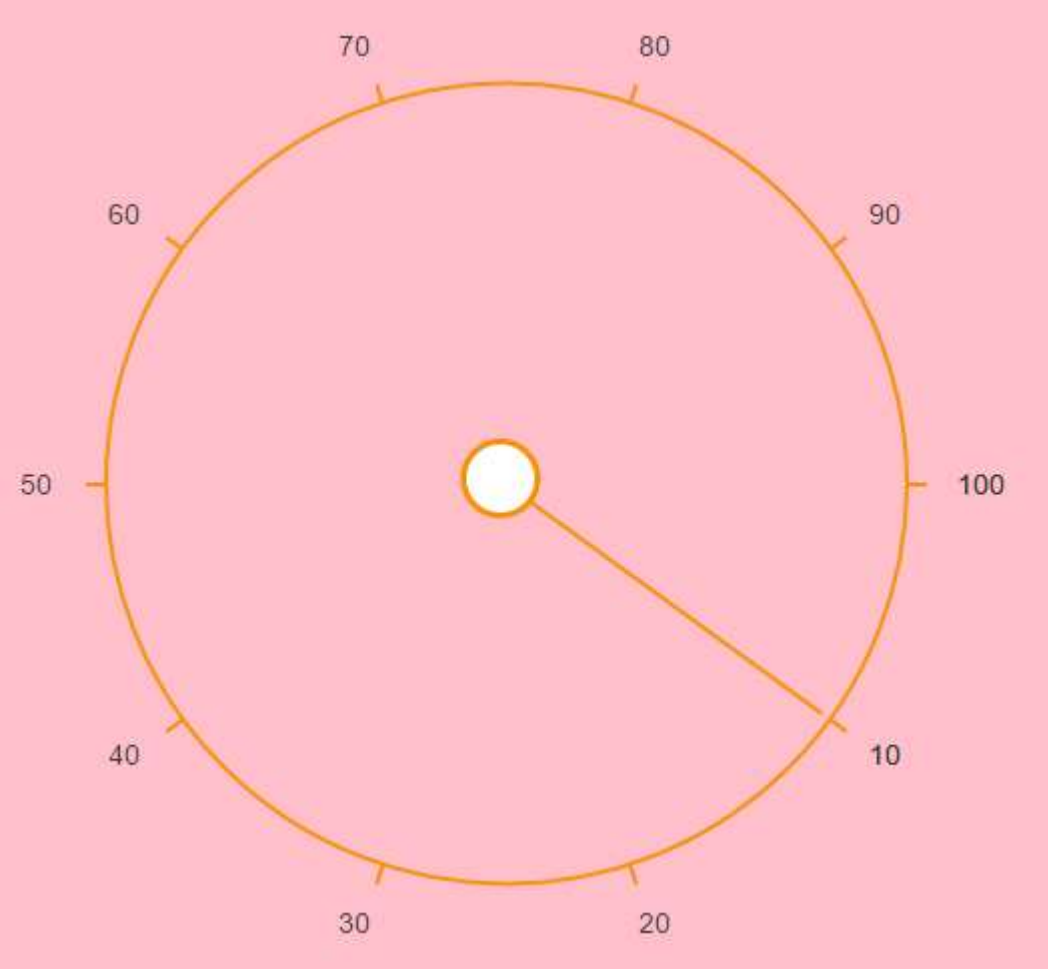
Add the following code in your style section.

CSS

```
.Purple-dark{
background-color:pink;
}
```

The following screenshot illustrates the output of the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



How to

Render RadialSlider from Code behind

RadialSlider can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of RadialSlider properties in the controller.

C#

```
//Namespace to get the JavaScript (RadialSlider) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class RadialsliderController: Controller
{
public ActionResult RadialSliderFeatures()
{
//Initializing the RadialSlider model
RadialSliderProperties sliderObj = new RadialSliderProperties();
//Initializing the other properties
sliderObj.StrokeWidth = 4;
sliderObj.ShowInnerCircle = true;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
sliderObj.InnerCircleImageUrl =
"http://js.syncfusion.com/demos/web/content/images/radialslider/chevron-
right.png";
//Passing RadialSlider properties using the ViewData
ViewData["RSliserModel"] = sliderObj;
return View();
}
}
}
```

Binding the RadialSlider properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().RadialSlider("Radialslider1",
(Syncfusion.JavaScript.Models.RadialSliderProperties)ViewData["RSliserMod
el"]).Render();
}
```

RadioButton

RadioButton

The Essential ASP.NET MVCRadioButton control allows you to choose an option to perform an action. This control allows you to select true/false. These RadioButtons are supported with themes.

Key Features

- Trendy Look : Rich Appearance with Theme Support
- RTL : Supports for Right to Left alignment
- Easy Customization: The customization of RadioButton is made simple

Getting Started

This section briefly describes you on how to create a QuizApp and RegistrationApp using ASP.NET MVC RadioButton control and use the features available in it.

Create your first RadioButton in MVC

ASP.NET MVC RadioButton provides support to display the RadioButton within your web page, and allows you to pick your choice. Using the following guidelines, you can customize RadioButton for a real-time QuizApp and RegistrationApp scenarios. This allows you to select the corresponding choice for each question.

Copyright © 2001 - 2020 Syncfusion Inc.



Create your QuizApp

Essential Studio ASP.NET MVCRadioButton widget has a built-in feature to select a single option from the QuizApp. You can create the RadioButton widget using the following steps.

1. Create an MVC Project and add necessary assemblies and scripts to it.
2. Refer MVC-Getting Started.
3. Add the following code example to the corresponding view page to render RadioButton.

ASPX-CS

```
<div class="frame">
<div>
<div >
<br />
1. What is the Expansion for MVC<br />
<table>
<tr>
<td >
@Html.EJ().RadioButton("Radio1").Name("rad1").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)<!--Creates a small Radio Button-->
<label for="Radio1" class="clslab">
Model View Controller
</label>
</td>
<td  >
@Html.EJ().RadioButton("Radio2").Name("rad1").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)
<label for="Radio2" class="clslab">
Model Virtual Container
</label>
</td>
<td >
@Html.EJ().RadioButton("Radio3").Name("rad1").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)
<label for="Radio3" class="clslab">
Model Visual Controller
</label>
</td>
</tr>
</table>
```

```
<br />
<br />
2.What is the Expansion for USB<br />
<table>
<tr>
<td >
@Html.EJ().RadioButton("Radio4").Name("rad2").Size(RadioButtonSize.Medium
).Checked(false).Enabled(true)<!--Creates a Medium size Radio Button-->
<label for="Radio4" class="clslab">
Universal Serial Bus
</label>
</td>
<td >
@Html.EJ().RadioButton("Radio5").Name("rad2").Size(RadioButtonSize.Medium
).Checked(false).Enabled(true)
<label for="Radio5" class="clslab">
Universal Serial Buffer
</label>
</td>
<td>
@Html.EJ().RadioButton("Radio6").Name("rad2").Size(RadioButtonSize.Medium
).Checked(false).Enabled(true)
<label for="Radio6" class="clslab">
Universal Serialized Buffer
</label>
</td>
</tr>
</table>
<br/>
<br />
3.What is the Expansion for JS<br />
<table>
<tr>
<td >
@Html.EJ().RadioButton("Radio7").Name("rad3").Size(RadioButtonSize.Medium
).Checked(false).Enabled(true)
<label for="Radio7" class="clslab">
Jquery Script
</label>
</td>
<td >
@Html.EJ().RadioButton("Radio8").Name("rad3").Size(RadioButtonSize.Medium
).Checked(false).Enabled(true)
<label for="Radio8" class="clslab">
Java Script
</label>
</td>
<td>
@Html.EJ().RadioButton("Radio9").Name("rad3").Size(RadioButtonSize.Medium
).Checked(false).Enabled(true)
<label for="Radio9" class="clslab">
Json Script
</label>
</td>
</tr>
</table>
<center>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<table>
@Html.EJ().Button("Submit").Width("100px").Size(ButtonSize.Large).Text("S
ubmit").ClientSideEvents(s => s.Click("button"))
</center>
</table>
<br />
</div>
</div>
</div>
```

Add Script

JS

```
<script type="text/javascript">
function button()
{
var checkedItem = [];
$(".e-radiobtn:checked").each(function (index, Element) {
//if ($(this).ejRadioButton('isChecked')) {
checkedItem.push($(this).parent().siblings().html());
//}
});
alert(checkedItem);
}
</script>
```

Configure Style

Add the following code example in the index page.

CSS

```
<style>
.frame
{
width: 80%;
}
</style>
```

Execute the above code example to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Create RegistrationApp



ASPX-CS

```
<div class="frame">
<div>
<div >
<br />
1. Are you a Fresher ?<br />
<table>
<tr>
<td >
@Html.EJ().RadioButton("Radio1").Name("rad1").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)<!--Creates a small Radio Button-->
<label for="Radio1" class="clslab">
Yes
</label>
</td>
<td  >
@Html.EJ().RadioButton("Radio2").Name("rad1").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)
<label for="Radio2" class="clslab">
No
</label>
</td>
</tr>
</table>
<br />
<br />
2. Do you complete any .NET certification courses ?<br />
<table>
<tr>
<td>
@Html.EJ().RadioButton("Radio3").Name("rad2").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)<!--Creates a small Radio Button-->
<label for="Radio1" class="clslab">
Yes
</label>
</td>
<td>
```

```
@Html.EJ().RadioButton("Radio4").Name("rad2").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)
<label for="Radio2" class="clslab">
No
</label>
</td>
</tr>
</table>
<br />
<br />
3. Are you interested to work in .NET platform ?<br />
<table>
<tr>
<td>
@Html.EJ().RadioButton("Radio5").Name("rad3").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)<!--Creates a small Radio Button-->
<label for="Radio1" class="clslab">
Yes
</label>
</td>
<td>
@Html.EJ().RadioButton("Radio6").Name("rad3").Size(RadioButtonSize.Small)
.Checked(false).Enabled(true)
<label for="Radio2" class="clslab">
No
</label>
</td>
</tr>
</table>
<br />
<br />
<center>
<table>
@Html.EJ().Button("Submit").Width("100px").Size(ButtonSize.Large).Text("S
ubmit").ClientSideEvents(s => s.Click("button"))
</center>
</table>
<br />
</div>
</div>
</div>
```

Add Script

JS

```
<script type="text/javascript">
function button()
{
var checkedItem = [];
var result = "";
$(".e-radiobtn:checked").each(function (index, Element)
{
quest = index + 1;
checkedItem.push("Question" + quest + ":" +
$(this).parent().siblings().html().trim()+" ");
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

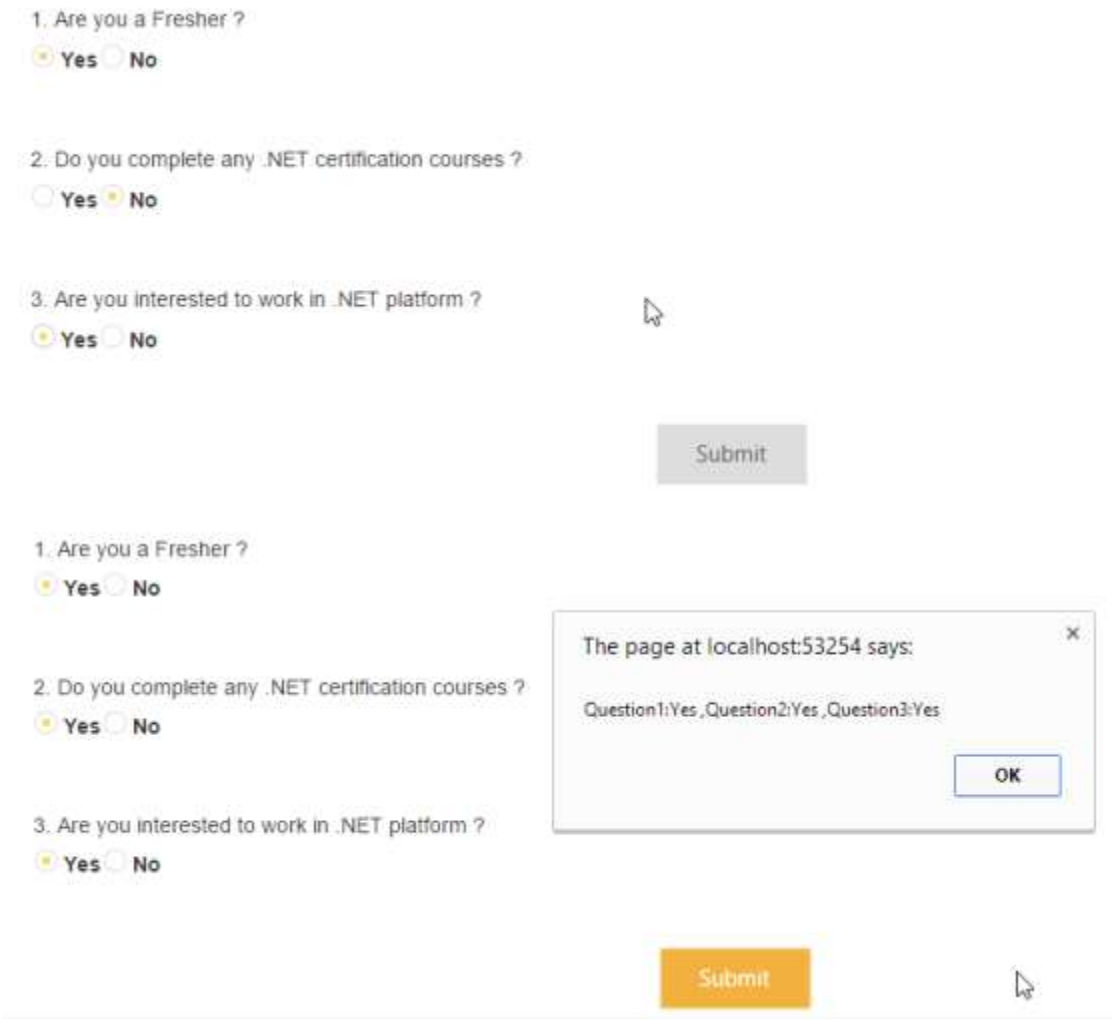
```
alert(checkedItem);
}
</script>
```

Add Style

CSS

```
<style>
.frame
{
width: 80%;
}
</style>
```

Execute the above code example to render the following outputs.



The whole purpose of Radio Button is to enable you to make one selection, only one, from a list. When you want to make multiple selections, you do not require a Radio Button, you need to use a CheckBox.

Copyright © 2001 - 2020 Syncfusion Inc.

You can make only one selection from the group of radio buttons. All radio buttons in that group need to share the same name. This tells the form handler, the name of the group, and the value of the selected Radio Button.

Radio Button is mutually exclusive; that is, when you select a radio button, any previously selected radio buttons in the same group become deselected.

For example, when you want to choose between a groups of colors, you can create Radio Buttons for each color. You can give all the radio buttons the same name, for example, "preferred_color", but you must give each radio button a different value, for example, "red".

1. All radio buttons within a group must share the same name. That is, the value of the name attribute has to be the same. For example, all three radio buttons in the "Preferred Color" group have the name "preferred_color".
2. All radio buttons within a group must have a different value for the value attribute. For example, when one radio button has value as "red", none of the others in that group can have a value of red, as this would defeat the purpose of having the extra radio button.
3. The label for each radio button is determined by the text next to the radio button and not by the value attribute. The value attribute is used by the form handler.

Easy Customization

Checked status

You have options to set the state of the radio button as either checked or unchecked. When you select any option from the group of radio buttons, a dot mark appears inside the circle. This is called the checked state. Previously selected radio buttons in this group are unselected that is they go to the unchecked state. The Checked property is used to set the state of the radio button.

The following steps explain the details about rendering the Radio Button with the checked option

In the CSHTML page, add the following input elements to configure Radio Button widget.

ASPX-CS

```
// Add the code in CSHTML page to configure Radio Button
@* Here checked and unchecked type of radio buttons are rendered in same
group
set checked state of radio button as follows*@
@Html.EJ().RadioButton("Radio_checked").Name("Gender").Size(RadioButtonSi
ze.Small).Checked(true)
<label for="Radio_checked" class="clslab">Male</label>
<br />
@Html.EJ().RadioButton("Radio_unchecked").Name("Gender").Size(RadioButton
Size.Small).Checked(false)
<label for="Radio_unchecked" class="clslab">Female</label>
```

Configure the CSS styles to align the radio buttons.

CSS

```
<style>
.page-align
{
margin: 100px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</style>
```

The following image is displayed as the output for the above steps.



Text

Specifies the text content for the radio button. In previous programs, separate labels were created for each radio button. But now you have the option to set the text for radio button using the Text property. So here you do not have to add a label tag for each radio button in the HTML code.

The following steps explain the details about rendering the Radio Button with Text and without using the label tag options.

In the CSHTML page, add the following input elements to configure the Radio Button widget.

ASPX-CS

```
// Add the code in CSHTML page to configure Radio Button
@*radio button with text property*@
@Html.EJ().RadioButton("RadBtn_male").Name("Gender").Size(RadioButtonSize
.Small).Checked(true).Text("Male")
<br />
@Html.EJ().RadioButton("RadBtn_female").Name("Gender").Size(RadioButtonSi
ze.Small).Checked(false).Text("Female")
```

Configure the CSS styles to align the radio buttons.

CSS

```
<style>
.page-align
{
margin: 100px;
}
</style>
```

The following image is displayed as the output for the above steps.



Size

You can render the Radio Button in different sizes. There are some predefined size options available for rendering a Radio Button in an easy way. Each size option has different height and width. It mainly avoids the complexity in rendering Radio Button with complex CSS class.

Copyright © 2001 - 2020 Syncfusion Inc.

Size

Property	Description
Small	Creates radio button with Built-in small size height, width specified.
Medium	Creates radio button with Built-in medium size height, width specified.

The following steps explain the details about rendering Radio Button with different size options.

In the CSHTML page, add the following input elements to configure Radio Button widget.

ASPX-CS

```
// Add the code in CSHTML page to configure Radio Button
@* small and medium size of radio buttons in same group
By default, here no one radio button is checked*@
Small size radio buttons
<br />
@Html.EJ().RadioButton("Radio_Male").Name("Gender").Size(RadioButtonSize.
Small).Checked(true)
<label for="Radio_Male" >Male</label>
<br />
@Html.EJ().RadioButton("Radio_Female").Name("Gender").Size(RadioButtonSiz
e.Small).Checked(false)
<label for="Radio_Female" >Female</label>
<br />
Medium size radio buttons
<br />
@Html.EJ().RadioButton("Radio1_Male").Name("Gender1").Size(RadioButtonSiz
e.Medium).Checked(true)
<label for="Radio1_Male">Male</label>
<br />
@Html.EJ().RadioButton("Radio1_Female").Name("Gender1").Size(RadioButtonS
ize.Medium).Checked(false)
<label for="Radio1_Female" >Female</label>
```

Configure the CSS styles to align the radio buttons.

CSS

```
<style>
.page-align
{
margin: 100px;
}
</style>
```

The following image is displayed as the output for the above steps.

Copyright © 2001 - 2020 Syncfusion Inc.



RTL Support

In some cases you need to use right-to-left alignment. You can give RTL support by using EnableRTL property. RTL mode works when you use the Text property in Radio Button. The Radio Buttons and text are aligned in the right-to-left format. For example, when text is right-aligned and Radio Button is left-aligned, after you apply right-to-left alignment, these positions are interchanged.

The following steps explain the details about rendering the Radio Button with right-to-left alignment support. Here the text property is necessary.

In the CSHTML page, add the following button elements to configure Radio Button widget.

ASPX-CS

```
// Add the code in CSHTML page to configure Radio Button
@*set radio button with right to left format*@
<div class="page-align">
<table class="rightAlign">
<tr>
<td>
@Html.EJ().RadioButton("RadBtn_male").Name("Gender").Size(RadioButtonSize
.Small).Checked(true).Text("Male").EnableRTL(true)
</td>
</tr>
<tr>
<td>
@Html.EJ().RadioButton("RadBtn_female").Name("Gender").Size(RadioButtonSi
ze.Small).Checked(false).Text("Female").EnableRTL(true)
</td>
</tr>
</table>
</div>
```

In the above mentioned code, the text property has been used. In LTR format, the Radio Button is on the left side. In RTL format, the Radio Button appears on the right side. Here the Text property is used and the EnableRTL property is set as “true”. It changes the alignment to right-to-left.

Configure the CSS styles to align the Radio Buttons.

CSS

```
<style>
.page-align
{
margin: 100px;
}
.rightAlign
{
text-align: right;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</style>
```

The following image is displayed as the output for the above steps.



Styles Customization

RadioButton allows you to customize its appearance by using user-defined CSS and custom skin options such as colors and backgrounds. To apply custom themes, use **CssClass** property. CssClass property sets the root class for RadioButton theme.

By using this CssClass, you can override the existing styles under the theme style sheet. The theme style sheet applies theme-specific styles like colors and backgrounds. From the root class, you can customize the RadioButton control theme.

In the following example, the border color and border width of the active RadioButton is customized through the custom classes to create the success, and danger indication with RadioButton.

HTML

```
<span><b>Agree terms & conditions:</b></span>
<span style="margin-left: 15px">
@Html.EJ().RadioButton("Radio_checked").Text("Yes").Name("radio1").CssCla
ss("success").Size(Syncfusion.JavaScript.RadioButtonSize.Medium)
@Html.EJ().RadioButton("Radio_unchecked").Text("No").Name("radio1").CssCl
ass("warning").Size(Syncfusion.JavaScript.RadioButtonSize.Medium)
</span>
<br>
<br>
<span><b>Confirm:</b></span>
<span style="margin-left: 15px">
@Html.EJ().RadioButton("RadioButton3").Text("Yes").Name("radio2").CssClas
s("success").Size(Syncfusion.JavaScript.RadioButtonSize.Medium)
@Html.EJ().RadioButton("RadioButton4").Text("No").Name("radio2").CssClass
("warning").Size(Syncfusion.JavaScript.RadioButtonSize.Medium)
</span>
```

CSS

```
<style>
.success .e-spanicon.e-rad-active,
.success .e-spanicon.e-rad-active:hover {
border-color: #17ad37;
border-width: 2px;
}
.warning .e-spanicon.e-rad-active,
.warning .e-spanicon.e-rad-active:hover {
border-color: #ff6e40;
border-width: 2px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Miscellaneous

Radio Button ID

Radio Button id is not shown in the user interface. Here Id denotes the Id attribute of the root element of Radio Button control. This Id value is unique. You can give Id through element and through the Id property. When you use two ids for a single radio button at initialization, the element Id is considered.

Set Id for Radio Button control as follows.

ASPX-CS

```
@*set new id value as follows*@
@Html.EJ().RadioButton("RadioButton_Male").Id("male_type").Name("Gender")
<br />
@Html.EJ().RadioButton("RadioButton_female").Id("female_type").Name("Gend
er")
```

Radio Button Prefix id

Id prefix value is appended to the element id value. It is used to mention the prefix for the wrapper's id attribute. When you assign a value for IdPrefix property, the older prefix id gets replaced by the new prefix id.

Setting a new prefix id for Radio Button control is as follows.

ASPX-CS

```
@*set new idPrefix  value as follows*@
@Html.EJ().RadioButton("Radio_Male").IdPrefix("sync_").Name("Gender")
<br />
@Html.EJ().RadioButton("Radio_Female").IdPrefix("sync_").Name("Gender")
```

Radio Button Name

The Name setting tells you where a Radio Button belongs. When you select one button, all other buttons in the same group are unselected. You can have only one group of Radio Buttons on each page.

The Name attribute is also used to identify form data after it has been submitted to the server, or for reference of form data using JavaScript on the client's side. Only form elements with a Name attribute have their values passed, when submitting a form.

Radio Button Value

The Value setting defines what can be submitted when checked.

For Radio Buttons, the contents of the value property do not appear in the user interface. The Value property only has meaning when submitting a form. If a radio button is in the checked state when the form is submitted, the name of the Radio Button is sent along with the value of the value property, if the radio button is not checked, no information is sent.

Copyright © 2001 - 2020 Syncfusion Inc.

To identify, on the server side, which one was checked, give different values for radio buttons in the same group,

Set name and value for each radio button control as follows.

ASPX-CS

```
@*set name and value for each radio button as follows*@
@Html.EJ().RadioButton("Radio_Male").Name("Gender").Value("male")
<br />
@Html.EJ().RadioButton("Radio_Female").Name("Gender").Value("female")
```

How to?

Render RadioButton from Code behind

RadioButton can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of RadioButton properties in the controller.

C#

```
//Namespace to get the JavaScript (RadioButton) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class RadioButtonController : Controller
{
// GET: RadioButton
public ActionResult RadioButtonFeatures()
{
//Initializing the RadioButton model
RadioButtonProperties BtnObj = new RadioButtonProperties();
//Initializing the datasource and other properties
BtnObj.Checked = true;
BtnObj.Size = RadioButtonSize.Medium;
BtnObj.Text = "Computer";
//Passing RadioButton properties using the ViewData
ViewData["BtnModel"] = BtnObj;
return View();
}
}
}
```

Binding the RadioButton properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().RadioButton("searchCustomer",(Syncfusion.JavaScript.Models.Radi
oButtonProperties)ViewData["BtnModel"]).Render();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

RangeNavigator

RangeNavigator

RangeNavigator is a data visualization control. It allows you to scroll and navigate through the data. This control easily combines with other controls such as Chart, GridView, etc., to create rich and powerful dashboards.

Key Features

The **RangeNavigator** is composed of the following features:

- **Higher level label:** Contains a time span format one level higher than the lower level **DateTime** values. For example, if the higher level label contains the year format (yyyy), the lower level label contains the Quarter format.
- **Lower level label:** Contains a time span format one level lower than the higher level **DateTime** values. For example, if the lower level label contains the Quarter format, the higher level label contains the year format (yyyy).
- **Chart:** To represent the data in RangeNavigator.
- **Slider:** To handle the selected data in RangeNavigator.

Getting Started

This section explains briefly about how to create a **RangeNavigator** in your ASP.NET MVC application.

Create your first RangeNavigator in MVC

This section encompasses the details on how to configure the **RangeNavigator** and update the **chart** control for **RangeNavigator's** selected range. It also helps you to learn how to pass the required data to **RangeNavigator** and customize the scale and selected range for your requirements. In this example, you will learn how to configure the RangeNavigator to analyze sales of a product for a particular quarter in a year.

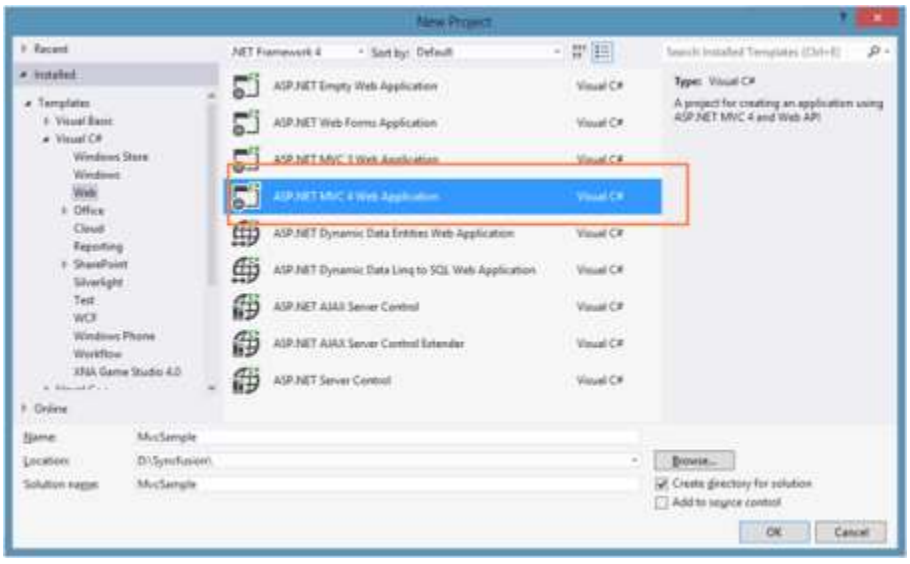
Copyright © 2001 - 2020 Syncfusion Inc.



Create a simple MVC Application for RangeNavigator

You can create a new ASP.NET MVC project Razor Application.

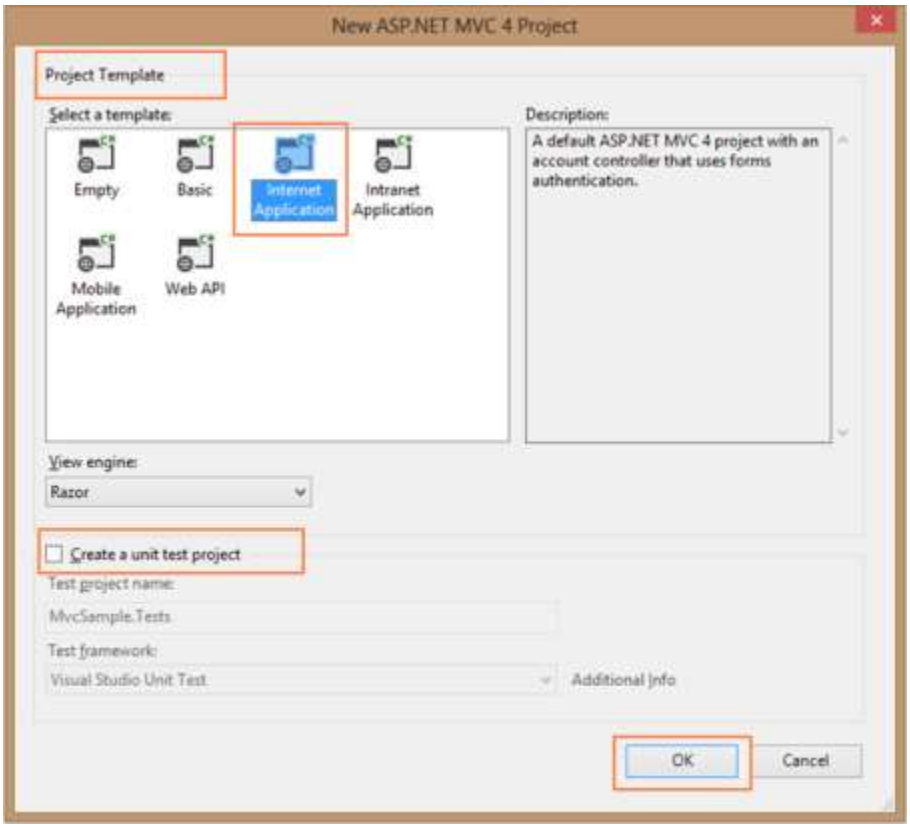
1. On the File menu, click New Project. The New Project dialog box opens.



2. On the upper-right corner, select .NET Framework 4.0.
3. In the Installed Templates pane, expand either Visual Basic or Visual C# and then click Web.
4. In the Visual Studio Installed Templates pane, select ASP.NET MVC 4 Web Application.
5. In the Name box, enter MvcSample

Copyright © 2001 - 2020 Syncfusion Inc.

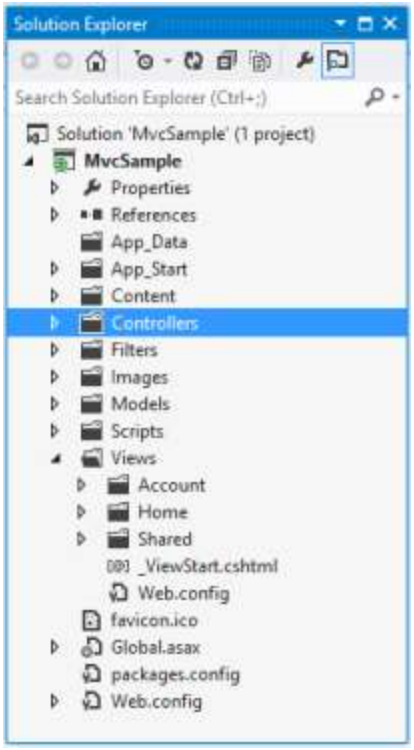
6. In the Location box, enter a name for the project folder.
7. If you want the name of the solution to differ from the project name, then enter a name in the Solution name box.
8. Select the Create directory for solution checkbox.
9. Click OK.
10. The New ASP.NET MVC4Project dialog box opens.



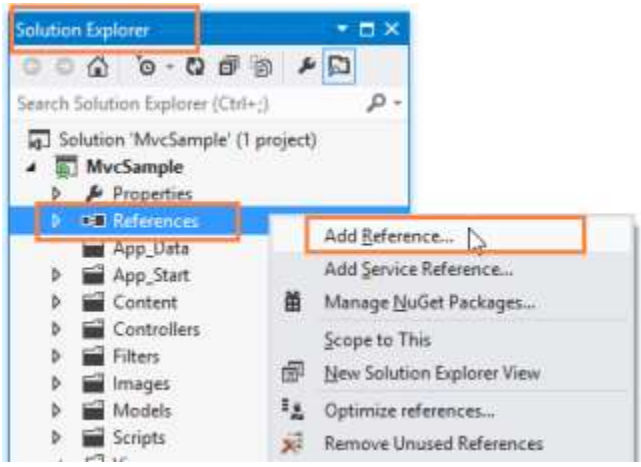
11. In the Project Template group, select the Internet Application template for MVC4 project.
12. Check the Create a unit test project checkbox, if you want to create a unit test project.
13. Click OK. The new MVC application project is generated.

The following screenshot shows the folder structure of the newly created MVC project.

Copyright © 2001 - 2020 Syncfusion Inc.



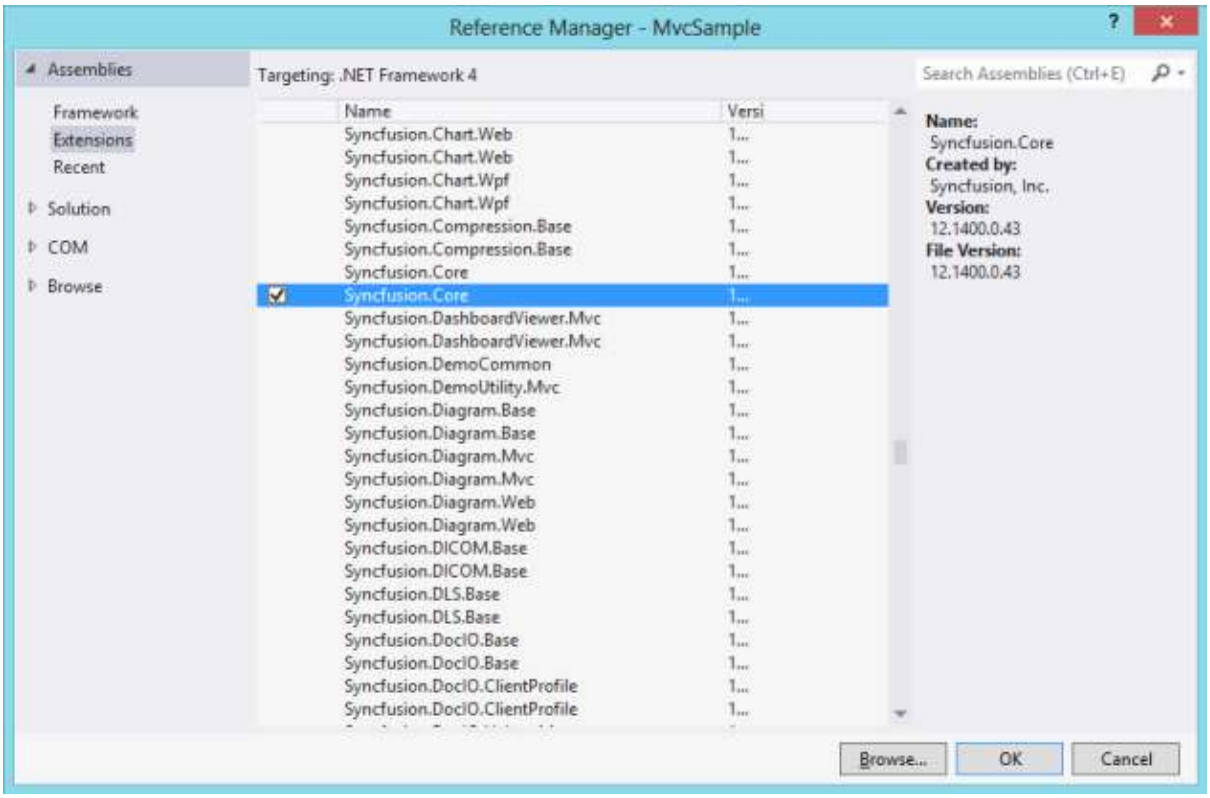
14. Adding Reference Assemblies
    - On the Solution Explorer, right-click the References folder and then click Add Reference. The Add Reference dialog box appears.

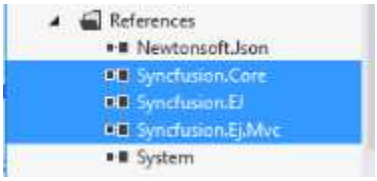


Adding Reference Assemblies

- Add the following Syncfusion assemblies :
    1. Syncfusion.Core
    2. Syncfusion.EJ
    3. Syncfusion.EJ.MVC

Copyright © 2001 - 2020 Syncfusion Inc.





15. You can add required Scripts and ScriptManger in _Layout page
    - Add the script references of the required libraries in the _Layout page as shown in the following code sample

ASPX-CS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="@Url.Content("~/Scripts/jquery-1.10.2.min.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.globalize.min.js")"
type="text/javascript" ></script>
<script src="@Url.Content("~/Scripts/ej.web.all.min.js")"
type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/ej.unobtrusive.min.js")"
type="text/javascript"></script>
</head>
```

- Add the ScriptManager() at the end of tag in _Layout page.

ASPX-CS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
</head>
<body>
@(Html.EJ().ScriptManager())
</body>
</html>
```

16. Configure web.config files for assemblies
    - The following assemblies references are added properly in web.config file (There will be two web.config files. Refer to the one in the root folder.).

ASPX-CS

```
<compilation debug="true" targetFramework="4.0" >
<assemblies>
<add assembly="Syncfusion.Core, Version=XX.XXXX.X.X, Culture=neutral,
PublicKeyToken=632609B4D040F6B4" />
<add assembly="Syncfusion.EJ, Version= XX.XXXX.X.X, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
<add assembly="Syncfusion.EJ.Mvc, Version= XX.XXXX.X.X, Culture=neutral,
PublicKeyToken=3d67ed1f87d44c89" />
</assemblies>
</compilation>
```

Note: Add the following namespaces in web.config file within the Views folder.

ASPX-CS

```
<namespaces>
<add namespace="System.Web.Mvc" />
<add namespace="System.Web.Mvc.Ajax" />
<add namespace="Syncfusion.JavaScript" />
<add namespace="Syncfusion.JavaScript.DataVisualization" />
<add namespace="Syncfusion.MVC.EJ" />
<add namespace="Syncfusion.EJ" />
</namespaces>
```

The above steps are done automatically, if you create your project with the help of MVC Project creation template, which is available once the MVC Extension is installed on your machine.

Configure RangeNavigator

Getting started with your MVC **RangeNavigator** is simple; all you need to do is initialize the **RangeNavigator** by setting range values.

1. Create a simple

   tag.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
<div>
</div>
```

2. Add the following code in the SimpleRangeNavigator.cshtml file, to create the RangeNavigator control in the View page.

The following code example renders a **RangeNavigator** with a range from 2010 January 1st to December 31st.

ASPX-CS

```
<div>
@(Html.EJ().RangeNavigator("container")
.RangeSettings(range=>range.Start("2010/1/1").End("2010/12/31"))
)
</div>
```

3. Open ~/Controllers/HomeController.cs.
4. Add the SimpleRangeNavigator() action as illustrated in the following code example.

C#

```
public ActionResult SimpleRangeNavigator()
{
return View();
}
```

The following Screen shot displays the **RangeNavigator** with a range from 2010 January 1st to December 31st.



Add series

To add a series to **RangeNavigator**, you need to set DataSource property, as given in the following code example.

You can add JSON data to the RangeNavigator using the Datasource property.

In Controllers/HomeController.cs specify the data for data source.

C#

```
public ArrayList GetData()
{
ArrayList dataTable = new ArrayList();
dataTable.Add(new NavigatorData (new DateTime(2011, 01, 01), 10));
dataTable.Add(new NavigatorData (new DateTime(2011, 02, 01), 5));
```

Copyright © 2001 - 2020 Syncfusion Inc.

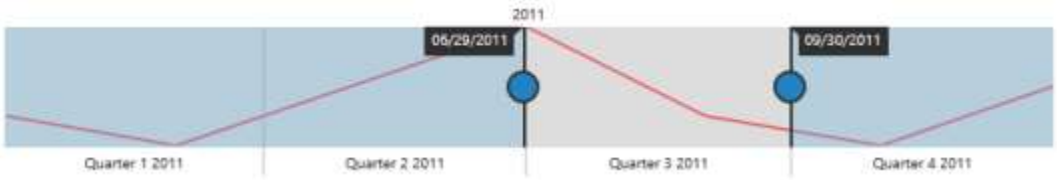
```
dataTable.Add(new NavigatorData (new DateTime(2011, 04, 01), 15));
dataTable.Add(new NavigatorData (new DateTime(2011, 06, 01), 25));
dataTable.Add(new NavigatorData (new DateTime(2011, 08, 01), 10));
dataTable.Add(new NavigatorData (new DateTime(2011, 10, 01), 5));
dataTable.Add(new NavigatorData (new DateTime(2011, 12, 31), 15));
return dataTable;
}
class NavigatorData
{
private DateTime date;
public DateTime xDate
{
get { return date; }
set { date = value; }
}
private double value;
public double yValue
{
get { return value; }
set { value = value; }
}
public NavigatorData(DateTime date, double value)
{
this.xDate = date;
this.yValue = value;
}
}
public ActionResult SimpleRangeNavigator()
{
var DataSource = GetData();
ViewBag.datasource = DataSource;
return View();
}
```

In SimpleRangeNavigator.cshtml specify the type of series you want to render using Type property. And specify the Datasource to the Series of RangeNavigator.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.DataSource((System.Collections.IEnumerable)ViewBag.datasource)
.XName("xDate")
.YName("yValue")
)
```

The following screenshot displays the RangeNavigator with the type series as “line”.



Copyright © 2001 - 2020 Syncfusion Inc.

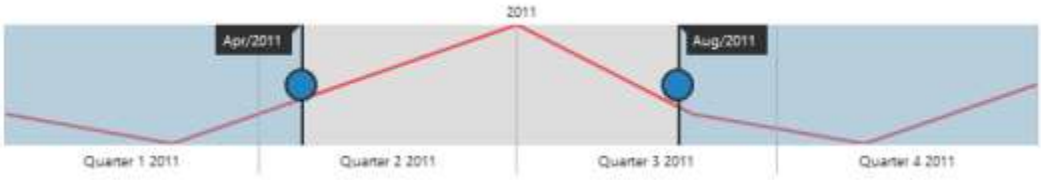
Enable tooltip

Tooltip can be customized for RangeNavigator using TooltipSettings option. You can also use TooltipDisplayMode option in tooltip to display the tooltip “always” or “ondemand” (displays tooltip only while dragging the sliders). You can also specify label format for tooltip using LabelFormat.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.TooltipSettings(tooltip=>tooltip.Visible(true).LabelFormat("MMM/yyyy").T
ooltipDisplayMode("always"))
)
```

The following screen shot displays the label format Tooltip in RangeNavigator:



Update Chart

RangeNavigator is used along with the controls like **chart** and **grid** to view the range of data selected in RangeNavigator.

In order to update **chart**, whenever the selected range changes in **RangeNavigator**, you need to use RangeChanged event of **RangeNavigator** and then update the **chart** with the selected data in this event.

Now, add the DataSource to the series and provide the field name to get the values from the DataSource in XName and YName options and also trigger the RangeChanged event for updating the chart.

ASPX-CS

```
@(Html.EJ().Chart("container")
.Title(title => title.Text("Sales Analysis"))
.Legend(legend => legend.Visible(true).Position(LegendPosition.Top))
.PrimaryYAxis(axis => axis.Title(axisTitle =>
axisTitle.Text("Sales(Million)")))
.Series(series =>
{
series.Name("Product A").Type(SeriesType.Line)
.DataSource((System.Collections.IEnumerable)ViewBag.datasource)
.XName("xDate")
.YName("yValue")
.Add();
}))
@(Html.EJ().RangeNavigator("container")
.DataSource((System.Collections.IEnumerable)ViewBag.datasource)
.XName("xDate")
.YName("yValue")
.RangeChanged("onRangeChanged")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example illustrates how to use the RangeChanged event of RangeNavigator for updating the chart with the selected data.

ASPX-CS

```
<script type="text/javascript" language="javascript">
function onRangeChanged(sender)
{
var chart = $("#container").data("ejChart");
if (chart != null)
{
chart.model.series[0].dataSource = sender.selectedData;
$("#container").ejChart("redraw");
}
}
</script>
```

The following screenshot displays how the RangeNavigator is updated when the selected range is changed.



Set value type

RangeNavigator can also be used with numerical values. You can specify the data type using ValueType option.

First let's create a DataSource for Chart Series with integer Values.

In Controllers/HomeController.cs specify the data for data source.

C#

```
public ArrayList GetData()
```

```
{
ArrayList dataTable = new ArrayList();
dataTable.Add(new NavigatorData (0, 10));
dataTable.Add(new NavigatorData (50, 5));
dataTable.Add(new NavigatorData (100, 15));
dataTable.Add(new NavigatorData (150, 25));
dataTable.Add(new NavigatorData (200, 10));
dataTable.Add(new NavigatorData (250, 5));
dataTable.Add(new NavigatorData (300, 15));
return dataTable;
}
class NavigatorData
{
private double xValue;
public double xValue
{
get { return xValue; }
set { xValue = value; }
}
private double yValue;
public double yValue
{
get { return yValue; }
set { yValue = value; }
}
public NavigatorData(double xValue, double yValue)
{
this.xValue = xValue;
this.yValue = yValue;
}
}
public ActionResult SimpleRangeNavigator()
{
var DataSource = GetData();
ViewBag.datasource = DataSource;
return View();
}
```

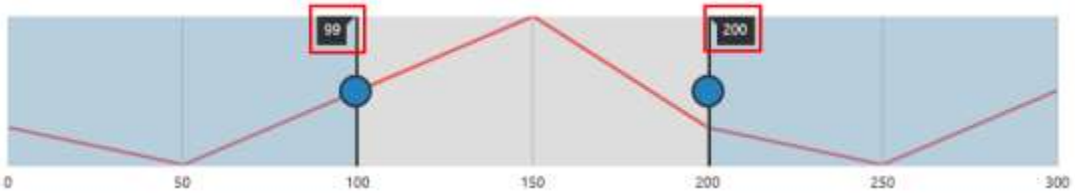
In SimpleRangeNavigator.cshtml, specify the DataSource to the Series and provide the field name to get the values from the DataSource in XName and YName options series and also set the ValueType property to “numeric” as given in the following code example.

ASPX-CS

```
@(Html.EJ().RangeNavigator ("container")
.Series(sr =>
{
sr.Type(SeriesType.Line)
.DataSource((System.Collections.IEnumerable)ViewBag.datasource)
.XName("xDate")
.YName("yValue")
.Add();
})
.ValueType("numeric"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays the RangeNavigator with numerical values:



Range Types

RangeNavigator control is designed to visualize large number of data and navigate to particular data from the large collection at ease. The data for the RangeNavigator is either **Numeric** values or **DateTime** values and the ValueType property in RangeNavigator indicates the type of the data that should be passed for the control. By default the ValueType of RangeNavigator is DateTime

- Numeric
- DateTime

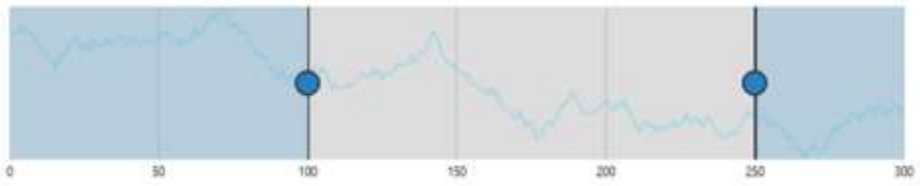
Numeric Type

RangeNavigator is also used with numeric data and the ValueType for this data is **"numeric"**

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.ValueType("numeric")
)
```

The following screenshot displays the RangeNavigator with numeric data.



DateTime

By default the ValueType of the RangeNavigator is **"datetime"** and represents the **DateTime** values.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.ValueType("datetime")
)
```



Copyright © 2001 - 2020 Syncfusion Inc.

DateTime Intervals

The **DateTime** range type contains an IntervalType property that sets the DateTime interval to one of the following:

- Years
- Quarters
- Months
- Weeks
- Days
- Hours

By default IntervalType for higher level labels are **Years** and for lower level labels its **Quarters**.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.LabelSettings(ls=>ls
.HigherLevel(higher=>higher.IntervalType(NavigatorIntervalType.Years))
.LowerLevel(li=>li.IntervalType(NavigatorIntervalType.Quarters))
))
```



Range Padding

Range Padding adds padding for range in **RangeNavigator**. It allows you to space the grid lines in the RangeNavigator. By default, this property is set to none.

Numeric

The RangePadding property allows you to customize the automatic range calculation using the default auto range calculation for **RangeNavigator**.

ASPX-CS

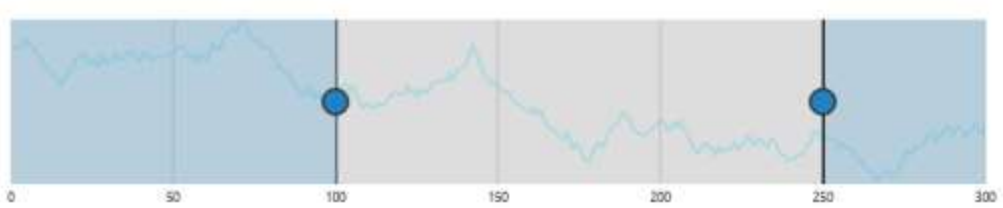
```
@(Html.EJ().RangeNavigator("container")
.RangePadding("none")
.ValueType("numeric")
)
```

None:

By default, the RangePadding for numerical range is none. The range is calculated from the minimum value to the maximum value of data in the RangeNavigator.

The following screenshot illustrates a **RangeNavigator** with RangePadding set to none.

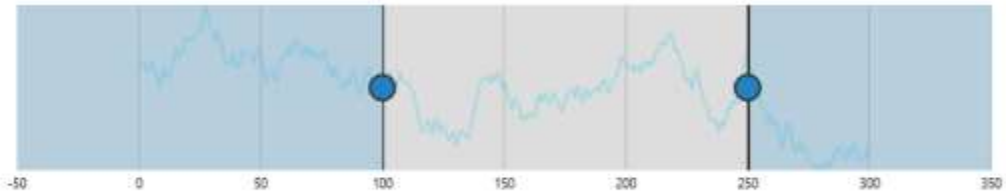
Copyright © 2001 - 2020 Syncfusion Inc.



Additional:

When you set the RangePadding for numerical range to **Additional**, range is padded with an interval.

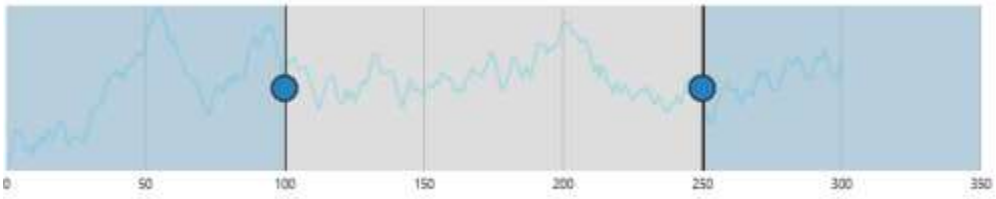
The following screenshot illustrates a **RangeNavigator** with RangePadding set to additional.



Normal:

In normal RangePadding, automatic range calculation differs based on the data.

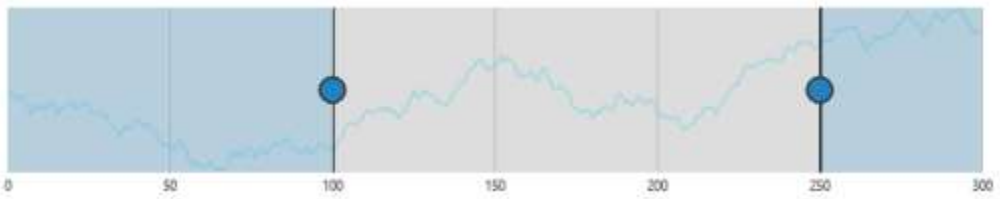
The following screenshot illustrates **RangeNavigator** with RangePadding set to **normal**



Round:

Round RangePadding for a numerical range rounds the range of the control to the nearest possible value that is divisible by the interval.

The following screenshot illustrates a **RangeNavigator** with RangePadding set to **Round**.



DateTime

Using the default range calculation for **RangeNavigator**, the RangePadding property allows you to customize the range.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().RangeNavigator("container")
.RangePadding("none")
)
```

None:

By default, the RangePadding for **DateTime** range is none. The range is calculated from the minimum value to the maximum value of data in the RangeNavigator

The following screenshot illustrates a **RangeNavigator** with RangePadding set to none.



Round:

Round RangePadding for a **DateTime** range rounds the range of the control to the nearest possible value.

The following screenshot illustrates a **RangeNavigator** with RangePadding set to Round.



Padding

The gap between the container and the **RangeNavigator** can be specified using Padding property.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Padding(15)
)
```

AllowSnapping

An AllowSnapping property toggles the placement of slider exactly on the place it left or on the nearest interval.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.AllowSnapping(true)
)
```

Responsive

Set IsResponsive value to make the **RangeNavigator** responsive on resize.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().RangeNavigator("container")
.IsResponsive(true)
)
```

Auto Resizing

Enable EnableAutoResizing option to resize the **RangeNavigator**.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.EnableAutoResizing(true)
)
```

Customize range Navigator border

RangeNavigator provides options to customize the Color, Opacity and Width of range navigator Border.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Border(border=>border.Color("green").Opacity(0.5).Width(2))
)
```

Customize size of range navigator

The Height and Width of **RangeNavigator** can be customized using SizeSettings property.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.SizeSettings(size=>size.Height("130").Width("900"))
)
```

Customize axis range of navigator

RangeNavigator calculates the range automatically based on the values of series data points. However you can explicitly specify the range using the Start, End properties in RangeSettings that is not possible when data is provided.

The following code example renders a RangeNavigator with a range from 2010 January 1st to 2013 January 1st.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
. RangeSettings(range=>range.Start("2010/1/1").End("2012/13/1"))
)
```



Copyright © 2001 - 2020 Syncfusion Inc.

Populate Data

When you provide data to **RangeNavigator**, it produces limited set of data. You can populate the RangeNavigator with data using the DataSource and Series properties.

Add series to the RangeNavigator

The Series property provides access to a collection of all series that are defined explicitly within a **RangeNavigator** control. Each series is assigned with Type and Name. It contains collection of data point, each point contains x value and y values. You can add data points to the series through DataSource property field name to get the values from the dataSource in XName and YName options.

Animation can be enabled by setting EnableAnimation property as true and the series color and opacity can be customized by using Fill and Opacity property in series.

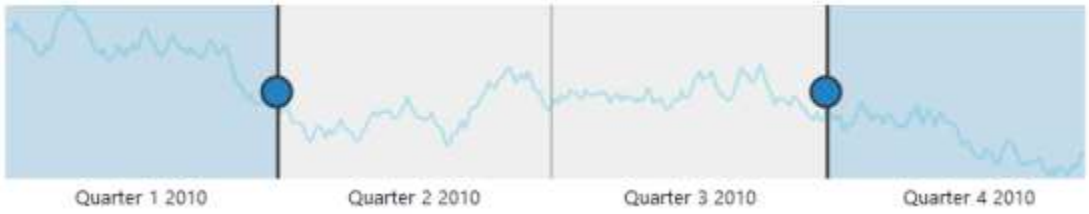
ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
// Adding Series
.Series(sr =>
{
sr.Name("ProductA")
.EnableAnimation(false)
.Type(SeriesType.Line)
.Opacity(0.5)
.Fill("#69D2E7").Add();
})
.Load("loadingData"))
<script type="text/javascript">
function loadingData(sender)
{
data = GetData();
sender.model.series[0].dataSource = data.Open;
sender.model.series[0].xName =  "XValue",
sender.model.series[0].yName = "YValue";
}
// Method to get data in JSON format
function GetData()
{
var series1 = [];
var series2 = [];
var value = 100;
var value1 = 120;
for (var i = 1; i < 730; i++)
{
if (Math.random() > .5)
{
value += Math.random();
value1 += Math.random();
}
else
{
value -= Math.random();
value1 -= Math.random();
}
var point1 = { XValue: new Date(2010, 0, i), YValue: value };
var point2 = { XValue: new Date(2010, 0, i), YValue: value1 };
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
series1.push(point1);
series2.push(point2);
}
data = { Open: series1, Close: series2 };
return data;
}
</script>
```

The following screenshot illustrates the RangeNavigator that is populated with data using DataSource property in series.



Behavior Customization

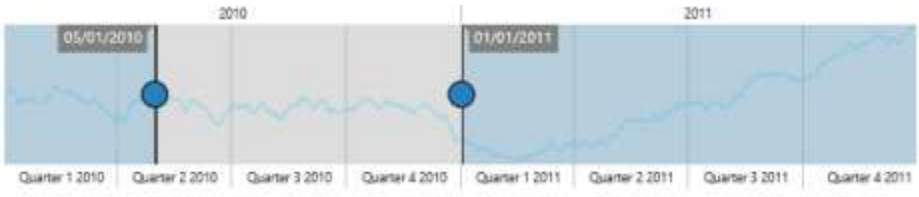
RangeNavigator allows you to customize the control using events. You can change the range for selected data of the **RangeNavigator** using events.

Deferred update

If you set EnableDeferredUpdate to true, the RangeChanged event gets fired after dragging and dropping the slider. By default the EnableDeferredUpdate is true. If EnableDeferredUpdate is false then the RangeChanged event gets fired while dragging the slider.

ASPX-CS

```
[
@(Html.EJ().RangeNavigator("container")
. EnableDeferredUpdate (true)
)
```



Deferred update

Methods

Destroy

This method is used to destroy the **RangeNavigator** widget.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
)
<script type="text/javascript">
var obj = $("#container").data("ejRangeNavigator");
```

```
obj.destroy(); // destroy the range navigator
</script>
```

Handle Events

Load

This event is fired when **RangeNavigator** is loading. A parameter **sender** is passed to the handler. Using **sender.model**, you can access the RangeNavigator properties.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Load("loadingData")
)
<script type="text/javascript">
function loadingData(sender)
{
sender.model.allowSnapping = false;
}
</script>
```

Loaded

This event is handled when the **RangeNavigator** gets loaded. A parameter **sender** is passed to the handler. Using **sender.model**, you can access the RangeNavigator properties.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Loaded("loadingData")
)
<script type="text/javascript">
function loadingData(sender)
{
sender.model.isResponsive = false;
}
</script>
```

RangeChanged

This event gets fired whenever the selected range changes in **RangeNavigator**. A parameter **sender** is passed to the handler. Using **sender.selectedRangeSettings**, you can access the start and end value of range for the selected region.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.RangeChanged("loadingData")
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.selectedRangeSettings.start);
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

SelectedRangeStart

This event is fired when starting to change the slider position in **RangeNavigator**. A parameter **sender** is passed to the handler. Using **sender.selectedRangeSettings**, you can access the start value of range for the selected region.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.SelectedRangeStart("loadingData")
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.selectedRangeSettings.start);
}
</script>
```

SelectedRangeEnd

This event is fired when selection ends in **RangeNavigator**. A parameter **sender** is passed to the handler. Using sender.selectedRangeSettings, you can access the end value of range for the selected region.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.SelectedRangeEnd("loadingData")
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.selectedRangeSettings.end);
}
</script>
```

ScrollStart

This event is fired when starting to change the scrollbar position of **RangeNavigator**. A parameter **sender** is passed to the handler. Using sender.data startRange and sender.data endRange, you can access the scrollbar position starting and ending range value on changed scrollbar.

JAVASCRIPT

```
@(Html.EJ().RangeNavigator("container")
.ScrollStart(loadingData)
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.type);
}
</script>
```

ScrollEnd

This event is fired while ending the change in scrollbar position of **RangeNavigator**. A parameter **sender** is passed to the handler. Using data oldRange and data newRange, you can access the scrollbar position old and new range values on changing scrollbar.

Copyright © 2001 - 2020 Syncfusion Inc.

JAVASCRIPT

```
@(Html.EJ().RangeNavigator("container")
.ScrollEnd(loadingData)
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.type);
}
</script>
```

ScrollChanged

This event is fired when changing the scrollbar position of **RangeNavigator**. A parameter **sender** is passed to the handler. Using data oldRange and data newRange, you can access the old and new range values of changed scrollbar position.

JAVASCRIPT

```
@(Html.EJ().RangeNavigator("container")
.ScrollChanged(loadingData)
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.type);
}
</script>
```

Click

Click event is handled when the **RangeNavigator** gets clicked. A parameter **sender** is passed to the handler. Using **sender.model**, you can access the RangeNavigator properties.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Click(loadingData)
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.type);
}
</script>
```

DoubleClick

DoubleClick event is handled when the **RangeNavigator** gets clicked. A parameter **sender** is passed to the handler. Using **sender.model**, you can access the RangeNavigator properties.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.DoubleClick(loadingData)
)
<script type="text/javascript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function loadingData(sender)
{
console.log(sender.type);
}
</script>
```

RightClick

RightClick event is handled when the **RangeNavigator** gets clicked. A parameter **sender** is passed to the handler. Using **sender.model**, you can access the RangeNavigator properties.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.RightClick(loadingData)
)
<script type="text/javascript">
function loadingData(sender)
{
console.log(sender.type);
}
</script>
```

Use of ZoomCoordinates

RangeNavigator is used along with the controls like chart and grid to view the selected data. To update chart/grid, whenever the selected range changes in **RangeNavigator**, you can use RangeChanged event of RangeNavigator and then update the chart/grid with the selected data in this event.

You can easily update the data for chart by assigning the ZoomFactor and ZoomPosition of the RangeNavigator to the chart axis.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.RangeChanged("loadingData")
)
<script type="text/javascript">
// setting zoom factor and position for chart axis in rangeChanged event.
function loadingData(sender)
{
var chart = $("#container").data("ejChart");
if (chart != null)
{
chart.model.axes[0].zoomPosition = sender. zoomPosition;
chart.model.axes[0].zoomFactor = sender. zoomFactor;
}
$("#container").ejChart("redraw");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Thumb Template

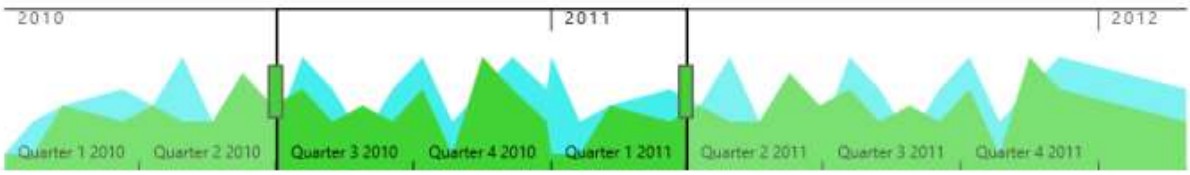
You can customize Thumb template by using `LeftThumbTemplate` and `RightThumbTemplate` property. You can add the required template as a “div” element with an “id” to the web page and assign the id or assign the HTML string to this property under `NavigatorStyleSettings`.

ASPX-CS

```
<script type="text/x-jsrender" id="left" >
<svg height="24" width="32" style="fill:#DD4A4A;stroke:black;">
<path d="M2 2 L2 22 L22 22 L32 12 L22 2 Z" />
</svg>
</script>
<script type="text/x-jsrender" id="right">
<svg height="24" width="32" style="fill:#DD4A4A;stroke:black; ">
<path d="M2 12 L12 22 L32 22 L32 2 L12 2 Z" />
</svg>
</script>
@(Html.EJ().RangeNavigator("container")
// ...
.NavigatorStyleSettings(navigator=>navigator.LeftThumbTemplate("left")
.RightThumbTemplate("right"))
// ...
)
```

The following screenshot displays the **RangeNavigator** using thumb template.

Copyright © 2001 - 2020 Syncfusion Inc.



Value Axis Settings

You can customize the line, Font Size, gridline, tickline, range, RangePadding and visibility of **RangeNavigator** axis.

To enable the visibility of axis line, you need to set Visible property of AxisLine in ValueAxisSettings.

You can customize the axis range by specifying Min, Max and Interval for Range property.

The MajorGridLines can be enabled by specifying Visible property. The Size, Width and Visible property of MajorTickLines is used to customize the axis tick lines.

The visibility of ValueAxisSettings is enabled by setting Visible property as true.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
// ...
.ValueAxisSettings(value=>value
.AxisLine(axis=>axis.Visible(true))
.Font(f=>f.Size('12px'))
.MajorGridLines(grid=>grid.Visible(true))
.MajorTickLines(tick=>tick.Size(3).Visible(false).Width(3))
.Range(range=>range.Min(0).Max(100).Interval(10))
.RangePadding(RangePadding.Normal),
.Visible(true)
)
// ...
)
```

Selected Range Settings

The start and end range values of selected range can be customized using Start and End property of SelectedRangeSettings.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
// ...
.SelectedRangeSettings(range=>range.Start("01/05/1992").End("01/05/1993")
)
// ...
)
```

Appearance and Styling

RangeNavigator is enriched with lots of customization options for labels, gridlines and slider to develop high quality graphic rich control.

Copyright © 2001 - 2020 Syncfusion Inc.

Customize labels

The labels are found along the range, displaying the value of the data it correspond, both on (higher level label) and below (lower level label) the RangeNavigator. **RangeNavigator** labels are further customized using higher level label Style property of Font and lower level label Style property of Font property in LabelSettings.

The higher level labels font Color, FontFamily, FontStyle, FontWeight, Opacity and Size can be customized using HigherLevel property.

The lower level labels font Color, FontFamily, FontStyle, FontWeight, Opacity and Size can be customized using LowerLevel property.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.LabelSettings(ls=>ls
// customizing higher level labels.
.HigherLevel(higher =>higher
.Style(style => style
.Font(f =>f
.Color("#ff0000")
.Opacity(1)
.Size("12px")
.Style(RangeNavigatorFontStyle.Normal)
.Weight(RangeNavigatorFontWeight.Regular)
)
)
)
// customizing lower level labels.
.LowerLevel(ll => ll
. Style(style => style
.Font(f => f
.Color("#ff0000")
.Opacity(1)
.Size("12px")
.Style(RangeNavigatorFontStyle.Normal)
.Weight(RangeNavigatorFontWeight.Regular)
)
)
)
))
```



Customize labels

Label Placement:

Labels in **RangeNavigator** are placed inside or outside of the control. You can customize both the higher and lower level labels using LabelPlacement property in label setting of RangeNavigator. By default LabelPlacement is “outside” for the both higher and lower level labels.

The following screen shot illustrates both the lower and higher level labels that are placed outside the control with LabelPlacement specified as outside.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.LabelSettings(ls=>ls
.HigherLevel(higher =>higher.LabelPlacement("inside"))
.LowerLevel(ll => ll.LabelPlacement("inside"))
)
)
```

The following screenshot illustrates a **RangeNavigator** with labels inside the control after specifying the LabelPlacement as inside.



Customize RangeNavigator

Customize NavigatorStyleSettings

RangeNavigator is customized using NavigatorStyleSettings properties. You can customize the selected and unselected region color using SelectedRegionColor and UnselectedRegionColor, SelectedRegionOpacity and UnselectedRegionOpacity in **NavigatorStyleSettings** and the thumb of the slider using ThumbColor, ThumbRadius and ThumbStroke in NavigatorStyleSettings. MajorGridLineStyle and MinorGridLineStyle are used to customize the major grid line Color, Visible property and minor gridline Color and Visible. You can customize the Background, Opacity and Border Color, DashArray and Width of navigatorStyleSettings.

Customize Labels

The visibility of labels are enabled by setting Visible in higher level and Visible in lower level. The labels can be aligned by specifying HorizontalAlignment of higher level style and HorizontalAlignment of lower level style.

You can customize the Border Color and Width, Fill, GridLineStyle Color, DashArray and Width, Position property of higher level labels in labelSettings.

You can also customize the Border Color and Width, Fill, GridLineStyle Color, DashArray and Width, Position property for lower level labels of labelSettings.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
//  To customize the navigator element
.NavigatorStyleSettings (navigator=>navigator
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Border(br=>br.Color("black").Width(3))
.Background("transparent")
.UnselectedRegionColor("white")
.SelectedRegionColor("#5EABDE")
.ThumbRadius(10)
.ThumbColor("white")
.MajorGridLineStyle(mr=>mr.Color("transparent").Visible(true))
.MinorGridLineStyle(mi=>mi.Color("transparent").Visible(true))
)
//  To customize the labels
.LabelSettings(ls=>ls
.HigherLevel(higher =>higher
.LabelPlacement("inside")
.IntervalType(NavigatorIntervalType.Years)
.Style(style => style
.HorizontalAlignment(HorizontalAlignment.Left)
.Font(f =>f
.Color("black")
.Opacity(1)
.Size("13px")
)
)
)
.LowerLevel(ll => ll
.LabelPlacement("inside")
.IntervalType(NavigatorIntervalType.Quarters)
.Style(style => style
.HorizontalAlignment(HorizontalAlignment.Center)
.Font(f => f
.Color("black")
.Opacity(1)
.Size("12px")
)
)
)
))
```



Themes

RangeNavigator Theme is a set of pre-defined options that are applied to the control before each RangeNavigator is instantiated. Following predefined themes are available in ASP.NET MVC RangeNavigator.

1. flatlight
2. flatdark
3. gradientlight

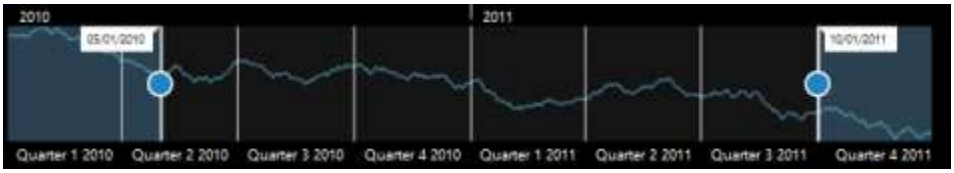
Copyright © 2001 - 2020 Syncfusion Inc.

4. gradientdark
5. azure
6. azuredark
7. lime
8. limedark
9. saffron
10. saffrondark
11. gradientazure
12. gradientazuredark
13. gradientlime
14. gradientlimedark
15. gradientsaffron

gradientsaffrondark

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Theme("azuredark")
)
```



Tooltip

RangeNavigator provides **Tooltip** support for sliders. Sliders are used to select data at particular range in the RangeNavigator control. Tooltips for sliders display the selected start and end **DateTime** values.

Customization

RangeNavigator provides support for you to customize the text display in the tooltip and background using TooltipSettings property. You can change font family, font color, font style, font weight. By default “Segoe UI” font family is set to tooltip text.

- Tooltip visibility can be enabled or disabled using Visible property.
- You can change background color of tooltip using BackgroundColor property.
- You can customize the Color, Family, FontStyle, Opacity, Size and Weight of tooltip text in Font property.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.TooltipSettings(tooltip=>tooltip
.Visible(true)
.BackgroundColor("Black")
//  To customize the tooltip text
.Font(font=>font
.Color("red")
. Family("Segoe UI")
.Style(RangeNavigatorFontStyle.Normal)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Size("12px")
.Opacity(1)
.Weight(RangeNavigatorFontWeight.Regular)
)
))
```



Label Format

By default, the **tooltip** texts are automatically determined based on the data points. To make it readable and understandable you can format the tooltip text. For **DateTime** data, all globalized format are supported. By default the LabelFormat is "MM/dd/yyyy".

Some of the LabelFormat for **DateTime** data are as follows:

- 'MMM, yyyy'
- 'dd, MMM'
- 'dd/MM/yyyy'
- 'dd, hh:mm'
- 'hh:mm:ss'

'hh:mm:ss:tt'

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.TooltipSettings(tooltip=>tooltip.LabelFormat("MMM, yyyy"))
)
```



Tooltip display mode

By default the **tooltip** for RangeNavigator gets displayed. You can change this behavior using the TooltipDisplayMode property in the TooltipSettings and it takes the following values.

Tooltip values

Value	Description
always	Tooltip get displayed for RangeNavigator always.

Copyright © 2001 - 2020 Syncfusion Inc.

onDemand	Tooltip get displayed only when we move the slider.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.TooltipSettings(tooltip=>tooltip.TooltipDisplayMode("onDemand"))
)
```



Globalization & Localization

RangeNavigator supports Localization and Globalization to customize the labels based on a culture specific to a country. The culture defines specific information for the number formats, week and month names, date and time formats etc.

Localization

The Locale property is used to specify the **Localization** of RangeNavigator. Localization is the process of customizing the user interface based on a culture specific to a particular country or region in order to display regional data. The culture is represented by a unique string, for example, —en-US‖ for U.S. English and —fr-FR‖ for French (common), this is achieved by creating a JavaScript file **"rangeNavigatorSource.fr-FR.js"** and setting the equivalent word as illustrated in the following code sample.

ASPX-CS

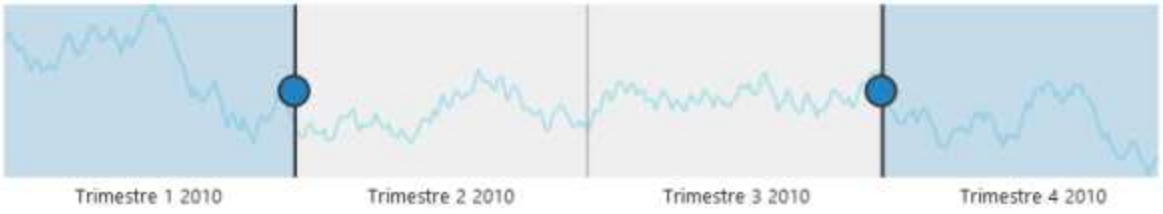
```
ej.datavisualization.RangeNavigator.locale["fr-FR"] =
{
intervals:
{
//string to display the intervals on RangeNavigator
quarter: {longQuarters: "Trimestre", shortQuarters: "T"},
week: { longWeeks: "Semaine", shortWeeks: "S" }
}
}
```

Localization is the key feature that provides solutions globally with the help of localized control.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.Locale("fr-FR")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

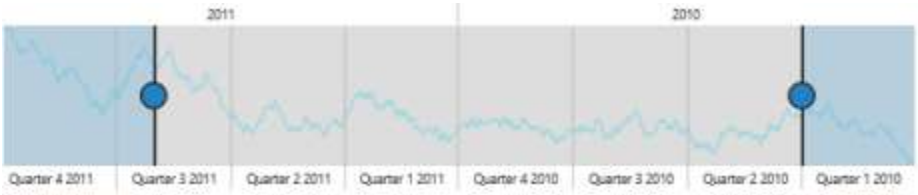


RTL

Right-to-Left or **RTL** describes the ability of application to handle and responds you to communicate with a right-to-left language, like Arabic or Japanese. EnableRTL property is used to change the rendering format to **"Right to Left"**, by default it renders from **"Left to Right"** in RangeNavigator.

ASPX-CS

```
@(Html.EJ().RangeNavigator("container")
.EnableRTL(true)
)
```



User Interactions

Highlight

EjRangeNavigator provides highlighting supports to the intervals on mouse hover. To enable the highlighting option, set the Enable property to true in the HighlightSettings of NavigatorStyleSettings.

ASPX-CS

```
@(Html.EJ().RangeNavigator("rangeContainer")
//...
.NavigatorStyleSettings(navigator=>
navigator.HighlightSettings(highlight=>
// enable the highlight settings
highlight.Enable(true))
)
)
```



Click here to view the highlight and selections online demo sample.

Customize the highlight style

To customize the highlighted intervals, use Color, Border and Opacity options in the HighlightSettings.To customize border of highlighted interval, use Color and Width options in Border.

ASPX-CS

```
@(Html.EJ().RangeNavigator("rangeContainer")
//...
.NavigatorStyleSettings(navigator=>
navigator.HighlightSettings(highlight=>
// enable the highlight settings
highlight.Enable(true)
// customizing style
.Color("#006fa0")
.Border(br=>br.Color("red").Width(2))
)
)
)
```



Selection

EjRangeNavigator provides selection supports to the intervals by, clicking and dragging the highlighted intervals. To enable the selection option, set the Enable property to true in the SelectionSettings.

ASPX-CS

```
@(Html.EJ().RangeNavigator("rangeContainer")
//...
.NavigatorStyleSettings(navigator=>
navigator.SelectionSettings(selection=>
// enable the selection settings
selection.Enable(true)
)
)
)
```



Click here to view the highlight and selections online demo sample.

Customize the selection style

To customize the selected intervals, use Color, Border and Opacity options in the SelectionSettings. To customize border of selected interval, useColor and Width options in Border.

ASPX-CS

```
@(Html.EJ().RangeNavigator("rangeContainer")
//...
.NavigatorStyleSettings(navigator=>
navigator.SelectionSettings(selection=>
```

```
// enable the selection settings
selection.Enable(true)
// customizing style
.Color("#27e8e5’")
.Border(br=>br.Color("red").Width(2))
)
)
)
```



Scrollbar

- To render the Scrollbar in RangeNavigator, you need to enable EnableScrollbar option.
- ScrollRangeSettings of rangenavigator Start and End value is used to set the minimum and maximum datasource value to be added in the rangenavigator.
- Based on the ScrollRangeSettings *Start, End* value and dataSource *Start, End* value scrollbar will be adjust.
- When you change the scrollbar position, ScrollEnd event returns the current position of start and end range value.

ASPX-CS

```
@(Html.EJ().RangeNavigator("rangeContainer")
//...
//Enable scrollbar option in the rangenavigator
.EnableScrollbar(true)
//Maximum data to be displayed in the rangenavigator control
.ScrollRangeSettings(range=>
range.Start("2010/1/1").End("2011/11/31")
)
//Subscribe the event on scrollbar position changed
.ScrollEnd("onScrollbarChange")
)
```

JS

```
function onScrollbarChange(sender) {
var start  = sender.data.newRange.start;
var end  = sender.data.newRange.end;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view scrollbar online demo sample.

Rating

Rating

Essential Studio ASP.NET MVC Rating control provides an intuitive Rating experience that enables you to select a number of stars representing a Rating. You can configure the item size, orientation and the number of displayed items in the Rating control. You can also customize the Rating star image by using custom CSS.

Key Features

Some important features of the Rating control are as follows:

- Precision support— You can set three different precisions like full, half, and exact.
- Orientation support—Rating control can be displayed in horizontal or vertical direction.
- Read-only mode—Rating control can be rendered in the read-only mode to enable or disable user interaction.
- Reset—You can reset the Rating value by using the Reset button.
- Tooltip—You can set a tooltip to display the currently selected Rating value.
- Theme—Rating control consists of 12 built-in themes (6 – metro and 6 – gradient effects), and also supports custom skins to set user-defined themes.

Getting Started

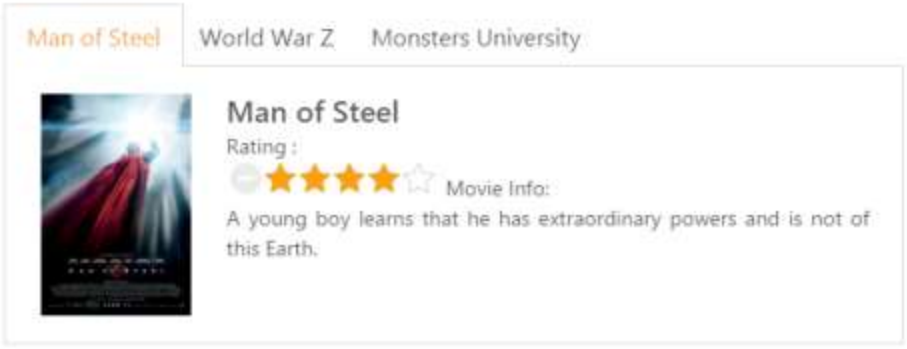
This section explains briefly how to create a Rating control in the ASP.NET MVC.

Create your first Rating control in MVC

Essential Studio ASP.NET MVCRating control provides support to display Rating bar within your web page, and allows you to rate the products. Refer to the following guidelines to customize the Rating control for a real-time movie download application. You can use Rating control to rate a movie. In this real-time application, Essential Studio ASP.NET MVCTab control is used to display the movies information with rating star.To know more about Tab control, refer to the Getting Started of the Tab control.

The following screenshot demonstrates the functionality of a Rating control with a Rating range of 0 to 5.

Copyright © 2001 - 2020 Syncfusion Inc.



In the above screenshot, you can rate the movie by selecting a corresponding movie.

Create a Rating

The Essential Studio ASP.NET MVCRating widget renders with built-in features like Precision, Orientation and flexible APIs. You can easily create the Rating widget by using HTML helper class as follows.

You can create an MVC Project and add the necessary assemblies, styles and scripts to it.

Refer to MVC-Getting Started.

Add the following code example to the corresponding view page to render Rating inside the Tab control.

ASPX-CS

```
<div class="frame">
@{Html.EJ().Tab("moviesTab").Items(event=>
{
event.Add().ID("steel").Text("Man of Steel").ContentTemplate(
@<div>
<table>
<tr>
<td class="movies-img" valign="top">
<img src="@Url.Content("~/Images/rating/mos.png")" alt="mos" />
</td>
<td valign="top">
<div>
<span class="movie-header">Man of Steel</span><br />
Rating :
<br />
@Html.EJ().Rating("univRating").Value(4)
<span>Movie Info:</span>
<p>
A young boy learns that he has extraordinary powers and is not of this
Earth.
</p>
</div>
</td>
</tr>
</table>
</div>);
event.Add().ID("war").Text("World War Z").ContentTemplate(
@<div>
<table>
<tr>
```

```
<td class="movies-img" valign="top">
<img src="@Url.Content("~/Images/rating/wwz.png")" alt="mos" />
</td>
<td valign="top">
<div>
<span class="movie-header">World War Z</span><br />
Rating :
<br />
@Html.EJ().Rating("wwzRating"). Value(4)
<span>Movie Info:</span>
<p>
The story revolves around United Nations employee Gerry Lane (Pitt).
</p>
</div>
</td>
</tr>
</table>
</div>);
event.Add().ID("university").Text("Monsters University").ContentTemplate(
@<div>
<table>
<tr>
<td class="movies-img" valign="top">
<img src="@Url.Content("~/Images/rating/mu.png")" alt="mos" />
</td>
<td valign="top">
<div>
<span class="movie-header">Monsters University</span><br />
Rating :
<br />
@Html.EJ().Rating("mosRating").Value(3)
<span>Movie Info:</span>
<p>
Mike Wazowski and James P. Sullivan are an inseparable pair, but that
wasn't always the case.
</p>
</div>
</td>
</tr>
</table>
</div>);
}).Render();
}
</div>
```

Add the following styles to the corresponding view page to show the Rating in a horizontal order.

CSS

```
<style type="text/css" class="cssStyles">
.movies-img
{
width: 125px;
}
.movie-header
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
font-size: 20px;
font-weight: 600;
}
.frame
{
width: 600px;
height: 250px;
}
</style>
```

Execute the above code to render the following output.



Note: Add necessary images to the mentioned directory. <project directory>/Images/rating/yourimage.png

Rating Customization

Setting Value

The Value property sets the display value of the Rating. For example, when the Value property is set to 4, the Rating control renders 4 ratings. By default, Value property's value is one.

The following code example is used to render the Rating control with the customized value.

Add the following code in your view page to render the Rating with the customized value.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized value.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").Value(4)
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating with custom defined value.

Copyright © 2001 - 2020 Syncfusion Inc.



Min Value

Rating control provides support for setting the MinimumValue. This is achieved by adding the MinValue property. When the MinValue property is set, the Rating value starts with MinValue+1.

The following code example is used to render the Rating control with minimum rating value specified.

Add the following code in your view page to render the Rating with Minimum value.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized minimum value.
<div id="container" style="width: 300px; padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").MinValue(3)
</td>
</tr>
</table>
</div>
```

The following screenshot Rating value starts with 4 since MinValue value is set as 3.



Rating with minimum value as 4

Max Value

Rating control provides support for setting Maximum value. This is achieved by adding the MaxValue property. By default, MaxValue is five.

The following code example is used to render the Rating control with maximum rating value specified

Add the following code in your view page to render the Rating with maximum value.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized maximum value.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").MaxValue(10)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating with maximum value.



Set Precision

You can set Precision in Rating value between two whole numbers such as 2.5 or 3.7 and it is done by using the property Precision by changing the value to Precision.Half or Precision.Exact. By default, value of Precision is Full.

The following code example is used to render the Rating control with Precision.

Add the following code in your view page to render the Rating with Precision.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with Precision.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("FullRating").Precision(Precision.Full).Value(4)
</td>
</tr>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("HalfRating").Precision(Precision.Half).Value(3.5)
</td>
</tr>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("ExactRating").Precision(Precision.Exact).Value(4.7)
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating with Precision.

Copyright © 2001 - 2020 Syncfusion Inc.



Increment Step

Rating control supports customized increment value for Rating. This is achieved by adding the IncrementStep property.

The following code example is used to render the Rating control with customized increment.

Add the following code in your view page to render the Rating with customized increment.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized increment.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").IncrementStep(2).MaxValue(10)
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating with customized increment.



Resetting values

Rating control provides support for value reset at runtime. This is achieved by enabling the AllowReset property to true. By default, the property value is set to true.

The following code example is used to render the Rating control with AllowReset.

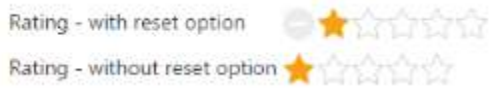
Add the following code in your view page to render the Rating with AllowReset.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
// Add the following code example to the corresponding CSHTML page to
render Rating with allowReset.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("ResetRating").AllowReset(true)
</td>
</tr>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").AllowReset(false)
</td>
</tr>
</table>
</div>
```

The following illustrates the Rating with allowReset.



Read only

Rating control provides support for changeable or unchangeable values for Rating control. This is achieved by the ReadOnly property. When this property is set to true, the Rating value becomes unchangeable. By default, this property value is set to false.

The following code example is used to render the Rating control with ReadOnly.

Add the following code in your view page to render Rating with readOnly.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with readOnly.
<div id="container" style="width: 300px; padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").ReadOnly(true).Value(2).MaxValue(10)
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating with ReadOnly.

Copyright © 2001 - 2020 Syncfusion Inc.



Enable or Disable

Rating control provides support to Enable or Disable the control. This is achieved by the Enabled property. By default the property value is true.

The following code example is used to render the Rating control with Enable or Disable support.

Add the following code in your view page to render the Rating with enable or disable support.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with Enable/Disable support.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").Enabled(false)
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating in a disabled form.



Appearance and Styling

Show ToolTip

Rating control provides support for Tooltip values. This is achieved by enabling the ShowTooltip property to true. When you move the mouse over the Rating control, it displays the Tooltip value as rating value. By default, this property value is set to true.

The following code example is used to render the Rating control without tooltip.

1. Add the following code in your view page to render Rating with Tooltip.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating without Tooltip
<div id="container" style="width: 300px; padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").ShowTooltip(false)
</td>
</tr>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</table>
</div>
```

The following screenshot illustrates Rating without Tooltip.



Adjusting Rating Size

Adjust Shape Width and Shape Height

You can customize the width and height of the Rating by ShapeWidth and ShapeHeight properties. These properties completely depend on rating image's size. The ShapeWidth and ShapeHeight are adjusted within the rating image size.

The following code example is used to render the Rating control with customized ShapeWidth and ShapeHeight.

In this sample, the default rating star is replaced by crystal-stars.png local image by overriding the Rating control CSS.

Add the following code in your view page to render the Rating with customized ShapeWidth and ShapeHeight.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized ShapeWidth and ShapeHeight.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").Value(4).ShapeWidth(29).ShapeHeight(29)
</td>
</tr>
</table>
</div>
```

Add the following styles.

CSS

```
<style type="text/css">
.e-rating
{
margin-top: -7px;
}
.e-rating.e-horizontal .e-shape-list, .e-rating.e-vertical .e-shape-list,
.e-rating.e-horizontal .e-shape, .e-rating.e-vertical .e-shape, .e-
rating.e-horizontal .e-ul,.e-rating.e-vertical .e-ul,.e-rating.e-
horizontal .e-reset, .e-rating.e-vertical .e-reset
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
height:28px;width:28px;
background:url(images/crystal-stars.png) no-repeat;
}
.e-rating.e-horizontal .e-reset, .e-rating.e-vertical .e-reset
{
background-position: 0 42px;
margin-left: 2px;
}
.e-rating.e-horizontal .e-shape-list
{
background-position: 0 -56px;
}
.e-rating .e-shape.inactive
{
background-position: 0 -56px;
}
.e-rating .e-shape.active
{
background-position: 0 -112px;
}
.e-rating .e-shape.selected
{
background-position: 0 -84px;
}
</style>
```

The following screenshot illustrates Rating with customized ShapeWidth and ShapeHeight.



Theme

Rating control's style and appearance are controlled based on CSS classes. In order to apply styles to the Rating control, refer to 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. When the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these both.

By default, there are 12 themes support available for Rating control namely:

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron

Copyright © 2001 - 2020 Syncfusion Inc.

- gradient-saffron-dark

Custom styles

The style of the Rating control is customized by CssClass property.

The following code example is used to render the Rating control with customized style.

Add the following HTML to render the Rating with customized style. In this, hovering behaviors, tooltip position and display have been customized by using custom CSS class.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized styles.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").CssClass("custom")
</td>
</tr>
</table>
</div>
```

Add the following styles.

CSS

```
<style type="text/css">
.customCss
{
margin-top: -7px;
}
.customCss.e-horizontal .e-shape-list, .e-rating.e-vertical .e-shape-
list,
.customCss.e-horizontal .e-shape, .e-rating.e-vertical .e-shape, .e-
rating.e-horizontal .e-ul, .customCss.e-vertical .e-ul, .e-rating.e-
horizontal .e-reset, .e-rating.e-vertical .e-reset
{
height: 28px;
width: 28px;
background: url(../../Content/ej/default-theme/images/crystal-stars.png)
no-repeat repeat;
}
.customCss.e-horizontal .e-reset, .customCss.e-vertical .e-reset
{
background-position: -2px 113px;
margin-left: 2px;
}
.customCss.e-horizontal .e-shape-list
{
background-position: 0 -56px;
}
.customCss.e-horizontal .e-reset:hover
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
background-position: 0 42px;
}
.customCss .e-shape.inactive
{
background-position: 0 -84px;
}
.customCss .e-shape.active
{
background-position: 0 -112px;
}
.customCss .e-shape.selected
{
background-position: 0 -84px;
}
.customCss.e-tooltip
{
background-color: white;
border: 2px solid #b0c4de;
color: black;
}
</style>
```

The following screenshot illustrates the Rating with customized style.



Orientation

Rating provides support for Vertical orientation. By default Rating renders with Horizontal orientation. You can change the orientation by the Orientation property.

The following code example is used to render the Rating control with Vertical orientation.

Add the following code in your view page to render the Rating with customizedOrientation.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Rating with customized orientation.
<div id="container" style="border: 1px solid black; width: 300px;
padding: 2px">
<table>
<tr>
<td valign="top">Rating:
</td>
<td>
@Html.EJ().Rating("Rating").Orientation(Orientation.Vertical)
</td>
</tr>
</table>
</div>
```

The following screenshot illustrates the Rating with Vertical orientation.

Copyright © 2001 - 2020 Syncfusion Inc.



How To

Render Rating from code behind

Rating can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of Rating using Rating properties from code behind.

RAZOR

```
@{Html.EJ().Rating("univRating", Model).Render();}
```

C#

```
public ActionResult Index()
{
RatingProperties Obj = new RatingProperties();
Obj.Value = 4;
Obj.Orientation = Syncfusion.JavaScript.Orientation.Vertical;
return View(Obj);
}
```

ReportViewer

ReportViewer

The ReportViewer is a visualization control to view Microsoft SSRS RDL/RDLC files on a web page and it is powered by HTML5/JavaScript. It has support to bind DataSources/Parameters to the Reports and also supports exporting, paging, zooming and printing the report.

Key Features

The ReportViewer supports the following features.

- DataSources (SQL Server, Oracle, MS Azure, XML, Business Object and SQL Server Compact Data Sources)
- Filters and Parameters
- Built-in Fields and Expressions
- Grouping and Sorting
- Report Items
- Table
- Matrix
- List
- Chart
- Sparkline
- DataBar

Copyright © 2001 - 2020 Syncfusion Inc.

- Map
- Gauge
- Indicator
- Image
- Textbox
- Line
- Rectangle
- Subreport
- Actions: Drilldown, Drill through, Toggle and Document Map
- Printing, Exporting, Paging, FitToPage and Zooming
- Report Processing Events
- KnockOut/AngularJS Support
- Toolbar Customization

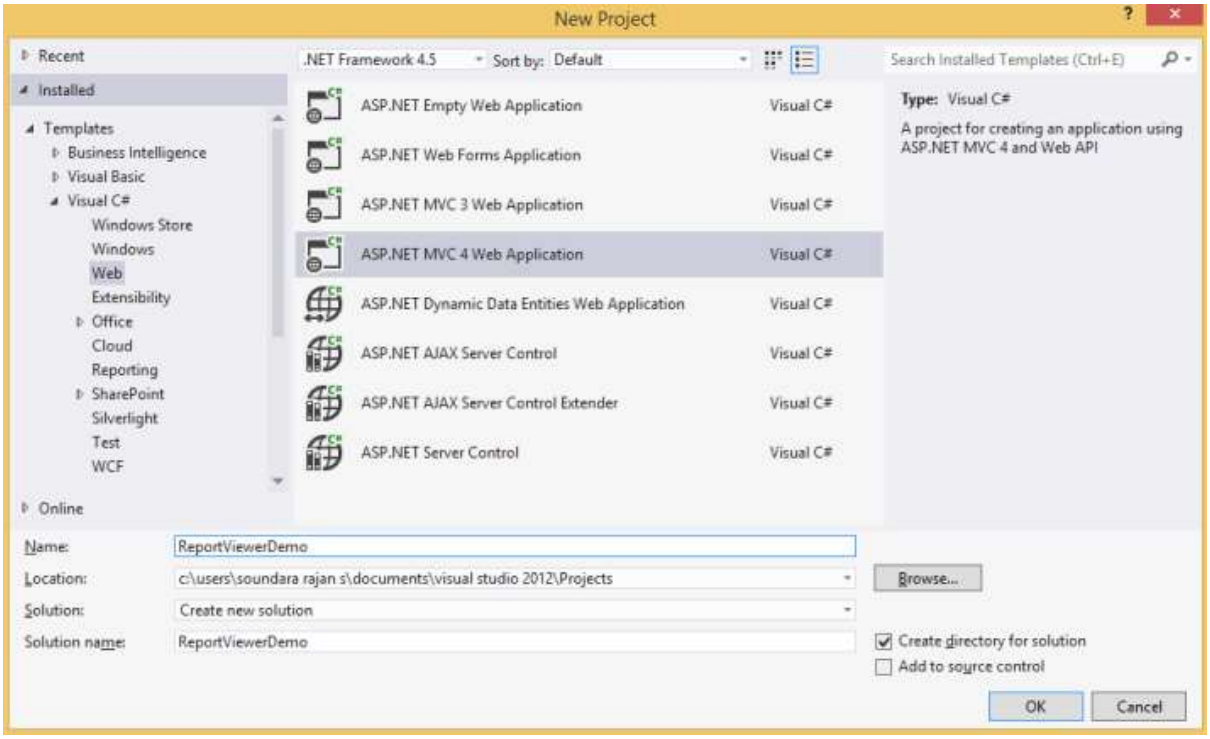
Getting Started

This section explains briefly about how to create a ReportViewer in your web application with MVC.

Create your first ReportViewer in MVC

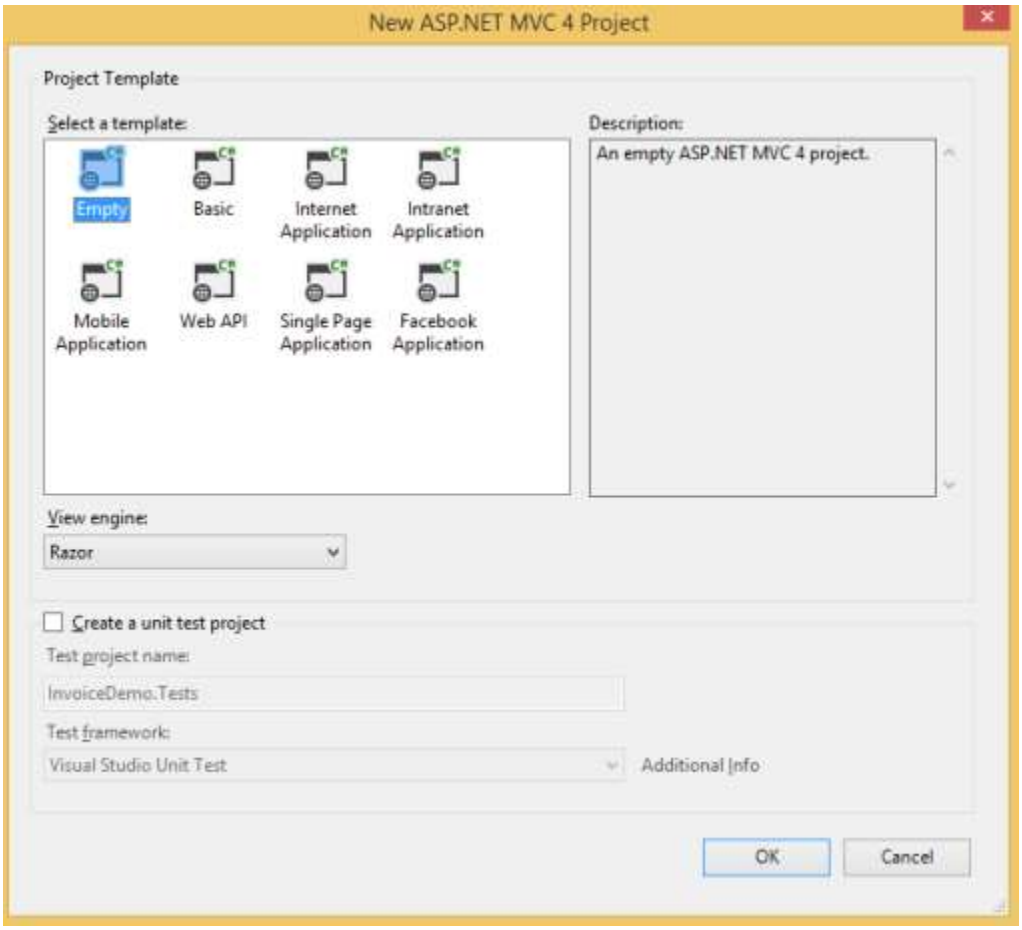
This section explains how to configure a ReportViewer component in web application. As ReportViewer uses WebApi to process the report file, you can also learn how to create WebApi Service to process the report for ReportViewer. In the following example, the ReportViewer component displays the Sales Dashboard Report.

Open Visual Studio and create a new project by clicking New Project. Select the Web category, select the ASP.NET MVC4 Web Application template, and then click OK. The following screenshot displays the Project Creation Wizard.



Project Creation Wizard

The following screenshot displays how to select Internet application template with razor view engine.

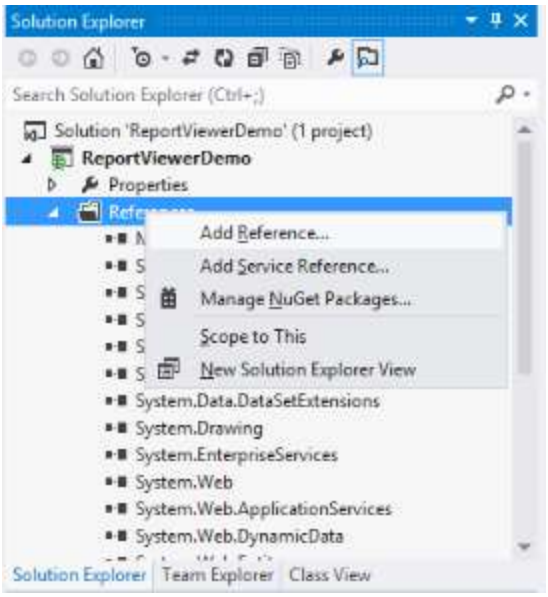


Add References, Scripts, Styles and Control in CSHTML Page

Add References

1. In the Solution Explorer, right-click the References folder and then click Add Reference

Copyright © 2001 - 2020 Syncfusion Inc.



Adding Reference

2. Add the following assemblies
    - System.Web.Routing
    - System.Web.Http
    - System.Web.WebHost
    - System.Net.Http
    - System.Net.Http.WebRequest
    - System.Net.Http.Formatting
    - Syncfusion.Linq.Base
    - Syncfusion.EJ
    - Syncfusion.EJ.MVC
    - Syncfusion.EJ.ReportViewer
    - Syncfusion.Pdf.Base
    - Syncfusion.XlsIO.Base
    - Syncfusion.DocIO.Base
    - Syncfusion.Shared.Wpf
    - Syncfusion.Chart.Wpf
    - Syncfusion.Gauge.Wpf
    - Syncfusion.SfMaps.Wpf

Note: Refer the above assemblies from the installed location, C:\Program Files (x86)\Syncfusion\Essential Studio\ASP.NET MVC\{{ site.releaseversion }}\Assemblies

Note:

Note: Refer System.Web.Http, System.Web.WebHost, System.Net.Http.WebRequest and System.Net.Http.Formatting assemblies from ASP.NET WebApi NuGet package.

3. Click OK

Copyright © 2001 - 2020 Syncfusion Inc.

Add Scripts and Styles

Add the script files and CSS files in the <title> tag of the _Layout.cshtml page.

Note: Use the following code example while adding scripts and styles.

ASPX-CS

```
<link rel="stylesheet" href="https://cdn.syncfusion.com/{{
site.releaseversion }}/js/web/flat-azure/ej.web.all.min.css" />
<script src="https://code.jquery.com/jquery-1.10.2.min.js"
type="text/javascript"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-
easing/1.3/jquery.easing.min.js" type="text/javascript"></script>
<script src="https://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"> </script>
<script src="https://cdn.syncfusion.com/{{ site.releaseversion
}}/js/common/ej.unobtrusive.min.js"></script>
```

Add Control in View page

Add the following code example in the Index.cshtml page that is already created. Set the desired ReportPath and ReportServiceUrl to ReportViewer.

ASPX-CS

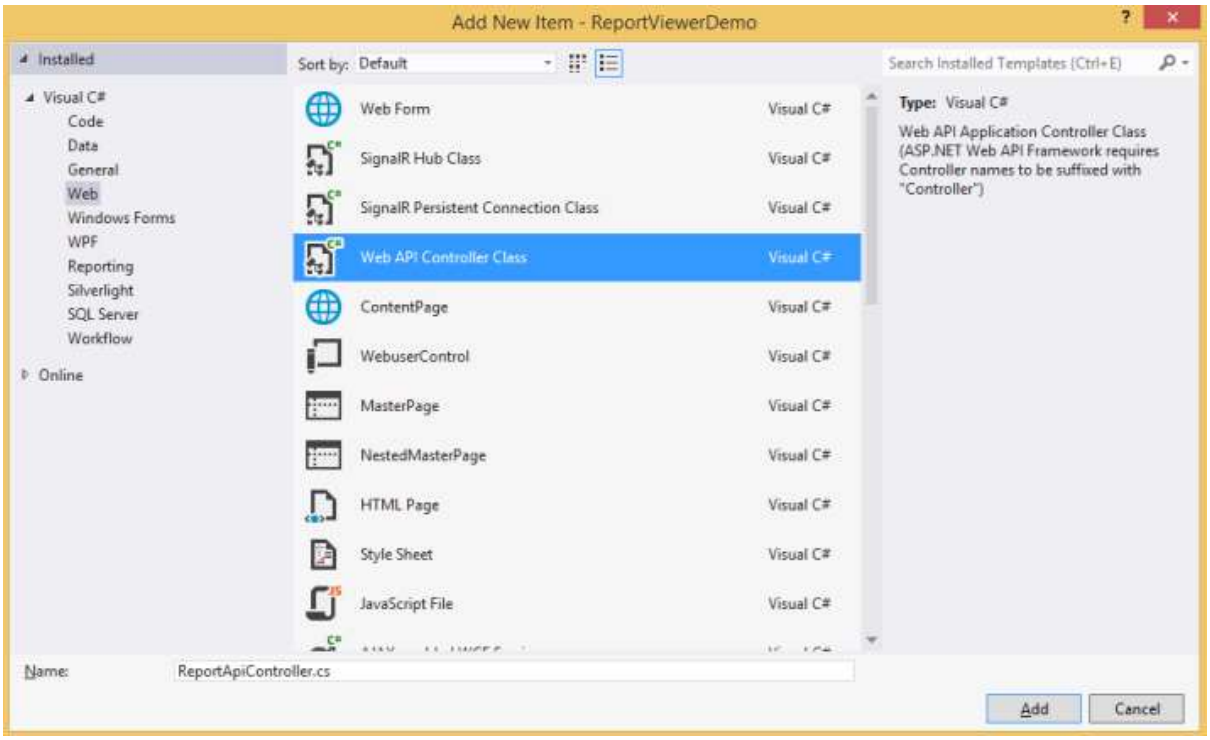
```
@using Syncfusion.MVC.EJ
@using Syncfusion.JavaScript.Models
@using Syncfusion.JavaScript
@using Syncfusion.JavaScript.ReportViewer
@{
ViewBag.Title = "Index";
}
<h2>Index</h2>
<div>
@{Html.EJ().ReportViewer("viewer").ReportPath("~/App_Data/Sales
Dashboard.rdl").ReportServiceUrl("/api/ReportApi").Render();}
</div>
```

Note: Add your report files to your application's App_Data folder. You can obtain sample rdl/rdlc files from Syncfusion installed location(%userprofile%\AppData\Local\Syncfusion\EssentialStudio\{{ site.releaseversion }}\Common\Data\ejReportTemplate). {{ site.releaseversion }} is the Essential Studio Release Version.

Add WebAPI controller for ReportViewer

The MVC ReportViewer uses WebApi services to process the report file and process the request from control.

Copyright © 2001 - 2020 Syncfusion Inc.



Adding WebApi Controller

Inherit IReportController

The ApiController inherits the IReportController and you can add the following code example to its methods definition in order to process the report file. The interface IReportController contains the required actions and helper methods declaration to process the report. The ReportHelper class contains helper methods that helps to process Post/Get request from control and return the response to control.

C#

```
using Syncfusion.EJ.ReportViewer;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
namespace ReportViewerDemo.Api
{
public class ReportApiController : ApiController, IReportController
{
//Post action for processing the rdl/rdlc report
public object PostReportAction(Dictionary<string, object> jsonResult)
{
return ReportHelper.ProcessReport(jsonResult, this);
}
//Get action for getting resources from the report
[System.Web.Http.ActionName("GetResource")]
[AcceptVerbs("GET")]
public object GetResource(string key, string resourcetype, bool isPrint)
{
return ReportHelper.GetResource(key, resourcetype, isPrint);
```

```
}
//Method will be called when initialize the report options before start
processing the report
public void OnInitReportOptions(ReportViewerOptions reportOption)
{
//You can update report options here
}
//Method will be called when reported is loaded
public void OnReportLoaded(ReportViewerOptions reportOption)
{
//You can update report options here
}
}
}
```

WebAPI Routing

You can route the WebAPI in Application_Start event into Global.asax file as follows.

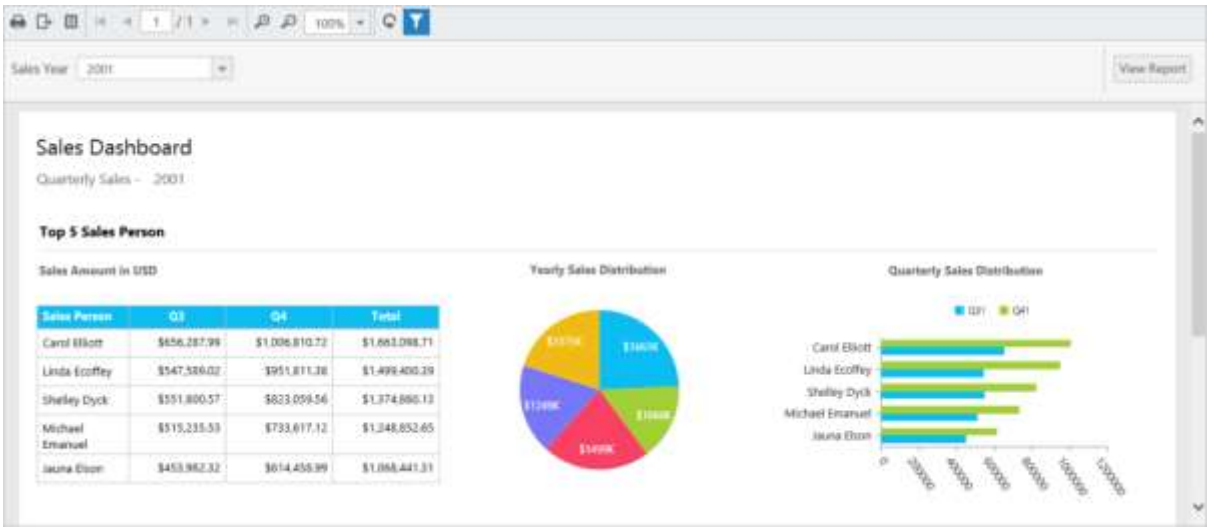
C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.SessionState;
using System.Web.Http;
namespace ReportViewerDemo
{
public class Global : System.Web.HttpApplication
{
protected void Application_Start(object sender, EventArgs e)
{
System.Web.Http.GlobalConfiguration.Configuration.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "api/{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional });
}
}
}
```

Run the Application

Run the sample application and you can see the ReportViewer on the page as displayed in the following screenshot.

Copyright © 2001 - 2020 Syncfusion Inc.



ReportViewer with Sales Dashboard Report

Load SSRS Server Reports

ReportViewer supports to load RDL/RDLC files from SSRS Server. The following steps help you to load reports from SSRS Server.

1. Set the ReportPath from SSRS and SSRS ReportServerUrl in the ReportViewer properties.

ASPX-CS

```
@using Syncfusion.MVC.EJ
@using Syncfusion.JavaScript.Models
@using Syncfusion.JavaScript
@using Syncfusion.JavaScript.ReportViewer
@{
ViewBag.Title = "Index";
}
<h2>Index</h2>
<div>
@(Html.EJ().ReportViewer("viewer").ProcessingMode(Syncfusion.JavaScript.R
eportViewerEnums.ProcessingMode.Remote)
.ReportServiceUrl("/api/ReportApi")
.ReportServerUrl("http://mvc.syncfusion.com/reportserver")
.ReportPath("/SSRSSamples2/Territory Sales new"))
</div>
```

2. Add the credential information in ReportApiController'sOnInitReportOptions method that is inherited in IReportController.

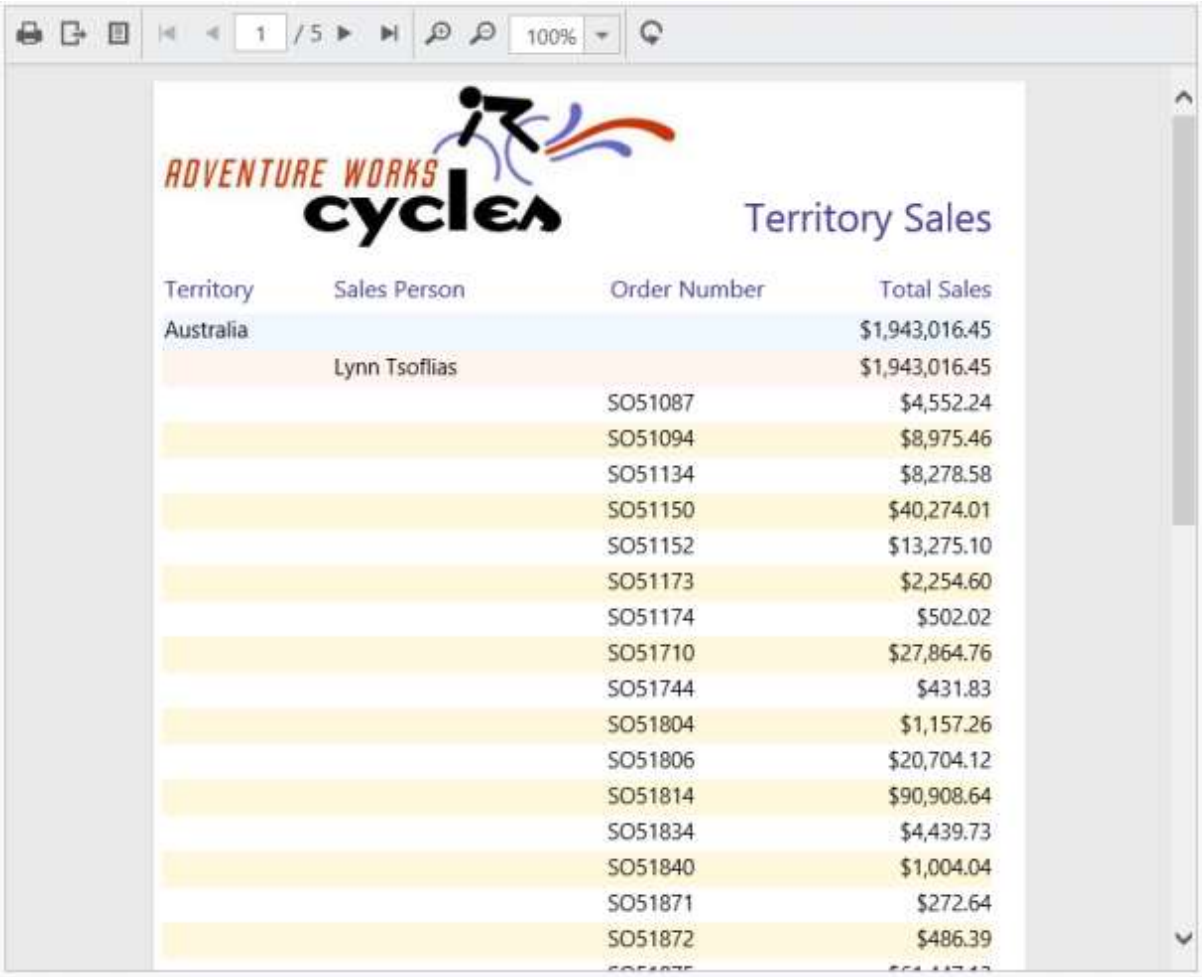
CSHARP

```
public void OnInitReportOptions(ReportViewerOptions reportOption)
{
//Add SSRS Server and database credentials here
reportOption.ReportModel.ReportServerCredential = new
System.Net.NetworkCredential("ssrs", "RDLReport1");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
reportOption.ReportModel.DataSourceCredentials.Add(new
DataSourceCredentials("AdventureWorks", "ssrs1", "RDLReport1"));
}
```

3. Run the application and you can see the ReportViewer on the page as displayed in the following screenshot.



Report from SSRS

Load RDLC Reports

The ReportViewer has data binding support to visualize the RDLC reports. The following code example helps you to bind data to ReportViewer.

1. Assign the RDLC report path to ReportViewer'sReportPath property and set the data sources to the ReportViewer'sDataSources property.

ASPX-CS

```
@(Html.EJ().ReportViewer("viewer").ProcessingMode(Syncfusion.JavaScript.R
eportViewerEnums.ProcessingMode.Local)
.ReportPath("~/App_Data/Product List.rdlc")
```

```
.ReportServiceUrl("/api/ReportApi")
.DataSources(ds =>
ds.Name("list").Value(ViewData["reportDsource"]).Add()))
```

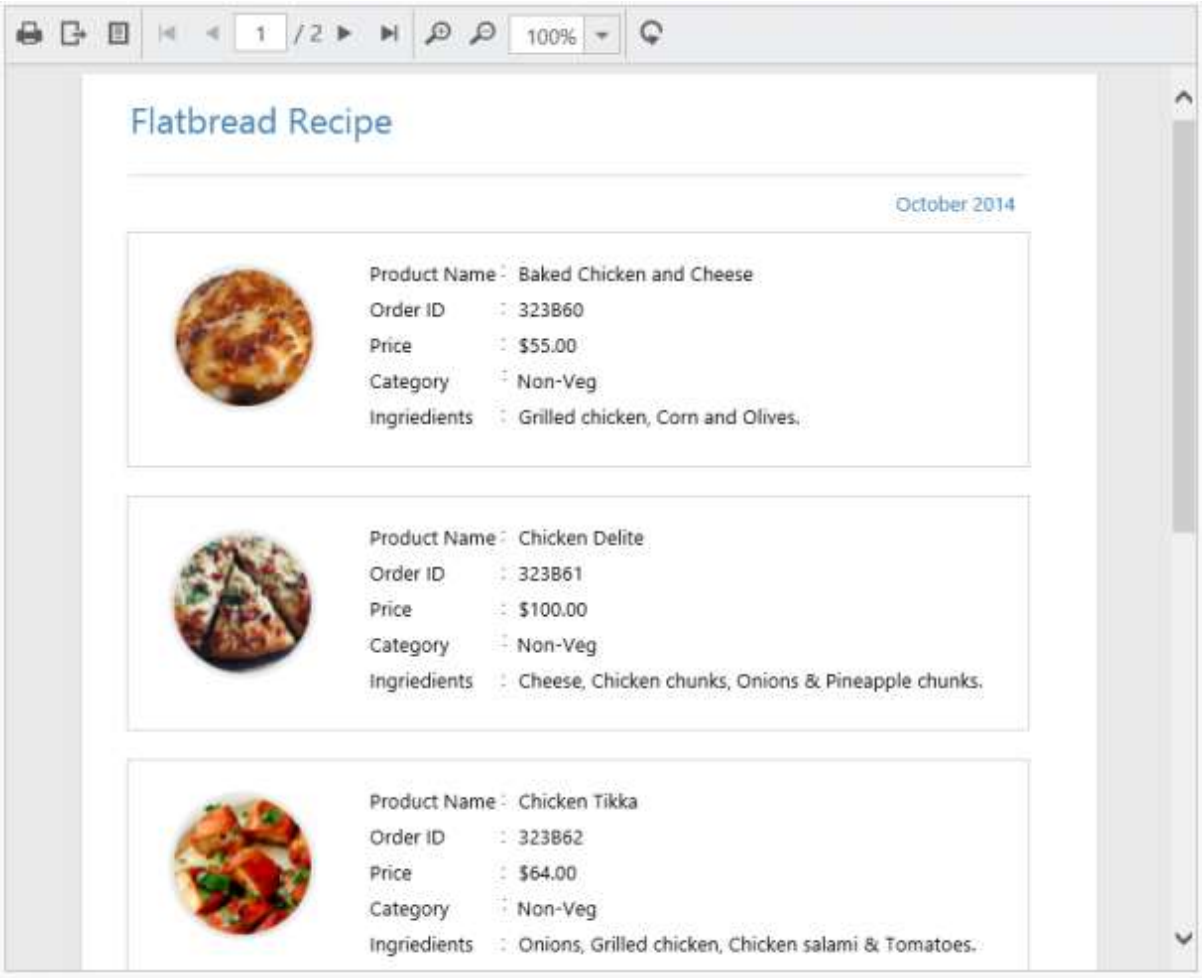
2. The following code example illustrates the creation of business object data source.

CSHARP

```
namespace ReportViewerDemo.Controllers
{
public partial class ReportViewerController : Controller
{
public ActionResult ProductList()
{
ProductList prodlist = new ProductList();
ViewData["reportDsource"] = prodlist.GetData();
return View();
}
}
public class ProductList
{
public string ProductName { get; set; }
public string OrderId { get; set; }
public double Price { get; set; }
public string Category { get; set; }
public string Ingredients { get; set; }
public string ProductImage { get; set; }
public IList GetData()
{
List<ProductList> datas = new List<ProductList>();
ProductList data = null;
data = new ProductList() {ProductName = "Baked Chicken and
Cheese",OrderId = "323B60",Price = 55,Category = "Non-Veg",Ingredients =
"grilled chicken, corn and olives.",ProductImage = "" };
datas.Add(data);
data = new ProductList() {ProductName = "Chicken Delite",OrderId =
"323B61",Price = 100,Category = "Non-Veg",Ingredients = "cheese, chicken
chunks, onions & pineapple chunks.",ProductImage = ""};
datas.Add(data);
data = new ProductList() {ProductName = "Chicken Tikka",OrderId =
"323B62",Price = 64,Category = "Non-Veg",Ingredients = "onions, grilled
chicken, chicken salami & tomatoes.",ProductImage = ""};
datas.Add(data);
return datas;
}
}
}
```

3. Run the application and you can see the ReportViewer on the page as displayed in the following screenshot.

Copyright © 2001 - 2020 Syncfusion Inc.



Product List RDLC Report

Report Controller

The ReportViewer uses Web API services to process the report file, process the request from control, and to return the processed data to control. The Syncfusion.EJ.ReportViewer assembly has helper APIs to define the service actions and process the service requests.

IReportController

The interface IReportController has the declaration of action methods that is defined in WebApi Controller for processing the RDL/RDLC files and for processing request from ReportViewer control. The IReportController has the following action methods declaration.

Methods	Description
GetResource	Action (HttpGet) method for getting resource for report.
PostReportAction	Action (HttpPost) method for posting the request for report process.
OnInitReportOptions	Report initialization method that is triggered when report begins to be processed.

Copyright © 2001 - 2020 Syncfusion Inc.

OnReportLoaded	Report loaded method that is triggered when report and sub report begin loading.

ReportHelper

The class ReportHelper contains helper methods that helps process Post/Get request from ReportViewer control and returns the response to ReportViewer control. The ReportHelper has the following methods.

Methods	Description
GetResource	Returns the report resource for the requested key.
ProcessReport	Processes the report request and returns the result.

C#

```
public class ReportsController : ApiController, IReportController
{
/// <summary>
/// Action (HttpGet) method for getting resource for report.
/// </summary>
/// <param name="key">The unique key to get the required
resource.</param>
/// <param name="resourceType">The type of the requested
resource.</param>
/// <param name="isPrinting">If set to <see langword="true"/>, then the
resource is generated for printing.</param>
/// <returns>The object data.</returns>
public object GetResource(string key, string resourceType, bool
isPrinting)
{
//Returns the report resource for the requested key.
return ReportHelper.GetResource(key, resourceType, isPrinting);
}
/// <summary>
/// Report Initialization method that is triggered when report begin
processed.
/// </summary>
/// <param name="reportOptions">The ReportViewer options.</param>
public void OnInitReportOptions(ReportViewerOptions reportOptions)
{
//You can update report options here
}
/// <summary>
/// Report loaded method that is triggered when report and sub report
begins to be loaded.
/// </summary>
/// <param name="reportOptions">The ReportViewer options.</param>
public void OnReportLoaded(ReportViewerOptions reportOptions)
{
//You can update report options here
}
/// <summary>
```

```
/// Action (HttpPost) method for posting the request for report process.
/// </summary>
/// <param name="jsonData">The JSON data posted for processing
report.</param>
/// <returns>The object data.</returns>
public object PostReportAction(Dictionary<string, object> jsonData)
{
//Processes the report request and returns the result.
return ReportHelper.ProcessReport(jsonData, this);
}
}
```

SSRS Configuration

The ReportViewer has support to load RDL/RDLC reports from SSRS server. You have to set your SSRS server URL to ReportViewer's ReportServerUrl property and set the relative path of RDL/RDLC file in SSRS to ReportViewer's ReportPath property.

ASPX-CS

```
@(Html.EJ().ReportViewer("viewer").ProcessingMode(Syncfusion.JavaScript.R
eportViewerEnums.ProcessingMode.Remote)
.ReportServiceUrl("/api/SSRSReport")
.ReportServerUrl("http://mvc.syncfusion.com/ReportServer")
.ReportPath("/SSRSSamples2/Territory Sales new"))
```

Network Credentials for SSRS

The Network credentials can be given at WebAPI Controller to connect the SSRS server.

C#

```
/// <summary>
/// Report Initialization method that is triggered when report begins to
process.
/// </summary>
/// <param name="reportOptions">The ReportViewer options.</param>
public void OnInitReportOptions(ReportViewerOptions reportOptions)
{
//Adds SSRS Server and Database Credentials here.
reportOptions.ReportModel.ReportServerCredential = new
System.Net.NetworkCredential("ssrs", "RDLReport1");
reportOptions.ReportModel.DataSourceCredentials.Add(new
DataSourceCredentials("AdventureWorks", "ssrs1", "RDLReport1"));
}
```

Note: DataSource credentials must be added to the ReportViewer for Shared DataSources that do not have credentials in the connection string and used in the SSRS Reports.

Report DataSources

The ReportViewer has support to add data sources to ReportViewer for RDLC reports at runtime. You can add SQL Server, Oracle, MS Azure, XML, Business Object, and SQL Server Compact DataSources to ReportViewer. The ReportViewer has DataSources property that is the list of ReportDataSource type to add collection of DataSources to it. You can add DataSources either through ReportViewer model when creating ReportViewer control or through Web API.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example illustrates how to add DataSource at control creation.

ASPX-CS

```
@(Html.EJ().ReportViewer("viewer").ProcessingMode(Syncfusion.JavaScript.R
eportViewerEnums.ProcessingMode.Local)
.ReportPath("~/App_Data/Product List.rdlc")
.ReportServiceUrl("/api/ReportApi").DataSources(ds => ds.Name("list")
.Value(ViewData["DataSource"]).Add()))
```

C#

```
public partial class ReportViewerController : Controller
{
public ActionResult ProductList()
{
ProductList productList = new ProductList();
ViewData["DataSource"] = productList.GetData();
return View();
}
}
public class ProductList
{
public string ProductName { get; set; }
public string OrderId { get; set; }
public double Price { get; set; }
public string Category { get; set; }
public string Ingredients { get; set; }
public string ProductImage { get; set; }
public IList GetData()
{
List<ProductList> dataList = new List<ProductList>();
ProductList data = null;
data = new ProductList() {ProductName = "Baked Chicken and
Cheese",OrderId = "323B60",Price = 55,Category = "Non-Veg",Ingredients =
"grilled chicken, corn and olives.",ProductImage = "" };
dataList.Add(data);
data = new ProductList() {ProductName = "Chicken Delite",OrderId =
"323B61",Price = 100,Category = "Non-Veg",Ingredients = "cheese, chicken
chunks, onions & pineapple chunks.",ProductImage = ""};
dataList.Add(data);
data = new ProductList() {ProductName = "Chicken Tikka",OrderId =
"323B62",Price = 64,Category = "Non-Veg",Ingredients = "onions, grilled
chicken, chicken salami & tomatoes.",ProductImage = ""};
dataList.Add(data);
return dataList;
}
}
```

The following code example illustrates how to add DataSource in Web API.

C#

```
public class ReportsController : ApiController, IReportController
{
/// <summary>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
/// Report loaded method that is triggered when report and sub report are
loaded.
/// </summary>
/// <param name="reportOptions">The ReportViewer options.</param>
public void OnReportLoaded(ReportViewerOptions reportOptions)
{
//Adds data sources to report model.
reportOptions.ReportModel.DataSources.Clear();
ProductList productList = new ProductList();
reportOptions.ReportModel.DataSources.Add(new ReportDataSource { Name =
"list", Value = productList.GetData() });
}
}
public class ProductList
{
public string ProductName { get; set; }
public string OrderId { get; set; }
public double Price { get; set; }
public string Category { get; set; }
public string Ingredients { get; set; }
public string ProductImage { get; set; }
public IList GetData()
{
List<ProductList> dataList = new List<ProductList>();
ProductList data = null;
data = new ProductList() { ProductName = "Baked Chicken and Cheese",
OrderId = "323B60", Price = 55, Category = "Non-Veg", Ingredients =
"grilled chicken, corn and olives.", ProductImage = "" };
dataList.Add(data);
data = new ProductList() { ProductName = "Chicken Delite", OrderId =
"323B61", Price = 100, Category = "Non-Veg", Ingredients = "cheese,
chicken chunks, onions & pineapple chunks.", ProductImage = "" };
dataList.Add(data);
data = new ProductList() { ProductName = "Chicken Tikka", OrderId =
"323B62", Price = 64, Category = "Non-Veg", Ingredients = "onions,
grilled chicken, chicken salami & tomatoes.", ProductImage = "" };
dataList.Add(data);
return dataList;
}
}
```

DataSource Credentials

The DataSource credentials can be given at Web API Controller to connect data source.

C#

```
/// <summary>
/// Report Initialization method that is triggered when report begins to
process.
/// </summary>
/// <param name="reportOptions">The ReportViewer options.</param>
public void OnInitReportOptions(ReportViewerOptions reportOptions)
{
//Adds SSRS Server and Database Credentials here.
reportOptions.ReportModel.ReportServerCredential = new
System.Net.NetworkCredential("ssrs", "RDLReport1");
```

```
reportOptions.ReportModel.DataSourceCredentials.Add(new
DataSourceCredentials("AdventureWorks", "ssrs1", "RDLReport1"));
}
```

Report Parameters

The ReportViewer has support to add report parameters to ReportViewer at runtime. The ReportViewer has Parameters property that is the list of ReportParameter type to add collection of report parameters to it. You can add report parameters either through ReportViewer model when creating ReportViewer control or through Web API.

The following code example illustrates how to add ReportParameter at control creation.

ASPX-CS

```
@(Html.EJ().ReportViewer("viewer").ProcessingMode(Syncfusion.JavaScript.R
eportViewerEnums.ProcessingMode.Remote)
.ReportPath("InvoiceTemplate.rdl")
.ReportServiceUrl("/api/ReportApi")
.Parameters(param => {param.Name("InvoiceID").Labels(new List<string>() {
"InvoiceID" }).Values(new List<string>() { "10250" }).Add();}))
```

The following code example illustrates how to add ReportParameter in Web API.

C#

```
public class ReportsController : ApiController, IReportController
{
/// <summary>
/// Report Initialization method that is triggered when report begins to
be processed.
/// </summary>
/// <param name="reportOptions">The ReportViewer options.</param>
public void OnInitReportOptions(ReportViewerOptions reportOptions)
{
List<ReportParameter> parameters = new List<ReportParameter>();
parameters.Add(new ReportParameter() { Name = "InvoiceID", Labels = new
List<string>() { "InvoiceID" }, Values = new List<string>() { "10250" }
});
reportOptions.ReportModel.Parameters = parameters;
}
}
```

Toolbar Customization

The ReportViewer has an option to show or hide items in the toolbar. To customize the toolbar items, use the ReportViewer's ToolbarSettings property. The toolbar template can also be customized by specifying custom template to ReportViewer toolbar.

ASPX-CS

```
@(Html.EJ().ReportViewer("viewer")
.ToolbarSettings(tb=>tb.Items(Syncfusion.JavaScript.ReportViewerEnums.Too
lbarItems.All &
~Syncfusion.JavaScript.ReportViewerEnums.ToolbarItems.Parameters)))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Events

The ReportViewer has the following client-side events support to listen to the control action.

Events	Description
destroy	Fires when the ReportViewer is destroyed.
drillThrough	Fires during drill through action done in report and event can be canceled.
renderingBegin	Fires before report rendering is completed.
renderingComplete	Report loaded method that is triggered when report and sub report begin loading.
reportError	Fires when any error occurs while rendering the report and event can be canceled.
reportLoaded	Fires when the report is loaded.

HTML

```
@(Html.EJ().ReportViewer("viewer")
.ProcessingMode(Syncfusion.JavaScript.ReportViewerEnums.ProcessingMode.Lo
cal)
.ReportPath("DatabindingRemote.rdlc")
.ReportServiceUrl(VirtualPathUtility.ToAbsolute("~/api/RDLCReport")).Repo
rtLoaded("reportLoaded") )
<script type="text/javascript">
function reportLoaded(senderObj) {
$.ajax({
type: "POST",
contentType: "application/json; charset=utf-8",
url: '../wcf/Reportservice.svc/GetOrderDetails',
dataType: "json",
processData: false,
crossDomain: true,
async: false,
timeout: 5000,
success: function (result) {
reportdata = result.d;
var dataManger = ej.DataManager(reportdata);
var query = ej.Query().select("OrderID", "CustomerID", "EmployeeID",
"Freight", "ShipCity", "ShipCountry");
reportResult = dataManger.executeLocal(query);
var reportModel = $("#viewer").data('ejReportViewer');
reportModel.model.dataSources = [{ value: reportResult, name: "remote"
}];
},
error: function (result) {
alert(result);
}
});
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Limitations

RDL Specification support

- The ReportViewer control does not support RDL Specification for SQL Server 2000 and SQL Server 2005.

Layout process

- The Syncfusion ReportViewer control has some limitations in the Tablix cell split layout process in comparison with MS ReportViewer. When the table cell width value exceeds the page width, the entire cell moves to the next page to display the complete cell items.
- The Syncfusion ReportViewer control does not support vertical alignment in the text box report item.

Unsupported expression

- The object function and VB function do not have complete support in the ReportViewer.

ReportDesigner

Overview

Report Designer helps to create and modify the RDL 2008/2008R2 specification standard reports. By using Report Designer, the data can be transformed into meaningful information and quickly build the reports using both table and visualization formats with the help of following features.

Key features

Data Sources --- All the commonly used data sources like Microsoft SQL Server, Oracle, OLEDB and ODBC are supported.

Business user friendly --- The drag-and-drop friendly report designer application makes it possible for business users to compose reports without any help from IT.

Report items --- All interactive report items that are commonly used in business reports like a variety of charts, pivot, grouping and grid have been included.

Report Parameter --- Parameters are used to specify the data to use in a report, connect related reports together and vary report presentation.

Expression --- Expressions are used throughout the report definition to specify or calculate values for parameters, queries, filters, report item properties, group and sort definitions, textbox properties, dynamic page header and footer content and images.

Getting Started

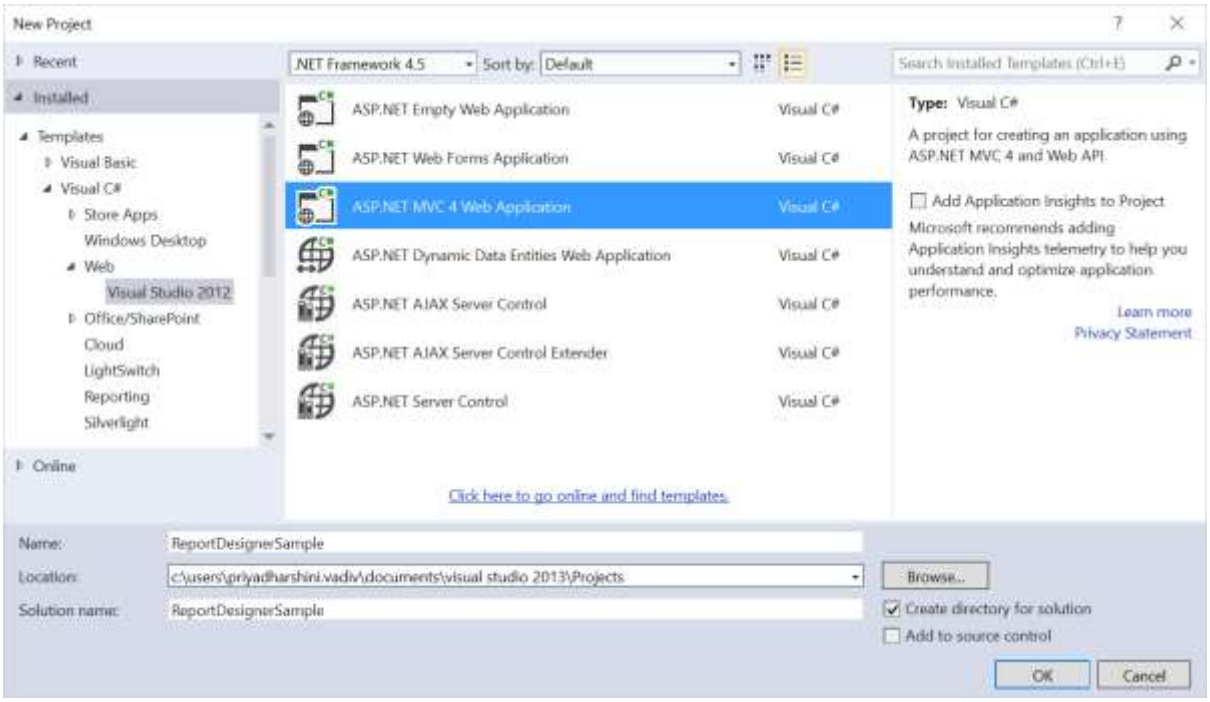
This section explains briefly about how to create a **ReportDesigner** in your ASP.NET MVC application.

Create your first ReportDesigner application in ASP.NET MVC

The following screenshots displays the Project Creation Wizard in Visual Studio 2013

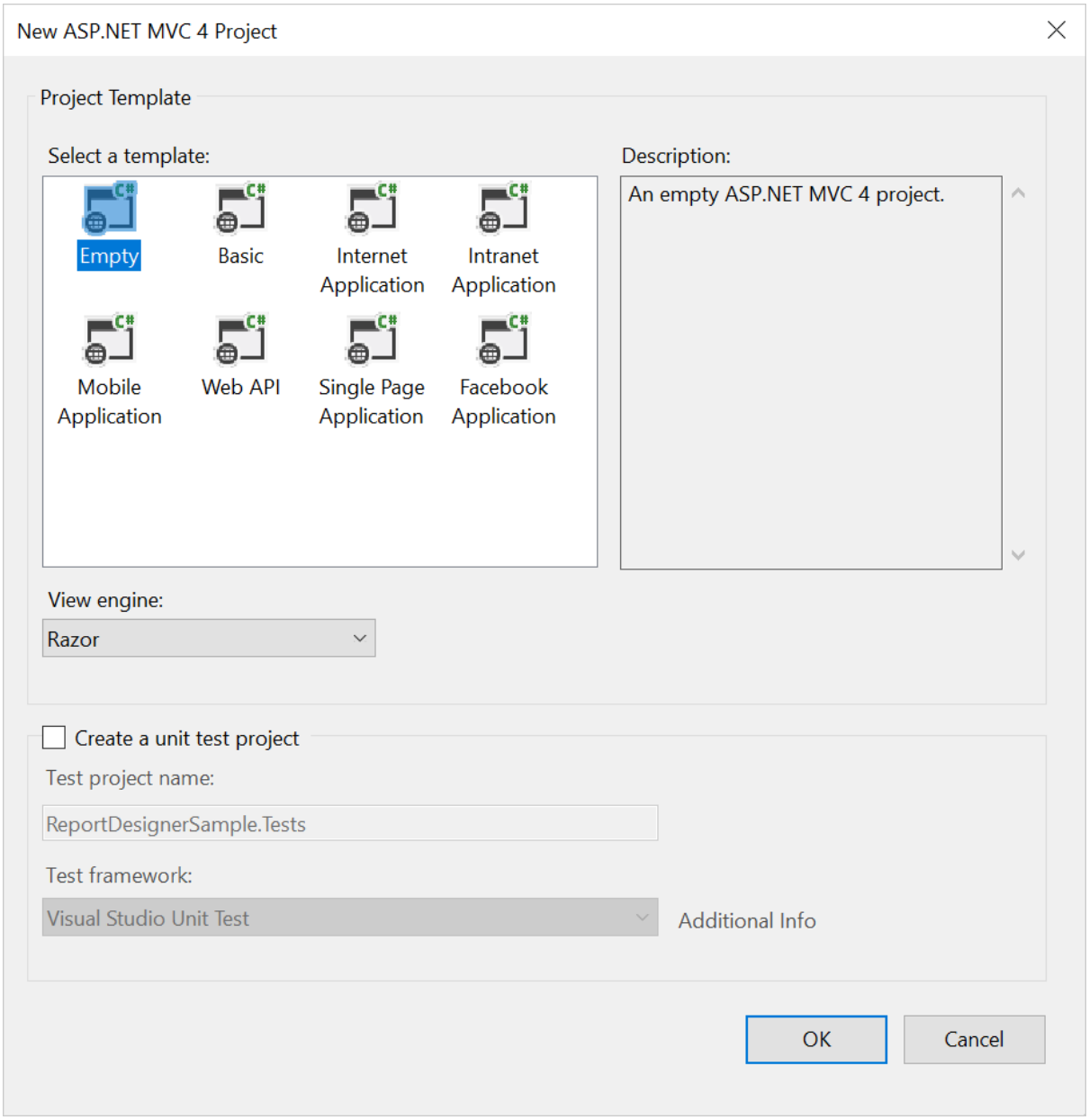
Create a new ASP.NET MVC4 Web Application project by selecting the **WEB** category from the listed project template in Microsoft Visual Studio IDE.

Copyright © 2001 - 2020 Syncfusion Inc.



Project Creation Wizard

The following screenshot displays how to select the project template with razor view engine:

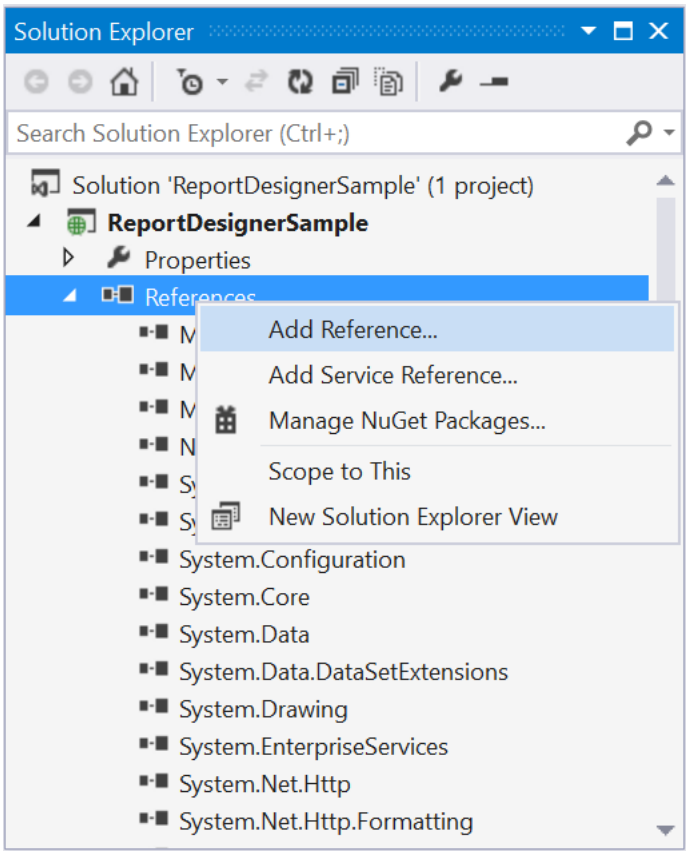


Select an empty template and click OK.

Add Assembly References

1. In the Solution Explorer, right-click the References folder and then click Add Reference.

Copyright © 2001 - 2020 Syncfusion Inc.



2. Add the following Syncfusion assemblies to the project that are necessary for using the report designer control.
    - Syncfusion.Chart.Wpf
    - Syncfusion.Compression.Base
    - Syncfusion.DocIO.Base
    - Syncfusion.EJ.ReportDesigner
    - Syncfusion.EJ.ReportViewer
    - Syncfusion.Gauge.Wpf
    - Syncfusion.Pdf.Base
    - Syncfusion.Presentation.Base
    - Syncfusion.Shared.Wpf
    - Syncfusion.SfMaps.Wpf
    - Syncfusion.XlsIO.Base
    - Syncfusion.EJ
    - Syncfusion.EJ.MVC

Refer the Syncfusion.EJ.MVC assembly from the installed location, [Installed Location]:\Program Files (x86)\Syncfusion\Essential Studio\ASP.NET MVC\{{ site.releaseversion }}\Assemblies\MVC\MVC4

Refer the other assemblies from the installed location, [Installed Location]:\Program Files (x86)\Syncfusion\Essential Studio\ASP.NET MVC\{{ site.releaseversion }}\Assemblies

3. Add the following WebAPI assemblies from NuGet package.

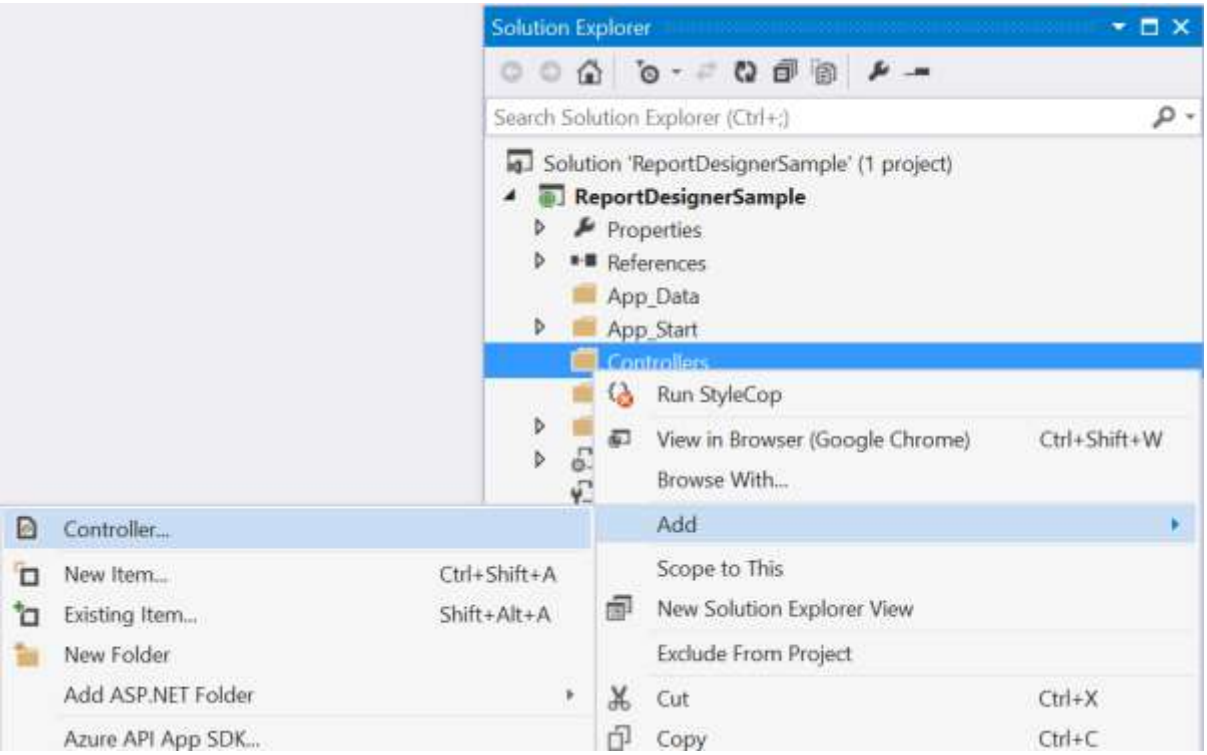
Copyright © 2001 - 2020 Syncfusion Inc.

- System.Web.Http
- System.Web.Http.WebHost
- System.Net.Http.WebRequest
- System.Net.Http.Formatting

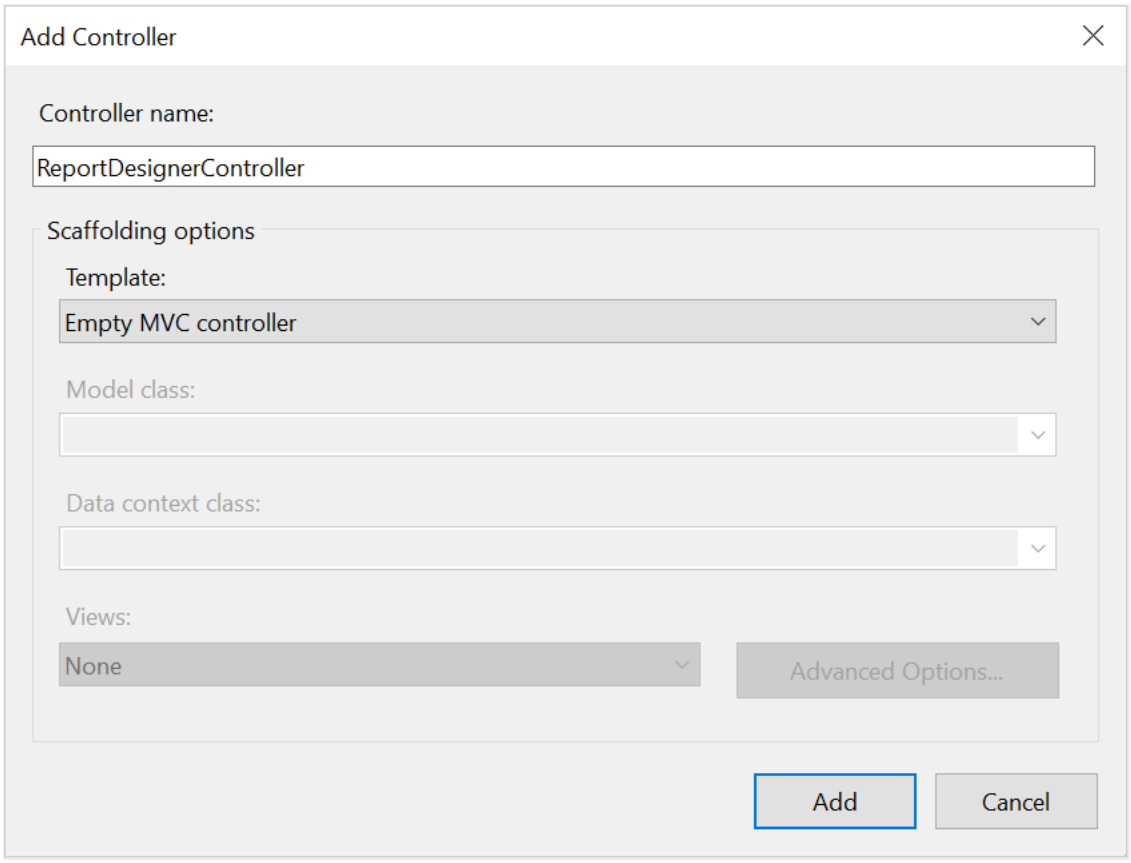
The System.Web.Routing and System.Net.Http assemblies are also required, which are referred by default when creating the project.

Add Controller

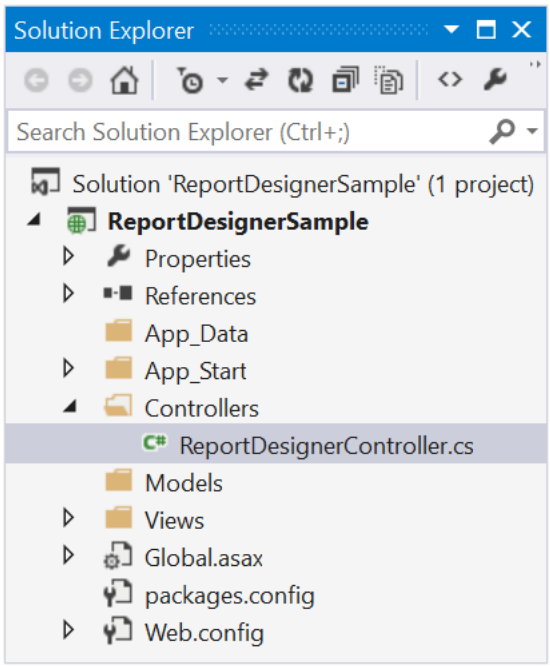
1. Right-Click on the **Controllers** folder in the project and select Add then select Controller.



2. Name the controller as **ReportDesignerController**.

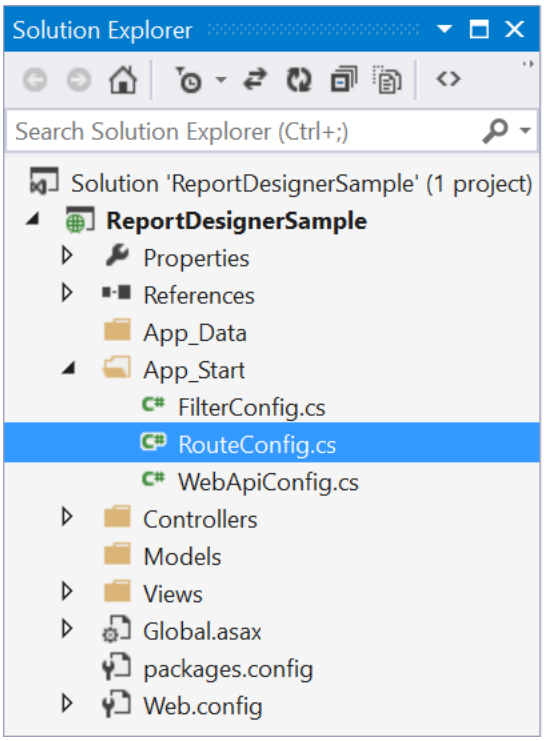


3. Click Add.



Modify RouteConfig.cs

1.Open the RouteConfig.cs file from App_Start folder of your application.



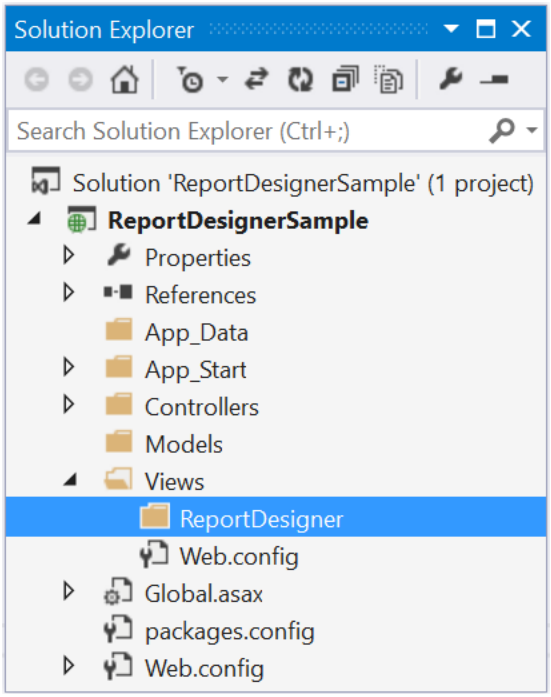
2.Modify the controller name to map to the **ReportDesigner** controller as follows.

C#

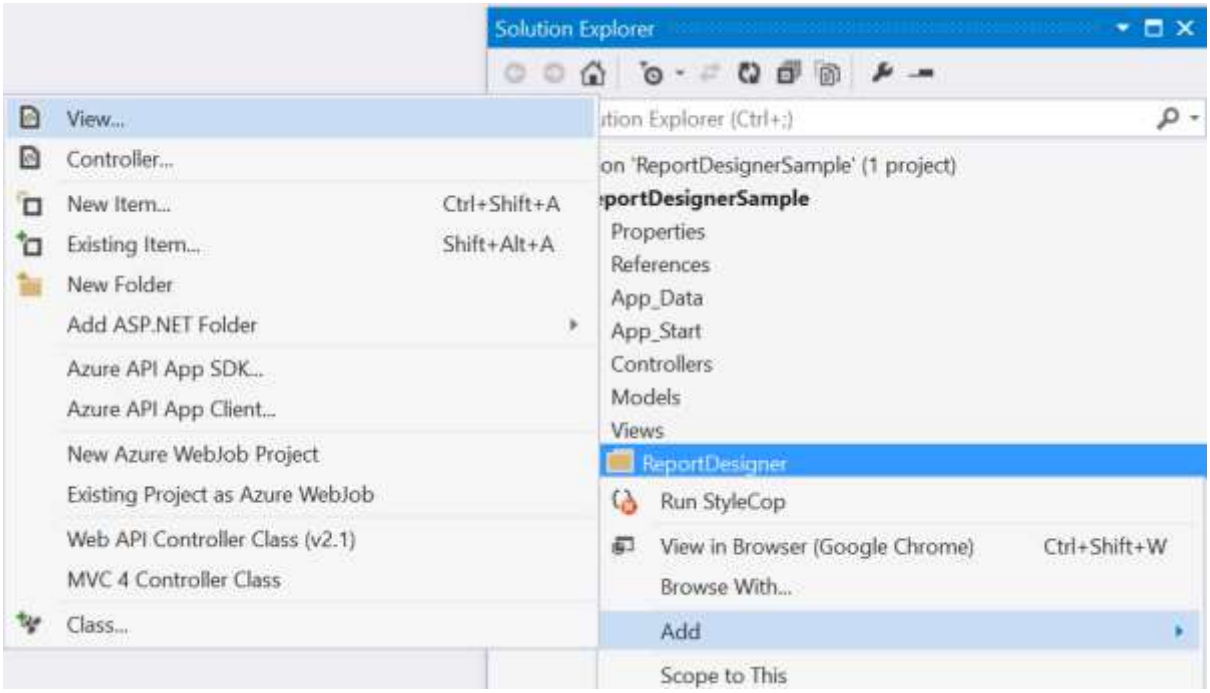
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
namespace ReportDesignerSample
{
public class RouteConfig
{
public static void RegisterRoutes(RouteCollection routes)
{
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
routes.MapRoute(
name: "Default",
url: "{controller}/{action}/{id}",
defaults: new { controller = "ReportDesigner", action = "Index", id =
UrlParameter.Optional }
);
}
}
}
```

Add View page

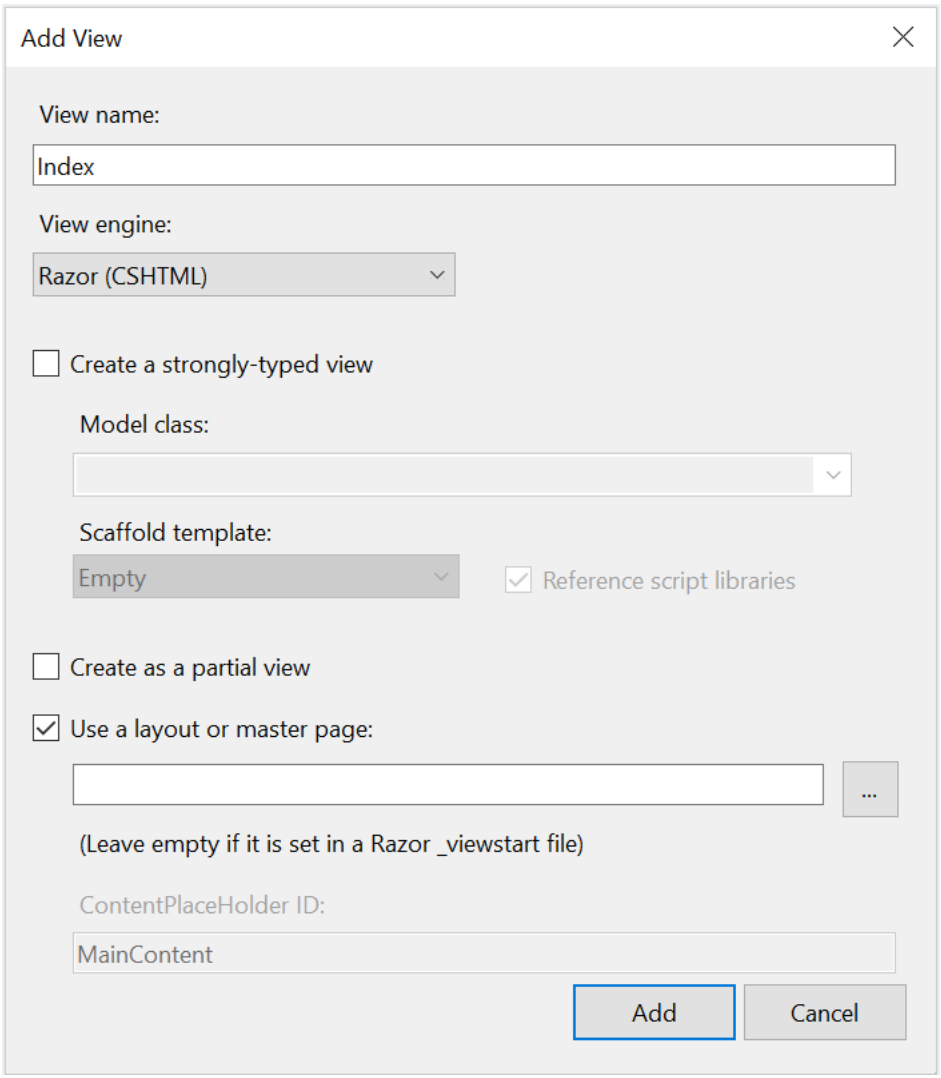
1. Create a new folder **ReportDesigner** in Views folder.



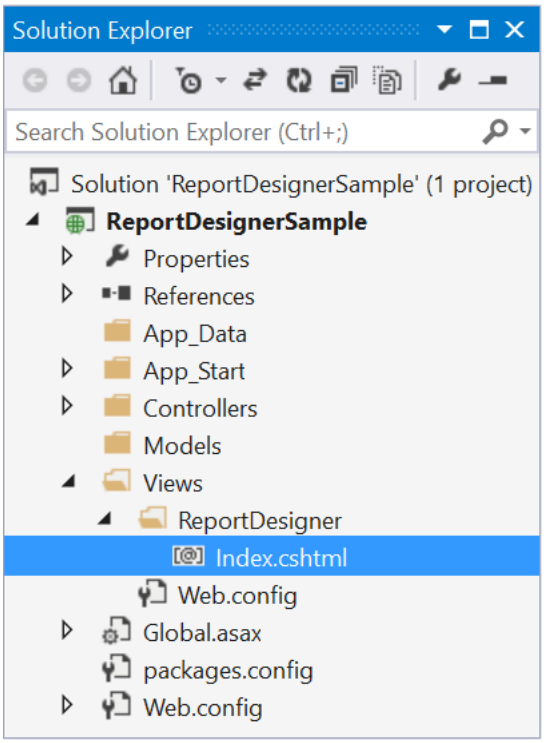
2. Right-Click on the **ReportDesigner** folder in the Views folder and select Add then select View.



3. Name the view page as **Index**.


Add View
View name:
Index
View engine:
Razor (CSHTML)
Create a strongly-typed view
Model class:
Scaffold template:
Empty
Reference script libraries
Create as a partial view
Use a layout or master page:
...
(Leave empty if it is set in a Razor _viewstart file)
ContentPlaceHolder ID:
MainContent
Add
Cancel

4. Click Add.



Add Scripts and Styles

For complete dependencies list of report designer control Click here.

Add the below code snippet in the Index.cshtml page.

ASPX-CS

```
@using Syncfusion.JavaScript
@using Syncfusion.MVC.EJ
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>@ViewBag.Title</title>
</head>
<body>
</body>
</html>
```

Themes

ASPX-CS

```
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet" />
<link href="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.reportdesigner.min.css" rel="stylesheet" />
```

Scripts

External dependencies

ASPX-CS

```
<script src="http://code.jquery.com/jquery-1.10.2.min.js"
type="text/javascript"></script>
<script src="http://cdnjs.cloudflare.com/ajax/libs/jquery-
easing/1.3/jquery.easing.min.js" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jsrender/0.9.90/jsrender.min.
js" type="text/javascript"></script>
```

Internal dependencies

Refer the below scripts to render report designer control.

ASPX-CS

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js" type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.reportdesigner.min.js" type="text/javascript"></script>
```

Code Mirror

To edit the SQL queries with syntax highlighter need to refer the below code mirror scripts and themes.

ASPX-CS

```
<link
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.37.0/codemirror
.min.css" rel="stylesheet" />
<link
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.37.0/addon/hint
/show-hint.min.css" rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.37.0/codemirror.
min.js" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.37.0/addon/hint/
show-hint.min.js" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.37.0/addon/hint/
sql-hint.min.js" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.37.0/mode/sql/sq
l.min.js" type="text/javascript"></script>
```

Add the script files and theme file references in the <head> tag of the Index.cshtml page.

Also the scripts and styles can be referred from the installed location, [Installed Location]:\Program Files (x86)\Syncfusion\Essential Studio\ASP.NET MVC\{{ site.releaseversion }}\JavaScript\assets

Unobtrusive Mode

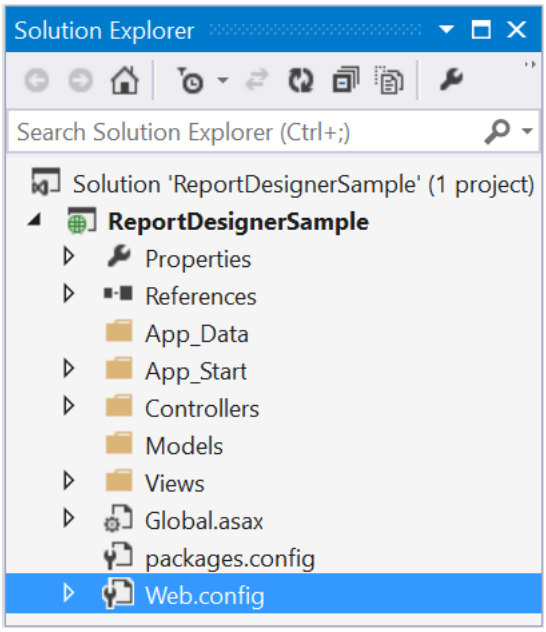
Enable unobtrusive mode

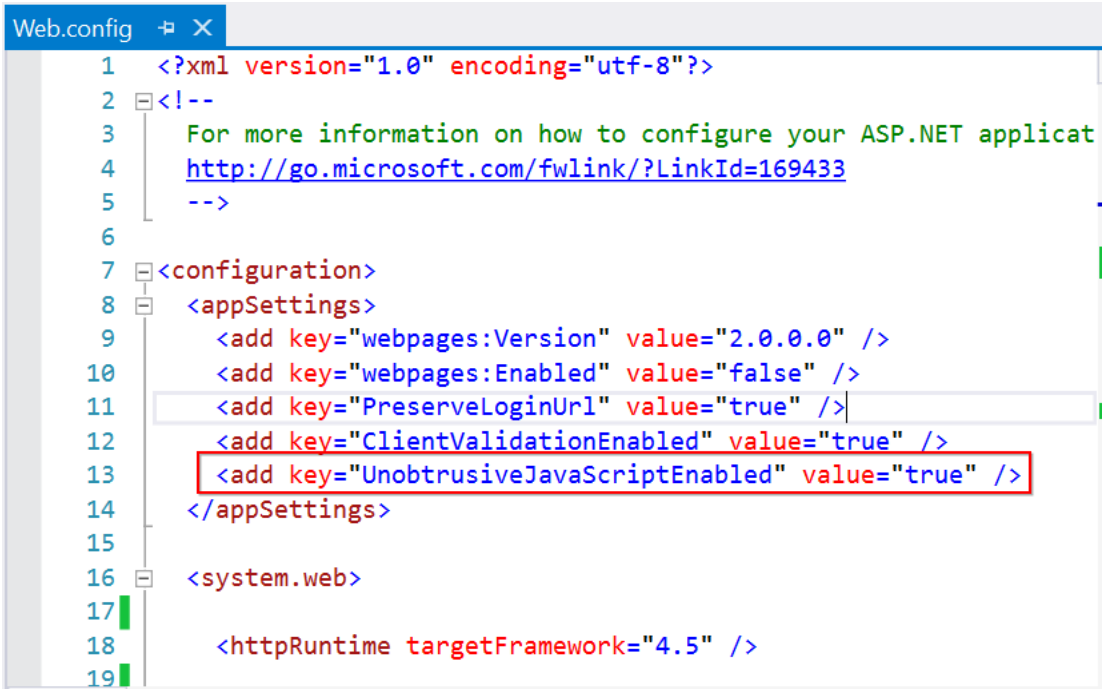
1.To render the Report Designer in unobtrusive mode, refer to the **ej.unobtrusive.js** script file in <head> of Index.cshtml page.

ASPX-CS

```
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/common/ej.unobtrusive.min.js"></script>
```

2.Set **UnobtrusiveJavaScriptEnabled** to true in Web.config file of your application.





Disable unobtrusive mode

Set **UnobtrusiveJavaScriptEnabled** to false in Web.config file of your application.

Web.config

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET applicat
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>
  <appSettings>
    <add key="webpages:Version" value="2.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="PreserveLoginUrl" value="true" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="false" />
  </appSettings>

  <system.web>

    <httpRuntime targetFramework="4.5" />
```

Refer to the unobtrusive section for more details.

Registering namespaces within Web.config

Register the following namespaces in the web.config file present within the Views folder as well as the root directory of your application.

XML

```
<namespaces>
<add namespace="Syncfusion.EJ.ReportViewer"/>
<add namespace="Syncfusion.EJ.ReportDesigner"/>
<add namespace="Syncfusion.EJ"/>
<add namespace="Syncfusion.MVC.EJ"/>
</namespaces>
```

Add Control in View page

Add the following code example in the <body> tag of the Index.cshtml page that is already created. Set the desired ServiceUrl to ReportDesigner.

ASPX-CS

```
<div>
@{Html.EJ().ReportDesigner("designer").ServiceUrl("/api/DesignerAPI").Ren
der();}
</div>
@(Html.EJ().ScriptManager())
```

The main reason for referring the **Script manager** in Index.cshtml is that, it can be referred as common by all the View files present within your application. If **unobtrusive** is set to true in the application, the script manager can be excluded, as the control is initialized using HTML5 attributes. Also for control

Copyright © 2001 - 2020 Syncfusion Inc.

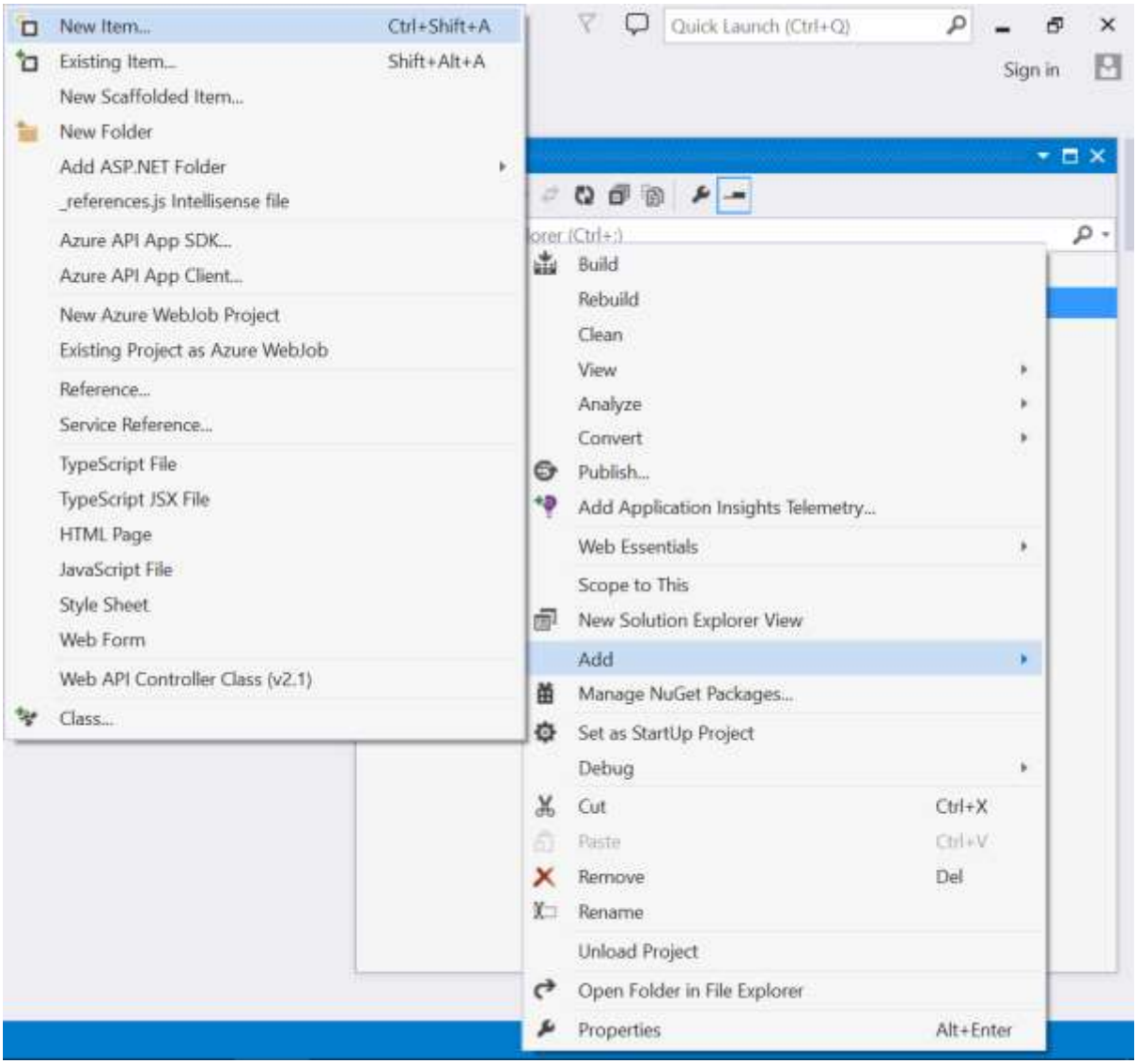
rendering, you need to include **ej.unobtrusive.min.js** file along with ej.web.all.min.js in **Index.cshtml** page.

Add WebAPI controller for ReportDesigner

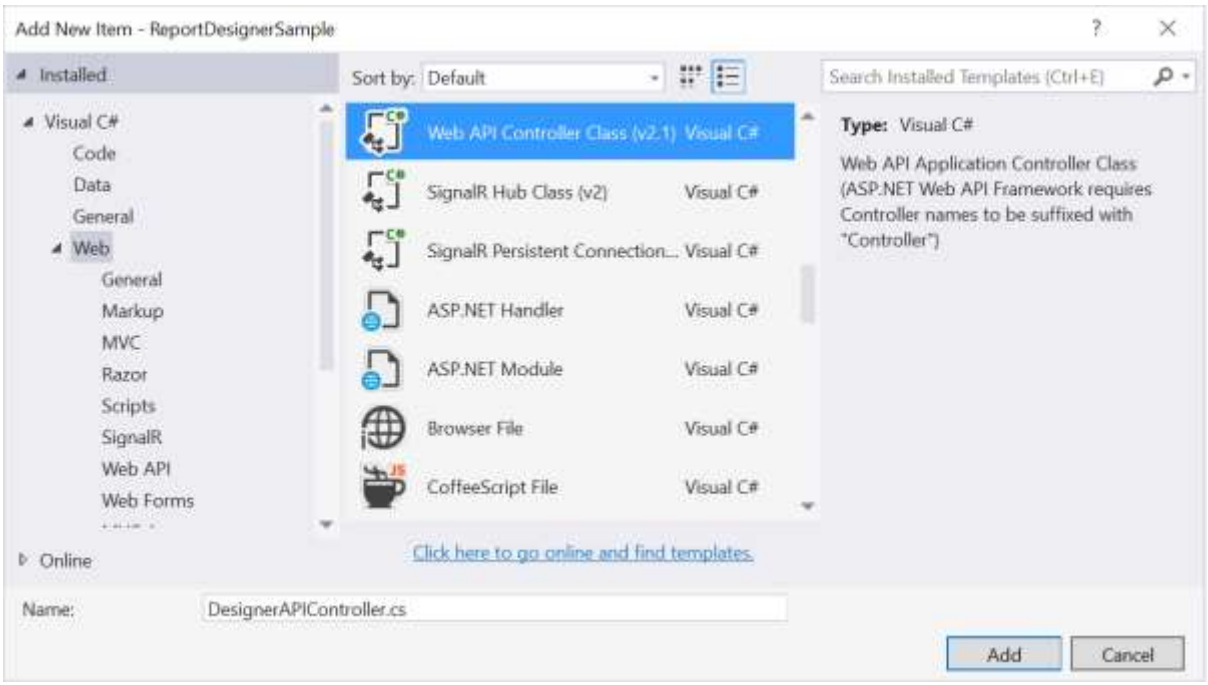
The MVC ReportDesigner uses WebApi services to process the report file and process the request from control.

Add Controller

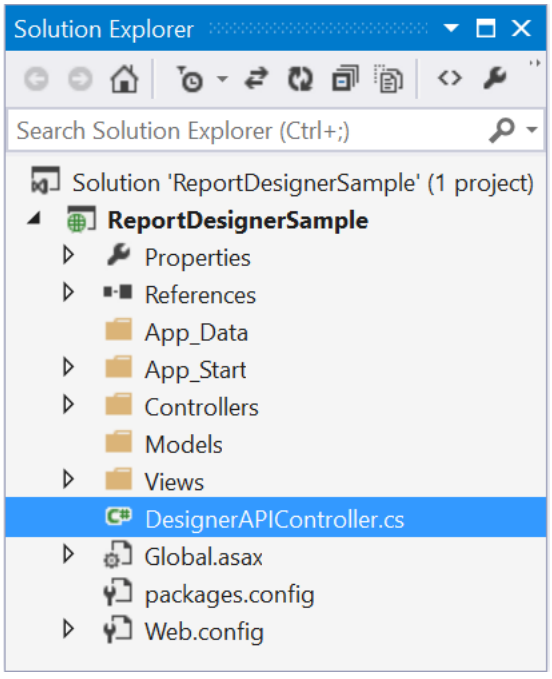
1. Right-Click on the project and select Add then click New Item.



2. Select Web API Controller Class from the listed templates and name the controller as **DesignerAPIController.cs**.



3. Click Add.



Inherit IReportDesignerController

The ApiController should inherit the `IReportDesignerController` and to process the report file. The interface `IReportDesignerController` contains the required actions and helper methods declaration to process the report. The `ReportDesignerHelper` and `ReportHelper` class contains helper methods that helps to process Post/Get request from control and return the response to control.

Please add the following code example in `DesignerAPIController.cs`.

C#

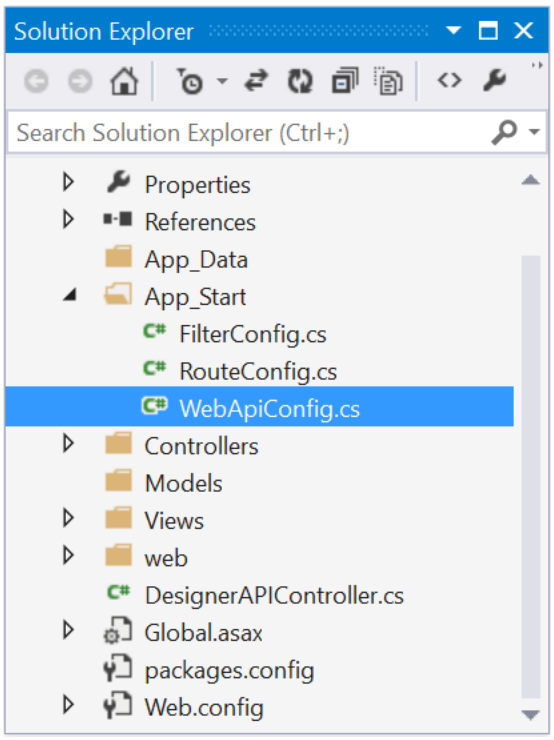
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using System.IO;
using System.Web;
using Syncfusion.EJ.ReportViewer;
using Syncfusion.Reports.EJ;
using Syncfusion.EJ.ReportDesigner;
namespace ReportDesignerSample
{
public class DesignerAPIController : ApiController,
Syncfusion.EJ.ReportDesigner.IReportDesignerController
{
public string GetFilePath(string fileName)
{
string targetFolder = HttpContext.Current.Server.MapPath("~/");
targetFolder += "Cache";
if (!Directory.Exists(targetFolder))
{
Directory.CreateDirectory(targetFolder);
}
if (!Directory.Exists(targetFolder + "\\" +
ReportDesignerHelper.EJReportDesignerToken))
{
Directory.CreateDirectory(targetFolder + "\\" +
ReportDesignerHelper.EJReportDesignerToken);
}
var folderPath = HttpContext.Current.Server.MapPath("~/") + "Cache\\" +
ReportDesignerHelper.EJReportDesignerToken + "\\";
return folderPath + fileName;
}
[HttpGet]
public object GetImage(string key, string image)
{
return ReportDesignerHelper.GetImage(key, image, this);
}
[HttpPost]
public object PostDesignerAction(Dictionary<string, object> jsonResult)
{
return ReportDesignerHelper.ProcessDesigner(jsonResult, this, null);
}
public bool UploadFile(System.Web.HttpPostedFile httpPostedFile)
{
string targetFolder = HttpContext.Current.Server.MapPath("~/");
string fileName =
!string.IsNullOrEmpty(ReportDesignerHelper.SaveFileName) ?
ReportDesignerHelper.SaveFileName :
Path.GetFileName(httpPostedFile.FileName);
targetFolder += "Cache";
if (!Directory.Exists(targetFolder))
{
Directory.CreateDirectory(targetFolder);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
if (!Directory.Exists(targetFolder + "\\" +
ReportDesignerHelper.EJReportDesignerToken))
{
Directory.CreateDirectory(targetFolder + "\\" +
ReportDesignerHelper.EJReportDesignerToken);
}
httpPostedFile.SaveAs(targetFolder + "\\" +
ReportDesignerHelper.EJReportDesignerToken + "\\" + fileName);
return true;
}
public void UploadReportAction()
{
ReportDesignerHelper.ProcessDesigner(null, this,
HttpContext.Current.Request.Files[0]);
}
public object GetResource(string key, string resourcetype, bool isPrint)
{
return ReportHelper.GetResource(key, resourcetype, isPrint);
}
public void OnInitReportOptions(ReportViewerOptions reportOption)
{
//You can update report options here
}
public void OnReportLoaded(ReportViewerOptions reportOption)
{
//You can update report options here
}
public object PostReportAction(Dictionary<string, object> jsonResult)
{
return ReportHelper.ProcessReport(jsonResult, this as IReportController);
}
public FileModel GetFile(string filename, bool isOverride)
{
throw new NotImplementedException();
}
public List<FileModel> GetFiles(FileType fileType)
{
throw new NotImplementedException();
}
}
}
```

Modify WebApiConfig.cs

1. Open the WebApiConfig.cs file from App_Start folder of your application.



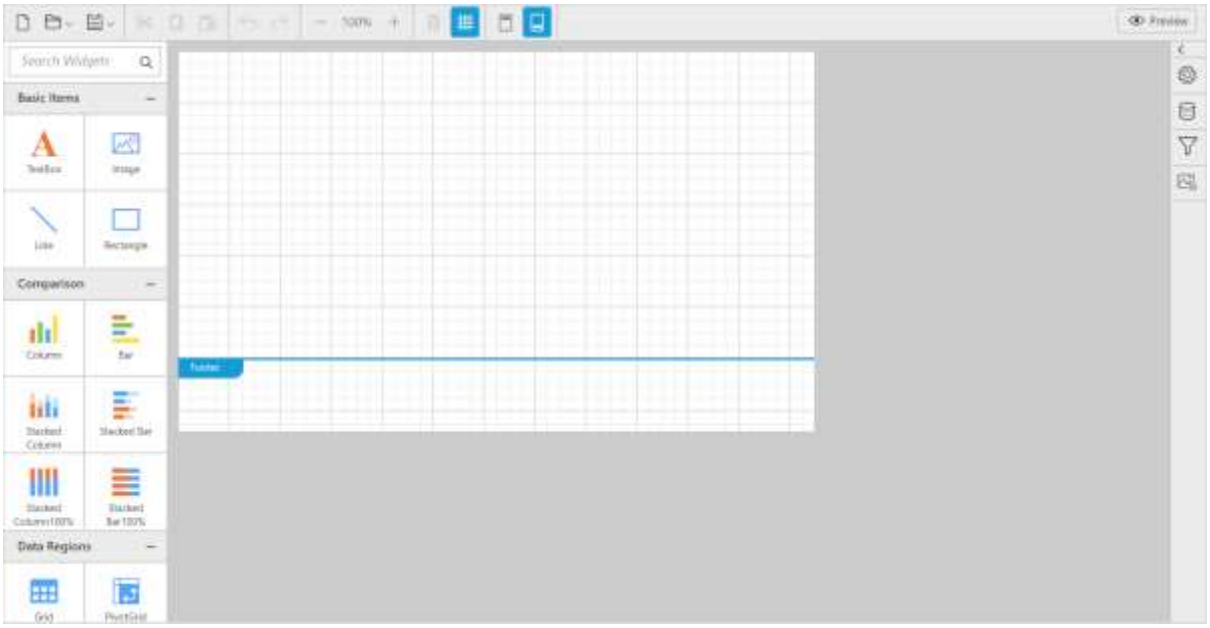
2. Modify the **routeTemplate** to map to the API controller of **ReportDesignerSample** project as follows.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;
namespace ReportDesignerSample
{
public static class WebApiConfig
{
public static void Register(HttpConfiguration config)
{
config.Routes.MapHttpRoute(
name: "DefaultApi",
routeTemplate: "api/{controller}/{action}/{id}",
defaults: new { id = RouteParameter.Optional }
);
}
}
}
```

Run the Application

On running the application, Report Designer will be rendered like below.



ReportDesigner Dependencies

The ReportDesigner have the following list of external dependencies.

- jQuery 1.7.1 and later versions.
- jsRender - to render the templates.
- CodeMirror - to edit the SQL queries with syntax highlighter.

And the internal dependencies are tabulated below.

File	Description/Usage
ej.core.min.js	Must be referred always before using all the *ASP.NET MVC* controls.
ej.data.js	Used to handle data operation and should be used while binding data to *ASP.NET MVC* controls.
ej.draggable.js	Used to handle drag on drop operation in all *ASP.NET MVC* controls.
ej.scroller.js	Used to handle scrolling operation in all *ASP.NET MVC* controls.
ej.globalize.js	It is referred when using localization in ReportDesigner.
ej.waitingpopup.js	These files are used to handle the internal UI and actions of ReportDesigner.
ej.button.js	
ej.checkbox.js	
ej.radiobutton.js	
ej.autocomplete.js	
ej.datepicker.js	

Copyright © 2001 - 2020 Syncfusion Inc.

ej.timepicker.js	
ej.datetimepicker.js	
ej.daterangepicker.js	
ej.dialog.js	
ej.dropdownlist.js	
ej.tooltip.js	
ej.listbox.js	
ej.map.js	
ej.editor.js	
ej.maskedit.js	
ej.menu.js	
ej.pager.js	
ej.slider.js	
ej.splitter.js	
ej.splitbutton.js	
ej.toolbar.js	
ej.tab.js	
ej.treeview.js	
ej.uploadbox.js	
ej.colorpicker.js	
ej.grid.js	
ej.reportViewer.js	
ej.chart.js	
ej.circulargauge.js	
ej.lineargauge.js	
ej.bulletgraph.js	

API Service

External assemblies needed for ReportDesigner service are tabulated below.

Assemblies	Description/Usage

Copyright © 2001 - 2020 Syncfusion Inc.

System.Web.Http	Must be referred these assemblies for WebAPI.
System.Web.Http.WebHost	
System.Net.Http.WebRequest	
System.Net.Http.Formatting	
System.Net.Http	
System.Web.Routing	

And the internal assemblies are tabulated below.

Assemblies	Description/Usage
Syncfusion.Shared.Wpf	It's common dependencies for our WPF components which will be used to handle the report export related actions in ReportViewer.
Syncfusion.Pdf.Base	Must be referred to handle pdf export.
Syncfusion.DocIO.Base	Must be referred to handle word export.
Syncfusion.XlsIO.Base	Must be referred to handle excel export.
Syncfusion.Presentation.Base	Must be referred to handle ppt export.
Syncfusion.Chart.Wpf	Must be referred to export the chart item from ReportViewer.
Syncfusion.Gauge.Wpf	Must be referred to export the Gauge item from ReportViewer.
Syncfusion.SfMaps.Wpf	Must be referred to export the Map item from ReportViewer.
Syncfusion.EJ.ReportViewer	It's used to preview the report.
Syncfusion.EJ.MVC	Must be referred to use Web Report Designer control from Syncfusion ASP.NET MVC controls.
Syncfusion.EJ	It's used as base class for Syncfusion ASP.NET MVC controls.

ReportWriter

Overview

ReportWriter is a class library that enables the user to render reports defined in Microsoft's RDL format (2008 or 2008 R2) as PDF, Word, Excel, PPT and HTML documents.

The important features of ASP MVC ReportWriter are listed as follows:

- RDL Specification - Supports RDL Specification for the SQL Server 2008 and RDL Specification for the SQL Server 2008 R2 only. List of available report definition formats msdn.microsoft.com/library/dd297486
- Data sources - You can use advanced database servers DataSources in ReportWriter (SQL and Oracle).

Copyright © 2001 - 2020 Syncfusion Inc.

- Charts - Show all basic types of Charts that are available in Microsoft RDL reports.
- Tablix - Shows the summaries and simple tables.
- Gauge - Shows measurement values by using expression values.
- Textbox - Shows textbox data with expression support.
- Export - Export report as PDF, Word, Excel, PPT and HTML.
- Report Parameter - Views the report based on the report parameter value.

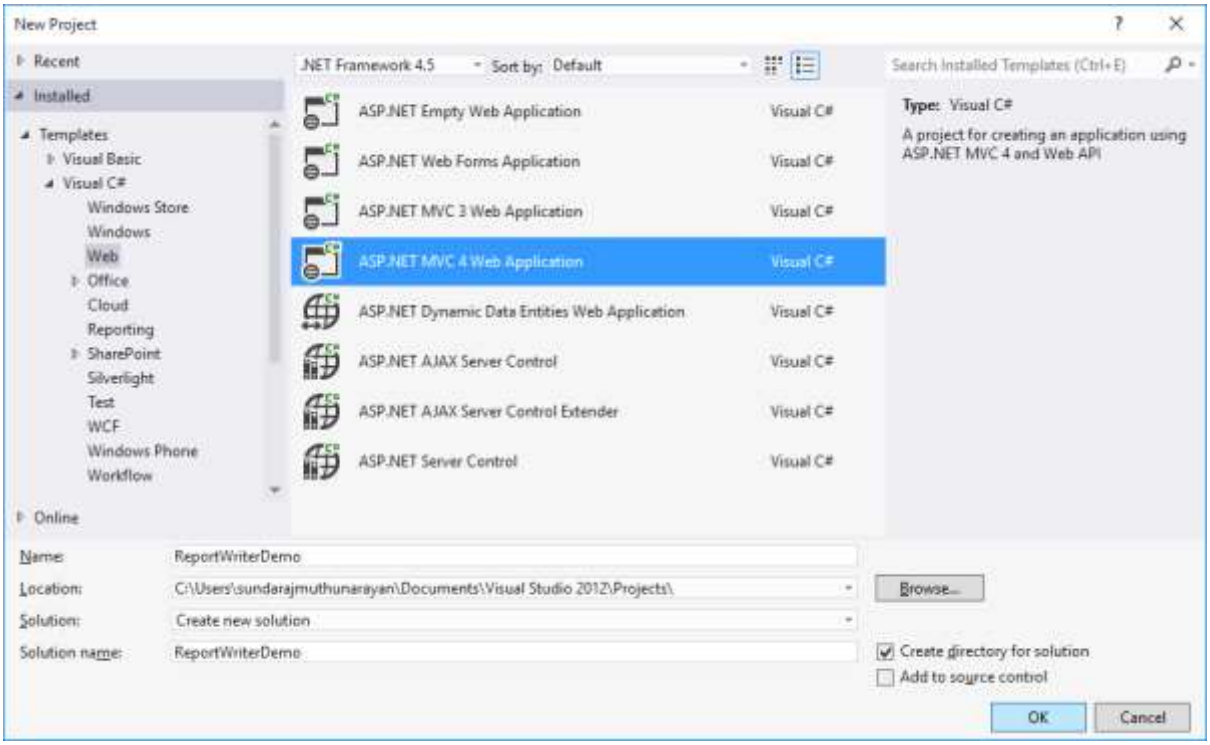
Getting Started with ASP.NET MVC Application

This section describes how to export the Report Server report as PDF, Word, Excel, PPT and HTML formats in ASP.NET MVC application using ReportWriter.

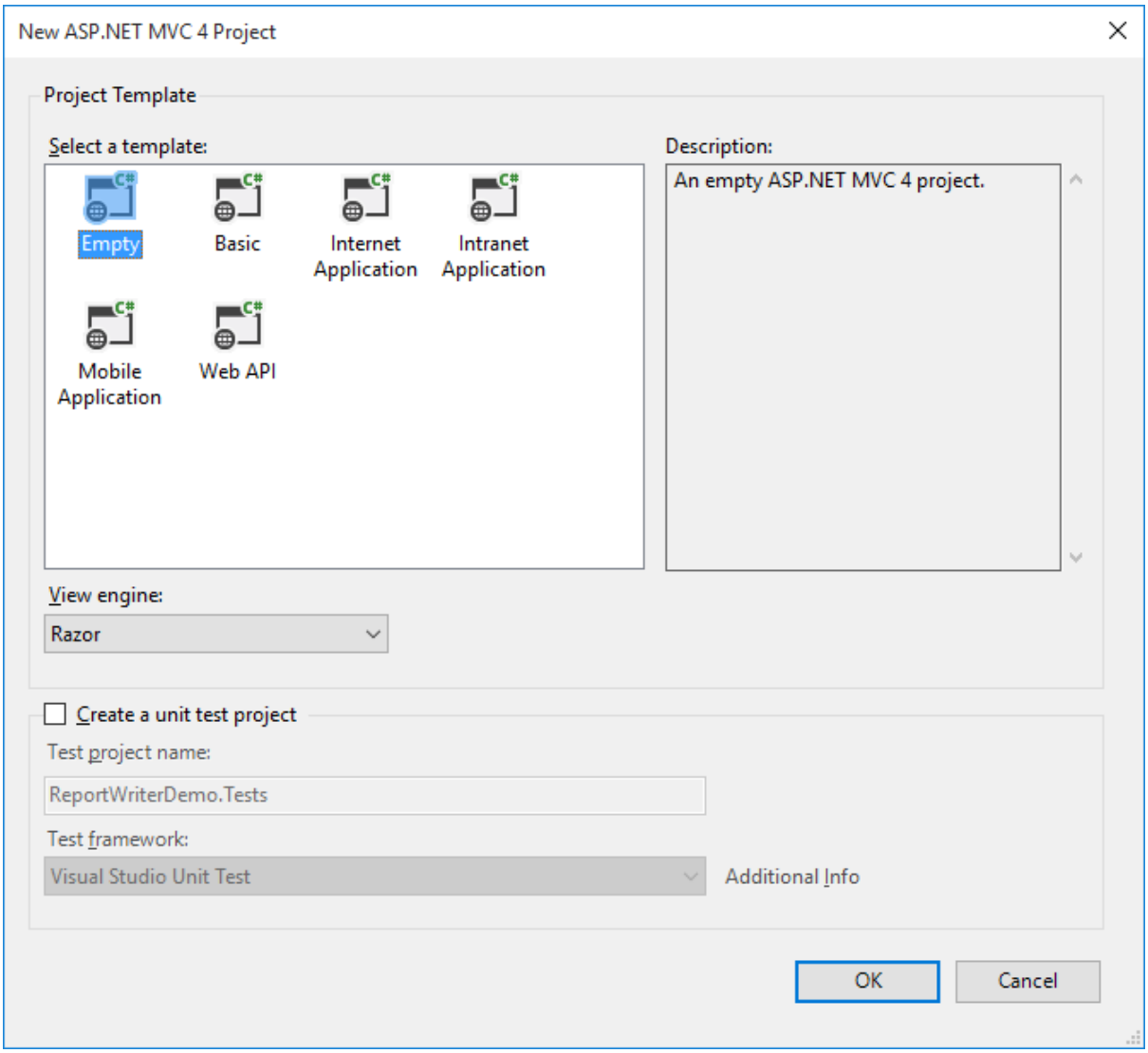
Project Creation

This section illustrates how to add ReportWriter to the ASP.NET MVC application. It includes the following steps.

1. Create a new ASP.NET MVC project by selecting the WEB category from the listed project template in Microsoft Visual Studio IDE.



2. The following screenshot displays how to select empty application template with razor view engine.

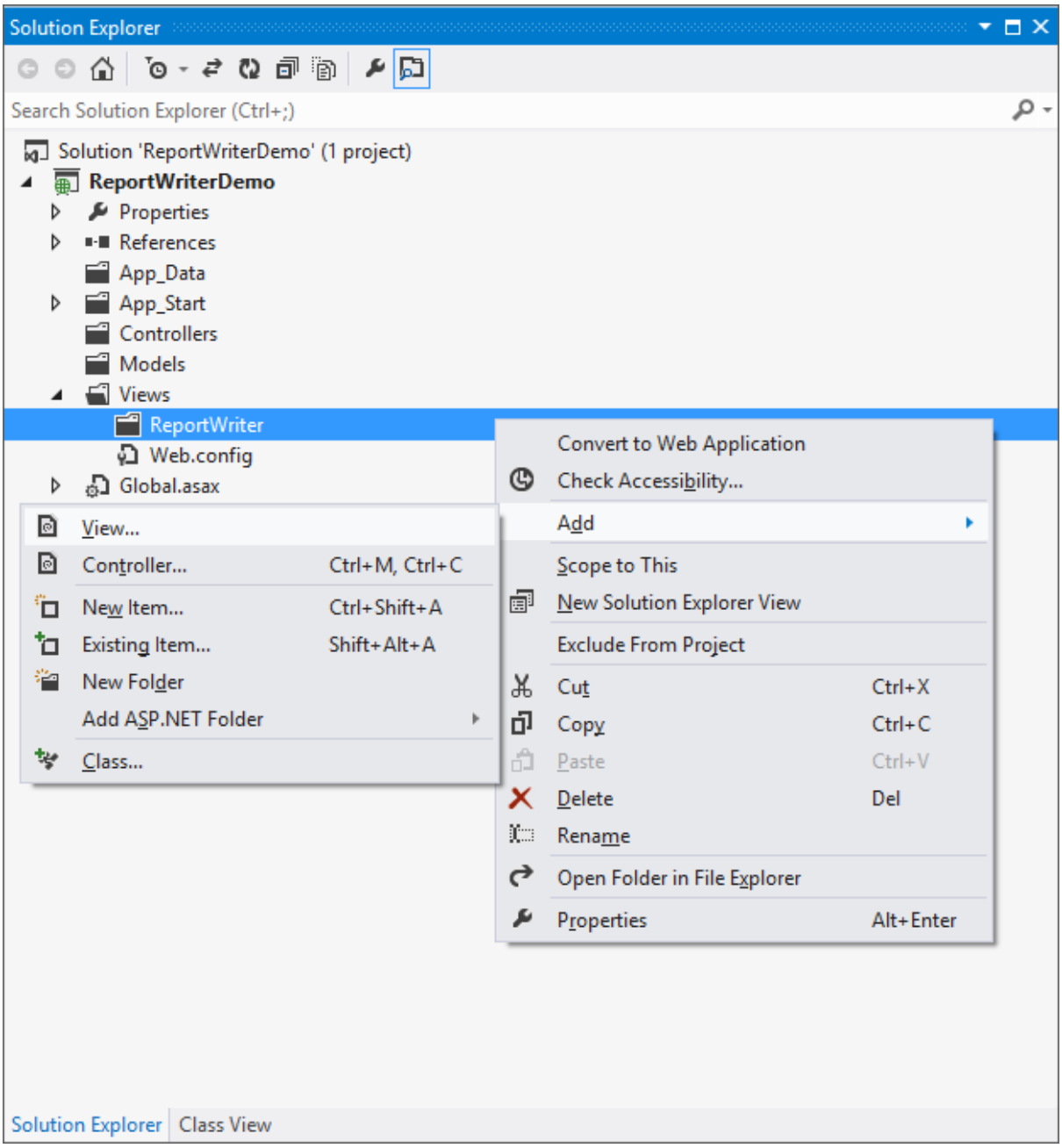


Create CSHTML Page

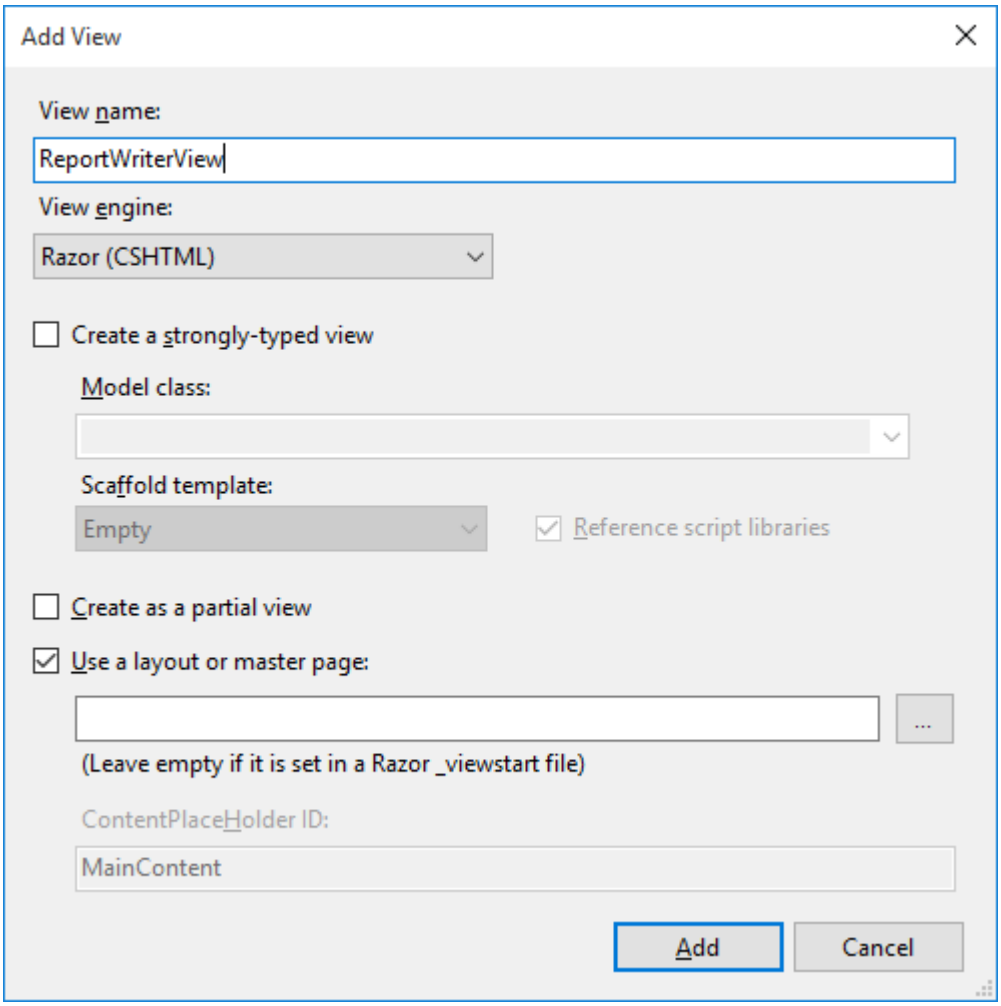
To create a new CSHTML view page in the application follow the below steps.

1. Right click on the ReportWriter folder from Views followed by Add and then click View.

Copyright © 2001 - 2020 Syncfusion Inc.

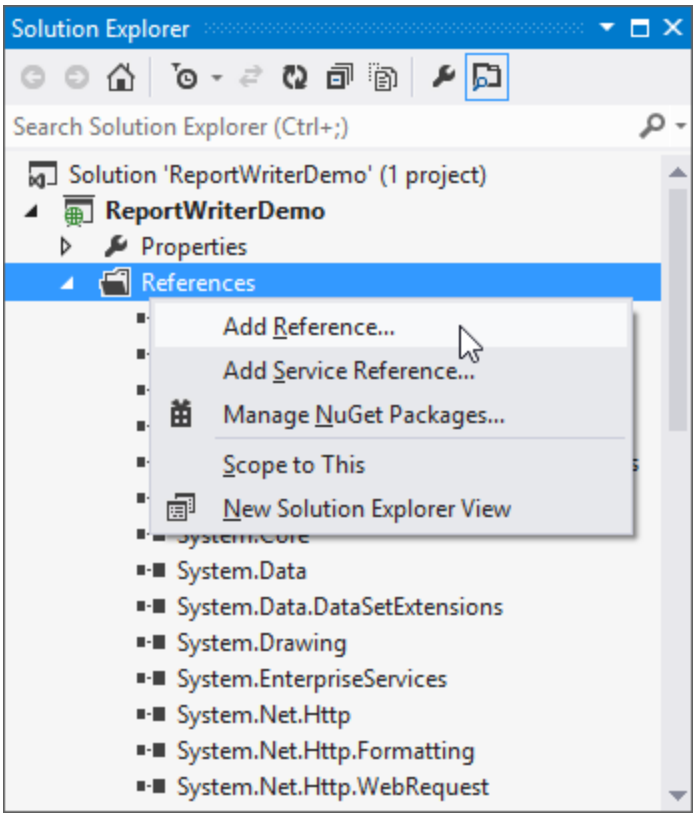


2. Specify the name and then click on OK.



Add References

1. In the Solution Explorer, Right-click the References folder and then click Add Reference.



2. Add the following references
    - Syncfusion.Chart.Wpf
    - Syncfusion.Compression.Base
    - Syncfusion.DocIO.Base
    - Syncfusion.EJ.ReportViewer
    - Syncfusion.Gauge.Wpf
    - Syncfusion.Linq.Base
    - Syncfusion.Pdf.Base
    - Syncfusion.SfMaps.Wpf
    - Syncfusion.Shared.Wpf
    - Syncfusion.XlsIO.Base

Note: Refer the above assemblies from the installed location, C:\Program Files (x86)\Syncfusion\Essential Studio\ASP.NET MVC\{{ site.releaseversion }}\Assemblies

Registering Assemblies within the Web.config file

In your application's root web.config file, add the below assembly information within the <assemblies> tag.

HTML

```
<system.web>
<compilation debug="true" targetFramework="4.5">
<assemblies>
<add assembly="Syncfusion.Chart.WPF, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
```

```
<add assembly="Syncfusion.Compression.Base, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.DocIO.Base, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.EJ.ReportViewer, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.Gauge.WPF, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.Linq.Base, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.Pdf.Base, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.SfMaps.WPF, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.Shared.WPF, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
<add assembly="Syncfusion.XlsIO.Base, Version=14.1400.0.42,
Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
</assemblies>
</compilation>
</system.web>
```

Registering namespaces within Web.config

Now you need to register the below mentioned two namespaces in the web.config file present within the Views folder as well as the Root directory of your application.

- Syncfusion.EJ.ReportViewer

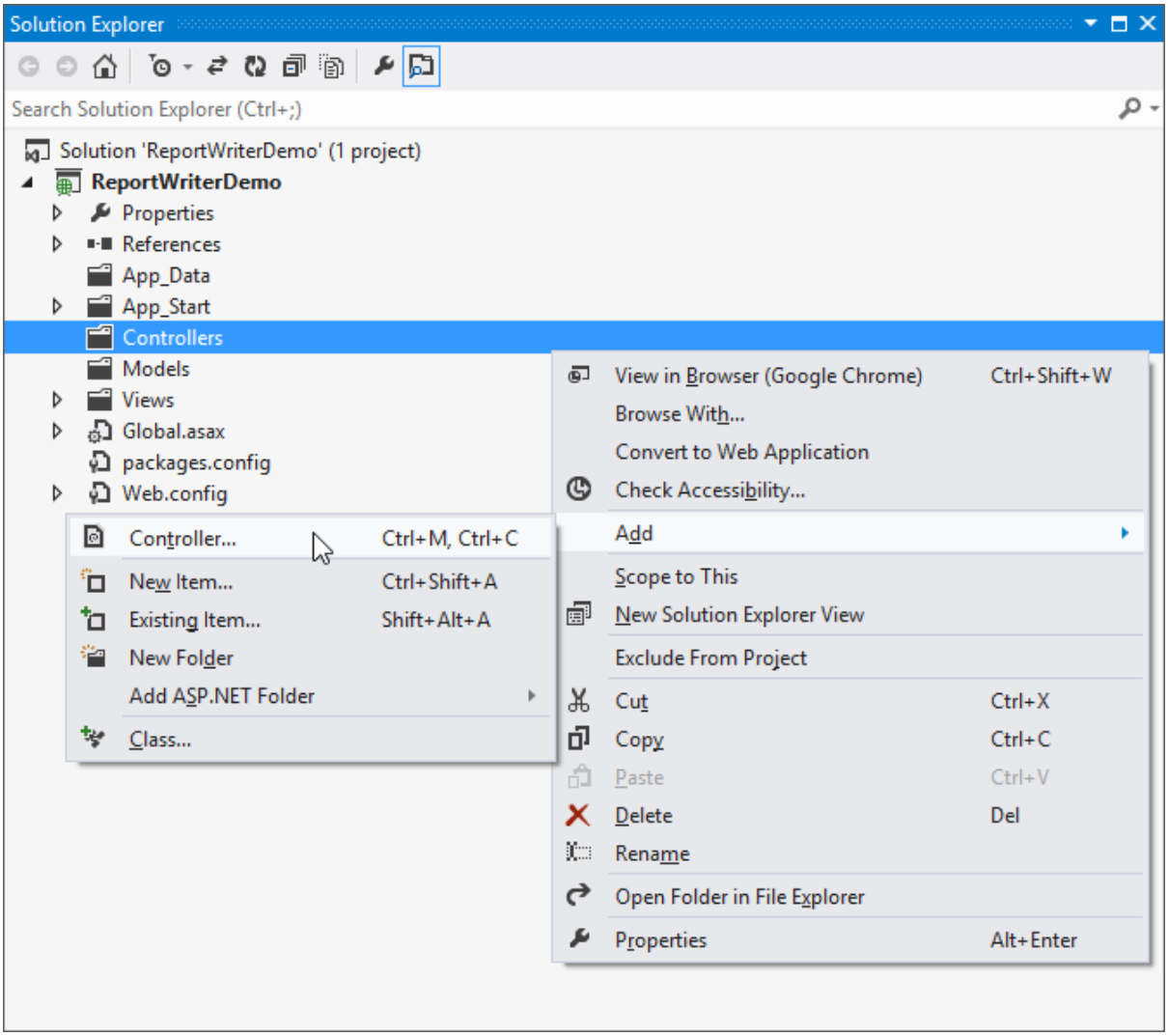
HTML

```
<namespaces>
<add namespace="System.Web.Mvc"/>
<add namespace="System.Web.Mvc.Ajax"/>
<add namespace="System.Web.Mvc.Html"/>
<add namespace="System.Web.Routing"/>
<add namespace="System.Web.WebPages"/>
<add namespace="Syncfusion.EJ.ReportViewer"/>
</namespaces>
```

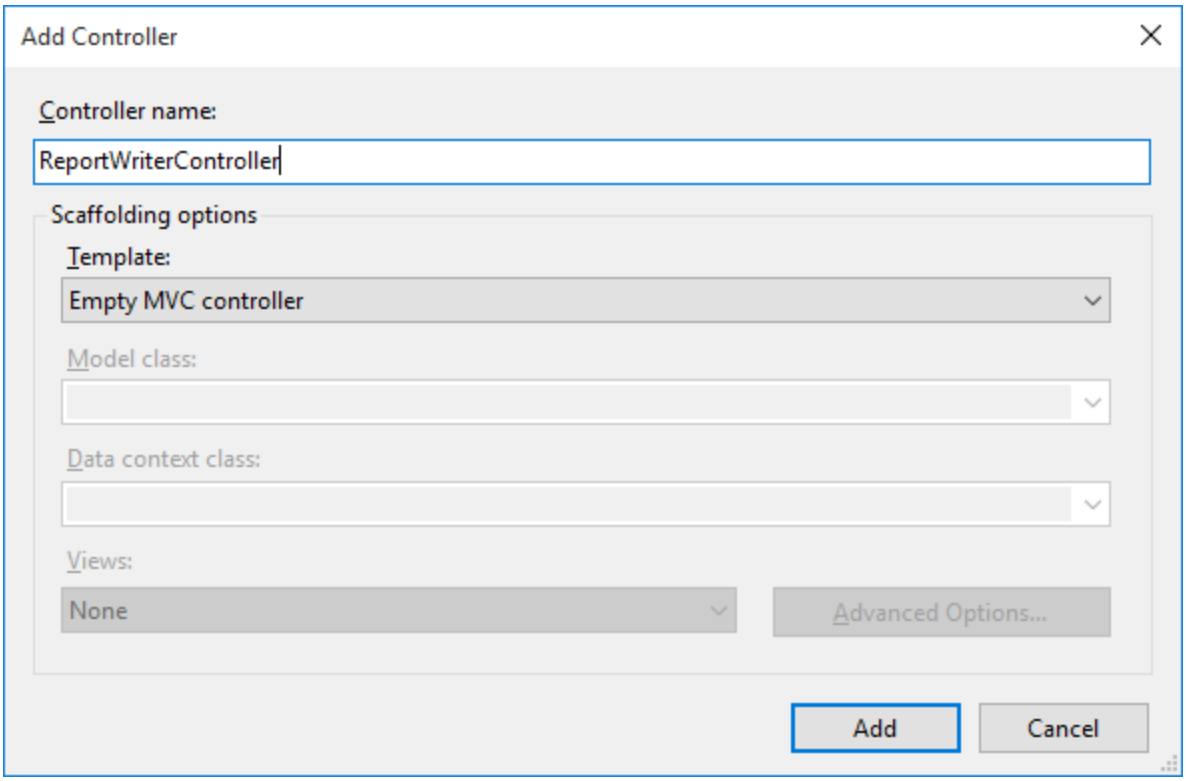
ReportWriter Initialization

1. Initialize ReportWriter by using the controller in the Application. To create a controller in the Application, please follow the below steps.

Copyright © 2001 - 2020 Syncfusion Inc.



2. Specify the name to the controller and then click on OK.



4. Set the following properties to ReportWriter.
    - ReportPath - Set the local file system or Report Server path of report.
    - ReportProcessingMode - Set ProcessingMode as Remote for RDL and Report Server report also Local for RDLC report.
    - WriterFormat - Set WriterFormat as PDF, Excel, Word and HTML.
5. Add the following code to the ReportWriter in the controller class.

CSHARP

```
public ActionResult Index(string writerFormat)
{
try
{
string fileName = null;
WriterFormat format;
HttpContext httpContext = System.Web.HttpContext.Current;
ReportWriter reportWriter = new ReportWriter();
reportWriter.ReportPath = Server.MapPath("~/App_Data/GroupingAgg.rdl");
reportWriter.ReportProcessingMode = ProcessingMode.Remote;
if (writerFormat == "PDF")
{
fileName = "GroupingAgg.pdf";
format = WriterFormat.PDF;
}
else if (writerFormat == "Word")
{
fileName = "GroupingAgg.doc";
format = WriterFormat.Word;
}
```

```
else if (writerFormat == "Html")
{
fileName = "GroupingAgg.Html";
format = WriterFormat.HTML;
}
else if (writerFormat == "PPT")
{
fileName = "GroupingAgg.ppt";
format = WriterFormat.PPT;
}
else
{
fileName = "GroupingAgg.xls";
format = WriterFormat.Excel;
}
reportWriter.Save(fileName, format, httpContext.Response);
}
catch
{
}
return View();
}
```

6. Add the following code example in the <body> tag of the CSHTML page to view ReportWriter export options.

HTML

```
<body>
<div class="container">
<div class="header">
<span style="color: white; font-size: 2em">ReportWriter Demo</span>
</div>
<div class="content_section">
@{Html.BeginForm("Index", "ReportWriter", FormMethod.Post);
{
<div id="description_Pane" style="text-align: justify;">
<h3>Description</h3>
<span>
Essential ReportWriter is a powerful control for exporting RDL files into
specified
format files.The following are some of the key features of this
component: Provides
support for various chart types that include Area, Bar, Column, Pie,
Funnel, Radar, Line and
so on. Provides support for both Matrix and Table and also provides
support for
both single, and multi-level row groupings and column groupings. Provides
support
for other common controls such as Textbox, Image, Rectangle, Line and
both Circular
and Linear Gauges. Provides support for report parameter.
</span>
</div>
<div id="export_Pane" style="margin-top: 4%;">
```

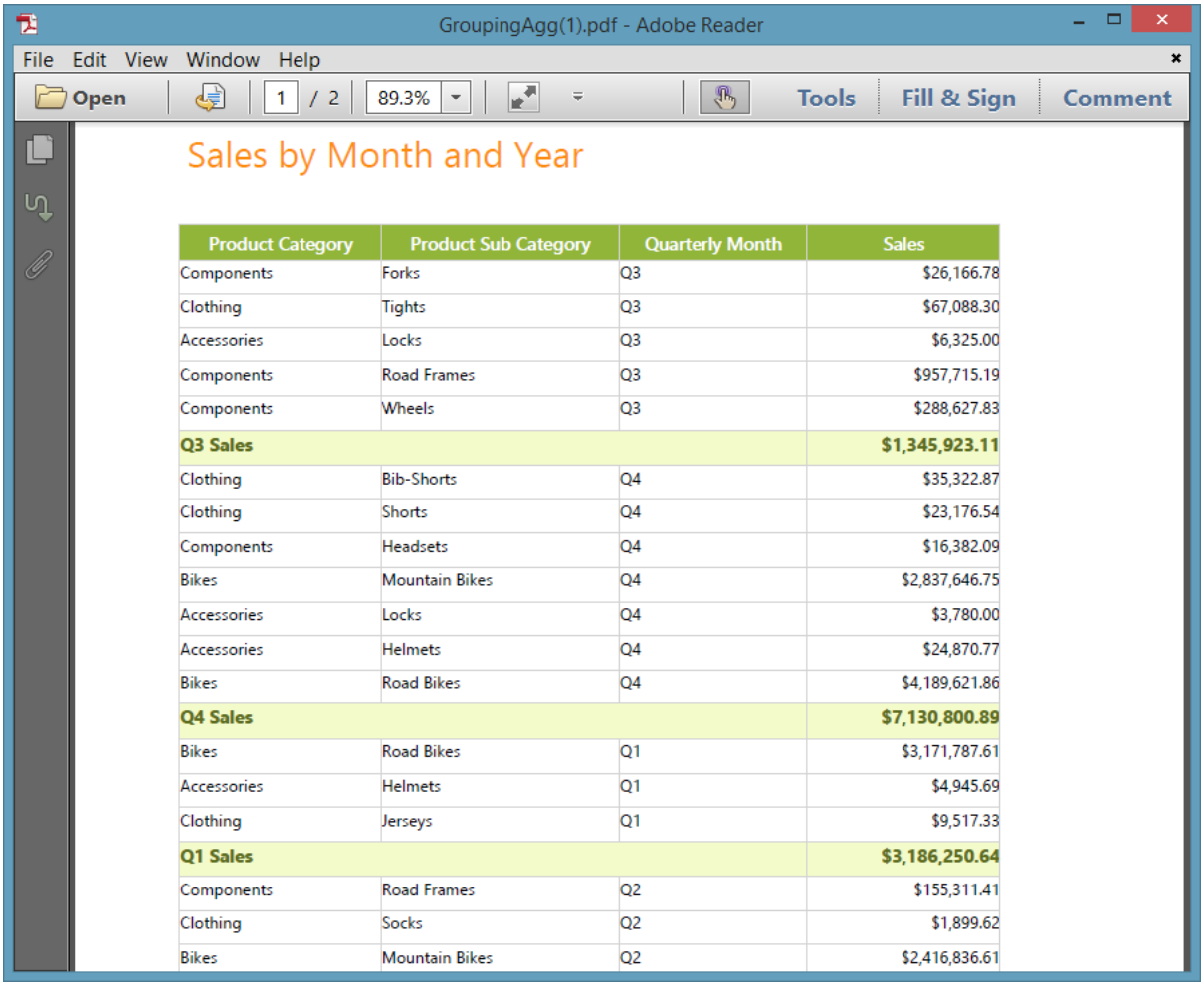
Copyright © 2001 - 2020 Syncfusion Inc.

```
<h3>Export Report</h3>
<span>
Choose a file format to view the selected document generated from Report
file by using Essential ReportWriter.
</span>
<div id="selection_Pane" style="margin-top: 2%;">
<span style="font-size:large;">
<strong>Save As : </strong>
</span>
<input type="radio" name="writerFormat" value="PDF" checked="checked" />
<label for="btnPDf">PDF </label>
<input type="radio" name="writerFormat" value="Word" />
<label for="btnWord">Word </label>
<input type="radio" name="writerFormat" value="xls" />
<label for="btnxls">Excel </label>
<input type="radio" name="writerFormat" value="Html" />
<label for="btnHtml">Html </label>
<input type="radio" name="writerFormat" value="PPT" />
<label for="btnHtml">PPT </label>
<input type="submit" name="button" value="Generate" style="width: 18%;
margin-left: 2%;"/>
</div>
</div>
Html.EndForm();
}
}
</div>
</div>
</body>
```

7. Run the application. The following output displays ReportWriter export options, Select the export option and click on Generate button.



8. The following output displays exported report in the PDF format.



Product Category	Product Sub Category	Quarterly Month	Sales
Components	Forks	Q3	$26,166.78
Clothing	Tights	Q3	$67,088.30
Accessories	Locks	Q3	$6,325.00
Components	Road Frames	Q3	$957,715.19
Components	Wheels	Q3	$288,627.83
Q3 Sales			**$1,345,923.11**
Clothing	Bib-Shorts	Q4	$35,322.87
Clothing	Shorts	Q4	$23,176.54
Components	Headsets	Q4	$16,382.09
Bikes	Mountain Bikes	Q4	$2,837,646.75
Accessories	Locks	Q4	$3,780.00
Accessories	Helmets	Q4	$24,870.77
Bikes	Road Bikes	Q4	$4,189,621.86
Q4 Sales			**$7,130,800.89**
Bikes	Road Bikes	Q1	$3,171,787.61
Accessories	Helmets	Q1	$4,945.69
Clothing	Jerseys	Q1	$9,517.33
Q1 Sales			**$3,186,250.64**
Components	Road Frames	Q2	$155,311.41
Clothing	Socks	Q2	$1,899.62
Bikes	Mountain Bikes	Q2	$2,416,836.61

Generate RDL Reports

The ReportWriter has options to save the RDL reports. The following code example helps you to generate the RDL report using ReportWriter.

Specify the `ReportPath`, `ReportProcessingMode` and `WriterFormat` properties for ReportWriter to generate report.

CSHARP

```
string reportPath = @"..\ReportTemplate\GroupingAgg.rdl";
ReportWriter reportWriter = new ReportWriter(reportPath);
reportWriter.ReportProcessingMode = ProcessingMode.Remote;
reportWriter.Save("GroupingAgg.xls", WriterFormat.Excel);
```

Generate RDLC Reports

The ReportWriter has support to view and save the RDLC reports. The following code helps you to bind data to ReportWriter.

1. Assign `ReportPath`, `ProcessingMode` and `ExportFormat` to ReportWriter.

CSHARP

Copyright © 2001 - 2020 Syncfusion Inc.

```
string reportPath = @"..ReportTemplate\RDLC\ProductCatalog.rdlc";
ReportWriter reportWriter = new ReportWriter(reportPath, dataSources);
reportWriter.ReportProcessingMode = ProcessingMode.Local;
reportWriter.Save("ProductCatalog.doc", WriterFormat.WORD);
```

2. Add Datasource to the RDLC report.

CSHARP

```
ReportDataSourceCollection dataSources = new
ReportDataSourceCollection();
dataSources.Add(new ReportDataSource { Name = "ProductCatalog", Value =
ProductCatalogSource.GetData() });
```

3. Assign values for the datasource which is given in the RDLC.

CSHARP

```
#region ProductCatalog Details
public class ProductCatalogSource
{
public string ProdSubCat { get; set; }
public string ProdModel { get; set; }
public string ProdCat { get; set; }
public string Description { get; set; }
public string ProdName { get; set; }
public string ProductNumber { get; set; }
public string Color { get; set; }
public string Size { get; set; }
public double? Weight { get; set; }
public double? StandardCost { get; set; }
public string Style { get; set; }
public string Class { get; set; }
public double? ListPrice { get; set; }
public static IList GetData()
{
List<ProductCatalogSource> datas = new List<ProductCatalogSource>();
ProductCatalogSource data = null;
data = new ProductCatalogSource()
{
ProdSubCat = "Road Frames",
ProdModel = "HL Road Frame",
ProdCat = "Components",
Description = "Our lightest and best quality aluminum frame made from the
newest alloy; it is welded and heat-treated for strength. Our innovative
design results in maximum comfort and performance.",
ProdName = "HL Road Frame - Black, 58",
ProductNumber = "FR-R92B-58",
Color = "Black",
Size = "58",
Weight = 2.24,
StandardCost = 1059.3100,
Style = "U ",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Class = "H ",
ListPrice = 1431.5000
};
datas.Add(data);
data = new ProductCatalogSource()
{
ProdSubCat = "Road Frames",
ProdModel = "HL Road Frame",
ProdCat = "Components",
Description = "Our lightest and best quality aluminum frame made from the
newest alloy; it is welded and heat-treated for strength. Our innovative
design results in maximum comfort and performance.",
ProdName = "HL Road Frame - Red, 58",
ProductNumber = "FR-R92R-58",
Color = "Red",
Size = "58",
Weight = 2.24,
StandardCost = 1059.3100,
Style = "U ",
Class = "H ",
ListPrice = 1431.5000
};
datas.Add(data);
data = new ProductCatalogSource()
{
ProdSubCat = "Helmets",
ProdModel = "Sport-100",
ProdCat = "Accessories",
Description = "Universal fit, well-vented, lightweight , snap-on visor.",
ProdName = "Sport-100 Helmet, Red",
ProductNumber = "HL-U509-R",
Color = "Red",
Size = "",
Weight = null,
StandardCost = 13.0863,
Style = "",
Class = "",
ListPrice = 34.9900
};
datas.Add(data);
return datas;
}
}
#endregion
```

Generate SSRS Reports

The ReportWriter has options to export the SSRS reports. The following code example helps you to generate the SSRS report using ReportWriter.

Specify the ReportPath, ReportServerUrl, ReportServerCredential, ReportProcessingMode and WriterFormat properties for ReportWriter to generate report.

CSHARP

```
HttpContext httpContext = System.Web.HttpContext.Current;
ReportWriter reportWriter = new ReportWriter();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
reportWriter.ReportPath = "45db67a0-3fd6-4684-b03c-aa640a521c97";
reportWriter.ReportServerUrl = "http://reportserver.syncfusion.com:80/";
reportWriter.ReportingServer = new ReportingServerExt();
reportWriter.ReportServerCredential = new
System.Net.NetworkCredential("guest", "demo");
reportWriter.ReportProcessingMode = ProcessingMode.Remote;
reportWriter.Save("GroupingAgg.doc", WriterFormat.Word,
httpContext.Response);
```

Exporting Reports

Essential ReportWriter provides support for Exporting a report as a PDF, Word, Excel, PPT and HTML documents with the help of the class ReportWriter. The report elements such as Tablix, matrices, charts, gauges, shapes, and text boxes are supported in this feature.

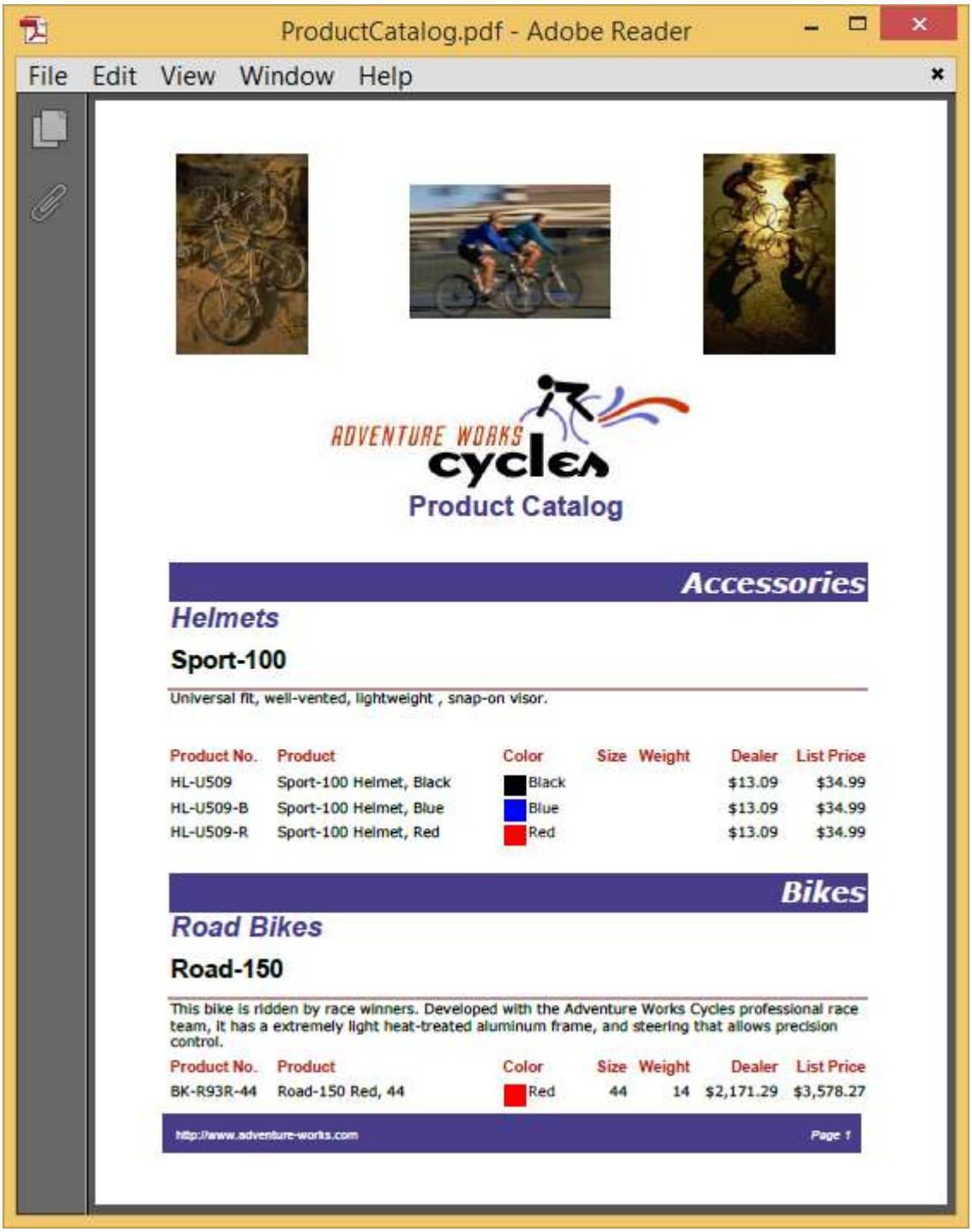
Exporting Report as PDF

The report used in ReportWriter can be exported as PDF document using the following code.

HTML

```
ReportWriter reportWriter = new ReportWriter(reportpath, dataSources);
reportWriter.Save("Sample.pdf", WriterFormat.PDF);
```

Copyright © 2001 - 2020 Syncfusion Inc.

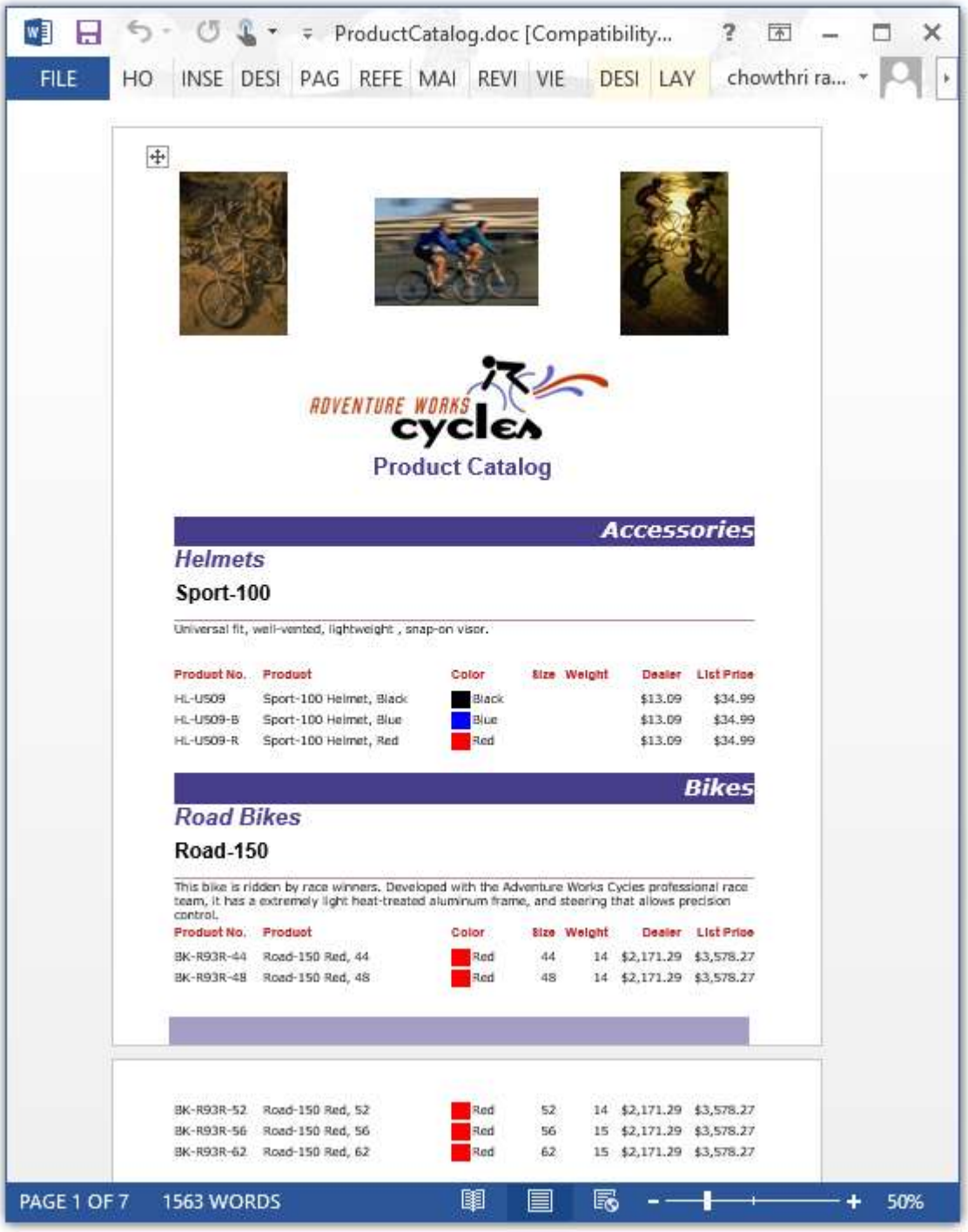


Exporting Report as Word

The report used in ReportWriter can be exported as Word document using the following code.

HTML

```
ReportWriter reportWriter = new ReportWriter(reportpath, dataSources);
reportWriter.Save("Sample.doc", WriterFormat.Word);
```



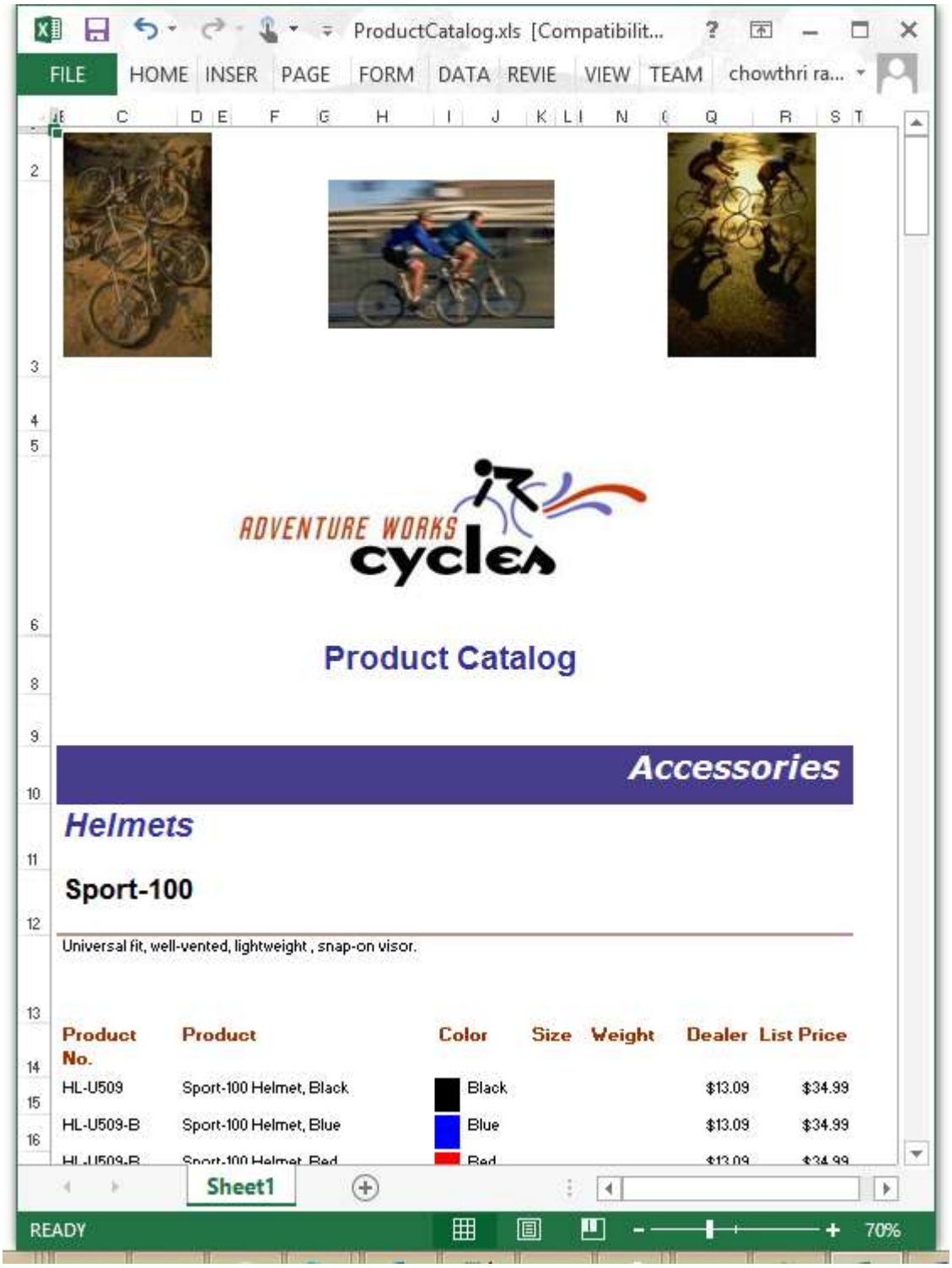
Exporting Report as Excel

The report used in ReportWriter can be exported as Excel document using the following code.

HTML

```
ReportWriter reportWriter = new ReportWriter(reportpath, dataSources);
reportWriter.Save("Sample.xls", WriterFormat.Excel);
```

Copyright © 2001 - 2020 Syncfusion Inc.

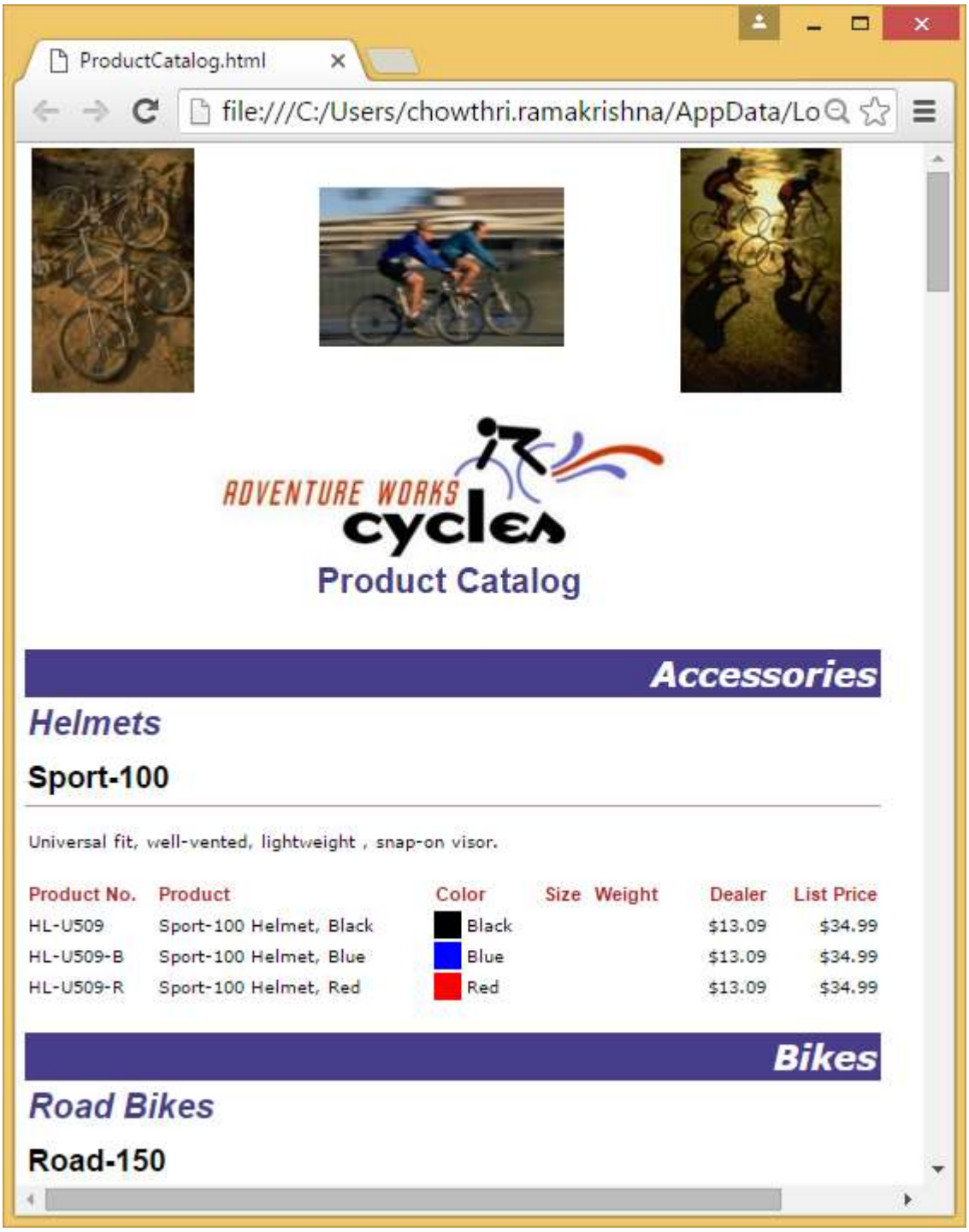


Exporting Report as HTML

The report used in ReportWriter can be exported as HTML document using the following code.

HTML

```
ReportWriter reportWriter = new ReportWriter(reportpath, dataSources);
reportWriter.Save("Sample.html", WriterFormat.HTML);
```



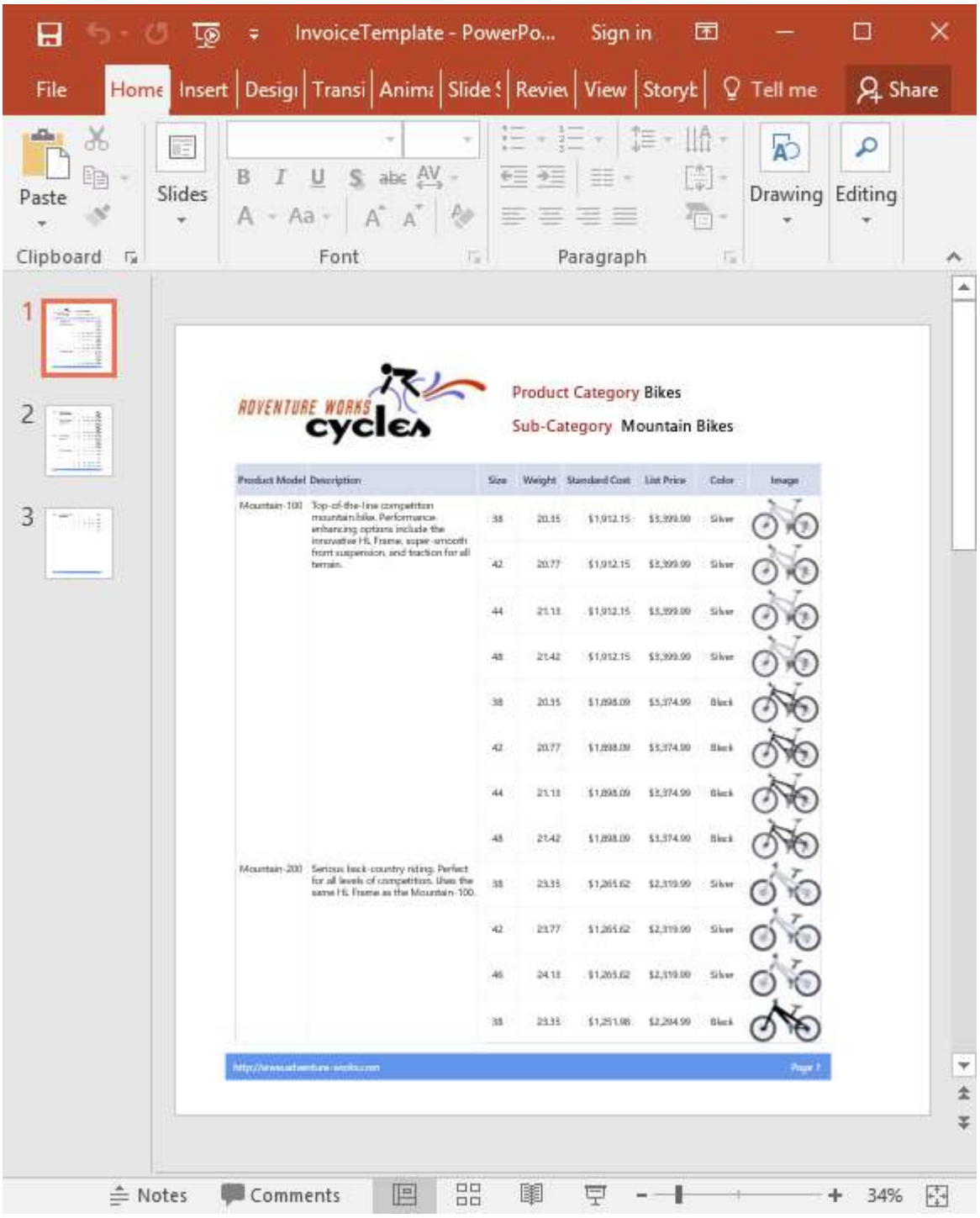
Exporting Report as PPT

The report used in ReportWriter can be exported as PPT document using the following code.

HTML

```
ReportWriter reportWriter = new ReportWriter(reportpath, dataSources);
reportWriter.Save("Sample.ppt", WriterFormat.PPT);
```

Copyright © 2001 - 2020 Syncfusion Inc.



ReportWriter API

Constructors

Name	Description
ReportWriter()	Initializes a new instance of the ReportWriter.

ReportWriter(string)	Initializes a new instance of the ReportWriter with the specified path of the report.
ReportWriter(stream)	Initializes a new instance of the ReportWriter with the stream of the report.
ReportWriter(string, ReportDataSourceCollection)	Initializes a new instance of the ReportWriter with the specified path of the report and data source collection.
ReportWriter(stream, ReportDataSourceCollection)	Initializes a new instance of the ReportWriter with the stream of the report and data source collection.

Properties

Property Name	Description
DataSources	Gets or sets the data source collection.
ReportPath	Gets or sets the report path (file name of the report with its full path).
ReportProcessingMode	Gets or sets the processing mode (either local or remote).
ReportServerCredential	Gets or sets the Report Server credential.
ReportServerFormsCredential	Gets or sets the Report Server form credential. This is used to export SQL Azure RDL reports.
ReportServerUrl	Gets or sets the ReportServerURL.

Methods

Name	Description
GetDataSetNames()	Returns the names of the data set as IList <string>.
GetParameters()	Returns the parameters of the report as ReportParameterInfoCollection.
SetParameters(IEnumerable<ReportParameters>)	Sets the parameters of the report.
LoadReport(stream)	Loads the report with the specified stream that contains the RDL contents.
Save(string,writer type)	Saves the report as a PDF, Word, Excel, PPT and HTML documents in the mentioned path.
Save(stream)	Saves the report as a PDF, Word, Excel, PPT and HTML documents in the stream.

Copyright © 2001 - 2020 Syncfusion Inc.

Events

Events	Description
ReportError	This event is triggered when Report throws Error.
ExportCompleted	This event is triggered when export is completed.
SubreportProcessing	This event is triggered when the report is RDLC and contains with subreport.

How To

How to

Save a Report as a Stream?

You can save a report as a stream using Save(Stream,WriterFormat) (overloaded method). This method is useful when generating the report on the server side and sending it to a client-side application. The following code explains how to save a report as a stream.

CSHARP

```
//Step 1: create the report data source
ReportDataSourceCollection dataSources = new
ReportDataSourceCollection();
dataSources.Add(new ReportDataSource() { Name = "Sales", Value =
GetDataSource() });
//Step 2: Instantiate the ReportWriter with the parameter "ReportPath"
and ReportDataSource Collection
ReportWriter reportWriter = new ReportWriter(reportPath, dataSources);
MemoryStream stream = new MemoryStream();
//Step 3: Save the report as PDF, Word or Excel document in the form of
stream contents
reportWriter.Save(stream, WriterFormat.PDF);
```

Ribbon

Overview

The Ribbon control for ASP.NET MVC provides with rich customizable user interfaces like Office 2010, SharePoint 2010, and Office Web Apps 2010. The Ribbon Tab appears across the top of the page. Each Tab organizes a set of groups that has labels to identify them and also contains a set of controls and group expander.

Key Feature

- **Application Tab** -Represents the menu commands which can used to do any operations, like file related commands.Supports Application Menu and Backstage.
- **Tabs** – Related groups are combined into Tabs for quick and easy access.
- **Contextual Tabs** - Allows to group number of Tabs based on some criteria.
- **Controls' support** -Supports Button, Split button and Dropdown List, Toggle button, and custom controls in the Ribbon groups.
- **Gallery** - Convenient way to visually display related options with text/images.
- **Resize** – Provides resizing to group controls dynamically based on window size.
- **Expand and Collapse** -Ribbon can be expanded and collapsed using expand/collapse button.

Copyright © 2001 - 2020 Syncfusion Inc.

- **Screen-Tip** -Supports HTML and enhanced custom tooltips for controls of Ribbon groups.
- **Theme** - **ASP.NET MVC** controls consist of 12 built-in themes (6 – flat and 6 – gradient effects). It also supports bootstrap theme.

Getting Started

This section explains briefly how to create a Ribbon in your application with ASP.NET MVC. You can refer MVC Getting Started documentation to create new project and add necessary dll's and script files.

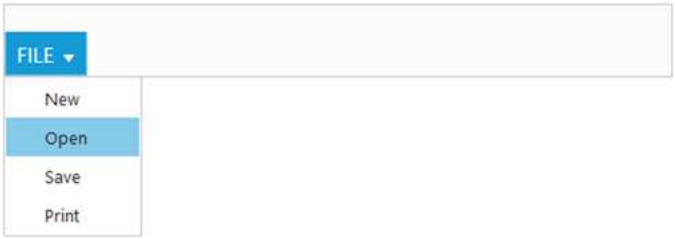
Control Initialization

Ribbon can be initialized with ApplicationTab and UL list is needed for binding Menu to application Menu which can be specified through MenuItemID which denotes Id of UL.

Define the Application Tab with Type as Menu to render simple Ribbon control.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = false
});
}))
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
```



Note: Set the required Width to Ribbon, else default parent container or window Width will be considered.

Adding Tabs

RibbonTab is a set of related groups which are combined into single item. For creating Tab, Id and Text properties should be specified.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = false
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tabgroup => {}).Add();
})
)
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
```



Configuring Groups

List of controls are combined as logical ContentGroups into RibbonTabs. TabGroup alignment type as “Rows/Columns”, Default is Rows.

Create TabGroup item with Text specified and add ContentGroup to ContentGroup collection with Button control settings.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = false
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tabgroup => {
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn => {
ctn.ContentGroups(contentGroup => {
contentGroup.Id("new").Text("New").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-Ribbon e-new",
```

```
Click = "executeAction"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("70px")).Add();
}).Add();
}).Add();
}))
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
}
@section StyleSection{
<link href="~/Content/ej/Ribbon-css/ej.icons.css" rel="stylesheet" />
}
```



Adding Controls to Group

Syncfusion ASP.NET MVC Controls can be added to TabGroup's content with corresponding Type specified like Button, SplitButton, ToggleButton, DropDownList, Gallery, Custom, etc. Default type is Button.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = false
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tabGroup => {
tabGroup.Text("SplitButton &
Dropdown").AlignType(RibbonAlignType.Columns).Content(ctn => {
ctn.ContentGroups(contentGroup => {
contentGroup.Id("paste").Text("Paste").SplitButtonSettings(new
SplitButtonProperties() {
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-Ribbon e-Ribbonpaste",
TargetID = "pasteSplit",
ArrowPosition = ArrowPosition.Bottom,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ButtonMode = ButtonMode.Dropdown
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.SplitButton).Width("50px").Height("70px")).Add()
;
ctn.ContentGroups(contentGroup => {
contentGroup.Id("fontFamily").DropdownSettings(new
DropDownListProperties() {
DataSource = (IEnumerable < FontFamily > ) ViewBag.datasource,
Text = "Segoe UI",
Select = "executeAction",
Width = "150px"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.DropDownList).Height("28px")).Add();
}).Add();
}).Add();
}))
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li><a>Paste</a></li>
</ul>
}
@section StyleSection{
<link href="~/Content/ej/Ribbon-css/ej.icons.css" rel="stylesheet" />
<style>
.e-pastetip {
background-image: url("../Content/ej/common-images/Ribbon/paste.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
</style>
}
```

C#

```
public partial class RibbonController: Controller
{
List<FontFamily> fontFamily1 = new List<FontFamily>();
List<FontPoint> fontPoint1 = new List<FontPoint>();
public ActionResult RibbonFeatures()
{
fontFamily1.Add(new FontFamily { text = "Segoe UI" });
fontFamily1.Add(new FontFamily { text = "Arial" });
```

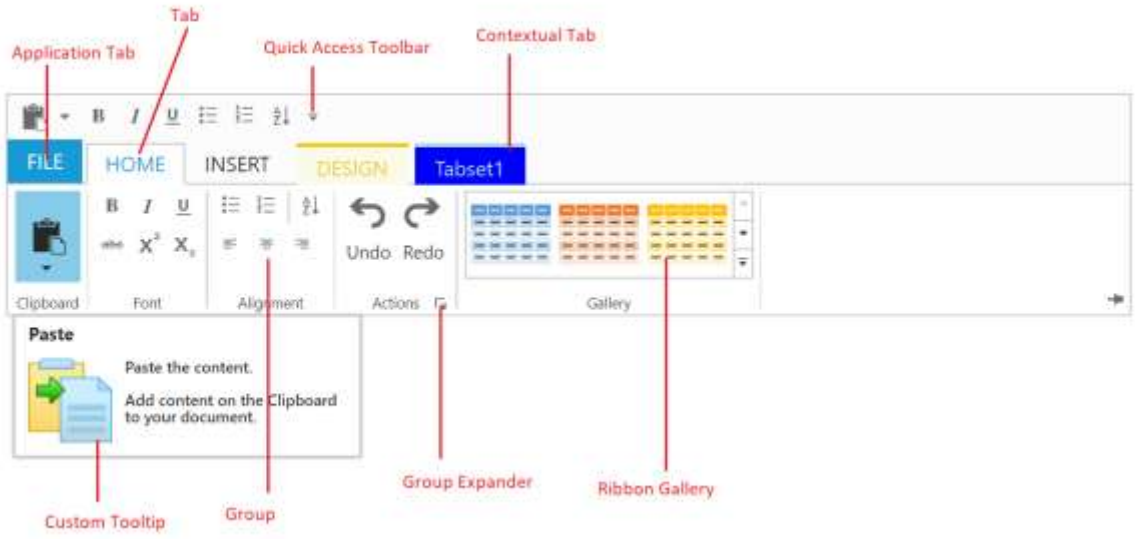
Copyright © 2001 - 2020 Syncfusion Inc.

```
fontFamily1.Add(new FontFamily { text = "Times New Roman" });
fontFamily1.Add(new FontFamily { text = "Tahoma" });
fontFamily1.Add(new FontFamily { text = "Helvetica" });
ViewBag.datasource = fontFamily1;
return View();
}
}
```



User Interface

Ribbon component able to integrate any custom components and customized their functionality in application end. Our Ribbon component is similar to Microsoft products(Word). The Ribbon UI consists of several sections like Application Tab, Quick Access Toolbar, Tab, Contextual Tab, Gallery and etc.The following screenshot shows the diagrammatic detail of Ribbon UI:



From above screenshot, you can see Ribbon has several subcomponents for different functionalities. The upcoming sections explains the brief details of each functionalities and their customizations.

Ribbon Dependencies

ej.web.all.js is a bundle of all *ASP.NET MVC* controls. When you use *ej.web.all.js* in your application, you can leave this section or else you can try to render *Ribbon* in your application by using *ej.ribbon* file. You can refer to the following frameworks and controls in your project.

Ribbon Dependency

FilesÂ Â Â Â Â Â Â Â Â Â Â Â Â Â	Description/Usage

Copyright © 2001 - 2020 Syncfusion Inc.

ej.core.min.js	Must always be referred to before using all the *ASP.NET MVC*controls.
ej.data.min.js	Used to handle data manger operation and should be used while binding data to *ASP.NET MVC* controls.
ej.globalize.min.js	Must be referred to localize any of the JS control's text and content.
ej.ribbon.min.js	Should be referred when using *Ribbon*control.
ej.waitingpopup.min.js	This file is used to render waiting popup when on-demand functionality is enabled in *Ribbon*control.
ej.menu.min.js	This file is used to render menu in the application tab.
ej.scroller.min.js	This file is used to render scroller in the *Ribbon* control.
ej.checkbox.min.js	This file is used to render checkboxes in the *Ribbon* control.
ej.tab.min.js	This file is used to render tabs into the *Ribbon* control.
ej.dropdownlist.min.js	These files are used to render button,split button,toggle button, and dropdown list controls in the ribbon groups.
ej.splitbutton.min.js	
ej.button.min.js	
ej.togglebutton.min.js	

Application Tab

The Application Tab is used to represent a Menu that do some operations, such as File Menu to create, open, and print documents. Application Tab classified by Type property with the following:

Menu Backstage

Application Menu

The Application Menu is similar to traditional file Menu options and Syncfusion Menu control is used internally to render this. To show Application Menu in Ribbon, set the Type as Menu and MenuSettings to customize properties of ejMenu.

*_Create Using Template_*

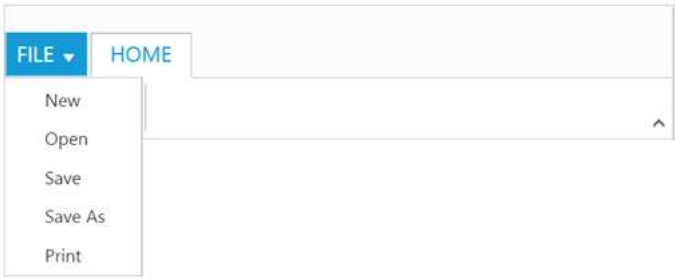
Set the UL element Id to MenuItemID property to create Application Menu and it will acts as template to render Menu.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app=> {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = false
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tab => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
tab.Text("New").ContentID("Contents").Type("custom").AlignType(RibbonAlig
nType.Rows).Content(ctn => {
ctn.ContentGroups(contentGroup => {
contentGroup.Id("new").Text("New").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-Ribbon e-new",
Click = "executeAction"
}).Add();
}).Add();
}).Add();
}).Add();
}))
<div id="Contents">Custom control</div>
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
```



*_Binding Data Source_*

Application Menu can be rendered using JSON Data Source. Please refer this page to set data source to ejMenu.

ASPX-CS

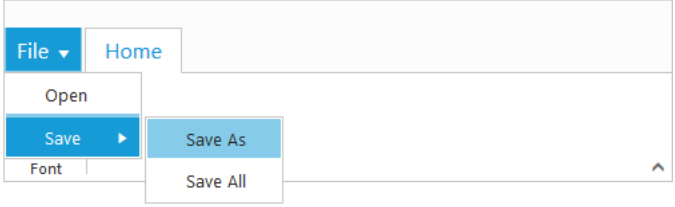
```
@(Html.EJ().Ribbon("MenuJson")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = true,
MenuFields = new MenuFields() {
DataSource = (IEnumerable < MenuJson > ) ViewBag.datasource,
Text = "text",
ParentId = "parent",
Id = "id"
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tab => {
tab.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn => {
ctn.ContentGroups(contentGroup => {
contentGroup.Id("new").Text("New").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-Ribbon e-new",
Click = "executeAction"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("70px")).Add();
}).Add();
}).Add();
})
<ul id="Ribbon"></ul>
}
@section StyleSection{
<link href="~/Content/ej/Ribbon-css/ej.icons.css" rel="stylesheet" />
}
```

C#

```
public partial class RibbonController : Controller
{
List<MenuJson> Ribbon = new List<MenuJson>();
public ActionResult Methods()
{
Ribbon.Add(new MenuJson { id = 1, parent = null, text = "File" });
Ribbon.Add(new MenuJson { id = 11, parent = "1", text = "Open" });
Ribbon.Add(new MenuJson { id = 12, parent = "1", text = "Save" });
Ribbon.Add(new MenuJson { id = 121, parent = "12", text = "Save As" });
Ribbon.Add(new MenuJson { id = 122, parent = "12", text = "Save All" });
ViewBag.datasource = Ribbon;
return View();
}
public class MenuJson
{
public string text { get; set; }
public int id { get; set; }
public string parent { get; set; }
}
}
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

Backstage Page

The Backstage page is where documents and related data of those can be managed, such as Create, Save and other information.

The Backstage page has a feature to add custom Control in left side of the page which contains menu items and the right side contains corresponding user controls.

You can set Application Tab `Type` as `Backstage` and set `Id` , `Text` to Backstage items. Backstage `Pages` can be added with required `ItemType` and `ContentID` as template id to render template into Backstage.

Separator between Backstage items can be enabled by setting `EnableSeparator` as true. Width of backstage side header can be customized using `HeaderWidth`, If not set based on content given width will be considered.

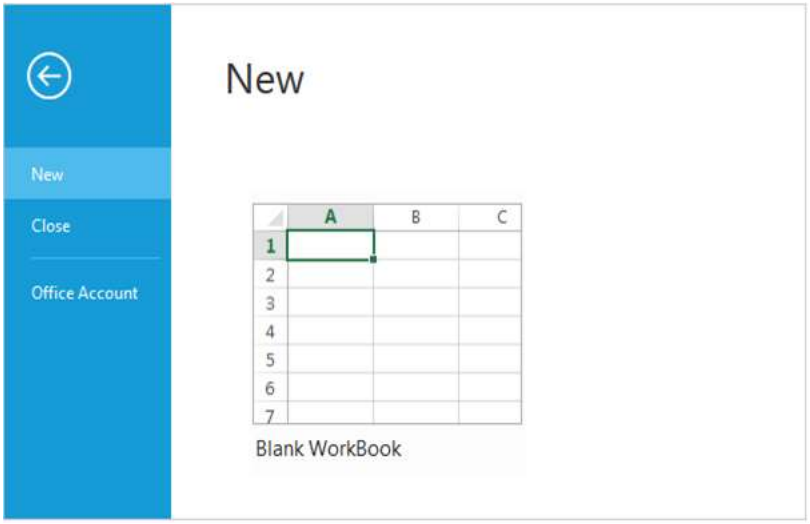
To render the Ribbon with the Backstage page, refer to the following code snippet.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Backstage).BackstageSettings(bsSettings => {
bsSettings.Text("FILE").Height("350").Width("100%").HeaderWidth("120").Pa
ges(bsPage => {
bsPage.Id("new").Text("New").ContentID("newCon").Add();
bsPage.Id("close").Text("Close").EnableSeparator(true).ItemType(ItemType.
Button).Add();
bsPage.Id("account").Text("Office
Account").ContentID("accountCon").Add();
});
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tab => {
tab.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn => {
ctn.ContentGroups(contentGroup => {
contentGroup.Id("new").Text("New").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-new",
Click = "executeAction"
}).Add();
}).Add();
}).Add();
}).Add();
})
.ClientSideEvents(event => event.Create("createControl")))
<div id="newCon">
<table>
<tr>
<td><button id="btn1" class="e-bsnewbtnstyle">Blank
WorkBook</button></td>
</tr>
</table>
</div>
@section ScriptSection{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script>
function createControl(args) {
$("#btn1").ejButton({
size: "large",
height: 200,
width: 225,
contentType: "textandimage",
imagePosition: "imagetop",
prefixIcon: "e-icon e-blank e-infopageicon"
});
}
</script>
}
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style>
.e-blank {
background-image: url("../Content/ej/common-images/ribbon/blank.png");
}
.e-infopageicon {
background-repeat: no-repeat;
height: 150px;
width: 198px;
}
.e-newpageicon {
background-repeat: no-repeat;
height: 42px;
width: 42px;
}
</style>
}
```



Note: Height & Width of backstage can be set using Height and Width, if these are not set, Ribbon's Height & Width will be considered

You can add/remove/update backStage item to the ribbon control by using addBackStageItem, removeBackStageItem and updateBackStageItem methods. Also you can show/hide the backstage page

Copyright © 2001 - 2020 Syncfusion Inc.

in ribbon control by using showBackstage and [hideBackstage](https://help.syncfusion.com/api/js/ejribbon#methods:hidebackstage methods.

Tab

RibbonTabs is a collection of control TabGroup which enables you to organize related commands into single view. Tabs can be added to Ribbon using RibbonTabs property. Id & Text properties are used to set unique ID and header text to RibbonTabs.

The manipulation of given text tab in the ribbon control can be done by using addTab, removeTab, hideTab,

showTab methods and tabAdd, tabCreate, tabRemove, tabClick and tabSelect events.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties() {
OpenOnClick = false
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tabgroup => {
tabgroup.Text("Save").AlignType(RibbonAlignType.Rows).Content(ctn => {
ctn.ContentGroups(contentGroup => {
contentGroup.Id("print").Text("Print").ButtonSettings(new
ButtonProperties() {
ContentType = ContentType.TextAndImage,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-Ribbon e-print"
}).Add();
contentGroup.Id("save").Text("Save").ButtonSettings(new
ButtonProperties() {
ContentType = ContentType.TextAndImage,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-Ribbon e-save"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("50px").Height("70px")).Add();
}).Add();
}).Add();
tab.Id("sendRec").Text("Send/Receive").TabGroups(tabgroup => {
tabgroup.Text("Send/Receive").Type("custom").ContentID("sendReceive").Ali
gnType(RibbonAlignType.Columns).Content(ctn => {
}).Add();
}).Add();
})
)
<div id="sendReceive">
Send/Receive All Folders
</div>
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
@section StyleSection{
<link href="~/Content/ej/Ribbon-css/ej.icons.css" rel="stylesheet" />
}
```



Group

Group is a collection of logical content groups that are combined under related Tab. Each group can be defined using content groups or custom content.

Adding Tab Groups

Group items can be added to Tabs by specifying Text and corresponding Content to be displayed. The content of group can be specified as either with Content collection, ContentID or CustomContent. You can add tab group dynamically in the ribbon control with given tab index, tab group object and group index position by using addTabGroup method.

Adding Content

Add content to Group item which is based on Type of content specified. The available types are Button , SplitButton, ToggleButton,Gallery, and DropDownList.

Groups and defaults settings can be added with the Content. You can add group content dynamically in the ribbon control with given tab index, group index, content, content index and sub group index position by using addTabGroupContent.

Defaults

The tabs.groups.content.defaults.height, tabs.groups.content.defaults.width,

tabs.groups.content.defaults.type, tabs.groups.content.defaults.isBig property to the controls in the group can be specified commonly.

The Height & Width applicable to Button, SplitButton, DropDownList ,ToggleButton controls and IsBig applicable to only button controls (button, split , toggle)

Adding Content Groups

Controls such as button, split button, dropdown list, toggle button, gallery in the subgroup of the Ribbon Tab can be rendered. All of these can be customized using its corresponding settings property such as ButtonSettings, DropdownSettings, etc.

Copyright © 2001 - 2020 Syncfusion Inc.

Custom controls or items (such as table, div etc.) can be added when the Type set as Custom. Defaults control settings of group can be specified for an individual group instead of specifying them to Content Groups collection commonly.

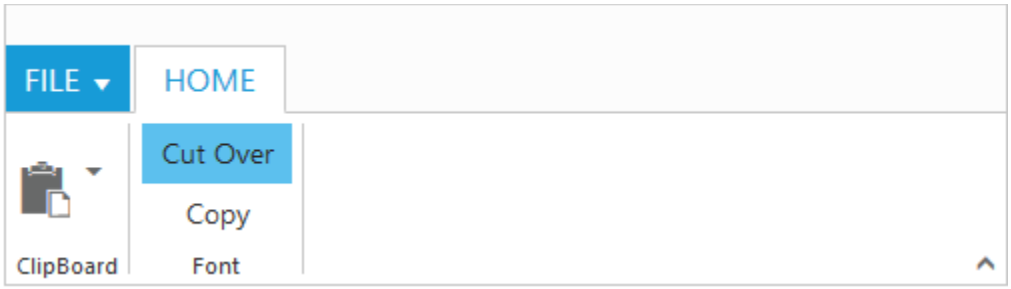
ToolTip and CustomTooltip can be specified for each group controls.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500px")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("paste").Text("Paste").ToolTip("Paste").IsBig(true).Type(
RibbonButtonType.SplitButton).SplitButtonSettings(new
SplitButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-ribbonpaste",
TargetID = "pasteSplit",
}).Add();
}).Add();
}).Add();
tabgroup.Text("Font").AlignType(RibbonAlignType.Columns).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("cut").ToggleButtonSettings(new ToggleButtonProperties()
{
DefaultText = "Cut",
ActiveText = "Cut Over"
}).Add();
contentGroup.Id("copy").ToggleButtonSettings(new ToggleButtonProperties()
{
DefaultText = "Copy",
ActiveText = "Copy Over"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("75px").Height("30px").Type(Ribbon
ButtonType.ToggleButton)).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<li><a>Open</a></li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li><a>Paste Special</a></li>
</ul>
```



Enable Separator

Separates the control from the next control in the group when group AlignType is Rows. Set “true” to EnableSeparator.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500px")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("new").Text("New").ToolTip("New").EnableSeparator(true).B
uttonSettings(new ButtonProperties()
{
Width = "100",
}).Add();
contentGroup.Id("font").Text("Font").ToolTip("Font").ButtonSettings(new
ButtonProperties()
{
Width = "150",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Height("70px")).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
```



Adding Custom Content

Set group Type as custom to add custom items such as div, table and custom controls. With type as custom, content can be added in two ways as specified below.

HTML contents can be directly added into the groups as string content using CustomContent. property

Custom template id can be specified to render those specific custom template using ContentID property

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("New").Type("custom").CustomContent("<button
id='customContent'>Using Custom Content</button>").Add();
tabgroup.Text("Data").Type("custom").ContentID("btn").Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
<button id='btn'>Using Content ID</button>
```



Group Expander

Set enableGroupExpander as true to show Group Expander for each group in Tab. These expanders can be customized using groupExpand event, such as to show popup dialog. To specify tooltip for the group expander of the group tabs.groups.groupExpanderSettings and

tabs.groups.groupExpanderSettings.toolTip can be used.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Type("custom").Enabl
eGroupExpander(true).ContentID("btn").Add();
})
.ClientSideEvents(event => event.GroupExpand("groupExpand"))
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
<button id='btn'>Home button</button>
<script>
function groupExpand(args) {
alert("Group expander click triggered");}
</script>
```



Copyright © 2001 - 2020 Syncfusion Inc.

Controls Support

Button, SplitButton, DropDownList, ToggleButton, Gallery and Custom controls can be added to each groups. You can set Type property in group to define the controls. Default Type is Button.

Built in Controls

The following table describes about the built in controls Type and their corresponding control settings.

Type	Control Settings	Example
Button	Button - ButtonSettings	ButtonSettings: { Width: 70, ContentType: ej.ContentType.ImageOnly, PrefixIcon: "e-icon e-ribbon e-new" }
SplitButton	SplitButton - SplitButtonSettings	SplitButtonSettings: { ContentType: ej.ContentType.ImageOnly, TargetID: "pasteSplit", ButtonMode: "dropdown", ArrowPosition: ej.ArrowPosition.Bottom }
ToggleButton	ToggleButton - ToggleButtonSettings	ToggleButtonSettings: { ContentType: ej.ContentType.ImageOnly, DefaultText: "Italic", ActiveText: "Italic" }
DropDownList	DropDownList - DropDownSettings	DropDownSettings: { DataSource: size, Text: "1pt", Width: 65 }

Note: 1. You can specify type either to group's collection or to each group.

2. For Type property you can assign either string value ("SplitButton") or enum value (RibbonButtonType.SplitButton).

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("100%")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon").MenuSettings(new
MenuProperties()
{
OpenOnClick = false
});
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentTab =>
{
contentTab.Id("new").Text("New").ToolTip("New").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
```

Copyright © 2001 - 2020 Syncfusion Inc.

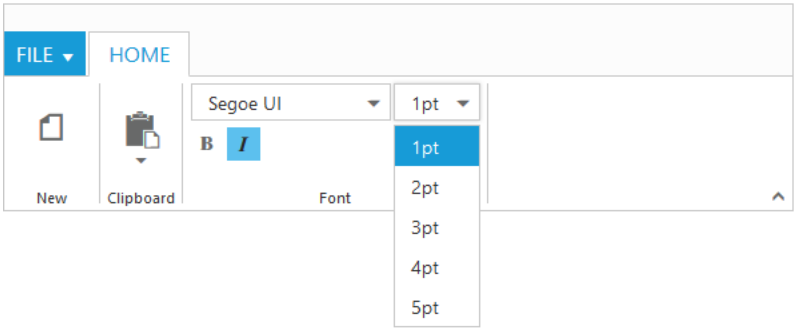
```
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-new",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("70px")).Add();
}).Add();
tabgroup.Text("Clipboard").AlignType(RibbonAlignType.Columns).Content(ctn
=>
{
ctn.ContentGroups(contentTab =>
{
contentTab.Id("paste").Text("Paste").ToolTip("Paste").SplitButtonSettings
(new SplitButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-ribbonpaste",
TargetID = "pasteSplit",
ButtonMode= ButtonMode.Dropdown,
ArrowPosition = ArrowPosition.Bottom,
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.SplitButton).Width("50px").Height("70px")).Add()
;
}).Add();
tabgroup.Text("Font").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentTab =>
{
contentTab.Id("fontFamily").ToolTip("Font").DropdownSettings(new
DropDownListProperties()
{
DataSource = (IEnumerable<FontFamily>)ViewBag.datasource,
Text = "Segoe UI",
Width = "150px"
}).Add();
contentTab.Id("fontsize").ToolTip("FontSize").DropdownSettings(new
DropDownListProperties()
{
DataSource = (IEnumerable<FontPoint>)ViewBag.datasource1,
Text = "1pt",
Width = "65px"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.DropDownList).Height("28px")).Add();
ctn.ContentGroups(contentTab =>
{
contentTab.Id("bold").ToolTip("Bold").Type(RibbonButtonType.ToggleButton)
.ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Bold",
ActiveText = "Bold",
DefaultPrefixIcon = "e-icon e-ribbon bold",
ActivePrefixIcon = "e-icon e-ribbon bold",
}).Add();
contentTab.Id("italic").ToolTip("Italic").Type(RibbonButtonType.ToggleBut
ton).ToggleButtonSettings(new ToggleButtonProperties()
```

```
{
ContentType = ContentType.ImageOnly,
DefaultText = "Italic",
ActiveText = "Italic",
DefaultPrefixIcon = "e-icon e-ribbon e-ribbonitalic",
ActivePrefixIcon = "e-icon e-ribbon e-ribbonitalic",
}).Add();
}).ContentDefaults(df => df.IsBig(false)).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li>
<a>New</a>
</li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li>
<a>Paste</a>
</li>
</ul>
```

C#

```
public partial class RibbonController: Controller
{
List<FontFamily> fontFamily1 = new List<FontFamily>();
List<FontPoint> fontPoint1 = new List<FontPoint>();
public ActionResult RibbonFeatures()
{
fontFamily1.Add(new FontFamily { text = "Segoe UI" });
fontFamily1.Add(new FontFamily { text = "Arial" });
fontFamily1.Add(new FontFamily { text = "Times New Roman" });
fontFamily1.Add(new FontFamily { text = "Tahoma" });
fontFamily1.Add(new FontFamily { text = "Helvetica" });
ViewBag.datasource = fontFamily1;
fontPoint1.Add(new FontPoint { text = "1pt" });
fontPoint1.Add(new FontPoint { text = "2pt" });
fontPoint1.Add(new FontPoint { text = "3pt" });
fontPoint1.Add(new FontPoint { text = "4pt" });
fontPoint1.Add(new FontPoint { text = "5pt" });
ViewBag.datasource1 = fontPoint1;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Custom

You can set Type as custom to render custom controls and Custom element id has to be specified as ContentID.You can change the element defined in the custom template to appropriate Syncfusion control in the event of Ribbon Create.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("600")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Font").Content(ctn =>
{
ctn.ContentGroups(contentTab =>
{
contentTab.Id("fontColor").ToolTip("Font
Color").ContentID("fontColor").Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Custom).Height("30px")).Add();
}).Add();
tabgroup.Text("Operators").Type("custom").ContentID("design").Add();
}).Add();
})
.ClientSideEvents(event => event.Create("createControl"))     )
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
<input id="fontColor" />
<table id="design" class="e-designtablestyle">
<tr>
<td><input type="checkbox" id="check1" /><label for="check1">Header
Row</label></td>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<td><input type="checkbox" id="Check2" checked="checked" /><label
for="Check2">First Column</label></td>
</tr>
<tr><td><input type="checkbox" id="check4" checked="checked" /><label
for="check4">Total Row</label></td>
<td><input type="checkbox" id="Check5" /><label for="Check5">Last
Column</label></td>
</tr>
</table>
<script>
function createControl(args) {
$("#fontColor").ejColorPicker({ value: "#FFFF00", modelType: "palette",
cssClass: "e-ribbon", toolIcon: "e-fontcoloricon" });
}
</script>
```



Contextual Tab And Tab Set

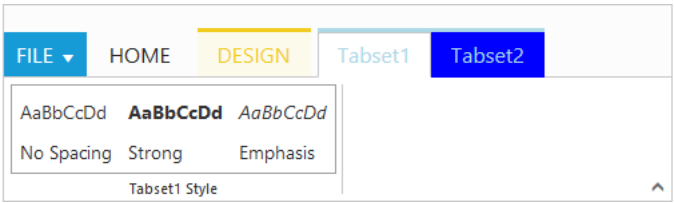
Contextual Tabs are collection of Tabs that extended styling and can be shown based on some criteria. Contextual Tabs can be added like RibbonTabs including TabGroup and TabContent section. You can set BackGroundColor and BorderColor to highlight them as Tab set. Contextual tabs can be added or set dynamically in ribbon control using addContextualTabs with it's object and index position.

ASPX-CS

```
@(Html.EJ().Ribbon("Ribbon")
.Width("500")
.ApplicationTab(app => {
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon").MenuSettings(new
MenuProperties()
{
OpenOnClick = false
});
})
.RibbonTabs(tab => {
tab.Id("home").Text("HOME").TabGroups(tabGroup => {
tabGroup.Text("CustomControls").Type("custom").ContentID("Contents").Add(
);
}).Add();
})
.ContextualTabs(contextualTab => {
contextualTab.BackgroundColor("#FCFBEB").BorderColor("#F2CC1C").RibbonTab
s(ribbonTab => {
ribbonTab.Id("Design").Text("DESIGN").TabGroups(contextTabGroup => {
contextTabGroup.Text("Table Style
Options").Type("custom").ContentID("contextualTab").Add();
}).Add();
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
contextualTab.BackgroundColor("blue").BorderColor("lightblue").RibbonTabs
(ribbonTab => {
ribbonTab.Id("tabset1").Text("Tabset1").TabGroups(contextTabGroup => {
contextTabGroup.Text("Tabset1
Style").Type("custom").ContentID("headings").Add();
}).Add();
ribbonTab.Id("tabset2").Text("Tabset2").TabGroups(contextTabGroup => {
contextTabGroup.Text("Tabset2
Styles").Type("custom").ContentID("contextualTabset2").Add();
}).Add();
});
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
<div id="headings" class="e-headings">
<div>
<p>First Heading</p>
<p>No Spacing</p>
</div>
<div>
<p class="e-strong">Second Heading</p>
<p>Strong</p>
</div>
<div>
<p class="e-emphasis">Third Heading</p>
<p>Emphasis</p>
</div>
</div>
<div id="Contents">Custom Control</div>
<div id="contextualTab">
<button id="contextualBtn">Contextual Tab</button>
</div>
<div id="contextualTabset2">
<button id="contextualTabsetBtn2">Contextual Tabset2</button>
</div>
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
}
```



Copyright © 2001 - 2020 Syncfusion Inc.

Gallery

Galleries are used to display items that can be arranged in a grid-type layout. Items in the gallery can be customized as Button/Menu to display images, text, or both images and text. You can set Type of group as Gallery.

Gallery Items

Gallery items are collection of the items to be included in the main gallery. You can set Text and ToolTip to gallery item which can also be customized with ButtonSettings.

Number of Columns to display in gallery for each row should be specified and the Number of columns in Expanded State (ExpandedColumns) can be customized, if not set, Columns count will be set as default.

Note: The ItemHeight and ItemWidth for gallery item can be set, if not set default values will be used.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Gallery").Type("gallery").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("Gallery").Type(RibbonButtonType.Gallery).ItemWidth("73")
.ItemHeight("54").Columns(2).ExpandedColumns(3).GalleryItems(gallery =>
{
gallery.Text("GalleryContent1").ToolTip("GalleryContent1").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent1 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("GalleryContent2").ToolTip("GalleryContent2").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent2 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("GalleryContent3").ToolTip("GalleryContent3").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent3 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
gallery.Text("GalleryContent4").ToolTip("GalleryContent4").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent4 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
}).Add();
}).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>Open</a></li>
</ul>
</li>
</ul>
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style type="text/css">
.e-gallerycontent1 {
background-position: 0 -105px;
}
.e-gallerycontent2 {
background-position: -69px -105px;
}
.e-gallerycontent3 {
background-position: -136px -105px;
}
.e-gallerycontent4 {
background-position: 0 -53px;
}
.e-gbtnposition {
margin-top: 5px;
}
.e-gbtnimg {
background-image: url("../Content/ej/common-
images/ribbon/homegallery.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
</style>
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Ribbon Gallery.



Gallery at Expanded State

Custom Gallery Items

Custom gallery items are the additional items to be displayed at gallery expanded state. You can set `CustomItemType` as `Button` or `Menu`, Default is `Button`.

You can also set `Text` and `ToolTip` to custom gallery item which can also be customized with ButtonSettings/MenuSettings based on the CustomItemType specified.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Gallery").Type("gallery").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("Gallery").Type(RibbonButtonType.Gallery).ItemWidth("73")
.ItemHeight("54").Columns(2).ExpandedColumns(3).GalleryItems(gallery =>
{
gallery.Text("GalleryContent1").ToolTip("GalleryContent1").ButtonSettings
(new ButtonProperties(){
ContentType=ContentType.ImageOnly,
PrefixIcon="e-icon e-gallerycontent1 e-gbtnimg",
```

```
CssClass="e-gbtnposition"
}).Add();
gallery.Text("GalleryContent2").ToolTip("GalleryContent2").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent2 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("GalleryContent3").ToolTip("GalleryContent3").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent3 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("GalleryContent4").ToolTip("GalleryContent4").ButtonSettings
(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent4 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
}).CustomGalleryItems(customGallery =>
{
customGallery.CustomItemType(CustomItemType.Menu).MenuId("customMenu").Me
nuSettings(new MenuProperties()
{
OpenOnClick = false
}).Add();
customGallery.Text("Clear Formatting").ToolTip("Clear
Formatting").CustomItemType(CustomItemType.Button).ButtonSettings(new
ButtonProperties()
{
CssClass = "e-extrabtnstyle"
}).Add();
}).Add();
}).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
</li>
</ul>
<ul id="customMenu">
<li>
<a>New Quick Step</a>
<ul>
<li>
<a>Flag & Move</a>
</li>
</ul>
</li>
</ul>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style type="text/css">
.e-gallerycontent1{
background-position: 0 -105px;
}
.e-gallerycontent2{
background-position: -69px -105px;
}
.e-gallerycontent3{
background-position: -136px -105px;
}
.e-gallerycontent4{
background-position: 0 -53px;
}
.e-gbtnposition{
margin-top:5px;
}
.e-gbtnimg{
background-image: url("../Content/ej/common-
images/ribbon/homegallery.png");
background-repeat:no-repeat;
height:64px;
width:64px;
}
.e-extracontent .e-extrabtnstyle{
padding-left: 28px;
text-align: left;
}
.e-pastetip {
background-image: url("../Content/ej/common-images/ribbon/paste.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
</style>
```



Copyright © 2001 - 2020 Syncfusion Inc.

Resize

Ribbon control dynamically resizes to display possible number of controls in the optimal layout as the application window size changes.

As the window is narrowed, controls in the Ribbon will be combined as group button with dropdown arrow, in which controls can be expanded with dropdown arrow.

Tablet Layout

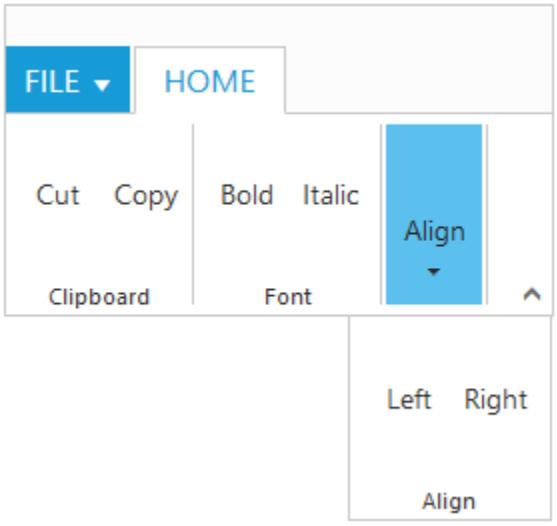
Set IsResponsive as true to enable resizing in Ribbon.If client width is above 420px or control content exceeds the page then, the ribbon will render in Tablet mode.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon").IsResponsive(true)
.Width("40%")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("cut").Text("Cut").Add();
contentGroup.Id("copy").Text("Copy").Add();
}).ContentDefaults(df => df.Width("40px").Height("70px")).Add();
}).Add();
tabgroup.Text("Font").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("bold").Text("Bold").Add();
contentGroup.Id("italic").Text("Italic").Add();
}).ContentDefaults(df => df.Width("40px").Height("70px")).Add();
}).Add();
tabgroup.Text("Align").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("left").Text("Left").Add();
contentGroup.Id("right").Text("Right").Add();
}).ContentDefaults(df => df.Width("40px").Height("70px")).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li><a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</li>
</ul>
```



Mobile Layout

If client width is less than 420px, the ribbon will render in mobile mode. In which, you can see that ribbon user interface is customized and redesigned for best view in small screens.

The customized features includes responsive tab & group rendering, backstage, gallery and button controls.

Responsive Tab and group

Set IsResponsive as true to enable responsive mode in Ribbon.

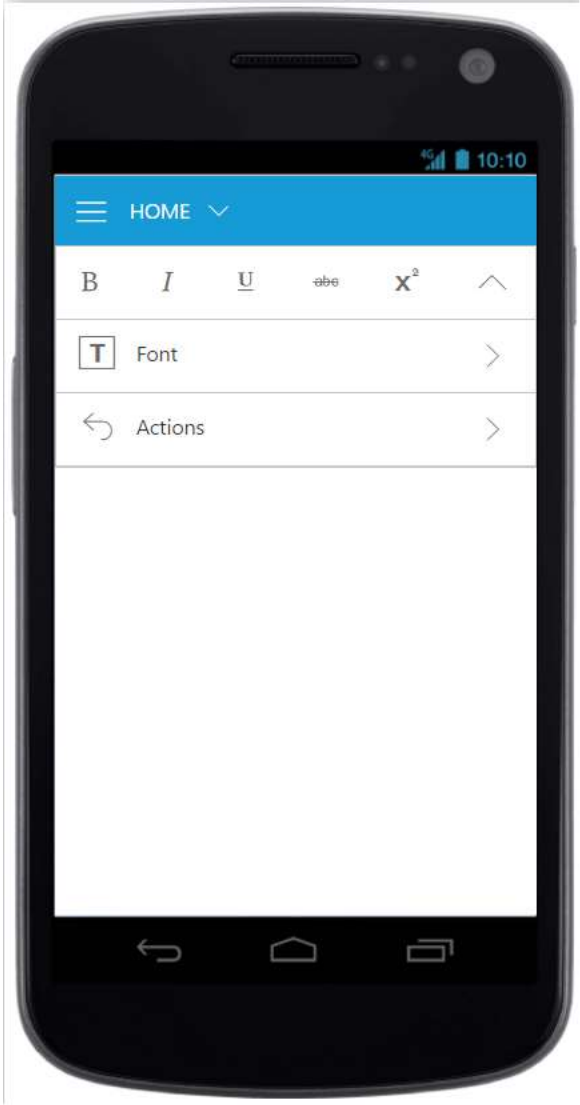
HTML

```
@(Html.EJ().Ribbon("defaultRibbon")
.IsResponsive(true)
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Font").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("bold").IsMobileOnly(true).Type(RibbonButtonType.ToggleBu
tton).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText="Bold",
ActiveText="Bold",
DefaultPrefixIcon = "e-icon e-ribbon e-resbold",
ActivePrefixIcon = "e-icon e-ribbon e-resbold",
}).Add();
contentGroup.Id("italic").IsMobileOnly(true).Type(RibbonButtonType.Toggle
Button).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DefaultText = "Italic",
ActiveText = "Italic",
DefaultPrefixIcon = "e-icon e-ribbon e-resitalic",
ActivePrefixIcon = "e-icon e-ribbon e-resitalic",
}).Add();
contentGroup.Id("underline").IsMobileOnly(true).Type(RibbonButtonType.Tog
gleButton).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Underline",
ActiveText = "Underline",
DefaultPrefixIcon = "e-icon e-ribbon e-resunderline",
ActivePrefixIcon = "e-icon e-ribbon e-resunderline",
}).Add();
contentGroup.Id("strikethrough").IsMobileOnly(true).Type(RibbonButtonType
.ToggleButton).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Strikethrough",
ActiveText = "Strikethrough",
DefaultPrefixIcon = "e-icon e-ribbon strikethrough",
ActivePrefixIcon = "e-icon e-ribbon strikethrough",
}).Add();
contentGroup.Id("superscript").IsMobileOnly(true).Text("Superscript").But
tonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-superscripticon",
}).Add();
}).ContentDefaults(df => df.IsBig(false)).Add();
}).Add();
}).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Note: To make the Ribbon control to react as responsive in mobile devices, it is necessary to refer the additional ej.responsive.css file in the application.

Mobile Toolbar Customization

Set IsMobileOnly as true to group control to show the controls

in the Mobile Toolbar of the ribbon. For each tab , first row of mobile ribbon will pick and display the controls which is set as IsMobileOnly with look adapt to mobile mode.If IsMobileOnly property is not defined to any of the control within tab, then by default fist group content will be displayed in first row toolbar.

To adapt to proper display of controls , following layout will be customized with constants display.

- First ribbon toolbar and Button controls with min-height.
- Drop down control will adapt to full screen width.
- All button controls icon will be displayed commonly as Top position.

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Ribbon("defaultRibbon")
.IsResponsive(true)
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Font").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("bold").IsMobileOnly(true).Type(RibbonButtonType.ToggleBu
tton).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText="Bold",
ActiveText="Bold",
DefaultPrefixIcon = "e-icon e-ribbon e-resbold",
ActivePrefixIcon = "e-icon e-ribbon e-resbold",
}).Add();
contentGroup.Id("italic").IsMobileOnly(true).Type(RibbonButtonType.Toggle
Button).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Italic",
ActiveText = "Italic",
DefaultPrefixIcon = "e-icon e-ribbon e-resitalic",
ActivePrefixIcon = "e-icon e-ribbon e-resitalic",
}).Add();
contentGroup.Id("underline").IsMobileOnly(true).Type(RibbonButtonType.Tog
gleButton).ToggleButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Underline",
ActiveText = "Underline",
DefaultPrefixIcon = "e-icon e-ribbon e-resunderline",
ActivePrefixIcon = "e-icon e-ribbon e-resunderline",
}).Add();
contentGroup.Id("strikethrough").Type(RibbonButtonType.ToggleButton).Togg
leButtonSettings(new ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Strikethrough",
ActiveText = "Strikethrough",
DefaultPrefixIcon = "e-icon e-ribbon strikethrough",
ActivePrefixIcon = "e-icon e-ribbon strikethrough",
}).Add();
contentGroup.Id("superscript").Text("Superscript").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-superscripticon",
}).Add();
}).ContentDefaults(df => df.IsBig(false)).Add();
}).Add();
}).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

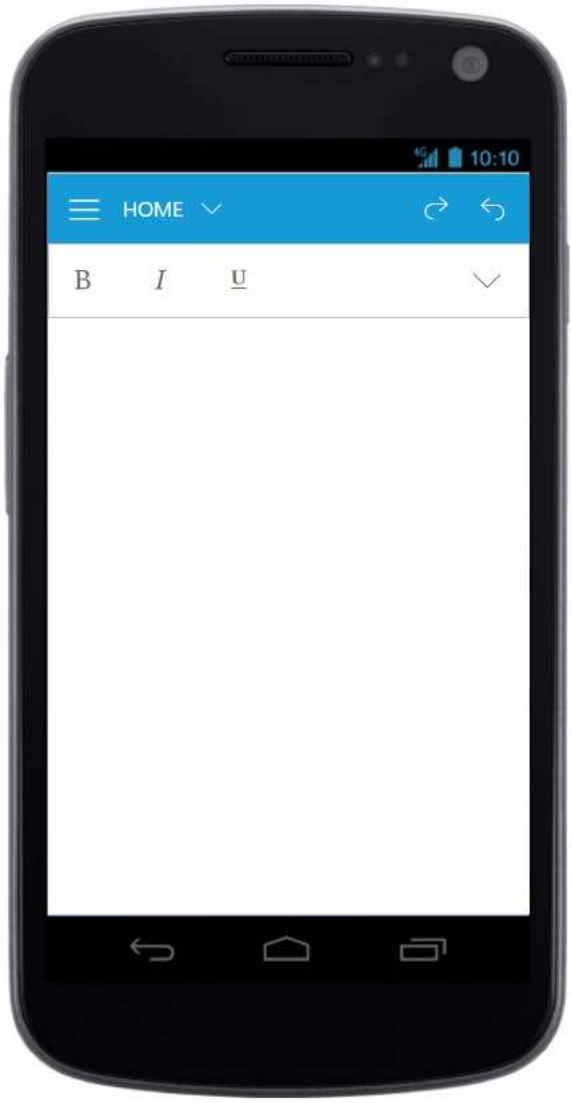


Ribbon Responsive with MobileToolbar

Customized Features

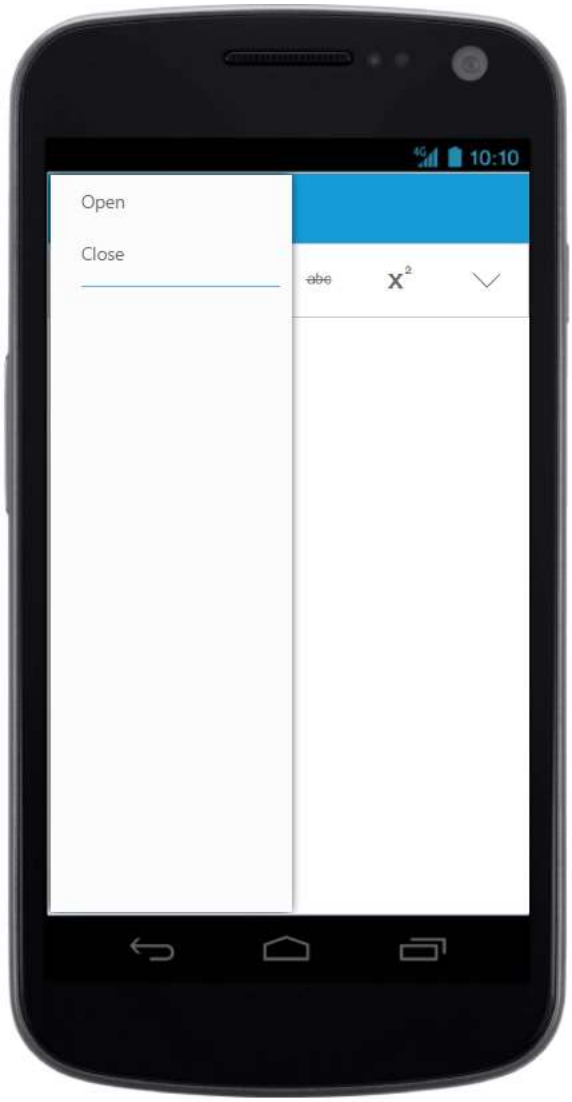
The customized layout for Quick Access Toolbar, backstage, gallery can be seen following screen shots.

Copyright © 2001 - 2020 Syncfusion Inc.

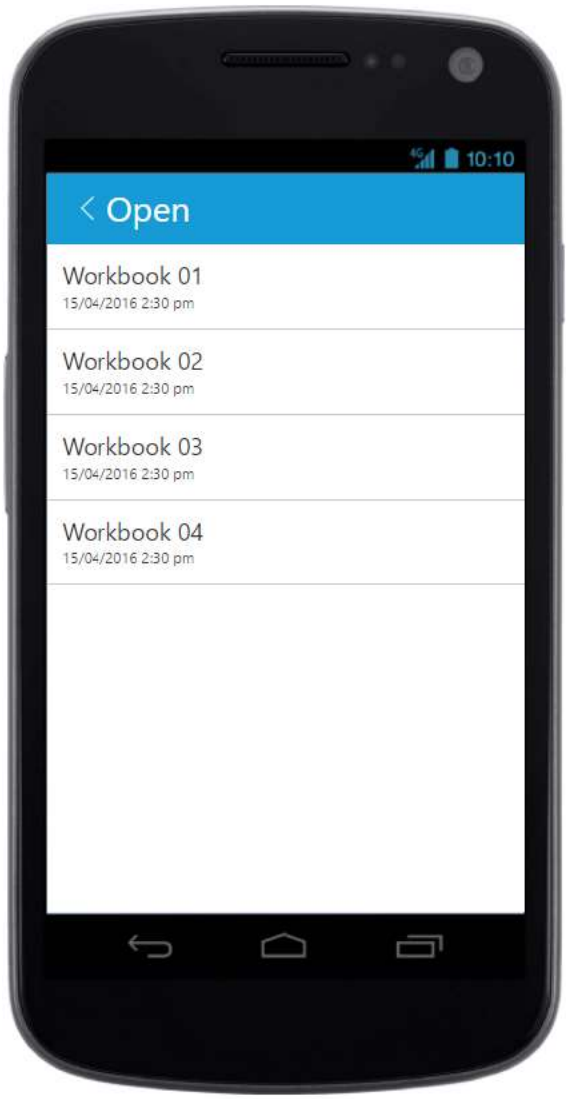


Ribbon Responsive with Quick Access Toolbar

Copyright © 2001 - 2020 Syncfusion Inc.

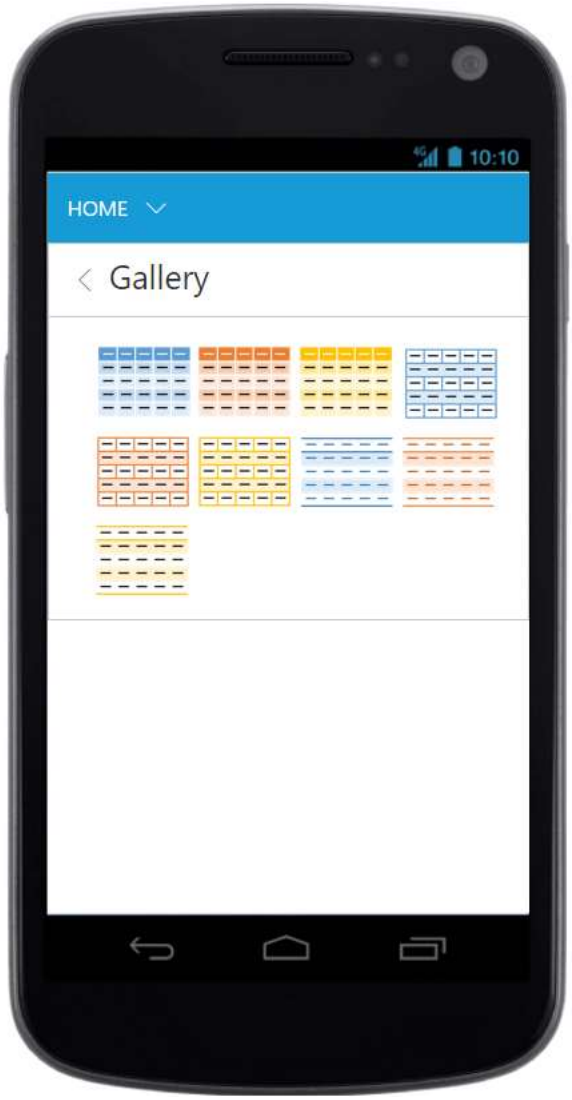


Copyright © 2001 - 2020 Syncfusion Inc.



Ribbon Responsive with backstage

Copyright © 2001 - 2020 Syncfusion Inc.



Ribbon Responsive with gallery

Group Button Customization

Based on window size, detailed group is shrined into single button and you can expand group items with group button click.

For each group shirked for resizing, Custom Class will be added based on group text.For example, `e-Action` whereas `Action` is group text. Using this custom class, group button can be customized such as to set icons etc.

ASPX-CS

```
@section ControlsSection{
<div id="resizediv" style="width:100%">
@(Html.EJ().Ribbon("defaultRibbon").AllowResizing(true)
.Width("36%")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").AlignType(RibbonAlignType.Columns).EnableGroup
Expander(true).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("paste").Text("Paste").ToolTip("Paste").ButtonSettings(ne
w ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-ribbonpaste",
}).Add();
}).ContentDefaults(df =>
df.IsBig(true).Width("50px").Height("70px")).Add();
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("cut").Text("Cut").ToolTip("Cut").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.TextAndImage,
PrefixIcon = "e-icon e-ribbon e-ribboncut",
}).Add();
contentGroup.Id("copy").Text("Copy").ToolTip("Copy").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.TextAndImage,
PrefixIcon = "e-icon e-ribbon e-ribboncopy",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("40px").IsBig(false
)).Add();
}).Add();
tabgroup.Text("Font").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("fontFamily").ToolTip("Font").DropdownSettings(new
DropDownListProperties()
{
DataSource = (IEnumerable<FontFamily>)ViewBag.datasource,
Text = "Segoe UI",
Width = "150px"
}).Add();
contentGroup.Id("fontsize").ToolTip("FontSize").DropdownSettings(new
DropDownListProperties()
{
DataSource = (IEnumerable<FontPoint>)ViewBag.datasource1,
Text = "1pt",
Width = "65px"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.DropDownList).Height("28px").IsBig(false)).Add()
;
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("new").Text("New").ToolTip("New").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-new",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("40px")).Add();
}).Add();
tabgroup.Text("Actions").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("undo").Text("Undo").ToolTip("Undo").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.TextAndImage,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-undo",
}).Add();
contentGroup.Id("redo").Text("Redo").ToolTip("Redo").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.TextAndImage,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-redo",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("40px").Height("70px")).Add();
}).Add();
}).Add();
tab.Id("layout").Text("LAYOUT").TabGroups(tabgroup =>
{
tabgroup.Text("Print Layout").AlignType(RibbonAlignType.Rows).Content(ctn
=>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("printLayout").Text("Print Layout").ToolTip("Print
Layout").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.TextAndImage,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-printlayout",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("80px").Height("70px")).Add();
}).Add();
}).Add();
})
)
</div>
```

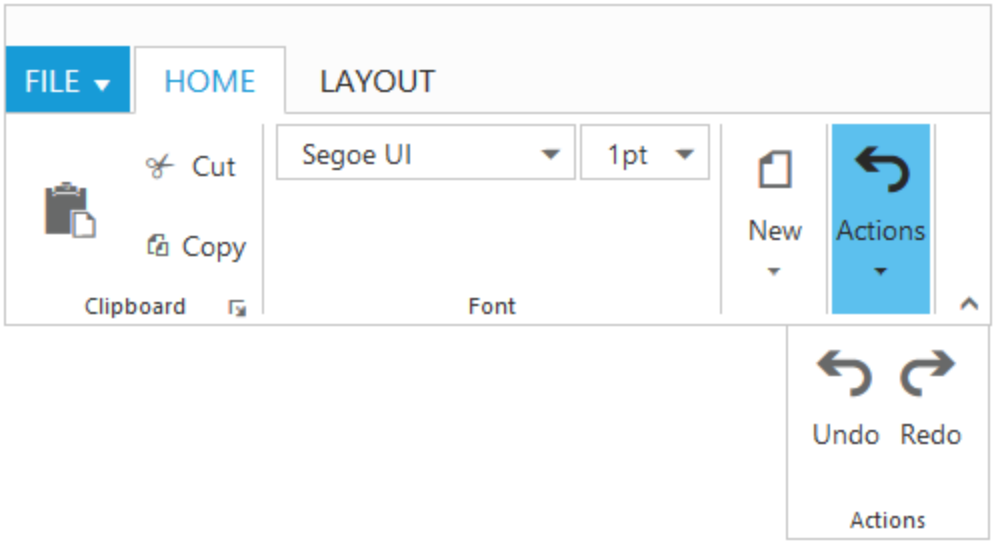
Copyright © 2001 - 2020 Syncfusion Inc.

```
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
}
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style type="text/css">
.e-ribbon .e-New:before, .e-ribbon .e-Actions:before {
font-family: "ej-ribbonfont";
font-size: 28px;
line-height: 35px;
}
.e-ribbon .e-New:before {
content: "\e167";
text-indent: -1.5px;
}
.e-ribbon .e-Actions:before {
content: "\e184";
text-indent: 7px;
}
</style>
}
```

C#

```
public partial class RibbonController: Controller
{
List<FontFamily> fontFamily1 = new List<FontFamily>();
List<FontPoint> fontPoint1 = new List<FontPoint>();
public ActionResult RibbonFeatures()
{
fontFamily1.Add(new FontFamily { text = "Segoe UI" });
fontFamily1.Add(new FontFamily { text = "Arial" });
fontFamily1.Add(new FontFamily { text = "Times New Roman" });
fontFamily1.Add(new FontFamily { text = "Tahoma" });
fontFamily1.Add(new FontFamily { text = "Helvetica" });
ViewBag.datasource = fontFamily1;
fontPoint1.Add(new FontPoint { text = "1pt" });
fontPoint1.Add(new FontPoint { text = "2pt" });
fontPoint1.Add(new FontPoint { text = "3pt" });
fontPoint1.Add(new FontPoint { text = "4pt" });
fontPoint1.Add(new FontPoint { text = "5pt" });
ViewBag.datasource1 = fontPoint1;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Screen Tips

ScreenTip/Tooltip is used to reduce the controls related Help that are needed to the end user to do control related actions.

HTML Tooltip

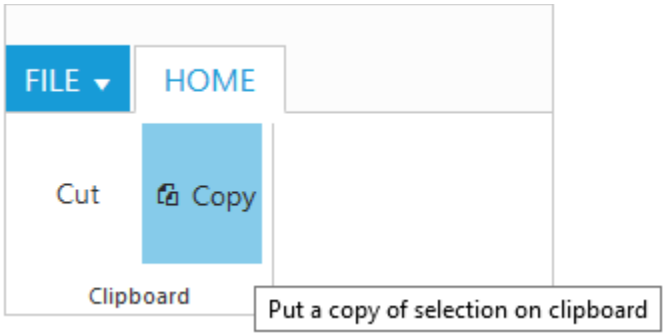
Standard html tooltip can be set using ToolTip property of each group item.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("20%")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("cut").Text("Cut").ToolTip("Remove the selection and put
it on clipboard").Add();
contentGroup.Id("copy").Text("Copy").ToolTip("Put a copy of selection on
clipboard").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.TextAndImage,
PrefixIcon = "e-icon e-ribbon e-ribboncopy",
}).Add();
}).ContentDefaults(df => df.Width("70px").Height("60px")).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
```



Custom Tooltip

Custom Tooltip is used to set detailed help to the user about the controls. You can set`Title`, `Content` and `PrefixIcon` class to customize the tooltip with icons.

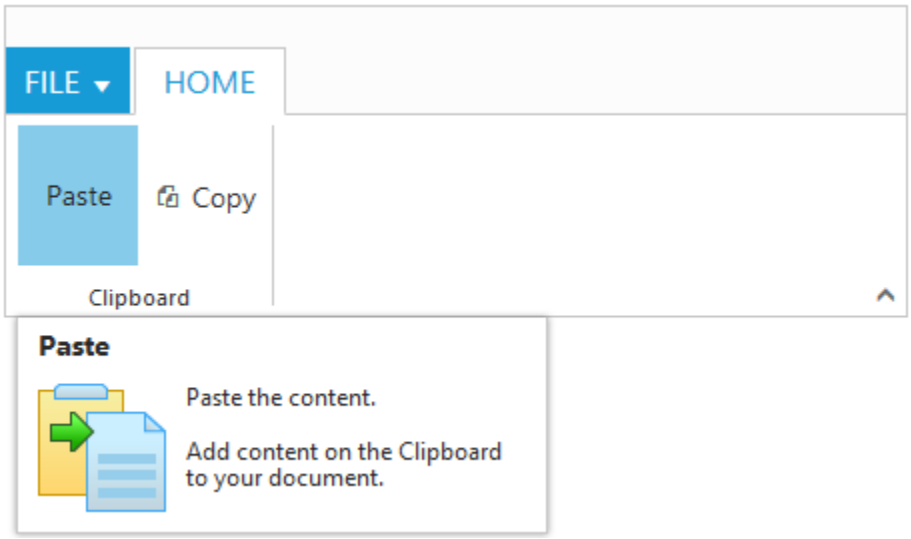
For Groups

`CustomTooltip` for each group controls can be specified. Such as to the controls Button, SplitButton, DropDownList etc.

ASPX-CS

```
@section ControlsSection {
@(Html.EJ().Ribbon("defaultRibbon")
.Width("450")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("paste").Text("Paste").CustomToolTip(new CustomToolTip()
{
Title = "Paste",
Content = "<h6>Paste the content.<br/><br/>Add content on the Clipboard
to your document.</h6>",
PrefixIcon = "e-icon e-pastetip",
}).Add();
contentGroup.Id("copy").Text("Copy").CustomToolTip(new CustomToolTip()
{
Title = "Copy",
Content = "<h6>Copy the content.</h6>",
```

```
}).ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.TextAndImage,
PrefixIcon = "e-icon e-ribbon e-ribboncopy",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("70px").Height("60px")).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
</ul>
</li>
</ul>
}
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style>
.e-pastetip {
background-image: url("../Content/ej/common-images/ribbon/paste.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
</style>
}
```



For Gallery

CustomTooltip for each Gallery and CustomGalleryItems button control can be specified.

Note: Custom gallery item Menu is not supported to Custom tooltip.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@section ControlsSection {
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Gallery").Type("Gallery").Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("Gallery").Type(RibbonButtonType.Gallery).ItemWidth("73")
.ItemHeight("54").Columns(2).ExpandedColumns(3).GalleryItems(gallery =>
{
gallery.Text("Style 1").CustomToolTip(new CustomToolTip()
{
Title = "Style 1",
Content = "<I>Style 1 to customize the table</I>",
}).ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent1 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("Style 2").CustomToolTip(new CustomToolTip()
{
Title = "Style 2",
Content = "<I>Style 2 to customize the table</I>",
}).ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent2 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("Style 3").CustomToolTip(new CustomToolTip()
{
Title = "Style 3",
Content = "<I>Style 3 to customize the table</I>",
}).ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-gallerycontent3 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
gallery.Text("Style 4").CustomToolTip(new CustomToolTip()
{
Title = "Style 4",
Content = "<I>Style 4 to customize the table</I>",
}).ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
PrefixIcon = "e-icon e-gallerycontent4 e-gbtnimg",
CssClass = "e-gbtnposition"
}).Add();
}).CustomGalleryItems(galleryItems =>
{
galleryItems.Text("Clear Formatting").ToolTip("Clear
Formatting").CustomItemType(CustomItemType.Button).CustomToolTip(new
CustomToolTip()
{
Title="Clear Format",
Content = "<I>To clear formatting</I>"
}).ButtonSettings(new ButtonProperties()
{
CssClass="e-extrabtnstyle"
}).Add();
galleryItems.CustomItemType(CustomItemType.Menu).MenuId("extraMenu").Menu
Settings(new MenuProperties()
{
OpenOnClick = false
}).Add();
}).Add();
}).Add();
}).Add();
}).Add();
})
)
<ul id="ribbon">
<li>
<a>FILE</a>
</li>
</ul>
<ul id="extraMenu">
<li>
<a>New Quick Step</a>
<ul>
<li><a>Flag and Move</a></li>
</ul>
</li>
</ul>
}
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style type="text/css">
.e-gallerycontent1 {
background-position: 0 -105px;
}
.e-gallerycontent2 {
background-position: -69px -105px;
}
.e-gallerycontent3 {
background-position: -136px -105px;
}
.e-gallerycontent4 {
background-position: 0 -53px;
}
.e-gbtnposition {
margin-top: 5px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.e-gbtnimg {
background-image: url("../Content/ej/common-
images/ribbon/homegallery.png");
background-repeat: no-repeat;
height: 64px;
width: 64px;
}
.e-extracontent .e-extrabtnstyle {
padding-left: 28px;
text-align: left;
}
</style>
}
```



Quick Access Toolbar

Quick Access Toolbar provides the shortcuts to the most commonly used commands by placing the controls at the Quick Access Toolbar section. It can be placed at the top or bottom of the Ribbon.

Set ShowQAT as true to enable Quick Access Toolbar in Ribbon. It supports the Button, Split Button, Toggle Button controls. The QuickAccessMode is used to change the controls state in Quick Access Toolbar through options as Toolbar,Menu and none. Default value is none and QAT toolbar is created with specified controls added in Toolbar.

The ToolBar option used to set controls visibility in Quick Access Toolbar.The Menu option shows the controls in Quick Access Menu and does not show controls in Quick Access Toolbar.

Once the controls are visible in Toolbar , then controls state will be set as ticked in Quick Access Menu and vice versa.

The client side event for Quick Access Toolbar menu click is QatMenuItemClick.

More Commands command provides with Quick Access Menu. This can be customized using QatMenuItemClick event, such as to show popup dialog.

ASPX-CS

```
@section ControlsSection {
@(Html.EJ().Ribbon("defaultRibbon")
```

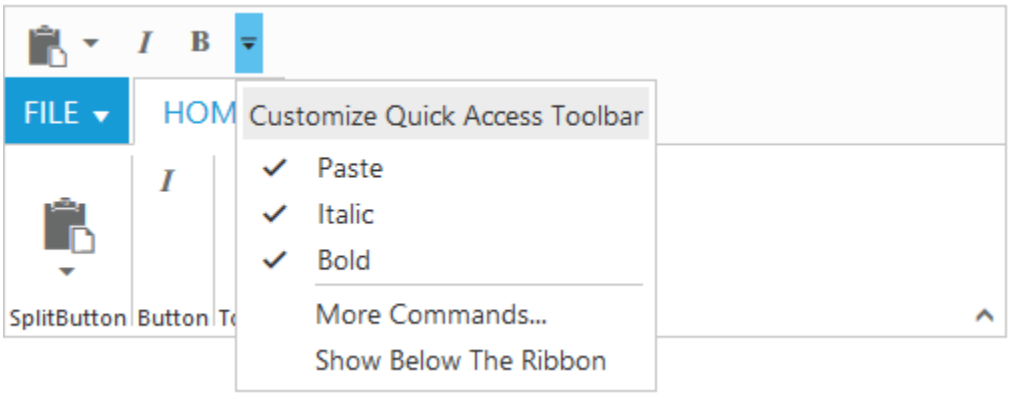
Copyright © 2001 - 2020 Syncfusion Inc.

```
.Width("500")
.ShowQAT(true)
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("ribbon");
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("SplitButton").AlignType(RibbonAlignType.Columns).Content(c
tn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("paste").Text("Paste").ToolTip("Paste").Type(RibbonButton
Type.SplitButton).QuickAccessMode(QuickAccessMode.ToolBar).SplitButtonSet
tings(new SplitButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-ribbonpaste",
TargetID = "pasteSplit",
ArrowPosition = ArrowPosition.Bottom,
ButtonMode = ButtonMode.Dropdown,
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.SplitButton).Width("50px").Height("70px")).Add()
;
}).Add();
tabgroup.Text("Button").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("italic").ToolTip("Italic").QuickAccessMode(QuickAccessMo
de.ToolBar).Type(RibbonButtonType.ToggleButton).ToggleButtonSettings(new
ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Italic",
ActiveText = "Italic",
DefaultPrefixIcon = "e-icon e-ribbon e-ribbonitalic",
ActivePrefixIcon = "e-icon e-ribbon e-ribbonitalic",
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("40px").Height("70px")).Add();
}).Add();
tabgroup.Text("Toggle").AlignType(RibbonAlignType.Columns).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("bold").ToolTip("Bold").QuickAccessMode(QuickAccessMode.T
oolBar).Type(RibbonButtonType.ToggleButton).ToggleButtonSettings(new
ToggleButtonProperties()
{
ContentType = ContentType.ImageOnly,
DefaultText = "Bold",
ActiveText = "Bold",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DefaultPrefixIcon = "e-icon e-ribbon bold",
ActivePrefixIcon = "e-icon e-ribbon bold",
}).Add();
}).ContentDefaults(df => df.IsBig(false)).Add();
}).Add();
}).Add();
}).ClientSideEvents(event =>
event.Create("createControl").QatMenuItemClick("qatMenuItemClick"))
)
<ul id="ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
</ul>
</li>
</ul>
<ul id="pasteSplit">
<li><a>Paste</a></li>
</ul>
}
@section StyleSection{
<link href="~/Content/ej/ribbon-css/ej.icons.css" rel="stylesheet" />
<style>
.e-ribbon .e-rbnquickaccessbar .e-ribbonpaste:before {
font-size: 27px;
left: -5px;
top: -6px;
}
.e-ribbon .e-ribbonpaste:before {
font-family: 'ej-ribbonfont';
content: "\e169";
font-size: 36px;
position: relative;
left: -9px;
top: -4px;
}
.e-ribbon .e-ribbonitalic:before ,.e-ribbon .bold:before{
font-family: 'ej-ribbonfont';
font-size: 16px;
left: -1px;
position: relative;
top: -1px;
}
.e-ribbon .e-ribbonitalic:before ,.e-ribbon .bold:before{
content: "\e163";
}
.e-ribbon .bold:before {
content: "\e15a";
}
.e-ribbon .e-rbnquickaccessbar .e-undo::before {
font-size: 18px;
line-height: 12px;
text-indent: -3px;
}
</style>
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Appearance and Styling

CssClass

When you want to display the **Ribbon** widget in a different style based on the appearance of your application, you can use this **cssClass** property to apply custom theme for the **Ribbon**. Specify a class name as the value for **cssClass** property. The specified class is added to the root element of the **Ribbon** widget. Now, you can easily override the styles of the **Ribbon** widget by accessing the styles from the root level (using the cssClass specified).

In an view page, define the Ribbon control and configure the Ribbon with custom theme by using the CssClass property.

ASPX-CS

```
@(Html.EJ().Ribbon("defaultRibbon")
.Width("500")
.CssClass("custom")
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Menu).MenuItemID("Ribbon").MenuSettings(new
MenuProperties()
{
OpenOnClick = false
});
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("new").Text("New").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-Ribbon e-new",
Click = "executeAction"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("70px")).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
}).Add();
}))
<ul id="Ribbon">
<li>
<a>FILE</a>
<ul>
<li><a>New</a></li>
<li><a>Open</a></li>
<li><a>Save</a></li>
<li><a>Save As</a></li>
<li><a>Print</a></li>
</ul>
</li>
</ul>
```

CSS

```
.custom.e-js .e-header {
background: #179bd7;
}
.custom.e-js .e-content {
background: #ddd;
}
.custom .e-rbn-button.e-btn.e-select {
background: #f5f5f5;
color: #333;
}
```

Execute the above code example to render the following output.



Themes

Ribbon control's style and appearance are controlled based on CSS classes and it has support of 12 default themes. You can check List of themes available for JavaScript's control which is mentioned in this page.

To modify Ribbon appearance, you need to override default CSS of Ribbon. Please find the list of CSS classes and its corresponding section in Ribbon. Also you have an option to create your own custom theme for all ASP.NET MVC controls using our Theme Studio.

To apply styles refer to two files namely

ej.widgets.core.min.css ej.theme.min.css

Copyright © 2001 - 2020 Syncfusion Inc.

When the file ej.widgets.all.min.css/ej.web.all.min.css is referred, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project as ej.widgets.all.min.css is the combination of these two files.

Customize Styles

Override default styles of Ribbon control by using its class name to customize it.

List of custom classes that are need to override styles. The following are specific class names you can use to modify styles and appearance in Ribbon.

e-js - used to represent element ej-widget control, that is present at the container element of Ribbon control. e-header - This class is applied to the header element of the Ribbon. *e-active-content -This class is applied to the active content of the Ribbon.* e-expandcollapse - This class is applied to the expand/collapse button in the Ribbon.

Application Tab

- e-apptab - This class is applied to the application tab in the Ribbon.

Tabs & Groups

e-tab -This class is applied to the Tabs in the Ribbon. e-groupdiv -This class is applied to the groups in the Ribbon. * e-active -This class is applied to the active or selected tab in the Ribbon.

Contextual Tabs

* e-contextliset, e-contextual, e-contextualtabset -This class is applied to the contextual tab and contextual tab set in the Ribbon.

Custom Tooltip

* e-tooltipdiv, e-tooltiptitle,e-tooltipdesc, e-tooltipimg – Applies to custom tooltip and its icons settings.

Gallery

e-gallexpandcontent,e-gallerycontent - Applies to gallery items and custom items for collapse and expanded state. e-gallerymovediv,e-gallerybtn,e-moveupdiv,e-movedowndiv,e-expgallerydiv - Applies to gallery item button and navigation buttons of gallery. * e-extracontent,e-galleryextrabtn,e-gallerymenu - Applies to gallery custom items menu and button customization.

Resizing

* e-resizebtn, e-resizediv – Applies to resizing and its group button.

Back Stage

e-ribbonbackstagepage,e-ribbonbackstagetop,e-backstagetopicon – Applies to back stage top header. e-ribbonbackstagebody ,e-backstageheader,e-backstagecontent,e-backstageli,e-backstageseparator, e-backstageactive – Applies Backstage page side header and its contents.

Quick Access Toolbar

- e-rbnwithqat- This class applied to the Quick access toolbar in the Ribbon.
- e-rbnabove - This class applied to the top Quick access toolbar.
- e-qatooldiv- This class applied to individual controls in the Quick access toolbar.
- e-rbnbelow- This class applied to the bottom Quick access toolbar

Copyright © 2001 - 2020 Syncfusion Inc.

AngularJS-Support

It is possible to add object, array collection in the Ribbon control with AngularJS. The one way binding support is provided for Ribbon control.

ej-ribbon is the control tag where, ej is tag prefix and Ribbon is the control name.

Tabs and Contextual Tabs

Ribbon Tabs and Contextual Tabs are Array Collections and the inner tag is used for them.

Objects with in these Array Collections can be extended by using hyphen. E.g. e-applicationTab-menuItemId="menu"

ASPX-CS

```
@section ControlsSection{
<div ng-app="ribbonApp">
<div ng-controller="RibbonCtrl">
<div id="defaultRibbon" ej-ribbon e-width="100%" e-applicationtab-
menuitemid="menu" e-expandpinsettings-tooltip="Collapse the Ribbon" e-
collapsepinsettings-tooltip="Pin the Ribbon" e-applicationtab-type="menu"
e-create="createControl">
<div e-tabs>
<div e-tab e-id="home" e-text="HOME">
<div e-groups>
<div e-group e-text="New" e-aligntype="rows">
<div e-content>
<div e-content e-defaults-type="button" e-defaults-width="60" e-defaults-
height="70">
<div e-groups>
<div e-group e-id="new" e-text="New" e-customtooltip-title="New" e-
customtooltip-content="
<h6>New.<&#47;h6>" e-buttonsettings-contenttype="imageonly" e-
buttonsettings-imageposition="imagetop" e-buttonsettings-prefixicon="e-
icon e-ribbon e-new" e-buttonsettings-click="executeAction">
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<div e-contextualtabs>
<div e-contextualtab e-backgroundcolor="#FCFBEB" e-bordercolor="#F2CC1C">
<div e-tabs>
<div e-tab e-id="Design" e-text="DESIGN">
<div e-groups>
<div e-group e-text="New" e-aligntype="rows">
<div e-content>
<div e-content e-defaults-type="button" e-defaults-width="60" e-defaults-
height="70">
<div e-groups>
<div e-group e-id="Design" e-text="Design" e-buttonsettings-
prefixicon="e-icon e-shape" e-buttonsettings-contenttype="textandimage"
e-buttonsettings-imageposition="imagetop"></div>
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<ul id="menu">
<li>
<a>FILE</a>
<ul>
<li>
<a>Save</a>
</li>
</ul>
</li>
</ul>
</div>
</div>
}
@section ScriptSection{
<script type="text/javascript">
angular.module('ribbonApp', ['ejangular'])
.controller('RibbonCtrl', function ($scope) {
});
</script>
}
```



Gallery and Custom Tooltip

GalleryItems is an Array Collection and the inner tag is used for it inside the Ribbon Groups. Object of this Array Collection can be extended by using hyphen.

Custom Tooltip is an Object of Ribbon Groups and can extend it by using hyphen. E.g. e-customToolTip-title.

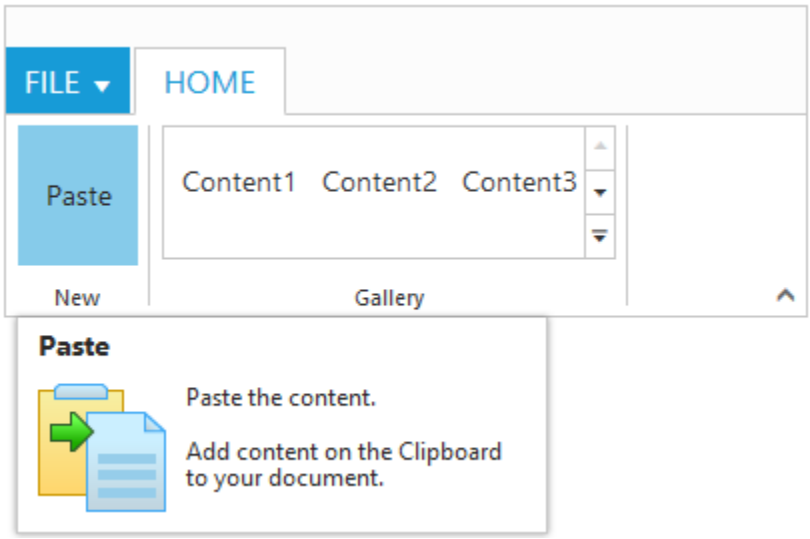
ASPX-CS

```
@section ControlsSection{
<div ng-app="ribbonApp">
<div ng-controller="RibbonCtrl">
<div id="defaultRibbon" ej-ribbon e-width="400" e-applicationtab-
menuitemid="menu" e-expandpinsettings-tooltip="Collapse the Ribbon" e-
collapsepinsettings-tooltip="Pin the Ribbon" e-applicationtab-type="menu"
e-create="createControl">
```

```
<div e-tabs>
<div e-tab e-id="home" e-text="HOME">
<div e-groups>
<div e-group e-text="New" e-aligntype="columns">
<div e-content>
<div e-content e-defaults-type="button" e-defaults-width="60" e-defaults-
height="70">
<div e-groups>
<div e-group e-id="paste" e-text="Paste" e-customtooltip-title="Paste" e-
customtooltip-content="
<h6>Paste the content.
<br &#47;>
<br &#47;>Add content on the Clipboard to your document.<&#47;h6>" e-
customtooltip-prefixicon="e-icon e-pastetip">
</div>
</div>
</div>
</div>
</div>
<div e-group e-text="Gallery" e-aligntype="rows">
<div e-content>
<div e-content>
<div e-groups>
<div e-group e-id="Gallery" e-columns="3" e-itemheight="54" e-
itemwidth="68" e-expandedcolumns="4" e-type="gallery">
<div e-galleryitems>
<div e-galleryitem e-text="Content1" e-tooltip="GalleryContent1" e-
buttonsettings-contenttype="textonly"></div>
<div e-galleryitem e-text="Content2" e-tooltip="GalleryContent2" e-
buttonsettings-contenttype="textonly"></div>
<div e-galleryitem e-text="Content3" e-tooltip="GalleryContent3" e-
buttonsettings-contenttype="textonly"></div>
<div e-galleryitem e-text="Content4" e-tooltip="GalleryContent4" e-
buttonsettings-contenttype="textonly"></div>
<div e-galleryitem e-text="Content5" e-tooltip="GalleryContent5" e-
buttonsettings-contenttype="textonly"></div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<ul id="menu">
<li>
<a>FILE</a>
<ul>
<li>
<a>Save</a>
</li>
</ul>
</li>
</ul>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
}
@section ScriptSection{
<script type="text/javascript">
angular.module('ribbonApp', ['ejangular'])
.controller('RibbonCtrl', function ($scope) {
});
</script>
}
```



Load on Demand

Set EnableOnDemand as true to enable load tab and backstage contents dynamically. Loading content on demand improves the initial rendering time of the ribbon by rendering tab and backstage content when tabs and backstage items are clicked.

ASPX-CS

```
<div id="newCon">
<table>
<tr>
<td>
<button id="btn1" class="e-bsnewbtnstyle">Blank WorkBook</button>
</td>
</tr>
</table>
</div>
<div id="accountCon">
<div class="e-userDiv">
<span>User Information</span>
<div>
<div class="e-accuser e-newpageicon"></div>
<div class="e-userCon">
<div>user</div>
<div>john@syncfusion.com</div>
</div>
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

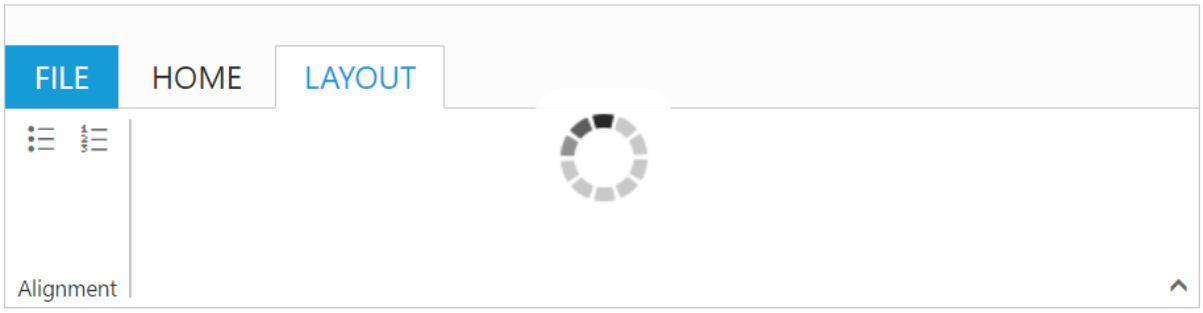
```
<a href="#">Sign out</a>
</div>
@(Html.EJ().Ribbon("defaultRibbon")
.Width("50%")
.EnableOnDemand(true)
.ApplicationTab(app =>
{
app.Type(ApplicationTabType.Backstage).BackstageSettings(bsSettings =>
{
bsSettings.Text("FILE").Height("350").Width("100%").HeaderWidth("120").Pa
ges(bsPage =>
{
bsPage.Id("new").Text("New").ContentID("newCon").Add();
bsPage.Id("close").Text("Close").EnableSeparator(true).ItemType(ItemType.
Button).Add();
bsPage.Id("account").Text("Office
Account").ContentID("accountCon").Add();
});
});
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").AlignType(RibbonAlignType.Columns).GroupExpand
erSettings(groupExp =>
{
groupExp.ToolTip("Clipboard");
}).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("paste").Text("Paste").ToolTip("Paste").ButtonSettings(ne
w ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-ribbonpaste"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).IsBig(true).Width("50px").Height("70px")
).Add();
}).Add();
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("new").Text("New").ToolTip("New").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-new"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("70px")).Add();
}).Add();
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
tab.Id("layout").Text("LAYOUT").TabGroups(tabgroup =>
{
tabgroup.Text("Alignment").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("bullet").Text("Bullet
Format").ToolTip("Bullets").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-bullet"
}).Add();
contentGroup.Id("number").Text("Number
Format").ToolTip("Numbering").EnableSeparator(true).ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-numbericon"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).IsBig(false)).Add();
}).Add();
}).Add();
})
)
<script>
$("#btn1").ejButton({
size: "large",
height: 200,
width: 225,
contentType: "textandimage",
imagePosition: "imagetop",
prefixIcon: "e-icon e-blank e-infopageicon"
});
</script>
<style>
.e-accuser {
background-image: url("../Content/ejthemes/common-
images/ribbon/User.jpg");
}
.e-blank {
background-image: url("../Content/ejthemes/common-
images/ribbon/blank.png");
}
.e-infopageicon {
background-repeat: no-repeat;
height: 150px;
width: 198px;
}
.e-newpageicon {
background-repeat: no-repeat;
height: 42px;
width: 42px;
}
.e-bspagestyle {
line-height: 0;
font-size: 30px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.e-ribbon .e-ribbonbackstagepage .e-bsnewbtnstyle {
color: #212121;
background: #fdfdfd;
margin: 20px;
}
</style>
```



Initially Collapsible

Set Collapsible as true to render ribbon control in collapsed state, which results ribbon tabs to render without any content initially.

While using initially collapsible ribbon with EnableOnDemand feature improves the performance by reducing initial loading time of ribbon.

ASPX-CS

```
<div id="newCon">
<table>
<tr>
<td>
<button id="btn1" class="e-bsnewbtnstyle">Blank WorkBook</button>
</td>
</tr>
</table>
</div>
<div id="accountCon">
<div class="e-userDiv">
<span>User Information</span>
<div>
<div class="e-accuser e-newpageicon"></div>
<div class="e-userCon">
<div>user</div>
<div>john@syncfusion.com</div>
</div>
</div>
</div>
<a href="#">Sign out</a>
</div>
@(Html.EJ().Ribbon("defaultRibbon")
.Width("50%")
.EnableOnDemand(true)
.Collapsible(true)
.ApplicationTab(app =>
{
```

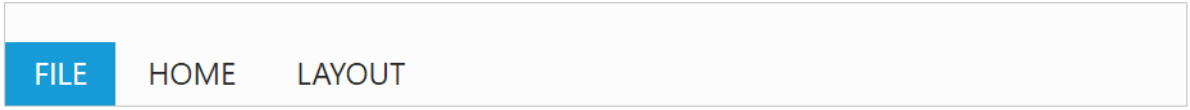
```
app.Type(ApplicationTabType.Backstage).BackstageSettings(bsSettings =>
{
bsSettings.Text("FILE").Height("350").Width("100%").HeaderWidth("120").Pa
ges(bsPage =>
{
bsPage.Id("new").Text("New").ContentID("newCon").Add();
bsPage.Id("close").Text("Close").EnableSeparator(true).ItemType(ItemType.
Button).Add();
bsPage.Id("account").Text("Office
Account").ContentID("accountCon").Add();
});
});
})
.RibbonTabs(tab =>
{
tab.Id("home").Text("HOME").TabGroups(tabgroup =>
{
tabgroup.Text("Clipboard").AlignType(RibbonAlignType.Columns).GroupExpand
erSettings(groupExp =>
{
groupExp.ToolTip("Clipboard");
}).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("paste").Text("Paste").ToolTip("Paste").ButtonSettings(ne
w ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-ribbonpaste"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).IsBig(true).Width("50px").Height("70px")
).Add();
}).Add();
tabgroup.Text("New").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
contentGroup.Id("new").Text("New").ToolTip("New").ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
ImagePosition = ImagePosition.ImageTop,
PrefixIcon = "e-icon e-ribbon e-new"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).Width("60px").Height("70px")).Add();
}).Add();
}).Add();
tab.Id("layout").Text("LAYOUT").TabGroups(tabgroup =>
{
tabgroup.Text("Alignment").AlignType(RibbonAlignType.Rows).Content(ctn =>
{
ctn.ContentGroups(contentGroup =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
contentGroup.Id("bullet").Text("Bullet
Format").ToolTip("Bullets").ButtonSettings(new ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-bullet"
}).Add();
contentGroup.Id("number").Text("Number
Format").ToolTip("Numbering").EnableSeparator(true).ButtonSettings(new
ButtonProperties()
{
ContentType = ContentType.ImageOnly,
PrefixIcon = "e-icon e-ribbon e-numbericon"
}).Add();
}).ContentDefaults(df =>
df.Type(RibbonButtonType.Button).IsBig(false)).Add();
}).Add();
}).Add();
})
)
<script>
$("#btn1").ejButton({
size: "large",
height: 200,
width: 225,
contentType: "textandimage",
imagePosition: "imagetop",
prefixIcon: "e-icon e-blank e-infopageicon"
});
</script>
<style>
.e-accuser {
background-image: url("../Content/ejthemes/common-
images/ribbon/User.jpg");
}
.e-blank {
background-image: url("../Content/ejthemes/common-
images/ribbon/blank.png");
}
.e-infopageicon {
background-repeat: no-repeat;
height: 150px;
width: 198px;
}
.e-newpageicon {
background-repeat: no-repeat;
height: 42px;
width: 42px;
}
.e-bspagestyle {
line-height: 0;
font-size: 30px;
}
.e-ribbon .e-ribbonbackstagepage .e-bsnewbtnstyle {
color: #212121;
background: #fdfdfd;
margin: 20px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</style>
```



How to

Get Ribbon object

After Ribbon initialization, Ribbon object is stored in a container element of Ribbon and it can be accessed for further processing.

JS

```
// "defaultRibbon" is Id of Ribbon control
var ribbonObject = $("#defaultRibbon").ejRibbon("instance");
[or]
var ribbonObject = $("#defaultRibbon").data("ejRibbon");
```

RichTextEditor

Overview

Rich text editor is a component that help you to display or edit the content including tables, hyperlinks, paragraphs, lists, video, and images. The editor supports file and folder management using FileExplorer component.

The following are some of the key features of the editor:

** Editing with formatting * Toolbar * Content Area with customization * Image and File browser * Links * Tables * Localization * XHTML validation*

Getting Started

This section explains the step-by-step instructions to create RichTextEditor in an ASP.NET MVC application.

Create your first RichTextEditor in MVC

1. Create Syncfusion ASP.NET MVC application. You can refer MVC Getting Started documentation to create new project and add necessary dll's and script files. 2. Add the following code example to the corresponding view page to render the RichTextEditor.

HTML

```
@Html.EJ().RTE("rteSample").Width("820px")
```

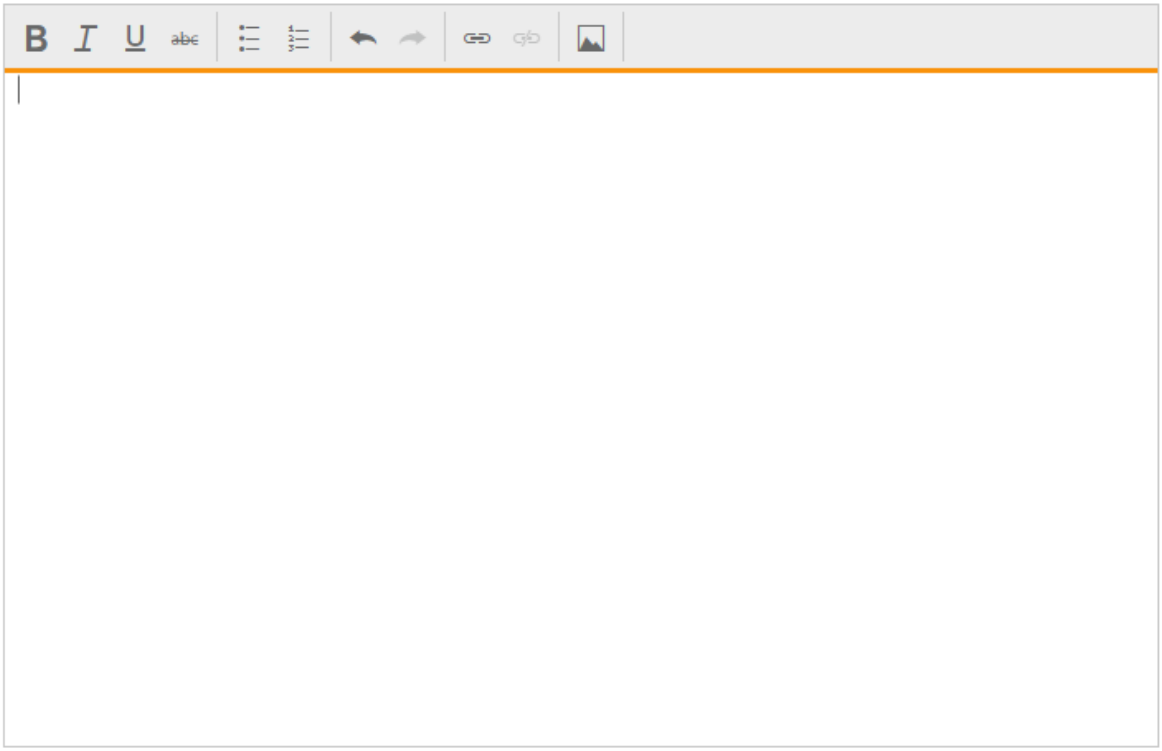
Toolbar–Configuration

You can configure the toolbar with the tools as your application requires.

HTML

```
@{
```

```
List<String> toolsList = new List<string>() { "style", "lists",
"doAction", "links", "images" };
List<String> style = new List<string>() { "bold", "italic", "underline",
"strikethrough" };
List<String> lists = new List<string>() { "unorderedList", "orderedList"
};
List<String> doAction = new List<string>() { "undo", "redo" };
List<String> links = new List<string>() { "createLink" };
List<String> images = new List<string>() { "image" };
}
@Html.EJ().RTE("rteSample").Width("820px").ToolsList(toolsList).Tools(too
l =>
tool.Styles(style).Lists(lists).DoAction(doAction).Links(links).Images(im
ages))
```



Setting and Getting Content

You can set the content of the editor as follows.

HTML

```
@Html.EJ().RTE("rteSample").Width("820px").ContentTemplate(@<div>
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content.
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</div>)
```

To retrieve the editor contents using Value property in Post Back action,

HTML

```
[HttpPost]
[ValidateInput(false)]
public ActionResult RichTextEditorFeatures(string rteSample)
{
ViewBag.data = rteSample;
return View();
}
```

Toolbar Configuration

The editor's toolbar contains a collection of tools such as bold, italic and text alignment buttons that are used to format the content.

However, in most integrations, it's desirable to change the toolbar configuration to suit needs. Fortunately, that's quite easy to do too.

Property	Description
toolsList	The toolsList option allows you to choose which tools appear on the toolbar, as well as the order and grouping of those items
tools	The toolsList property is used to get the root group order and tools property is used to get the inner order of the corresponding groups displayed.

Note: By default, when you tab from the textbox to the RTE, the first tools in the Toolbar of RTE will get focus not in the text area.

But we can able to focus the RTE text area by setting the tab index attribute as -1 to avoid the focus on RTE toolbar when we tab from textbox to RTE - Demo

Toolbar Items

The following table lists the available buttons and dropdowns on the toolbar:

Name	Summary	Initialization	isDefault ?
Font	Applies font type, size, and color to the content	List<String> toolsList = new List<string>() {" font"}; List<String> font = new List<string>() { "fontName", "fontSize", "fontColor", "backgroundColor" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Font(font)).Render();}	No
Font style	Applies bold, italic, underline, and strikethrough formatting to the content	List<String> toolsList = new List<string>() { " style"}; List<String> style = new List<string>() { "bold", "italic", "underline", "strikethrough" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Styles(style)).Render();}	yes

Copyright © 2001 - 2020 Syncfusion Inc.

Alignment	Align the content with left, center, and right margin.	List<String> toolsList = new List<string>() { " alignment"}; List<String> alignment = new List<string>() { "justifyLeft", "justifyCenter" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Alignment (alignment)).Render();}	yes
List	Create a new list item (bulleted/numbered).	List<String> toolsList = new List<string>() { "lists"}; List<String> lists = new List<string>() { "unorderedList", "orderedList" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Lists(lists)).Render();}	yes
Indents	Allows to increase/decrease the indent level of the content.	List<String> toolsList = new List<string>() { "indenting"}; List<String> indenting = new List<string> { "outdent", "indent" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Indenting(indenting)).Render();}	yes
Undo/Redo Action	Allows to undo/redo the actions	List<String> toolsList = new List<string>() {"doAction"}; List<String> doAction = new List<string>() { "undo", "redo" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. DoAction (doAction)).Render();}	yes
Hyperlink	Creates a hyperlink to a text or image to a specific location in the content.	List<String> toolsList = new List<string>() {"links"}; List<String> links = new List<string>() { "createLink", "removeLink" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Links (links)).Render();}	yes
Images	Inserts an image from an online source or local computer.	List<String> toolsList = new List<string>() {"images"}; List<String> images = new List<string>() { "image" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Images (images)).Render();}	yes
Media	Allows to embed a video into the document.	List<String> toolsList = new List<string>() {"media"}; List<String> media = new List<string>() { "video" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool.Media (media)).Render();}	yes
Table	Allows to add or modify Tables	List<String> toolsList = new List<string>() {"tables"}; List<String> tables = new List<string>() { "createTable", "addRowAbove", "addRowBelow", "addColumnLeft", "addColumnRight", "deleteRow", "deleteColumn", "deleteTable" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Tables(tables)).Render();}	yes

Copyright © 2001 - 2020 Syncfusion Inc.

Casing	Change the case of selected text in the content	List<String> toolsList = new List<string>() {"casing"}; List<String> casing = new List<string>() { "upperCase", "lowerCase" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Casing(casing)).Render();}	no
Scripts	Makes the selected text as superscript (higher) or subscript (lower).	List<String> toolsList = new List<string>() {"effects"}; List<String> effects = new List<string>() { "superscript", "subscript" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool. Effects(effects)).Render();}	no
Format	Clears the formatting options like bold, italic, underline and more.	List<String> toolsList = new List<string>() {"formatStyle"}; List<String> formatStyle = new List<string>() { "format" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool.FormatStyle(formatStyle)).Render();}	yes
Clipboard Actions	Controls the clipboard actions by applying specific action on the selected content.	List <String> toolsList = new List<string>() {"clipboard"}; List<String> clipboard = new List<string>() { "cut", "copy", "paste" }; @{Html.EJ().RTE("rteSample").ToolsList(toolsList).Tools(tool => tool.Clipboard(clipboard)).Render();}	no

Rearrange Group

The toolbar contains groups, which are similar or related functionalities of toolbar items for efficient access. By default, the groups are arranged using the following order:

HTML

```
List<String> toolsList = new List<string>() { "formatStyle", "font",
"style", "effects", "alignment", "lists", "indenting", "clipboard",
"doAction", "clear", "links", "images", "media", "tables", "casing",
"customTool","view" };
```

The group can be rearranged on customization using the ToolsList property.

HTML

```
@{
List<String> toolsList = new List<string>() { "style", "lists",
"doAction", "links", "images" };
List<String> style = new List<string>() { "bold", "italic", "underline",
"strikethrough" };
List<String> lists = new List<string>() { "unorderedList", "orderedList"
};
List<String> doAction = new List<string>() { "undo", "redo" };
List<String> links = new List<string>() { "createLink" };
List<String> images = new List<string>() { "image" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</p>)
.ShowFooter(true)
.ToolsList(toolsList)
.Tools(tool =>
tool.Links(links).Images(images).Styles(style).Lists(lists).DoAction(doAc
tion)).Render();}
```

Note: If you are not specifying any group in **ToolsList** property, the editor will create the toolbar with default group.

Undo and Redo

Undo and Redo buttons allow you to editing the text by disregard/cancel the recently made changes and restore it to previous state. It is a useful tool to restore the performed action which got changed by mistake. Up to 50 actions can be undo/redo in the editor by default.

To undo and redo operations, do one of the following:

- Press the undo/redo button on the toolbar
- Press the `Ctrl + Z` , `Ctrl + Y` combination on the keyboard

HTML

```
@{
List<String> toolsList = new List<string>() { "doAction" };
List<String> doAction = new List<string>() { "undo", "redo" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ToolsList(toolsList).Tools(tool => tool.DoAction(doAction))
.Render();}
<br />
```

Clipboard Operations

The editor provides support for the clipboard operations (cut, copy, and paste) in all text and images using the toolbar buttons and the keyboard shortcuts. Toolbar includes buttons through which the clipboard operations, such as Cut, Copy, and Paste can be accessed.

You can use the keyboard shortcuts to perform the clipboard operations.

Copyright © 2001 - 2020 Syncfusion Inc.

- Cut - CTRL + X
- Copy - CTRL + C
- Paste - CTRL + V

Some browsers block the clipboard access from JavaScript. If you want to use the Cut, Copy, and Paste buttons on the toolbar, you need to allow JavaScript to use the clipboard. If you don't want to do this configuration, use CTRL+C, CTRL+X, and CTRL+V keyboard commands.

HTML

```
@{
List<String> toolsList = new List<string>() { "clipboard" };
List<String> clipboard = new List<string>() { "cut", "copy", "paste" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ToolsList(toolsList).Tools(tool => tool.Clipboard(clipboard))
.Render();}
```

Specify Custom Tools

Beside the available built-in tools, the RichTextEditor's toolbar functionality can be extended through custom tools, defined in the tools through the customTools options, which renders the custom toolbar items along with the built-in toolbar items.

The example below demonstrates how to add a custom tool button

HTML

```
@{
List<String> toolsList = new List<string>() { "customTools" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</p>).ShowFooter(true).ToolsList(toolsList).Tools(tool =>
tool.CustomTools(customTools => customTools.Text("Insert -
O").Tooltip("Special Characters").Css("insert-special-
character").Action("CustomTool").Add())).Render();}
```

Upon clicking the "Insert" button, the special character will be added to the RTE's content.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<script>
var rteObj =  $("#rteSample").data("ejRTE");
$(".insert-special-character").ejButton();
$("#specialCharacter").ejDialog({ enableResize: false, enableModal: true,
showOnInit: false, width: "auto", position: { X: 218, Y: 38 } });
$(".special-table tbody tr td" ).addClass("special-td").on( "click",
customTdClick);
function customTdClick(args) {
rteObj.executeCommand("inserthtml", args.currentTarget.innerText);
$("#specialCharacter").ejDialog("close");
}
function CustomTool(args){
$("#specialCharacter").ejDialog("open");
}
</script>
```

Define the CSS that will be applied to the custom tool.

HTML

```
<style>
.special-table tr td
{
border:1px solid #c8c8c8;
}
.special-td:hover
{
background-color:#86bcea;
cursor:pointer;
}
</style>
```

Note: The CSS class that defined for custom tool is directly applies to the newly added toolbar item.

Information: The custom buttons get a insert-special-character CSS class to allow styling, where name is the name specified in the custom tool configuration.

Types of responsive toolbar

Two types of toolbar modes are available for RTE in responsive mode. This includes "inline" and "popup". Toolbar Type can be set through API toolbarOverflowMode whose default value is "popup".

HTML

```
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>The Rich
Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</p>).ToolbarOverflowMode(ToolbarOverflowMode.Inline).IsResponsive(true).
Render();}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: If you are not specifying toolbarOverflowMode, responsive RTE will be rendered with default popup toolbar.

Working with Content

The editor creates the iframe element as the content area on control initialization, it is used to display and editing the content. In Content Area, the editor displays only the body tag of a < iframe > document. To set or modify details in the < head > tag, use Source view of the editor.

IFrame Attributes

The editor allows to passing an additional attributes to body tag of a < iframe > element using IFrameAttributes property. The property contains name/value pairs in string format, it is used to override the default appearance of the content area.

Note: The content area's font, color, margins, and background can be overridden using iframeAttributes property. You can specifies the editable behavior (content editable) of the content also in this property.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.IFrameAttributes(new Dictionary<string, object> { { "style",
"background-color:#e0ffff;color:#6495ed;" } })
.Render();}
```

Note: Background image for the RTE control : Link

Set default font for the Iframe : Link

Adding CSS File

The editor offers to add external CSS file to style the < iframe > element. Easily change the appearance of editor's content using an external CSS file. For example, apply default styles for headings (h1, h2, etc.) and lists (bulleted or numbered) of the editor's content.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ExternalCSS("/Content/Css/iframe-custom.css")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Render();}
```

The new file named iframe-custom.css is created and moved to the content folder with the following styles.

HTML

```
div {
color:navy;
margin-left:20px;
}
ul {
display:block;
list-style-type:square;
margin-left:10px;
}
ol {
display:block;
list-style-type:circle;
margin-right:10px;
}
```

Note: Our RTE control editor section is an iframe. An iframe has another scope, so we can't access it to style using class which is defined in main document. To set the styles for the contents that kept inside the editor(iframe) we have to append the styles link in iframe head section.

Content Editable

The editor provides option to control the editable behavior using AllowEditing property. When the AllowEditing property is set to false, the editor disables its editing functionality.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.AllowEditing(false)
.Render();}
```

The ContentEditable attribute allows you to make any element of HTML content to become editable or non-editable.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
<p> Cannot Edit </p>
</div>)
.ClientSideEvents(e => e.Create("Create"))
.Render();}
<script>
function Create() {
var editor = $("#rteSample").ejRTE("instance");
var iframeDoc = editor.getDocument();
var paragraph = $("p", iframeDoc.body);
$($(paragraph)[0]).attr("contenteditable", "false");
}
</script>
```

Note: Content editable is fully compatible with latest browsers, to know more details, see here.

Submit Content

The editor allows you to process its content before it is being submitted to the server on form submit event. You can use this option to validate content on the client side to prevent invalid data from being submitted to the server.

This example shows how to encode the HTML content before form submit event.

HTML

```
<form>
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</p>)
.ShowFooter(true)
.ShowHtmlSource(true)
.Render();}
<br />
@{Html.EJ().Button("Submit").Text("Submit").Type(ButtonType.Submit).Rende
r();}
</form>
<script type="text/javascript">
$("form").on("submit", function () {
var editor = $("#rteSample").data("ejRTE");
var encoded = $('<div />').text(editor.model.value).html();
$("#rteSample").val(encoded);
});
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Refresh

When you move the editor's wrapper element into another DOM element, the editor needs to be reinitialized by the refresh method to retain its content. The method reload the content area and rebind the events of the editor.

HTML

```
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</p>)
.Render();}
<br />
@{Html.EJ().Dialog("Target").Width("600px").Render();}
@{Html.EJ().Button("Append").Text("Append").ClientSideEvents(e =>
e.Click("appendTo")).Render();}
@{Html.EJ().Button("Refresh").Text("Refresh").ClientSideEvents(e =>
e.Click("refresh")).Render();}
<script type="text/javascript">
var editor;
function appendTo() {
editor = $("#rteSample").ejRTE("instance");
editor._rteWapper.appendTo($("#Target"))
}
function refresh() {
editor = $("#rteSample").ejRTE("instance");
editor.refresh()
}
</script>
```

Persistence

The editor is capable to persist its content with HTML format. By default, the persistence support is disabled in the editor. When you set the EnablePersistence property to true, the persistence will be enabled in the editor.

Note: local storage is not supported below ie9 version, therefore persistence support is fallback to cookie.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px")
.EnablePersistence(true)
.Render();}
<br/>
```

Set Default Font Name and Size

- Set a default font name and font size to the font name and size drop-down programmatically.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: •By default, the editor's < iframe > is initialized with "Segoe UI" font name and 3(12pt) font size.

- To change it, select a different font name and font size from the drop-down in the editor's toolbar.

- To apply different font style for particular section of the content, select the text that you would like to change, and select a required font style from the drop-down to apply the changes to the selected text.

HTML

```
@{
List<String> toolsList = new List<string>() { "font" };
List<String> font = new List<string>() { "fontName", "fontSize",
"fontColor", "backgroundColor" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ClientSideEvents(e => e.Create("Create"))
.ToolsList(toolsList).Tools(tool => tool.Font(font))
.Render();}
<br />
<script>
function Create() {
var editor = $("#rteSample").ejRTE("instance");
var obj = editor._fontStyleDDL.ejDropDownList("instance");
obj.selectItemByIndex(7);
var dropdownSize = editor._fontSizeDDL.ejDropDownList("instance");
dropdownSize.selectItemByIndex(5);
}
</script>
```

- Set default font name and size for < iframe >'s body tag using IFrameAttributes property.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.IFrameAttributes(new Dictionary<string, object> { { "style", "font-
family:Arial;;font-size:14px;" } })
.Render();
}
```

- If you want to override the default font from CSS, create a style tag with CSS styles and append it to the < iframe >'s head tag of the editor.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ClientSideEvents(e => e.Create("Create"))
.Render();}
<br />
<script>
function Create() {
var css = "html,body{font-family:sans-serif;font-size:14px; }";
var editorDoc = $("#rteSample").ejRTE("instance").getDocument();
var styleTag = document.createElement("style");
styleTag.type = "text/css";
if (styleTag.styleSheet)
styleTag.styleSheet.cssText = css;
else
styleTag.appendChild(document.createTextNode(css));
editorDoc.head.appendChild(styleTag);
}
</script>
```

Adding Font names and size

If you want to add additional font names and sizes to font drop-down, pass the font information as JSON data and bind it with instance of drop-down.

HTML

```
@{List<String> toolsList = new List<string>() { "font" };
List<String> font = new List<string>() { "fontName", "fontSize",
"fontColor", "backgroundColor" };}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
in the text area.</div>)
.ClientSideEvents(e => e.Create("Create"))
.ToolsList(toolsList).Tools(tool => tool.Font(font))
.Render();}
<br />
<script>
function Create() {
var editor = $("#rteSample").ejRTE("instance");
editor.defaults.fontName.push({ text: "Calibri Light", value:
"CalibriLight" }, { text: "Calibri", value: "Calibri" });
editor.defaults.fontSize.push({ text: "8 (42pt)", value: "8" });
var obj = editor._fontStyleDDL.ejDropDownList("instance");
var dropdownSize = editor._fontSizeDDL.ejDropDownList("instance");
obj.option({ "dataSource": editor.defaults.fontName });
dropdownSize.option({ "dataSource": editor.defaults.fontSize });
obj.selectItemByValue("CalibriLight");
dropdownSize.selectItemByValue("8");
}
</script>
```

Apply Font color and Background color

If you want to apply font color or background color for a selected content of RTE you can use font color and background color tools. These tools contains a color palette with basic colors along with an option called **"More colors.."** in order to choose custom colors from color picker dialog.You can apply transparent background color for selected text through **transparent** button available in background color palette.

HTML

```
@{List<String> toolsList = new List<string>() { "font" };
List<String> font = new List<string>() { "fontName", "fontSize",
"fontColor", "backgroundColor" };}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.</div>)
.ToolsList(toolsList).Tools(tool => tool.Font(font))
.Render();}
```

Insert the content at cursor

If you want to insert/paste the content at the current cursor position (or) to replace the selected content with some formatting, you can use pasteContent method in the editor.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.Render();}
<br />
@{Html.EJ().Button("sample").Width("100px").Text("PasteContent").ClientSi
deEvents(e => e.Click("pasteContent")).Render();}
<script>
function pasteContent() {
var editor = $("#rteSample").ejRTE("instance");
var selectedHtml = editor.getSelectedHtml();
editor.pasteContent("<p style='background-color:yellow;color:skyblue'> "
+ selectedHtml + " </p>");
}
</script>
```

Footer

This option allows you to specify which footer elements should be shown at the bottom of the editor. The available footer elements are listed below:

1. HTML View
2. HTML Tag Info
3. Characters Count / Word Count
4. Clear Format
5. Resizer

When the footer is enabled, it displays the HTML tag, word Count, character count, clear format, resize icon and clear all the content icons, by default.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.Render();}
```

Source View

RichTextBox includes the ability for users to directly edit HTML code via “Source View” in a separate dialog. If you made any modification in Source view directly, click Update button to synchronize with Design view. This provides power users with more flexibility over the content they create.

Copyright © 2001 - 2020 Syncfusion Inc.

You can paste HTML text into Source view. If you cut or copy from HTML source such as page source of Browser, paste as HTML (without escape characters) into Source view.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.ShowHtmlSource(true)
.Render();}
```

Note: Source view is useful for working directly with raw HTML text, so this tool is mainly used for advanced users who would like to have more control over the source of their content.

HTML Tag Info

The HTML tag info tool that shows the path of currently selected tag along with hierarchy of parent tags to which it belongs. The tag information is displayed at the bottom of the editor. It is used to determine which element has the focus in the editor's content.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.ShowHtmlTagInfo(true)
.Render();}
```

Note: The outermost tag is the body tag of < iframe > element in design view, so it shows the path from currently selected path to the body tag.

Characters Count/Word Count

The editor automatically counts the number of characters and words in the content while you type. The characters and words count displayed at the bottom of the editor. You can limit the number of characters in your content using MaxLength property. By default, the editor sets the characters limit value as 7000 characters.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.ShowWordCount(true)
.ShowCharCount(true)
.Render();}
```

Note: The editor counts the characters by including the space, and this validation occurs while pasting the content into the editor also.

Clear Format

The clear format tool is useful to remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. As a result, all the text formatting will be cleared and return to its default formatting styles. When you set the ShowClearFormat property to true, the clear format tool will be displayed at bottom of the editor.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.ShowClearFormat(true)
.Render();}
```

Resize Handle

When you set the EnableResize property to true, resize handle will be displayed at bottom-right corner of the editor. You can drag the handle to change its size. On resizing, the editor will automatically adjust the toolbar, content area, and footer within it accordingly. Resize limits can be defined via MinHeight, MaxHeight, MinWidth, and MaxWidth properties. You can specify the size of the editor programmatically through the Height and Width properties.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.EnableResize(true)
.MinWidth("250")
.MinHeight("250")
.MaxHeight("500")
.MaxWidth("750")
.Render();}
```

Note: 1. As resizable option will be added in the footer of RTE, so set the showFooter property also as “true”.

2. In order to showcase only the resizer, disable the other properties in the Footer such as showClearFormat, showClearFormat, showCharCount, showWordCount
3. When you set the enableRTL property to true, the resize handle will automatically positioned to the bottom-left corner of the editor.

Characters Count/Word Count

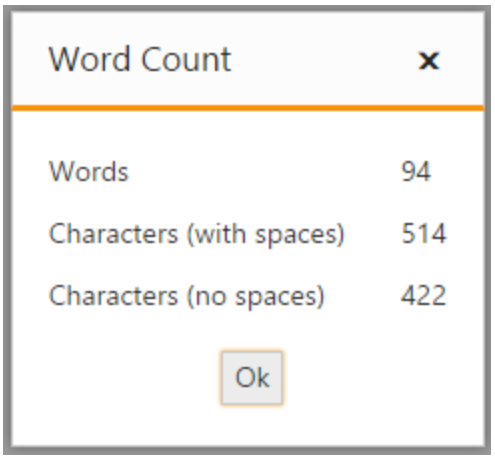
The editor automatically counts the number of characters and words in the content while you type. The characters and words count displayed at the bottom of the editor. You can limit the number of characters in your content using MaxLength property. By default, the editor sets the characters limit value as 7000 characters.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowFooter(true)
.ShowWordCount(true)
.ShowCharCount(true)
.Render();}
```

By clicking the Characters Count/Word Count labels in footer , The word and character count information dialog is opened. It contains the details of the number of words and characters with and without spacing.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: The editor counts the characters by including the space, and this validation occurs while pasting the content into the editor also.

Context Menu

Editor provides custom context menu support, which enables more interaction on the content modification and also it can be enabled dynamically. The showContextMenu property helps to enable custom context menu within editor area.

Based on the target content type context menu provides different actions. Refer the details with below table.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowContextMenu(true)
.Render();}
```

- Based on the target content type ContextMenu provides different actions- refer the details with below table.

Content-Type	Supported Actions
Text content	cut, copy, paste, add/edit/open/remove hyperlink.
Image content	cut, copy, paste, image properties.
Table content	cut, copy, paste, insert row/column, remove row/column/table, edit table properties, add/edit/open/remove hyperlink.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: We have given support to own context menu by restricting the default browser context menu, which provides you the options for quick access but, with that clipboard action are restricted based on browser behavior.

However we can disable the context menu by using ShowContextMenu API and it needs to be set as false, if you wish to continue with default browser context menu.

Adding an item with the context menu

To add a new item to the editor ContextMenu, you need to use the 'insertMenuOption' method and in order to handle the ContextMenu item click using the 'contextMenuClick' client side-event.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowContextMenu(true)
.Render();}
<script>
$("#rteSample").ejRTE({ contextMenuClick: function(args){//handle menu-
item click action.
} });
var rteObj =  $("#rteSample").data("ejRTE");// Inserts new item to the
context menu
rteObj.insertMenuOption({newItem:"Show Table Details",
targetItem: "Table Properties",
insertType:("insertAfter"),
menuType:{text:false,image:false,hyperlink:false,table:true},
spriteCssClass:"e-rte-toolbar-icon tableProperties"});
</script>
```

Removing an item from the context menu

To remove a menu-item from the editor ContextMenu, you have to use the 'removeMenuOption' method from the ejRTE object and find the method and parameter details with the API-document.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ShowContextMenu(true)
.Render();}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script>
var rteObj = $("#rteSample").data("ejRTE");
rteObj.removeMenuOption("Table-Properties");
</script>
```

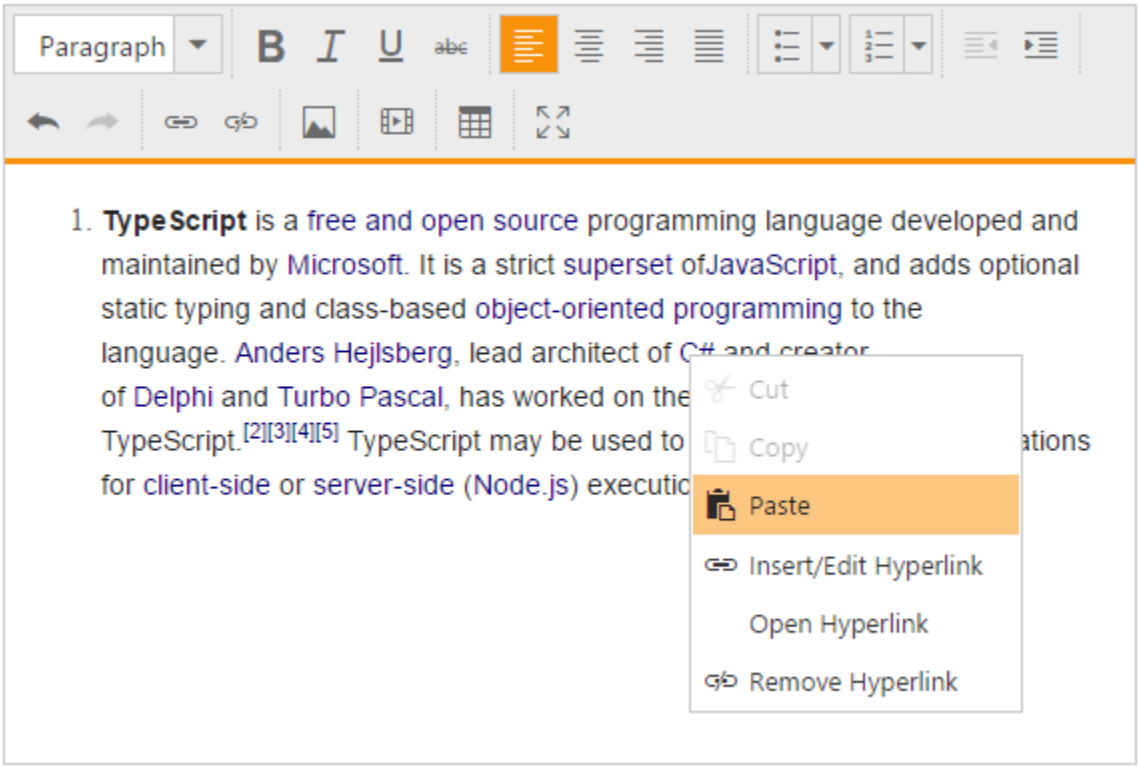


Image and File browser

The editor allows you to manage the images and files using **FileExplorer**. The FileExplorer enables you to insert images from online source as well as local computer where you want to insert the image in your content. The Image and file browser is the ability to upload pictures and link file to the editor.

Insert an Image from Online Source

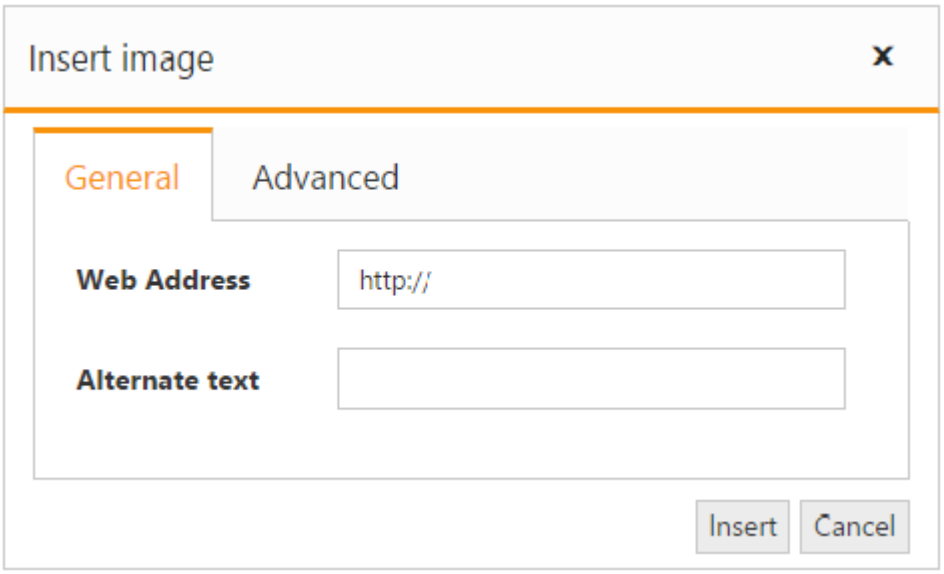
If you want to insert an image from online source like Google, ping, etc., you need to enable images tool on the editor's toolbar. By default, the images tool is open a simple dialog which allows you to inserting an image from online source.

HTML

```
@{
List<String> toolsList = new List<string>() { "images" };
List<String> images = new List<string>() { "image" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content.
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</p>)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ShowFooter(true)
.ToolsList(toolsList)
.Tools(tool => tool.Images(images)).Render();}
```



Insert an Image from Your Computer

Configure the ImageBrowser and FileBrowser property to insert an image from your computer. You can specify the settings required by the FileExplorer to create, read, upload, and destroy the files and images from the explorer.

HTML

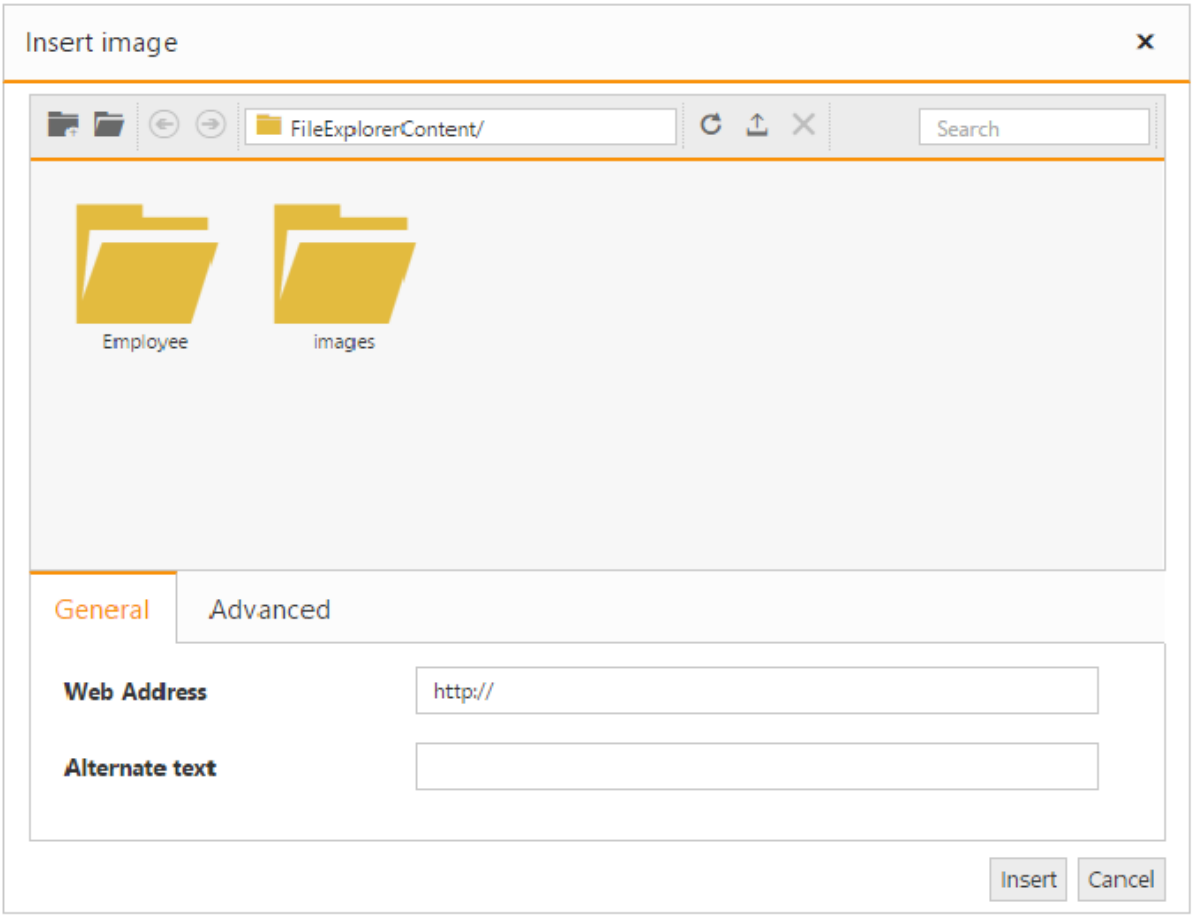
```
@{
List<String> toolsList = new List<string>() { "images" };
List<String> images = new List<string>() { "image" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content.
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</p>)
.ShowFooter(true)
.Tools(tool => tool.Images(images))
.ToolsList(toolsList)
.ImageBrowser(image => image.FilePath("~/FileExplorerContent/")
.ExtensionAllow("*.png,*.gif,*.jpg,*.jpeg")
.AjaxAction(@Url.Content("FileActionMethods")))
.FileBrowser(file => file.FilePath("~/FileExplorerContent/")
.ExtensionAllow("*.png,*.txt,*.jpg,*.docx")
.AjaxAction("FileActionMethods"))
.Render();}
```

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult
FileActionMethods(Syncfusion.JavaScript.FileExplorerParams args)
{
Syncfusion.JavaScript.FileExplorerOperations operation = new
Syncfusion.JavaScript.FileExplorerOperations();
switch (args.ActionType)
{
case "Read":
return Json(operation.Read(args.Path, args.ExtensionsAllow));
case "CreateFolder":
return Json(operation.CreateFolder(args.Path, args.Name));
case "Paste":
return Json(operation.Paste(args.LocationFrom, args.LocationTo,
args.Names, args.Action, args.CommonFiles));
case "Remove":
return Json(operation.Remove(args.Names, args.Path));
case "Rename":
return Json(operation.Rename(args.Path, args.Name, args.NewName,
args.CommonFiles));
case "GetDetails":
return Json(operation.GetDetails(args.Path, args.Names));
case "Download":
operation.Download(args.Path, args.Names);
break;
case "Upload":
operation.Upload(args.FileUpload, args.Path);
break;
}
return Json("");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

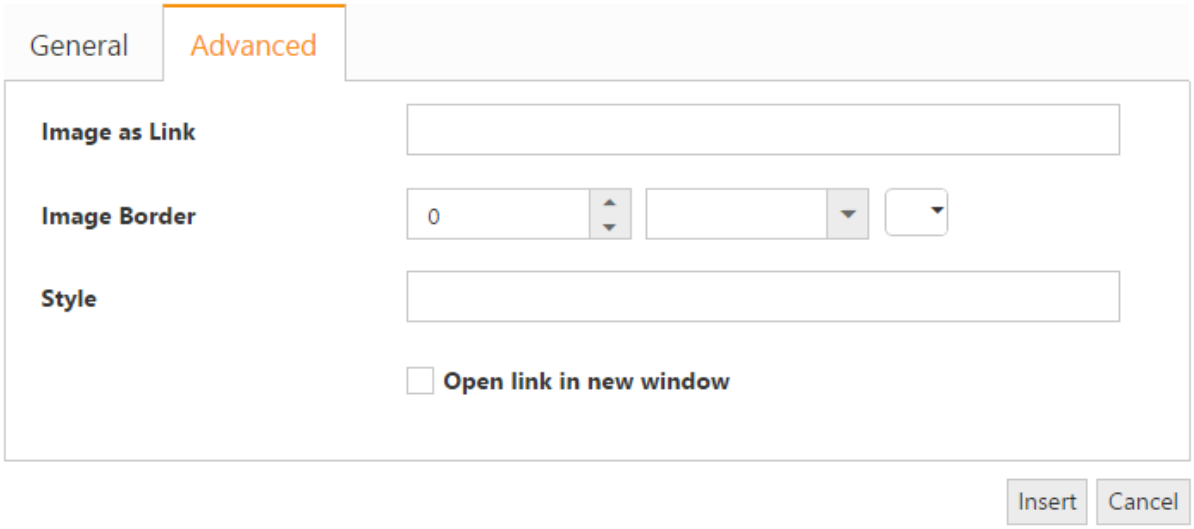


Note: FileExplorer component has been implemented and integrated with the editor in Volume 1, 2015 release. For more information about FileExplorer component, see here.

Image Properties

You can set or modify the properties of an image by using the image dialog. It allows you to add links, apply border and set additional styles to the images. The editor provides an option to specify the alternate text for an image, if the image cannot be displayed.

Copyright © 2001 - 2020 Syncfusion Inc.



Resize an Image

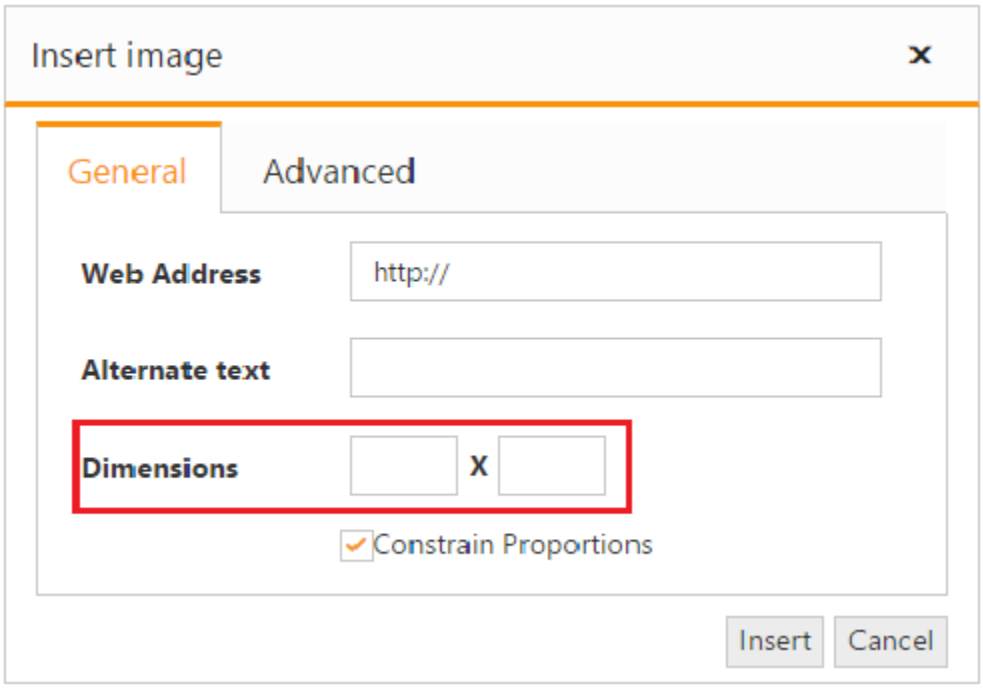
You can resize an image either manually or by setting width and height in the image dialog.

Resize Manually

You can resize an image manually by selecting an image. And drag the handle until the image is resized to the desired size.



Copyright © 2001 - 2020 Syncfusion Inc.



Note: Set the default height and width of the Images which was inserted into the RTE text area in “change” event of RTE - Link

Set Width and Height

The editor provides you to set the Width and Height properties to change the size of an image (rather than forcing you to set in style attributes) using ShowDimensions property. By default, the Constrain Proportion checkbox is selected to resize an image to an exact proportion. To apply the exact width and height that you specify into the Height and Width textboxes, uncheck the Constrain Proportions checkbox.

HTML

```
@{
List<String> toolsList = new List<string>() { "images" };
List<String> images = new List<string>() { "image" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</p>).ShowFooter(true)
.ToolsList(toolsList)
.Tools(tool => tool.Images(images))
.ShowDimensions(true)
.Render();}
```


Copyright © 2001 - 2020 Syncfusion Inc.

Suppression of the Image Browser

The General and Advanced tabs in the RTE Image browser can be removed by setting its corresponding display CSS property to none.

CSS

```
<style type="text/css" class="cssStyles">
div.e-rte-imageTab.e-tab.e-js.e-widget {
display: none;
}
</style>
```

Can remove the Add NewFolder button by using removeToolbarItem property of Image Browser in the RTE create event.

HTML

```
@{
List<String> toolsList = new List<string>() { "images" };
List<String> images = new List<string>() { "image" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content.
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</p>)
.ShowFooter(true)
.Tools(tool => tool.Images(images))
.ToolsList(toolsList)
.ImageBrowser(image => image.FilePath("~/FileExplorerContent/")
.ExtensionAllow("*.png,*.gif,*.jpg,*.jpeg")
.AjaxAction(@Url.Content("FileActionMethods")))
.FileBrowser(file => file.FilePath("~/FileExplorerContent/")
.ExtensionAllow("*.png,*.txt,*.jpg,*.docx")
.AjaxAction("FileActionMethods"))
.ClientSideEvents(e => e.Create("onCreate"))
.Render();}
<script>
function onCreate(args){
this._explorerObj.removeToolbarItem("NewFolder");
}
</script>
```

In RTE control, there is no direct support for the autoUpload option. But this can be achieved by enabling the autoUpload at create event of RTE.

HTML

```
@{
List<String> toolsList = new List<string>() { "images" };
List<String> images = new List<string>() { "image" };
}
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor (RTE) control is an easy to render in client side.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Customer easy to edit the contents and get the HTML content for the
displayed content.
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</p>)
.ShowFooter(true)
.Tools(tool => tool.Images(images))
.ToolsList(toolsList)
.ImageBrowser(image => image.FilePath("~/FileExplorerContent/")
.ExtensionAllow("*.png,*.gif,*.jpg,*.jpeg")
.AjaxAction(@Url.Content("FileActionMethods")))
.FileBrowser(file => file.FilePath("~/FileExplorerContent/")
.ExtensionAllow("*.png,*.txt,*.jpg,*.docx")
.AjaxAction("FileActionMethods"))
.ClientSideEvents(e => e.Create("onCreate"))
.Render();}
<script>
function onCreate(args){
this._explorerObj._uploadtag.data("ejUploadbox").option("autoUpload",
true);
}
</script>
```

Working with Hyperlinks

A hyperlink can be inserted into the editor for quick access to the related information. The hyperlink itself can be a text or an image.

Add and Edit a hyperlink

To add a hyperlink to the editor, follow these instructions:

1. Point the cursor anywhere within the editor where you'd like to insert the link. It is also possible to select a text or an image within the editor and can be converted to the hyperlink.
2. Click the "Insert/Edit hyperlink" tool on the Toolbar.
3. In the **Web Address** box, type or paste the destination for the link you are creating.
4. In the **Text** box, type or edit the required text that you want to display text for the link.

Note: Leave this textbox blank if you want to display the full link as hyperlink in the editor.

5. To display an additional helpful information when you place the pointer on the hyperlink, type the required text in the "Tooltip" field.
6. Click OK button to insert a hyperlink.

HTML

```
@{
List<String> toolsList = new List<string>() { "links" };
List<String> links = new List<string>() { "createLink", "removeLink" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</div>)
.ToolsList(toolsList).Tools(tool => tool.Links(links))
.Render();}
```

Note: hyperlinks on a picture are not always visible, but the pointer's appearance will be changed on positioning the mouse pointer over it.

Remove a hyperlink

If you want to remove a hyperlink from a text or image, select the text or image with the hyperlink and click "Remove Hyperlink" tool from toolbar. It will keep the text or image.

Working with Selection

The editor control provides option to select the all content and in addition to selection of a particular range of content.

Select All

The selectAll method enables you to select the entire content including images in the editor by programmatically.

Note: the selection highlight is invisible if the editor does not have focus. So, if you want to call the selectAll method, focus the editor before.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor (RTE) control is an easy to render in
client side. Customer easy to edit the contents and get the HTML content
for the displayed content. A rich text editor control provides users with
a toolbar that helps them to apply rich text formats to the text entered
in the text area.
</div>)
.Render();}
<br />
@Html.EJ().Button("Button1").ClientSideEvents(e =>
e.Click("selectAll")).Text("Select All")
<script>
function selectAll() {
var editor = $("#rteSample").ejRTE("instance");
editor.selectAll();
}
</script>
```

Select a Range

You can programmatically select a range of content in the editor using the selectRange method. To select a range, create a range object with desired offset position and pass it as arguments to selectRange method. The range object is created from createRange method.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
<ul>
<li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
<li>Aliquam tincidunt mauris eu risus.</li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<li>Vestibulum auctor dapibus neque.</li>
</ul>
</div>).Render();}
<br />
@Html.EJ().Button("button1").ClientSideEvents(e =>
e.Click("select")).Text("Select")
<script>
function select() {
var editor = $("#rteSample").ejRTE("instance");
range = editor.createRange();
var liTag = $(editor.getDocument().body).find("li");
if (!editor._isIE8()) {
range.setStart(liTag[1], 0);
range.setEnd(liTag[2], 1);
}
else {
range = editor.getDocument().body.createTextRange()
range.moveToElementText(liTag[2]);
}
editor.selectRange(range);
}
</script>
```

Get Selection

The following public methods helps you to retrieve the selected content from the editor:

- getText method is used to get the currently selected content as raw text.
- getSelectedHtml method is used to get the HTML source of currently selected content.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
<ul>
<li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
<li>Aliquam tincidunt mauris eu risus.</li>
<li>Vestibulum auctor dapibus neque.</li>
</ul>
</div>).Render();}
<br />
@Html.EJ().Button("button1").ClientSideEvents(e=>e.Click("select")).Text(
"Select")
<script>
function select() {
var editor = $("#rteSample").ejRTE("instance");
var selectedText = editor.getText();
var selectedHtml = editor.getSelectedHtml();
}
</script>
```

Working with Tables

The editor provides tools to add tables in your content. You can add, edit, and remove the table as well as perform other table related tasks.

Copyright © 2001 - 2020 Syncfusion Inc.

Create a Table

By default, Insert Table tool is enabled in the editor's toolbar. There are two ways to insert a table into the editor's content.

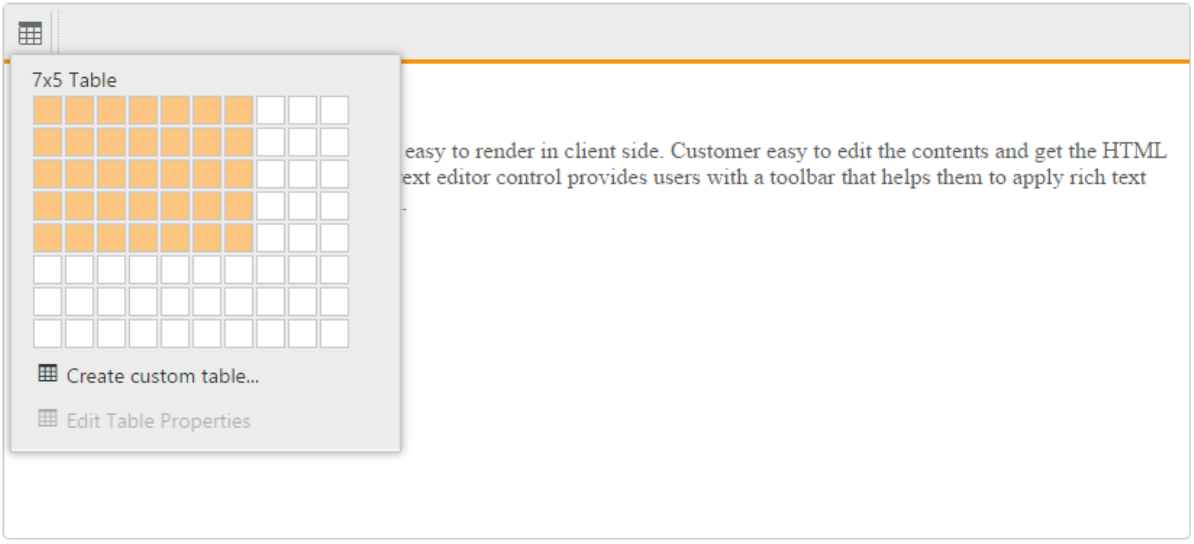
- Insert a table
- Insert a custom table

HTML

```
@{
List<String> toolsList = new List<string>() { "tables" };
List<String> tables = new List<string>() { "createTable", "addRowAbove",
"addRowBelow", "addColumnLeft", "addColumnRight", "deleteRow",
"deleteColumn", "deleteTable" };
}
<br />
@{Html.EJ().RTE("rteSample").Width("820px").ContentTemplate(@<div>
<ul>
<li>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</li>
<li>Aliquam tincidunt mauris eu risus.</li>
<li>Vestibulum auctor dapibus neque.</li>
</ul>
</div>)
.ToolsList(toolsList)
.Tools(tool => tool.Tables(tables)).Render();}
```

Insert a Table

You can insert a basic table by select the "Insert Table" tool from toolbar and drag the cursor over the grid until you highlight the number of columns and rows you want. A table can be inserted as large as with ten columns and eight rows (10 * 8 cells). When you finish inserting tables, click in a cell and start typing or insert an image.

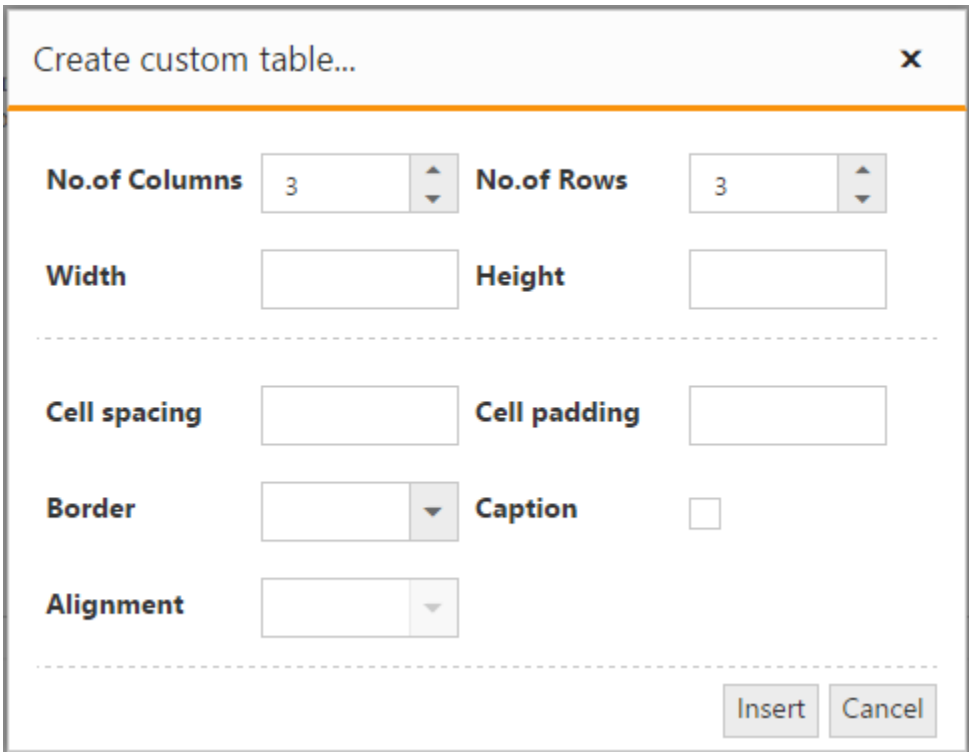


Note: If you want to make any adjustment with drawn table, you can add/remove rows and columns.

Copyright © 2001 - 2020 Syncfusion Inc.

Custom Table

You can use custom table tools to insert tables with custom behaviors. You can create a table with more than ten columns and eight rows, as well as set the table related attributes (such as width, height, cell spacing/padding, and more) using the custom table dialog.

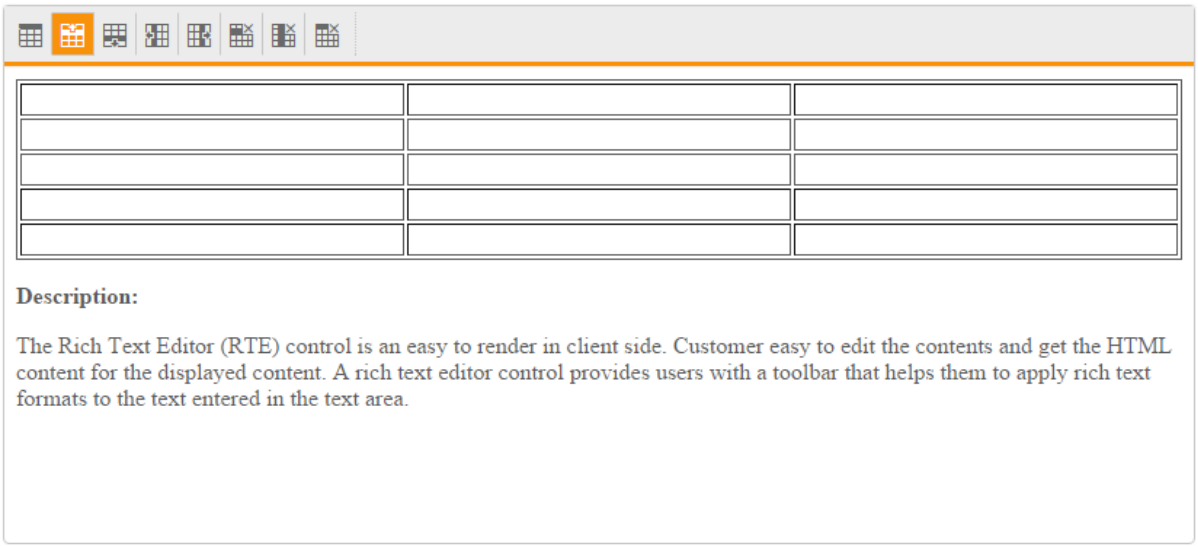


Insert and Delete a Row or Column

You can choose the following options to modify the inserted table from the toolbar, it will be enabled when you're in table cell.

- Add a row above
- Insert row below
- Insert column left
- Insert column right
- Delete a column
- Delete a row

Copyright © 2001 - 2020 Syncfusion Inc.



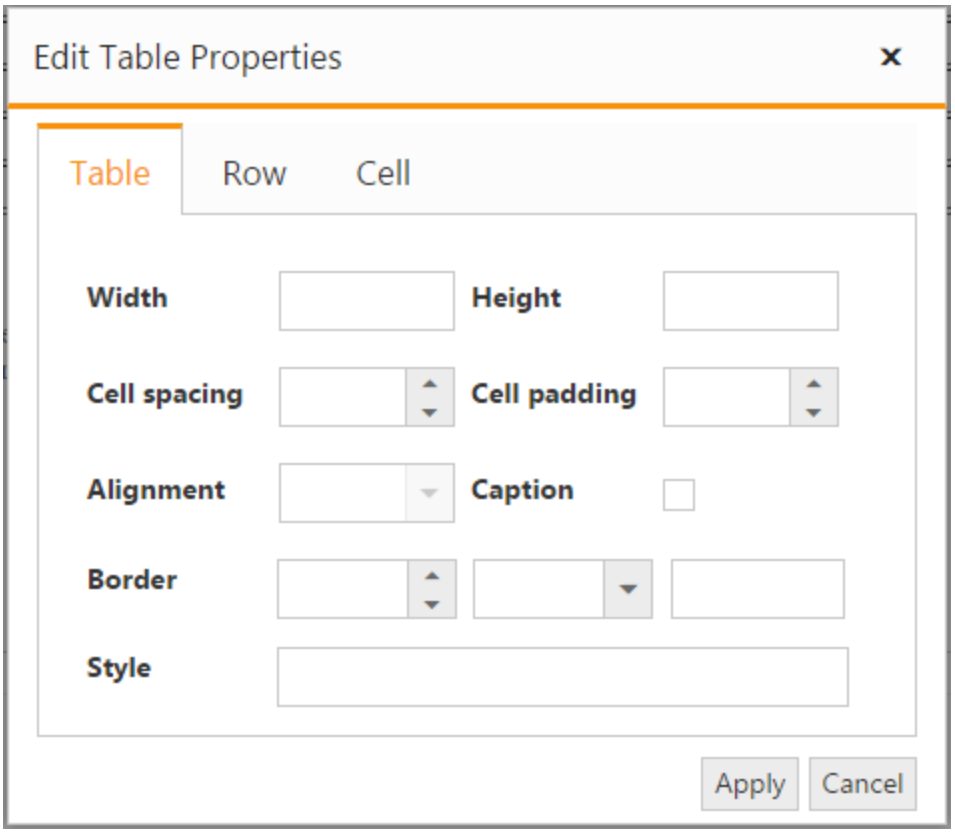
Note: You can also quickly add a new row by clicking on the lower-right cell of the table and pressing the Tab key.

Format a Table

After you create a table, you can format the entire table by using **Table Properties** dialog. In the Table Properties dialog, set or modify each table styles to apply the styles to table elements.

- Table - width, height, style, border, and alignment.
- Cell - border, spacing, and padding.
- Row - height, border, and text align.

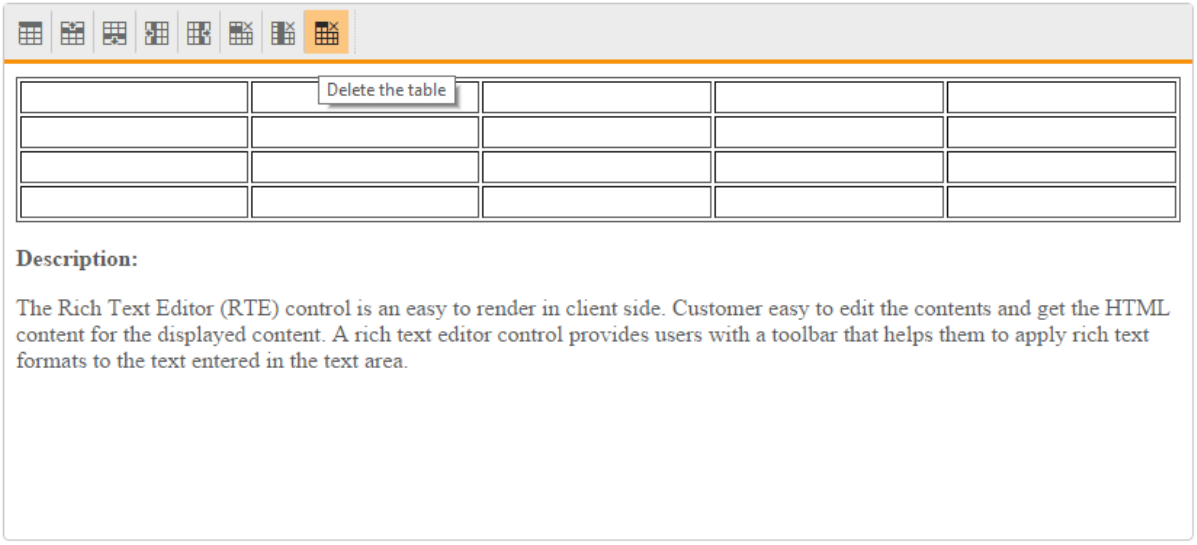
Copyright © 2001 - 2020 Syncfusion Inc.



Note: When you click in a table cell, the **Table Properties** option will be enabled under **Table** tool on the editor's toolbar.

Delete a table

To delete a table from your content, focus on the table and select "Delete a Table" tool from toolbar. It will delete the entire table with all formatting content.



Copyright © 2001 - 2020 Syncfusion Inc.

Note: To delete the table contents alone, select the required content of the table which you want to delete and press Delete key. All the content disappears but the rows and columns remain along with its formatting.

Working with Lists

The editor provides tools to makes your content as list such as an ordered and unordered list.

Create a Lists

By default, Insert Lists tool is enabled in the editor's toolbar.The editor's have ordered and unordered list types.

HTML

```
@{
List<String> toolsList = new List<string>() { "lists" };
List<String> lists = new List<string>() { "unorderedList", "orderedList"
};
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ToolsList(toolsList)
.Tools(tool => tool.Lists(lists))
.Render();}
<br />
```

Custom Lists

You can use custom lists tools to insert lists with custom behaviors.You can create a list with related attributes (such as listImage, listStyle, title, name, and text) using the custom list tool.Ordered and Unordered list having own customize ways to insert a list into the editor's content.

- Insert a customOrderedList
- Insert a customUnorderedList

Insert a customOrderedList

you need to enable customOrderedList tool on the editor's toolbar.

The customOrderedList having below options for an ordered list customization.

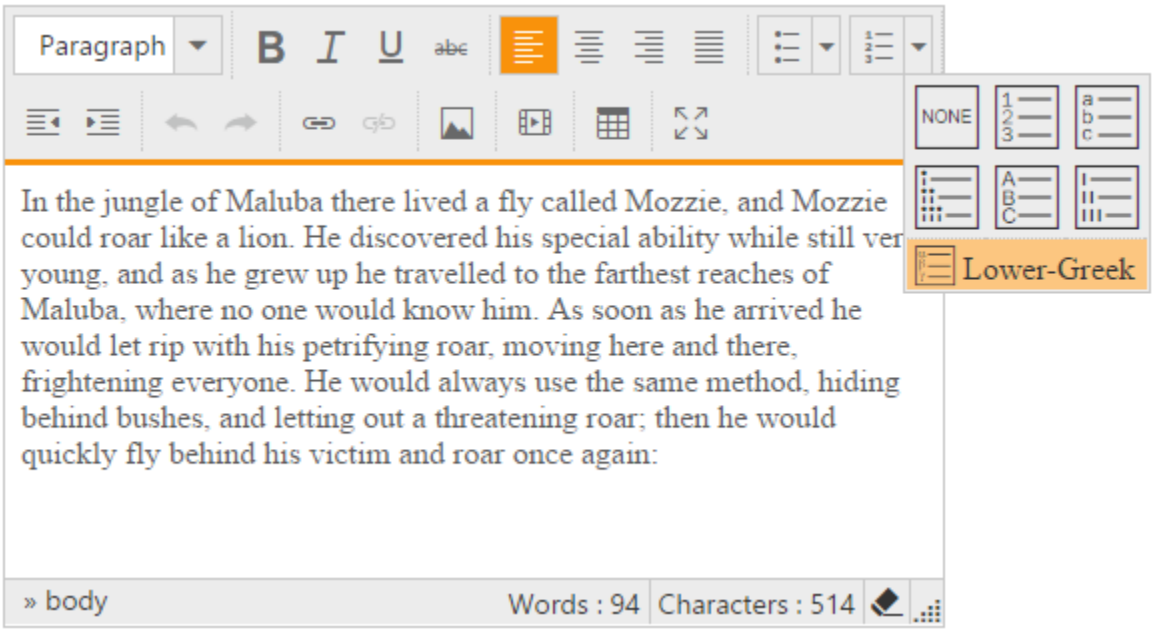
Option	Summary
name	Specifies the name for customOrderedList item.
tooltip	Specifies the title for customOrderedList item.
CSS	Specifies the styles for customOrderedList item.

Copyright © 2001 - 2020 Syncfusion Inc.

text	Specifies the text for customOrderedList item.
listStyle	Specifies the list style for customOrderedList item.
listImage	Specifies the image for customOrderedList item.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.Tools(tool => tool.CustomOrderList(customOrder => customOrder.Css("e-
rte-toolbar-icon e-rte-listitems customOrder").ListStyle("lower-
greek").Name("orderInsert").Tooltip("Custom OrderList").Text("Lower-
Greek").Add()))
.Render();}
<br />
```



Insert a customUnorderedList

you need to enable customUnorderedList tool on the editor's toolbar.

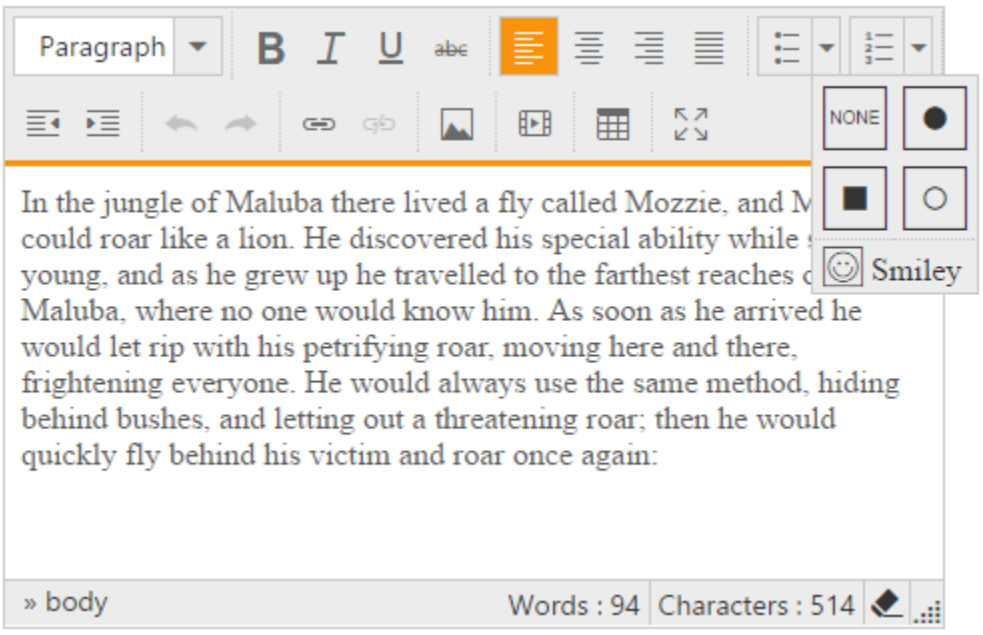
The customUnorderedList having below options for an unordered list customization.

Option	Summary

name	Specifies the name for customUnorderedList item.
tooltip	Specifies the title for customUnorderedList item.
CSS	Specifies the styles for customUnorderedList item.
text	Specifies the text for customUnorderedList item.
listStyle	Specifies the list style for customUnorderedList item.
listImage	Specifies the image for customUnorderedList item.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.Tools(tool => tool.CustomOrderList(customOrder => customOrder.Css("e-
rte-toolbar-icon e-rte-listitems customOrder").ListStyle("lower-
greek").Name("orderInsert").Tooltip("Custom OrderList").Text("Lower-
Greek").Add()))
.Render();}
<br />
```



Clean unwanted elements and styles when copy paste from Microsoft Word

While copy pasting content from MSWord document, the content will be processed in the paste action event.pasteCleanupSettings API can be used for removing unwanted elements.

This will convert an unformatted html element (MOS XML format) content to a proper html element.Table elements also will be converted with proper elements.

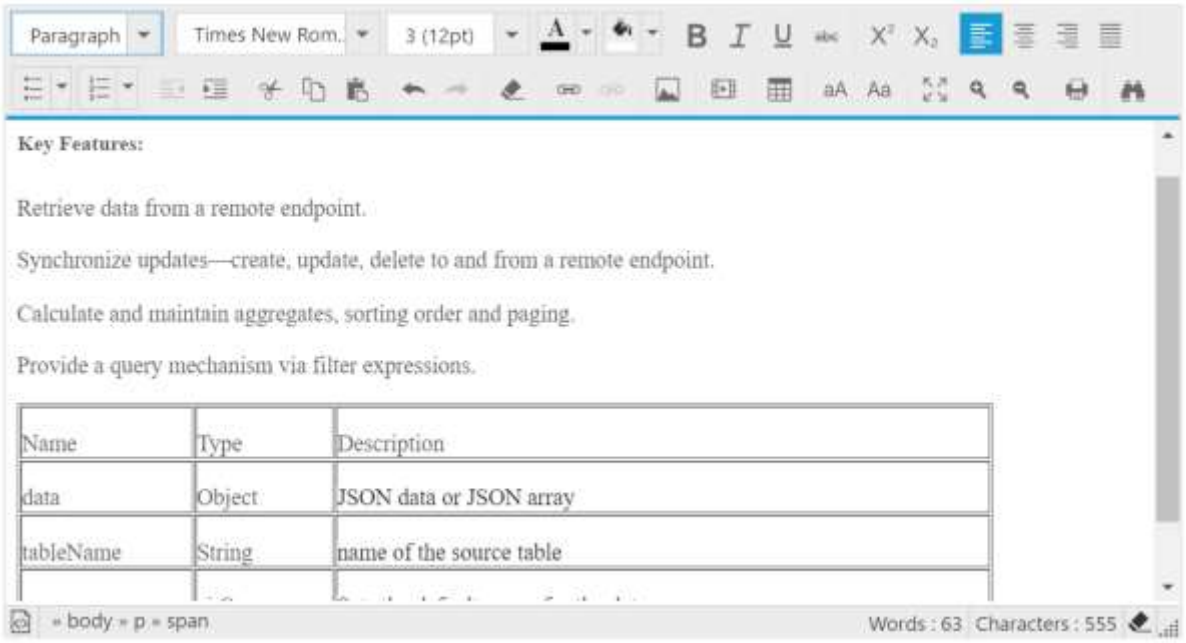
Options	Description
listConversion	List Conversion option convert the list elements into a proper format pasted from MSWord document to editor.
cleanCSS	Clean Css is used to clean the unwanted css in the elements pasted from MSWord document to editor
removeStyles	Remove styles will remove all styles in the elements pasted from MSWord document to editor.
cleanElements	Clean Elements is used to clean the unwanted elements pasted from MSWord document to editor.

HTML

```
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.PasteCleanupSettings(c=>c.CleanCSS(true).CleanElements(true).ListConvers
ion(true).RemoveStyles(true))
.ShowContextMenu(true)
.Render();}
```

Pasted Content before Cleanup:

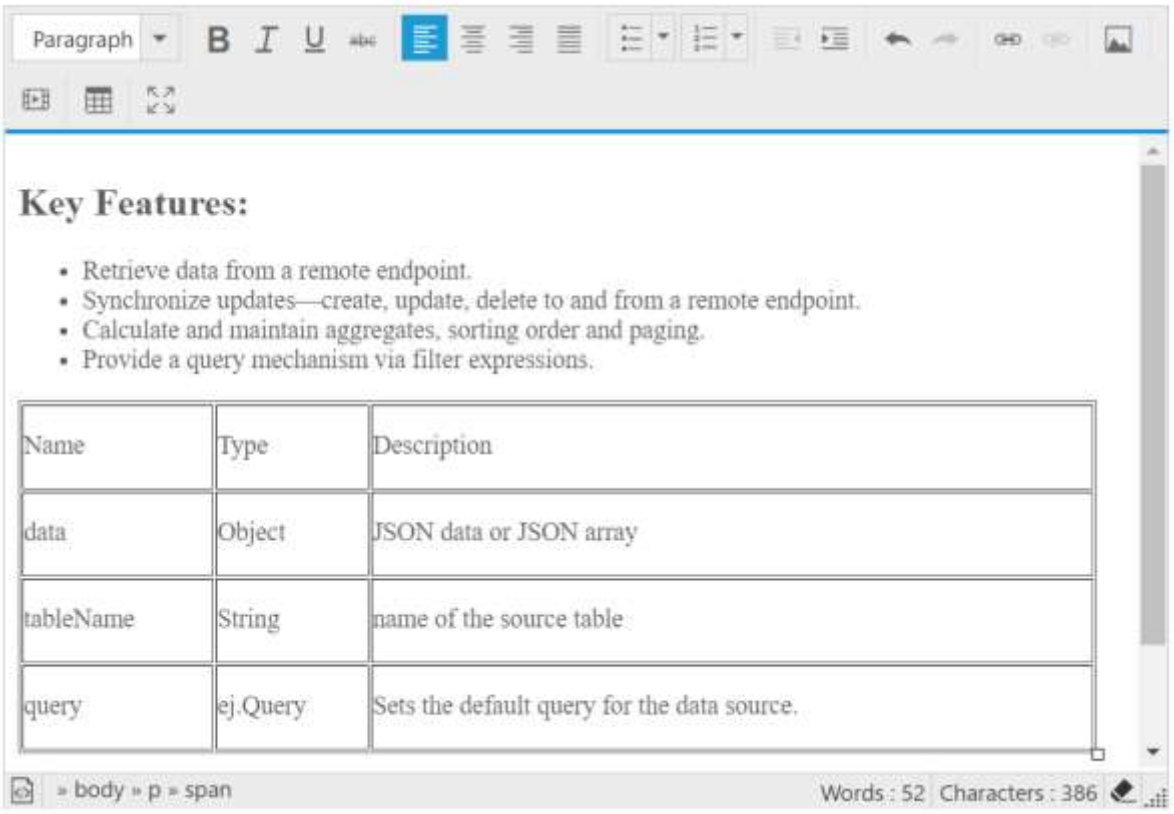
Copyright © 2001 - 2020 Syncfusion Inc.




```
<body spellcheck="false" autocorrect="off" contenteditable="true">
  <h2>…</h2>
  <p class="MsoListParagraphCxSpFirst" style="text-indent:-.25in;mso-list:l0 level1 lfo1">…</p>
  <p class="MsoListParagraphCxSpMiddle" style="text-indent:-.25in;mso-list:l0 level1 lfo1">…</p>
  <p class="MsoListParagraphCxSpMiddle" style="text-indent:-.25in;mso-list:l0 level1 lfo1">…</p>
  <p class="MsoListParagraphCxSpLast" style="text-indent:-.25in;mso-list:l0 level1 lfo1">…</p>
  <table class="MsoTableGrid e-rte-table" border="1" cellpadding="0" cellspacing="1" width="678.6666870117188"
  style="border: 1px solid rgb(92, 92, 92); border-collapse: separate; border-spacing: 2px;">…</table>
    <p></p>
  <p class="MsoNormal">…</p>
  </body>
```

Pasted Content after Cleanup:

Copyright © 2001 - 2020 Syncfusion Inc.




```
<body spellcheck="false" autocorrect="off" contenteditable="true">
  <p>…</p>
    <h2>Key Features:</h2>
  <ul level="1">…</ul>
  <table border="1" cellpadding="0" class="e-rte-table" cellspacing="1" width="672.8240793359375" style="border:
  1px solid rgb(92, 92, 92); border-collapse: separate; border-spacing: 2px;">…</table>
    <p></p>
  <p>…</p>
  <ol>…</ol>
  </body>
```

Zoom

The editor provides zoom tools which enlarges the view of an editor's object enabling you to see more detail. You can continuous zoomIn and zoomOut either using zoom tools or keyboard.

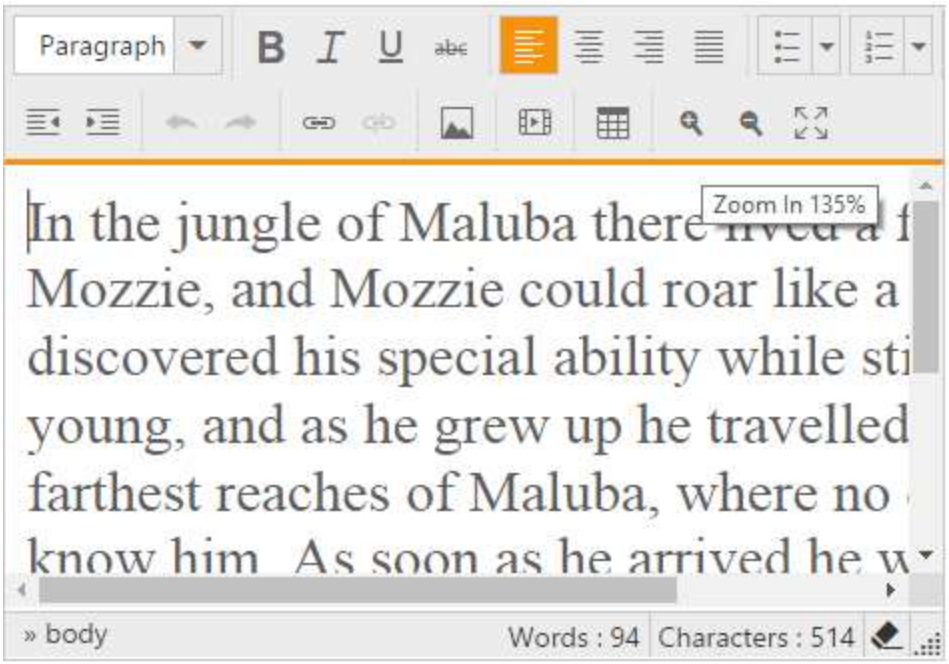
You can assign Increases and decreases of zooming range using zoomStep property

HTML

```
@{
List<String> toolsList = new List<string>() { "view" };
List<String> view = new List<string>() { "zoomIn","zoomOut" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ZoomStep("0.05").ContentTempl
ate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ToolsList(toolsList)
.Tools(tool => tool.View(view))
.Render();}
<br />
```



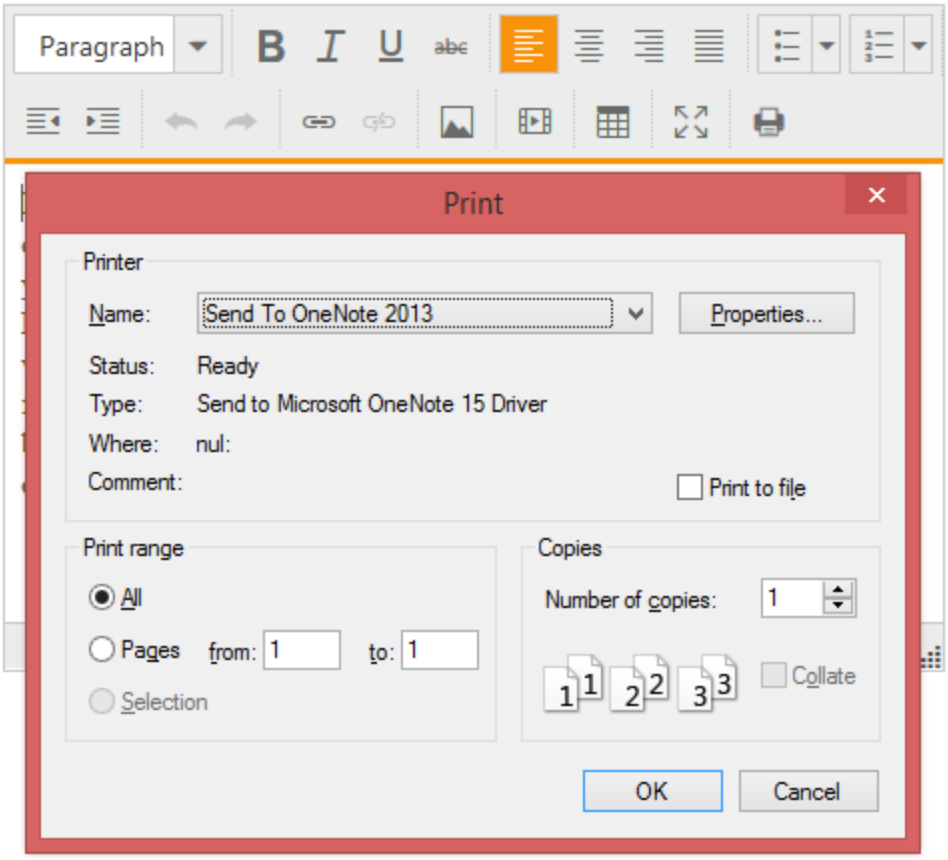
Print

The editor provides print tools which use to print the contents of the editor.

HTML

```
@{
List<String> toolsList = new List<string>() { "print" };
List<String> print = new List<string>() { "print" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ToolsList(toolsList)
.Tools(tool => tool.Print(print))
.Render();}
<br />
```

Copyright © 2001 - 2020 Syncfusion Inc.



Find and Replace

RTE provides find and replace support, which is used to search for a keyword in RTE content and replace the matched keyword with a specified text. In order to use it, we have to enable the find and replace item in the editor toolbar (or) Press CTRL+F key.

HTML

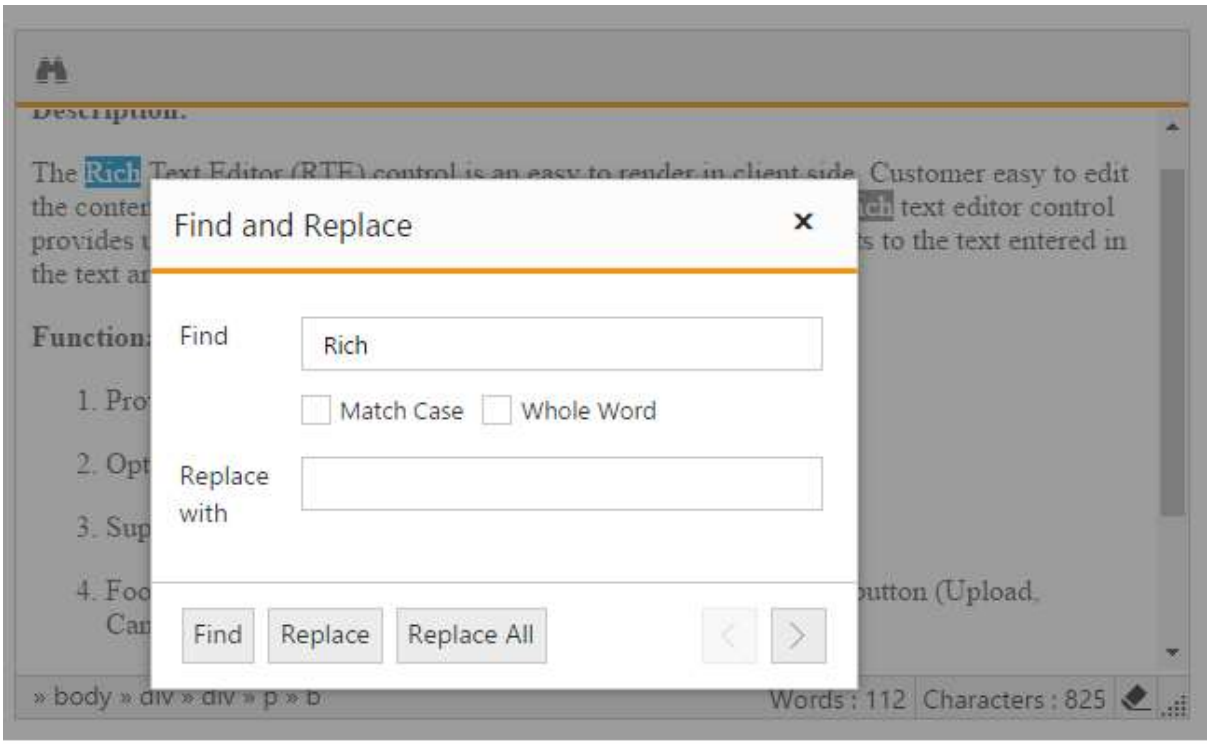
```
@{
List<String> edit = new List<string>() { "edit" };
List<String> editTool = new List<string>() { "findAndReplace" };
}
@{Html.EJ().RTE("rteSample").Width("800px").ContentTemplate(@<div>
The Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area.
</div>)
.ToolsList(edit)
.Tools(tool => tool.Edit(editTool))
```

- Find and Replace action should happen in a following sequence:

Copyright © 2001 - 2020 Syncfusion Inc.

Find => Replace (or) Replace All.

Action	Descriptions
Find	Finds a keyword matches with the editor content.it consist of following filters. *Match Case.* Whole Word.
Replace	Replaces the particular selected match with the specified text.
ReplaceAll	Replaces the entire matches with the specified text.



Note: Before performing the Replace/ReplaceAll action, we must do the find action to validate the match's availability.

Localization

The editor provides option to localize its strings, it is used to adapting the editor to a particular local language. By default, the editor will use the US English (en-US) as its language. Please find the table with list of keys and their corresponding values for default language (en-US).

HTML

```
ej.RTE.Locale["en-US"] = {
bold: "Bold",
italic: "Italic",
underline: "Underline",
strikethrough: "Strikethrough",
superscript: "Superscript",
subscript: "Subscript",
justifyCenter: "Align text center",
justifyLeft: "Align text left",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
justifyRight: "Align text right",
justifyFull: "Justify",
unorderedList: "Insert unordered list",
orderedList: "Insert ordered list",
indent: "Indent",
fileBrowser: "File Browser",
outdent: "Decrease Indent",
cut: "Cut",
copy: "Copy",
paste: "Paste",
paragraph: "Paragraph",
undo: "Undo",
redo: "Redo",
upperCase: "Upper Case",
lowerCase: "Lower Case",
clearAll: "Clear All",
clearFormat: "Clear Format",
createLink: "Insert/Edit hyperlink",
removeLink: "Remove hyperlink",
image: "Insert image",
video: "Insert video",
editTable: "Edit Table Properties",
embedVideo: "Paste your embed code below",
viewHtml: "View HTML",
fontName: "Select font family",
fontSize: "Select font size",
fontColor: "Select color",
format: "Format",
backgroundColor: "Background color",
style: "Styles",
deleteAlert: "Are you sure you want to clear all the contents?",
copyPasteAlert:
"Your browser doesn't support direct access to the clipboard. Please use
the Ctrl+X/C/V keyboard shortcuts instead.",
videoError: "The text area cannot be empty",
imageWebUrl: "Web Address",
imageAltText: "Image description",
dimensions: "Dimensions",
constrainProportions: "Constrain Proportions",
linkWebUrl: "Web Address",
imageLink: "Image as Link",
imageBorder: "Image Border",
imageStyle: "Style",
linkText: "Text",
linkToolTip: "ToolTip",
html5Support: "This tool icon only enabled in HTML5 supported browsers",
linkOpenInNewWindow: "Open link in new window",
tableColumns: "No.of Columns",
tableRows: "No.of Rows",
tableWidth: "Width",
tableHeight: "Height",
tableCellSpacing: "Cell spacing",
tableCellPadding: "Cell padding",
tableBorder: "Border",
tableCaption: "Caption",
tableAlignment: "Alignment",
textAlign: "Text align",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
dialogUpdate: "Update",
dialogInsert: "Insert",
dialogCancel: "Cancel",
dialogApply: "Apply",
dialogOk: "Ok",
createTable: "Create table",
addColumnLeft: "Add column on the left",
addColumnRight: "Add column on the right",
addRowAbove: "Add row above",
addRowBelow: "Add row below",
deleteRow: "Delete the row",
deleteColumn: "Delete the column",
deleteTable: "Delete the table",
customTable: "Create custom table...",
characters: "Characters",
general: "General",
advanced: "Advanced",
table: "Table",
row: "Row",
column: "Column",
cell: "Cell",
solid: "Solid",
dotted: "Dotted",
dashed: "Dashed",
doubled: "Doubled"
};
var format = [
{ text: "Paragraph", value: "p", spriteCssClass: "e-paragraph" },
{ text: "Quotation", value: "blockquote", spriteCssClass: "e-quotation"
},
{ text: "Heading 1", value: "h1", spriteCssClass: "e-h1" },
{ text: "Heading 2", value: "h2", spriteCssClass: "e-h2" },
{ text: "Heading 3", value: "h3", spriteCssClass: "e-h3" },
{ text: "Heading 4", value: "h4", spriteCssClass: "e-h4" },
{ text: "Heading 5", value: "h5", spriteCssClass: "e-h5" },
{ text: "Heading 6", value: "h6", spriteCssClass: "e-h6" }
];
```

Note: The culture name has to be specified in a standard format such as [Language Code]-[County/Region Code].

To localize the editor's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the editor in German language ("de-DE").

HTML

```
ej.RTE.Locale["de-DE"] = {
bold: "fett",
italic: "kursiv",
underline: "unterstreichen",
strikethrough: "Durchgestrichen",
superscript: "Hochgestellt ",
subscript: "Tiefgestellt",
justifyCenter: "Ausrichten von Text Zentrum",
justifyLeft: "Ausrichten von Text links",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
justifyRight: "Ausrichten von Text rechts",
justifyFull: "rechtfertigen",
unorderedList: "Legen Sie ungeordnete Liste",
orderedList: "Geordnete Liste einfügen ",
indent: "Gedankenstrich",
fileBrowser: "Datei-Browser",
outdent: "Einzug verkleinern",
cut: "Cut",
copy: "Kopie",
paste: "Paste",
paragraph: "Absatz",
undo: "rückgängig machen",
redo: "wiederholen",
upperCase: "Großbuchstaben",
lowerCase: "Kleinbuchstaben",
clearAll: "Alles löschen",
clearFormat: "Klar Format",
createLink: "Einfügen / bearbeiten Hyperlink",
removeLink: "Hyperlink entfernen",
image: "Bild einfügen",
video: "Video einfügen",
editTable: " Table Edit Properties",
embedVideo: "Fügen Sie Ihren Code einbetten unten",
viewHtml: "View HTML",
fontName: "Wählen Sie Schriftfamilie",
fontSize: "Schriftgröße wählen",
fontColor: "Farbe wählen",
format: "formatieren",
backgroundColor: "Hintergrundfarbe",
style: "Styles",
deleteAlert: "Sind Sie sicher, dass Sie den gesamten Inhalt löschen?",
copyPasteAlert: "Ihr Browser unterstützt keine direkten Zugriff auf die
Zwischenablage. Bitte verwenden Sie stattdessen die Strg + X / C / V
Tastenkombinationen",
videoError: "Der Textbereich kann nicht leer sein",
imageWebUrl: "Webadresse",
imageAltText: "Bildbeschreibung",
dimensions: "Abmessungen",
constrainProportions: "Proportionen beschränken",
linkWebUrl: "Webadresse",
imageLink: "Bild als Link-",
imageBorder: "Bild Rand",
imageStyle: "Stil",
linkText: "Text",
linkToolTip: "Tooltip",
html5Support: "Dieses Werkzeug-Symbol in HTML5 ist nur aktiviert,
unterstützten Browser",
linkOpenInNewWindow: "Link in neuem Fenster öffnen",
tableColumns: "Keine der Spalten",
tableRows: "Keine der Zeilen",
tableWidth: "Weite",
tableHeight: "Höhe",
tableCellSpacing: "Zellenabstand",
tableCellPadding: "Zellauffüllung",
tableBorder: "Border",
tableCaption: "Bildunterschrift",
tableAlignment: "Ausrichtung",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
textAlign: "Text align",
dialogUpdate: "aktualisieren",
dialogInsert: "Insert",
dialogCancel: "Absagen",
dialogApply: "Bewerben",
dialogOk: "Ok",
createTable: "create table",
addColumnLeft: "Spalte hinzufügen an der linken",
addColumnRight: "Spalte hinzufügen am rechten",
addRowAbove: "In Zeile über",
addRowBelow: "Zeile hinzufügen unter",
deleteRow: "Löschen Sie die Zeile",
deleteColumn: "Löschen Sie die Spalte",
deleteTable: "Löschen Sie die Tabelle",
customTable: "Erstellen Sie benutzerdefinierte Tabelle ...",
characters: "Charaktere",
general: "Allgemein",
advanced: "Erweiterte",
table: "Tabelle",
row: "Zeile",
column: "Spalte",
cell: "Zelle",
solid: "solide",
dotted: "gepunktete",
dashed: "Eine gestrichelte",
doubled: "verdoppelt",
maximize: "maximieren",
resize: "minimieren",
swatches: "Farbfelder"
};
```

You can set the Locale property of the editor to the new language.

HTML

```
@{
Syncfusion.JavaScript.Models.RTEproperties rte = new
Syncfusion.JavaScript.Models.RTEproperties();
List<Syncfusion.JavaScript.Models.Format> list = new
List<Syncfusion.JavaScript.Models.Format>{
new Syncfusion.JavaScript.Models.Format { Text= "Absatz", Value= "p",
SpriteCssClass= "e-paragraph" },
new Syncfusion.JavaScript.Models.Format { Text= "Zitat", Value=
"blockquote", SpriteCssClass= "e-quotation" },
new Syncfusion.JavaScript.Models.Format {  Text= "Kopf 1", Value= "h1",
SpriteCssClass= "e-h1"} ,
new Syncfusion.JavaScript.Models.Format { Text= "Kopf 2", Value= "h2",
SpriteCssClass= "e-h2" },
new Syncfusion.JavaScript.Models.Format { Text="Kopf 3", Value="h3",
SpriteCssClass= "e-h3" },
new Syncfusion.JavaScript.Models.Format { Text= "Kopf 4", Value= "h4",
SpriteCssClass= "e-h4" },
new Syncfusion.JavaScript.Models.Format { Text= "Kopf 5", Value= "h5",
SpriteCssClass= "e-h5" },
new Syncfusion.JavaScript.Models.Format { Text= "Kopf 6", Value= "h6",
SpriteCssClass= "e-h6" }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
};
rte.Format = list;
rte.Value =
"Das RichTextEditor (RTE) Steuerung ermöglicht Ihnen, den Inhalt mit Eins
atz Tisch und Bilder zu bearbeiten ," +
" sondern bietet auch eine Werkzeugleiste , die Rich-Text- Format ," +
" um die in der Textarea eingegeben Gehalt gelten können.";
rte.Locale = "de-DE";
}
@{Html.EJ().RTE("rteSample", rte)
.Render();}.
```

Keyboard Support

The editor has full keyboard accessibility that includes shortcuts to open and to do other actions with toolbar items, drop-down lists, and dialogs.

Press	To do this
Arrow keys	When a toolbar is focused Move between options in an open drop-down list
Enter	Execute the selected button, drop-down item
ESC	Cancel an action or close the dialog
HOME END	Go to first/last of the line
CTRL + HOME CTRL + END	Go to the beginning/end of a content
PAGE UP PAGE DOWN	Scroll up/down page in the content
CTRL + A	Select all content
CTRL + C	Copy the selected text or image
CTRL + X	Cut the selected text
CTRL + V	Paste text or image
CTRL + Z	Undo the last action
CTRL + Y	Redo the last action

Copyright © 2001 - 2020 Syncfusion Inc.

CTRL + U	Make text underline
CTRL + I	Make text italic
CTRL + B	Make text bold

To disable the keyboard navigation, set the AllowKeyboardNavigation property of the editor to false (its default value is true). It will disable all the keyboard navigation shortcuts except for the UP/DOWN keys and PAGE UP/PAGE DOWN keys.

HTML

```
@{Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<p>
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content.
A rich text editor control provides users with a toolbar that helps them
to apply rich text formats to the text entered in the text area.
</p>)
.AllowKeyboardNavigation(false)
.Render();}
```

Strongly typed HTML helper

The RTE control supports strongly typed HTML helpers which use lambda expressions to reference models or view models passed to a view template. These helpers allow you to define the value of the RichTextEditorFor from the model. The following steps explain how to use the strongly typed helpers to create RichTextEditorFor.

Add a class named "RTE" in the Models folder and replace the code with the following code:

HTML

```
public class RTE
{
public string value { get; set; }
}
```

In the controller, pass the model to the View.

HTML

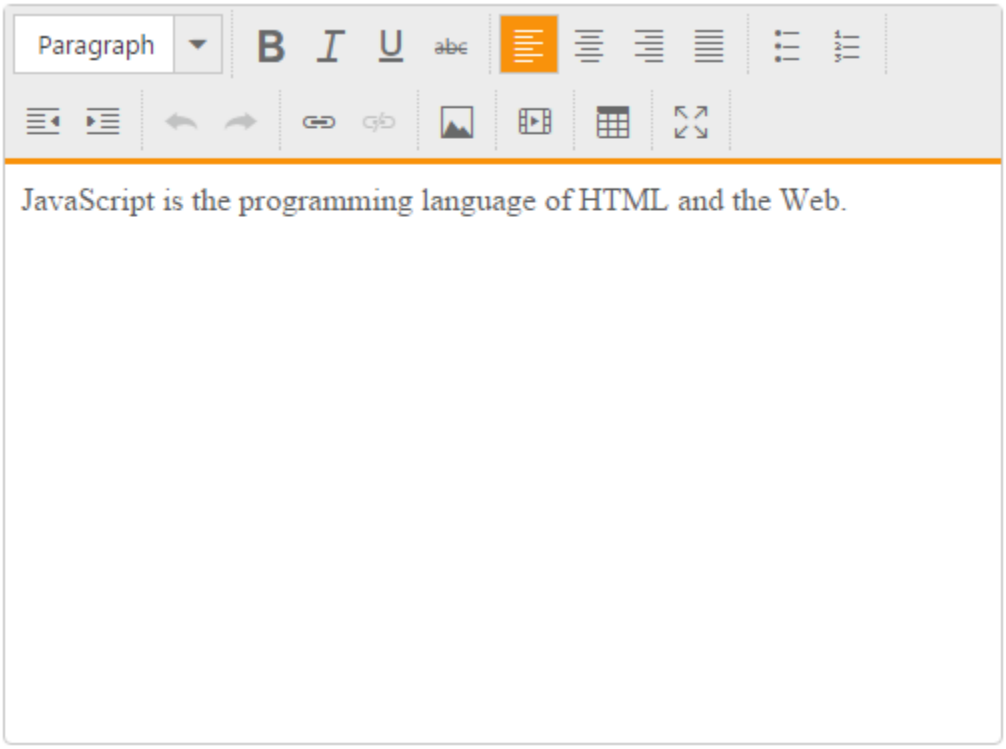
```
using UserInterface.Models;
public ActionResult RichTextEditorFeatures()
{
RTE obj = new RTE ();
obj.value = "JavaScript is the programming language of HTML and the
Web.";
return View(obj);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

In View, invoke the strongly typed RichTextEditor helper with the lambda expression to set the default value.

HTML

```
@model UserInterface.Models.RTE
@using (Html.BeginForm())
{
@Html.EJ().RTEFor(model => model.value )
}
```



XHTML Validation

The editor provides option to validate its content through the EnableXHTML property. When you set or modify the content into the editor, it continuously checks whether the HTML source of the content that you are creating is valid. The editor examines the HTML markup and then removes the elements or attributes that are not valid.

RAZOR

```
@{
Html.EJ().RTE("rteSample").Width("820px").ContentTemplate(@<div>
The RichTextEditor (RTE) control enables you to edit the contents with in
sert table and images
It also provides a toolbar that helps to apply rich text formats to the c
ontent entered in the TextArea
</div>)
.EnableXHTML(true)
.Render();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The editor checks the following settings on validation:

- **Attributes**
- Must be specified in lowercase
- Proper use of quotation marks around the attributes
- Must be valid attributes for corresponding HTML element
- All the required attributes must be included in the HTML element
- Accepts the allowed values alone such as true or false.
- **HTML Elements**
- Must be in lowercase
- All opening tags must be closed
- Allows only the valid HTML elements

Import

Import feature provides support to import a word document into the editor `textarea`. To enable import option in the RTE tool bar, `import` toolbar items needs to be added in RTE toolbar toolsList using `importExport` which adds the tool in the toolbar. In `ImportSettings` Url option, the server page for import is needed to be mapped. When you click the toolbar import icon, it opens a dialog to browse the select a word file. The selected word file will be imported into the editor `textarea`.

RAZOR

```
@{
List<String> toolsList = new List<string>() { "importExport" };
List<String> ImportExport = new List<string>() { "import" };
}
@{
Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<div>
&#60;p&#62;The Rich Text Editor (RTE) control is an easy to render in
client side. Customer easy to edit the contents and get the HTML content
for
the displayed content. A rich text editor control provides users with a
toolbar
that helps them to apply rich text formats to the text entered in the
text
area. &#60;/p&#62;
</div>).MinWidth("200px").ToolsList(toolsList).Tools(tool =>
tool.ImportExport(ImportExport))
.ImportSettings(importSettings => importSettings.Url("Import")).Render();
}
```

C#

```
public partial class RTEController : Controller
{
//
// GET: /ImportExport/
public ActionResult ImportExport()
{
return View();
}
[HttpPost]
public string Import()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
string HtmlString = string.Empty;
if (HttpContext.Request.Files.AllKeys.Any())
{
string RTEID = HttpContext.Request.QueryString["rteid"];
var fileName = RTEID + "_importUpload";
var httpPostedFile = HttpContext.Request.Files[fileName];
if (httpPostedFile != null)
{
using (var mStream = new MemoryStream())
{
new WordDocument(httpPostedFile.InputStream).Save(mStream,
FormatType.Html);
mStream.Position = 0;
HtmlString = new StreamReader(mStream).ReadToEnd();
};
HtmlString = ExtractBodyContent(HtmlString);
foreach (var item in DecodeKeys())
{
HtmlString = HtmlString.Replace(item.Key, item.Value);
}
}
else HttpContext.Response.Write("Select any file to upload.");
}
return HtmlString;
}
public IDictionary<string, string> DecodeKeys()
{
IDictionary<string, string> KeyValuePair = new Dictionary<string,
string>()
{
{"\"", "'"},{"\r", " "},{"\n", "<br/> "},{"\r\n", " "},{"( )+", "
"},{" ", " "},{"&#8226;", "*"},{"&#8249;", "<"},
{"&#8250;", ">"},{"&#8482;", "(tm)"},{"&#169;", "(c)"},{"&#174;", "(r)"}
};
return KeyValuePair;
}
public string ExtractBodyContent(string html)
{
if (html.Contains("<html") && html.Contains("<body"))
{
return html.Remove(0, html.IndexOf("<body>") +
6).Replace("</body></html>", "");
}
return html;
}
}
```

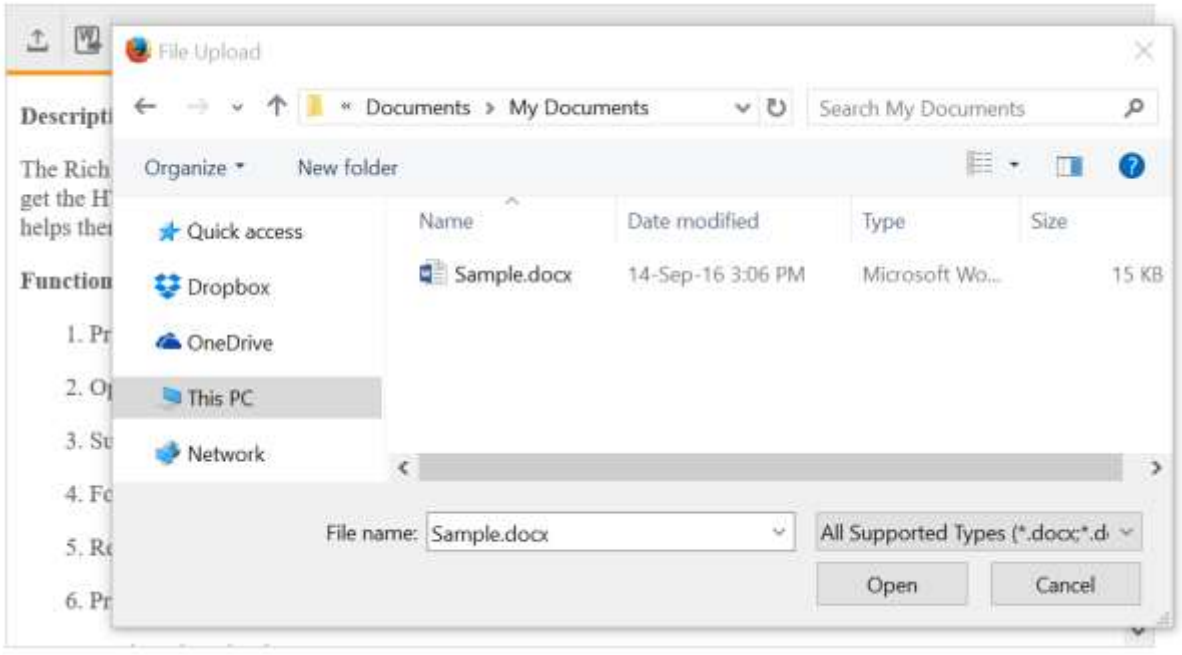
Import Server Dependencies

Full list of assemblies needed for RTE Import are as follows

1. Syncfusion.EJ
2. Syncfusion.EJ.Export
3. Syncfusion.Linq.Base

Copyright © 2001 - 2020 Syncfusion Inc.

4. Syncfusion.Compression.Base
5. Syncfusion.DocIO.Base



Export

Export feature provides support to export editor textarea content into word and PDF files. To enable Export option in the RTE tool bar, wordExport , pdfExport toolbar items needs to be added in RTE toolbar toolsList using importExport which adds the tool in the toolbar. ExportToWordSettings consists of Url and FileName sub properties. In Url property, the server page for export to word is needed to be mapped and In FileName property, the name for the exported word file is given. ExportToPdfSettings consists of Url and FileName sub properties. In Url property, the server page for export to PDF is needed to be mapped and In FileName property, the name for the exported PDF file is given. When you click the toolbar pdfExport or wordExport icon, the contents of RTE are sent to the server. It performs XHTML Validation on the editor textarea content on the server. Once the XHTML validation and formatting is successful, it exports the content into a Word or PDF File.

RAZOR

```
@{
List<String> toolsList = new List<string>() { "importExport" };
List<String> ImportExport = new List<string>() { "wordExport","pdfExport"
};
}
@{
Html.EJ().RTE("rteSample").Width("100%").ContentTemplate(@<div>
&#60;p&#62;The Rich Text Editor (RTE) control is an easy to render in
client side. Customer easy to edit the contents and get the HTML content
for
the displayed content. A rich text editor control provides users with a
toolbar
that helps them to apply rich text formats to the text entered in the
text
```

```
area. &#60;/p&#62;
</div>).MinWidth("200px").ToolsList(toolsList).Tools(tool =>
tool.ImportExport(ImportExport))
.ExportToWordSettings(exportToWordSettings =>
exportToWordSettings.Url("ExportToWord").FileName("WordExport"))
.ExportToPdfSettings(exportToPdfSettings =>
exportToPdfSettings.Url("ExportToPDF").FileName("PdfExport")).Render();
}
```

C#

```
public partial class RTEController : Controller
{
//
// GET: /ImportExport/
public ActionResult ImportExport()
{
return View();
}
//Export to Word Document
[HttpPost]
[ValidateInput(false)]
public void ExportToWord()
{
string RTEID = HttpContext.Request.QueryString["rteid"];
string FileName = HttpContext.Request.Params[RTEID + "_inputFile"];
string htmlText = HttpContext.Request.Params[RTEID + "_inputVal"];
WordDocument document = GetDocument(htmlText);
document.Save(FileName + ".docx", FormatType.Docx,
System.Web.HttpContext.Current.Response,
HttpContentDisposition.Attachment);
}
//Export to PDF Document
[HttpPost]
[ValidateInput(false)]
public void ExportToPDF()
{
string RTEID = HttpContext.Request.QueryString["rteid"];
string FileName = HttpContext.Request.Params[RTEID + "_inputFile"];
string htmlText = HttpContext.Request.Params[RTEID + "_inputVal"];
WordDocument document = GetDocument(htmlText);
DocToPDFConverter converter = new DocToPDFConverter();
PdfDocument pdfDocument = converter.ConvertToPDF(document);
pdfDocument.Save(FileName + ".pdf",
System.Web.HttpContext.Current.Response, HttpReadType.Save);
}
public WordDocument GetDocument(string htmlText)
{
WordDocument document = null;
MemoryStream stream = new MemoryStream();
StreamWriter writer = new StreamWriter(stream,
System.Text.Encoding.Default);
htmlText = htmlText.Replace("\"", "'");
XmlConversion XmlText = new XmlConversion(htmlText);
XhtmlConversion XhtmlText = new XhtmlConversion(XmlText);
writer.Write(XhtmlText.ToString());
```

Copyright © 2001 - 2020 Syncfusion Inc.

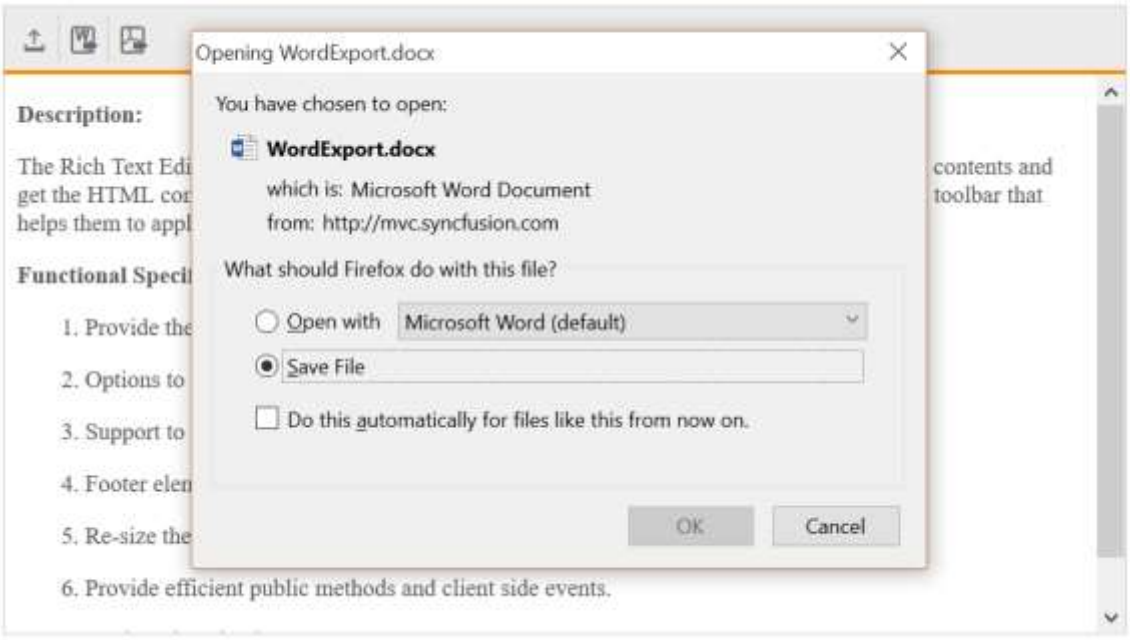
```
writer.Flush();
stream.Position = 0;
document = new WordDocument(stream, FormatType.Html,
XHTMLValidationType.None);
return document;
}
}
```

Export Server Dependencies

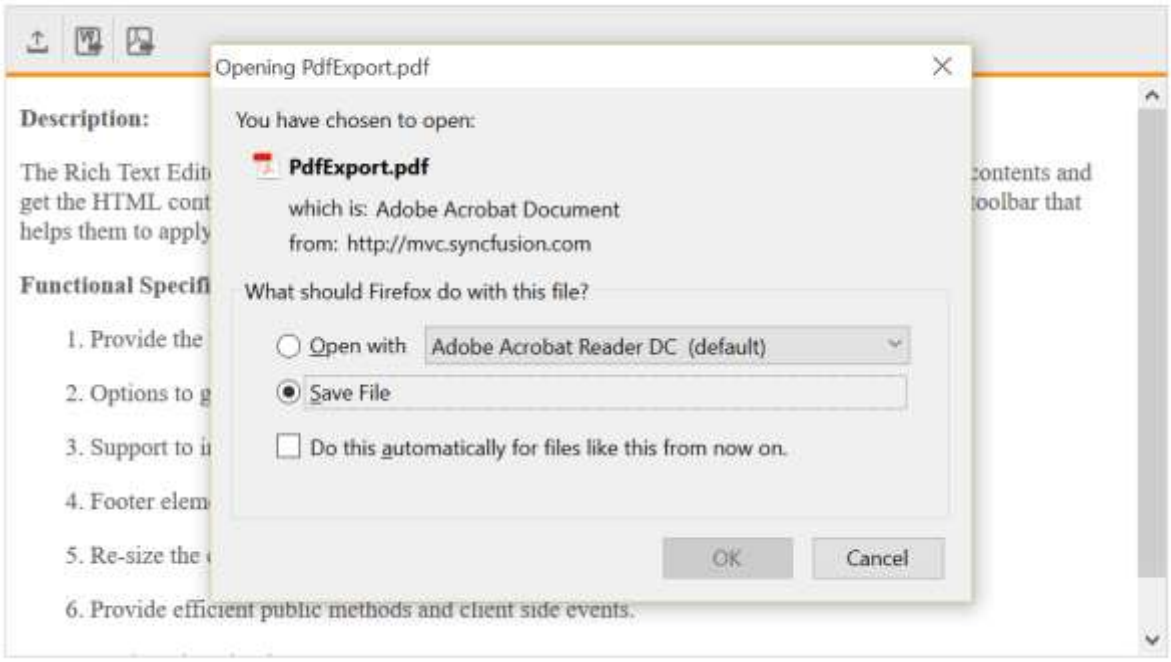
Export Helper functions are available in the Assembly Syncfusion.EJ.Export, which is available in the Essential Studio builds. Full list of assemblies needed for RTE Export as follows

1. Syncfusion.EJ
2. Syncfusion.EJ.Export
3. Syncfusion.Linq.Base
4. Syncfusion.Compression.Base
5. Syncfusion.DocIO.Base
6. Syncfusion.PDF.Base

Word Export



PDF Export



Validation

It's easy to validate the RichTextEditor's value on form submission by applying ValidationRules and ValidationMessage to the RichTextEditor.

Note: jquery.validate.min script file should be referred for validation, for more details, refer here.

jQuery Validation Methods

The following are jQuery validation methods.

List of jQuery validation methods

Rules	Description
required	Requires value for the RichTextEditor control.
minWordCount	Requires the value to be of given minimum words count.
minlength	Requires the value to be of given minimum characters count.
maxlength	Requires the value to be of given maximum characters count.

Validation Rules

The validation rules help you to verify the content by adding validation attributes to the text area. This can be set by using ValidationRules property.

Validation Messages

You can set your own custom error message by using ValidationMessage property. To display the error message, specify the corresponding annotation attribute followed by the message to display.

Copyright © 2001 - 2020 Syncfusion Inc.

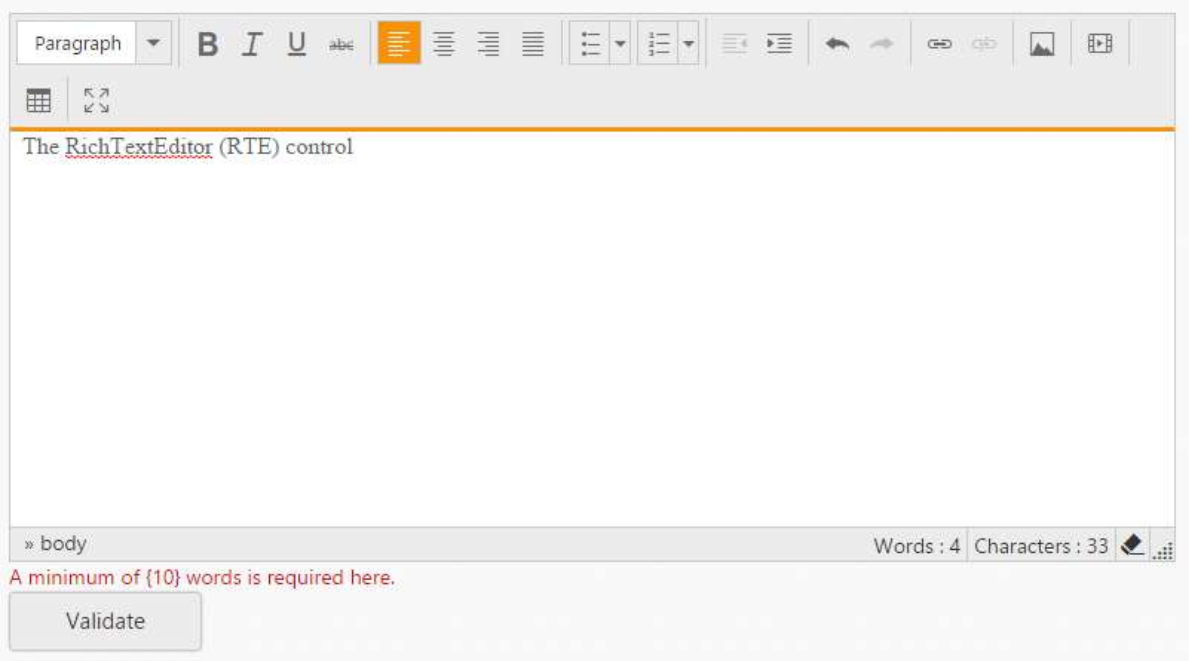
Note: jQuery predefined error messages to that annotation attribute will be shown when this property is not defined.

When you initialize the RichTextEditor widget, it creates a text area hidden element which is used to store the value. Hence, the validation is performed based on the value stored in this hidden element.

Required field and minWordCount values validation is demonstrated in the below given example.

HTML

```
@using (Html.BeginForm())
{
<br />
@Html.EJ().RTE("RTE1").ValidationRules(new Dictionary<string, object> { { "required", "true" }, { "minWordCount", 15 } }).ValidationMessage(new Dictionary<string, object> { { "minWordCount", "A minimum of {10} words is required here" }, { "Required", "Please enter the content" } })
<br />
@Html.EJ().Button("Btn1").Text("Validate")
}
```



Client Side Validation

In the Client Side Validation you can provide a better user experience by responding quickly at the browser level. When you perform a Client Side Validation, all the user inputs validated in the user's browser itself. Client Side validation does not require a round trip to the server, so the network traffic which will help your server perform better.

ASP.NET MVC supports client side validation. First of all you need to take a reference of two JavaScript files from Scripts folder, jquery.validate.min.js and jquery.validate.unobtrusive.min.js in your layout file as shown below.

HTML

```
<script src="~/Scripts/jquery.validate.min.js"></script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

Now, add following two settings in <appSettings> section of web.config, if it is not there.

HTML

```
<appSettings>
<add key="ClientValidationEnabled" value="true" />
<add key="UnobtrusiveJavaScriptEnabled" value="true" />
</appSettings>
```

1. DataAnnotations provides a built-in set of validation attributes that you can apply declaratively to any class or property. Create RTE class to take advantage of the built-in Required validation attributes.

HTML

```
using System.ComponentModel.DataAnnotations;
public class RTE
{
[Required(ErrorMessage = "RTE value is Required")]
public string value { get; set; }
}
```

2. After that you need to create the controller's action methods. These render views on the UI and bind a model with the view. So let's create a controller as follows.

HTML

```
public ActionResult RichTextEditorFeatures()
{
controlInitialization();
return View();
}
[HttpPost]
public ActionResult RichTextEditorFeatures(RTE model)
{
if (ModelState.IsValid)
{
//perform some action
return RedirectToAction("Home");
}
controlInitialization();
return View(model);
}
public void controlInitialization()
{
RTEproperties property = new RTEproperties();
List<String> toolsList = new List<string>() { "alignment" };
List<String> alignment = new List<string>() { "justifyLeft",
"justifyCenter" };
property.ToolsList = toolsList;
RTEtools tools = new RTEtools();
tools.Alignment = alignment;
ViewData["edit"] = property;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

In our example, the form is not posted to the server when there are validation errors detected on the client side.

3. Below is the RichTextEditorFeatures.cshtml view and It's used by the action methods shown above both to display the initial form and to redisplay it in the event of an error.

HTML

```
@using (Html.BeginForm())
{
@Html.ValidationSummary(true)
@Html.EJ().RTEFor(model => model.value,
(Syncfusion.JavaScript.Models.RTEproperties)ViewData["edit"])
@Html.ValidationMessageFor(model => model.value)
<br />
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Post").Type(ButtonT
ype.Submit)
}
```

Server side Validation

In the Server Side Validation, the input submitted by the user is being sent to the server and validated using one of server side scripting languages. After the validation process on the Server Side, the feedback is sent back to the client.

ASP.NET MVC uses Data Annotations attributes to implement validations. Data Annotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET MVC Framework will automatically enforce these validation rules and display validation messages in the view.

1. First of all, apply DataAnnotation attribute on the properties of RTE model class. Here, we have validate the value of the RichTextEditor should not be empty.

HTML

```
using System.ComponentModel.DataAnnotations;
public class RTE
{
[Required(ErrorMessage = "RTE value is Required")]
public string value { get; set; }
}
```

2. Create an action method in the controller that returns a view with a model after the post request.

HTML

```
public ActionResult RichTextEditorFeatures()
{
controlInitialization();
return View();
}
[HttpPost]
public ActionResult RichTextEditorFeatures(RTE model)
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (ModelState.IsValid)
{
//perform some action
return RedirectToAction("Home");
}
controlInitialization();
return View(model);
}
public void controlInitialization()
{
RTEproperties property = new RTEproperties();
List<String> toolsList = new List<string>() { "alignment" };
List<String> alignment = new List<string>() { "justifyLeft",
"justifyCenter" };
property.ToolsList = toolsList;
RTEtools tools = new RTEtools();
tools.Alignment = alignment;
ViewData["edit"] = property;
}
```

As you can see in the POST Edit method, we first check if the ModelState is valid or not. If ModelState is valid then perform some action, if not then return RichTextEditorFeatures view again with the same RTE data.

ModelState.IsValid determines that whether submitted values satisfy all the DataAnnotation validation attributes applied to model properties.

3. After that, created a view for RichTextEditor.

HTML

```
@model ClientValidation.Models.RTE
@{
ViewBag.Title = "RTE";
Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>RTE Features:</h2>
<br/>
@using (Html.BeginForm())
{
@Html.EJ().RTEFor(model => model.value,
(Syncfusion.JavaScript.Models.RTEproperties)ViewData["edit"])
@Html.ValidationMessageFor(model => model.value)
<br />
@Html.EJ().Button("btn").Size(ButtonSize.Small).Text("Post").Type(ButtonT
ype.Submit)
}
```

As you can see in the above RichTextEditorFeatures.cshtml, it calls HTML Helper method ValidationMessageFor for every field. ValidationMessageFor is responsible to display error message for the specified field.

So now, it will display default validation message when you submit the form without entering a value for the RichTextEditor.

Copyright © 2001 - 2020 Syncfusion Inc.



How to

Render the RichTextEditor in Partial View

While rendering a control in the partial view, we need to specify @Html.EJ().ScriptManager() in the partial view page after the initializing the control.

Create a Partial View and define as follows

HTML

```
@Html.EJ().RTE("RTE1").Value("Model Binding allows you to map and bind
the HTTP request data with a model. If you want to work with the form
data, Model Binding makes it easier because the requested data is
submitted automatically into a data model that we
specify.").Width("650px")
@Html.EJ().ScriptManager()
```

Render the partial view into parental view.

HTML

```
@Html.Partial("~/Views/Temp.cshtml");
```

Note: When unobtrusive is set to true in the application, then the script manager can be excluded, as the control is initialized using HTML5 attributes.

If suppose, you require the Unobtrusive to set to true in your application, then you need to include the ej.unobtrusive.min.js file as a reference to your application and set the value of UnobtrusiveJavaScriptEnabled to true in Root directory web.config file.

Copyright © 2001 - 2020 Syncfusion Inc.

Import word document to the RichTextEditor and Export to word document/pdf

Using customTools property in the RichTextEditor to specify which tools to be displayed in the toolbars or create your own command elements, be it Buttons or DropDownList.

In controller, defines the property for the RichTextEditor as well as define your custom tools as follows as.

HTML

```
public ActionResult RichTextEditorFeatures()
{
Syncfusion.JavaScript.Models.RTEproperties property = new
Syncfusion.JavaScript.Models.RTEproperties();
List<Syncfusion.JavaScript.Models.CustomTools> customTool = new
List<Syncfusion.JavaScript.Models.CustomTools>();
customTool.Add(new Syncfusion.JavaScript.Models.CustomTools { Name =
"Import", Text = "Import", Css = "codeCheck", Action = "OnCheck" });
customTool.Add(new Syncfusion.JavaScript.Models.CustomTools { Name =
"Export", Text = "Export", Css = "codeCheck", Action = "OnCheck" });
customTool.Add(new Syncfusion.JavaScript.Models.CustomTools { Name =
"DropType", Text = "DropType", Css = "codeCheck", Action = "OnCheck" });
Syncfusion.JavaScript.Models.RTEtools tools = new
Syncfusion.JavaScript.Models.RTEtools();
tools.CustomTools = customTool;
property.PreRender = "OnRender";
List<String> toolsList = new List<string>() { "customTools" };
property.ToolsList = toolsList;
property.Tools = tools;
property.Value = "Please load the file";
property.ShowFooter = true;
property.EnableXHTML = true;
ViewBag.property = property;
return View();
}
```

Add the following code snippet to the corresponding view page to render the RichTextEditor.

HTML

```
@using (Html.BeginForm())
{
@Html.EJ().RTE("rte2",
(Syncfusion.JavaScript.Models.RTEproperties)ViewBag.property)
}
<div id="TargetList">
<ul>
<li>Word Document</li>
<li>PDF</li>
</ul>
</div>
```

In RichTextEditor’s preRender event, define your custom Tools as follows.

HTML

```
function OnRender(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var importDocs = $("#Import");
$("#Import").find("div").remove();
var target = $("<div id='importFiles'></div>");
importDocs.append(target);
var exportDocs = $("#Export");
$("#Export").find("div").remove();
var export = $("<button id='exportFiles'></button>");
exportDocs.append(export);
var drop = $("#DropType");
$("#DropType").find("div").remove();
var dropTarget = $("<input id='selectType'></div>");
drop.append(dropTarget);
$("#importFiles").ejUploadbox({
saveUrl: "/saveFiles.ashx",
removeUrl: "/removeFiles.ashx",
extensionsAllow: ".doc, .docx, .pdf",
autoUpload: true,
showFileDetails: false,
dialogAction: { content: "body" },
complete: "onComplete"
});
$("#exportFiles").ejButton({
size: "small",
text: "Export",
type : "submit",
});
$('#selectType').ejDropDownList({
watermarkText: 'Select Type',
targetID: "TargetList",
width: "150px",
selectedIndex: 0
});
}
```

HTML

```
function onComplete(args) {
var fileName = args.files.name;
var extension = args.files.extension;
$.ajax({
type: "POST",
contentType: 'application/json; charset=utf-8',
url: '@(Url.Action("Extract"))',
data: JSON.stringify({ name : fileName, ext : extension }),
dataType: "html",
success: function (result) {
if (result != null) {
$("#rte2").ejRTE({ value: result });
}
},
error: function (jqXHR, textStatus, errorThrown) {
alert("Request failed: " + jqXHR.responseStart + "-" + textStatus + "-" +
errorThrown);
}
});
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

In AJAX post back, Retrieve the content from the uploaded files and set the extracted text to the RichTextEditor.

HTML

```
public string Extract(string name, string ext)
{
string value = " ";
if (ext == ".pdf")
{
String path = HttpContext.Server.MapPath("~/uploadfiles/" + name);
Stream sfile1 = new FileStream(path, FileMode.Open, FileAccess.Read,
FileShare.Read);
PdfLoadedDocument loadDocument = new PdfLoadedDocument(sfile1);
// Loading Page collections
PdfLoadedPageCollection loadedPages = loadDocument.Pages;
// Extract text from PDF document pages
foreach (PdfLoadedPage loadPage in loadedPages)
{
value += loadPage.ExtractText();
}
}
else
{
WordDocument document = new WordDocument();
String path = HttpContext.Server.MapPath("~/uploadfiles/" + name);
document.Open(path);
MemoryStream stream = new MemoryStream();
document.Save(stream, FormatType.Html);
stream.Position = 0;
StreamReader reader = new StreamReader(stream);
value = reader.ReadToEnd();
}
return xhtmlValidation(value);
}
```

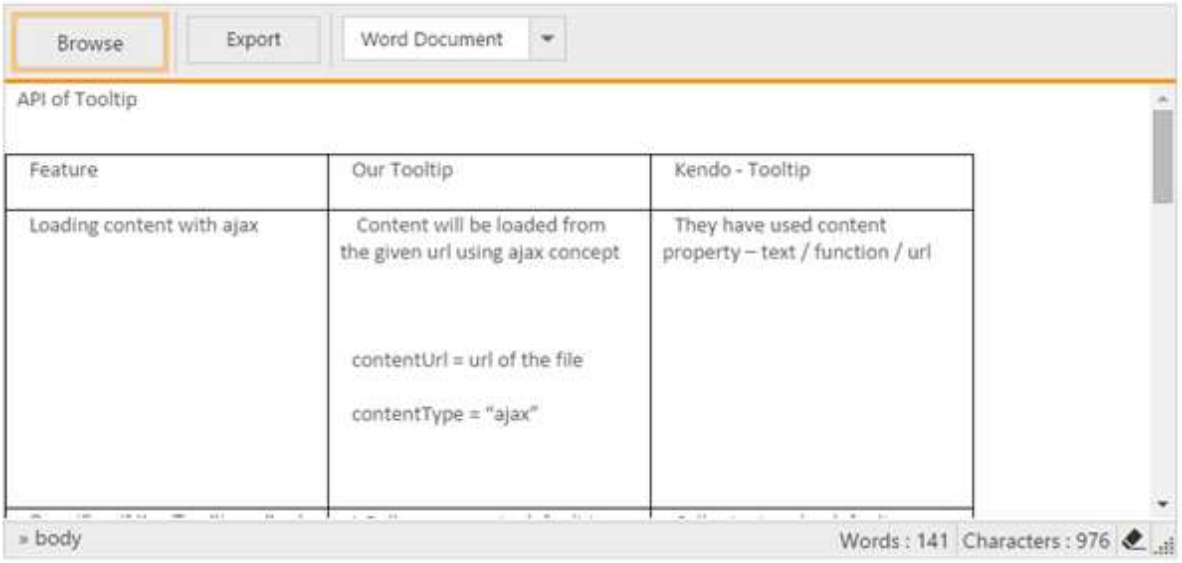
HTML

```
public string xhtmlValidation(string value)
{
if (value != null)
{
if (value.Contains("<html"))
{
int number = value.IndexOf("<body>");
value = value.Remove(0, number + 6);
value = value.Replace("</body></html>", "");
}
value = value.Replace("\"", "'");
value = value.Replace("\r", " ");
value = value.Replace("\n", "<br/> ");
value = value.Replace("\r\n", " ");
value = value.Replace("( )+", " ");
value = value.Replace(" ", " ");
value = value.Replace("&#8226;", "*");
```

Copyright © 2001 - 2020 Syncfusion Inc.

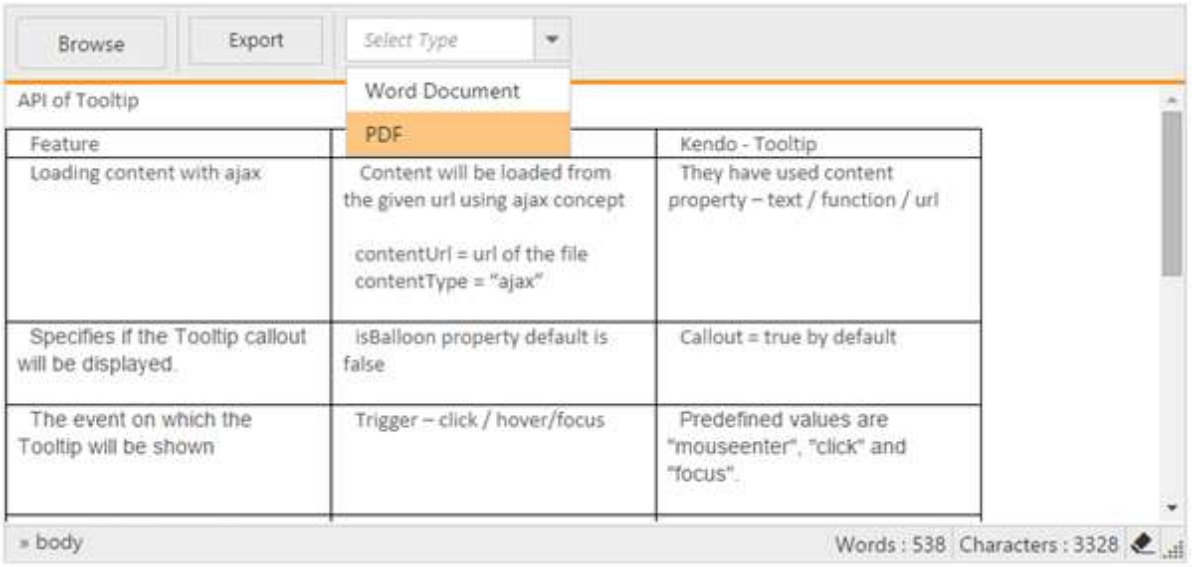
```
value = value.Replace("&#8249;", "<");
value = value.Replace("&#8250;", ">");
value = value.Replace("&#8482;", "(tm)");
value = value.Replace("&#169;", "(c)");
value = value.Replace("&#174;", "(r)");
}
return value;
}
```

Note: If you have selected the document in .pdf format, the text only extracted from the document and set as value to the RichTextEditor.



In above screenshot, use Browse options to browse your files and upload the content to RichTextEditor.

Use DropDownList control in the toolbar to choose the any one type be it is PDF or word document and click Export button to export the content of the RichTextEditor to the selected type.



Upon clicking to the Export button, the post back action has been raised, in that content from the RichTextEditor retrieved and stored as the corresponding format.

HTML

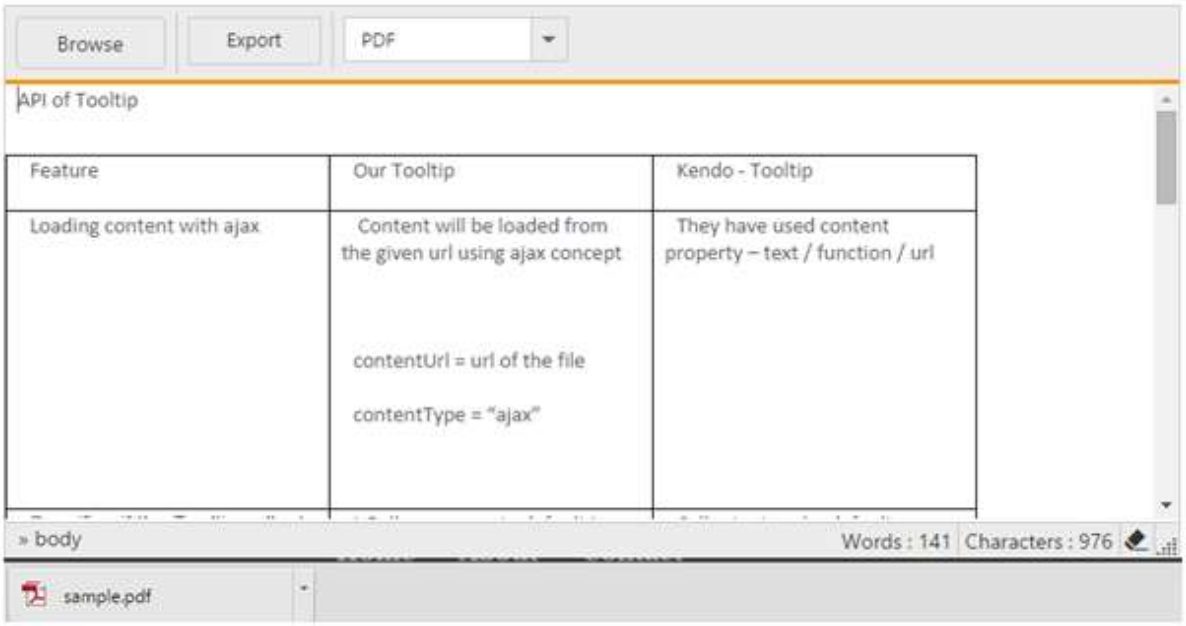
```
[HttpPost]
[ValidateInput(false)]
public void RichTextEditorFeatures(string rte2, string selectType)
{
if (selectType == "PDF")
{
PdfDocument document = new PdfDocument();
ThreadStart starter = () => CreateDocument(rte2);
Thread t = new Thread(starter);
t.SetApartmentState(ApartmentState.STA);
t.Start();
t.Join();
}
else
{
WordDocument document = new WordDocument();
MemoryStream stream = new MemoryStream();
StreamWriter writer = new StreamWriter(stream,
System.Text.Encoding.Default);
string htmlText = rte2;
writer.Write(htmlText);
writer.Flush();
stream.Position = 0;
document = new WordDocument(stream, FormatType.Html,
XHTMLValidationType.Strict);
document.Save("Sample.docx", FormatType.Docx,
HttpContext.ApplicationInstance.Response,
HttpContentDisposition.Attachment);
}
}
```

HTML

```
public void CreateDocument(string rte2)
{
PdfDocument pdfDocument = new PdfDocument();
pdfDocument.PageSettings.Margins.All = 0;
PdfPage page = pdfDocument.Pages.Add();
SizeF pageSize = page.GetClientSize().ToSize();
AspectRatio dimension = AspectRatio.KeepWidth;
HtmlConverter html = new HtmlConverter();
PdfUnitConvertor converter = new PdfUnitConvertor();
float width = converter.ConvertToPixels(pageSize.Width,
PdfGraphicsUnit.Point);
float height = converter.ConvertToPixels(pageSize.Height,
PdfGraphicsUnit.Point);
PdfMetafileLayoutFormat metafileFormat = new PdfMetafileLayoutFormat();
metafileFormat.Break = PdfLayoutBreakType.FitPage;
metafileFormat.Layout = PdfLayoutType.Paginate;
metafileFormat.SplitTextLines = false;
metafileFormat.SplitImages = false;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
string htmlText = @"<html><head><title></title></head><body><div>" + rte2
+ "</div></body></html>";
HtmlToPdfResult result =
html.Convert(htmlText,"",Syncfusion.HtmlConverter.ImageType.Metafile,
(int)width, (int)height, dimension);
result.Render(page, metafileFormat);
pdfDocument.Save("sample.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
}
```



Enable the Browser Spell checker in RTE

Currently we do not have in-built spell checker support in RTE. But we can use the Browser spell checker.

1. First, disable the context menu option from RTE. To do this, please refer the below code snippet.

HTML

```
@{Html.EJ().RTE("rteSample").Width("100%").EnableRTL(true).ShowContextMen
u(false).IFrameAttributes(parameters).Locale("en-
US").ContentTemplate(@<p>
Description:
The Rich Text Editor (RTE) control is an easy to render in client side.
Customer easy to edit the contents and get the HTML content for the
displayed content. A rich text editor control provides users with a
toolbar that helps them to apply rich text formats to the text entered in
the text area.
</p>).Render();}
```

2. Enable the spellcheck property in RTE iframe element. To do this, please refer the below code snippet.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
IDictionary<string, object> parameters = new Dictionary<string,
object>();
parameters.Add("spellcheck", "true");
```

In the above sample, we have disabled the RTE content menu by setting “ShowContextMenu” property as false, so that on right click to the RTE we can get the browser spell check suggestion after enabling the spell checker in Iframe by setting the spellcheck as true in iframeAttributes property.

Pass the RTE from partial view to main view using AJAX call

In main view, the Button control is rendered, upon clicking to the Button, the AJAX call is initiated.

HTML

```
@Html.EJ().Button("button1").Text("Load
RTE").ShowRoundedCorner(true).Size(ButtonSize.Small).ClientSideEvents(e
=> e.Click("onClick"))
<div id="display"></div>
<script type="text/javascript">
var value;
function onClick(e) {
$.ajax({
type: "Post",
url: "/RichTextEditor/LoadRTE",
contentType: "application/json; charset=utf-8",
async: false,
success: function (value) {
$("#display").html(value);
ej.widget.init("#display");
}
});
}
</script>
```

The controller returns the partial view to the AJAX call. In the success function of the AJAX, the returned HTML appended to the div element. In this case, the Syncfusion widgets need to be initiated.

HTML

```
public ActionResult RichTextEditorFeatures()
{
return View();
}
public ActionResult LoadRTE(string value)
{
RTEModel.text = value;
return PartialView("~/Views/RichTextEditor/RTE.cshtml");
}
```

In the partial view, the RTE is rendered as follows

HTML

```
@model  RTEPartialView.Models.RTEModel
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{Html.EJ().RTE("Id1").Width("100%").ContentTemplate(@<p>
</p>).ShowHtmlSource(true).ShowFooter(true).Value(RTEPartialView.Models.R
TEModel.text).Render();}
```

Render RichTextEditor from code behind

RichTextEditor can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of RichTextEditor using RichTextEditor properties from code behind.

RAZOR

```
@model  Syncfusion.JavaScript.Models.RTEproperties
@{
Html.EJ().RTE("rteSample",
(Syncfusion.JavaScript.Models.RTEproperties)ViewData["RTEModel"]).Render(
);   // render the RTE in view
}
```

C#

```
public ActionResult Index()
{
RTEproperties RTEObj = new RTEproperties();
RTEObj.ContentTemplate = new MvcTemplate<object>
{
RazorViewTemplate = (data) =>
{
return "This is RTE content";
}
};
RTEObj.Width = "550";
RTEObj.IsResponsive = true;
ViewData["RTEModel"] = RTEObj;
return View(RTEObj);
}
```

Display content from db and post content to db in RTE

We can use setHtml() method in the RTE control to display the content from the database into RTE.We can edit this content in RTE and get the modified content through getHtml() method of RTE and pass this to controller and save in your database.

The following code illustrates how to set content from db

RAZOR

```
<div>
@{Html.EJ().RTE("rteSample").ContentTemplate(@<p>
&#60;p&#62;&#60;b&#62;Description:&#60;/b&#62;&#60;/p&#62; &#60;p&#62;The
Rich Text Editor
(RTE) control is an easy to render in client side. Customer easy to edit
the contents
and get the HTML content for the displayed content. A rich text editor
control provides
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
users with a toolbar that helps them to apply rich text formats to the
text entered
in the text area. &#60;/p&#62; &#60;p&#62;&#60;b&#62;
</p>).ClientSideEvents(e=> e.Create("onCreate")).Render();}
</div>
<br /><br />
<button id="set" >Import content from Server</button>
<button id="get">Get Content from RTE</button>
<script>
var rteObj;
$(function () {
$("#set").on("click", function () {
$.ajax({
url: "@Url.Action("GetData", "Home")",
type: "GET",
contentType: "application/json; charset=utf-8",
success: function (e) {
rteObj.setHtml(e); //set content from server
}
});
});
$("#get").on("click", function () {
alert(rteObj.getHtml()); //get content from RTE
});
});
function onCreate() {
rteObj = this;
}
</script>
```

C#

```
public ActionResult GetData()
{
var NewContent = "<p>This is <b>New</b> content from the
<i>Controller</i></p>";
return Content(NewContent);    //return data from db
}
```

Rotator

Rotator

The ASP.NET MVCRotator control displays a set of slides. Each slide may contain images or images with content, or content with user-defined transition between them. It supports Data Binding, Thumbnail, Pager, dynamic number of slide move options, and all custom animations. It supports all types of image formats (JPEG, GIF, and so on).

Key Features

- Data Binding: Supports data binding with JSON data as well as remote data.
- Image with Content: Supports to render an image, content, or image with content.
- Display Dynamic number of Slide: Supports dynamic (one or more) number of slides.

Copyright © 2001 - 2020 Syncfusion Inc.

- Move Dynamic Number of Slide: Supports to move dynamic (one or more) number of slides at a time.
- Autoplay: Supports autoplay mode for slide transition.
- Pager and Thumbnail: Support to navigate between slides.

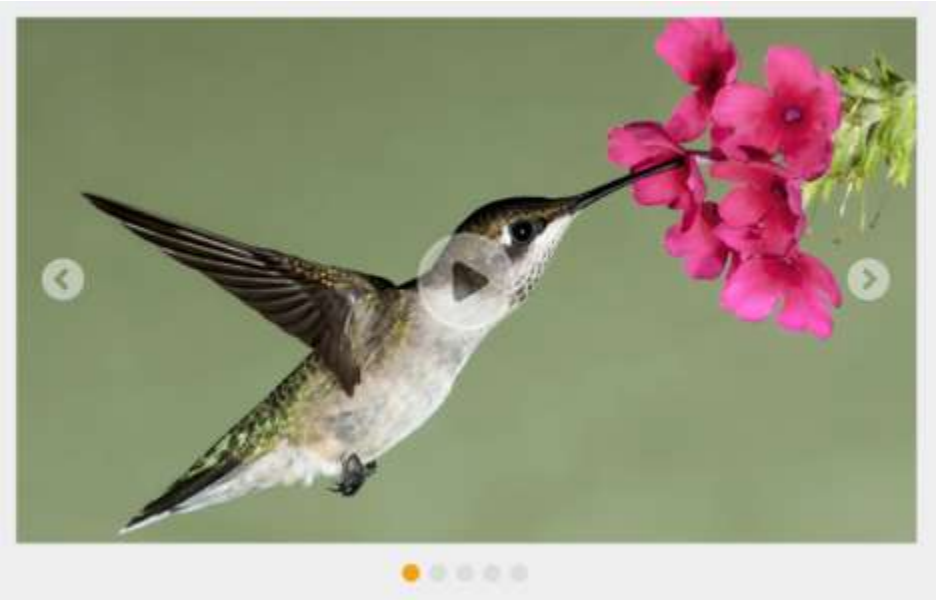
Getting Started

This section explains briefly about how to create a Rotator in your ASP.NET MVC application.

Create your first Rotator in MVC

ASP.NET MVC Rotator provides support to display the provided number of images within your web page in a rotating manner. Refer the following guidelines to create a Rotator widget for a real-time website banner scenario that has five images that slide automatically. When you click the center button, images slide in a rotating manner and on second click, the rotation stops.

The following screenshot illustrates a Rotator widget.



Rotator

Create a Rotator

Essential Studio ASP.NET MVCRotator widget has built-in features such as unobtrusive and flexible APIs. You can easily create the Rotator widget using simple HTML helper as follows.

1. Create a MVC Project and add necessary assemblies, styles, and scripts to it.

Refer MVC-Getting Started.

2. Add the following code example to the corresponding view page to render Rotator. Move the images under the folder ~/Images/rotator.

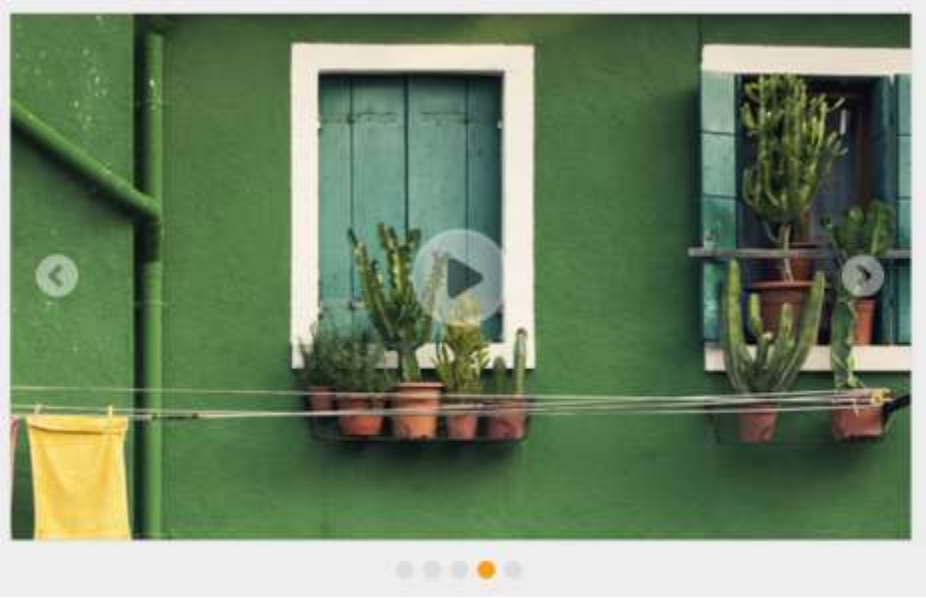
ASPX-CS

```
<div class="frame">
@{Html.EJ().Rotator("content").Items(item =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
item.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/Untitled.png")" />
</div>);
item.Add().ContentTemplate(@<div>
<a href="~/Views/Home/Index.cshtml"></a>
<img class="image" src="@Url.Content("~/Images/rotator/Untitled1.png")"
/>
</div>);
item.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/Untitled2.png")"
/>
</div>);
item.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/Untitled3.png")"
/>
</div>);
item.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/Untitled4.png")"
/>
</div>);
})
.SlideWidth("600px").SlideHeight("350px").ShowPlayButton(true).Render();}
</div>
```

3. The following banner is displayed as output.



Data Binding

Rotator provides a flexible approach for binding the data from various data sources. There are various properties in Rotator for Data Binding. The value set to this property is object type.

Copyright © 2001 - 2020 Syncfusion Inc.

Data Fields and Configuration

The following sub-properties provide a way to bind the data either locally/remotely to the Rotator control.

DataSource

This property specifies the list of data that contains a set of data fields. Each data value is used to render an item for the Rotator. Datasource receives Essential DataManager object and JSON object. The value set to this property is object type.

Fields

It defines mapping fields for the data items of the Rotator. The value set to this property is object type.

Text

It specifies the text content of the Image in Rotator control. The value set to this property is string type.

Url

This property specifies the URL for an image. The value set to this property is string type.

Query

This property receives query to retrieve data from the table (query is same as SQL).This property is applicable only when a remote data source is used. Each data value is used to render an item for the Rotator. The value set to this property is object type.

Local data binding

Rotator provides the data binding support for the Rotator item. So you can bind the data from JSONData. For this behavior, you need to map the corresponding filed with their column names. The data can be bound as a list and it is assigned to Datasource property. You can refer the following code example to bind local data.

C#

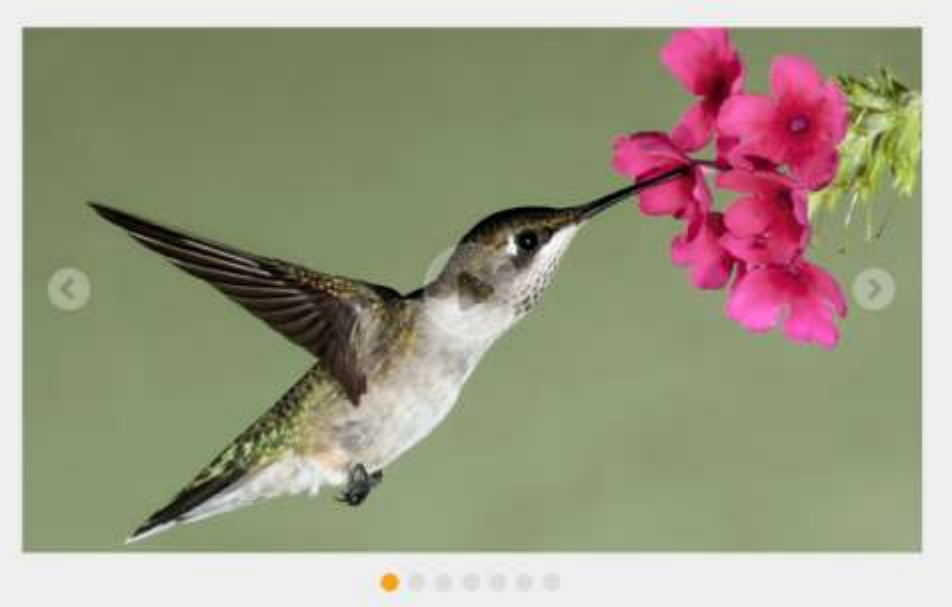
```
// Add the following data list to be bind in the controller page and
define the corresponding data.
// Define local data source elements with  fields
public class LocalData
{
public string Text { get; set; }
public string Url { get; set; }
}
//Refer the Model in the controller
using <Application name>.Models;
List<LocalData> localValues = new List<LocalData>();
public ActionResult Index()
{
localValues.Add(new LocalData { Text = "Beautiful Bird", Url =
"../Images/rotator/bird.jpg" });
localValues.Add(new LocalData { Text = "Colorful Night", Url =
"../Images/rotator/night.jpg" });
localValues.Add(new LocalData { Text = "Technology", Url =
"../Images/rotator/tablet.jpg" });
localValues.Add(new LocalData { Text = "Nature", Url =
"../Images/rotator/nature.jpg" });
localValues.Add(new LocalData { Text = "Snow Fall", Url =
"../Images/rotator/snowfall.jpg" });
localValues.Add(new LocalData { Text = "Credit Card", Url =
"../Images/rotator/card.jpg" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
localValues.Add(new LocalData { Text = "Amazing Sculptures", Url =
"../Images/rotator/sculpture.jpg" });
ViewBag.datasource = localValues;
return View();
}
```

ASPX-CS

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text")
.Url("Url")).SlideWidth("600px")
.SlideHeight("350px")
.ShowPlayButton(true)
```



Rotator control with local data binding

Behavior settings

Enabling rotator

Enabled property is a Boolean type, which allow us to enables or disable the Rotator control. The default value is ‘true’. You can specify the property Enabled in the script as follows.

ASPX-CS

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px").Enabled(false)
```

The following banner displaying as output.

Copyright © 2001 - 2020 Syncfusion Inc.



Responsive rotator

IsResponsive

IsResponsive property is Boolean type, which allow us to resizes the Rotator when the browser window is resized. The default value is ‘false’. You can specify the property IsResponsive in the script as follows.

ASPX-CS

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").IsResponsive(true)
```

EnableAutoPlay

The Rotator Items continuously rotate without user interference by enable the EnableAutoPlay property. The default value is ‘false’. The value set to this property is Boolean. You can specify the property EnableAutoPlay in the script as follows.

ASPX-CS

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").EnableAutoPlay(true)
```

StopOnHover

StopOnHover property is Boolean type, which allow us to pause the auto play while hover on the Rotator content. The default value is ‘false’. You can specify the property StopOnHover in the script as follows.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").EnableAutoPlay(true).StopOnHover(true)
```

Pager settings

PagerPosition

This property specifies the position of the Pager in the Rotator Item. The default value is ‘Outside’. This property is only applicable when ShowPager property is set true. The value set to the PagerPosition property is string or enum.

- Syncfusion.JavaScript.PagerPosition.BottomLeft
- Syncfusion.JavaScript.PagerPosition.BottomRight
- Syncfusion.JavaScript.PagerPosition.Outside
- Syncfusion.JavaScript.PagerPosition.TopCenter
- Syncfusion.JavaScript.PagerPosition.TopLeft
- Syncfusion.JavaScript.PagerPosition.TopRight

You can specify the property PagerPosition in the script as follows.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px")
.PagerPosition(Syncfusion.JavaScript.PagerPosition.TopLeft)
```



ShowPager

This property turns on or off the pager support in the Rotator control. The Pager is used to navigate the Rotator Items. The default value is ‘true’. The value set to this property is Boolean.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").ShowPager(false)
```



Show options

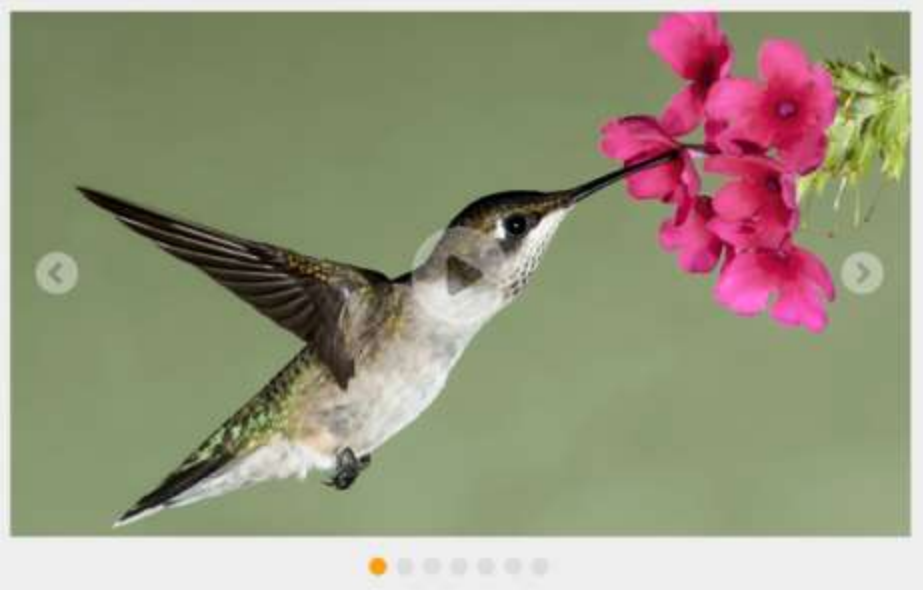
ShowPlayButton

ShowPlayButton property is Boolean type which allow us to enable play / pause button on Rotator. The default value of ShowPlayButton is ‘false’.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource).RotatorFields(t =>
t.Text("Text").Url("Url")).SlideWidth("600px").SlideHeight("350px").ShowP
layButton(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

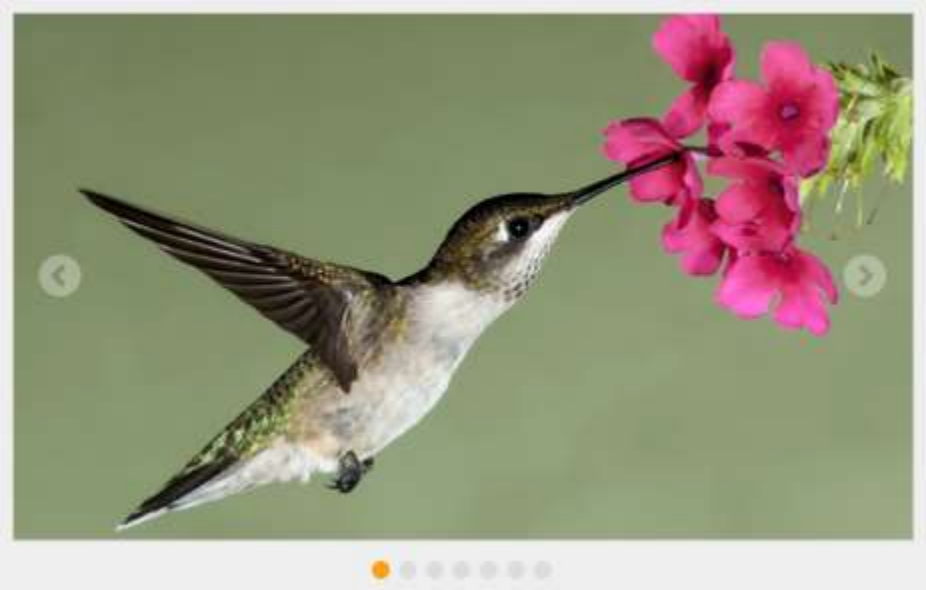


ShowNavigateButton

This property turns on or off the slide buttons (next and previous) in the Rotator Items. Slide buttons are used to navigate the Rotator Items. The default value is ‘false’. The value set to this property is Boolean.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px").ShowNavigateButton(true)
```



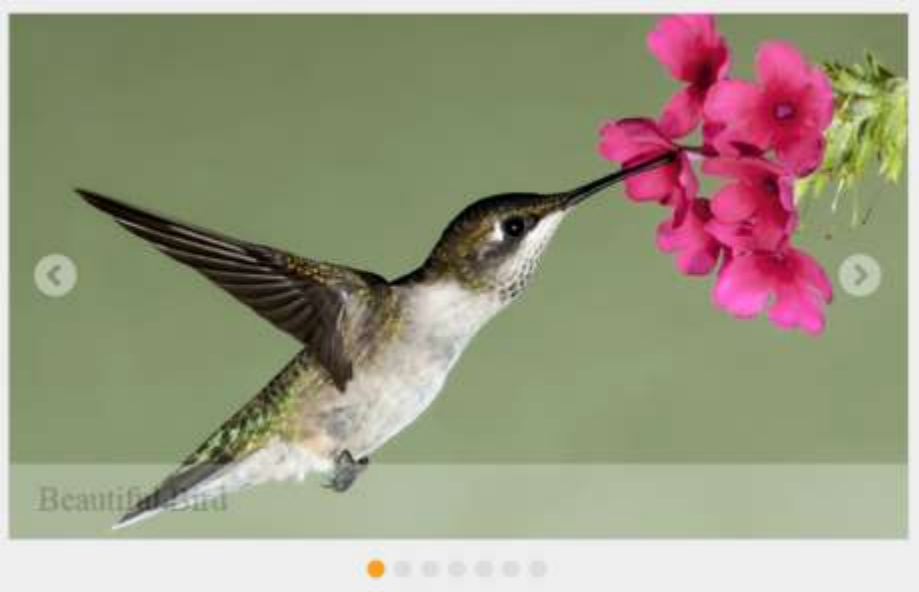
Copyright © 2001 - 2020 Syncfusion Inc.

ShowCaption

When the Rotator Item is an image, you can specify a caption for the Rotator Item. The caption text for each Rotator Item is shown by using ShowCaption property. This property is Boolean type, which allow us to Show or Hide the Caption about Images on bottom of Image. The caption cannot be displayed when multiple Rotator Items are present. The default value is ‘false’.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").ShowCaption(true)
```



Thumbnail

This feature implements Thumbnail in Rotator control. You can view or access any of the Rotator items instantly. All the images are given as Thumb Element to use this feature.

The property ShowThumbnail is Boolean type, which allow us to Show or Hide the Thumbnail in Rotator control. Thumbnail is used to navigate between slides. Thumbnail supports only single slide transition. You must specify the source for thumbnail elements through the ThumbnailSourceID property. The default value is ‘false’.

The property ThumbnailSourceID specifies the source for thumbnail elements. The default value is null. The value set to this property is object.

You can refer the following code example of Thumbnail in Rotator.

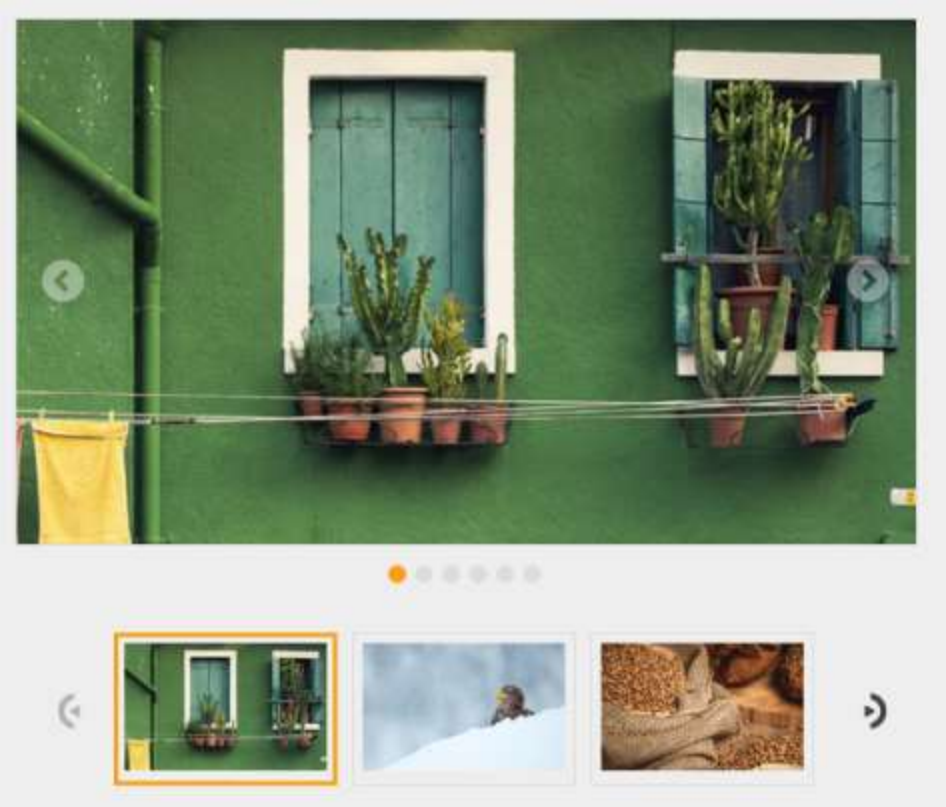
ASPX-CS

```
<ul id="slide" style="display: none">
<li>
<img src="@Url.Content("~/Images/rotator/green.jpg")" title="Green"
/></li>
<li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<img src="@Url.Content("~/Images/rotator/snow.jpg")" title="Snow" /></li>
<li>
<img src="@Url.Content("~/Images/rotator/wheat.jpg")" title="Wheat"
/></li>
<li>
<img src="@Url.Content("~/Images/rotator/tablet.jpg")" title="Tablet"
/></li>
<li>
<img src="@Url.Content("~/Images/rotator/sea.jpg")" title="Sea" /></li>
<li>
<img src="@Url.Content("~/Images/rotator/bread.jpg")" title="Bread"
/></li>
</ul>
@Html.EJ().Rotator("sliderContent").Items(itemElement =>
{
itemElement.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/green.jpg")" />
</div>);
itemElement.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/snow.jpg")"/>
</div>);
itemElement.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/wheat.jpg")" />
</div>);
itemElement.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/tablet.jpg")" />
</div>);
itemElement.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/sea.jpg")" />
</div>);
itemElement.Add().ContentTemplate(@<div>
<img class="image" src="@Url.Content("~/Images/rotator/bread.jpg")" />
</div>);
}).SlideWidth("600px").SlideHeight("350px").ShowThumbnail(true).Thumbnail
SourceID("slide")
```

Copyright © 2001 - 2020 Syncfusion Inc.



Orientation

The Rotator control supports both vertical and horizontal orientations allowing it to fit into any scenario. The Orientation property defines the Orientation where the control is rendered. The value set to this property is enum or string type. It accepts the following values.

- ej.Orientation.Horizontal or “Horizontal”
- ej.Orientation.Vertical or “Vertical”

The following steps explain you how to set Orientation for the Rotator.

Horizontal

This property sets the Rotator in HorizontalOrientation. You can refer the following steps to set Horizontal Orientation for Rotator control. In Rotator control, the Default value of Orientation is Horizontal.

1. In View, create ul-li list of Rotator items and invoke the Rotator helper.
2. Add the following script in your CSHTML page.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.SlideHeight("350px")
.Orientation(Syncfusion.JavaScript.Orientation.Horizontal)
```

Vertical

This property sets the Rotator in vertical orientation. You can refer the following steps to set VerticalOrientation for Rotator control.

1. Create ul-li list of Rotator items and invoke the Rotator helper.
2. Add the following script in your CSHTML page.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px")
.Orientation(Syncfusion.JavaScript.Orientation.Vertical)
```

Appearance and Styling

Adjusting rotator item size

SlideWidth

This property sets the Width of a Rotator Item. The value set to this property is string or number.

ASPX-CS

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px")
```

SlideHeight

This property sets the Height of a Rotator Item. The value set to this property is string or number.

ASPX-CS

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<Localdata>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px")
```

Image with Contents

This feature allows you to add text along with the image in Rotator control. This is achieved by splitting the content into two panels. In the following code example, image is given in the left panel and text is given in the right panel.

ASPX-CS

```
@{Html.EJ().Rotator("sliderContent").Items(itemElement =>
{
itemElement.Add().ContentTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div class="leftPanel">
<img src="@Url.Content("~/Images/rotator/tablet.jpg")" />
</div>
<div class="rightPanel black">
<div class="contentPanel">Tablet </div>
<ul>
<li>A tablet computer, or simply tablet, is a mobile computer with
display, circuitry and battery in a single unit.</li>
<li>
Tablets are equipped with sensors, including cameras, microphone,
accelerometer and touchscreen,
</ul>
</div>
</div>);
itemElement.Add().ContentTemplate(@<div>
<div class="leftPanel">
<img src="@Url.Content("~/Images/rotator/rose.jpg")" />
</div>
<div class="rightPanel">
<div class="contentPanel">Rose </div>
<ul>
<li>A rose is a woody perennial of the genus Rosa, within the family
Rosaceae</li>
<li>Flowers vary in size and shape and are usually large and showy,
There are over 100 species
</li>
</ul>
</div>
</div>);
itemElement.Add().ContentTemplate(@<div>
<div class="leftPanel">
<img src="@Url.Content("~/Images/rotator/snowfall.jpg")" />
</div>
<div class="rightPanel rightSide">
<div class="contentPanel">Snowfall </div>
<ul>
<li>Mt. Baker ski area in Washington State has the world record for
snowfall at 1,140 inches of snow in the 1998/1999 winter season</li>
<li>Mt. Baker ski area is located near but not on the real 10,781’ Mount
Baker</li>
</ul>
</div>
</div>);
itemElement.Add().ContentTemplate(@<div>
<div class="leftPanel">
<img src="@Url.Content("~/Images/rotator/nature.jpg")" />
</div>
<div class="rightPanel">
<div class="contentPanel">Nature </div>
<ul>
<li>The health of the natural environment is critical to the long-term
future of the planet</li>
<li>Nature, in the broadest sense, is equivalent to the natural,
physical, or material world or universe.</li>
</ul>
</div>
</div>);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
itemElement.Add().ContentTemplate(@<div>
<div class="leftPanel">
<img src="@Url.Content("~/Images/rotator/card.jpg")" />
</div>
<div class="rightPanel credit">
<div class="contentPanel">Credit card </div>
<ul>
<li>A credit card is a payment card issued to users as a system of
payment</li>
<li>It allows the card holder to pay for goods and services based on the
holder's promise to pay for them</li>
</ul>
</div>
</div>);
}).SlideWidth("700px").SlideHeight("300px").ShowPlayButton(true).Render()
;}
```

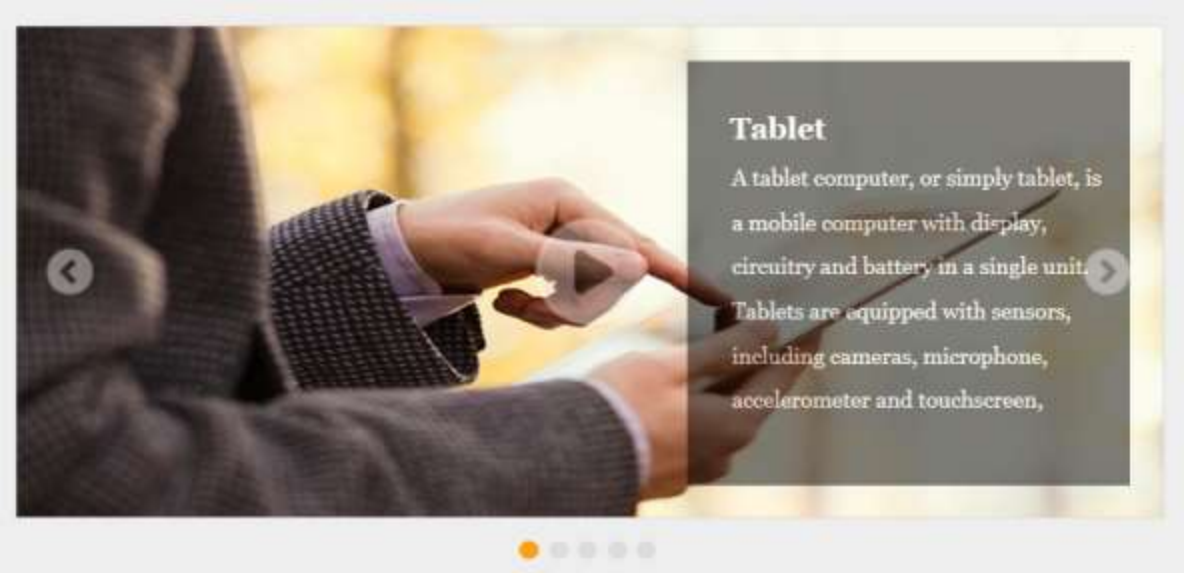
CSS

```
<style type="text/css" class="cssStyles">
#sliderContent > li
{
background-color: #9ee8d8;
}
#sliderContent > li .leftPanel
{
float: left;
width: 700px;
height: 300px;
padding-right: 0px;
}
#sliderContent > li .leftPanel img
{
width: 700px;
height: 300px;
}
#sliderContent .rightPanel
{
background-color: #FFFFFF;
height: 259px;
margin-left: 410px;
margin-top: 21px;
opacity: 0.5;
padding-left: 10px;
position: absolute;
width: 260px;
}
#sliderContent .rightPanel.credit
{
opacity: 0.6;
}
#sliderContent .rightPanel.black
{
background-color: black;
}
#sliderContent .rightPanel.black li
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
color: white;
list-style: none;
line-height: 2;
}
#sliderContent .rightPanel.black .contentPanel
{
padding-top: 30px;
color: white;
}
#sliderContent .rightPanel .contentPanel
{
color: #000000;
font-size: large;
font-weight: bold;
left: 16px;
padding-top: 30px;
position: relative;
}
#sliderContent .rightPanel li
{
color: black;
list-style: none;
line-height: 2;
}
#sliderContent .rightPanel.rightSide
{
margin-left: 20px;
background-color: black;
}
#sliderContent .rightPanel.rightSide li
{
color: white;
list-style: none;
line-height: 2;
}
#sliderContent .rightPanel.rightSide .contentPanel
{
padding-top: 30px;
color: white;
}
.rightPanel > ul
{
padding: 6px 17px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Display items

DisplayItemCount

This property specifies the number of Rotator Items to be displayed. The default value is ‘1’. The value set to this property is string or number.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").DisplayItemCount("3")
```



NavigateSteps

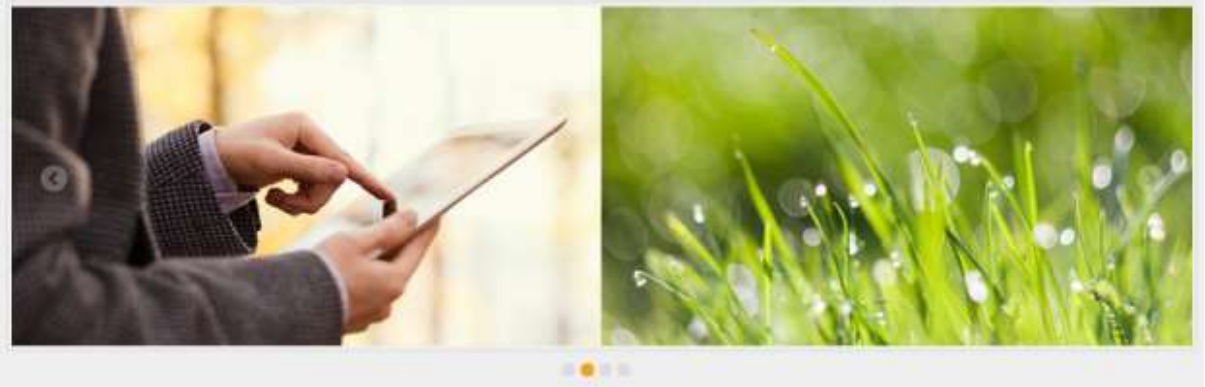
This property specifies the number of Rotator Items to navigate on a single click (next/previous/play buttons). The NavigateSteps property value must be less than or equal to the DisplayItemCount property value. The default value is ‘1’. The value set to this property is string or number.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.SlideWidth("600px").SlideHeight("350px")
.DisplayItemCount("2").NavigateSteps("2")
```

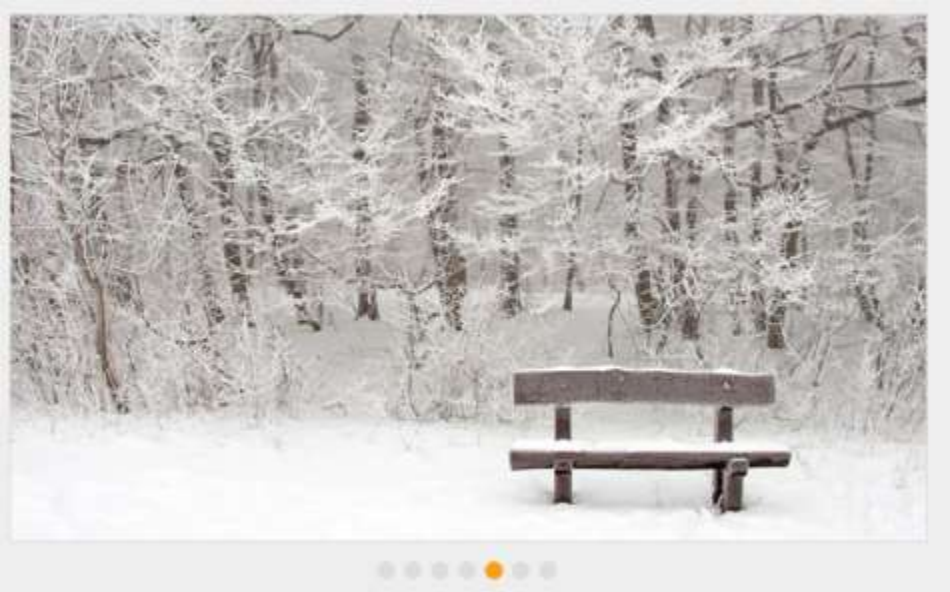


StartIndex

This property sets the index of the slide that is displayed first. The default value is ‘0’. The value set to this property is string or number.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").StartIndex("4")
```



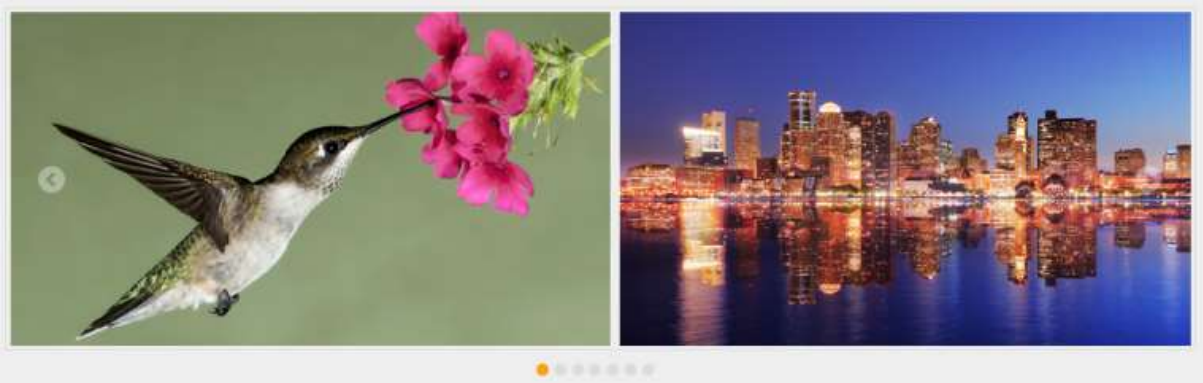
FrameSpace

This property sets the space between the Rotator Items. The value set to this property is string or number.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url")).SlideWidth("600px")
.SlideHeight("350px").DisplayItemCount("2").FrameSpace("40px")
```



AnimationType

AnimationType property specifies the Type of Animation for the Rotator Item. AnimationType options include slide, fastSlide, slowSlide, and other custom easing Animation Types. The default value is ‘slide’. The value set to this property is string.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px")
.SlideHeight("350px")
.AnimationType("fastSlide")
```

AnimationSpeed

This property sets the speed of slide transition. The default value of AnimationSpeed is ‘600’. The value set to this property is string or number.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text")
.Url("Url")).SlideWidth("600px")
.SlideHeight("350px")
.AnimationSpeed(2000)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Delay

This property sets the delay between the Rotator Items to move after the slide transition. The default value is 500. The value set to this property is string or number.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px")
.SlideHeight("350px")
.Delay(2000)
```

Theme

Rotator control's style and appearance are controlled based on CSS classes. In order to apply styles to the Rotator control, you can refer 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. When the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these both.

By default, there are 12 themes support available for Rotator control as follows,

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

CssClass

This property is used to set root class for Rotator control theme. The value set to this property is string type.

ASPX-CS

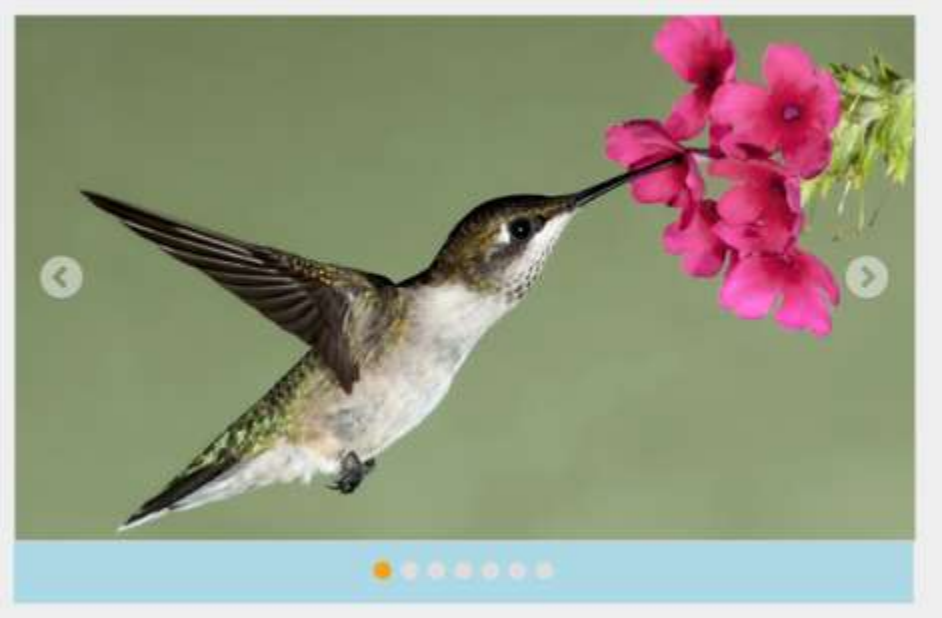
```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px")
.SlideHeight("350px")
.CssClass("myCss")
```

Copyright © 2001 - 2020 Syncfusion Inc.

Add the following code in your CSS.

CSS

```
<style>
.flat-lime
{
background-color: yellowgreen;
}
</style>
<style type="text/css">
.blue
{
background-color: lightblue;
}
</style>
```



Template

This Template feature implements in Rotator control. Rotator template will allow you to customize the rendering structure for its li elements. Using template API with data binding can render image, audio or video inside the Rotator control.

The property template specifies the structure for slide li elements. The default value is null. The value set to this property is string.

1. Add the below code in your view page to render the Rotator

ASPX-CS

```
// Add the following code in View page to configure Rotator widget
<div class="frame">
@Html.EJ().Rotator("rot").Datasource((IEnumerable<LocalData>)ViewBag.data
source).SlideWidth("100%").SlideHeight("350px").IsResponsive(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Template("<div class='image'><img src = ${url} title = ${text}
class='image'/> </div>")
</div>
```

CSHARP

```
// Add the following code to add slide image items in the controller page
List<LocalData> localValues = new List<LocalData>();
public ActionResult LocalData()
{
localValues.Add(new LocalData{ text= "Beautiful Bird", url=
"../Images/rotator/bird.jpg" });
localValues.Add(new LocalData { text = "Colorful Night", url =
"../Images/rotator/night.jpg" });
localValues.Add(new LocalData { text = "Technology", url =
"../Images/rotator/tablet.jpg" });
localValues.Add(new LocalData { text = "Nature", url =
"../Images/rotator/nature.jpg" });
localValues.Add(new LocalData { text = "Snow Fall", url =
"../Images/rotator/snowfall.jpg" });
localValues.Add(new LocalData { text = "Credit Card", url =
"../Images/rotator/card.jpg" });
localValues.Add(new LocalData { text = "Amazing Sculptures", url =
"../Images/rotator/sculpture.jpg" });
ViewBag.datasource=localValues;
return View();
}
```

2. CSS.

CSS

```
<style type="text/css" class="cssStyles">
.frame {
width: 100%;
box-sizing : border-box;
}
#rot > li img {
width: 100%;
height: 350px;
}
</style>
```

RTL Support

EnableRTL feature supports to change the left-to-right alignment of the Rotator to right-to-left (RTL). (I.e.) Sets the Rotator to do its actions from right to left. The EnableRTL property sets the Rotator from right to left. The value set to this property is Boolean type.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
binding Rotator items.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Rotator("sliderContent").Datasource((IEnumerable<LocalData>)Vi
ewBag.datasource)
.RotatorFields(t => t.Text("Text").Url("Url"))
.SlideWidth("600px").SlideHeight("350px")
.EnableRTL(true)
```

Keyboard interaction

The AllowKeyboardNavigation property turns on keyboard interaction with the Rotator items. You must set this property to ‘true’ to access the keyboard shortcuts. The default value is ‘true’. The value set to this property is Boolean.

The entire Rotator commands are accessed through the keyboard by specifying the KeyboardShortcut in the following table.

Keyboard Shortcut	Function
Alt + j	Focuses the control
Up/ Right	Move to next slide
Down/Left	Move to previous slide.
Space	Move to Play/Pause slide.
Alt + Right	Move thumbnail item to right and select item.
Alt + Left	Move thumbnail item to left and select item.
Enter	Selected the thumbnail item

You can refer the following code example for keyboard navigation.

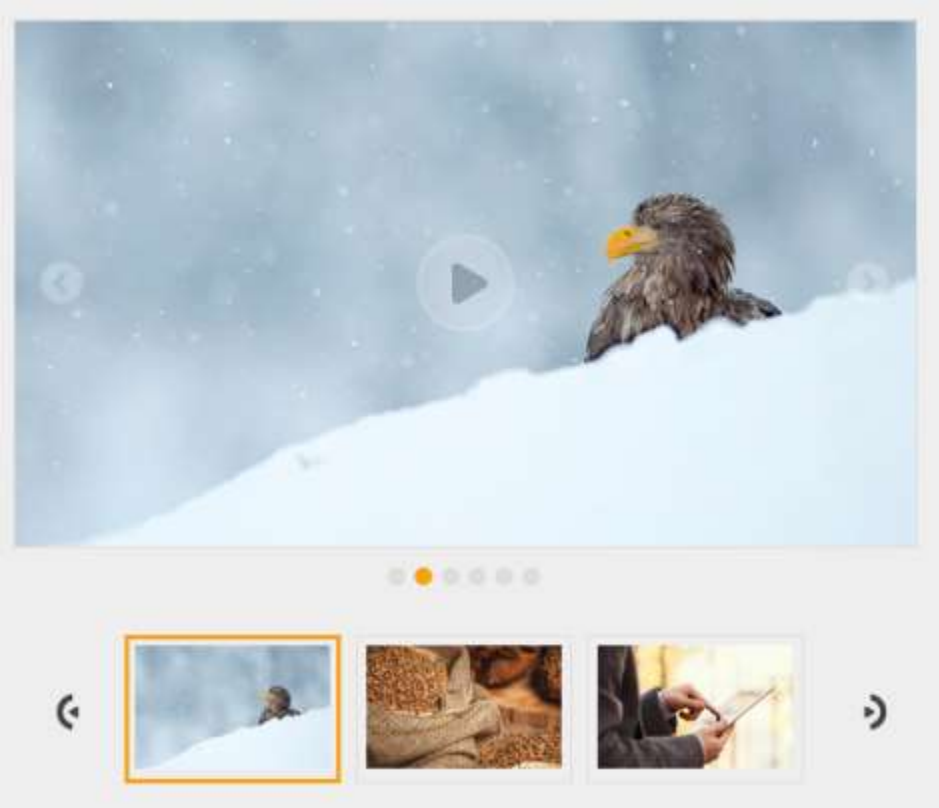
ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
binding Rotator items.
<ul id="slide" style="display: none">
<li> <img src="@Url.Content("~/Images/rotator/green.jpg")" title="Green"
/> </li>
<li> <img src="@Url.Content("~/Images/rotator/snow.jpg")" title="Snow" />
</li>
<li> <img src="@Url.Content("~/Images/rotator/wheat.jpg")" title="Wheat"
/> </li>
<li> <img src="@Url.Content("~/Images/rotator/tablet.jpg")"
title="Tablet" /> </li>
<li>        <img src="@Url.Content("~/Images/rotator/sea.jpg")"
title="Sea" /></li>
<li>        <img src="@Url.Content("~/Images/rotator/bread.jpg")"
title="Bread" /></li>
</ul>
@Html.EJ().Rotator("sliderContent").Items(itemElement =>  {
itemElement.Add().ContentTemplate(@<div> <img class="image"
src="@Url.Content("~/Images/rotator/green.jpg")" /> </div>);
itemElement.Add().ContentTemplate(@<div> <img class="image"
src="@Url.Content("~/Images/rotator/snow.jpg")"/>  </div>);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
itemElement.Add().ContentTemplate(@<div> <img class="image"
src="@Url.Content("~/Images/rotator/wheat.jpg")" />  </div>);
itemElement.Add().ContentTemplate(@<div> <img class="image"
src="@Url.Content("~/Images/rotator/tablet.jpg")" /> </div>);
itemElement.Add().ContentTemplate(@<div> <img class="image"
src="@Url.Content("~/Images/rotator/sea.jpg")" /> </div>);
itemElement.Add().ContentTemplate(@<div> <img class="image"
src="@Url.Content("~/Images/rotator/bread.jpg")" /> </div>);})
.SlideWidth("600px").SlideHeight("350px")
.ShowThumbnail(true).ThumbnailSourceID("slide")
.AllowKeyboardNavigation(true).ShowPlayButton(true)
<script type="text/JavaScript">
$(function ()
{
//Control focus key
$(document).on("keydown", function (e)
{
if (e.altKey && e.keyCode === 74)
{
// j- key code.
$("#sliderContent")[0].focus();
}
});
});
</script>
```

Run the above sample, we get the output like this,



Copyright © 2001 - 2020 Syncfusion Inc.

How to

Render Rotator from Code behind

Rotator can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side

The following code illustrates the initialization of Rotator properties in the controller.

C#

```
//Namespace to get the JavaScript (Rotator) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class RotatorController: Controller
{
public ActionResult RotatorFeatures()
{
// initialize the data list
List<LocalData> localValues = new List<LocalData>();
localValues.Add(new LocalData { Text = "Beautiful Bird", Url =
"../Images/rotator/bird.jpg" });
localValues.Add(new LocalData { Text = "Colorful Night", Url =
"../Images/rotator/night.jpg" });
localValues.Add(new LocalData { Text = "Technology", Url =
"../Images/rotator/tablet.jpg" });
localValues.Add(new LocalData { Text = "Nature", Url =
"../Images/rotator/nature.jpg" });
localValues.Add(new LocalData { Text = "Snow Fall", Url =
"../Images/rotator/snowfall.jpg" });
localValues.Add(new LocalData { Text = "Credit Card", Url =
"../Images/rotator/card.jpg" });
localValues.Add(new LocalData { Text = "Amazing Sculptures", Url =
"../Images/rotator/sculpture.jpg" });
//Initializing the Rotator model
RotatorProperties RotatorObj = new RotatorProperties();
//Initializing the datasource and other properties
RotatorObj.DataSource = localValues;
RotatorObj.RotatorFields.Text = "Text";
RotatorObj.RotatorFields.Url = "Url";
RotatorObj.SlideWidth = "600px";
RotatorObj.SlideHeight = "350px";
RotatorObj.ShowPlayButton = true;
//Passing Rotator properties using the ViewData
ViewData["RotatorModel"] = RotatorObj;
return View();
}
}
}
```

Binding the Rotator properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Html.EJ().Rotator("sliderContent",
(Syncfusion.JavaScript.Models.RotatorProperties)ViewData["RotatorModel"])
.Render();
}
```

Schedule

Overview

The **EJMVC Scheduler** is an event calendar that manages the list of various activities (events/appointments) in different available views (day, week, workweek, month and agenda) for various resources. It is mainly for tracking the user appointments and allows them to create a new or edit/delete the older ones. It is simply a server-side wrapper of JS Scheduler component and almost all the features in JS Scheduler are applicable to Scheduler in MVC platform.

Key Features

Some of the key features of Scheduler are as follows,

- **DataSource** - Supports various client-side and remote data sources such as JSON, RESTful services, OData services, WCF services and much more.
- **Views** - 6 types of views are available namely Day, Week, WorkWeek, Month, Agenda and Custom View (user-specific date rendering).
- **Adaptive** - Scheduler UI layout adapts automatically according to the desktop/mobile mode (responsive support).
- **Resize & Drag** - Appointments can be resized and dragged anywhere within the Scheduler. External drag and drop of appointments are also applicable now.
- **Multiple Resources & Grouping** - Allows the Scheduler to categorize and display resources in a hierarchical structure either in a horizontal or vertical manner.
- **Orientation** - Two types of control orientation is supported namely – Vertical and Horizontal (Timeline View).
- **Categories** - 6 default types of Appointment categories along with user customization options are available to differentiate the appointment status.
- **Template** - Template customization provided for appointments, resource header, cells, date header, priority, tooltip, time scale and agenda view.
- **TimeZone & DST** - Supports observation of Daylight Saving Time in Scheduler for whichever time zone it is applicable.
- **Export & Print** - Supports exporting of single/all appointment(s) to an ICS file and also Prints all/specific appointment(s).
- **PDF Export** - Supports exporting of entire Scheduler into PDF format.
- **Appointment window Customization** – Entire appointment window can be customized with the user required fields.
- **Recurrence Editor** - Complete recurrence related options of Scheduler are collectively defined as a separate plug-in, which can be utilized directly within the customized appointment window.

Getting Started

Follow the steps pointed out in the Getting Started page of the Introduction part to create an MVC application with required assembly references, scripts and stylesheets.

Copyright © 2001 - 2020 Syncfusion Inc.

Create a new Class in controller page to define the data to be passed to the Scheduler as mentioned below,

C#

```
public class ScheduleData
{
public int ProgramId { get; set; }
public string ProgramName { get; set; }
public string Comments { get; set; }
public DateTime ProgramStartTime { get; set; }
public DateTime ProgramEndTime { get; set; }
public bool IsAllDay { get; set; }
public bool IsRecurrence { get; set; }
public string RecurrenceRule { get; set; }
}
```

Now create an instance of the class and add the list of Scheduler data to it within the Index method in controller page, which will then be passed to the View page and bound to the Scheduler dataSource.

C#

```
public ActionResult Index()
{
List<ScheduleData> data = new List<ScheduleData> {
new ScheduleData {
ProgramName = "Turtle Walk",
Comments = "Night out with turtles",
ProgramStartTime = new DateTime(2016, 6, 2, 3, 0, 0),
ProgramEndTime = new DateTime(2016, 6, 2, 4, 0, 0),
IsAllDay = true
},
new ScheduleData {
ProgramName = "Winter Sleepers",
Comments = "Long sleep during winter season",
ProgramStartTime = new DateTime(2016, 6, 3, 1, 0, 0),
ProgramEndTime = new DateTime(2016, 6, 3, 2, 0, 0)
},
new ScheduleData {
ProgramName = "Estivation",
Comments = "Sleeping in hot season",
ProgramStartTime = new DateTime(2016, 6, 4, 3, 0, 0),
ProgramEndTime = new DateTime(2016, 6, 4, 4, 0, 0)
}
};
return View(data);
}
```

Add the Scheduler code in View page as shown below,

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014,6,2))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("ProgramId")
.Subject("ProgramName")
.StartTime("ProgramStartTime")
.EndTime("ProgramEndTime")
.Description("Comments")
.AllDay("IsAllDay")
.Recurrence("IsRecurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

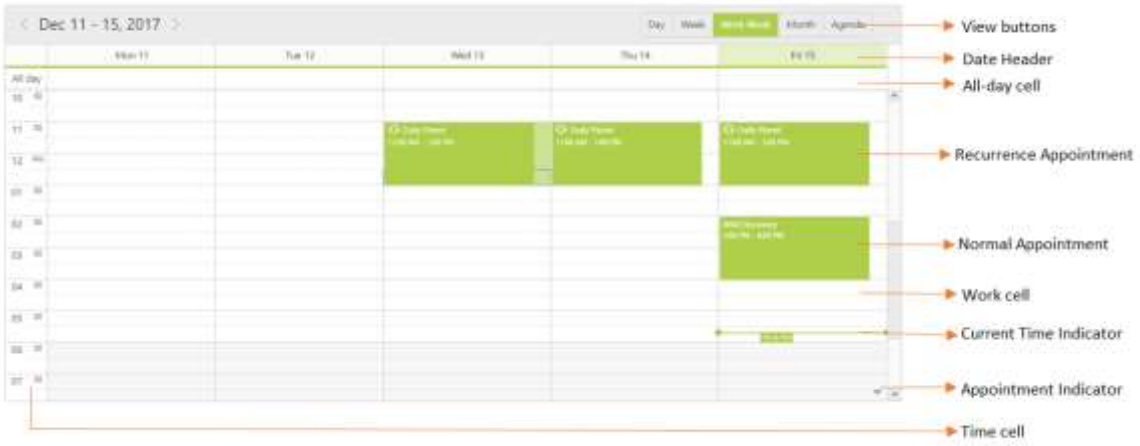
Note: The appointment fields like Id, Subject, Description and other required fields are needed to be mapped with the appropriate column names from the dataSource as done in the above code example.

To bind the remote data directly to the Scheduler instead of passing data from the Model as previously depicted, refer the below code snippet.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014,6,2))
.AppointmentSettings(fields =>
fields.Datasource("http://mvc.syncfusion.com/ODataServices/Northwnd.svc/"
)
.Query("ej.Query().from('Events').take(10)")
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

To bind the data to the Scheduler from database or any other dataSources, refer here.



Data Binding

Appointment Fields

The below listed names are the appointment fields which holds the appropriate column names from the dataSource.

Field name	Description
Id	Binds the Id field name for indexing and performing CRUD operation on the appointments. Itâ€™s optional.
StartTime	Binds the appointment start time field name which is mandatory and also its related validation rules.
StartTimeZone	Binds the name of the start timezone field in the dataSource and also its related validation rules. If the StartTimeZone field is not mentioned, then the appointment makes use of the Scheduler timeZone or System timeZone.
EndTime	Binds the appointment end time field name which is mandatory and also its related validation rules.
EndTimeZone	Binds the name of the end timezone field in the dataSource and also its related validation rules. If the EndTimeZone field is not mentioned, then the appointment makes use of the Scheduler timeZone or System timeZone.
Subject	Binds the appointment subject field name which holds the summary of the appointment and also its related validation rules.
Location	Binds the name of the location field and also its related validation rules. It indicates the appointment location/occurrence place. This field needs to be bind to the Scheduler, when an API ShowLocationField is set to true.
Description	Binds the appointment description field name and also its related validation rules.
AllDay	Binds the name of the AllDay field. It accepts the boolean value and indicates whether the appointment is an all-day appointment or not.
Categorize	Binds the name of the categorize field and also its related validation rules. It indicates the category or status value (red categorize, green, yellow and so on).
Priority	Binds the name of the priority field and its related validation rules, which indicates the priority (high, low, medium and none) of the appointments. This field should be bind to the Scheduler, when PrioritySettings.Enable is set to true.
ResourceFields	Binds one or more fields in the resource collection. It maps the resource field names with the appointments, denoting to which resource the appointments actually belongs.

Copyright © 2001 - 2020 Syncfusion Inc.

Recurrence	Binds the name of the recurrence field. It accepts the boolean value and indicates whether the appointment is a recurrence appointment or not.
RecurrenceRule	Binds the name of the recurrenceRule field. It holds the recurrence pattern associated with the appointments.
RecurrenceId	Binds the recurrence Id field which acts as a parent id for Scheduler recurrence appointments.
RecurrenceExDate	Binds the recurrence Exception field which accepts the recurrence Exception date values.

The below example depicts the appointment fields accepting the string type mapper fields.

Create a new Class **ScheduleData** to define the appointment related data to be passed to the Scheduler as mentioned below,

C#

```
public class ScheduleData
{
public string Id { get; set; }
public string Subject { get; set; }
public string Description { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string Categorize { get; set; }
public string RoomId { get; set; }
public string OwnerId { get; set; }
public string Priority { get; set; }
public bool AllDay { get; set; }
public bool Recurrence { get; set; }
public string RecurrenceRule { get; set; }
}
```

Also, define the below model classes in your application, if there is a need to make use of the features such as multiple resources, categories, priorities, context menu and timezone collection in your sample application.

C#

```
// Define the below class, whenever the multiple resources and grouping
feature is used in your sample project.
public class ResourceFields
{
public string Text { set; get; }
public string Id { set; get; }
public string GroupId { set; get; }
public string Color { set; get; }
public int WorkHourStart { set; get; }
public int WorkHourEnd { set; get; }
public List<string> CustomDays { set; get; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
// Define the below class, whenever the category related data is to be
used in Scheduler.
public class CategorizeSettings
{
public string Text { set; get; }
public string Id { set; get; }
public string FontColor { set; get; }
public string Color { set; get; }
}
// Define the below class, whenever the priorities are to be used for
Scheduler appointments.
public class PrioritySettings
{
public string Text { set; get; }
public string Value { set; get; }
}
// Define the below two classes (Appointment and Cells), if context menu
items are to be used in the Scheduler.
public class Appointment
{
public string Text { set; get; }
public string Id { set; get; }
}
public class Cells
{
public string Text { set; get; }
public string Id { set; get; }
public string ParentId { set; get; }
}
// Define the below class, if timezone collection to be specified for
Scheduler.
public class TimezoneCollection
{
public string Text { set; get; }
public string Id { set; get; }
public string Value { set; get; }
}
```

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleData> Appoint = new List<ScheduleData>();
Appoint.Add(new ScheduleData { Id = "1", Subject = "Meeting", StartTime =
new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015, 11,
10, 11, 00, 00), Description = "", AllDay = false, Recurrence = false,
RecurrenceRule = "" });
<!-- Datasource for Grouping Resources -->
List<String> Group = new List<String>();
Group.Add("Owners");
<!-- Datasource for Resources -->
List<ResourceFields> Resources = new List<ResourceFields>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Resources.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398" });
Resources.Add(new ResourceFields { Id = "3", Text = "Steven", Color =
"#56ca85" });
Resources.Add(new ResourceFields { Id = "5", Text = "Michael", Color =
"#51a0ed" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Group(gr => { gr.Resources(Group); })
.Resources(res => {
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Resources).Text("Text").Id("Id").Color("Color")).Add();
})
.ShowLocationField(true)
.CategorizeSettings(cat => cat.Enable(true))
.PrioritySettings(pri => pri.Enable(true))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.StartTimeZone("StartTimeZone")
.EndTime("EndTime")
.EndTimeZone("EndTimeZone")
.Description("Description")
.Location("Location")
.AllDay("AllDay")
.Priority("Priority")
.Categorize("Categorize")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.RecurrenceId("RecurrenceId")
.RecurrenceExDate("RecurrenceExDate")
.ResourceFields("OwnerId"))
)
```

Appointment Field Validation

It is possible to validate the required fields of the appointment window from client-side before submitting it, by adding appropriate validation rules to each of the fields. The appointment fields have been extended to accept both String and object type values. Therefore, in order to perform validations, it is necessary to specify object values for the appointment fields.

Refer the appointment fields specified with validation rules from the following code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for CategorizeSettings -->
List<CategorizeSettings> CategorizeValue = new
List<CategorizeSettings>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
CategorizeValue.Add(new CategorizeSettings { Text = "Blue Category", Id =
"1", Color = "#43b496", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Green Category", Id
= "2", Color = "#7f993e", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Orange Category", Id
= "3", Color = "#cc8638", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Purple Category", Id
= "4", Color = "#ab54a0", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Red Category", Id =
"5", Color = "#dd654e", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Yellow Category", Id
= "6", Color = "#d0af2b", FontColor = "#ffffff" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowLocationField(true)
.CategorizeSettings(fields =>
fields.Datasource(CategorizeValue).Enable(true).AllowMultiple(false).Id("
Id").Text("Text").Color("Color").FontColor("FontColor"))
.AppointmentSettings(fields => fields
.Id("Id")
.Subject(subject => subject.Field("Subject").ValidationRules(v =>
v.AddRule("required", true)))
.Location(location => location.Field("Location").ValidationRules(v =>
v.AddRule("required", true).AddRule("customRule", "/^[a-zA-Z0-9- ]*$/")))
.StartTime("StartTime")
.EndTime("EndTime")
.Description(des => des.Field("Description").ValidationRules(v =>
v.AddRule("required", true).AddRule("minlength", 5).AddRule("maxlength",
500)))
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.Categorize(categorize =>
categorize.Field("Categorize").ValidationRules(v => v.AddRule("required",
true))))
)
```

HTML

```
<script src="@Url.Content("~/Scripts/jquery.validate.min.js")"></script>
<script
src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"></scri
pt>
<script type="text/javascript">
$(function () {
$.validator.addMethod("customRule", function (value, element, options) {
var expression = /^[a-zA-Z0-9- ]*$/;
return expression.test(value);
}, "Special character(s) not allowed in Location field");
});
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Binding to Simple List Data Array

Scheduler supports binding the appointment data to it through the list of appointment object collection as depicted in the below code example.

Example - Array of List Data Binding

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Binding Remote Data Service

The appointment data can be bound to the Scheduler through Odata remote services, by configuring the service URL to the Schedule dataSource API.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields =>
fields.Datasource("http://mvc.syncfusion.com/OdataServices/Northwnd.svc/"
).Query("ej.Query().from('Events').take(10)")
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

OData V4

The OData v4 is an improved version of OData protocols. Scheduler supports retrieving and consuming appointment data from OData v4 services, just similar to the other remote services.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.EnableLoadOnDemand(true)
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("http://services.odata.org/V4/Northwind/Northwind.svc/Orde
rs/").Adaptor(AdaptorType.ODataV4Adaptor))
.Id("Id")
.Subject("ShipName")
.StartTime("OrderDate")
.EndTime("RequiredDate")
.Description("ShipAddress"))
)
```

WebAPI Binding

The Schedule appointment data can also be bound through the Web API services. It is a programmatic interface to define the request and response messages system that is mostly exposed in **JSON** or **XML**.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/OdataServices/api/ScheduleData/
").CrossDomain(true))
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

The server-side code to retrieve the appointments are as follows.

C#

```
// To retrieve the appointments from database and bind it to Scheduler
public IEnumerable<Event> GetData(String CurrentDate, String CurrentView,
String CurrentAction)
{
return new NORTHWNDEntities().Events.ToList();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Data Binding using OLEDB

The appointment data can also be bound to the Scheduler using OLEDB database as depicted below.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("Home/GetData").CrudURL("Home/Batch").Adaptor("UrlAdaptor"
))
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.StartTimeZone("StartTimeZone")
.EndTimeZone("EndTimeZone")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

The server-side controller code to retrieve and bind the appointment data to Scheduler are as follows. Also, define a class with all the required appointment fields as depicted in the below code example.

C#

```
// Define a class with all appointment fields
public class ScheduleData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Boolean AllDay { get; set; }
public Boolean Recurrence { get; set; }
public string RecurrenceRule { get; set; }
public string StartTimeZone { get; set; }
public string EndTimeZone { get; set; }
public string Description { get; set; }
}
// To retrieve the appointments from database and bind it to Scheduler
public JsonResult GetData()
{
// Mention your own dataSource to be used here.
string stringAccessConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|/ScheduleDb.MDB";
DataSet myDataSet = new DataSet();
OleDbConnection myAccessConn = new OleDbConnection(stringAccessConn);
OleDbCommand myAccessCommand = new OleDbCommand("SELECT * FROM
DefaultSchedule", myAccessConn);
OleDbDataAdapter myDataAdapter = new OleDbDataAdapter(myAccessCommand);
myAccessConn.Open();
myDataAdapter.Fill(myDataSet, "DefaultSchedule");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
List<ScheduleData> datasource = new List<ScheduleData>();
datasource = myDataSet.Tables[0].AsEnumerable().Select(dataRow => new
ScheduleData { Id = dataRow.Field<int>("Id"), Subject =
dataRow.Field<string>("Subject"), StartTime =
dataRow.Field<DateTime>("StartTime"), EndTime =
dataRow.Field<DateTime>("EndTime"), AllDay =
dataRow.Field<bool>("AllDay"), Recurrence =
dataRow.Field<bool>("Recurrence"), RecurrenceRule =
dataRow.Field<string>("RecurrenceRule"), Description =
dataRow.Field<string>("Description"), StartTimeZone =
dataRow.Field<string>("StartTimeZone"), EndTimeZone =
dataRow.Field<string>("EndTimeZone") }).ToList();
myAccessConn.Close();
return Json(datasource, JsonRequestBehavior.AllowGet);
}
```

The controller code to handle the CRUD operation are as follows.

C#

```
public JsonResult Batch(EditParams param)
{
// Mention your own dataSource to be used here.
string stringAccessConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=|DataDirectory|/ScheduleDb.MDB";
if (param.action == "insert" || (param.action == "batch" && param.added
!= null))  // this block of code will execute while inserting the
appointments
{
var value = param.action == "insert" ? param.value : param.added[0];
using (OleDbConnection myCon = new OleDbConnection(stringAccessConn))
{
OleDbCommand cmd = new OleDbCommand();
cmd.CommandType = CommandType.Text;
cmd.CommandText = "INSERT INTO
DefaultSchedule(Subject,StartTime,EndTime,AllDay,Recurrence,RecurrenceRul
e,Description,StartTimeZone,EndTimeZone) VALUES
(@Subject,@StartTime,@EndTime,@AllDay,@Recurrence,@RecurrenceRule,@Descri
ption,@StartTimeZone,@EndTimeZone)";
if (string.IsNullOrEmpty(value.Subject))
cmd.Parameters.AddWithValue("@Subject", DBNull.Value);
else
cmd.Parameters.AddWithValue("@Subject", value.Subject);
cmd.Parameters.AddWithValue("@StartTime", value.StartTime);
cmd.Parameters.AddWithValue("@EndTime", value.EndTime);
cmd.Parameters.AddWithValue("@AllDay", value.AllDay);
cmd.Parameters.AddWithValue("@Recurrence", value.Recurrence);
if (string.IsNullOrEmpty(value.RecurrenceRule))
cmd.Parameters.AddWithValue("@RecurrenceRule", DBNull.Value);
else
cmd.Parameters.AddWithValue("@RecurrenceRule", value.RecurrenceRule);
if (string.IsNullOrEmpty(value.Description))
cmd.Parameters.AddWithValue("@Description", DBNull.Value);
else
cmd.Parameters.AddWithValue("@Description", value.Description);
if (string.IsNullOrEmpty(value.StartTimeZone))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
cmd.Parameters.AddWithValue("@StartTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@StartTimeZone", value.StartTimeZone);
if (string.IsNullOrEmpty(value.EndTimeZone))
cmd.Parameters.AddWithValue("@EndTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@EndTimeZone", value.EndTimeZone);
cmd.Connection = myCon;
myCon.Open();
cmd.ExecuteNonQuery();
myCon.Close();
}
}
if (param.action == "remove" || param.deleted != null)  // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
using (OleDbConnection myCon = new OleDbConnection(stringAccessConn))
{
myCon.Open();
OleDbCommand cmd = new OleDbCommand("DELETE FROM DefaultSchedule WHERE Id
= @Key", myCon);
cmd.Parameters.AddWithValue("@Key", param.key);
cmd.ExecuteNonQuery();
myCon.Close();
}
}
else
{
foreach (var apps in param.deleted)
{
using (OleDbConnection myCon = new OleDbConnection(stringAccessConn))
{
myCon.Open();
OleDbCommand cmd = new OleDbCommand("DELETE FROM DefaultSchedule WHERE Id
= @Key", myCon);
cmd.Parameters.AddWithValue("@Key", apps.Id);
cmd.ExecuteNonQuery();
myCon.Close();
}
}
}
}
if ((param.action == "batch" && param.changed != null) || param.action ==
"update")   // this block of code will execute while updating the
appointment
{
var value = param.action == "update" ? param.value : param.changed[0];
using (OleDbConnection myCon = new OleDbConnection(stringAccessConn))
{
myCon.Open();
OleDbCommand cmd = new OleDbCommand("UPDATE DefaultSchedule SET
Subject=@Subject,StartTime=@StartTime,EndTime=@EndTime,AllDay=@AllDay,Rec
urrence=@Recurrence,RecurrenceRule=@RecurrenceRule,Description=@Descripti
on,StartTimeZone=@StartTimeZone,EndTimeZone=@EndTimeZone  WHERE Id =
@Key", myCon);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (string.IsNullOrEmpty(value.Subject))
cmd.Parameters.AddWithValue("@Subject", DBNull.Value);
else
cmd.Parameters.AddWithValue("@Subject", value.Subject);
cmd.Parameters.AddWithValue("@StartTime", value.StartTime);
cmd.Parameters.AddWithValue("@EndTime", value.EndTime);
cmd.Parameters.AddWithValue("@AllDay", value.AllDay);
cmd.Parameters.AddWithValue("@Recurrence", value.Recurrence);
if (string.IsNullOrEmpty(value.RecurrenceRule))
cmd.Parameters.AddWithValue("@RecurrenceRule", DBNull.Value);
else
cmd.Parameters.AddWithValue("@RecurrenceRule", value.RecurrenceRule);
if (string.IsNullOrEmpty(value.Description))
cmd.Parameters.AddWithValue("@Description", DBNull.Value);
else
cmd.Parameters.AddWithValue("@Description", value.Description);
if (string.IsNullOrEmpty(value.StartTimeZone))
cmd.Parameters.AddWithValue("@StartTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@StartTimeZone", value.StartTimeZone);
if (string.IsNullOrEmpty(value.EndTimeZone))
cmd.Parameters.AddWithValue("@EndTimeZone", DBNull.Value);
else
cmd.Parameters.AddWithValue("@EndTimeZone", value.EndTimeZone);
cmd.Parameters.AddWithValue("@Key", value.Id);
cmd.ExecuteNonQuery();
myCon.Close();
}
}
OleDbConnection myAccessConn = new OleDbConnection(stringAccessConn);
OleDbCommand myAccessCommand = new OleDbCommand("SELECT * FROM
DefaultSchedule", myAccessConn);
OleDbDataAdapter myDataAdapter = new OleDbDataAdapter(myAccessCommand);
DataSet myDataSet = new DataSet();
myAccessConn.Open();
myDataAdapter.Fill(myDataSet, "DefaultSchedule");
List<ScheduleData> datasource = new List<ScheduleData>();
datasource = myDataSet.Tables[0].AsEnumerable().Select(dataRow => new
ScheduleData { Id = dataRow.Field<int>("Id"), Subject =
dataRow.Field<string>("Subject"), StartTime =
dataRow.Field<DateTime>("StartTime"), EndTime =
dataRow.Field<DateTime>("EndTime"), AllDay =
dataRow.Field<bool>("AllDay"), Recurrence =
dataRow.Field<bool>("Recurrence"), RecurrenceRule =
dataRow.Field<string>("RecurrenceRule"), Description =
dataRow.Field<string>("Description"), StartTimeZone =
dataRow.Field<string>("StartTimeZone"), EndTimeZone =
dataRow.Field<string>("EndTimeZone") }).ToList();
myAccessConn.Close();
return Json(datasource, JsonRequestBehavior.AllowGet);
}
// Class definition for EditParams to be used as parameter in the above
Crud method for receiving the object value in it.
public class EditParams
{
public string key { get; set; }
public string action { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public List<ScheduleData> added { get; set; }
public List<ScheduleData> changed { get; set; }
public List<ScheduleData> deleted { get; set; }
public ScheduleData value { get; set; }
}
```

Controller Action Binding with CRUD operation

Scheduler supports binding appointment data from the controller by using various available options of DataSource property like URL, BatchURL, InsertURL, UpdateURL, RemoveURL and Adaptor.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("Home/GetData").BatchURL("Home/Crud").InsertURL("Home/add"
).UpdateURL("Home/update").RemoveURL("Home/remove").Adaptor(AdaptorType.U
rlAdaptor))
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

The server-side controller code to handle the CRUD operations are as follows.

C#

```
// To initially bind the appointments with Scheduler
public JsonResult GetData()
{
// ScheduleDataDataContext is a LINQ-to-SQL data class name that is
defined in the .dbml file to access the tables from the database
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(100);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers while saving a new appointment through quick window
public JsonResult add(Appointment value)
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
int intMax = db.Appointments.ToList().Count > 0 ?
db.Appointments.ToList().Max(p => p.Id) : 1;
Appointment appoint = new Appointment()
{
Id = intMax + 1,
StartTime = value.StartTime,
EndTime = value.EndTime,
Subject = value.Subject,
Description = value.Description,
OwnerId = value.OwnerId,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Recurrence = value.Recurrence,
AllDay = value.AllDay,
RecurrenceRule = value.RecurrenceRule
};
db.Appointments.InsertOnSubmit(appoint);
db.SubmitChanges();
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(500);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers while editing/dragging/resizing the existing appointment
public JsonResult update(Appointment value)
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
var filterData = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(value.Id));
Appointment appoint = db.Appointments.Single(A => A.Id ==
Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
appoint.StartTime = startTime;
appoint.EndTime = endTime;
appoint.Subject = value.Subject;
appoint.Description = value.Description;
appoint.OwnerId = value.OwnerId;
appoint.Recurrence = Convert.ToByte(value.Recurrence);
appoint.AllDay = value.AllDay;
appoint.RecurrenceRule = value.RecurrenceRule;
}
db.SubmitChanges();
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(500);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers when an appointment is deleted
public JsonResult remove(string key)
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
Appointment app = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(key)).FirstOrDefault();
if (app != null) db.Appointments.DeleteOnSubmit(app);
db.SubmitChanges();
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(500);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Triggers for any of the Scheduler CRUD operation
public JsonResult Crud(EditParams param)
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
if (param.action == "insert" || (param.action == "batch" && param.added
!= null))  // this block of code will execute while inserting the
appointments
{
var value = param.action == "insert" ? param.value : param.added[0];
int intMax = db.Appointments.ToList().Count > 0 ?
db.Appointments.ToList().Max(p => p.Id) : 1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DateTime endTime = Convert.ToDateTime(value.EndTime);
Appointment appoint = new Appointment()
{
Id = intMax + 1,
StartTime = startTime,
EndTime = endTime,
Subject = value.Subject,
Description = value.Description,
OwnerId = value.OwnerId,
Recurrence = value.Recurrence,
AllDay = value.AllDay,
RecurrenceRule = value.RecurrenceRule
};
db.Appointments.InsertOnSubmit(appoint);
db.SubmitChanges();
}
else if (param.action == "remove")  // this block of code will execute
while removing the appointment
{
Appointment app = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(param.key)).FirstOrDefault();
if (app != null) db.Appointments.DeleteOnSubmit(app);
db.SubmitChanges();
}
else if ((param.action == "batch" && param.changed != null) ||
param.action == "update")   // this block of code will execute while
updating the appointment
{
var value = param.action == "update" ? param.value : param.changed[0];
var filterData = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
Appointment appoint = db.Appointments.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appoint.StartTime = startTime;
appoint.EndTime = endTime;
appoint.Subject = value.Subject;
appoint.Description = value.Description;
appoint.OwnerId = value.OwnerId;
appoint.Recurrence = Convert.ToByte(value.Recurrence);
appoint.AllDay = value.AllDay;
appoint.RecurrenceRule = value.RecurrenceRule;
}
db.SubmitChanges();
}
IEnumerable data = new ScheduleDataDataContext().Appointments.Take(500);
return Json(data, JsonRequestBehavior.AllowGet);
}
// Class definition for EditParams to be used as parameter in the above
Crud method for receiving the object value in it.
public class EditParams
{
public string key { get; set; }
public string action { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public List<Appointment> added { get; set; }
public List<Appointment> changed { get; set; }
public Appointment value { get; set; }
}
```

Loading Data on Demand

Load on demand feature allows the Scheduler to retrieve only the filtered appointment data (for the current Scheduler date range) from the service/database during **loading time**, and that too only for the current Scheduler view. There are 3 parameters made available on the server-side namely **CurrentDate**, **CurrentView** and **CurrentAction** through which only the necessary appointments are retrieved from the database and then assigned to the Scheduler dataSource. With this kind of Scheduler action, consuming only lesser data will reduce the usage of network bandwidth size and loading time.

The **EnableLoadOnDemand** property is used to enable or disable the load on demand functionality of the schedule.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.EnableLoadOnDemand(true)
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/OdataServices/api/ScheduleData/
").CrossDomain(true))
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

The server-side code to load the Scheduler data on demand is as follows.

C#

```
// retrieve the appointments based on the current date.
public IEnumerable<Event> GetData(String CurrentDate, String CurrentView,
String CurrentAction)
{
var dateString = Regex.Match(CurrentDate.ToString(),
@"^(\w+\b.*?){4}").ToString();
string format = "ddd MMM dd yyyy";
DateTime dateTimeValue;
try
{
dateTimeValue = DateTime.ParseExact(dateString, format,
CultureInfo.InvariantCulture);
}
catch(FormatException)
{
```

```
var dateSplit = CurrentDate.Split(' ');//For IE<=10 Fri Mar 20 11:00:22
UTC+0530 2015
if (dateSplit[2].Length == 1) dateSplit[2] = string.Concat("0",
dateSplit[2]);
dateString = string.Concat(dateSplit[0], ' ', dateSplit[1], ' ',
dateSplit[2], ' ', dateSplit[dateSplit.Length - 1]);
dateTimeValue = DateTime.ParseExact(dateString, format,
CultureInfo.InvariantCulture);
}
// AppointmentReposit is a user-defined class within which the
FilterAppointment method is defined.
AppointmentReposit rep = new AppointmentReposit();
var data = rep.FilterAppointment(dateTimeValue, CurrentAction,
CurrentView);
return data;
}
// Method to filter the appointments based on the date range
public List<Event> FilterAppointment(DateTime CurrentDate, String
CurrentAction, String CurrentView)
{
DateTime CurrDate = Convert.ToDateTime(CurrentDate);
DateTime StartDate = FirstWeekDate(CurrDate.Date);
DateTime EndDate = FirstWeekDate(CurrDate.Date);
List<Event> appointmentList = new NORTHWNDEntities().Events.ToList();
switch (CurrentView)
{
case "day":
StartDate = CurrentDate;
EndDate = CurrentDate;
break;
case "week":
EndDate = EndDate.AddDays(7);
break;
case "workweek":
EndDate = EndDate.AddDays(5);
break;
case "month":
StartDate = CurrDate.Date.AddDays(-CurrDate.Day + 1);
EndDate = StartDate.AddMonths(1);
break;
}
appointmentList = new NORTHWNDEntities().Events.ToList().Where(app =>
((Convert.ToDateTime(app.StartTime).Date >=
Convert.ToDateTime(StartDate.Date)) &&
(Convert.ToDateTime(app.EndTime).Date <=
Convert.ToDateTime(EndDate.Date)))).ToList();
return appointmentList;
}
internal static DateTime FirstWeekDate(DateTime CurrentDate)
{
try
{
DateTime FirstDayOfWeek = CurrentDate;
DayOfWeek WeekDay = FirstDayOfWeek.DayOfWeek;
switch (WeekDay)
{
case DayOfWeek.Sunday:
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
break;
case DayOfWeek.Monday:
FirstDayOfWeek = FirstDayOfWeek.AddDays(-1);
break;
case DayOfWeek.Tuesday:
FirstDayOfWeek = FirstDayOfWeek.AddDays(-2);
break;
case DayOfWeek.Wednesday:
FirstDayOfWeek = FirstDayOfWeek.AddDays(-3);
break;
case DayOfWeek.Thursday:
FirstDayOfWeek = FirstDayOfWeek.AddDays(-4);
break;
case DayOfWeek.Friday:
FirstDayOfWeek = FirstDayOfWeek.AddDays(-5);
break;
case DayOfWeek.Saturday:
FirstDayOfWeek = FirstDayOfWeek.AddDays(-6);
break;
}
return (FirstDayOfWeek);
}
catch
{
return DateTime.Now;
}
}
```

Entity Framework Data Binding

The appointment data can be bound to the Scheduler through Entity Framework which supports Entity Data Model(EDM) defining the data at conceptual level. To know more on how to create and use the Entity Data Model, refer here.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

C#

```
ScheduleControlDataEntities db = new ScheduleControlDataEntities(); //
Create an object of the entity class to access its underlying tables
within the controller page.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult Index()
{
return View();
}
public JsonResult GetData()
{
IEnumerable data = new
ScheduleControlDataEntities().Appointments.Take(100);
return Json(data, JsonRequestBehavior.AllowGet);
}
public JsonResult Batch(EditParams param)
{
// this block of code will execute while inserting the appointments
if (param.action == "insert" || (param.action == "batch" && param.added
!= null))
{
var value = param.action == "insert" ? param.value : param.added[0];
int intMax = db.Appointments.ToList().Count > 0 ?
db.Appointments.ToList().Max(p => p.Id) : 1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
var currentTimeZone = TimeZone.CurrentTimeZone;
Appointment appoint = new Appointment()
{
Id = value.Id,
StartTime = startTime,
EndTime = endTime,
StartTimeZone=value.StartTimeZone,
EndTimeZone=value.EndTimeZone,
Subject = value.Subject,
Description = value.Description,
Recurrence = value.Recurrence,
RoomId = value.RoomId,
OwnerId = value.OwnerId,
CategoryId = value.CategoryId,
AllDay = value.AllDay,
RecurrenceRule = value.RecurrenceRule,
};
db.Appointments.Add(appoint);
db.SaveChanges();
}
if (param.action == "remove" || param.deleted != null)
// this block of code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
Appointment app = db.Appointments.Where(c => c.Id ==
key).FirstOrDefault();
if (app != null) db.Appointments.Remove(app);
}
else
{
foreach (var apps in param.deleted)
{
Appointment app = db.Appointments.Where(c => c.Id ==
apps.Id).FirstOrDefault();
```

```
if (apps != null) db.Appointments.Remove(app);
}
}
db.SaveChanges();
}
if ((param.action == "batch" && param.changed != null) || param.action ==
"update")
{
// this block of code will execute while updating the appointment
var value = param.action == "update" ? param.value : param.changed[0];
var filterData = db.Appointments.Where(c => c.Id ==
Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
Appointment appoint = db.Appointments.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appoint.StartTime = startTime;
appoint.EndTime = endTime;
appoint.Subject = value.Subject;
appoint.StartTimeZone = value.StartTimeZone;
appoint.EndTimeZone = value.EndTimeZone;
appoint.Description = value.Description;
appoint.OwnerId = value.OwnerId;
appoint.Recurrence = Convert.ToByte(value.Recurrence);
appoint.RoomId = value.RoomId;
appoint.AllDay = value.AllDay;
appoint.RecurrenceRule = value.RecurrenceRule;
}
db.SaveChanges();
}
IEnumerable data = new
ScheduleControlDataEntities().Appointments.Take(500); //
new.Appointment.Take(5);
return Json(data, JsonRequestBehavior.AllowGet);
}
```

SQL Data Binding

Binding SQL data to Scheduler is quite simple and also it supports the complete CRUD operation to be performed on the appointment data easily.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("/Home/GetData").CrudURL("/Home/Batch").Adaptor("UrlAdapto
r"))
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

C#

```
// Mention the correct path of the database to connect. Also ensure,
whether the provided database already exists.
SqlConnection connection = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\ScheduleDa
tabase.mdf;Integrated Security=True");
public ActionResult Index()
{
return View();
}
public JsonResult GetData()
{
using (connection)
{
DataTable myDataSet = new DataTable();
SqlConnection myAccessConn = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\ScheduleCo
ntrolData.mdf;Integrated Security=True");
SqlCommand myAccessCommand = new SqlCommand("SELECT * FROM ScheduleData",
myAccessConn);
SqlDataAdapter myDataAdapter = new SqlDataAdapter(myAccessCommand);
myAccessConn.Open();
myDataAdapter.Fill(myDataSet);
List<DataRow> datasource = myDataSet.AsEnumerable().ToList();
List<ScheduleDatabase> scheduleList = new List<ScheduleDatabase>();
for (int i = 0; i < datasource.Count; i++)
{
ScheduleDatabase datum = new ScheduleDatabase();
datum.Id = Convert.ToInt32(datasource[i]["Id"]);
datum.Subject = datasource[i]["Subject"].ToString();
datum.StartTime = Convert.ToDateTime(datasource[i]["StartTime"]);
datum.EndTime = Convert.ToDateTime(datasource[i]["EndTime"]);
datum.AllDay = Convert.ToBoolean(datasource[i]["AllDay"]);
datum.Recurrence = Convert.ToBoolean(datasource[i]["Recurrence"]);
datum.RecurrenceRule = datasource[i]["RecurrenceRule"].ToString();
datum.StartTimeZone = datasource[i]["StartTimeZone"].ToString();
datum.EndTimeZone = datasource[i]["EndTimeZone"].ToString();
datum.Description = datasource[i]["Description"].ToString();
scheduleList.Add(datum);
}
myAccessConn.Close();
return Json(scheduleList, JsonRequestBehavior.AllowGet);
}
}
public class ScheduleDatabase
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Boolean AllDay { get; set; }
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public Boolean Recurrence { get; set; }
public string RecurrenceRule { get; set; }
public string StartTimeZone { get; set; }
public string EndTimeZone { get; set; }
public string Description { get; set; }
}
// To perform other CRUD operations
public JsonResult Batch(EditParams param)
{
if (param.action == "insert" || (param.action == "batch" && param.added
!= null))
{
var value = param.action == "insert" ? param.value : param.added[0];
connection.Open();
sqlCommand = "insert into ScheduleData
(Id,EndTime,Recurrence,StartTime,Subject,AllDay,RecurrenceRule,StartTimeZ
one,EndTimeZone,Description) values(" + value.Id + ",'" + value.EndTime +
"','" + value.Recurrence + "','" + value.StartTime + "','" +
value.Subject + "','" + value.AllDay + "','" + value.RecurrenceRule
+"','" + value.StartTimeZone + "','" + value.EndTimeZone + "','" +
value.Description + "')";
adapter1.InsertCommand = new SqlCommand(sqlCommand, connection);
adapter1.InsertCommand.ExecuteNonQuery();
connection.Close();
}
if (param.action == "remove" || param.deleted != null)
{
connection.Open();
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
SqlCommand delete = new SqlCommand("delete from ScheduleData where Id=" +
key + "", connection);
delete.ExecuteNonQuery();
}
else
{
foreach (var apps in param.deleted)
{
SqlCommand delete = new SqlCommand("delete from ScheduleData where Id=" +
apps.Id + "", connection);
delete.ExecuteNonQuery();
}
}
connection.Close();
}
if ((param.action == "batch" && param.changed != null) || param.action ==
"update")
{
connection.Open();
var value = param.action == "update" ? param.value : param.changed[0];
sqlCommand = "update scheduleData set EndTime='" + value.EndTime +
"',Recurrence='" + value.Recurrence + "',StartTime='" + value.StartTime +
"',Subject='" + value.Subject + "',AllDay='" + value.AllDay +
"',RecurrenceRule='" + value.RecurrenceRule + "',StartTimeZone = '" +
value.StartTimeZone + "',EndTimeZone = '" + value.EndTimeZone + "'where
Id='" + value.Id + "'";
```

```
adapter1.UpdateCommand = new SqlCommand(sqlCommand, connection);
adapter1.UpdateCommand.ExecuteNonQuery();
connection.Close();
}
return GetData();
}
```

Views

The number of days and its associated appointments are usually grouped together in Scheduler to organize different views. The available view options in Scheduler are as follows,

- Day
- Week
- Workweek
- Month
- Custom View
- Agenda
- Timeline View

Usually these view options are displayed as a toolbar in the date-header section of the Schedule control. The items within the views toolbar can be added/removed based on the value passed to the Views property.

By default, the Schedule control's active view is **Week** view. Also, it is possible to change the active view of the Scheduler by setting CurrentView option with the required view name as depicted below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@{List<string> view = new List<string>() { "Day", "Week", "WorkWeek",
"Month" };}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.Views(view)
.CurrentView(CurrentView.Week)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

Note: The **CurrentView** property accepts Enum value of CurrentView.<ViewName>.

Day

It represents a single day Scheduler view (single date display) with all its related appointments.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentView(CurrentView.Day)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Week

It's a view displaying a count of 7 days (from Sunday to Saturday) with all its related appointments. The first day of the week can be changed using the FirstDayOfWeek API which accepts either the integer (Sunday=0, Monday=1, Tuesday=2, etc) or string ("Sunday", "Monday", etc) or DayOfWeek enum type value. The default value of this **FirstDayOfWeek** depends on the current culture (language) used in the Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.CurrentView(CurrentView.Week)
.FirstDayOfWeek(DayOfWeek.Monday)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Work Week

Work week view displays the working days of the week (count of 5 days) and its associated appointments. It is also possible to customize the days to be displayed in the work week view using WorkWeek API which accepts the string array such as ["Monday", "Tuesday", "Wednesday", "Thursday" and "Friday"]. By default, it renders from Monday to Friday (5 days).

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@{List<string> workWeek = new List<string>() { "Monday", "Tuesday"
,"Wednesday", "Friday", "Saturday" };}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentView(CurrentView.Workweek)
.WorkWeek(workWeek)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Month

Month view displays the entire days of a particular month and all its related appointments. An alternative way to navigate to a particular date in a day view directly from Month view, clicking on the appropriate month cell date header will do so. If the week date range column is clicked, it will navigate to the corresponding week view.

Copyright © 2001 - 2020 Syncfusion Inc.

The next and previous month date cells in the Month view can be shown/hidden on the Scheduler using ShowNextPrevMonth property by setting it to *false*.

For example – To set the Month view as current view in Scheduler and to hide the other month days in it, refer the below code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentView(CurrentView.Month)
.ShowNextPrevMonth(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: An appointment directly created in Month view will be considered as an all-day appointment.

Customizing number of days (Custom View)

The Scheduler can be displayed with the user-specified date ranges, such as 4 days or any specific date ranges instead of default view options, by making use of the RenderDates property. This property includes two sub properties namely **Start** and **End**, which accepts the date object or date value in string format to specify the date range.

To display the custom view option in the toolbar-like view options in the scheduler header area, add the CustomView value to the views property array collection as shown below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{List<string> view = new List<string>() { "Day", "Week", "WorkWeek",
"Month", "CustomView" };}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 5, 2))
.Views(view)
.CurrentView(CurrentView.CustomView)
.RenderDates(dt => dt.Start("5/01/2014").End("5/06/2014"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

When the date difference between the provided start and end date is greater than 7, then the month-like view will get displayed in Vertical Scheduler mode - whereas with the date difference less than 7 days displays the Scheduler with exact count of the specified days.

Note: When the CurrentDate property of Scheduler is set with a date, that lies beyond the specified custom date range - then the Scheduler navigates to the current date with the mentioned date differences.

Agenda

This View option lists out the appointments in a grid-like view for the next 7 days by default from the current date. The count of the number of appointments to be listed in this view can be customized using the AgendaViewSettings.DaysInAgenda.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.CurrentView(CurrentView.Agenda)
.AgendaViewSettings(agendaViewSettings =>
agendaViewSettings.DaysInAgenda(5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: In Agenda view, the templates can be applied for the date and time columns. Also, the template passed through the AppointmentTemplateId will gets applied to the event column in Agenda view.

Restriction on View Navigation

It is possible to restrict the users to display only the specific list of views in the Schedule header section and also not to navigate to other views that are not listed.

For example, if the views property is set only with Month view – then the Schedule header section displays only the Month option in the view toolbar and also other additional available actions like navigating to day/week view on clicking the month header dates and week date-range is stopped.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@{List<string> views = new List<string>() { "Month" };}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.Views(views)
.CurrentDate(new DateTime(2014, 6, 2))
.CurrentView(CurrentView.Agenda)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: Even though Week view is the active view in Scheduler by default, if it is not listed in the views collection – then the first listed option in the views collection will be taken as current view of the Scheduler.

Timeline View

Timeline view displays the day, time and its associated events horizontally arranged from left to right. By default, Scheduler renders in vertical mode and it can be changed to the timeline mode using Orientation property which accepts both the string and Orientation enum value.

Copyright © 2001 - 2020 Syncfusion Inc.

All the applicable features in Vertical mode works similar with Timeline mode (Horizontal) and only the visualization of the layout changes based on the orientation.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.Orientation(Orientation.Horizontal)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Working with Appointments

An appointment represents a certain time interval in a schedule cell depicting a plan made for the specified time interval.

Appointment Types

The types of appointments available within Scheduler can be categorized as follows

Normal

Represents an appointment that is created for a certain time interval in one or more number of days. If the normal appointment is created for more than 24 hours, then those longer appointments will be rendered on the all-day row.

Note: If the normal appointment is to be created for two days (say from November 25, 2015 – 11.00 PM to November 26, 2015 2.00 AM) but less than 24 hour time interval, then the appointment is split into two partitions and will be displayed appropriately on both the days.

All-Day

Represents an appointment that is created for an entire day such as holiday events. It renders separately in All-day row, a separate place for all-day appointments. In Timeline (horizontal) view, all-day appointment renders in the usual work cells, as no all-day cells are present in that view.

Note: An all-day row is normally visible on the Scheduler, as the ShowAllDayRow property is set to true by default.

Copyright © 2001 - 2020 Syncfusion Inc.

Recurrence

Represents an appointment that is created for a certain time interval that occurs repeatedly in a daily, weekly, monthly, yearly or every weekday basis at the same time interval based on the recurrence rule. The other available options and validations that can be performed on recurrence appointments can be referred from here.

CRUD operation

Appointments play a dynamic role within the Schedule control with which the users mostly interact. You can manipulate (add/edit/delete/drag/resize) the required appointments that reveals one of the main purpose of the Schedule control.

Add/Edit Appointments

The appointments can be added/edited in the Scheduler using any one of the following ways,

- Quick window
- Inline creation/editing
- Default appointment window
- Context menu
- Through programmatically

Quick Window

The Quick window usually pops out while single clicking on the Scheduler cells or appointments. It requires the user to enter the Subject to proceed with the appointment creation. It also includes an **Edit Appointment** option displayed at the bottom left corner – on selection which opens up the normal appointment window.

On single clicking the scheduler appointments, the pop-up that shows up contains the appointment information along with the other options that are listed below,

- Edit Appointment
- Edit Series (only for the recurrence appointments)
- Delete icon

The quick window option can be enabled/disabled by using `ShowQuickWindow` API, whereas its default value is set to **true**.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ShowQuickWindow(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: Select multiple cells either using mouse or keyboard access keys (shift+arrow keys) and press enter key, so that the quick window opens up for the selected date/time range.

Another way to disable the quick window option at dynamic time can be achieved through the CellClick and AppointmentClick events. The below code example shows the way to disable the quick appointment window only while clicking on the cells, but displays for appointments.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.ShowQuickWindow(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event => event.CellClick("onCellClick"))
)
```

JS

```
function onCellClick(args) {
args.cancel = true;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Inline Appointment Creation/Editing

Another easier way, for adding or editing the appointment's subject alone can be achieved using inline Add/Edit support. It allows the user to add and edit the appointments inline.

To get familiar with inline Add mode, single click on any of the Scheduler cells or press enter key on the selected cells. When the inline adding mode is ON, a text box will get created within the clicked Scheduler cells with a blinking cursor in it, requiring the user to enter the subject of an appointment. Once the subject is entered, the appointment will be saved on pressing the enter key.

To enable inline edit mode, single click on any of the existing appointment's subject, so that the user can edit the subject of that appointment. The edited subject of that appointment is then updated on pressing the enter key.

The inline option can be enabled/disabled on Scheduler by using the AllowInline API, whereas its default value is set to **false**.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AllowInline(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Enabling Inline Edit alone

It is possible to disable the inline appointment creation and enabling only the editing mode of inline by making use of the CellClick event. The below code example shows the way to disable the inline appointment creation while clicking on the cells, but appointments can be edited while clicking on it's subject.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AllowInline(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event => event.CellClick("onCellClick"))
)
```

JS

```
function onCellClick(args) {
args.cancel = true;
}
```

Default Appointment Window

The default appointment window is availed with options like

- Subject
- Start and End Time
- All-Day
- TimeZone (for both Start and End time)
- Repeat options
- Description

The other additional options available are listed below for which the appropriate API's are needed to be configured to display these options on the appointment window.

- Location (ShowLocationField)
- Priority (PrioritySettings)
- Categorize (CategorizeSettings)
- Resources

The appointments can be created by double-clicking the Scheduler cells across the required time slots, which makes the Create Appointment window to pop-up. The start and end time fields will get automatically populated, according to the time-slot selection. Clicking on the done button in an appointment window will create the appointment for the selected time cells.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Select multiple cells both using mouse or keyboard access keys (shift+arrow keys) and press Alt+N key, so that the default appointment window opens up for the selected date/time range with the Start and End time fields automatically filled in.

To prevent the display of default appointment window on double clicking the Scheduler cells, either the AppointmentWindowOpen or CellDoubleClick event can be used, within which the **args.cancel** needs to be set to true. This behavior is depicted in the below code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event =>
event.AppointmentWindowOpen("onAppointmentWindowOpen"))
)
```

JS

```
function onAppointmentWindowOpen(args) {
args.cancel = true;
}
```

Through Programmatically

You can add/edit the appointments dynamically through the public method saveAppointment. It accepts the JSON Object data (either a new or updated appointment object) as its argument.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event =>
event.AppointmentWindowOpen("onAppointmentWindowOpen"))
)
@Html.EJ().Button("add").Text("Add").ClientSideEvents(event =>
event.Click("addAppointment"))
```

JS

```
function addAppointment() {
//appointment details
var appointment = {
Subject: "Gym",
StartTime: new Date("2015/11/7 03:30 AM"),
EndTime: new Date("2015/11/7 04:30 AM")
};
//create the schedule object
var schObj = $("#Schedule1").data("ejSchedule");
//pass the JSON object in the public method
schObj.saveAppointment(appointment);
}
```

Delete Appointments

The appointments can be deleted in either of the following ways,

- Selecting an appointment and clicking the delete icon in the quick appointment window.
- Hovering the mouse over appointments and clicking on Inline delete option which is enabled by default for all the appointments.
- Selecting an appointment and pressing `Delete` key.
- Through Programmatically.

A pop-up with a confirmation message will get displayed before deleting an appointment, which can be either switched on/off using the API `ShowDeleteConfirmationDialog`. Also, the confirmation text in that pop-up can be customized as mentioned here.

For example, to localize only the delete confirmation message in the delete window -

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.ShowDeleteConfirmationDialog(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<script>
var deleteCustomizationMessage = {
// customize the confirmation message
DeleteConfirmation: "Do you want to delete this Event?",
// customize the delete window title
MouseOverDeleteTitle: "Delete Event",
// customize the recurrence delete window title
RecurrenceDeleteTitle: "Delete Repeat Event"
};
// Extend only the required changes to the original locale collection
$.extend(ej.Schedule.Locale["en-US"], deleteCustomizationMessage);
</script>
```

Note: All these CRUD operations on appointments (add/edit/delete) can also be done through the default Context Menu options **Add Appointment**, **Edit Appointment** and **Delete Appointment** which is available, when context menu settings is enabled within Scheduler.

Through Programmatically

You can delete the appointments dynamically using the method `deleteAppointment`, which accepts the Guid of the appointment or complete appointment data as its argument. The Guid is availed as one of the appointment element's attribute.

Example 1 - Using GUID

The below code example depicts the way to delete the appointments using GUID programmatically by calling the **deleteAppointment** function within the `AppointmentClick` event, which triggers whenever the user clicks on an appointment.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event =>
event.AppointmentClick("onAppointmentClick"))
)
```

JS

```
function onAppointmentClick(args) {
var schObj = $("#schedule").data("ejSchedule");
schObj.deleteAppointment(args.appointment.Guid);
}
```

Example 2 - Using Appointment object

The below code example depicts the way to delete the appointments using appointment data programmatically by calling the **deleteAppointment** function within the AppointmentClick event, which triggers whenever the user clicks on an appointment.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event =>
event.AppointmentClick("onAppointmentClick"))
)
```

JS

```
function onAppointmentClick(args) {
var schObj = $("#schedule").data("ejSchedule");
schObj.deleteAppointment(args.appointment);
}
```

Handling Appointment Actions

It is possible to define some specific actions to take place before the CRUD operation occurs on the Scheduler appointments through the following available client-side events,

- BeforeAppointmentCreate
- BeforeAppointmentChange
- BeforeAppointmentRemove

BeforeAppointmentCreate – Triggers before saving a new appointment.

BeforeAppointmentChange – Triggers when an appointment is edited and before it is being updated to the dataSource.

BeforeAppointmentRemove – Triggers before deleting an existing appointment.

To stop the save, edit and delete actions on the Scheduler appointments, following code example can be used.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event =>
event.BeforeAppointmentCreate("onAppointmentSave").BeforeAppointmentChang
e("onAppointmentEdit").BeforeAppointmentRemove("onAppointmentDelete"))
)
```

JS

```
function onAppointmentSave(args) {
args.cancel = true; // cancels the save action on appointments.
}
function onAppointmentEdit(args) {
args.cancel = true; // cancels the edit action on appointments.
}
function onAppointmentDelete(args) {
args.cancel = true; // cancels the delete action on appointments.
}
```

Read Only

An interaction with the appointments of the Scheduler can be enabled/disabled through the ReadOnly property. When the ReadOnly property is set to true, it is not possible to do any actions on the appointments, but you can navigate between the schedule dates, views and can also be able to see the appointment details in the quick window. By default, this property is set to false.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.ReadOnly(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

Note: When the ReadOnly property is set to true – double clicking the cells will open the appointment window filled with appointment details, which can be allowed to view but cannot be edited or saved.

Drag and Drop

The appointment time can be modified through the drag and drop behavior, by dragging and dropping it to the new location, so that the start time and end time of the appointment gets changed automatically. We can enable/disable the drag and drop functionality through the AllowDragAndDrop property. By default, it is set to true.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AllowDragAndDrop(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Handling Drag Actions Dynamically

The drag and drop functionality can be handled with the following three events,

- DragStart
- Drag
- DragStop

dragStart – Triggers when the appointments are started to drag from its source location.

drag – Triggers when the appointments are being dragged over.

dragStop – Triggers when the appointments are dropped on a destined location.

The following code example shows how to cancel the dragging functionality with the help of one of these events.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event=>event.DragStop("onDragStop"))
)
```

JS

```
function onDragStop(args) {
args.cancel = true; // cancels the drag action on appointments.
}
```

External Drag and Drop

It is possible to drag and drop the external items to and fro the Scheduler control. This action is handled through the property AppointmentDragArea which specifies the draggable area which states whether the appointments can be dragged outside of the control or within it.

The following code example lets you dragging and dropping external items to and from the Scheduler control.

HTML

```
<div class="row">
<div class="col-md-2">
<span class=""><b>Tutorials </b> </span>
@Html.EJ().TreeView("drag").Items(items => {
items.Add().Text("HTML").Id("HTML").Expanded(true).Children(child => {
child.Add().Text("Introduction");
child.Add().Text("Editors");
child.Add().Text("Styles");
child.Add().Text("Formatting");
child.Add().Text("Tables");
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).AllowDragAndDrop(true).AllowDropChild(false).AllowDropSibling(false).A
llowDragAndDropAcrossControl(true).Width("auto").ClientSideEvents(s =>
s.NodeDropped("onDropped").NodeDragStart("onNodeDrag"))
</div>
<div class="col-md-9">
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Owners -->
List<ResourceFields> resources = new List<ResourceFields>();
resources.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398" });
resources.Add(new ResourceFields { Id = "3", Text = "Steven", Color =
"#56ca95" });
resources.Add(new ResourceFields { Id = "5", Text = "Michael", Color =
"#51a0ed" });
<!-- Datasource for Grouping -->
List<String> Group = new List<String>();
Group.Add("Owners");
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.AppointmentDragArea("body")
.ShowCurrentTimeIndicator(false)
.CurrentDate(new DateTime(2015, 11, 5))
.Resources(res => {
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(resources).Text("Text").Id("Id").Color("Color")).Add();
})
.Group(gr => { gr.Resources(Group); })
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
</div>
</div>
<div id="customWindow" style="display: none">
<form id="custom">
<table width="100%" cellpadding="5">
<tbody>
<tr>
<td>Subject:</td>
<td colspan="2">
<input id="subject" type="text" value="" name="Subject" style="width:
100%" readonly />
</td>
</tr>
<tr>
```

```
<td>Description:</td>
<td colspan="2">
<textarea id="customDescription" name="Description" rows="3" cols="50"
style="width: 100%; resize: vertical"></textarea>
</td>
</tr>
<tr>
<td>StartTime:</td>
<td>
@Html.EJ().DateTimePicker("StartTime").Width("150px")
</td>
</tr>
<tr>
<td>EndTime:</td>
<td>
@Html.EJ().DateTimePicker("EndTime").Width("150px")
</td>
</tr>
<tr>
<td>Resource:</td>
<td colspan="2">
<input id="resource" type="text" value="" name="Resource" style="width:
100%" readonly />
</td>
</tr>
<tr style="display: none">
<td>ownerId:</td>
<td colspan="2">
<input id="ownerId" type="text" name="OwnerId" />
</td>
</tr>
</tbody>
</table>
</form>
<div>
<button type="submit" onclick="cancel()" id="buttonCancel"
style="float:right;margin-right:20px;margin-bottom:10px;">Cancel</button>
<button type="submit" onclick="save()" id="buttonSubmit"
style="float:right;margin-right:20px;margin-bottom:10px;">Save</button>
</div>
</div>
```

JS

```
$(function () {
$("#customWindow").ejDialog({
width: 600,
height: "auto",
position: { X: 400, Y: 200 },
showOnInit: false,
enableModal: true,
title: "Create Appointment",
enableResize: false,
allowKeyboardNavigation: false,
close: "clearFields"
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#buttonCancel").ejButton({ width: '85px' });
$("#buttonSubmit").ejButton({ width: '85px' });
});
function onNodeDrag(e) {
if (e.targetElementData.parentId == "") return false;
}
function onDropped(e) {
if ($(e.target).parents(".e-schedule").length != 0) {
var scheduleObj = $("#Schedule1").data("ejSchedule");
var result = scheduleObj.getSlotByElement($(e.target));
// set value to custom appointment window fields
$("#subject").val(e.droppedElementData.text);
$("#customDescription").val(e.droppedElementData.text);
$("#StartTime").ejDateTimePicker({ value: new Date(result.startTime) });
$("#EndTime").ejDateTimePicker({ value: new Date(result.endTime) });
$("#resource").val(result.resources.text);
$("#ownerId").val(result.resources.id);
$("#customWindow").ejDialog("open");
}
}
function save() {
var obj = {};
var formElement = $("#customWindow").find("#custom").get(0);
for (var index = 0; index < formElement.length; index++) {
var columnName = formElement[index].name, $element =
$(formElement[index]);
if (columnName != undefined) {
if (columnName == "Subject")
var value = formElement[index].value;
if (columnName == "Description")
value = formElement[index].value;
if (columnName == "StartTime")
value = new Date(formElement[index].value);
if (columnName == "EndTime")
value = new Date(formElement[index].value);
if (columnName == "OwnerId")
value = formElement[index].value;
if (columnName != "Resource")
obj[columnName] = value;
}
}
$("#customWindow").ejDialog("close");
var object = $("#Schedule1").data("ejSchedule");
object.saveAppointment(obj);
}
function cancel() {
$("#customWindow").ejDialog("close");
}
}
```

Resize

Resizing an appointment is another way to change its start and end time. Mouse hover on the appointments, so that the resizing handlers gets displayed on either sides of the appointment which allows resizing. The resizing functionality can be enabled/disabled by setting the `EnableAppointmentResize` property. By default it is set to `true`.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.EnableAppointmentResize(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Handling Resize Actions Dynamically

The appointment resizing functionality can be handled through the following three events,

- ResizeStart
- Resize
- ResizeStop

ResizeStart – Triggers when the appointments are started resizing from its original time.

Resize – Triggers when the appointment resizing is in progress.

ResizeStop – Triggers when the appointment resizing is done.

The following code example shows how to cancel the resizing functionality with the help of one of these events.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event=>event.ResizeStart("onResizeStart"))
)
```

JS

```
function onResizeStart(args) {
args.cancel = true; // Blocks the resize action on appointments.
}
```

Categorization

It allows to differentiate the appointments with various categorize options and individual colors. You can also denote the status of the appointments using this categorize option and can specify your own user-defined category collection. It is also possible to select multiple categorize for a single appointment.

Categorize Settings

The CategorizeSettings holds the below categorize related properties such as,

- Enable - It accepts true or false value, denoting whether to enable/disable the categorize option. Its default value is false.
- AllowMultiple – It enables or disables the multiple selection of categories for each appointments in the appointment window as well as in the context menu. Its default value is false.
- DataSource – Binds the categorize dataSource collection. This property should be assigned with the JSON data array collection or instance of ej.DataManger.

We have below 6 default values for Categorize dataSource collection.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for CategorizeSettings -->
List<CategorizeSettings> CategorizeValue = new
List<CategorizeSettings>();
CategorizeValue.Add(new CategorizeSettings { Text = "Blue Category", Id =
"1", Color = "#43b496", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Green Category", Id
= "2", Color = "#7f993e", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Orange Category", Id
= "3", Color = "#cc8638", FontColor = "#ffffff" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
CategorizeValue.Add(new CategorizeSettings { Text = "Purple Category", Id
= "4", Color = "#ab54a0", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Red Category", Id =
"5", Color = "#dd654e", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Yellow Category", Id
= "6", Color = "#d0af2b", FontColor = "#ffffff" });
}
```

The below categorize fields holds the appropriate column names from the dataSource -

Field name	Description
Id	It holds the binding name for Id field in the categorize dataSource
Text	It holds the binding name for Text field in the categorize dataSource
Color	It holds the binding name for Color field in the categorize dataSource.
FontColor	It holds the binding name for FontColor field in the categorize dataSource. This font color apply in the appointment.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Datasource for CategorizeSettings -->
List<CategorizeSettings> CategorizeValue = new
List<CategorizeSettings>();
CategorizeValue.Add(new CategorizeSettings { Text = "Blue Category", Id =
"1", Color = "#43b496", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Green Category", Id
= "2", Color = "#7f993e", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Orange Category", Id
= "3", Color = "#cc8638", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Purple Category", Id
= "4", Color = "#ab54a0", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Red Category", Id =
"5", Color = "#dd654e", FontColor = "#ffffff" });
CategorizeValue.Add(new CategorizeSettings { Text = "Yellow Category", Id
= "6", Color = "#d0af2b", FontColor = "#ffffff" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.CategorizeSettings(fields =>
fields.Datasource(CategorizeValue).Enable(true).AllowMultiple(true).Id("I
d").Text("Text").Color("Color").FontColor("FontColor"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Priority

This option prioritize the appointments based on its importance and it can be differentiated with each individual icons/images. By default, there are some specific set of default priority collection and you can also customize it with your own priority collection.

Priority Settings

The PrioritySettings holds the below priority related properties such as,

- Enable - It accepts true or false value, denoting whether to enable/disable the priority option. Its default value is **false**.
- Template – Customize the priority icon/images using template options.
- DataSource – binds the priority dataSource collection. This property should be assigned with the JSON data array collection or instance of ej.DataManger.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for PrioritySettings -->
List<PrioritySettings> PriorityValue = new List<PrioritySettings>();
PriorityValue.Add(new PrioritySettings { Text = "None", Value = "none"
});
PriorityValue.Add(new PrioritySettings { Text = "High", Value = "high"
});
PriorityValue.Add(new PrioritySettings { Text = "Medium", Value =
"medium" });
PriorityValue.Add(new PrioritySettings { Text = "Low", Value = "low" });
}
```

The below priority fields holds the appropriate column names from the dataSource,

Field name	Description
Text	It holds the binding name for text field in the priority dataSource
Value	It holds the binding name for value field in the priority dataSource.

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Datasource for PrioritySettings -->
List<PrioritySettings> PriorityValue = new List<PrioritySettings>();
PriorityValue.Add(new PrioritySettings { Text = "None", Value = "none"
});
PriorityValue.Add(new PrioritySettings { Text = "High", Value = "high"
});
PriorityValue.Add(new PrioritySettings { Text = "Low", Value = "low" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.PrioritySettings(fields =>
fields.Datasource(PriorityValue).Enable(true).Text("Text").Value("Value")
)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Priority("Priority")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Search or Filter Appointments

Appointment Search

The public method searchAppointments is used to search the appointments in the Scheduler dataSource. It contains the below four arguments such as search string, search field, filter operator and ignore case.

searchString - It is used to search the given word/sentence within the appointment data.

fields - It is the field with which the search operation takes place. It's an optional argument.

filterOperator – It denotes the filter type like contains, greaterthan or lessthan. It's an optional argument.

ignoreCase – It is a boolean value to set the search string as case sensitive or not. It's an optional argument.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<div id="grid1" />
<input id="txtSearch" type="text" placeholder="Search Appointments" />
<button id="search" onclick="onSearch()">Search</button>
<script>
function onSearch() {
var _searchString = $("#txtSearch").val();
var schObj = $("#Schedule1").data("ejSchedule");
// method to retrieve the appointment based on search string
var result = schObj.searchAppointments(_searchString);
showResult(result, _searchString);
}
// method to show the result in a grid
function showResult(list, _searchString) {
if (!ej.isNullOrUndefined(list) && list.length != 0 && _searchString !=
"") {
$("#grid1").show();
$("#grid1").data("ejGrid") && $("#grid1").ejGrid("destroy");
$("#grid1").ejGrid({
allowScrolling: true,
dataSource: list,
allowPaging: true
});
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Appointment Filters

The appointments can be filtered or shortlisted based on the simple or complex conditions with four available properties such as **field**, **operator**, **value** and **predicate** which is passed to the public method filterAppointments.

field - It is the field, with which the search operation takes place. It's an optional argument.

operator – It is generally used to specify the filter type.

value – It is the filter keyword based on which the records are filtered.

predicate – To add more than one conditional query, need to use and, or predicate.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<div id="grid1" />
<input id="txtSearch" type="text" placeholder="Search Appointments" />
<button id="search" onclick="onFilter()">Filter</button>
<script>
function onFilter() {
// Add the filter data as like in the below format
var filter = [{
field: "Subject", // field configure
operator: "contains",
value: "Music",
predicate: "or" // predicate
}, {
field: "Subject", // field configure
operator: "contains",
value: "School",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
predicate: "or" // predicate
}];
var schObj = $("#Schedule1").data("ejSchedule");
// Method to get the Filtered appointment
var result = schObj.filterAppointments(filter);
showResult(result);
}
// method to show the result in a grid
function showResult(list, _searchString) {
if (!ej.isNullOrUndefined(list) && list.length != 0 && _searchString !=
"") {
$("#grid1").show();
$("#grid1").data("ejGrid") && $("#grid1").ejGrid("destroy");
$("#grid1").ejGrid({
allowScrolling: true,
dataSource: list,
allowPaging: true
});
}
}
</script>
```

Recurrence Options

There are scenarios where you require the same appointments to be repeatedly created for multiple days on daily, weekly, monthly, and yearly or on every weekday basis.

In appointment data collection, **Recurrence** and **RecurrenceRule** are dependent fields. While creating or binding the recurrence appointment, the **Recurrence** field is set to **true** and **RecurrenceRule** contains recurrence pattern in string format.

Recurrence Rule

The recurrence appointments are created based on the recurrence rule. The RecurrenceRule is a string value that contains the details of the recurrence appointments like

- repeat type - daily/weekly/monthly/yearly/every weekday
- how many times it needs to be repeated
- the interval duration
- the time period to render the appointment, etc.,

It has the following properties based on which the recurrence appointments are rendered in the Schedule control with its respective time period.

S.No	Property	Purpose
1	FREQ	Maintains the Repeat type value of the appointment. (Example: Daily, Weekly, Monthly, Yearly, Every week day) Example: FREQ=DAILY;INTERVAL=1

Copyright © 2001 - 2020 Syncfusion Inc.

2	INTERVAL	Maintains the Repeat type value of the appointment. (Example: Daily, Weekly, Monthly, Yearly, Every weekday) Example: FREQ=DAILY;INTERVAL=1
3	COUNT	It holds the appointmentâ€™s count value. For example, When the recurrence appointment count value is 10, which means that 10 instances of appointments are created in the recurrence series. Example: FREQ=DAILY;INTERVAL=1;COUNT=10
4	UNTIL	This property is used to store the recurrence end date value. For example, when you set the end date of appointment as 11/30/2015, the UNTIL property holds the end date value denoting when the recurrence actually ends. Example: FREQ=DAILY;INTERVAL=1;UNTIL=11/30/2015
5	BYDAY	It holds the DAY values, representing on which the appointments actually renders. For example, Create the weekly appointment, and select the day(s) from the day options (Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday). When Monday is selected, the first two letters of the selected day â€œMOâ€• is saved in the BYDAY property. When multiple days are selected, the values are separated by commas. Example: FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE;COUNT=10
6	BYMONTHDAY	This property is used to store the date value of the Month, while creating the Month recurrence appointment. For example, when you create a Monthly recurrence appointment for every 3rd day of the month, then BYMONTHDAY holds the value 3 and creates the appointment on 3rd day of every month. Example: FREQ=MONTHLY;BYMONTHDAY=3;INTERVAL=1;COUNT=10
7	BYMONTH	This property is used to store the index value of the selected Month while creating the yearly appointments. For example, when you create the yearly appointment on June month, the index value of June month 6 will get stored in the BYMONTH field. The appointment is created on every 6th month of a year.

Copyright © 2001 - 2020 Syncfusion Inc.

		Example: FREQ=YEARLY;BYMONTHDAY=16;BYMONTH=6;INTERVAL=1;COUNT=10
8	BYSETPOS	This property is used to store the index value of the week. For example, when you create the monthly appointment in second week of a month, the index value of the second week (2) is stored in BYSETPOS. Example: FREQ=MONTHLY;BYDAY=MO;BYSETPOS=2;UNTIL=8/11/2015
9	WKST	This property is used to store the start day value of a week. For example, when you render the workweek the â€œWKSTâ€• value is Monday.
10	EXDATE	EXDATE is used to hold the modified appointment date details (date value) in the recurrence appointment series. For example, when you change the recurrence appointment instance under the date â€œ6/19/2015â€•, then this date is added to the recurrence rule EXDATE field. â€œEXDATEâ€• is also used to differentiate the edited occurrence of the recurrence series for some internal process while doing the â€œEdit Series or Delete seriesâ€• actions. Example: FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR;COUNT=10;EXDATE=6/18/2015,6/20/2015;RECUREDITID=1651
11	RECUREDITID	This property contains the Parent Id value of the edited appointment. It is used to track the edited appointment occurrence with its parent recurrence appointment series. For example, when you edit the particular occurrence of the recurrence appointment series, the â€œRECUREDITIDâ€• is added to that edited appointment depicting its parent Id. Example: FREQ=WEEKLY;BYDAY=MO,TU,WE,TH,FR;COUNT=10;EXDATE=6/18/2015,6/20/2015;RECUREDITID=1651

To know more about other possible combinations of above specified recurrence rule properties, refer here.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
// Recurrence mapper field
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Recurrence Validation

The default recurrence validation has been included for recurrence appointments similar to the one available in outlook. The validation occurs during the recurrence appointment creation, drag and drop or resizing of the recurrence appointments and also if any single occurrence changes. The validation can be disabled by setting the `EnableRecurrenceValidation` to `false`.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.EnableRecurrenceValidation(false) // Disable Recurrence Validation
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
// Recurrence mapper field
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: You can parse the **RecurrenceRule** of an appointment from the server-side by making use of a new generic utility class **RecurrenceHelper**. Refer this KB document.

Recurrence Edit and Delete options

The recurring appointments can be edited or deleted in three ways as below:

- Single occurrence
- Following appointments
- Entire series

Single occurrence

When an option "Only this Appointment" is selected, a single occurrence of the recurrence appointment alone will be edited or deleted.

Following appointments

When an option "Following Appointments" is selected, all the following events of the recurrence appointment from the current instance will be edited or deleted. The previous instances of the recurrence appointment before this current instances will be retained as it is on the Scheduler.

Entire series

The entire recurrence series will be edited / deleted, on selecting this option.

Reminder

Reminder option notifies all the appointments before some specific time. By default, it notifies before 5 minutes. Each and every appointment triggers the Reminder event and can utilize this event for other user actions like mailing particular event to someone or to do any kind of manipulations with the reminder appointments and so on. The ReminderSettings includes the following 2 properties namely,

- **Enable** - To enable the reminder settings of the Schedule control, set the **Enable** property as true within the Reminder
- **AlertBefore** - Accepts the integer value to denote the time, before how long the reminder should be notified to the user.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ReminderSettings(rem => rem.Enable(true).AlertBefore(10))
.ScheduleClientSideEvents(event => event.Reminder("reminderCustom"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

RAZOR

```
function reminderCustom(args) {
alert("Reminder Appointment");
}
```

Note: Whenever the reminder setting is enabled in the Scheduler with some specific value (in minutes) assigned to the **AlertBefore** property, the **Reminder** event gets triggered before this specified value. It includes the reminder appointment's entire information within its arguments.

Block Time Intervals

It allows to block the particular timeslots in Schedule. When specific timeslots are blocked, the appointments that lies in that range can either be made read-only or else can be allowed to interact with the users based on the value assigned to the IsBlockAppointment property.

Blockout Settings

The BlockoutSettings holds the below block intervals related properties such as,

- Enable - It accepts true or false value, denoting whether to enable/disable the block intervals option. It's default value is false.
- TemplateId – It applies the template design to block the intervals.
- Datasource – Binds the block intervals dataSource collection. This property should be assigned either with the JSON data array collection or instance of DataManager (ej.DataManager).

The below blockout fields holds the appropriate column names from the dataSource -

Field name	Description
Id	It holds the binding name for Id field in the blockout dataSource
Subject	It holds the binding name for Subject field in the blockout dataSource
StartTime	It holds the binding name for StartTime field in the blockout dataSource.
EndTime	It holds the binding name for EndTime field in the blockout dataSource.
IsBlockAppointment	It holds the binding name for IsBlockAppointment field in the blockout dataSource.

Copyright © 2001 - 2020 Syncfusion Inc.

IsAllDay	It holds the binding name for IsAllDay field in the blockout dataSource.
ResourceId	It holds the binding name for ResourceId field in the blockout dataSource.
CustomStyle	It holds the binding name for CustomStyle field in the blockout dataSource.

The below example depicts the appointment fields accepting the string type mapper fields.

Create a new Class to define the block intervals to be passed to the Scheduler as mentioned below,

C#

```
public class BlockData
{
public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public Boolean IsBlockAppointment { get; set; }
public string ResourceId { get; set; }
public bool IsAllDay { get; set; }
}
```

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Block Intervals -->
List<BlockData> blockData = new List<BlockData>();
blockData.Add(new BlockData { Id = 1, Subject = "Service", StartTime =
new DateTime(2015, 11, 10, 9, 00, 00), EndTime = new DateTime(2015, 11,
10, 10, 00, 00), IsBlockAppointment = true });
blockData.Add(new BlockData { Id = 2, Subject = "Maintenance", StartTime
= new DateTime(2015, 11, 11, 11, 00, 00), EndTime = new DateTime(2015,
11, 11, 13, 00, 00), IsBlockAppointment = true });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 10))
.BlockoutSettings(fields => fields.Enable(true)
.Datasource(blockData)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.IsBlockAppointment("IsBlockAppointment"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Blocking Appointments

The Appointments that lies within the blocked time range can be restricted to perform CRUD operations in it and can be made read-only. This can be achieved by setting IsBlockAppointment property to true.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Block Intervals -->
List<BlockData> blockData = new List<BlockData>();
blockData.Add(new BlockData { Id = 1, Subject = "Service", StartTime =
new DateTime(2015, 11, 10, 9, 00, 00), EndTime = new DateTime(2015, 11,
10, 10, 00, 00), IsBlockAppointment = true });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 10))
//Configure the blockout settings
.BlockoutSettings(fields => fields.Enable(true)
.Datasource(blockData)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
//Bind the block appointment field in datasource
.IsBlockAppointment("IsBlockAppointment"))
)
```

Customizing block time intervals

The BlockoutSettings holds the below properties to customize the block intervals such as,

- TemplateId - Template design that applies on the block intervals.
- CustomStyle - The custom CSS that applies on the blocked intervals.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Block Intervals -->
List<BlockData> blockData = new List<BlockData>();
blockData.Add(new BlockData { Id = 1, Subject = "Service", StartTime =
new DateTime(2015, 11, 10, 9, 00, 00), EndTime = new DateTime(2015, 11,
10, 10, 00, 00), IsBlockAppointment = true });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 10))
//Configure the blockout settings
.BlockoutSettings(fields => fields.Enable(true)
.TemplateId("#blockTemplate")
.Datasource(blockData)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.IsBlockAppointment("IsBlockAppointment"))
)
```

HTML

```
<!--Template to apply block intervals-->
<script id="blockTemplate" type="text/x-jsrender">
<div style="height:100%">
<div>{{:Subject}}</div>
</div>
</script>
```

Blocking time interval based on resources

- ResourceId property used within the BlockoutSettings which accepts the resource id's can be used to apply the block intervals based on the resources.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Block Intervals -->
List<BlockData> blockData = new List<BlockData>();
blockData.Add(new BlockData { Id = 1, Subject = "Service", StartTime =
new DateTime(2015, 11, 10, 9, 00, 00), EndTime = new DateTime(2015, 11,
10, 10, 00, 00), IsBlockAppointment = true, ResourceId = "1" });
<!-- Datasource for Appointments -->
List<ScheduleData> Appoint = new List<ScheduleData>();
Appoint.Add(new ScheduleData { Id = "1", Subject = "Meeting", StartTime =
new DateTime(2015, 11, 11, 10, 00, 00), EndTime = new DateTime(2015, 11,
11, 11, 00, 00), Description = "", AllDay = false, Recurrence = false,
RecurrenceRule = "",OwnerId="1" });
<!-- Define the Group -->
List<String> Group = new List<String>();
Group.Add("Owners");
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398", GroupId = "1" });
Owner.Add(new ResourceFields { Id = "2", Text = "Steven", Color =
"#56ca95", GroupId = "1" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 10))
//Configure the blockout settings
.BlockoutSettings(fields => fields.Enable(true)
.Datasource(blockData)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.IsBlockAppointment("IsBlockAppointment")
//resource id mapper field
.ResourceId("ResourceId"))
.Group(gr => { gr.Resources(Group); })
.Resources(res =>
{
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
```

Context Menu

Scheduler provides default context menu options for both appointments as well as work cells. It also allows to define additional custom context menu options.

The options that are available under `ContextMenuSettings` are as follows,

- **Enable** - Enables/disables the context menu option in Scheduler.
- **MenuItems** - Contains the sub-menu collections related to both the appointment and cells.

Default Menu Options

The menu items contains two separate collection based on the appointment and cells.

Appointment

The appointment collection includes the following options.

Open Appointment (default)
Delete Appointment (default)
Print Appointment
Categorize

Cells

The default options available within the cell collection includes -

Copyright © 2001 - 2020 Syncfusion Inc.

New Appointment
New Recurring Appointment
Today
Go to date
Settings (View, TimeMode, Work Hours)

The following code snippet shows how to enable the context menu settings in Scheduler and to make use of it with default available options.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
<!-- List of Cell menu options -->
List<Cells> CellMenu = new List<Cells>();
CellMenu.Add(new Cells { Id = "new", Text = "New Appointment" });
CellMenu.Add(new Cells { Id = "recurrence", Text = "New Recurring
Appointment" });
CellMenu.Add(new Cells { Id = "today", Text = "Today" });
CellMenu.Add(new Cells { Id = "goToDate", Text = "Go to date" });
CellMenu.Add(new Cells { Id = "settings", Text = "Settings" });
CellMenu.Add(new Cells { Id = "view", Text = "View", ParentId =
"settings" });
CellMenu.Add(new Cells { Id = "timeMode", Text = "TimeMode", ParentId =
"settings" });
CellMenu.Add(new Cells { Id = "view_Day", Text = "Day", ParentId = "view"
});
CellMenu.Add(new Cells { Id = "view_Week", Text = "Week", ParentId =
"view" });
CellMenu.Add(new Cells { Id = "view_Workweek", Text = "WorkWeek",
ParentId = "view" });
CellMenu.Add(new Cells { Id = "view_Month", Text = "Month", ParentId =
"view" });
CellMenu.Add(new Cells { Id = "timeMode_Hour12", Text = "12 Hour",
ParentId = "timeMode" });
CellMenu.Add(new Cells { Id = "timeMode_Hour24", Text = "24 Hour",
ParentId = "timeMode" });
CellMenu.Add(new Cells { Id = "workhours", Text = "Work Hours", ParentId
= "settings" });
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items =>
items.Appointment(AppMenu).Cells(CellMenu)))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: In agenda view, only the appointment menu items shows up in the context menu options. For default menu items, the id must be defined with the same value as mentioned in the above code example – as we have processed the menus based on this id within our source.

Custom Menu Options

Apart from the default available options, it is also possible to add custom menu options to the context-menu of both the appointment and cell collection.

The following code example depicts how **to add the custom menu items** to the appointment and cell collection of the context menu settings.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
AppMenu.Add(new Appointment { Id = "option1", Text = "User Option 1" });
<!-- List of Cell menu options -->
List<Cells> CellMenu = new List<Cells>();
CellMenu.Add(new Cells { Id = "customOption", Text = "User Option" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items =>
items.Appointment(AppMenu).Cells(CellMenu)))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: The **Id** given for the custom menu items **must be unique** in both the appointment and cell collection.

Handling Menu Actions

To define specific actions for a click made on the custom menu items, the client-side event MenuItemClick can be used as depicted in the below code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
AppMenu.Add(new Appointment { Id = "option1", Text = "User Option 1" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items =>
items.Appointment(AppMenu).Cells(CellMenu)))
.ScheduleClientSideEvents(event =>
event.MenuItemClick("onMenuItemClick"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.RecurrenceRule("RecurrenceRule"))
)
```

JS

```
function onMenuItemClick(args) {
//args.events contains information of the clicked menu item.
if (args.events.ID == "option1")
alert("Custom menu clicked");
}
```

Also, it is possible to predict the target on which the right click is made, either on the cells or appointments with the use of the event BeforeContextMenuOpen. The below code example shows how to block the display of context menu when right clicked on the cells by setting **args.cancel** as **true**.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
AppMenu.Add(new Appointment { Id = "option1", Text = "User Option 1" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items =>
items.Appointment(AppMenu).Cells(CellMenu)))
.ScheduleClientSideEvents(event =>
event.BeforeContextMenuOpen("beforeContextMenuOpen"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
function beforeContextMenuOpen(args) {
//args.events.target - target information to depict either
cell/appointment
if ($(args.events.target).hasClass("e-workcells") ||
$(args.events.target).hasClass("e-monthcells"))
args.cancel = true;
}
```

Adding Categorize Option

To include the default categorize options within the context menu, it is necessary to enable the CategorizeSettings property as shown in the below code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
AppMenu.Add(new Appointment { Id = "categorize", Text = "Categorize" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items =>
items.Appointment(AppMenu).Cells(CellMenu)))
.CategorizeSettings(cat => cat.Enable(true))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: The **Categorize** option must be added to the **Appointment** collections simply with an id "categorize", which displays on right clicking the appointments.

Copyright © 2001 - 2020 Syncfusion Inc.

Resources

The Scheduler provides **Resources** support, with which the single Scheduler is shared by multiple resources simultaneously. Each resource in the Scheduler is arranged in a column/row wise, with individual spacing to display all its respective appointments on a single page. It also supports the grouping of resources, thus enabling the categorization of resources in a hierarchical structure and shows it either in expandable groups (**Horizontal view**) or else vertical hierarchy one after the other (**Vertical view**).

One or more resources can be assigned to the Scheduler appointments by making selection of the resource options available in the appointment window.

Fields of Resources

The default options available within the Resources collection are as follows,

*Name (**String**)*

A unique resource name which is used for differentiating various resource objects while grouping it in levels.

*Title (**String**)*

It holds the title name of the resource field to be displayed on the Scheduler appointment window.

*Field (**String**)*

It holds the name of the resource field to be bound to the Scheduler appointments which contains the resource Id.

*AllowMultiple (**Boolean**)*

When set to true, allows multiple selection of resource names, thus creating multiple instances of same appointment for the selected resources.

*ResourceSettings (**Object**)*

It holds the field names of the resources dataSource to be bound to the Scheduler.

The following are the resource fields which must be defined within the **ResourceSettings** that holds the appropriate column names from the dataSource.

Field name	Description
Text	Binds the Text field name in the dataSource to the ResourceSettings Text. These text gets listed out in the resources field of the appointment window. Itâ€™s mandatory.
Id	Binds the Id field name in the dataSource to the ResourceSettings Id. Itâ€™s mandatory.
GroupId	Binds the GroupId field name in the dataSource to the ResourceSettings GroupId. This field is not necessary for a resource object (resource data) defined as first level within the resources collection.
Color	Binds the Color field name in the dataSource to the ResourceSettings Color. It is optional.

Copyright © 2001 - 2020 Syncfusion Inc.

AppointmentClass	Binds the AppointmentClass field name in the dataSource. It applies the custom CSS class name to the appointments based on the resources.
Start	Binds the starting work hour field name in the dataSource. It's optional, but when provided with some numeric value will set the starting work hour for specific resources.
End	Binds the end work hour field name in the dataSource. It's optional, but when provided with some numeric value will set the end work hour for specific resources.
WorkWeek	Binds the resources working days field name in the dataSource. It's optional, and accept array of strings which is nothing but only the week day names. When provided with some values (array of day names), only those days will render for the specific resources.

Example: To set the resources options using all the above specified fields,

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", RoomId = "2" });
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398" });
Owner.Add(new ResourceFields { Id = "2", Text = "Steven", Color =
"#56ca95" });
<!-- Datasource for Rooms -->
List<ResourceFields> Room = new List<ResourceFields>();
Room.Add(new ResourceFields { Id = "1", Text = "Room1", Color =
"#f8a398", GroupId = "1", WorkHourStart=10, WorkHourEnd=18, CustomDays:
["Monday","Wednesday","Friday"]});
Room.Add(new ResourceFields { Id = "2", Text = "Room2", Color =
"#56ca85", GroupId = "2", WorkHourStart=6, WorkHourEnd=10, CustomDays:
["Monday","Saturday"]});
Room.Add(new ResourceFields { Id = "3", Text = "Room3", Color =
"#56ac88", GroupId = "2", WorkHourStart=11, WorkHourEnd=15, CustomDays:
["Tuesday","Friday"]});
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Resources(res => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(false).R
esourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add();
res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).Resou
rceSettings(fields =>
fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("Gro
upId").Start("WorkHourStart").End("WorkHourEnd").WorkWeek("CustomDays")).
Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId,RoomId"))
)
```

Note: The resource object defined at **First Level** within the **Resources** collection doesn't make use of the **GroupId** field, as there is no previous levels applicable to map.

Data Binding

The resources data can be bound to the Schedule control through the **ResourceSettings** option available within the **Resources** property. The data-binding can be done either using JSON object collection or DataManager (ej.DataManager) instance which contains the resources related data.

List Data

Example: To set the resource data with array of **List** data collection

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398", GroupId = "1" });
Owner.Add(new ResourceFields { Id = "2", Text = "Steven", Color =
"#56ca95", GroupId = "1" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Resources(res => {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
```

Remote Data

Example: To set the resource data through remote service URL,

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Resources(res => {
res.ResourceSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/OdataServices/Northwnd.svc")).T
ext("CategoryName").Id("CategoryId").Query("ej.Query().select('CategoryID
', 'CategoryName').from('Categories').take(5)")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Multiple Resources (Without Grouping)

It is possible to display the Scheduler in default look without visually showcasing all the resources on it, but it allow the user to assign the required resources to the appointments through the appointment window resource options.

The appointments belonging to all the resources will be displayed on the Scheduler which will be differentiated based on the resource color assigned in the **ResourceSettings** (depicting to which resource that particular appointment belongs).

Example: To display default Scheduler with multiple resource options in the appointment window,

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398", GroupId = "1" });
Owner.Add(new ResourceFields { Id = "2", Text = "Steven", Color =
"#56ca95", GroupId = "1" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Resources(res => {
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
```

Note: Setting **AllowMultiple** to **true** in the above code snippet allows the user to select multiple resources in the appointment window and also creates multiple copies of the same appointment in the Scheduler for each resources while saving.

Copyright © 2001 - 2020 Syncfusion Inc.

Grouping

Scheduler supports both single and multiple levels of resource grouping that can be customized either in horizontal or vertical Scheduler views. In Vertical view - the levels are displayed in a tree structure one after the other, but in horizontal view – the levels are grouped in a vertically expandable/collapsible structure.

Single-Level

This type of grouping allows the Scheduler to display all the resources at a single level simultaneously. The appointments will make use of the **Color** defined for the first resource instance as its background color.

Example: To display the Scheduler with single level resource grouping options,

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Define the Group -->
List<String> Group = new List<String>();
Group.Add("Owners");
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398", GroupId = "1" });
Owner.Add(new ResourceFields { Id = "2", Text = "Steven", Color =
"#56ca95", GroupId = "1" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Group(gr => { gr.Resources(Group); })
.Resources(res => {
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The **Name** field mentioned in the **Resource** object needs to be specified within the **Group** property in order to enable the grouping option in Scheduler.

Multi-Level

This type of grouping displays the resources in the Scheduler at multiple levels with a set of resources grouped under each parent. The appointments will make use of the **Color** defined for the first/top level resource instance as its background color.

Example: To display the Scheduler with multiple level resource grouping options,

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", RoomId = "1", OwnerId = "1" });
<!-- Datasource for Grouping -->
List<String> Group = new List<String>();
Group.Add("Owners");
Group.Add("Rooms");
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398", GroupId = "1" });
Owner.Add(new ResourceFields { Id = "2", Text = "Steven", Color =
"#56ca95", GroupId = "1" });
<!-- Datasource for Rooms -->
List<ResourceFields> Room = new List<ResourceFields>();
Room.Add(new ResourceFields { Id = "1", Text = "Room1", Color =
"#f8a398", GroupId = "1" });
Room.Add(new ResourceFields { Id = "2", Text = "Room2", Color =
"#56ca85", GroupId = "2" });
Room.Add(new ResourceFields { Id = "3", Text = "Room3", Color =
"#56ac88", GroupId = "2" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Group(gr => { gr.Resources(Group); })
.Resources(res => {
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add();
res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).Resou
rceSettings(fields =>
fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("Gro
upId")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId,RoomId"))
)
```

Note: Here, the appointments will make use of the **Color** defined for the Owners resource instance as its background color.

Different Working days and Hours for Resources

It is possible to assign different workdays and workhours for each resources present within the Scheduler. The process of assigning different working days for every individual resources is applicable only for the vertical Scheduler mode and not for timeline view, whereas the customization of workhours for each resources is applicable on both the Scheduler orientation. The custom workdays and workhours needs to be defined within the ResourceSettings property using the following 3 sub-properties available within it.

- **Start** – To define the work start hour for each individual resources
- **End** – To define the work end hour for each individual resources
- **WorkWeek** – To define the working days for each individual resources

Example: To display the Scheduler with each resources having different workhours and workdays, the code example is depicted below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", RoomId = "1", OwnerId = "5" });
<!-- Datasource for Grouping -->
List<String> Group = new List<String>();
Group.Add("Owners");
Group.Add("Rooms");
<!-- Datasource for Rooms -->
List<ResourceFields> Room = new List<ResourceFields>();
Room.Add(new ResourceFields { Id = "1", Text = "Room1", Color =
"#f8a398", GroupId = "1" });
Room.Add(new ResourceFields { Id = "2", Text = "Room2", Color =
"#56ca95", GroupId = "1" });
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Text = "Nancy", Id= "1", GroupId = "1",
Color = "#ffaa00", Start = "10", End = "18", WorkWeek = new List<string>
{ "Monday", "Wednesday", "Friday" } });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Owner.Add(new ResourceFields { Text = "Steven", Id = "3", GroupId = "2",
Color = "#f8a398", Start = "6", End = "10", WorkWeek = new List<string> {
"Tuesday", "Thursday" } });
Owner.Add(new ResourceFields { Text = "Michael", Id = "5", GroupId = "1",
Color = "#7499e1", Start = "11", End = "15", WorkWeek = new List<string>
{ "Sunday", "Tuesday", "Thursday", "Saturday" } });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Group(gr => { gr.Resources(Group); })
.Resources(res => {
res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).Resou
rceSettings(fields =>
fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("Gro
upId")).Add();
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).Re
sourceSettings(fields =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color").GroupId("Gr
oupId").Start("Start").End("End").WorkWeek("WorkWeek")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("RoomId,OwnerId"))
)
```

Customization

The Scheduler can be customized in various aspects like -

- Setting different Start/end hour limits
- Highlighting the working hours
- Setting different Date format
- Specifying minimum and maximum date ranges
- Customize the entire appointment window with the user required fields
- Setting different time Slot duration
- Complete Scheduler customization using QueryCellInfo event
- Setting different FirstDayOfWeek

Hour Customization

It includes customization of displaying Scheduler with specific Start/End hours and also defining the working hour time range which is differentiated as business hours.

Copyright © 2001 - 2020 Syncfusion Inc.

Schedule Display Hours

It denotes the start and end hour time limits to be displayed on the Scheduler. To set this time limit, two properties namely StartHour and EndHour can be used.

- **StartHour** - The hour from which the Scheduler time display actually starts.
- **EndHour** - The hour on which the Scheduler time display should end.

The following code example renders the scheduler from 7.00 AM to 6.00 PM.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.StartHour(7)
.EndHour(18)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Working Hours

Working hours indicates the work hour limit within the Scheduler, which is highlighted visually with white colored work cells. To enable the highlighting of work hours on the Scheduler, set the **Highlight** option available within the WorkHours property to **true**. By default, it is set to true and includes the below specified options,

- **Highlight** – enables/disables the highlighting of work hours.
- **Start** - sets the start time of the working/business hour in a day.
- **End** - sets the end time limit of the working/business hour in a day.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.WorkHours(hrs => hrs.Highlight(true).Start(8).End(16))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: By default, work hour **Start** is set to **9** and **End** is set to **18**. Also, the Scheduler cells automatically scrolls up or down based on the starting work hour, to make the user to view that particular time initially.

Hide Weekend days

The Scheduler can be customized to display only the working days, thus hiding the weekend days from it. The working days render based on the values given in the WorkWeek property. The days that are not mentioned in the WorkWeek collection is considered to be the weekend days and it can be hidden from the Scheduler by setting false to the ShowWeekend property.

The following code example renders the Scheduler by hiding the weekend days.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowWeekend(false)
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

TimeScale

The TimeScale allows the user to set the required time slot duration for the work cells that displays on the Scheduler. It provides option to customize both the major and minor slots using template option. It includes the below properties such as,

Copyright © 2001 - 2020 Syncfusion Inc.

- Enable - It accepts true or false value, denoting whether to show or hide the time slots. Its default value is true.
- MajorSlot – Specifies the major time slot duration.
- MinorSlotCount – Specifies the value, based on which the minor time slots are divided into appropriate count.
- TimeScale Templates – 2 template options available for customizing time scales namely MinorSlotTemplateId and MajorSlotTemplateId.

The MajorSlot and MinorSlot can be customized with the following code example.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.TimeScale(ts => ts.Enable(true).MajorSlot(60).MinorSlotCount(6))
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Date Customization

The date in the Scheduler can be customized by setting specific minimum and maximum date ranges and also defining various date formats to it.

Current Date

The Current date indicates the date with which the Scheduler loads initially and based on which the appropriate date range displays in the week/workweek/month/agenda views. To set the current date to the Scheduler – use the following code example,

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: By default, the System current date will be taken as Scheduler's current date.

MinDate and MaxDate

Providing the MinDate and MaxDate property with some date values, allows the Scheduler to set the minimum and maximum date range. The Scheduler date that lies beyond these minimum and maximum date range will be in a disabled state, so that the date navigation is blocked beyond these specified date range. Also, the appointments that lies beyond these date ranges will not be displayed on the Scheduler.

The following code example shows how to set the MinDate and MaxDate properties of the Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.MinDate(new DateTime(2015, 11, 4))
.MaxDate(new DateTime(2015, 11, 8))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: The **MaxDate** value provided should always be greater than that of **MinDate** value.

Appointment Window Customization

It is possible to use the custom appointment window option to design it with the user-required extra fields apart from the other default available fields. To make use of the customized appointment window, it is necessary to use the AppointmentWindowOpen event within which the display of default appointment window is prevented.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example lets you create the custom appointment window (using recurrence editor feature) with a single extra field for defining the appointment type.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ScheduleClientSideEvents(event =>
event.AppointmentWindowOpen("onAppointmentWindowOpen"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<div id="customWindow" style="display: none">
<div id="appWindow">
<form id="custom">
<table width="100%" cellpadding="5">
<tbody>
<tr style="display: none">
<td>
Id:
</td>
<td colspan="2">
<input id="customId" type="text" name="Id" />
</td>
</tr>
<tr>
<td>
Subject:
</td>
<td colspan="2">
<input id="subject" type="text" value="" name="Subject" onfocus="temp()"
style="width: 100%" />
</td>
</tr>
<tr>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<td>
Description:
</td>
<td colspan="2">
<textarea id="customdescription" name="Description" rows="3" cols="50"
style="width: 100%; resize: vertical"></textarea>
</td>
</tr>
<tr>
<td>
StartTime:
</td>
<td>
@Html.EJ().DateTimePicker("StartTime").Width("150px")
</td>
</tr>
<tr>
<td>
EndTime:
</td>
<td>
@Html.EJ().DateTimePicker("EndTime").Width("150px")
</td>
</tr>
<tr>
<td colspan="3">
<div class="customcheck">AllDay:</div>
<div class="customcheck">
<input id="allDay" type="checkbox" name="AllDay" onchange="allDayCheck()"
/>
</div>
<div class="customcheck">Recurrence:</div>
<div>
<input id="recurrence" type="checkbox" name="Recurrence" />
</div>
</td>
</tr>
<tr id="summaryTr" style="display:none;">
<td colspan="3">
<div class="recSummary">Summary:</div>
<div>
<label id="recSummary" name="Summary"></label>
</div>
</td>
</tr>
<tr id="editor" style="display:none;">
<td colspan="3">
<div><a id="recedit" onclick="recurrenceRule()">Edit</a></div>
</td>
</tr>
</tbody>
</table>
</form>
<div>
<button type="submit" onclick="cancel()" id="buttonCancel"
style="float:right;margin-right:20px;margin-bottom:10px;">Cancel</button>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<button type="submit" onclick="save()" id="buttonSubmit"
style="float:right;margin-right:20px;margin-bottom:10px;">Submit</button>
</div>
</div>
<div id="recWindow" style="display: none">
<div style="margin: 0 15%;">
@(Html.EJ().RecurrenceEditor("recurrenceEditor").SelectedRecurrenceType(0
).Frequencies(new List<string> { "daily", "weekly", "monthly", "yearly",
"everyweekday" }))
</div>
<div>
<button type="submit" id="recCancel">Cancel</button>
<button type="submit" id="recSubmit">Submit</button>
</div>
</div>
</div>
```

The styles to be applied for the controls within the custom appointment window are as follows.

HTML

```
<style>
.customcheck {
float: left;
margin-right: 10px;
}
.error {
background-color: #FF8A8A;
}
#custom table td {
padding:5px;
}
</style>
```

Now, define the Schedule control with custom appointment window within script as shown below.

HTML

```
<script>
$(function () {
// defining sub-controls used within custom appointment window
$("#buttonCancel").ejButton({ width: '85px' });
$("#buttonSubmit").ejButton({ width: '85px' });
$("#recurrence").ejCheckBox({ change: "recurCheck" });
$("#recCancel,#recSubmit").ejButton({ click: "onRecurrenceClick" });
// defining dialog control to be used as custom appointment window
$("#customWindow").ejDialog({
width: 600,
height: "auto",
position: { X: 200, Y: 100 },
showOnInit: false,
enableModal: true,
title: "Appointment Window",
enableResize: false,
allowKeyboardNavigation: false,
close: "clearFields"
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
});
</script>
```

The following function needs to be defined within script section, which gets called before the default appointment window opens.

JS

```
function onAppointmentWindowOpen(args) {
args.cancel = true; // prevents the display of default appointment window
var schObj = $("#Schedule1").data("ejSchedule");
// When double clicked on the Scheduler cells, fills the StartTime and
EndTime fields appropriately
$("#StartTime").ejDateTimePicker({ value: args.startTime });
$("#EndTime").ejDateTimePicker({ value: args.endTime });
$("#recWindow").css("display", "none");
$("#appWindow").css("display", "");
if (!ej.isNullOrUndefined(args.target)) {
// When double clicked on the Scheduler cells, if the target is allDay or
month cells - only then enable check mark on the allDay checkbox
if ($(args.target.currentTarget).hasClass("e-alldaycells") ||
(args.startTime.getHours() == 0 && args.endTime.getHours() == 23))
$("#allDay").prop("checked", true);
else
args.model.currentView == "month" ? $("#allDay").prop("checked", true) :
$("#allDay").prop("checked", false);
// If the target is allDay or month cells - disable the StartTime and
EndTime fields
$("#StartTime,#EndTime").ejDateTimePicker({
enabled: ($(args.target.currentTarget).hasClass("e-alldaycells") ||
(args.startTime.getHours() == 0 && args.endTime.getHours() == 23) ||
$(args.target.currentTarget).hasClass("e-monthcells") ||
args.model.currentView == "month") ? false : true
});
}
// If double clicked on the appointments, fill the custom appointment
window fields with appropriate values.
if (!ej.isNullOrUndefined(args.appointment)) {
$("#customId").val(args.appointment.Id);
$("#subject").val(args.appointment.Subject);
$("#StartTime").ejDateTimePicker({ value: new
Date(args.appointment.StartTime) });
$("#EndTime").ejDateTimePicker({ value: new
Date(args.appointment.EndTime) });
$("#recurrence").ejCheckBox({ checked: args.appointment.Recurrence });
if (args.appointment.Recurrence) {
$("#editor").css("display", "");
$("#recSummary").html(args.appointment.RecurrenceRule);
$("#summaryTr").css("display", "");
recObj._recRule = args.appointment.RecurrenceRule; // app recurrence rule
is stored in Recurrence editor object
recObj.recurrenceRuleSplit(args.appointment.RecurrenceRule,
args.appointment.recurrenceExDate); //splitting the recurrence rule
recObj.showRecurrenceSummary(args.appointment.Id); // updating the
recurrence rule in Recurrence editor
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
$("#customWindow").ejDialog("open");
}
```

On clicking the **Submit** button within the Custom Appointment window, the following function gets executed – which will validate the appointment fields and then save it appropriately.

JS

```
function save() {
// checks if the subject value is not left blank before saving it.
if ($.trim($("#subject").val()) == "") {
$("#subject").addClass("error");
return false;
}
var obj = {}, temp = {}, rType;
var formElement = $("#customWindow").find("#custom").get(0);
// looping through the custom form elements to get each value and form a
JSON object
for (var index = 0; index < formElement.length; index++) {
var columnName = formElement[index].name, $element =
$(formElement[index]);
if (columnName != undefined) {
if (columnName == "")
columnName = formElement[index].id.replace(this._id, "");
if (columnName != "" && obj[columnName] == null) {
var value = formElement[index].value;
if (columnName == "Id" && value != "")
value = parseInt(value);
if ($element.hasClass("e-datetimepicker"))
value = new Date(value);
if (formElement[index].type == "checkbox")
value = formElement[index].checked;
obj[columnName] = value;
}
}
}
obj["RecurrenceRule"] = (obj.Recurrence) ? recurRule : null;
$("#customWindow").ejDialog("close");
var object = $("#Schedule1").data("ejSchedule");
object.saveAppointment(obj);
clearFields();
}
// This function executes when the submit/cancel button in the recurrence
editor window is pressed.
function onRecurrenceClick(args) {
if ($(args.e.currentTarget).attr("id") == "recSubmit") {
recObj = $("#recurrenceEditor").ejRecurrenceEditor('instance');
recObj.closeRecurPublic();
recurRule = recObj._recRule;
$("#recSummary").html(recurRule);
}
else
if (($(args.e.currentTarget).attr("id") == "recCancel")) {
if ($("#recSummary").html() == "") {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#editor").css("display", "none");
$("#recurrence").ejCheckBox({ checked: false });
}
else
$("#recurrence").ejCheckBox({ checked: true });
}
$("#recWindow").css("display", "none");
$("#appWindow").css("display", "");
if ($("#recSummary").html() != "")
$("#summaryTr").css("display", "");
}
// This function executes when the Edit anchor tag in the edit
appointment window is clicked.
function recurrenceRule() {
$("#recWindow").css("display", "");
$("#appWindow").css("display", "none");
}
// Clears all the field values of the custom window after saving
appointments
function clearFields() {
$("#customId").val("");
recObj.clearRecurrenceFields();
$("#subject").val("");
$("#recSummary").html("");
$("#summaryTr").css("display", "none");
$("#recurrence").ejCheckBox({ checked: false });
$("#editor").css("display", "none");
$("#StartTime,#EndTime").ejDateTimePicker({ enabled: true });
}
// This function executes when the recurrence checkbox is checked in the
custom appointment window
function recurCheck(args) {
if (args.isInteraction) {
if ($("#recurrence").get(0).checked == true) {  // Displays the
recurrence field, when recurrence checkbox is checked.
$("#recWindow").css("display", "");
$("#appWindow").css("display", "none");
$("#editor").css("display", "");
}
else {
$("#recWindow").css("display", "none");
$("#editor").css("display", "none");
$("#recSummary").html("");
$("#summaryTr").css("display", "none");
}
}
}
// This function executes when the All-day checkbox is checked in the
custom appointment window
function allDayCheck() {
// Disables and sets the specific hours to the StartTime and EndTime
fields, when the all-day checkbox is checked
if ($("#allDay").prop("checked")) {
var a = $("#StartTime").data("ejDateTimePicker").model.value;
a.setHours(0, 0, 0);
var b = $("#EndTime").data("ejDateTimePicker").model.value;
b.setHours(23, 59, 0);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#StartTime").ejDateTimePicker({
value: new Date(a),
enabled: false
});
$("#EndTime").ejDateTimePicker({
value: new Date(b),
enabled: false
});
} else {
$("#StartTime").ejDateTimePicker({
enabled: true
});
$("#EndTime").ejDateTimePicker({
enabled: true
});
}
}
// This function executes when the subject text field is currently being
focused
function temp() {
$("#subject").removeClass("error");
}
// This function executes when the cancel button in the custom
appointment window is pressed.
function cancel() {
recObj = $("#recurrenceEditor").ejRecurrenceEditor('instance');
clearFields();
$("#customWindow").ejDialog("close");
}
```

Scheduler Customization using QueryCellInfo

It is possible to format and customize almost every child elements of scheduler such as work cells, header cells, time cells and so on using QueryCellEvent event.

The following code snippet shows how to customize the appointment and work cells based on the query cell info event.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ScheduleClientSideEvents(event => event.QueryCellInfo("checkFormat"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

While loading the scheduler using above code, the below function gets triggered by the QueryCellInfo event which customizes the corresponding DOM element.

HTML

```
<script type="text/javascript">
function checkFormat(args) {
switch (args.requestType) {
case "workcells":
args.element.css("background-color", "#ffe9cc");
break;
case "monthcells":
args.element.css("background-color", "#faa41a");
args.element.css("border-color", "#faa41a");
break;
}
}
</script>
```

The Scheduler elements are listed below which can be formatted through this event. The names are listed in the format with which it can be accessed or used within the requestType argument of the event.

Request Type	Description
appointment	Depicts the appointment element within the Scheduler.
agendacells	Depicts the Agenda Cell element within the Scheduler.
alldaycells	Depicts the AllDay cell element within the Scheduler.
headercells	Depicts the header cell element within the Scheduler.
resourceheadercells	Depicts the resource header cell element within the Scheduler.
leftheadercells	Depicts the left empty space on header cell element within the Scheduler.
leftindentcells	Depicts the left empty space on date cell element within the Scheduler.
timecells	Depicts the left side time panel cell element within the Scheduler.
headerdate	Depicts the header date cell element within the Scheduler.
emptytd	Depicts the empty space above the vertical scroller within the Scheduler.
resourcegroupheader	Depicts the header group cell in horizontal orientation in the Scheduler.

Copyright © 2001 - 2020 Syncfusion Inc.

Request Type	Description
monthcells	Depicts the month cell element within the Scheduler.
workcells	Depicts the work cell element within the Scheduler.

Navigation

Navigation in Scheduler can be classified based on Scheduler views, date and also the appointments in it.

View Navigation

By default, all the available view options except the custom view are available at the top right corner of the Schedule header, which can be traverse through continuously as and when needed.

Clicking on the particular date header in the Week/Work Week/Month/Custom View will navigate to the day view automatically. Also, clicking on the week header ranges displayed at the left side in the month view will navigate to the Week view. These particular actions can take place only if the Week and Day view options are present in the Views Collection.

Handling View navigation actions

Usually, the Navigation event gets triggered whenever the views/dates are being navigated. To block the navigation to day and week views from month view, the **Navigation** event can be used in the following way.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event => event.Navigation("onNavigation"))
)
```

JS

```
function onNavigation(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// args.target.currentTarget - target element which is clicked.
var target = $(args.target.currentTarget);
if (args.requestType == "viewNavigate" && (target.hasClass("e-
headercells") || target.hasClass("e-monthheader") || target.hasClass("e-
timecells")))
args.cancel = true;
}
```

Note: Based on the navigation, the appointments that lies between the particular date ranges of the current view are fetched and rendered in the Scheduler.

Date Navigation

The Scheduler date can be navigated on two aspects either in a continuous or random manner. On pressing the previous and next navigation arrow icons in the Scheduler header will move the scheduler one date back and forth respectively.

Another way of navigating through date is by making use of the built-in calendar available within the Scheduler, which pops out when the header date range is clicked. Selecting any date in the calendar will make the Scheduler to move to that particular date appropriately.

Handling date navigation actions

To handle the date navigation actions, the Navigation event can be used. For example, to block the date navigation, follow the below code example.

HTML

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
.ScheduleClientSideEvents(event => event.Navigation("onNavigation"))
)
```

JS

```
function onNavigation(args) {
//args.target - target element which is clicked.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//args.currentDate - current date of the Scheduler.
//args.requestType - Specifies the navigation type.
if (args.requestType == "dateNavigate")
args.cancel = true;
}
```

Appointment Navigation

The Appointment navigation bars (labeled **Previous/Next Appointment**) are rendered parallel to each other on the left and right centric corners of the Schedule control. It is controlled by an API ShowAppointmentNavigator which is set to true by default. When it is set to false, these bars will not be displayed on the Scheduler.

Whenever the previous/Next Appointment bars are clicked, it navigates the Scheduler to the corresponding closest date where the appointments are available. If no appointments are available beyond the current date, then these appointment bars will be in a disabled state.

The following code example shows the way to define the **ShowAppointmentNavigator** property for Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowAppointmentNavigator(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Template

Template is applicable to all the below specified elements of the Scheduler,

- Appointments
- Cells
- Date header
- Resource header

Copyright © 2001 - 2020 Syncfusion Inc.

- Priority field
- TimeScale
- Date and time columns in Agenda view
- Tooltip

Appointment Template

The template design that applies on for the Scheduler appointments. The field names that are mapped from the dataSource to the appropriate field properties within the AppointmentSettings can be accessed within the template.

Apart from the dataSource field names, the template can also access the current view of the Scheduler using the name **View** – which can contain either of the following values in lowercase.

- day
- week
- workweek
- agenda
- month
- customview

It is controlled by an API named AppointmentTemplateId which accepts the id value of the template design block preceded by a symbol **#**.

Usually, the appointments are displayed with its **Subject** and **Start/End time** on the Scheduler. If suppose, the subject needs to be accompanied with location text, it can be done with the following code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentTemplateId("#appTemplate")
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

HTML

```
<!-- Template for Appointment -->
<script id="appTemplate" type="text/x-jsrender">
{{"{{"}}if View !== "agenda"{{}}}}
<div style="height:100%; background-color:orange; margin-left: 5px;">
<div style="margin-left: 2px;">{{"{{"}}:Subject{{}}}}</div>
<div style="margin-left: 2px;">{{"{{"}}:Location{{}}}}</div>
</div>
{{"{{"}}else{{}}}}
<div>{{"{{"}}:Subject{{}}}}{{"{{"}}:Location{{}}}}</div>
{{"{{"}}/if{{}}}}
</script>
```

Cells Template

The template design that applies on the Scheduler elements such as all-day cells, work cells and month cells which allows the customization to be done based on the date, view, resources and timescale. The cells can be customized to add images, colors, and other elements etc and can also access the current view of the Scheduler using the name **view**.

All-day cells - An API named `AllDayCellsTemplateId` can be used to customize the all-day cells, which accepts the id of the template design block preceded with a symbol **#**.

Work cells and Month cells - An API named `WorkCellsTemplateId` can be used to customize the work cells in all the views, which accepts the id of the template design block preceded by a symbol **#**.

The cells can be customized with the following code example.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AllDayCellsTemplateId("#allDayTemplate") // Alldaycells Template ID
.WorkCellsTemplateId("#workTemplate") // Workcells Template ID
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<!-- Template for Alldaycells -->
<script id="allDayTemplate" type="text/x-jsrender">
<div class="e-icon e-scheduleallday" style="opacity:0.5"></div>
<span style="opacity:0.5">AllDay</span>
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<!-- Template for Workcells and Monthcells -->
<script id="workTemplate" type="text/x-jsrender">
{{"{{"}}if resource.classname == 'e-parentnode'{{}}}}
{{"{{"}}:resource.text{{}}}}
{{"{{"}}else{{}}}}
{{"{{"}}if date.getDay() == 0 || date.getDay() == 6{{}}}}
<div style="background-color:lightblue">Weekend</div>
{{"{{"}}else{{}}}}
{{"{{"}}if view == 'month' && resource.text == 'Party Hall-A' &&
date.getDay() == 5{{}}}}
<div style="background-color:burlywood">Meeting</div>
{{"{{"}}else resource.text != 'Party Hall-B' && date.getDate() ==
15{{}}}}
<div style="background-color:thistle">Holiday</div>
{{"{{"}}else view != 'month' && resource.text == 'Party Hall-A' &&
date.getDay() == 5 && date.getHours() == 10{{}}}}
<div style="background-color:burlywood">Meeting</div>
{{"{{"}}else view == 'month' && resource.text == 'Party Hall-B' &&
date.getDay() == 5{{}}}}
<div style="background-color:lightblue">Conf.</div>
{{"{{"}}else resource.text == 'Party Hall-B' && date.getDate() ==
16{{}}}}
<div style="background-color:darkkhaki">HappyDay</div>
{{"{{"}}else view != 'month' && resource.text == 'Party Hall-B' &&
date.getDay() == 5 && date.getHours() == 12{{}}}}
<div style="background-color:goldenrod">Conf.</div>
{{"{{"}}else date.getDate() == 10 && date.getMonth() == 11{{}}}}
<div style="background-color:palegreen">Day Special</div>
{{"{{"}}else date.getDate() == 25 && date.getMonth() == 11{{}}}}
<div style="background-color:sandybrown">Christmas</div>
{{"{{"}}/if{{}}}}
{{"{{"}}/if{{}}}}
{{"{{"}}/if{{}}}}
</script>
```

Date Header Template

The template design that applies on for the date header part of the Scheduler. An API named DateHeaderTemplateId] can be used to customize the date header which accepts the id value of the template design block preceded by a symbol **#**. The template can also access the current view of the Scheduler in using the name **view**.

The Date header can be customized with the following code example.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.DateHeaderTemplateId("#dateTemplate") // Date header Template ID
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<!-- Template for Date header -->
<script id="dateTemplate" type="text/x-jsrender">
<div>{{"{{"}}:~dTemplate(date){{}}}}</div>
</script>
<script>
function _dateFormat(date) {
var dFormat = ej.format(new Date(date), "dd/MM");
return dFormat;
}
$.views.helpers({ dTemplate: _dateFormat });
</script>
```

Resource Header Template

The template structure that applies on the resource headers of the Scheduler. By default, only the resource names will be displayed on the resource header bar. Also, the way of rendering resource headers on the Scheduler is comparatively different for both the vertical and horizontal scheduler views.

The field names that are mapped from the dataSource to the appropriate field properties within the **ResourceSettings** can be accessed within the resource header template.

Resource Header Template in Vertical View

To customize the resource header with some additional images or other customizations in **Vertical Scheduler View** – refer the below code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Datasource for Grouping -->
List<String> Group = new List<String>();
Group.Add("Rooms");
<!-- Datasource for Rooms -->
List<ResourceFields> Room = new List<ResourceFields>();
Room.Add(new ResourceFields { Id = "1", Text = "Room1", Color =
"#f8a398", GroupId = "1" });
Room.Add(new ResourceFields { Id = "2", Text = "Room2", Color =
"#56ca85", GroupId = "2" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Group(gr => { gr.Resources(Group); })
.ResourceHeaderTemplateId("#resTemplate")
.Resources(res => {
res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).Resou
rceSettings(fields =>
fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("Gro
upId")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("RoomId"))
)
```

HTML

```
<!-- Template for ResourceHeader in Vertical -->
<script id="resTemplate" type="text/x-jsrender">
<div style="height:100%">
<div style="width:15px;height:15px;margin-left:150px;margin-
top:2px;float:left;background:{{"{{"}}:Color{{}}}};"></div><div
style="float:left;margin-left:5px;">{{"{{"}}:Text{{}}}}</div>
</div>
</script>
```

Resource Header Template in Horizontal View

To perform the above specified same customization in **Horizontal Scheduler view**, the template structure varies a little bit as depicted below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", RoomId = "1" });
<!-- Datasource for Grouping -->
List<String> Group = new List<String>();
Group.Add("Rooms");
<!-- Datasource for Rooms -->
List<ResourceFields> Room = new List<ResourceFields>();
Room.Add(new ResourceFields { Id = "1", Text = "Room1", Color =
"#f8a398", GroupId = "1" });
Room.Add(new ResourceFields { Id = "2", Text = "Room2", Color =
"#56ca85", GroupId = "2" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Orientation(Orientation.Horizontal)
.Group(gr => { gr.Resources(Group); })
.ResourceHeaderTemplateId("#resTemplate")
.Resources(res => {
res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).Resou
rceSettings(fields =>
fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("Gro
upId")).Add();
})
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("RoomId"))
)
```

HTML

```
// Template for ResourceHeader in Horizontal
<script id="resTemplate" type="text/x-jsrender">
<div style="height:100%">
<div style="width:15px;height:15px;margin-
top:2px;float:left;background:{{"{{"}}:Color{{}}}};"></div><div
style="float:left;margin-left:5px;">{{"{{"}}:Text{{}}}}</div>
</div>
</script>
```

Note: In horizontal Scheduler, the header template makes use of an additional field namely **classname** which holds either **e-parentnode** or **e-childnode** value. The field **classname** can be used in the application scenario to check for the parent or child header node. You can apply the different template customization accordingly based on it.

TimeScale with Template

The TimeScale is an object collection that holds below timescale related properties such as,

The TimeScale is also availed with template options to allow customization. It includes the following 2 properties for customization -

- MajorSlotTemplateId - Accepts the id value of the template design block preceded by a symbol **#**, which gets applied for the major time slots.
- MinorSlotTemplateId - Accepts the id value of the template design block preceded by a symbol **#**, which gets applied for the minor time slots.

Copyright © 2001 - 2020 Syncfusion Inc.

The template customization for major and minor timeslots can be referred from the following code example.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.TimeScale(ts =>
ts.Enable(true).majorSlot(60).majorSlotTemplateId("#majorTemplate").minor
SlotCount(6).minorSlotTemplateId("#minorTemplate"))
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<!-- Template for Majorslot -->
<script id="majorTemplate" type="text/x-jsrender">
<div>{{"{{"}}:~major(date){{}}}}</div>
</script>
<!-- Template for Minorslot -->
<script id="minorTemplate" type="text/x-jsrender">
<div>{{"{{"}}:~minor(date){{}}}}</div>
</script>
<script>
function _majorFormat(date) {
var dFormat = ej.format(new Date(date), "hh:mm:ss");
return dFormat;
}
$.views.helpers({ major: _majorFormat });
function _minorFormat(date) {
var dFormat = ej.format(new Date(date), "hh:mm:ss");
return dFormat;
}
$.views.helpers({ minor: _minorFormat });
</script>
```

Priority Settings Template

The template design which can be applied to the content of the priority field in the appointment window. By default, the appropriate icons are displayed for each priority options such as **None**, **High**, **Medium** and **Low**.

When template is applied for the PrioritySettings, these default icons will be replaced by the custom icons or styles defined newly. The following code example depicts the way to enable the priority settings and to define the new custom styles to replace the default icons in the Priority field.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", Priority = "critical" });
<!-- Datasource for PrioritySettings -->
List<PrioritySettings> Priority = new List<PrioritySettings>();
Priority.Add(new PrioritySettings { Text = "none", Value = "none" });
Priority.Add(new PrioritySettings { Text = "critical", Value = "critical"
});
Priority.Add(new PrioritySettings { Text = "ultra-critical", Value =
"ultra-critical" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.PrioritySettings(priority =>
priority.Enable(true).Datasource(Priority).Text("Text").Value("Value").Te
mplate("<div class='${Text}'></div>"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Priority("Priority")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<!-- Style for Priority Template -->
<style type="text/css">
.critical,
.ultra-critical,
.none {
height: 13px;
width: 13px;
float: left;
margin-right: 4px;
background-repeat: no-repeat;
background-size: 60px;
padding: 1px;
margin-top: 2px;
}
.critical {
background-color: orange;
background-position: -13px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ultra-critical {
background-color: #56ca85;
background-position: -59px;
}
</style>
```

The custom style class names defined for the priority template should be same as that of the values defined for each priority option within the dataSource, so that it applies properly.

Note: Additionally, the priority field within the AppointmentSettings should be defined with appropriate dataSource field name. When an appointment is assigned with a priority value, the custom style/icon defined for that priority option will get applied over that appointment.

Tooltip Template

The tooltip can be applied with the customized template design. Currently the tooltip support is provided only for the appointments and the default tooltip displays the Subject and duration on hovering across the appointments.

By making use of template feature with tooltip, all the field names that are mapped from the dataSource to the appropriate field properties within the **AppointmentSettings** can be accessed.

To define the template option for tooltip, the TooltipSettings must be enabled first. The following code example depicts the way to add the tooltip template.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", Location = "US" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.TooltipSettings(tooltip =>
tooltip.Enable(true).TemplateId("#tooltipTemplate"))
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Location("Location")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
<!-- Template for Tooltip -->
<script id="tooltipTemplate" type="text/x-jsrender">
<div style="width:145px">
<div style="padding-top:3px;">
<div style="float:left; font:13px Segoe UI; font-
weight:bold;">Subject: </div>
<div style="padding-top:2px; font:12px Segoe UI
SemiBold;">{{"{{"}}:Subject{{}}}}</div>
</div>
<div style="padding-top:3px">
<div style="float:left; font:13px Segoe UI; font-
weight:bold;">Location: </div>
<div style="padding-top:2px; font:12px Segoe UI
SemiBold;">{{"{{"}}:Location{{}}}}</div>
</div>
</div>
</script>
```

Agenda View Templates

Agenda View provides two separate templates – one for date column and another for time column. These templates allows the customization of the content of both the date and time columns. Apart from this, the event column can also be customized through the existing API named `AppointmentTemplateId`.

The following code snippet shows how to customize the content of the date, time and event column.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
// Datasource for Appointments
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AgendaViewSettings(agenda =>
agenda.TimeColumnTemplateId("#timeTemplate").DateColumnTemplateId("#dateT
emplate"))
.AppointmentTemplateId("#appTemplate")
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
<!-- Template for date column -->
<script id="dateTemplate" type="text/x-jsrender">
<div style="height:100%">
<div>
<div>{{"{{"}}:~dateDisplay(StartTime){{}}}}</div>
</div>
</div>
</script>
<!-- Template for time column -->
<script id="timeTemplate" type="text/x-jsrender">
<div style="height:100%">
<div>
<div>{{"{{"}}:~timeDisplay(StartTime){{}}}}</div>
</div>
</div>
</script>
<!-- Template for appointment which applies for event column in agenda
view. -->
<script id="appTemplate" type="text/x-jsrender">
{{"{{"}}if View !== "agenda"{{}}}}
<div style="height:100%; background-color:orange; margin-left: 5px;">
<div style="margin-left: 2px;">{{"{{"}}:Subject{{}}}}</div>
<div style="margin-left: 2px;">{{"{{"}}:Location{{}}}}</div>
</div>
{{"{{"}}else{{}}}}
<div>{{"{{"}}:Subject{{}}}} {{"{{"}}:Location{{}}}}</div>
{{"{{"}}/if{{}}}}
</script>
```

JS

```
function _getDate(date) {
var dateCol = new Date(date);
return dateCol.toDateString();
}
function _getTime(date) {
var time = new Date(date);
return time.toLocaleTimeString();
}
$.views.helpers({
dateDisplay: _getDate,
timeDisplay: _getTime
});
```

Globalization and Localization

Globalization

The Scheduler control is built with default **globalization** support as it format the dates according to the user's locale automatically and processes it internally without any need for manual conversions. This kind of default handling of Scheduler dates is achieved through the built-in **ej.globalize** library which globalize the date, day and month names accordingly.

Copyright © 2001 - 2020 Syncfusion Inc.

Localization

Scheduler also comes with default localization support which allows it to customize the display of text within the Scheduler in a user-specific culture and locale. The Schedule control can be localized in specific culture using the common API Locale along with the collection of localized words defined for that culture using the ej.Schedule.Locale [**culture-code**].

Note: By default, the Schedule control is localized in **en-US** culture.

To localize Scheduler into any particular culture, it is necessary to refer the culture-specific script files in your application after the reference of **ej.web.all.min.js** file, which are available under the following location.

<Installed location>\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

The following code example shows how to localize the Schedule control in **fr-FR** culture.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.Locale("fr-FR")
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<script src="@Url.Content("~/Scripts/i18n/ej.culture.fr-
FR.min.js")"></script>
<script>
ej.Schedule.Locale["fr-FR"] = {
ReminderWindowTitle: "Fenêtre de rappel",
CreateAppointmentTitle: "créer un rendez-",
RecurrenceEditTitle: "Modifier répétition nomination",
RecurrenceEditMessage: "Comment voulez-vous changer le cas dans la
série?",
RecurrenceEditOnly: "Seulement cette nomination",
RecurrenceEditSeries: "La série entière",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
PreviousAppointment: "Nomination précédente",
NextAppointment: "prochain rendez-vous",
AppointmentSubject: "sujet",
StartTime: "Heure de début",
EndTime: "Heure de fin",
AllDay: "toute la journée",
Today: "aujourd'hui",
Recurrence: "répétition",
Done: "Terminé",
Cancel: "annuler",
Ok: "Ok",
RepeatBy: "Répétez par",
RepeatEvery: "répéter chaque",
RepeatOn: "répéter l'opération sur",
StartsOn: "démarre sur",
Ends: "extrémités",
Summary: "résumé",
Daily: "quotidien",
Weekly: "hebdomadaire",
Monthly: "mensuel",
Yearly: "annuel",
Every: "tous",
EveryWeekDay: "chaque jour de la semaine",
Never: "jamais",
After: "après",
Occurence: "apparition",
On: "sur",
Edit: "Modifier",
RecurrenceDay: "Jour (s)",
RecurrenceWeek: "Semaine (s)",
RecurrenceMonth: "Mois (s)",
RecurrenceYear: "Année (s)",
The: "la",
OfEvery: "de chaque",
First: "première",
Second: "deuxième",
Third: "troisième",
Fourth: "quatrième",
Last: "dernier",
WeekDay: "jour de la semaine",
WeekEndDay: "Jour de week-end",
Subject: "sujet",
Categorize: "Catégories",
DueIn: "En raison",
DismissAll: "rejeter tout",
Dismiss: "rejeter",
OpenItem: "Ouvrir l'élément",
Snooze: "répétition",
Day: "jour",
Week: "semaine",
WorkWeek: "Semaine de travail",
Month: "Mois",
AddEvent: "Ajouter événement",
CustomView: "Vue personnalisée",
Agenda: "ordre du jour",
Detailed: "détaillé",
EventBeginsin: "Nomination commence dans",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Editevent: "Modifier nomination",
Editseries: "Modifier série",
Times: "fois",
Until: "jusqu'à",
Eventwas: "rendez-vous était",
Hours: "hrs",
Minutes: "minutes",
Overdue: "en retard",
Days: "jour (s)",
Event: "Sujet",
Select: "sélectionner",
Previous: "prev",
Next: "suivant",
Close: "proche",
Delete: "effacer",
Date: "date",
Showin: "montrer en",
Gotodate: "Aller à la date",
Resources: "RESSOURCES",
RecurrenceDeleteTitle: "Supprimer répétition rendez-",
Location: "emplacement",
Priority: "priorité",
RecurrenceAlert: "alerte",
WrongPattern: "Le modèle de récurrence est pas valable",
CreateError: "La durée de la nomination doit être plus courte que la
façon dont elle se produit fréquemment. Raccourcir la durée ou changer le
modèle de récurrence dans la boîte de dialogue Récurrence de rendez.",
DragResizeError: "Impossible de replanifier une occurrence du rendez-vous
récurrent. si elle saute sur une occurrence ultérieure du même rendez-
vous.",
StartEndError: "L'heure de fin doit être supérieur à l'heure de début",
MouseOverDeleteTitle: "supprimer un rendez-",
DeleteConfirmation: "Êtes-vous sûr de vouloir supprimer ce rendez-vous?",
Time: "Temps"
};
</script>
```

Note: Refer the **ej.culture.fr-FR.min.js** file in your HTML application and also define the **Locale** property for the Schedule control with the appropriate **culture-code** [**fr-FR**].

For further information on – how to refer the required culture scripts into your application, refer here.

Localizing Specific Words

To customize or localize only some specific words in the default ej.Schedule.Locale[“en-US”] collection, the words to be localized/customized can be defined in a separate variable and then extended to the original collection as depicted in the following code example.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<script>
var customizationMessage = {
// customize the appointment window title
CreateAppointmentTitle: "Create Event",
// customize the view options text in the Schedule header
Day: "1 Day",
Week: "7 Days",
WorkWeek: "5 Days",
Month: "Month"
};
// Extend only the required changes to the original locale collection
$.extend(ej.Schedule.Locale["en-US"], customizationMessage);
</script>
```

Time Zone

The Scheduler makes use of the System time zone by default. If it needs to follow some other user-specific time zone, then the API TimeZone can be used. Also, the Scheduler can be set to observe the Daylight Saving Time (DST) with its **isDST** property which is set to **false** by default.

When IsDST property is set to **true**, the Scheduler internally processes the time difference values (for the Start and end time of the appointments) related to the Scheduler time zone that observes daylight savings time.

The following code example shows the way to set the specific time zone value with the daylight savings time observed in the Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
<@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.TimeZone("UTC +05:30")
.IsDST(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Different TimeZone for Scheduler Appointments

Apart from the default action of applying specific timezone to the entire Scheduler, it is also possible to set different time zone values for each appointments through the properties **startTimeZone** and **endTimeZone** which can be defined as separate fields within the appointment dataSource. When these properties are not explicitly defined for appointments, the appointments Start and End time will be processed based on the Scheduler time zone.

Note: The **IsDST** property closely relies on the appointment fields like StartTimeZone and EndTimeZone, for appropriate time difference calculations. If these two fields are not defined for appointments, then **IsDST** depends on the System **TimeZone** value.

The following code snippet shows how to define isDST and the time zones for specific appointments.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", StartTimeZone = "UTC +02:00", EndTimeZone =
"UTC +02:00" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.IsDST(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.StartTimeZone("StartTimeZone")
.EndTimeZone("EndTimeZone"))
)
```

Customizing TimeZone Collection

It is also possible to define or customize the default time zone collection of the Scheduler, by using the TimeZoneCollection API as follows.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- Datasource for TimezoneCollection -->
List<TimezoneCollection> TimeZone = new List<TimezoneCollection>();
TimeZone.Add(new TimezoneCollection { Text = "UTC -04:00", Id = "10",
Value = "UTC -04:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC -03:30", Id = "11",
Value = "UTC -03:30" });
TimeZone.Add(new TimezoneCollection { Text = "UTC -03:00", Id = "12",
Value = "UTC -03:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC -02:00", Id = "13",
Value = "UTC -02:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC -01:00", Id = "14",
Value = "UTC -01:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +00:00", Id = "15",
Value = "UTC +00:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +01:00", Id = "16",
Value = "UTC +01:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +02:00", Id = "17",
Value = "UTC +02:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +03:00", Id = "18",
Value = "UTC +03:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +03:30", Id = "19",
Value = "UTC +03:30" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +04:00", Id = "20",
Value = "UTC +04:00" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +04:30", Id = "21",
Value = "UTC +04:30" });
TimeZone.Add(new TimezoneCollection { Text = "UTC +05:00", Id = "22",
Value = "UTC +05:00" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.TimeZoneCollection(fields =>
fields.Datasource(TimeZone).Text("Text").Id("Id").Value("Value"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Priority("Priority")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: The values defined within the **TimeZoneCollection** dataSource are usually the only options displayed within the start and end time zone dropdown fields of the appointment window.

Time Mode

The time mode of the Scheduler can be either **12** or **24 hours** format which is based on the Locale set to the Scheduler. Since the default locale value of the Scheduler is **en-US**, therefore the time mode will be set to **12 hours** format (by default) automatically based on the culture.

The user can also set specific time mode for the Scheduler using TimeMode property which accepts either **String** or **enum** value. It accepts the following **enum** values,

- TimeMode.Hour12
- TimeMode.Hour24

The following code snippet shows the way to set specific **24 hour format** time mode for the Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.TimeMode(TimeMode.Hour24)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: If the **TimeMode** property is not set with specific value, then the value will be taken based on the locale assigned for the Scheduler.

Date Format

Scheduler can be used with all valid date formats. The default date format used in Scheduler is "MM/dd/yyyy".

If the DateFormat property is not specified particularly, then it will be taken based on the locale that is assigned to the Scheduler. The default locale applied on the Scheduler is “en-US”, which makes it to follow the "MM/dd/yyyy" pattern by default.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.DateFormat("yyyy/MM/dd")
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

First Day of Week

The FirstDayOfWeek property allows to set any of the week days as start of the week/workweek/month view in Scheduler. It accepts either the integer (Sunday=0, Monday=1, Tuesday=2, etc) or string (“Sunday”, “Monday”, etc) value. The default value of this FirstDayOfWeek property depends on the current culture (language) assigned to the Scheduler.

To set different first day of week in Scheduler,

RAZOR

```
@using Syncfusion.JavaScript.Models;
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 9))
.FirstDayOfWeek(DayOfWeek.Monday)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: The sub-control datepicker which is used within Scheduler for start/end time fields in appointment window and for date navigation purposes will also follow the same first day of week.

Keyboard Navigation

The shortcut keys for accessing the sub-elements of the scheduler and other Scheduler actions are tabulated in the following table.

Keys	Functionality description
arrow keys	To traverse through the Scheduler cells.
Shift+arrow keys	Multiple cell selection
Enter	Pressing enter key, after a single/multiple scheduler cell selection will make the quick appointment window to pop-up. Also, when the focus is being moved on to the fields like checkbox or buttons of the appointment window, pressing enter will select that particular action.
Esc	Closes any of the popup that displays on the Schedule control
Alt+N	Opens the Create Appointment window
Ctrl+E	Opens the Edit Appointment window
Alt+C	Opens the calendar popup within the Scheduler.
Ctrl+left arrow	Previous date navigation
Ctrl+right arrow	Next date navigation
Alt++	Forward traversing of view items in the toolbar
Alt+-	Reverse traversing of view items in the toolbar

Copyright © 2001 - 2020 Syncfusion Inc.

Space	When the previous/next navigation icons are currently being focused, pressing space navigates through the corresponding dates. Also, when the focus is being moved on to the fields like checkbox or buttons of the appointment window, pressing enter will select that particular action.
Del	Deletes the currently selected appointment
Tab	Traversing through the appointments.
Shift+tab	Traversing through the appointments in a forward direction

Note: By default AllowKeyboardNavigation property is set to **true**, which allows the Scheduler to be accessed through the above specified keys.

Setting Dimension

The dimension normally refers to the height and width of the element. With Scheduler, it is possible to set 2 kinds of dimensions.

- Scheduler dimension (Scheduler control height and width)
- Cell dimension (Scheduler cells height and width)

Scheduler Dimension

The Height and Width properties can be defined to set the outer dimension of the Scheduler control.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("70%")
.Height("500px")
.CurrentDate(new DateTime(2015,11, 5))
)
```

Note: The height and width properties accepts both **Pixel** and **Percentage** values.

Scheduler Cell Dimensions

Cell Height

The CellHeight property allows the Scheduler to set the height of the cells in pixels. The appointment height in vertical mode changes accordingly as per the cell size within which it renders.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CellHeight("40px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: In **Desktop** mode, the default height value of the cells is set to **20px**, whereas for **Mobile** mode – the Scheduler cells are rendered with **40px** by default.

Cell Auto-Height

The height of the cells specifically in timeline view can be made to adjust automatically based on its exceeding appointment count. It is controlled by an API named ShowOverflowButton which accepts true or false value, denoting whether to enable/disable the cell auto-height adjusting option. To enable this option, set the value of ShowOverflowButton as false whereas its default value is true.

In **Vertical** view, the same functionality is made applicable only in the **Month View** whereas in **Horizontal** mode, it is applicable in all the views.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.CurrentView(CurrentView.Month)
.ShowOverflowButton(false)
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Cell Width

The CellWidth property allows the Scheduler to set the width of the cells in pixels. The appointment width adjusts based on the cell width of the Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CellWidth("97px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: When the **CellHeight** and **CellWidth** properties are set with some specific pixel values, the cell size does not adapt to the responsive behavior of the Scheduler while resizing it in desktop/mobile mode.

Responsiveness

Scheduler is provided with default **responsive** support, so that it adjust or auto-resize it's UI content based on the window or the container size on which it is placed.

Auto-Resizing Scheduler

By default, setting 100% width to the Scheduler makes it adaptable to the window or its parent container, as and when the browser is resized. This will not allow the scheduler to adapt height-wise.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

To auto-resize the Scheduler height – set the **Height** property of the Scheduler to **100%** and also set some specific height (either percentage or pixel values) to its parent container (or) to the HTML and body tag elements as shown below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Height("800px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: When the Scheduler width is set to have **less than 600px**, then the View selection options will be displayed in a **navigation drawer**.

Also, whenever the Scheduler resizes - the width of the work cells adjusts automatically (if no **CellWidth** property is specified with pixel values), but the height of the cells are kept to remain as same as 20px until **CellHeight** property is set explicitly with some pixel values.

Scheduler in Mobile/Tablets

The Scheduler layout adapts automatically in the mobile and tablet devices with the appropriate UI changes, as the IsResponsive property is set to **true** by default. In case, if the user wants to display the normal scheduler in mobile devices without any adaptive enhancements, then the IsResponsive property can be set to false.

Some of the default changes done for adaptive Scheduler to render in mobile devices are as follows,

- Animation effect introduced between view/date navigation.
- Cell Height increased
- The header date display changes (displayed next to the navigation icons).
- View options displayed in Navigation drawer
- Time cell width decreased

Copyright © 2001 - 2020 Syncfusion Inc.

- Quick window display blocked on cell/appointment single click.
- Appointment resizing action blocked.
- Multiple cell selection blocked.
- Delete appointment option made available within the appointment window.
- Week header display in month view hidden.
- With Multiple resources - only one resource is viewable initially on the screen and to further look onto other resources, user need to swipe the screen.

Note: To make the Scheduler control to react as responsive in mobile/tablet devices, it is necessary to refer the additional [ej.responsive.css](http://cdn.syncfusion.com/{{ site.releaseversion }}/js/web/responsive-css/ej.responsive.css) file in the application.

The following code snippet depicts the Scheduler code with responsive set to true.

HTML

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.CurrentDate(new DateTime(2014, 6, 2))
.IsResponsive(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Persistence

State persistence allows the Scheduler to retain the current model value in the browser cookies for state maintenance. This action is handled through the property EnablePersistence which is set to false by default.

When it is set to **true**, some of the Schedule control model values will be retained even after refreshing the page which are listed below.

CurrentView	TimeMode
FirstDayOfWeek	DateFormat
IsDST	TimeZone
TimeScale	StartHour

Copyright © 2001 - 2020 Syncfusion Inc.

EndHour	WorkHours
Height	Width
CellHeight	CellWidth
CurrentDate	MinDate
MaxDate	RenderDates
Orientation	Views
WorkWeek	AgendaViewSettings.DaysInAgenda
EnableLoadOnDemand	ShowLocationField
ShowAllDayRow	IsResponsive
EnableRecurrenceValidation	ShowOverflowButton
AllowDragAndDrop	ShowDeleteConfirmationDialog
ShowNextPrevMonth	AppointmentDragArea

The Schedule properties that are not retained while maintaining state persistence are included within the **ignoreOnPersist** list, which makes it not to persist by default.

Export and Print

Export Appointments to ICS file

The Appointments can be exported as a whole collection or else a single appointment alone can be exported from the Scheduler. By default, the appointments are exported to .ics format which can then be imported and used in any of the other external calendars.

Single Appointment Exporting

A single appointment can be exported by making use of its Id. You can achieve this functionality by making use of a client-side exportSchedule method by passing the id of an appointment to export as one of its parameter.

It can also be achieved in another way by enabling the context menu settings and then adding a custom menu option for export appointment functionality. When right clicked on an appointment, and **export Appointment** option is chosen, the exporting functionality can be handled through the MenuItemClick event, within which the exportSchedule method should be defined with the following parameters,

- Action name (to be called in the server-side)
- Server Event (optional)
- Appointment Id (mandatory for single Appointment exporting)

The following code example shows the way to export single appointment from the Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment Menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
AppMenu.Add(new Appointment { Id = "export", Text = "Export Appointment"
});
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items => items.Appointment(AppMenu)))
.ScheduleClientSideEvents(event =>
event.MenuItemClick("onMenuItemClick"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

JS

```
function onMenuItemClick(args) {
if (args.events.ID == "export") {
var obj = $("#Schedule1").data("ejSchedule");
obj.exportSchedule("ExportToICS", null, args.targetInfo.Id);
}
}
```

Note: The Id value of the appointment passed in the above code example can be retrieved in the server-side action through Request.Form["AppointmentId"], as the id passed from the script is stored as hidden value in the input field of the form under this name internally.

Exporting all Appointments

To export the entire Scheduler appointments, the same `exportSchedule` method can be used without passing the id value to its parameter list. To achieve this, keep an individual button to export, and when it is clicked - the Scheduler can be allowed to export all the appointments.

The following code example depicts the way to export all the Scheduler appointments as a whole.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
@(Html.EJ().Button("export").Text("Export").ClientSideEvents(event =>
event.Click("onClick")))
```

JS

```
function onClick() {
var obj = $("#Schedule1").data("ejSchedule");
obj.exportSchedule("ExportToICS", null, null);
}
```

The server-side action **ExportToICS** contains the following code example to export the Scheduler appointments.

C#

```
public void ExportToICS(FormCollection form)
{
IEnumerable data;
string JSONModel = Request.Form["ScheduleModel"];
var model = JsonConvert.DeserializeObject<Dictionary<string,
object>>(JSONModel);
var Id = Request.Form["AppointmentId"];
if(Id != null)
// To export single appointment
data = db.DefaultSchedules.Where(app => app.Id.ToString() ==
Id.ToString()).ToList();
else
// To export all the appointments
data = db.DefaultSchedules.ToList();
ScheduleExport obj = new ScheduleExport(model, data);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

PDF Export

Scheduler supports exporting it along with all its appointments in PDF format, for which the same exportSchedule method can be used without passing any id value to its parameter list. To achieve this, keep an individual button to export and when it is clicked, the Scheduler with appointments can be allowed to export as PDF.

The following code example depicts the way to export the Scheduler with appointments in PDF format.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.AppointmentSettings(fields => fields.Datasource(Model)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence"))
)
@Html.EJ().Button("submit").Text("Export").Width("80px").Height("30px").C
lientSideEvents(event => event.Click("onPDFExport"))
```

JS

```
function onPDFExport(args) {
var obj = $("#Schedule1").data("ejSchedule");
obj.exportSchedule("ExportAsPDF", null, null);
}
```

The server-side action **ExportAsPDF** contains the following code example to export the Scheduler.

C#

```
public ActionResult ExportAsPDF(string scheduleModel)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
SchedulePDFExport convert = new SchedulePDFExport();
// Here calling the public method to convert the model values to
ScheduleProperties type from string type
ScheduleProperties scheduleObject =
convert.ScheduleSerializeModel(Request.Form["ScheduleModel"]);
// Here converting the schedule appointments data into enumerable object
by deserializing
IEnumerable scheduleAppointments =
(IEnumerable)serializer.Deserialize(Request.Form["ScheduleProcesedApps"],
typeof(IEnumerable));
PdfExport exp = new PdfExport();
PdfPageSettings pageSettings = new PdfPageSettings(50f);
pageSettings.Orientation= PdfPageOrientation.Landscape;
// Here passing the theme values from enum collection
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
PdfDocument document = exp.Export(scheduleObject, scheduleAppointments,
ExportTheme.FlatSaffron, Request.Form["locale"], pageSettings);
document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
return RedirectToAction("PDFExportController");
}
```

Note: To pass the theme value as a string instead of the enum value, refer to the following code example of passing the string type theme value.

C#

```
PdfDocument document = exp.Export(scheduleObject, scheduleAppointments,
"flat-saffron",Request.Form["locale"]);
```

Setting Page Options:

It is possible to set/change the PDF page settings such as size, margins (left, right, top and bottom), transition, and rotate angle and header/footer values by passing the page settings and page document template object values into the export method.

Page Settings:

You can set/apply the following page property values to the PDF file before exporting it with the Scheduler by using the PdfPageSettings object.

- Page Size
- Orientation
- Margins (Left, Right, Top, Bottom)
- Transition (Direction, Dimension, Duration, Motion, PageDuration, Scale, Style)
- Rotate (RotateAngle0, RotateAngle90, RotateAngle180, RotateAngle270)
- Height
- Width

To apply the above page setting values, you need to create an object for the PdfPageSettings class. Then, you can assign the values to the available properties within it such as Size = PdfPageSize.A3, Orientation = PdfPageOrientation.Landscape and so on. Once done, pass this object into the export method which will set these page settings values to the PDF file before exporting it. Refer to the following code example (server-side) to set/change the page settings values.

C#

```
public ActionResult ExportAsPDF(string scheduleModel)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
SchedulePDFExport convert= new SchedulePDFExport();
ScheduleProperties scheduleObject =
convert.ScheduleSerializeModel(scheduleModel); // Here calling the public
method to convert the model values to ScheduleProperties type from string
type
IEnumerable scheduleAppointments =
(IEnumerable)serializer.Deserialize(Request.Form["ScheduleProcesedApps"],
typeof(IEnumerable)); // Here deserialize the appointments to enumerable
object
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
PdfExport exp = new PdfExport();
// Here the 50f is the default value for the margins. If you are not
assigning any separate values like Margins.Left = 15 means 50f value will
be setting to all the Margin properties.
PdfPageSettings pageSettings = new PdfPageSettings(50f);
// Here you can assign the PdfPageSize enum value (ex: A3, A4)
pageSettings.Size = PdfPageSize.A3;
// Here Landscape value assigned to the orientation. You can set either
Landscape/Portrait for this Orientation property
pageSettings.Orientation = PdfPageOrientation.Landscape;
// Here assigned different values for the margins
pageSettings.Margins.Left = 15;
pageSettings.Margins.Right = 15;
pageSettings.Margins.Top = 20;
pageSettings.Margins.Bottom = 20;
// Here assigned the Transition Direction as BottomToTop. You can also
set any of the Transition values to this property
pageSettings.Transition.Direction= PdfTransitionDirection.BottomToTop;
// Here assigned the Rotate Angle as RotateAngle180. You can also set any
of the Rotate values to this property
pageSettings.Rotate = PdfPageRotateAngle.RotateAngle180;
// Here passing the page settings object value into the export method.
PdfDocument document = exp.Export(scheduleObject, scheduleAppointments,
ExportTheme.FlatSaffron, Request.Form["locale"], pageSettings);
document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
return RedirectToAction("PDFExportController");
}
```

Header/Footer Settings:

You can also set the header and footer options to the PDF page before it is being exported, by passing the PDFDocumentTemplate object values as mentioned in the following code example.

C#

```
public ActionResult ExportAsPDF(string scheduleModel)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
SchedulePDFExport convert= new SchedulePDFExport();
ScheduleProperties scheduleObject =
convert.ScheduleSerializeModel(scheduleModel); // Here calling the public
method to convert the model values to ScheduleProperties type from string
type
IEnumerable scheduleAppointments =
(IEnumerable)serializer.Deserialize(Request.Form["ScheduleProcesedApps"],
typeof(IEnumerable)); // Here deserialize the appointments to enumerable
object
PdfExport exp = new PdfExport();
PdfDocument document = new PdfDocument();
PdfPage pdfPage = document.Pages.Add();
//Create a header and draw the string.
// Here the bounds value is used to store the position/axis value, where
the header and footer content to be displayed
RectangleF bounds = new RectangleF(0, 0,
document.Pages[0].GetClientSize().Width, 50);
// Creating object for the PdfPageTemplateElement
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
PdfPageTemplateElement header = new PdfPageTemplateElement(bounds);
// Here setting the font style and color to display the header/footer
content
PdfSolidBrush brush = new PdfSolidBrush(Color.Gray);
PdfFont font = new PdfStandardFont(PdfFontFamily.Helvetica, 6,
PdfFontStyle.Bold);
PdfStringFormat format = new PdfStringFormat();
format.Alignment = PdfTextAlignment.Center;
format.LineAlignment = PdfVerticalAlignment.Middle;
header.Graphics.DrawString("Schedule", font, brush, new RectangleF(0, 18,
header.Width, header.Height), format);
//Create a Page template that can be used as footer (here creating
template to display the page number).
PdfPageTemplateElement footer = new PdfPageTemplateElement(bounds);
//Create page number field.
PdfPageNumberField pageNumber = new PdfPageNumberField(font, brush);
//Create page count field.
PdfPageCountField count = new PdfPageCountField(font, brush);
//Add the fields in composite fields.
PdfCompositeField compositeField = new PdfCompositeField(font, brush,
"Page {0} of {1}", pageNumber, count);
compositeField.Bounds = footer.Bounds;
//Draw the composite field in footer.
compositeField.Draw(footer.Graphics, new PointF(470, 40));
PdfDocumentTemplate headerFooterTemplate = new PdfDocumentTemplate();
//Here assigning the header template value to the PdfDocumentTemplate
Object.
headerFooterTemplate.Top = header;
//Here assigning the footer template value to the PdfDocumentTemplate
Object.
headerFooterTemplate.Bottom = footer;
//Here passing the PdfDocumentTemplate Object value into the export
method.
document = exp.Export(scheduleObject, scheduleAppointments,
ExportTheme.FlatSaffron, Request.Form["locale"], headerFooterTemplate);
document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
return RedirectToAction("PDFExportController");
}
```

Note: The header and footer values can be passed to the PDF file only through the template option. In the above mentioned code example, the template has been written with the text “Schedule” to display it as the header text and the page number (ex: Page 1 of 2) has been displayed in the footer part.

It is also possible to pass both the PageSettings and header/footer template objects in the export method altogether to apply the page settings as well as the header and footer options to the PDF file which can be referred from the following code example,

C#

```
public ActionResult ExportAsPDF(string scheduleModel)
{
//Here add the code example of both the page settings and header/footer
values settings.
//Then pass both PdfPageSettings and PdfDocumentTemplate object into the
export method.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
document = exp.Export(scheduleObject, scheduleAppointments,
ExportTheme.FlatSaffron, Request.Form["locale"], pageSettings,
headerFooterTemplate);
document.Save("Schedule.pdf", HttpContext.ApplicationInstance.Response,
HttpReadType.Save);
return RedirectToAction("PDFExportController");
}
```

Print

Two types of Print options are available – either to print the entire Scheduler layout including all the appointments in it or else to print any particular appointment alone. The public method Print is used to print the entire Scheduler.

Print the Scheduler

The following code example shows the way to print the entire Scheduler, by keeping an extra button in a page – on which clicking the button will print the entire Scheduler.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
@Html.EJ().Button("print").Text("Print").ClientSideEvents(event =>
event.Click("onClick"))
```

JS

```
function onClick(args) {
var obj = $("#Schedule1").data("ejSchedule");
obj.print();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Printing Specific Appointments

To print a particular appointment, either the default context menu **print** option can be used or else the print method can be used by passing the id of the appointment to be printed as its argument.

Note: To handle this functionality, the context menu settings needs to be enabled with print option added to the appointment items.

The following code example depicts the way to print a particular appointment.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
<!-- List of Appointment Menu options -->
List<Appointment> AppMenu = new List<Appointment>();
AppMenu.Add(new Appointment { Id = "open", Text = "Open Appointment" });
AppMenu.Add(new Appointment { Id = "delete", Text = "Delete Appointment"
});
AppMenu.Add(new Appointment { Id = "print", Text = "Print Appointment"
});
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.ContextMenuSettings(contextMenu =>
contextMenu.Enable(true).MenuItems(items => items.Appointment(AppMenu)))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Using print() method

The client-side public method print needs to be used here by passing the appointment object as its argument, that needs to be printed.

For example, here the below code example depicts the way to print the particular appointment programmatically by calling the **print** function within the AppointmentClick event, which triggers whenever the user clicks on an appointment.

RAZOR

```
@using Syncfusion.JavaScript.Models;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 9))
.ScheduleClientSideEvents(event =>
event.AppointmentClick("onAppointmentClick"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

JS

```
function onAppointmentClick(args) {
var schObj = $("#Schedule1").data("ejSchedule");
schObj.print(args.appointment);
}
```

Import Appointments

To import appointments into the Scheduler, server-side method renderingImportAppointments needs to be used, which returns the list of appointments retrieved from the specified file path.

Refer the following code example to import the appointments into Scheduler.

RAZOR

```
@Html.EJ().Button("submit").Text("Submit").ClientSideEvents(event =>
event.Click("ScheduleImport"))
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2014, 6, 2))
.AppointmentSettings(fields => fields
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
function ScheduleImport() {
var dataManger = ej.DataManager({ url: '@Url.Action("ScheduleImportData",
"Schedule")', crossDomain: true });
$("#Schedule1").ejSchedule({
appointmentSettings: {
dataSource: dataManger
}
});
}
```

The server-side action ScheduleImportData contains the following code example to import the Scheduler appointments.

C#

```
public JsonResult ScheduleImportData() {
var destinationPath = @"FilePath\iCalender.ics";
ScheduleImport importApps = new ScheduleImport();
var app = importApps.renderingImportAppointments(destinationPath);
int intMax = app.Max(a => a.Id);
List<ScheduleAppointmentsObjData> result = new
List<ScheduleAppointmentsObjData>();
for (var i = 0; i < app.Count; i++) {
app[i].Id = intMax + 1;
ScheduleAppointmentsObjData row = new
ScheduleAppointmentsObjData(app[i].Id, app[i].Subject, app[i].Location,
app[i].StartTime, app[i].EndTime, app[i].Description, null, null,
app[i].Recurrence, null, null, app[i].AppointmentCategorize, null,
app[i].AllDay, null, null, app[i].RecurrenceRules, null, null);
result.Add(row);
intMax = app[i].Id;
}
return Json(result, JsonRequestBehavior.AllowGet);
}
```

Miscellaneous

Time Indicator

The current system time can be depicted in a scheduler with an indication of a piece of text displaying the time over a horizontal line across today's date (Vertical Scheduler mode). In Horizontal mode, the time indicator is displayed as a small vertical line within the header time cells (depicting current time).

It is enabled by default in Scheduler and to hide it, ShowCurrentTimeIndicator property needs to be set as **false** as shown below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowCurrentTimeIndicator(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Note: When Scheduler rendered with multiple resources, multiple time indicators are displayed in Vertical mode – whereas single indicator displays for horizontal mode.

Show/Hide All-Day Row

The All-day row cell is a single row region displayed at the top of the work cells area to hold the all-day appointments together. The Scheduler displays it by default and if it needs to be hidden, the ShowAllDayRow property can be set to **false** as shown below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowAllDayRow(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The All-day row expands vertically, whenever more number of appointments are populated in one or more days.

Show/Hide Header bar

The header bar is the top most region of the Scheduler which includes the date navigation icons and view navigation options – and also shown on the Scheduler by default. To hide the header bar, set the ShowHeaderBar property to **false** as shown below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowHeaderBar(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Show/Hide Location Field

The Location field in the default appointment window can be displayed or hidden from it using the ShowLocationField property. By default, this property is set to false. To display the location option in the appointment window, then set **ShowLocationField** to **true** and also bind the appropriate field to the location property within the appointmentSettings as shown below.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowLocationField(true)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.AllDay("AllDay")
.Location("EventLocation")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Recurrence Editor

The Recurrence Editor includes the entire recurrence related information in a separate portable manner which can be either utilized as a separate widget or else can be embed within the appointment window of Scheduler to enable recurrence options within it. The recurrence rule can be easily generated based on the frequency selected. The customizations such as changing the labels of the recurrence editor is also possible to achieve through its properties. The frequencies available are Never, Daily, Weekly, Monthly, Yearly and Every weekday.

Getting Started

Follow the steps as mentioned in the Getting Started page of the Introduction part to create an MVC application with required assembly references, scripts and stylesheets to render the Recurrence Editor.

Add the basic Recurrence Editor code in the View page as shown below,

RAZOR

```
@(Html.EJ().RecurrenceEditor("RecurrenceEditor")
.SelectedRecurrenceType(0)
)
```

Generating Recurrence Rule

The Recurrence Editor can be used to generate the recurrence rule as a string, based on the repeat options selected.

The following code example depicts the way to generate the recurrence rule.

RAZOR

```
@(Html.EJ().RecurrenceEditor("RecurrenceEditor")
.SelectedRecurrenceType(2)
.RecurrenceEditorClientSideEvents(evt=>evt.Create("onCreate"))
)
```

Define a button separately and add it to the page as shown below and while clicking on it, make use of the client-side public method `getRecurrenceRule` of the Recurrence Editor to generate the recurrence rule string explicitly.

HTML

Copyright © 2001 - 2020 Syncfusion Inc.

```
<button onclick="closeRecurrence()" id="donerecur1" class='recurbutton'
style="float:right;margin-right:20px;margin-bottom:10px;">Recurrence
String</button>
```

JS

```
$(function () {
$("#donerecur1").ejButton({ width: '155px', height: '35px',
showRoundedCorner: true });
$("#RecurrenceEditor").after($("#donerecur1"));
});
function closeRecurrence() {
var obj = $(".e-recurrenceeditor").data("ejRecurrenceEditor");
alert(obj.getRecurrenceRule()) ; // recurrence rule generated as per the
options selected in the recurrence editor will be displayed here
}
function onCreate() {
this.element.find("#recurrencetype_wrapper").css("width", "33%");
}
```

How To

Validate the Custom Appointment Window Fields

The client-side validation of the fields present within the custom appointment window can be handled before submitting it, by adding appropriate validation classes to each field.

Refer the steps here and create a sample for Custom Appointment window, before proceeding with the following validations.

In the custom appointment window sample, create an additional CSS class **validation** as mentioned below to add it to the appropriate fields, if the validation of such fields fails.

CSS

```
<style>
.validation {
border-color: red;
}
</style>
```

The sample contains the fields like Subject, Description, StartTime and EndTime which needs to be validated at client-side. To do so, add the required validation code within the script section as follows.

HTML

```
<script type="text/javascript">
// To Validate the Subject field.
$("#subject").focusout(function() {
if ($.trim($("#subject").val()) == "") {
$("#subject").addClass("validation");
return false;
}
})
// To Validate the Description field.
$("#customDescription").focusout(function() {
if ($.trim($("#customDescription").val()) == "") {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$("#customDescription").addClass("validation");
return false;
}
})
// To Validate the Time duration of the appointments
$("#EndTime").focusout(function() {
if (new Date($("#EndTime").val()).getDate() >= new
Date($("#StartTime").val()).getDate()) {
if (new Date($("#StartTime").val()).getTime() >= new
Date($("#EndTime").val()).getTime())
alert("EndTime value is lesser than the StartTime value");
}
})
</script>
```

Now, after adding the above validations – whenever the fields within the custom appointment window are skipped without filling any information, it will be notified to the users appropriately.

Highlight Different Work Hours for Each Resources

By default, the work hours of the Scheduler is highlighted based on the start and end values provided within the WorkHours property. It remains same for all the resources, when the Scheduler is rendered with multiple resources. To customize this behavior so as to highlight different working hour range for each of the resources, the following workaround can be utilized by making use of the Scheduler events Create and ActionComplete.

Initially, set the Highlight as false for the WorkHours, so as to disable the highlighting of default work hour range.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "", OwnerId = "1" });
<!-- Datasource for Grouping -->
List<String> Group = new List<String>();
Group.Add("Owners");
<!-- Datasource for Owners -->
List<ResourceFields> Owner = new List<ResourceFields>();
Owner.Add(new ResourceFields { Id = "1", Text = "Nancy", Color =
"#f8a398" });
Owner.Add(new ResourceFields { Id = "3", Text = "Steven", Color =
"#56ca85" });
Owner.Add(new ResourceFields { Id = "5", Text = "Michael", Color =
"#51a0ed" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ShowCurrentTimeIndicator(false)
.WorkHours(hrs => hrs.Highlight(false))
.Group(gr => gr.Resources(Group))
.Resources(res => {
res.Field("OwnerId").Title("Owner").Name("Owners").ResourceSettings(field
s =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color")).Add(); })
.ScheduleClientSideEvents(event =>
event.Create("onCreate").ActionComplete("onCreate"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule")
.ResourceFields("OwnerId"))
)
```

JS

```
// This function executes during the initial control creation and also on
completion of each action like date/view navigation.
function onCreate(args) {
if (args.requestType == "viewNavigate" || args.requestType ==
"dateNavigate" || args.type == "create") {
// declare different start and end work hour values for each resources
var resourceOneStart = 9,
resourceOneEnd = 18;
var resourceTwoStart = 13,
resourceTwoEnd = 18;
var resourceThreeStart = 10,
resourceThreeEnd = 18;
this.option("workHours.highlight", (this.currentView() != "month") ?
false : true);
var trElements = this.$WorkCellDiv.find("tr");    // Get the Scheduler
work cell rows
for (var i = 0; i < trElements.length; i++) {
var tdElements = $(trElements[i]).find("td");  // Get the Scheduler work
cell columns
for (var j = 0; j < tdElements.length; j++) {
switch (this.currentView()) {
case "day":
switch (j) {
case 0: // column index 0 represents first resource in day view
$(tdElements[j]).addClass(((i > (resourceOneStart * 2) - 1) && (i <=
(resourceOneEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
case 1: // column index 1 represents second resource in day view
$(tdElements[j]).addClass(((i > (resourceTwoStart * 2) - 1) && (i <=
(resourceTwoEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
case 2: // column index 2 represents third resource in day view
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
$(tdElements[j]).addClass(((i > (resourceThreeStart * 2) - 1) && (i <=
(resourceThreeEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
}
break;
case "week":
switch (j) {
case 0:
case 1:
case 2:
case 3:
case 4:
case 5:
case 6: // column indexes 0 to 6 belongs to first resource in week view
(7 days)
$(tdElements[j]).addClass(((i > (resourceOneStart * 2) - 1) && (i <=
(resourceOneEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
case 7:
case 8:
case 9:
case 10:
case 11:
case 12:
case 13: // column indexes 7 to 13 belongs to second resource in week
view (7 days)
$(tdElements[j]).addClass(((i > (resourceTwoStart * 2) - 1) && (i <=
(resourceTwoEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
case 14:
case 15:
case 16:
case 17:
case 18:
case 19:
case 20: // column indexes 14 to 20 belongs to third resource in week
view (7 days)
$(tdElements[j]).addClass(((i > (resourceThreeStart * 2) - 1) && (i <=
(resourceThreeEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
}
break;
case "workweek":
switch (j) {
case 0:
case 1:
case 2:
case 3:
case 4: // column indexes 0 to 4 belongs to first resource in workweek
view (5 days)
$(tdElements[j]).addClass(((i > (resourceOneStart * 2) - 1) && (i <=
(resourceOneEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
case 5:
case 6:
case 7:
case 8:
```

```
case 9:
// column indexes 5 to 9 belongs to second resource in workweek view (5
days)
$(tdElements[j]).addClass(((i > (resourceTwoStart * 2) - 1) && (i <=
(resourceTwoEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
case 10:
case 11:
case 12:
case 13:
case 14:
// column indexes 10 to 14 belongs to third resource in workweek view (5
days)
$(tdElements[j]).addClass(((i > (resourceThreeStart * 2) - 1) && (i <=
(resourceThreeEnd * 2) - 1)) ? "e-businesshighlightworkcells" : "");
break;
}
break;
}
}
}
}
}
```

Display Scheduler with Appointments Filtered by Subject

It is possible to display the Scheduler with appointments, which is filtered based on the Subject. For example, if we keep a text box and start typing a character – the list of appointments whose subject matches the typed character alone will be shown on the Scheduler. The following code example depicts the way to achieve this.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
Appoint.Add(new ScheduleFields { Id = "2", Subject = "Bering Sea Gold",
StartTime = new DateTime(2015, 11, 10, 01, 00, 00), EndTime = new
DateTime(2015, 11, 10, 02, 30, 00), Description = "", AllDay = false,
Recurrence = false, RecurrenceRule = "" });
Appoint.Add(new ScheduleFields { Id = "3", Subject = "Deadest Catch",
StartTime = new DateTime(2015, 11, 10, 11, 00, 00), EndTime = new
DateTime(2015, 11, 10, 11, 30, 00), Description = "", AllDay = false,
Recurrence = false, RecurrenceRule = "" });
Appoint.Add(new ScheduleFields { Id = "4", Subject = "What Happened
Next?", StartTime = new DateTime(2015, 11, 10, 15, 00, 00), EndTime = new
DateTime(2015, 11, 10, 17, 30, 00), Description = "", AllDay = false,
Recurrence = false, RecurrenceRule = "" });
Appoint.Add(new ScheduleFields { Id = "5", Subject = "Daily Planet",
StartTime = new DateTime(2015, 11, 10, 20, 00, 00), EndTime = new
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
DateTime(2015, 11, 10, 22, 30, 00), Description = "", AllDay = false,
Recurrence = false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowCurrentTimeIndicator(false)
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<!-- Textbox for entering search character -->
<input id='txtSearch' type='text' onkeyup='searchKeyUp()' />
<script>
function searchKeyUp() {
var searchString = $("#txtSearch").val();
var schObj = $("#Schedule1").data("ejSchedule");
var result = schObj.searchAppointments(searchString);
schObj.option("appointmentSettings", { dataSource: [] });
if (!ej.isNullOrUndefined(result) && result.length != 0 && searchString
!= "")
schObj.option("appointmentSettings", { dataSource: result });
else
schObj.option("appointmentSettings", { dataSource:
schObj._currentAppointmentData });
}
</script>
```

Customize the Default Appointment Window

Apart from the custom appointment window, it is possible to customize the default appointment window by adding/removing the required number of fields into it. This can be achieved through the Create event of the scheduler.

The following code example depicts the way to achieve the customization of default appointment window.

RAZOR

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ShowCurrentTimeIndicator(false)
.ScheduleClientSideEvents(event =>
event.AppointmentWindowOpen("onAppointmentOpen"))
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

JS

```
// This function executes before the appointment window gets opened.
function onAppointmentOpen(args) {
// to add custom element in default appointment window
if (this._appointmentAddWindow.find(".custom-fields").length == 0) {
var customDesign = "<tr class='custom-fields'><td class='e-
textlabel'>Event Type</td><td><input class='app-type'
type='text'/></td><td class='e-textlabel'>Event Status </td><td><input
class='status' type='text'/></td></tr>";
$(customDesign).insertAfter(this._appointmentAddWindow.find("." +
this._id + "arrow"));
}
if (!ej.isNullOrUndefined(args.appointment)) {
// if double clicked on the appointments, retrieve the custom field
values from the appointment object and fills it in the appropriate
fields.                this._appointmentAddWindow.find(".app-
type").val(args.appointment.AppointmentType);
this._appointmentAddWindow.find(".status").val(args.appointment.Status);
} else {
// if double clicked on the cells, clears the field values.
this._appointmentAddWindow.find(".app-type").val("");
this._appointmentAddWindow.find(".status").val("");
}
}
```

SignalR Implementation

To implement SignalR concept in Scheduler, add and refer the appropriate SignalR scripts in your project and then create a hub for initiating the action. The SignalR implementation can be achieved with the following code example.

RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@using MyProject.Models; // Here MyProject defines your project name to
access the model class
@{
<!-- Datasource for Appointments -->
List<ScheduleFields> Appoint = new List<ScheduleFields>();
Appoint.Add(new ScheduleFields { Id = "1", Subject = "Meeting", StartTime
= new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new DateTime(2015,
11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence =
false, RecurrenceRule = "" });
}
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 5))
.ScheduleClientSideEvents(d => {
d.BeforeAppointmentCreate("actionComplete");
d.Navigation("actionComplete");
d.BeforeAppointmentRemove("actionComplete");
d.BeforeAppointmentChange("actionComplete"); })
.AppointmentSettings(fields => fields.Datasource(Appoint)
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

HTML

```
<script src="~/Scripts/jquery.signalR-1.1.4.min.js"></script>
<script src="@Url.Content("~/signalr/hubs")"></script>
<script type="text/javascript">
$(function () {
window.signal = $.connection.scheduleHub;
window.signal.client.modify = function (action, data) {
if (data != null) {
var id = data.Id != null ? data.Id : data[0].Id;
var subject = data.Subject;
var startTime = new Date(data.StartTime);
var endTime = new Date(data.EndTime);
var description = data.Description;
var allDay = data.AllDay;
var recurrence = data.Recurrence;
var recurrenceRule = data.RecurrenceRule;
var object = { Id: id, Subject: subject, StartTime: startTime, EndTime:
endTime, Description: description, AllDay: allDay, Recurrence:
recurrence, RecurrenceRule: recurrenceRule };
var schedule = $("#Schedule1").data("ejSchedule");
if (action == "appointmentSaved") {
new ej.DataManager(schedule._currentAppointmentData).insert(object);
schedule._appointmentProcessing(object);
schedule._renderAppointmentAll();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
else if (action == "appointmentEdited") {
new ej.DataManager(schedule._currentAppointmentData).update("Id",
object);
new ej.DataManager(schedule._processed).remove("Id", object.Id);
schedule._appointmentProcessing(object);
schedule._renderAppointmentAll();
}
else if (action == "appointmentDeleted") {
var appointment = new
ej.DataManager(schedule._processed).executeLocal(new
ej.Query().where("Id", ej.FilterOperators.equal, id));
new ej.DataManager(schedule._currentAppointmentData).remove("Id",
appointment[0].Id);
new ej.DataManager(schedule._processed).remove("Id", appointment[0].Id);
schedule._renderAppointmentAll();
}
}
};
$.connection.hub.start().done(function () {
window.actionComplete = function (args) {
if(args.methodType=="public")
args.appointment=null;
var appointmentDetails = args.appointment;
if (args.type == "appointmentSaved" || args.type == "appointmentEdited"
|| args.type == "appointmentDeleted") {
window.signal.server.modify(args.type, appointmentDetails);
}
};
});
});
</script>
```

Create a Hub file and add the following action for accessing the client details.

C#

```
public void Modify(string action, Object data)
{
Clients.Others.modify(action, data);
}
```

Synchronize the Schedule with Outlook

Schedule appointments can be synchronized with Outlook and vice versa, by making use of the Microsoft Outlook 12/15 Object library.

The following code example depicts the way to synchronize the Schedule with Outlook.

RAZOR

```
@(Html.EJ().Schedule("Schedule1")
.Width("100%")
.Height("525px")
.CurrentDate(new DateTime(2015, 11, 12))
.AppointmentSettings(fields => fields.Datasource(dataSource =>
dataSource.URL("/Home/GetApp").CrudURL("/Home/Batch").Adaptor("UrlAdaptor
"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Id("Id")
.Subject("Subject")
.StartTime("StartTime")
.EndTime("EndTime")
.Description("Description")
.AllDay("AllDay")
.Recurrence("Recurrence")
.RecurrenceRule("RecurrenceRule"))
)
```

Schedule control is not directly compatible with the Outlook calendar object, as it returns the COM object. Therefore, in order to bind the schedule control with those data retrieved from outlook, then it is necessary to convert the COM object into the schedule acceptable object format. To do so, add the following code example in your code behind.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.ComponentModel;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Collections;
using Microsoft.Office.Interop.Outlook; // required Microsoft Assembly
using System.Web.Script.Serialization;
using System.Web.UI;
using System.Web.UI.WebControls;
using ScheduleCRUD.Models;
namespace ScheduleCRUD.Controllers
{
public class HomeController : Controller
{
DataClasses1DataContext db = new DataClasses1DataContext();
public ActionResult Index()
{
return View();
}
public JsonResult GetApp() // function to display the outlook
appointments in Schedule initially
{
IEnumerable data = GetData();
return Json(data, JsonRequestBehavior.AllowGet);
}
public List<ScheduleData> GetData() // function to return the outlook
appointments
{
Microsoft.Office.Interop.Outlook.Application oApp = new
Microsoft.Office.Interop.Outlook.Application();
Microsoft.Office.Interop.Outlook.NameSpace mapNamespace =
oApp.GetNamespace("MAPI");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Microsoft.Office.Interop.Outlook.MAPIFolder CalendarFolder =
mapNamespace.GetDefaultFolder(Microsoft.Office.Interop.Outlook.OlDefaultF
olders.olFolderCalendar);
Microsoft.Office.Interop.Outlook.Items outlookCalendarItems =
CalendarFolder.Items;
List<Microsoft.Office.Interop.Outlook.AppointmentItem> appointmentList =
new List<Microsoft.Office.Interop.Outlook.AppointmentItem>();
foreach (Microsoft.Office.Interop.Outlook.AppointmentItem item in
outlookCalendarItems)
{
appointmentList.Add(item);
}
List<ScheduleData> newApp = new List<ScheduleData>();
// converting COM object into IEnumerable object
for (var i = 0; i < appointmentList.Count; i++)
{
ScheduleData app = new ScheduleData();
app.Id = i;
app.Subject = appointmentList[i].Subject;
app.AllDay = appointmentList[i].AllDayEvent;
app.StartTime = Convert.ToDateTime(appointmentList[i].Start.ToString());
string endTime = appointmentList[i].End.ToString();
DateTime appEndDate = Convert.ToDateTime(endTime);
if (endTime.Contains("12:00:00 AM") && app.AllDay == true)
app.EndTime = appEndDate.AddMinutes(-1);
else
app.EndTime = appEndDate;
app.Recurrence = false;
app.RecurrenceRule = null;
newApp.Add(app);
}
return newApp;
}
public JsonResult Batch(EditParams param)
{
if (param.action == "insert" || (param.action == "batch" && param.added
!= null)) // this block of code will execute while inserting the
appointments
{
var value = param.action == "insert" ? param.value : param.added[0];
int intMax = db.ScheduleDatas.ToList().Count > 0 ?
db.ScheduleDatas.ToList().Max(p => p.Id) : 1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleData appoint = new ScheduleData()
{
Id = value.Id,
StartTime = startTime,
EndTime = endTime,
StartTimeZone=value.StartTimeZone,
EndTimeZone=value.EndTimeZone,
Subject = value.Subject,
Description = value.Description,
Recurrence = value.Recurrence,
CategoryId = value.CategoryId,
AllDay = value.AllDay,
RecurrenceRule = value.RecurrenceRule,
```

```
};
db.ScheduleDatas.InsertOnSubmit(appoint);
db.SubmitChanges();
Microsoft.Office.Interop.Outlook.Application outlookApp = new
Microsoft.Office.Interop.Outlook.Application();
Microsoft.Office.Interop.Outlook.AppointmentItem newAppointment = null;
newAppointment =
(Microsoft.Office.Interop.Outlook.AppointmentItem)outlookApp.CreateItem(M
icrosoft.Office.Interop.Outlook.OlItemType.olAppointmentItem);
newAppointment.Start =
Convert.ToDateTime(value.StartTime).ToUniversalTime();
newAppointment.End = Convert.ToDateTime(value.EndTime).ToUniversalTime();
newAppointment.Subject = value.Subject.ToString();
newAppointment.Save();
}
IEnumerable data = new DataClasses1DataContext().ScheduleDatas.Take(500);
// new.Appointment.Take(5);
return Json(data, JsonRequestBehavior.AllowGet);
}
}
}
```

Note: In order to achieve the above scenario, need to refer the Microsoft assembly (Microsoft Outlook 12/15 Object library [Microsoft.Office.Interop.Outlook]) in your application and refer it in the controller page as shown above.

Scroller

Overview

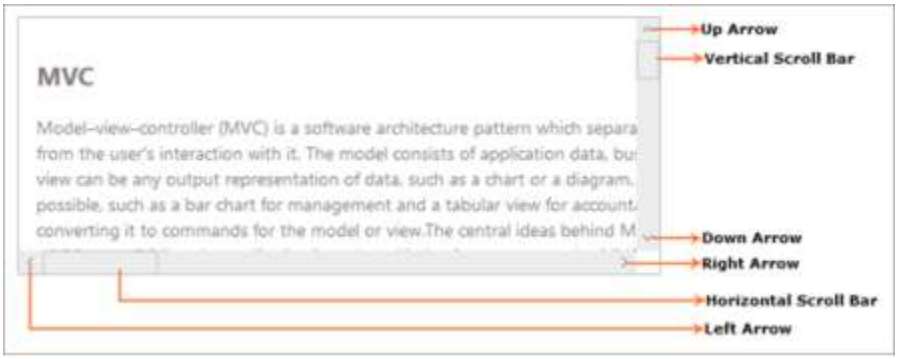
The Essential ASP.NET MVC Scroller control has a sliding document whose position corresponds to a value. The document has text, HTML content or images. You can also customize the Scroller control by resizing the scrolling bar and changing the theme.

Key Features

- Height and Width—Set the height and width of the scroll panel.
- Button Size—Customize the width and height of the buttons (UP, DOWN, RIGHT and LEFT).
- Scroller Size—Customize the scrollbar width and height.
- Step Increment—Set the number of pixels to be moved on pressing the ARROW key.
- RTL Support—Sets the alignment of the horizontal Scroller to the right, the reading order to right-to-left and the layout of the control to flow from right to left.
- Theme—Essential ASP.NET MVC controls consists of 12 built in themes (6 – flat and 6 – gradient effects), and also supports custom skins to set user-defined themes.

Copyright © 2001 - 2020 Syncfusion Inc.

Control Structure



Getting Started

This section explains briefly about how to create a Scroller in your ASP.NET MVC application.

Create your first Scroller in MVC

ASP.NET MVCScroller control allows you to slide document whose position corresponds to a value. The document has text, HTML content or images. Refer the following guidelines to create a Scroller control for horizontal and vertical Scrolling.

Add Scroller Control to your MVC Application

Essential Studio ASP.NET MVC Scroller control has a built-in feature to customize the resizing and changing the Scrollbar theme.

1. Create an MVC Project and add required assemblies, scripts, and styles to it. Refer MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page to render the Scroller.

ASPX-CS

```
<div id="scrollContent">
<div> @* Wrapper div for Scroller.*@
<div class="inner-content"> @* Content div*@
<h3> MVC </h3>
<p>
Model–view–controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it. The model consists of application data, business rules, logic,
and
functions. A view can be any output representation of data, such as a
chart
or a diagram.
</p>
</div>
</div>
</div>
@{ Html.EJ().Scroller("scrollContent").Height(150).Width(300).Render(); }
```

3. Add the following style in the view page to set the height and width of the Scroller.

CSS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<style type="text/css" class="cssStyles">
.inner-content
{
width: 400px;
padding:15px;
}
</style>
```

4. Execute the above example code to get the following output.



Scroller Styles

The Essential ASP.NET MVCScroller control allows you to customize the look and function of scrollbars. You can vary it significantly by setting the scrollbar button size, scrollbar position, height and width of the Scroller control. This section describes you the custom styles to be used when creating Scroller.

Button Size

In Scroller control, it allows you to customize the scroll arrows width and height. In horizontal scroller the ButtonSize customizes the top and down arrow and in vertical scroller the ButtonSize customizes the left and right arrow.

Scroller Size

The ScrollerSize property is used to customize the scrollbar width and height. It is applicable for both horizontal and vertical scroller.

Scroll Top

The ScrollerTop property is used to move the Scroller content and scrollbars in top position with the specified value. It is used for only vertical scroller.

Scroll Left

The ScrollerLeft property is used to move the Scroller content and scrollbars in left position with the specified value. It is used for only horizontal scroller.

Height

The Height property is used to set the height for Scroller outer wrapper.

Width

The width property is used to set the Width for Scroller outer wrapper.

In the View page, add the Scroller helper to configure Scroller widget.

ASPX-CS

```
// In the CSHTML page, add a <div> element to configure Scroller widget
and initialize the control.
```

Copyright © 2001 - 2020 Syncfusion Inc.

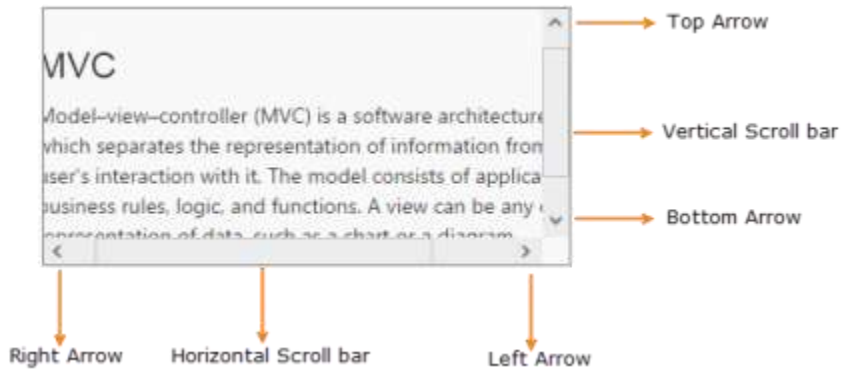
```
<div id="scrollContent">
<div> @*Wrapper div for Scroller.*@
<div class="inner-content"> @*Content div*@
<h3>MVC </h3>
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it. The model consists of application data, business rules, logic,
and
functions. A view can be any output representation of data, such as a
chart
or a diagram.
</p>
</div>
</div>
</div>
@{Html.EJ().Scroller("scrollContent").Height(170).Width(350).ScrollTop(10
).ScrollLeft(20).ButtonSize(20).Render();}
```

1. Configure the styles

CSS

```
<style type="text/css">
.inner-content
{
width: 400px;
padding: 15px;
}
#scrollContent {
border: 1px solid grey;
}
</style>
```

The following screenshot displays the Scroller control with basic styles



Scroller control rendered with basic styles

Copyright © 2001 - 2020 Syncfusion Inc.

Thumb Scrolling

Normally the scrollbar position can be changed by dragging the scrollbar handle or clicking the arrows. The Scroller control allows you for panning or dragging the scroll content area to drag by dragging inside the scroll content. To achieve this in your Scroller control, enable the EnableTouchScroll to true. By default the value forEnableTouchScroll is true. When you want to prevent the panning or dragging the scroll content area, set EnableTouchScroll as false.

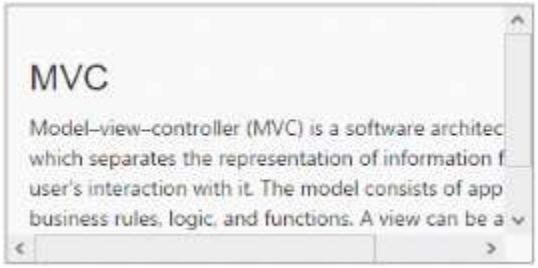
The following steps explains you the configuration of EnableTouchScroll property in Scroller.

1. In the View page, add a scroller helper to configure the EnableTouchScroll property.

ASPX-CS

```
// In the CSHTML page, add a <div> element to configure Scroller widget
and initialize the control.
<div id="scrollContent">
<div> @*Wrapper div for Scroller.*@
<div class="inner-content">  @*Content div*@
<h3>MVC </h3>
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it. The model consists of application data, business rules, logic,
and
functions. A view can be any output representation of data, such as a
chart
or a diagram.
</p>
</div>
</div>
</div>
@{Html.EJ().Scroller("scrollContent").Height(170).Width(350).EnableTouchS
croll(false).Render();}
```

The following screenshot displays Scroller control with disabled touch support.



Scroller control with disabled touch support

Customizing the scroll Step

The Scroller control allows you to specify the scroll movement step in a pixel value, this step value is added to the scrollbar position when you press the arrow key. Based on position value, the scrollbar moves in its corresponding orientation. You can achieve this by using ScrollOneStepBy.

The following steps explains you the configuration of ScrollOneStepBy property in Scroller.

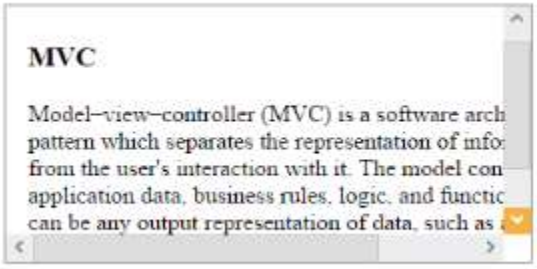
Copyright © 2001 - 2020 Syncfusion Inc.

1. In the View page, add a scroller helper to configure Scroller widget.

ASPX-CS

```
// In the CSHTML page, add a <div> element to configure Scroller widget
and initialize the control.
<div id="scrollContent">
<div>
@*Wrapper div for Scroller.*@
<div class="inner-content">
@*Content div*@
<h3>MVC </h3>
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it. The model consists of application data, business rules, logic,
and
functions. A view can be any output representation of data, such as a
chart or a diagram.
</p>
</div>
</div>
</div>
@{Html.EJ().Scroller("scrollContent").Height(170).Width(350).ScrollOneSte
pBy(50).Render();}
```

The following screenshot displays the Scroller control with scroll step value.



Scroller Control with scroll step value

RTL

EnableRTL API provides right-to-left functionality and features for languages that work in a right-to-left for selecting, editing. Arabic and Hebrew are written from right to left. If you have a working style from right to left, you can use this feature in scroller. You can achieve this in your Scroller by using EnableRTL property. Setting this property to true, the Scroller content text is displayed in the right to left format. The vertical scrollbar move to right to left side.

The following steps explains you the configuration of EnableRTL property in Scroller.

In the View page, add a scroller helper to configure Scroller widget.

ASPX-CS

```
// In the CSHTML page, add a <div> element to configure Scroller widget
and initialize the control.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div id="scrollcontent">
<div>  @*Wrapper div for Scroller.*@
<div class="inner-content"> @*Content div*@
<h3>MVC </h3>
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it. The model consists of application data, business rules, logic,
and
functions. A view can be any output representation of data, such as a
chart
or a diagram.
</p>
</div>
</div>
</div>
@{Html.EJ().Scroller("scrollContent").Height(170).Width(350).EnableRTL(tr
ue).Render();}
```

The following screenshot displays the Scroller control in RTL direction.



Scroller control in RTL direction

How to?

Render Scroller from Code behind

Scroller can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of Scroller properties in the controller.

C#

```
<div>
@* Wrapper div for Scroller.*@
<div class="inner-content" style="width:400px;padding:15px">
@* Content div*@
<h3> MVC </h3>
<p>
Model-view-controller (MVC) is a software architecture pattern which
separates the representation of information from the user's interaction
with it. The model consists of application data, business rules, logic,
and
functions. A view can be any output representation of data, such as a
chart
or a diagram.
</p>
</div>
```

```
</div>
</div>
//Namespace to get the JavaScript (Scroller) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class ScrollerController : Controller
{
// GET: Scroller
public ActionResult ScrollerFeatures()
{
//Initializing the Scroller model
ScrollerProperties ScrollerObj = new ScrollerProperties();
//Initializing the datasource and other properties
ScrollerObj.Height = 150;
ScrollerObj.Width = 300;
//Passing Scroller properties using the ViewData
ViewData["ScrollerModel"] = ScrollerObj;
return View();
}
}
}
```

Binding the Scroller properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Scroller("scrollContent",(Syncfusion.JavaScript.Models.Scroller
Properties)ViewData["ScrollerModel"]).Render();
}
```

Signature

Overview

The Essential MVC Signature Plugin is used to capture or drawing the smooth signatures. It captures user's signature as vector outlines of strokes. Using Signature we can customize the background, stroke width and stroke color and also convert captured signature to an image format. It is also provided with Undo, Redo & Clear options.

Key Features

- **Height and Width** - Set the height and width of the signature canvas.

*** Stroke Color and Background Color** - Customize the stroke color and background color for the canvas

- **Background Image** - Sets the background image for the signature canvas.
- **Stroke Width** - Sets the stroke's minimum and maximum width for the drawing stroke.

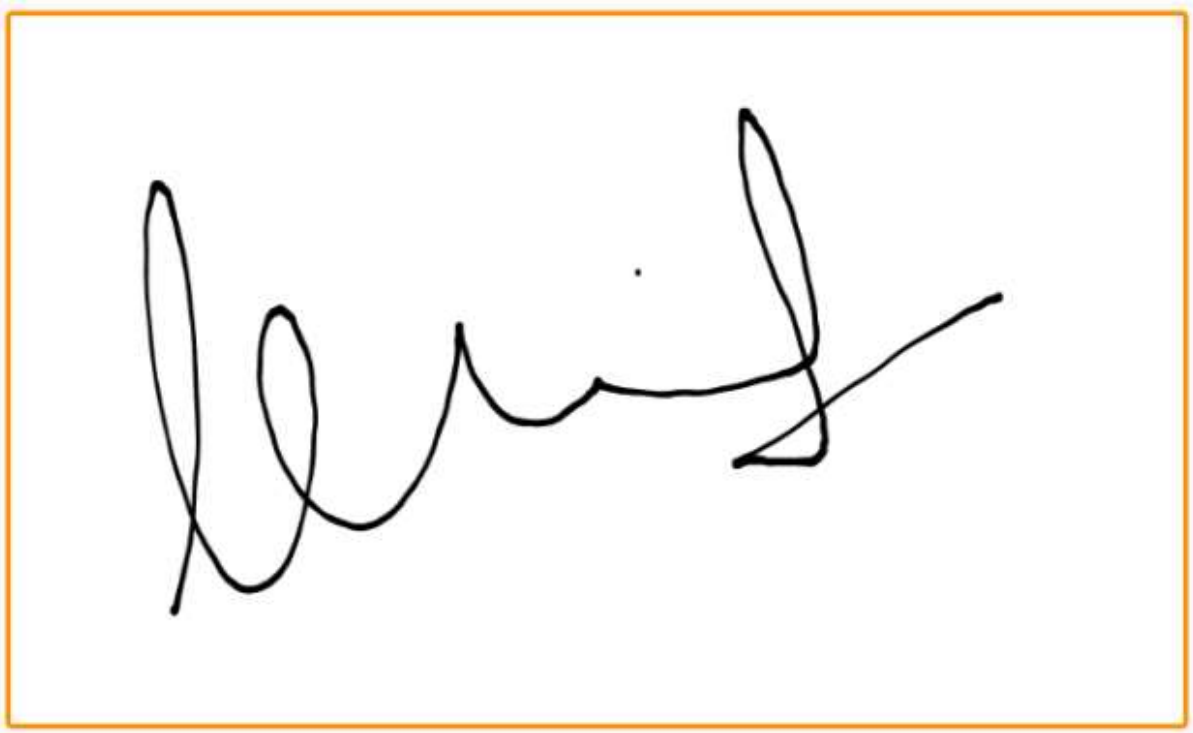
***Undo and Redo** – Undo and Redo the strokes in the signature, if wrongly drawn

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Started

The Signature simplifies creation of a signature capture in a browser, allowing a user to draw a signature using mouse or touch.

This section explains briefly about how to create a **Signature** control in your application with ASP.NET MVC application by the step-by-step instructions. The following screenshot demonstrates the functionality of Signature widget.



Creating your first Signature in MVC application

Create an MVC Project and add necessary assemblies, scripts and CSS files given in MVC-Getting Started Documentation.

Add Signature control using the helper from EJ namespace.

RAZOR

```
@Html.EJ().Signature("mySign").Height("400px")
```

Execute the code to render a Signature widget as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Adjusting Signature Size

We can customize the width and height of the Signature by width and height properties. These properties completely depend on signature canvas. The width and height are adjusted within the signature canvas.

The following code example is used to render the Signature control with customized width and height.

RAZOR

```
@Html.EJ().Signature("mySign").Height("500px").Width("200px")
```

The following screenshot illustrates signature with customized width and height.

Copyright © 2001 - 2020 Syncfusion Inc.



Signature Customization

Stroke color

Signature widget allows you to set the stroke color for the signature stroke using the **StrokeColor** property. When we set our required color, then the signature's stroke color will be changed.

The following example is used to render the Signature control with stroke color.

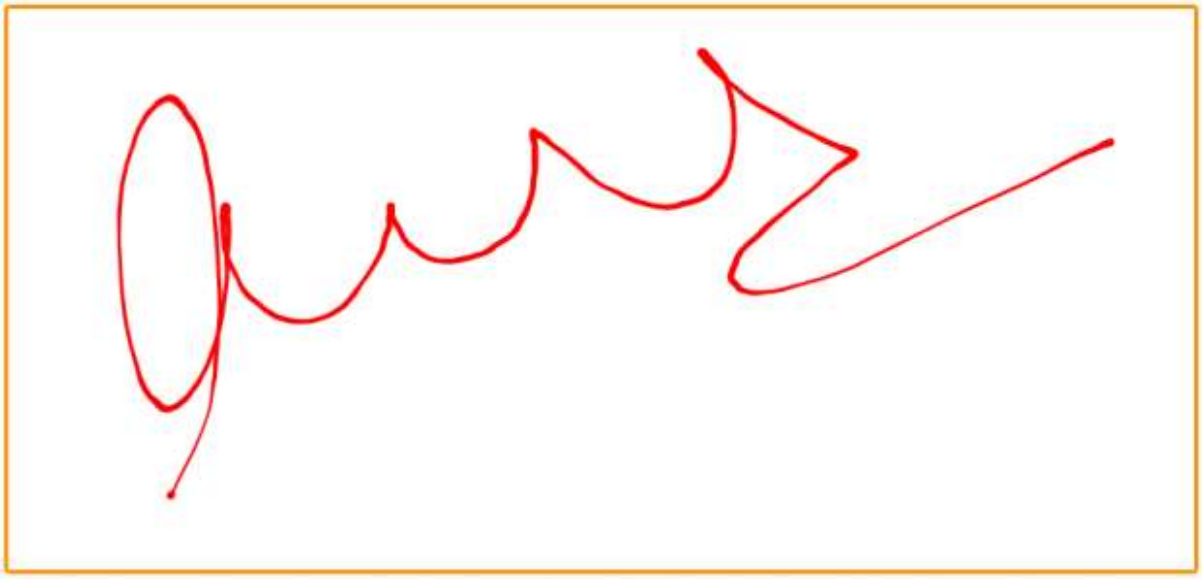
In the View page, define the following code.

RAZOR

```
@Html.EJ().Signature("mySignature").StrokeColor("red")
```

The following screenshot illustrates the Signature with custom defined stroke color.

Copyright © 2001 - 2020 Syncfusion Inc.



Stroke Width

Signature widget allows you to set the stroke width for the control using the **StrokeWidth** property. When we set our required width, then the signature's stroke width will be changed.

The following code example is used to render the Signature control with stroke width value.

In the View page, define the following code.

RAZOR

```
@Html.EJ().Signature("mySignature").StrokeWidth(5)
```

The following screenshot illustrates the Signature with custom defined stroke maximum value.



Background color

Signature widget allows you to set the background Color for the control using the **BackgroundColor** property. When we set our required background, then the signature's background color will be changed automatically.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example is used to render the Signature control with background color.

In the View page, define the following code.

RAZOR

```
@Html.EJ().Signature("mySignature").BackgroundColor("grey")
```

The following screenshot illustrates the Signature with custom defined background color



Background Image

Signature widget allows you to set the background image for the control using the **BackgroundImage** property. When we set our required background image, then the signature's canvas will be changed with the given image.

The following code example is used to render the Signature control with customized background image.

In the View page, define the following code.

RAZOR

```
@Html.EJ().Signature("mySignature").BackgroundImage("image.jpg")
```

The following screenshot illustrates the Signature with custom defined background image.

Copyright © 2001 - 2020 Syncfusion Inc.



How To

Save signature image with user defined format

By default, the downloaded image form the signature canvas will be in **png** format. We can define our own format to download the image with **SaveImageFormat** property. And we can also save the image along with the background by using the **SaveWithBackground** property.

In the View page, define the following code.

RAZOR

```
@Html.EJ().Signature("mySignature").Height("400px").StrokeWidth(3).Backgr
oundImage("../../Images/progressbar/water.png").IsResponsive(true).saveWi
thBackground(true)
@Html.EJ().Button("signSave").Text("Save").ShowRoundedCorner(true).Client
SideEvents(event => event.Click("onSave"))
```

Add the following script to define the download format for the canvas.

JS

```
<script type="text/javascript">
function onSave() {
var sign = $("#mySignature").ejSignature("instance");
sign.option("saveImageFormat", "jpg")
sign.save("mySignature");
}
</script>
```

The following screenshot illustrates the Signature with saving (downloading) the drawn image along with its background.

Copyright © 2001 - 2020 Syncfusion Inc.



To clear the Signature

To clear the signature, you can simply use the **clear()** method. This method will clear all the drawn strokes in the signature canvas and leaves it empty.

RAZOR

```
@Html.EJ().Signature("mySignature").Height("400px").StrokeWidth(3).IsResp
onsive(true)
@Html.EJ().Button("signClear").Text("Clear").ShowRoundedCorner(true).Clie
ntSideEvents(clientSideEvent => clientSideEvent.Click("onClear"))
```

JS

```
<script type="text/javascript">
function onClear() {
var sign = $("#mySignature").ejSignature("instance");
sign.clear();
}
</script>
```

Make signature as responsive

When the signature control is resized or even the window is resized the strokes drawn in the signature will be disappeared. To make the strokes visible even after resizing the window, we must set the **IsResponsive** property as true.

In the View page, define the following code to render signature with responsive.

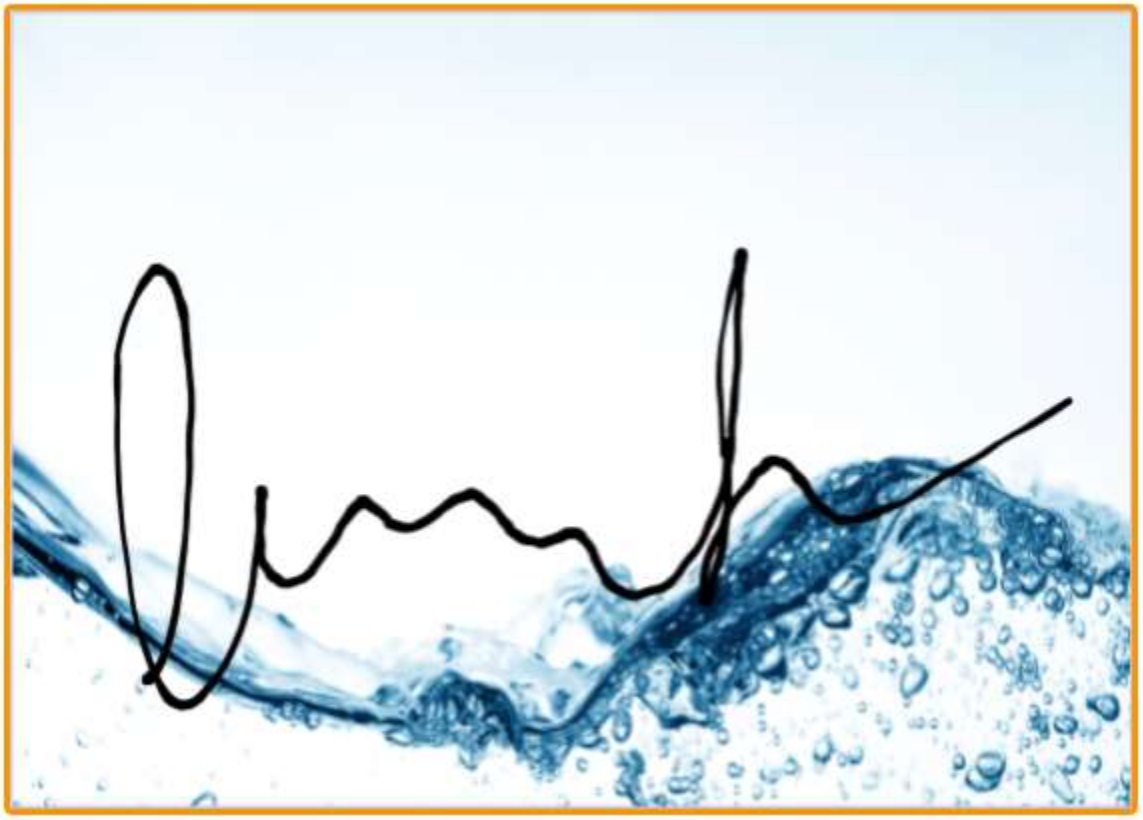
RAZOR

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Signature("mySign").IsResponsive(true)
```

The following screenshot illustrates the Signature with responsiveness.

Before Responsiveness:



After giving the Responsiveness:



To check whether any input to the signature control since render

We can detect whether not there has been any input to the signature control since render. To detect we can use the storeSnap public variable, which is an array that stores all the canvas inputs. At initial rendering this array is empty and we can use this variable to check for the drawn strokes.

JS

```
<script type="text/javascript">
var sign = $("#signature").ejSignature("instance");
if (ej.isNullOrUndefined(sign.storeSnap)) {
//Something
}
</script>
```

Render Signature from Code behind

Signature can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side

The following code illustrates the initialization of Signature properties in the controller.

C#

```
//Namespace to get the JavaScript (Signature) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class SignatureController: Controller
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult SignatureFeatures()
{
//Initializing the Signature model
SignatureProperties signObj = new SignatureProperties();
//Initializing the other properties
signObj.Height = "200px";
signObj.IsResponsive = true;
signObj.StrokeWidth = 5;
//Passing Signature properties using the ViewData
ViewData["SignModel"] = signObj;
return View();
}
}
}
```

Binding the Signature properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Signature("sign1",
(Syncfusion.JavaScript.Models.SignatureProperties)ViewData["SignModel"]).
Render();
}
```

Slider

Slider

The Essential ASP.NET MVC Slider provides support to select a value from a particular range as well as selects a range value. The Slider has a sliding base on which the handles are moved. There are three types of Sliders such as default Slider, min-range Slider and range Slider.

Key Features

- Orientation—Slider control is displayed in horizontal or vertical direction.
- Animation—Supports for handle movement with animation.
- Tooltip—Supports to display a tooltip to display the currently selected value.
- Range Slider—Supports to select a range of values.
- Scale—Supports to display a scale with small and big ticks.
- Persist—Supports for state maintenance while refreshing a page.
- Customizable—Easily customizable.
- RTL Support—Sets the alignment to the right, the reading order to right-to-left and the layout of the control to flow from right to left.
- Themes—Essential JavaScript controls consist of 12 built-in themes (6 – flat and 6 – gradient effects), and also support custom skins to set user-defined themes.

Getting Started

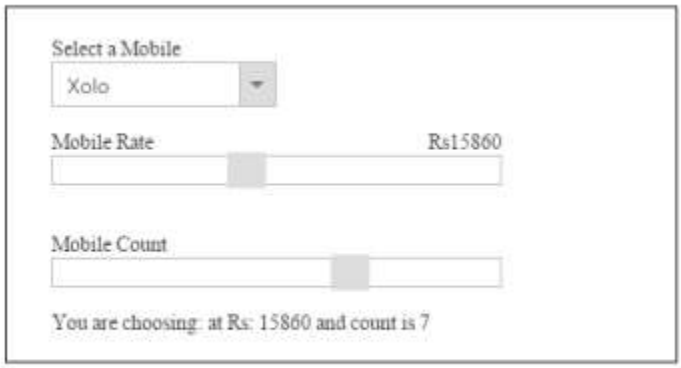
This section explains briefly about how to create a Slider in your application with ASP.NET MVC.

Copyright © 2001 - 2020 Syncfusion Inc.

Create your first Slider in MVC

ASP.NET MVC Slider provides support to display a Slider within the web page. The following section explains you on how to use Sliders in a real-time application to select a mobile model within specified range in Mobile Purchase.

The following screenshot illustrates the functionality of Slider control. You can select a mobile model in the dropdown to purchase it at any rate specified in the Mobile RateSlider and you can also specify the number of mobiles you need by selecting the Mobile CountSlider. Simultaneously, you can observe the change in mobile rate and count using Sliders.



Create a Slider

ASP.NET MVC Slider control allows you to switch between different ranges of input. The basic Slider is horizontal and has a single handle that can be moved with the mouse or by using the arrow keys. You can easily create the Slider control using @Html.helper as follows.

1. Create an MVC Project and add necessary assembly, scripts, and styles to it.

Refer MVC-Getting Started

2. Add the following code example for Slider to the corresponding view page.

ASPX-CS

```
<div class="frame">
<div class="inner">
<div class="control-label">
Select a Mobile
</div>
@Html.EJ().DropDownList("selectMobile").TargetID("mobileList").Width("150
px")
<div id="mobileList">
<ul>
<li>Nokia Lumia</li>
<li>Nokia Asha</li>
<li>Xolo</li>
<li>Samsung Galaxy</li>
<li>Moto G</li>
</ul>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
<span class="column-left">
<span>Mobile Rate</span>
</span>
<span class="column-right">
<span>Rs </span><span class="value"></span>
</span>
@Html.EJ().Slider("rateSlider").Height("20px").Value("100").MinValue(5000
).MaxValue(30000).IncrementStep(20).ClientSideEvents(event =>
event.Change("onChange").Slide("onChange"))
<span class="Columnleft">
<span>Mobile Count</span>
</span>
@Html.EJ().Slider("countSlider").Height("20px").Value("1").MinValue(1).Ma
xValue(10).IncrementStep(1).ClientSideEvents(event =>
event.Change("onChange").Slide("onChange"))
You are choosing:
<span id="EventLog"></span>
</div>
</div>
```

3. Add the following styles for Sliders.

CSS

```
<style>
.frame
{
width:400px;
height:200px;
border:1px solid black;
margin-right:100px;
}
.control-label, .loan
{
font-weight:bold;
}
.column-right
{
font-weight:bold;
float:right;
}
.inner
{
width:300px;
height:150px;
padding:20px 40px 20px 30px;
}
.mobileList
{
display:block;
}
</style>
```

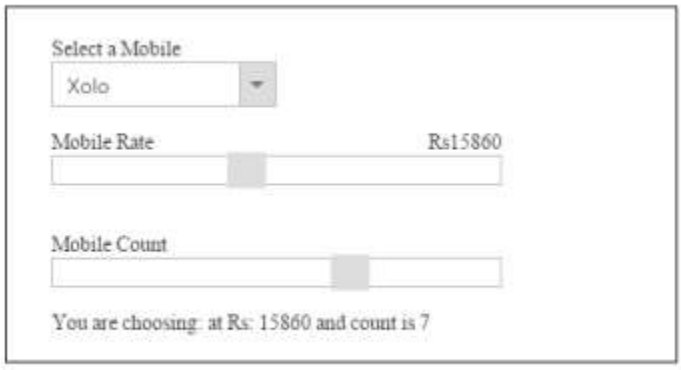
Copyright © 2001 - 2020 Syncfusion Inc.

4. Add the following script to the Slider.

JS

```
<script>
var mobileObj, rateObj, countObj;
function onChange(args) {
$('#' + args.id).parent().prev().find('.value').html(args.value)
show();
}
function show() {
selectObj = $('#selectMobile').data('ejDropDownList');
rateObj = $('#rateSlider').data('ejSlider');
countObj = $('#countSlider').data('ejSlider');
var x = selectObj.getValue();
var y = rateObj.getValue();
var z = countObj.getValue();
$('#EventLog').html("\n" + x + "\n" + "at Rs: " + y + "\n" + "and count
is " + z);
}
</script>
```

5. The following screenshot displays the final output in Slider creation.



EMI Calculator

This section explains how to use the Slider control for EMI Calculation. The final result of EMI amount is calculated automatically based on the loan amount, interest rate and tenure amount based on the amount you choose using Slider.

1. You can create an MVC Project and add necessary and script to it. Refer MVC-Getting Started Documentation.
2. Add the following code for Slider to the corresponding view page.

ASPX-CS

```
<div class="frame">
<div class="inner">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div id="loanHeading">
Details of Loan
</div>
<span class="column-left">
<span>Loan Amount</span>
</span>
<span class="column-right">
<span>Rs </span><span class="value">25000</span>
</span>
@Html.EJ().Slider("loanSlider").Height("16px").Value("25000").MinValue(10
000).MaxValue(1000000).IncrementStep(10).ClientSideEvents(event =>
event.Change("onChange").Slide("onChange"))
<span class="column-left">
<span>Interest Rate</span>
</span>
<span class="column-right">
<span class="value"></span><span>% pa</span>
</span>
@Html.EJ().Slider("interestSlider").Height("16px").Value("4").MinValue(1)
.MaxValue(20).IncrementStep(1).ClientSideEvents(event =>
event.Change("onChange").Slide("onChange"))
<span class="column-left">
<span>Tenure</span>
</span>
<span class="column-right">
<span class="value"></span><span>Years</span>
</span>
@Html.EJ().Slider("tenureSlider").Height("16px").Value("3").MinValue(1).M
axValue(20).IncrementStep(1).ClientSideEvents(event =>
event.Change("onChange").Slide("onChange"))
Your Monthly EMI Amount is
<span id="EventLog"></span>
</div>
</div>
</div>
```

3. Include the following styles for Slider.

CSS

```
<style>
.frame
{
width:350px;
height:200px;
border:1px solid black;
}
.inner
{
width:250px;
height:150px;
padding:20px 30px;
}
#loanHeading
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
font-weight:bold;
}
.column-left
{
font-weight:bold;
}
.column-right
{
float:right;
font-weight:normal;
}
</style>
```

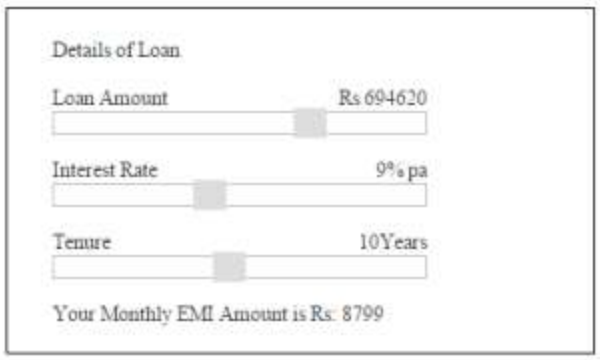
4. Include the following script to calculate the EMI amount.

ASPX-CS

```
<script type="text/javascript">
var loanObj, interestObj, tenureObj;
function onChange(args)
{
$('#' + args.id).parent().prev().find('.value').html(args.value)
calculate();
}
function calculate()
{
loanObj = $('#loanSlider').data('ejSlider');
interestObj = $('#interestSlider').data('ejSlider');
tenureObj = $('#tenureSlider').data('ejSlider');
var loan = loanObj.getValue(), interest = interestObj.getValue(), tenure
= tenureObj.getValue();
var P = loan;
var y = interest / 1200;
var tenureAmount = tenure * 12;
//actual processing
var top = y * (Math.pow((1 + y), tenureAmount));
var bottom = (Math.pow((1 + y), tenureAmount)) - 1;
var ans = top / bottom;
var final = P * ans;
var z = Math.round(final);
$('#EventLog').html("Rs: " + z);
}
</script>
```

The following screenshot displays the calculated EMI amount using Slider.

Copyright © 2001 - 2020 Syncfusion Inc.



The Slider component lets you to select a value such as number, percentage by moving the slider handle. It also allows you to specify a range of values between its minimum and maximum values. You can use Slider to influence other object. For example, Slider can be associated with a picture so that the picture enlarges or shrinks based upon the value in the slider.

The Slider has horizontal orientation by default. You can also change the orientation as vertical. Also, the Slider contains features like enabling slider of different types, enabling scale in the slider etc.

Behavior Settings

Height

By default, Slider renders with a height of 14px. You can change the Slider height using Height property. Specify the value for this property in string format.

Width

By default, Slider widget renders with 100% width. You can customize the width of the Slider using Width property. Specify the value for this property in string format.

The following steps explains you on how to configure the Height and Width of the Slider.

1. In an VIEW page, add a helper element to render it as a Slider widget.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("BasicSlider").Height("20").Width("500"))
```

Execute the above code example to render the following output.

IncrementStep

This property sets the incremental step value for the Slider. When the Slider handle slides through mouse or keyboard, it increments / decrements the value based on the step value. By default, when the

slider handle is moved, single value increments / decrements. Using IncrementStep property you can change the increment step value. Data type of this property is number.

The following steps explains you on how to configure the IncrementStep property.

1. In an VIEW page, add a helper element to render it as a Slider widget.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("BasicSlider").Height("20").Width("500").IncrementStep
(5))
```

Execute the above code example to render the following output.



In the above example, value for IncrementStep property is specified as “5” therefore, when you move the Slider handle, value “5” increments/decrements from the current Slider value.

ReadOnly

This feature prevents you from interacting with the Slider. That is you can only view the Slider value and cannot change it.

The following steps explains you on how to enable the ReadOnly property.

1. In an VIEW page, add a helper element to render it as a Slider widget.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("BasicSlider").Height("20").Width("500").ReadOnly(true
))
```

After you execute the above code example, the Slider values cannot be changed in any ways.



Persistence Support

This feature supports you to save current model value to browser cookies for state maintenance. When you refresh the web page, the Slider control retains the model value apply from browser cookies. The data type of EnablePersistence is Boolean type.

Copyright © 2001 - 2020 Syncfusion Inc.

The following steps explains you on how to enable the EnablePersistence property.

1. In an VIEW page, add a helper element to render it as a Slider widget.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("BasicSlider").Height("20").Width("500").EnablePersist
ence(true))
```

After execute the code, Increase the value and reload the page. The state is maintained in the Slider control.



Orientation

This property is used to set the Slider in either horizontal or vertical direction. By default, Slider renders in horizontal direction. Data type of this property is “Enum”.

Possible Slider orientations are as follows,

Property Table for MVC

Property	Allowed values	Description
Orientation	Vertical	Displays the Slider in vertical direction
	Horizontal (default value)	Displays the Slider in horizontal direction

The following steps explains you on how to configure the Orientation property.

1. In an VIEW page, add a helper element to render it as a Slider widget.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("BasicSlider").Height("150").Width("20").Orientation(O
rientation.Vertical))
```

Execute the above code example to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Slider in vertical Orientation

Slider Types

This feature allows you to specify the type of Slider. There are three different types of Slider, Default Slider, Min-Range Slider and Range Slider. By default, Default Slider renders. You can use the SliderType property to choose the type of Slider. Data type of this property is “Enum”

Both Default Slider and Min-Range Slider have same behavior that is used to select a single value. In Min-Range Slider, a shadow is considered from the start value to current handle position. But Range Slider contains two handles that is used to select a range of values and a shadow is considered in between the two handles.

Possible Slider types are as follows,

Property Table for MVC

Property	Allowed values	Description
SliderType	Default (default value)	It is the default Slider type. It helps to select a single value.
	MinRange	Use this Slider to select a single value; Displays shadow from the start value to the current value.
	Range	Use this Slider to select a range of values; Displays shadow in-between the selection range.

The following steps explains you on how to configure the SliderType property to display Range Slider and MinRange Slider.

1. In an VIEW page, specify the helper elements to render the RangeSlider and MinRangeSlider.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("minSlider").SliderType(SlideType.MinRange).Value("60"
).Width("500"))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("30,
60").Width("500"))
```

Execute the above code example to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Shows different Slider types (Range and Min-Range sliders)

Updating slider value

Slider widget includes an option to specify/update its value. And also you can specify the starting and ending value for the Slider using the MinValue and MaxValue properties.

Value

This property is used to set the value in the “Default” and “Min-Range” Sliders. By default its value is null when no value is specified. Data type of this property is “Number”. You can get/set the value in the slider handle by using getValue and setValue methods.

Also change event will be triggered whenever **Slider** value is changed.

Values

This property is used to set the value in “Range Slider”. By default range values is from 0 to 100. This property is of “Array” data type.

The following steps explains you the configuration of “Value” and “Values” property.

1. In an VIEW page, specify the helper elements to render the “Range Slider” and “Default Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").SliderType(SlideType.Default).Value("
50").Width("500"))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("20,
80").Width("500"))
```

Execute the above code example to render the following output.



MinValue

To set the minimum/starting value of the Slider, you can use the MinValue property. By default its value is 0. Data type of this property is “number”.

Copyright © 2001 - 2020 Syncfusion Inc.

MaxValue

To set the maximum/ending value of the Slider, you can use the MaxValue property. By default its value is 100. Data type of this property is “Number”.

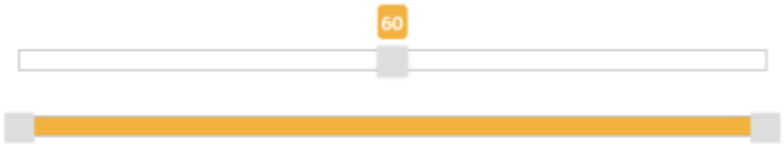
The following steps explains you on how to configure MinValue and MaxValue property.

1. In an VIEW page, specify the helper elements to render the Default Slider and Range Slider.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").MinValue(40)
.MaxValue(80))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("10,
90")
.MinValue(10).MaxValue(90).Width("500"))
```

Execute the above code example to render the following output.



In the above example, for Default Slider the slider value starts from “40” (min value) and ends in “80” (max value), so the slider handle is placed at the center of the Slider while specifying the value as “60”.

For Range Slider, the value starts from “10” (min value) and ends in “90” (max value). The range shadow occupies the entire Slider, since the range (values) is specified as “[10, 90]”.

Buttons

Slider includes the button support for increment or decrement the values of the slider.

Enabling Buttons

Use the ShowButtons property to enable the button support. By default this property is disabled. Data type of this property is “Boolean”.

The following steps explains you on how to enable button support in Slider.

1. In an VIEW page, specify the helper elements to render the Range Slider.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("30,
60")
.Width("500").ShowButtons(true))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code example to render the following output.



Scale Settings

Slider widget includes option to display the scale on the Slider. Scale in Slider supports you to easily identify the current value/values of the Slider. Scale contains “small ticks” and “large ticks”. Values of the Slider is displayed above each large ticks.

Show Scale

This property enables the scale in the Slider. By default, its value is “false”. Data type of this property is “Boolean”.

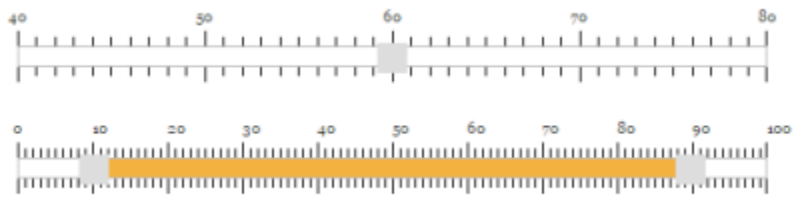
The following steps explains you the configuration of ShowScale property.

1. In an VIEW page, specify the helper elements to render the “Default Slider” and “Range Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").MinValue(40)
.MaxValue(80).ShowScale(true))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("10,
90")
.ShowScale(true))
```

Execute the above code example to render the following output.



Enable Small Ticks

Slider widget provides you an option to enable/disable the small ticks present in the scale. By default, when you enable “ShowScale” property, small ticks is displayed in the scale. Use the ShowSmallTicks property to disable the small ticks present in the scale. Data type of this property is “Boolean”. The renderingTicks event will be triggered while creating each slider scale tick.

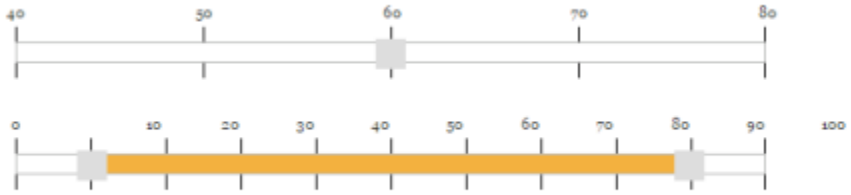
The following steps explains you on how to disable the small ticks in Slider.

1. In an VIEW page, specify the helper elements to render the “Default Slider” and “Range Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").MinValue(40)
.MaxValue(80)
.ShowScale(true).ShowSmallTicks(false))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("10,
90")
.ShowScale(true).ShowSmallTicks(false))
```

Execute the above code example to render the following output.



Small step

This property specifies the distance between the two small ticks present in the scale and the position of the small ticks in the Slider scale. Data type of this property is “Number”.

Large step

This property specifies the distance between the two small ticks present in the scale and the position of the large ticks in the Slider scale. Data type of this property is “Number”.

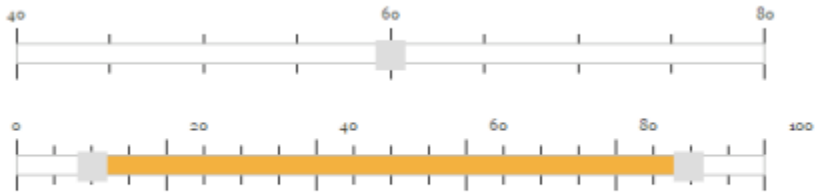
The following steps explains you on how to configure the SmallStep and LargeStep property in Slider scale.

1. In an VIEW page, specify the helper elements to render the “Default Slider” and “Range Slider”

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").MinValue(40)
.MaxValue(80)
.ShowScale(true).SmallStep(5).LargeStep(20))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("10,
90")
.ShowScale(true).SmallStep(5).LargeStep(20))
```

Execute the above code example to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

In the above example, for “Default Slider” the “SmallStep” value is specified as “5”, so for each 5 values from the starting value, small ticks is enabled. Also, “LargeStep” value is specified as “20”, so for each 20 values from the starting value, large ticks is enabled.

Similarly for “Range Slider”, “SmallStep” value is specifies as “5”, so for each 5 values from the starting value, small ticks is enabled. Also, “LargeStep” value is specified as “25” so, for each 25 values large ticks is enabled.

Appearance and Styling

Slider widget looks sleek and enriched with good UI appearance. It is included with both metro (flat) theme and gradient theme support. Totally twelve built-in themes are provided including six flat themes and six gradient themes. The themes include three color variations such as “Azure”, “Lime” and “Saffron”. The themes supported by the Slider widgets are as follows,

- default-theme
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

In order to apply different themes, you can refer the “ej.widgets.all.min.css” file from the corresponding theme folders. This file is a combination of two style sheets “ej.widgets.core.min.css” and “ej.theme.min.css”. Instead of including “ej.widgets.all.min.css” file you can also refer the “ej.widgets.core.min.css” and “ej.theme.min.css” files separately.

The following steps explains you on how to apply “flat-lime-dark” theme to the Slider widget

1. In an VIEW page, specify the desired “ej.widgets.all.min.css” file to load the corresponding theme.

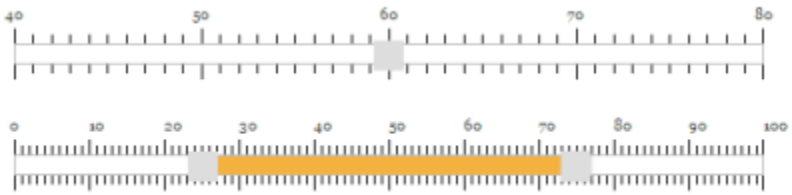
ASPX-CS

```
//In _Layout page, specify the desired
“ej.widgets.all.min.css” file to load the corresponding theme.
<head>
<title>Slider</title>
<!--Flat-saffron theme-->
<link href="http://cdn.syncfusion.com/13.1.0.21/js/web/flat-saffron-
dark/ej.web.all.min.css" rel="stylesheet" />
<!--scripts-->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.web.all.min.js"></scri
pt>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</head></td><td>
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").MinValue(40)
.MaxValue(80)     .ShowScale(true).SmallStep(5).LargeStep(20))
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("10,
90")     .ShowScale(true).SmallStep(5).LargeStep(20))
```

Execute the above code example to render the following output.



CSS Class

When you want to display the Slider widget in a different style based on the appearance of your application, you can use this CssClass property to apply custom theme for the Slider. Specify a class name as the value for CssClass property. The specified class is added to the wrapper of the Slider widget. Now, you can easily override the styles of the Slider widget by accessing the styles from the root level (using the CssClass specified).

The following steps explains you on how to configure the Slider with custom theme using the CssClass property. Here, a class name “purple” is specified for the CssClass.

1. In an VIEW page, specify the helper elements to render the “Default Slider” and “Range Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("25,
75")
.Width("500").CssClass("purple"))
```

2. Include the “CssClass” value before each style of the Slider widget and customize the styles as follows.

CSS

```
.purple.e-slider.e-widget
{
background-color: burlywood;
border-color: #bbbcbb;
}
.purple.e-tooltip
{
background: none repeat scroll 0 0 violet;
/* Old browsers */
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
border-color: #1b95cf;
color: white;
}
.purple.e-slider .e-handle.e-select
{
background-color: purple;
border-color: purple;
}
.purple.e-slider .e-handle.e-hover
{
background-color: purple;
border-color: purple;
}
.purple.e-slider .e-handle.e-focus
{
box-shadow: 0 0 2px rgba(0, 0, 0, 0.2);
}
.purple.e-slider .e-range
{
background: none repeat scroll 0 0 violet;
/* Old browsers */
}
.purple.e-scale .e-tick
{
background-image: url(images/dot.png);
}
```

Execute the above code example to render the following output.



Show Tooltip

Slider displays the tooltip to indicate the current value when you click on the Slider handle. By default, Slider displays the tooltip. Using the ShowTooltip option you can enable or disable the Tooltip. Data type of this property is “boolean”.

The following steps explains you on how to disable the tooltip in Slider.

1. In an VIEW page, specify the helper elements to render the Default Slider.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").ShowTooltip(
false))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Show Rounded Corner

This property is used to display the Slider and its handle with rounded corners. By default ShowRoundedCorner is in disabled state. Data type of this property is “Boolean”.

The following steps explains you on how to disable the tooltip in Slider.

1. In an VIEW page, specify the helper elements to render the “Default Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").ShowRoundedC
orner(true))
```

Execute the above code example to render the following output.

Default Slider

Animation

This feature includes the animation effect for the Slider when moving the Slider handle.

Enabling Animation

By default, animation is enabled in the Slider. Using the EnableAnimation property you can enable/disable the animation effects. Data type of this property is “Boolean”.

The following steps explains you on how to disable the animation effect in Slider.

1. In an VIEW page, specify the helper elements to render the “Default Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").EnableAnimat
ion(false))
```


Customizing Animation speed

Animation speed of the Slider indicates the speed at which the slider handle can be moved. Higher the value specified for this property decreases the rate of speed at which the Slider handle can be moved.

Copyright © 2001 - 2020 Syncfusion Inc.

You can customize the animation speed of the Slider using the AnimationSpeed property. Default value of this property is “500”.

The following steps explains you on how to customize the animation speed.

1. In an VIEW page, specify the helper elements to render the “Default Slider”.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").AnimationSpe
ed(600))
```

Enable/Disable the Slider

Slider widget includes an option to enable/disable it. When you disable the Slider, it is displayed in a blur state and you cannot perform any operations in it.

Enabled

Using this Enabled property you can enable/disable the Slider. Data type of this property is “Boolean”. Also you can enable/disable the **Slider** by using enable and disable methods.

The following steps explains you on how to disable the Slider.

1. In an VIEW page, specify the helper elements to render the Default Slider.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("defaultSlider").Value("60").Width("500").Enabled(fals
e))
```

Execute the above code example to render the following output.



RTL support

Slider includes the Right to Left alignment support. Operations in the Slider is performed from Right to Left.

Enabling RTL

Use the EnableRTL property to enable the RTL support. By default this property is disabled. Data type of this property is “Boolean”.

The following steps explains you on how to enable RTL support in Slider.

1. In an VIEW page, specify the helper elements to render the Range Slider.

ASPX-CS

```
// Add this code in your view page
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("25,
75")
.Width("500").EnableRTL(true))
```

Execute the above code example to render the following output.



Keyboard Interaction

You can use Keyboard shortcut keys as an alternative to the mouse for using the Slider widget. All options in the Slider can be accessed using keyboard shortcuts. The following table explains the keyboard shortcut keys and the operations that can be performed using the corresponding keys.

List of Keyboard shortcuts

Shortcut Key	Description
AltÂ + j	Focuses into the Slider handle
Up/Right	Increments the Slider value
Down/Left	Decrements the Slider value
Home	Slider handle moves to the start value
End	Slider handle moves to the end value
Esc	Focuses out from the Slider handle

Configure keyboard interaction

The following steps explains you on how to enable keyboard support in Slider.

1. In an VIEW page, specify the helper elements to render the Range Slider.

ASPX-CS

```
// Add this code in your view page
@(Html.EJ().Slider("rangeSlider").SliderType(SlideType.Range).Values("25,
75")
.Width("500").EnableRTL(true))
// In JavaScript, Focus the Slider control in document key down function
<script>
$(document).on("keydown", function (e)
{
if (e.altKey && e.keyCode === 74)
{
// j- key code.
$("#rangeSlider a")[0].focus();
```

```
}
});
</script>
```

Run the sample and press, Alt + j keys to set focus in the Slider handle and you can handle the Slider operations using the keyboard shortcut keys.

When clicked the Alt + J key, the slider was focused.



Slider with Keyboard Interaction.

SpellCheck

SpellCheck

The ASP.NET MVC SpellCheck component provides all the functionalities that is available in EJ SpellCheck, since it is a server side wrapper of EJ SpellCheck widget. The ASP.NET MVC SpellCheck control helps to find and highlight misspelled words from a given content, typically by comparing it with a stored list of dictionary words.

Key Features

Some of the key features of SpellCheck are as follows,

- Custom Dictionary - Custom dictionary file is used to store and maintain your custom words like technical terms, brand names etc.
- Ignore Settings - Ignore words composed of alphanumeric characters, mixed case words, uppercase words, URL links, and file path.
- Ignore Words - Ignore a specific collection of words from the error word consideration.
- Customize Menu Items - Adding the user defined menu items in the context menu to correct the error words.
- Misspell Word Style - Highlight the erroneous word based on the user-customized style.
- Maximum Suggestion Count - Display a number of suggestions based on the user-specified count.

Getting Started

This section explains briefly about how to create a SpellCheck Control in your application with ASP.NET MVC

Create your first SpellCheck in MVC

Create a ASP.NET MVC Project and add necessary assemblies, CSS and scripts with the help of the given MVC-Getting Started documentation. After creating this project, you can create a SpellCheck in following ways.

SpellCheck Using Helper element

In the view page, add SpellCheck helper as shown below.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().SpellCheck("TextArea")
```

Service Reference

Assign the service method (CheckWords) path reference to the property dictionaryUrl, which is mandatory to check the spelling of the word.

The CheckWords method will perform the splitting of target sentence into separate words and check each word is that an erroneous or not. If the word is erroneous fetching the possible suggestions for it and returns those details as a result.

ASPX-CS

```
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
```

Spell Checking

To spell check the content of the target element, you need to add one button and calling any one of the spell check method showInDialog or validate by clicking the button to highlight the error words.

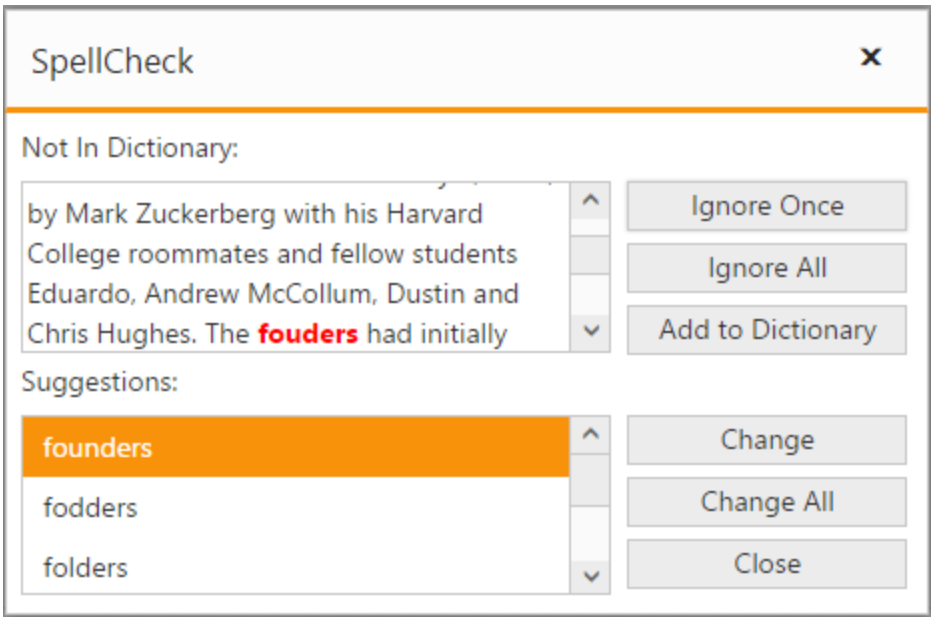
The following code example depicts that checking the spelling of the target element through the showInDialog method.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
Facebook is a social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark
Zuckerberg with his Harvard College roommates and fellow students
Eduardo, Andrew McCollum, Dustin and Chris Hughes.
The founders had initially limited the websites membership to Harvard
students, but later expanded it to collages in the Boston area, the Ivy
League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
</script>
}
```

Once you have completed the above steps, you will get an output like below.

Copyright © 2001 - 2020 Syncfusion Inc.



Dependencies

To render the SpellCheck control, the following list of external dependencies are needed,

- jQuery - 1.7.1 and later versions

The other required internal dependencies are tabulated below,

File	Description/Usage
ej.core.min.js	Must be referred always first before using all the JS controls.
ej.data.min.js	Used to handle data operation and should be used while binding data to JS controls.
ej.spellcheck.min.js	spell check core script file.
ej.unobtrusive.min.js	Used to create our SpellCheck component with basic level HTML tag-like structure.
ej.scroller.js ej.button.js ej.dialog.js ej.menu.js ej.listbox.js ej.draggable.js	These files are referred for proper working of the sub-controls used within SpellCheck.

Copyright © 2001 - 2020 Syncfusion Inc.

ej.globalize.js	

Note: SpellCheck uses one or more sub-controls, therefore refer the ej.web.all.min.js (which encapsulates all the ej controls and frameworks in a single file) in the application instead of referring all the above specified internal dependencies.

To get the real appearance of the SpellChecker, the dependent CSS file ej.web.all.min.css (which includes styles of all the widgets) should also needs to be referred.

Dictionary

Essential ASP.NET MVC SpellCheck uses two different types of dictionary.

- Default Dictionary
- Custom Dictionary

Default Dictionary

The default dictionary is a primary dictionary file location to read the dictionary words for comparing it with the input words to identify its erroneous.

The following code example depicts the way to mention the default dictionary path and to read its content.

C#

```
SpellCheckController()
{
if (baseDictionary == null)
{
// Here you need to specify the corresponding file name (ex. Default.dic)
string filePath =
HttpContext.Current.Server.MapPath("~/App_Data/SpellCheck/Default.dic");
Stream stream = new FileStream(filePath, FileMode.Open, FileAccess.Read);
baseDictionary = new SpellCheckerBase(stream);
}
this.CustomFileRead();
}
```

Custom Dictionary

The custom dictionary is a collection of custom words that are used for spell checking (i.e. technical terms/words, company names, etc.,).

The following code example describes the way to mention the custom dictionary path to read its content.

C#

```
private readonly string _customFilePath =
HttpContext.Current.Server.MapPath("~/App_Data/SpellCheck/Custom.dic");
// Here you need to specify the corresponding custom dictionary file name
private void CustomFileRead()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Stream stream1 = new FileStream(_customFilePath, FileMode.Open,
FileAccess.ReadWrite);
customDictionary = new SpellCheckerBase(stream1);
}
```

Note: The custom words will be displayed within the custom dictionary file line by line (i.e. one word per line).

Add a Word to Custom dictionary

You can directly add the necessary words in the custom dictionary file or else by using the SpellCheck control public method addToDictionary to add the custom words one by one.

Operations

Essential SpellCheck provides two ways to perform the spell check operation(error correction). They are,

- Dialog Mode
- Context Menu Mode

Dialog Mode

Description

SpellCheck provides the dialog mode option to perform the following spell check operations.

- Ignore Once
- IgnoreAll
- Change
- ChangeAll
- Add to Dictionary

Open Dialog Mode

The following code snippet shows how to open the dialog to spell check the content.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Handling Dialog Actions

To define the specific actions before the dialog window open, the client-side event dialogBeforeOpen can be used as depicted in the below code example.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ClientSideEvents(event =>
event.DialogBeforeOpen("onDialogBeforeOpen"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
function onDialogBeforeOpen(args) {
if (args.requestType == "dialogBeforeOpen")
{
alert("dialog before open event triggered");
}
}
</script>
```

The client-side event dialogOpen can be used to define the specific actions after the dialog window open as shown in the following code example.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ClientSideEvents(event =>
event.DialogOpen("onDialogOpen"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
function onDialogOpen(args) {
if (args.requestType == "dialogOpen")
{
alert(args.targetText);
} }
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example used to define some actions after the dialog closing by using the client-side event dialogClose.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ClientSideEvents(event =>
event.DialogClose("onDialogClose"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
function onDialogClose(args) {
if (args.requestType == "dialogClose")
{
alert(args.updatedText);
} }
</script>
```

It is possible to predict the error word details before starting the spell check operations through dialog mode by using the client side event start. The below code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ClientSideEvents(event =>
event.Start("onSpellCheckStart"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
function onSpellCheckStart(args) {
if (args.requestType == "spellCheckDialog")
{
alert(args.errorWords);
} }
</script>
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

You can get the corrected text content details before updating it into target element with the help of the client side event complete as mentioned in the below code example.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ClientSideEvents(event =>
event.Complete("onCheckComplete"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
function onCheckComplete(args) {
if (args.requestType == "changeErrorWord")
{
alert(args.targetText);
} }
</script>
```

Context Menu Mode

SpellCheck provides default context menu options to perform the spell check operations. It also allows to define additional custom context menu options.

The options that are available under contextMenuSettings are as follows,

- enable - Enables/disables the context menu option in SpellCheck.
- menuItems - Contains the options to perform spell check operations.

Default Menu Options

The menu items contains the following options to perform the spell check operation.

- IgnoreAll
- Add to Dictionary

The following code snippet shows how to enable the context menu settings in SpellCheck and to make use of it with default available options.

ASPX-CS

```
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ContextMenuSettings(items=>items.Enable(t
rue).MenuItems(menuItem=>menuItem.ID("IgnoreAll").Text("Ignore All")))
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: For default menu items, the id must be defined the same as mentioned in the above code example – as we have processed the menus based on this id within our source.

Custom Menu Options

Apart from the default available options, it is also possible to add custom menu options to the context-menu list.

The following code example depicts how to add the custom menu items into the context menu settings.

ASPX-CS

```
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ContextMenuSettings(items=>items.Enable(t
rue).MenuItems(menuItem=>menuItem.ID("IgnoreAll").Text("Ignore All")))
```

Note: The id given for the custom menu items must be unique.

Handling Menu Actions

The client-side event contextClick can be used to define specific actions when a click made on the custom menu items. The following code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ContextMenuSettings(items =>
items.Enable(true).MenuItems(menuItem =>
menuItem.ID("IgnoreAll").Text("Ignore All"))).ClientSideEvents(events =>
events.ContextClick("onCustomMenuClick"))
}
@section ScriptSection{
<script type="text/javascript">
function onCustomMenuClick(args) {
if (args.selectedOption == "Ignore")
{
alert("Custom menu clicked");
} }
</script>
```

It is possible to predict the target (error word) on which the right click is made with the use of the event contextOpen. The below code example shows how to block the display of context menu, when right clicked on the word by setting args.cancel as true.

ASPX-CS

```
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ContextMenuSettings(items =>
items.Enable(true).MenuItems(menuItem =>
menuItem.ID("IgnoreAll").Text("Ignore All"))).ClientSideEvents(events =>
events.ContextOpen("onContextOpen"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@section ScriptSection{
<script type="text/javascript">
function onContextOpen (args) {
if (args.selectedOption == "Ignore")
{
alert("Custom menu clicked");
} }
</script>
```

You can get the current spell check operation arguments details with the client-side event validating. The following code example describes the way to block the ignoreAll operation for the particular word.

ASPX-CS

```
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary  =>
dictionary
.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUrl("
../api/SpellCheck/CheckWords")).ContextMenuSettings(items =>
items.Enable(true).MenuItems(menuItem =>
menuItem.ID("IgnoreAll").Text("Ignore All"))).ClientSideEvents(events =>
events.Validating("onSpellChecking"))
@section ScriptSection{
<script type="text/javascript">
function onSpellChecking (args) {
if (args.selectedOption == "Ignore")
{
alert("Custom menu clicked");
} }
</script>
```

Functionalities

Check Spelling

The main functionality of the SpellCheck control is checking the spelling of a word and returns the suggestions for an error word.

For example, if you pass the sentence that contains misspell words as input, you can know the error words in this sentence and its suggestions (apt words to replace).

Ignore Words

The SpellCheck control provides the support to ignore the words from an error word consideration and also to ignore an error words whenever needed. Please refer the following options to ignore the words.

Ignore

The ignore option is used to ignore a specific word once from the given input string.

The following code example describes the above method implementation.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
@section ScriptSection{
<script type="text/javascript">
var target = 'The first <span class="errorspan e-
errorword">textarea</span> <span class="errorspan e-
errorword">sample</span> uses a dialog <span class="errorspan e-
errorword">textarea</span> to display the <span class="errorspan e-
errorword">sample</span> spell <span class="errorspan e-
errorword">textarea</span> errors';
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
var targetIgnoreContent = spellObj.ignore("textarea", target, null);
}
</script>
}
```

IgnoreAll

The ignoreAll option is used to ignore all the error word occurrences from the given input string.

The following code example describes the way of using ignore all method.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
var target = 'The first <span class="errorspan e-
errorword">textarea</span> <span class="errorspan e-
errorword">sample</span> uses a dialog <span class="errorspan e-
errorword">textarea</span> to display the <span class="errorspan e-
errorword">sample</span> spell <span class="errorspan e-
errorword">textarea</span> errors';
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
var targetIgnoreContent = spellObj.ignoreAll("textarea", target, null);
}
</script>
}
```

Word Collection to Ignore

The ignore words option is used to ignore the collection of words from an error word checking. You can pass the technical terms, brand names which is not present in the dictionary file to this property ignoreWords as shown in the following code example and then while checking the spelling these passed words are considered as a correct word.

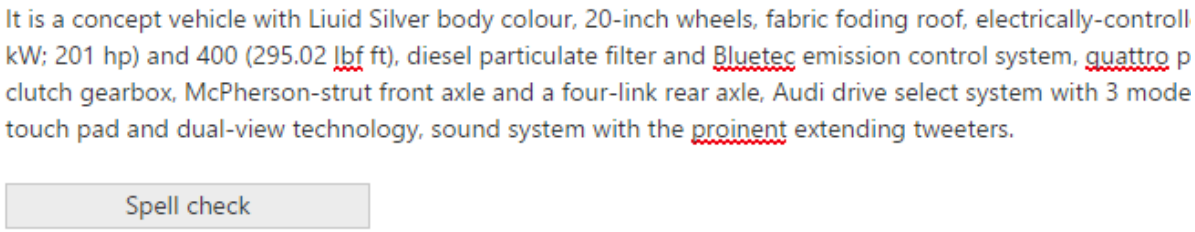
ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters JavaScript TypeScript .
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).ContextMenuSettings(context =>
context.Enable(true)).IgnoreWords(@ViewBag.datasource)
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("contextMenu"))
}
@section ScriptSection{
<script type="text/javascript">
function contextMenu() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.validate();
}
</script>
}
public ActionResult ContextMenu()
{
string[] spellIgnoreWords = { "color", "Liquid", "folding" };
ViewBag.datasource = spellIgnoreWords;
return View();
}
```

Once you have run the above code, you get an output like below.



Ignore Settings

The ignore settings helps to ignore the uppercase, mixed case words, alphanumeric words, file path and email addresses from the error word checking. Ignore settings contains the following options to ignore the words based on their category.

Copyright © 2001 - 2020 Syncfusion Inc.

- ignoreAlphaNumericWords - ignoring the alpha numeric words from an error word consideration.
- ignoreUpperCase - ignoring the upper case words from an error word consideration.
- ignoreMixedCaseWords - ignoring the mixed case words from an error word consideration.
- ignoreFileNames - ignoring the file address path from an error word consideration.
- ignoreUrl - ignoring the url links from an error word consideration.
- ignoreEmailAddress - ignoring the email address from an error word consideration.

The following code example uses to enable the checking of all the words formed with alphanumeric, uppercase, mixed case words and file paths and email addresses.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).ContextMenuSettings(context =>
context.Enable(true)).IgnoreSettings(ignore=>ignore.IgnoreAlphaNumericWor
ds(false).IgnoreEmailAddress(false).IgnoreFileNames(false).IgnoreHtmlTags
(false).IgnoreMixedCaseWords(false).IgnoreUpperCase(false).IgnoreUrl(fals
e))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("contextMenu"))
}
@section ScriptSection{
<script type="text/javascript">
function contextMenu () {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.validate();
}
</script>
}
```

Change Words

The SpellCheck control provides the support to change an error words from its possible suggestions. Please refer the following options to change an error word.

Change

The change option is used to replace an error word occurrences once from the given input string with the correct word.

The following code example describes the behavior of change method.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
var target = 'The first <span class="errorspan e-
errorword">textarea</span> <span class="errorspan e-
errorword">sample</span> uses a dialog <span class="errorspan e-
errorword">textarea</span> to display the <span class="errorspan e-
errorword">sample</span> spell <span class="errorspan e-
errorword">textarea</span> errors';
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
var targetChangeContent = spellObj.change("textarea", target,"text
area",null);
}
</script>
}
```

ChangeAll

The changeAll option is used to replace all the occurrences of an error word with the correct word(selected from the suggestions list) from the given inputs string.

The following code example uses to change all the error word occurrences.

ASPX-CS

```
@section ControlsSection{
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
var target = 'The first <span class="errorspan e-
errorword">textarea</span> <span class="errorspan e-
errorword">sample</span> uses a dialog <span class="errorspan e-
errorword">textarea</span> to display the <span class="errorspan e-
errorword">sample</span> spell <span class="errorspan e-
errorword">textarea</span> errors';
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
var targetChangeAllContent = spellObj.changeAll("textarea", target,"text
area", null);
}
</script>
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Custom Words

The SpellCheck control provides the support to add the custom words into the custom dictionary file.

The addToDictionary option is used to add the custom words into the custom dictionary file.

The following code example uses to add the custom word into the custom dictionary file.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).ContextMenuSettings(context =>
context.Enable(true)).IgnoreSettings(ignore=>ignore.IgnoreAlphaNumericWor
ds(false).IgnoreEmailAddress(false).IgnoreFileNames(false).IgnoreHtmlTags
(false).IgnoreMixedCaseWords(false).IgnoreUpperCase(false).IgnoreUrl(fals
e))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.addToDictionary("Liquid");
}
</script>
}
```

You can also add the custom words into the custom dictionary file through the dialog mode or context menu mode add to dictionary option.

- Dialog Mode - Add To Dictionary button will be available in the dialog window, while checking the error in the given input string and clicking this button then the word will be added into the custom dictionary file.
- Context Menu Mode - Add To Dictionary option will be available in the context menu while right click on the error word in the target area and clicking this option then the word will be added into the custom dictionary file.

Checking content on typing

SpellCheck control provides support for checking the content, on pressing the Enter and Space key. The cursor position will also be properly retained, while processing the SpellCheck operations. If you enable “enableValidateOnType” property, the spellcheck operation will be carried out on typing.

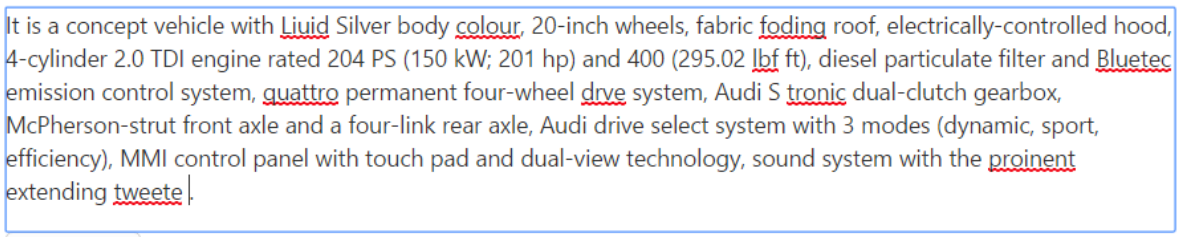
Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dic =>
dic.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUr
l("../api/SpellCheck/CheckWords")).ContextMenuSettings(context =>
context.Enable(true)).EnableValidateOnType(true)
}
```

The following screenshot displays the output for the above code



You can also validate the content within the IFrame element or IFrame element target text, by passing the IFrame element id or class name value to the controlsToValidate property.

Detailed information is given here

Suggestion Words

The getSuggestionWords option is used to retrieve the possible suggestion words for an error word which is provided to correct that spelling.

The following code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dic =>
dic.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUr
l("../api/SpellCheck/CheckWords"))
}
@section ScriptSection{
<script type="text/javascript">
$( document ).ready(function() {
var spellObj = $("#SpellCheck").data("ejSpellCheck");
spellObj.getSuggestionWords("textarea");
setTimeout(()=>{
alert(spellObj._suggestedWords);
},800);
});
</script>
}
```

Note: You can get the suggestion words after some time interval once this method is called. Since, AJAX request processing takes place in the background.

Synchronous request

On setting enableAsync option to false, enables the synchronous request to the server to perform SpellCheck operations.

The following code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dic =>
dic.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").DictionaryUr
l("../api/SpellCheck/CheckWords")).EnableAsync(false).AjaxDataType("json"
)
}
```

Note: You need to set the ajaxDataType value as json to retrieve the synchronous request result properly.

Customization

The Essential ASP.NET MVC SpellCheck provides option to customize for the following scenarios.

- Misspell Word Appearance
- Restrict Suggestion Count

Copyright © 2001 - 2020 Syncfusion Inc.

Misspell Word Appearance

The SpellCheck control provide the support(misspellWordCss) to display the error word in user defined style. By default displaying the error words with the red underline.

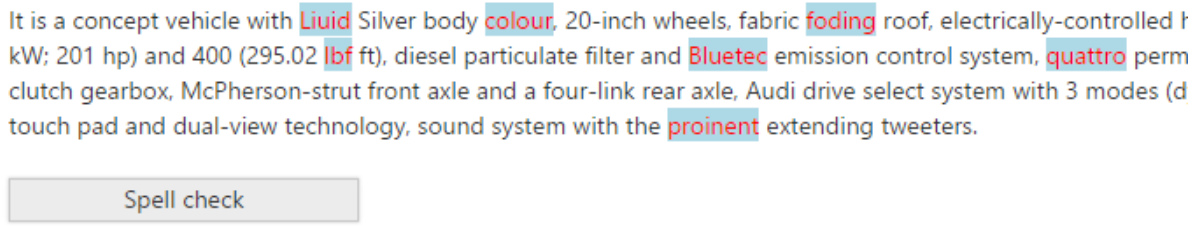
The following code example depicts the way to customize the error word highlight (displaying error word with red color font and lightblue background).

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).MisspellWordCss("highlight")
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("contextMenu"))
}
@section ScriptSection{
<script type="text/javascript">
function contextMenu () {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.validate();
}
</script>
}
@section StyleSection{
<style>
.highlight {
background-color: lightblue;
color: red;
}
</style>
}
```

Once you have run the above code, you get an output like below.

Copyright © 2001 - 2020 Syncfusion Inc.



Restrict Suggestion Count

The SpellCheck control provides option (maxSuggestionCount) to restrict the count that the number of items displayed in the suggestion list.

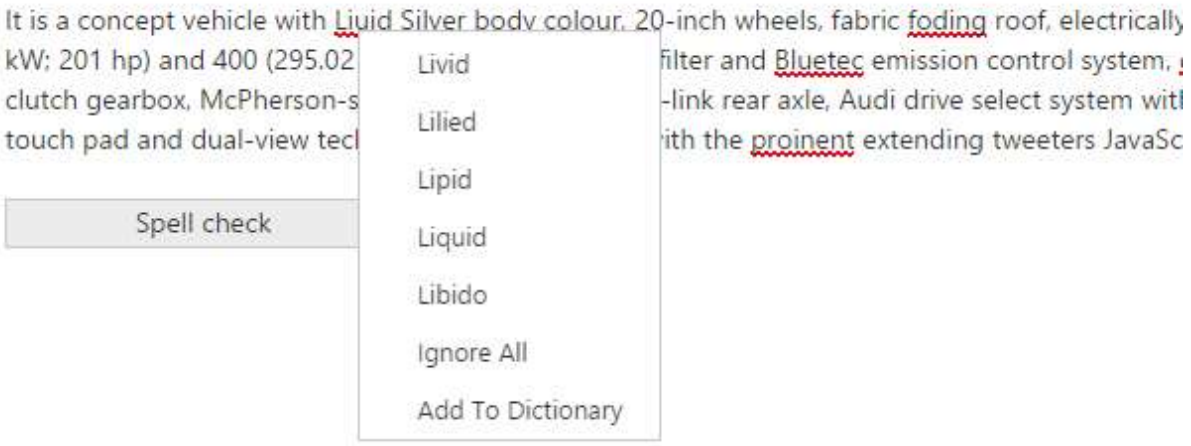
The following code example describes the way to control the suggestion count.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
It is a concept vehicle with Liquid Silver body color, 20-inch wheels,
fabric folding roof, electrically-controlled hood,
4-cylinder 2.0 TDI engine rated 204 PS (150 kW; 201 hp) and 400 (295.02
lbf ft), diesel particulate filter and Bluetech emission control system,
quattro permanent four-wheel drive system, Audi S tronic dual-clutch
gearbox, McPherson-strut front axle and a four-link rear axle, Audi drive
select system with 3 modes (dynamic, sport, efficiency),
MMI control panel with touch pad and dual-view technology, sound system
with the prominent extending tweeters.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).MaxSuggestionCount(5)
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check").ClientSideEvents(event => event.Click("contextMenu"))
}
@section ScriptSection{
<script type="text/javascript">
function contextMenu () {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.validate();
}
</script>
}
```

Once you have run the above code, you get an output like below.

Copyright © 2001 - 2020 Syncfusion Inc.



Localization

SpellCheck dialog mode comes with default localization support which allows it to customize the display of text within the SpellCheck dialog in a user-specific culture and locale. The SpellCheck control can be localized in specific culture using the common API locale along with the collection of localized words defined for that culture using the ej.SpellCheck.Locale [**culture-code**].

By default, the SpellCheck control is localized in **en-US** culture. Please find the following table lists the default keywords and its localized text values for en-US culture.

Locale key words	Text
SpellCheckButtonText	SpellCheck
NotInDictionary	Not in Dictionary
SuggestionLabel	Suggestions
IgnoreOnceButtonText	Ignore Once
IgnoreAllButtonText	Ignore All
AddToDictionary	Add to Dictionary
ChangeButtonText	Change
ChangeAllButtonText	Change All
CloseButtonText	Close
CompletionPopupMessage	Spell check is complete
CompletionPopupTitle	Spell check alert
OK	OK

Copyright © 2001 - 2020 Syncfusion Inc.

NoSuggestionMessage	No suggestions available

To localize SpellCheck into any particular culture, it is necessary to refer the culture-specific script files in your application after the reference of **ej.web.all.min.js** file, which are available under the following location.

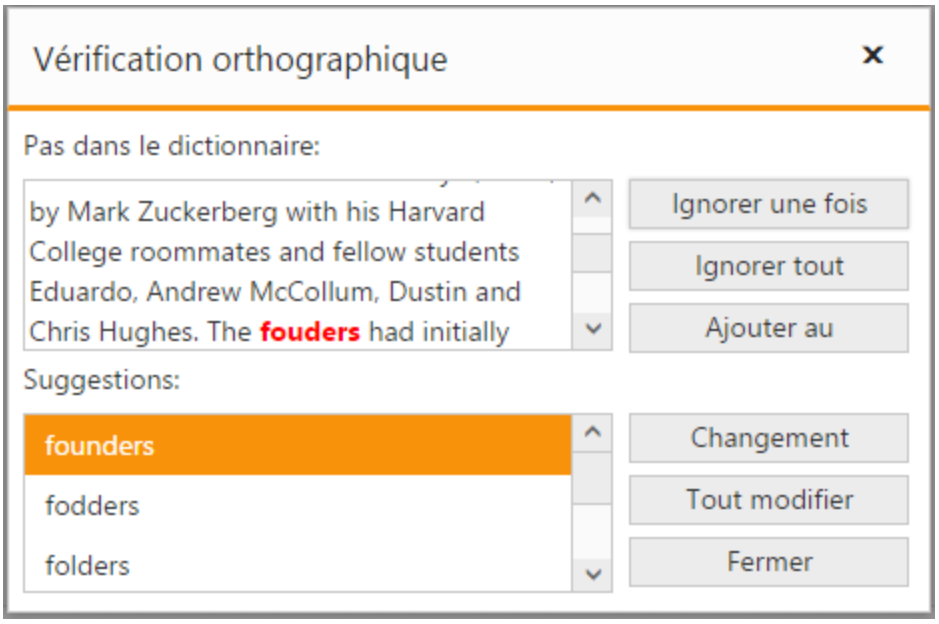
<Installed location>\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

The following code example shows how to localize the SpellCheck control in **fr-FR** culture.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
Facebook is a social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark
Zuckerberg with his Harvard College roommates and fellow students
Eduardo, Andrew McCollum, Dustin and Chris Hughes.
The founders had initially limited the websites membership to Harvard
students, but later expanded it to collages in the Boston area, the Ivy
League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).Locale("fr-FR")
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
ej.SpellCheck.Locale["fr-FR"] = {
SpellCheckButtonText: "Vrification orthographique",
NotInDictionary: "Pas dans le dictionnaire",
SuggestionLabel: "Suggestions",
IgnoreOnceButtonText: "Ignorer une fois",
IgnoreAllButtonText: "Ignorer tout",
AddToDictionary: "Ajouter au dictionnaire",
ChangeButtonText: "Changement",
ChangeAllButtonText: "Tout modifier",
CloseButtonText: "Fermer",
CompletionPopupMessage: "Vrification orthographique est termine",
ErrorPopupMessage: "Vrification orthographique est pas termine",
CompletionPopupTitle: "Vrification orthographique alerte",
OK: "D'accord",
NoSuggestionMessage: "Aucune suggestion disponible",
};
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
</script>
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Note: Refer the **ej.culture.fr-FR.min.js** file in your HTML application and also define the **locale** property for the SpellCheck control with the appropriate **culture-code** [**fr-FR**].

For further information on how to refer the required culture scripts into your application, refer here.

Localizing Specific Words

To customize or localize only some specific words in the default ej.SpellCheck.Locale["en-US"] collection, the words to be localized/customized can be defined in a separate variable and then extended to the original collection as depicted in the following code example.

ASPX-CS

```
@section ControlsSection{
<div id="TextArea" contenteditable="true" name="sentence">
Facebook is a social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark
Zuckerberg with his Harvard College roommates and fellow students
Eduardo, Andrew McCollum, Dustin and Chris Hughes.
The founders had initially limited the websites membership to Harvard
students, but later expanded it to collages in the Boston area, the Ivy
League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
</div><br />
@Html.EJ().SpellCheck("TextArea").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords")).Locale("fr-FR")
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
var customizationMessage = { CompletionPopupMessage: "Completed", };
$.extend(ej.SpellCheck.Locale["en-US"], customizationMessage);
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
}
```

Multiple Targets

The SpellCheck control has support for multiple target HTML element's texts spelling and correct its error words. This can be achieved by passing the target HTML elements “id” values to the property “ControlsToValidate”.

You can spell check most of the HTML element’s texts such as div, span, textarea, input, address and article by using this property.

The multiple target HTML element supports in both the modes.

- Dialog Mode
- Context Menu Mode

The following code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{
<div id="control1">
London, one of the most popular tourist destinations in the world for a
reason. A cultural and historical hub, London has an excellent public
transportation system that allows visitors to see all the fantastic
sights without spending a ton of money on a rental car.
London contains four World Heritage Sites.
</div><br />
<textarea id="control2" style="width:940px">
Paris, the city of lights and love - this short guide is full of ideas
for how to make the most of the romanticism that oozes from every one of
its beautiful corners.You couldn't possibly visit Rome, one of the
world's most fascinating cities. The old adage that Rome was not built in
a day - and that you won't see it in one or even in three - is true. For
the intrepid traveler who can keep pace, here is a list of must-sees.
But remember what the Romans say: Even a lifetime isn't enough to see
Rome.
</textarea><br />
<span id="control3">
Rome, one of the world's most fascinating cities. The old adage that Rome
was not built in a day - and that you won't see it in one or even in
three - is true. For the intrepid traveler who can keep pace, here is a
list of must-sees.
But remember what the Romans say: Even a lifetime isn't enough to see
Rome.
</span><br /><br />
@Html.EJ().SpellCheck("TextArea").ControlsToValidate("control1,control2,c
ontrol3").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Text("Spell check using
dialog").ClientSideEvents(event => event.Click("showDialog"))
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Note: You need to pass the target HTML element's id value with the comma separator. For example, in the above code example passed id values of div(control1), textarea(control2) and span(control3) element.

Dialog Mode

If the user specifies multiple targets in ControlsToValidate property, the control must loop through the multiple HTML target elements one by one in the order which they are specified and process of every individual target' content.

ASPX-CS

```
@section ControlsSection{
<div id="control1">
London, one of the most popular tourist destinations in the world for a
reason. A cultural and historical hub, London has an excellent public
transportation system that allows visitors to see all the fantastic
sights without spending a ton of money on a rental car.
London contains four World Heritage Sites.
</div><br />
<textarea id="control2" style="width:940px">
Paris, the city of lights and love - this short guide is full of ideas
for how to make the most of the romanticism that oozes from every one of
its beautiful corners.You couldn't possibly visit Rome, one of the
world's most fascinating cities. The old adage that Rome was not built in
a day - and that you won't see it in one or even in three - is true. For
the intrepid traveler who can keep pace, here is a list of must-sees.
But remember what the Romans say: Even a lifetime isn't enough to see
Rome.
</textarea><br />
<span id="control3">
Rome, one of the world's most fascinating cities. The old adage that Rome
was not built in a day - and that you won't see it in one or even in
three - is true. For the intrepid traveler who can keep pace, here is a
list of must-sees.
But remember what the Romans say: Even a lifetime isn't enough to see
Rome.
</span><br /><br />
@Html.EJ().SpellCheck("TextArea").ControlsToValidate("control1,control2,c
ontrol3").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Text("Spell check using
dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
</script>
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Context Menu Mode

By using ControlsToValidate property, you can spell check Multiple HTML elements at the same time through context menu. You can correct error word by right click on the error word and choose the corresponding context menu option.

ASPX-CS

```
@section ControlsSection{
<div id="control1">
London, one of the most popular tourist destinations in the world for a
reason. A cultural and historical hub, London has an excellent public
transportation system that allows visitors to see all the fantastic
sights without spending a ton of money on a rental car.
London contains four World Heritage Sites.
</div><br />
<textarea id="control2" style="width:940px">
Paris, the city of lights and love - this short guide is full of ideas
for how to make the most of the romanticism that oozes from every one of
its beautiful corners.You couldn't possibly visit Rome, one of the
world's most fascinating cities. The old adage that Rome was not built in
a day - and that you won't see it in one or even in three - is true. For
the intrepid traveler who can keep pace, here is a list of must-sees.
But remember what the Romans say: Even a lifetime isn't enough to see
Rome.
</textarea><br />
<span id="control3">
Rome, one of the world's most fascinating cities. The old adage that Rome
was not built in a day - and that you won't see it in one or even in
three - is true. For the intrepid traveler who can keep pace, here is a
list of must-sees.
But remember what the Romans say: Even a lifetime isn't enough to see
Rome.
</span><br /><br />
@Html.EJ().SpellCheck("TextArea").ControlsToValidate("control1,control2,c
ontrol3").DictionarySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Text("Spell check using context
menu").ClientSideEvents(event => event.Click("showContextMenu"))
}
@section ScriptSection{
<script type="text/javascript">
function showContextMenu() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.validate();
}
</script>
}
```

Note: We couldn’t highlight the error word in the target element itself, while checking the elements that doesn’t support the html or innerHTML value.

Responsiveness

The SpellCheck control has support for responsive behavior based on client browser’s width and height. To enable responsive, IsResponsive property should be true.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

ASPX-CS

```
@section ControlsSection{<div id="TextArea" contenteditable="true"
name="sentence">
Facebook is a social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark
Zuckerberg with his Harvard College roommates and fellow students
Eduardo, Andrew McCollum, Dustin and Chris Hughes.
The founders had initially limited the websites membership to Harvard
students, but later expanded it to collages in the Boston area, the Ivy
League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
</div><br />
@Html.EJ().("TextArea").SpellCheck("TextArea").IsResponsive(true).Diction
arySettings(dictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
}
```

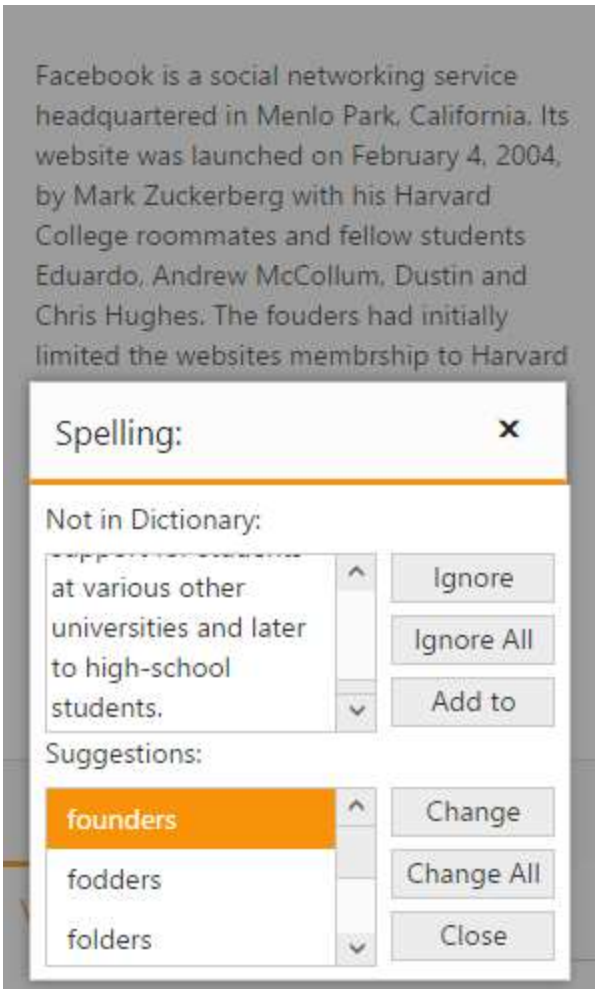
The dialog of spell check control is rendering based on the client browser's width and height, it can be achieved by IsResponsive property as true.

ASPX-CS

```
@section ControlsSection{<div id="TextArea" contenteditable="true"
name="sentence">
Facebook is a social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark
Zuckerberg with his Harvard College roommates and fellow students
Eduardo, Andrew McCollum, Dustin and Chris Hughes.
The founders had initially limited the websites membership to Harvard
students, but later expanded it to collages in the Boston area, the Ivy
League, and Stanford University. It gradually added support for students
at various other universities and later to high-school students.
</div><br />
@Html.EJ().SpellCheck("TextArea").IsResponsive(true).DictionarySettings(d
ictionary =>
dictionary.CustomDictionaryUrl("../api/SpellCheck/AddToDictionary").Dicti
onaryUrl("../api/SpellCheck/CheckWords"))
@Html.EJ().Button("SpellCheck").Width("200px").Height("25px").Text("Spell
check using dialog").ClientSideEvents(event => event.Click("showDialog"))
}
@section ScriptSection{
<script type="text/javascript">
function showDialog() {
var spellObj = $("#TextArea").data("ejSpellCheck");
spellObj.showInDialog();
}
</script>
}
```

Mobile Rendering Screenshot

Copyright © 2001 - 2020 Syncfusion Inc.



SplitButton

SplitButton

The Split Button allows you to perform an action by clicking the button and choosing extra options from the dropdown button. The SplitButton can also display both text and images. By default the DropDown button is given to the right most corner of the button.

Key Features

- Trendy Look : Rich Appearance with Theme Support
- RTL : Supports for Right to Left alignment
- Text and Image : Supports both text and image as Split Button content
- Built-in Icon :Supports the built-in icon libraries
- Easy Customization: The customization of Split Button control to any form is made simple

Getting Started

This section explains briefly about how to create a Split Button in ASP.NET MVC application.

Copyright © 2001 - 2020 Syncfusion Inc.

Create your first Split Button in MVC

The HTML button element and the <UL>, <LI> can be easily configured as Essential ASP.NET MVC Split Button control. The basic rendering of Essential ASP.NET MVC Split Button is achieved by using default functionality. Initially the TargetID is a mandatory one, without this field it acts as normal button on two sides.

Create Split Button Control in MVC

Essential ASP.NET MVC Split Button control contains built-in features such as Click and different option choosing. You can easily create the Split Button control by using HTML helper as follows.

1. You can create an MVC Project and add necessary assemblies, styles and scripts to it. Refer MVC-Getting Started.
2. Add the following code to the corresponding View page to render Split Button.

ASPX-CS

```
@Html.EJ().SplitButton("splitButton").Text("Save").ShowRoundedCorner(true
).Size(ButtonSize.Large).TargetID("target")
```

3. Add the following

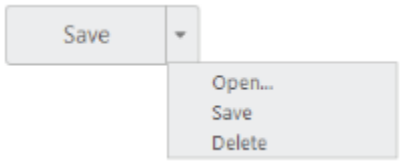
,

- elements to render Split Button with popup option.

ASPX-CS

```
<ul id="target">
<li><span>Open</span></li>
<li><span>Save</span></li>
<li><span>Delete</span></li>
</ul>
```

Output of the above scripts,



Easy customization

Split Button is used in many applications. Split Button Size, images and Content are varied according to each application. The following sections describes you some customizable options for Split Button that can perform easily.

Content for Split button

The TargetId is a mandatory one, without this field it acts as normal button on two sides. This TargetId property is used to specify the list of content for Split Button. The list of content is rendered as a vertical menu list. This vertical menu list is open, when you click on the down arrow of the Split Button.

Copyright © 2001 - 2020 Syncfusion Inc.

The following script explains you the details about rendering the Split Button with content.

1. In the VIEW page, add the following button elements to configure Split Button widget.

ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
@*Set the content for split button control as follows. *@
<div class="control">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton").Text("login").Size(ButtonSize.Small
).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).PrefixIco
n("e-icon e-
login").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul11")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</div>
```

Execute the above code to render the following output.



Button Size

You can render the Split Button in different sizes by using Size property. The following table contains some predefined size option for rendering a Split Button in easiest way. Each size option has different height and width. Mainly it avoids the complexity in rendering Split Button with complex CSS class.

List of Button size

Normal	Creates split button with content size.
Mini	Creates split button with Built-in mini size height, width specified.
Small	Creates split button with Built-in small size height, width specified.
Medium	Creates split button with Built-in medium size height, width specified.
Large	Creates split button with Built-in large size height, width specified.

Apart from the above mentioned predefined size option, you can set your own width and height for Split Button using Height and Width property.

Copyright © 2001 - 2020 Syncfusion Inc.

The following steps explains you the details about rendering the Split Button with above mentioned Size options.

1. In the View page, add the following button elements to configure Split Button widget.

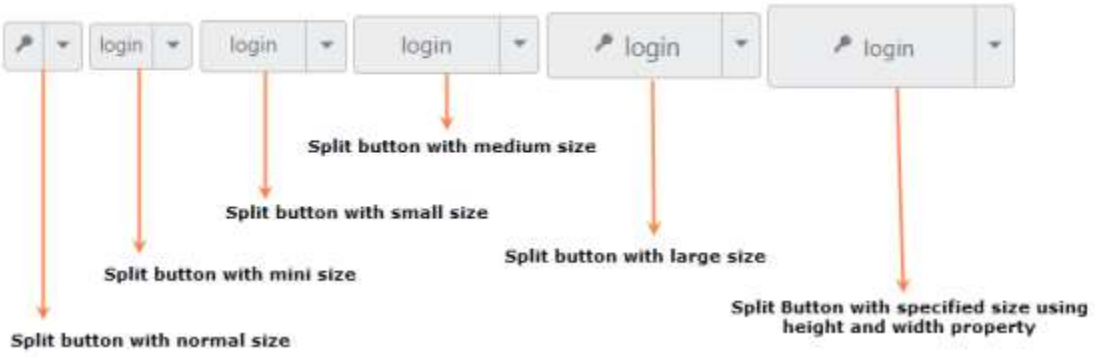
ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
@*Set the different size types for split button control as follows. *@
<div class="control">
<table>
<tr>
<td class="btnsht">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_normal").Text("login").ShowRoundedCor
ner(true).Size(ButtonSize.Normal).ContentType(ContentType.ImageOnly).Pref
ixIcon("e-icon e-login").TargetID("Ul11")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
@Html.EJ().SplitButton("splitButton_mini").Text("login").ShowRoundedCorne
r(true).Size(ButtonSize.Mini).TargetID("Ul21")
<ul id="Ul21">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_small").Text("login").ShowRoundedCorn
er(true).Size(ButtonSize.Small).TargetID("Ul31")
<ul id="Ul31">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_medium").Text("login").ShowRoundedCor
ner(true).Size(ButtonSize.Medium).TargetID("Ul41")
<ul id="Ul41">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_large").Text("login").ShowRoundedCorn
er(true).Size(ButtonSize.Large).ContentType(ContentType.TextAndImage).Pre
fixIcon("e-icon e-login").TargetID("Ul51")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<ul id="Ul51">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_customSize").Text("login").ShowRounde
dCorner(true).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-
login").TargetID("Ul61").Height("50").Width("150")
<ul id="Ul61">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>
</div>
```

Execute the above code to render the following output.



ContentType

The content of the Split Button is mainly rendered as text and images. Instead of using complex CSS classes to render Split Button with different content types, you can use some predefined content type options listed in the following table. Using this ContentType property you can easily add different types of content for Split Button. Split Button supports the following content types.

List of content types

TextOnly	Supports only for text content only.
ImageOnly	Supports only for image content only
ImageBoth	Supports image for both ends of the button.
TextAndImage	Supports image with the text content.
ImageTextImage	Supports image with both ends and middle in text.

Copyright © 2001 - 2020 Syncfusion Inc.

Prefix and Suffix icons

Icons inside the Split Button is added easily using PrefixIcon and SuffixIcon property. Location of the icon in Split Button is a necessary thing. You can customize the location of Icon easily using the following mentioned options.

Split Button control also supports the Built-in icon libraries. The ej.widgets.core.min.css contains definitions for important icons that are used in Split Buttons. You can use these Built-in icons by mentioning the icon class name as value in PrefixIcon and SuffixIcon property. You can use any font icons that is defined in ej.widgets.core.min.css. It avoids the complexity in specifying icon using sprite image and CSS.

For example the following Built-in CSS classes are used to display the font icons that is used by media player.

- e-mediaback
- e-mediaforward
- e-medianext
- e-mediaprev
- e-mediaeject
- e-mediaclose
- e-mediapause
- e-mediaplay

Prefix Icon

It inserts the icon at the starting position of Split Button. After this prefix icon, you can use text or suffix icon.

Suffix Icon

It inserts the icon at the ending position of Split Button. Before this SuffixIcon, you can use Text or PrefixIcon.

The following steps explains you the details on rendering the Split Button with ContentType, PrefixIcon and SuffixIcon options

1. In the VIEW page, add the following button elements to configure Split Button widget.

ASPX-CS

```
//Add the code in the CSHTML page to configure and initialize the control
@*Set the different content types and use the necessary built-in icons
for split button control as follows.*@
<div class="control">
<table>
<tr>
<td class="btnsht">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_imageOnly").Text("login").Size(Button
Size.Medium).ShowRoundedCorner(true).ContentType(ContentType.ImageOnly).P
refixIcon("e-icon e-handup").TargetID("Ul11")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
@Html.EJ().SplitButton("splitButton_textOnly").Text("login").Size(ButtonS
ize.Medium).ShowRoundedCorner(true).ContentType(ContentType.TextOnly).Tar
getID("Ul21")
<ul id="Ul21">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_imageBoth").Text("login").Size(Button
Size.Medium).ShowRoundedCorner(true).ContentType(ContentType.ImageBoth).P
refixIcon("e-icon e-handup").SuffixIcon("e-icon e-
palette").TargetID("Ul31")
<ul id="Ul31">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_textAndImage").Text("login").Size(But
tonSize.Medium).ShowRoundedCorner(true).ContentType(ContentType.TextAndIm
age).PrefixIcon("e-icon e-handup").TargetID("Ul41")
<ul id="Ul41">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_imageTextImage").Text("login").Size(B
uttonSize.Medium).ShowRoundedCorner(true).ContentType(ContentType.ImageTe
xtImage).PrefixIcon("e-icon e-handup").SuffixIcon("e-icon e-
palette").TargetID("Ul51")
<ul id="Ul51">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>
</div>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.

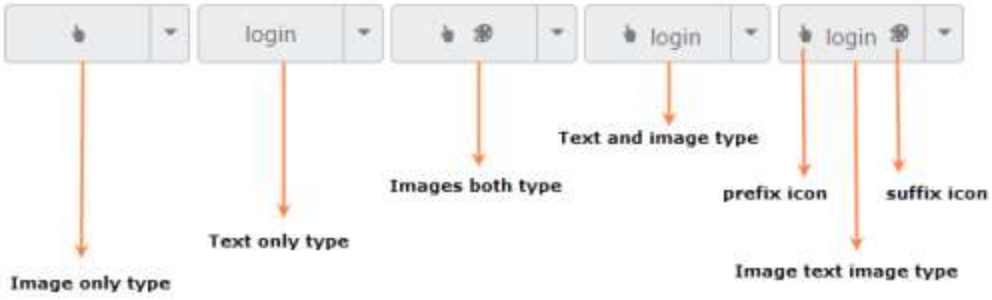


Image Position

To provide the best look and feel for Split Button, position of images in Split Button is important. Using ImagePosition property, you can easily customize the position of images inside Split Button without using any complex CSS. ImagePosition property is applicable only with the TextAndImage content type property. This property represent the position of images with respect to text.

ImageLeft	Support for aligning text in right and image in left.
ImageRight	Support for aligning text in left and image in right.
ImageTop	Support for aligning text in bottom and image in top.
ImageBottom	Support for aligning text in top and image in bottom.

The following steps explains you the details on rendering the Split Button with above mentioned ImagePosition property's options.

1. In the View page, add the following button elements to configure Split Button widget.

ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
<div class="control">
@*Set the different image position for split button control as follows.*@
<table>
<tr>
<td class="btnsht">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_normal_imageLeft").Text("login").Size
(ButtonSize.Normal).ShowRoundedCorner(true).ContentType(ContentType.TextA
ndImage).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul11")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
@Html.EJ().SplitButton("splitButton_small_imageLeft").Text("login").Size(
ButtonSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAnd
```

Copyright © 2001 - 2020 Syncfusion Inc.

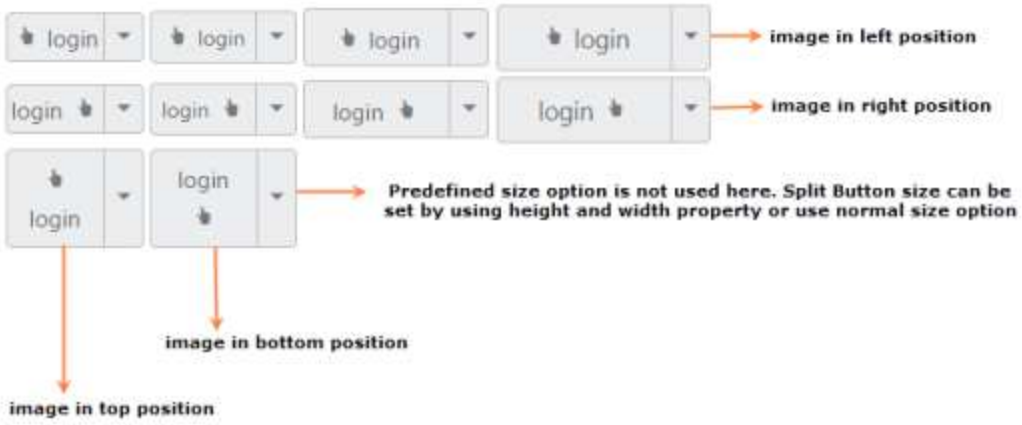
```
Image).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul21")
<ul id="Ul21">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_medium_imageLeft").Text("login").Size
(ButtonSize.Medium).ShowRoundedCorner(true).ContentType(ContentType.TextA
ndImage).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul31")
<ul id="Ul31">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_large_imageLeft").Text("login").Size(
ButtonSize.Large).ShowRoundedCorner(true).ContentType(ContentType.TextAnd
Image).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul41")
<ul id="Ul41">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
<tr>
<td class="btnsht">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_normal_imageRight").Text("login").Siz
e(ButtonSize.Normal).ShowRoundedCorner(true).ContentType(ContentType.Text
AndImage).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageRight).TargetID("Ul51")
<ul id="Ul51">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
@Html.EJ().SplitButton("splitButton_small_imageRight").Text("login").Size
(ButtonSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAn
dImage).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageRight).TargetID("Ul61")
<ul id="Ul61">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().SplitButton("splitButton_medium_imageRight").Text("login").Size(ButtonSize.Medium).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-handup").ImagePosition(ImagePosition.ImageRight).TargetID("Ul71")
<ul id="Ul71">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_large_imageRight").Text("login").Size(ButtonSize.Large).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-handup").ImagePosition(ImagePosition.ImageRight).TargetID("Ul81")
<ul id="Ul81">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
<tr>
<td class="btnsht">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_imageTop").Text("login").Size(ButtonSize.Normal).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-handup").ImagePosition(ImagePosition.ImageTop).TargetID("Ul91").Height("60")
<ul id="Ul91">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
@Html.EJ().SplitButton("splitButton_imageBottom").Text("login").Size(ButtonSize.Normal).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).PrefixIcon("e-icon e-handup").ImagePosition(ImagePosition.ImageBottom).TargetID("Ul101").Height("60")
<ul id="Ul101">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>
</div>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Theme support

You can control the style and appearance of Split Button based on CSS classes. In order to apply styles to the Split Button control, you can refer two files namely, ej.widgets.core.min.css and ej.theme.min.css. When you refer the ej.widgets.all.min.css file, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Split Button control namely

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Custom CSS

You can use the CSS class to customize the Split Button control appearance. Define a CSS class as per requirement and assign the class name to CssClass property.

The following steps explains you the details about rendering the Split Button with custom CSS.

1. In the VIEW page, add the following button elements to configure Split Button widget.

ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
<div class="control">
@*Set Custom CSS class for split button control as follows*@
<table>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<tr>
<td class="btnsht">
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_customCSS1").Text("login").Size(Butto
nSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage
).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul11").CssClass
("customCss1")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
</td>
<td>
@Html.EJ().SplitButton("splitButton_customCSS2").Text("login").Size(Butto
nSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage
).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul21").CssClass
("customCss2")
<ul id="Ul21">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_customCSS3").Text("login").Size(Butto
nSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage
).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul31").CssClass
("customCss3")
<ul id="Ul31">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_customCSS4").Text("login").Size(Butto
nSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage
).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul41").CssClass
("customCss4")
<ul id="Ul41">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
<td class="btnsht">
@Html.EJ().SplitButton("splitButton_customCSS5").Text("login").Size(Butto
nSize.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage
).PrefixIcon("e-icon e-
handup").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul51").CssClass
("customCss5")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<ul id="Ul51">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</td>
</tr>
</table>
</div>
```

2. Configure the CSS styles to apply on buttons

CSS

```
<style type="text/css">
/* Customize the button background */
.e-split .customCss1
{
background-color: #121111;
}
.e-split .customCss2
{
background-color: #94bbd5;
}
.e-split .customCss3
{
background-color: #f3533c;
}
.e-split .customCss4
{
background-color: #d1eeed;
}
.e-split .customCss5
{
background-color: #deb66e;
}
/* Customize the button image & text color */
.e-split .customCss1.e-btn.e-select .e-icon, .e-split .customCss1.e-
btn.e-select .e-btntxt
{
color: #94bbd5;
}
.e-split .customCss2.e-btn.e-select .e-icon, .e-split .customCss2.e-
btn.e-select .e-btntxt
{
color: #121111;
}
.e-split .customCss3.e-btn.e-select .e-icon, .e-split .customCss3.e-
btn.e-select .e-btntxt
{
color: #cef6f7;
}
.e-split .customCss5.e-btn.e-select .e-icon, .e-split .customCss5.e-
btn.e-select .e-btntxt
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
color: #534f4f;
}
</style>
```

Execute the above code to render the following output.



Dropdown Button

You can change the Split Button as Dropdown Button that consists of a single button that when clicked displays a drop-down list of mutually exclusive items. You can achieve this by using default functionality of Split Button with ButtonMode as ButtonMode.Dropdown. Initially the TargetID is a mandatory one.

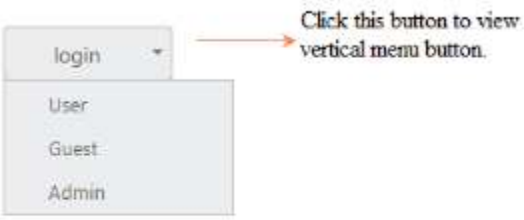
The following steps explain how to change the Split Button as Dropdown Button.

1. In the VIEW page, add the following button elements to configure Button widget.

ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
@Html.EJ().SplitButton("dropdownButton").Text("login").ShowRoundedCorner(
true).Size(ButtonSize.Medium).ContentType(ContentType.TextOnly).TargetID(
"menu1").ButtonMode(ButtonMode.Dropdown)
<ul id="menu1">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
```

2. Execute the above code to render the following output.



Arrow Position

To provide a good look and feel for Split Button, position of arrow in Split Button is important. Using ArrowPosition property, you can easily customize the position of arrow inside Split Button without using any complex CSS. ArrowPosition property is applicable for both Split Button and Dropdown Button. This property represent the position of arrow with menu content.

Copyright © 2001 - 2020 Syncfusion Inc.

List of arrow position

Left	Support for arrow in left.
Right	Support for arrow in right.Â
Top	Support for arrow in top.Â
Bottom	Support for arrow in bottom.

The following steps explain you the details on rendering the Split Button with above mentioned arrow position options.

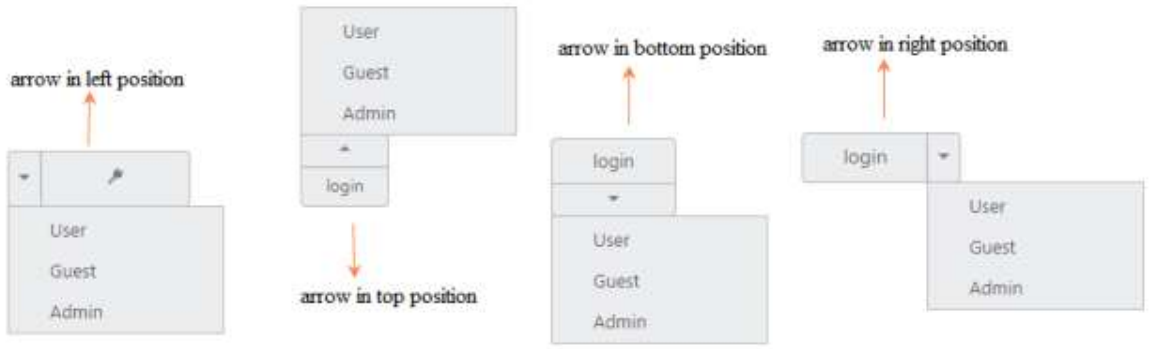
1. In the View page, add the following button elements to configure Split Button widget.

ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
@Html.EJ().SplitButton("spltbutton11").Text("login").ShowRoundedCorner(tr
ue).Size(ButtonSize.Large).ContentType(ContentType.ImageOnly).TargetID("U
l11").PrefixIcon("e-icon e-login").ArrowPosition(ArrowPosition.Left)
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
@Html.EJ().SplitButton("spltbutton21").Text("login").ShowRoundedCorner(tr
ue).Size(ButtonSize.Mini).ContentType(ContentType.TextOnly).TargetID("Ul2
1").ArrowPosition(ArrowPosition.Top)
<ul id="Ul21">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
@Html.EJ().SplitButton("spltbutton31").Text("login").ShowRoundedCorner(tr
ue).Size(ButtonSize.Small).ContentType(ContentType.TextOnly).TargetID("Ul
31").ArrowPosition(ArrowPosition.Bottom)
<ul id="Ul31">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
@Html.EJ().SplitButton("spltbutton41").Text("login").ShowRoundedCorner(tr
ue).Size(ButtonSize.Medium).ContentType(ContentType.TextOnly).TargetID("U
l41").ArrowPosition(ArrowPosition.Right)
<ul id="Ul41">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
```

2. Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



RTL Support

In some cases it is necessary to use right to left alignment. RTL support is provided by using EnableRTL property. In RTL mode when there is more than one content (image/text, image/image) in Split Button, then the content is aligned in right to left format. For example, when text is in left and image is in right position, after applying right to left alignment these position are interchanged.

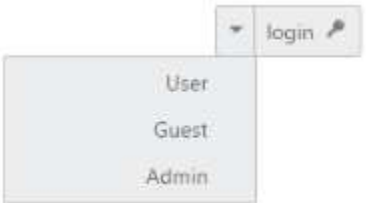
The following steps explains you the details about rendering the button with Right to left alignment support

1. In the VIEW page, add the following button elements to configure Split Button widget.

ASPX-CS

```
@*Add the code in the CSHTML page to configure and initialize the
control*@
@*Enable the alignment format for split button control as follows.*@
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_rtl").Text("login").Size(ButtonSize.S
mall).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Prefi
xIcon("e-icon e-
login").ImagePosition(ImagePosition.ImageLeft).TargetID("Ul11").EnableRTL
(true)
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Miscellaneous

Text

It is necessary to display the user defined text for Split Button. Using Text property, you can easily set text content for Split Button.

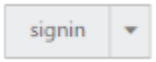
The following steps explains you the details about rendering the Split Button with specified text.

1. In the VIEW page, add the following button elements to configure Split Button widget

ASPX-CS

```
//Add the code in the CSHTML page to configure and initialize the control
@*Set the text for split button control as follows. *@
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_text").Text("sign
in").Size(ButtonSize.Small).TargetID("Ul11")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

Execute the above code to render the following output.



ShowRoundedCorner

Specifies the corner of Split Button in rounded shape. By default, the edges of Split Button is not rounded. To set rounded corner, you can enable ShowRoundedCorner property.

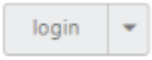
The following steps explains you the details about rendering the Split Button with rounded corner.

1. In the VIEW page, add the following button elements to configure Split Button widget.

ASPX-CS

```
//Add the code in the CSHTML page to configure and initialize the control
@*Enable the rounded corner for split button control as follows.*@
<div class="spltspan">
@Html.EJ().SplitButton("splitButton_roundedCorner").Text("login").Size(Bu
ttonSize.Small).ShowRoundedCorner(true).TargetID("Ul11")
<ul id="Ul11">
<li><span>User</span></li>
<li><span>Guest</span></li>
<li><span>Admin</span></li>
</ul>
</div>
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

How to?

Render SplitButton from Code behind

SplitButton can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of SplitButton properties in the controller.

C#

```
<ul id="target">
<li><span>Open</span></li>
<li><span>Save</span></li>
<li><span>Delete</span></li>
</ul>
//Namespace to get the JavaScript (SplitButton) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class SplitButtonController : Controller
{
// GET: SplitButton
public ActionResult SplitButtonFeatures()
{
//Initializing the SplitButton model
SplitButtonProperties BtnObj = new SplitButtonProperties();
//Initializing the datasource and other properties
BtnObj.Text = "login";
BtnObj.Size= ButtonSize.Medium;
BtnObj.TargetID = "target";
BtnObj.ShowRoundedCorner=true;
//Passing SplitButton properties using the ViewData
ViewData["BtnModel"] = BtnObj;
return View();
}
}
}
```

Binding the SplitButton properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().SplitButton("searchCustomer",(Syncfusion.JavaScript.Models.Spli
tButtonProperties)ViewData["BtnModel"]).Render();
}
```

Splitter

Splitter

The Essential ASP.NET MVCSplitter is a layout control that enables you to divide a Web page into distinct areas by inserting resizable panes. You can create any number of Splitter panes and place them inside the Splitter control; the split bars are inserted automatically in between the adjacent panes.

Copyright © 2001 - 2020 Syncfusion Inc.

Key Features

- Expand or Collapse Panes—Support for expanding or collapsing panes by using the Expand or Collapse arrows.
- Animation—Supports for animating panes while expanding or collapsing the panes, and also setting the animation speed.
- Window Resizing—Supports for resizing the Splitter while resizing the window.
- RTL—Supports for right to left alignment of Splitter panes.
- Themes—Essential ASP.NET MVC controls consist of 12 built-in themes (6 – flat and 6 – gradient effects), and also support custom skins that enable to set user-defined themes.

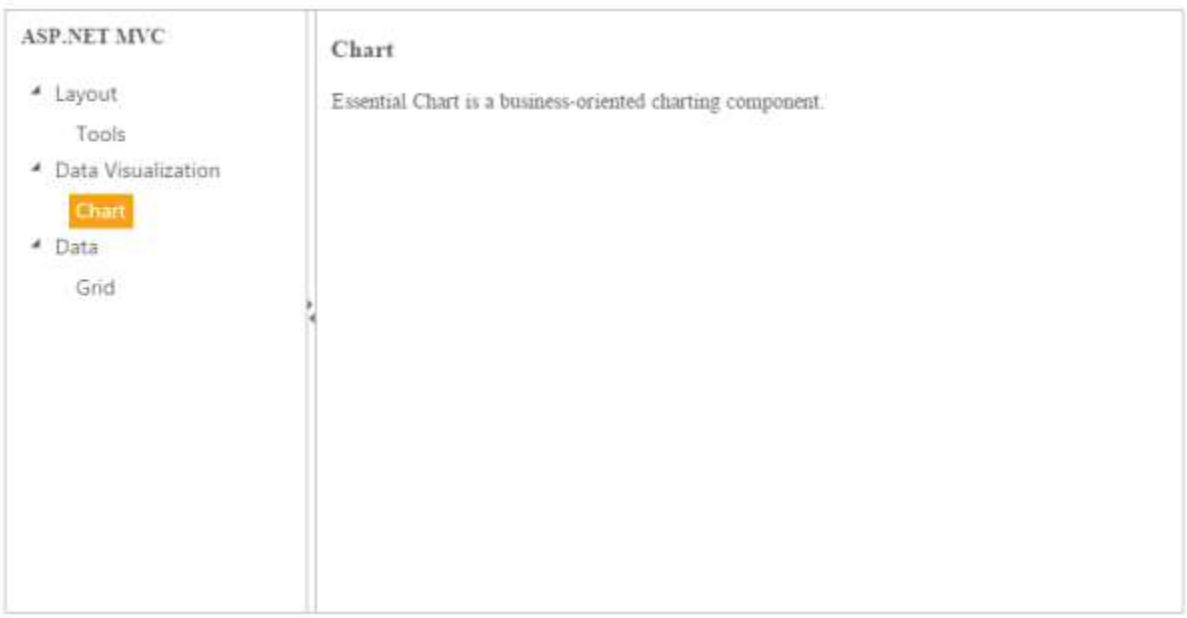
Getting Started

This section explains briefly about how to create a Splitter in ASP.NET MVC application.

Create your first Splitter in MVC

ASP.NET MVCSplitter control consists of movable split bar(s) that divides the container's display area into two or more resizable and collapsible panels. Refer the following guidelines to create a Splitter, add tree view in the Splitter and how to set actions to view the image that is used to split the document or image and also Expand or Collapse the Splitter.

The following screenshot illustrates a Splitter control.



Create a Splitter

Essential Studio ASP.NET MVC Splitter control has a built-in feature to split the display into horizontal and vertical order.

1. Create an MVC Project and add required assemblies, scripts, and styles to it. Refer MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page to render the Splitter. You can download the desired mobile and name it as galaxy.jpg. ‘ContentTemplate’ property is used to

Copyright © 2001 - 2020 Syncfusion Inc.

specify the HTML tags within the helper elements. Here we have used the contentTemplate property to specify the contents of the splitter control.

ASPX-CS

```
@{Html.EJ().Splitter("outerSplitter").PaneProperties(p =>
{
p.Add().ContentTemplate(
@<div class="cont">
<h3 class="h3">
ASP.NET MVC
</h3>
@*Add Tree Element here*@
</div>).PaneSize("200");
p.Add().ContentTemplate(
@<div class="cont">
<div class="_content">
Select any product from the tree to show the description.
</div>
<div class="tools des">
<h3>
Tools
</h3>
<p>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</p>
</div>
<div class="chart des">
<h3>
Chart
</h3>
<p> Essential Chart is a business-oriented charting component.</p>
</div>
<div class="grid des">
<h3>
Grid
</h3>
<p>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</p>
</div>
</div>).PaneSize("200");
}).Height("400").Width("100%").Render();}
```

3. Add the following style in the view page to set the height and width of the Splitter.

CSS

```
<style type="text/css" class="cssStyles">
#outerSplitter
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
margin: 0 auto;
}
.cont #treeView_Container
{
margin-bottom: 0;
border: none;
}
.h3, ._content, p
{
font-size: 14px;
margin-top: 10px;
text-indent: 10px;
}
.des
{
display: none;
}
</style>
```

Configure Tree View

Add the following code example in the corresponding view page. 'HtmlAttributes' property is used to add HTML attributes like, id, class etc.. to the components. We need to use IDictionary<string, object> to specify the HTML attributes. Please check the below code.

ASPX-CS

```
@{IDictionary<string, object> htmlAttribute = new Dictionary<string,
object>();
htmlAttribute.Add("class", "_child");
}
@Html.EJ().TreeView("treeview").Items(items =>
{
items.Add().Text("Layout").Expanded(true).Children(child =>
{
child.Add().Id("tools").Text("Tools").HtmlAttributes(htmlAttribute);
});
items.Add().Text("Data Visualization").Expanded(true).Children(child =>
{
child.Add().Id("chart").Text("Chart").HtmlAttributes(htmlAttribute);
});
items.Add().Text("Data").Expanded(true).Children(child =>
{
child.Add().Id("grid").Text("Grid").HtmlAttributes(htmlAttribute);
});
}).ClientSideEvents(e => e.NodeSelect("treeClicked"))
```

Set Actions to Splitter control to view the image

Add the following code example in the view page to set action in Splitter control to view the image.

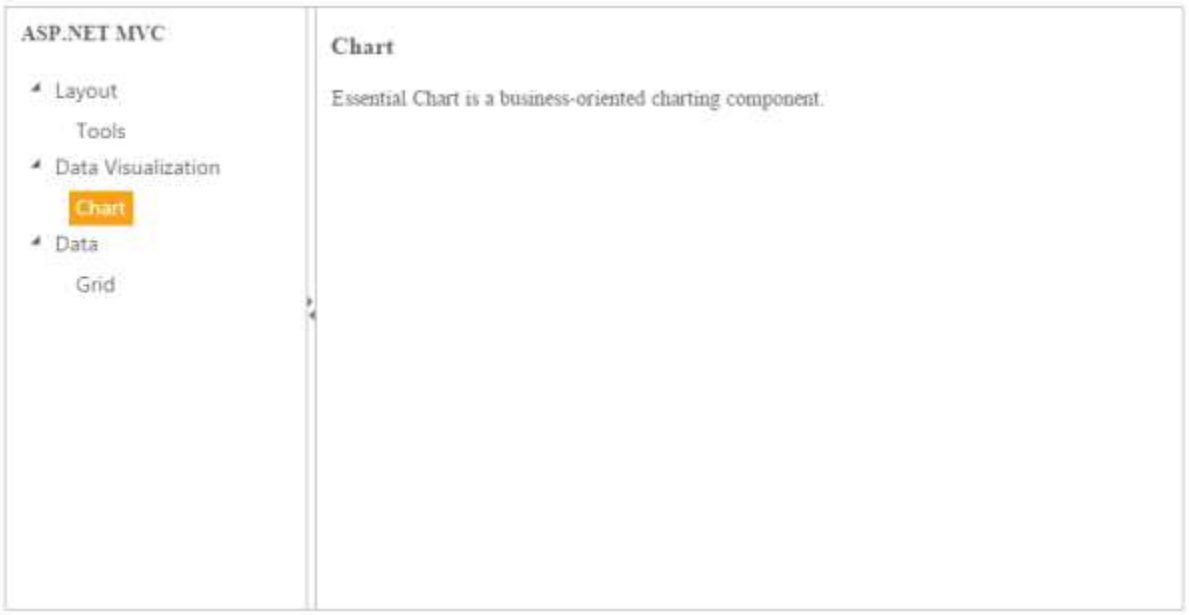
JS

```
<script type="text/javascript">
function treeClicked(sender, args)
{
if (sender.currentElement.hasClass('_child'))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
//nodeSelect event handle
var content = $('.' + sender.currentElement[0].id).html();
$('._content').html(content);
}
}
</script>
```

Execute the above code example to render the following output.



Enabling Collapsible

The Splitter provides you the option to enable or disable the pane collapse functionality. You can click the icon in Split bar to collapse or expand the corresponding pane element in Splitter. Setting the Collapsible property to “false” disables the pane collapse functionality in the Splitter widget.

Configure Collapsible

The following steps explain the implementation of the Collapsible option in Splitter.

1. In the View page, add the Splitter helper and configure the ‘Collapsible’ property as shown below. ‘ContentTemplate’ property is used to specify the HTML tags within the helper elements. Here we have used the contentTemplate property to specify the contents of the splitter control.

ASPX-CS

```
@Html.EJ().Splitter("Splitter").Height("280").Width("600").Orientation(Or
ientation.Horizontal).PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>).Collapsible(false);
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>);
})
```

The output for Splitter when collapsible is set to “true” is as follows.



The output for Splitter when collapsible is “false”.



Splitter Orientation

The Splitter supports both vertical and horizontal orientation of the pane. You can declare the orientation by using enum, ej.Orientation.Vertical or ej.Orientation.Horizontal, that have the corresponding value of vertical and horizontal as a string.

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Splitter Orientation

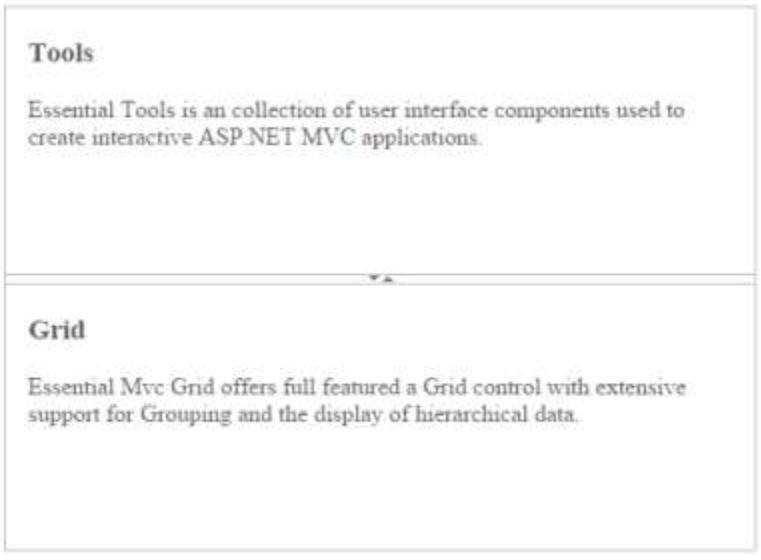
The following steps explain the implementation of Splitter orientation option.

1. In the View page, add the Splitter helper and configure the 'Orientation' property as shown below.

ASPX-CS

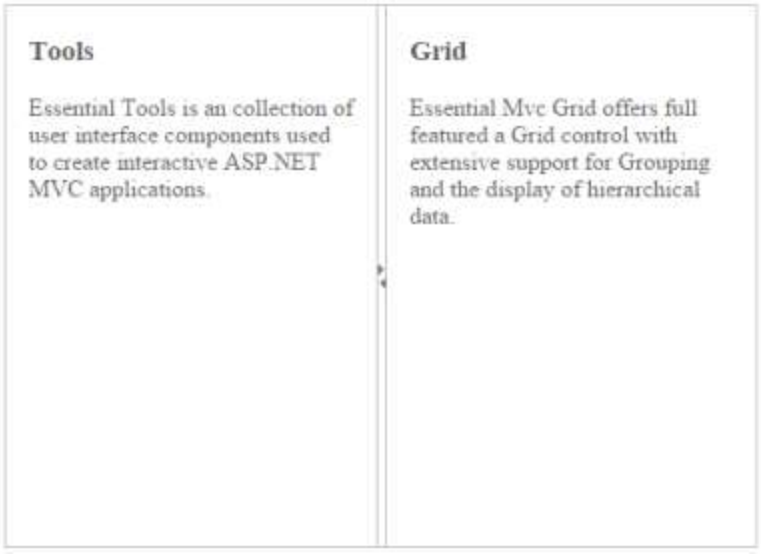
```
@{Html.EJ().Splitter("Splitter").Height("360").Width("500").Orientation(O
rientation.Vertical).PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>);
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>);
}).Render();}
```

The output for Splitter with ej.Orientation.Vertical.



The output for Splitter with ej.Orientation.Horizontal.

Copyright © 2001 - 2020 Syncfusion Inc.



Nested Splitter Support

The Splitter provides nested pane support that allows you to add a pane between two pane elements.

Configure Nested Splitter

The following steps explain the implementation of the “Nested Splitter” option.

1. In the View page, add the Splitter helper and configure the outer and inner splitters. As you can see we have specified a helper function to render the inner splitter. A helper is a reusable snippet of Razor syntax exposed as a method, and is intended for rendering HTML to the browse. To use helper within another helper we have specified as a separate function as shown below.

ASPX-CS

```
@{Html.EJ().Splitter("outerSplitter").Height("300").Width("600").Orientat
ion(Orientation.Vertical).PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div class="content" style="padding: 0px 15px;">
<h3 class="h3">
ASP.NET MVC
</h3>
</div>
</div>).PaneSize("60");
p.Add().ContentTemplate(
@<div style="height: 100%; width: 100%">
@innerSplitter()
</div>);
}).Render();}
@helper innerSplitter()
{
@Html.EJ().Splitter("innerSplitter").Width("600").PaneProperties(p1 =>
{
p1.Add().ContentTemplate(@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

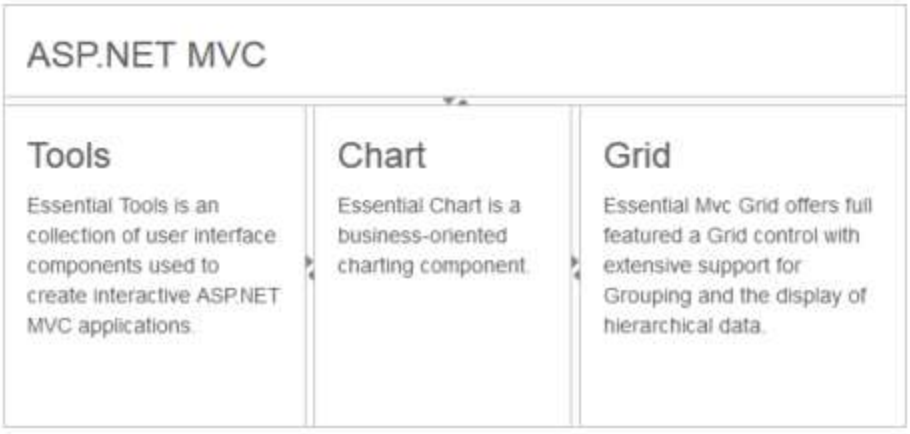
```
<div class="content">
<h3 class="h3">
Tools
</h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>).PaneSize("200");
p1.Add().ContentTemplate(@<div>
<div class="content">
<h3 class="h3">
Chart
</h3>
Essential Chart is a business-oriented charting component.
</div>
</div>).PaneSize("200");
p1.Add().ContentTemplate(@<div>
<div class="content">
<h3 class="h3">
Grid
</h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>).PaneSize("200");
})
}
```

CSS

```
<style type="text/css" >
#outerSplitter
{
margin: 0 auto;
}
.h3
{
font-size: 14px;
}
#innerSplitter
{
border: 0 none;
}
.content
{
padding: 15px;
}
</style>
```

The output for nested Splitter.

Copyright © 2001 - 2020 Syncfusion Inc.



Splitter Integration

The Splitter allows you to use other Essential ASP.NET MVC products inside the pane. The integrated function of those widgets can be used in other panes of the Splitter.

Configuring other widgets in Splitter

The following steps explain the implementation of Splitter integration.

1. In the View page, add the Splitter helper and configure the elements within the split pane. The first pane has the TreeView content and the next one has some content that is related to TreeView.

ASPX-CS

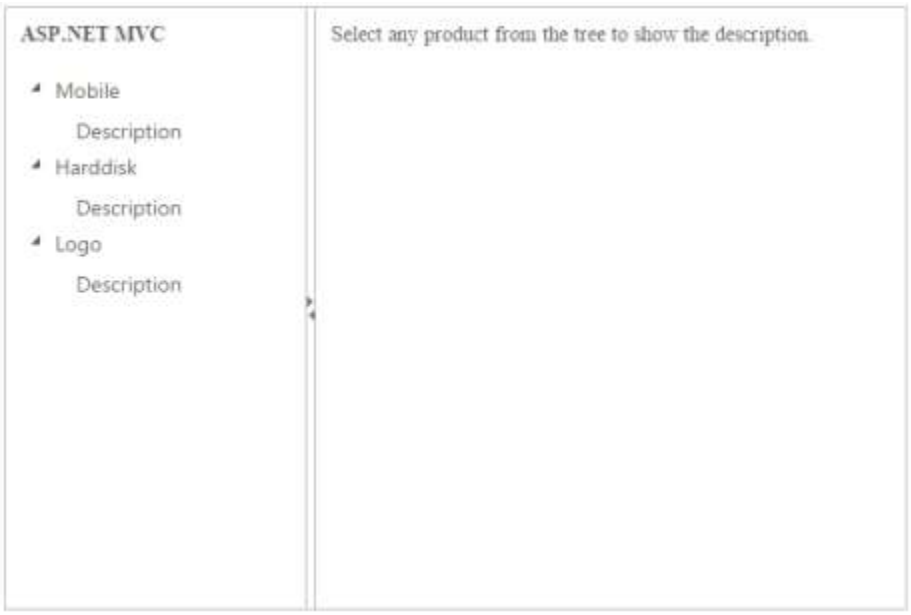
```
@{IDictionary<string, object> htmlAttribute = new Dictionary<string,
object>();
htmlAttribute.Add("class",
"_child");}@{Html.EJ().Splitter("outerSplitter").PaneProperties(p =>     {
p.Add().ContentTemplate(
@<div class="cont">
<h3 class="h3">
ASP.NET MVC
</h3>
@Html.EJ().TreeView("treeview").Items(items =>
{
items.Add().Text("Mobile").Expanded(true).Children(child =>
{
child.Add().Id("tools").Text("Galaxy").HtmlAttributes(htmlAttribute);
});
items.Add().Text("Hard Disk").Expanded(true).Children(child =>
{
child.Add().Id("chart").Text("Seagate").HtmlAttributes(htmlAttribute);
});
items.Add().Text("Logo").Expanded(true).Children(child =>
{
child.Add().Id("grid").Text("Amazon").HtmlAttributes(htmlAttribute);
});
}).ClientSideEvents(e=>e.NodeSelect("treeClicked"))
</div>).PaneSize("200");
p.Add().ContentTemplate(
@<div class="cont">
```

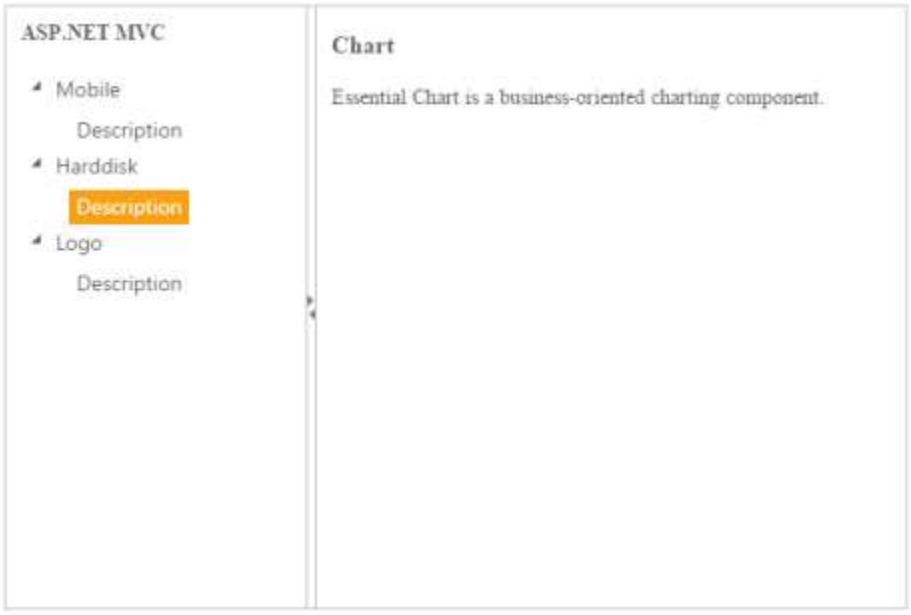
Copyright © 2001 - 2020 Syncfusion Inc.

```
<div class="_content">
Select any product from the tree to show the description.
</div>
<div class="tools des">
<h3>
Tools
</h3>
<p>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</p>
</div>
<div class="chart des">
<h3>
Chart
</h3>
<p> Essential Chart is a business-oriented charting component.</p>
</div>
<div class="grid des">
<h3>
Grid
</h3>
<p>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</p>
</div>
</div>).PaneSize("200");
}).Height("400").Width("100%").Render();}<style type="text/css">
#outerSplitter {
margin: 0 auto;
}    .cont #treeView_Container {
margin-bottom: 0;
border: none;
}
.h3, ._content, p {
font-size: 14px;
margin-top: 10px;
text-indent: 10px;
}
.des {
display: none;
}
</style>
<script type="text/javascript">
function treeClicked(sender, args)
{
if (sender.currentElement.hasClass('_child'))
{
//nodeSelect event handle
var content = $('.' + sender.currentElement[0].id).html();
$('._content').html(content);
}
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

When the node is selected in TreeView, the integrated output is displayed in the second pane.





Appearance and Styling

Responsive

The EnableAutoResize option allows the Splitter control to adapt its rendering based on the parent container where it is actually placed. When this option is set to true, the Splitter control adjusts its height and width based on the outer container that contains it, and also the sub-elements within it adjust to its height, width and position, appropriately.

Enabling Auto Resize

The following steps explain the implementation of AutoResize option in the Splitter control.

Copyright © 2001 - 2020 Syncfusion Inc.

1. In the View page, add the Splitter helper and configure the ‘EnableAutoResize’ property.

ASPX-CS

```
@{Html.EJ().Splitter("outerSplitter").Height("280")
.Width("100%").Orientation(Orientation.Vertical).EnableAutoResize(true)
.PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div class="content" style="padding: 0px 15px;">
<h3 class="h3">
ASP.NET MVC
</h3>
</div>
</div>).PaneSize("60");
p.Add().ContentTemplate(
@<div style="height: 100%; width: 100%">
@innerSplitter()
</div>);
}).Render();}
@helper innerSplitter()
{
@Html.EJ().Splitter("innerSplitter").EnableAutoResize(true).PanePropertie
s(p1=>
{
p1.Add().ContentTemplate(@<div>
<div class="content">
<h3 class="h3">
Tools
</h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>);
p1.Add().ContentTemplate(@<div>
<div class="content">
<h3 class="h3">
Grid
</h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>);
});
}
```

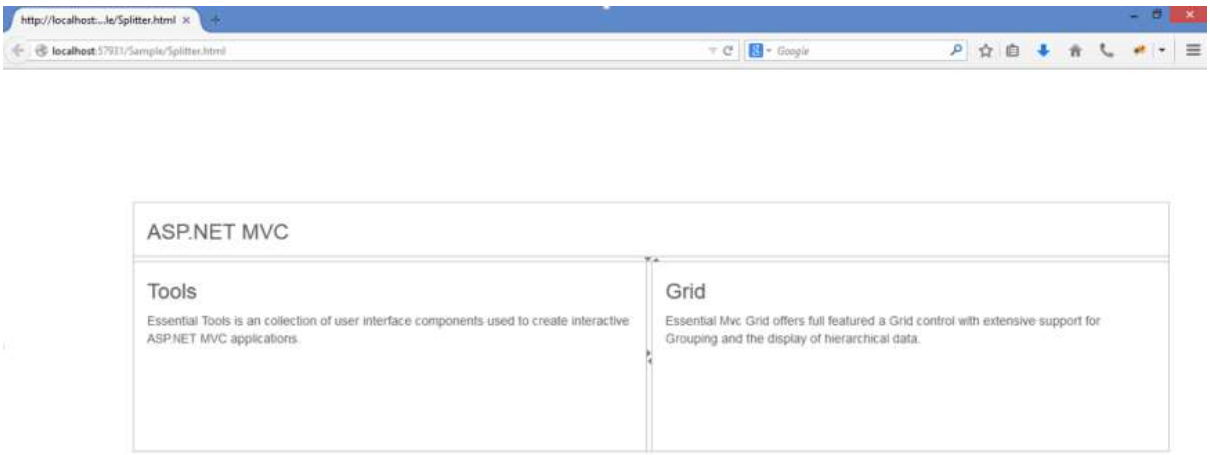
CSS

```
<style type="text/css">
#outerSplitter
{
margin: 0 auto;
```

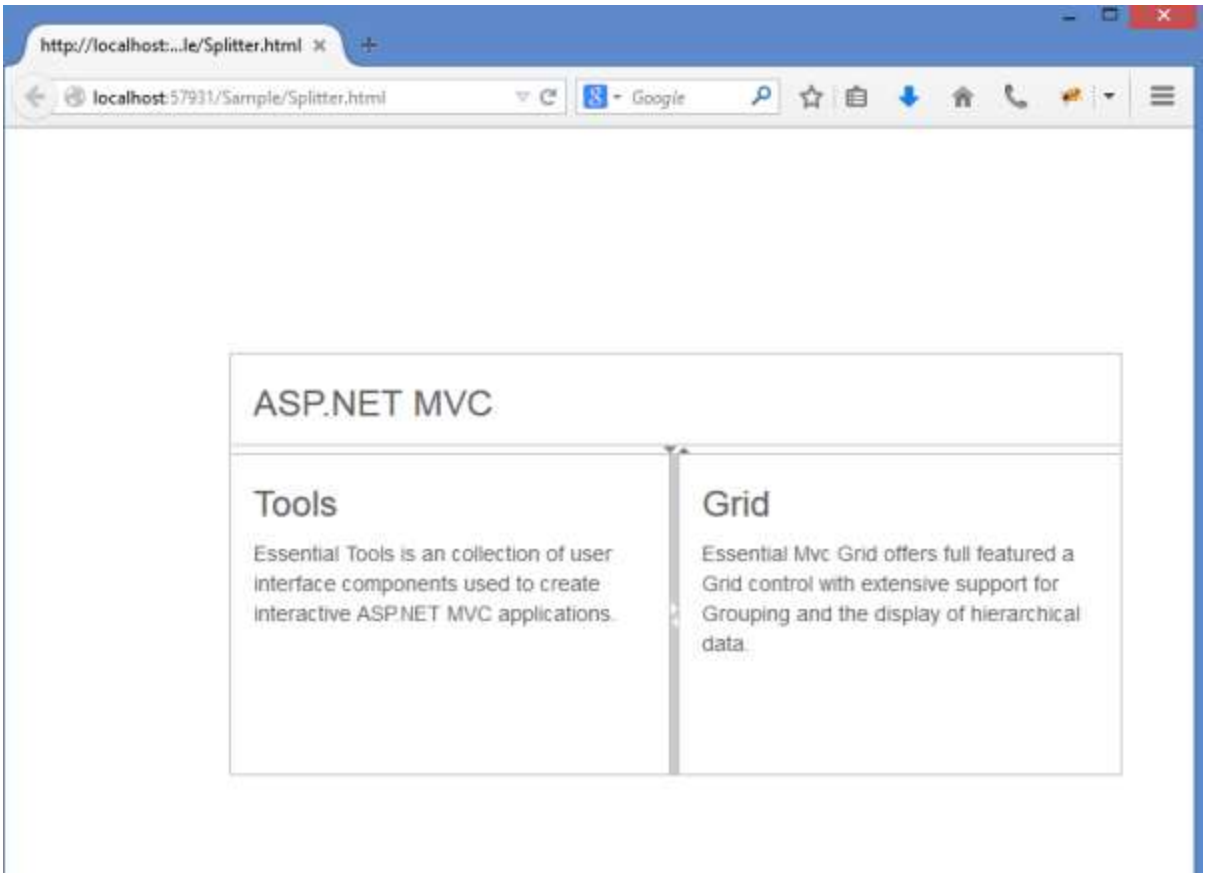
Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.h3
{
font-size: 14px;
}
#innerSplitter
{
border: 0 none;
}
.content
{
padding: 15px;
}
</style>
```

The output for Splitter when EnableAutoResize is “true”.



Copyright © 2001 - 2020 Syncfusion Inc.



Animation Support

The Splitter provides you animation support when you expand or collapse the pane. The animation speed can be modified by using the AnimationSpeed property, that has values in milliseconds.

Enabling Animation with Animation speed

The following steps explain the implementation of EnableAnimation option in the Splitter widget.

1. In the View page add the Splitter helper to customize the Splitter appearance.

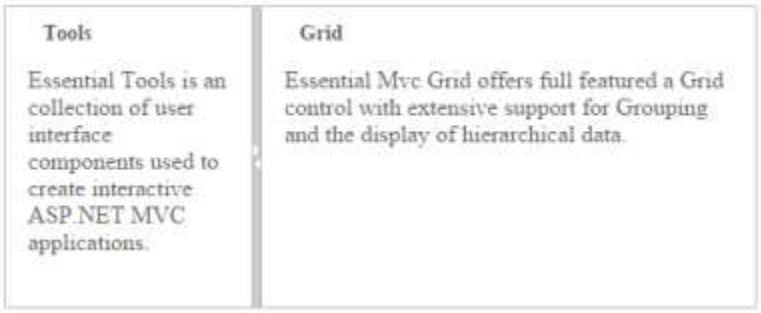
ASPX-CS

```
@{Html.EJ().Splitter("Splitter").Height("200").Width("500").EnableAnimati
on(true).AnimationSpeed(300).PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>);
p.Add().ContentTemplate(
@<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>);
}).Render();}
```

The output for Splitter when EnableAnimation is “true”. Expanding or collapsing the outer pane in the Splitter produces the animation effect with the animation speed.



Adjusting Splitter Size

Height

The height of Splitter can be modified by using the Height property. The default value for Height property is null in Splitter. You can set the Height property by pixel or percentage values.

Width

The width of Splitter can be modified by using the Width property. The default value for Width property is null in Splitter. You can set the Width property by pixel or percentage values.

Max Size

Defines the maximum resizable size of the pane when you resize the Splitter widget. The default value of MaxSize is null in Splitter. You can set the MaxSize property by pixel values.

Min Size

Defines the minimum resizable size of the pane when you resize the Splitter widget. The default value of MinSize is 10 in Splitter. You can set the MinSize property by pixel values.

Pane Size

Defines the pane size in the Splitter widget. The default value of PaneSize is 0px in Splitter. You can set the PaneSize property by pixel or percentage values.

Resizable

Defines whether the pane in the Splitter is resizable or not. Setting the resizable property as “false” disables the resize option to the pane. The default value of Resizable property is true in Splitter.

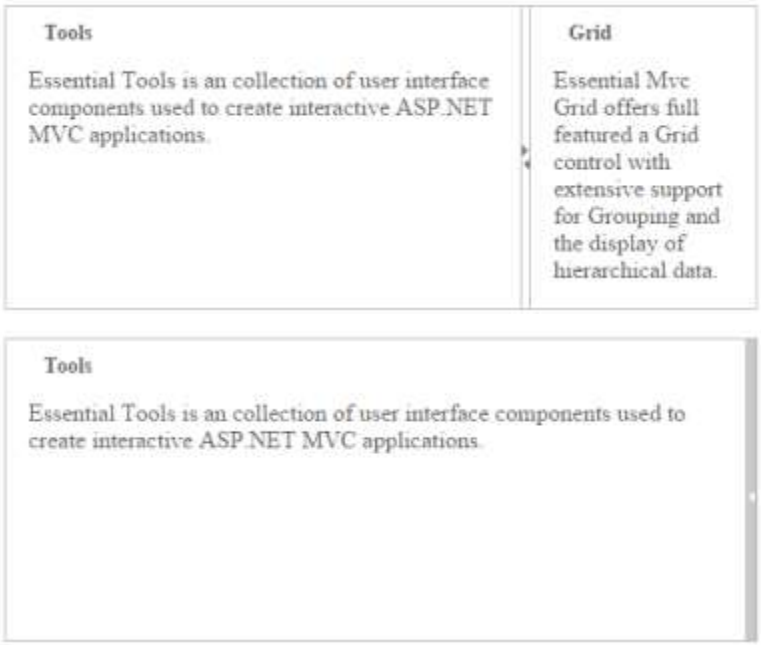
The following steps explain the implementation of Splitter properties.

1. In the View page add Splitter helper and configure resizable property of the split pane.

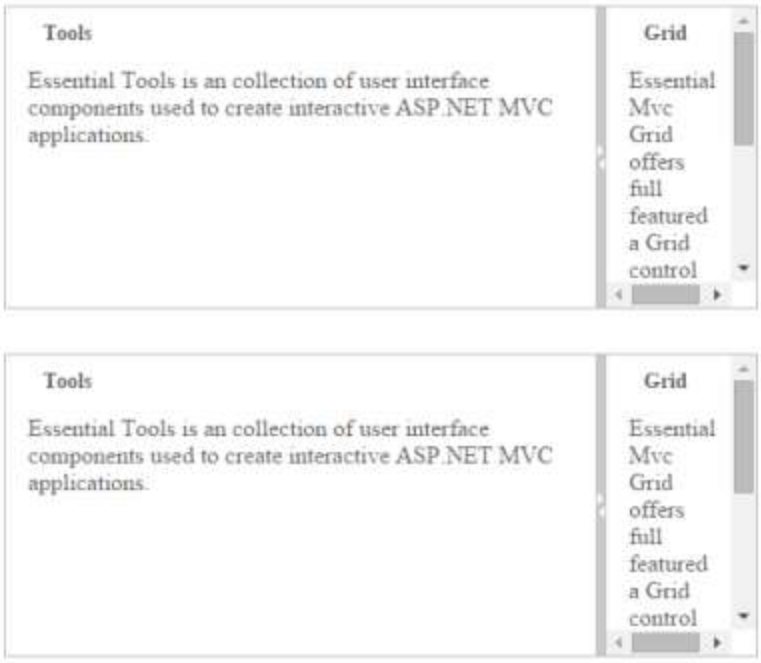
ASPX-CS

```
@{Html.EJ().Splitter("Splitter").Height("200").Width("500").PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to create interactive
ASP.NET MVC applications.
</div>
</div>);
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive support for
Grouping and the display of hierarchical data.
</div>
</div>).Collapsible(true).MinSize("100").MaxSize("150").PaneSize("150").Resizable(true);
}).Render();}
```

The output for Splitter after adding the properties.



Copyright © 2001 - 2020 Syncfusion Inc.



Theme

Splitter control's style and appearance can be controlled based on CSS classes. In order to apply styles to the Splitter control, refer 2 files namely: ej.widgets.core.min.css and ej.theme.min.css. If the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Autocomplete control namely

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

CSS Class

The CSS properties can be customized by using CssClass in the Splitter widget. The following steps explain the implementation of CssClass option in the Splitter widget.

In the View page add the Splitter helper and customize the Splitter appearance.

ASPX-CS

```
@{Html.EJ().Splitter("Splitter").Height("200").Width("500").CssClass("cus
tomCSS").PaneProperties(
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
p =>
{
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>);
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>).Collapsible(true).MinSize("100").Resizable(true);
}).Render();}
```

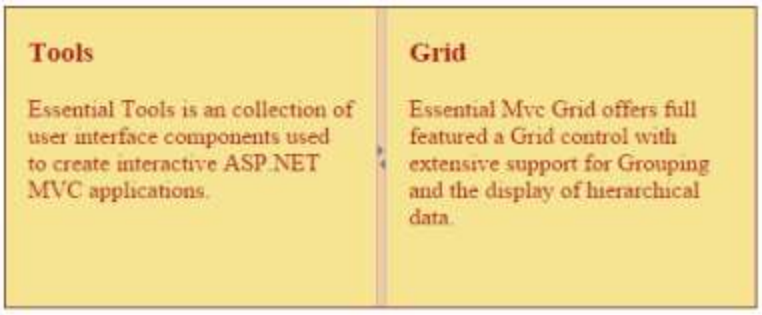
1. Customize the CSS class by setting CSS properties.

CSS

```
<style>
.customCSS
{
border-color: #661e19;
}
/*Customize Splitbar*/
.customCSS .e-splitbar
{
background-color: #f9c89f;
}
/*Customize Splitter pane*/
.customCSS .e-pane
{
color: #b21010;
background-color: #f6e492;
}
</style>
```

The output for Splitter after customizing the CSS class.

Copyright © 2001 - 2020 Syncfusion Inc.



Keyboard Navigation

With the keyboard navigation enabled in the Splitter control, it is possible to control the actions of the Splitter with the provided shortcut keys. Almost all the Splitter actions that are done by mouse can be controlled with shortcut keys.

The various keyboard shortcuts available within the Splitter control are discussed in the following table.

Keyboard Shortcuts

Shortcut Key	Description
Left	Moves the Split bar left.
Right	Moves the Split bar right.
Ctrl + Left	Collapses the left pane.
Ctrl + Right	Collapses the right pane.
Up	Moves the Split bar up.
Down	Moves the Split bar down.
Ctrl + Up	Collapses the top pane.
Ctrl + Down	Collapses the bottom pane.
Enter	Resize the pane to the current Split bar position.
Esc	Focuses out from the Split bar.

Configuring Keyboard Navigation

The following steps explain to enable keyboard interaction for Splitter widget.

1. In the View page add Splitter helper to render the control.

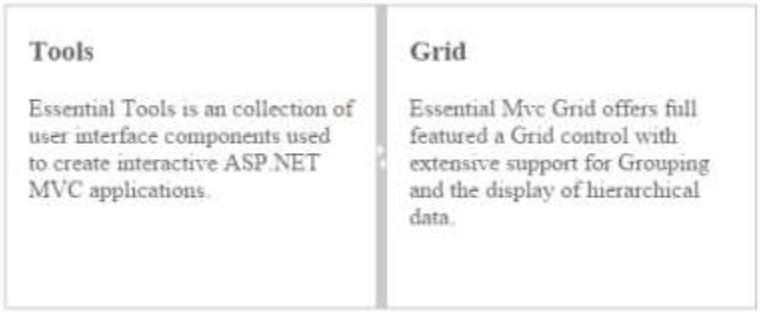
ASPX-CS

```
@{Html.EJ().Splitter("Splitter").Height("200").Width("500").PaneProperties(
p =>    {        p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Tools </h3>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>);
p.Add().ContentTemplate(
@<div>
<div style="padding: 0px 15px;">
<h3 class="h3">Grid </h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>);
}).Render();}
<script type="text/javascript">
//Control focus key
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) { // j- key code.
$("#Splitter .e-splitbar")[0].focus();
}
});
</script>
```

1. Run the sample and press Alt + J to focus the Splitter widget. We can fully utilize the functionality of the splitter control by using keyboard shortcuts.



RTL Support

The Splitter provides you with RTL (Right-To-Left) support. The alignment of Splitter can be changed from Left-To-Right to Right-To-Left.

Enable RTL

The following steps explain enabling the right-to-left property for Splitter widget.

In the View page add the Splitter helper to enable the RTL functionality.

ASPX-CS

```
@{Html.EJ().Splitter("outerSplitter").Height("300").Width("600").Orientat
ion(Orientation.Vertical).EnableRTL(true)
.PaneProperties(
p =>
{
p.Add().ContentTemplate(
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@<div>
<div class="content" style="padding: 0px 15px;">
<h3 class="h3">
ASP.NET MVC
</h3>
</div>
</div>).PaneSize("60");
p.Add().ContentTemplate(
@<div style="height: 100%; width: 100%">
@innerSplitter()
</div>);
}).Render();}
@helper innerSplitter()
{
@Html.EJ().Splitter("innerSplitter").Width("600").PaneProperties(p1 =>
{
p1.Add().ContentTemplate
(@<div>
<div class="content">
<h3 class="h3">
Tools
</h3>
Essential Tools is an collection of user interface components used to
create interactive
ASP.NET MVC applications.
</div>
</div>).PaneSize("200");
p1.Add().ContentTemplate
(@<div>
<div class="content">
<h3 class="h3">
Chart
</h3>
Essential Chart is a business-oriented charting component.
</div>
</div>).PaneSize("200");
p1.Add().ContentTemplate
(@<div>
<div class="content">
<h3 class="h3">
Grid
</h3>
Essential MVC Grid offers full featured a Grid control with extensive
support for
Grouping and the display of hierarchical data.
</div>
</div>).PaneSize("200");
})
}
```

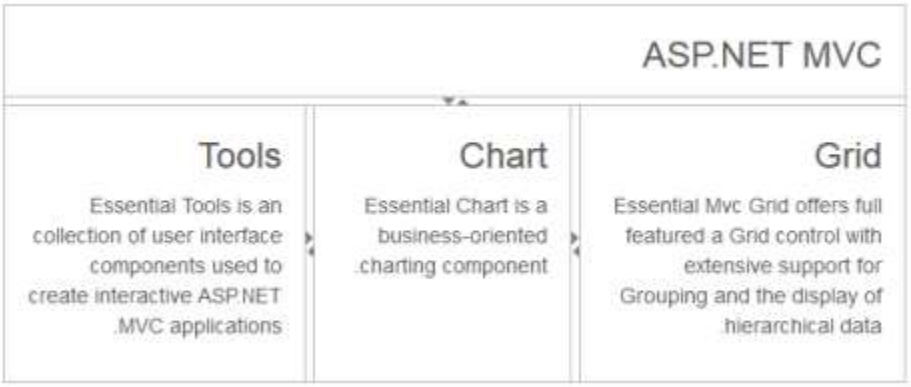
CSS

```
<style type="text/css">
#outerSplitter
{
margin: 0 auto;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.h3
{
font-size: 14px;
}
#innerSplitter
{
border: 0 none;
}
.content
{
padding: 15px;
}
</style>
```

The output for Splitter when EnableRTL is “true”.



How To?

Change Expand/Collapse icons

By default, you are provided with collapse/expand icons in **Split bar** to collapse or expand the splitter panes. We have provided template support to replace existing expand/collapse icons.

- **expanderTemplate** Specifies HTML element string to replace template with existing expand/collapse icons.
- **clickOnExpander** event is triggered when we click on the template icon.`

ASPX-CS

```
<div id="outterSplitter">
@Html.EJ().Splitter("innerSplitter").PaneProperties(p1 =>
{
p1.Add().ContentTemplate(@<div>
<div class="cont">
Pane 1
</div>
</div>);
p1.Add().ContentTemplate(@<div>
<div class="cont">
Pane 2
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
</div>);
}).Height("250").Width("80%").ExpanderTemplate("<img class='eimg'
src='../Content/basketball.png' alt='employee'/>").ClientSideEvents(cs =>
cs.ClickOnExpander("onClick"))
</div>
```

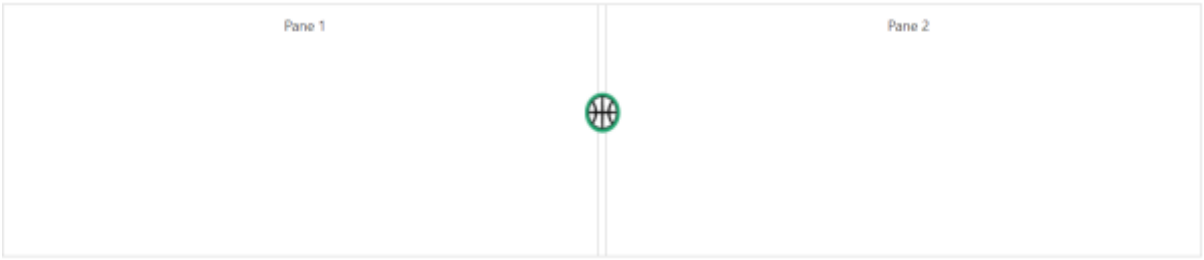
JS

```
<script type="text/javascript">
var flag = true;
function onClick(args) {
if (flag) { this.collapse(0); flag = false; }
else { this.expand(0); flag = true; }
}
</script>
```

CSS

```
.cont {
padding: 10px 0 0 10px;
text-align: center;
}
.eimg {
height:40px;
width:35px;
margin-left: -13px;
}
.e-splitter .e-splitbar .e-splitter-h-template {
top: 15%;
}
```

The output for Splitter with **Template support**.



Copyright © 2001 - 2020 Syncfusion Inc.

Change the splitter pane size dynamically

Splitter pane size can be changed by updating model property of paneSize and by refreshing the control using refresh public method. As shown in the below code, based on the selected dropdown list value, pane size of splitter is changed.

RAZOR

```
@Html.EJ().DropDownList("customersList").Datasource((IEnumerable<Customer
s>)ViewBag.datasource).DropDownListFields(df =>
df.ID("id").Text("text").Value("text")).Value("50").ClientSideEvents(e=>e
.Change("change"))
@Html.EJ().Splitter("Splitter").PaneProperties(
p =>
{
p.Add().ContentTemplate(
@<div>
<div class="content">
<h3 class="h3">
Pane 1
</h3>
</div>
</div>);
p.Add().ContentTemplate(
@<div>
<h3 class="h3">Pane 2</h3>
</div>);
}).Height("350")
```

C#

```
List<Customers> customer = new List<Customers>();
public ActionResult SplitterFeatures()
{
customer.Add(new Customers { id = "1", text = "25" });
customer.Add(new Customers { id = "2", text = "50" });
customer.Add(new Customers { id = "3", text = "75" });
ViewBag.datasource = customer;
return View();
}
public class Customers
{
public string id { get; set; }
public string text { get; set; }
}
```

CSS

```
<style type="text/css" class="cssStyles">
.cont
{
padding: 10px 0 0 10px;
text-align: center;
}
#inner
{
border:0 none;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
#Splitter
{
margin: 0 auto;
}
</style>
```

JAVASCRIPT

```
<script>
function change(args) {
var obj = $("#Splitter").data("ejSplitter");
obj.model.properties[0].paneSize = args.value + "%";
obj.refresh();
}
</script>
```



Render Splitter from Code behind

Splitter can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side

The following code illustrates the initialization of Splitter properties in the controller.

C#

```
//Namespace to get the JavaScript (Splitter) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class SplitterController: Controller
{
public ActionResult SplitterFeatures()
{
//Initializing the splitter model
SplitterProperties splitterObj = new SplitterProperties();
//Initializing the split panes and other properties
splitterObj.PaneProperties.Add(new PaneProperties() { Collapsible = true,
Expandable = true, ContentTemplate = new
Syncfusion.JavaScript.MvcTemplate<PaneProperties> { RazorViewTemplate =
(data) => { return "Pane 1"; } } });
splitterObj.PaneProperties.Add(new PaneProperties() { Collapsible = true,
Expandable = true, ContentTemplate = new
Syncfusion.JavaScript.MvcTemplate<PaneProperties> { RazorViewTemplate =
(data) => { return "Pane 2"; } } });
```

```
splitterObj.Orientation = Syncfusion.JavaScript.Orientation.Vertical;
splitterObj.Height = "300px";
splitterObj.Width = "500px";
//Passing Splitter properties using the ViewData
ViewData["SplitterModel"] = splitterObj;
return View();
}
}
}
```

Binding the Splitter properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Splitter("Splitter",
(Syncfusion.JavaScript.Models.SplitterProperties)ViewData["SplitterModel"
]).Render();
}
```

Spreadsheet

Overview

The Spreadsheet control is a Microsoft Excel-like Spreadsheet component for web. It provides editing experience that is very similar to that of excel and it is able to import and export Excel workbook files. The Spreadsheet control includes all the important features of Microsoft Excel like editing, sorting, filtering, formulas, data validation, formatting, table, charts, import and export.

Key Features

The key features of Spreadsheet control for ASP.NET MVC are:

- **Editing** - Provides support to edit the cell content.
- **Sorting** - Helps you to visualize and figure out your data better, organize and find the data quickly that you want.
- **Filtering** - Offers Excel-like filtering to filter data.
- **Formulas** - Allows you to handle calculations in cells.
- **Table** - Allows you to manage and analyze a group of related data.
- **Formatting** - Provides various formatting support like cell formatting, number formatting, text formatting.
- **Data Validation** - Helps you to restrict the users to enter the valid data in a range.
- **Conditional Formatting** - Allows you to highlight range of cells with a certain colors, depending on the provided conditions.
- **Chart** – Allows you to have graphical representation of data.
- **Fill Series** - Allows you to drag fill data based on adjacent cell data.
- **Printing** – Allows you to print whole sheet / selected range of data.
- **Import and Export** - Allows you to import excel files to view and export Spreadsheet as excel files.

Copyright © 2001 - 2020 Syncfusion Inc.

Getting Started

This section explains briefly about how to create Spreadsheet in your ASP.NET MVC application and also how to populate the Spreadsheet with data, formats and how to export Spreadsheet data as excel file.

Initialize Spreadsheet

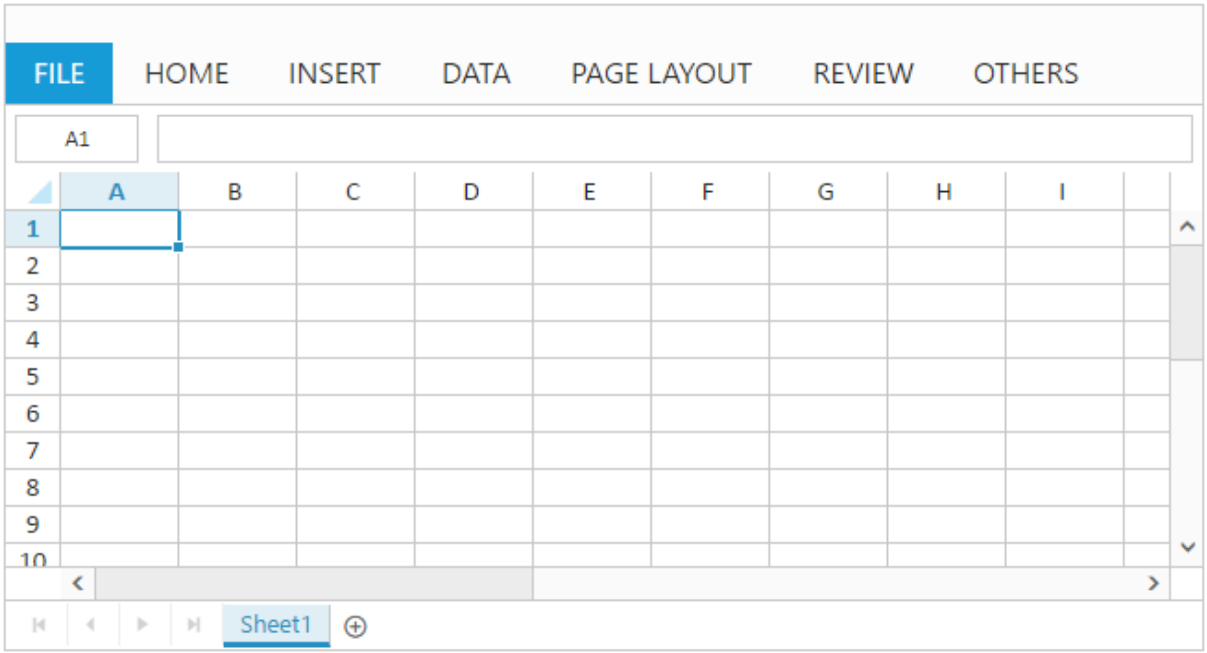
The following steps explains how to create spreadsheet,

1. Create Syncfusion ASP.NET MVC application. You can refer MVC Getting started documentation to create new project and add necessary dll's and script files.
2. Add the following code example in view page to render Spreadsheet control. Spreadsheet is rendered with default height and width. You can also customize Spreadsheet dimension by setting Height and Width property in ScrollSettings.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet"))
```

Now, the Spreadsheet is rendered with default row and column count.



Populate Spreadsheet with Data

Now, this section explains how to populate JSON data to the Spreadsheet. You can set DataSource property in Sheets settings to populate JSON data in Spreadsheet.

1) Create datasource for Spreadsheet control.

CSHARP

```
public ActionResult Index()
{
var DataSource = GetAllItemDetails().ToList();
ViewBag.Datasource = DataSource;
return View();
```

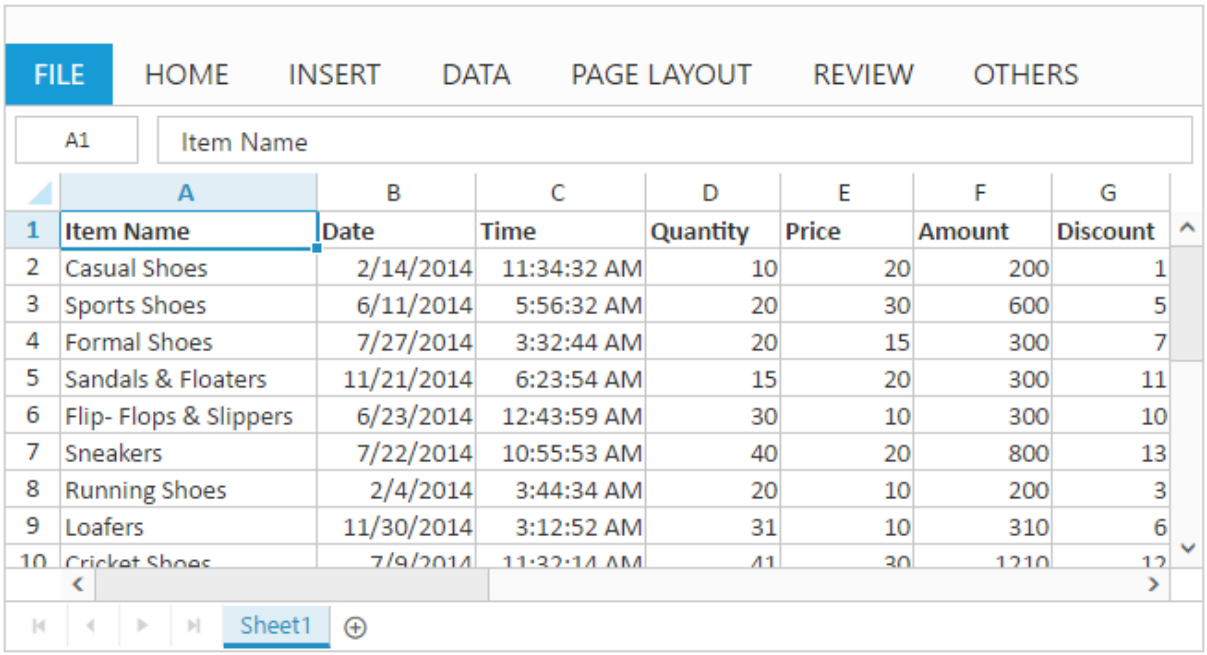
```
}
public IEnumerable<object> GetAllItemDetails()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
lItems.Add(new ItemDetail() { ItemName = "Sneakers", Date = new
DateTime(2014, 07, 22), Time = new DateTime(2014, 07, 22, 10, 55, 53),
Quantity = 40, Price = 20, Amount = 800, Discount = 13, Profit = 66 });
lItems.Add(new ItemDetail() { ItemName = "Running Shoes", Date = new
DateTime(2014, 02, 04), Time = new DateTime(2014, 02, 04, 03, 44, 34),
Quantity = 20, Price = 10, Amount = 200, Discount = 3, Profit = 14 });
lItems.Add(new ItemDetail() { ItemName = "Loafers", Date = new
DateTime(2014, 11, 30), Time = new DateTime(2014, 11, 30, 03, 12, 52),
Quantity = 31, Price = 10, Amount = 310, Discount = 6, Profit = 29 });
lItems.Add(new ItemDetail() { ItemName = "Cricket Shoes", Date = new
DateTime(2014, 07, 09), Time = new DateTime(2014, 07, 09, 11, 32, 14),
Quantity = 41, Price = 30, Amount = 1210, Discount = 12, Profit = 166 });
lItems.Add(new ItemDetail() { ItemName = "T-Shirts", Date = new
DateTime(2014, 10, 31), Time = new DateTime(2014, 10, 31, 12, 01, 44),
Quantity = 50, Price = 10, Amount = 500, Discount = 9, Profit = 55 });
return lItems;
}
public class ItemDetail
{
public string ItemName { get; set; }
public DateTime Date { get; set; }
public DateTime Time { get; set; }
public int Quantity { get; set; }
public int Price { get; set; }
public int Amount { get; set; }
public int Discount { get; set; }
public int Profit { get; set; }
}
```

2) Initialize Spreadsheet with datasource.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
```

```
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
)
```



	A	B	C	D	E	F	G
1	Item Name	Date	Time	Quantity	Price	Amount	Discount
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12

Apply Conditional formatting

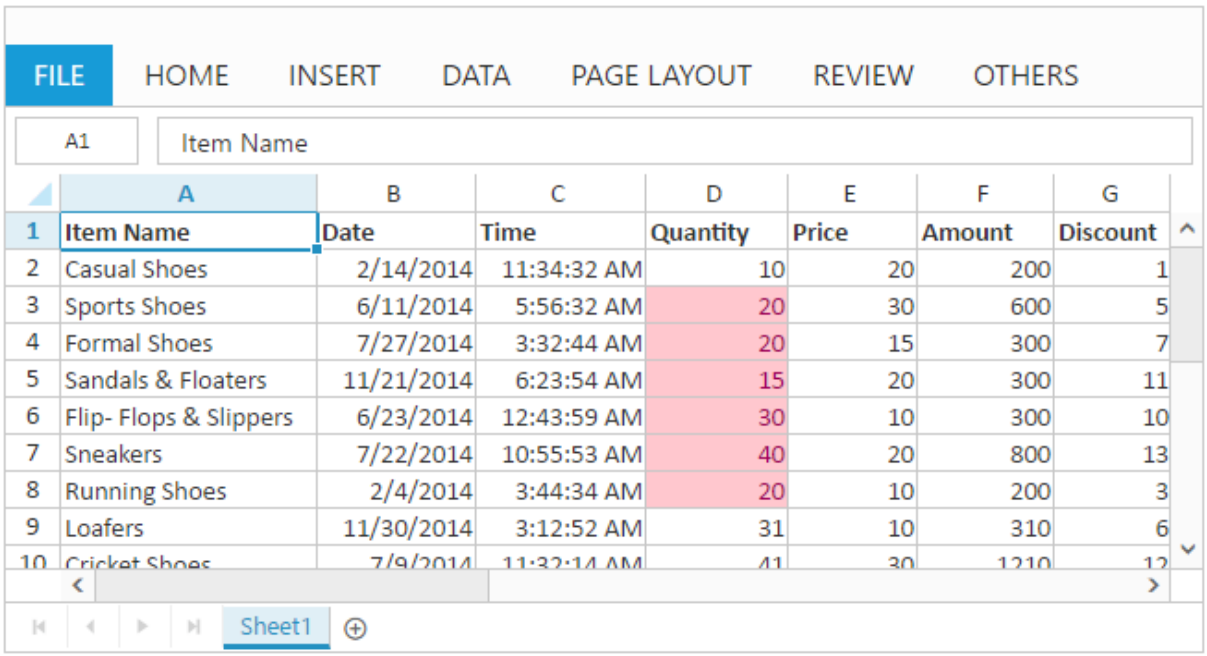
Conditional formatting helps you to apply formats to a cell or range with certain color based on the cells values. You can use AllowConditionalFormats property to enable/disable Conditional formats.

To apply conditional formats for a range use CFormatRule property.The following code example illustrates this behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).CFormatRule(formatRule =>
{
formatRule.Action(SpreadsheetCFormatRuleAction.GreaterThan).Inputs(new
List<string>() { "10"
}).Color(SpreadsheetCFormatHighlightColor.RedFill).Range("D2:D8").Add();
}).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G
1	Item Name	Date	Time	Quantity	Price	Amount	Discount
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12

Export Spreadsheet as Excel File

The Spreadsheet can save its data, style, format into an excel file. To enable save option in Spreadsheet set AllowExporting option in ExportSettings as true. Since Spreadsheet uses server side helper to save documents set ExcelUrl in ExportSettings option. The following code example illustrates this behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
//...
.ExportSettings(export =>
{
export.ExcelUrl("ExportToExcel");
}
)
//...
)
```

CSHARP

```
[AcceptVerbs(HttpVerbs.Post)]
//...
public void ExportToExcel(string sheetModel, string sheetData, string
password)
{
if (String.IsNullOrEmpty(password))
Spreadsheet.Save(sheetModel, sheetData, "sample", ExportFormat.XLSX,
ExcelVersion.Excel2013);
else
Spreadsheet.Save(sheetModel, sheetData, "sample", ExportFormat.XLSX,
ExcelVersion.Excel2013, password);
}
//...
```

Copyright © 2001 - 2020 Syncfusion Inc.

Use shortcut Ctrl + S to save Spreadsheet as excel file.

Note: 1. For more details about Export refer following link

2. For more details about Client dependencies refer the following link

3. For more details about Server dependencies refer the following link

Worksheet

Worksheet is a collection of cells organized in the form of rows and columns that allow us to store, format, manipulate and display data in grid format. You can create multiple sheets in Spreadsheet and use sheet tab for switching between those worksheets. By default, Spreadsheet creates single worksheet since default SheetCount value is 1.

List of Sheet Operations

You can perform following operations in worksheet,

- Add
- Remove
- Rename
- Move and Copy

Add

The Spreadsheet has support for inserting new sheet. You can insert sheet into two ways.

- Add a new sheet as last sheet.
- Insert a new sheet before the active sheet.

Add Sheet

You can dynamically add a sheet by one of the following ways,

- Click the new sheet button in the Spreadsheet sheet tab.
- Using addNewSheet method.

Insert Sheet

You can dynamically insert a sheet by one of the following ways,

- Right clicking on the worksheet in the sheet tab and then click Insert option in the context menu.
- Click OTHERS tab in the ribbon and select Insert Sheet option in Insert dropdown button.
- Using insertSheet method.

The following code example describes the above behavior

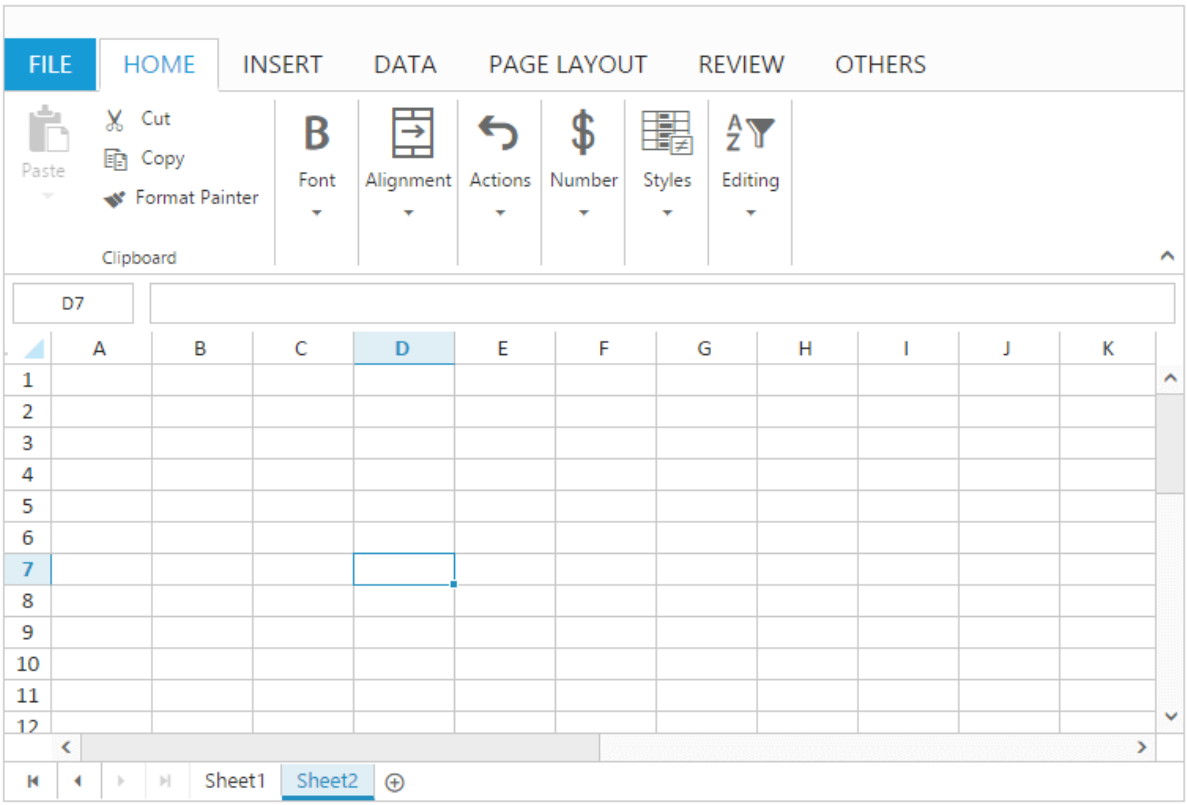
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (!this.isImport) {
this.addNewSheet(); //To add as a last sheet.
//this.insertSheet(); // To insert a sheet before the active sheet.
}
}
</script>
```

The following output is displayed as a result of the above code example.



Copy

The Spreadsheet provides support to create a copy of an existing worksheet. You can dynamically copy a worksheet by using one of the following ways,

- Right clicking on the worksheet in the sheet tab and then click Move or Copy in the context menu. Check the “Create a copy” checkbox in the “Move or Copy” dialog.
- Copy an existing worksheet using copySheet method.

The following code example describes the above behavior

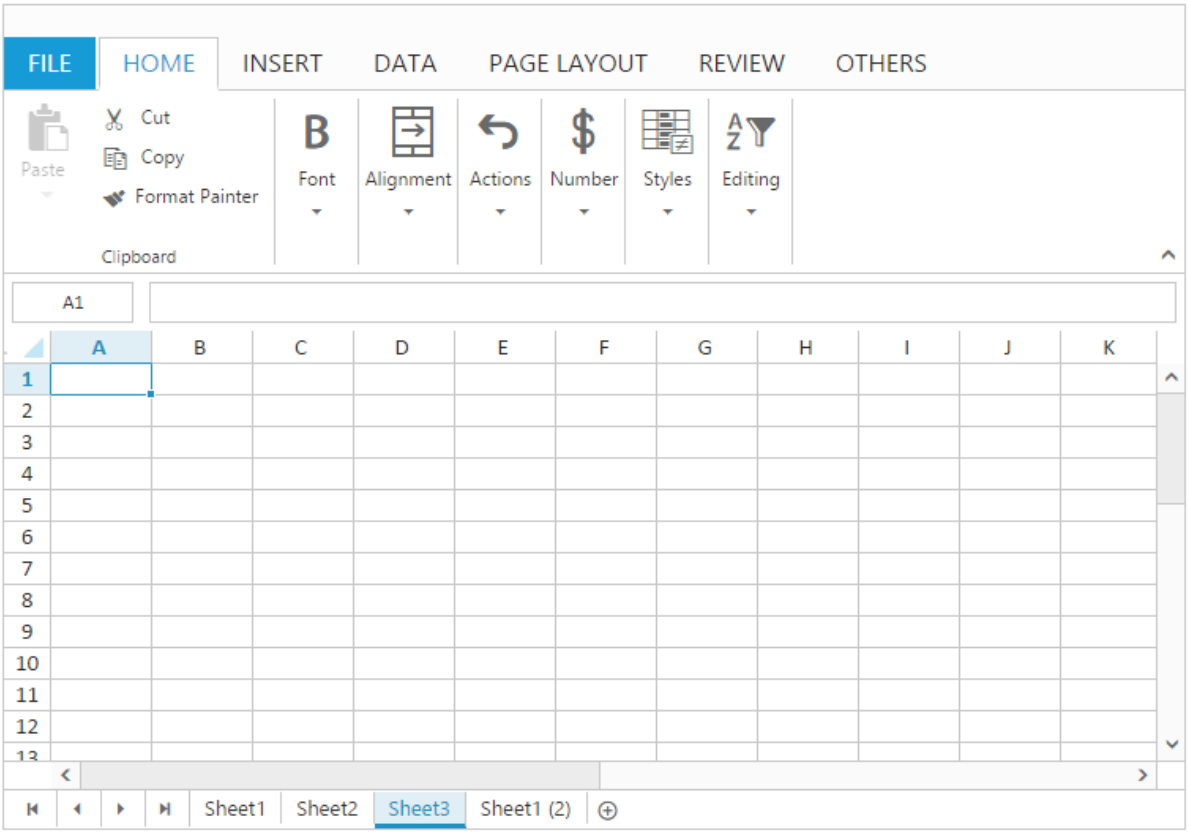
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.SheetCount(3)
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (!this.isImport) {
this.copySheet(1, 3, true); //arg1- from index, arg2 -to index, arg3 -
isCopySheet
}
}
</script>
```

The following output is displayed as a result of the above code example.



Move

The Spreadsheet provides support to move an existing worksheet. You can dynamically move a worksheet by using one of the following ways,

- Right clicking on the worksheet in the sheet tab and then click Move or Copy in the context menu. Select the sheet that you have to move in the “Move or Copy” dialog.
- Move an existing worksheet using copySheet method.

The following code example describes the above behavior

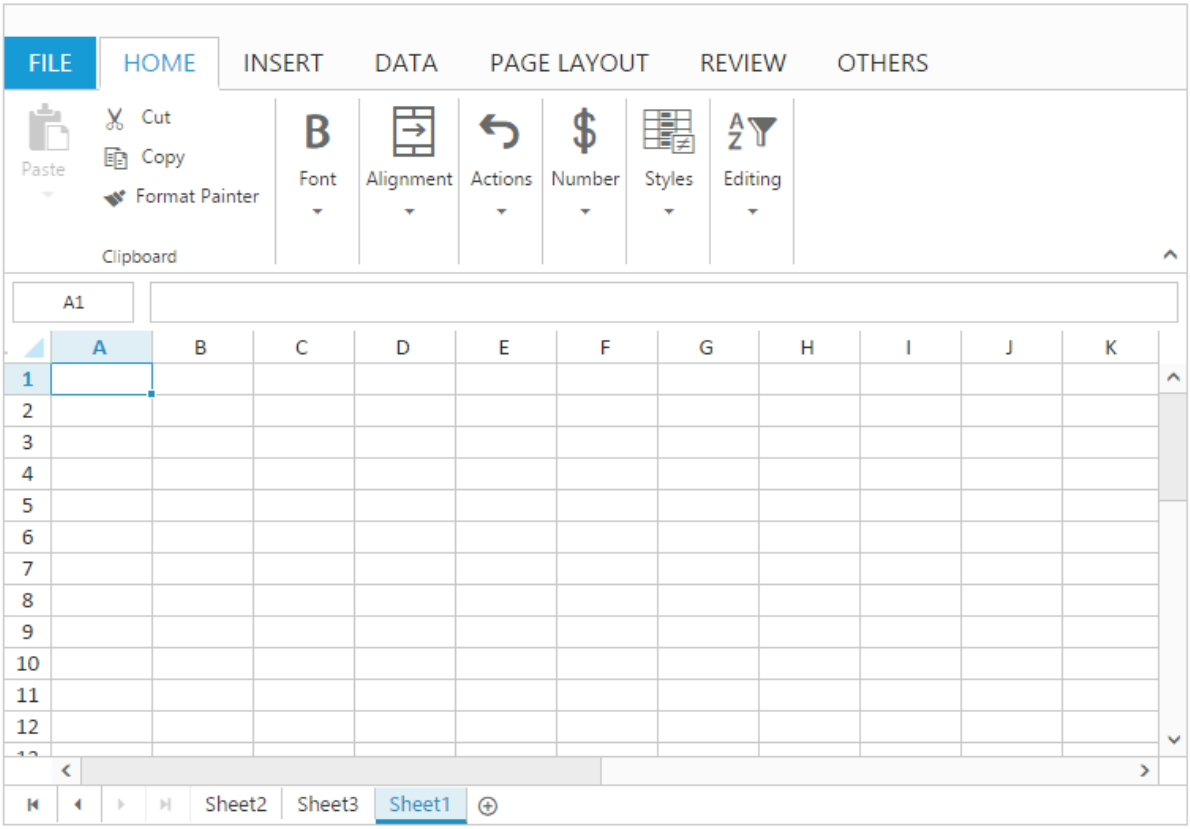
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.SheetCount(3)
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (!this.isImport) {
this.copySheet(1, 3, false); //arg1- from index, arg2 -to index, arg3 -
isCopySheet
}
}
</script>
```

The following output is displayed as a result of the above code example.



Remove

The Spreadsheet has support for removing an existing worksheet. You can dynamically remove the existing sheet by following ways,

- Right click on the worksheet in the sheet tab and then click Delete option in the context menu.
- Select the existing worksheet, click OTHERS tab in the ribbon and select Delete dropdown button. Then click Delete Sheet.

You can also remove an active worksheet using deleteSheet method.

The following code example describes the above behavior

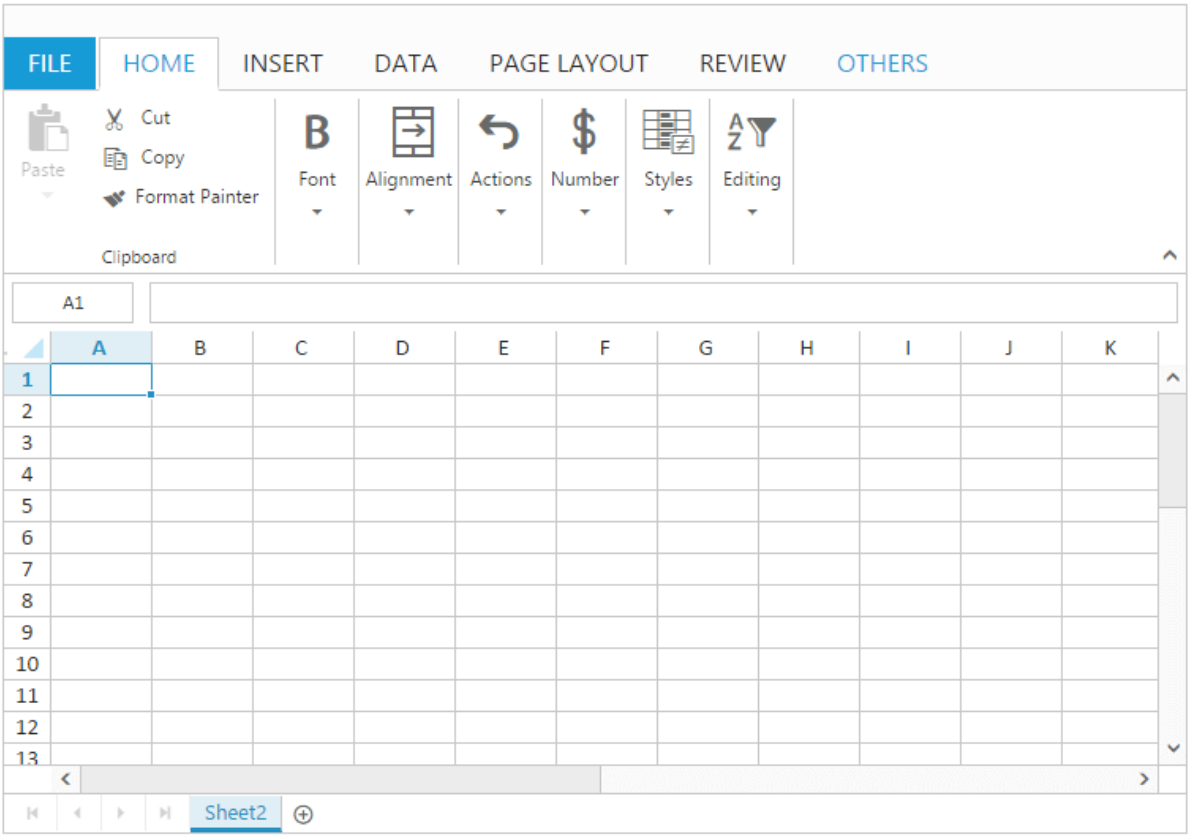
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.SheetCount(2)
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.deleteSheet();
}
}
</script>
```

The following output is displayed as a result of the above code example.



Rename

The Spreadsheet has support for renaming an existing worksheet. You can dynamically rename worksheet by using one of the following ways,

- Right clicking on the worksheet in the sheet tab and then click Rename option in the context menu.
- Rename an active worksheet using sheetRename method.

The following code example describes the above behavior

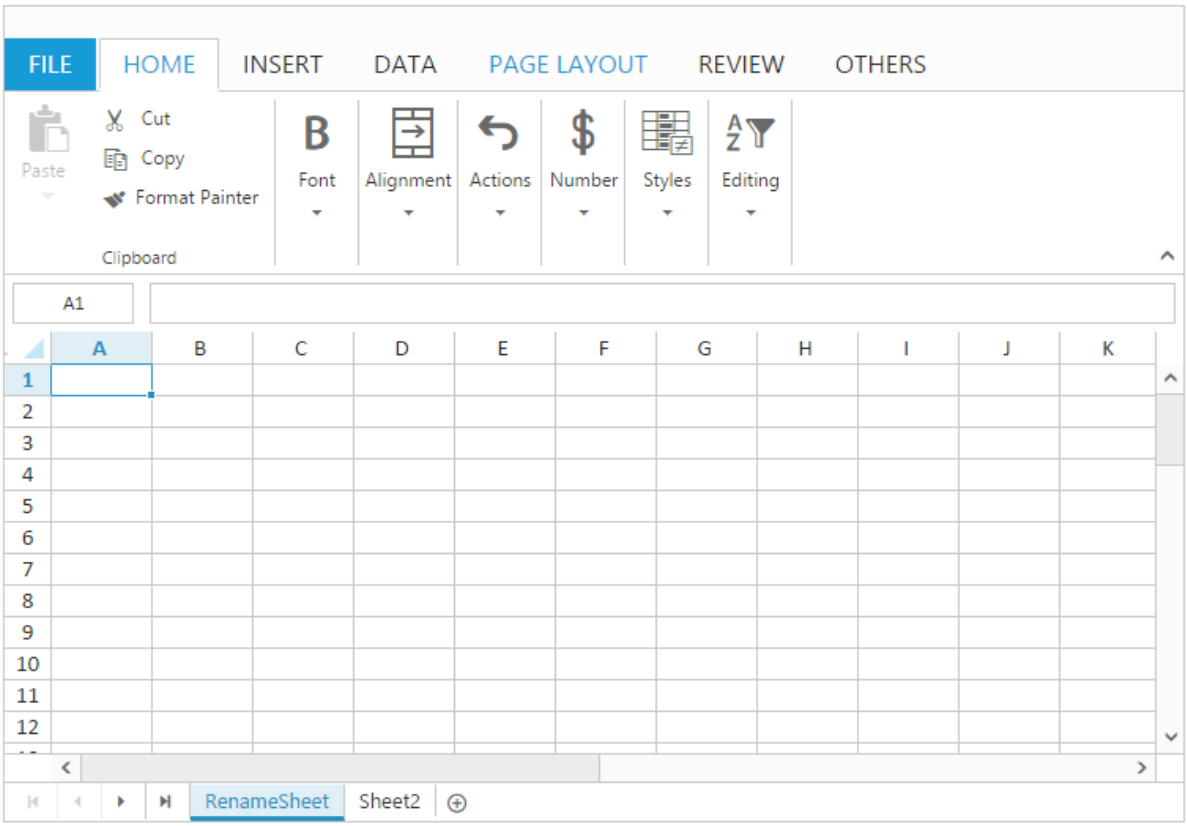
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.SheetCount(2)
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.sheetRename("RenameSheet")
}
}
</script>
```

The following output is displayed as a result of the above code example.



Headers

Headers in the Spreadsheet are numbered rows and lettered columns in worksheets. It makes ease of view and reference to your data. You can dynamically show/hide worksheet header by using one of the following ways,

- Select PAGE LAYOUT tab in the ribbon and then check or uncheck Headings in the Show group.
- Show/Hide the worksheet headers using ShowHeadings property and showHeadings method.

The following code example describes the above behavior

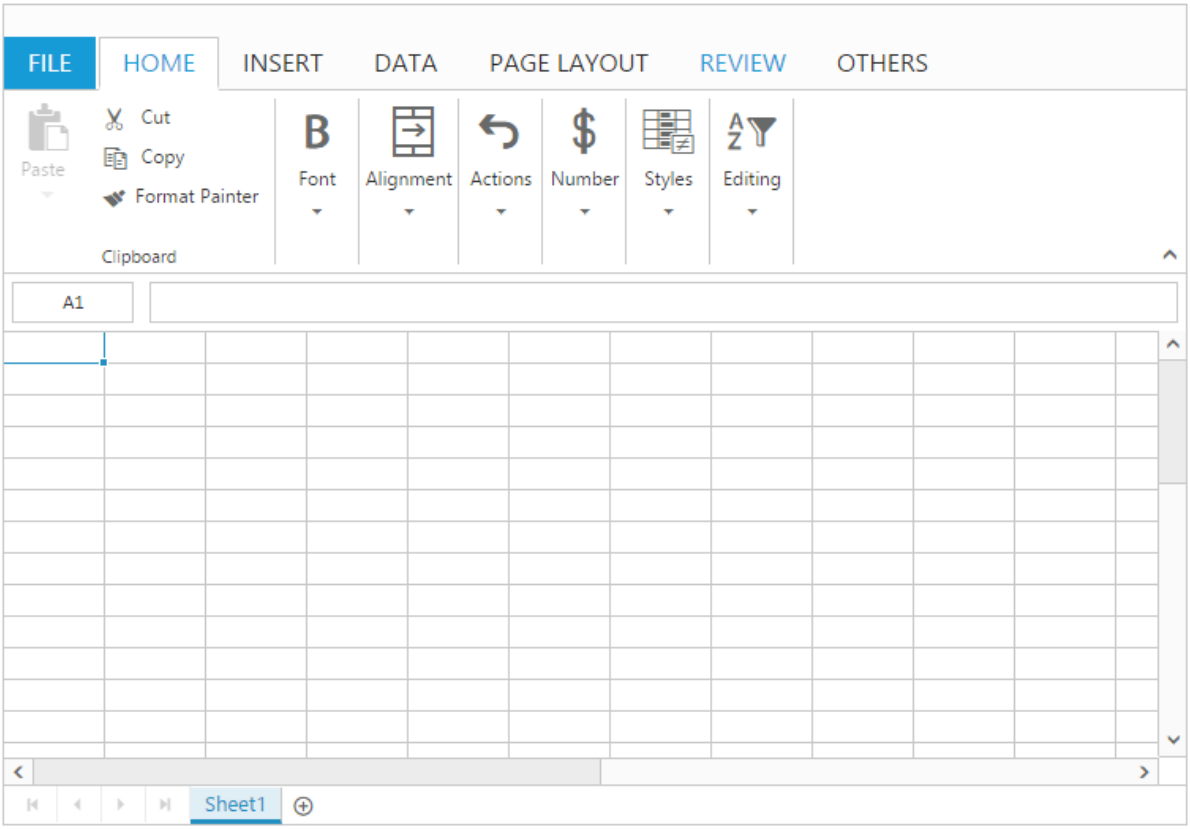
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.ShowHeadings(false).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

The following output is displayed as a result of the above code example.



Show / Hide Sheets

You can dynamically show/hide worksheet by using one of the following ways,

- Right clicking on the worksheet in the sheet tab and then click Hide or Unhide in the context menu
- Hide the sheet using hideSheet method.
- Show the hidden sheet using unhideSheet method.

The following code example describes the above behavior

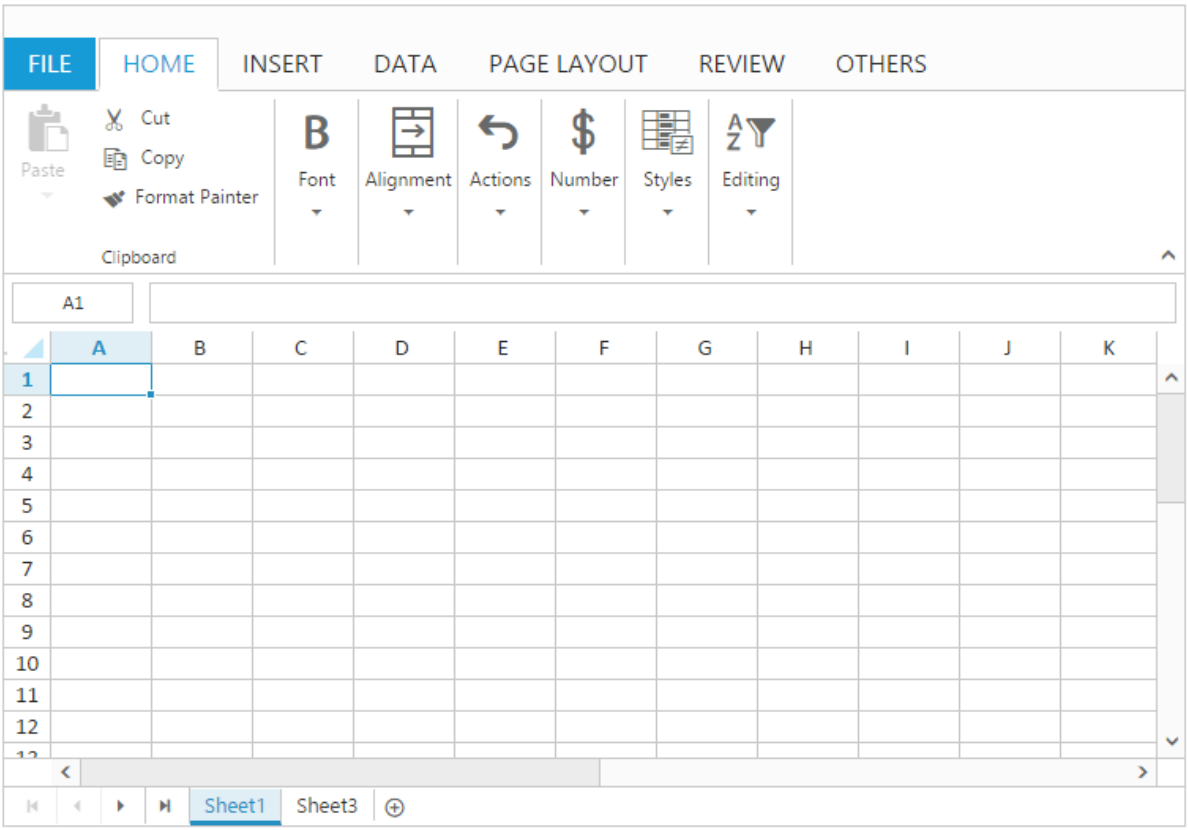
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.SheetCount(3)
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.hideSheet(1);
this.hideSheet(2);
this.unhideSheet(1);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
```

The following output is displayed as a result of the above code example.



Show / Hide Gridlines

Gridlines act as a border like appearance of cells. They are used to distinguish cells on the worksheet. You can dynamically show/hide gridlines by using one of the following ways,

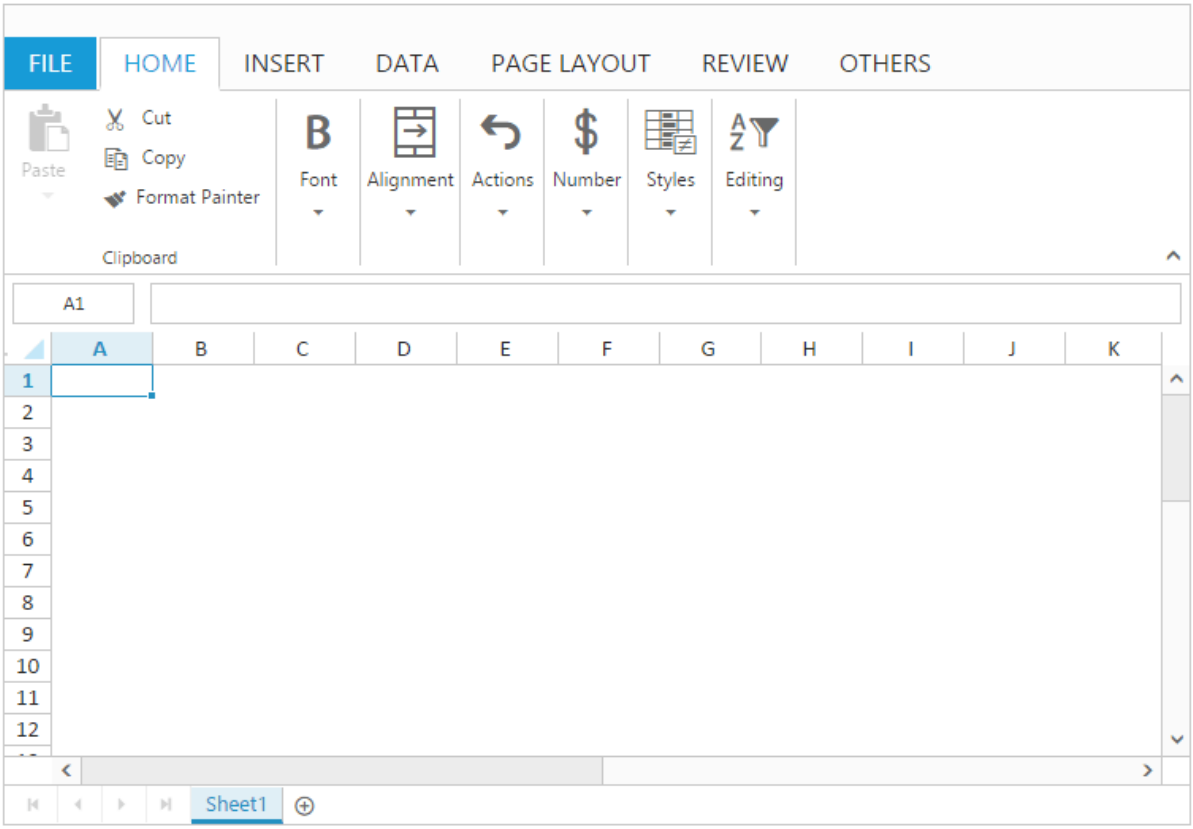
- Select PAGE LAYOUT tab in the ribbon and then check or uncheck Gridlines in the Show group.
- Show/Hide gridlines in a worksheet using ShowGridlines property and showGridlines method.

The following code example describes the above behavior

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.ShowGridlines(false).Add();
})
)
```

The following output is displayed as a result of the above code example.



Rows and Columns

Spreadsheet is a tabular format consisting of rows and columns. Rows and columns are used to represent the editing area in Spreadsheet. The intersection point of rows and columns are called as cells. In that you can perform editing. You have RowCount and ColCount in sheets property for defining the rows and columns count. By default, Spreadsheet creates 20 rows and 21 columns. Based on this grid content will be created.

Rows

Rows are a collection of cells that run horizontally. Each row is identified by the row number in the row header.

Columns

Columns are a collection of cells that run vertically. Each column is identified by column heading in the column header.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).RowCount(50).ColCount(36).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

List of operations

You can perform the following list of operations in rows and columns,

- Insert
- Delete
- Show and Hide
- Resizing

Insert

You can insert blank cells, rows or columns based on the selection in a worksheet. You have to enable the AllowInsert property to perform the insert operation. You can perform insert operation through,

- OTHERS tab in ribbon.
- Context menu

Note: In the header context menu you can insert only rows or columns.

Copyright © 2001 - 2020 Syncfusion Inc.

Insert Shift Bottom

You can dynamically insert blank cells to the top of the selected range and shift the selected cells to down by following ways,

- Click Insert in the context menu and select “Shift Cells Down” option in Insert dialog.
- Select Insert Cells option in Insert button of OTHERS tab in Ribbon and select “Shift Cells Down” option in Insert dialog.

You can also perform insert shift bottom using insertShiftBottom method.

Insert Shift Right

You can dynamically insert blank cells to the left of the selected range and shift the selected cells to right by following ways,

- Click Insert in the context menu and select “Shift Cells Right” option in Insert dialog.
- Select Insert Cells option in Insert button of OTHERS tab in Ribbon and select “Shift Cells Right” option in Insert dialog.

You can also perform insert shift right using insertShiftRight method.

Insert Entire Row

You can dynamically insert the selected number of blank rows to the top of the selected range by following ways,

- Click Insert in the context menu and select “Entire Row” option in Insert dialog.
- Select Insert Cells option in Insert button of OTHERS tab in Ribbon and select “Entire Row” option in Insert dialog.
- Select Insert Sheet Rows option in Insert button of OTHERS tab in Ribbon.
- Click Insert option in row header context menu.

You can also perform insert entire row using insertEntireRow method.

Insert Entire Column

You can dynamically insert the selected number of blank columns to the left of the selected range by following ways,

- Click Insert in the context menu and select “Entire Column” option in Insert dialog.
- Select Insert Cells option in Insert button of OTHERS tab in Ribbon and select “Entire Column” option in Insert dialog.
- Select Insert Sheet Columns option in Insert button of OTHERS tab in Ribbon.
- Click Insert option in column header context menu.

You can also perform insert entire column using insertEntireColumn method.

Delete

You can delete a range of cells, rows or columns based on the selection in worksheet. You have to enable the AllowDelete property to perform delete operation.

You can perform delete operation through,

Copyright © 2001 - 2020 Syncfusion Inc.

- OTHERS tab in Ribbon
- Context menu

Note: In header Context menu you can delete only rows or columns.

Delete Shift Up

You can dynamically delete the selected range of cells and shift the other cells to top by following ways,

- Click Delete in the context menu and select “Shift Cells Up” option in Delete dialog.
- Select Delete Cells option in Delete button of OTHERS tab in Ribbon and select “Shift Cells Up” option in Delete dialog.

You can also perform delete shift up using deleteShiftUp method.

Delete Shift Left

You can dynamically delete the selected range of cells and shift the other cells to left by following ways,

- Click Delete in the context menu and select “Shift Cells Left” option in Delete dialog.
- Select Delete Cells in Delete button of OTHERS tab in Ribbon and select “Shift Cells Left” option in Delete dialog.

You can also perform delete shift left using deleteShiftLeft method.

Delete Entire Row

You can dynamically delete the selected rows and shift the other rows to top by following ways,

- Click Delete in the context menu and select “Entire Row” option in Delete dialog.
- Select Delete Cells option in Delete button of OTHERS tab in Ribbon and select “Entire Row” option in Delete dialog.
- Select Delete Sheet Rows option in Delete button of OTHERS tab in Ribbon.
- Click Delete option in row header context menu.

You can also perform delete entire row using deleteEntireRow method.

Delete Entire Column

You can dynamically delete a selected columns and shift other columns to left by following ways,

- Click Delete in the context menu and select “Entire Column” option in Delete dialog.
- Select Delete Cells option in Delete button of OTHERS tab in Ribbon and select “Entire Column” option in Delete dialog.
- Select Delete Sheet Columns option in Delete button of OTHERS tab in Ribbon.
- Click Delete option in column header context menu.

You can also perform delete entire column using deleteEntireColumn method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

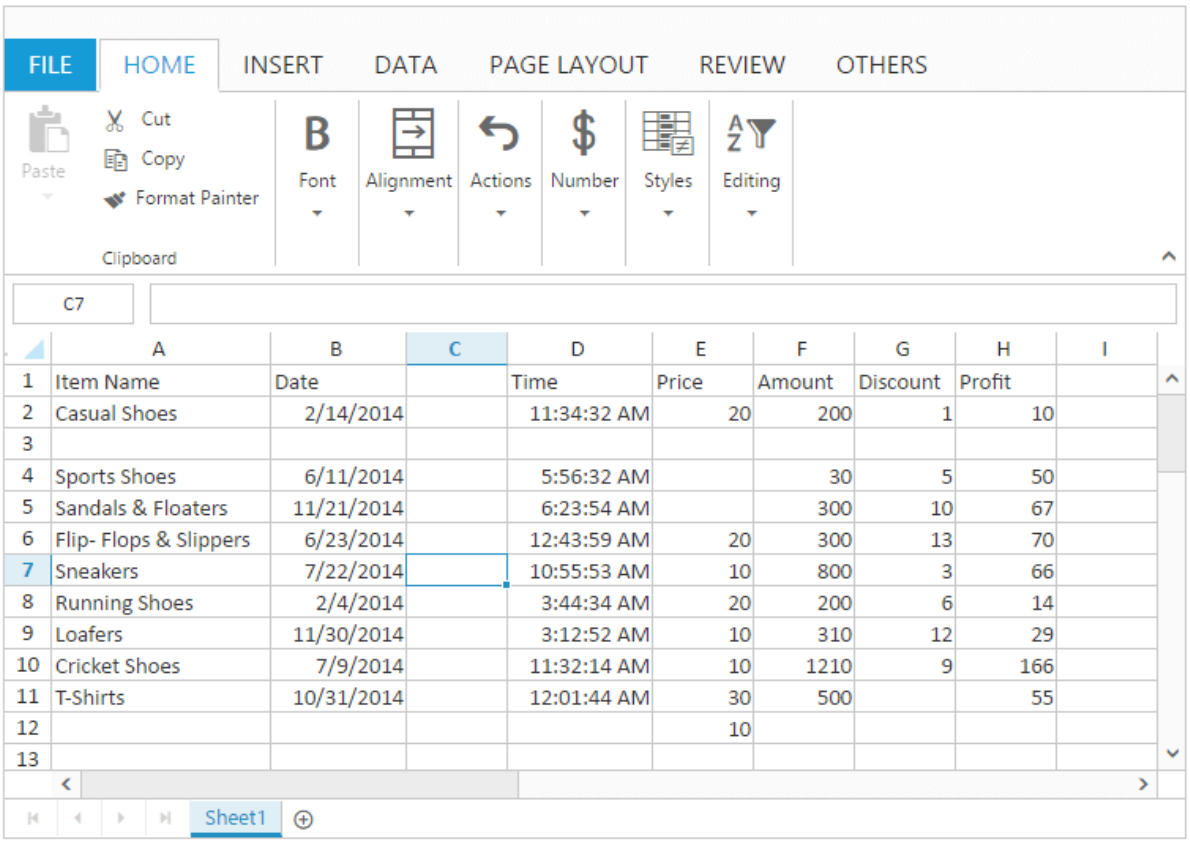
```
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.insertEntireRow(2, 2);
this.insertEntireColumn(2, 2);
this.deleteEntireRow(4, 4);
this.deleteEntireColumn(4, 4);
this.insertShiftBottom({ rowIndex: 4, colIndex: 4 }, { rowIndex: 4,
colIndex: 4 });
this.insertShiftRight({ rowIndex: 3, colIndex: 4 }, { rowIndex: 3,
colIndex: 4 });
this.deleteShiftUp({ rowIndex: 4, colIndex: 6 }, { rowIndex: 4, colIndex:
6 });
this.deleteShiftLeft({ rowIndex: 3, colIndex: 6 }, { rowIndex: 3,
colIndex: 6 });
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G	H	I
1	Item Name	Date		Time	Price	Amount	Discount	Profit	
2	Casual Shoes	2/14/2014		11:34:32 AM	20	200	1	10	
3									
4	Sports Shoes	6/11/2014		5:56:32 AM		30	5	50	
5	Sandals & Floaters	11/21/2014		6:23:54 AM		300	10	67	
6	Flip- Flops & Slippers	6/23/2014		12:43:59 AM	20	300	13	70	
7	Sneakers	7/22/2014		10:55:53 AM	10	800	3	66	
8	Running Shoes	2/4/2014		3:44:34 AM	20	200	6	14	
9	Loafers	11/30/2014		3:12:52 AM	10	310	12	29	
10	Cricket Shoes	7/9/2014		11:32:14 AM	10	1210	9	166	
11	T-Shirts	10/31/2014		12:01:44 AM	30	500		55	
12					10				
13									

Show and Hide

You can show or hide the rows and columns in Spreadsheet using methods and context menu.

Hide Row

You can hide the rows dynamically by using one of the following ways,

- Click “Hide” option in row header context menu.
- Hide the rows using hideRow method.

Hide Column

You can hide the columns dynamically by using one of the following ways,

- Click “Hide” option in column header context menu.
- Hide the columns using hideColumn method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
```

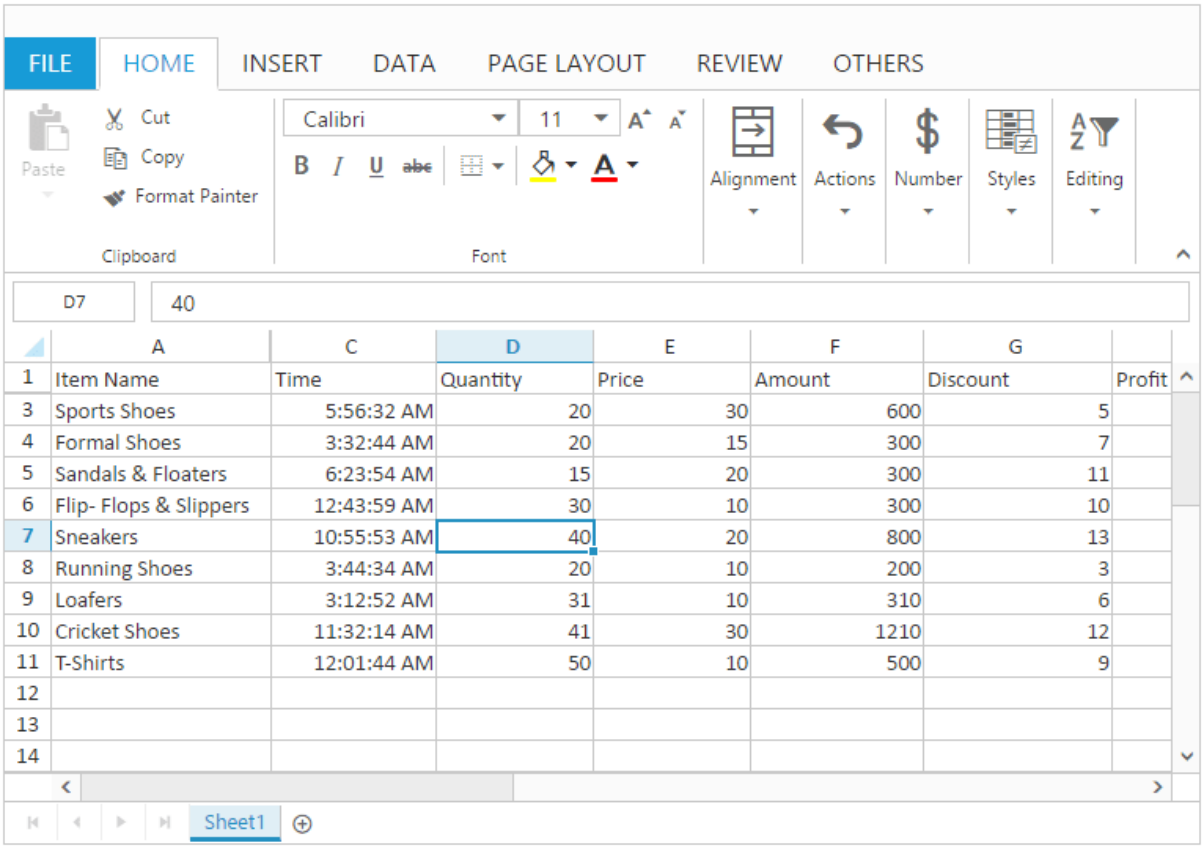
```
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.hideRow(2);
this.hideColumn(2);
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	C	D	E	F	G	
1	Item Name	Time	Quantity	Price	Amount	Discount	Profit
3	Sports Shoes	5:56:32 AM	20	30	600	5	
4	Formal Shoes	3:32:44 AM	20	15	300	7	
5	Sandals & Floaters	6:23:54 AM	15	20	300	11	
6	Flip- Flops & Slippers	12:43:59 AM	30	10	300	10	
7	Sneakers	10:55:53 AM	40	20	800	13	
8	Running Shoes	3:44:34 AM	20	10	200	3	
9	Loafers	3:12:52 AM	31	10	310	6	
10	Cricket Shoes	11:32:14 AM	41	30	1210	12	
11	T-Shirts	12:01:44 AM	50	10	500	9	
12							
13							
14							

Sheet1

Show Row

You can show the hidden rows dynamically by using one of the following ways,

- Click “Unhide” option in row header context menu.
- Show the hidden rows using showRow method.

Show Column

You can show the hidden columns dynamically by using one of the following ways,

- Click “Unhide” option in column header context menu.
- Show the hidden columns using showColumn method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

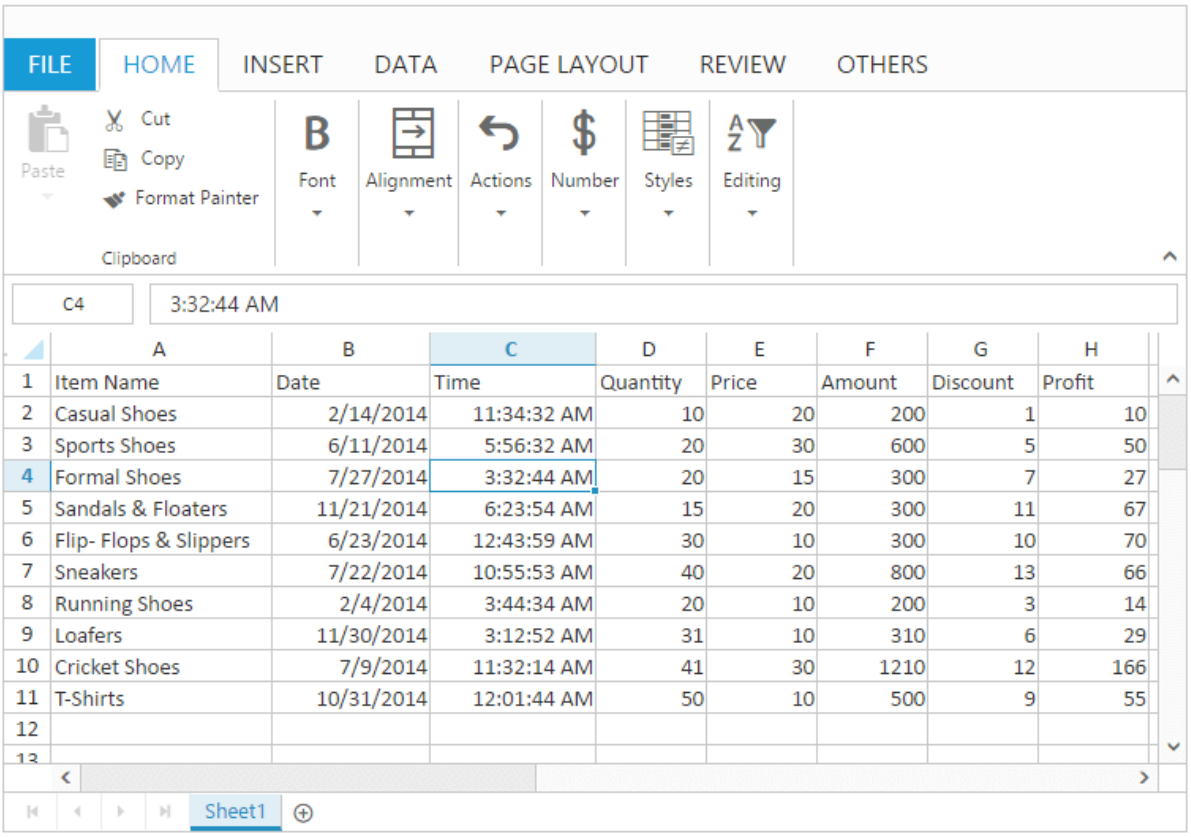
```
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.hideRow(2);
this.hideColumn(2);
this.showRow(2);
this.showColumn(2);
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Resizing

You can change ColumnWidth and RowHeight with the specified value. You have to enable AllowResizing property to perform resizing.

You can perform resizing using one of the following ways,

- Resize option in column header and row header.
- Set the column width by using setColWidth method or ColumnWidth property.
- Set the row height by using setRowHeight method or RowHeight property.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ScrollSettings(scroll =>
{
scroll.Height(510);
})
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.RowHeight(20)
```

Copyright © 2001 - 2020 Syncfusion Inc.

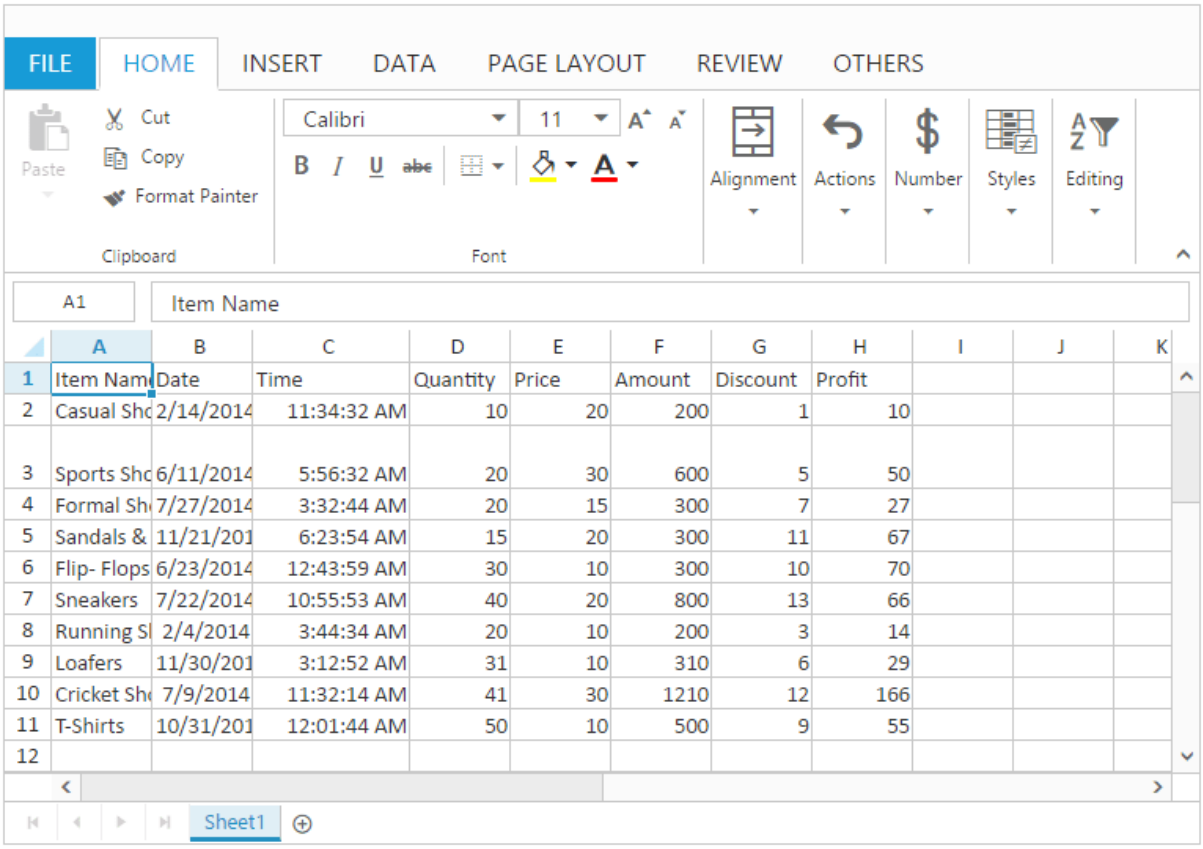
```
.ColumnWidth(64)
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.XLResize.setColWidth(2, 100);
this.XLResize.setRowHeight(2, 40);
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H
1	Item Nam	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Sho	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Sho	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Sh	7/27/2014	3:32:44 AM	20	15	300	7	27
5	Sandals &	11/21/201	6:23:54 AM	15	20	300	11	67
6	Flip- Flops	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Sl	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/201	3:12:52 AM	31	10	310	6	29
10	Cricket Sh	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/201	12:01:44 AM	50	10	500	9	55
12								

Freeze Pane

Freeze pane allows you to keep a portion of the sheet visible, while scrolling through the rest of the worksheet.

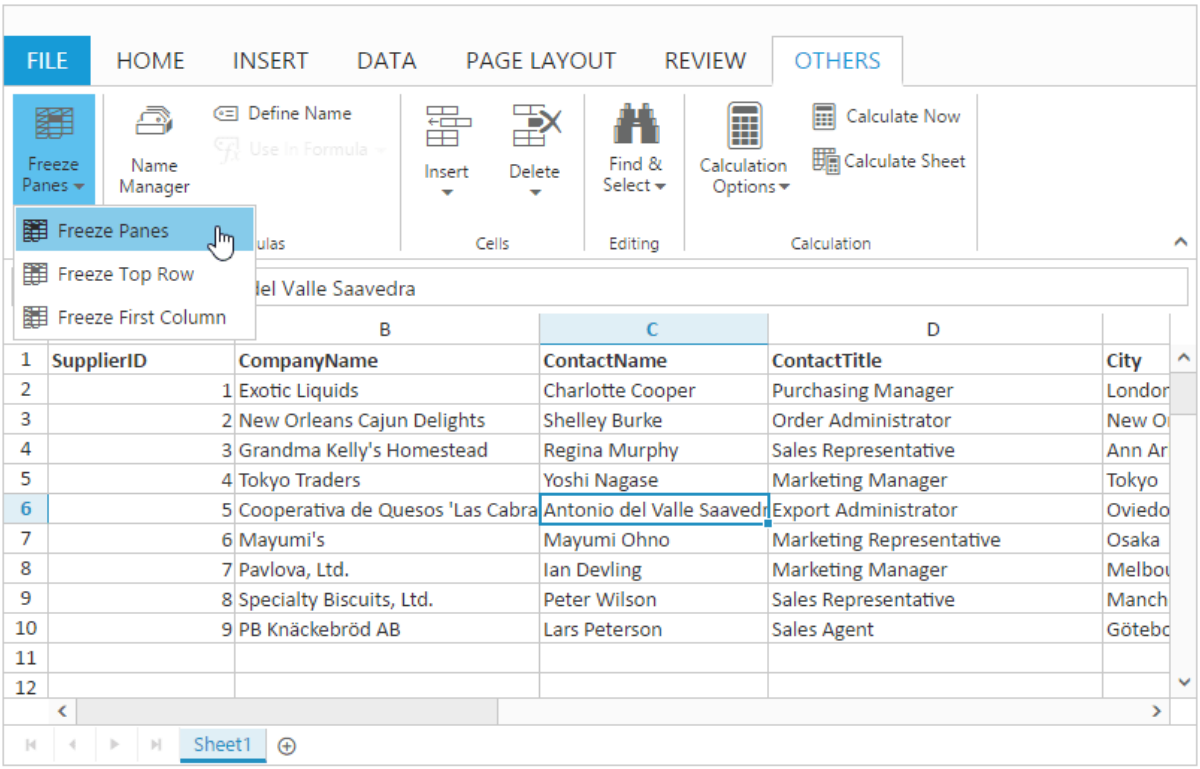
The freeze pane can be applied in a following ways,

1. User Interface
2. Initial Load
3. Method

User Interface

Select any cell and on OTHERS tab click Freeze Panes in Freeze Panes dropdown list.

Copyright © 2001 - 2020 Syncfusion Inc.



Initial Load

You can use AllowFreezing property to enable or disable freeze pane in Spreadsheet.

To specify number of rows and columns to be frozen, use FrozenRows and FrozenColumns property in sheets.

The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
.AllowFreezing(true)
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).FrozenRows(5).F
rozenColumns(2).Add();
})
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PersonDetail> lItems = new List<PersonDetail>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Note: The default value of AllowFreezing property is true.

Method

You can freeze rows or columns using freezePanes method.

The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport)
```

Copyright © 2001 - 2020 Syncfusion Inc.

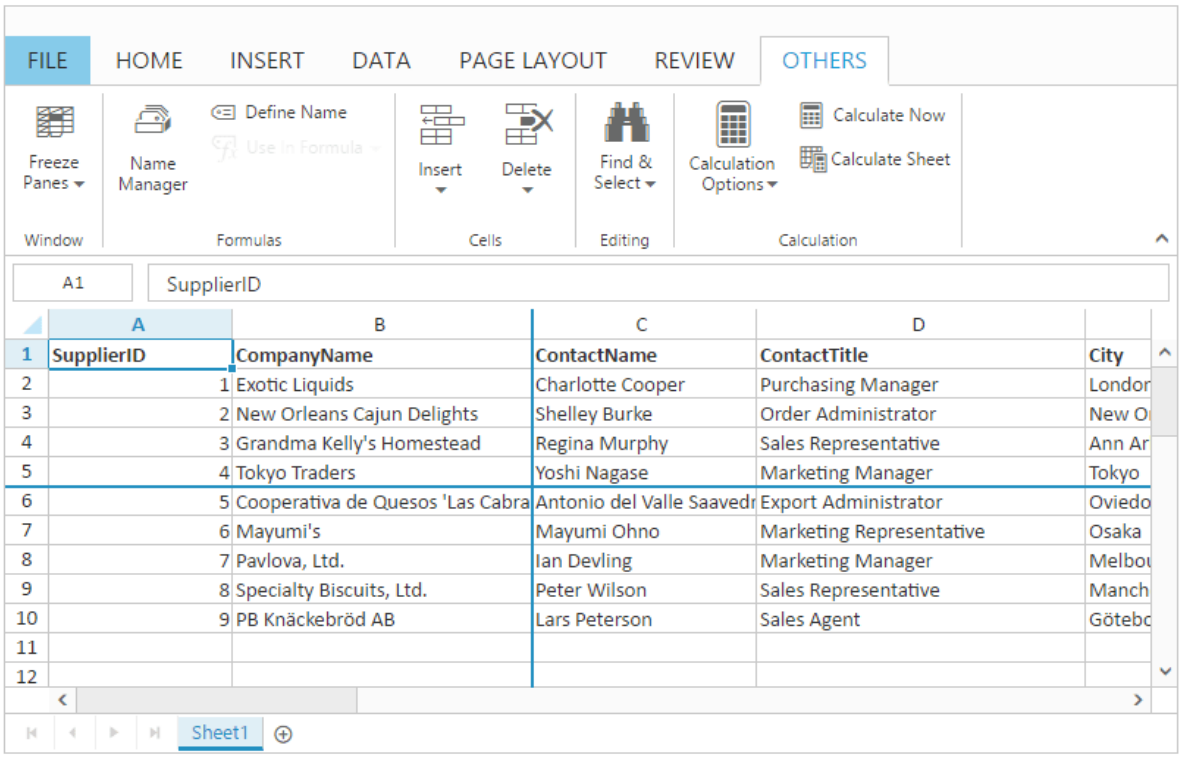
```
this.XLFreeze.freezePanes(5,2);
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PersonDetail> lItems = new List<PersonDetail>();
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above behavior.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	
1	SupplierID	CompanyName	ContactName	ContactTitle	City
2	1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	Londor
3	2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	New O
4	3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	Ann Ar
5	4	Tokyo Traders	Yoshi Nagase	Marketing Manager	Tokyo
6	5	Cooperativa de Quesos 'Las Cabra	Antonio del Valle Saaved	Export Administrator	Oviedo
7	6	Mayumi's	Mayumi Ohno	Marketing Representative	Osaka
8	7	Pavlova, Ltd.	Ian Devling	Marketing Manager	Melbou
9	8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	Manch
10	9	PB Knäckebröd AB	Lars Peterson	Sales Agent	Götebo
11					
12					

Note: When we apply freeze pane by selecting cell “A1”, rows and columns are frozen based on the middle cell in the worksheet.

Freeze Rows

It allows you to keep the certain number of rows visible while scrolling through the rest of the worksheet.

The freeze rows can be applied in a following ways,

1. Initial Load
2. Method

Initial Load

You can freeze rows using `FrozenRows` property in sheets.

The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).FrozenRows(3).A
dd();
})
)
```

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PersonDetail> lItems = new List<PersonDetail>();
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Method

You can freeze rows using freezeRows method.

The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

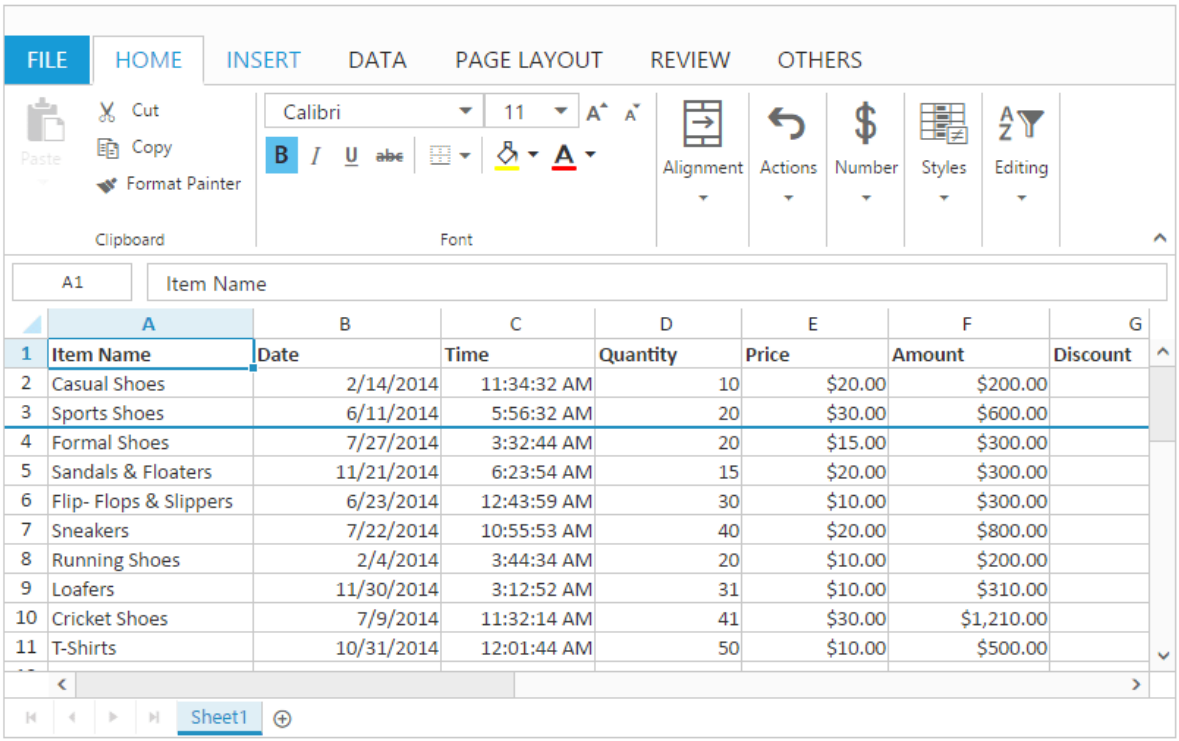
```
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport)
this.XLFreeze.freezeRows(3);
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PersonDetail> lItems = new List<PersonDetail>();
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G
1	Item Name	Date	Time	Quantity	Price	Amount	Discount
2	Casual Shoes	2/14/2014	11:34:32 AM	10	$20.00	$200.00	
3	Sports Shoes	6/11/2014	5:56:32 AM	20	$30.00	$600.00	
4	Formal Shoes	7/27/2014	3:32:44 AM	20	$15.00	$300.00	
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	$20.00	$300.00	
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	$10.00	$300.00	
7	Sneakers	7/22/2014	10:55:53 AM	40	$20.00	$800.00	
8	Running Shoes	2/4/2014	3:44:34 AM	20	$10.00	$200.00	
9	Loafers	11/30/2014	3:12:52 AM	31	$10.00	$310.00	
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	$30.00	$1,210.00	
11	T-Shirts	10/31/2014	12:01:44 AM	50	$10.00	$500.00	

Note: On OTHERS tab click Freeze Top Row in FreezePanes dropdown list, to freeze top row. You can also freeze top row using freezeTopRow method.

Freeze Columns

It allows you to keep the certain number of column visible while scrolling through the rest of the work sheet.

The freeze columns can be applied in a following ways,

1. Initial Load
2. Method

Initial Load

You can apply freeze columns using `FrozenColumns` property in sheets.

The following code example describes the above behavior

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).FrozenColumns(3
).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PersonDetail> lItems = new List<PersonDetail>();
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Method

You can apply freeze columns using freezeColumns method.

The following code example describes the above behavior

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport)
this.XLFreeze.freezeColumns(3);
}
</script>
```

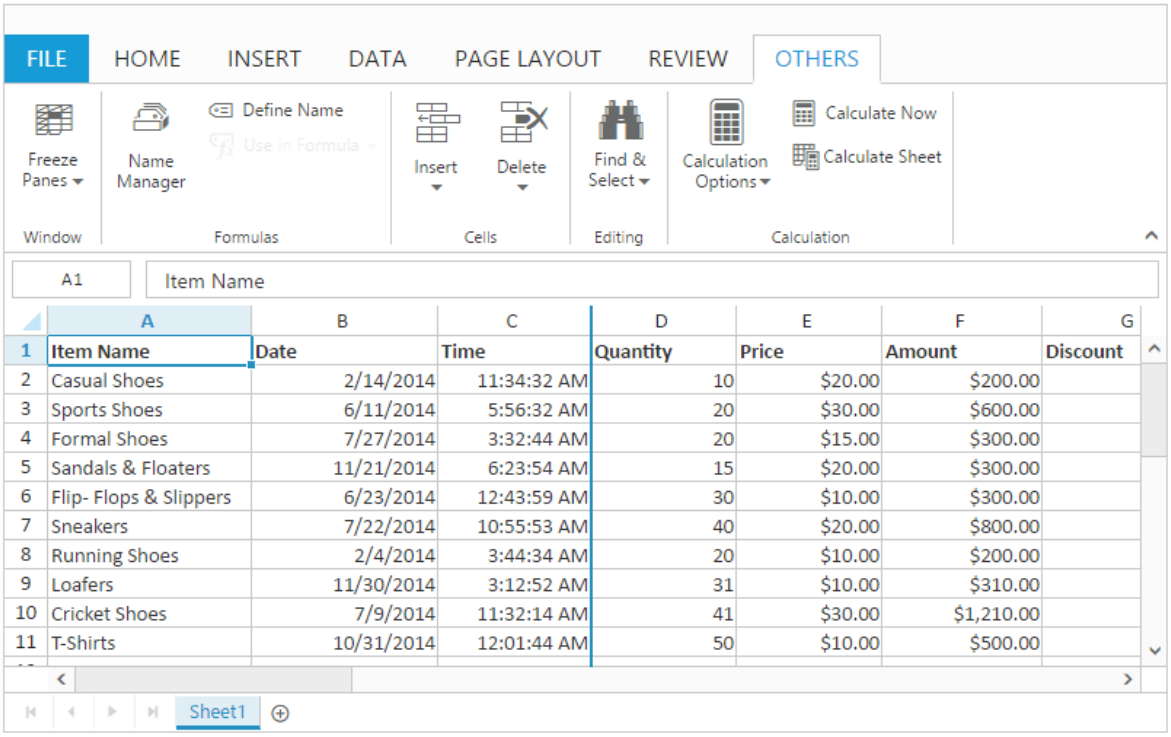
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PersonDetail> lItems = new List<PersonDetail>();
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G
1	Item Name	Date	Time	Quantity	Price	Amount	Discount
2	Casual Shoes	2/14/2014	11:34:32 AM	10	$20.00	$200.00	
3	Sports Shoes	6/11/2014	5:56:32 AM	20	$30.00	$600.00	
4	Formal Shoes	7/27/2014	3:32:44 AM	20	$15.00	$300.00	
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	$20.00	$300.00	
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	$10.00	$300.00	
7	Sneakers	7/22/2014	10:55:53 AM	40	$20.00	$800.00	
8	Running Shoes	2/4/2014	3:44:34 AM	20	$10.00	$200.00	
9	Loafers	11/30/2014	3:12:52 AM	31	$10.00	$310.00	
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	$30.00	$1,210.00	
11	T-Shirts	10/31/2014	12:01:44 AM	50	$10.00	$500.00	

Note: On OTHERS tab click Freeze First Column in FreezePanes dropdown list, to freeze left Column. You can also apply freeze first column using freezeLeftColumn method.

Unfreeze Pane

Unfreeze all the rows and columns to scroll through entire sheet.

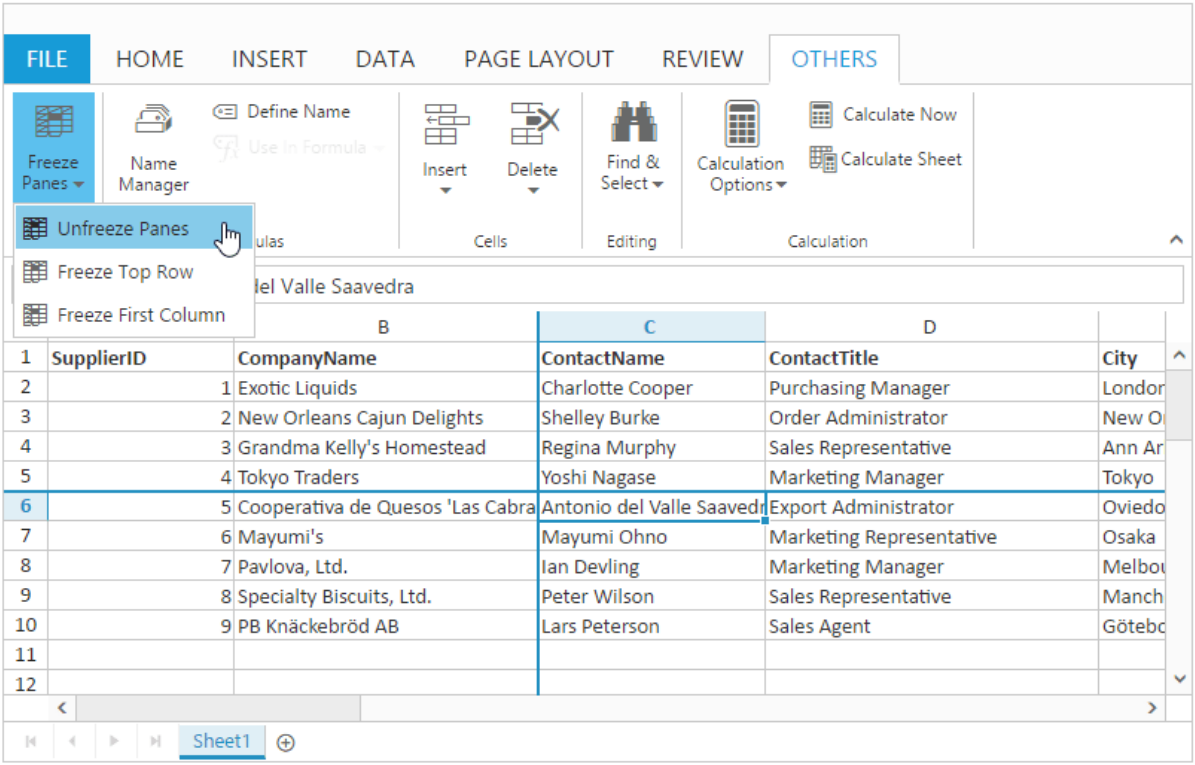
The unfreeze pane can be applied in a following ways,

1. User Interface
2. Method

User Interface

On OTHERS tab click Unfreeze Panes in Freeze Panes dropdown list.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	
1	SupplierID	CompanyName	ContactName	ContactTitle	City
2	1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	Londor
3	2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	New O
4	3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	Ann Ar
5	4	Tokyo Traders	Yoshi Nagase	Marketing Manager	Tokyo
6	5	Cooperativa de Quesos 'Las Cabra	Antonio del Valle Saavedr	Export Administrator	Oviedo
7	6	Mayumi's	Mayumi Ohno	Marketing Representative	Osaka
8	7	Pavlova, Ltd.	Ian Devling	Marketing Manager	Melbou
9	8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	Manch
10	9	PB Knäckebröd AB	Lars Peterson	Sales Agent	Götebc
11					
12					

Method

You can unfreeze rows or columns using unfreezePanes method.

The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport){
this.XLFreeze.freezePanes(5,2);
this.XLFreeze.unfreezePanes();
}
}
</script>
```

C#

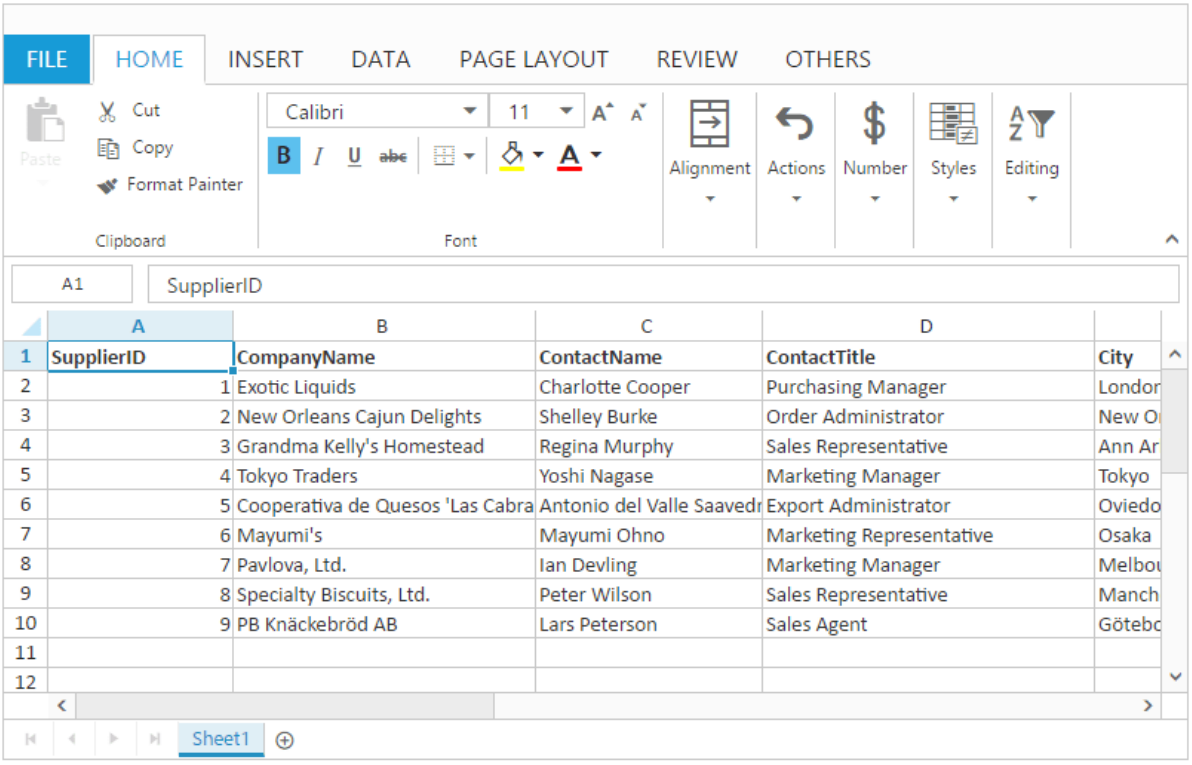
```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
List<PersonDetail> lItems = new List<PersonDetail>();
lItems.Add(new PersonDetail() { SupplierID = 1, CompanyName = "Exotic
Liquids", ContactName = "Charlotte Cooper", ContactTitle = "Purchasing
Manager", City = "London", PostalCode = "EC1 4SD", Country = "UK" });
lItems.Add(new PersonDetail() { SupplierID = 2, CompanyName = "New
Orleans Cajun Delights", ContactName = "Shelley Burke", ContactTitle =
"Order Administrator", City = "New Orleans", PostalCode = "70117",
Country = "USA" });
lItems.Add(new PersonDetail() { SupplierID = 3, CompanyName = "Grandma
Kelly's Homestead", ContactName = "Regina Murphy", ContactTitle = "Sales
Representative", City = "Ann Arbor", PostalCode = "48104", Country =
"USA" });
lItems.Add(new PersonDetail() { SupplierID = 4, CompanyName = "Tokyo
Traders", ContactName = "Yoshi Nagase", ContactTitle = "Marketing
Manager", City = "Tokyo", PostalCode = "100", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 5, CompanyName =
"Cooperativa de Quesos 'Las Cabras'", ContactName = "Antonio del Valle
Saavedra", ContactTitle = "Export Administrator", City = "Oviedo",
PostalCode = "33007", Country = "Spain" });
lItems.Add(new PersonDetail() { SupplierID = 6, CompanyName = "Mayumi's",
ContactName = "Mayumi Ohno", ContactTitle = "Marketing Representative",
City = "Osaka", PostalCode = "545", Country = "Japan" });
lItems.Add(new PersonDetail() { SupplierID = 7, CompanyName = "Pavlova,
Ltd.", ContactName = "Ian Devling", ContactTitle = "Marketing Manager",
City = "Melbourne", PostalCode = "3058", Country = "Australia" });
lItems.Add(new PersonDetail() { SupplierID = 8, CompanyName = "Specialty
Biscuits, Ltd.", ContactName = "Peter Wilson", ContactTitle = "Sales
Representative", City = "Manchester", PostalCode = "M14 GSD", Country =
"UK" });
lItems.Add(new PersonDetail() { SupplierID = 9, CompanyName = "PB
Knäckebröd AB", ContactName = "Lars Peterson", ContactTitle = "Sales
Agent", City = "Göteborg", PostalCode = "S-345 67", Country = "Sweden"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	
1	SupplierID	CompanyName	ContactName	ContactTitle	City
2	1	Exotic Liquids	Charlotte Cooper	Purchasing Manager	Londor
3	2	New Orleans Cajun Delights	Shelley Burke	Order Administrator	New O
4	3	Grandma Kelly's Homestead	Regina Murphy	Sales Representative	Ann Ar
5	4	Tokyo Traders	Yoshi Nagase	Marketing Manager	Tokyo
6	5	Cooperativa de Quesos 'Las Cabra	Antonio del Valle Saavedr	Export Administrator	Oviedo
7	6	Mayumi's	Mayumi Ohno	Marketing Representative	Osaka
8	7	Pavlova, Ltd.	Ian Devling	Marketing Manager	Melbou
9	8	Specialty Biscuits, Ltd.	Peter Wilson	Sales Representative	Manch
10	9	PB Knäckebröd AB	Lars Peterson	Sales Agent	Götebc
11					
12					

Open and Save

The native data format for Spreadsheet is JSON. You can load and store JSON data with Spreadsheet. In Spreadsheet we have saveAsJSON and loadFromJSON method which is used to save Spreadsheet as JSON and same JSON used to render Spreadsheet.

JAVASCRIPT

```
function saveAsJSON() {
var excelObj = $("#Spreadsheet").data("ejSpreadsheet");
window.excelData = excelObj.saveAsJSON();
}
function loadFromJSON() {
var excelObj = $("#Spreadsheet").data("ejSpreadsheet");
excelObj.loadFromJSON(window.excelData);
}
```

When you open excel file, it need to be read and converted to client side Spreadsheet model. The converted client side Spreadsheet model is sent as JSON which is used to render Spreadsheet. Similarly, when you save the Spreadsheet, the client Spreadsheet model is sent to the server as JSON for processing and saved. Server configuration is used for this process.

Open

The Spreadsheet can open excel documents as like excel application with its data, style, format. To enable open option in Spreadsheet set AllowImport option to true. Since Spreadsheet uses a server side helper to open document, set ImportMapper in ImportSettings to map server action.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowImport(true)
.ImportSettings(import =>
{
import.ImportMapper("Import");
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
return View();
}
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Import(ImportRequest importRequest)
{
return importRequest.SpreadsheetActions();
}
}
}
```

Following file types can be opened in Spreadsheet

- XLS
- XLSX
- CSV

Click here to view online demo sample. You can open excel documents in following ways

1. Initial settings
2. Methods
3. User Interface

Initial settings

The Spreadsheet can load excel documents initially. The document can be specified either from client side or in server side. To load excel documents initially from client side, set ImportUrl in ImportSettings. The code snippets for document initial load on client side are as follows,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ImportSettings(import =>
{
import.ImportMapper("Import").ImportUrl("http://mvc.syncfusion.com/Spread
sheet/LargeData.xlsx");
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
return View();
}
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Import(ImportRequest importRequest)
{
return importRequest.SpreadsheetActions();
}
}
}
```

To load excel documents initially from server side, set ImportOnLoadas true and assign document stream or URL in the server. The code snippets for document initial load from server side are as follows,

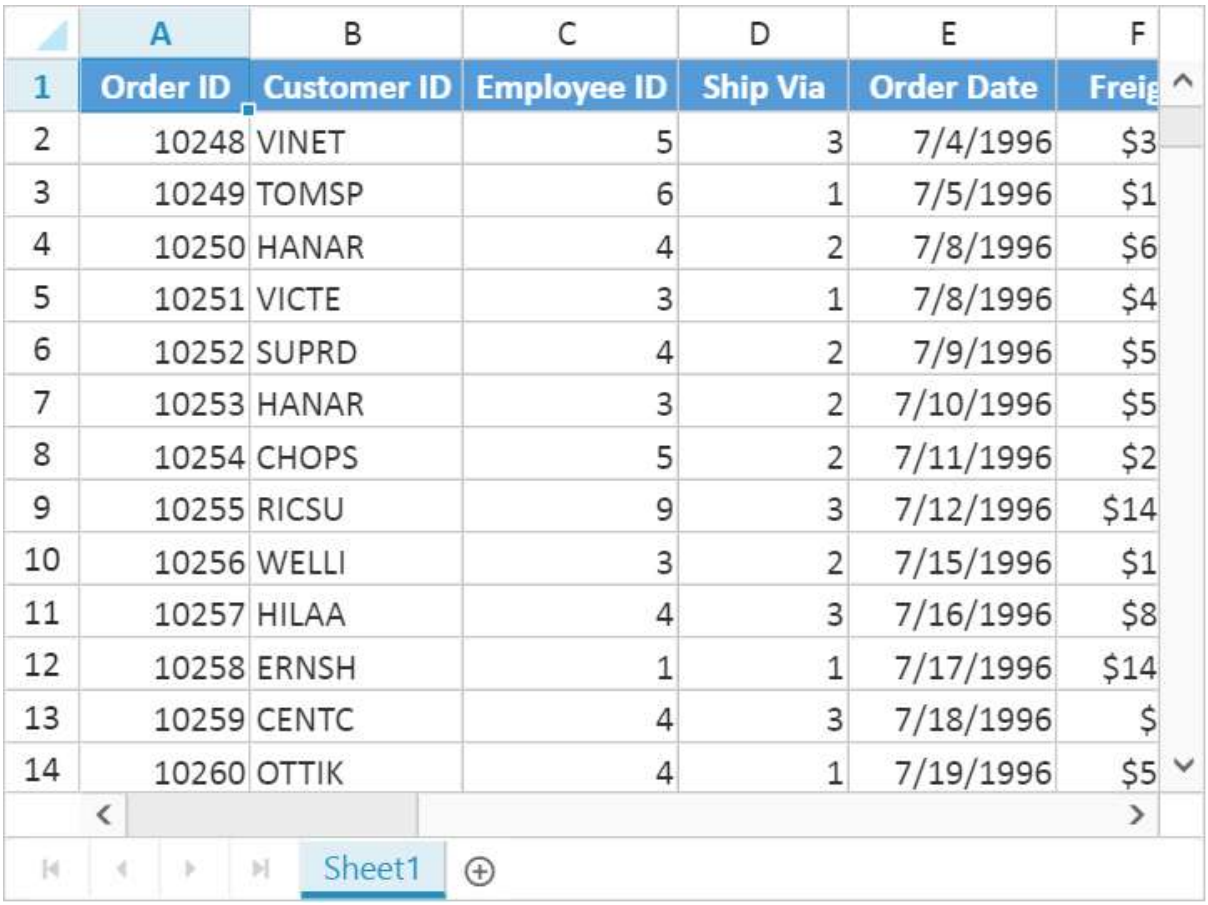
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ImportSettings(import =>
{
import.ImportOnLoad(true).ImportMapper("Import");
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
return View();
}
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Import(ImportRequest importRequest)
{
importRequest.Url =
"http://mvc.syncfusion.com/Spreadsheet/LargeData.xlsx";
return importRequest.SpreadsheetActions();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F
1	Order ID	Customer ID	Employee ID	Ship Via	Order Date	Freig
2	10248	VINET	5	3	7/4/1996	$3
3	10249	TOMSP	6	1	7/5/1996	$1
4	10250	HANAR	4	2	7/8/1996	$6
5	10251	VICTE	3	1	7/8/1996	$4
6	10252	SUPRD	4	2	7/9/1996	$5
7	10253	HANAR	3	2	7/10/1996	$5
8	10254	CHOPS	5	2	7/11/1996	$2
9	10255	RICSU	9	3	7/12/1996	$14
10	10256	WELLI	3	2	7/15/1996	$1
11	10257	HILAA	4	3	7/16/1996	$8
12	10258	ERNSH	1	1	7/17/1996	$14
13	10259	CENTC	4	3	7/18/1996	$
14	10260	OTTIK	4	1	7/19/1996	$5

Sheet1

Methods

To open an excel document, import method should be called with import options as a parameter. The Spreadsheet can open excel document as a stream or file URL.

Stream

Spreadsheet can open excel document as a stream and the document stream was either from the client side or it can be specified in server side. The code snippets to open excel document as a stream from client side are as follows,

JAVASCRIPT

```
function fileOpen(args) {
var excelObj = $("#Spreadsheet").data("ejSpreadsheet"),
stream = args.files[0]; // file stream from ejUploadbox
excelObj["import"]({ file: stream });
}
```

The Code snippets to specify excel document as stream in server side are as follows,

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
var DataSource = new OrderItemsDataContext().GetAllItemDetails.ToList();
ViewBag.Datasource = DataSource;
return View();
}
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Import(ImportRequest importRequest)
{
importRequest.FileStream = getFileStream(); // assign file stream
return importRequest.SpreadsheetActions();
}
}
}
```

File URL

Spreadsheet can open excel document from specified URL. The URL can be specified either from client side or in server side. The code snippets to open excel document as URL from client side are as follows,

JAVASCRIPT

```
function fileOpen() {
var excelObj = $("#Spreadsheet").data("ejSpreadsheet");
excelObj["import"]({Url:
"http://mvc.syncfusion.com/Spreadsheet/LargeData.xlsx"});
}
```

The Code snippets to specify excel document as URL in server side are as follows,

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
return View();
}
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Import(ImportRequest importRequest)
{
importRequest.Url =
"http://mvc.syncfusion.com/Spreadsheet/LargeData.xlsx";
return importRequest.SpreadsheetActions();
}
}
}
```

User Interface

You can dynamically open excel document by clicking the file menu in ribbon and choose Open to upload excel file.

Copyright © 2001 - 2020 Syncfusion Inc.

Save

The Spreadsheet can save its data, style, format into an excel file. To enable save option in Spreadsheet set AllowExporting option in ExportSettings as true. Since Spreadsheet uses server side helper to save documents set ExcelUrl in ExportSettings option.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ExportSettings(export =>
{
export.AllowExporting(true).ExcelUrl("ExportToExcel").CsvUrl("ExportToCSV
").PdfUrl("ExportToPDF");
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
return View();
}
[AcceptVerbs(HttpVerbs.Post)]
public void ExportToExcel(string sheetModel, string sheetData, string
password)
{
if (String.IsNullOrEmpty(password))
Spreadsheet.Save(sheetModel, sheetData, "sample", ExportFormat.XLSX,
ExcelVersion.Excel2013);
else
Spreadsheet.Save(sheetModel, sheetData, "sample", ExportFormat.XLSX,
ExcelVersion.Excel2013, password);
}
[AcceptVerbs(HttpVerbs.Post)]
public void ExportToCsv(string sheetModel, string sheetData)
{
Spreadsheet.Save(sheetModel, sheetData, "sample", ExportFormat.CSV);
}
[AcceptVerbs(HttpVerbs.Post)]
public void ExportToPdf(string sheetModel, string sheetData)
{
Spreadsheet.Save(sheetModel, sheetData, "sample", ExportFormat.PDF);
}
}
}
```

Note: To export as Stream skip file name parameter in Save method. For more details refer below code snippets,

Stream stream = Spreadsheet.Save(sheetModel, sheetData, ExportFormat.XLSX, ExcelVersion.Excel2013);

Copyright © 2001 - 2020 Syncfusion Inc.

You can save Spreadsheet contents with following file types,

- XLS
- XLSX
- CSV
- PDF

Click here to view online demo sample. You can save excel documents in following ways

1. Methods
2. User Interface

Methods

To save Spreadsheet document as excel file, export method should be called with file type as parameter. The code snippets to save Spreadsheet document are as follows,

JAVASCRIPT

```
function saveAsFile() {
var excelObj = $("#spreadsheet").data("ejSpreadsheet");
excelObj.XLExport["export"](ej.Spreadsheet.exportType.Excel);
}
```

User Interface

You can dynamically save Spreadsheet by clicking file menu in ribbon and choose SaveAs option.

Server dependencies

Import and Export Helper functions are available in the assembly Syncfusion.EJ.Export, which is available in Essential Studio and Essential JavaScript builds. The full list of assemblies needed for Spreadsheet import and export are as follows.

1. Syncfusion.EJ
2. Syncfusion.EJ.Export
3. Syncfusion.Linq.Base
4. Syncfusion.Compression.Base
5. Syncfusion.DocIO.Base
6. Syncfusion.XlsIO.Base
7. Syncfusion.PDF.Base

Note: 1. The above mentioned assemblies will be available in below location after Essential Studio build installation.

2. C:\Program Files (x86)\Syncfusion\Essential Studio\x.x.x.x\precompiledassemblies\x.x.x.x\y.y.

3. x.x.x.x defines build version of Essential Studio and y.y defines .NET Framework version.

Data Binding

Spreadsheet can be populated with external datasource using DataSource property. The DataSource property can be assigned either with the instance of DataManager or JSON data array collection. Spreadsheet supports three different kinds of Data binding.

Copyright © 2001 - 2020 Syncfusion Inc.

- Local Data
- Remote Data
- HTML Table Data

Local Data

To bind local data to the Spreadsheet, you can assign a JSON array to the worksheet DataSource property. The following code illustrates how to bind local data to the Spreadsheet,

ASPX-CS

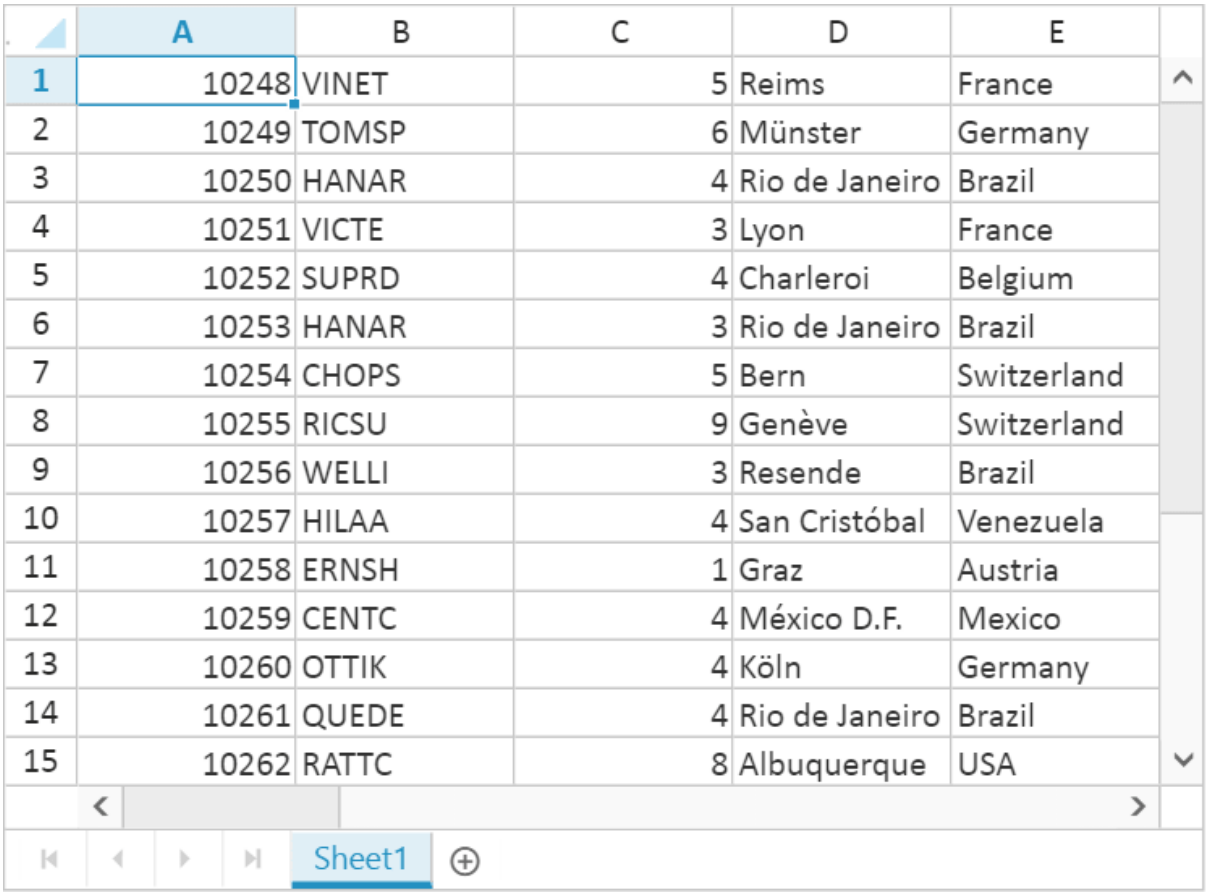
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).ShowHeader(fals
e).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<OrderDetail> lItems = new List<OrderDetail>();
lItems.Add(new OrderDetail() { OrderID = 10248, CustomerID = "VINET",
EmployeeID = 5, ShipCity = "Reims", ShipCountry = "France" });
lItems.Add(new OrderDetail() { OrderID = 10249, CustomerID = "TOMSP",
EmployeeID = 6, ShipCity = "Münster", ShipCountry = "Germany" });
lItems.Add(new OrderDetail() { OrderID = 10250, CustomerID = "HANAR",
EmployeeID = 4, ShipCity = "Rio de Janeiro", ShipCountry = "Brazil" });
lItems.Add(new OrderDetail() { OrderID = 10251, CustomerID = "VICTE",
EmployeeID = 3, ShipCity = "Lyon", ShipCountry = "France" });
lItems.Add(new OrderDetail() { OrderID = 10252, CustomerID = "SUPRD",
EmployeeID = 4, ShipCity = "Charleroi", ShipCountry = "Belgium" });
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code snippets.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E
1	10248	VINET	5	Reims	France
2	10249	TOMSP	6	Münster	Germany
3	10250	HANAR	4	Rio de Janeiro	Brazil
4	10251	VICTE	3	Lyon	France
5	10252	SUPRD	4	Charleroi	Belgium
6	10253	HANAR	3	Rio de Janeiro	Brazil
7	10254	CHOPS	5	Bern	Switzerland
8	10255	RICSU	9	Genève	Switzerland
9	10256	WELLI	3	Resende	Brazil
10	10257	HILAA	4	San Cristóbal	Venezuela
11	10258	ERNSH	1	Graz	Austria
12	10259	CENTC	4	México D.F.	Mexico
13	10260	OTTIK	4	Köln	Germany
14	10261	QUEDE	4	Rio de Janeiro	Brazil
15	10262	RATTC	8	Albuquerque	USA

Sheet1

Remote Data

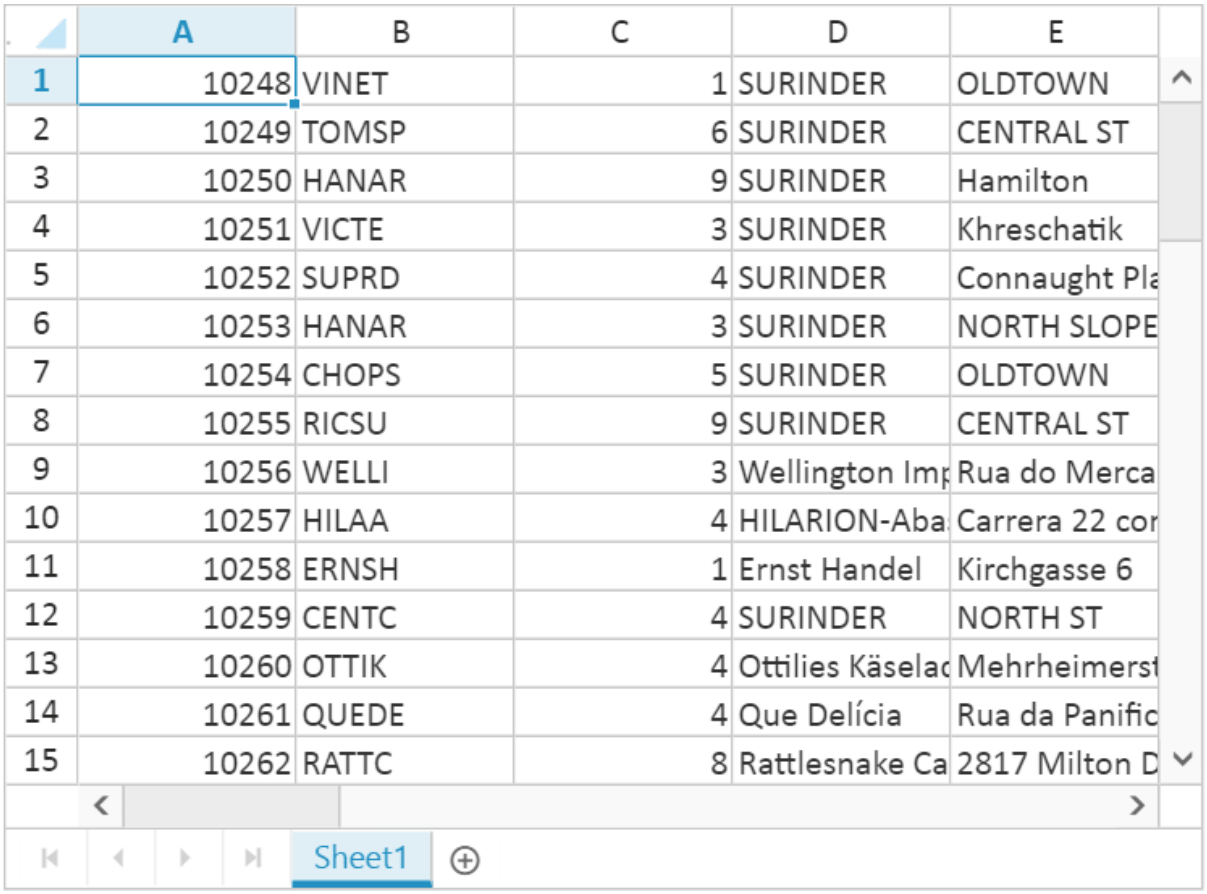
To bind remote data to the Spreadsheet, you can assign a service data as an instance of DataManager to the worksheet DataSource property. The following code illustrates how to bind remote data to the Spreadsheet,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders"
)
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight'])
.take(50)").Add();
})
)
```

The following output is displayed as a result of the above code snippets.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E
1	10248	VINET	1	SURINDER	OLDTOWN
2	10249	TOMSP	6	SURINDER	CENTRAL ST
3	10250	HANAR	9	SURINDER	Hamilton
4	10251	VICTE	3	SURINDER	Khreschatik
5	10252	SUPRD	4	SURINDER	Connaught Pla
6	10253	HANAR	3	SURINDER	NORTH SLOPE
7	10254	CHOPS	5	SURINDER	OLDTOWN
8	10255	RICSU	9	SURINDER	CENTRAL ST
9	10256	WELLI	3	Wellington Imp	Rua do Merca
10	10257	HILAA	4	HILARION-Aba	Carrera 22 cor
11	10258	ERNSH	1	Ernst Handel	Kirchgasse 6
12	10259	CENTC	4	SURINDER	NORTH ST
13	10260	OTTIK	4	Ottilies Käselac	Mehrheimerst
14	10261	QUEDE	4	Que Delícia	Rua da Panific
15	10262	RATTC	8	Rattlesnake Ca	2817 Milton D

Sheet1

Offline Mode

To avoid sending post back request to server on every action, Spreadsheet allows user to create, update and delete data on client side. To enable this, set Offline property of DataManager as true to fetch all data from server on initial rendering of Spreadsheet and perform all operation on client side.

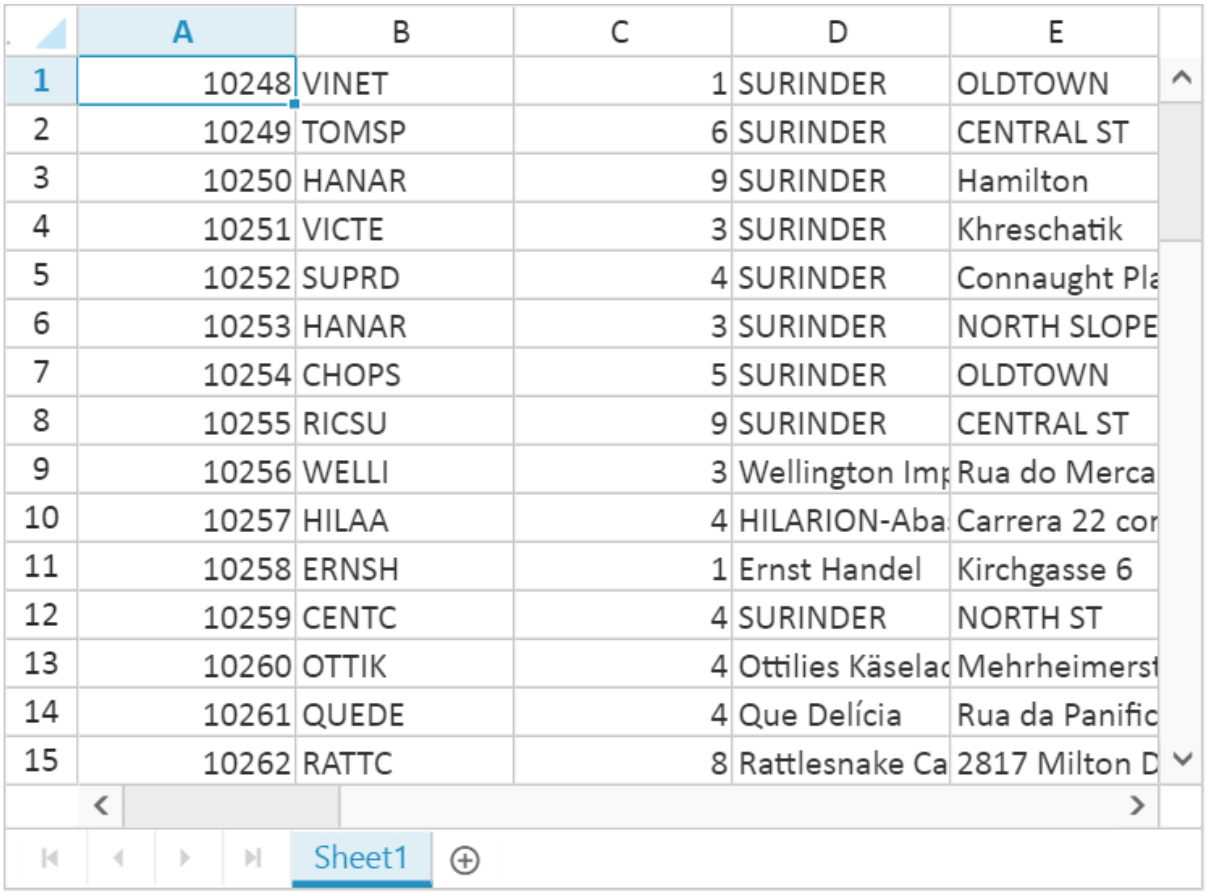
The following code illustrates Offline data binding for Spreadsheet,

ASPX-CS

```
@(Html.EJ().DataManager("FlatData").URL("http://mvc.syncfusion.com/Servic
es/Northwnd.svc/Orders/").Offline(true)))
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.DataManagerID("FlatData")
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipName',  'ShipAddress'])")
.PrimaryKey("OrderID").Add();
})
)
```

The following output is displayed as a result of the above code snippets.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E
1	10248	VINET	1	SURINDER	OLDTOWN
2	10249	TOMSP	6	SURINDER	CENTRAL ST
3	10250	HANAR	9	SURINDER	Hamilton
4	10251	VICTE	3	SURINDER	Khreschatik
5	10252	SUPRD	4	SURINDER	Connaught Pla
6	10253	HANAR	3	SURINDER	NORTH SLOPE
7	10254	CHOPS	5	SURINDER	OLDTOWN
8	10255	RICSU	9	SURINDER	CENTRAL ST
9	10256	WELLI	3	Wellington Imp	Rua do Merca
10	10257	HILAA	4	HILARION-Aba	Carrera 22 cor
11	10258	ERNSH	1	Ernst Handel	Kirchgasse 6
12	10259	CENTC	4	SURINDER	NORTH ST
13	10260	OTTIK	4	Ottilies Käselac	Mehrheimerst
14	10261	QUEDE	4	Que Delícia	Rua da Panific
15	10262	RATTC	8	Rattlesnake Ca	2817 Milton D

Sheet1

Note: For further reference about Offline property in DataManager refer following link

HTML Table Data

An HTML Table element can also be used as the data source of Spreadsheet. To use HTML Table as data source, the table element should be passed to worksheet DataSource property of Spreadsheet as an instance of the DataManager. The following code illustrates how to bind HTML Table data to the Spreadsheet,

ASPX-CS

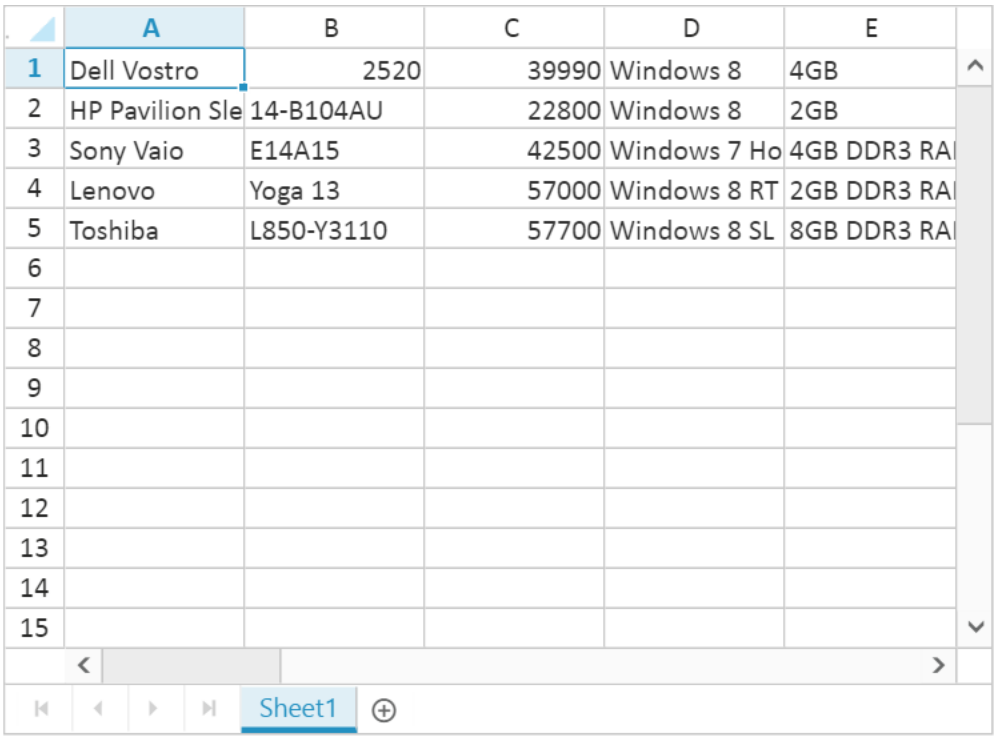
```
@(Html.EJ().DataManager("FlatData").Table("#_table1"))
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.DataManagerID("FlatData").Add();
})
)
<script id="_table1" type="text/template">
<table id="Table1">
<thead>
<tr>
<th>Laptop</th>
<th>Model</th>
<th>Price</th>
<th>OS</th>
<th>RAM</th>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<th>ScreenSize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dell Vostro</td>
<td>2520</td>
<td>39990</td>
<td>Windows 8</td>
<td>4GB</td>
<td>15.6</td>
</tr>
<tr>
<td>HP Pavilion Sleekbook</td>
<td>14-B104AU</td>
<td>22800</td>
<td>Windows 8</td>
<td>2GB</td>
<td>14</td>
</tr>
<tr>
<td>Sony Vaio</td>
<td>E14A15</td>
<td>42500</td>
<td>Windows 7 Home Premium</td>
<td>4GB DDR3 RAM</td>
<td>14</td>
</tr>
<tr>
<td>Lenovo</td>
<td>Yoga 13</td>
<td>57000</td>
<td>Windows 8 RT</td>
<td>2GB DDR3 RAM</td>
<td>11.6</td>
</tr>
<tr>
<td>Toshiba</td>
<td>L850-Y3110</td>
<td>57700</td>
<td>Windows 8 SL</td>
<td>8GB DDR3 RAM</td>
<td>15.6</td>
</tr>
</tbody>
</table>
</script>
```

The following output is displayed as a result of the above code snippets.

Copyright © 2001 - 2020 Syncfusion Inc.



Ways to bind data in Spreadsheet

You can bind data to Spreadsheet in following ways,

- Cell binding
- Range binding
- Sheet binding

Cell Binding

Spreadsheet can bind data for individual cells in a sheet. The data may contain value, style, format, comment and hyperlink. The individual cell properties are listed below,

Properties	Description
Index	To specify particular cell
Value	To specify value. It may be string, integer, formula etc.
Style	To specify style in the cell
Format	To specify number format in the cell
Comment	To specify comment in the cell
Hyperlink	To specify hyperlink in the cell

The individual row properties are listed below,

Properties	Description

Copyright © 2001 - 2020 Syncfusion Inc.

Index	To specify particular row
Height	To specify height in the row

You can specify particular row with Index property and its height with Height property in the Rows property collection. The following code illustrates cell binding in Spreadsheet,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ScrollSettings(scroll =>
{
scroll.Height(510);
})
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Height(30).Cells(cells =>
{
cells.Value("Item Name").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Value("Quantity").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Value("Price").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Value("Amount").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Value("Stock Details").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Value("Website").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("Casual Shoes").Comment(comment =>
{
comment.Value("Casual Footwears with wide variety of colors.");
}).Add();
cells.Value("10").Add();
cells.Value("20").Format(format =>
{
format.Type("currency");
```

Copyright © 2001 - 2020 Syncfusion Inc.

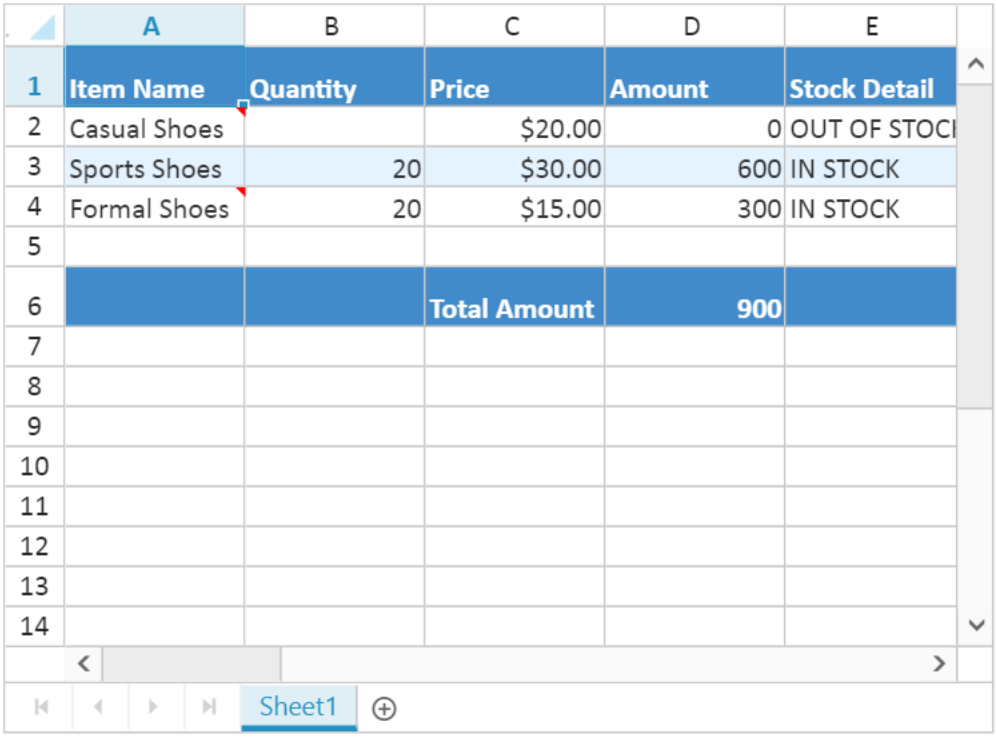
```
}).Add();
cells.Value("=B2*C2").Add();
cells.Value("OUT OF STOCK").Add();
cells.Value("Amazon").Hyperlink(hyperlink =>
{
hyperlink.WebAddr("www.amazon.com");
}).Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("Sports Shoes").Style(style =>
{
style.BackgroundColor("#E5F3FF");
}).Add();
cells.Value("20").Style(style =>
{
style.BackgroundColor("#E5F3FF");
}).Add();
cells.Value("30").Format(format =>
{
format.Type("currency");
}).Style(style =>
{
style.BackgroundColor("#E5F3FF");
}).Add();
cells.Value("=B3*C3").Style(style =>
{
style.BackgroundColor("#E5F3FF");
}).Add();
cells.Value("IN STOCK").Style(style =>
{
style.BackgroundColor("#E5F3FF");
}).Add();
cells.Value("AliExpress").Hyperlink(hyperlink =>
{
hyperlink.WebAddr("www.aliexpress.com");
}).Style(style =>
{
style.BackgroundColor("#E5F3FF");
}).Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("Formal Shoes").Comment(comment =>
{
comment.Value("Formal Footwears with wide range of sizes.");
}).Add();
cells.Value("20").Add();
cells.Value("15").Format(format =>
{
format.Type("currency");
}).Add();
cells.Value("=B4*C4").Add();
cells.Value("IN STOCK").Add();
cells.Value("Amazon").Hyperlink(hyperlink =>
{
hyperlink.WebAddr("www.amazon.com");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
}).Add();
rows.Height(30).Cells(cells =>
{
cells.Style(style =>
{
style.BackgroundColor("#428bca");
}).Add();
cells.Style(style =>
{
style.BackgroundColor("#428bca");
}).Add();
cells.Value("Total Amount").Index(2).Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Value("=Sum(D2:D4)").Style(style =>
{
style.FontWeight("bold").Color("#FFFFFF").BackgroundColor("#428bca");
}).Add();
cells.Style(style =>
{
style.BackgroundColor("#428bca");
}).Add();
cells.Style(style =>
{
style.BackgroundColor("#428bca");
}).Add();
}).Add();
}).Add();
})
)
```

The following output is displayed as a result of the above code snippets.

Copyright © 2001 - 2020 Syncfusion Inc.



Range Binding

Spreadsheet can bind data for one or more range in a sheet using RangeSettings. The individual range properties are listed below,

Properties	Description
DataSource	To specify JSON or DataManager
Query	To specify query for DataManager
StartCell	To specify start cell of a range
PrimaryKey	To specify data source primary key
ShowHeader	To show data source header
HeaderStyles	To specify header styles

The following code illustrates range binding in Spreadsheet

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).StartCell("C2")
.ShowHeader(true).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

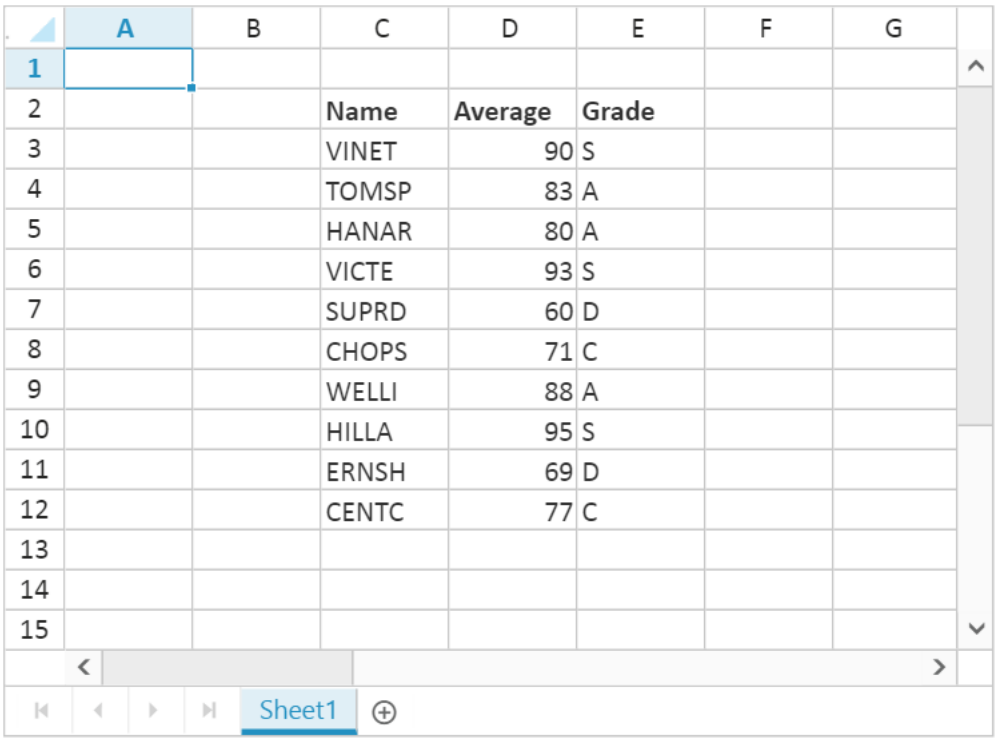
```
)
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<MarkDetail> lItems = new List<MarkDetail>();
lItems.Add(new MarkDetail() { Name = "VINET", Average = 90, Grade = "S"
});
lItems.Add(new MarkDetail() { Name = "TOMSP", Average = 83, Grade = "A"
});
lItems.Add(new MarkDetail() { Name = "HANAR", Average = 80, Grade = "A"
});
lItems.Add(new MarkDetail() { Name = "VICTE", Average = 93, Grade = "S"
});
lItems.Add(new MarkDetail() { Name = "SUPRD", Average = 60, Grade = "D"
});
lItems.Add(new MarkDetail() { Name = "CHOPS", Average = 71, Grade = "C"
});
lItems.Add(new MarkDetail() { Name = "WELLI", Average = 88, Grade = "A"
});
lItems.Add(new MarkDetail() { Name = "HILLA", Average = 95, Grade = "S"
});
lItems.Add(new MarkDetail() { Name = "ERNSH", Average = 69, Grade = "D"
});
lItems.Add(new MarkDetail() { Name = "CENTC", Average = 77, Grade = "C"
});
lItems.Add(new MarkDetail() { Name = "OTTIK", Average = 95, Grade = "S"
});
lItems.Add(new MarkDetail() { Name = "RATTC", Average = 85, Grade = "A"
});
lItems.Add(new MarkDetail() { Name = "FOLKO", Average = 90, Grade = "A"
});
lItems.Add(new MarkDetail() { Name = "BLONP", Average = 97, Grade = "S"
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code snippets.

Copyright © 2001 - 2020 Syncfusion Inc.



Sheet Binding

Spreadsheet can bind data for a sheet. The individual sheet properties are listed below,

Properties	Description
DataSource	To specify JSON or DataManager
Query	To specify query for DataManager
StartCell	To specify start cell of a range
PrimaryKey	To specify data source primary key
ShowHeader	To show data source header
HeaderStyles	To specify header styles
FieldAsColumnHeader	To show data source fields in column header

The following code illustrates sheet binding in Spreadsheet

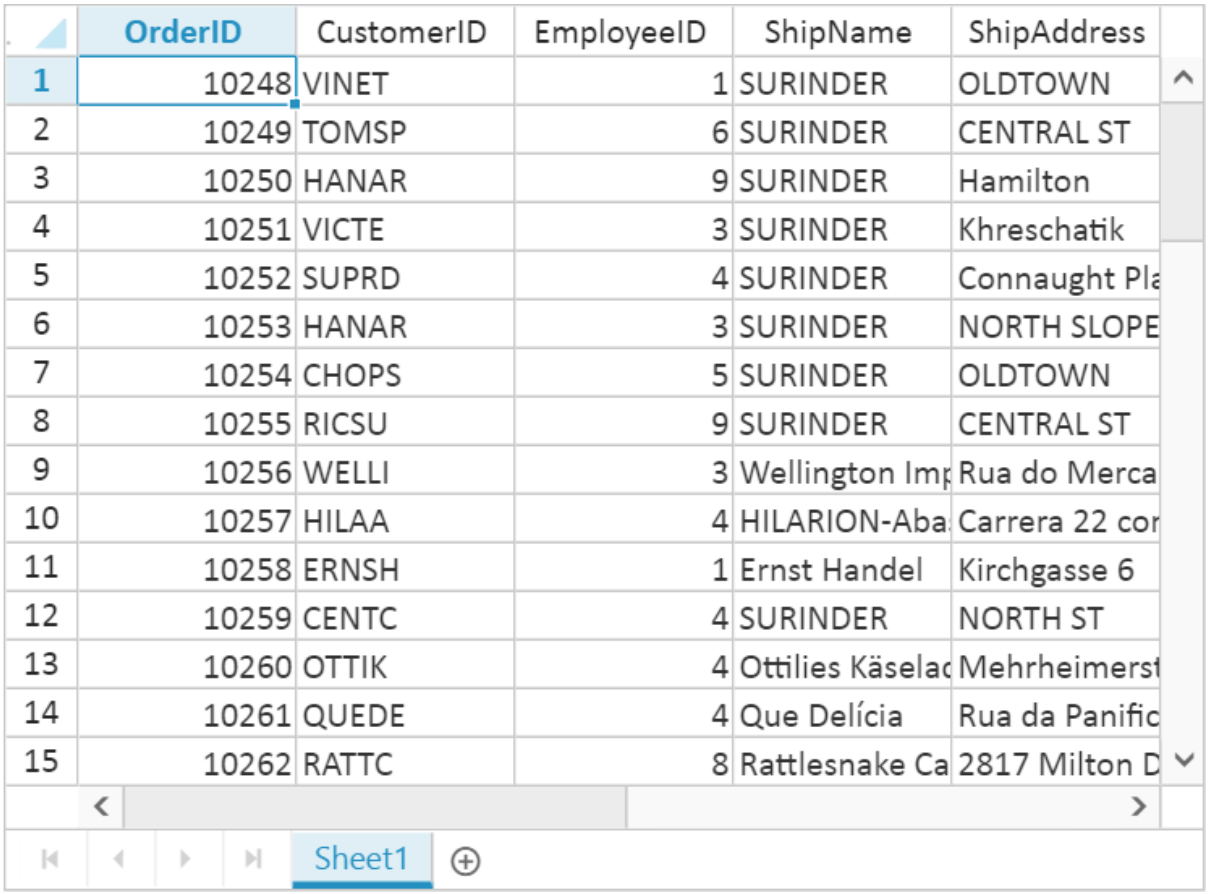
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource("http://mvc.syncfusion.com/Services/Northwnd.svc/Orders"
)
.Query("new ej.Query().select(['OrderID', 'CustomerID', 'EmployeeID',
'ShipCity', 'Freight'])
.take(50)").FieldAsColumnHeader(true).PrimaryKey("OrderID").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```

The following output is displayed as a result of the above code snippets.



	OrderID	CustomerID	EmployeeID	ShipName	ShipAddress
1	10248	VINET	1	SURINDER	OLDTOWN
2	10249	TOMSP	6	SURINDER	CENTRAL ST
3	10250	HANAR	9	SURINDER	Hamilton
4	10251	VICTE	3	SURINDER	Khreschatik
5	10252	SUPRD	4	SURINDER	Connaught Pla
6	10253	HANAR	3	SURINDER	NORTH SLOPE
7	10254	CHOPS	5	SURINDER	OLDTOWN
8	10255	RICSU	9	SURINDER	CENTRAL ST
9	10256	WELLI	3	Wellington Imp	Rua do Merca
10	10257	HILAA	4	HILARION-Aba	Carrera 22 cor
11	10258	ERNSH	1	Ernst Handel	Kirchgasse 6
12	10259	CENTC	4	SURINDER	NORTH ST
13	10260	OTTIK	4	Ottilies Käselac	Mehrheimerst
14	10261	QUEDE	4	Que Delícia	Rua da Panific
15	10262	RATTC	8	Rattlesnake Ca	2817 Milton D

Sheet1

Cell Range

A Cell Range is a collection of cells in a sheet. It represents single cell or selection of cells. When cells have been selected, they are surrounded by border.

You have following features in Cell Range,

- Comment
- Cell Navigation
- Data Validation
- Drag and Drop
- Auto Fill
- Hyperlink
- Merge Cell

Comment

Comment is used to give additional information for an individual cell about the data it contains. When a cell has a comment, a red indicator appears in the corner of the cell. When mouse hover on the cell, the comment will appear. You can use `AllowComments` property to enable/disable comments.

Copyright © 2001 - 2020 Syncfusion Inc.

To insert a Comment

You can insert a comment by using one of the following ways,

- Using "New" button under Comments group of REVIEW Tab in ribbon.
- Using context menu to select "Insert Comment" option in "Comment".
- Using setComment method.

To remove a Comment

You can remove a comment by using one of the following ways,

- Using "Delete" button under Comments group of REVIEW Tab in ribbon.
- Using context menu to select "Delete Comment" option in "Comment".
- Using deleteComment method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLComment.setComment("A2", " Casual Foot wears with wide variety of
colors.", false);
this.XLComment.setComment("A4", " Formal Foot wears with wide variety of
sizes.", true); // If true comment is in Editing mode.
//To Remove a Comment
this.XLComment.deleteComment ("A2");
}
}
</script>
```

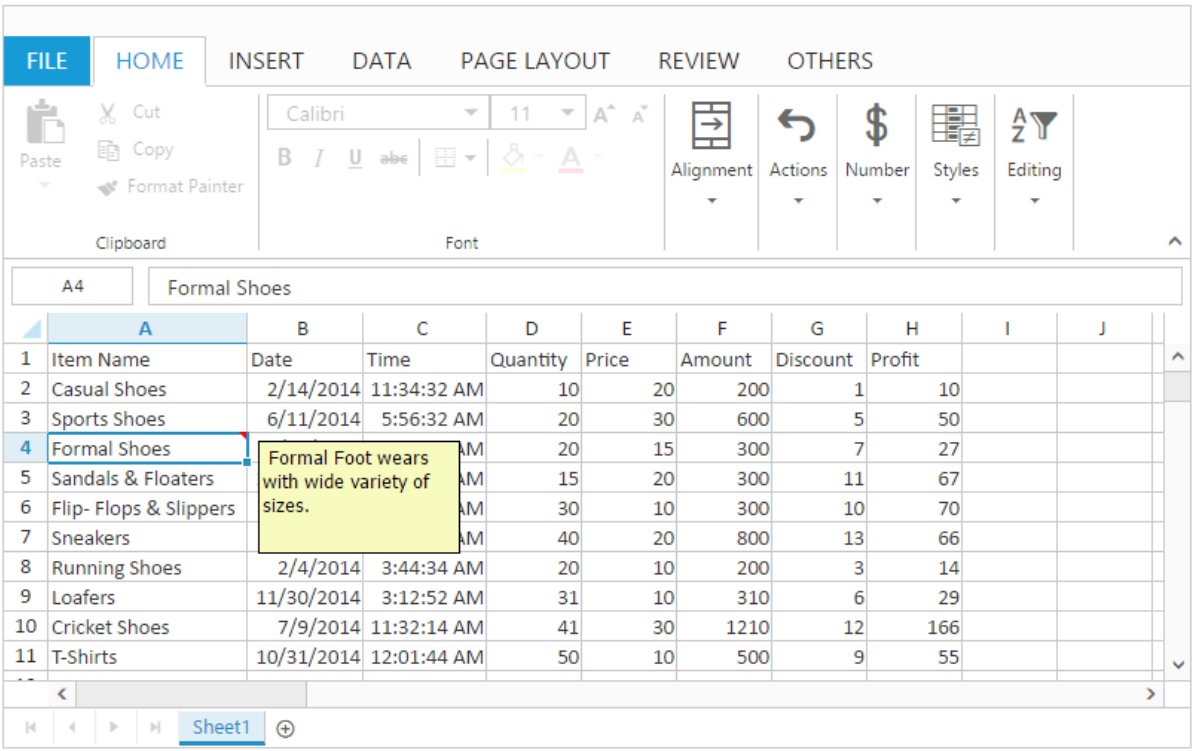
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Shoes		AM	20	15	300	7	27
5	Sandals & Floaters		AM	15	20	300	11	67
6	Flip- Flops & Slippers		AM	30	10	300	10	70
7	Sneakers		AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55

Cell Navigation

Cell navigation is used to navigate through the cells using keyboard. You can use `AllowKeyboardNavigation` property to enable/disable cell navigation. The following list of keys used for cell navigation are,

Keys	Description
Ctrl + Home	Go to first cell in the first row.

Copyright © 2001 - 2020 Syncfusion Inc.

Ctrl + End	Go to last cell in the last row.
Up Arrow	Go to previous cell in the same column
Down Arrow	Go to next cell in the same column
Right Arrow	Go to next cell in the same row
Left Arrow	Go to previous cell in the same row
Page Down	Go to next block
Page Up	Go to previous block
Home	Go to first cell of the same row
End	Go to last cell of the same row
Enter	Save the current cell changes and navigate to the next row.
Tab	Go to next cell
Shift + Tab	Go to previous cell
Shift + Up Arrow	Go to previous row or previous row cell with selection
Shift+ Down Arrow	Go to next row or next row cell with selection
Shift + Right Arrow	Go to next cell within the same row with selection
Shift + Left Arrow	Go to previous cell within the same row with selection
Ctrl + Up Arrow	Go to next cell from the first empty cell in upwards.
Ctrl + Down Arrow	Go to previous cell from the first empty cell in downwards.
Ctrl + Left Arrow	Go to next cell from the first empty cell in leftwards.
Ctrl+ Right Arrow	Go to previous cell from the first empty cell in rightwards.

Note: In the above table, "Ctrl + Home" and "Ctrl + End" is based on the used range.

Data Validation

Data Validation is used to restrict the user to enter the invalid data. You can use `AllowDataValidation` property to enable/disable data validation.

Note: The below validation script files are needed for validation.

* jquery.validate.min.js

* jquery.validate.unobtrusive.min.js

To apply validation.

You can apply data validation by using one of the following ways,

Copyright © 2001 - 2020 Syncfusion Inc.

- Using "Data Validation" option in Data Validation button under Data Tools group of DATA Tab in ribbon, Data Validation dialog will be opened in that you can perform Validation.
- Using applyDVRules method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLValidate.applyDVRules("G2:G12", ["Greater", 6], "number", true,
true);
//the last two boolean values used to ignore blank value and error alert.
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
```

To clear validation

You can clear data validation rule by one of the following ways,

- Using "Clear Validation" option in Data Validation button under Data Tools group of DATA Tab in ribbon.
- Using clearDV method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLValidate.applyDVRules("G2:G12", ["Greater", 6], "number", true,
true);
//the last two boolean values used to ignore blank value and error alert.
this.XLValidate.clearDV("G2:G12");
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

To format invalid data

You can highlight the invalid data by using following ways,

- Using "Format Invalid Data" option in Data Validation button under Data Tools group of DATA Tab in ribbon.
- Using highlightInvalidData method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLValidate.applyDVRules("G2:G12", ["Greater", 6], "number", true,
true);
//the last two boolean values used to ignore blank value and error alert.
this.XLValidate.highlightInvalidData ("G2:G12");
}
}
</script>
```

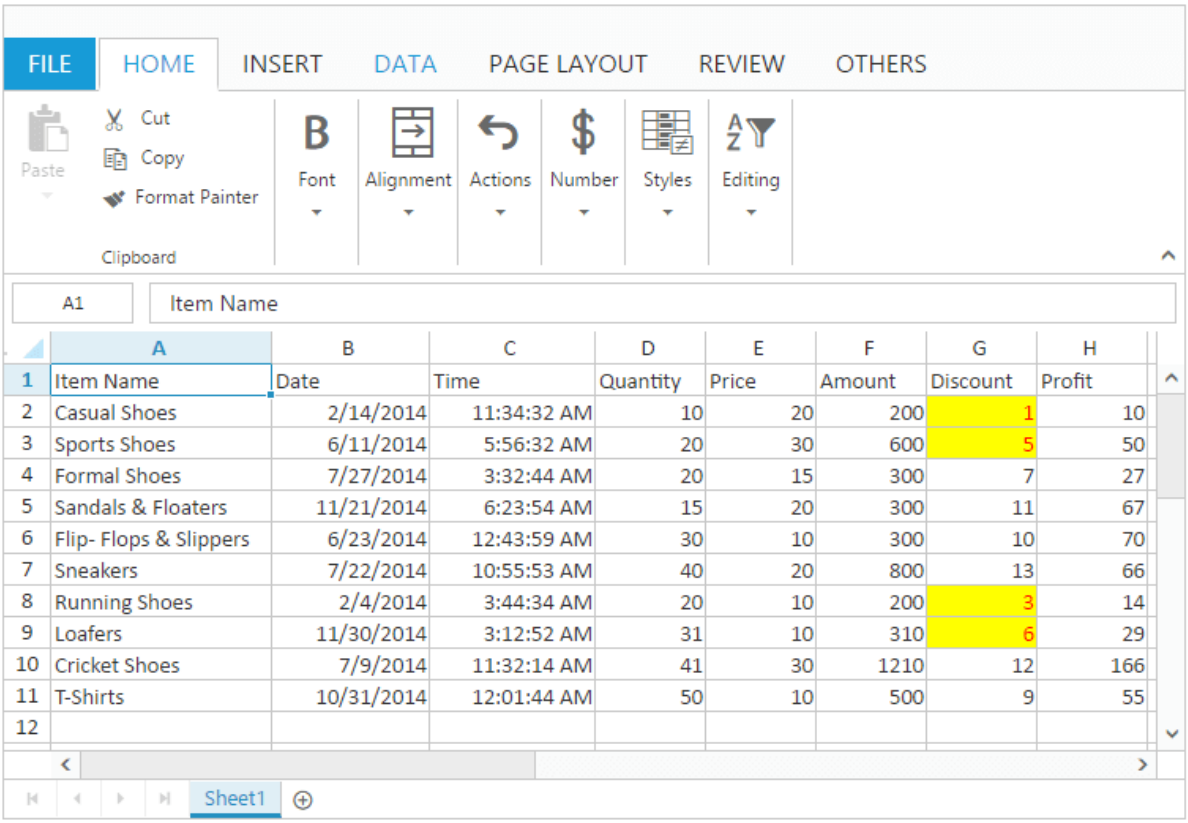
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Drag and Drop

Drag and drop is used to pick a selected cells and drop it into a new place on the worksheet. You can use `AllowDragAndDrop` property to enable/disable drag and drop.

Copyright © 2001 - 2020 Syncfusion Inc.

You can do this by one of the following ways,

- Using mouse drag and drop.
- Using moveRangeTo Method.

The following code example describes the above behavior.

ASPX-CS

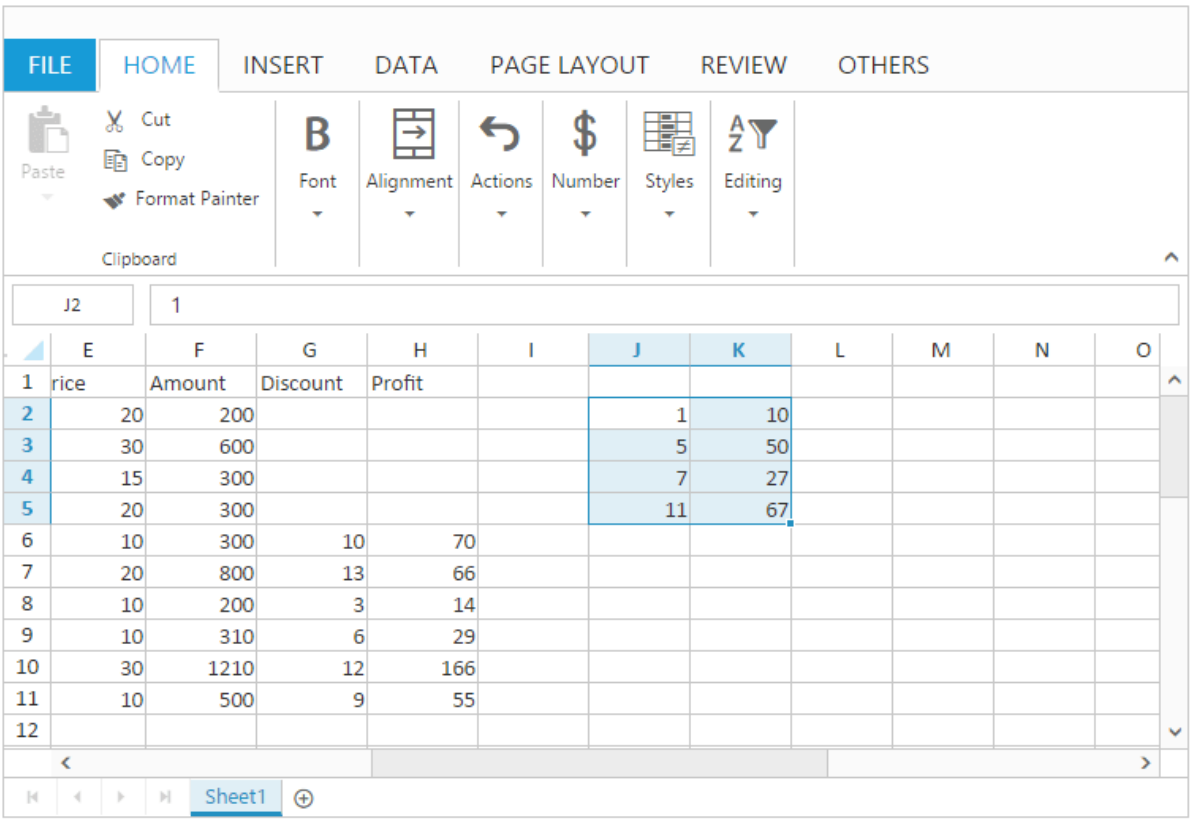
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport)
this.XLDragDrop.moveRangeTo([1, 6, 4, 7], [1, 9, 4, 10]);
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output is displayed as a result of the above code example.



	E	F	G	H	I	J	K	L	M	N	O
1	rice	Amount	Discount	Profit							
2	20	200				1	10				
3	30	600				5	50				
4	15	300				7	27				
5	20	300				11	67				
6	10	300	10	70							
7	20	800	13	66							
8	10	200	3	14							
9	10	310	6	29							
10	30	1210	12	166							
11	10	500	9	55							
12											

Auto Fill

Auto Fill is used to fill the cells with data based on adjacent cells. It also follows a pattern from adjacent cells if available. There is no need to enter the repeated data manually. You can use `AllowAutoFill` property to enable/disable the auto fill support. You can also use `ShowFillOptions` property to enable/disable the fill option and `FillType` property to change the default auto fill option which is available in `AutoFillSettings`.

You can do this by one of the following ways,

- Using “AutoFillOptions” menu which is open, while drag and drop the cell using fill handle element.
- Using autoFill method.

In Auto Fill we have following options,

- Copy Cells
- Fill Series
- Fill Formatting Only
- Fill Without Formatting
- Flash Fill

Copyright © 2001 - 2020 Syncfusion Inc.

Note: The default auto fill option is "fillSeries" which can be referred from FillType property.

Copy Cells

To copy the selected cell content to the adjacent cells. You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and "Copy Cells" option in "AutoFillOptions" menu to fill the adjacent cells.
- Using "copyCells" as fill type in autoFill method to fill the adjacent cells.

Fill Series

To fill the series of numbers, characters, or dates based on selected cell content to the adjacent cells with their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and "Fill Series" option in "AutoFillOptions" menu to fill the adjacent cells.
- Using "fillSeries" as fill type in autoFill method to fill the adjacent cells.

Fill Formatting Only

To fill the cell style and number formatting based on the selected cell content to the adjacent cells without their content.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and "Fill Formatting Only" option in "AutoFillOptions" menu to fill the adjacent cells.
- Using "fillFormattingOnly" as fill type in autoFill method to fill the adjacent cells.

Fill Without Formatting

To fill series of numbers, characters, or dates based on the selected cells to the adjacent cells without their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and "Fill Without Formatting" option in "AutoFillOptions" menu to fill the adjacent cells.
- Using "fillWithoutFormatting" as fill type in autoFill method to fill the adjacent cells.

Flash Fill

To fill the column when it senses a pattern from adjacent column data based on what you type.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and "Flash Fill" option in "AutoFillOptions" menu to fill the adjacent cells.
- Using "flashFill" as fill type in autoFill method to fill the adjacent cells.

The following code example describes the above behavior.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(rangeSettings =>
{
rangeSettings.Datasource((IEnumerable<ItemDetail>)ViewBag.Datasource).Add
();
rangeSettings.Datasource((IEnumerable<SampleData>)ViewBag.SampleData).Sta
rtCell("I1").Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLEdit.updateValue("N2", "Name");
this.XLEdit.updateValue("N3", "Casual Shoes");
this.XLEdit.updateValue("N4", "Formal Shoes");
this.XLEdit.updateValue("N5", "Sports Shoes");
this.XLEdit.updateValue("O2", "FirstName");
this.XLEdit.updateValue("O3", "Casual");
this.XLFormat.format({ "style": { "background-color": "yellow" } },
"K2:L2");
this.XLFormat.format({ "style": { "background-color": "red" } },
"K4:L4");
this.XLFormat.format({ "style": { "background-color": "blue" } },
"K5:L5");
//copy Cells
this.XLDragFill.autoFill({ sheetIdx: 1, dataRange: [1, 8, 4, 8],
fillRange: "I6:I10", fillType: "copycells", direction:"down" });
//fill series
this.XLDragFill.autoFill({ sheetIdx: 1, dataRange: [1, 9, 4, 9],
fillRange: "J6:J10", fillType: "fillseries", direction: "down" });
//fill formatting only
this.XLDragFill.autoFill({ sheetIdx: 1, dataRange: [1, 10, 4, 10],
fillRange: "K6:K10", fillType: "fillformattingonly", direction: "down"
});
//fill without formatting
this.XLDragFill.autoFill({ sheetIdx: 1, dataRange: [1, 11, 4, 11],
fillRange: "L6:L10", fillType: "fillwithoutformatting", direction: "down"
});
//flash fill
this.XLDragFill.autoFill({ sheetIdx: 1, dataRange: [2, 14, 2, 14],
fillRange: "O4:O6", fillType: "flashfill", direction: "down" });
}
}
</script>
```

C#

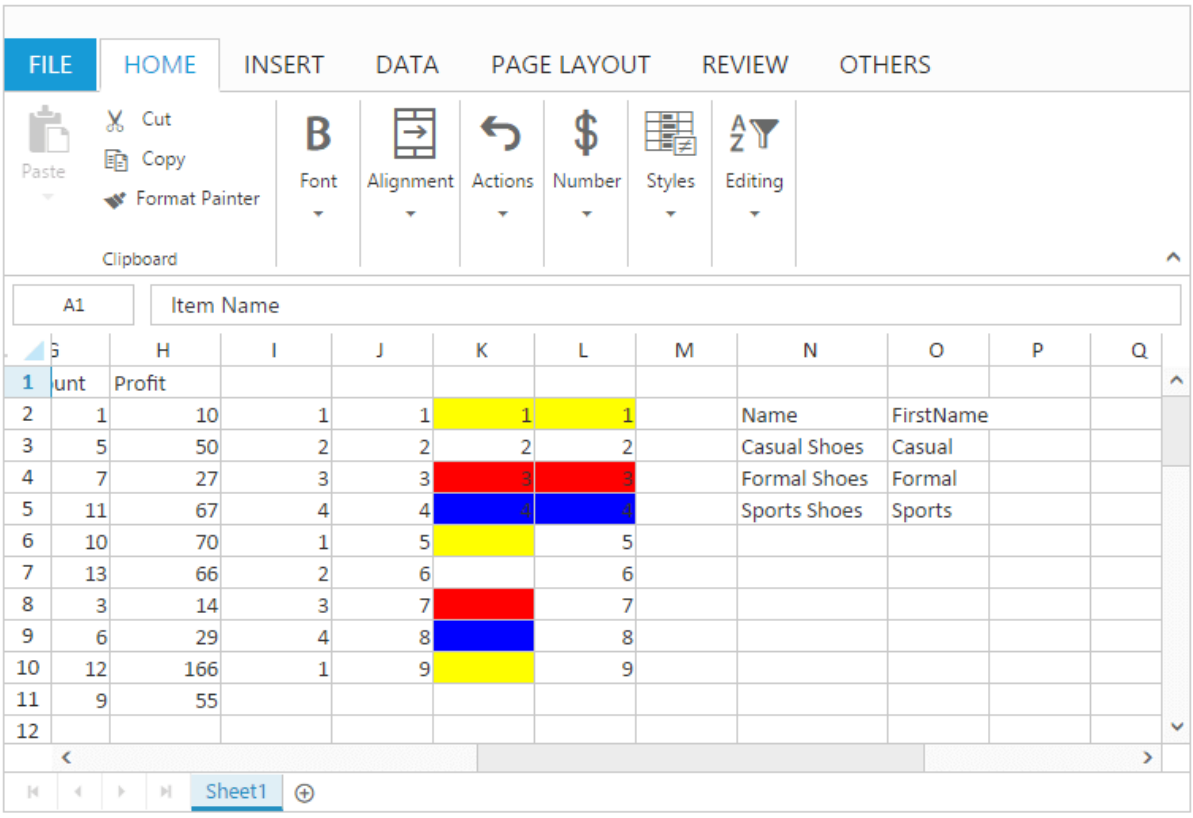
```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
List <SampleData> sampleData = new List<SampleData>();
sampleData.Add(new SampleData() { i = 1, j=1, k=1,l=1});
sampleData.Add(new SampleData() { i = 2, j = 2, k = 2, l = 2 });
sampleData.Add(new SampleData() { i= 3, j= 3, k= 3, l= 3});
sampleData.Add(new SampleData() { i= 4, j= 4, k= 4, l= 4 });
ViewBag.SampleData = sampleData;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Hyperlink

Hyperlink is used to navigate to web links or cell reference within the sheet or to other sheets in Spreadsheet. You can use AllowHyperlink property to enable/disable Hyperlink.

To insert a Hyperlink

You can insert a hyperlink by one of the following ways,

- Using "Hyperlink" button under Links group of INSERT Tab in ribbon.
- Using context menu to select "Insert Hyperlink" option in "Hyperlink".
- Using setHyperlink method.

To remove a Hyperlink

You can remove a hyperlink by one of the following ways,

- Using context menu to select "Remove Hyperlink" option in Hyperlink.
- Using removeHyperlink method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

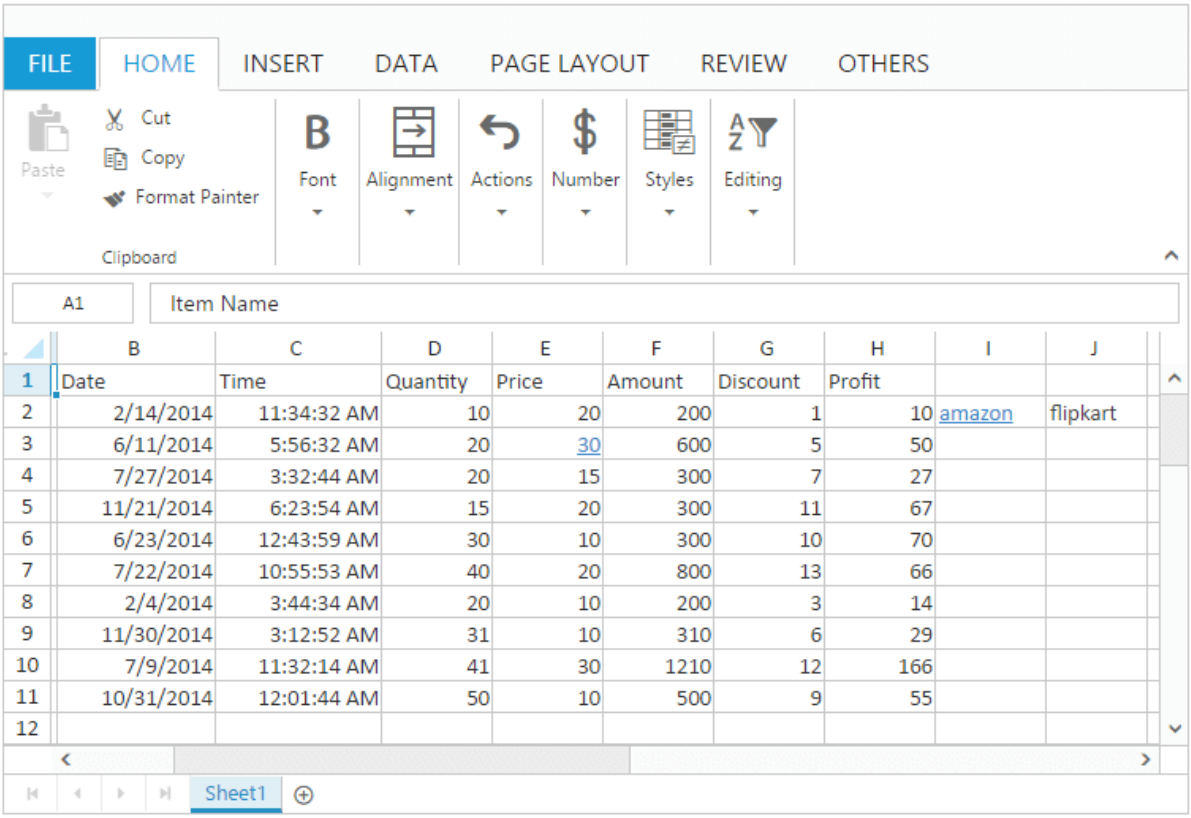
```
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLEdit.updateValue("I2", "amazon");
this.XLEdit.updateValue("J2", "flipkart");
this.setHyperlink("E3:E3", { "cellAddr": "A1:D2" }, 2);
this.setHyperlink("I2:I2", { "webAddr": "http://www.amazon.com" }, 1);
this.setHyperlink("J2:J2", { "webAddr": "http://www.flipkart.com" }, 1);
//To Remove a Hyperlink
this.removeHyperlink("J2:J2");
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	B	C	D	E	F	G	H	I	J
1	Date	Time	Quantity	Price	Amount	Discount	Profit		
2	2/14/2014	11:34:32 AM	10	20	200	1	10	amazon	flipkart
3	6/11/2014	5:56:32 AM	20	30	600	5	50		
4	7/27/2014	3:32:44 AM	20	15	300	7	27		
5	11/21/2014	6:23:54 AM	15	20	300	11	67		
6	6/23/2014	12:43:59 AM	30	10	300	10	70		
7	7/22/2014	10:55:53 AM	40	20	800	13	66		
8	2/4/2014	3:44:34 AM	20	10	200	3	14		
9	11/30/2014	3:12:52 AM	31	10	310	6	29		
10	7/9/2014	11:32:14 AM	41	30	1210	12	166		
11	10/31/2014	12:01:44 AM	50	10	500	9	55		
12									

Merge Cell

Merge cell is a single cell created by combining two or more individual cells together. You can use AllowMerging property to enable/disable merge cells. The cell reference for a merged cell is the cell in the upper left corner of the original selected range or group of cells.

You have following options in Merge Cell,

- Merge Cells
- Merge & Center
- Merge Across

Merge Cells

You can combine two or more cells located in the same row or column into a single cell. When cells with multiple values are merged, upper-left most cell's data will be the data of merged cell.

You can do this by one of the following ways,

- Using "Merge Cells" option in Merge & Center button under Alignment group of HOME Tab in ribbon.
- Using mergeCells method.

Merge & Center

You can combine two or more cells located in the same row or column into a single cell with center text align. When cells with multiple values are merged, upper-left most cell's data will be the data of the merged cell. You can do this by one of the following ways,

Copyright © 2001 - 2020 Syncfusion Inc.

- Using Merge & Center button under Alignment group of HOME Tab in ribbon.
- Using "Merge & Center" option in Merge & Center button under Alignment group of HOME Tab in ribbon.
- Using mergeCells method with mergeCenter property as true to enable the center alignment.

Merge Across

You can combine two or more cells located in the same row into a single cell. When cells with multiple values are merged, left most cell's data will be the data of the merged cell.

You can do this by one of the following ways,

- Using "Merge Across" option in Merge & Center button under Alignment group of HOME Tab in ribbon.
- Using mergeAcrossCells method.

The following code example describes the behavior of Merge Cells, Merge & Center and Merge Across.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.mergeCells("A2:B4", true);
this.mergeCenter = true;
this.mergeCells("D2:E4", true); // true is to prevent the alert message.
this.mergeAcrossCells("G2:H4", true); // true is to prevent the alert
message.
}
}
</script>
```

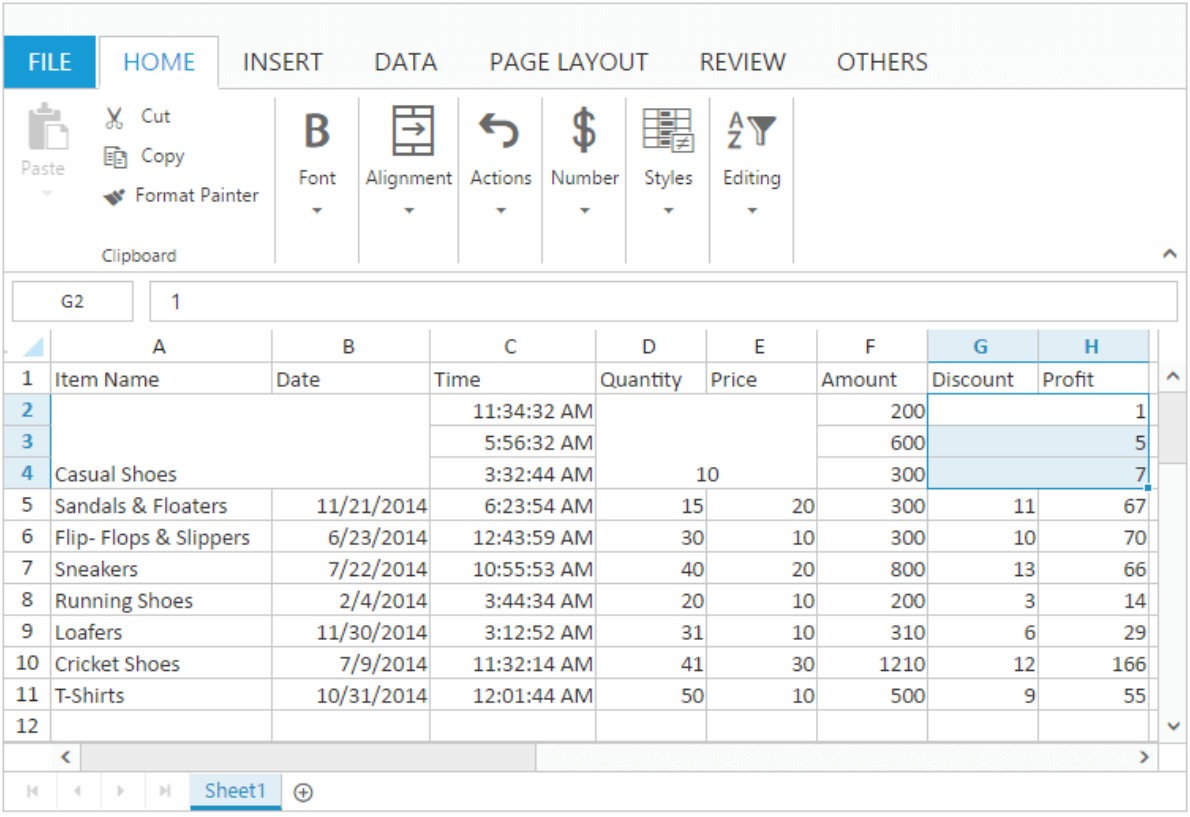
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2			11:34:32 AM			200		1
3			5:56:32 AM			600		5
4	Casual Shoes		3:32:44 AM	10		300		7
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Unmerge Cells

You can split the merged cell into multiple cells. You can do this by one of the following ways,

- Using "Unmerge cells" option in Merge & Center button under Alignment group of HOME Tab in ribbon.
- Using unMergeCells method.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

ASPX-CS

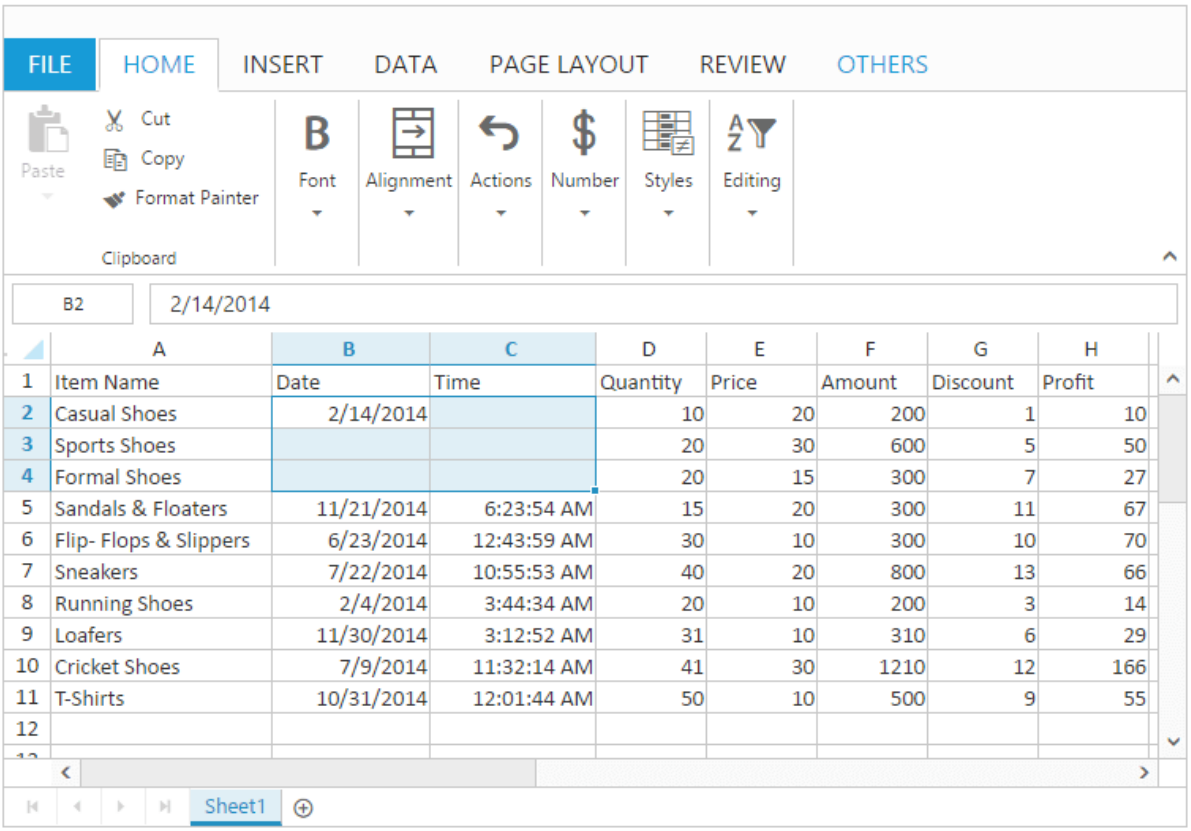
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.mergeCells("D2:E4", true); // true is to prevent the alert message.
this.unmergeCells("D2:E4");
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014		10	20	200	1	10
3	Sports Shoes			20	30	600	5	50
4	Formal Shoes			20	15	300	7	27
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Editing

You can edit the contents of a cell directly in the cell. You can also do this by typing in the formula bar. When you edit the cell, Spreadsheet is operating in edit mode. In this mode some of the options are unavailable. You can use `AllowEditing` property to enable/disable editing feature.

Note: By default `AllowEditing` property is set to true.

Edit cell content

You can perform this by one of the following ways,

- Double click on the cell to perform editing. This starts the edit mode and positions the cursor at the end of the cell.
- Press F2 Key to edit the active cell.
- Use Formula bar to perform editing.
- Use Backspace and Delete Key to delete the contents of a cell.
- Use "Alt + Enter" keys to perform multi line editing.
- Using editCell method.

Save cell content

You can do this by one of the following ways,

- Perform mouse click on any other cell other than the current editing cell.
- Perform Enter/Tab key press on the cell.
- Using saveCell method.

Copyright © 2001 - 2020 Syncfusion Inc.

Note: Edited cells are automatically formatted (right/left/center/Number Formatting) based on cell values.

The following code example describes the above behavior.

ASPX-CS

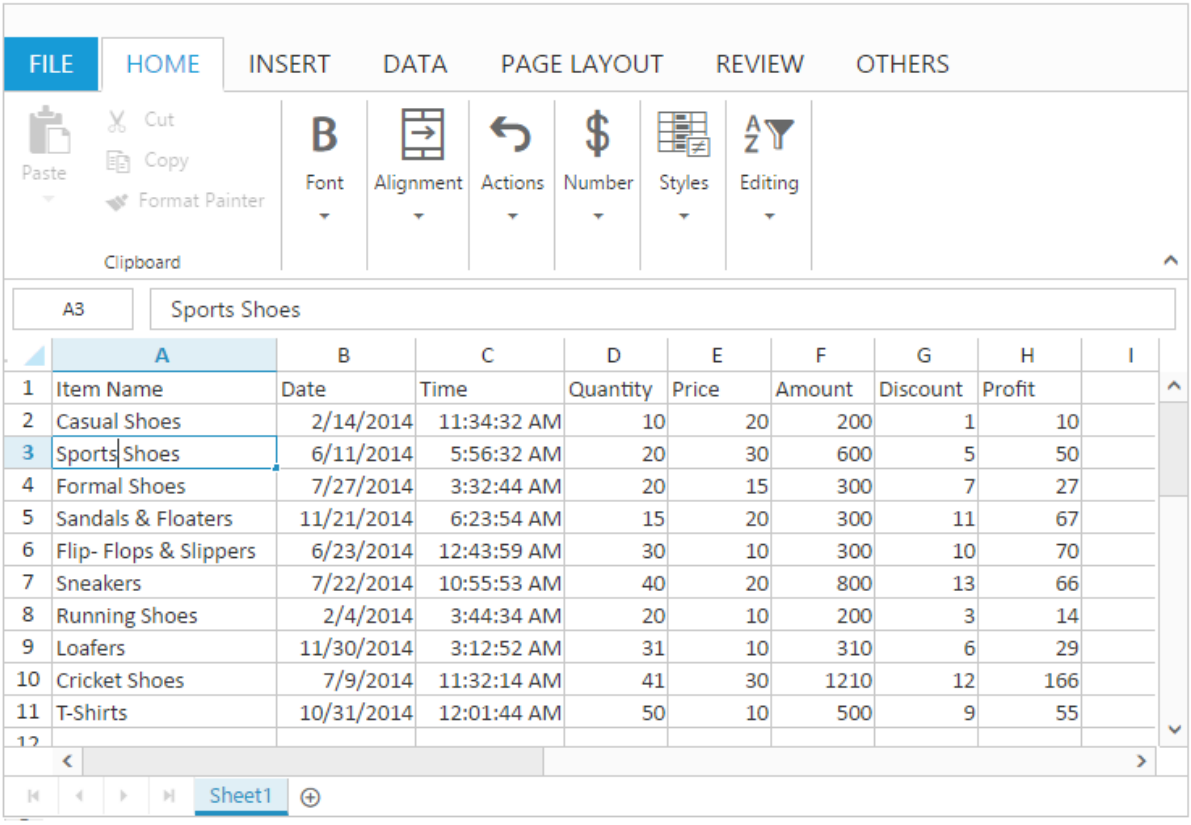
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLEdit.editCell(3, 0, true); //if true, it maintains the existing
data otherwise it clears the data.
//this.XLEdit.saveCell();
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H	I
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit	
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10	
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50	
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27	
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67	
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70	
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66	
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14	
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29	
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166	
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55	
12									

Read-Only Cells

You can restrict/prevent the editing in the specified range. You can use AllowLockCell property to enable/disable the lock cells. You can do this by one of the following ways,

- Using lockCells method to lock the selected cells. Then you need to protect the sheet using protectSheet method to restrict the editing.
- Using "Lock Cells" option under Changes group of REVIEW tab in ribbon. Then Using "Protect Sheet" option under Changes group of REVIEW tab in ribbon to restrict editing.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowLockCell(true)
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.protectSheet(false);
this.lockCells("A1:A5", true);
this.protectSheet(true);
```

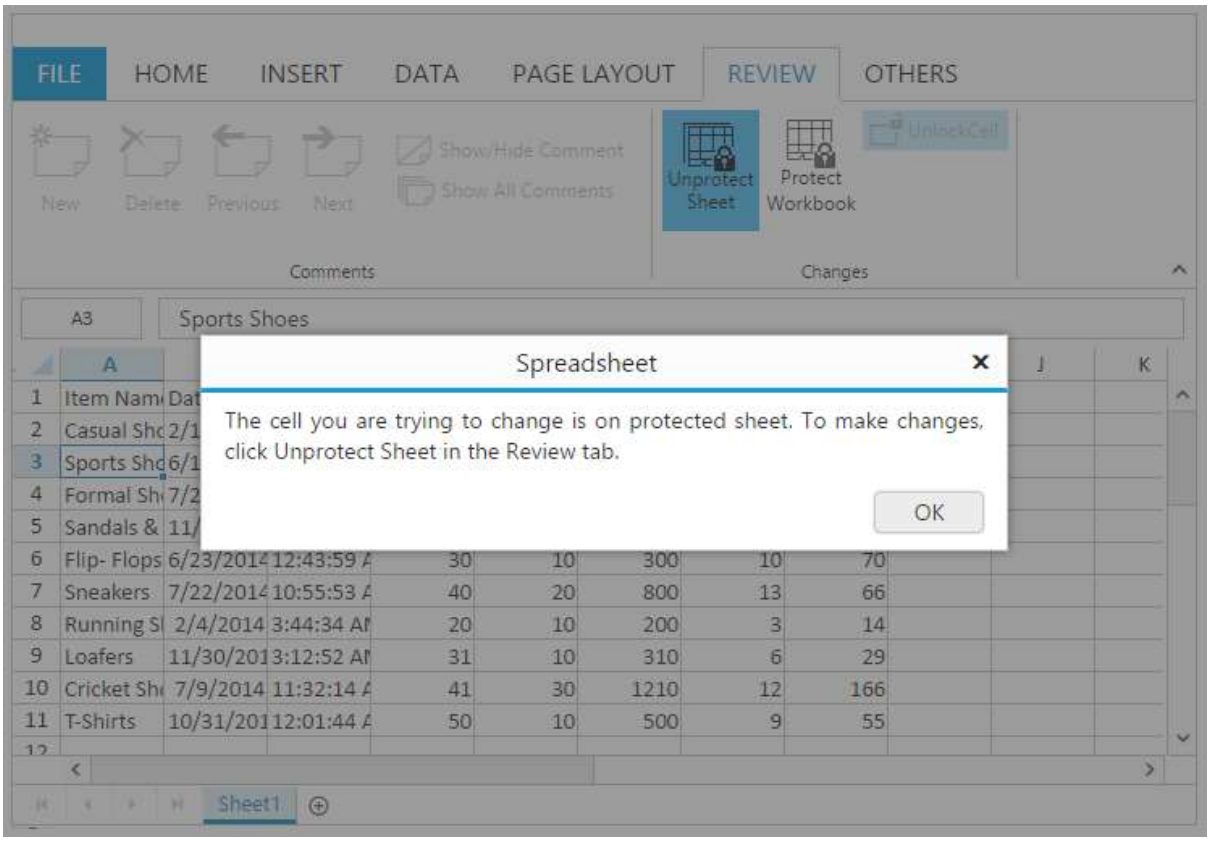
```
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Events

The following events will trigger when editing and saving the cell.

- cellEdit - Triggered when the cell is edited.
- cellSave - Triggered when save the edited cell.

Data binding

You can bind the data to Spreadsheet using data manager. You can refer Data Binding to know more about this. You can use saveBatchChanges method to update the changes in server.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().DataManager("DataManager").URL("Default").Adaptor(AdaptorType
.UrlAdaptor).BatchURL("BatchUpdate"))
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.DataManagerID("DataManager").PrimaryKey("ItemName").Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLEdit.updateValue("I2", "amazon");
```

```
this.XLEdit.updateValue("J2", "flipkart");
this.saveBatchChanges(this.getActiveSheetIndex());
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
public ActionResult BatchUpdate( List<ItemDetail> changed,
List<ItemDetail> added,  List<ItemDetail> deleted, string action, string
key)
{
//Save the batch changes
}
}
```

To save and retrieve the Spreadsheet data in the database, you can refer this Knowledge Base link.

You can update data dynamically in the Spreadsheet by using updateRange method.

Selection

Selection provides an interactive support to highlight cell, row, or column that you select. Selection can be done through Mouse, Touch or Keyboard interaction. To enable selection, set `AllowSelection` as `true`.

Copyright © 2001 - 2020 Syncfusion Inc.

Selection Modes

The mode of selection can be single or range selection. The default selection mode can be set using SelectionUnit in SelectionSettings.

The two types of selection mode are as follows

- Single - SpreadsheetSelectionUnit.Single
- Range - SpreadsheetSelectionUnit.Range

Selection Type

There are four types of selection in Spreadsheet,

1. Cell Selection
2. Row Selection
3. Column Selection
4. Sheet Selection

You can set default SelectionType in SelectionSettings.

Cell Selection

Cell selection is used to select a single or multiple cells. It can be performed using selectRange method.

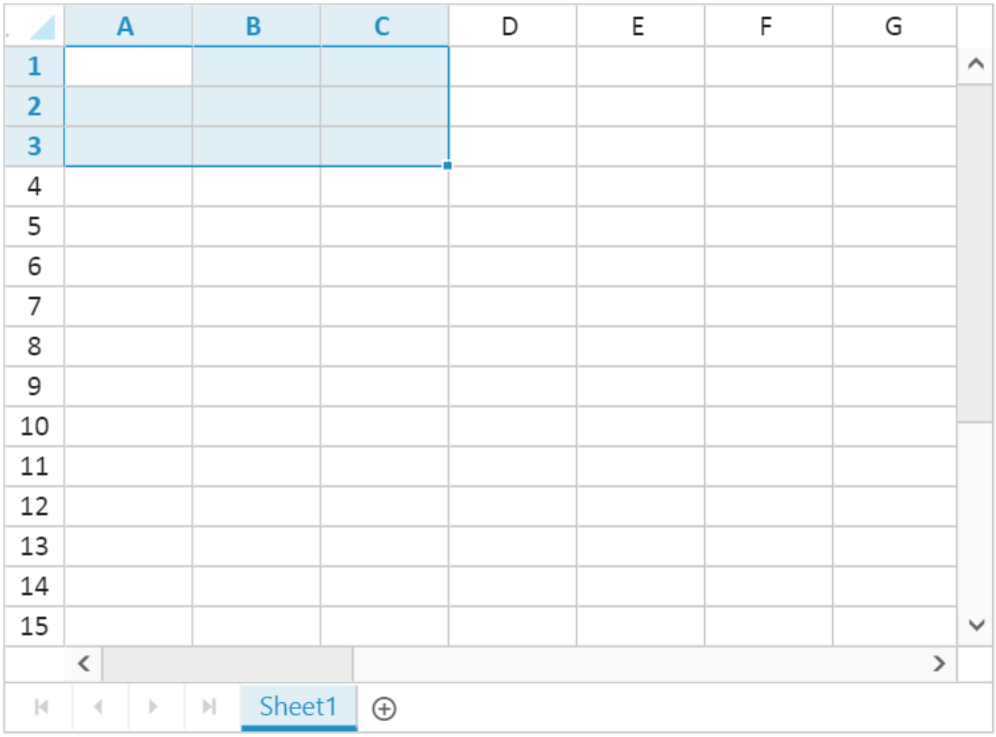
The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLSelection.selectRange("A1:C3");
this.XLDragFill.positionAutoFillElement();
}
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Row Selection

Row selection is used to select a single or multiple rows. It can be performed using selectRows method.

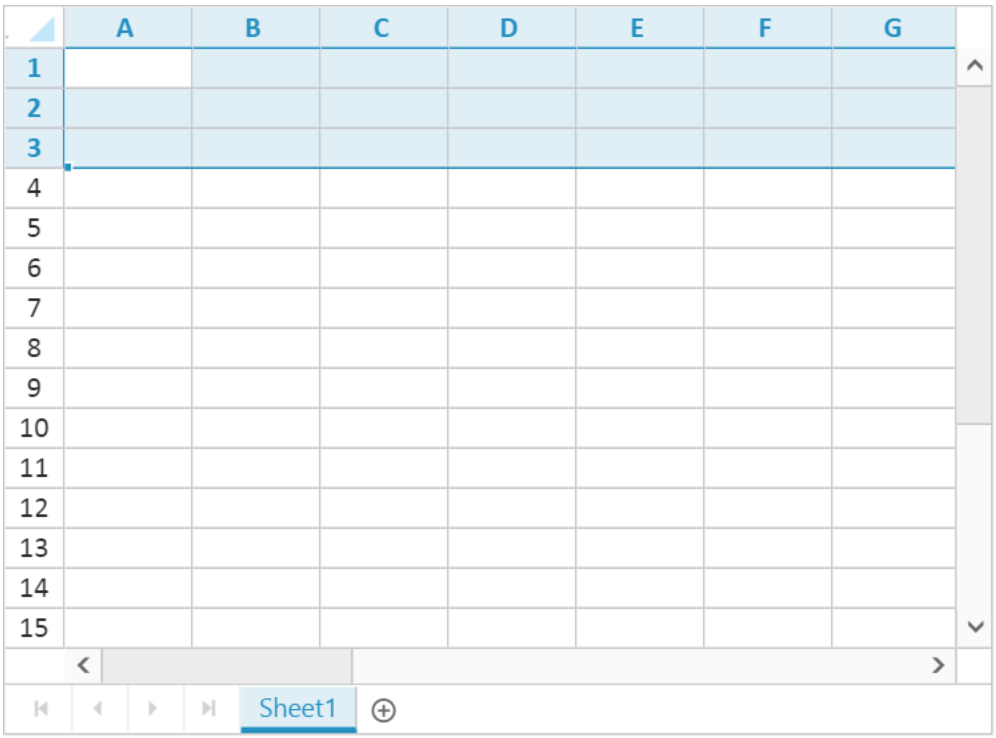
The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLSelection.selectRows(0,2);
this.XLDragFill.positionAutoFillElement();
}
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: This type can be set as default by setting SelectionType property in SelectionSettings as SpreadsheetSelectionType.Row

Column Selection

Column selection is used to select a single or multiple columns. It can be performed using selectColumns method.

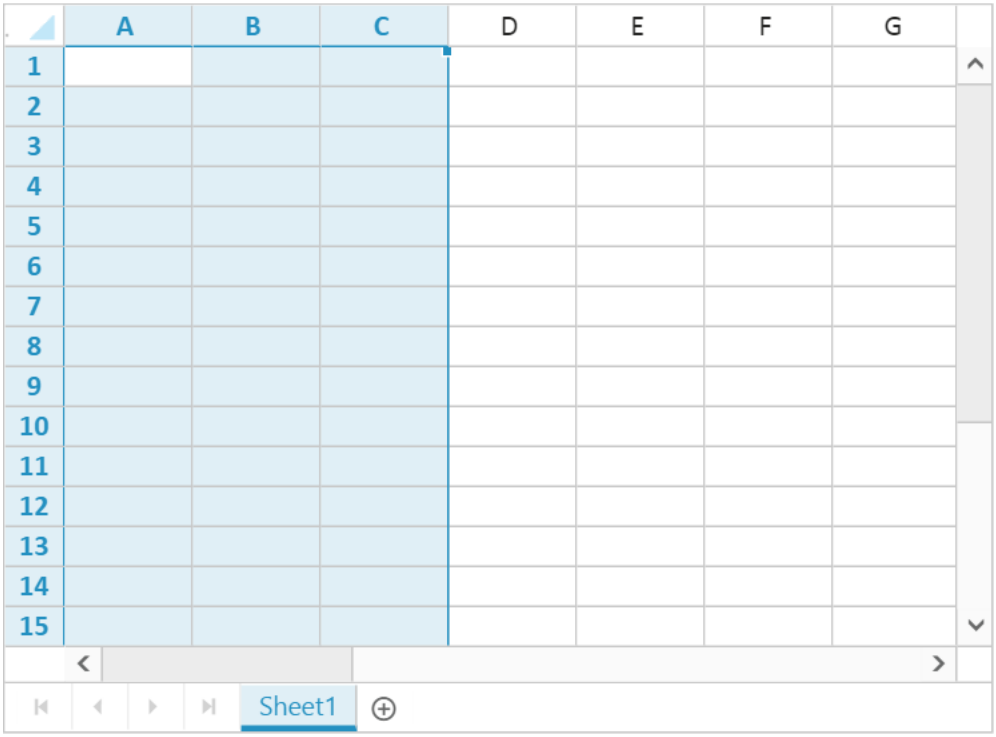
The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport) {
this.XLSelection.selectColumns(0,2);
this.XLDragFill.positionAutoFillElement();
}
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: This type can be set as default by setting SelectionType property in SelectionSettings as SpreadsheetSelectionType.Column

Sheet Selection

Sheet selection is used to select all cells in a worksheet. It can be performed using selectSheet method.

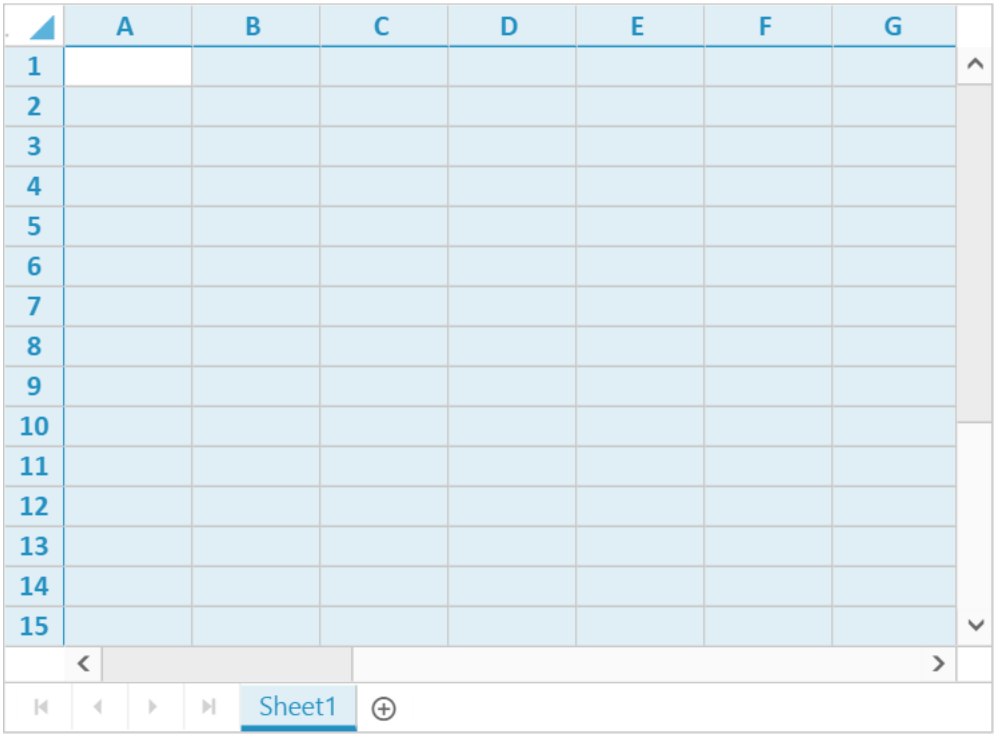
The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if(!this.isImport)
this.XLSelection.selectSheet();
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Context Menu

Context Menu is used to improve user interaction with Spreadsheet using popup menu. This will open when right clicking on Cell/Column Header/Row Header/Pager in Spreadsheet. You can use `EnableContextMenu` property to enable/disable context menu.

Default Context Menu items

Please find the below table for default context menu items and its actions.

Section	Context Menu items	Action
Row Cell	Cut	Cut the selected cells content to the clipboard, so that you can paste it somewhere else.
	Copy	Copy the selected cells content to the clipboard, so that you can paste it to somewhere else.
	Paste	Paste the content from clipboard to spreadsheet.
	Insert	Insert new cells or rows or columns to worksheet.
	Delete	Delete existing cells or rows or columns from worksheet.
	Sort	Perform sorting to the selected range of cells by ascending or descending.

Copyright © 2001 - 2020 Syncfusion Inc.

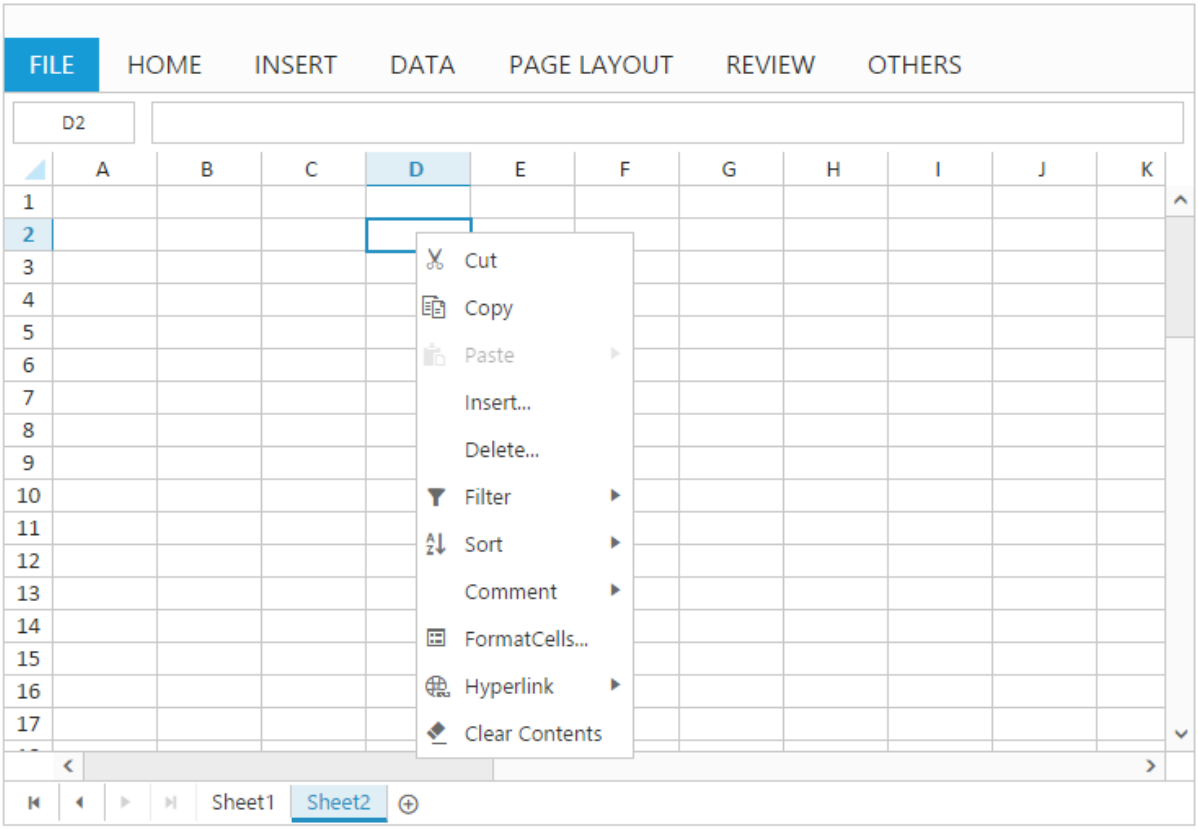
	Filter	Perform filtering to the selected cells based an active cell's value or active cell's color.
	Hyperlink	Create a link in the spreadsheet for quick access to webpages or worksheet reference.
	Comment	Add a note or extra information about the active cell.
	Format Cells	Apply number format to the selected cells.
	Clear Contents	Delete the contents in the selected cells.
Row Header / Column Header	Cut	Cut the selected cells content to the clipboard, so that you can paste it to somewhere else.
	Copy	Copy the selected cells content to the clipboard, so that you can paste it to somewhere else.
	Paste	Paste the content from clipboard to spreadsheet.
	Clear Contents	Delete the contents in the selected cells.
	Insert	Insert new cells or rows or columns to worksheet.
	Delete	Delete existing cells or rows or columns from worksheet.
	Hide	Hide the selected columns/rows.
	Unhide	Unhide the selected columns/rows.
Pager	Insert	Insert new worksheet to spreadsheet.
	Delete	Delete the selected worksheet from spreadsheet.
	Move or copy	Opens move or copy dialog to move worksheet or to create duplicate worksheet.
	Rename	Rename the selected worksheet.
	Protect Sheet	Prevent unwanted changes from others by limiting their ability to edit.
	Hide	Hide the selected worksheet.
	Unhide	Opens unhide dialog to unhide worksheet.

The following code example describes the above behavior.

ASPX-CS

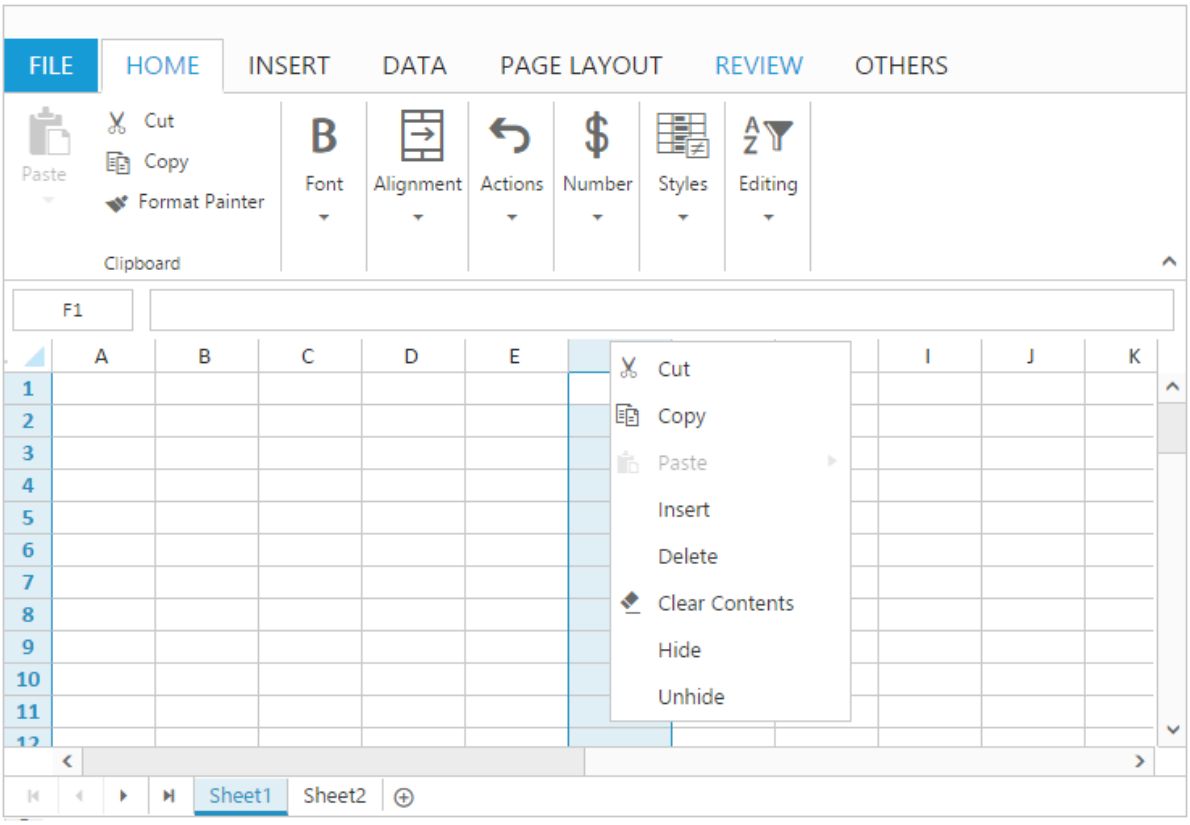
Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.EnableContextMenu(true)
)
```

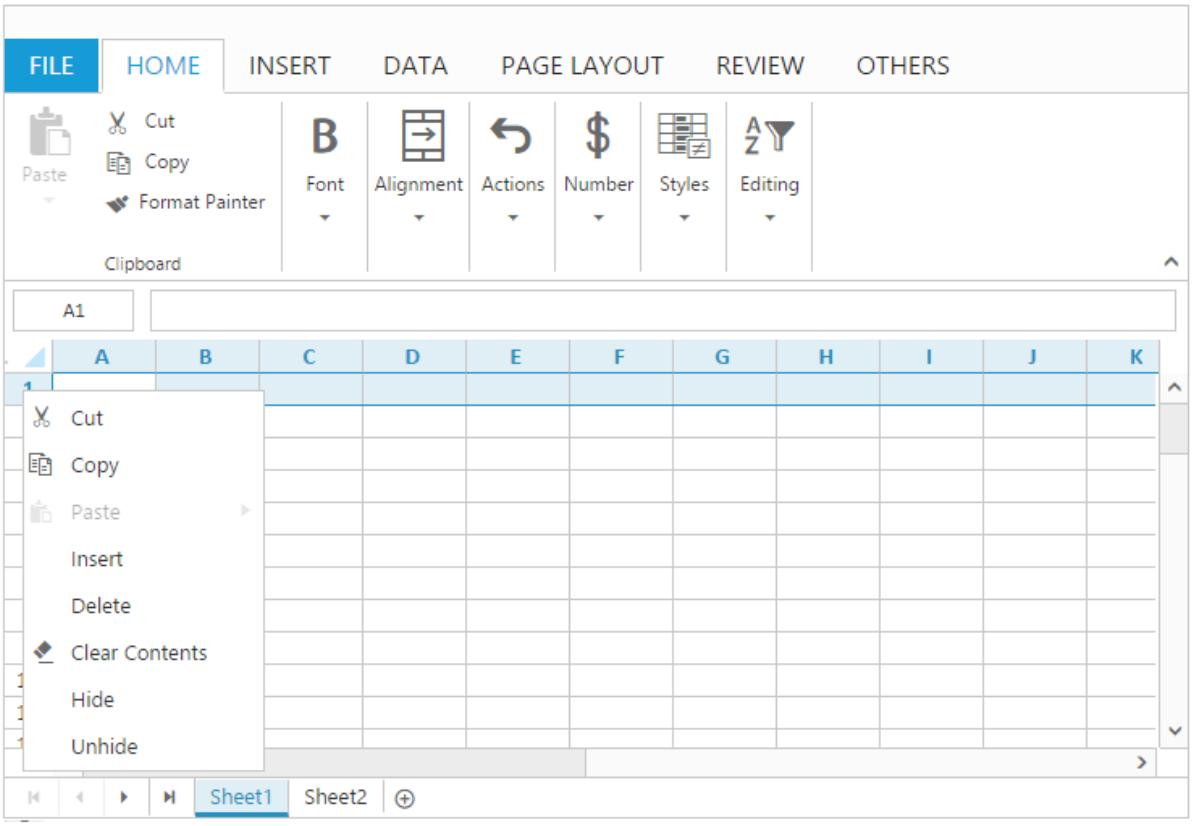


Context menu at Cell

Copyright © 2001 - 2020 Syncfusion Inc.

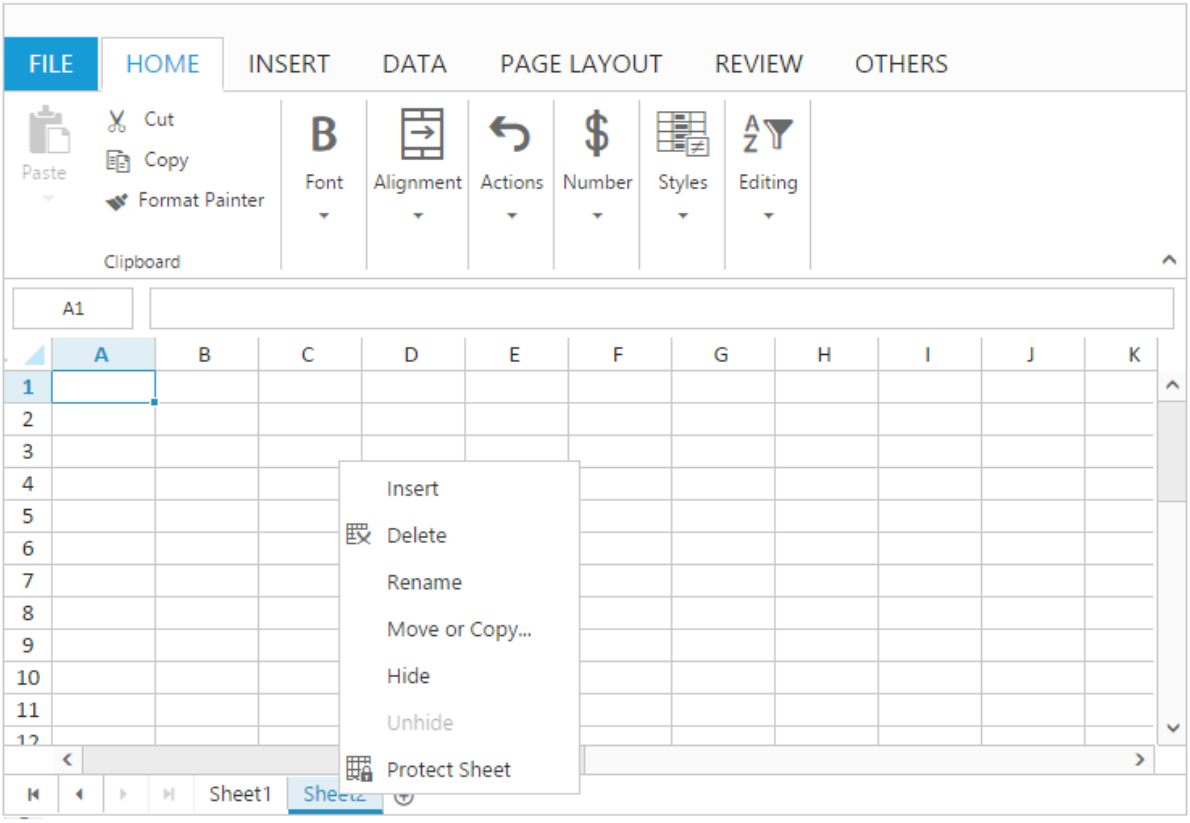


Context menu at Column Header



Context menu at Row Header

Copyright © 2001 - 2020 Syncfusion Inc.



Context menu at Pager

Data Presentation

Data presentation is helpful for proper representation of data in Spreadsheet. You have following features in Data Presentation.

- Cell types
- Chart
- Conditional formatting
- Filtering
- Picture
- Pivot table
- Sorting
- Table

Cell Type

You can insert the controls like Button, Checkbox, Dropdown list and Date picker. You can use `AllowCellType` property to enable/disable cell type operations.

To insert a Cell Type.

You can insert the cell type to the selected range of cells by one of the following ways

- Using `CellTypes` property in sheets.
- Using addCellTypes method.

To remove Cell Type

You can delete the cell type in the selected range of cells by using removeCellTypes method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowCellType(true)
.Sheets(sheet =>
{
sheet.Datasource((IEnumerable<object>)ViewBag.Datasource)
.CellTypes(cellType =>
{
cellType.Range("A1").Setting(setting =>
{
setting.Type(CustomCellTypes.DropDownList).DataSourceRange("A2:A11");
}).Add();
cellType.Range("D1").Setting(setting =>
{
setting.Type(CustomCellTypes.Button).Text("BUTTON").BackgroundColor("gree
n");
}).Add();
cellType.Range("C1").Setting(setting =>
{
setting.Type(CustomCellTypes.DropDownList).DataSourceRange("A2:A11");
}).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var excelCellType = this.XLCellType;
if (!this.isImport) {
excelCellType.addCellTypes("B1", {"type" :
ej.Spreadsheet.CustomCellType.DatePicker, 'value' : '2/12/2016'},  1);
excelCellType.addCellTypes("E1", {"type" :
ej.Spreadsheet.CustomCellType.CheckBox, "isChecked" : true },  1);
this.XLCellType.removeCellTypes("C1");
}
}
</script>
```

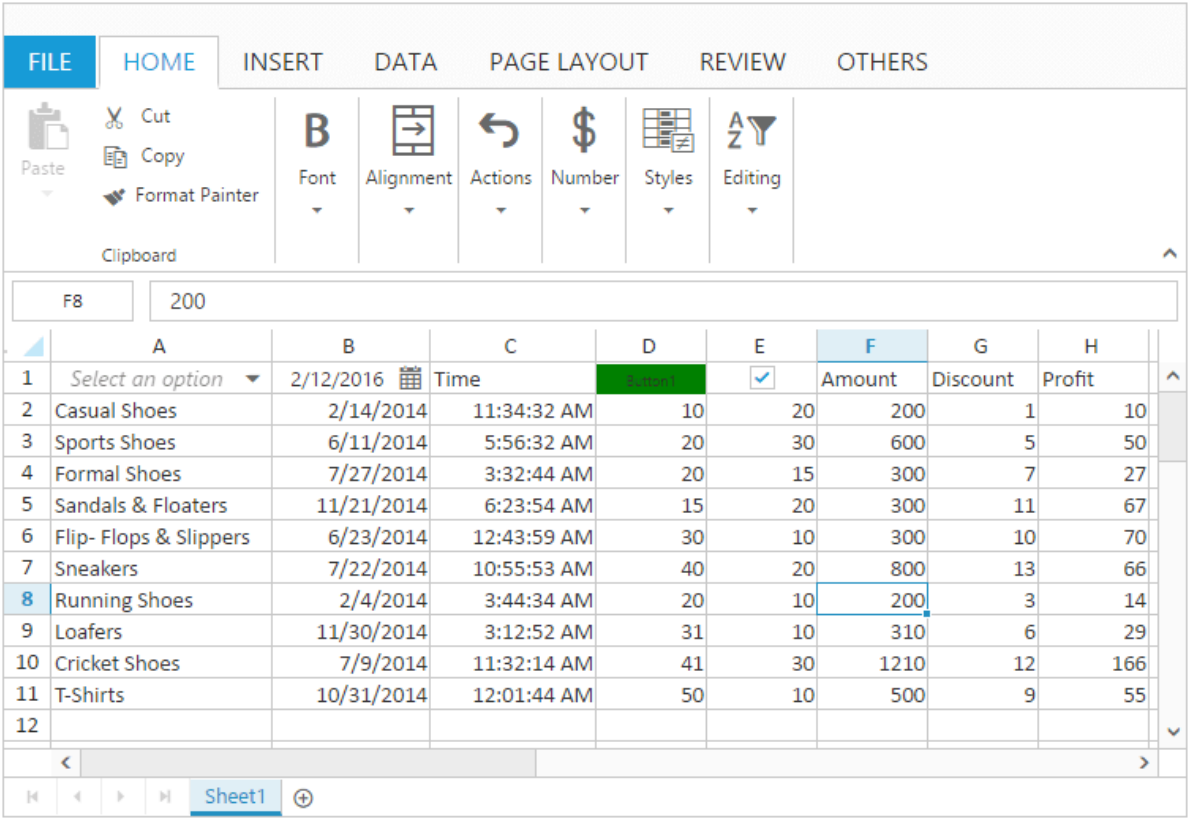
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Chart

Chart is a graphical representation of data, that organizes and represents a set of numerical or qualitative data. It mostly displays the selected range of data in terms of x axis and y axis. You can use AllowCharts property to enable/disable chart operations.

Copyright © 2001 - 2020 Syncfusion Inc.

Types of Chart

The following types of charts are available in Spreadsheet.

- Area Chart
- Bar Chart
- Column Chart
- Line Chart
- Pie Chart
- Radar Chart
- Scatter Chart

You can create the Chart by one of the following ways,

- Using "Chart Type" button to Select the type of chart under Charts group of INSERT Tab in ribbon.
- Using createChart method to create the chart.

Chart Customization

You can perform the following customizations for chart. These are available in DESIGN Tab which is enabled while clicking the chart element.

Feature	Description
Add Chart Elements	You can add a chart element like chart axes, legends, chart title, axis title, data labels and grid lines.
Switch Row/Column	You can switch the row of the chart to column of the chart and vice versa.
Select Data	You can modify the data source of Chart.
Chart Type	You can change the type of the chart using Chart Type dialog.
Height and Width	You can change the height and width of the chart.
Chart Themes	You can change the theme of the chart. The available themes are saffron, lemon and azure in dark, light themes.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowCharts(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

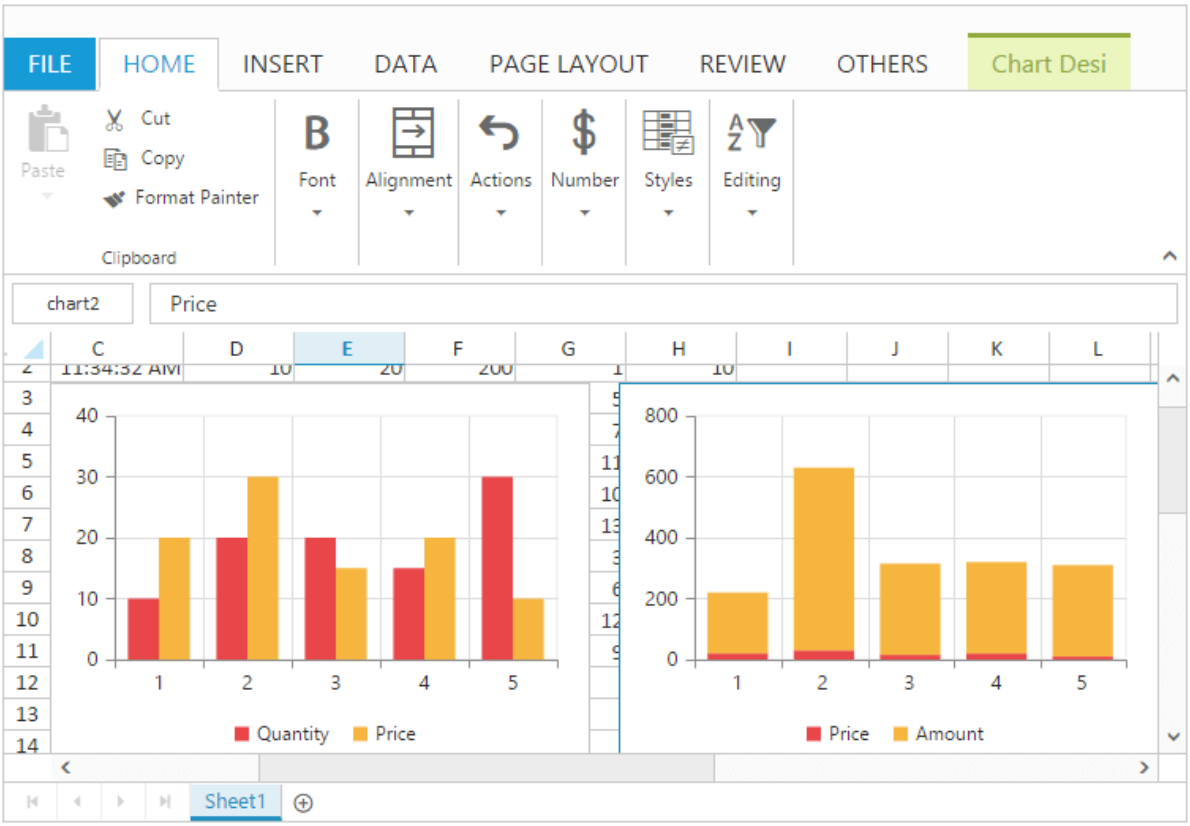
```
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.XLChart.createChart("D1:E6", { type: "column", enable3D: false,
marker: false, top: 40, left: 260, width: 340, height: 250 });
this.performSelection("E1");
this.XLChart.createChart("E1:F6", { type: "stackingcolumn", enable3D:
false, marker: { visible: false }, top: 40, left: 620, width: 340,
height: 250 });
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Conditional Formatting

Conditional formatting helps you to apply formats to a cell or range with certain color based on the cells values. You can use AllowConditionalFormats property to enable/disable Conditional formats.

Condition Definition

You can define conditions such as greater than, less than, between, equal to, text contains and date occurring for selected cells and defining value for condition. It highlights the specified cell.

You can do this by one of the following ways,

- Using "Conditional Formatting" option in Conditional Formatting button of HOME Tab in ribbon to open the conditional formatting dialog.
- Using setCFRule method to define the condition.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowConditionalFormats(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

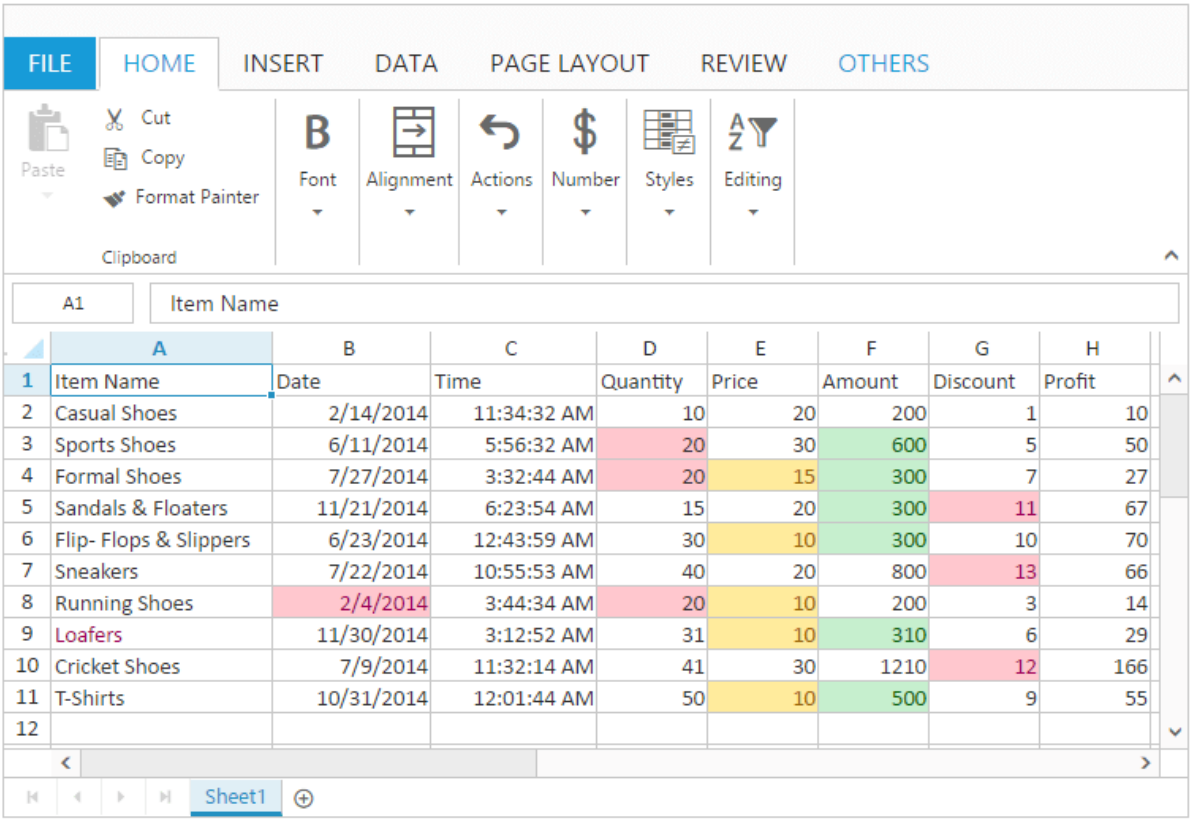
```
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var conditionFormat = this.XLCFormat;
if (!this.isImport) {
conditionFormat.setCFRule({ "action": "greaterthan", "inputs": ["10"],
"color": "redft", "range": "G2:G11" });
conditionFormat.setCFRule({ "action": "lessthan", "inputs": ["20"],
"color": "yellowft", "range": "E1:E11" });
conditionFormat.setCFRule({ "action": "between", "inputs": ["300",
"600"], "color": "greenft", "range": "F2:F11" });
conditionFormat.setCFRule({ "action": "equalto", "inputs": ["20"],
"color": "redf", "range": "D2:D11" });
conditionFormat.setCFRule({ "action": "textcontains", "inputs":
["loafers"], "color": "redt", "range": "A1:A11" });
conditionFormat.setCFRule({ "action": "dateoccur", "inputs":
["02/04/2014"], "color": "redft", "range": "B1:B11" });
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Clear Rules

You can clear the defined rules by using one of the following ways,

- Using "Clear Rules" option in Conditional Formatting button of HOME Tab in ribbon to clear the rule.
- Using clearCF method to clear the defined rules.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var conditionFormat = this.XLCFormat;
if (!this.isImport) {
conditionFormat.setCFRule({ "action": "greaterthan", "inputs": ["10"],
"color": "redft", "range": "G2:G11" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

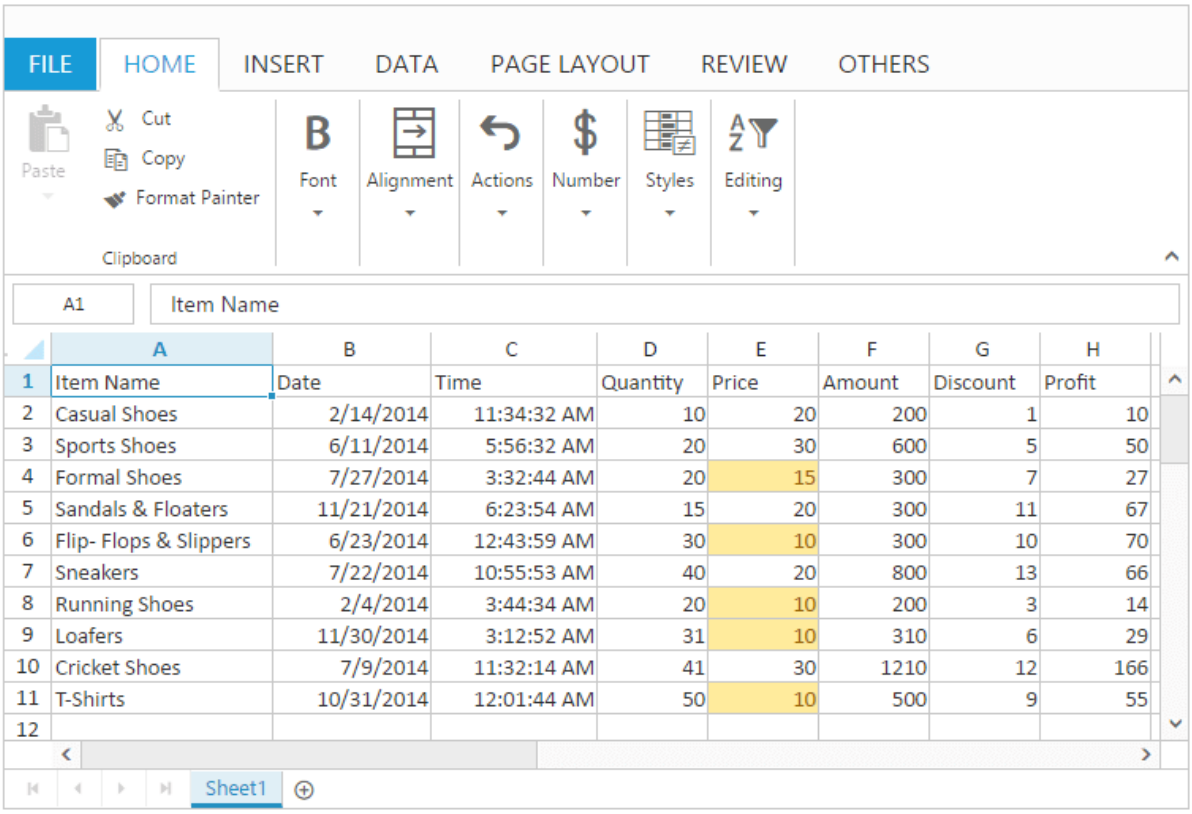
```
conditionFormat.setCFRule({ "action": "lessthan", "inputs": ["20"],
"color": "yellowft", "range": "E1:E11" });
conditionFormat.setCFRule({ "action": "between", "inputs": ["300",
"600"], "color": "greenft", "range": "F2:F11" });
conditionFormat.clearCF("G2:G11");
conditionFormat.clearCF("F2:F11");
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Filtering

Filtering allows you to view specific rows in Spreadsheet, while hiding the other rows. When a filter is added to the header row, a drop-down menu appears in each cell of the header row. You can use `AllowFiltering` property to enable/disable filtering.

You can apply filtering by one of the following ways,

- Using "Filter" option in Sort & Filter button under Editing group of HOME Tab in ribbon.
- Using "Filter" button under Sort and Filter group of DATA Tab in ribbon.

You have following options in Filtering.

- Filter by Value.
- Filter by Color.
- Clear Filter.

Filter by Value

You can perform filtering by using number, string. The filtered rows are only visible in the Spreadsheet. All the other rows within the filtered range were hidden.

You can do this by one of the following ways,

- Using dropdown button in filter header to open the filter dialog.
- Using context menu to select "Filter by Selected Cell's Value" option in Filter.
- Using filterByActiveCell method.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example describes the above behavior.

ASPX-CS

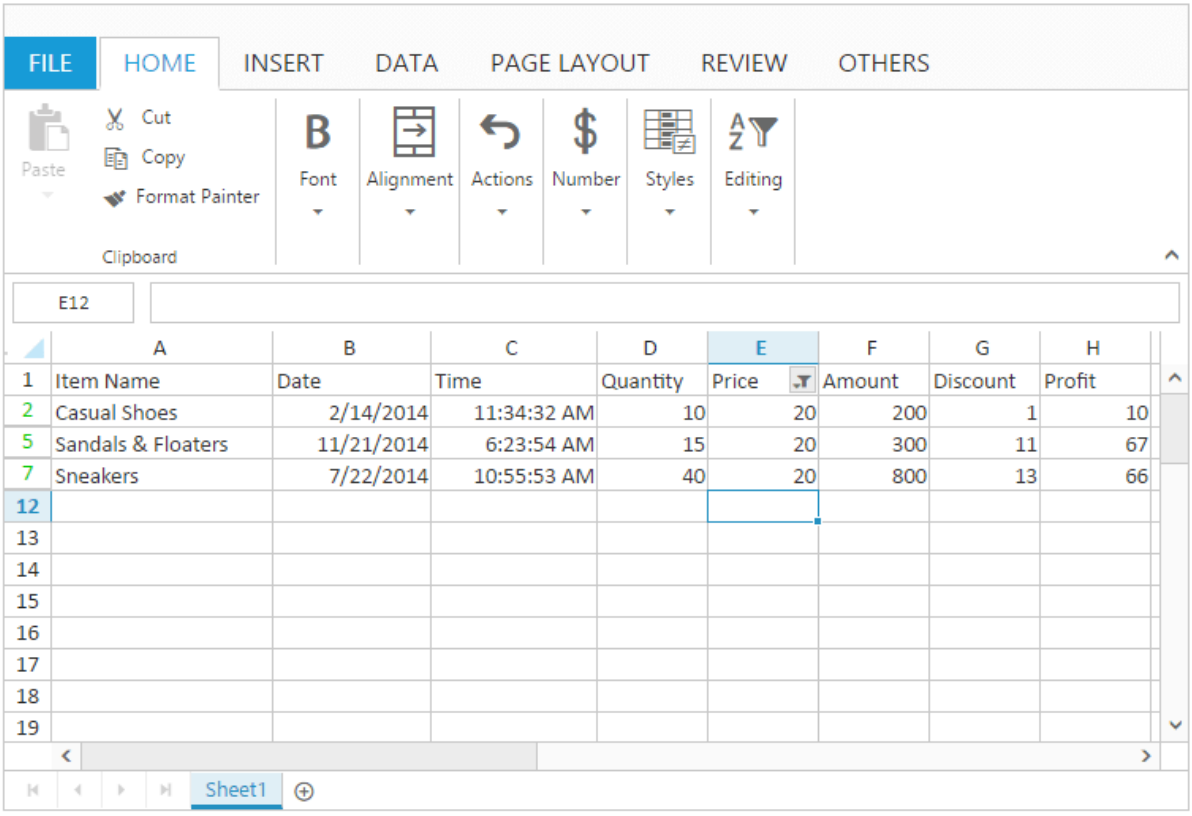
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowFiltering(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
this.performSelection("E2");
this.XLFilter.filterByActiveCell();
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Filter by Color

You can perform filtering by the selected cell color or font color. The filtered rows are only visible in the Spreadsheet. You can do this by clicking "Filter by Color" option in filter dialog to select filter by cell color or font color.

Note: This option is only available if the selected range of cells having any color.

Clear Filter

You can clear the filtering to show all the filtered rows in the spreadsheet within the filtered range.

You can do this by one of the following ways,

- Using "Clear Filter" option in the filter dialog.
- Using context menu to select "Clear Filter" option in Filter.
- Using "Clear Filter" option in "Sort & Filter" button under Editing group of HOME Tab in ribbon.
- Using "Clear Filter" option under Sort and Filter group of DATA Tab in ribbon.
- Using clearFilter method to perform clear filtering.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowFiltering(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
```

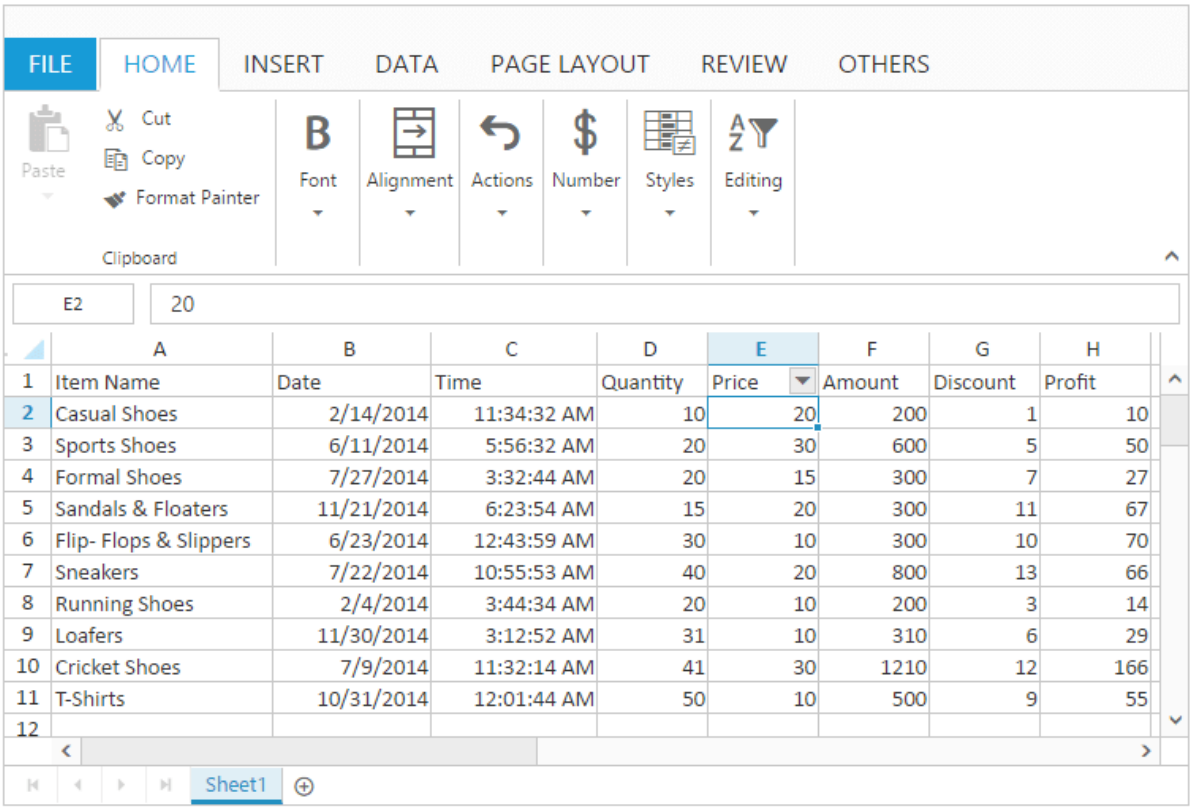
```
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var excelFilter = this.XLFilter;
if (!this.isImport) {
this.performSelection("E2");
excelFilter.filterByActiveCell();
excelFilter.clearFilter();
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55
12								

Picture

You can insert a picture by selecting the "Pictures" button under Illustrations group of INSERT Tab in ribbon.

Picture Customization

You can perform the following customizations for picture. These are available in DESIGN Tab which is enabled while clicking the picture element.

Feature	Description
Change Picture	You can change the picture with existing picture.
Reset Picture	You can reset the changes done in the picture such as border changes, height and width changes.
Picture Border	You can add border to the picture. You have Border Color, Border Type and Border weight options to draw a border.
Height and Width	You can change the height and width of the picture.

Pivot Table

Pivot table is a program tool that allows you to reorganize and summarize selected columns and rows of data to obtain a desired report. You can use `EnablePivotTable` property to enable/disable pivot table operations.

You can do this by one of the following ways,

Copyright © 2001 - 2020 Syncfusion Inc.

- Using "Pivot Table" option under Tables group of INSERT Tab in ribbon.
- Using createPivotTable method to create pivot table
- Using deletePivotTable method to remove the pivot table.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.EnablePivotTable(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
if (!this.isImport) {
var settings = {
rows: [
{
fieldName: "Country",
},
{
fieldName: "State",
}
],
columns: [
{
fieldName: "Product",
}
],
values: [
{
fieldName: "Amount",
},
{
fieldName: "Quantity",
}
],
filters: [
{
fieldName: "Date",
}
]
};
this.XLPivot.createPivotTable("Sheet1!$A$1:$F$25", null, null, settings);
//this.XLPivot.deletePivotTable("name");
}
}
</script>
```

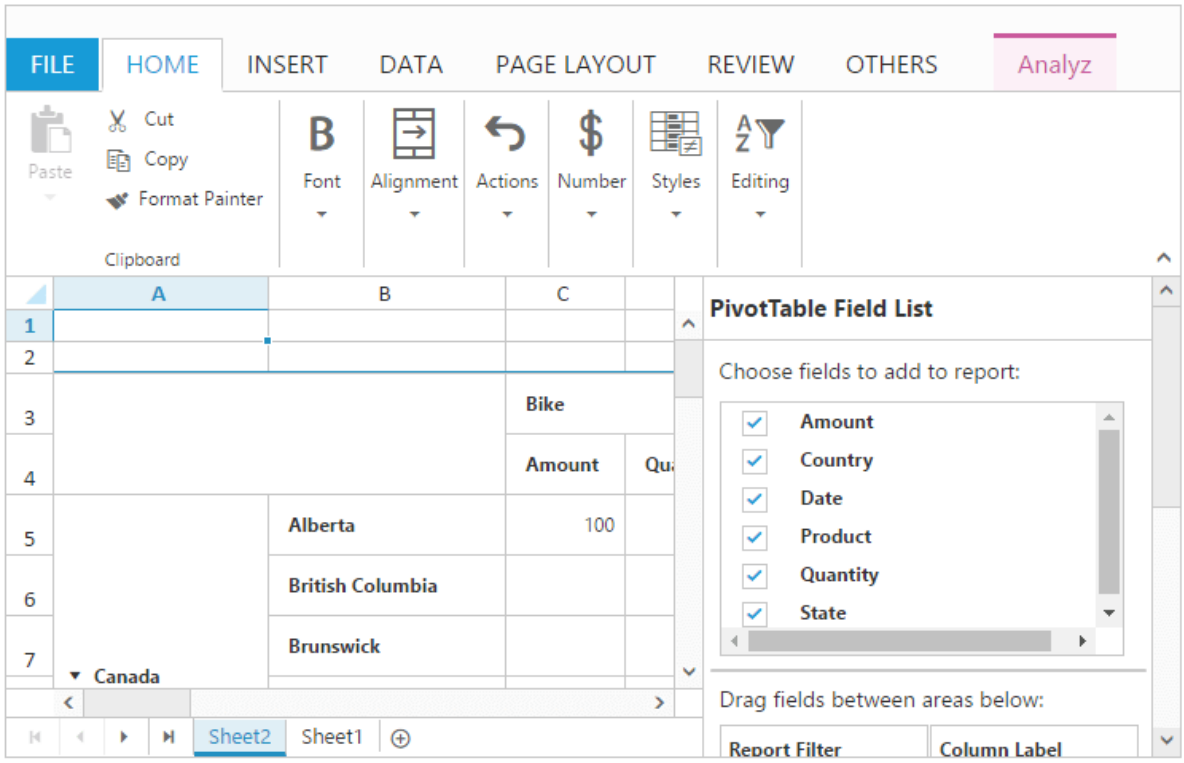
Copyright © 2001 - 2020 Syncfusion Inc.

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<PivotData> data = new List<PivotData>();
data.Add(new PivotData() { Amount = 100, Country = "Canada", Date = "FY
2005", Product = "Bike", Quantity = 2, State = "Alberta" });
data.Add(new PivotData() { Amount = 200, Country = "Canada", Date = "FY
2006", Product = "Van", Quantity = 3, State = "British Columbia" });
data.Add(new PivotData() { Amount = 300, Country = "Canada", Date = "FY
2007", Product = "Car", Quantity = 4, State = "Brunswick" });
data.Add(new PivotData() { Amount = 150, Country = "Canada", Date = "FY
2008", Product = "Bike", Quantity = 3, State = "Manitoba" });
data.Add(new PivotData() { Amount = 200, Country = "Canada", Date = "FY
2006", Product = "Car", Quantity = 4, State = "Ontario" });
data.Add(new PivotData() { Amount = 100, Country = "Canada", Date = "FY
2007", Product = "Van", Quantity = 1, State = "Quebec" });
data.Add(new PivotData() { Amount = 200, Country = "France", Date = "FY
2005", Product = "Bike", Quantity = 2, State = "Charente-Maritime" });
data.Add(new PivotData() { Amount = 250, Country = "France", Date = "FY
2006", Product = "Van", Quantity = 4, State = "Essonne" });
data.Add(new PivotData() { Amount = 300, Country = "France", Date = "FY
2007", Product = "Car", Quantity = 3, State = "Garonne (Haute)" });
data.Add(new PivotData() { Amount = 150, Country = "France", Date = "FY
2008", Product = "Van", Quantity = 2, State = "Gers" });
data.Add(new PivotData() { Amount = 200, Country = "Germany", Date = "FY
2006", Product = "Van", Quantity = 3, State = "Bayern" });
data.Add(new PivotData() { Amount = 250, Country = "Germany", Date = "FY
2007", Product = "Car", Quantity = 3, State = "Brandenburg" });
data.Add(new PivotData() { Amount = 150, Country = "Germany", Date = "FY
2008", Product = "Car", Quantity = 4, State = "Hamburg" });
data.Add(new PivotData() { Amount = 200, Country = "Germany", Date = "FY
2008", Product = "Bike", Quantity = 4, State = "Hessen" });
data.Add(new PivotData() { Amount = 150, Country = "Germany", Date = "FY
2007", Product = "Van", Quantity = 3, State = "Nordrhein-Westfalen" });
data.Add(new PivotData() { Amount = 100, Country = "Germany", Date = "FY
2005", Product = "Bike", Quantity = 2, State = "Saarland" });
data.Add(new PivotData() { Amount = 150, Country = "United Kingdom", Date
= "FY 2008", Product = "Bike", Quantity = 5, State = "England" });
data.Add(new PivotData() { Amount = 250, Country = "United States", Date
= "FY 2007", Product = "Car", Quantity = 4, State = "Alabama" });
data.Add(new PivotData() { Amount = 200, Country = "United States", Date
= "FY 2005", Product = "Van", Quantity = 4, State = "California" });
data.Add(new PivotData() { Amount = 100, Country = "United States", Date
= "FY 2006", Product = "Bike", Quantity = 2, State = "Colorado" });
data.Add(new PivotData() { Amount = 150, Country = "United States", Date
= "FY 2008", Product = "Car", Quantity = 3, State = "New Mexico" });
data.Add(new PivotData() { Amount = 200, Country = "United States", Date
= "FY 2005", Product = "Bike", Quantity = 4, State = "New York" });
data.Add(new PivotData() { Amount = 250, Country = "United States", Date
= "FY 2008", Product = "Car", Quantity = 3, State = "North Carolina" });
data.Add(new PivotData() { Amount = 300, Country = "United States", Date
= "FY 2007", Product = "Van", Quantity = 4, State = "South Carolina" });
```

```
ViewBag.Datasource = data;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



Sorting

Sorting helps you to arrange the data to particular order in a selected range of cells. You can use AllowSorting property to enable/disable sorting.

You have following options in sorting.

- Sort by Ascending or Descending.
- Sort by Color.

Sort by Ascending and Descending

You can perform number sorting by ascending or descending and string sorting by A to Z or Z to A to arrange the data.

You can do this by one of the following ways,

- By choosing "Sort A to Z" or "Sort Z to A" in Sort & Filter button under Editing group of HOME Tab in ribbon.
- Using "Sort A to Z" or "Sort Z to A" button in Sort & Filter group of DATA Tab in ribbon.
- Using context menu to select "Sort A to Z" or "Sort Z to A" for strings and option in Sort.

Copyright © 2001 - 2020 Syncfusion Inc.

- Using "Sort A to Z" or "Sort Z to A" for strings an "Sort Smallest to Largest" or "Sort Largest to Smallest" for numbers in filter dialog.
- Using sortByRange method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowSorting(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var excelSort = this.XLSort,excelFormat = this.XLFormat;
if (!this.isImport) {
excelSort.sortByRange("A2:A10", "A", "ascending");
excelSort.sortByRange("E2:E10", "E", "descending");
}
}
</script>
```

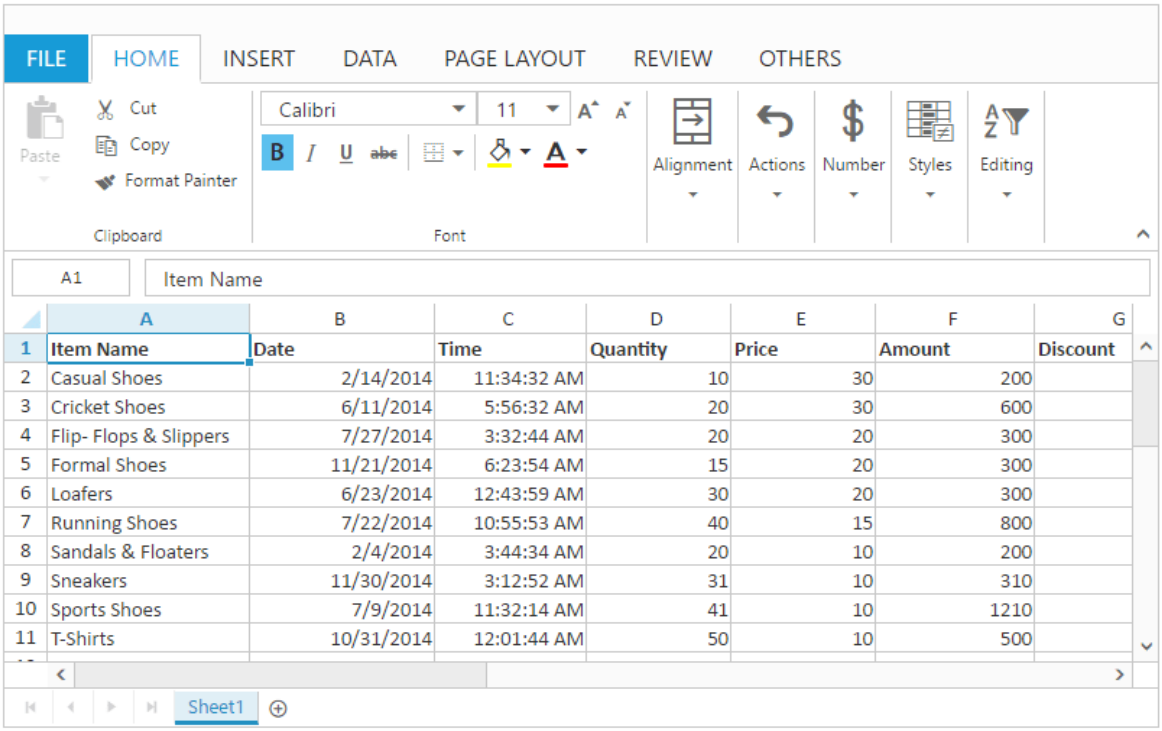
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
}
```

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G
1	Item Name	Date	Time	Quantity	Price	Amount	Discount
2	Casual Shoes	2/14/2014	11:34:32 AM	10	30	200	
3	Cricket Shoes	6/11/2014	5:56:32 AM	20	30	600	
4	Flip- Flops & Slippers	7/27/2014	3:32:44 AM	20	20	300	
5	Formal Shoes	11/21/2014	6:23:54 AM	15	20	300	
6	Loafers	6/23/2014	12:43:59 AM	30	20	300	
7	Running Shoes	7/22/2014	10:55:53 AM	40	15	800	
8	Sandals & Floaters	2/4/2014	3:44:34 AM	20	10	200	
9	Sneakers	11/30/2014	3:12:52 AM	31	10	310	
10	Sports Shoes	7/9/2014	11:32:14 AM	41	10	1210	
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	

Sort by Color

You can perform sort by color to arrange the data based on the selected cell's background color or font color. This option is only available if the selected range of cells having any color.

You can do this by one of the following ways,

- Using "Sort By Color" option in filter dialog to perform sorting by cell color or font color.
- Using context menu to select "Put Selected Cell Color To The Top" or "Put Selected Font Color To The Top" in Sort option.

Table

A table is a data structure that organizes information into rows and columns. You can use `AllowFormatAsTable` property to enable/disable table operations.

You can do this by one of the following ways,

- Using "Format As Table" under Styles group of HOME Tab in ribbon.
- Using Table option under Tables group of INSERT Tab in ribbon.
- Using createTable method to insert a table and removeTable to delete a table.

Copyright © 2001 - 2020 Syncfusion Inc.

Table Customization

You can perform the following customizations for table. These are available in DESIGN Tab which is enabled while clicking the table.

Feature	Description
Resize Table	You can add new rows to the table.
Convert to Range	You can convert the table to regular range of data without losing any table style format.
First Column	You can highlight the first column of the table.
Last Column	You can highlight the last column of the table.
Total Row	You can insert a new row in the bottom of the table to display the total value of the last column. You can toggle this by using checkbox.
Filter Button	You can able to hide or unhide the filter icons in the filter header of a table.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var excelFormat = this.XLFormat;
if (!this.isImport) {
excelFormat.createTable({ "header": true, "formatName":
"TableStyleLight8" }, "A1:B4");
excelFormat.createTable({ "header": true, "formatName":
"TableStyleLight10" }, "D1:E4");
}
}
</script>
```

C#

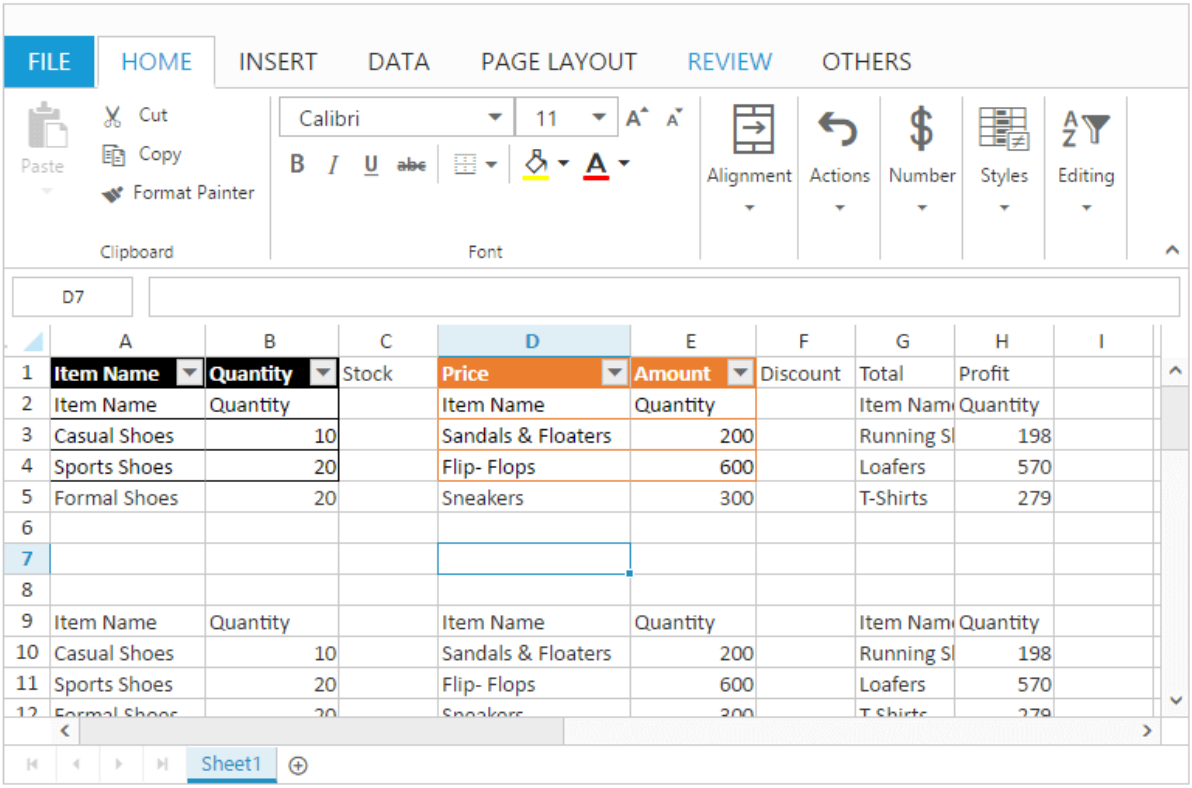
```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<FormatData> formats = new List<FormatData>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
formats.Add(new FormatData() { format1 = "Item Name", format2 =
"Quantity", format4 = "Item Name", format5 = "Quantity", format7 = "Item
Name", format8 = "Quantity" });
formats.Add(new FormatData() { format1 = "Casual Shoes", format2 = "10",
format4 = "Sandals & Floaters", format5 = "200", format7 = "Running
Shoes", format8 = "198" });
formats.Add(new FormatData() { format1 = "Sports Shoes", format2 = "20",
format4 = "Flip- Flops", format5 = "600", format7 = "Loafers", format8 =
"570" });
formats.Add(new FormatData() { format1 = "Formal Shoes", format2 = "20",
format4 = "Sneakers", format5 = "300", format7 = "T-Shirts", format8 =
"279" });
formats.Add(new FormatData());
formats.Add(new FormatData());
formats.Add(new FormatData());
formats.Add(new FormatData() { format1 = "Item Name", format2 =
"Quantity", format4 = "Item Name", format5 = "Quantity", format7 = "Item
Name", format8 = "Quantity" });
formats.Add(new FormatData() { format1 = "Casual Shoes", format2 = "10",
format4 = "Sandals & Floaters", format5 = "200", format7 = "Running
Shoes", format8 = "198" });
formats.Add(new FormatData() { format1 = "Sports Shoes", format2 = "20",
format4 = "Flip- Flops", format5 = "600", format7 = "Loafers", format8 =
"570" });
formats.Add(new FormatData() { format1 = "Formal Shoes", format2 = "20",
format4 = "Sneakers", format5 = "300", format7 = "T-Shirts", format8 =
"279" });
ViewBag.Datasource = formats;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Formulas

Formulas are used for calculation of data in sheet. You can refer the formula cell reference in following ways,

1. Cell reference from same sheet
2. Cell reference from different sheets

You can set formula for a cell in following ways,

1. Initial Load
2. Method
3. User Interface

Initial Load

You can set formula for a cell by specifying Value property in cell data binding. The following code example describes the above behavior,

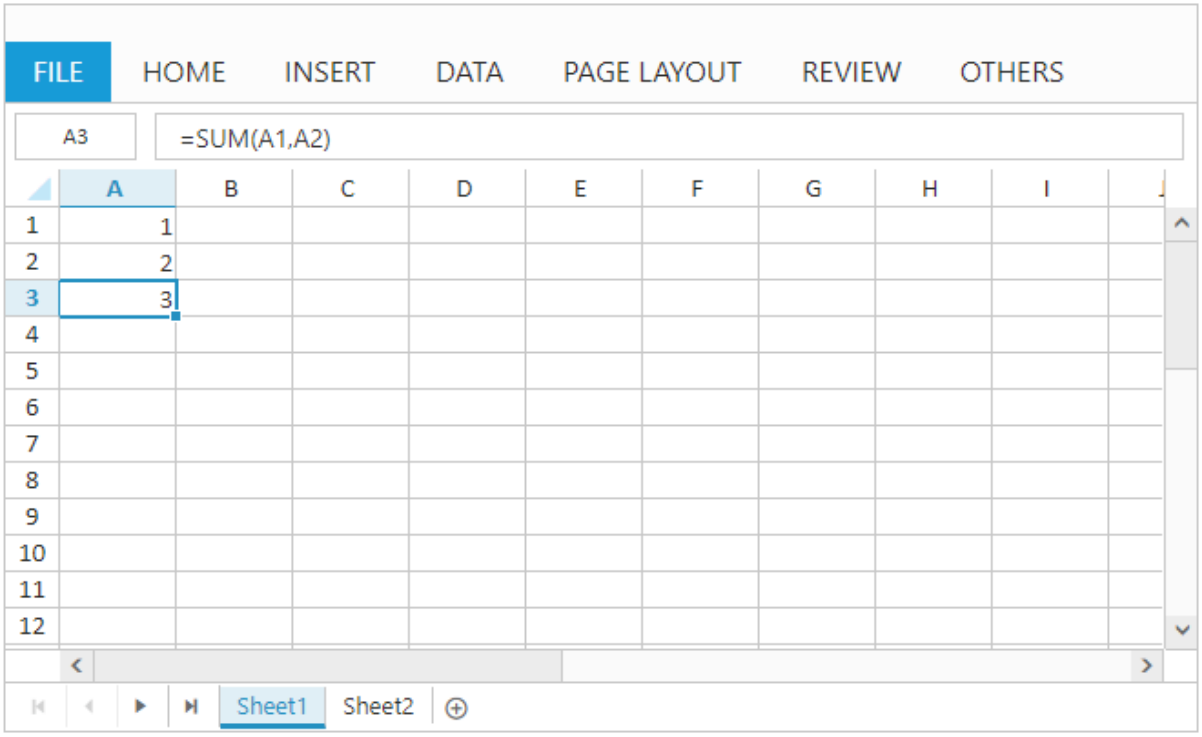
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("1").Add();
```

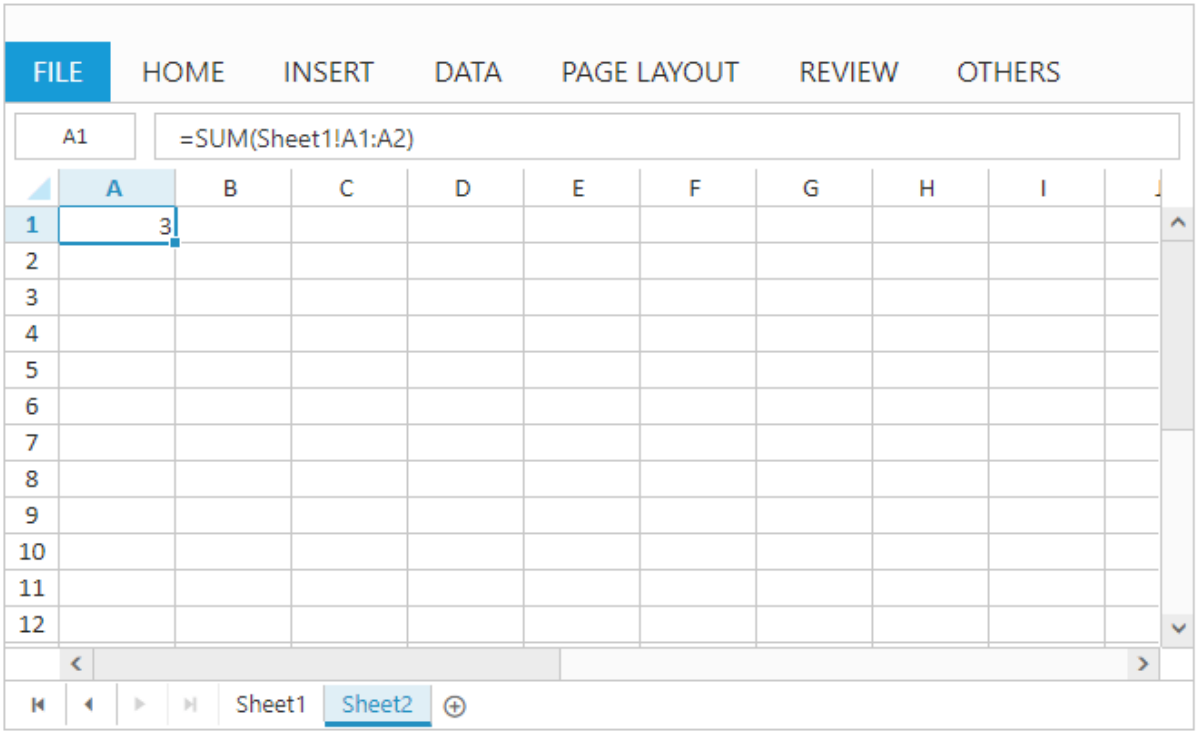
Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("=SUM(A1,A2)").Add();
}).Add();
}).Add();
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("=Sheet1!A1:A2").Add();
}).Add();
}).Add();
})
)
```

The following output is displayed as a result of the above code example.



Cell Reference from Same sheet



Cell Reference from different sheets

Method

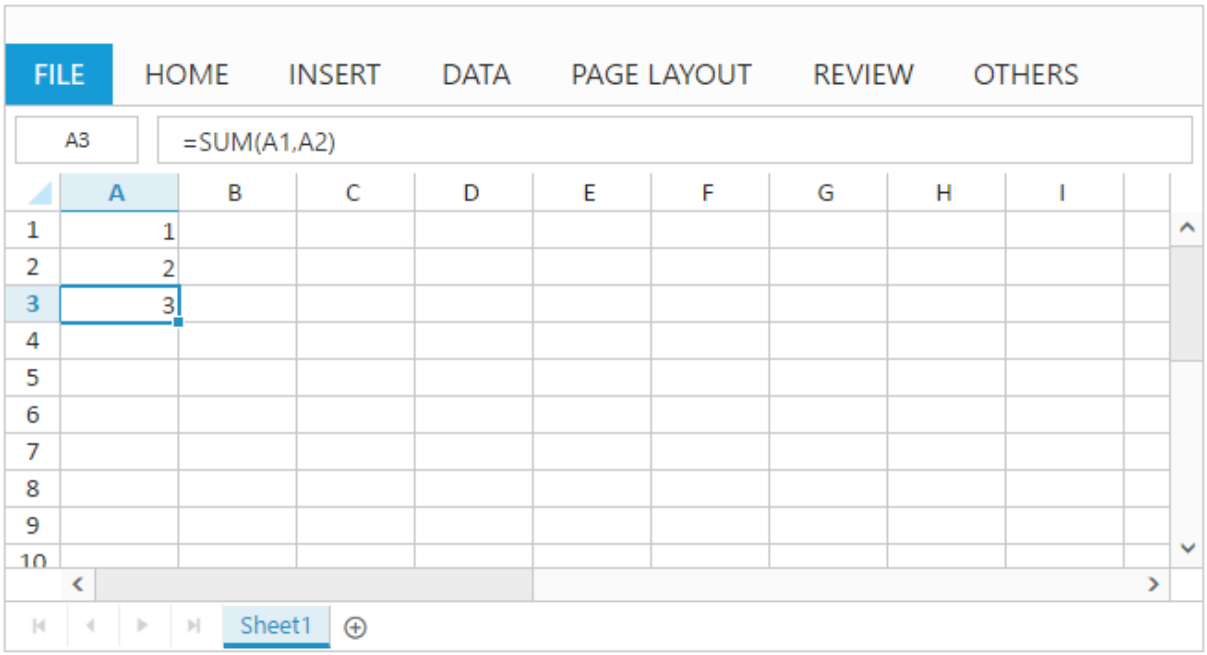
You can set formula for a cell using updateCellValue method. The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("1").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script>
function loadComplete() {
this.XLEdit.updateCellValue({ rowIndex: 2, colIndex: 0 }, "=SUM(A1,A2)");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output is displayed as a result of the above code example.



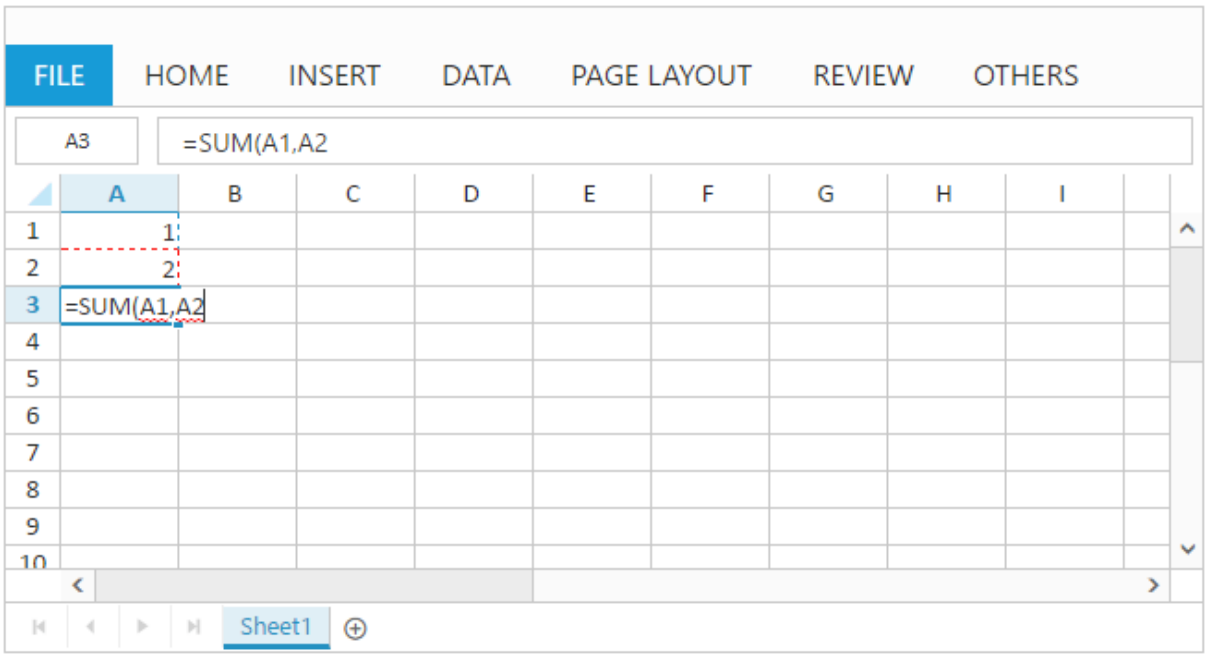
User Interface

You can set formula for a cell by edit and save a cell through user interface using Editing feature.

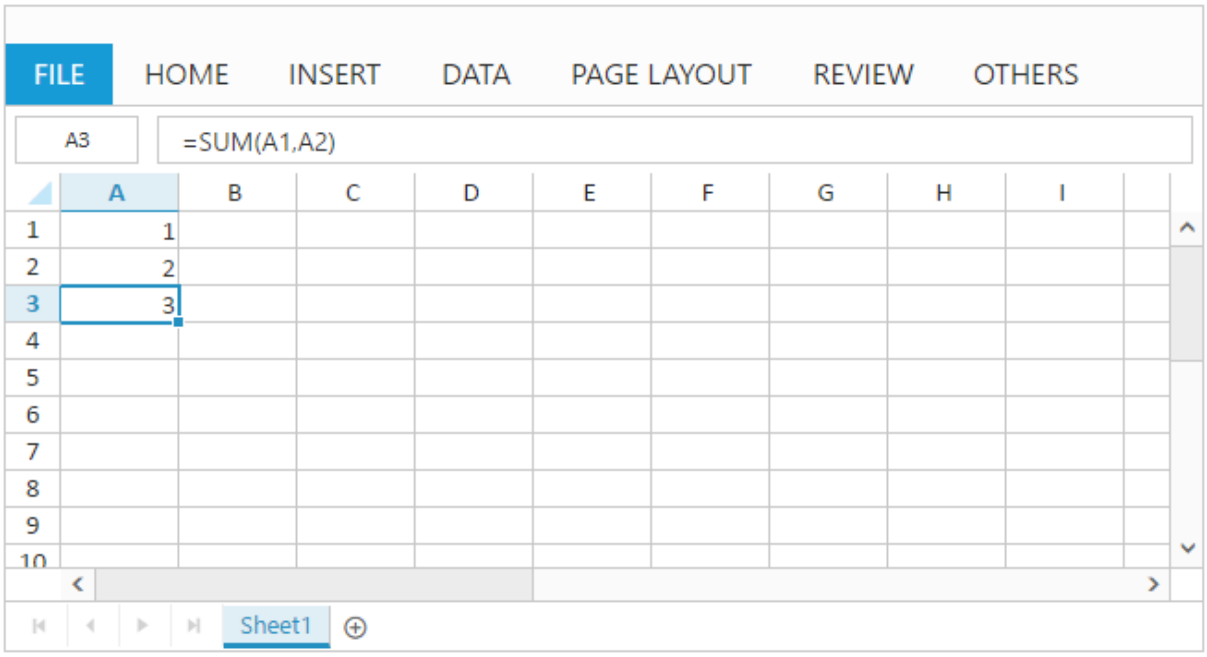
The following code example and screenshot describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("1").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
}).Add();
})
)
```



The following output is displayed while saving edited cell with above code example.



Note: 1. The list of supported formulas can be find in following link

2. Constant values, cell references, formulas and named ranges can be passed as argument to formulas

3. Selection can be used to mention cell references within formula

User Defined Functions

The list of formulas supported in Spreadsheet is sufficient for most of your calculations. If not, you can create and use your own function using user defined function option. You can add user defined function to Spreadsheet in following ways,

Copyright © 2001 - 2020 Syncfusion Inc.

1. Initial Load
2. Method

Initial Load

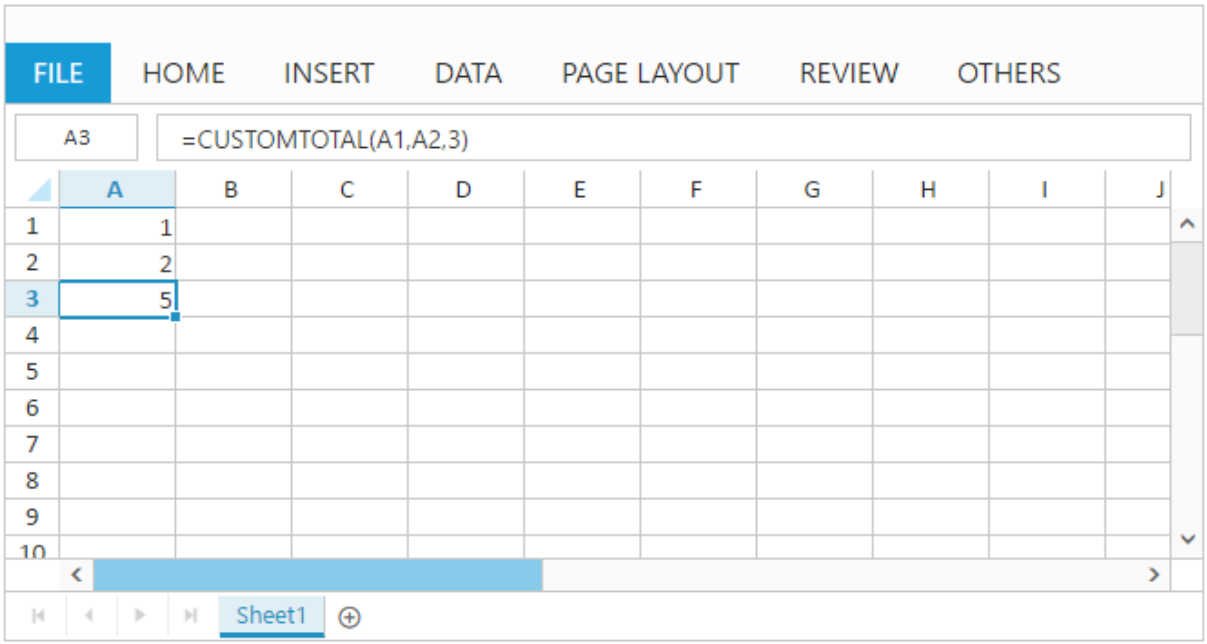
You can add your own function to Spreadsheet at initial load with CustomFormulas property. The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("1").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("=CUSTOMTOTAL(A1,A2,3)").Add();
}).Add();
}).Add();
})
.CustomFormulas(formula => {
formula.FormulaName("CUSTOMTOTAL").FunctionName("customTotal").Add(); })
)
<script>
function customTotal(args) {
var param1, param2, param3, value, excelObj =
$('#Spreadsheet').data("ejSpreadsheet"),
argument = excelObj.getValueFromFormulaArg(args);
param1 = argument["arg1"];
param2 = argument["arg2"];
param3 = argument["arg3"];
value = param1 * param2 + param3;
return value;
}
</script>
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Method

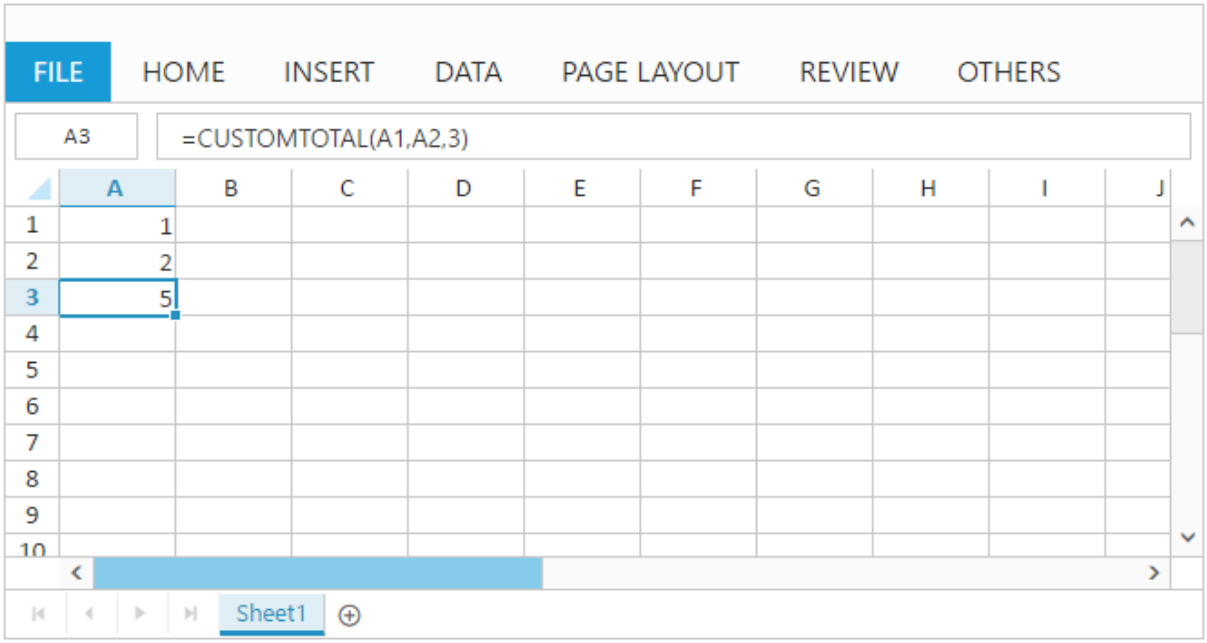
You can add your own function to Spreadsheet using addCustomFormula method. The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("1").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script>
function loadComplete() {
this.addCustomFormula("CUSTOMTOTAL", "customTotal");
this.XLEdit.updateCellValue({ rowIndex: 2, colIndex: 0 },
"=CUSTOMTOTAL(A1,A2,3)");
}
function customTotal(args) {
var param1, param2, param3, value, excelObj =
$('#Spreadsheet').data("ejSpreadsheet"),
argument = excelObj.getValueFromFormulaArg(args);
param1 = argument["arg1"];
param2 = argument["arg2"];
```

```
param3 = argument["arg3"];
value = param1 * param2 + param3;
return value;
}
</script>
```

The following output is displayed as a result of the above code example.



To remove user defined function from Spreadsheet use removeCustomFormula method.

Named Ranges

To understand the purpose of cell reference or table, you can define a meaningful name using named ranges support. By using names, you can make your formula much easier to understand and maintain. You can add named ranges to Spreadsheet in following ways,

1. Initial Load
2. Method
3. User Interface

Initial Load

To add named ranges at initial load by using nameManager API, and also you can specify the name and address of the range by using name and refersto API. You can find the added named range list collection in Spreadsheet by accessing nameManager API.

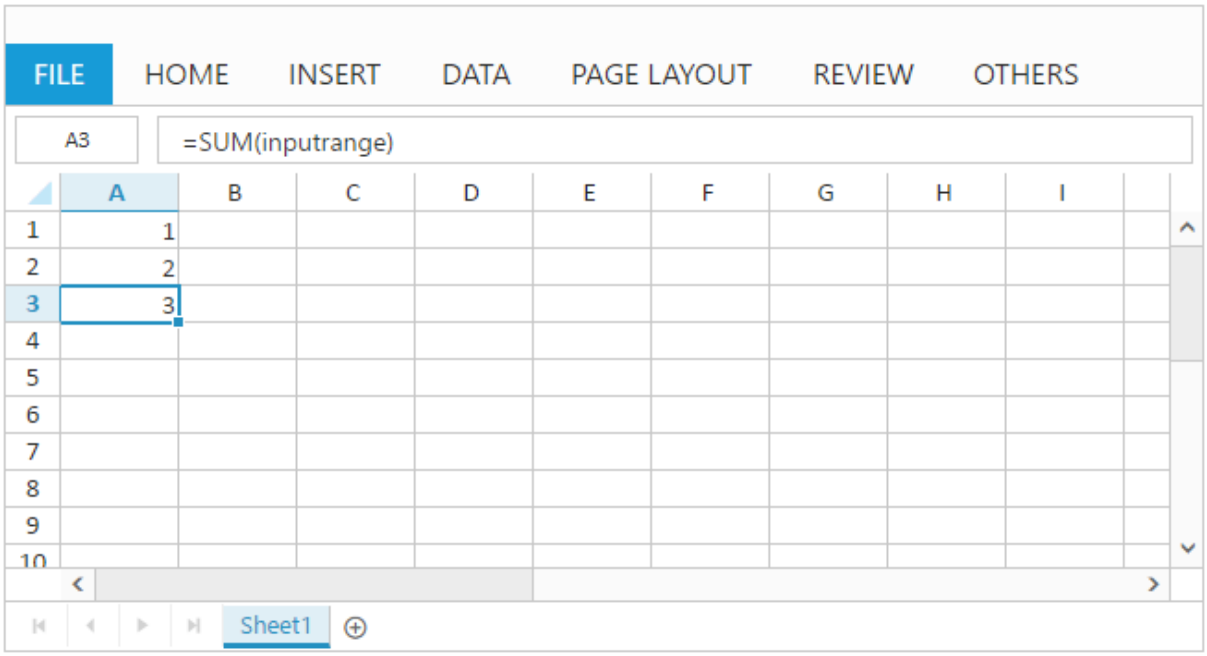
The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
```

```
rows.Cells(cells =>
{
cells.Value("1").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("=SUM(inputRange)").Add();
}).Add();
})
.NameManager(manager => {
manager.Name("inputRange").RefersTo("=Sheet1!$A$1:$A$2").Add(); })
)
```

The following output is displayed as a result of the above code example.



Method

You can add named range to Spreadsheet with addNamedRange method and it can be removed with removeNamedRange method.

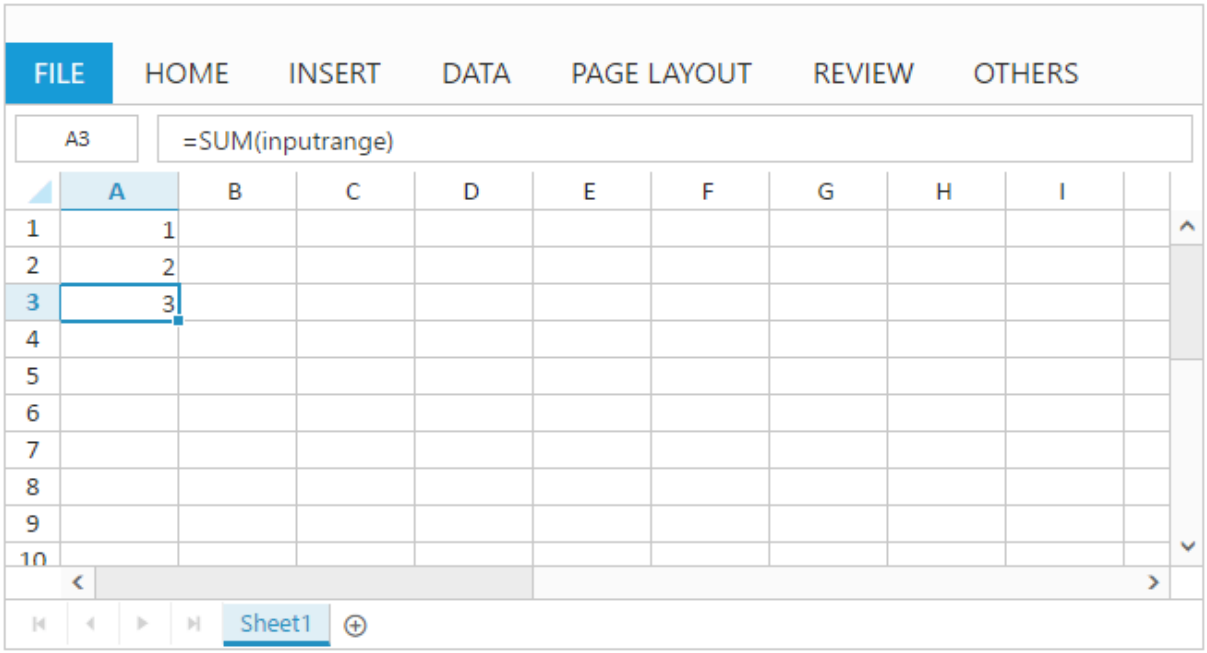
The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
```

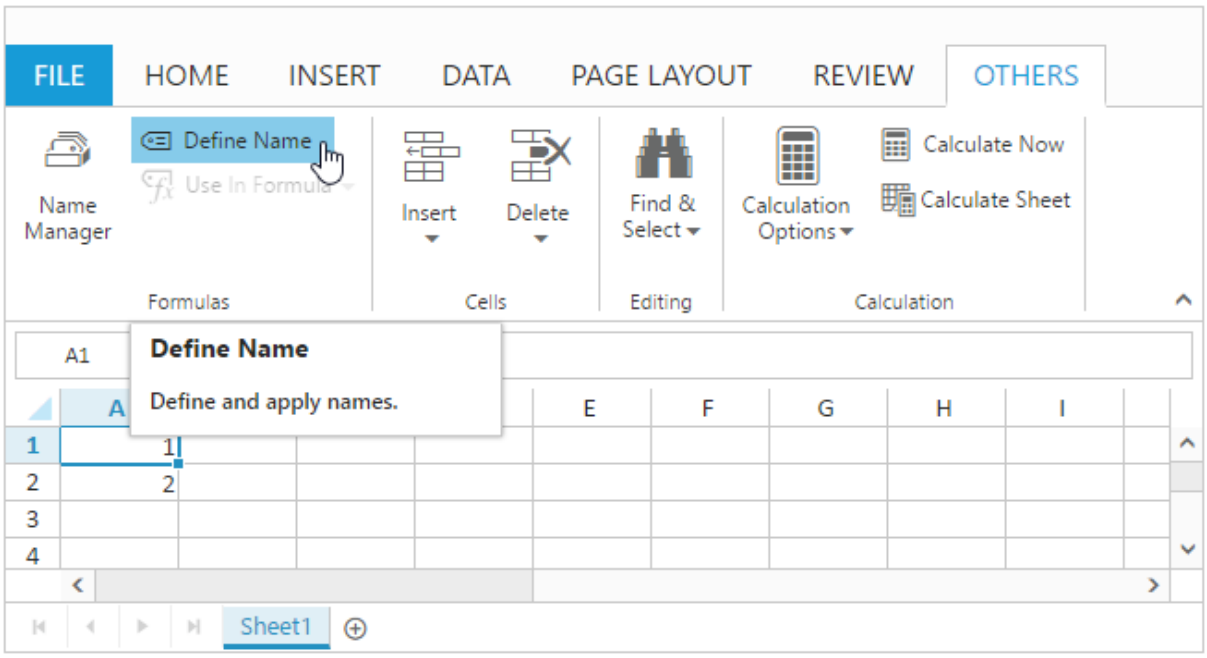
```
cells.Value("1").Add();
}).Add();
rows.Cells(cells =>
{
cells.Value("2").Add();
}).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script>
function loadComplete() {
this.XLRibbon.addNamedRange("inputRange", "=Sheet1!$A$1:$A$2", "named
range demo", this.getActiveSheetIndex());
this.XLEdit.updateCellValue({rowIndex: 2, colIndex: 0},
"=SUM(inputRange)");
}
</script>
```

The following output is displayed as a result of the above code example.



User Interface

You can define name for range of cells through user interface using Define Name option in OTHERS tab. The following screenshot describes the above behavior,



Note: 1. Defining name for cell reference or table will be accessible across all sheets.

2. Named Ranges will be displayed in Name Manger dialog box.

Formula Bar

Formula bar is used to edit or enter cell data in much easier way. To enable formula bar set `AllowFormulaBar` as `true`.

Auto Sum

To sum a row or column of data, select a cell next to the data you want to sum, click `AutoSum` on the `HOME` tab and press enter. To enable auto sum set `AllowAutoSum` API as `true`.

The auto sum options in ribbon is used to perform basic operations like sum, average, count, minimum, maximum etc.

Formatting

Spreadsheet supports many formatting options to make your data easier to view and understand. Use `AllowCellFormatting` property to enable / disable formatting option in Spreadsheet. The different types of formatting supported in Spreadsheet are,

1) Number Formatting 2) Text Formatting 3) Cell Formatting

Number Formatting

Number formatting is used to represent type for your data in Spreadsheet. The different types of number formatting supported in Spreadsheet are,

1) Number 2) Currency 3) Accounting 4) Percentage 5) Short Date 6) Long Date 7) Time 8) Scientific 9) Fraction

You can apply number format for a cell in following ways,

1) Initial Load 2) Method 3) User Interface

Copyright © 2001 - 2020 Syncfusion Inc.

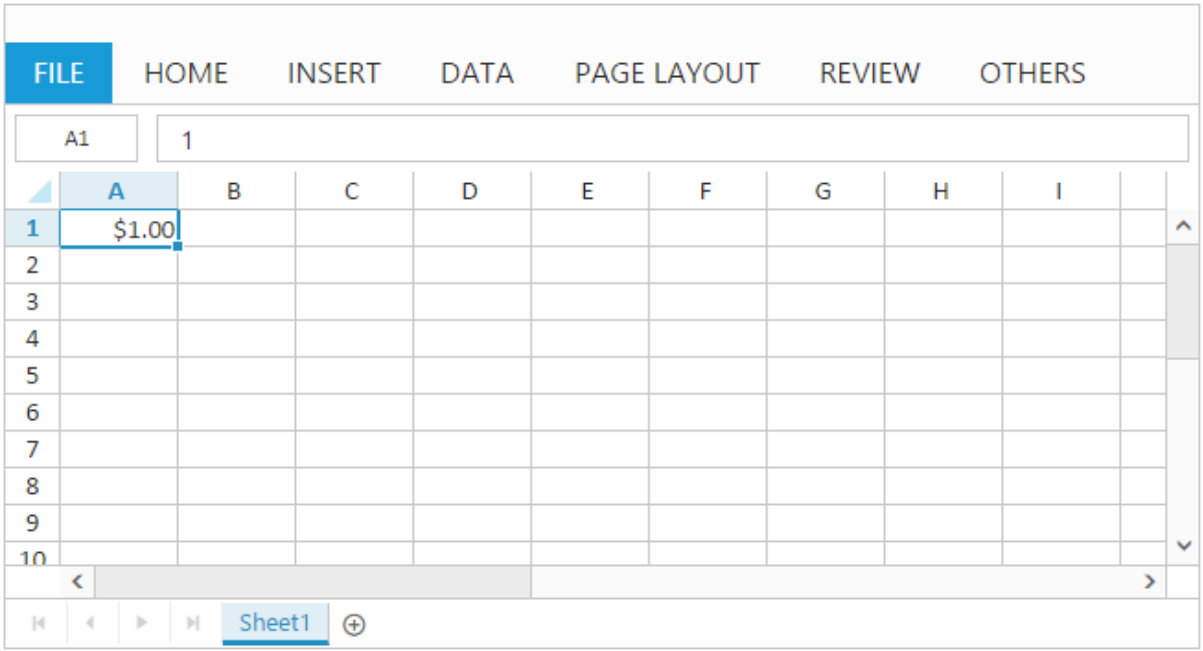
Initial Load

You can set number format for a cell by specifying Format property in cell data binding. The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("1").Format(format => {
format.Type("currency");
}).Add();
}).Add();
}).Add();
})
)
```

The following output is displayed as a result of the above code example.



Method

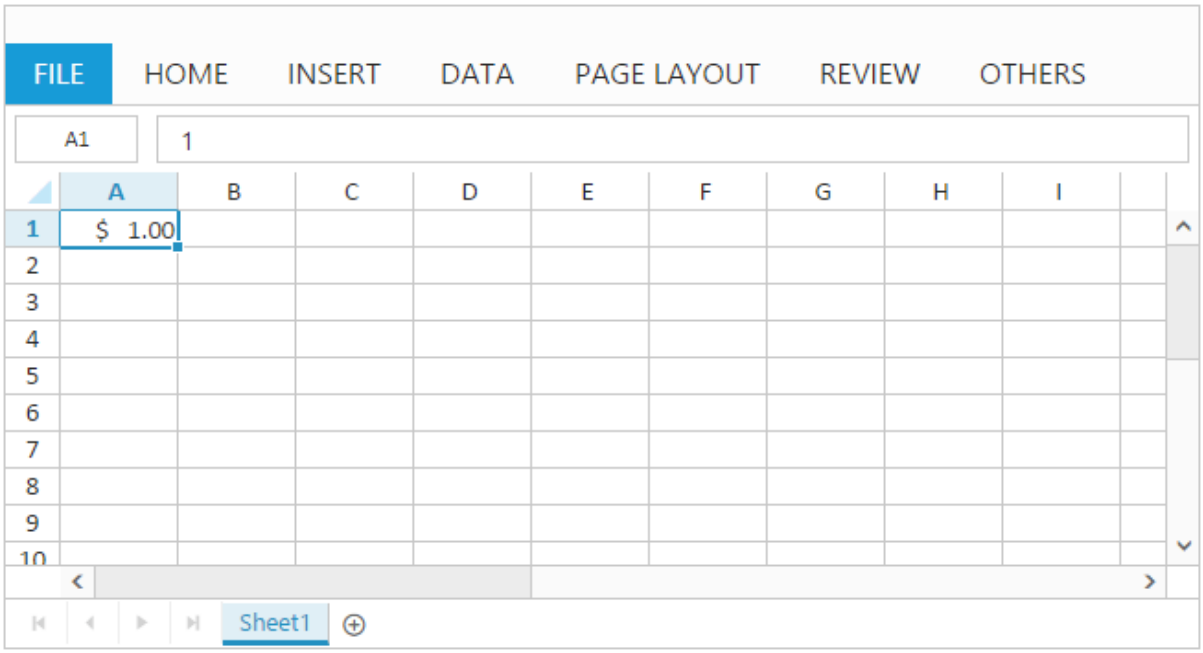
You can set number format for a cell using format method. The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
```

```
rows.Cells(cells =>
{
cells.Value("1").Add();
}).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script>
function loadComplete() {
this.XLFormat.format({ type: "accounting" }, "A1"); // applying
accounting type to cell A1
}
</script>
```

The following output is displayed as a result of the above code example.



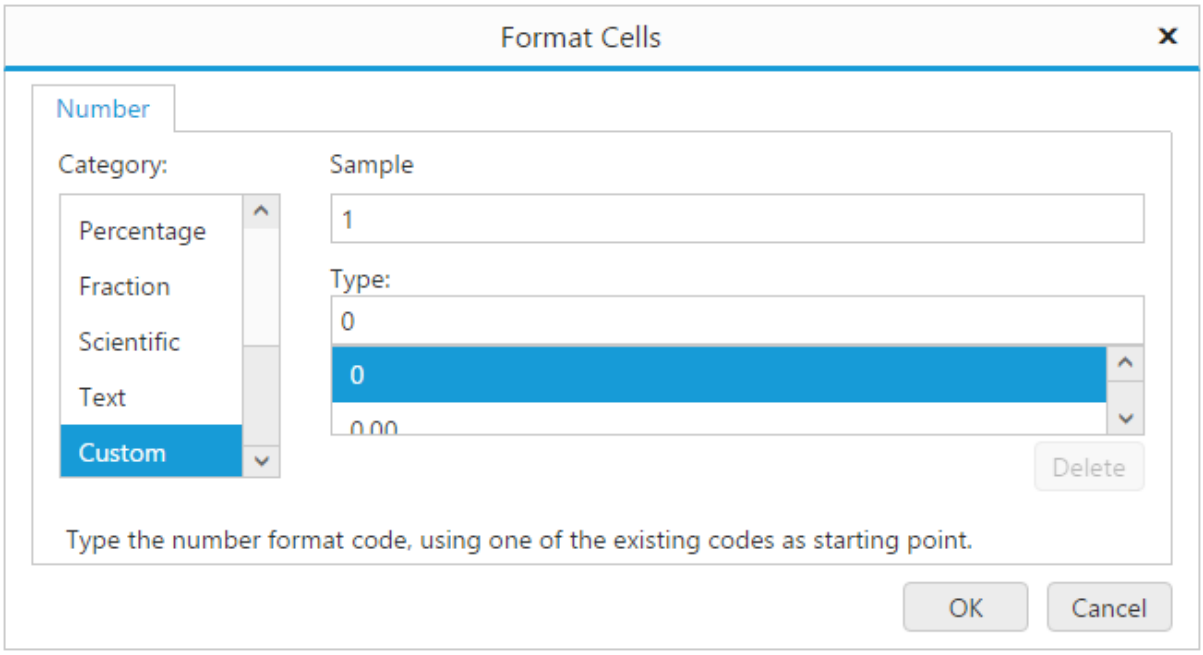
User Interface

You can set number format for a cell through number formatting options in ribbon HOME tab.

Custom Number Format

Spreadsheet supports many number format to display your data as currency, date, percentage and so on. If these pre-defined number formats do not meet your needs you can create and apply your own number formats using format cell dialog. The following screenshot illustrate this,

Copyright © 2001 - 2020 Syncfusion Inc.



Note: Spreadsheet supports basic number format customization and it doesn't have all functionality similar to excel

Text Formatting

To organize and easier to follow your financial, statistical or scientific data, you can apply text formats like font size, font color, text alignment etc. to a cell or range of cells.

Fonts

To distinguish your data from built-in font formats, you can apply different font formats like bold, italic, strike-through, color, font-family and size etc.

Text Alignment

To enhance the visual presentation of your data, you can align text in a cell vertically or horizontally. To align text vertically pick top, middle or bottom align and to align text horizontally pick left, center or right align.

Indents

To enhance the appearance of text in a cell, you can change the indentation of a cell content by increasing or decreasing text indent.

Applying Text Formatting

You can apply text format for a cell in following ways,

1) Initial Load 2) Method 3) User Interface

Initial Load

You can apply text format for a cell by specifying Style property in cell data binding. The following code example describes the above behavior,

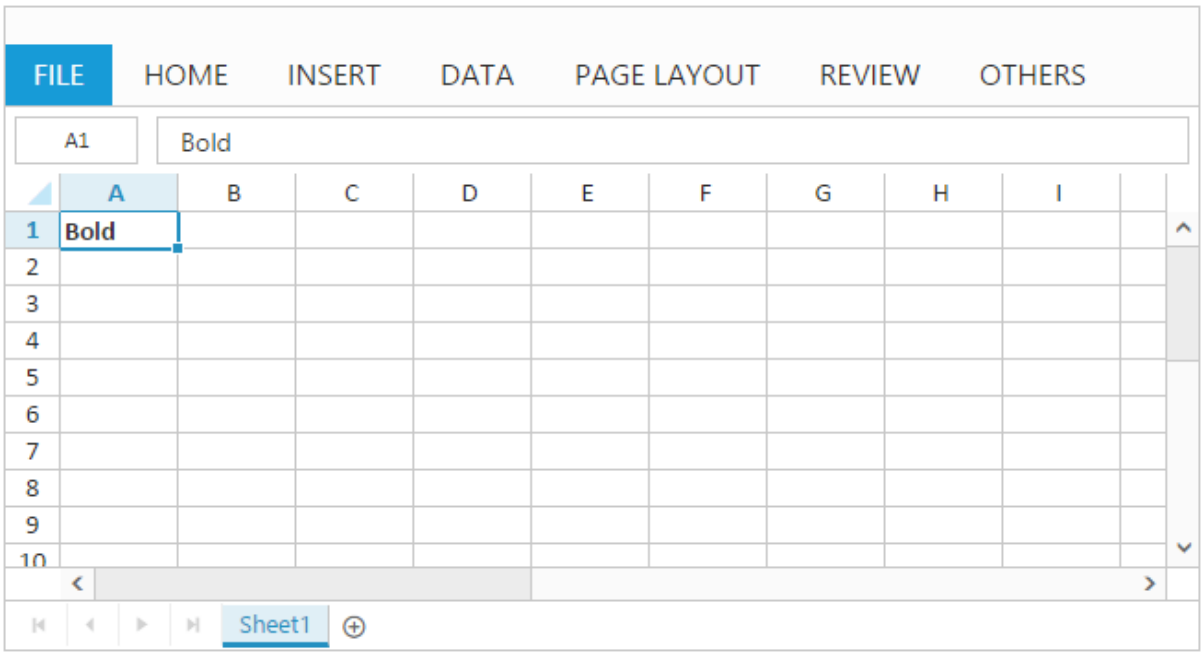
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("Bold").Style(style =>
{
style.FontWeight("bold");
}).Add();
}).Add();
}).Add();
})
)
```

The following output is displayed as a result of the above code example.



Method

You can apply text format for a cell or range of cells using format method. The following code example describes the above behavior,

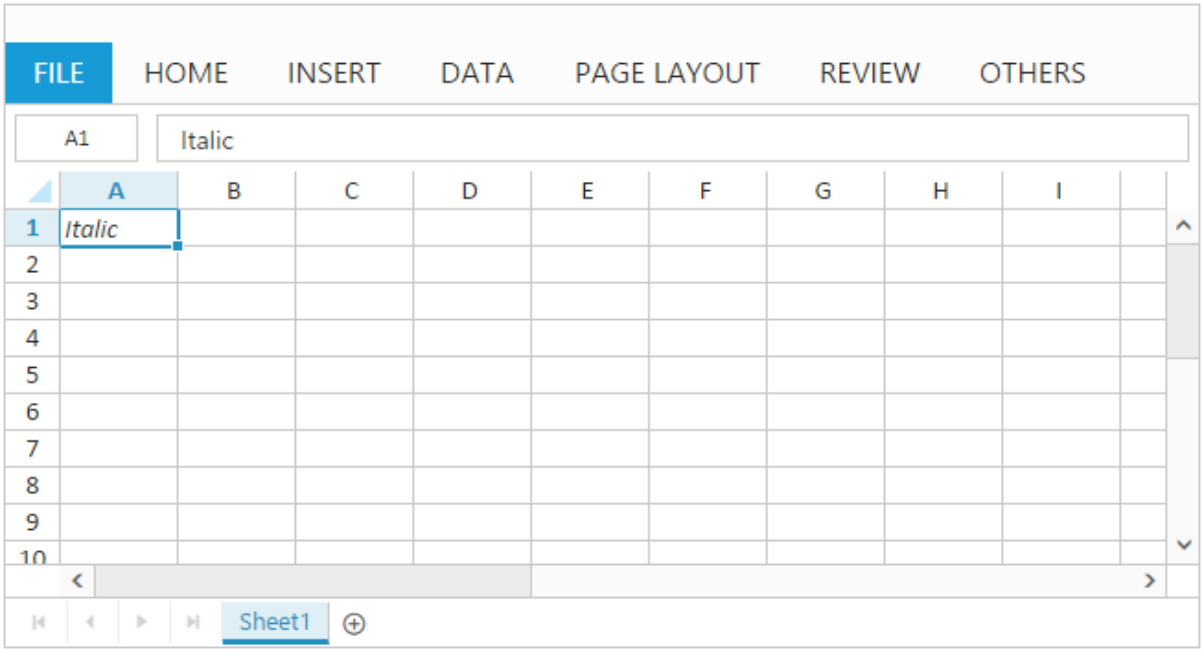
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("Italic").Add();
}).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script>
function loadComplete() {
this.XLFormat.format({ style: { "font-style": "italic" } }, "A1");
}
</script>
```

The following output is displayed as a result of the above code example.



User Interface

You can apply text format for a cell through text formatting options in ribbon HOME tab.

Wrap Text

To make text appearance on multiple lines in a cell, you can apply wrap text to the cell. So, that the text wraps automatically or you can enter a manual line break using ALT + ENTER key in edit mode. Use AllowWrap API to enable/disable wrap text. You can apply wrap text for a cell in following ways,

1) Method 2) User Interface

Method

You can wrap, text in a cell using wrapText method and it can be unwrap using unWrapText method. The following code example describes the wrapText behavior,

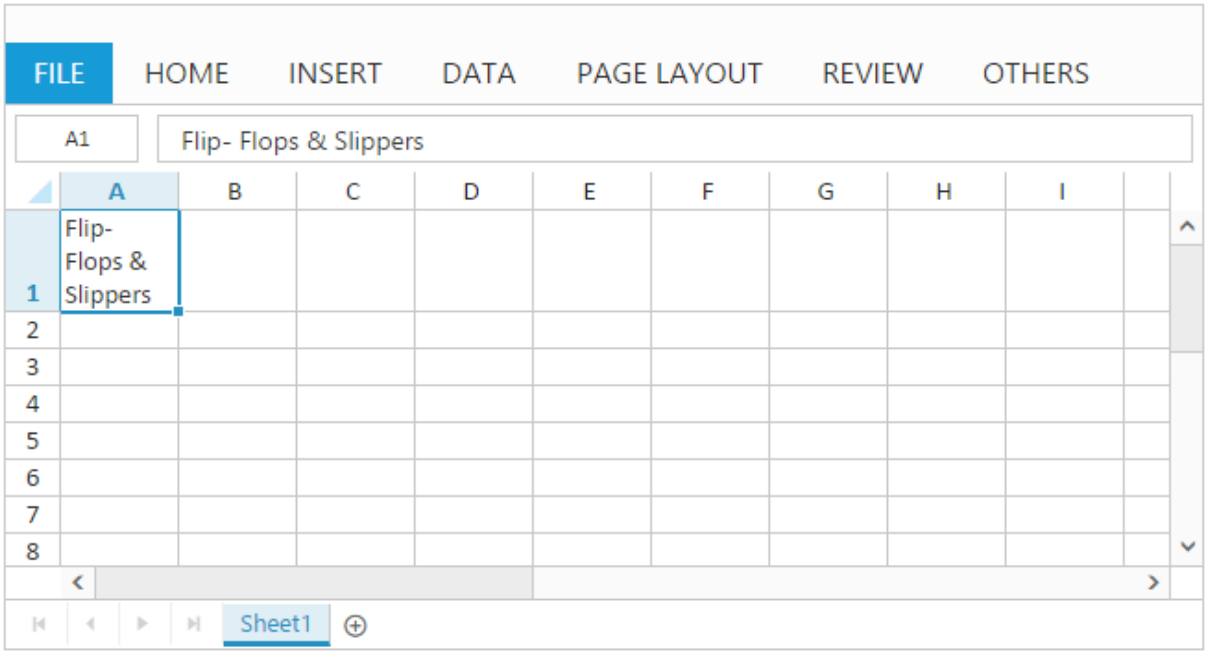
ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.Rows(rows =>
{
rows.Cells(cells =>
{
cells.Value("Flip-Flops & Slippers").Add();
}).Add();
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script>
function loadComplete() {
this.wrapText("A1");
//this.unWrapText("A1");
}
</script>
```

The following output is displayed as a result of the above code example.



User Interface

You can wrap or unwrap text in a cell using Wrap text option in ribbon HOME tab.

Cell Formatting

To highlight particular cell or section of cells from whole workbook you can use cell formatting options like borders, fill color etc.

Borders

You can add border around a cell or range of cells to define a section of worksheet or table. The different types of borders supported in Spreadsheet are,

1) Bottom Border 2) Top Border 3) Left Border 4) Right Border 5) All Borders 6) Outside Borders 7) Thick Box Border 8) Bottom Double Border 9) Thick Bottom Border 10) Top and Bottom Border 11) Top and Thick Bottom Border 12) Top and Bottom Double Border

You can apply border for a cell or range of cells through following ways,

1) Use format method to apply border via code 2) Apply border for a cell or range of cells using border options in ribbon HOME tab 3) Use Draw Border options in ribbon HOME tab

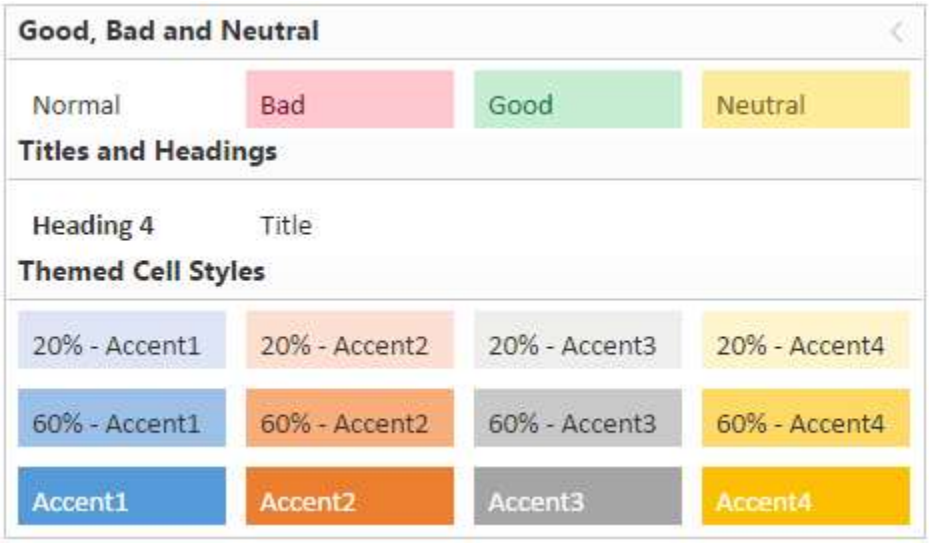
Copyright © 2001 - 2020 Syncfusion Inc.

Fill color

To highlight cell or range of cells from whole workbook you can apply background color for a cell using fill color option in Spreadsheet.

Cell Styles

Cell styles is a collection of pre-defined styles with multiple formatting characteristics to apply several formats in one step. You can apply cell style for a cell using Cell Styles option in ribbon HOME tab. The following screenshot illustrate this,



Format painter

The format painter lets you copy all of the formatting from a cell or range of cells and apply the same formatting to another cell or range of cells including font size, color, style etc.

Clear

Clear option is used to clear cell contents, formats or any attached comments from a cell or range of cells in worksheet. Use AllowClear API to enable/disable clear option in Spreadsheet.You have following clear options in Spreadsheet,

1) Clear All 2) Clear Formats 3) Clear Contents 4) Clear Comments 5) Clear Hyperlinks

Clear All

To clear content, format, comment, hyperlink etc. from a cell or range of cells, use Clear All option in Spreadsheet. Use clearAll method to clear cells via code.

Clear Formats

To clear formats in a cell or range of cells use Clear Formats option in Spreadsheet. Use clearAllFormat method to clear formats via code.

Clear Contents

To clear contents in a cell or range of cells use Clear Contents option in Spreadsheet. Use clearContents method to clear contents via code.

Clear Comments

To clear comment in a cell or range of cells use Clear Comments option in Spreadsheet. Use clearComments method to clear comments via code.

Copyright © 2001 - 2020 Syncfusion Inc.

Clear Hyperlinks

To clear hyperlink in a cell or range of cells use Clear Hyperlinks option in Spreadsheet. Use clearHyperlinks method to clear hyperlinks via code.

Scrolling

Scrolling helps you to move quickly to different areas of worksheet. Scrolling can be enabled by setting AllowScrolling as true in ScrollSettings.

You can scroll through worksheet using one of the following ways,

- Using the arrow keys
- Using the scroll bars
- Using the mouse

Set height and width for Scrolling

To set height and width in spreadsheet use Height and Width property in ScrollSettings. The default value for Height and Width is 100%.

The height and width can be set in percentage or pixel.

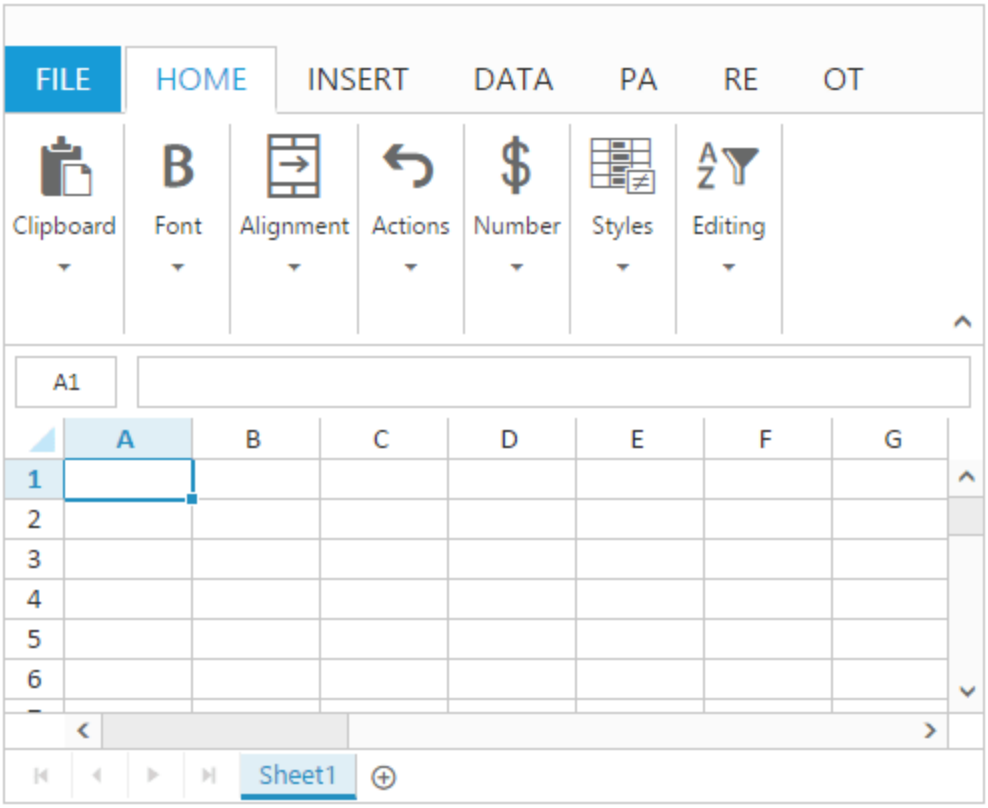
The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.ScrollSettings(scroll =>
{
scroll.AllowScrolling(true);
scroll.Height(400); // Height in pixel
scroll.Width("50%"); // Width in percentage
})
)
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Responsive

Spreadsheet has support for responsive behavior based on client browser's width and height. To enable responsive, set `IsResponsive` property in `ScrollSettings` as `true`. The three modes of responsive layout available in Spreadsheet based on client width are,

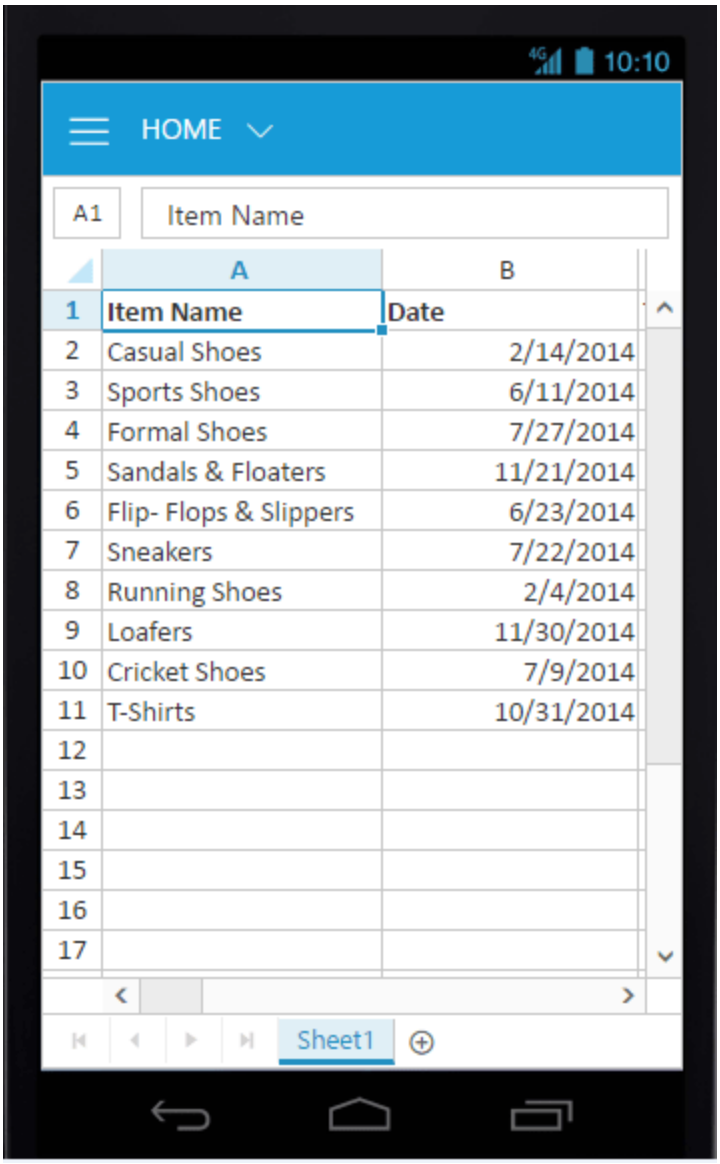
- Mobile(<420px)
- Tablet (420px to 617px)
- Desktop(>617px)

Note: Default value of `IsResponsive` property is `true`.

Mobile Mode

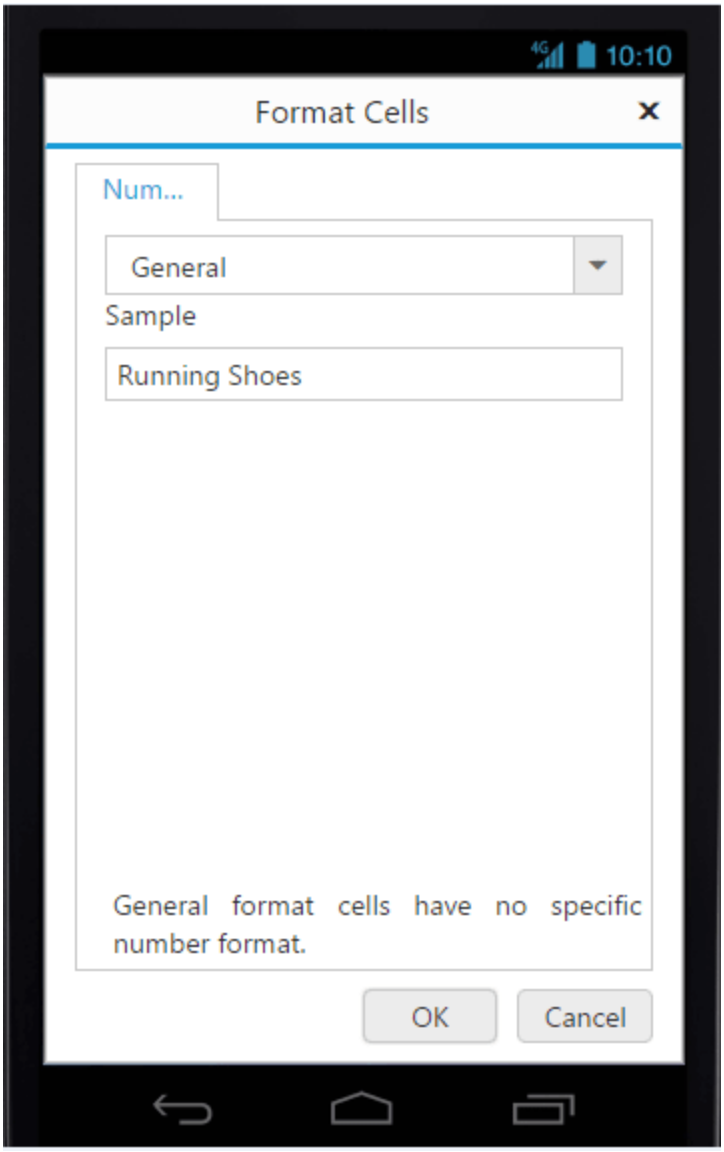
If client width is less than 420px, the spreadsheet will render in mobile mode. In which, you can see that spreadsheet user interface is customized and redesigned for best view in small screens. The customized feature includes filter dialog, format dialog, chart type dialog and ribbon.

Copyright © 2001 - 2020 Syncfusion Inc.



Ribbon in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.

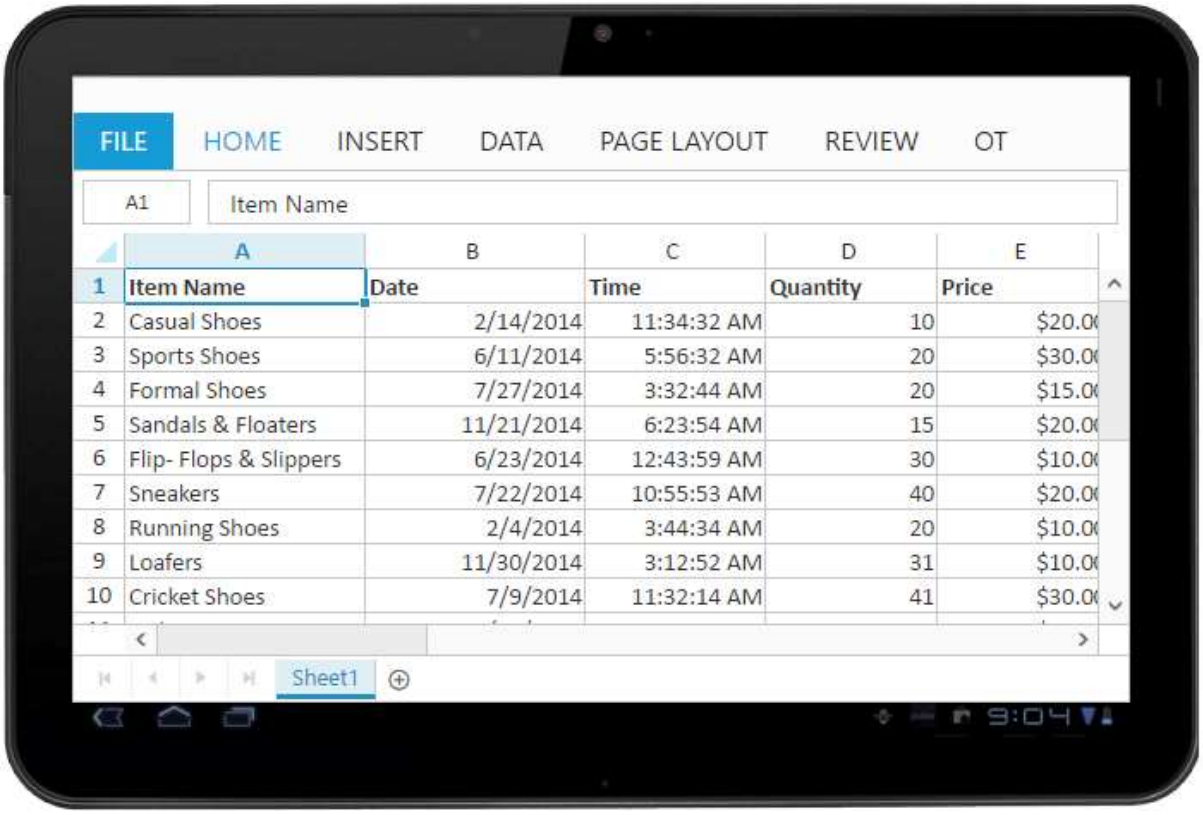


Format cell dialog in mobile layout.

Tablet Layout

If the client width is between 420px and 617px, then the spreadsheet will render in tablet mode. Also it has customized the dialogs to match tablet screen size.

Copyright © 2001 - 2020 Syncfusion Inc.



Ribbon in tablet layout.

Scroll Mode

Spreadsheet supports two type of modes in scrolling. You can use ScrollMode property in ScrollSettings to specify the mode of scrolling.

- Normal - This mode doesn't create new row/column when the scrollbar reaches the end.
- Infinite - This mode creates new row/column when the scrollbar reaches the end.

Note: Default value of ScrollMode property is Infinite mode.

Virtual Scrolling

Spreadsheet has support for virtual scrolling. This allows you to load data that you require (load data based on viewport size) without buffering the entire huge database. You can set AllowVirtualScrolling property in ScrollSettings as true to enable virtual scrolling.

Note: Default value of AllowVirtualScrolling property is true.

Clipboard

The Spreadsheet provides support for the clipboard operations (cut, copy, and paste). Clipboard operations can be enabled or disabled by setting AllowClipboard property in Spreadsheet.

By default AllowClipboard property is true.

Cut

This function cuts the selected range values and make it available in clipboard.

Copyright © 2001 - 2020 Syncfusion Inc.

You can do this by one of the following ways.

- Using "Ctrl + X" key.
- Using Cut button of HOME tab in ribbon to perform cut operation.
- Using Cut option in Context Menu.
- Using cut method.

Copy

This function copies the selected range values and make it available in clipboard.

You can do this by one of the following ways.

- Using "Ctrl + C" key.
- Using Copy button of HOME tab in ribbon to perform copy operation.
- Using Copy option in Context Menu.
- Using copy method.

Paste

This function pastes the clipboard content to newly selected range. If you perform cut paste, clipboard contents are cleared whereas in copy paste the clipboard contents are maintained.

You have following options in Paste.

- Paste Special - You can paste the values with formatting.
- Paste - You can paste only the values.

Note: The default paste option is Paste Special. This is working only within the current Spreadsheet. If you copy the content from other sources, it will paste only the values in the Spreadsheet.

You can do this by one of the following ways,

- Using "Ctrl + V" key.
- Using Paste button of HOME tab in ribbon to perform paste operation.
- Using Paste option in Context Menu.
- Using paste method.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowClipboard(true)
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

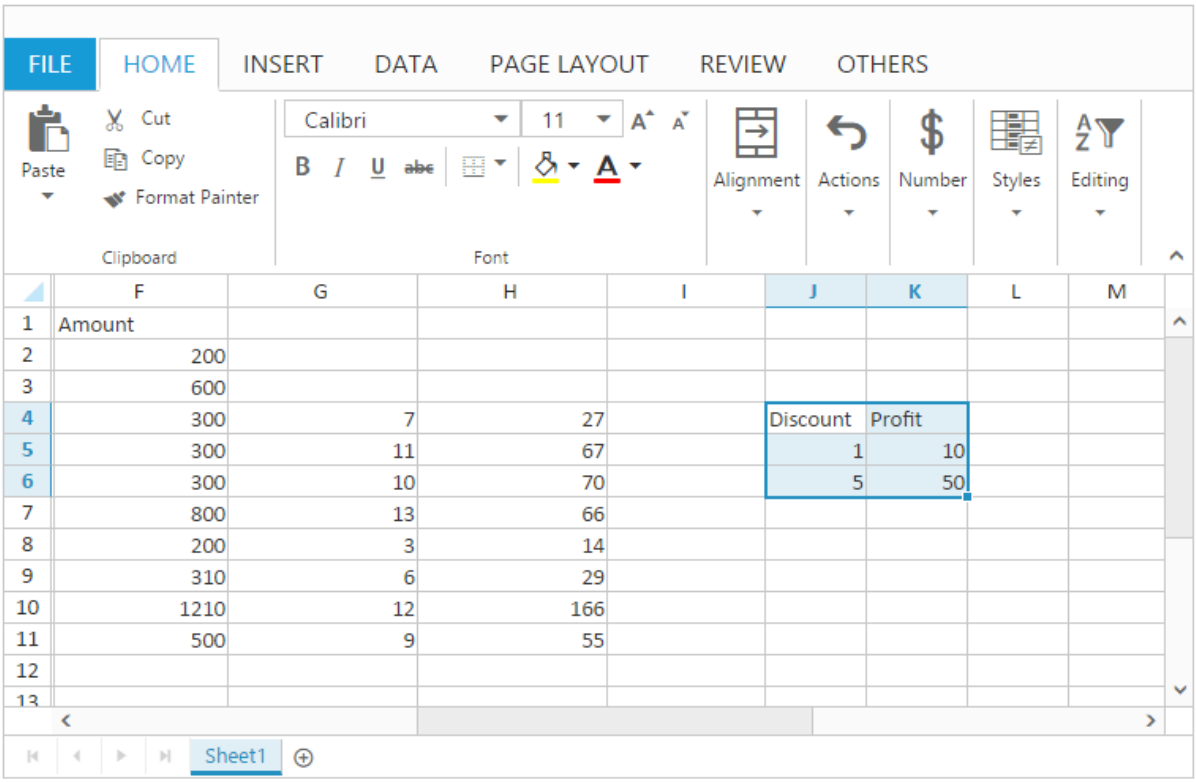
```
function loadComplete() {
var excelClip = this.XLClipboard;
this.performSelection("G1:H3");
excelClip.cut(); // Cut the selected cells
//excelClip.copy();//Copy the selected cells.
this.performSelection("J4");
excelClip.paste();
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Note: Similarly you can perform clipboard operations for shapes (Chart and Image).

Searching

This feature is used to search the contents in the Spreadsheet. You can use AllowSearching property to enable or disable this feature.

You have following options in Searching.

- Find
- Replace
- GoTo
- GoTo Special

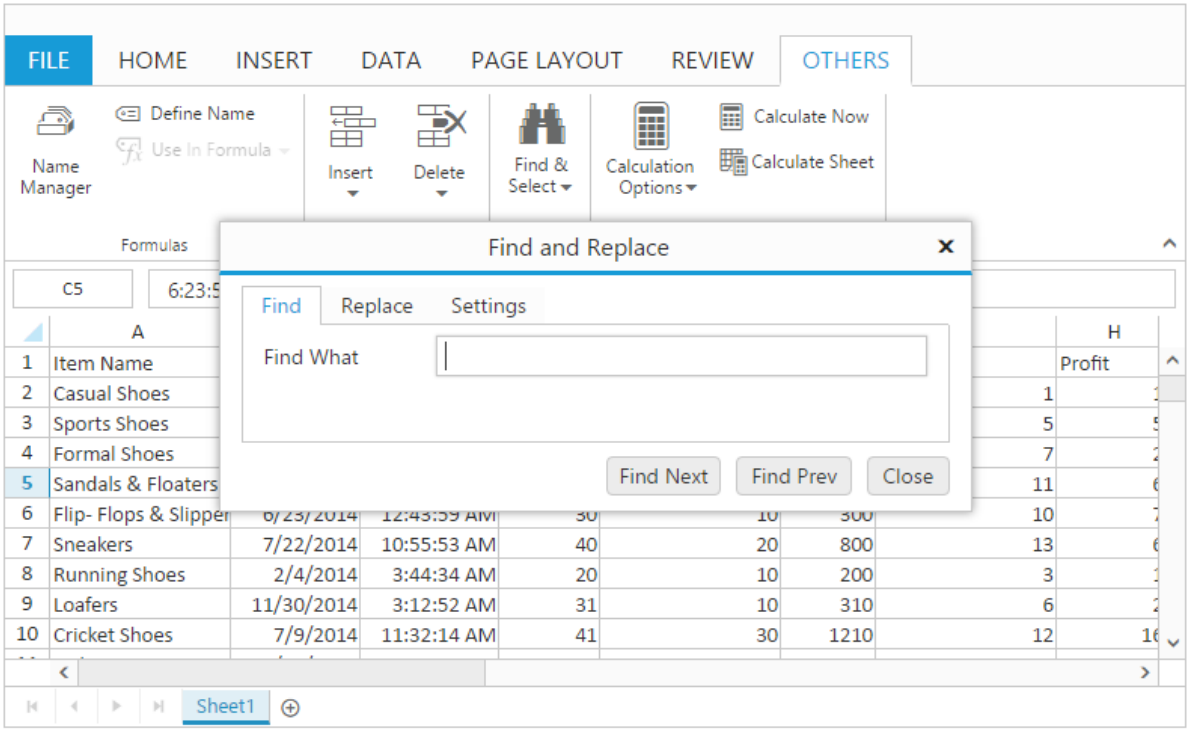
Find

This is used to search the contents of a cell. You can do this by one of the following ways,

- Using "Find" option in Find and Select button of OTHERS tab in ribbon to open the Find and Replace dialog.
- Using findNext and findPrevious methods to search the given value in workbook.
- Using "Ctrl + F" key to open the Find and Replace dialog with Find tab enabled.

The following output is displayed as a result of Find and Replace dialog with Find tab enabled.

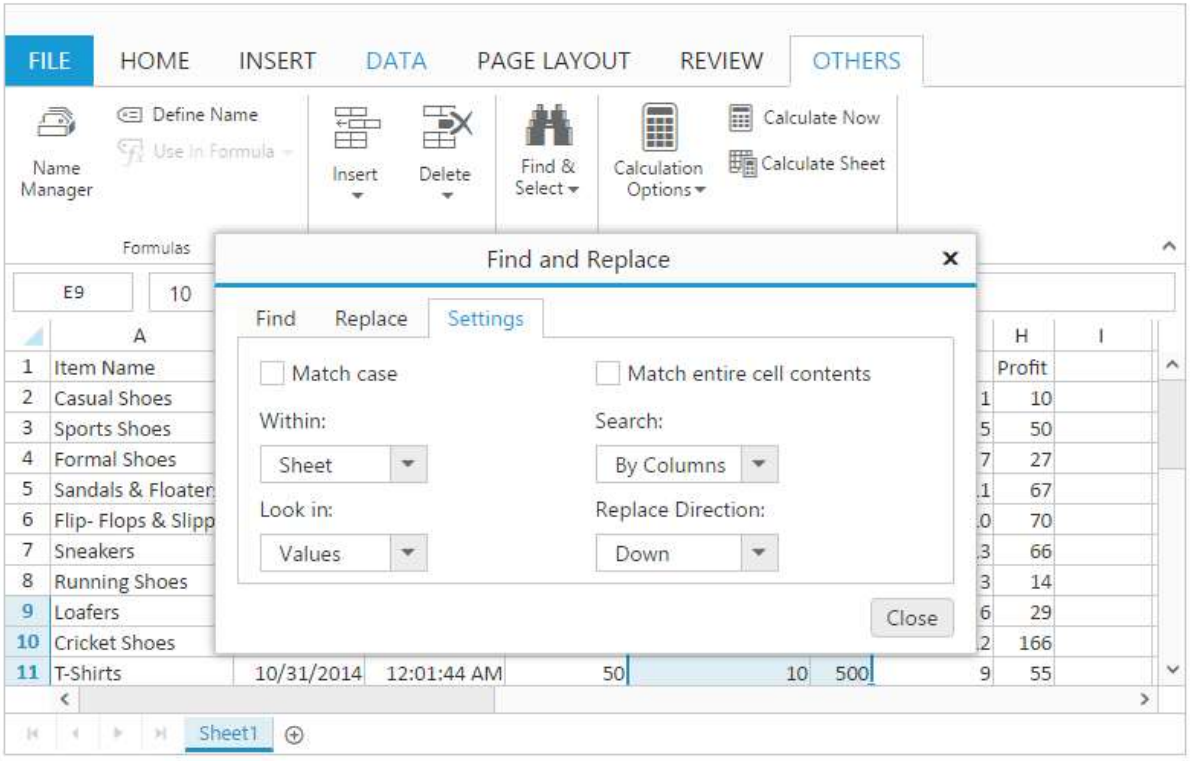
Copyright © 2001 - 2020 Syncfusion Inc.



You can customize the following settings in find option.

- Match Case - Search content should have same casing compared with the cell content.
- Match Entire Cell Content - Search content should match with entire cell content.
- Look in - You can search the contents by values, formulas and comments.
- Within - You can search the content within the sheet or workbook.
- Search - You can search the contents by rows or columns.
- Replace Direction - You can replace the contents by up or down direction.

Copyright © 2001 - 2020 Syncfusion Inc.



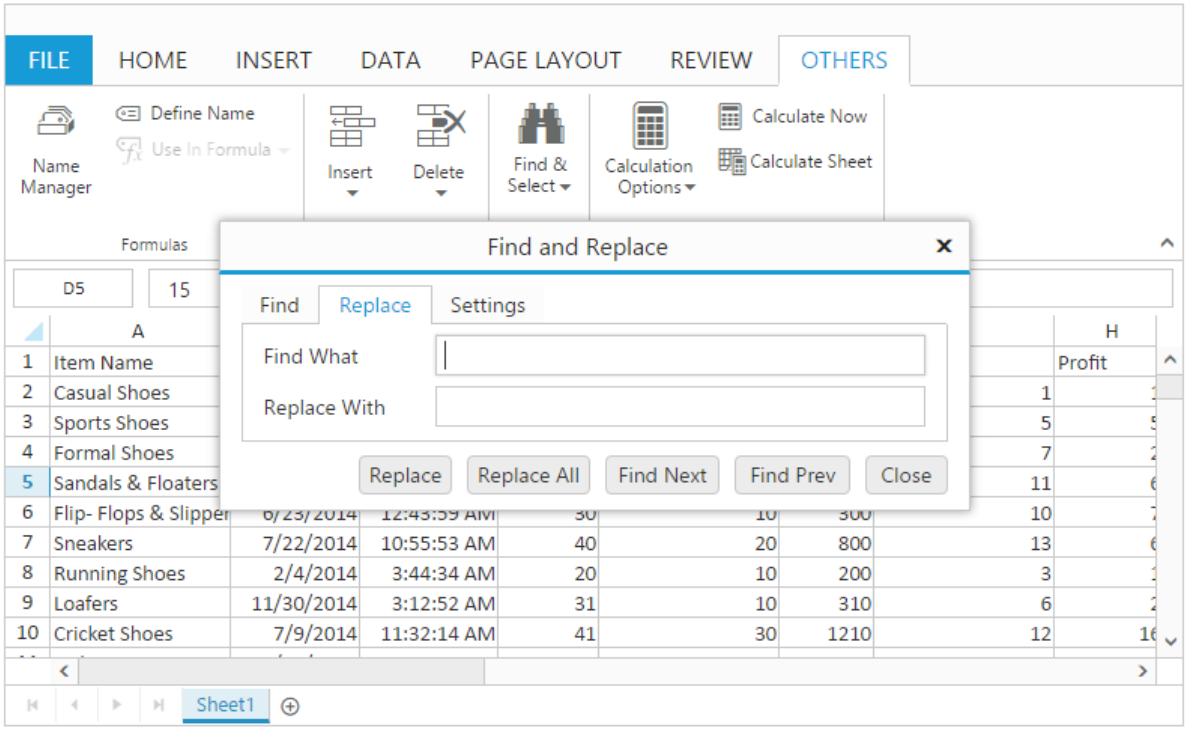
Find and Replace dialog with Settings tab

Replace

This is used to replace the contents of a cell. You can do this by one of the following ways.

- Using "Replace" option in Find and Select button of OTHERS tab in ribbon to open the Find and Replace dialog.
- Using replaceAllByBook or replaceAllBySheet method to replace the contents.
- Using "Ctrl + R" key to open the Find and Replace dialog with Replace tab enabled.

Copyright © 2001 - 2020 Syncfusion Inc.



Find and Replace dialog with Replace tab

The following code example describes the above behavior.

ASPX-CS

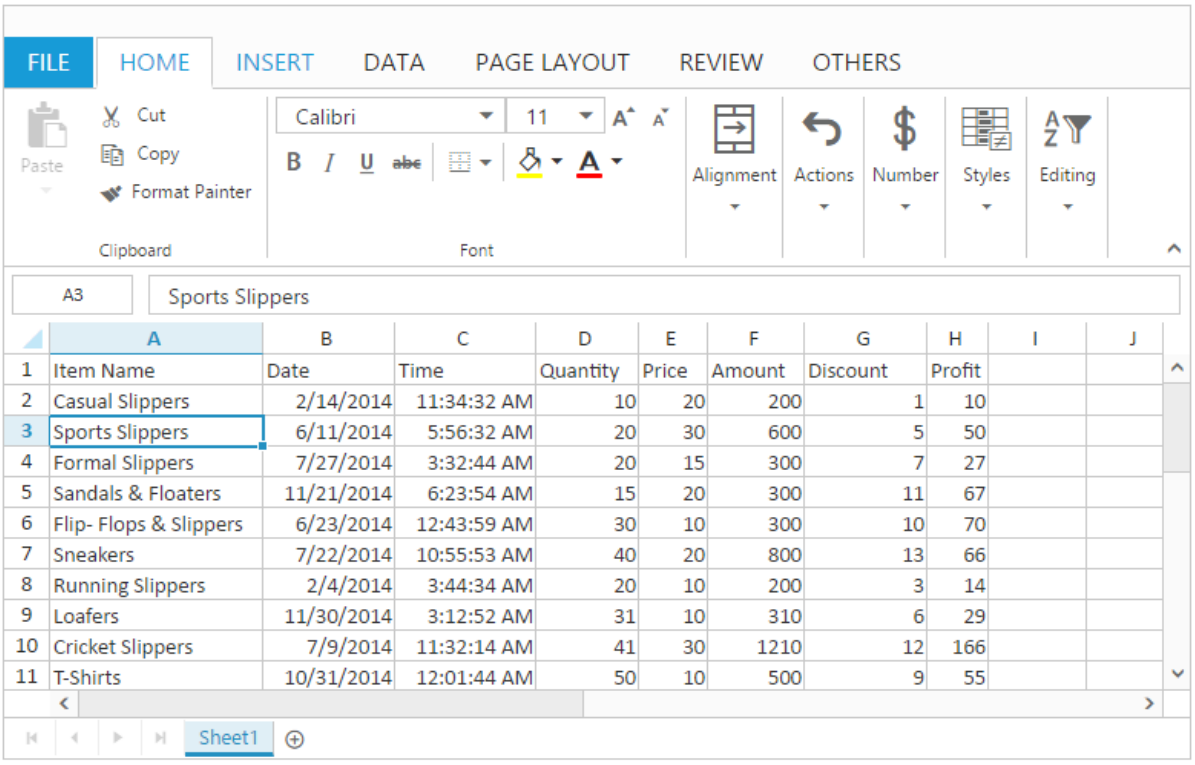
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script type="text/javascript">
function loadComplete(args) {
var excelSearch = this.XLSearch;
if (!this.isImport) {
excelSearch.replaceAllBySheet("Shoes", "Slippers", true, false);
//excelSearch.replaceAllByBook("Shoes", "Slippers", true, false);
}
}
</script>
```

C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
```

```
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.



	A	B	C	D	E	F	G	H	I	J
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit		
2	Casual Slippers	2/14/2014	11:34:32 AM	10	20	200	1	10		
3	Sports Slippers	6/11/2014	5:56:32 AM	20	30	600	5	50		
4	Formal Slippers	7/27/2014	3:32:44 AM	20	15	300	7	27		
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67		
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70		
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66		
8	Running Slippers	2/4/2014	3:44:34 AM	20	10	200	3	14		
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29		
10	Cricket Slippers	7/9/2014	11:32:14 AM	41	30	1210	12	166		
11	T-Shirts	10/31/2014	12:01:44 AM	50	10	500	9	55		

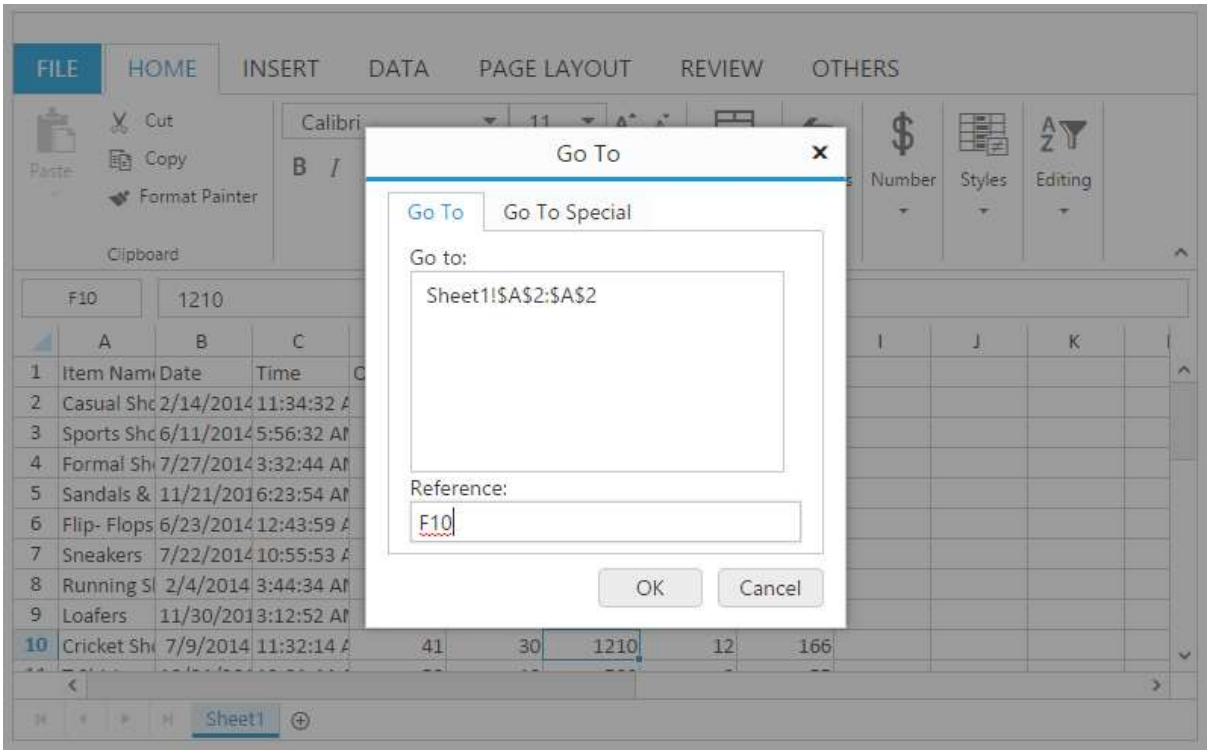
Go To

This feature is used to navigate to the particular cell in the worksheet or workbook.

Copyright © 2001 - 2020 Syncfusion Inc.

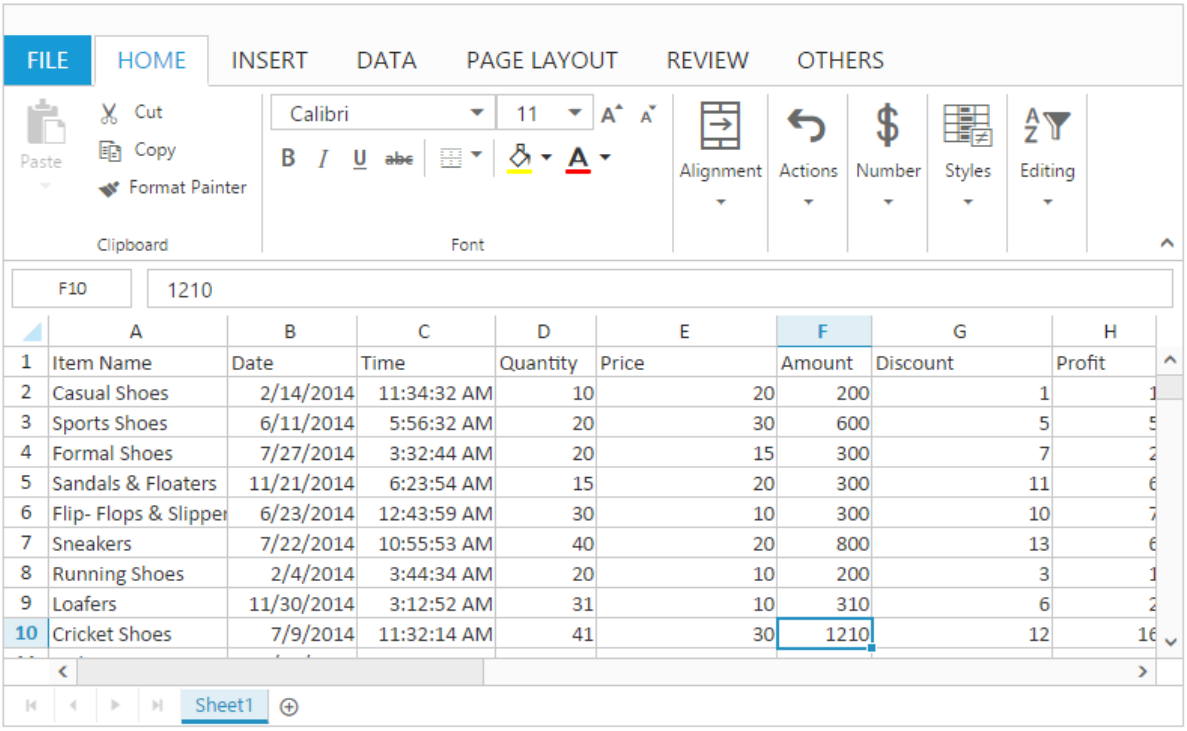
You can do this by one of the following ways,

- Using "Go To" option in Find and Select button of OTHERS tab in ribbon to open the Go To dialog.
- Using "Ctrl + G" key to open the Go To dialog.



Go To dialog

The following output is displayed as a result of goto to cell "F10".



	A	B	C	D	E	F	G	H
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	1
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	5
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	2
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	6
6	Flip- Flops & Slipper	6/23/2014	12:43:59 AM	30	10	300	10	7
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	6
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	1
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	2
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	16

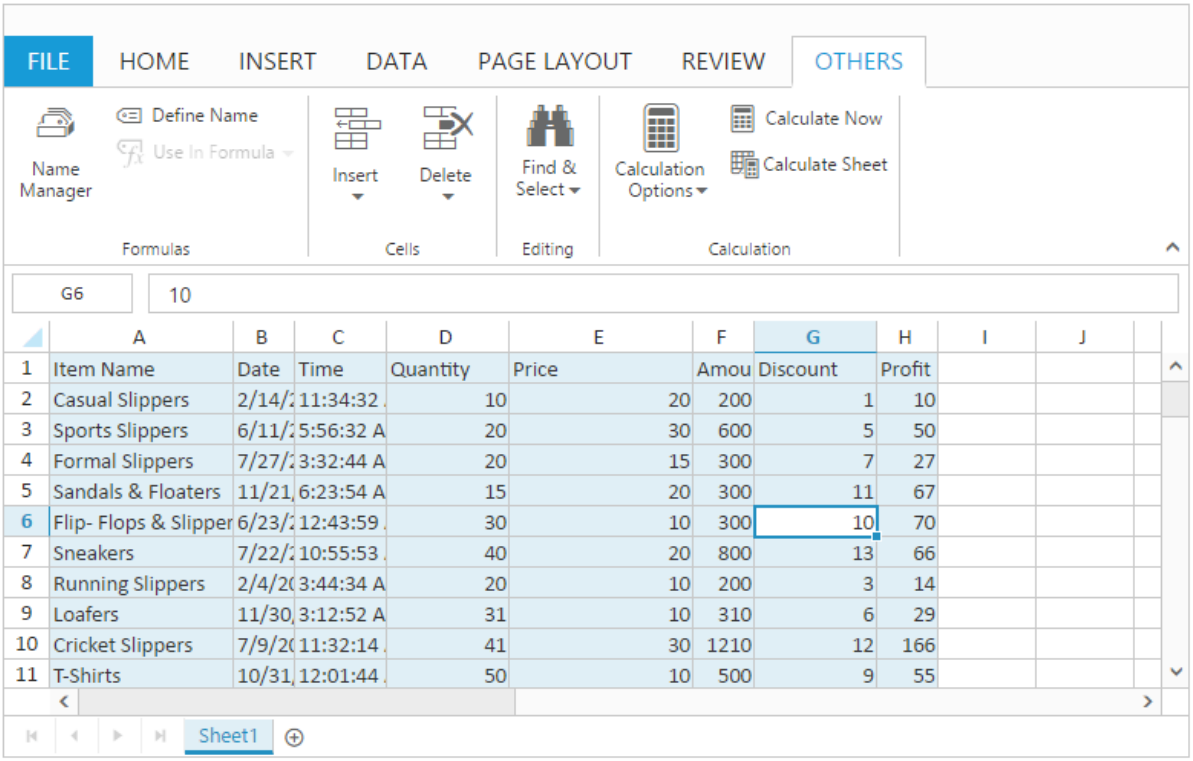
GoTo Special

This feature is used to quickly select cells of a specified type within the worksheet. You can do this by following ways,

- Using "Go to Special" option in Find and Select button of OTHERS tab in ribbon to open the Go To dialog.
- Using "Go to Formulas" option in Find and Select button of OTHERS tab in ribbon to highlight the cells, which contains formulas.
- Using "Go to Comments" option in Find and Select button of OTHERS tab in ribbon to highlight the cells, which contains comments.
- Using "Go to Conditional Formats" option in Find and Select button of OTHERS tab in ribbon to highlight conditional format applied cells.
- Using "Go to Data Validation" option in Find and Select button of OTHERS tab in ribbon to highlight the data validation applied cells.
- Using "Go to Constants" option in Find and Select button of OTHERS tab in ribbon to highlight the cells, which contains constant values.

The following output is displayed as a result of goto constants which selects the cells containing constant values.

Copyright © 2001 - 2020 Syncfusion Inc.



	A	B	C	D	E	F	G	H	I	J
1	Item Name	Date	Time	Quantity	Price	Amou	Discount	Profit		
2	Casual Slippers	2/14/2	11:34:32	10	20	200	1	10		
3	Sports Slippers	6/11/2	5:56:32 A	20	30	600	5	50		
4	Formal Slippers	7/27/2	3:32:44 A	20	15	300	7	27		
5	Sandals & Floaters	11/21/	6:23:54 A	15	20	300	11	67		
6	Flip- Flops & Slipper	6/23/2	12:43:59	30	10	300	10	70		
7	Sneakers	7/22/2	10:55:53	40	20	800	13	66		
8	Running Slippers	2/4/20	3:44:34 A	20	10	200	3	14		
9	Loafers	11/30/	3:12:52 A	31	10	310	6	29		
10	Cricket Slippers	7/9/20	11:32:14	41	30	1210	12	166		
11	T-Shirts	10/31/	12:01:44	50	10	500	9	55		

Undo and Redo

Spreadsheet provides the support to perform undo and redo operations. You can set `AllowUndoRedo` property as `true` to enable undo redo feature. You can also use `UndoredoStep` property to limit the undo redo action.

Note: Default value of `UndoredoStep` property is 20. You can perform 20 undo or redo actions.

Undo the last action

Undo reverses the last action you performed with spreadsheet. You can do this by following ways.

- Use Undo button of HOME tab in ribbon.
- Use "Ctrl + Z" key.

Redo the action

Redo reverses the last undo action you performed with spreadsheet. You can do this by following ways.

- Use Redo button of HOME tab in ribbon.
- Use "Ctrl + Y" key.

Print

This feature is used to produce text and graphics from a computer screen or file into a sheet of paper. You can use `AllowPrinting` property in `PrintSettings` to enable/disable the print operation.

You have following options in printing.

- Print Sheet / Print Selected Area.

Copyright © 2001 - 2020 Syncfusion Inc.

- Print with Gridlines
- Print with Headers

Print Sheet / Print Selected Area

You can print the sheet or print the selected area by following ways,

- Using the Print button or Print Selected button under Print group of PAGE LAYOUT tab in ribbon.
- Using printSheet or printSelection method to print the spreadsheet.

The following code example describes the above behavior.

ASPX-CS

```
@Html.EJ().Button("btnPrint").Size(ButtonSize.Medium).ShowRoundedCorner(t
rue).Width("80").ClientSideEvents(e =>
e.Click("printClick")).Text("Print")
@Html.EJ().Button("btnPrintSel").Size(ButtonSize.Medium).ShowRoundedCorne
r(true).ClientSideEvents(e => e.Click("printSelClick")).Text("Print
Selected")
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.PrintSettings(printOption => {
printOption.AllowPrinting(true);
})
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).Add();
})
)
<script type="text/javascript">
function printClick(args) {
var excelObj = $("#Spreadsheet").ejSpreadsheet("instance");
excelObj.XLPrint.printSheet();
}
function printSelClick(args) {
var excelObj = $("#Spreadsheet").ejSpreadsheet("instance");
excelObj.XLPrint.printSelection();
}
</script>
```

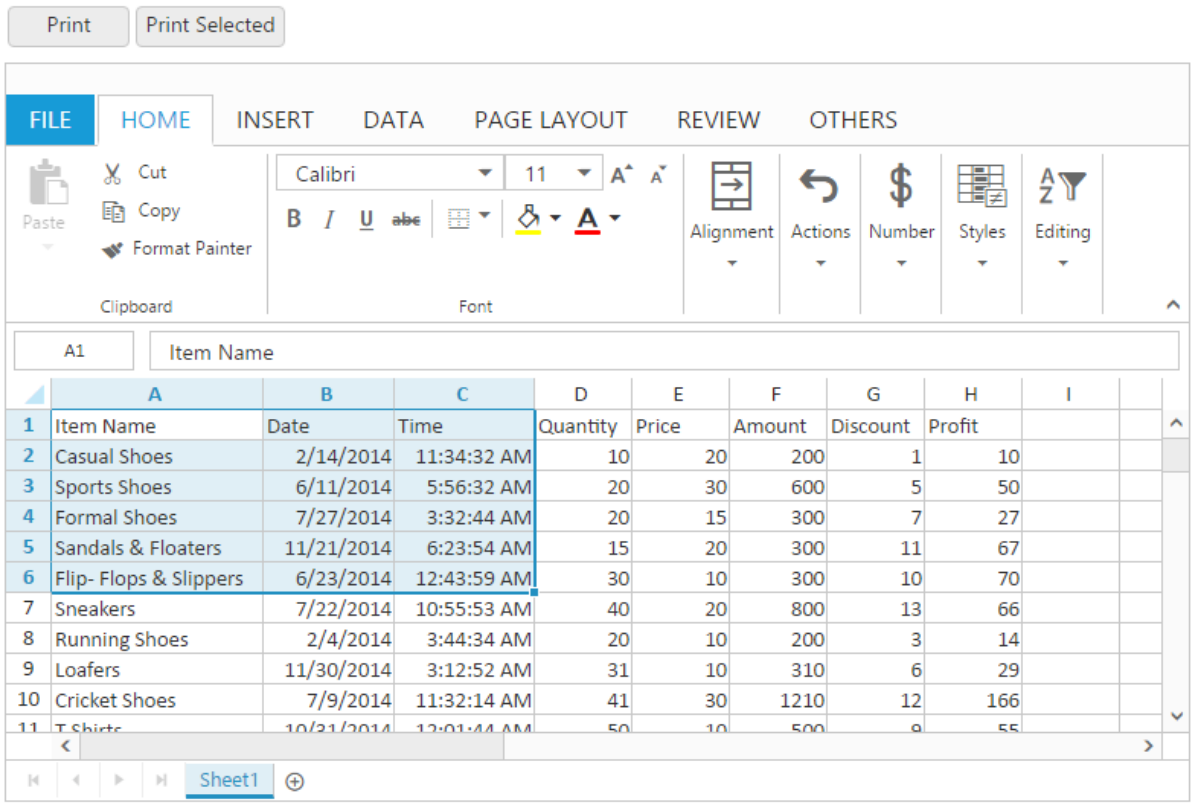
C#

```
namespace MVCSampleBrowser.Controllers
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

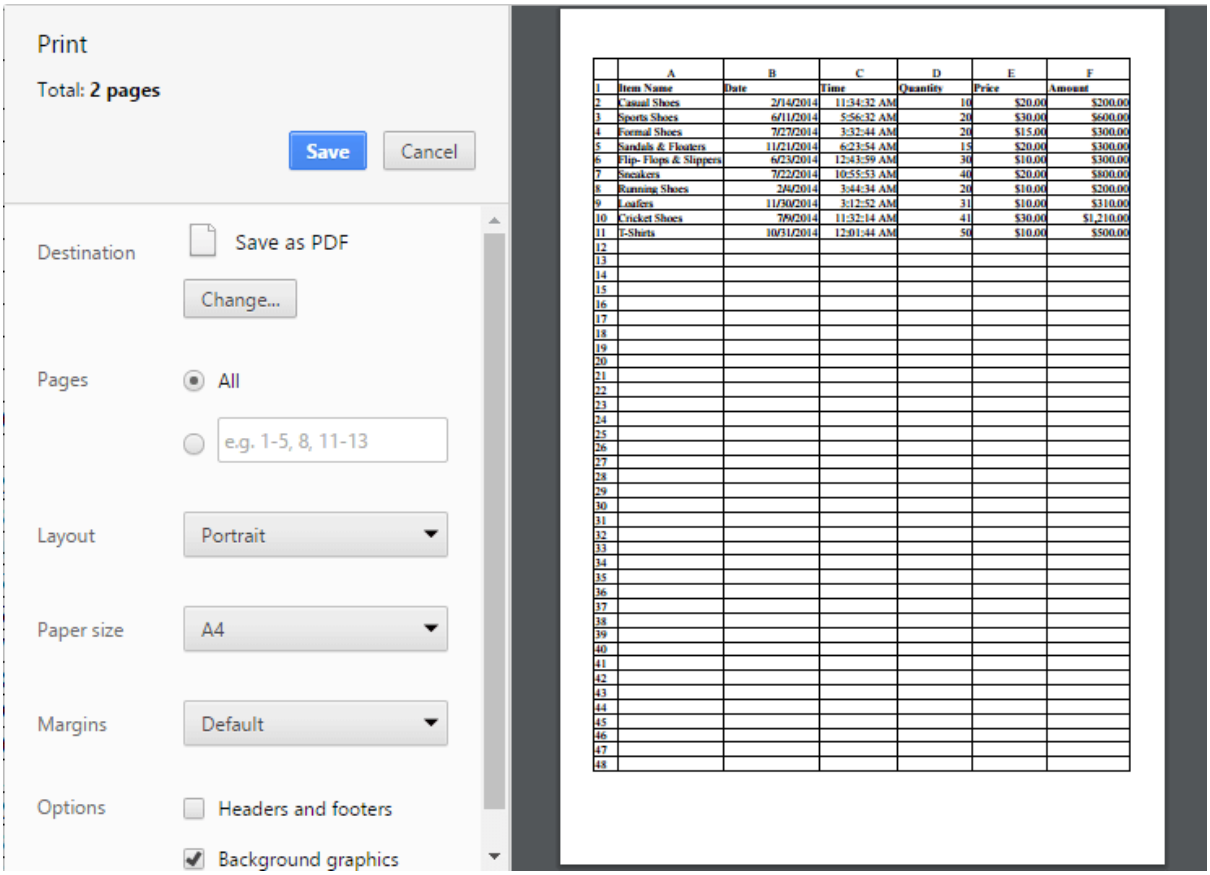
```
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

The following output is displayed as a result of the above code example.

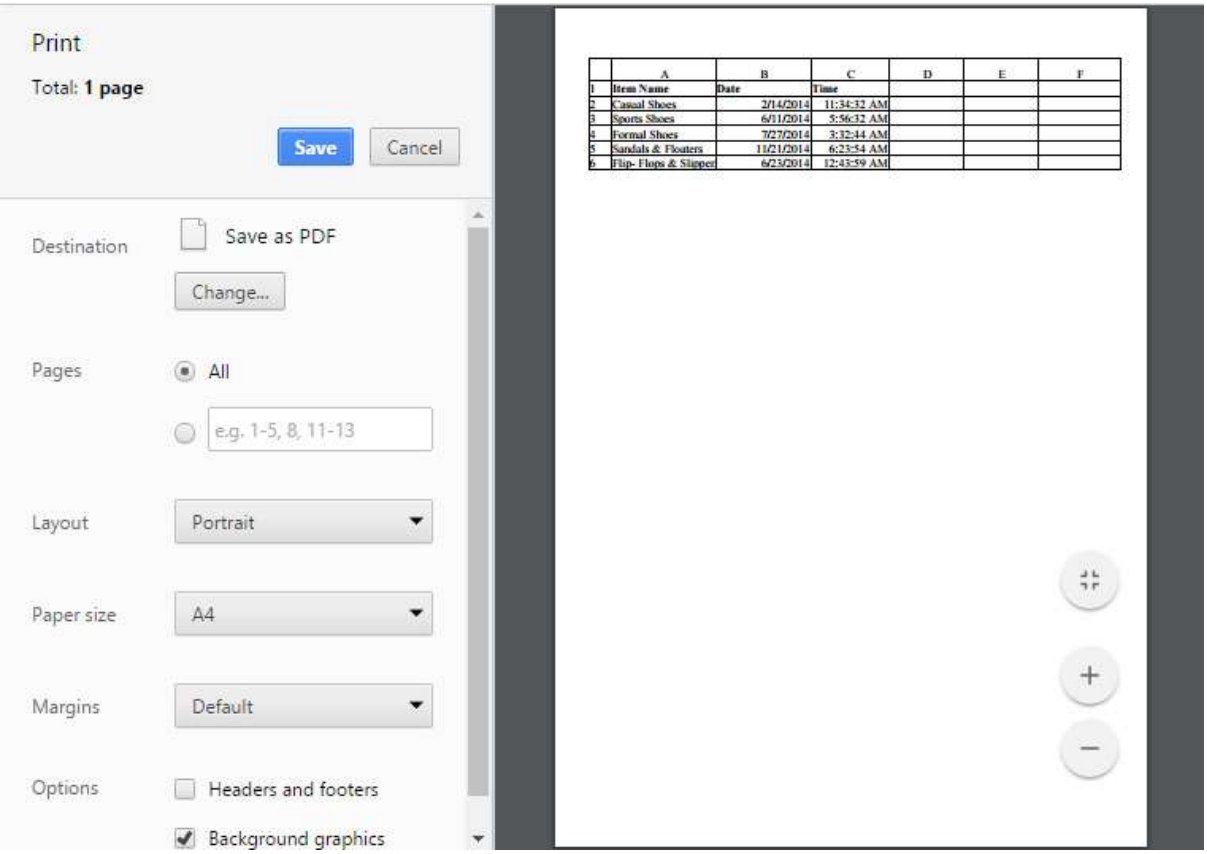


	A	B	C	D	E	F	G	H	I
1	Item Name	Date	Time	Quantity	Price	Amount	Discount	Profit	
2	Casual Shoes	2/14/2014	11:34:32 AM	10	20	200	1	10	
3	Sports Shoes	6/11/2014	5:56:32 AM	20	30	600	5	50	
4	Formal Shoes	7/27/2014	3:32:44 AM	20	15	300	7	27	
5	Sandals & Floaters	11/21/2014	6:23:54 AM	15	20	300	11	67	
6	Flip- Flops & Slippers	6/23/2014	12:43:59 AM	30	10	300	10	70	
7	Sneakers	7/22/2014	10:55:53 AM	40	20	800	13	66	
8	Running Shoes	2/4/2014	3:44:34 AM	20	10	200	3	14	
9	Loafers	11/30/2014	3:12:52 AM	31	10	310	6	29	
10	Cricket Shoes	7/9/2014	11:32:14 AM	41	30	1210	12	166	
11	T Shirts	10/21/2014	12:01:44 AM	50	10	500	9	55	

Copyright © 2001 - 2020 Syncfusion Inc.



Print dialog with entire sheet in Chrome browser



Print dialog with selected area print in chrome browser

Print with Gridlines

You have an option to print the spreadsheet with gridlines or without gridlines. You can do this by one of the following ways,

- Check or uncheck the Gridlines option under Show group of PAGE LAYOUT tab in ribbon to print with Gridlines.
- Using ShowGridlines property to enable / disable the gridlines.

The following code example describes the above behavior.

ASPX-CS

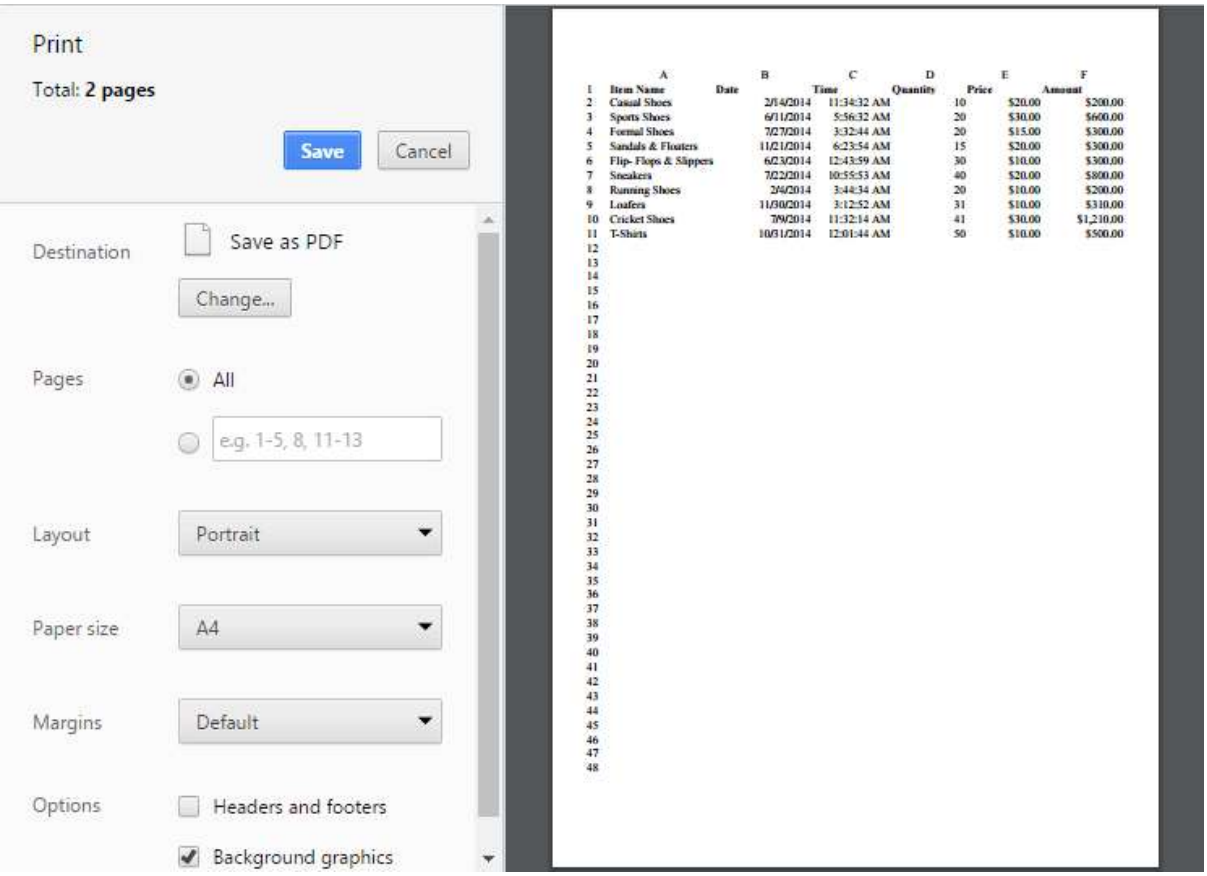
```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).ShowGridlines(false).Add();
})
)
```

C#

```
namespace MVCSampleBrowser.Controllers
```

```
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Print dialog without gridlines in chrome browser

Print with Headings

You have an option to print the spreadsheet with headers or without headers. You can do this by one of the following ways,

- Check or uncheck the Headings option under Show group of PAGE LAYOUT tab in ribbon.
- Using ShowHeadings property to enable / disable showHeadings.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).Add();
}).ShowHeadings(false).Add();
})
)
```

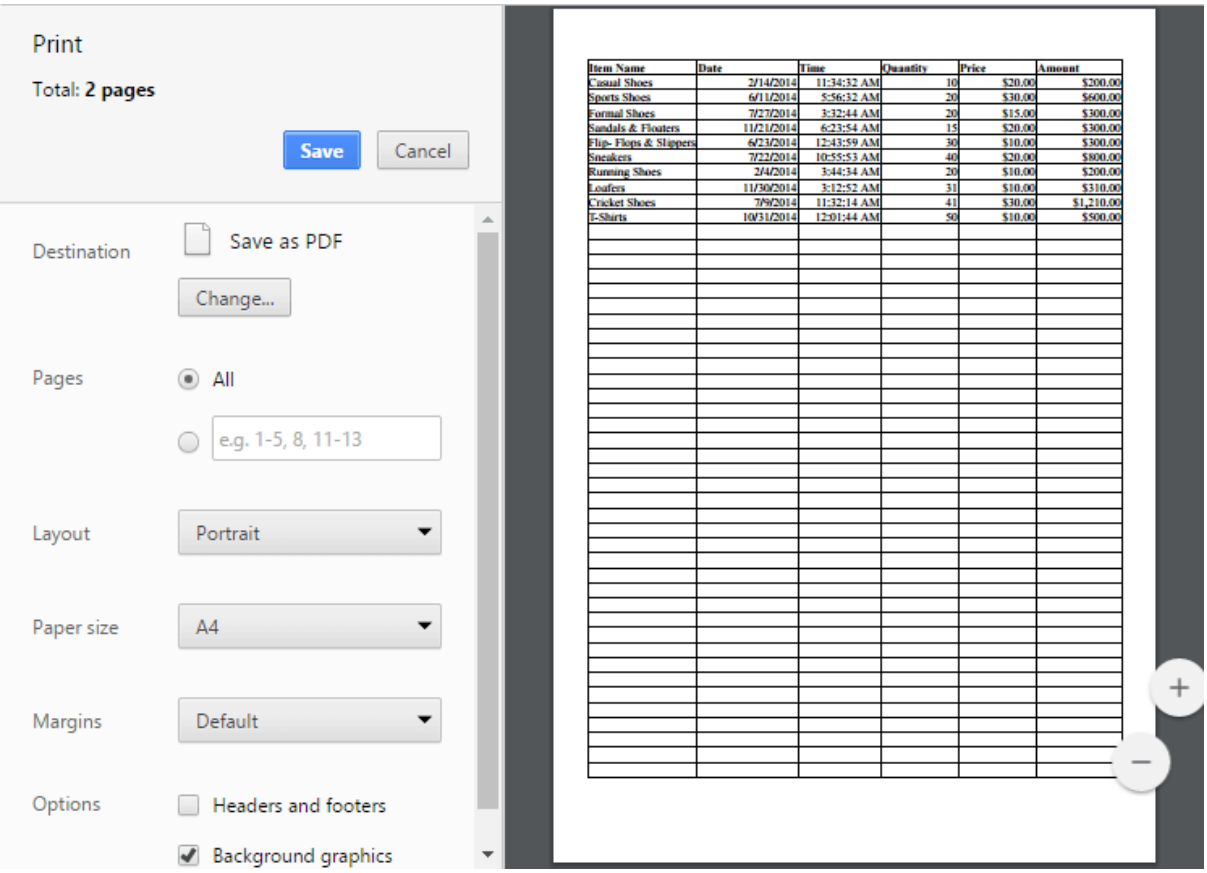
C#

```
namespace MVCSampleBrowser.Controllers
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public partial class SpreadsheetController : Controller
{
public ActionResult Default()
{
List<ItemDetail> lItems = new List<ItemDetail>();
lItems.Add(new ItemDetail() { ItemName = "Casual Shoes", Date = new
DateTime(2014, 02, 14), Time = new DateTime(2014, 02, 14, 11, 34, 32),
Quantity = 10, Price = 20, Amount = 200, Discount = 1, Profit = 10 });
lItems.Add(new ItemDetail() { ItemName = "Sports Shoes", Date = new
DateTime(2014, 06, 11), Time = new DateTime(2014, 06, 11, 05, 56, 32),
Quantity = 20, Price = 30, Amount = 600, Discount = 5, Profit = 50 });
lItems.Add(new ItemDetail() { ItemName = "Formal Shoes", Date = new
DateTime(2014, 07, 27), Time = new DateTime(2014, 07, 27, 03, 32, 44),
Quantity = 20, Price = 15, Amount = 300, Discount = 7, Profit = 27 });
lItems.Add(new ItemDetail() { ItemName = "Sandals & Floaters", Date = new
DateTime(2014, 11, 21), Time = new DateTime(2014, 11, 21, 06, 23, 54),
Quantity = 15, Price = 20, Amount = 300, Discount = 11, Profit = 67 });
lItems.Add(new ItemDetail() { ItemName = "Flip- Flops & Slippers", Date =
new DateTime(2014, 06, 23), Time = new DateTime(2014, 06, 23, 12, 43,
59), Quantity = 30, Price = 10, Amount = 300, Discount = 10, Profit = 70
});
ViewBag.Datasource = lItems;
return View();
}
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Print dialog without headings in chrome browser

Keyboard Shortcuts

Shortcut	Description
Alt + Enter	In Cell editing mode, starts a new line in the same cell
Alt + F1	Creates a Chart from the selected cells
Alt + T	Opens create Table dialog box
Backspace	Clears content of the active cell. Deletes one character to the left in the Formula Bar. In Cell editing mode, it deletes one character to the left of the insertion point.
Ctrl + \	To apply go to row difference for selected cells
Ctrl + 1	Opens Format cell dialog box
Ctrl + 5	Applies or removes strikethrough
Ctrl + 8	Hides the selected columns
Ctrl + 9	Hides the selected rows

Copyright © 2001 - 2020 Syncfusion Inc.

Ctrl + A	Selects the entire worksheet.If worksheet has data, Ctrl + A will selects current region. Pressing Ctrl + A second time selects the entire worksheet.
Ctrl + B	Applies or removes bold formatting
Ctrl + C	Copies the selected cells
Ctrl + D	Uses the Fill Down command to copy the contents and format
Ctrl + F3	Opens name manager dialog box
Ctrl + I	Applies or removes italic formatting
Ctrl + K	Opens the Insert Hyperlink dialog box for new hyperlinks or the Edit Hyperlink dialog box for selected existing hyperlinks
Ctrl + O	Displays Open dialog box to open a file
Ctrl + P	Opens Print dialog box
Ctrl + R	Uses the Fill Right command to copy the contents and format
Ctrl + S	Saves the current workbook
Ctrl + Space Bar	Selects the entire column of selected cells in a worksheet
Ctrl + U	Applies or removes underlining
Ctrl + Shift + &	Applies the outline border to the selected cells
Ctrl + Shift + _	Removes the outline border from the selected cells
Ctrl + Shift + ~	Applies the General format
Ctrl + Shift + $	Applies the Currency format
Ctrl + Shift + %	Applies the Percentage format
Ctrl + Shift + ^	Applies the Scientific format
Ctrl + Shift + #	Applies the Short Date format
Ctrl + Shift + @	Applies the Time format
Ctrl + Shift + !	Applies the Number format with two decimal places
Ctrl + Shift + 8	To show hidden column within the selection
Ctrl + Shift + 9	To show hidden row within the selection
Ctrl + Shift + D	Opens Delete dialog box
Ctrl + Shift + Home	Extends the selection of cells to the beginning of the worksheet
Ctrl + Shift + I	Opens Insert dialog box
Ctrl + Shift + F2	To expand or collapse ribbon content

Copyright © 2001 - 2020 Syncfusion Inc.

Ctrl + Shift + O	Highlight all cells that contain comments
Ctrl + Shift + |	To apply go to column difference for selected cells
Ctrl + V	Inserts the content of the clipboard based on active cell and replaces any selection. It can be performed after you have cut or copied cell contents.
Ctrl + X	Cuts the selected cells
Ctrl + Y	Repeats the last action, if possible
Ctrl + Z	Reverses the last action or to delete the last entry
Delete	Delete selected cells
Esc	Cancel edit mode and discard changes.
F2	Start edit
Shift + F2	Adds or edits a cell comment
Shift + F11	Inserts a new worksheet
Shift + Spacebar	Selects the entire row of selected cells in a worksheet

Sparkline

Sparkline are easy to interpret and it conveys much more information to the user by visualizing the data in a small amount of space. Here sparkline were placed inside the cell. It creates the Sparkline based on the selected cell range's data. You can use AllowSparkline property to enable/disable Sparkline.

Types of Sparkline

The Following Types of Sparkline are available in Spreadsheet.

Column Line * Win/loss

You can create the Sparkline by one of the following ways,

Using the Sparkline Type button to select the type of Sparkline under Sparkline groups of INSERT Tab in ribbon. Using createSparkline method to create the Sparkline.

Sparkline Customization

You can perform the following customizations for Sparkline. These options are available in SPARKLINE DESIGN Tab which is enabled, while clicking the Sparkline element.

Feature	API	Description
Edit Data & Location Range	createSparkline	You can modify the data range, location range of Sparkline
Sparkline Type	changeType	You can change the type of sparkline by using sparkline type button.

Copyright © 2001 - 2020 Syncfusion Inc.

High Point, First Point, Last Point, Low Point, Negative Point, Sparkline Color and Marker Color	changePointColor	You can high light the high point, low point, first point, last point, negative point and Marker Color of sparkline .

The following code example describes the above behavior,

ASPX-CS

```
@(Html.EJ().Spreadsheet<object>("Spreadsheet")
.AllowSparkline(true)
.PageSettings(setting => setting.PageCount(1))
.Sheets(sheet =>
{
sheet.RangeSettings(range =>
{
range.Datasource((IEnumerable<object>)ViewBag.Datasource).StartCell("A1")
.Add();
}) Add();
})
.ClientSideEvents(events => events.LoadComplete("loadComplete"))
)
<script>
function loadComplete() {
var  xlSparkline = this.XLSparkline;
if(!this.isImport) {
this.sheetRename("Sparkline Chart");
this.mergeCells("F3:G4", true);
xlSparkline.createSparkline("A5:D5", "F3", "Column", {highPointColor:
"red", negativePointColor: "black", startPointColor: "green"} );
this.mergeCells("F8:G8", true);
xlSparkline.createSparkline("B2:B4", "F8", "Winloss");
this.mergeCells("F10:G11", true);
xlSparkline.createSparkline("C3:C5", "F10",  "Line",
{markerSettings:{visible:true},highPointColor: "red", negativePointColor:
"black", startPointColor: "green"} );
}
}
</script>
```

C#

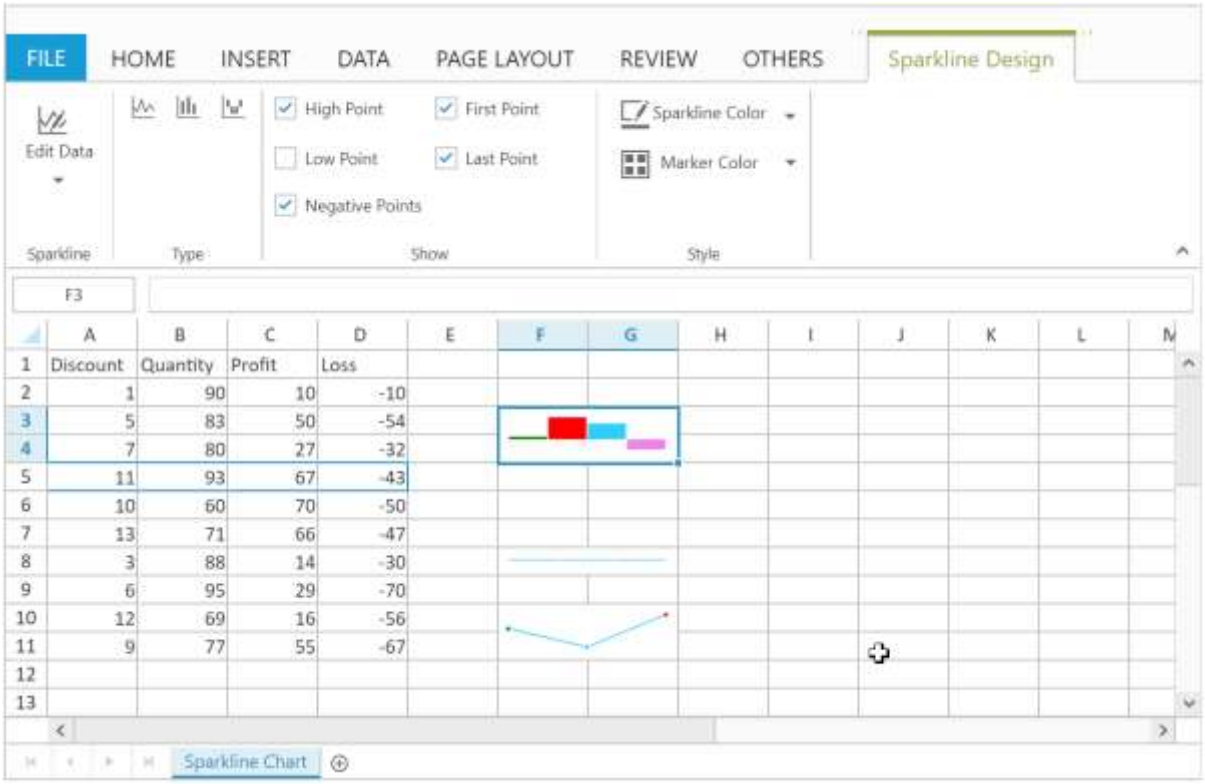
```
public ActionResult Index()
{
var dataSource = SparklineDetail.GetSparklineDetails.ToList();
ViewBag.DataSource = dataSource;
return View()
}
public IEnumerable<object> GetSparklineDetails
{
get
{
List<SparklineDetail> lItems = new List<SparklineDetail>();
lItems.Add(new SparklineDetail() { Discount = 1, Quantity = 90, Profit =
10, Loss = -10 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lItems.Add(new SparklineDetail() { Discount = 5, Quantity = 83, Profit =
50, Loss = -54 });
lItems.Add(new SparklineDetail() { Discount = 7, Quantity = 80, Profit =
27, Loss = -32 });
lItems.Add(new SparklineDetail() { Discount = 11, Quantity = 93, Profit =
67, Loss = -43 });
lItems.Add(new SparklineDetail() { Discount = 10, Quantity = 60, Profit =
70, Loss = -50 });
lItems.Add(new SparklineDetail() { Discount = 13, Quantity = 71, Profit =
66, Loss = -47 });
lItems.Add(new SparklineDetail() { Discount = 3, Quantity = 88, Profit =
14, Loss = -30 });
lItems.Add(new SparklineDetail() { Discount = 6, Quantity = 95, Profit =
29, Loss = -70 });
lItems.Add(new SparklineDetail() { Discount = 12, Quantity = 69, Profit =
16, Loss = -56 });
lItems.Add(new SparklineDetail() { Discount = 9, Quantity = 77, Profit =
55, Loss = -67 });
var itemCol =
from item in lItems
select new { item.Discount, item.Quantity, item.Profit, item.Loss };
return itemCol;
}
}
public class SparklineDetail
{
public int Discount { get; set; }
public int Quantity { get; set; }
public int Profit { get; set; }
public int Loss { get; set; }
}
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Sparkline

Overview

Sparklines are easy to interpret and also it conveys much more information to the user by visualizing the data in a small amount of space.

Some of the key features are,

- Consumes less amount of space for visualizing the data.
- Easy integration with grids.
- Five type of Sparkline available namely Line, Column, Area, Win-loss and Pie.
- Range Band to highlight selected start and end value region.
- Can customize markers for high, low, start, end and negative point.
- Trackball to get current mouse point to display tooltip.

Getting Started

This section explains you the steps required to populate the Sparkline with data. This section covers only the minimal features that you need to know to get started with the Sparkline.

Installation and Deployment

It describes you on how to configure the Syncfusion dependencies in your ASP.NET MVC application to create a Sparkline.

Referencing Syncfusion Assemblies

The following assemblies need to be referenced in your application for using Essential Sparkline MVC

Copyright © 2001 - 2020 Syncfusion Inc.

1. Syncfusion.EJ.dll
2. Syncfusion.EJ.MVC.dll

Follow the steps given below to deploy the application in the development server by referencing the assembly in **GAC**.

- Web.config file should be configured according to the referenced assemblies.
- To deploy your application, you have to ensure that the above referenced assemblies (in your web.config files) are present in the GAC.

```
<system.web>
  <compilation debug="true" targetFramework="4.5.1">
    <assemblies>
      <add assembly="Syncfusion.EJ, Version=13.3400.0.7, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
      <add assembly="Syncfusion.EJ.Mvc, Version=13.3400.0.7, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
    </assemblies>
  </compilation>
```

And, add Syncfusion namespace in the Web.config file of Views folder to generate SunburstChart builder code in View page.

```
<system.web.webPages.razor>
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="Syncfusion.JavaScript, Version=13.3400.0.7, Culture=neutral,
                                                PublicKeyToken=3d67ed1f87d44c89" />
      <add namespace="Syncfusion.JavaScript.DataVisualization, Version=13.3400.0.7,
                                                Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
      <add namespace="Syncfusion.MVC.EJ, Version=13.3400.0.7, Culture=neutral,
                                                PublicKeyToken=3d67ed1f87d44c89" />

    </namespaces>
  </pages>
</system.web.webPages.razor>
```

Note: If you are referring Syncfusion dll's manually from bin folder to create MVC SunburstChart, then remove Culture, Version and PublicKeyToken attributes used in all, and nodes.

Adding script reference

Add the below scripts references and Syncfusion **ScriptManager** to execute the script as mentioned in the following code snippet in layout page.

ASPX-CS

```
<!--  jquery script  -->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<!-- Essentialcshtml UI widget -->
<script
src="http://cdn.syncfusion.com/{{site.releaseversion}}/js/web/ej.web.all.
min.js"></script>
<!--Add Syncfusion Script Manager -->
@Html.EJ().ScriptManager()
```

To render the MVC Chart in unobtrusive mode refer **ej.unobtrusive.js** script file. Otherwise set false to **UnobtrusiveJavaScriptEnabled** in Web.config file.

In the above code, ej.web.all.min.js script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the

Copyright © 2001 - 2020 Syncfusion Inc.

widgets. Instead, you can use CSG utility to generate a custom script file with the required widgets for deployment purpose.

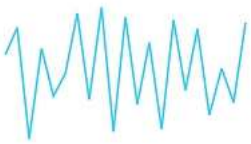
Initialize Sparkline

Add the following code in the index.cshtml file to create the Sparkline control in View page.

ASPX-CS

```
<div>
@(Html.EJ().Sparkline("sparklineContainer"))
</div>
```

Now, the Sparkline is rendered with some auto-generated random values and with default Line type Sparkline.



Working with Data

Local Data

1. You can bind the data to the Sparkline by using DataSourceproperty and then you need to map the **X** and **Y** value with XName and YName properties respectively.

C#

```
public class HomeController : Controller
{
public IActionResult Index()
{
//// Create dataSource to Sparkline
List<SparklineData> data = new List<SparklineData>();
data.Add(new SparklineData(0, 35));
data.Add(new SparklineData(1, 28));
data.Add(new SparklineData(2, 34));
data.Add(new SparklineData(3, 32));
data.Add(new SparklineData(4, 40));
data.Add(new SparklineData(5, 32));
data.Add(new SparklineData(6, 35));
data.Add(new SparklineData(7, 55));
data.Add(new SparklineData(8, 38));
data.Add(new SparklineData(9, 30));
data.Add(new SparklineData(10, 25));
data.Add(new SparklineData(11, 32));
///...
ViewBag.SparklineData = data;
return View();
}
}
public class SparklineData
{
public double Month;
```

```
public double Sales;
public SparklineData(double month, double sales)
{
this.Month = month;
this.Sales = sales;
}
}
```

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.DataSource((IEnumerable<object>)ViewBag.SparklineData)
.XName("Month").YName("Sales")
)
```



;

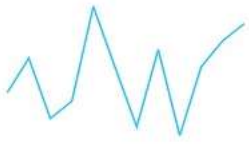
2. You can also bind an array of data to Sparkline by using DataSource property.

ASPX-CS

```
public class HomeController : Controller
{
public IActionResult Index()
{
Double[] y1 = { 2, 6, -1, 1, 12, 5, -2, 7, -3, 5, 8, 10};
ViewBag.SparklineData = y1;
return View();
}
}
```

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.DataSource((IEnumerable<object>)ViewBag.SparklineData)
)
```



;

Sparkline Dimensions

You can set the size directly on the Sparkline or to the container of the sparkline. When you do not specify the size, it takes 30px as the height and 50px as its width, by default.

Copyright © 2001 - 2020 Syncfusion Inc.

Set size for the container

You can customize the Sparkline dimension by setting the width and height for the container element.

HTML

```
<body>
<div id="container" style="width:820px;height:500px;"></div>
@(Html.EJ().Sparkline("container"))
</body>
```



Set size in pixels

You can also set the Height and Width for Sparkline by using the Size property of the Sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.Size(size=>size.Height("40").Width("60"))
)
```

Responsive Sparkline

To resize the Sparkline when the browser or the sparkline container is resized, set the IsResponsive property to **true**, where the sparkline adapts to the changes in size of the container.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.IsResponsive(true)
)
```

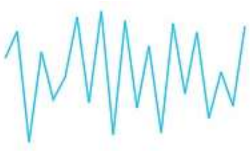
Sparkline Types

Line Type

To render a Line type Sparkline, set the Type as **line**. To change the color and width of the line, you can use the Fill and Width property.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the  line type sparkline
.Width(3)
.Fill("#33ccff")
)
```



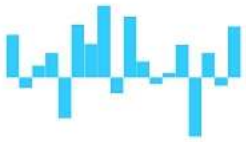
Copyright © 2001 - 2020 Syncfusion Inc.

Column Type

To render a Column Sparkline, set the type as **column** To change the color of the column, you can use the Fill property.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the column type sparkline
.Type(SparklineType.Column)
.Fill("#33ccff")
)
```

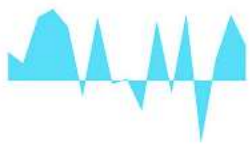


Area Type

To render an Area Sparkline, you can specify the type as **area**. To change the Area color, you can use the Fill property

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the area type sparkline
.Type(SparklineType.Area)
.Fill("#69D2E7")
)
```

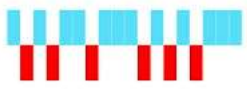


WinLoss Type

WinLoss Sparkline render as a column segment and it show the positive, negative and neutral values. You can customize the positive and negative color of the win-loss type.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the area type sparkline
.Type(SparklineType.WinLoss)
.Fill("#69D2E7")
)
```



Copyright © 2001 - 2020 Syncfusion Inc.

Pie Type

You can create a pie type sparkline by setting the type as **pie**. Colors for the pie can be customize using Palette property.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the area type sparkline
.Type(SparklineType.Pie)
.Palette(pl => { pl.Add("#ff3399"); pl.Add("#33ccff");  })
)
```

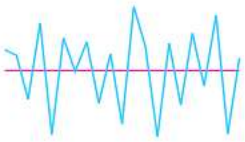


Axis Customize

The Sparkline axis can be collapsed using visible property in AxisLineSettings and this not applicable for win-loss. You can customize Color, Width and DashArray of axis line.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the axis line of the sparkline
.AxisLineSettings(as => as.Visible(true).Color("#ff14ae"))
)
```

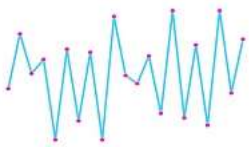


Marker Customization

You can customize markers by initializing the MarkerSettings property. The customization options allow you to change the Visibility, Fill, Width, Opacityand Border options Color, Width, Opacity for marker. This customization only applicable for line, column and area type Sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the marker of the sparkline
.MarkerSettings(mr =>
mr.Visible(true).Fill("#ff14ae").Width(4).Border(br=> br.Width(1)))
)
```



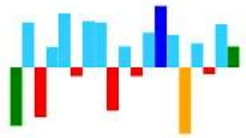
Copyright © 2001 - 2020 Syncfusion Inc.

Point Customization

You can customize points by initializing the point colors. The customization options allow you to differentiate the First, Last, Highest, Lowest, and Negative points. This customization only applicable for line, column and area type Sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the point color of the sparkline
.NegativePointColor("Red")
.HighPointColor("Blue")
.LowPointColor("Orange")
.StartPointColor("Green")
.EndPointColor("Green")
)
```

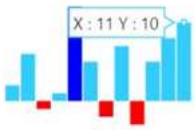


Tooltip

The Tooltip follows the pointer movement and is used to indicate the value of a point. This feature is applicable for line, column, pie, and area Sparkline. You can enable the tooltip by setting it's Visible property as true. You can customize the tooltip Fill, Border and Font.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.Tooltip(tooltip => tooltip.Visible(true))
)
```



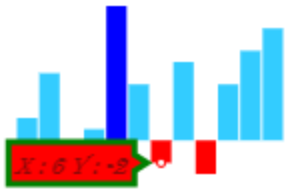
Tooltip Customization

You can customize the tooltip Fill, Border Color, Width and Font properties Color, FontFamily, FontStyle, FontWeight, Opacity and Size

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
.Tooltip(tooltip => tooltip
.Fill('red')
.Border(border=>border.Color('green').Width(3))
.Font(font=>font.Size('12px').FontFamily('Algerian').FontStyle('Italic').
FontWeight('Lighter').Opacity(0.5))
)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip Template

HTML elements can be displayed in the tooltip by using the Template option of the tooltip. The template option takes the value of the id attribute of the HTML element. You can use the **#point.x#** and **#point.y#** as place holders in the HTML element to display the x and y values of the corresponding point.

ASPX-CS

```
<div id="item" style="display: none;">
<div>
<div>#point.x#</div>
<div>#point.y#</div>
</div>
</div>
@(Html.EJ().Sparkline("container")
.Tooltip(tooltip => tooltip.Visible(true).Template("item"))
)
```

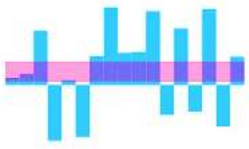


Range Band

The range band feature is used to highlight a particular range along the y-axis using StartRange and EndRange property. You can also customize the Color and Opacity of the RangeBandSettings.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//To customize the range Band of the sparkline
.RangeBandSettings(range =>
range.StartRange(4).EndRange(30).Color("#ff14ae").Opacity(0.4))
)
```



Sparkline Customization

This section explains you the customization options available to make changes in the Sparkline for getting better appearance.

Copyright © 2001 - 2020 Syncfusion Inc.

Sparkline background

You can specify the background color for the Sparkline using Background property. When you don't specify the Background, it takes "transparent" as background color.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
//Specifies background color for sparkline
.Background("gray")
// ...
)
```



Stroke color and width

You can customize the series border color and width using Stroke and Width. This is applicable for Sparkline types line and area.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
//Specifies border color and width for line and area series
.Stroke("green")
.Width(3)
// ...
)
```



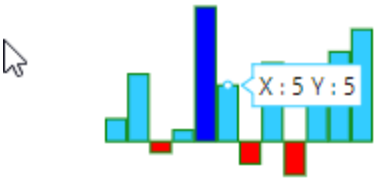
Sparkline border

You can customize the Border color and width of the Sparkline using Color and Height properties. This is applicable for column, win-loss and pie series.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
//Specifies border width and color for column, winLoss and pie series
.Border(border=>border.Color("green").Width(2))
// ...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Opacity

By default Opacity of the Sparkline is 1. You can specify the opacity value from 0 to 1. This is applicable for all types of series.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
//Specifies the opacity of the sparkline
.Opacity(0.5)
// ...
)
```

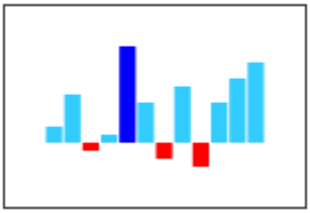


Padding for Sparkline

Padding is used to specify the padding value between the container and Sparkline. By default padding value of the Sparkline is 5.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
//Padding for the sparkline
.Padding(20)
// ...
)
```



Canvas support

You can control whether Sparkline has to be rendered as SVG or Canvas. EnableCanvasRendering property supports all the functionalities supported in SVG rendering.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
//enables canvas rendering
.EnableCanvasRendering(true)
// ...
)
```

Themes

You can specify different Theme for Sparkline control.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
// ...
//theme for sparkline
.Theme(SparkTheme.FlatDark)
// ...
)
```



Methods and Events

Methods

redraw()

This methods redraws the entire sparkline. You can call redraw() whenever you update, add or remove points from the data source or whenever you want to refresh the UI.

ASPX-CS

```
@(Html.EJ().Sparkline("container"))
```

JS

```
// redraw method for sparkline
$("#container").ejSparkline("redraw");
```

Events

Load

The Load event is fired before loading the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.Load("onLoad");
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
<script type="text/javascript">
function onLoad(sender) {
//Do something
}
</script>
```

Loaded

The Loaded event is fired after loaded the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.Loaded("onLoaded");
)
```

JS

```
<script type="text/javascript">
function onLoaded(sender) {
//Do something
}
</script>
```

TooltipInitialize

Fires before rendering trackball tooltip. You can use TooltipInitialize event to customize the text displayed in trackball tooltip.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.TooltipInitialize("onTooltipInitialize")
)
```

JS

```
<script type="text/javascript">
function onTooltipInitialize(sender) {
//Do something
}
</script>
```

SeriesRendering

Fires before rendering a series. The SeriesRendering event is fired for each series in Sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.SeriesRendering("onSeriesRender")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
<script type="text/javascript">
function onSeriesRender(sender) {
//Do something
}
</script>
```

PointRegionMouseMove

The PointRegionMouseMove event is fired when mouse is moved over a point.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.PointRegionMouseMove("onPointMouseMove")
)
```

JS

```
<script type="text/javascript">
function onPointMouseMove(sender) {
//Do something
}
</script>
```

PointRegionMouseClick

The PointRegionMouseClick event is fired on clicking a point in sparkline. You can use this event to handle clicks made on points.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.PointRegionMouseClick("onPointMouseClick")
)
```

JS

```
<script type="text/javascript">
function onPointMouseClick(sender) {
//Do something
}
</script>
```

SparklineMouseMove

The SparklineMouseMove is fired on moving mouse over the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.SparklineMouseMove("onSparklineMouseMove")
)
```

JS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function onSparklineMouseMove(sender) {
//Do something
}
</script>
```

SparklineMouseLeave

The SparklineMouseLeave event is fired on moving mouse outside the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.SparklineMouseLeave("onSparklineMouseLeave")
)
```

JS

```
<script type="text/javascript">
function onSparklineMouseLeave(sender) {
//Do something
}
</script>
```

Click

The Click event is fired on clicking the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.Click("onClick")
)
```

JS

```
<script type="text/javascript">
function onClick(sender) {
//Do something
}
</script>
```

DoubleClick

The DoubleClick event is fired on double clicking the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.DoubleClick("onDoubleClick")
)
```

JS

```
<script type="text/javascript">
function onDoubleClick(sender) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Do something
}
</script>
```

RightClick

The RightClick event is fired on right clicking the sparkline.

ASPX-CS

```
@(Html.EJ().Sparkline("container")
.RightClick("onRightClick")
)
```

JS

```
<script type="text/javascript">
function onRightClick(sender) {
//Do something
}
</script>
```

SunburstChart

Overview

Sunburst chart is useful for visualizing hierarchical data. The center circle represents the root level in the hierarchy, with outer circles representing the higher levels of the hierarchy.

Some of the key features are,

- Visualize hierarchical data.
- Data label support for better readability.
- Interactively select or highlight segments.
- Interactive legend.
- Colors of the segments can be customized in the Sunburst Chart.

Getting Started

This section explains you the steps required to populate the SunburstChart , add data labels, tooltips and title to the SunburstChart. This section covers only the minimal features that you need to know to get started with the SunburstChart.

Installation and Deployment

It describes you on how to configure the Syncfusion dependencies in your ASP.NET MVC application to create a SunburstChart.

Referencing Syncfusion Assemblies

The following assemblies need to be referenced in your application for using Essential SunburstChart MVC

1. Syncfusion.EJ.dll
2. Syncfusion.EJ.MVC.dll

Copyright © 2001 - 2020 Syncfusion Inc.

Follow the steps given below to deploy the application in the development server by referencing the assembly in **GAC**.

- Web.config file should be configured according to the referenced assemblies.
- To deploy your application, you have to ensure that the above referenced assemblies (in your web.config files) are present in the GAC.

```
<system.web>
  <compilation debug="true" targetFramework="4.5.1">
    <assemblies>
      <add assembly="Syncfusion.EJ, Version=13.3400.0.7, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
      <add assembly="Syncfusion.EJ.Mvc, Version=13.3400.0.7, Culture=neutral, PublicKeyToken=3d67ed1f87d44c89"/>
    </assemblies>
  </compilation>
```

And, add Syncfusion namespace in the Web.config file of Views folder to generate SunburstChart builder code in View page.

```
<system.web.webPages.razor>
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="Syncfusion.JavaScript, Version=13.3400.0.7, Culture=neutral,
                                                PublicKeyToken=3d67ed1f87d44c89" />
      <add namespace="Syncfusion.JavaScript.DataVisualization, Version=13.3400.0.7,
                                                Culture=neutral, PublicKeyToken=3d67ed1f87d44c89" />
      <add namespace="Syncfusion.MVC.EJ, Version=13.3400.0.7, Culture=neutral,
                                                PublicKeyToken=3d67ed1f87d44c89" />

    </namespaces>
  </pages>
</system.web.webPages.razor>
```

Note: If you are referring Syncfusion dll's manually from bin folder to create MVC SunburstChart, then remove Culture, Version and PublicKeyToken attributes used in all, and nodes.

Adding script reference

Add the below scripts references and Syncfusion **ScriptManager** to execute the script as mentioned in the following code snippet in layout page.

ASPX-CS

```
<!--  jquery script  -->
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<!-- Essentialcshtml UI widget -->
<script
src="http://cdn.syncfusion.com/{{site.releaseversion}}/js/web/ej.web.all.
min.js"></script>
<!--Add Syncfusion Script Manager -->
@Html.EJ().ScriptManager()
```

To render the MVC Chart in unobtrusive mode refer **ej.unobtrusive.js** script file. Otherwise set false to **UnobtrusiveJavaScriptEnabled** in Web.config file.

In the above code, ej.web.all.min.js script reference has been added for demonstration purpose. It is not recommended to use this for deployment purpose, as its file size is larger since it contains all the widgets. Instead, you can use CSG utility to generate a custom script file with the required widgets for deployment purpose.

Copyright © 2001 - 2020 Syncfusion Inc.

Initialize Sunburst Chart

Add the following code in the index.cshtml file to create the SunburstChart control in View page.

ASPX-CS

```
<div>
@(Html.EJ().SunburstChart("chartContainer"))
</div>
```

Populate Data source:

Now, let's see how to plot data source to the SunburstChart. First, let us generate a data source containing following fields in controller.

C#

```
public class DefaultData
{
public string Category { get; set; }
public string Country { get; set; }
public string JobDescription { get; set; }
public string JobGroup { get; set; }
public string JobRole { get; set; }
public int EmployeesCount { get; set; }
public static List<DefaultData> GetDefaultData()
{
List<DefaultData> data = new List<DefaultData>();
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Sales",          JobGroup ="Executive",
EmployeesCount= 50});
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Sales",          JobGroup = "Analyst",
EmployeesCount =40 });
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Marketing",
EmployeesCount =40 });
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Technical",      JobGroup = "Testers",
EmployeesCount =55 });
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Technical",      JobGroup = "Developers", JobRole =
"Windows", EmployeesCount =175});
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Technical",      JobGroup = "Developers", JobRole =
"Web",      EmployeesCount =70 });
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Management",
EmployeesCount =40 });
data.Add(new DefaultData() { Category = "Employees", Country = "USA",
JobDescription = "Accounts",
EmployeesCount =60 });
data.Add(new DefaultData() { Category = "Employees", Country = "India",
JobDescription = "Technical",      JobGroup = "Testers",
EmployeesCount = 43 });
data.Add(new DefaultData() { Category = "Employees", Country = "India",
JobDescription = "Technical",      JobGroup = "Developers", JobRole =
"Windows", EmployeesCount = 125});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
data.Add(new DefaultData() { Category = "Employees", Country = "India",
JobDescription = "Technical",     JobGroup = "Developers", JobRole =
"Web",     EmployeesCount = 60 });
data.Add(new DefaultData() { Category = "Employees", Country = "India",
JobDescription = "HR Executives",
EmployeesCount = 70 });
data.Add(new DefaultData() { Category = "Employees", Country = "India",
JobDescription = "Accounts",
EmployeesCount = 45 });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Sales",         JobGroup = "Executive",
EmployeesCount = 30 });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Sales",         JobGroup = "Analyst",
EmployeesCount = 40 });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Marketing",
EmployeesCount = 50  });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Technical",     JobGroup = "Testers",
EmployeesCount = 40 });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Technical",     JobGroup = "Developers", JobRole =
"Windows", EmployeesCount = 65 });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Technical",     JobGroup = "Developers", JobRole =
"Web",     EmployeesCount = 27 });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Management",
EmployeesCount = 33  });
data.Add(new DefaultData() { Category = "Employees", Country = "Germany",
JobDescription = "Accounts",
EmployeesCount = 55  });
data.Add(new DefaultData() { Category = "Employees", Country = "UK",
JobDescription = "Technical",     JobGroup = "Testers",
EmployeesCount = 45 });
data.Add(new DefaultData() { Category = "Employees", Country = "UK",
JobDescription = "Technical",     JobGroup = "Developers", JobRole =
"Windows", EmployeesCount = 96 });
data.Add(new DefaultData() { Category = "Employees", Country = "UK",
JobDescription = "Technical",     JobGroup = "Developers", JobRole =
"Web",     EmployeesCount = 55 });
data.Add(new DefaultData() { Category = "Employees", Country = "UK",
JobDescription = "HR Executives",
EmployeesCount = 60 });
data.Add(new DefaultData() { Category = "Employees", Country = "UK",
JobDescription= "Accounts",
EmployeesCount=30   });
data.Add(new DefaultData() { Category = "Employees", Country = "France",
JobDescription= "Technical",     JobGroup = "Testers",
EmployeesCount = 40 });
data.Add(new DefaultData() { Category = "Employees", Country = "France",
JobDescription= "Technical",     JobGroup = "Developers", JobRole =
"Windows", EmployeesCount = 65 });
data.Add(new DefaultData() { Category = "Employees", Country = "France",
JobDescription= "Technical",     JobGroup = "Developers", JobRole =
"Web",     EmployeesCount = 27 });
```

```
data.Add(new DefaultData() { Category = "Employees", Country = "France",
JobDescription="Marketing",
EmployeesCount= 50 });
return data;
}
}
```

Initialize Sunburst Chart with data

Now, bind the DefaultData to DataSource property of the Sunburst Chart. TheLevelsproperty determines the number of hierarchical levels. Each hierarchy level is formed based on the property specified in GroupMemberPath property, and each arc segment size is calculated using ValueMemberPath.

2. Add SunburstChart() action in SunburstChartController and add the following code.

C#

```
public ActionResult SunburstChart()
{
ViewData["defaultData"] = DefaultData.GetDefaultData();
return View();
}
```

3.Render the SunburstChart with the Datasource and Levels,the following code snippet is used

ASPX-CS

```
@{
var datasource = ViewData["defaultData"];
}
@(Html.EJ().SunburstChart("chartContainer")
.ValueMemberPath("EmployeesCount")
.DataSource(datasource)
.Levels(lv =>
{
lv.GroupMemberPath("Country").Add();
lv.GroupMemberPath("JobDescription").Add();
lv.GroupMemberPath("JobGroup").Add();
lv.GroupMemberPath("JobRole").Add();
}))
```

3. The final CSHTML file appears as follows

ASPX-CS

```
@using Syncfusion.JavaScript.DataVisualization
@{
var datasource = ViewData["datasource"];
}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!--  jquery script  -->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script src="http://cdn.syncfusion.com/js/assets/external/jquery-
1.10.2.min.js"></script>
<!-- Essentialcshtml UI widget -->
<script src="http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/ej.web.all.min.js"></script>
<div>
@(Html.EJ().SunburstChart("chartContainer")
.ValueMemberPath("EmployeesCount")
.DataSource(datasource)
.Levels(lv =>
{
lv.GroupMemberPath("Country").Add();
lv.GroupMemberPath("JobDescription").Add();
lv.GroupMemberPath("JobGroup").Add();
lv.GroupMemberPath("JobRole").Add();
}))
</div>
@(Html.EJ().ScriptManager())
</body>
</html>
```

Add Title to the Sunburst Chart

The title of the Sunburst chart is used to provide quick information to the user about the data being plotted in the Sunburst Chart. You can add it by using the Text property of the **Title**

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Enable Text
.Title(ti => ti.Text("Employees Count"))
)
```

Enable Legend

You can enable or disable the legend by using the Visible property present inside the **Legend**

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Enable Legend
.Legend(li => li.Visible(true))
)
```

Add Data Labels

The data labels are used to improve the readability of the Sunburst chart. This can be achieved by enabling the visible property in the **DataLabelSettings**

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Enable DataLabel
.DataLabelSettings(dt => dt.Visible(true))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Now the Sunburst Chart is rendered along with the specified customizations



Click here to view the default sample of the Sunburst Chart.

Sunburst Elements

The Sunburst region represents the entire chart and all its elements. It includes all the chart elements like Legend, DataLabel, Levels etc. The major properties of the Sunburst Chart are as follows

datasource – Provides the data that are used to generate the chart.

valueMemberPath- Property based on the which the data segments are rendered in the chart

legend – displays the legend of the Sunburst Chart

levels- displays the hierarchical levels for the chart

data label– displays the data labels for the Sunburst Chart

Start and End Angle

Start and End Angle

You can change the start and end angle of Sunburst chart using **StartAngle** and **EndAngle** property as shown in below code

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.StartAngle(-90)
.EndAngle(90)
)
```

Sunburst Radius

The Radius of the Sunburst chart can be customized by using the **Radius** property. The default value of radius is 1 and its value ranges between 0 and 1

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Radius(0.8)
)
```



;

Sunburst Inner Radius

The Inner Radius of the Sunburst chart can be customized by using the **InnerRadius** property. The default value of innerRadius is 0.4 and its value ranges between 0 and 1

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.InnerRadius(0.5)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



;

Levels

Sunburst chart is used to display hierarchical data. You can add more than one hierarchical data by using the **Levels** property of Sunburst chart. Each level of the hierarchy is represented by circle.

The following code snippet illustrates

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Levels(
//.. to add the hierarchical levels
})
)
```

GroupMemberPath

It is the string property that is used to map the group category value in the dataSource .

You can define the levels as shown in the below code example

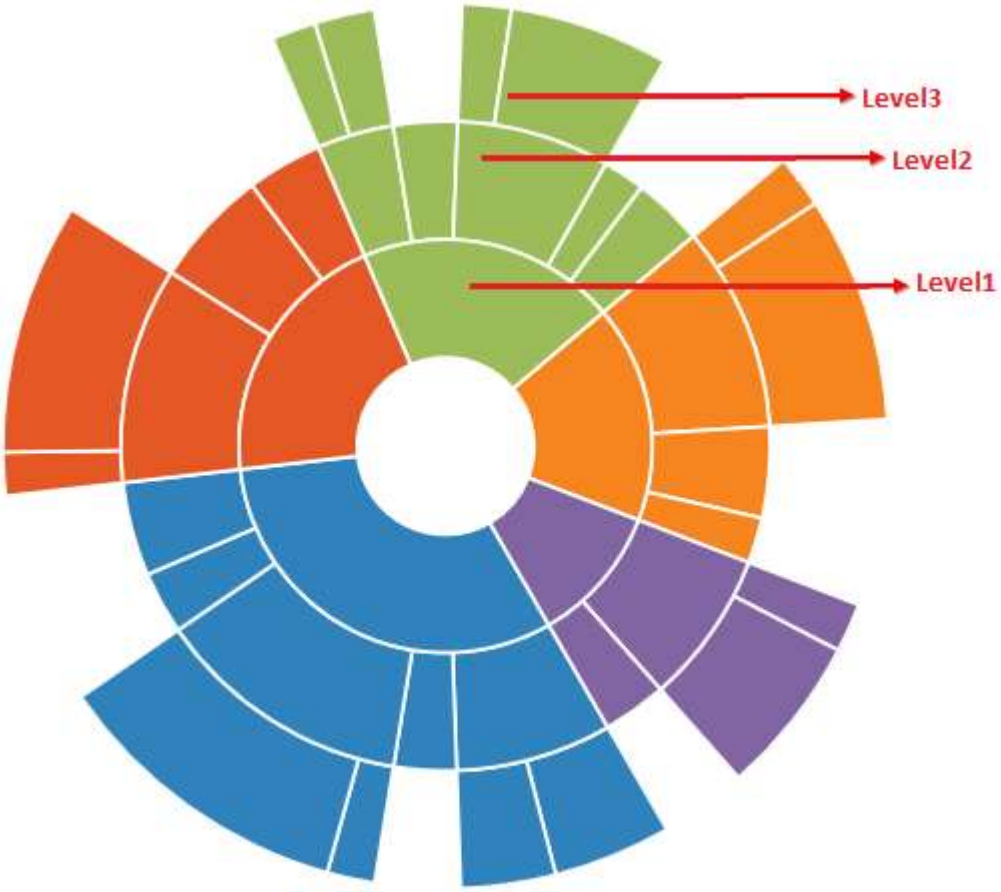
JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Levels(
//.. to add the hierarchical levels
lv.GroupMemberPath("Level1").Add();
lv.GroupMemberPath("Level2").Add();
lv.GroupMemberPath("Level3").Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

The following screenshot illustrates the Sunburst Chart with different levels



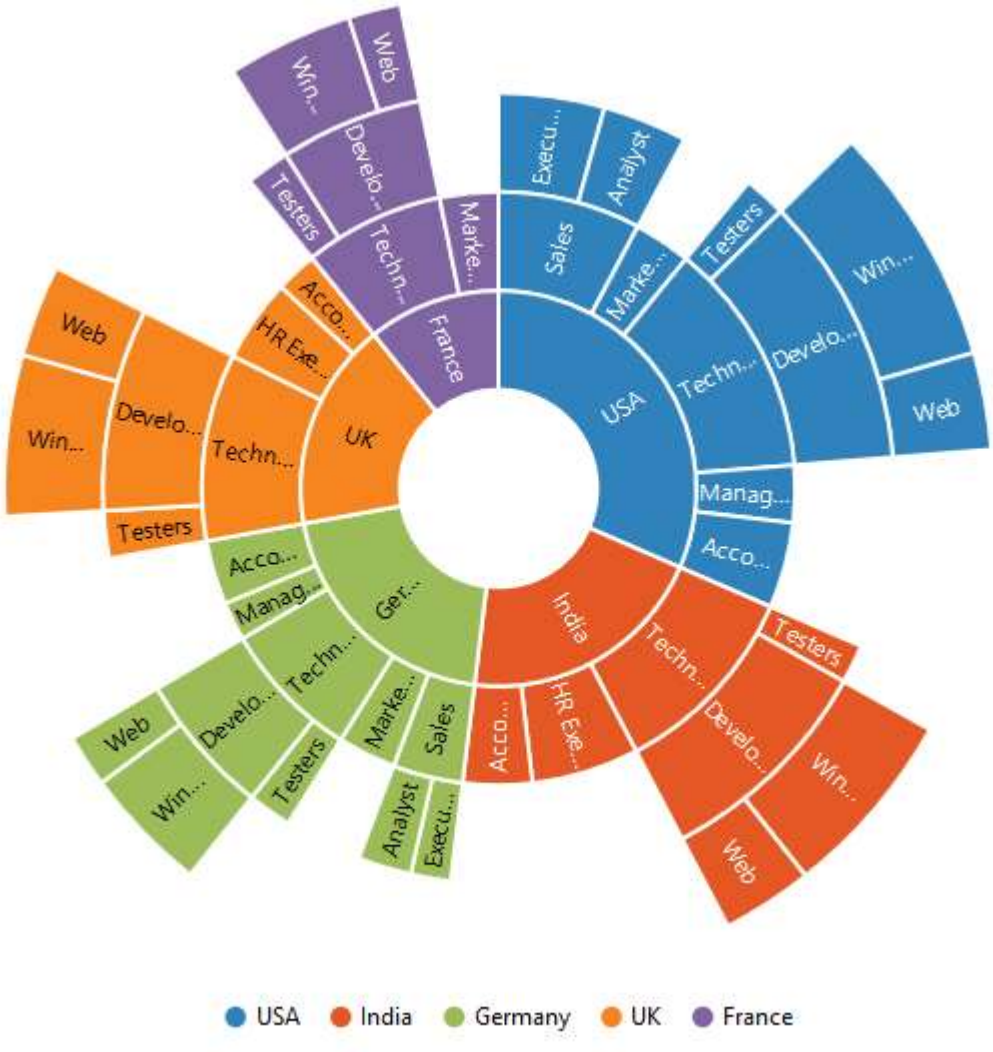
Legend

The legend is used to represent the first level of items in the Sunburst Chart.The **Legend** can be initialized using the below code snippet

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg => lg.Visible(true))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Legend Icon

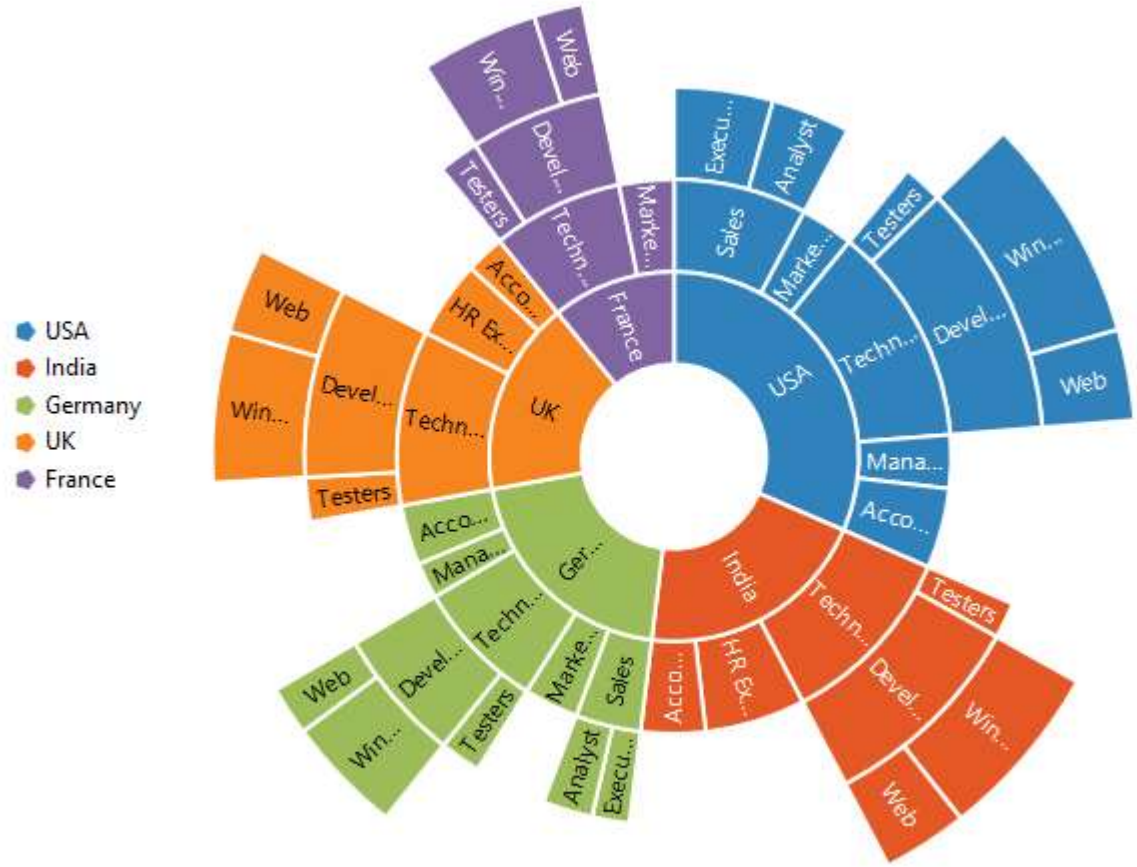
You can specify different shapes of legend icon by using the **Shape** property of the legend. By default, legend shape is **Circle**. The Sunburst chart has some predefined shapes such as:

- Circle
- Cross
- Diamond
- Pentagon
- Rectangle
- Triangle

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg => lg.Visible(true).Shape(SunburstLegendShape.Pentagon))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

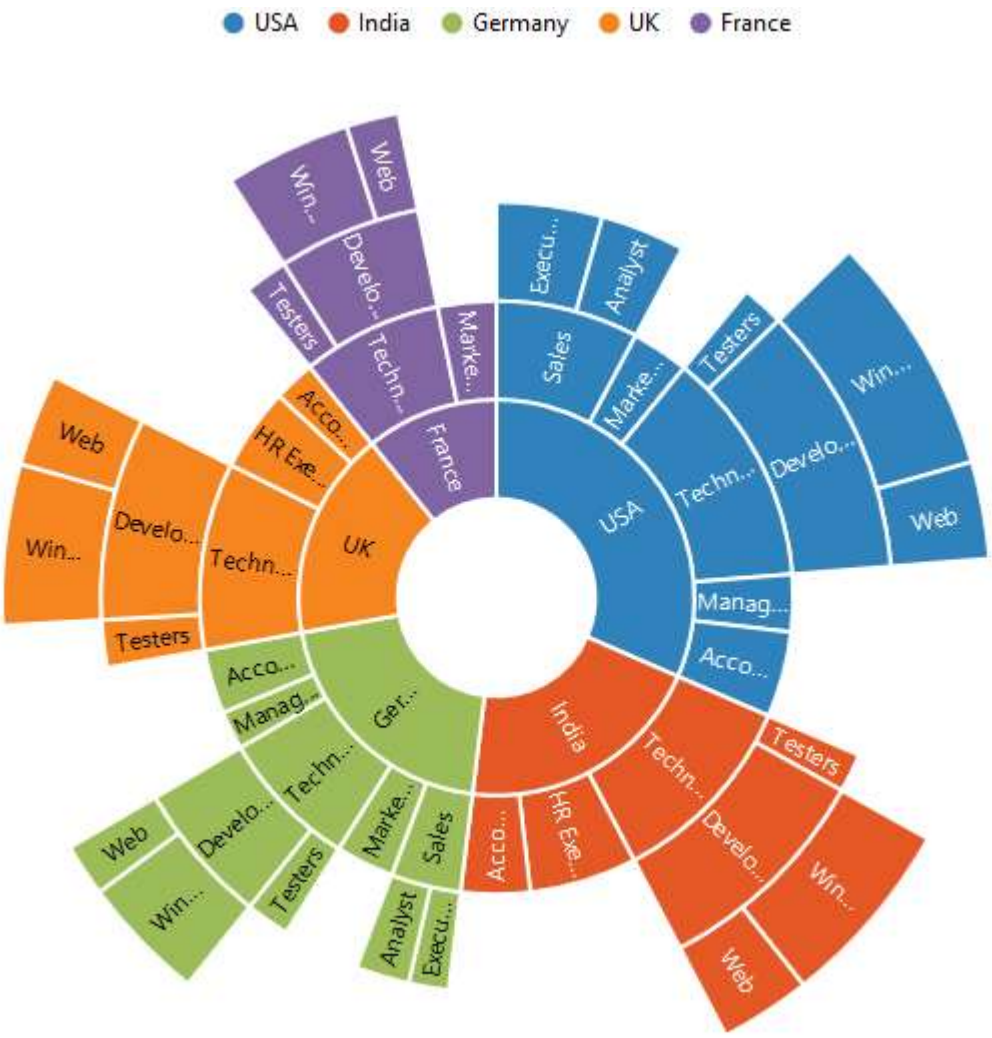


Positioning the Legend

By using the **Position** property, you can position the legend at left, right, top or bottom of the chart.

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg => lg.Visible(true).Position(SunburstLegendPosition.Top))
)
```



Customization

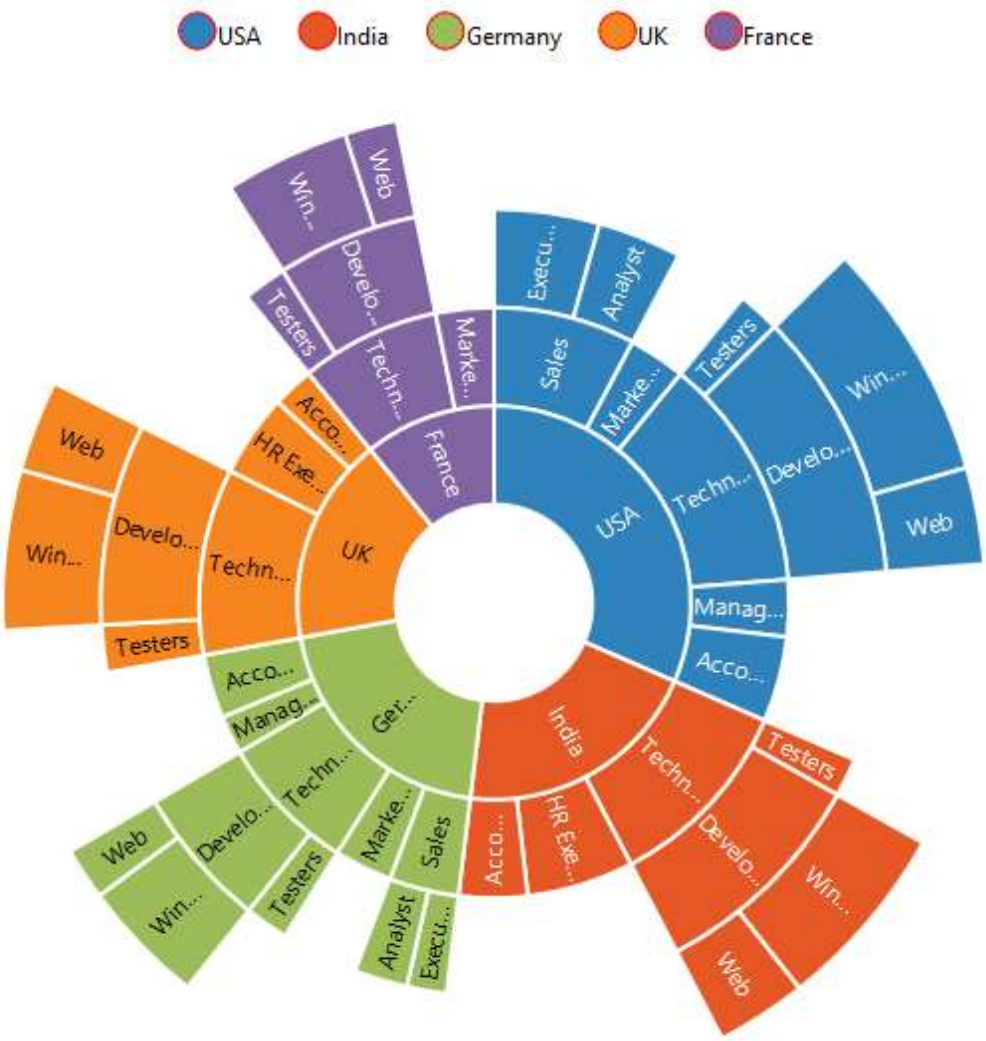
Legend Item Size and border

You can change the size of the legend items by using the **LegendItemStyle-Width** and **LegendItemStyle-Height** properties. To change the legend item border, use **Border** property of the legend .

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg =>
lg.Visible(true).Position(SunburstLegendPosition.Top).LegendItemStyle(lt=
>lt.Width(13).Height(13)).Border(br=>br.Color("#FF0000").Width(1)))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



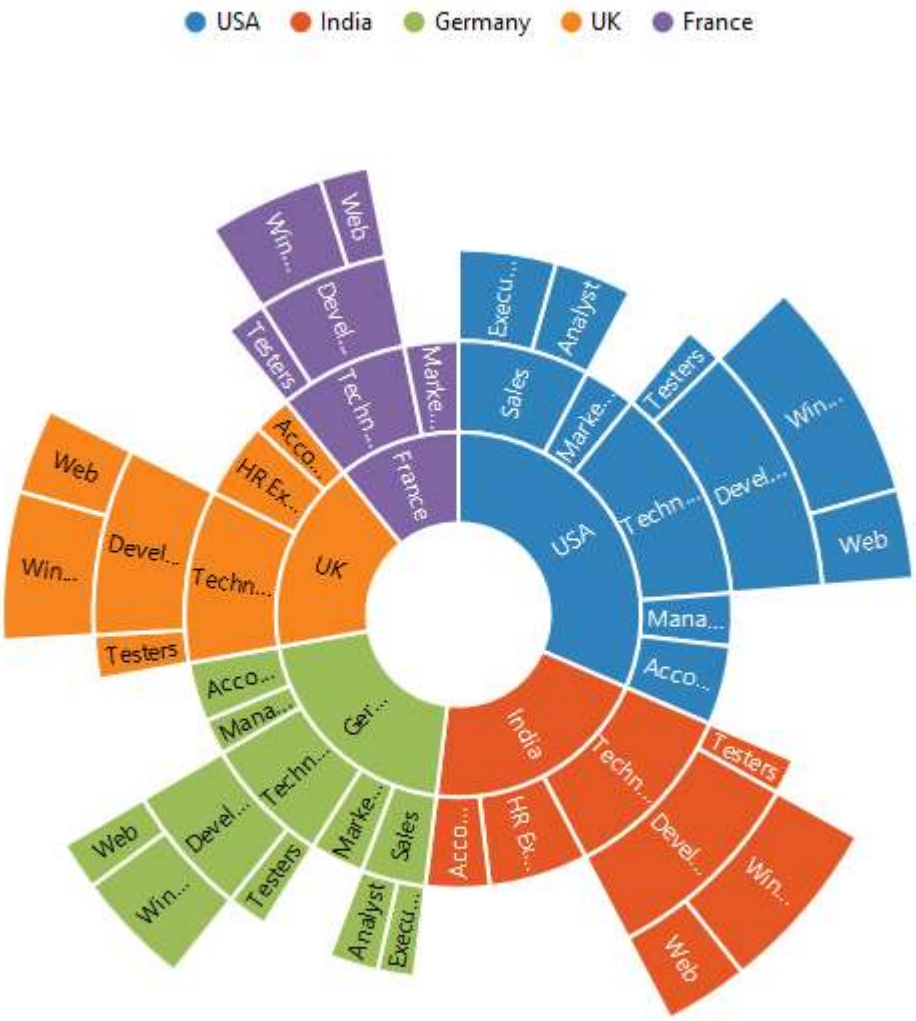
Legend Size

By default, legend takes 20% of the height horizontally when it was placed on the top or bottom position and 20% of the width vertically while placing on the left or right position of the chart. You can change this default legend size by using the **Size** property of the legend.

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg =>
lg.Visible(true).Position(SunburstLegendPosition.Top).Size(size=>size.Wid
th(200).Height(75)))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Legend Row and Columns

You can arrange the legend items horizontally and vertically by using the **RowCount** and **ColumnCount** properties of the legend.

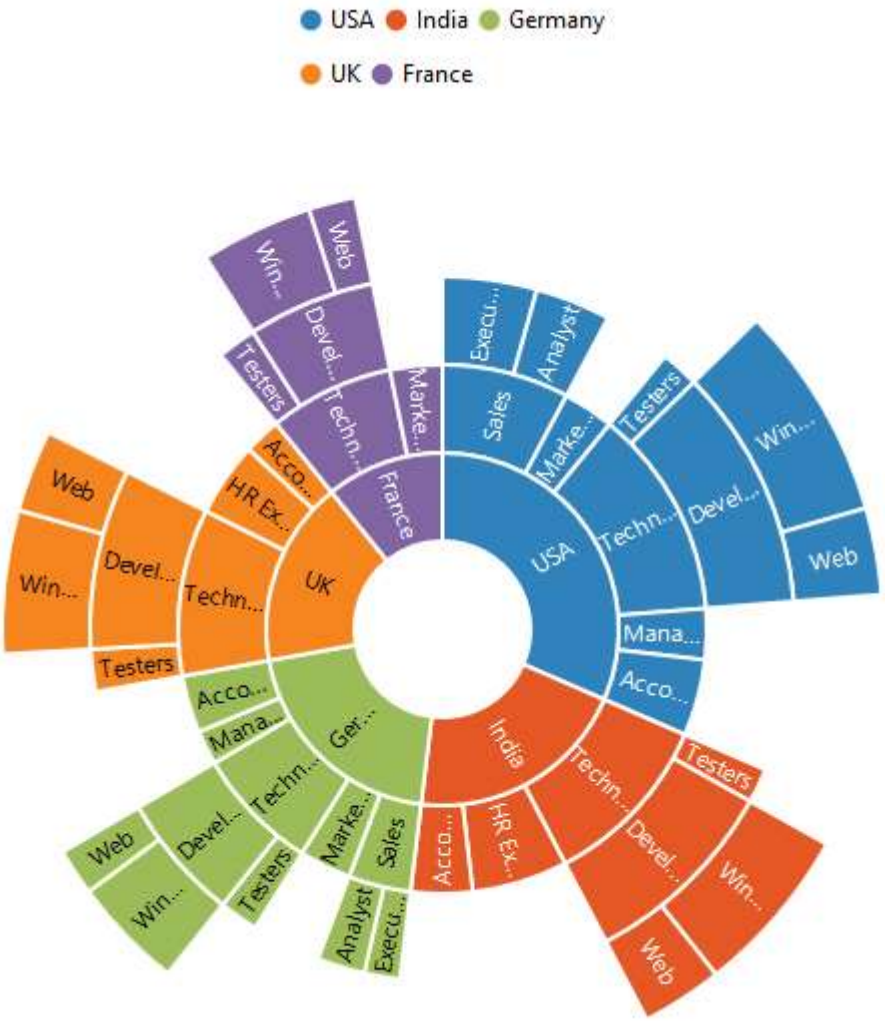
- When only the rowCount is specified, the legend items are arranged according to the rowCount and number of columns may vary based on the number of legend items.
- When only the columnCount is specified, the legend items are arranged according to the columnCount and number of rows may vary based on the number of legend items.
- When both the properties are specified, then the one which has higher value is given preference. For example, when the rowCount is 4 and columnCount is 3, legend items are arranged in 4 rows.
- When both the properties are specified and have the same value, the preference is given to the columnCount when it is positioned at the top/bottom position. The preference is given to the rowCount when it is positioned at the left/right position.

JS

```
@(Html.EJ().SunburstChart("chartContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Legend(lg => lg.Visible(true).RowCount(2).ColumnCount(3))
)
```



LegendInteractivity

You can select a specific category while clicking on corresponding legend item through **ClickAction** property.

It has three types of action

ToggleSegmentSelection ToggleSegmentVisibility * None

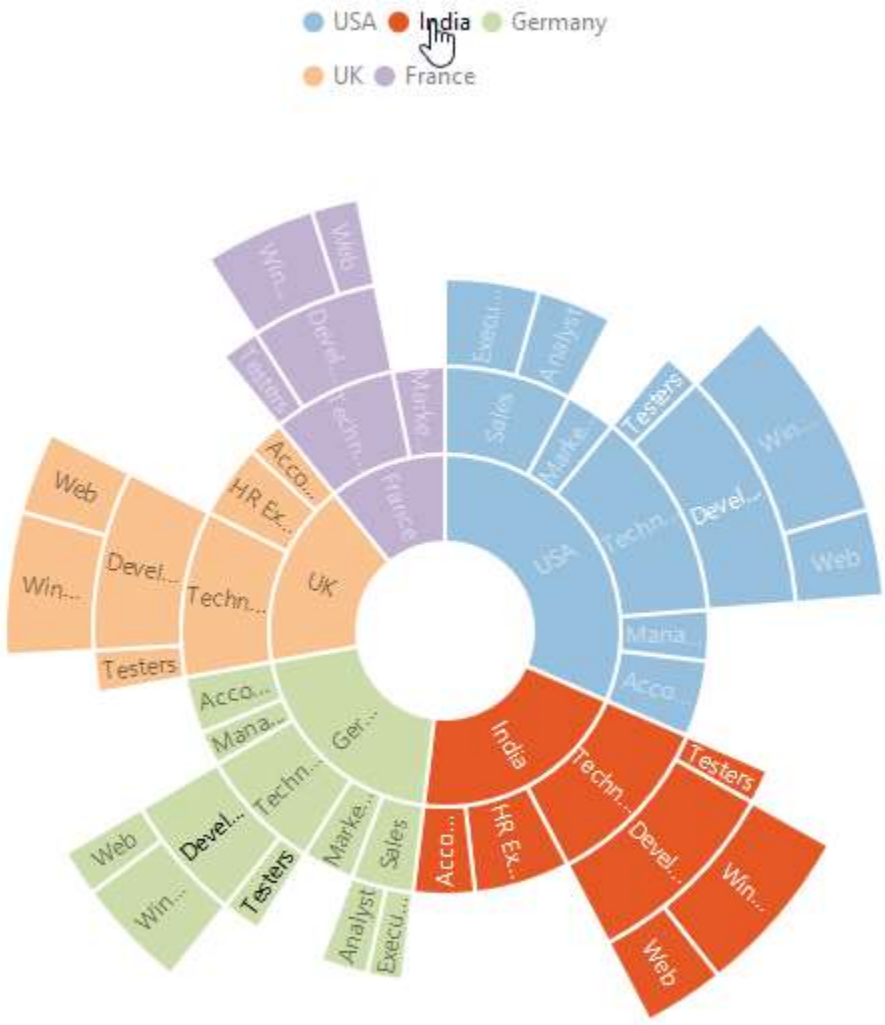
ToggleSegmentSelection

Used to highlight specific category while clicking on legend item

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg =>
lg.Visible(true).ClickAction(SunburstLegendClickAction.ToggleSegmentSelec
tion))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



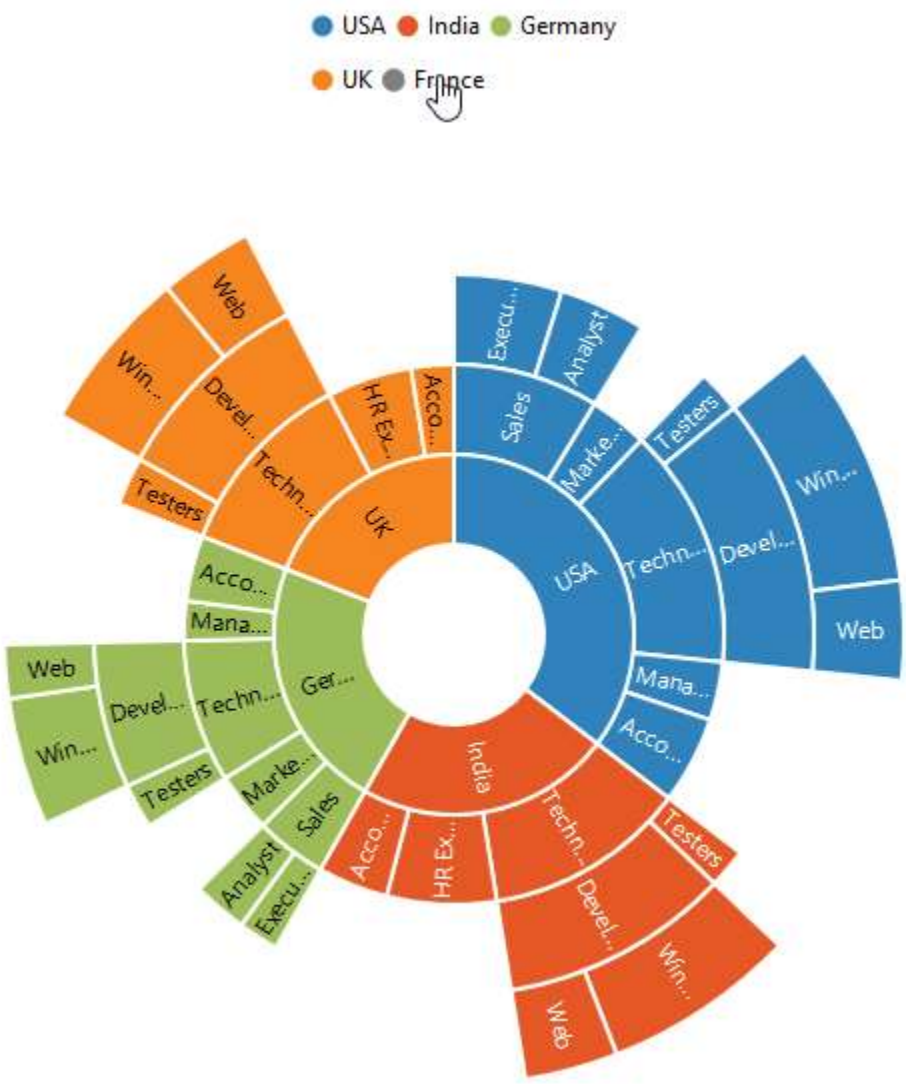
Toggle Segment Visibility

Used to disable the specific category while clicking on legend item.

JS

```
@(Html.EJ().SunburstChart("chartContainer")
.Legend(lg =>
lg.Visible(true).ClickAction(SunburstLegendClickAction.ToggleSegmentVisibility))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Data Labels

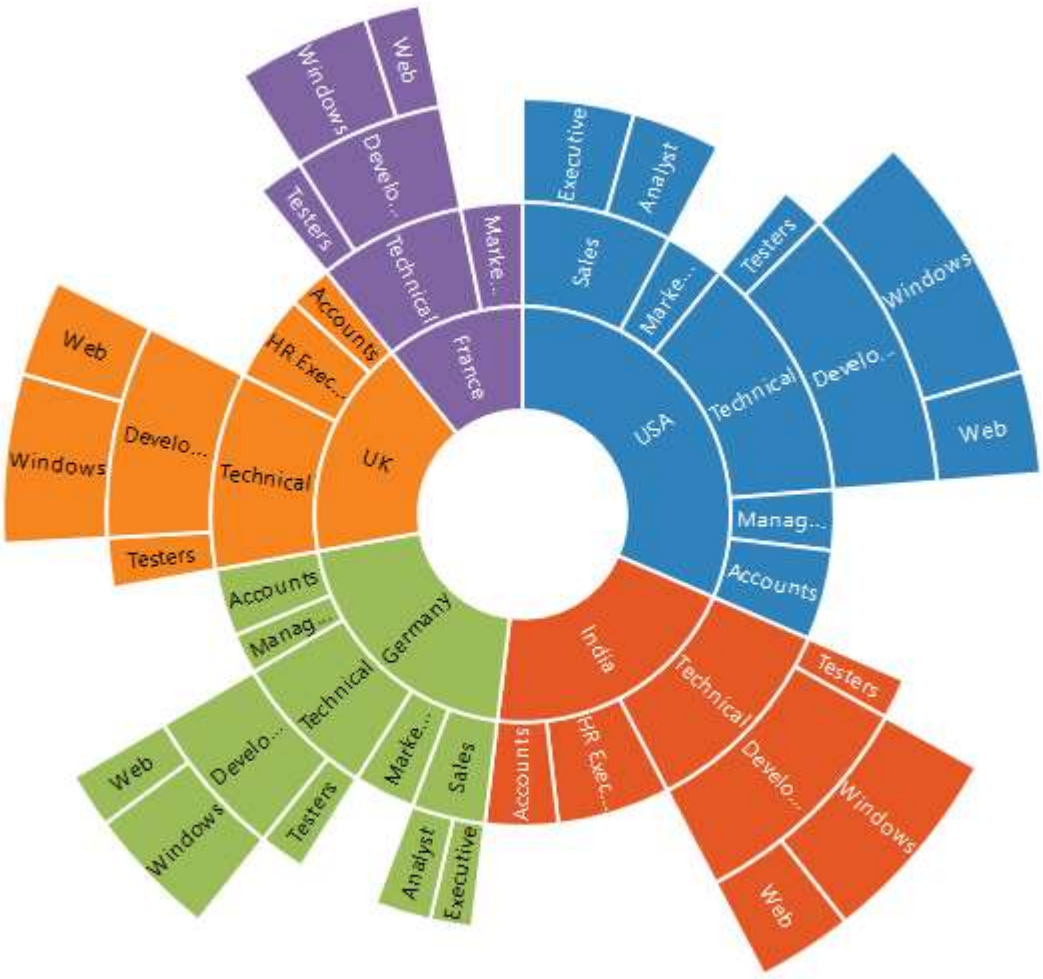
Sunburst data labels are used to display the data related to the segment. It helps to provide the information about the data points to the users.

You can enable or disable the data labels by setting the visible property of the **DataLabelSettings** to true as shown in the below code

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Palette colors
.DataLabelSettings(dt => dt.Visible(true))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Label Overflow mode

When you represent huge data with data labels, they may intersect each other. You can avoid this using the *SunburstLabelOverflowMode* property.

The following properties are used to avoid the overlapping.

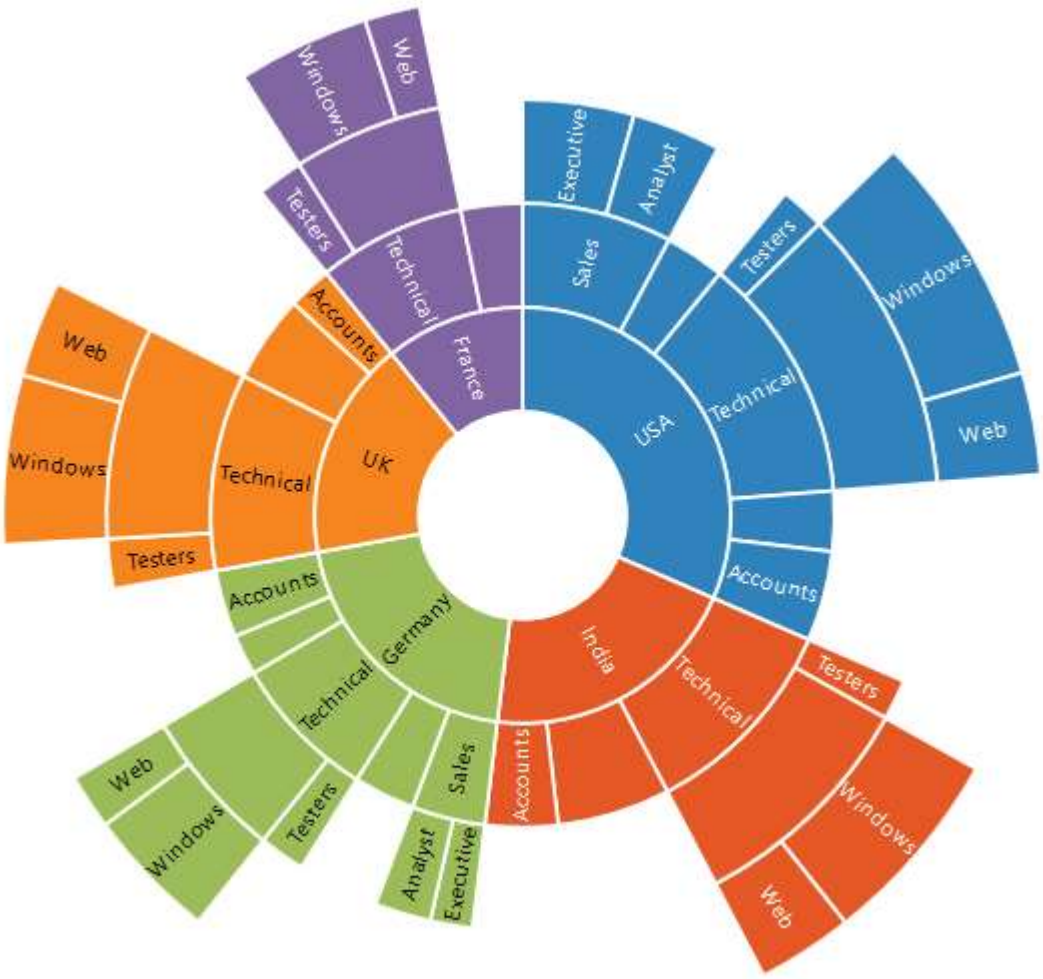
Trim – To trim the large data labels. Hide – To hide the overlapped data labels.

The following code shows how to set Hide and Trim mode.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//.
.DataLabelSettings(dt =>
dt.Visible(true).SunburstLabelOverflowMode(SunburstLabelOverflowMode.Hide
))
//...
)
```

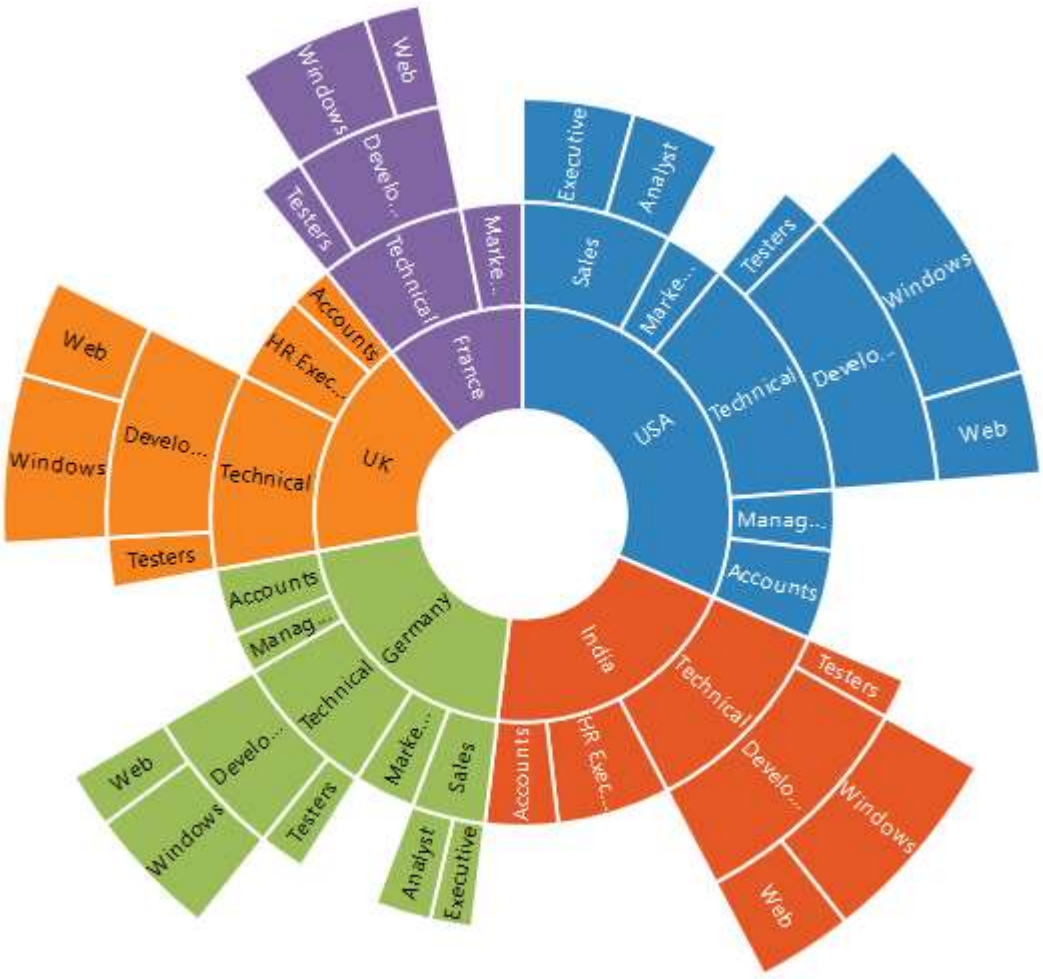
Copyright © 2001 - 2020 Syncfusion Inc.



ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//.
.DataLabelSettings(dt =>
dt.Visible(true).SunburstLabelOverflowMode(SunburstLabelOverflowMode.Trim
))
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Label Rotation Mode

You can rotate the data label by using *SunburstLabelRotationMode* property. By default, the labelRotationMode is set as **angle**.

The following code shows how to set labelRotationMode as normal and angle.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//.
.DataLabelSettings(dt =>
dt.Visible(true).SunburstLabelRotationMode(SunburstLabelRotationMode.Normal)
//...
)
```

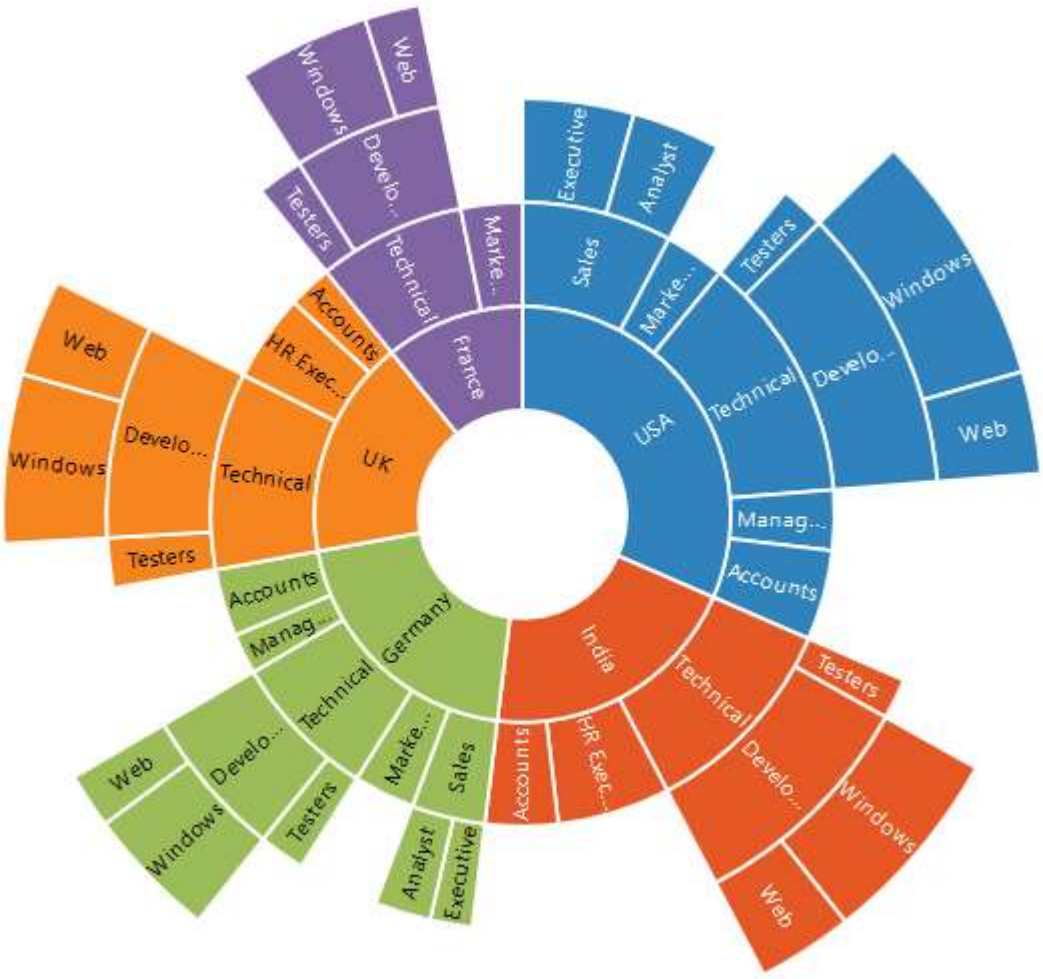
Copyright © 2001 - 2020 Syncfusion Inc.



ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//.
.DataLabelSettings(dt =>
dt.Visible(true).SunburstLabelRotationMode(SunburstLabelRotationMode.Angle)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing the data labels

You can customize the appearance of the data point using the Font property.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//.
.DataLabelSettings(dt =>
dt.Visible(true).Font(font=>font.Color("Black").Font-
Weight("Bold").Size("15px")))
//...
)
```



Tooltip

ToolTip allows you to display any information over a sunburst segment. It appears when mouse hovered over or touch any chart segment. By default, it displays the corresponding segment category name and its value

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.Tooltip(tooltip => tooltip.Visible(true))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip Template

HTML elements can be displayed in the tooltip by using the *Template* property of the tooltip. The template property takes the value of the id attribute of the HTML element. You can use the **#point.x#** and **#point.y#** as place holders in the HTML element to display the x and y values of the corresponding point.

ASPX-CS

```
<div id="Tooltip" style="display: none;">
<div id="value" style="background-color:red;padding-top:3px;padding-
right:3px">
<div>
<label id="efpercentage" style="color:white">
  Category: #point.x#
<br />  Value:#point.y#
</label>
</div>
</div>
</div>
@(Html.EJ().SunburstChart("chartContainer")
.Tooltip(tooltip => tooltip.Visible(true).Template("Tooltip"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



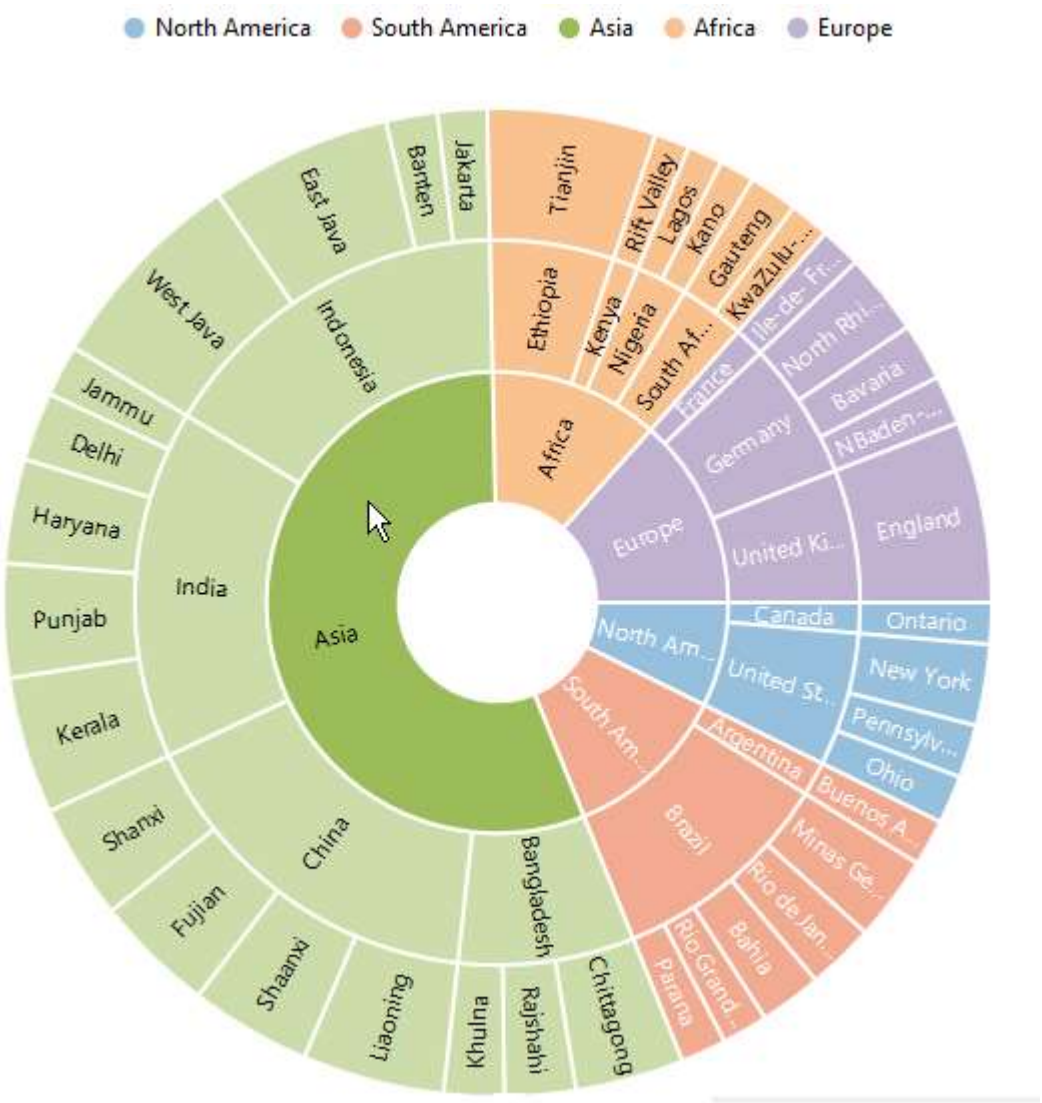
Highlight

SunburstChart provides highlighting support for the points on mouse hover. To enable the highlighting , set the *Enable* property to true in the **HighlightSettings**.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.HighlightSettings(highlight=>highlight.Enable(true))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Highlight Display mode

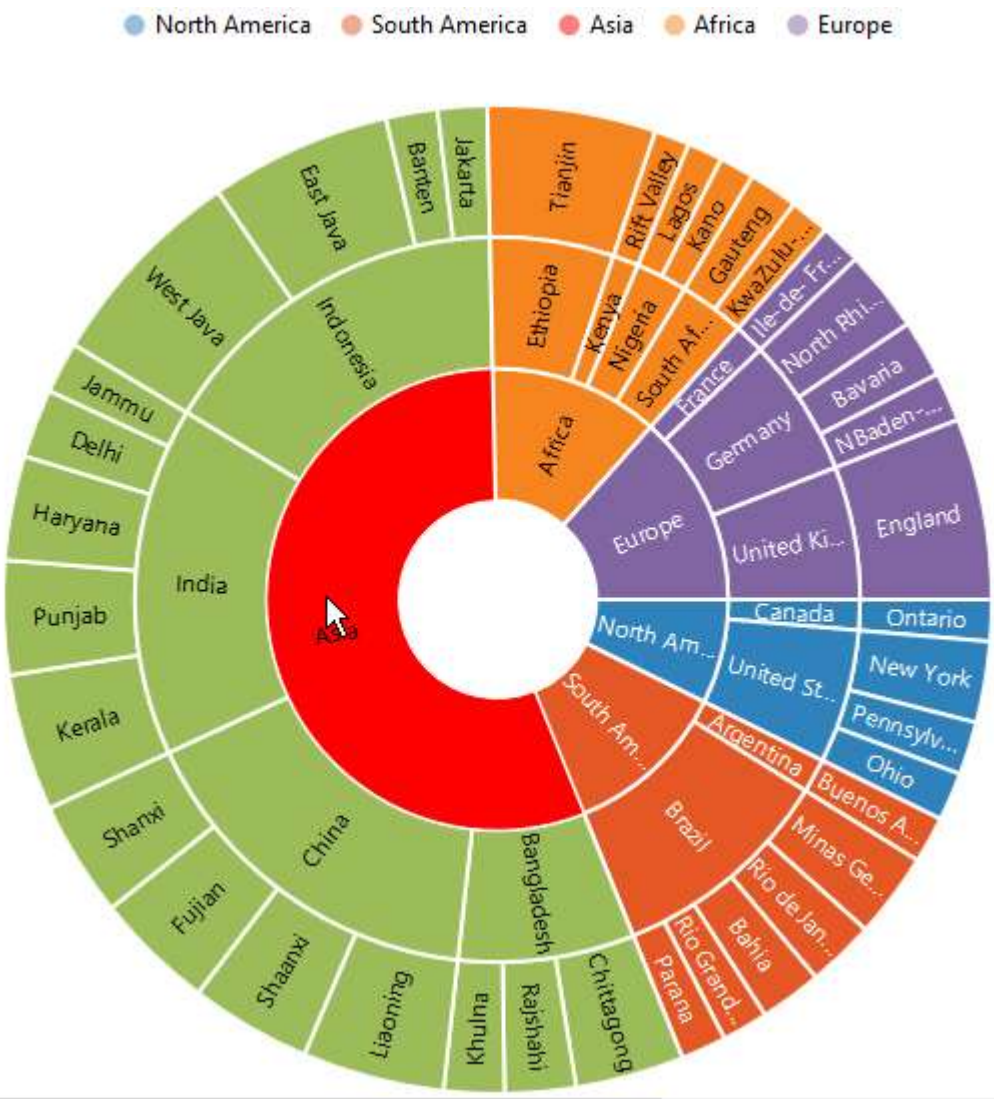
You can customize the highlighted segment appearance by using color or opacity. You can choose between color or opacity using the **Type** property in the highlight Settings.

HighlightByColor – To display the highlighted segment appearance using color. HighlightByOpacity – To display the highlighted segment appearance using opacity.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.HighlightSettings(highlight=>highlight.Enable(true).Type(SunburstHighlig
htType.Color).Color("Red"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Highlight Mode

Sunburst chart provides multiple option to represent the highlighted categories. You can highlight the segment categories by using the **Mode** property in highlightSettings

Child – To highlight the child of selected parent. All – To highlight the entire categories in group. *Parent – To highlight the parent of selected child.* Single - To highlight single item in the category.

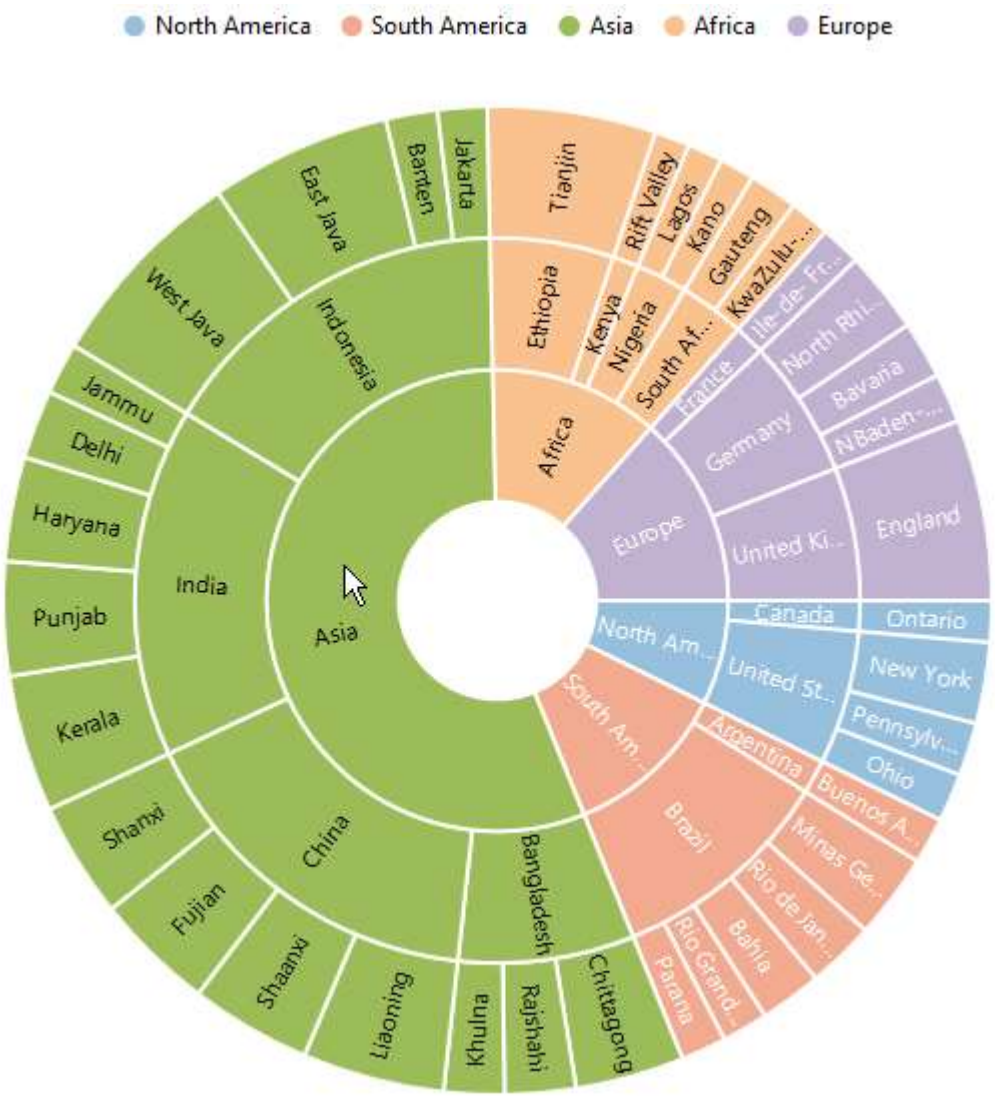
Child

The following code shows how to set the highlight type as child

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.HighlightSettings(highlight=>highlight.Enable(true).Mode(SunburstHighlig
htMode.Child))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Parent

The parent mode can be enabled by using the below code

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.HighlightSettings(highlight=>highlight.Enable(true).Mode(SunburstHighlig
htMode.Parent))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



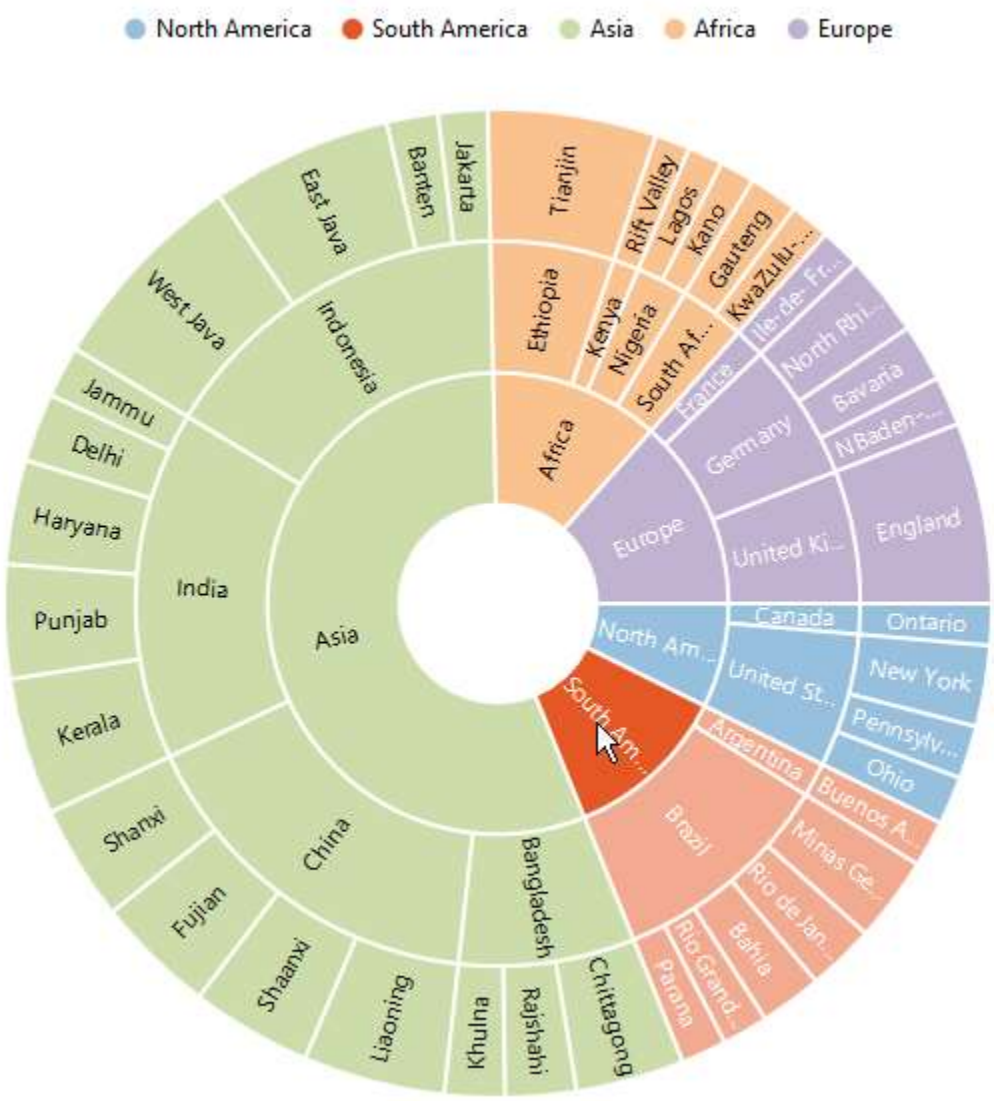
Point

To highlight the particular segment, the point mode of the highlight settings is used.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.HighlightSettings(highlight=>highlight.Enable(true).Mode(SunburstHighlig
htMode.Point))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

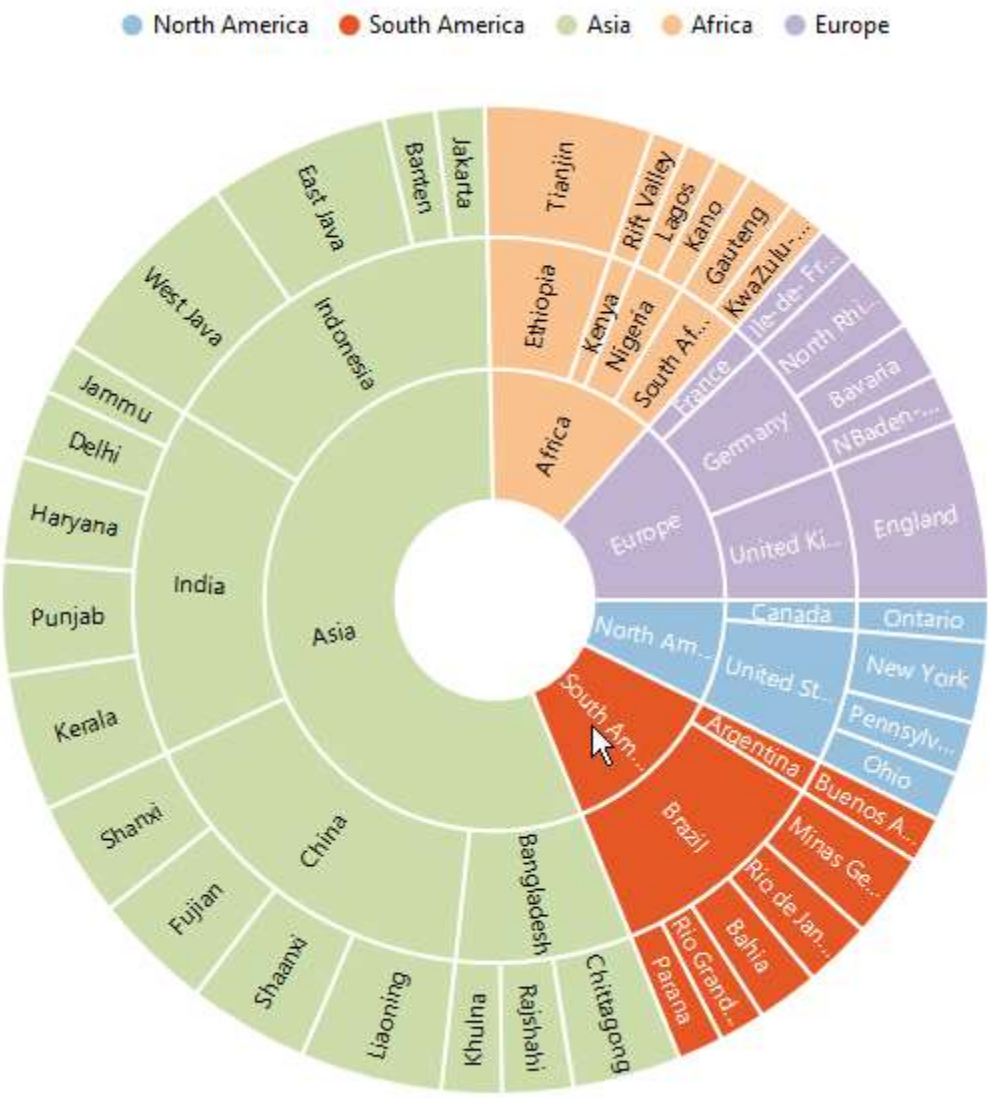


All

The following code snippet is used for the all mode of highlight settings

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.HighlightSettings(highlight=>highlight.Enable(true).Mode(SunburstHighlig
htMode.All))
)
```



Click here to view the online demo sample of Highlight in the Sunburst Chart.

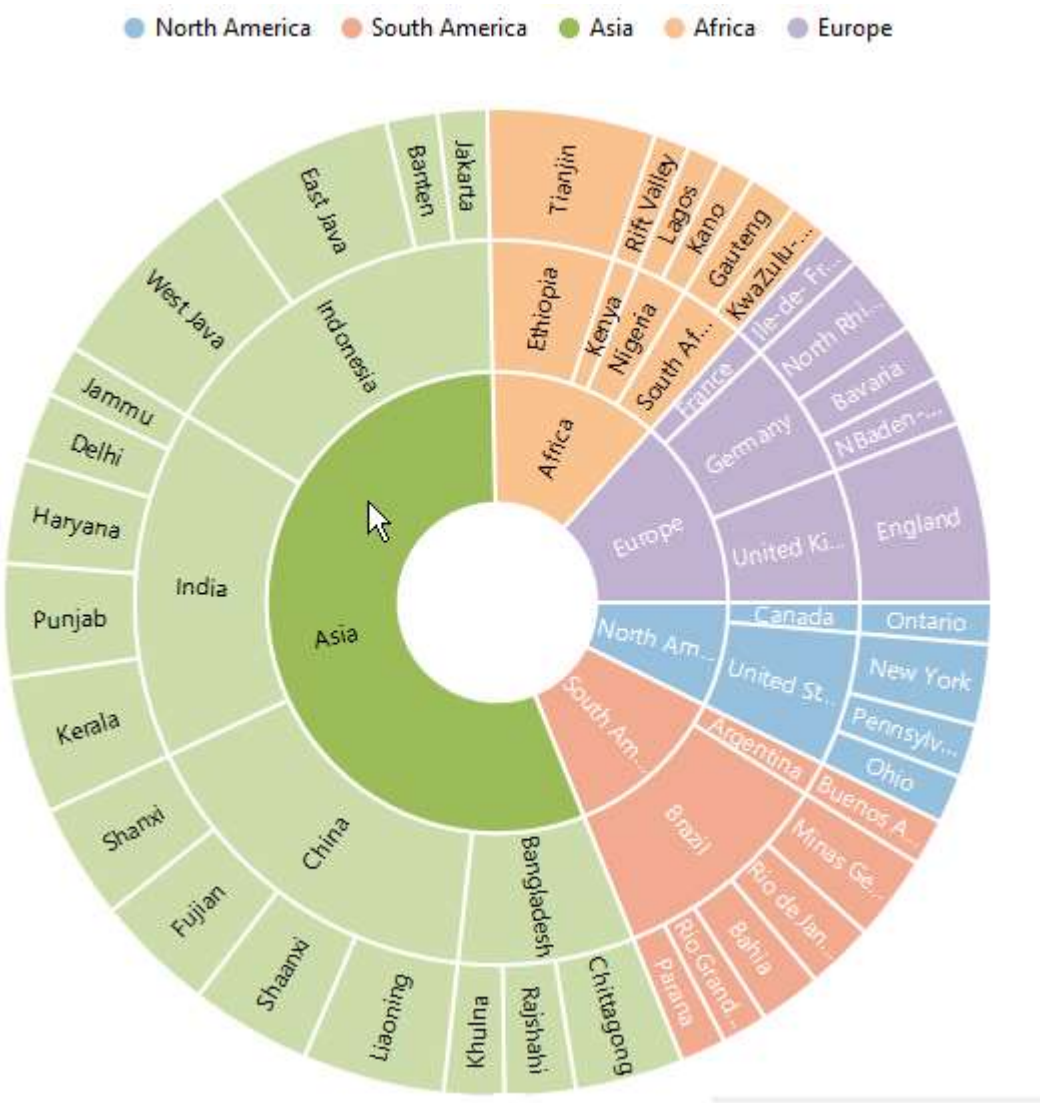
Selection

SunburstChart provides selection support for the points on mouse click. To enable the selection , set the *Enable* property to true in the **SelectionSettings**.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.SelectionSettings(selection=>selection.Enable(true))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Selection Display mode

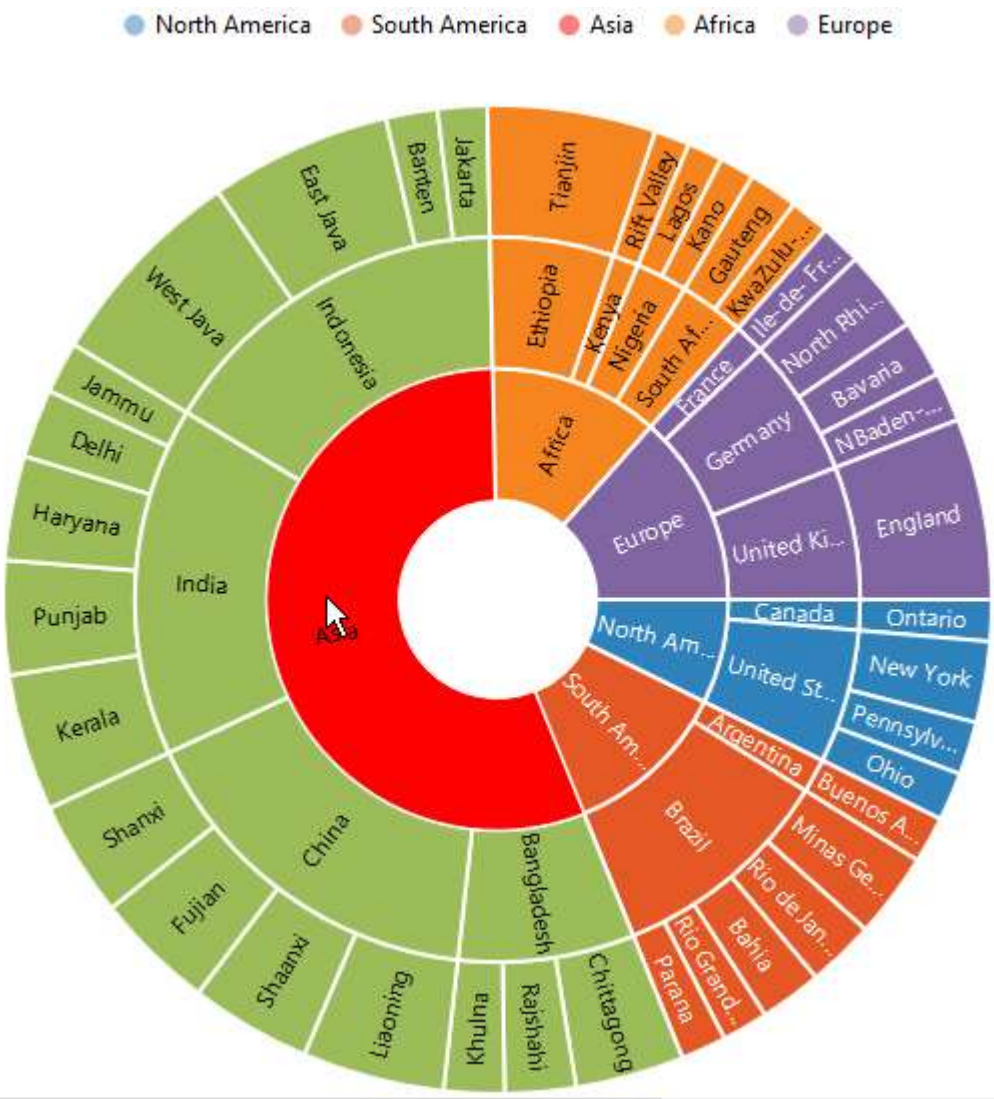
You can customize the selected segment appearance by using color or opacity. You can choose between color or opacity using the **Type** selection Settings.

selectionByColor – To display the selected segment appearance using color. selectionByOpacity – To display the selected segment appearance using opacity.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.SelectionSettings(selection=>selection.Enable(true).Type(SunburstSelecti
onType.Color).Color("Red"))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Selection Mode

Sunburst chart provides multiple option to represent the selected categories. You can select the segment categories by using the **Mode** property in selectionSettings

Child – To selection the child of selected parent. All – To selection the entire categories in group. *Parent – To selection the parent of selected child.* Single - To selection single item in the category.

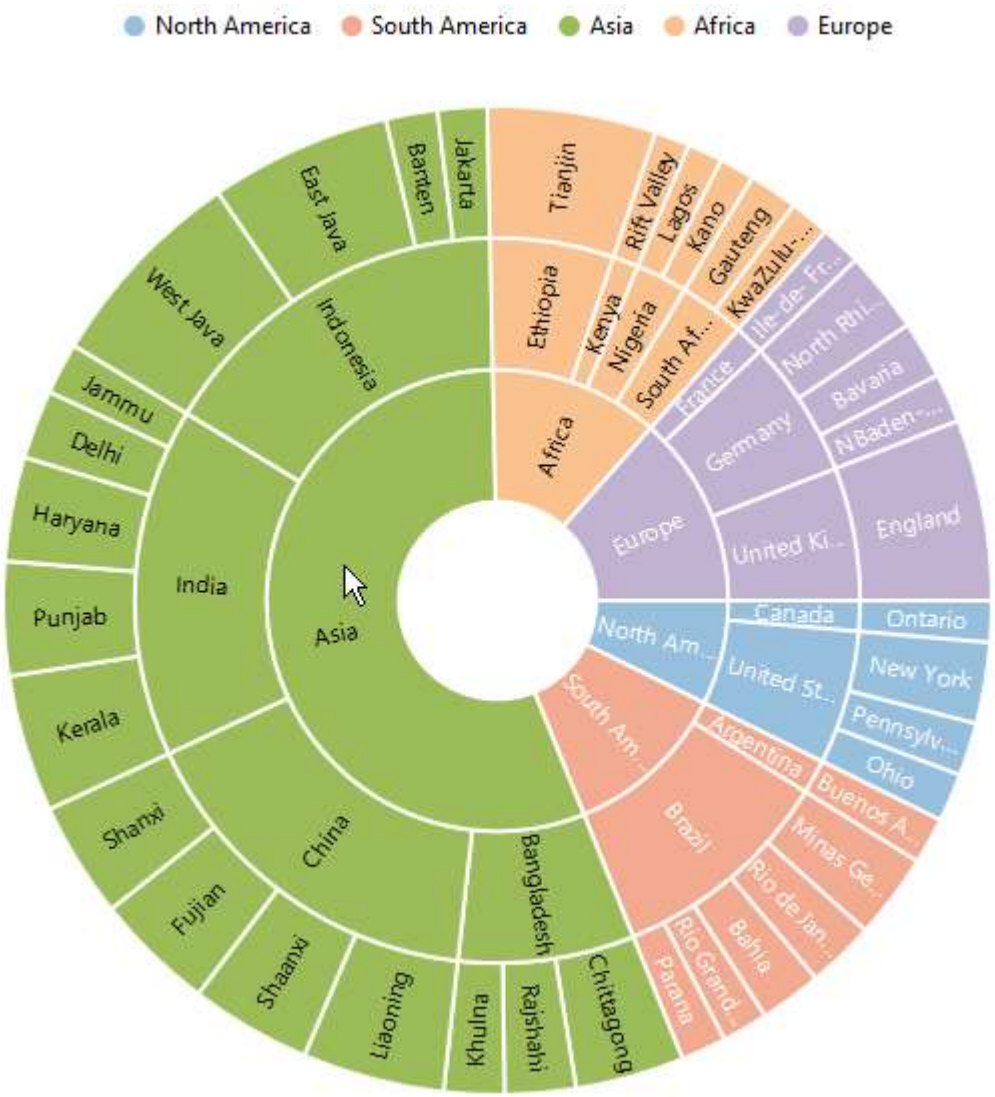
Child

The following code shows how to set the selection type as child

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.SelectionSettings(selection=>selection.Enable(true).Mode(SunburstSelecti
onMode.Child))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Parent

The parent mode can be enabled by using the below code

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.SelectionSettings(selection=>selection.Enable(true).Mode(SunburstSelecti
onMode.Parent))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



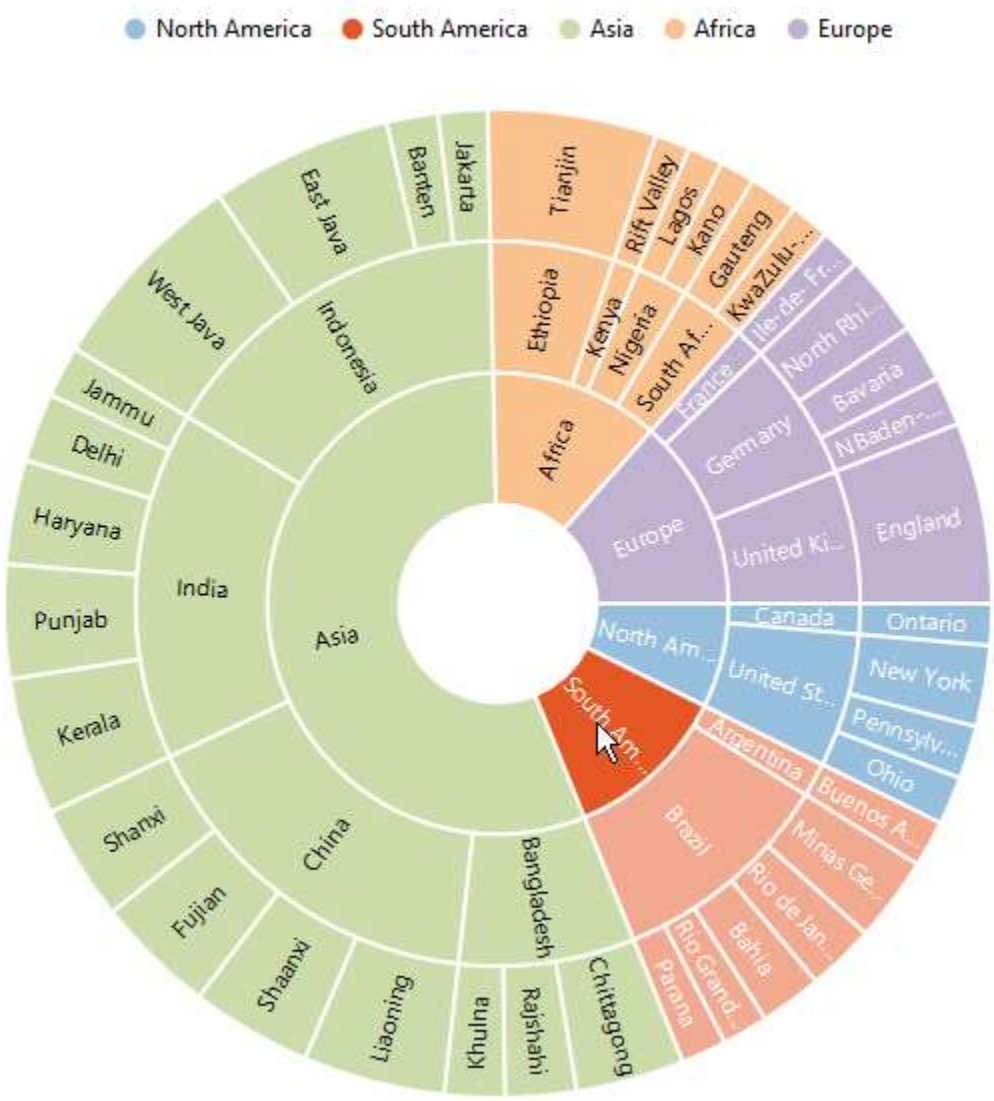
Point

To selection the particular segment, the point mode of the selection settings is used.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.SelectionSettings(selection=>selection.Enable(true).Mode(SunburstSelecti
onMode.Point))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



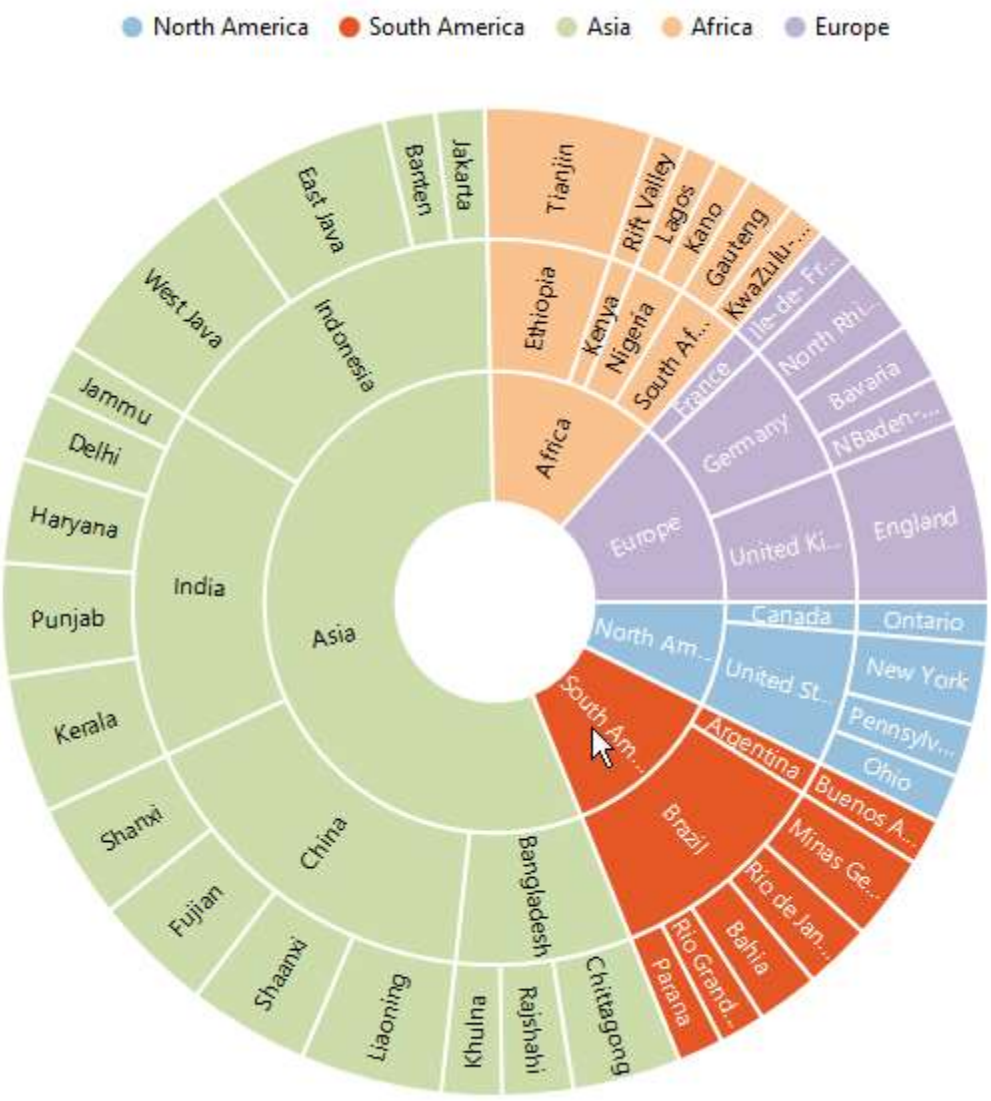
All

The following code snippet is used for the all mode of selection settings

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.SelectionSettings(selection=>selection.Enable(true).Mode(SunburstSelecti
onMode.All))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample of Selection in the Sunburst Chart.

Zooming

Sunburst chart provides zooming (drill down) experience with animation for both mouse and touch enabled devices. It allows you to Virtualize large sets of data into minimum data view.The zooming is achieved by using the property of *zoomSettings*

The following code shows how to initialize the zooming.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.ZoomSettings(zoom => zoom.Enable(true))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Zooming toolbar

By default, zooming toolbar will be enabled while zooming the segment.It contains both back and reset option.

You can align the zooming toolbar position by using *ToolbarHorizontalAlignment* and *ToolbarVerticalAlignment* property.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
.ZoomSettings(zoom =>
zoom.Enable(true).ToolbarHorizontalAlignment(SunburstHorizontalAlignment.
Left))
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample of Zooming in the Sunburst Chart.

Animation

Sunburst chart allows you to animate the chart segments. You can enable animation using **EnableAnimation** property.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Enable Animation
.EnableAnimation(true)
//...
)
```

Animation Types

Sunburst chart provide options to animate the chart segments in different ways using **AnimationType** property.

FadeIn – It gradually changes opacity of the chart segment.

Copyright © 2001 - 2020 Syncfusion Inc.

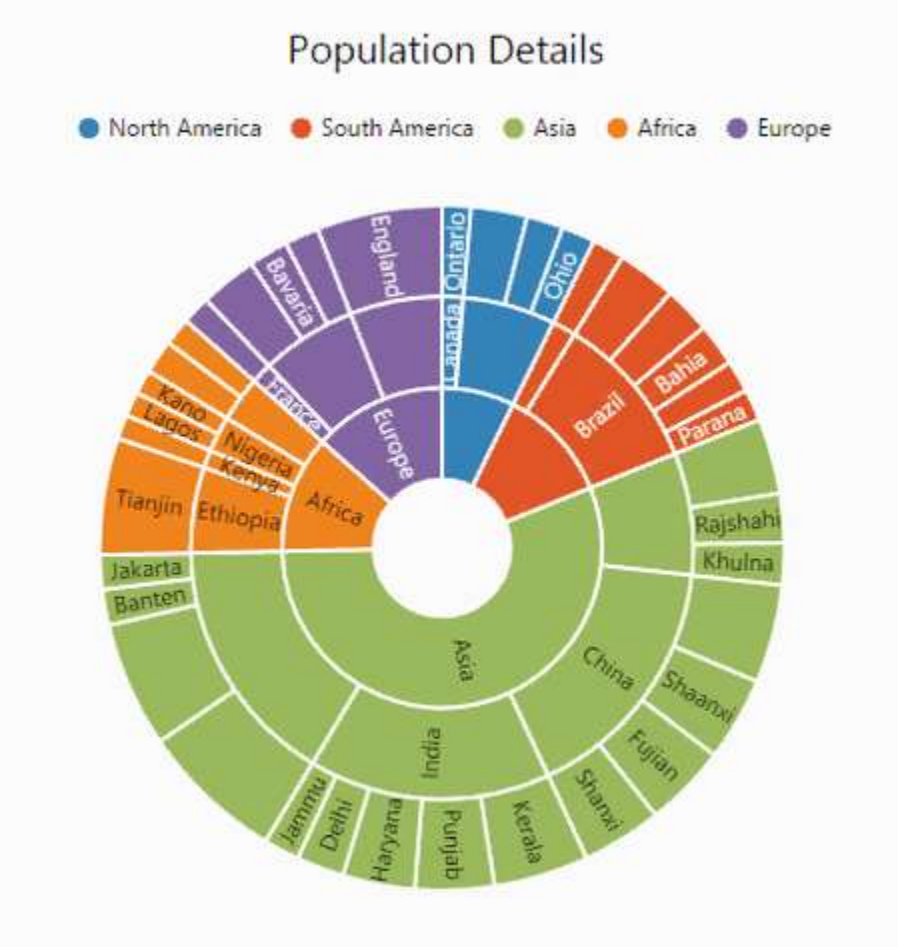
Rotation – During an animation, control rotate from 0 to 360 angle.

Fade In

The Fade In animation is enabled as follows

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Enable Animation
.EnableAnimation(true)
.SunburstAnimationType(SunburstAnimationType.FadeIn)
//...
)
```



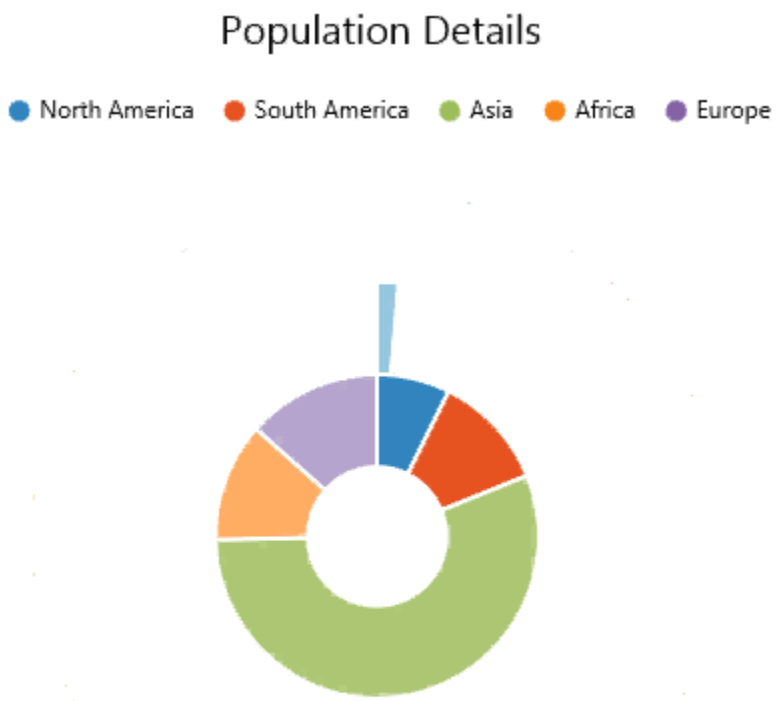
Rotation

The following example shows how to enable rotation animation in ejSunburstChart

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Enable Animation
.EnableAnimation(true)
.SunburstAnimationType(SunburstAnimationType.Rotation)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample of Animation in the Sunburst Chart.

Appearance

The appearance of the Sunburst Chart can be customized as shown below

Palette

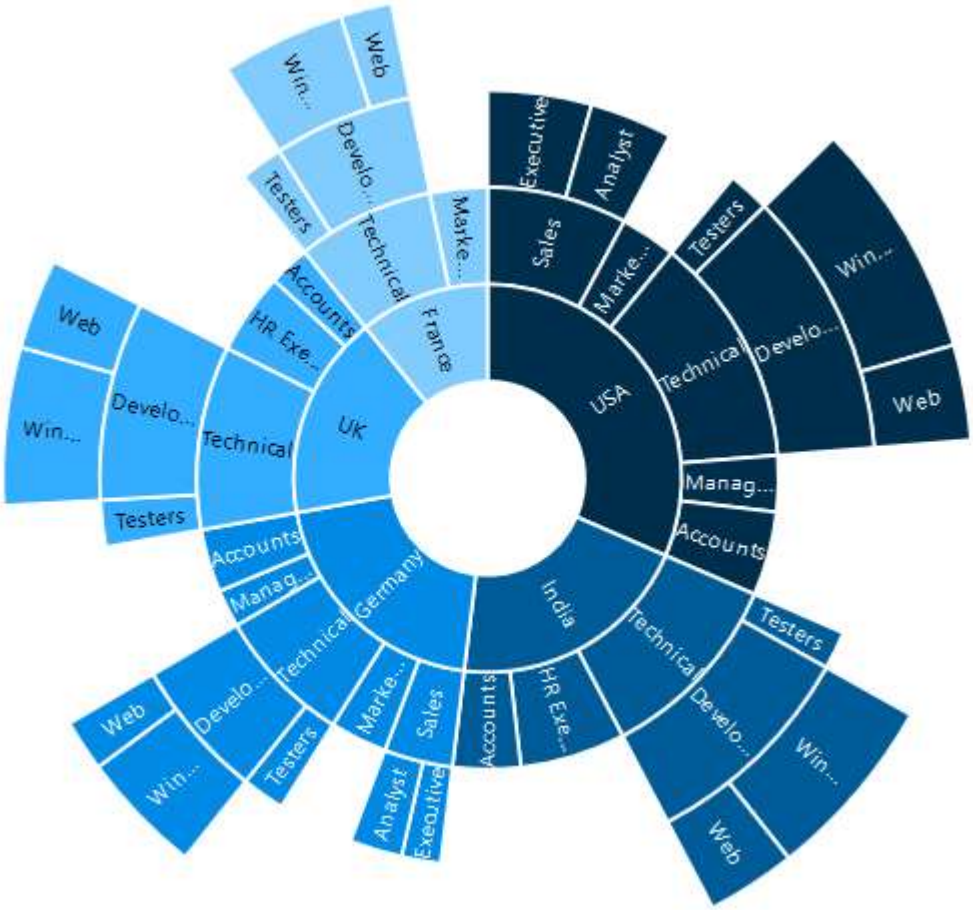
The Sunburst Chart displays different segments in different colors by default. You can customize the color of each segment by providing a custom color palette of your choice by using the **Palette** property.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Palette colors
.Palette(new List<string> { "#002e4d", "#005c99", "#008ae6", "#33adff",
"#80ccff" })
//...
)
```

The Sunburst Chart rendered with palette colors

Copyright © 2001 - 2020 Syncfusion Inc.



Built- in Themes

The Sunburst Chart supports different themes.

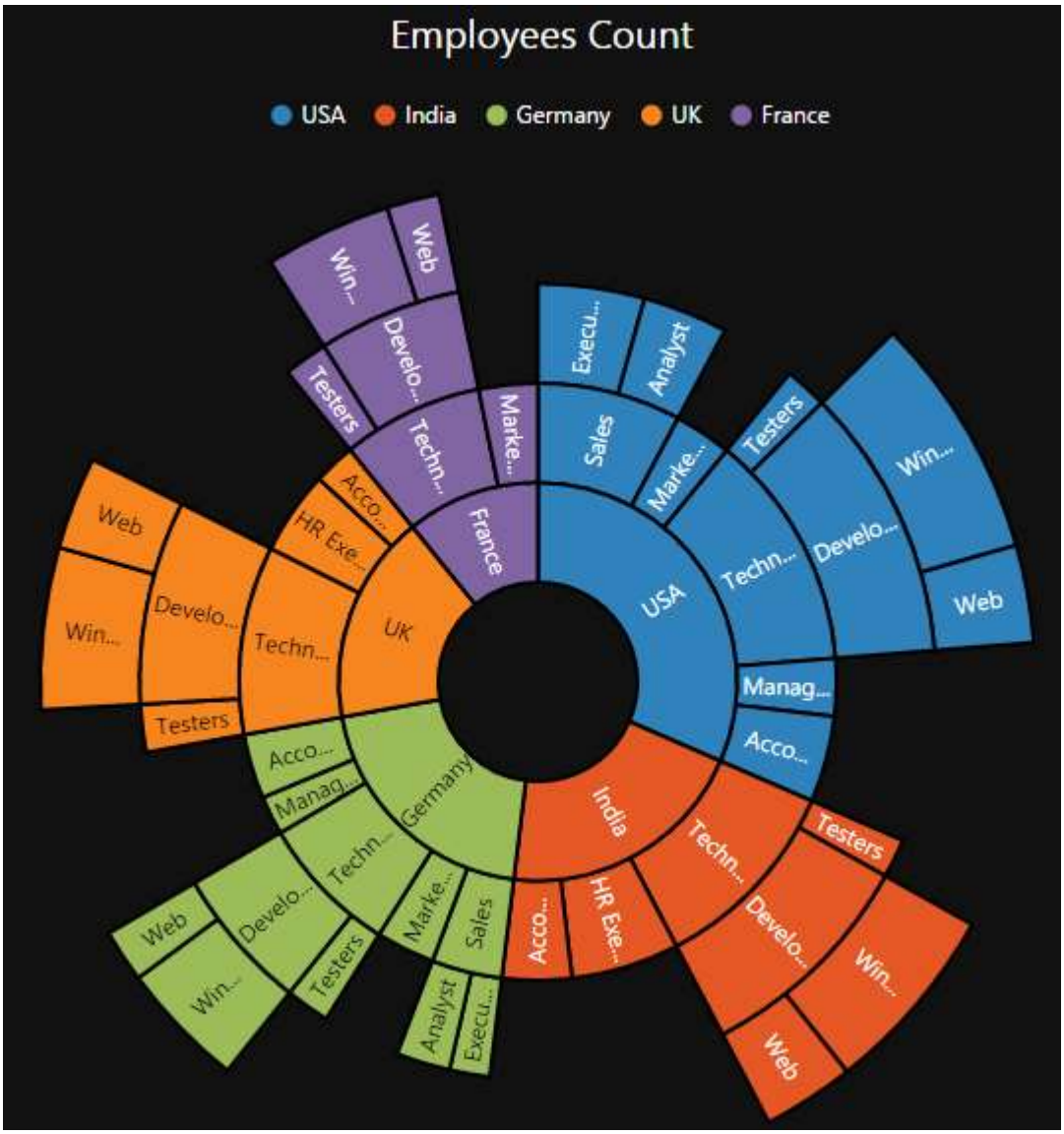
flatlight flatdark *gradientlight* gradientdark *azure* azuredark *lime* limedark *saffron* saffrondark *gradient-azure* gradient-azuredark *gradient-lime* gradient-limedark *gradient-saffron* gradient-saffrondark

You can set your desired theme by using the **Theme** property. **Flat light** is the default theme used in the Sunburst Chart.

ASPX-CS

```
@(Html.EJ().SunburstChart("chartContainer")
//...Palette colors
.Theme(Syncfusion.JavaScript.DataVisualization.SunburstTheme.FlatDark)
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Methods

redraw()

redraw the entire sunburst. You can call this method whenever you update, add or remove points from the data source or whenever you want to refresh the UI.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1"))
```

JS

```
var sun = $("#SunburstChart1").data("ejSunburstChart");
sun.redraw();
```

destroy ()

destroy the sunburst

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1"))
```

JS

```
var sun = $("#SunburstChart1").data("ejSunburstChart");
sun.destroy();
```

Events

Load

Fires before loading, you can use Load event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.Load("load")
)
```

JS

```
function load(){
// Do Something
}
```

PreRender

Fires before rendering sunburst, you can use PreRender event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.PreRender("prerender")
)
```

JS

```
function prerender(){
// Do Something
}
```

Loaded

Fires after rendering sunburst, you can use Loaded event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.Loaded("loaded")
)
```

JS

```
function loaded(){
// Do Something
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

DataLabelRendering

Fires before rendering the data label, you can use DataLabelRendering event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.DataLabelRendering("dataLabelRendering")
)
```

JS

```
function dataLabelRendering(){
// Do Something
}
```

SegmentRendering

Fires before rendering each segment, you can use SegmentRendering event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.SegmentRendering("segmentRendering")
)
```

JS

```
function segmentRendering(){
// Do Something
}
```

TitleRendering

Fires before rendering sunburst title, you can use TitleRendering event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.TitleRendering("titleRendering")
)
```

JS

```
function titleRendering(){
// Do Something
}
```

TooltipInitialize

Fires during initialization of tooltip, you can use TooltipInitialize event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.TooltipInitialize("tooltipInitialize")
)
```

JS

```
function tooltipInitialize(){
// Do Something
}
```

PointRegionClick

Fires after clicking the point in sunburst, you can use PointRegionClick event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.PointRegionClick("pointRegionClick")
)
```

JS

```
function pointRegionClick(){
// Do Something
}
```

PointRegionMouseMove

Fires while moving the mouse over sunburst points, you can use PointRegionMouseMove event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.PointRegionMouseMove("pointRegionMouseMove")
)
```

JS

```
function pointRegionMouseMove(){
// Do Something
}
```

DrillDownClick

Fires when clicking the point to perform drilldown, you can use DrillDownClick event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.DrillDownClick("drillDownClick")
)
```

JS

```
function drillDownClick(){
// Do Something
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

DrillDownBack

Fires when resetting drilldown points, you can use DrillDownBack event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.DrillDownBack("drillDownBack")
)
```

JS

```
function drillDownBack(){
// Do Something
}
```

DrillDownReset

Fires after resetting the sunburst points, you can use DrillDownReset event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.DrillDownReset("drillDownReset")
)
```

JS

```
function drillDownReset(){
// Do Something
}
```

LegendItemRendering

Fires before rendering the legend item, you can use LegendItemRendering event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.LegendItemRendering("legendItemRendering")
)
```

JS

```
function legendItemRendering(){
// Do Something
}
```

LegendItemClick

Fires when clicking the legend item, you can use LegendItemClick event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.LegendItemClick("legendItemClick")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

JS

```
function legendItemClick(){
// Do Something
}
```

Click

Fires when clicking the sunburst points, you can use Click event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.Click("click")
)
```

JS

```
function click(){
// Do Something
}
```

DoubleClick

Fires when double clicking the sunburst points, you can use DoubleClick event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.DoubleClick("doubleClick")
)
```

JS

```
function doubleClick(){
// Do Something
}
```

RightClick

Fires when right clicking the sunburst points, you can use RightClick event.

ASPX-CS

```
@(Html.EJ().SunburstChart("SunburstChart1")
.RightClick("rightClick")
)
```

JS

```
function rightClick(){
// Do Something
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Tab

Tab

The Tab control is an interface where list of items are expanded from a single item. Each Tab panel defines its header text or header template, as well as a content template. Tab items are dynamically added and removed. Tabs can be loaded with AJAX content that is useful for building dashboards where space is limited.

Key Features

- Collapsible header: All headers are collapsible.
- AJAX load: Load AJAX content in the Tab content panel.
- Close button: Provides a close button for each Tab item.
- Custom event for expanding header: Activate a Tab item on a single click, or bind custom events such as mouse over or mouse up to activate a Tab.
- Header orientation: Change the header position to the top or bottom of the control.
- Persist: Save the current model value to browser cookies to maintain the state.
- Height style: Adjust the content panel height.
- RTL: Tab headers and content can display right-to-left languages.
- Add Tab dynamically: New Tab items can be added and removed at run time.
- Theme: Essential ASP.NET MVC controls feature 12 built-in themes, six flat themes and six with gradient effects, and also support custom skin options to set user-defined themes.
- Keyboard navigation: You can interact with the control using the keyboard.

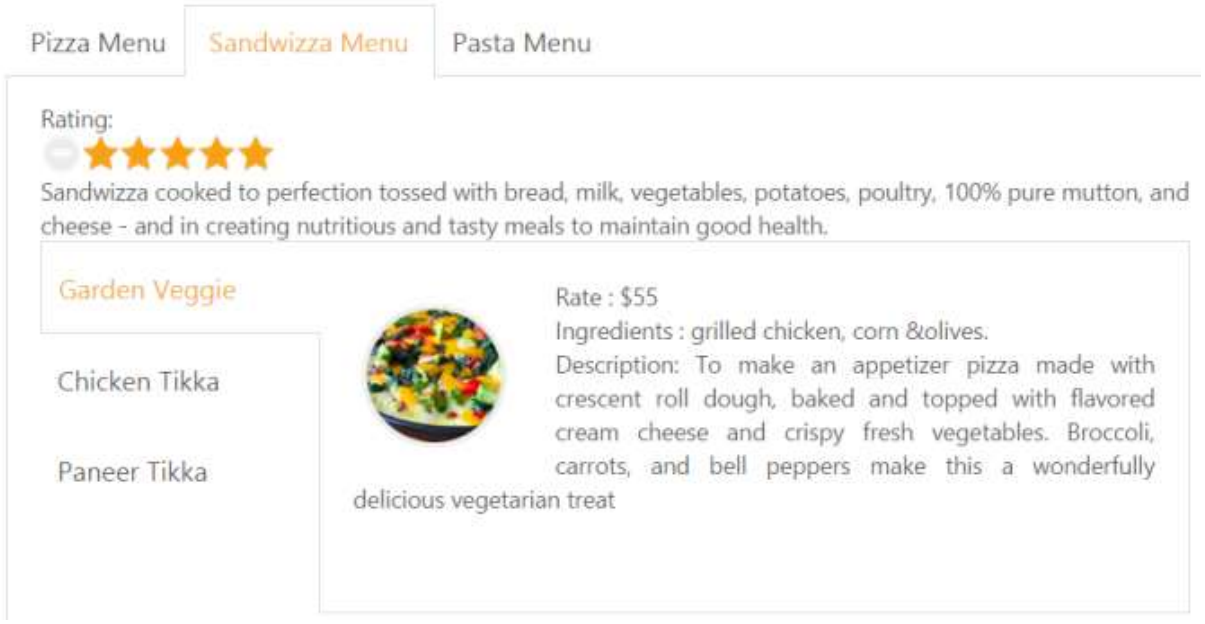
Getting Started

This section explains briefly about how to create a Tab Control in your application with ASP.NET MVC.

Create your first Tab Control in MVC

The ASP.NET MVCTab control is an interface that displays the content in multiple sections. A TabPanel consists of HeaderText as well as a ContentTemplate. Tab is useful for a dashboard that is having limited space. The following section guides you to customize the Tab for displaying Hotel menu items, its rate details and the ingredients on demand.

Copyright © 2001 - 2020 Syncfusion Inc.



Tab control with Hotel Menu items

Create Tab Control

The ASP.NET MVCTab widget basically builds a dynamic, interactive, menu-driven interface from your content. The content can be text, graphics or HTML. You can create the Tab headers using Text and Content template property

The following steps are used to create Tab control.

1. You can create an MVC Project and add necessary assembly and script with the help of the given MVC-Getting Started Documentation.
2. Add the mentioned code to the corresponding view page for Tab rendering.

ASPX-CS

```
@Html.EJ().Tab("Dish").Items(data =>
{
data.Add().ID("dish").Text("dish Menu").ContentTemplate(@<div></div>);
data.Add().ID("type").Text("dish Type").ContentTemplate(@<div></div>);
data.Add().ID("sandwich").Text("Sandwich
Type").ContentTemplate(@<div></div>);
})
```

The following output is displayed.



Tab control with Header

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Content

In this application, a detailed description is provided to each item. You can specify the contents in the Tab section within the Content template.

ASPX-CS

```
@Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("dish").Text("dish Menu").ContentTemplate(@<div> Rating:
@Html.EJ().Rating("RatingDish").Value(4).Precision(Precision.Exact)
<!--Food item description-->
<p>dish cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>);
})
```

You can provide more customization to the Tab by using rating control, to describe an item's price.

Create the Rating

The ASP.NET MVC Rating control provides an intuitive rating experience that allows you to select the number of stars that represents the rating. The following code example explains you the creation of rating control. We can create the Rating control using HTML helper. We can render the rating control and its description inside the content template of first tab control.

For more information about rating, refer to the following link:

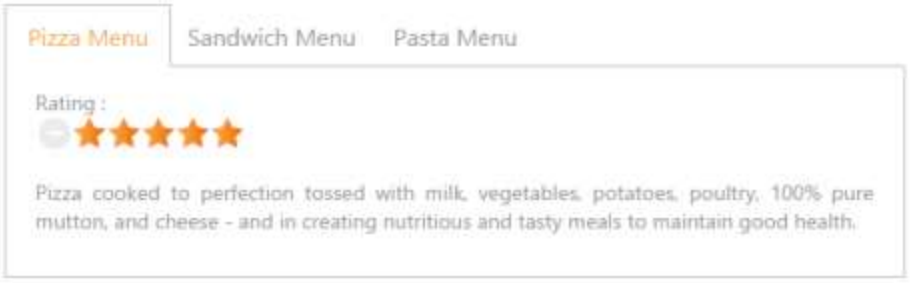
http://help.syncfusion.com/aspnetmvc

ASPX-CS

```
<!--Use the following codes with above Html contents-->
@Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("dish").Text("dish Menu").ContentTemplate(@<div> Rating:
@Html.EJ().Rating("RatingDish").Value(4).Precision(Precision.Exact)
<!--Food item description-->
<p>dish cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>);
data.Add().ID("type").Text("dish Type").ContentTemplate(@<div></div>);
data.Add().ID("sandwich").Text("Sandwich
Type").ContentTemplate(@<div></div>);
})
```

The following screenshot is the output for the given code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Sub Tab with Content

Each item has a variety of options, and these options are also specified in the limited space. So you can choose the Tab control that is used within the root Tab to specify more details.

The following code example represents sub Tab control rendering using helper function.

ASPX-CS

```
<!--Use the following codes with in the  above Html -->
@Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("dish").Text("dish Menu").ContentTemplate(@<div> Rating:
@Html.EJ().Rating("RatingDish").Value(4).Precision(Precision.Exact)
<!--Food item description-->
<p>dish cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
@firstTab()
</div>);
data.Add().ID("type").Text("dish Type").ContentTemplate(@<div></div>);
data.Add().ID("sandwich").Text("Sandwich
Type").ContentTemplate(@<div></div>);
})
@helper firstTab()
{
@Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("Corn-Spinach").Text("Corn Spinach").ContentTemplate(@<div
class="e-content">
<img src=" http://js.syncfusion.com/demos/web/images/accordion/corn-and-
spinach-05.png" alt="corn-spinach">
<div class="ingredients">
Rate     : $70<br />
Ingredients : cheese, sweet corn &#38; green capsicums.
<br />
Description: Small dish bases are topped with spinach and paneer and
fresh cream, a nice layer of mozzarella cheese. This is baked until the
cheese is all hot and gooey.
</div>
</div>
);
data.Add().ID("ChickenDelite").Text("Chicken-
Delite").ContentTemplate(@<div class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/chicken-
delite.png" alt="chicken-delite">
<div class="ingredients">
Rate     : $100<br />
```

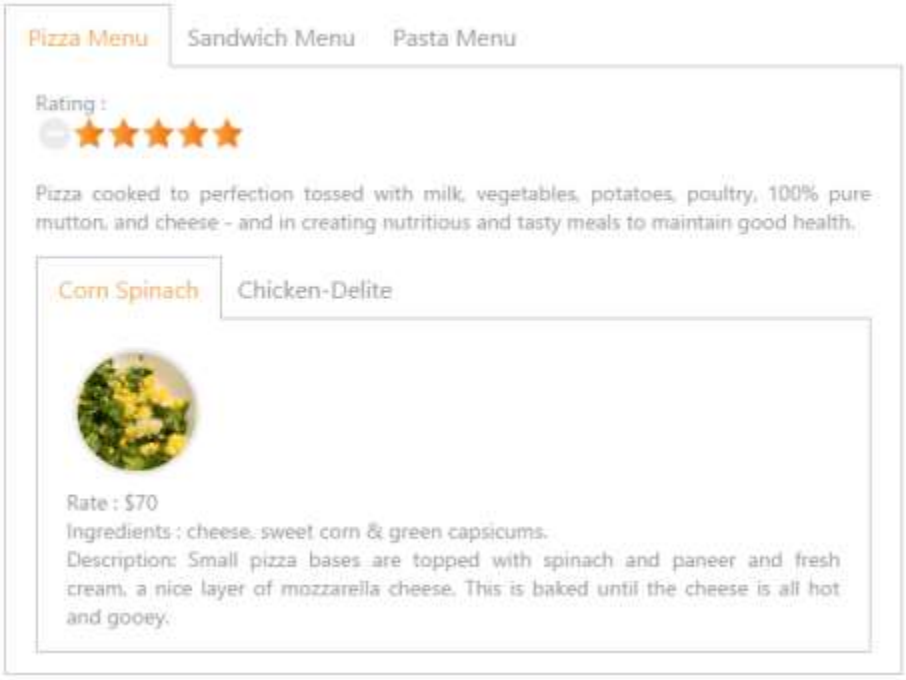
```
Ingredients : cheese, chicken chunks, onions &#38; pineapple chunks.
<br />
Description: This is a tasty, elegant chicken dish that is easy to
prepare.
</div>
</div>
);
})
});
}
```

The following code example is used to position the image and content.

CSS

```
<style type="text/css" class="cssStyles">
/*reuse the previous rating control style section code*/
.ingredients
{
height: 180px;
margin-top: 8px;
}
img
{
float: left;
margin: 10px 26px 5px 1px;
}
</style>
```

The following screenshot illustrates the first Tab with the sub Tab control.



Copyright © 2001 - 2020 Syncfusion Inc.

Orientation Change

Now, you can learn how to set the sub Tab orientation to vertical. By default, Tab control is rendered in horizontal orientation. You can change this orientation to vertical by using the HeaderPosition property. In the following code section, set the Tab header orientation as Left.

The following code section renders the sub Tab element in the vertical orientation.

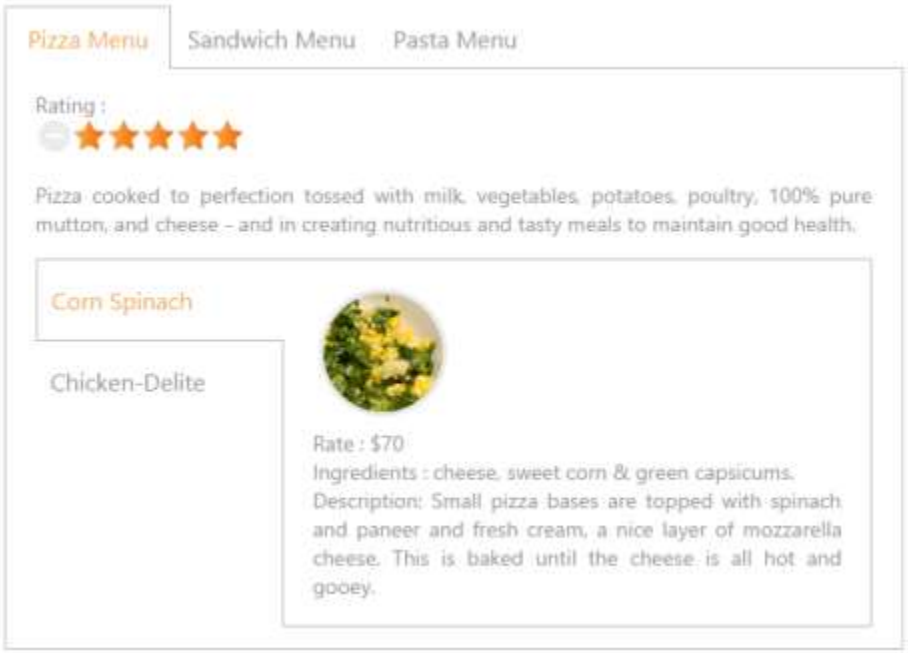
ASPX-CS

```
<!--Use the following codes with in the  above Html -->
@Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("dish").Text("dish Menu").ContentTemplate(@<div>
Rating:
@Html.EJ().Rating("RatingDish").Value(4).Precision(Precision.Exact)
<!--Food item description-->
<p>dish cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
@firstTab()
</div>);
data.Add().ID("type").Text("dish Type").ContentTemplate(@<div></div>);
data.Add().ID("sandwich").Text("Sandwich
Type").ContentTemplate(@<div></div>);
})
@helper firstTab()
{
@Html.EJ().Tab("dish").HeaderPosition(HeaderPosition.Left).Height("221").
Items(data =>
{
data.Add().ID("Corn-Spinach").Text("Corn Spinach").ContentTemplate(@<div
class="e-content">
<img src=" http://js.syncfusion.com/demos/web/images/accordion/corn-and-
spinach-05.png" alt="corn-spinach">
<div class="ingredients">
Rate     : $70<br />
Ingredients : cheese, sweet corn &#38; green capsicums.
<br />
Description: Small dish bases are topped with spinach and paneer and
fresh cream, a nice layer of mozzarella cheese. This is baked until the
cheese is all hot and gooey.
</div>
</div>
);
data.Add().ID("ChickenDelite").Text("Chicken-
Delite").ContentTemplate(@<div class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/chicken-
delite.png" alt="chicken-delite">
<div class="ingredients">
Rate     : $100<br />
Ingredients : cheese, chicken chunks, onions &#38; pineapple chunks.
<br />
Description: This is a tasty, elegant chicken dish that is easy to
prepare.
</div>
</div>
);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
});
```

The following screenshot is the output of above steps.



Configure Contents to remaining Tab items

The second and third Tab contents are declared in the same method as of the first Tab content declaration. These Tabs also consist of rating and sub Tab controls.

ASPX-CS

```
@{Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("type").Text("dish Menu").ContentTemplate(@<div>
dish cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
@firstTab()
</div>);
data.Add().ID("sandwichType").Text("Sandwizza
Menu").ContentTemplate(@<div>
Sandwizza cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.
@secondTab()
</div>);
data.Add().ID("Pasta").Text("Pasta Menu").ContentTemplate(@<div>
</div>);
}).Render();
@helper secondTab(){
@Html.EJ().Tab("SandwichMenu").HeaderPosition(HeaderPosition.Left).Height
("221").Items(data =>
{
```

```
data.Add().ID("GardenVeggie").Text("Garden Veggie").ContentTemplate(@<div
class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/garden-
veggie.png" alt="garden-veggie ">
<div class="ingredients">
Rate     : $55<br />
Ingredients : grilled chicken, corn &#38;olives.
<br />
Description: To make an appetizer dish made with crescent roll dough,
baked and topped with flavored cream cheese and crispy fresh vegetables.
Broccoli, carrots, and bell peppers make this a wonderfully delicious
vegetarian treat
</div>
</div>);
data.Add().ID("ChickenTikka").Text("Chicken Tikka
").ContentTemplate(@<div class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/chicken-
tikka.png" alt="chicken-tikka">
<div class="ingredients">
Rate     : $100<br />
Ingredients : onions, grilled chicken, chicken salami &#38; tomatoes.
<br />
Description: Juicy chunks of boneless chicken roasted on open fire.This
takeaway favorite is freezer-friendly and quick to reheat, giving you the
chance to get ahead.
</div>
</div>);
data.Add().ID("PaneerTikka").Text("Paneer Tikka  ").ContentTemplate(@<div
class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/paneer-
tikka.png" alt="paneer-tikka">
<div class="ingredients">
Rate     : $150
<br />
Ingredients : onions, paneer & tomatoes.
<br />
Description: Delve into the tasty Paneer Tikka Kebabs made from marinated
paneer or cottage cubes.
Relish these grilled delicacies with green mint chutney and onion rings.
</div>
</div>);
})
}
</div>
```

Add third Tab contents in element during initialization using content template option.

ASPX-CS

```
@{Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("type").Text("dish Menu").ContentTemplate(@<div>
dish cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
@firstTab()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>);
data.Add().ID("sandwichType").Text("Sandwizza
Menu").ContentTemplate(@<div>
Sandwizza cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.
@secondTab()
</div>);
data.Add().ID("Pasta").Text("Pasta Menu").ContentTemplate(@<div>
Pasta cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
@thirdTab()
</div>);
}).Render();
@helper thirdTab(){
@Html.EJ().Tab("PastaMenu").HeaderPosition(HeaderPosition.Left).Height("2
21").Items(data =>
{
data.Add().ID("KheemaPasta").Text("Kheema Pasta ").ContentTemplate(@<div
class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/corn-and-
spinach.png" alt="kheema-pasta ">
<div class="ingredients">
<p>
Rate : $30<br />
Ingredients : chicken, onions, chilly, garlic &#38; tomatoes.<br />
Description: Kheema pasta dish make with veg or non-veg type.It is
delicious and can be served for dinner, brunch or evening snack.
</p>
</div>
</div>);
data.Add().ID("TunaPasta").Text("Tuna Pasta").ContentTemplate(@<div
class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/garden-
fresh.png" alt="tuna-pasta">
<div class="ingredients">
<p>
Rate : $55<br />
Ingredients : tomato ,olive, onion &#38;garlic.<br />
Description: Canned tuna is used to make this yummy tomato sauce.
</p>
</div>
</div> );
data.Add().ID("ChannaPasta").Text("Channa Pasta").ContentTemplate(@<div
class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/zesty-
mushroons-and-tomatoes.png" alt="channa-asta">
<div class="ingredients">
<p>
Rate : $30<br />
Ingredients : sauntered spinach mix, sweet corn, parsley &#38;mozzarella
cheese. .<br />
Description: This is a pasta dish make with leftover Channa masala
(choler). This can be made from scratch too by making the Channa masala
first and then tossing in the cooked pasta.
</p>
```

Copyright © 2001 - 2020 Syncfusion Inc.

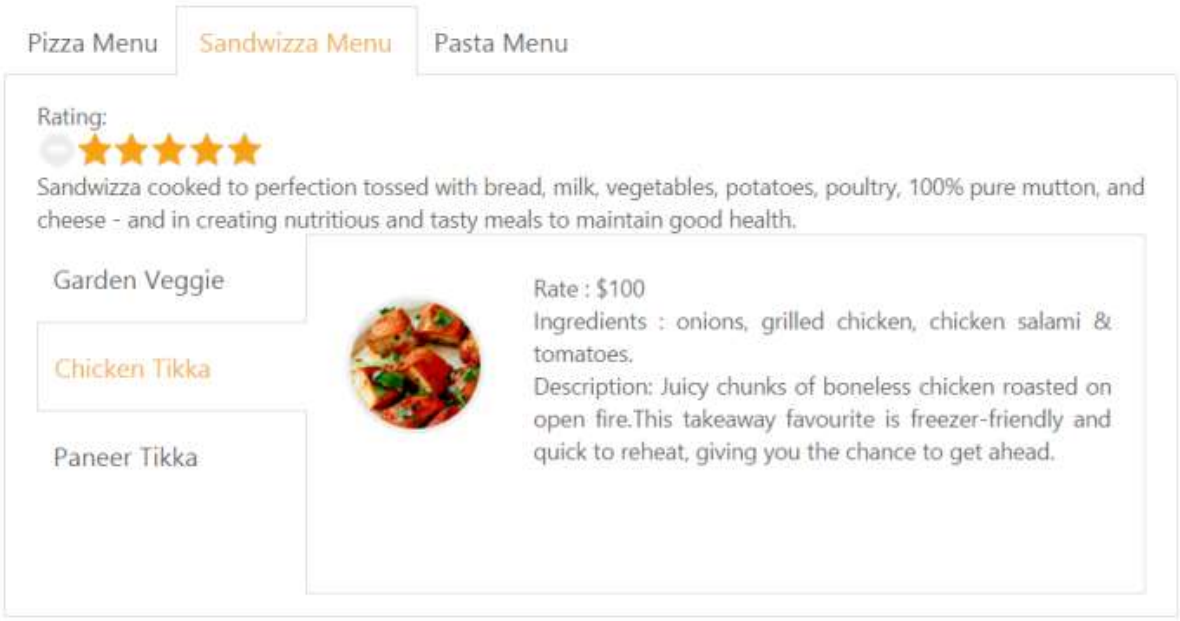
```
</div>
</div>);
})
}
```

Apply the following styles to the Tab.

CSS

```
<style type="text/css" class="cssStyles">
/*to reuse the previous style section code and following css*/
.sandwich-image, .pasta-image
{
height: 25px;
width: 25px;
}
.sandwich-image
{
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_garden-fresh.png") no-repeat;
}
.pasta-image
{
background: url("http://js.syncfusion.com/UG/Web/Content/rsz_garden-veggie.png") no-repeat;
}
</style>
```

The following screenshot illustrates the second Tab contents in Tab and the final hotel menu with rating, description and ingredients of the item in the Tab interface.



Copyright © 2001 - 2020 Syncfusion Inc.

Behavior Settings

Close Button

By default, Tab contents are rendered without CloseButton. You can add the CloseButton by setting the 'ShowCloseButton' property to 'true'. When you move cursor over the Tab headers, the CloseButton is displayed.

The following code example is used to render the Tab widget with CloseButton.

1. Add the following code in your view page for simple Tab creation with CloseButton.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with close button.
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).ShowCloseButton(true).Render();}
</div>
```

2. The following screenshot illustrates the Tab with CloseButton.



Orientation

By default, Tab control renders in horizontal orientation. You can change the Orientation to vertical using the 'HeaderPosition' property. Using this property, you can customize the header by " Top"," Bottom", "Left", and "Right".

The following code example is used to render the sub Tab widget in the vertical orientation.

1. Add the following code in your view page for Tab orientation.

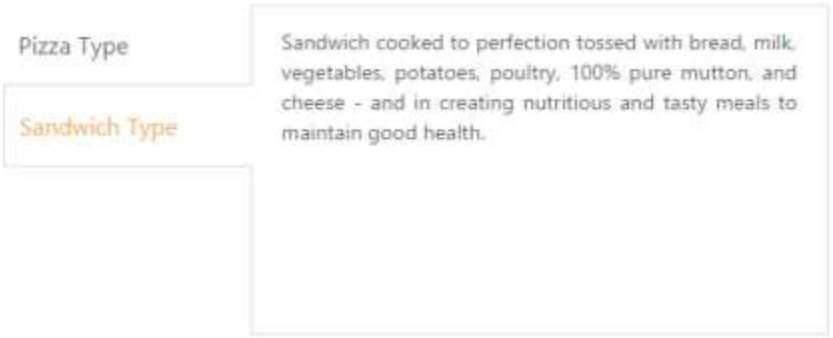
ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with customized orientation.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).HeaderPosition(HeaderPosition.Left).Height("220").Render();}
</div>
```

2. The following screenshot illustrates the sub Tab with vertical orientation.



State Maintenance

When the page gets refreshed or reloaded, the Tab state is changed (i.e.) the focus is moved to start Tab. You can maintain the state of the Tab by using 'EnablePersistence' property. When this property is set to 'true', it retains the state.

The following code example is used to render the Tab widget with state maintenance.

1. Add the following code in your view page for Tab state maintenance.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with state maintenance.
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>);
}).EnablePersistence(true).Render();}
</div>
```

2. The following screenshot illustrates the Tab with State maintenance.



State before page refresh



Appearance and Styling

Header Image Customization

To set the Tab header image for each Tab item you need to specify image in “ImageCssClass” property during the TabItem declaration.

The following code example is used to add the header image for the root Tab header element.

1. Add the following code in your view page to render Tab with header image.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with header image.
<div style="width: 500px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("pizza").Text("Pizza").ContentTemplate(@<div id="pizzatype"
style="background-color: #F5F5F5">
<p>Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.</p>
</div>);
data.Add().ID("sandwichType").Text("Sandwich").ContentTemplate(@<div
id="sandwichType" style="background-color: #F5F5F5">
<p>Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.</p>
</div>);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).SelectedItemIndex(1).ClientSideEvents(e
=>e.Create("create")).Render();}
</div>
<script>
function create(args)
{
$("#dish ul li:first").find("a").prepend('<span class="dish
pizzaImg"></span>');
$("#dish ul li:last").find("a").prepend('<span class="dish
sandwichImg"></span>');
}
</script>
```

2. Add following CSS for header image customization.

CSS

```
<style type="text/css" class="cssStyles">
.dish {
display: inline-block;
}
.pizzaImg {
background:
url("http://js.syncfusion.com/demos/web/content/images/accordion/chicken-
delite.png") no-repeat;
height: 25px;
width: 25px;
margin:2px;
background-position: -18px;
}
.sandwichImg {
background:
url("http://js.syncfusion.com/demos/web/content/images/accordion/garden-
fresh.png") no-repeat;
padding:1px;
height: 25px;
width: 25px;
margin:2px;
background-position: -18px;
}
</style>
```

3. The following screenshot illustrates the Tab with the customized header image.



Copyright © 2001 - 2020 Syncfusion Inc.

Header Image Customization

Rounded corner

By enabling 'ShowRoundedCorner' property, you can customize the shape of the Tab widget from regular rectangular shape to rounded rectangle shape that is set to 'false' by default.

The following code example is used to render the Tab widget with rounded corner.

1. Add the following code in your view page to render Tab with rounder corner.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with rounded corner.
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).ShowRoundedCorner(true).Render();}
</div>
```

2. The following screenshot illustrates the Tab with Rounded corner.



Tab with rounded corner

Enable/Disable

You can enable or disable the Tab widget by 'Enabled' property. By default, the property set to 'true'.

The following code example is used to render the Tab widget with enable/disable.

1. Add the following code in your view page to render Tab with enable/disable.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with Enable/Disable format.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).Enabled(false).Render();}
</div>
```

2. The following screenshot illustrates the Tab with disabled format.

Tab with disabled format

Enabling Reload Icon

Without refresh/reload the whole page, you can reload a particular Tab using Reload icon. The Reload icon is appeared at right corner of the Tab by enabling the property 'ShowReloadIcon' to 'true'. When you move cursor over the Tab headers, the Reload icon is displayed. By default the property value is set to 'false'.

The following code example is used to render the Tab widget with Reload icon.

1. Add the following code in your view page to render Tab with Reload icon.

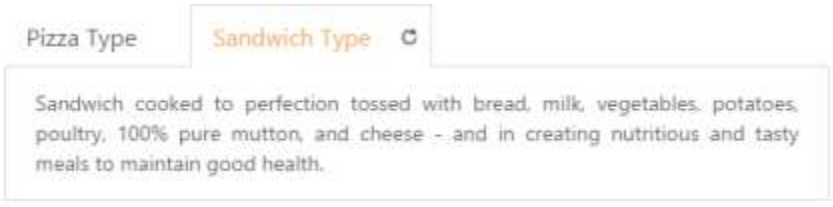
ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with reload icon.
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).ShowReloadIcon(true).Render();}
</div>
```

2. The following screenshot illustrates the Tab with Reload icon.



Tab with reload icon

Collapsible Tabs

You can collapse the Tab content by enabling the 'Collapsible' property to 'true'. When the property is set to 'true' then click the active Tab header, the Tab contents are hidden. By default, the property value is set to 'false'.

The following code example is used to render the Tab widget with customized collapsible mode.

1. Add the following code in your view page to render Tab with customized collapsible mode.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with collapsible mode.
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).Collapsible(true).Render();}
</div>
```

2. The following screenshot illustrates the Tab with customized collapsible mode.



Tab with customized collapsible mode

Copyright © 2001 - 2020 Syncfusion Inc.

Adjusting Tab Size

Height Adjust Mode and Height

The height of the Tab widget is customized by 'height' property. The Tab widget height depends on 'HeightAdjustMode' property. Using the HeightAdjustMode property, you can adjust height by "Content", "Auto", "Fill". By default the HeightAdjustMode is set as content.

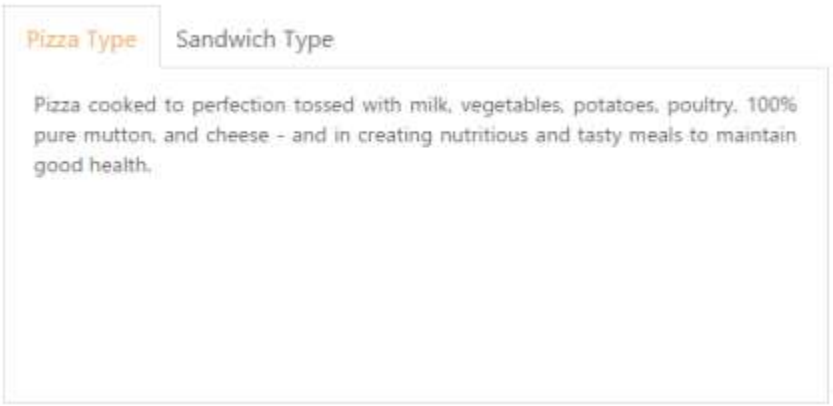
The following code example is used to render the Tab widget with customized height and height adjust mode.

1. Add the following code in your view page to render Tab with customized height and height adjust mode.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with customized height and height adjust mode.
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwichType").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).HeightAdjustMode(HeightAdjustMode.Fill).Height("300px").Render();}
</div>
```

2. The following screenshot illustrates the Tab with customized height and height adjust mode.



Tab with customized height and height adjust mode

Width

The width of the Tab widget is customized by using 'Width' property that accepts only the pixel values.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example is used to render the Tab widget with customized width.

1. Add the following code in your view page to render Tab with customized width.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with customized width.
<div>
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwichType").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).Width("450").Render();}
</div>
```

2. The following screenshot illustrates the Tab with customized width.



Tab with customized width

Theme

Tab control's style and appearance are controlled based on CSS classes. In order to apply styles to the Tab control, you can refer 2 files namely, ej.widgets.core.min.css and ej.theme.min.css. When the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 13 themes support available for Tab control namely

- default-theme
- bootstrap-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure

Copyright © 2001 - 2020 Syncfusion Inc.

- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Custom styles

The style of the Tab widget is customized by 'CssClass' property.

The following code example is used to render the Tab widget with customized style.

1. Add the following code in your view page to render Tab with customized style.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with customized style.
<div>
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("dish").Text("dish Type")
.ContentTemplate(@<div>dish cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwichType").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).CssClass("custom").Render();}
</div>
```

2. Add the following styles

CSS

```
<style type="text/css">
.custom
{
width:650px;
}
</style>
```

3. The following screenshot illustrates the Tab with customized style.

Copyright © 2001 - 2020 Syncfusion Inc.



Tab with customized style

Integration with other widgets

You can provide more customization to the Tab with Rating control as content in it for describing the item rating value.

The Essential ASP.NET MVC Rating control provide you an intuitive rating experience that allows you to select the number of stars that represents the rating. For more information about the rating, you can refer the following link:

http://help.syncfusion.com/aspnetmvc/tab/integration-with-other-widgets

The following code example explains you the Rating control creation. The input element is used to create the rating control. Render the input element as rating control using the input element id.

1. Add the following code in your view page to render Tab with other widget (rating).

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with other widgets.
<div style="width:550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("pizza").Text("Pizza Type")
.ContentTemplate(@<div>Rating:@Html.EJ().Rating("PRating").Value(4)
Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Rating:@Html.EJ().Rating("SRating").Value(4)
Sandwich cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.
</div>);
}).Render();}
</div>
```

To render the rating controls in the first Tab element refer the styles mentioned in the following code example.

1. Add the following styles to render Tab.

CSS

```
<style type="text/css" class="cssStyles">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.dishRating
{
position: absolute;
margin: -31px 0px 0px 80px;
}
</style>
```

2. The following screenshot illustrates the Tab content with rating control.



Tab content section with Rating Control

AJAX Content Load (Load on Demand)

You can change the contents in sub Tab element periodically and you are provided with a support to change the contents without any problems. To achieve the content load, use the Load on Demand concept.

In Load On-Demand, the external HTML file with the necessary details is referred in URL section during Tab header declaration section. Also include “DataType”, “ContentType”, and “Async” in the script like the following example when rendering the control. When you click the Tab header, the AJAX automatically calls the content from the external files and displays in a Tab content section.

Sub Tab with AJAX Content

Each item has a variety of options and these options are also specified in the limited space. So you can choose the Tab control that is used within the root Tab to specify more details.

1. Add the following code in your view page to render sub Tab with AJAX content.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render sub Tab with AJAX content.
@{Html.EJ().Tab("DishType").Items(data =>
{
data.Add().ID("Pizza").Text("Pizza Menu").ContentTemplate(@<div>
Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
@firstTab()
</div>);
data.Add().ID("sandwich").Text("Sandwizza Menu").ContentTemplate(@<div>
Sandwizza cooked to perfection tossed with bread, milk, vegetables,
potatoes, poultry, 100% pure mutton, and cheese - and in creating
nutritious and tasty meals to maintain good health.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@secondTab()
</div>);
data.Add().ID("Pasta").Text("Pasta Menu").ContentTemplate(@<div>
Pasta cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
</div>);
}).Render();
@helper firstTab()
{
@Html.EJ().Tab("Pizza").Items(data =>
{
data.Add().ID("Corn-Spinach").Text("Corn
Spinach").Url("Content/cornSpinach.html");
data.Add().ID("ChickenDelite").Text("Chicken-
Delite").Url("Content/chickenDelite.html");
})
}
@helper secondTab()
{
@Html.EJ().Tab("sandwich").Items(data =>
{
data.Add().ID("GardenVeggie").Text("Corn
Spinach").Url("Content/gardenVeggie.html");
data.Add().ID("ChickenTikka").Text("Chicken-
Delite").Url("Content/chickenTikka.html");
data.Add().ID("PaneerTikka").Text("PaneerTikka").Url("Content/paneerTikka
.html");
})
}
}
```

2. The file ‘cornSpinach.html’ content is as follows.

ASPX-CS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<div class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/corn-and-
spinach-05.png" alt="corn-spinach">
<div class="ingredients">
Rate     : $70<br/> Ingredients : cheese, sweet corn &#38; green
capsicums.
<br />
Description: Small pizza bases are topped with spinach and paneer and
fresh cream, a nice layer of mozzarella cheese. This is baked until the
cheese is all hot and gooey.                    </div>
</div>
</body>
</html>
```

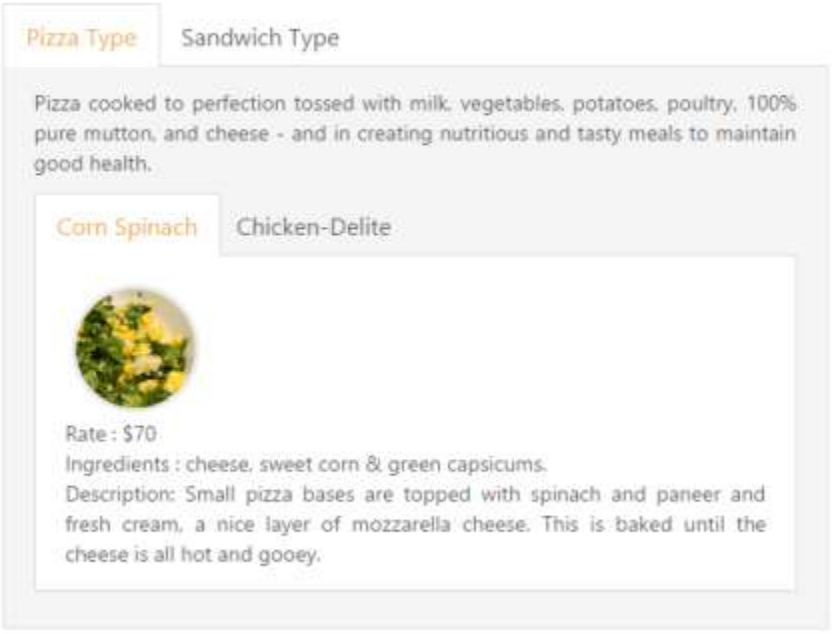
Copyright © 2001 - 2020 Syncfusion Inc.

3. The file ‘chickenDelite.html’ content is as follows.

ASPX-CS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
<div class="e-content">
<img src="http://js.syncfusion.com/demos/web/images/accordion/chicken-delite.png" alt="chicken-delite">
<div class="ingredients">
Rate     : $100<br /> Ingredients : cheese, chicken chunks, onions &#38; pineapple chunks.   <br />
Description: This is a tasty, elegant chicken dish that is easy to prepare.
</div>
</div>
</body>
</html>
```

The following screenshot illustrates the First Tab with the sub Tab control using Load on Demand.



Tab section with sub Tab control using Load on Demand.

RTL Support

Tab control provides support for load contents in right to left format. This is achieved by setting ‘EnableRTL’ property to “true”.

The following code example is used to render the Tab element in RTL format.

1. Add the following code in your view page to render Tab with RTL format.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab in Rtl format.
<div style="width:550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("pizza").Text("Pizza Type")
.ContentTemplate(@<div>
Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).EnableRTL(true).Render();}
</div>
```

2. The following screenshot illustrates the Tab with RTL format.



Keyboard Navigation

Tab control provides supports keyboard interaction. Using this functionality you can interact with control using keyboard. This is achieved by enabling 'AllowKeyboardNavigation'to 'true'. By default this property value is set to 'true'.

Following table illustrates the accessible key and their usage

Keys	Behavior
Up	Selected previous item.
Right	Selected previous item.
Down	Selected next item.
Left	Selected next item.
Home	Selected first item.
End	Selected last item.

The following code example is used to render the Tab element in RTL format.

Copyright © 2001 - 2020 Syncfusion Inc.

1. Add the following code in your view page to render Tab with keyboard navigation.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render Tab with keyboard navigation.
<div style="width:550px">
@{Html.EJ().Tab("dishTab").Items(data =>
{
data.Add().ID("pizzaType").Text("Pizza Type")
.ContentTemplate(@<div>
Pizza cooked to perfection tossed with milk, vegetables, potatoes,
poultry, 100% pure mutton, and cheese - and in creating nutritious and
tasty meals to maintain good health.
</div>);
data.Add().ID("sandwichType").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.
</div>);
}).AllowKeyboardNavigation(true).Render();}</div>
// Add the following script to render Tab with keyboard navigation.
<script type="text/javascript">
$(function () {
//Control focus key
$(document).on("keydown", function (e) {
if (e.altKey && e.keyCode === 74) {
// j- key code.
$("#dishTab ul a").focus();
}
});
});
</script>
```

2. The following screenshot illustrates the Tab with keyboard navigation.



Scroll Support

Tab control provides you scrolling support on Tab items to display a larger number of tabs with scroll buttons to get rid of the extending page size. The enabled Scroll buttons can be used to traverse through the elements.

By default, Tab header is rendered without scroll button. You can add the scroll button by setting the "EnableTabScroll" property to "true". When you move the cursor over the Tab header the scroll button is displayed.

You can use the following code example to render the Tab widget with scroll button.

Copyright © 2001 - 2020 Syncfusion Inc.

1. Add the following code in your view page to create a simple Tab with scroll button.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render Tab with scroll button.*@
<div style="width: 550px">
@{Html.EJ().Tab("dish").Items(data =>
{
data.Add().ID("pizza").Text("Pizza Type")
.ContentTemplate(@<div>Pizza cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("pasta").Text("Pasta Type")
.ContentTemplate(@<div>Pasta cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("burger").Text("Burger Type")
.ContentTemplate(@<div>Burger cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("sandwich").Text("Sandwich Type")
.ContentTemplate(@<div>Sandwich cooked to perfection tossed with bread,
milk, vegetables, potatoes, poultry, 100% pure mutton, and cheese - and
in creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("spaghetti").Text("Spaghetti Type")
.ContentTemplate(@<div>Spaghetti cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
data.Add().ID("ramen").Text("Ramen Type")
.ContentTemplate(@<div>Ramen cooked to perfection tossed with milk,
vegetables, potatoes, poultry, 100% pure mutton, and cheese - and in
creating nutritious and tasty meals to maintain good health.</div>);
}).EnableTabScroll(true).Render();}
</div>
```

The following screenshot illustrates you the Tab control with scroll button.



Tab control with scroll support

Template Support

ASP.NET MVC

The Content template option provided in MVC is used to specify the HTML elements inside the Tab control. We can use this option to load any HTML elements and showcase it in the Tab panels as per our requirement.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code block showcases how to use content template option in the Tab control.

ASPX-CS

```
<div style="width:650px">
@{Html.EJ().Tab("pizza").Items(data =>
{
data.Add().ID("garden").Text("GARDEN FRESH (Veg)")
.ContentTemplate(@<div>
<img src="~/Content/accordion/garden-veggie.png" alt="garden-fresh" />
<div class="ingredients">
Rate     : $50
<br />
Ingredients : cheese, onions, green capsicums & tomatoes.
</div>
</div>);
data.Add().ID("corn").Text("CORN & SPINACH (Veg)")
.ContentTemplate(@<div>
<img src="~/Content/accordion/corn-and-spinach-05.png" alt="garden-fresh"
/>
<div class="ingredients">
Rate     : $70
<br />
Ingredients : cheese, sweet corn & green capsicums.
</div>
</div>);
data.Add().ID("chicken").Text("CHICKEN DELITE (Non-veg)")
.ContentTemplate(@<div>
<img src="~/Content/accordion/chicken-delite.png" alt="garden-fresh" />
<div class="ingredients">
Rate     : $100
<br />
Ingredients : cheese, chicken chunks, onions & pineapple chunks.
</div>
</div>);
}).Render();}
</div>
```

Output:



Tab control with template support

Render Tab from code behind

Tab can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of Tab using Tab properties from code behind.

RAZOR

```
@{
Html.EJ().Tab("Tab",
(Syncfusion.JavaScript.Models.TabProperties)ViewData["tab"]).Render();
}
```

C#

```
public ActionResult Index()
{
TabProperties tab = new TabProperties();
tab.Items.Add(new TabBaseItem() { ID = "javaScript", Text = "JavaScript",
ContentTemplate = new MvcTemplate<TabBaseItem> { RazorViewTemplate =
(data) => { return "JavaScript (JS) is an interpreted computer
programming language"; } } });
tab.Items.Add(new TabBaseItem() { ID = "cSharp", Text = "C Sharp",
ContentTemplate = new MvcTemplate<TabBaseItem> { RazorViewTemplate =
(data) => { return " C# is intended to be a simple, modern, general-
purpose, object-oriented programming language."; } } });
tab.Items.Add(new TabBaseItem() { ID = "vb", Text = "VB.Net",
ContentTemplate = new MvcTemplate<TabBaseItem> { RazorViewTemplate =
(data) => { return "  The command-line compiler, VBC.EXE, is installed as
part of the Freeware .NET Framework SDK."; } } });
ViewData["tab"] = tab;
return View();
}
```

How To

Add a tab item dynamically based on condition

To add tab item dynamically based on condition, we can use the conditional operators to check and add the items within the items property. Refer to the following code.

HTML

```
@{Html.EJ().Tab("defaultTabs").Items(data =>
{
data.Add().ID("reviews").Text("Reviews").ContentTemplate(@<div>@{
Html.RenderPartial("_ReviewList"); }</div>);
if (user == "admin") { // check the condition to load the item in tab
data.Add().ID("calendar").Text("Calendar").ContentTemplate(@<div> @{
Html.RenderPartial("_Calendar"); }</div>);
}
}).ShowRoundedCorner(true).Render();}
```

In the above code, based on the “user” the second tab will be displayed.

Sample

Copyright © 2001 - 2020 Syncfusion Inc.

TagCloud

TagCloud

The ASP.NET MVC TagCloud control allows the user to display a list of links or tags with a structured cloud format where the importance of the tags can differentiate with varied font sizes, colors, and styles.

Key Features

- Data binding: Supports data binding with JSON data as well as remote data.
- Accessible: Supports adding or removing tags dynamically.
- Format: Supports tag structure in both cloud format and list format.
- Font-size: Supports limiting the font size with a minimum and maximum range.
- RTL: Supports changing the TagCloud direction for right-to-left alignment.
- Themes: JavaScript controls include 12 built-in themes (6 flat and 6 gradient effects) and also support a custom skin option to set user defined themes.

Getting Started

This section explains briefly about how to create a TagCloud in your ASP.NET MVC application.

Create your first TagCloud in MVC

ASP.NET MVC TagCloud provides support to display a weighted list, where the weight of each item is reflected by the size of the item's text. TagCloud is rendered as a link and as you click it, you can drill into the selected category. Refer the following guidelines to create TagCloud scenario. This allows you to display a weighted list.



In the above screenshot, you can see the weighted list and a click on any item drills into the respective category.

Create a TagCloud

Essential Studio ASP.NET MVC TagCloud widget has built-in features such as clicking on one of the sites in the TagCloud brings all relevant results only from the selected source. You can easily create the TagCloud widget element as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

1. You can create a MVC Project and add required assemblies, scripts and styles to it. Refer MVC-Getting Started.
2. Add the following code to the corresponding view page to render TagCloud.

ASPX-CS

```
@Html.EJ().TagCloud("tagEvents").Datasource((IEnumerable<MvcApplication.Models.WebsiteCollection>)ViewBag.datasource).TagCloudFields(tag => tag.Text("text").Url("url").Frequency("frequency")).Title("Tech Sites")
```

3. Add the following style to show the weighted list.

CSS

```
<style type="text/css">
#tagEvents
{
margin: 0 auto;
}
</style>
```

Set the Min and Max Font Size

Create the class for WebsiteCollection and define the necessary data members. Add the following code in WebsiteCollection.cs part.

C#

```
// Define local data source elements with  fields
public class WebsiteCollection
{
public string text { get; set; }
public string url { get; set; }
public int frequency { get; set; }
}
```

You can set the minimum and maximum font size in frequency property by adding the following code example in the Controller.

C#

```
//Refer the Model in the controller
using <Applicationname>.Models;
public ActionResult Index()
{
List<WebsiteCollection> sites = new List<WebsiteCollection>();
sites.Add(new WebsiteCollection { text = "Google", url =
"http://tech.firstpost.com/tag/google", frequency = 12 });
sites.Add(new WebsiteCollection { text = "Apple", url =
"http://tech.firstpost.com/tag/apple-iwork", frequency = 3 });
sites.Add(new WebsiteCollection { text = " Drone ", url =
"http://tech.firstpost.com/tag/drone", frequency = 8 });
sites.Add(new WebsiteCollection { text = "google Drone", url =
"http://tech.firstpost.com/tag/google-drones/", frequency = 2 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
sites.Add(new WebsiteCollection { text = "apple iwork", url =
"http://tech.firstpost.com/tag/apple-iwork", frequency = 12 });
sites.Add(new WebsiteCollection { text = "tech-buzz", url =
"http://tech.firstpost.com/tag/tech-buzz", frequency = 5 });
sites.Add(new WebsiteCollection { text = "netizens", url =
"http://tech.firstpost.com/tag/netizens", frequency = 8 });
sites.Add(new WebsiteCollection { text = "selfile", url =
"http://tech.firstpost.com/tag/selfie", frequency = 20 });
sites.Add(new WebsiteCollection { text = "globalselfile", url =
"http://tech.firstpost.com/tag/nasa-globalselfie", frequency = 1 });
sites.Add(new WebsiteCollection { text = "extreme", url =
"http://www.extremetech.com/", frequency = 3 });
sites.Add(new WebsiteCollection { text = "at-t move", url =
"http://www.extremetech.com/extreme/182815-att-moves-to-acquire-directv-
to-defend-against-comcast-everyone-loses", frequency = 5 });
sites.Add(new WebsiteCollection { text = "Gearlog", url =
"http://www.gearlog.com/", frequency = 9 });
sites.Add(new WebsiteCollection { text = "Information Week", url =
"http://www.informationweek.com/", frequency = 0 });
sites.Add(new WebsiteCollection { text = "PCWorld", url =
"http://www.pcworld.com/", frequency = 11 });
sites.Add(new WebsiteCollection { text = "Tech Republic", url =
"http://techrepublic.com/", frequency = 3 });
sites.Add(new WebsiteCollection { text = "Valleywag", url =
"http://valleywag.gawker.com/", frequency = 6 });
sites.Add(new WebsiteCollection { text = "computing", url =
"http://www.extremetech.com/category/computing", frequency = 9 });
sites.Add(new WebsiteCollection { text = "WebProNews", url =
"http://www.webpronews.com/", frequency = 2 });
ViewBag.datasource = sites;
return View();
}
```



Set an event to perform operation

You can perform the event operations like MouseOver, MouseOut and Click by adding the following code example inside View page.

ASPX-CS

```
@Html.EJ().TagCloud("tagEvents").Datasource((IEnumerable<MvcApplication.M
odels.WebsiteCollection>)ViewBag.datasource).TagCloudFields(tag =>
```

```
tag.Text("text").Url("url").Frequency("frequency")).Title("Tech
sites").ClientSideEvents(event =>
event.Create("create").MouseOver("mouseover").MouseOut("mouseout").Click(
"click"))
<div id="tagCloudTarget">
<ul>
<li>mouseover</li>
<li>mouseout</li>
<li>click</li>
</ul>
</div>
@Html.EJ().DropDownList("selectControls_input").TargetID("tagCloudTarget"
).ShowCheckbox(true).CheckAll(true).ClientSideEvents(event =>
event.Change("event"))
<style type="text/css">
#tagEvents
{
margin: 0 auto;
}
</style>
<script>
function event(args)
{
tagObj = $("#tagEvents").data("ejTagCloud");
if (args.isChecked)
{
switch (args.selectedText)
{
case "mouseover": tagObj.option(args.selectedText, "mouseover"); break;
case "mouseout": tagObj.option(args.selectedText, "mouseout"); break;
case "click": tagObj.option(args.selectedText, "click"); break;
}
}
else tagObj.option(args.selectedText, null);
}
function create(args)
{
alert("Tagcloud has been created.");
}
function mouseover(args)
{
alert("Mouse pointer is hovered on " + args.value);
}
function mouseout(args)
{
alert("Mouse pointer is hovered away from " + args.value);
}
function click(args)
{
alert( args.value + " is clicked.");
}
</script>
```

When you move the mouse to latest technology (weighted) list, it is denoted in Alert box. You can execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Data-Binding

To render the TagCloud widget, it is necessary to bind the data to it in a proper way. The following sub-properties provides you a way to bind local or remote data to the TagCloud widget by binding the appropriate data fields to the corresponding options.

Fields

dataSource

The Datasource property receives Essential DataManager object and JSON object. Then assign the local JSON data or remote (URL binding) data to the TagCloud control.

Query

This property receive query to retrieve data from the table (query is same as SQL).This property is applicable only when a remote Datasource is used. Each data value is used to render an item for the TagCloud. The value set to this property is object type.

Text

It maps the corresponding text field name from the data table or JSON data that is assigned to the Datasource with the Text property of the TagCloud control. The Text value that is fetched from the table renders the value to be displayed in the TagCloud.

URL

URL field in the data table or JSON data assigned the Datasource is mapped to the URL property of the TagCloud control. The URL property defines the link to be navigated on clicking the corresponding text item.

Frequency

It maps to the Frequency field name from the data table or JSON data that is assigned to the Datasource. The frequency value that is fetched from the table should be a number to categorize the font size.

Local Binding

Local data binding allows you to map JSON data to TagCloud, that the corresponding Text, URL, and Frequency fields are assigned with a local JSON data.

Defining the Local data for TagCloud

The following steps explains you the local data binding to TagCloud widget,

C#

Copyright © 2001 - 2020 Syncfusion Inc.

```
// Add the following data list to be bind in the controller page and
define the corresponding data.
// Define local data source elements with  fields
public class WebsiteCollection
{
//TagCloud data source should have Text and Url as Mandatory
public string Text { get; set; }
public string Url { get; set; }
public int Frequency { get; set; }
}
//Refer the Model in the controller using <Applicationname>.Models;
List<WebsiteCollection> sites = new List<WebsiteCollection>();
public ActionResult Index()
{
sites.Add(new WebsiteCollection
{
Text = "Google", Url = "http://www.google.com", Frequency = 12 });
sites.Add(new WebsiteCollection { Text = "All Things Digital", Url =
"http://allthingsd.com/", Frequency = 3 });
sites.Add(new WebsiteCollection { Text = "Arts Technica", Url =
"http://arstechnica.com/", Frequency = 8 });
sites.Add(new WebsiteCollection { Text = "Business Week", Url =
"http://www.businessweek.com/", Frequency = 2 });
sites.Add(new WebsiteCollection { Text = "Yahoo", Url =
"http://in.yahoo.com/", Frequency = 12 });
sites.Add(new WebsiteCollection { Text = "Center Networks", Url =
"http://www.centernetworks.com/", Frequency = 5 });
sites.Add(new WebsiteCollection { Text = "Crave", Url =
"http://news.cnet.com/crave/", Frequency = 8 });
sites.Add(new WebsiteCollection { Text = "Crunch Gear", Url =
"http://techcrunch.com/gadgets/", Frequency = 20 });
sites.Add(new WebsiteCollection { Text = "Daily Tech", Url =
"http://www.dailytech.com/", Frequency = 1 });
sites.Add(new WebsiteCollection { Text = "Electronista", Url =
"http://www.electronista.com/", Frequency = 3 });
sites.Add(new WebsiteCollection { Text = "Engadget", Url =
"http://www.engadget.com/", Frequency = 5 });
sites.Add(new WebsiteCollection { Text = "Gearlog", Url =
"http://www.gearlog.com/", Frequency = 9 });
sites.Add(new WebsiteCollection { Text = "Information Week", Url =
"http://www.informationweek.com/", Frequency = 0 });
sites.Add(new WebsiteCollection { Text = "PCWorld", Url =
"http://www.pcworld.com/", Frequency = 11 });
sites.Add(new WebsiteCollection { Text = "Tech Republic", Url =
"http://techrepublic.com/", Frequency = 3 });
sites.Add(new WebsiteCollection { Text = "Valleywag", Url =
"http://valleywag.gawker.com/", Frequency = 6 });
sites.Add(new WebsiteCollection { Text = "Rediff", Url =
"http://in.rediff.com/", Frequency = 9 });
sites.Add(new WebsiteCollection { Text = "WebProNews", Url =
"http://www.webpronews.com/", Frequency = 2 });
ViewBag.datasource = sites;
return View();
}
```

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
// Map Local datasource to corresponding fields in TagCloud control.
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource)
.TagCloudFields(tag =>
tag.Text("Text").Url("Url").Frequency("Frequency"))
```

The following screenshot displays the TagCloud control with local data binding.



Remote Binding

TagCloud provides remote data binding support to populate TagCloud items and the values can be mapped to the TagCloud fields from a remote web service by using DataManager and Query.

DataManager is used to manage relational data in ASP.NET MVC. It supports CRUD (Create, Read, Update, and Destroy) in individual requests and batch. DataManager use two different classes, ej.DataManager for processing, and ej.Query for serving data. ej.DataManager communicates with data source and ej.Query generates data queries that are read by DataManager.

Configuring Remote data for TagCloud

The following script explains you the Remote data binding to TagCloud widget.

ASPX-CS

```
<%-- Configure datasource referring remote data binding section and
assign it to datasource property -- %>
@Html.EJ().TagCloud("tagcloud").Datasource(d =>
d.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/").Offline(false))
.Query("ej.Query().from('Orders').take(10)").TagCloudFields(tag =>
tag.Text("CustomerID").Frequency("EmployeeID")).Title("Tagcloud Data")
```

The following screenshot displays a TagCloud control with remote data binding.

Copyright © 2001 - 2020 Syncfusion Inc.



Title Customization

ShowTitle

The TagCloud items are displayed with a Title element by default. ShowTitle is a Boolean type, which allow us to Hide or Show the Title. This property is set true by default.

How to disable title in TagCloud

The following step explains you on how to configure title for a TagCloud.

ASPX-CS

```
<%-- Configure datasource referring local data binding section and assign
it to datasource property -- %>
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource)
.TagCloudFields(tag => tag.Text("Text").Url("Url")
.Frequency("Frequency"))
.ShowTitle(false)
```

The following screenshot illustrates a TagCloud control when you set false to ShowTitle,



Title

TagCloud widget allows us to set a custom Title text by using the Title property. By default Title property is set to string value “Title”.

Defining title text for TagCloud

The following step explain to us on how to configure Title for a TagCloud.

ASPX-CS

```
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.TagCloudFields(tag => tag.Text("Text").Url("Url")
.Frequency("Frequency"))
.Title("Tech sites")
```

The following screenshot illustrates the TagCloud control with customized title text.



TitleImage

TagCloud widget provides TitleImage to set an image for the title. You can set the desired image URL to TitleImage property.

Defining Title Image for TagCloud

The following step explains us to configure TitleImage for a TagCloud.

ASPX-CS

```
<%-- Configure datasource referring local data binding section and assign
it to datasource property -- %>
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource)
.TagCloudFields(tag => tag.Text("Text").Url("Url")
.Frequency("Frequency")).Title("Tech sites")
.TitleImage("http://js.syncfusion.com/demos/web/images/waitingpopup/js_lo
go.png")
```

1. Using CSS class you can resize the Title image content as follows.

CSS

```
<style type="text/css">
.e-title-img
{
height::35px;
width:35px;
}
</style>
```

The following screenshot illustrates the TagCloud control with customized title image.

Copyright © 2001 - 2020 Syncfusion Inc.



Appearance and Styling

Minimum and maximum Font size

The TagCloud content are set to different font sizes from minimum to maximum based on its frequency values. By default, MinFontSize is “10px” and MaxFontSize is “40px”, using these properties you can customize the minimum and maximum font sizes.

Customizing font sizes of TagCloud

The following steps explains you on how to configure font sizes for a TagCloud.

ASPX-CS

```
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource).TagCloudFields(tag =>
tag.Text("Text").Url("Url").Frequency("Frequency")).Title("Tech
sites").MinFontSize("20px").MaxFontSize("50px")
```

The following screenshot illustrates the TagCloud control with customized font sizes.

Copyright © 2001 - 2020 Syncfusion Inc.



Display Formats

You can set the TagCloud content display format using Format property. The Format has two variant. By default Format is set to Cloud, which displays content in TagCloud. Another one format is as List to display the content in linear format.

Defining Cloud and List format

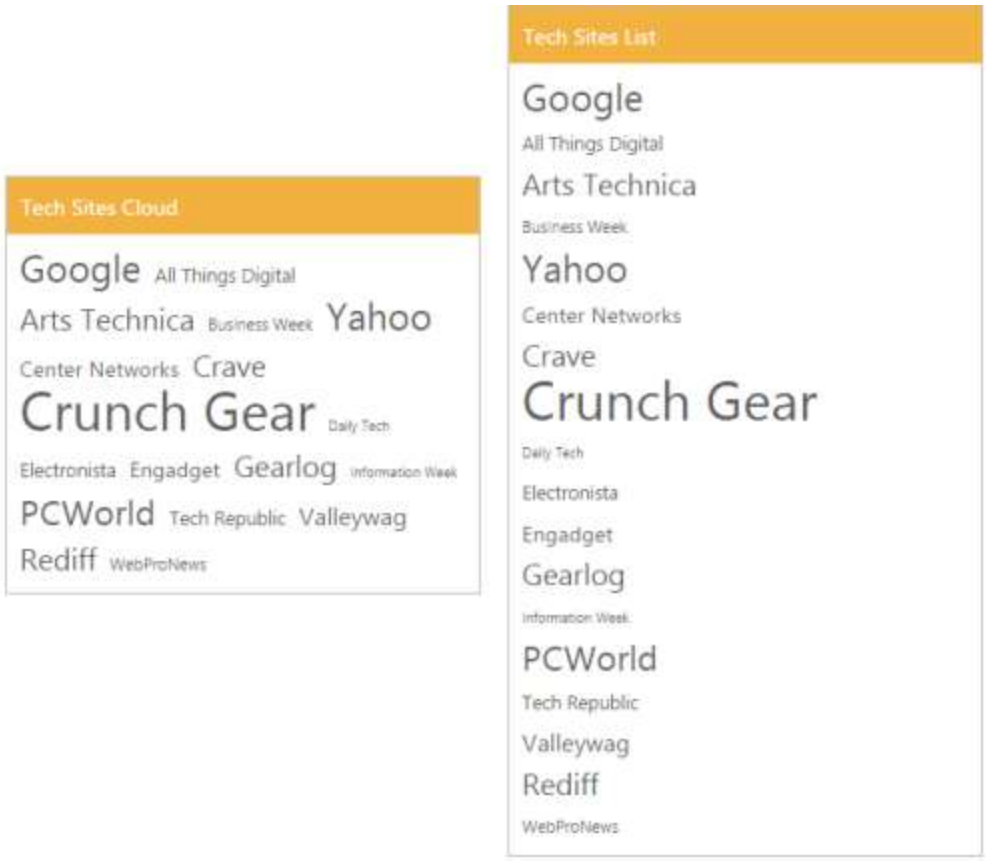
The following steps explains you to configure format for a TagCloud.

ASPX-CS

```
<%-- Configure datasource referring local data binding section and assign
it to datasource property -- %>
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource).TagCloudFields(tag =>
tag.Text("Text").Url("Url").Frequency("Frequency")).Title("Tech
sites").Format(Formats.Cloud)
@Html.EJ().TagCloud("tagCloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource).TagCloudFields(tag =>
tag.Text("Text").Url("Url").Frequency("Frequency")).Title("Tech
sites").Format(Formats.List)
```

The following screenshot illustrates the TagCloud control with customized formats.

Copyright © 2001 - 2020 Syncfusion Inc.



Theme

You can control the style and appearance of TagCloud based on CSS classes. To apply styles to the TagCloud control, you can refer two files, ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.widgets.all.min.css file, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for TagCloud control namely,

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Copyright © 2001 - 2020 Syncfusion Inc.

CssClass

You can use the CssClass to customize the TagCloud appearance. Any of the CssClass properties can be used to modify look and feel of tag cloud based on the requirement. Define a CSS class as per requirement and assign the class name to CssClass property.

Configure TagCloud using CSS class

The following steps allows you to configure CssClass for TagCloud.

1. In the View page, add the below script to configure TagCloud widget.

ASPX-CS

```
<%-- Configure datasource referring local data binding section and assign
it to datasource property -- %>
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource)
.TagCloudFields(tag => tag.Text("Text").Url("Url")
.Frequency("Frequency")).Title("Tech sites")
.CssClass("CustomCss")
```

2. Define CSS class for customizing the TagCloud widget.

CSS

```
<style type="text/css" class="cssStyles">
/* Customize the TagCloud div element */
.CustomCss
{
background-color: #DDC;
width: 400px;
}
/* Customize the TagCloud header element */
.CustomCss .e-header.e-title
{
text-align: center;
font-weight: bold;
}
</style>
```

The following screenshot illustrates the TagCloud with customized CSS class,



RTL Support

The EnableRTL feature supports you to change the left-to-right alignment of the TagCloud widget to right-to-left (RTL). This displays the content from right-to-left in the widget. You can achieve this using EnableRTL property that is set false by default.

Enabling RTL Support

The following steps explains you the EnableRTL property in TagCloud.

ASPX-CS

```
<%-- Configure datasource referring local data binding section and assign
it to datasource property -- %>
@Html.EJ().TagCloud("tagcloud").Datasource((IEnumerable<WebsiteCollection
>)ViewBag.datasource)
.TagCloudFields(tag => tag.Text("Text").Url("Url")
.Frequency("Frequency"))
.Title("Tech sites")
.EnableRTL(true)
```

The following screenshot illustrates the TagCloud control with RTL support.



Render TagCloud from Server Side

TagCloud can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of TagCloud using TagCloud properties from code behind.

RAZOR

```
@{Html.EJ().TagCloud("tagEvents", Model).Render(); }
```

C#

```
public ActionResult Index()
{
List<WebsiteCollection> sites = new List<WebsiteCollection>();
sites.Add(new WebsiteCollection { text = "Google", url =
"http://tech.firstpost.com/tag/google", frequency = 12 });
sites.Add(new WebsiteCollection { text = "Apple", url =
"http://tech.firstpost.com/tag/apple-iwork", frequency = 3 });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
sites.Add(new WebsiteCollection { text = " Drone ", url =
"http://tech.firstpost.com/tag/drone", frequency = 8 });
sites.Add(new WebsiteCollection { text = "google Drone", url =
"http://tech.firstpost.com/tag/google-drones/", frequency = 2 });
sites.Add(new WebsiteCollection { text = "apple iwork", url =
"http://tech.firstpost.com/tag/apple-iwork", frequency = 12 });
sites.Add(new WebsiteCollection { text = "tech-buzz", url =
"http://tech.firstpost.com/tag/tech-buzz", frequency = 5 });
sites.Add(new WebsiteCollection { text = "netizens", url =
"http://tech.firstpost.com/tag/netizens", frequency = 8 });
sites.Add(new WebsiteCollection { text = "selfile", url =
"http://tech.firstpost.com/tag/selfie", frequency = 20 });
sites.Add(new WebsiteCollection { text = "globalselfile", url =
"http://tech.firstpost.com/tag/nasa-globalselfie", frequency = 1 });
sites.Add(new WebsiteCollection { text = "extreme", url =
"http://www.extremetech.com/", frequency = 3 });
sites.Add(new WebsiteCollection { text = "at-t move", url =
"http://www.extremetech.com/extreme/182815-att-moves-to-acquire-directv-
to-defend-against-comcast-everyone-loses", frequency = 5 });
sites.Add(new WebsiteCollection { text = "Gearlog", url =
"http://www.gearlog.com/", frequency = 9 });
sites.Add(new WebsiteCollection { text = "Information Week", url =
"http://www.informationweek.com/", frequency = 0 });
sites.Add(new WebsiteCollection { text = "PCWorld", url =
"http://www.pcworld.com/", frequency = 11 });
sites.Add(new WebsiteCollection { text = "Tech Republic", url =
"http://techrepublic.com/", frequency = 3 });
sites.Add(new WebsiteCollection { text = "Valleywag", url =
"http://valleywag.gawker.com/", frequency = 6 });
sites.Add(new WebsiteCollection { text = "computing", url =
"http://www.extremetech.com/category/computing", frequency = 9 });
sites.Add(new WebsiteCollection { text = "WebProNews", url =
"http://www.webpronews.com/", frequency = 2 });
TagCloudProperties tagObj = new TagCloudProperties();
tagObj.DataSource = sites;
tagObj.Title = "Tech Sites";
tagObj.TagCloudFields.Text = "text";
tagObj.TagCloudFields.Url = "url";
return View(tagObj);
}
```

Tile

Tile

The Essential Studio for ASP.NET MVC Web Tiles are simple, opaque rectangles or squares and they are arrayed on the start screen in a grid-like pattern. Tapping or selecting a Tile, launches the app or does some other action that is represented by the Tile. Tiles are arranged in a group separated by columns that looks like a start screen of a device and it can be either static or live.

Getting Started

This section explains briefly on how to create a Tile control in your application.

Copyright © 2001 - 2020 Syncfusion Inc.

Create your first Tile in MVC

The Essential Studio for ASP.NET MVC Web Tiles are simple, opaque rectangles or squares that are arrayed on the Start screen in a grid-like pattern and it can be either static or live. Tapping or selecting a Tile launches the app or other experience that is represented by the Tile.



Create Tile Widget

The following steps guide you to add group of Tiles for creating a home page that displays all the available applications.

1. You can create an MVC Project and add necessary assemblies, styles and scripts to it. Refer MVC-Getting Started.
2. To add a Tile control, call Tile helper. You can specify the tile name, size and image URL as follows.
3. Add the following code example to the corresponding view page.

ASPX-CS

```
@Html.EJ().Tile("tile1")
.TileSize(TileSize.Medium)
.ImageUrl("http://js.syncfusion.com/ug/web/content/tile/map.png")
.Text("Maps")
```

4. Run the application to render the following output.



In this scenario, a home page is designed using tile for easy navigation. Therefore, you require many different sizes of tiles aligned in a grid-like manner. To align the tiles automatically, define the necessary tile elements inside the wrapper element that contains a ‘column’ class. You can define all columns elements under the wrapper element with ‘group’ class to make ‘n’ number of tiles as a grouped tile.

Refer to the following code example.

ASPX-CS

```
<div id="tile" style="margin-top: 45px;">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div class="e-tile-group">
<div class="e-tile-column">
@Html.EJ().Tile("tile1").ImagePosition(TileImagePosition.Fill).TileSize(T
ileSize.Medium).ImageUrl("http://js.syncfusion.com/ug/
web/content/tile/people_1.png").Text("People")
<div class="e-tile-small-col-2">
@Html.EJ().Tile("tile2").ImagePosition(TileImagePosition.Center).TileSize
(TileSize.Small).ImageUrl("http://js.syncfusion.com/ug/web/content/tile/a
lerts.png")
@Html.EJ().Tile("tile3").ImagePosition(TileImagePosition.Center).TileSize
(TileSize.Small).ImageUrl("http://js.syncfusion.com/ug/web/content/tile/b
ing.png")
@Html.EJ().Tile("tile4").ImagePosition(TileImagePosition.Center).TileSize
(TileSize.Small).ImageUrl("http://js.syncfusion.com/ug/web/content/tile/c
amera.png")
@Html.EJ().Tile("tile5").ImagePosition(TileImagePosition.Center).TileSize
(TileSize.Small).ImageUrl("http://js.syncfusion.com/ug/
web/content/tile/messages.png")
</div>
@Html.EJ().Tile("tile6").ImagePosition(TileImagePosition.Center).TileSize
(TileSize.Medium).ImageUrl("http://js.syncfusion.com/ug/web/content/tile/
games.png").Text("Play")
@Html.EJ().Tile("tile7").TileSize(TileSize.Medium).ImageUrl("http://js.sy
ncfusion.com/ug/web/content/tile/map.png").Text("Maps")
</div>
<div class="e-tile-column">
@Html.EJ().Tile("tile9").ImagePosition(TileImagePosition.Fill).TileSize(T
ileSize.Medium).ImageUrl("http://js.syncfusion.com/ug/web/content/tile/pe
ople_2.png").Text("People")
@Html.EJ().Tile("tile10").ImagePosition(TileImagePosition.Center).TileSiz
e(TileSize.Medium).ImageUrl("http://js.syncfusion.com/ug/web/content/tile
/pictures.png").Text("Photo")
@Html.EJ().Tile("tile11").ImagePosition(TileImagePosition.Center).TileSiz
e(TileSize.Wide).ImageUrl("http://js.syncfusion.com/ug/web/content/tile/w
eather.png").Text("Weather")
</div>
</div>
</div>
```

Run the application to render the following output.



Tile dependencies

This section deals with the dependency files for Tile control. The file ej.web.all.min.js is a mass collection of Web Controls. When you are using this file, you can skip this section. Otherwise, refer to the following mentioned script files to render the Tile control.

Copyright © 2001 - 2020 Syncfusion Inc.

File	Description/Usage
ej.core.min.js	Must be referred always before using all the Web controls.
ej.data.min.js	Must be referred while using data binding feature in Web controls that is used to handle data manger operation.
ej.touch.min.js	Must be referred for touch events supports.
ej.tile.min.js	Must be referred to render Tile control and to handle Tile specific operations.

Image Configuration

ImagePosition property is used to adjust the position of Tile image at the Center on initialization. The possible values for the ImagePosition are as follows

1. Center
2. TopCenter
3. BottomCenter
4. RightCenter
5. LeftCenter
6. TopLeft
7. TopRight
8. BottomRight
9. BottomLeft
10. Fill

ImageUrl property is used to set the background image for Tile, where the image is given in the path specified by ImageUrl property.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").Text("Weather").ImagePosition(TileImagePosition.C
enter).TileSize(TileSize.Wide).ImageUrl("http://js.syncfusion.com/UG/web/
Content/tile/Weather_2.png ")
```



You can give images for each tile through CSS classes by using ImageClass property. You can define your desired styles in the specified class.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").Text("Pictures").ImagePosition(TileImagePosition.
Center).TileSize(TileSize.Medium).ImageClass("pictures")
<style>
.pictures
{
background:
url("http://js.syncfusion.com/UG/web/Content/tile/pictures.png ");
background-size:30px 30px;
}
</style>
```



Text Configuration

ShowText property is used to show or hide the Tile caption. By default, the ShowText property is to true on initialization.Text property is used to set the caption of a Tile as a Text on initialization. TextAlignment property is used to align the Tile text as Normal on initialization. The possible position values for TextAlignment are as follows:

1. Normal
2. Left
3. Right
4. Center

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").Text("Camera").TextAlignment(TextAlignment.Center
).ImagePosition(TileImagePosition.Center).TileSize(TileSize.Medium).Image
Url("http://js.syncfusion.com/UG/web/Content/tile/camera.png")
```



Template Support

ImageTemplateId property is used to customize the image of Tile with template feature by setting the id. CaptionTemplateId property is used to customize the text of Tile with template feature by setting the id.

Copyright © 2001 - 2020 Syncfusion Inc.

Refer to the following code examples.

Add the following code example for MVC samples

CSS

```
<style>
#appImage
{
background-image:
url("http://js.syncfusion.com/UG/mobile/content/google.png");
background-position: center center;
background-repeat: no-repeat;
background-size: 50% auto;
display: table-cell;
width: 45%;
}
.tileMargin
{
display: table-cell;
padding-top: 25px;
}
.e-tile-template
{
display: table;
height: 100%;
width: 100%;
}
</style>
```

ASPX-CS

```
@Html.EJ().Tile("tile").ImageTemplateId("imageTemplate").ImagePosition(Ti
leImagePosition.Fill).TileSize(TileSize.Wide).Caption(caption => {
caption.Enabled(true); caption.Text("Windows Store"); })
<div id="imageTemplate">
<div id="appImage">
</div>
<div class="tileMargin">
<span class="caption">Google Search</span><br />
<span class="description">The world's information</span><br />
<span class="sub">Free</span>
</div>
</div>
```



Copyright © 2001 - 2020 Syncfusion Inc.

Configure Badge

The Badge property handles badge specific functionalities like enable or disable the badge and setting badge value for Tile.

Enabled property enables or disables the badge for a Tile. The Tile renders with hidden badge when it is set to false.

The Value property is used to set the badge value to a Tile. By default, the Value is set to 1 on initialization. The Text property is used to set the text instead of number for Tile badge. The MaxValue property is used to set the maximum badge value to a Tile. When you set the badge value greater than MaxValue, it shows maximum value in badge with plus symbol. The MinValue property is used to set the minimum badge value to a Tile. When you set the badge value less than MinValue, it shows minimum value in badge.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").ImageUrl("http://js.syncfusion.com/UG/Web/Content
/tile/messages.png").Badge(badge => {
badge.Enabled(true).MaxValue(80).MinValue(10).Value(88);
}).ImagePosition(TileImagePosition.Center).TileSize(TileSize.Medium)
```



LiveTile Configuration

Live Tiles are used to display the current or up to date information like scores, stocks, weather, etc. You can enable Live Tile using enabled property by setting it to true. Type property allows you to specify the type of animation while updating the information in Tile. There are three types of Tile animation supported: Flip, Slide and Carousel.

ImageUrl property sets background image for Live Tile. This property accepts array values so you can specify the image url's for all the Tiles that are used in single Live Tile.

You can specify time interval for each Tile update/animation using UpdateInterval property. Time interval is given in milliseconds. The default value is 2000.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").LiveTile(live => {
live.Enabled(true).Type(LiveTileType.Flip).UpdateInterval(2500).ImageUrl(
new string[] {
"http://js.syncfusion.com/UG/Web/Content/tile/people_1.png","http://js.sy
ncfusion.com/UG/Web/Content/tile/people_2.png"});
}).ImagePosition(TileImagePosition.Fill).TileSize(TileSize.Medium)
```

Copyright © 2001 - 2020 Syncfusion Inc.

In ImageTemplateId property, you can give Live Tile images outside the Tile rendering. To achieve this, you are required to give image content inside the element where the path is specified by template id. You can update the ImageTemplateId dynamically through updateTemplateID public method.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").LiveTile(live => {
live.Enabled(true).ImageTemplateId(new string[] { "temp1", "temp2" });
}).ImagePosition(TileImagePosition.Fill).TileSize(TileSize.Medium)
<div id="temp1" style="background-
image:url('http://js.syncfusion.com/ug/web/content/tile/people_1.png');
width:100%; height:100%;">
</div>
<div id="temp2" style="background-image:
url('http://js.syncfusion.com/ug/web/content/tile/people_2.png');
width:100%; height:100%;">
</div>
```

You can specify the array of images for Live Tile through CSS classes by using ImageClass property and you can define the desired styles in the specified class.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").LiveTile(live => {
live.Enabled(true).ImageClass(new string[] { "people1", "people2" });
}).ImagePosition(TileImagePosition.Fill).TileSize(TileSize.Medium)
<style>
.people1
{
background-image:
url('http://js.syncfusion.com/UG/Web/Content/tile/people_1.png');
}
.people2
{
background-image:
url('http://js.syncfusion.com/UG/Web/Content/tile/people_2.png');
}
</style>
```

Customize size

You can customize the size of the Tile by using TileSize property. The following built-in tile sizes are supported.

1. Medium
2. Small
3. Large
4. Wide

The default TileSize value is set to Small.

Copyright © 2001 - 2020 Syncfusion Inc.

Refer to the following code examples.

ASPX-CS

```
@Html.EJ().Tile("tile").Text("Pictures").ImagePosition(TileImagePosition.
Center).TileSize(TileSize.Medium).ImageUrl("http://js.syncfusion.com/UG/W
eb/Content/tile/pictures.png ")
```



Add Group Tiles

To make a Tile as grouped tile, you can use the following mentioned pre-defined classes.

Class Name	Explanation
e-tile-group	To group the column elements
e-tile-column	To align the tile in column manner
e-tile-small-col-2	To align the small size tiles

To render group tile, refer to the following code example.

ASPX-CS

```
<div class="group">
<div class="column">
<!– Add tile control here -->
</div>
</div>
```

To render column grouped tile, you need to render the number of tiles inside a element with class ‘column’. Then that column group element is appended to a with class ‘group’.

To render small-col-2 grouped tile, you need to render the number of tiles inside a element with class ‘small-col-2’. Then that small-col-2 group element is appended to a with class ‘column’. Then you need to append those column inside the main group element.

Refer the following code examples.

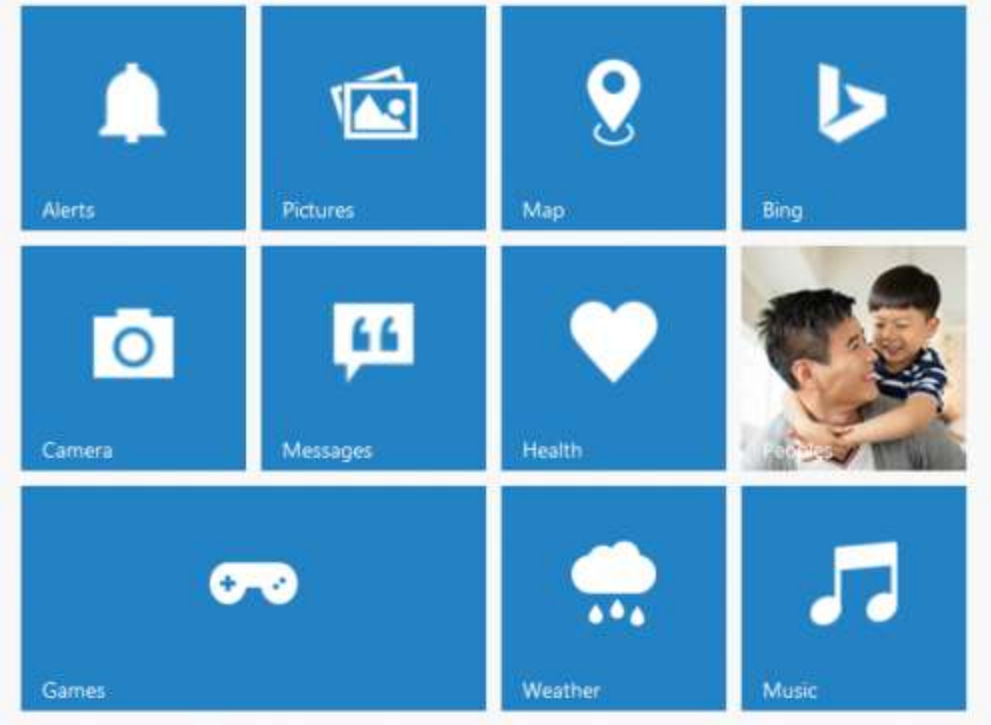
ASPX-CS

```
<div class="e-tile-group">
<div class="e-tile-column">
@Html.EJ().Tile("tile1").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/alerts.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Alert"))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Tile("tile2").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/pictures.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("pictures"))
@Html.EJ().Tile("tile3").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/camera.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Camera"))
@Html.EJ().Tile("tile4").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/messages.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Messages"))
@Html.EJ().Tile("tile5").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/games.png").TileSize(TileSize.Wide).Caption(caption =>
caption.Enabled(true).Text("Games"))
</div>
<div class="e-tile-column">
@Html.EJ().Tile("tile6").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/map.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Map"))
@Html.EJ().Tile("tile7").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/bing.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Bing"))
@Html.EJ().Tile("tile8").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/favs.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Health"))
@Html.EJ().Tile("tile9").ImageUrl("http://js.syncfusion.com/UG/Web/Conten
t/tile/people_2.png").TileSize(TileSize.Medium).ImagePosition(TileImagePo
sition.Fill).Caption(caption => caption.Enabled(true).Text("Peoples"))
@Html.EJ().Tile("tile10").ImageUrl("http://js.syncfusion.com/UG/Web/Conte
nt/tile/weather.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Weather"))
@Html.EJ().Tile("tile11").ImageUrl("http://js.syncfusion.com/UG/Web/Conte
nt/tile/music.png").TileSize(TileSize.Medium).Caption(caption =>
caption.Enabled(true).Text("Music"))
</div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Styles and Appearance

Styles and appearance of Tile works based on CSS class. You need two style sheets to apply styles to the Tile control: ej.widget.core.min.css and ej.theme.min.css.

Customize styles

To change the color or design of the Tile control appearance, you can override default styles of the Tile control by using its class names. The following are the specific class names to modify styles and appearance in the Tile.

File	Description/Usage
e-tile	It is the root class of the Tile.
e-child-tile	It class is applied to the child of the Tile.
e-enabled-text	It class is applied to the text of Tile when text element is active
e-tile-text	It class is applied to the text of Tile
e-tile-image	It class is applied to image of Tile
e-tile-template	It class is applied to tile image when tile has image template id property.
e-live-tile-text	It class is applied to text elements of individual live tile in the Tile control.
e-template-caption	This class is applied to tile caption when tile has caption template id property.

Copyright © 2001 - 2020 Syncfusion Inc.

How to

Render TileView from Code behind

TileView can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side

The following code illustrates the initialization of TileView properties in the controller.

C#

```
//Namespace to get the JavaScript (TileView) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class TileController: Controller
{
public ActionResult TileFeatures()
{
//Initializing the TileView model
TileProperties tileObj = new TileProperties();
//Initializing the other properties
tileObj.TileSize = Syncfusion.JavaScript.TileSize.Medium;
tileObj.ImageUrl =
"http://js.syncfusion.com/ug/web/content/tile/alerts.png";
tileObj.Text = "Alerts";
//Passing TileView properties using the ViewData
ViewData["TileModel"] = tileObj;
return View();
}
}
}
```

Binding the TileView properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().Tile("Tile1",
(Syncfusion.JavaScript.Models.TileProperties)ViewData["TileModel"]).Rende
r();
}
```

TimePicker

TimePicker

The Essential ASP.NET MVC TimePicker control allows users to select a time value. The available times can be restricted to a range by setting minimum and maximum time values. The TimePicker sets the time as a mask to prevent users from entering invalid values.

Key Features

- Time format: Supports all valid time formats.
- Localization: Supports localization with different cultures.
- Picker and editor: Allows users to pick or edit the time.

Copyright © 2001 - 2020 Syncfusion Inc.

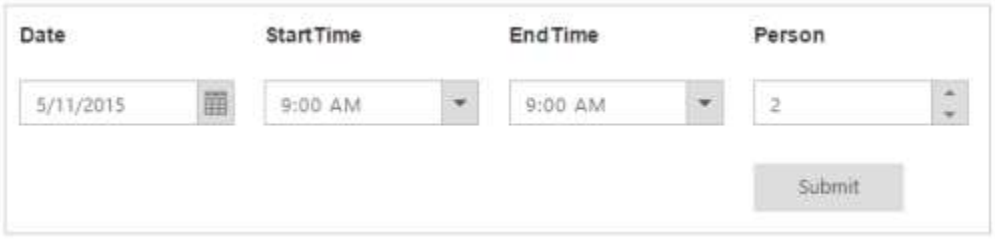
- Persist: Supports state maintenance when refreshing the page.
- RTL: Supports right-to-left alignment in the TimePicker control.
- Themes: The TimePicker control features 12 built-in themes (six flat and six with gradient effects) and also supports custom skin options for user-defined themes.

Getting Started

This section explains briefly about how to create a TimePicker in your application with ASP.NET MVC.

Create your first TimePicker in MVC

Essential MVCTimePicker provides support to display the time in your web page and allows you to pick a time from the list. In this section you can learn how to customize TimePickers in a real-time application. The following example shows you how to use the TimePicker control to book a hotel dining table booking application, within a limited time, in any day.



The above screenshot illustrates the functionality of a TimePicker, with a time range from morning to evening. You can select the time to book a table ranging from 9.00 AM to 10.00 PM in the current day. This avoids selecting a time prior to that day.

Create a TimePicker

Essential MVC TimePicker widget has built-in features such as keyboard navigation, other time navigation with animations, and flexible APIs. You can easily create the TimePicker widget using simple TimePicker element as follows.

1. You can create an MVC Project and add necessary assemblies and scripts to it. Refer MVC-Getting Started Documentation.
2. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
<table>
<tr>
<td class="table-cell">Date</td>
<td class="table-cell">Time</td>
<td class="table-cell">Person</td>
</tr>
<tr>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().DatePicker("startDate")
</span>
</td>
<td class="table-cell">
<span class="inner-display">
```

```
@Html.EJ().TimePicker("timeStart")
</span>
</td>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().NumericTextbox("NumericTextbox").Value("0")
</span>
</td>
</tr>
<tr>
<td></td>
<td></td>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().Button("Submit").Width("100px").Size(ButtonSize.Small).Text("S
ubmit").ClientSideEvents(s => s.Click("button"))
</span>
</td>
</tr>
</table>
```

3. Add the following styles to show the TimePicker control in horizontal order in the Content folder.

CSS

```
<style type="text/css" class="cssStyles">
.table-cell
{
width: 70px;
height: 10px;
font-weight: bold;
padding: 10px;
}
.inner-display
{
display: inline-block;
}
</style>
```

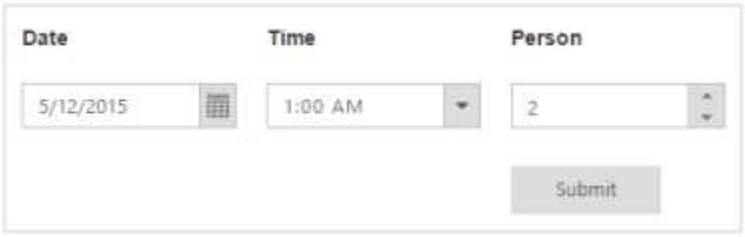
4. Add the following code to the Layout page for DatePicker and TimePicker initialization.

JS

```
<script type="text/javascript">
function button()
{
alert("You have Successfully booked");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

5. The following screenshot displays the output for the above code.



6. Click the submit button in the application to render the following output.



Set Min and Max Date

In a real-time scenario, the booking is open only for a limited time. You can select a date and time from the given range using the properties MinDate, MinTime and MaxDate, MaxTime that enables only the dates and times ranging between the MinDate, MinTime and MaxDate, MaxTime in the DatePicker and TimePicker.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
<table>
<tr>
<td class="table-cell">Date</td>
<td class="table-cell">StartTime</td>
<td class="table-cell">EndTime</td>
<td class="table-cell">Person</td>
</tr>
<tr>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().DatePicker("startDate").Value("5/11/2015").MinDate("5/11/2015"
).MaxDate("6/11/2015")
</span>
```

```
</td>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().TimePicker("timeStart").MinTime("9:00 AM").MaxTime("10:00
PM").Interval(60).ClientSideEvents(e => e.Select("selectedStartTime"))
</span>
</td>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().TimePicker("timeEnd").MinTime("9:00 AM").MaxTime("10:00
PM").Interval(60)
</span>
</td>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().NumericTextbox("NumericTextbox").Value("0")
</span>
</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td class="table-cell">
<span class="inner-display">
@Html.EJ().Button("Submit").Width("100px").Size(ButtonSize.Small).Text("S
ubmit").ClientSideEvents(s => s.Click("button"))
</span>
</td>
</tr>
</table>
```

2. Add the following styles to show the TimePicker control in horizontal order in the Content folder.

CSS

```
<style type="text/css" class="cssStyles">
.table-cell
{
width: 70px;
height: 10px;
font-weight: bold;
padding: 10px;
}
.inner-display
{
display: inline-block;
}
</style>
```

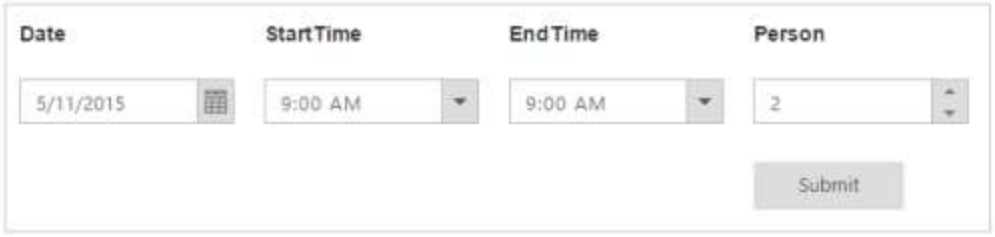
3. Add the following code to the Layout page for DatePicker and TimePicker initialization.

Copyright © 2001 - 2020 Syncfusion Inc.

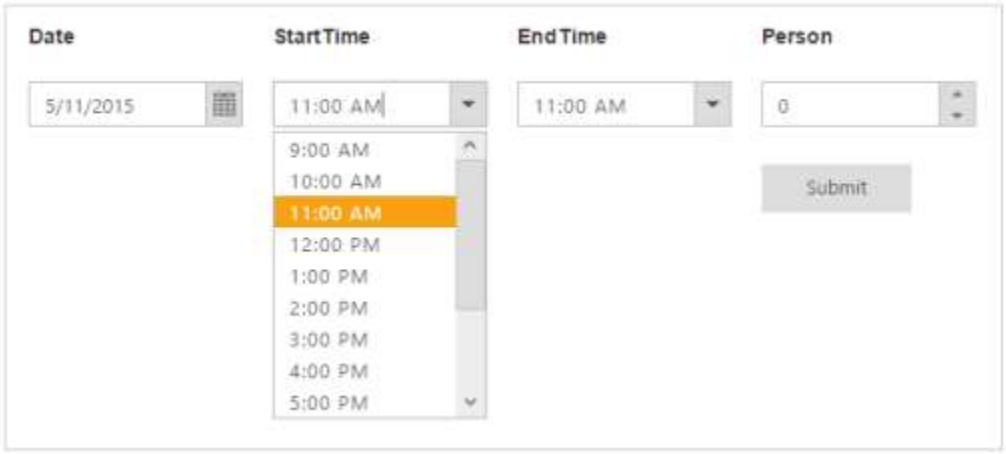
JS

```
<script type="text/javascript">
function selectedStartTime(sender) {
var selDate = sender.value; // mentions the selected time.
minTimepicker = $("#timeEnd").data("ejTimePicker");// creating TimePicker
object
minTimepicker.setModel({ "minTime": selDate });// setting minTime
property through setModel of TimePicker object.
}
function button()
{
alert("You have Successfully booked");
}
</script>
```

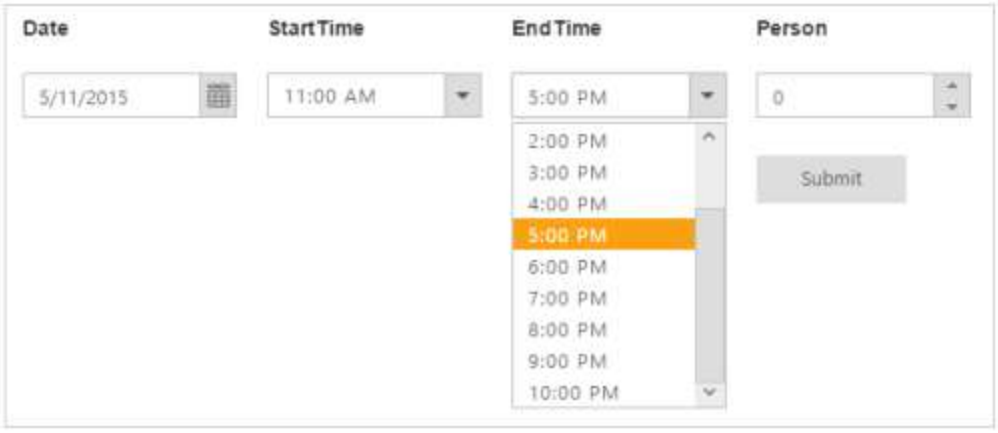
4. The following screenshot displays the output for the above code.



You can select the Start time in the first TimePicker and select the End time within the given range. The second TimePicker start value is based on the first TimePicker's selected value. This is illustrated in the following screenshots.



Copyright © 2001 - 2020 Syncfusion Inc.



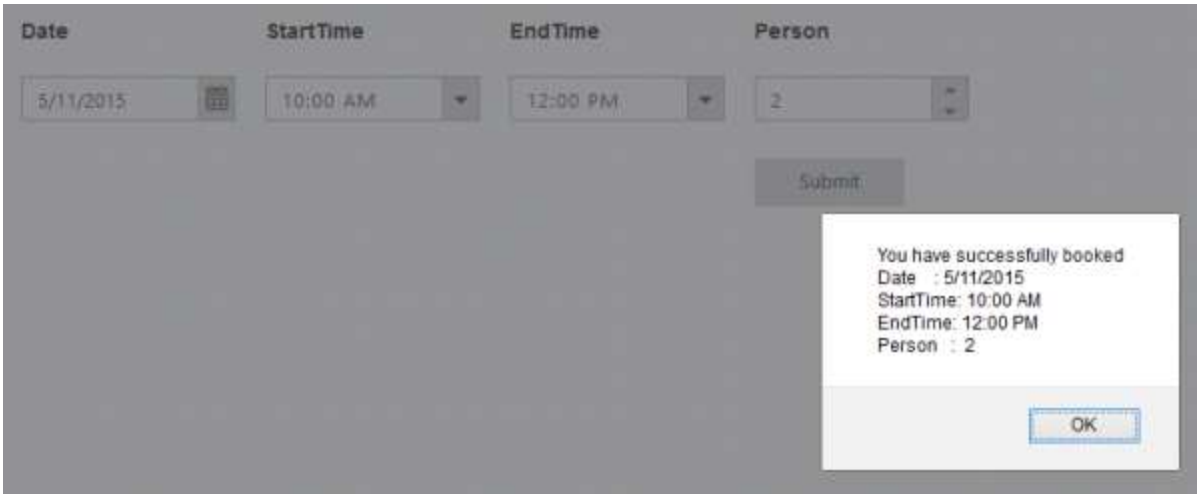
Display Reserved Time

An acknowledgment message appears when you click the Submit button.

You can specify the alert message in the script as follows.

JS

```
<script type="text/javascript">
function button()
{
var a = $('#startDate').val();
var b = $('#timeStart').val();
var d = $('#timeEnd').val();
var c = $('#NumericTextbox').val();
alert("You have successfully booked\nDate    : " + a + "\nStartTime: " +
b + "\nEndTime: " + d + " \nPerson   :  " + c);
}
</script>
```



TimePicker Customization

The TimePicker provides support to display a TimePicker in your web page and allows you to pick a time from it.

Copyright © 2001 - 2020 Syncfusion Inc.

Creating TimePicker Widget

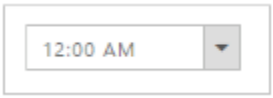
The following steps explains you to create a TimePicker widget.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
@Html.EJ().TimePicker("time")
```

2. The following screenshot illustrates you a default TimePicker.



TimePicker Control

Localization

You can globalize the TimePicker, so that users of different cultures can make use of it and post their content. For your convenience, you can format the TimePicker control to your culture. When your blog is in your culture, the viewers of your culture can understand about your company and its products. You can achieve localization using the “Locale” property.

More than 350 culture specific files are available to localize the time. To know more about EJ globalize support, please refer the below link

http://help.syncfusion.com/js/localization

Note: Seven culture-specific script files are available in the below specified location. For all other culture files, please download from the GitHub location.

(installed location)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n
For example, If you have installed the Essential Studio package within C:\Program Files (x86), then navigate to the below location, C:\Program Files (x86)\Syncfusion\Essential Studio\{{ site.releaseversion }}\JavaScript\assets\scripts\i18n

The following code example is used to set TimePicker in Spanish language.

Enabling Localization Support

The following steps explains you on how to enable Locale property for TimePicker.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with Spanish culture*@
@Html.EJ().TimePicker("time").Locale("es-ES")
```

Copyright © 2001 - 2020 Syncfusion Inc.

Execute the above code to render the following output.



TimePicker with es-ES Localization

Behavior Settings

Set value of the TimePicker widget

You can use Value property to set default time for the TimePicker.

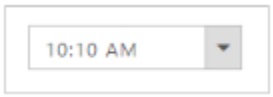
The following steps explains you on how to set the default value of the TimePicker.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with given time value*@
@Html.EJ().TimePicker("time").Value("10:10 AM")
```

Execute the above code to render the following output.



TimePicker default value

Enable/Disable TimePicker widget

TimePicker widget provides you an option to enable /disable the widget. You can disable the TimePicker by setting the “Enabled” property value as false.

The following steps explain you to enable/disable property in TimePicker widget.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render disabled TimePicker widget*@
@Html.EJ().TimePicker("time").Enabled(false)
```


Disabled TimePicker widget

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.

Restrict editing

TimePicker widget provides ReadOnly property to disable editing in the control. Therefore you can only read the value set to TimePicker and cannot modify it. The Value property allows you to set the default value for TimePicker widget when it is created.

Configure TimePicker textbox to restrict editing

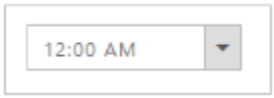
The following steps allows you to disable editing value in TimePicker.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
@Html.EJ().TimePicker("time").ReadOnly(true)
```

The following screenshot illustrates a TimePicker textbox configured to restrict editing.



,

TimePicker with PopUp button and enable ReadOnly property

Rounded Corner

You can customize the shape of the TimePicker widget from regular rectangular shape to rounded rectangle shape using ShowRoundedCorner property that is set to false by default.

Configure Rounded corner to TimePicker Text box

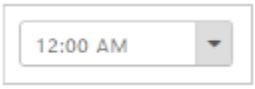
The following steps explain you to change the edges of the textbox to rounded corner.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with rounded corner*@
@Html.EJ().TimePicker("time").ShowRoundedCorner(true)
```

The following screenshot illustrates a TimePicker when ShowRoundedCorner is set to “true”.



Rounded corner of TimePicker Textbox

Scaling

TimePicker widget provides you options to change its height, width and also to change popup height and width.

Change TimePicker widget Height and Width

You can use Height and Width property to customize TimePicker width and height.

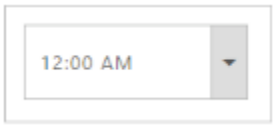
Copyright © 2001 - 2020 Syncfusion Inc.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with specified height and width*@
@Html.EJ().TimePicker("time").Height("50px").Width("150px")
```

Execute the above code to render the following output.



Changing Width and Height of the TimePicker.

Changing TimePicker PopupHeight and PopupWidth

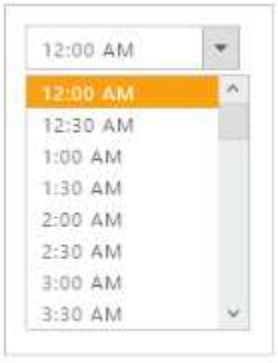
You can use PopupHeight and PopupWidth property to customize TimePicker width and height.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with specified popup height and width*@
@Html.EJ().TimePicker("time").PopupHeight("170px").PopupWidth("150px")
```

Execute the above code to render the following output.



Changing popup height and width

State persistence

TimePicker widget provide options to maintain the selected value after you refresh the page by using EnablePersistence property.

Steps to use State persistence of the TimePicker

The following steps explains you to use EnablePersistence property.

1. Add the following code to the corresponding view page to render the TimePicker.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
@Html.EJ().TimePicker("time").Value("10:10 AM").EnablePersistence(true)
```

Strict mode of the TimePicker

TimePicker widget provides you an option to set default value when there is no default value between MinTime and MaxTime by using EnableStrictMode property.

Steps to use EnableStrictMode property

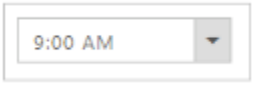
The following steps explains you to use EnableStrictMode property.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with specifies min and max time*@
@Html.EJ().TimePicker("time").Value("09:00 AM").MinTime("10:00
AM").MaxTime("09:00 PM").EnableStrictMode(true)
```

Execute the above code to render the following output.



Using EnableStrictMode property

Interval

TimePicker widget provides you an option to change the interval of the hour, minute and seconds.

Steps to change the Time interval of the TimePicker control

The following steps explains you to change the Time Interval of the TimePicker.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
@Html.EJ().TimePicker("time").TimeFormat("hh:mm:ss
tt").HourInterval(2).MinutesInterval(60).SecondsInterval(20)
```

Range

TimePicker widget provide options to set the Range (MinTime & MaxTime) for the time.

Steps to change MinTime & MaxTime of the TimePicker

The following steps explains you to change the Range of the TimePicker.

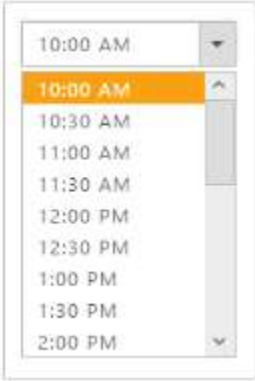
1. Add the following code to the corresponding view page to render the TimePicker.

Copyright © 2001 - 2020 Syncfusion Inc.

HTML

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with Min and Max time*@
@Html.EJ().TimePicker("time").MinTime("10:00 AM").MaxTime("10:00 PM")
```

Execute the above code to render the following output.



Range for TimePicker widget

Time Format

TimePicker widget provides you an option to change the TimeFormat property.

Steps to change Time Format of TimePicker widget

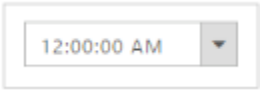
The following steps explains you to change the time format for the TimePicker.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with time format*@
@Html.EJ().TimePicker("time").TimeFormat("hh:mm:ss tt")
```

Execute the above code to render the following output.



TimePicker with TimeFormat.

RTL

This feature supports to change the left-to-right alignment of the TimePicker widget to right-to-left when set the EnableRTL as true. By default EnableRTL value is set as “false”. The custom template TimePicker also supports RTL.

Enabling Right-To-Left Support

The following steps explains you in enabling the right-to-left property for the TimePicker.

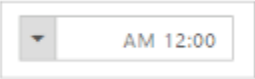
1. Add the following code to the corresponding view page to render the TimePicker.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget with right to left appearance*@
@Html.EJ().TimePicker("time").EnableRTL(true)
```

The following screenshot illustrates a TimePicker control when EnableRTL is set to “true”



TimePicker template with RTL support

Keyboard Interaction

You can use Keyboard shortcut keys as an alternative to the mouse on using TimePicker widget. TimePicker widget allows you to perform all kinds of actions using keyboard shortcuts. For enabling the keyboard navigation, we need to set the access key to TimePicker control by using HtmlAttributes property. The HtmlAttributes receives the IDictionary object. The IDictionary represents a generic collection of key/value pairs.

The various keyboard shortcuts available within the TimePicker widget are discussed in the following table.

List of keyboard shortcuts

Shortcut Key	Description
Access keyÂ + j	Focuses into Timepicker widget
Alt + Down	Opens/Hides the popup
Right/Left	Moves to adjacent part
Up	Increments the value
Down	Decrements the value

When popup is open

List of keyboard shortcuts

Shortcut Key	Description
Up	Selects the previous time
Down	Selects the next time.
Home/End	Moves to the first / last item
Esc	Closes the popup

Copyright © 2001 - 2020 Syncfusion Inc.

Configure Keyboard Interaction

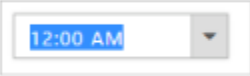
The following steps explains you on how to enable keyboard interaction for the TimePicker widget with access key as “j”. We can set the any other key as access key in TimePicker control.

1. Add the following code to the corresponding view page to render the TimePicker.

ASPX-CS

```
@*Add the following code example to the corresponding CSHTML page to
render TimePicker widget*@
@{IDictionary<string, object> parameters = new Dictionary<string,
object>();
parameters.Add("accesskey", "j");
}
@Html.EJ().TimePicker("time").HtmlAttributes(parameters)
```

2. Run the code sample, press Access key + J to focus in the TimePicker widget that enables it and you can navigate using arrow keys and Esc key to close the popup.



TimePicker focused with keyboard shortcut

How to?

Render TimePicker from Code behind

TimePicker can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of TimePicker properties in the controller.

C#

```
//Namespace to get the JavaScript (TimePicker) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class TimePickerController : Controller
{
// GET: TimePicker
public ActionResult TimePickerFeatures()
{
//Initializing the TimePicker model
TimePickerProperties TimeObj = new TimePickerProperties();
//Initializing the datasource and other properties
TimeObj.Width = "300px";
TimeObj.Interval= 60;
TimeObj.Value = "10:10 AM";
//Passing TimePicker properties using the ViewData
ViewData["TimeModel"] = TimeObj;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

Binding the TimePicker properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().TimePicker("searchCustomer",(Syncfusion.JavaScript.Models.TimeP
ickerProperties)ViewData["TimeModel"]).Render();
}
```

ToggleButton

ToggleButton

The Toggle Button allows you to perform the toggle option by using checked and unchecked state. This ToggleButton can be helpful to you to check their states. Toggle Button is used as normal click able button, submitting form data, resetting the form data to its initial value. According to the usage of button, you can render the Toggle Button in three types by using the type property.

Key Features

- Trendy Look : Rich Appearance with Theme Support
- RTL : Supports for Right to Left alignment
- Text and Image : Supports both text and image as Toggle Button content in both On/Off state
- Built-in Icon :Supports the built-in icon libraries
- Easy Customization: The customization of button control to any form is made simple

Getting Started

This section explains briefly about how to create a ToggleButton in your ASP.NET MVC application.

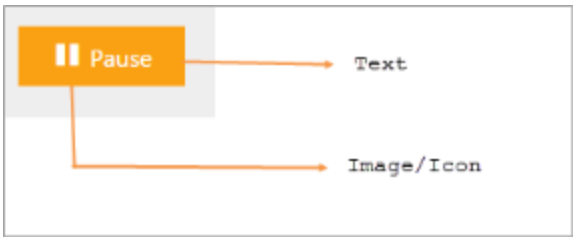
Create your first Toggle Button in MVC

The Toggle Button control displays both text and images. The text displayed on the Toggle Button is contained in the Text property. The Toggle Button control display images using the SpriteCssClass and ImagePosition properties. The Toggle Buttons has theme support also.

The following screenshot illustrates a Toggle Button control.



Toggle Button in OFF state_



Toggle Button in ON state

Create a Toggle Button

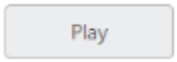
Essential Studio ASP.NET MVC Toggle Button control has a built-in feature to customize the size, text and images of button.

1. Create an MVC Project and add required assemblies, scripts, and styles to it. Refer MVC-Getting Started Documentation.
2. Add the following code example to the corresponding view page to render the Toggle Button.

ASPX-CS

```
@Html.EJ().ToggleButton("button").Size(ButtonSize.Mini).ShowRoundedCorner
(true).DefaultText("Play")
```

3. Output of the above steps.



Easy Customization

Toggle Button is used in all applications. ToggleButton size, content and content location is varied according to each application. The following section contains some customizable option for Toggle Button that can perform easily.

Toggle State

Toggle button has two states like off / on state in a switch. By default you can set any state at initial and then you can move from one state to another state by clicking on the Toggle Button. These two states are Default and Active.ToggleState property is used to set the state of Toggle Button as default state or active state. Default value of ToggleState is false.

The following steps explains you the details about rendering the Toggle Button with different ToggleState.

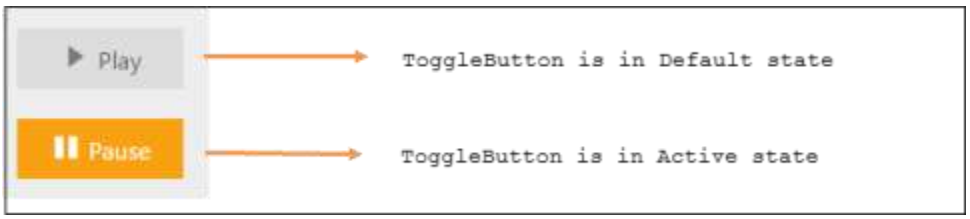
ASPX-CS

```
//Add the code in CSHTML page to configure the widget and initialize the
control
@*set the state for toggle button*@
@Html.EJ().ToggleButton("toggleButton_default").Size(ButtonSize.Small).Co
ntentType(ContentType.TextAndImage).DefaultText("Play").ActiveText("Pause
").DefaultPrefixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon e-
mediapause").ToggleState(false)
<br/>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().ToggleButton("toggleButton_active").Size(ButtonSize.Small).Con
tentType(ContentType.TextAndImage).DefaultText("Play").ActiveText("Pause"
).DefaultPrefixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon e-
mediapause").ToggleState(true)
```

Execute the above code to render the following output.



Toggle button with two different toggle states

ToggleState with icons

Prefix and Suffix Icons

You can add icons in prefix and suffix position of Toggle Button. Location of Icon is customized easily using the following mentioned option. This is applicable for the content type's ImageOnly, TextAndImage, ImageTextImage and ImageBoth.

Toggle Button control also supports the Built-in icon libraries. The ej.widgets.core.min.css contains the definition for important icons that are used in toggle buttons. You can use these Built-in icons by mentioning the icon class name as value in DefaultPrefixIcon, DefaultSuffixIcon, ActivePrefixIcon,and ActiveSuffixIcon property. You can use any font icons that is defined in ej.widgets.core.min.css it avoids the complexity in specifying icon using sprite image and CSS.

For example the following mentioned Built-in CSS class are used to show the font icons that are used by media player.

- e-mediaback
- e-mediaforward
- e-medianext
- e-mediaprev
- e-mediaeject
- e-mediaclose
- e-mediapause
- e-mediaplay

Prefix Icon

It inserts the icon at the starting position of Toggle Button. After this prefix icon, you can use text or suffix icon.

Suffix Icon

It inserts the icon at the ending position of Toggle Button. Before this suffix icon, you can use text or prefix icon.

You can also set icon in different location (prefix, suffix) and in different state (default, active) by using the option provided. The following properties are defined for merging the option to add text, icon with different position and in toggle states.

Copyright © 2001 - 2020 Syncfusion Inc.

Property Table

ActiveText	Specifies the text of toggle button in active state
ActivePrefixIcon	Specifies the prefix icon of toggle button in active state
ActiveSuffixIcon	Specifies the suffix icon of toggle button in active state
DefaultText	Specifies the text of toggle button in default state.
DefaultPrefixIcon	Specifies the prefix icon of toggle button in default state
DefaultSuffixIcon	Specifies the suffix icon of toggle button in default state

The following script explains you the details about rendering the Toggle Button with above mentioned customization properties.

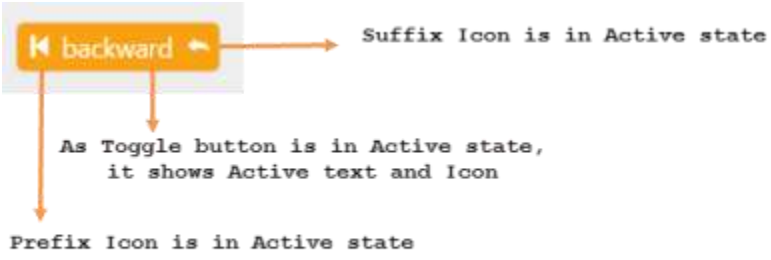
ASPX-CS

```
//Add the code in CSHTML page to configure the widget and initialize the
control
@*set the prefix, suffix built-in icons in default and active state*@
@Html.EJ().ToggleButton("toggleButton_iconsAndStates").ContentType(Conten
tType.ImageTextImage).ShowRoundedCorner(true).DefaultText("forward").Acti
veText("backward").DefaultPrefixIcon("e-icon e-
mediaforward").ActivePrefixIcon("e-icon e-
mediaback").DefaultSuffixIcon("e-icon e-redo").ActiveSuffixIcon("e-icon
e-undo")
```

Execute the above code to render the following output.



Before clicking the toggle button



After clicking the toggle button

Copyright © 2001 - 2020 Syncfusion Inc.

Toggle button size

You can render the Toggle Button in different sizes by using Size property. You can use some predefined size option for rendering a Toggle Button in easiest way. Each size option has different height and width. It mainly avoids the complexity in rendering Toggle Button with complex CSS class. You can mention the size of the Toggle Button using the following five predefined size options.

Predefined Toggle Button size

Normal	Creates toggle button with content size.
Mini	Creates toggle button with Built-in mini size height, width specified.
Small	Creates toggle button with Built-in small size height, width specified.
Medium	Creates toggle button with Built-in medium size height, width specified.
Large	Creates toggle button with Built-in large size height, width specified.

You can also set your own width and height for toggle button using Height and Width property.

The following steps explains you the details about rendering the Toggle Button with above mentioned size options.

1. In the View page, add the following button elements to configure Toggle Button widget.

ASPX-CS

```
//Add the code in CSHTML page to configure the widget and initialize the
control
<div class="control">
@*Set the different size types for toggle button control as follows. *@
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_normal").Size(ButtonSize.Normal).Sh
owRoundedCorner(true).ContentType(ContentType.ImageOnly).DefaultPrefixIco
n("e-icon e-mediaplay").ActivePrefixIcon("e-icon e-mediaforward")
</td> </tr><tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_mini").Size(ButtonSize.Mini).ShowRo
undedCorner(true).DefaultText("Play").ActiveText("Next")
</td></tr><tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small").Size(ButtonSize.Small).Show
RoundedCorner(true).DefaultText("Play").ActiveText("Next")
</td></tr><tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_medium").Size(ButtonSize.Medium).Sh
owRoundedCorner(true).DefaultText("Play").ActiveText("Next")
</td></tr><tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_large").Size(ButtonSize.Large).Show
RoundedCorner(true).ContentType(ContentType.TextAndImage).DefaultText("Pl
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
ay").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-mediaforward")
</td></tr><tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_customSize").ShowRoundedCorner(true
).ContentType(ContentType.TextAndImage).DefaultText("Play").ActiveText("N
ext").DefaultPrefixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").Height("50").Width("150")
</td>
</tr>
</table>
</div>
```

Execute the above code to render the following output.



Toggle button in different sizes

Content type

The content of the Toggle Button is mainly text and images. Instead of using complex CSS classes to render Toggle Button with different content types, you can use some predefined content type options as listed in the following table. Using this ContentType property you can easily add different types of content for Toggle Button. The Toggle Button supports the following content types.

List of Content types

TextOnly	Supports only for text content only.
ImageOnly	Supports only for image content only
ImageBoth	Supports image for both ends of the toggle button.
TextAndImage	Supports image with the text content.
ImageTextImage	Supports image with both ends and middle in text.

The following steps explains you the details about rendering the Toggle Button with above mentioned content type options.

1. In the View page, add the following button elements to configure Toggle Button widget.

Copyright © 2001 - 2020 Syncfusion Inc.

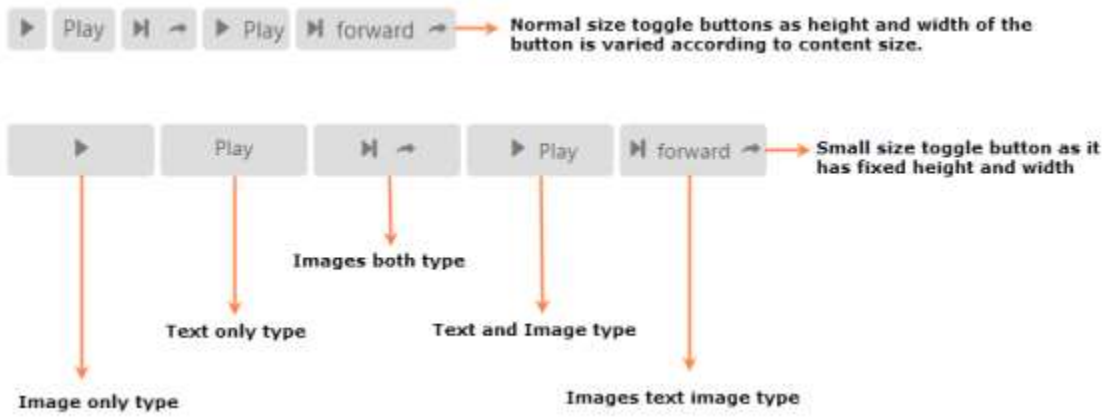
ASPX-CS

```
//Add the code in CSHTML page to configure the widget and initialize the
control
<div class="normal">
@*set different content types for toggle button*@
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_imageOnly").ShowRoundedCorner(true)
.ContentType(ContentType.ImageOnly).DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-mediaforward")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_textOnly").ShowRoundedCorner(true).
DefaultText("Play").ActiveText("Next")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_imageBoth").ShowRoundedCorner(true)
.ContentType(ContentType.ImageBoth).DefaultPrefixIcon("e-icon e-
mediaforward").ActivePrefixIcon("e-icon e-
mediaback").DefaultSuffixIcon("e-icon e-undo").ActiveSuffixIcon("e-icon
e-redo")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_textAndImage").ShowRoundedCorner(tr
ue).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Default
Text("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-mediaforward")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_imageTextImage").ShowRoundedCorner(
true).ContentType(ContentType.ImageTextImage).DefaultText("forward").Acti
veText("backward").DefaultPrefixIcon("e-icon e-
mediaforward").ActivePrefixIcon("e-icon e-
mediaback").DefaultSuffixIcon("e-icon e-undo").ActiveSuffixIcon("e-icon
e-redo")
</td>
</tr>
</table>
</div>
<div class="small">
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_imageOnly").Size(ButtonSize.S
mall).ShowRoundedCorner(true).ContentType(ContentType.ImageOnly).DefaultP
refixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon e-mediaforward")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_textOnly").Size(ButtonSize.Sm
all).ShowRoundedCorner(true).DefaultText("Play").ActiveText("Next")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_imageBoth").Size(ButtonSize.S
mall).ShowRoundedCorner(true).ContentType(ContentType.ImageBoth).DefaultP
refixIcon("e-icon e-mediaforward").ActivePrefixIcon("e-icon e-
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
mediaback").DefaultSuffixIcon("e-icon e-undo").ActiveSuffixIcon("e-icon
e-redo")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_textAndImage").Size(ButtonSiz
e.Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).De
faultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-mediaforward")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_imageTextImage").Size(ButtonS
ize.Small).ShowRoundedCorner(true).ContentType(ContentType.ImageTextImage
).DefaultText("forward").ActiveText("backward").DefaultPrefixIcon("e-icon
e-mediaforward").ActivePrefixIcon("e-icon e-
mediaback").DefaultSuffixIcon("e-icon e-undo").ActiveSuffixIcon("e-icon
e-redo")
</td>
</tr>
</table>
</div>
```

Execute the above code to render the following output.



Toggle button with different content type

ImagePosition

To provide the best look and feel for Toggle Button, position of images in toggle button is important. You can easily customize the position of images inside toggle button using ImagePosition property without using any complex CSS. ImagePosition property is applicable only with the TextAndImage content type property. This property represent the position of images with respect to text.

Property Table

ImageLeft	Support for aligning text in right and image in left.
ImageRight	Support for aligning text in left and image in right.
ImageTop	Support for aligning text in bottom and image in top.
ImageBottom	Support for aligning text in top and image in bottom.

Copyright © 2001 - 2020 Syncfusion Inc.

The following steps explains you the details about rendering the Toggle Button with above mentioned image Position options.

1. In the View page, add the following button elements to configure Toggle Button widget.

ASPX-CS

```
@*Add the code in CSHTML page to configure the widget and initialize the
control*@
@*set images of toggle button in different location*@
<div class="one">
@*align the images in left side*@
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_normal_imageLeft").Size(ButtonSize.
Normal).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Def
aultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageLeft)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_mini_imageLeft").Size(ButtonSize.Mi
ni).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Default
Text("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageLeft)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_imageLeft").Size(ButtonSize.S
mall).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Defau
ltText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageLeft)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_medium_imageLeft").Size(ButtonSize.
Medium).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Def
aultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageLeft)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_large_imageLeft").Size(ButtonSize.L
arge).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Defau
ltText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageLeft)
</td>
</tr>
</table>
</div>
<div class="one">
@*align the images in right side*@
<table>
<tr>
```

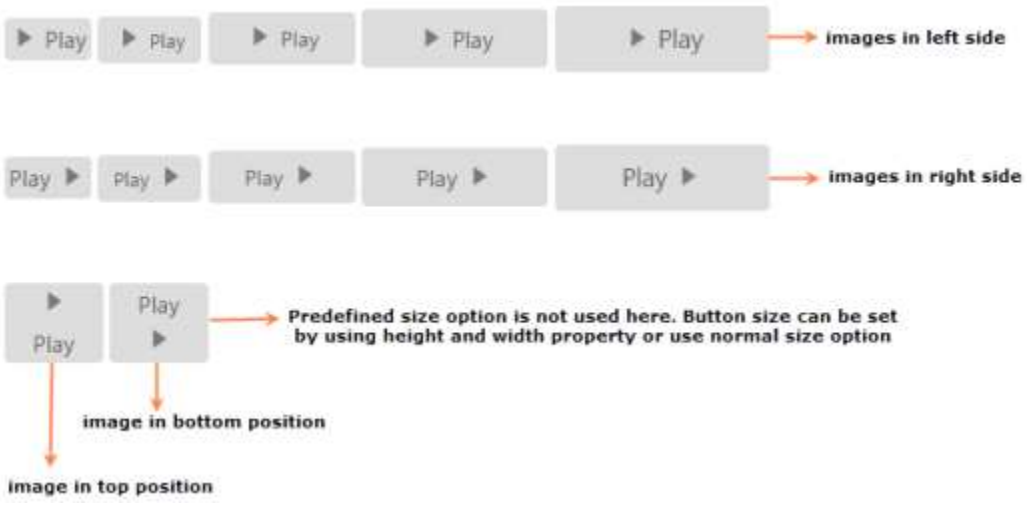
Copyright © 2001 - 2020 Syncfusion Inc.

```
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_normal_imageRight").Size(ButtonSize
.Normal).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).De
faultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageRight)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_mini_imageRight").Size(ButtonSize.M
ini).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Defaul
tText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageRight)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_small_imageRight").Size(ButtonSize.
Small).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Defa
ultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageRight)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_medium_imageRight").Size(ButtonSize
.Medium).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).De
faultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageRight)
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_large_imageRight").Size(ButtonSize.
Large).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Defa
ultText("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageRight)
</td>
</tr>
</table>
</div>
<div class="two">
@*align the images in Top and Bottom*@
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_imageTop").ShowRoundedCorner(true).
ContentType(ContentType.TextAndImage).DefaultText("Play").ActiveText("Nex
t").DefaultPrefixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").ImagePosition(ImagePosition.ImageTop).Height("60").Width("
50")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_imageBottom").ShowRoundedCorner(tru
e).ContentType(ContentType.TextAndImage).DefaultText("Play").ActiveText("
Next").DefaultPrefixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon
e-
mediaforward").ImagePosition(ImagePosition.ImageBottom).Height("60").Widt
h("50")
</td>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</tr>
</table>
</div>
```

Execute the above code to render the following output.



Toggle button with different type of image position

Theme support

You can control the style and appearance of Toggle Button based on CSS classes. To apply styles to the Toggle Button control, you can refer two files, ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.widgets.all.min.css file, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Toggle Button control namely,

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

Custom CSS

You can use the CSS to customize the Toggle Button control appearance. Define a CSS class as per requirement and assign the class name to CssClass property.

The following steps explains you the details about rendering the Toggle Button with custom CSS.

Copyright © 2001 - 2020 Syncfusion Inc.

1. In the View page, add the following button elements to configure Toggle Button widget.

ASPX-CS

```
@*Add the code in CSHTML page to configure the widget and initialize the
control*@
@*set different custom CSS class for toggle button*@
<div class="one">
@*apply custom CSS class for toggle buttons*@
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_customCSS1").Size(ButtonSize.Small)
.ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).DefaultTex
t("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").CssClass("customCss1")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_customCSS2").Size(ButtonSize.Small)
.ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).DefaultTex
t("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").CssClass("customCss2")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_customCSS3").Size(ButtonSize.Small)
.ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).DefaultTex
t("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").CssClass("customCss3")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_customCSS4").Size(ButtonSize.Small)
.ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).DefaultTex
t("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").CssClass("customCss4")
</td>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_customCSS5").Size(ButtonSize.Small)
.ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).DefaultTex
t("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-
mediaforward").CssClass("customCss5")
</td>
</tr>
</table>
</div>
```

2. Configure the CSS styles to apply on buttons.

CSS

```
<style type="text/css" class="cssStyles">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
/* Customize the button background */
.e-togglebutton.customCss1
{
background-color: #121111;
}
.e-togglebutton.customCss2
{
background-color: #94bbd5;
}
.e-togglebutton.customCss3
{
background-color: #f3533c;
}
.e-togglebutton.customCss4
{
background-color: #d1eeed;
}
.e-togglebutton.customCss5
{
background-color: #deb66e;
}
/* Customize the button image & text color */
.e-togglebutton.customCss1.e-btn.e-select .e-icon, .e-
togglebutton.customCss1.e-btn.e-select .e-btntxt
{
color: #94bbd5;
}
.e-togglebutton.customCss2.e-btn.e-select .e-icon, .e-
togglebutton.customCss2.e-btn.e-select .e-btntxt
{
color: #121111;
}
.e-togglebutton.customCss3.e-btn.e-select .e-icon, .e-
togglebutton.customCss3.e-btn.e-select .e-btntxt
{
color: #cef6f7;
}
.e-togglebutton.customCss5.e-btn.e-select .e-icon, .e-
togglebutton.customCss5.e-btn.e-select .e-btntxt
{
color: #534f4f;
}
</style>
```

Execute the above code to render the following output.



Toggle button with Custom CSS

Button types

Toggle Button is used as normal click able button, submitting form data, resetting the form data to its initial value. According to the usage of button, you can render the Toggle Button in the following three types by using the Type property.

Copyright © 2001 - 2020 Syncfusion Inc.

Toggle Button Types

Button	The button is a click able button
Submit	The button is a submit button (submits form-data)
Reset	The button is a reset button (resets the form-data to its initial values)

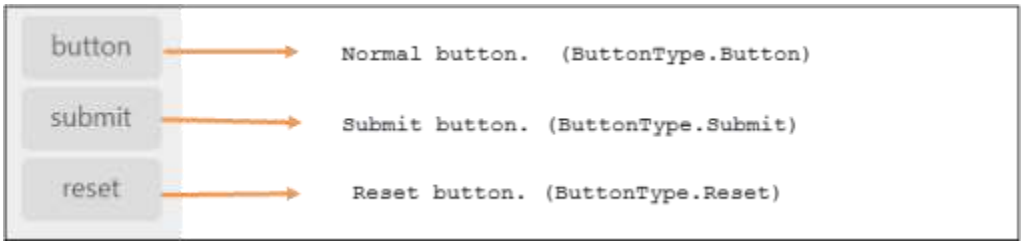
The following steps explains you the details about rendering the Toggle Button with above mentioned types.

1. In the View page, add the following button elements to configure Toggle Button widget.

ASPX-CS

```
@*Add the code in CSHTML page to configure the widget and initialize the
control*@
<div class="one">
@*set toggle button type using type property*@
<table>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_button").Size(ButtonSize.Mini).Show
RoundedCorner(true).ContentType(ContentType.TextOnly).DefaultText("button
").ActiveText("Next").Type(ButtonType.Button)
</td>
</tr>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_submit").Size(ButtonSize.Mini).Show
RoundedCorner(true).ContentType(ContentType.TextOnly).DefaultText("submit
").ActiveText("Next").Type(ButtonType.Submit)
</td>
</tr>
<tr>
<td class="btnsht">
@Html.EJ().ToggleButton("toggleButton_reset").Size(ButtonSize.Mini).ShowR
oundedCorner(true).ContentType(ContentType.TextOnly).DefaultText("reset")
.ActiveText("Next").Type(ButtonType.Reset)
</td>
</tr>
</table>
</div>
```

Execute the above code to render the following output.



Copyright © 2001 - 2020 Syncfusion Inc.

Types of Toggle button

RTL support

In some cases, it is necessary to use right to left alignment. You can render RTL support by using EnableRTL property. In RTL mode, when there is more than one content (image/text, image/image) in button, then the content is aligned in right to left format. For example, when text is in left and image is in right position, after applying right to left alignment these positions are interchanged.

The following steps explains you the details about rendering the Toggle Button with Right to left alignment support.

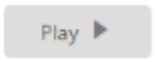
1. In the View page, add the following button elements to configure Toggle Button widget.

ASPX-CS

```
//Add the code in CSHTML page to configure the widget and initialize the
control
<div class="one">
@*enable right to left alignment*@
@Html.EJ().ToggleButton("toggleButton_rtl").Size(ButtonSize.Small).ShowRo
undedCorner(true).ContentType(ContentType.TextAndImage).DefaultText("Play
").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-medianext").EnableRTL(true)
</div>
```

In above mentioned code example PrefixIcon property is used and the icon that is to be on left side, (before text) is rendered on right side as EnableRTL property is used with PrefixIcon. Consequently, the alignment is changed in right to left order.

Output of above steps



Toggle button with RTL support

Miscellaneous

ShowRoundedCorner

It sets the corner of Toggle Button in rounded shape. The Toggle Button, by default doesn't have rounded corner. To set rounded corner, you can enable ShowRoundedCorner property.

The following steps explains you the details about rendering the Toggle Button with Rounded corner support.

1. In the View page, add the following button elements to configure Toggle Button widget.

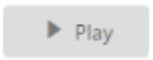
ASPX-CS

```
@*Add the code in CSHTML page to configure the widget and initialize the
control*@
<div class="one">
@*set rounded corner for toggle button*@
@Html.EJ().ToggleButton("toggleButton_roundedCorner").Size(ButtonSize.Sma
ll).ShowRoundedCorner(true).ContentType(ContentType.TextAndImage).Default
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Text("Play").ActiveText("Next").DefaultPrefixIcon("e-icon e-
mediaplay").ActivePrefixIcon("e-icon e-medianext")
</div>
```

Execute the above code to render the following output.



Toggle button with Rounder corner

PreventToggle

This property is used to prevent the state change of Toggle Button when it is clicked. When you set PreventToggle property as true, then the state of the Toggle Button is not changed even though it is clicked. Default value of PreventToggle is false.

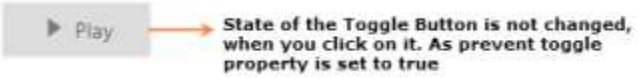
The following steps explains you the details about rendering the Toggle Button with PreventToggle property enabled.

1. In the View page, add the following button elements to configure Toggle Button widget.

ASPX-CS

```
@*Add the code in CSHTML page to configure the widget and initialize the
control*@
<div class="one">
@* set prevent toggle property for preventing states*@
@Html.EJ().ToggleButton("toggleButton_preventToggle").Size(ButtonSize.Sma
ll).ContentType(ContentType.TextAndImage).DefaultText("Play").ActiveText(
"Next").DefaultPrefixIcon("e-icon e-mediaplay").ActivePrefixIcon("e-icon
e-medianext").PreventToggle(true)
</div>
```

Execute the above code to render the following output.



Toggle button with prevent Toggle

How to?

Render ToggleButton from Code behind

ToggleButton can be rendered from the code behind by initializing the required properties in controller and passing those properties via **ViewData** or **Model** to the client side.

The following code illustrates the initialization of ToggleButton properties in the controller.

C#

```
//Namespace to get the JavaScript (ToggleButton) component properties
using Syncfusion.JavaScript.Models;
namespace MvcApplication.Controllers
{
public class ToggleButtonController : Controller
```

```
{
// GET: ToggleButton
public ActionResult ToggleButtonFeatures()
{
//Initializing the ToggleButton model
ToggleButtonProperties BtnObj = new ToggleButtonProperties();
//Initializing the datasource and other properties
BtnObj.ActiveText = "pause";
BtnObj.DefaultText = "Play";
BtnObj.ShowRoundedCorner = true;
BtnObj.Size = ButtonSize.Large;
//Passing ToggleButton properties using the ViewData
ViewData["BtnModel"] = BtnObj;
return View();
}
}
}
```

Binding the ToggleButton properties passed via **ViewData** from the controller in the client side as below.

ASPX-CS

```
@{
Html.EJ().ToggleButton("searchCustomer",(Syncfusion.JavaScript.Models.Tog
gleButtonProperties)ViewData["BtnModel"]).Render();
}
```

Toolbar

Toolbar

The Toolbar control for ASP.NET MVC supports displaying a list of tools within a web page. This control is capable of customizing toolbar items with any functionality by using enriched client-side methods. This control is composed of collection of unordered lists containing text and images contained into a div.

Key Features

- Modern look: Rich appearance with theme support.
- RTL: Supports right-to-left alignment.
- Text and Image: Supports both text and images as toolbar content.
- Easy Customization: The customization of toolbar control to any form is made simple.
- Data binding: Supports for data binding with local data and remote data.
- Template: Supports for Customizing toolbar with user needed tools.
- Keyboard navigation: Support for navigation through keyboard.

Getting Started

This section explains briefly about how to create a Toolbar in your ASP.NET MVC application.

Create your first Toolbar in MVC

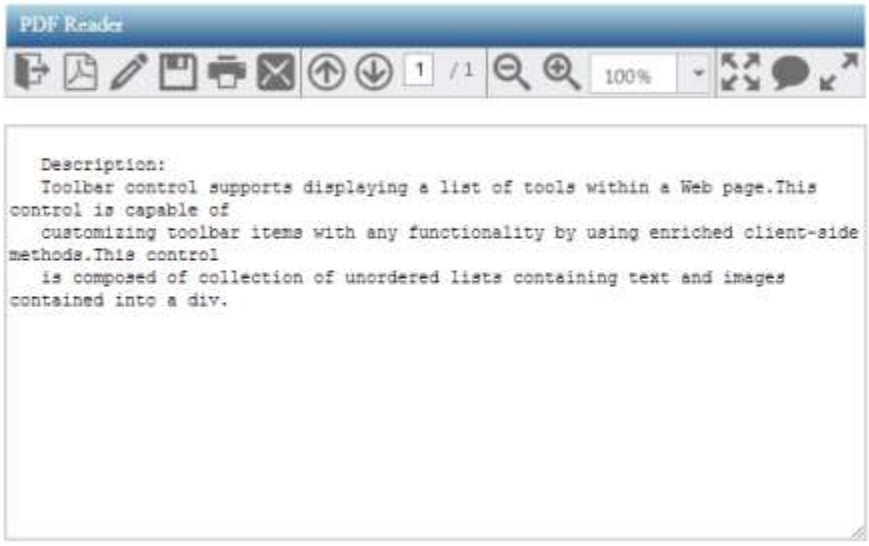
Create Toolbar for PDF Reader

Toolbar control displays a list of tools in a web page. It is used to customize Toolbar items of any functionality, by using enriched client-side methods. This control consists of a collection of unordered

Copyright © 2001 - 2020 Syncfusion Inc.

lists, containing text and images into a . From the following section, you can learn how to customize Toolbar control for a PDF Reader scenario. The following screen shot shows the appearance of Toolbar in PDF Reader simulator application.

Here, the Toolbar consists of a Toolbar with title and text area as PDF Reader.



Create a Toolbar

Using the following steps, you can create a Toolbar control. The basic rendering of ASP.NET MVC Toolbar is achieved with default functionality.

1. You can create an MVC Project and add necessary assembly and script with the help of the given MVC-Getting Started Documentation.

2. Add the mentioned code to the corresponding view page for Toolbar rendering.

ASPX-CS

```
@Html.EJ().Toolbar("ToolbarItem")
```

The following output is displayed.

Toolbar without Toolbar items

Initialize Toolbar Items

Toolbar consists of a list of items. From the following guidelines, you can learn how to initialize the Toolbar items with <UL> <LI> template.

Initialize the Toolbar items with <UL> <LI> template as follows.

ASPX-CS

```
<div id="ToolbarItem">
<!--list of toolbar items-->
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<ul>
<li id="OtherFormat" title="Convert PDF files to Word or Excel Online…">
<div class="PdfDocument e-icon convertToOthers "></div>
</li>
<li id="PDFOnline" title="Convert files to PDF Online">
<div class="PdfDocument e-icon convertToPdf "></div>
</li>
<li id="Signature" title="Sign, add text or send a document for
signature">
<div class=" PdfDocument e-icon signature "></div>
</li>
<li id="Save" title="Save file ( Ctrl+S )">
<div class=" PdfDocument e-icon save "></div>
</li>
<li id="Print" title="Print file ( Ctrl+P ) ">
<div class=" PdfDocument e-icon print "></div>
</li>
<li id="Message" title="Message">
<div class=" PdfDocument e-icon msg "></div>
</li>
</ul>
</div>
@Html.EJ().Toolbar("ToolbarItem").Width("auto").EnableSeparator(true).Hei
ght("33px")
```

Apply the given styles in the code table to show the Toolbar items as follows. You can refer images from any location.

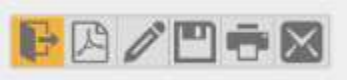
CSS

```
<style type="text/css" class="cssStyles">
.e-tooltxt .e-icon
{
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-
icon.png');
background-repeat: no-repeat;
display: block;
height: 30px;
width: 30px;
}
.convertToOthers
{
background-position: -349px 0px;
}
.convertToPdf
{
background-position: -527px 0px;
}
.signature
{
background-position: 2px 0px;
}
.save
{
background-position: -87px 0px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.print
{
background-position: -43px 0px;
}
.msg
{
background-position: -483px 0px;
}
</style>
```

Execute the code to render a Toolbar with a list of Toolbar items.



Toolbar with list of toolbar items

Render remaining Toolbar Items

To achieve the requirements, you need to render all the Toolbar items. You can separate or group the Toolbar items. The separation or grouping of Toolbar items is achieved when you give Toolbar items as a list of <UL> <LI> values inside the toolbar or span element. From the following sections, you can learn how to initialize the remaining Toolbar items with <UL> <LI> template and how to group the toolbar items.

Initialize the Toolbar items with <UL> <LI> template as follows.

ASPX-CS

```
<div id="ToolbarItem">
<!--Initialize toolbar items from above code snippet -->
<!-- Separator will be added at the end of each ul inside the toolbar
element-->
<!-- list of Remaining toolbar items with item separator -->
<ul>
<li id="Previous" title="Show previous page ( Left Arrow )">
<div class=" PdfDocument e-icon previous "></div>
</li>
<li id="Next" title="Show next page ( Right Arrow )">
<div class="PdfDocument e-icon next "></div>
</li>
<li id="page">
<div class="PdfDocument">
<input type="text" value="1" />
</div>
</li>
<li id="count">
<span>/ 1</span></li>
</ul>
<ul>
<li id="ZoomOut" title="Zoom Out">
<div class=" PdfDocument e-icon zoomOut "></div>
</li>
<li id="ZoomIn" title="Zoom In">
<div class=" PdfDocument e-icon zoomIn "></div>
</li>
<li id="ZoomValue">
```

```
<div id="car">
<ul>
<li>10%</li>
<li>25%</li>
<li>50%</li>
<li>100%</li>
<li>400%</li>
<li>800%</li>
<li>1600%</li>
<li>3200%</li>
<li>6400%</li>
</ul>
</div>
@Html.EJ().DropDownList("car").Width("90").Value("100%").TargetID("car")
</li>
</ul>
<ul>
<li id="FitFull" title="Fit one full page to window">
<div class=" PdfDocument e-icon fitOne "></div>
</li>
<li id="StickyNote" title="Add stick note ( Ctrl+6 ) ">
<div class=" PdfDocument e-icon sticky "></div>
</li>
<li id="ReadMode" title="View File in Read Mode">
<div class=" PdfDocument e-icon readMode "></div>
</li>
</ul>
</div>
```

Add the following styles in the code table to display the Toolbar items as follows.

CSS

```
/*Additional style for Remaining toolbar items*/
<style>
.zoomIn
{
background-position: -175px 0px;
}
.readMode
{
background-position: -307px -1px;
}
.zoomOut
{
background-position: -219px 0px;
}
.fitOne
{
background-position: -264px 0px;
}
.sticky
{
background-position: -131px -1px;
}
#ZoomValue.e-tooltxt .e-icon
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
background-image: none;
}
#page .PdfDocument input
{
text-align: center;
width: 20px;
height: 21px;
}
#count span
{
width: 30px;
height: 30px;
position: relative;
top: 2px;
text-align: center;
vertical-align: middle;
background-image: none;
}
.PdfDocument.e-icon.previous
{
background-position: -395px 0px;
}
.PdfDocument.e-icon.next
{
background-position: -439px 0px;
}
</style>
```

Set Zoom value is one of the items in the Toolbar. You need to render the DropDownList control for select zoom value. DropDownList control is rendered with <UL> <LI> elements. The ASP.NET MVCDropdown control with a list of zoom values is used to render Set Zoom value in the above sample code. Refer to the provided link for Dropdown creation.

Dropdown – GettingStarted

Execute the code to render Toolbar items with separator.



Add Actions to Toolbar Items

Now that the Toolbar is rendered, you need to render the header and content area to create a PDF Reader. In the following section, you can learn how to render the header (Toolbar), content section (PDF viewer area) and how to set the action to Toolbar items.

Note: PDF reading or rendering is not shown here. Simulation of the PDF Reader app to demonstrate the usage of Toolbar control is provided. PDF rendering area is ignored._

Initialize the content area and header as specified in the code table.

ASPX-CS

```
<!—"control" class used for aligns the pdf reader in center of a page. --
>
```

Copyright © 2001 - 2020 Syncfusion Inc.

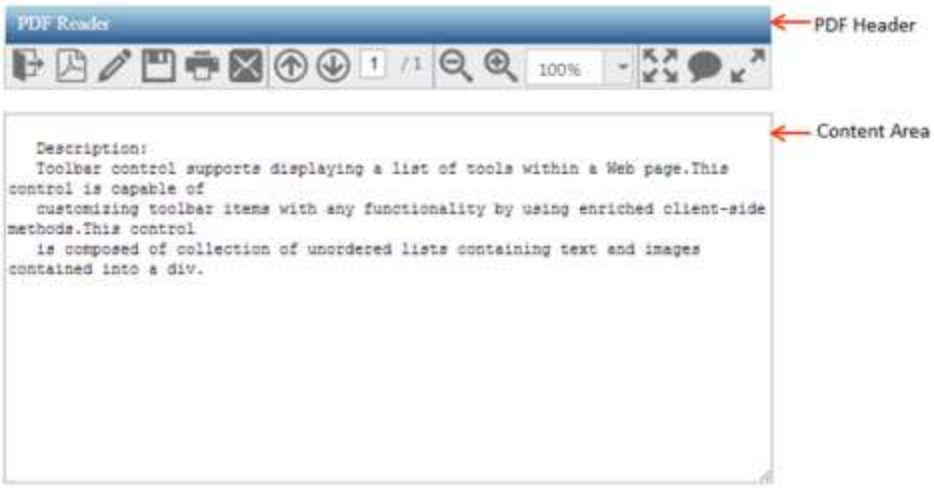
```
<div class="control">
<div class="label"></div>
<div id="ToolbarItem">
<!--Add toolbar items from above code snippet -->
</div>
<!-- Here Initialize the Toolbar items as like above code sample -->
<div id="contentSection">
<textarea id="content" rows="10" cols="30">
Description:
Toolbar control supports displaying a list of tools within a Web page.
This control is capable of
customizing toolbar items with any functionality by using enriched
client-side methods.This control
Is composed of collection of unordered lists containing text and images
contained into a div.
</textarea>
</div>
</div>
```

You can apply the following styles with the above styles to design the PDF header and content area.

CSS

```
<style type="text/css" class="cssStyles">
#content
{
float: left;
height: 300px;
width: 633px;
position: absolute;
}
.control
{
position: relative;
}
.label
{
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-
header.png');
background-repeat: no-repeat;
width: 638px;
height: 32px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



PDF Reader Appearance

So far, you have added the required toolbar items and configured its appearance. When you click on

Toolbar items, the operation is performed through client-slide click event. The following code example explains how to perform operations, when you click on the Toolbar items.

ASPX-CS

```
<div id="ToolbarItem">
<!--Add toolbar items from above code snippet -->
</div>
<!-- Here Initialize the Toolbar items as shown in above code sample and add Client side Event as follows -->
@Html.EJ().Toolbar("ToolbarItem").ClientSideEvents(e =>
e.Click("onItemClick")).Width("auto").EnableSeparator(true)
<script type="text/javascript">
function onItemClick(args)
{
//Finds Out the Item that was Clicked in Toolbar
//args.currentTarget returns the clicked Toolbar element
var option = args.currentTarget.id; /* Find Out the Id of Clicked item.
*/
switch (option)
{
case "OtherFormat":
//writes a code for Convert pdf files to Other format.
case "PdfOnline":
//writes a code for Convert files to Pdf online.
case "Signature":
//writes a code for Send a document for signature.
case "Save":
//writes a code for Save content.
case "Print":
//writes a code for Print content.
case "Message":
//writes a code for Send a Message.
case "Previous":
//writes a code for Show previous page.
case "Next":
//writes a code for Show Next page.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
case "ZoomOut":
//writes a code for Zoom out the page.
case "ZoomIn":
//writes a code for Zoom In the page.
case "FitFull":
//writes a code for Fit one full page to window.
case "StickyNote":
//writes a code for Add Sticky Note.
case "ReadMode":
//writes a code for view file in read mode.
}
}
</script>
```

Data Binding

Toolbar provides you a flexible approach for binding data from various data sources. There are various properties in Toolbar for Data Binding.

Data fields and configuration

The following sub-properties provides a way to bind either the local/remote data to the Toolbar control.

Property Table of MVC Toolbar control

Property	Value type	Description
Datasource	object	It specifies the data source of the Toolbar. The data source contains the list of data for generating the Toolbar items.
Fields	object	It specifies the mapping fields for the data items of the Toolbar.
Id	string	It specifies the id of the element
Text	string	It specifies the text content of the element.
ImageUrl	string	This property defines the imageURL for the image location. While setting images, the folder name in which the images are stored should be set to imageUrl property.
ImageAttributes	string	This property defines style for the image. While setting an image, styles can be applied such as height, width by using this property.
SpriteCssClass	string	This property sets the Sprite CSS for the image element in Toolbar.
HtmlAttributes	object	This property sets the HTML attribute for the Toolbar item. It can be any HTML attribute such as id, class, style.
TooltipText	string	This property sets the text value for Toolbar item while mouse over in Toolbar.
Query	object	It specifies the query to retrieve the data from online server.
Group	string	It groups the given Toolbar items.

Copyright © 2001 - 2020 Syncfusion Inc.

Local data

To bind the Local Data to the Toolbar control, map the user-defined JSON data names with its appropriate data source field. You can bind data to Toolbar by mapping fields such as Id, SpriteCssClass, ImageUrl, group and TooltipText.

The following steps explain how you can bind local data to Toolbar Control.

In the View page, add Toolbar helper to configure Toolbar.

C#

```
// Add the following data list to be bind in the controller page and
define the corresponding data.
// Define local data source elements with  fields
public class ToolbarLocalBinding
{
public string IconId { get; set; }
public string SpriteCss { get; set; }
public string Tooltip { get; set; }
}
//Refer the Model in the controller
using <Applicationname>.Models;
public ActionResult Index()
{
List<ToolbarLocalBinding> tools = new List<ToolbarLocalBinding>();
tools.Add(new ToolbarLocalBinding { IconId = "1", SpriteCss =
"LeftAlign_tool", Tooltip = "left" });
tools.Add(new ToolbarLocalBinding { IconId = "2", SpriteCss =
"CenterAlign_tool", Tooltip = "center" });
tools.Add(new ToolbarLocalBinding { IconId = "3", SpriteCss =
"RightAlign_tool", Tooltip = "right" });
tools.Add(new ToolbarLocalBinding { IconId = "4", SpriteCss =
"Justify_tool", Tooltip = "justify" });
tools.Add(new ToolbarLocalBinding { IconId = "5", SpriteCss =
"Bold_tool", Tooltip = "bold" });
tools.Add(new ToolbarLocalBinding { IconId = "6", SpriteCss =
"Italic_tool", Tooltip = "italic" });
tools.Add(new ToolbarLocalBinding { IconId = "7", SpriteCss =
"StrikeThrough_tool", Tooltip = "strike" });
tools.Add(new ToolbarLocalBinding { IconId = "8", SpriteCss =
"Underline_tool", Tooltip = "underline" });
ViewBag.datasource = tools;
return View();
}
```

ASPX-CS

```
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<MVCSam
ples.Models.ToolbarLocalBinding>)
ViewBag.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip"))
.Orientation(Orientation.Horizontal)
</div>
```

CSS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<style type="text/css" class="cssStyles">
.darktheme .cols-sample-area .e-tooltxt .ToolbarItems
{
background-image: url('../images/toolbar/ui-icons-metro.png');
}
.cols-sample-area .e-tooltxt .ToolbarItems
{
display: block;
background-image: url('../images/toolbar/ui-icons-dark.png');
height: 22px;
width: 22px;
}
.e-tooltxt:hover .ToolbarItems, .darktheme .cols-sample-area .e-
tooltxt:hover .ToolbarItems
{
background-image: url('../images/toolbar/ui-icons-light.png');
}
.ToolbarItems.LeftAlign_tool
{
background-position: -26px -39px;
}
.ToolbarItems.CenterAlign_tool
{
background-position: -55px -39px;
}
.ToolbarItems.RightAlign_tool
{
background-position: -89px -39px;
}
.ToolbarItems.Justify_tool
{
background-position: -123px -39px;
}
.ToolbarItems.Bold_tool
{
background-position: -159px -39px;
}
.ToolbarItems.Italic_tool
{
background-position: -196px -39px;
}
.ToolbarItems.StrikeThrough_tool
{
background-position: -55px -70px;
}
.ToolbarItems.Underline_tool
{
background-position: -23px -68px;
}
.html
{
background-color: yellowgreen;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



ToolBar control bounded to Local Data

Remote data

You can bind Toolbar to Remote Data using DataManager and the query in fields is used to retrieve the data. DataManager supports the following types of data-binding: JSON, Web Services and OData. It uses two different classes; ej.DataManager for processing, and ej.Query for serving data. ej.DataManager communicates with data source and ej.Query generates data queries that are read by the DataManager. In the following link, how to create DataManager is explained in full detail.

http://help.syncfusion.com/aspnetmvc/toolbar/data-binding

The following steps explain how you can bind remote data to Toolbar control.

1. In the View page, add Toolbar helper to configure Toolbar.

ASPX-CS

```
@Html.EJ().Toolbar("toolbar").Datasource(dataSource =>
dataSource.URL("http://mvc.syncfusion.com/Services/Northwnd.svc/")).Query
("ej.Query().from('Orders').take(6)").ToolbarFields(f =>
f.Text("CustomerID")).Orientation(Orientation.Horizontal).Width("340")
```



ToolBar control bounded to Remote Data

Behavior settings

The following are some miscellaneous properties that enables you to change the behavior of Toolbar control.

Enabling Toolbar

Enabled property is Boolean type, which allow us to enable or disable the Toolbar control. By default Enabled value is true. You can specify the property Enabled in the script as follows.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource)
.ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).Enable
d(false)
</div>
```

The following screenshot illustrates a Toolbar with Disable mode.



ToolBar control in Enabled (false)

Hiding Toolbar

The Hide property is Boolean type, which allow us to show or hide the Toolbar. Default value of Hide is false. You can specify the property Hide in the script as follows.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource)
.ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).Hide(t
rue)
</div>
```

Disable Or Enable Separate Toolbar Item

We can enable or disable a separate toolbar item by using the following methods.

Disable Item

The **Toolbar** method **disableItem** and **disableItemByID** can be used to disable separate toolbar item. In the below code we have disabled the third toolbar item by using these two methods

HTML

```
@Html.EJ().Toolbar("editingToolbar").Items(s =>
{
s.Add().SpriteCssClass("e-icon e-cut_01").TooltipText("Cut");
s.Add().SpriteCssClass("e-icon e-copy_02").TooltipText("Copy");
s.Add().SpriteCssClass("e-icon e-paste_01").TooltipText("Paste");
}).Items(s1 =>
{
s1.Add().SpriteCssClass("e-icon e-bold_01").TooltipText("Bold");
s1.Add().SpriteCssClass("e-icon e-
underline_01").TooltipText("Underline");
s1.Add().SpriteCssClass("e-icon e-
strikethrough_01").TooltipText("StrikeThrough");
}).Items(s2 =>
{
s2.Add().SpriteCssClass("e-icon e-align-left_01").TooltipText("Left");
s2.Add().SpriteCssClass("e-icon e-align-
center_01").TooltipText("Center");
s2.Add().SpriteCssClass("e-icon e-align-right_01").TooltipText("Right");
s2.Add().SpriteCssClass("e-icon e-align-
justify_01").TooltipText("Justify");
}).ClientSideEvents(e=>e.Create("create"))
```

JAVASCRIPT

```
function create() {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
var obj = $("#editingToolbar").data("ejToolbar");
obj.disableItem($(".e-toolbar li")[2]);
}
```

OR

JAVASCRIPT

```
function create() {
var obj = $("#editingToolbar").data("ejToolbar");
obj.disableItemByID("paste");
}
```



Enable Item

The **Toolbar** method **enableItem** and **enableItemByID** can be used to enable separate toolbar item. In the below code we have disabled the first five items initially and enabled the third toolbar item by using these two methods

HTML

```
@Html.EJ().Toolbar("editingToolbar").DisabledItemIndices(new List<int>
{0,1,2,3,4}).Items(s =>
{
s.Add().SpriteCssClass("e-icon e-cut_01").TooltipText("Cut");
s.Add().SpriteCssClass("e-icon e-copy_02").TooltipText("Copy");
s.Add().SpriteCssClass("e-icon e-paste_01").TooltipText("Paste");
}).Items(s1 =>
{
s1.Add().SpriteCssClass("e-icon e-bold_01").TooltipText("Bold");
s1.Add().SpriteCssClass("e-icon e-
underline_01").TooltipText("Underline");
s1.Add().SpriteCssClass("e-icon e-
strikethrough_01").TooltipText("StrikeThrough");
}).Items(s2 =>
{
s2.Add().SpriteCssClass("e-icon e-align-left_01").TooltipText("Left");
s2.Add().SpriteCssClass("e-icon e-align-
center_01").TooltipText("Center");
s2.Add().SpriteCssClass("e-icon e-align-right_01").TooltipText("Right");
s2.Add().SpriteCssClass("e-icon e-align-
justify_01").TooltipText("Justify");
}).ClientSideEvents(e=>e.Create("create"))
```

JAVASCRIPT

```
function create() {
var obj = $("#editingToolbar").data("ejToolbar");
obj.enableItem($(".e-toolbar li")[2]);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
```

OR

JAVASCRIPT

```
function create() {
var obj = $("#editingToolbar").data("ejToolbar");
obj.enableItemByID("paste");
}
```



Orientation

The Toolbar control supports both vertical and horizontal orientations, allowing it to fit into any scenario. The Orientation property of Toolbar defines the orientation in which the control is rendered. Set the value to this property as enum or string type. It accepts the following values.

- Syncfusion.JavaScript.Orientation.Horizontal
- Syncfusion.JavaScript.Orientation.Vertical

The following section explains you on how to set orientation for the toolbar.

Horizontal

By default the Orientation property sets the Toolbar in horizontal orientation. You can refer the following steps to set horizontal orientation for Toolbar control. Default value of Orientation is Horizontal.

1. In View, create UL-LI list of toolbar items and invoke the toolbar helper.
2. Add the following code in your CSHTML page.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).Orient
ation(Syncfusion.JavaScript.Orientation.Horizontal)
</div>
```

3. Build and run the application.

The following screenshot illustrates a Toolbar with horizontal orientation.



Copyright © 2001 - 2020 Syncfusion Inc.

Toolbar with Horizontal Orientation

Vertical

This property sets the Toolbar in vertical orientation. You can refer the following steps to set Vertical Orientation for Toolbar control.

1. Create UL-LI list of toolbar items and invoke the toolbar helper.
2. Add the following script in your HTML page.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).Orient
ation(Syncfusion.JavaScript.Orientation.Vertical)
</div>
```

3. Build and run the application.

The following screenshot illustrates a Toolbar with vertical orientation.



Toolbar with Vertical Orientation

Appearance and Styling

Adjusting Toolbar size

Height

The Height property is used to set height of the Toolbar. Set the value to this property as number or string type.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Height("300").Datasource((IEnu
merable<ToolbarLocalBinding>)ViewBag
.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip"))
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Width

The Width property is used to set width of the Toolbar. Set the value to this property as number or string type.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource)
.ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip"))
</div>
```

Enabling Rounded Corner

The ShowRoundedCorner property is Boolean type, which allows us to enable rounded corner for Toolbar control. Default value of this property is false. You can specify the property ShowRoundedCorner in the script as below,

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource)
.ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).ShowRo
undedCorner(true)
</div>
```



ToolBar control with rounded corner

Enabling Separator

The EnableSeparator property is Boolean type, which allow us to set separator between Toolbar items in Toolbar control. It separates one or more list items. When you want to separate two set of items, then define them in two separate ‘UL’ elements. Default value of EnableSeparator is false.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).Enable
Separator(true)
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.



ToolBar control with item separator

Themes

You can control the style and appearance of the Toolbar based on CSS classes. In order to apply styles to the Toolbar control, you can refer two files - ej.widgets.core.min.css and ej.theme.min.css. When you refer ej.widgets.all.min.css file, it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these two.

By default, there are 12 themes support available for Toolbar control namely

- default-theme
- flat-azure-dark
- fat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark

CssClass

The CssClass property is used to set root class for Toolbar control theme. Set the value to this property as string type.

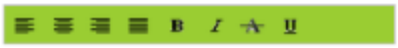
ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)ViewBag.datasource)
.ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).CssCla
ss("gradient-lime")
</div>
```

CSS

```
<style>
.gradient-lime
{
background-color: yellowgreen;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



ToolBar control with CssClass

Template Support

Template allows you to insert custom or ASP.NET MVC controls inside the toolbar items. Also you can design simple drop down buttons listing the items and radio button inside the Toolbar.

Set the list for DropDown control inside a list element and define this element as a Toolbar item. You can use all simple controls as a ToolBar item. For example to add RadioButton and DropDownList to Toolbar, use the following code example.

ASPX-CS

```
/ / Add this code in your CSHTML page and refer local data section for
data source
@Html.EJ().Toolbar("toolbar").Items(s =>
{
s.Add().ContentTemplate(@<div>
@Html.EJ().RadioButton("radio1").Checked(false)
</div>);
s.Add().ContentTemplate(@<div>
@Html.EJ().DropDownList("select").Height("23").Width("100").TargetID("car
s").SelectedItemIndex(0)
<div id="cars">
<ul>
<li>Audi A4</li>
<li>Audi A5</li>
<li>Audi A6</li>
<li>Audi A7</li>
</ul>
</div>
</div>);
})
```

The following screenshot displays a Toolbar with embedded controls.



Toolbar with Template

RTL

EnableRTL property is Boolean type, which allow us to change the left-to-right alignment of the Toolbar to right-to-left (RTL) that sets the Toolbar to do its actions from right to left. Default value of EnableRTL is false. You can specify the EnableRTL property in the script as below,

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)
ViewBag.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip")).Enable
RTL(true)
</div>
```



Toolbar from RTL

Keyboard Navigation

The entire Toolbar commands can be accessed through the keyboard by specifying the Keyboard Shortcut listed in the following table.

List of keyboard shortcuts

Keyboard Shortcut	Function
Alt + j	Focuses the control
UP	Navigates up and left.
Down	Navigates down and right.
Left	Navigates up and left.
Right	Navigates down and right.
Home	Navigates to the starting item.
End	Navigates to the ending item.
Enter	Selects the focused item

The following code example illustrates shortcuts associated with the Toolbar items.

ASPX-CS

```
// Add this code in your CSHTML page and refer local data section for
data source
<div class="cols-sample-area">
@Html.EJ().Toolbar("toolbar").Width("250").Datasource((IEnumerable<Toolba
rLocalBinding>)
ViewBag.datasource).ToolbarFields(f =>
f.ID("IconId").SpriteCssClass("SpriteCss").TooltipText("Tooltip"))
</div>
```



ToolBar control with Keyboard shortcuts

Copyright © 2001 - 2020 Syncfusion Inc.

Responsive Layout

Responsive Layout is aimed at crafting sites to provide an optimal viewing experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from mobile phones to desktop computer monitors). You can achieve Responsive Layout by using default functionality of Toolbar with IsResponsive as true and also you need to give Toolbar Width as in percentage value and add ej.responsive.css file in this sample. CDN link for the responsive CSS file is as follows.

http://cdn.syncfusion.com/13.1.0.21/js/web/responsive-css/ej.responsive.css

Add the above css link in the code sample.

1. Add the following code example in your HTML page.

ASPX-CS

```
<div class="control">
@Html.EJ().Toolbar("Toolbar").Width("65%").Height("33px").IsResponsive(tr
ue).Items(s =>
{
s.Add().Id("OtherFormat").SpriteCssClass("PdfDocument e-icon
convertToOthers ").TooltipText("Convert PDF files to Word or Excel
Online…");
s.Add().Id("PDFOnline").SpriteCssClass("PdfDocument e-icon convertToPdf
").TooltipText("Convert files to PDF Online");
s.Add().Id("Signature").SpriteCssClass("PdfDocument e-icon signature
").TooltipText("Sign, add text or send a document for signature");
s.Add().Id("Save").SpriteCssClass("PdfDocument e-icon save
").TooltipText("Save file ( Ctrl+S )");
s.Add().Id("Print").SpriteCssClass("PdfDocument e-icon print
").TooltipText("Print file ( Ctrl+P )");
s.Add().Id("Message").SpriteCssClass("PdfDocument e-icon msg
").TooltipText("Message");
}).Items(s1 =>
{
s1.Add().Id("Previous").SpriteCssClass("PdfDocument e-icon previous
").TooltipText("Show previous page ( Left Arrow )");
s1.Add().Id("Next").SpriteCssClass("PdfDocument e-icon next
").TooltipText("Show next page ( Right Arrow )");
s1.Add().Id("page").SpriteCssClass("PdfDocument
").ContentTemplate(@<input type="text" value="1" />);
s1.Add().Id("count").ContentTemplate(@<span>/1</span>);
}).Items(s2 =>
{
s2.Add().Id("ZoomOut").SpriteCssClass("PdfDocument e-icon zoomOut
").TooltipText("Zoom Out");
s2.Add().Id("ZoomIn").SpriteCssClass("PdfDocument e-icon zoomIn
").TooltipText("Zoom In");
s2.Add().Id("ZoomValue").SpriteCssClass("PdfDocument
").ContentTemplate(@<div>
@dropdown()
</div>);
}).Items(s3 =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
s3.Add().Id("FitFull").SpriteCssClass("PdfDocument e-icon fitOne
").TooltipText("Fit one full page to window");
s3.Add().Id("StickyNote").SpriteCssClass("PdfDocument e-icon sticky
").TooltipText("Add stick note ( Ctrl+6 ) ");
s3.Add().Id("ReadMode").SpriteCssClass("PdfDocument e-icon readMode
").TooltipText("View File in Read Mode");
})
@helper dropdown()
{
<div id="percent">
<ul>
<li>10%</li>
<li>25%</li>
<li>50%</li>
<li>100%</li>
<li>400%</li>
<li>800%</li>
<li>1600%</li>
<li>3200%</li>
<li>6400%</li>
</ul>
</div>
@Html.EJ().DropDownList("selectPercent").Width("90").Value("100%").Target
ID("percent")
}
</div>
```

CSS

```
<style type="text/css" class="cssStyles">
.e-tooltxt .PdfDocument.e-icon
{
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-
icon.png');
background-repeat: no-repeat;
display: block;
height: 30px;
width: 30px;
}
.e-tooltxt .PdfDocument.e-icon:hover
{
background-image: url('http://js.syncfusion.com/UG/Web/Content/pdf-icon-
white.png');
}
.PdfDocument.e-icon.convertToOthers
{
background-position: -349px 0px;
}
.PdfDocument.e-icon.convertToPdf
{
background-position: -527px 0px;
}
.PdfDocument.e-icon.signature
{
background-position: 2px 0px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.PdfDocument.e-icon.save
{
background-position: -87px 0px;
}
.PdfDocument.e-icon.msg
{
background-position: -483px 0px;
}
.PdfDocument.e-icon.previous
{
background-position: -395px 0px;
}
.PdfDocument.e-icon.next
{
background-position: -439px 0px;
}
.PdfDocument.e-icon.zoomIn
{
background-position: -175px 0px;
}
.PdfDocument.e-icon.zoomOut
{
background-position: -219px 0px;
}
.PdfDocument.e-icon.fitOne
{
background-position: -264px 0px;
}
.PdfDocument.e-icon.sticky
{
background-position: -131px -1px;
}
.PdfDocument.e-icon.readMode
{
background-position: -308px 0px;
}
.PdfDocument.e-icon.print
{
background-position: -43px 0px;
}
#ZoomValue .PdfDocument
{
width: 90px;
}
#page .PdfDocument input
{
text-align: center;
width: 20px;
height: 21px;
}
#count span
{
width: 30px;
height: 30px;
position: relative;
top: 2px;
text-align: center;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
vertical-align: middle;
}
</style>
```

Execute the above code to render the following output.



responsiveType:Inline

We can set the responsiveType property to inline to display the overflown toolbar items as inline toolbar.

ASPX-CS

```
@Html.EJ().Toolbar("Toolbar").Width("65%").Height("33px").IsResponsive(tr
ue).ResponsiveType(ResponsiveType.Inline).Items(s =>
{
s.Add().Id("OtherFormat").SpriteCssClass("PdfDocument e-icon
convertToOthers ").TooltipText("Convert PDF files to Word or Excel
Online…");
s.Add().Id("PDFOnline").SpriteCssClass("PdfDocument e-icon convertToPdf
").TooltipText("Convert files to PDF Online");
s.Add().Id("Signature").SpriteCssClass("PdfDocument e-icon signature
").TooltipText("Sign, add text or send a document for signature");
s.Add().Id("Save").SpriteCssClass("PdfDocument e-icon save
").TooltipText("Save file ( Ctrl+S )");
s.Add().Id("Print").SpriteCssClass("PdfDocument e-icon print
").TooltipText("Print file ( Ctrl+P )");
s.Add().Id("Message").SpriteCssClass("PdfDocument e-icon msg
").TooltipText("Message");
}).Items(s1 =>
{
s1.Add().Id("Previous").SpriteCssClass("PdfDocument e-icon previous
").TooltipText("Show previous page ( Left Arrow )");
s1.Add().Id("Next").SpriteCssClass("PdfDocument e-icon next
").TooltipText("Show next page ( Right Arrow )");
s1.Add().Id("page").SpriteCssClass("PdfDocument
").ContentTemplate(@<input type="text" value="1" />);
s1.Add().Id("count").ContentTemplate(@<span>/1</span>);
}).Items(s2 =>
{
s2.Add().Id("ZoomOut").SpriteCssClass("PdfDocument e-icon zoomOut
").TooltipText("Zoom Out");
s2.Add().Id("ZoomIn").SpriteCssClass("PdfDocument e-icon zoomIn
").TooltipText("Zoom In");
s2.Add().Id("ZoomValue").SpriteCssClass("PdfDocument
").ContentTemplate(@<div>
@dropdown()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>);
}).Items(s3 =>
{
s3.Add().Id("FitFull").SpriteCssClass("PdfDocument e-icon fitOne
").TooltipText("Fit one full page to window");
s3.Add().Id("StickyNote").SpriteCssClass("PdfDocument e-icon sticky
").TooltipText("Add stick note ( Ctrl+6 ) ");
s3.Add().Id("ReadMode").SpriteCssClass("PdfDocument e-icon readMode
").TooltipText("View File in Read Mode");
})
@helper dropdown()
{
<div id="percent">
<ul>
<li>10%</li>
<li>25%</li>
<li>50%</li>
<li>100%</li>
<li>400%</li>
<li>800%</li>
<li>1600%</li>
<li>3200%</li>
<li>6400%</li>
</ul>
</div>
@Html.EJ().DropDownList("selectPercent").Width("90").Value("100%").Target
ID("percent")
}
```

The following output will be displayed while executing the above code:



responsiveType:Popup

We can set the responsiveType property to popup to display the overflown toolbar items as popup. This the default value of enum responsiveType.

ASPX-CS

```
@Html.EJ().Toolbar("Toolbar").Width("65%").Height("33px").IsResponsive(tr
ue).ResponsiveType(ResponsiveType.Popup).Items(s =>
{
s.Add().Id("OtherFormat").SpriteCssClass("PdfDocument e-icon
convertToOthers ").TooltipText("Convert PDF files to Word or Excel
Online…");
s.Add().Id("PDFOnline").SpriteCssClass("PdfDocument e-icon convertToPdf
").TooltipText("Convert files to PDF Online");
s.Add().Id("Signature").SpriteCssClass("PdfDocument e-icon signature
").TooltipText("Sign, add text or send a document for signature");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
s.Add().Id("Save").SpriteCssClass("PdfDocument e-icon save
").TooltipText("Save file ( Ctrl+S )");
s.Add().Id("Print").SpriteCssClass("PdfDocument e-icon print
").TooltipText("Print file ( Ctrl+P )");
s.Add().Id("Message").SpriteCssClass("PdfDocument e-icon msg
").TooltipText("Message");
}).Items(s1 =>
{
s1.Add().Id("Previous").SpriteCssClass("PdfDocument e-icon previous
").TooltipText("Show previous page ( Left Arrow )");
s1.Add().Id("Next").SpriteCssClass("PdfDocument e-icon next
").TooltipText("Show next page ( Right Arrow )");
s1.Add().Id("page").SpriteCssClass("PdfDocument
").ContentTemplate(@<input type="text" value="1" />);
s1.Add().Id("count").ContentTemplate(@<span>/1</span>);
}).Items(s2 =>
{
s2.Add().Id("ZoomOut").SpriteCssClass("PdfDocument e-icon zoomOut
").TooltipText("Zoom Out");
s2.Add().Id("ZoomIn").SpriteCssClass("PdfDocument e-icon zoomIn
").TooltipText("Zoom In");
s2.Add().Id("ZoomValue").SpriteCssClass("PdfDocument
").ContentTemplate(@<div>
@dropdown()
</div>);
}).Items(s3 =>
{
s3.Add().Id("FitFull").SpriteCssClass("PdfDocument e-icon fitOne
").TooltipText("Fit one full page to window");
s3.Add().Id("StickyNote").SpriteCssClass("PdfDocument e-icon sticky
").TooltipText("Add stick note ( Ctrl+6 ) ");
s3.Add().Id("ReadMode").SpriteCssClass("PdfDocument e-icon readMode
").TooltipText("View File in Read Mode");
})
@helper dropdown()
{
<div id="percent">
<ul>
<li>10%</li>
<li>25%</li>
<li>50%</li>
<li>100%</li>
<li>400%</li>
<li>800%</li>
<li>1600%</li>
<li>3200%</li>
<li>6400%</li>
</ul>
</div>
@Html.EJ().DropDownList("selectPercent").Width("90").Value("100%").Target
ID("percent")
}
```

The following output will be displayed while executing the above code:

Copyright © 2001 - 2020 Syncfusion Inc.



How To

Render Toolbar from code behind

Toolbar can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of Toolbar using Toolbar properties from code behind.

RAZOR

```
@{
Html.EJ().Toolbar("toolbar",
(Syncfusion.JavaScript.Models.ToolbarProperties)ViewData["toolbar"]).Rend
er();
}
```

C#

```
public ActionResult Index()
{
List<ToolbarBaseItem> obj = new List<ToolbarBaseItem>();
obj.Add(new ToolbarBaseItem() { SpriteCssClass= "e-icon e-cut"
,TooltipText="cut"});
obj.Add(new ToolbarBaseItem() { SpriteCssClass = "e-icon e-copy",
TooltipText = "copy" });
obj.Add(new ToolbarBaseItem() { SpriteCssClass = "e-icon e-paste",
TooltipText = "paste" });
ToolbarProperties toolbar = new ToolbarProperties();
toolbar.Items = obj;
ViewData["toolbar"] = toolbar;
return View();
}
```

Tooltip

Overview

The Tooltip control displays a pop-up hint when users hover, click, or focus on an element. The Tooltip will be positioned based on browser window, mouse, target element, or x and y coordinates. The Tooltip's layout can be customized to create a distinct visualized element.

Key Features

Responsive – Flexible positioning of Tooltip and can even reposition themselves when the viewport is resized. **Template** - The Tooltip's layout can be customized to create a distinct visualized element. *Animation – Custom animation effects can be used for showing/hiding the Tooltip* **RTL support** -

Provides a full-fledged right-to-left mode which aligns content in the Tooltip control from right to left. * **Accessibility** - Supports keyboard and ARIA accessibility.

Getting started

Create a Tooltip

Using the following steps, you can create a Tooltip control. The basic rendering of ASP.NET MVC Tooltip is achieved with default functionality.

1. You can create an MVC Project and add necessary assembly and script with the help of the given MVC-Getting Started Documentation.

2. Add the mentioned code to the corresponding view page for Tooltip rendering.

ASPX-CS

```
<div class="frame">
<div class="img" id="sample1">
<a target="_blank" href="image/taj.png">
<img
src="http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-
05.png" alt="Delphi">
</a>
<div class="desc">Delphi Succinctly</div>
</div>
</div>
@Html.EJ().Tooltip("sample1").Content("Learn the fundamentals of Delphi
to build a variety of solutions for many devices and platforms.")
```

Apply the following style sheet

CSS

```
<style>
div.img {
border: 1px solid #ccc;
width: 159px;
height: 213px;
left: 35%;
position: relative;
top: 20%;
}
div.img img {
width: 159px;
height: 179px;
}
div.desc {
padding: 8px;
text-align: center;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Setting Dimensions

Tooltip dimensions can be set using width and height API.

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip1"><u> JavaScript</u> </a>the way
you really want to.
</div>
@Html.EJ().Tooltip("tooltip1").Content("JavaScript is the programming
language of HTML and the Web.").Width("100px").Height("100px")
```

Tooltip Appearance

You can configure the appearance of the Tooltip with the title, close button and call out as your application requires.

ASPX-CS

```
<div class="img" id="sample">
<a target="_blank" href="image/taj.png">
<img
src="http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-
05.png" alt="Delphi">
</a>
<div class="desc">Delphi Succinctly</div>
</div>
@Html.EJ().Tooltip("sample").Content("Learn the fundamentals of Delphi to
build a variety of solutions for many devices and
platforms.").Width("180px").Title("Delphi
Succinctly").CloseMode(CloseMode.Sticky).IsBalloon(false)
```

Apply the following styles to show the Tooltip.

CSS

```
<style>
div.img {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
border: 1px solid #ccc;
float: left;
box-sizing: border-box;
height: 200px;
width: 146px;
}
div.img img{
width: 100%;
height: 166px;
}
div.desc {
padding: 6px;
text-align: center;
}
</style>
```



Customization

Template Support

By default you can add any text or image to the Tooltip. To customize the tooltip layout or to create your own visualized elements you can use this template support.

ASPX-CS

```
<div class="ctrl" id="centerImage">
<img class="ctrImg"
src="http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-
04.png" />
<div class="new">Roslyn Succinctly</div>
</div>
@Html.EJ().Tooltip("centerImage").Content("<div class='main'> <div
class='poster'> <img
src='http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
2.png' width='150px' height='120px'> </div> <div class='def'> <h4> Roslyn
Succinctly </h4><div class='description'>Microsoft has only recently
embraced the world of open source software, offering <a
href='#'>More...</a> </div>").Width("350px")
```

CSS

```
<style>
h4 {
margin-top: 0px;
margin-bottom: 2px;
}
.e-tooltip-wrap .e-tipContainer .e-tipcontent {
padding: 5px 0px;
}
.poster {
float: left;
padding: 4px 0px;
}
.new {
text-align: center;
}
.def {
float: right;
}
.ctrl {
border: 1px solid #ebebe0;
width: 150px;
padding: 5px;
height: 180px;
margin-top: 239px;
margin-left: 250px;
}
.ctrImg {
width: 150px;
height: 160px;
}
.category {
margin-left: 10px;
}
.description {
width: 200px;
height: 60px;
line-height: 22px;
margin-top: 10px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip's title customization

Tooltip title can be customized with the image or any HTML element.

ASPX-CS

```
<div class="ctrl" id="centerImage">
<img class="ctrImg"
src="http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-04.png" />
<div class="new">Roslyn Succinctly</div>
</div>
@Html.EJ().Tooltip("centerImage").Content("<div>Microsoft has only recently embraced the world of open source software, offering <a href='#'>More...</a> </div>").Title("<div><img class='titleImg' src='http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-2.png' /> <div class='description'> Roslyn Succinctly </div> </div> ")
```

CSS

```
<style>
.titleImg {
width: 20px;
height: 20px;
float: left;
margin-right: 10px;
}
#centerImage{
margin-left : 300px;
margin-top : 250px;
position : absolute;
border: 1px solid grey;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.description {
height: 20px;
}
</style>
```



Animation Effects

Determines the type of effect that takes place when showing/hiding the tooltip.

We can specify the effect and the duration for the animation.

Effects	Description
Slide	Sliding animation effect takes place with a duration of 200ms.
Fade	Fading animation effect takes place with a duration of 800ms.
None (Default)	No effect takes place

Let's create a Tooltip that slides down when shown using the animation property:

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip1"><u> JavaScript</u> </a>the way
you really want to.
</div>
@Html.EJ().Tooltip("tooltip1").Content("JavaScript is the programming
language of HTML and the Web.").Animation(animationEffect =>
animationEffect.Effect(Effect.Slide).Speed(1000))
```

Custom Animation

Custom animation effect for both Tooltip show/hide can also be done by show and hide method.

Copyright © 2001 - 2020 Syncfusion Inc.

Show or Hide method may receive an optional 'callback' parameter, which represents a function you'd like to call which will animate the tooltip.

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip2"><u> JavaScript</u> </a>the way
you really want to.
</div>
@Html.EJ().Button("open").Size(ButtonSize.Large).ShowRoundedCorner(true).
Text("Open").ClientSideEvents(events => events.Click("onClick"))
@Html.EJ().Tooltip("tooltip2").Content("JavaScript is the programming
language of HTML and the Web.")
<script>
var tip;
function onClick(args){
tip = $("#tooltip2").data("ejTooltip");
tip.show(null,"myFunc");
}
function myFunc(args) {
tip = $("#tooltip2").data("ejTooltip");
$(tip.tooltip).slideDown(200, "easeOutElastic");
}
</script>
```

Note: Show or Hide method can also receive an optional parameter “effect name”, (e.g any easing effect name) which specifies the type of effect taken place when showing/hiding of the tooltip, please refer to the following link for online demo - link.

Modernize the tooltip’s content

It's easy to update a tooltip’s content – whether it’s open or closed.

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip3"><u> JavaScript</u> </a>the way
you really want to.
@Html.EJ().Button("open").Size(ButtonSize.Large).ShowRoundedCorner(true).
Text("Update Content").ClientSideEvents(events =>
events.Click("onClick"))
</div>
@Html.EJ().Tooltip("tooltip3").Content("JavaScript is the programming
language of HTML and the Web.")
<script type="text/javascript">
function onClick(args){
tip = $("#tooltip3").data("ejTooltip");
tip.setModel({ content: "JavaScript" });
tip.show();
}
</script>
```

Closing Mode

By default, the Tooltip will be hidden when mouse leaves the target element. Different types of close mode as follows

Copyright © 2001 - 2020 Syncfusion Inc.

Types	Description
Auto	Tooltip will be hidden after a particular period of time.
Sticky	Tooltip rendered with the button, it will close the tooltip.
None (Default)	Tooltip will be hidden when mouse leaves the target element.

Auto

The tooltip will be visible only for the period of time specified in the autoCloseTimeout.

Let see an example, this Tooltip will only hide after hovering the target for 2000ms

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip4"><u> JavaScript</u> </a>the way
you really want to.
</div>
@Html.EJ().Tooltip("tooltip4").Content("JavaScript is the programming
language of HTML and the
Web.").CloseMode(CloseMode.Auto).AutoCloseTimeout(2000)
```

Note: Time specified in the autoCloseTimeout will be in milliseconds and the default value is 4000ms

Sticky

A close button will be shown with the Tooltip. The button element (i.e. close button) located by default at the top right of the Tooltip or title bar (if title is enabled). The tooltip gets closed when the button is clicked.

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip5"><u> JavaScript</u> </a>the way
you really want to.
</div>
@Html.EJ().Tooltip("tooltip5").Content("JavaScript is the programming
language of HTML and the
Web.").CloseMode(CloseMode.Sticky).Width("200px")
```

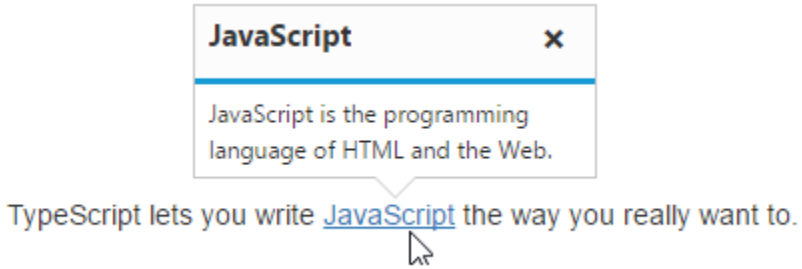


You can also have Tooltip with a title, in which case the button will lye within it:

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="tooltip6"><u> JavaScript</u> </a>the way
you really want to.
```

```
</div>
@Html.EJ().Tooltip("tooltip6").Content("JavaScript is the programming
language of HTML and the
Web.").CloseMode(CloseMode.Sticky).Width("200px").Title("JavaScript")
```



Position

The position object defines various attributes of the Tooltip position, including the element it is positioned in relation to, and how the position is adjusted within the defined container.

Lets position the Tooltips (stems) left center corner at the right center of the target element.

ASPX-CS

```
<div class="img" id="sample">
<a target="_blank" href="image/taj.png">
<img
src="http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-
05.png" alt="Delphi">
</a>
<div class="desc">Delphi Succinctly</div>
</div>
@Html.EJ().Tooltip("sample").Content("Learn the fundamentals of Delphi to
build a variety of solutions for many devices and
platforms.").Position(position => position.Target(target =>
target.Horizontal("right").Vertical("center")).Stem(stem =>
stem.Horizontal("left").Vertical("center")))
```

Apply the following styles to show the Tooltip.

CSS

```
<style>
div.img {
border: 1px solid #ccc;
float: left;
box-sizing: border-box;
height: 200px;
width: 146px;
}
div.img img{
width: 100%;
height: 166px;
}
div.desc {
padding: 6px;
text-align: center;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</style>
```



Note: By default, the Tooltips "center bottom" corner is placed at the center top of the target element.

Containment

Determines the HTML element in which the Tooltip is appended to e.g. its containing element and The tooltip will be restricted to move only within the specified container element.

Let's append our Tooltip to a custom 'frame' container:

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip1"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip1").Content("JavaScript is the programming
language of HTML and the Web.").Containment(".frame")
```

Note: By default all Tooltips are appended to the document.body element.

Associates

The Tooltip will be positioned in relation to target element. Can also be set to 'mouse' or the window, or an absolute x/y position on the page.

Let's position the Tooltip in relation to the 'a' element inside the div element:

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip2"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip2").Content("JavaScript is the programming
language of HTML and the Web.")
```

Copyright © 2001 - 2020 Syncfusion Inc.

We can also position the Tooltip in relation to the mouse.

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip3"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip3").Content("JavaScript is the programming
language of HTML and the Web.").Associate(Associate.Mousefollow)
```

Position the tooltip at the current mouse position, once enter to the target element as follows

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip4"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip4").Content("JavaScript is the programming
language of HTML and the Web.").Associate(Associate.Mouseenter)
```

It also possible to place the tooltip relation to the window as follows

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip5"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip5").Content("JavaScript is the programming
language of HTML and the Web.").Position(position =>
position.Target(target =>
target.Horizontal("right").Vertical("bottom"))).Associate(Associate.Windo
w)
```

And last but not least, absolute positioning via X,Y co-ordinates e.g. a Tooltip at 10px from left and top of the page:

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip6"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip6").Content("JavaScript is the programming
language of HTML and the Web.").Position(position =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
position.Target(target =>
target.Horizontal(10).Vertical(10))).Associate(Associate.Axis)
```

Collision

When the positioned element overflows the window in some direction, move it to an alternative position.

The following values determines the kind of positioning that takes place.

Value	Description
Flip	Flips the element to the opposite side of the target if the collision detected.
Fit	Shift the element away from the edge of the window.
FlipFit(Default)	Ensure as much of the element is visible as possible to showcase.
None	Does not apply any collision detection.

ASPX-CS

```
<div class="frame">
<div class="control">
TypeScript lets you write <a id="tooltip7"><u> JavaScript</u> </a>the way
you really want to.
</div>
</div>
@Html.EJ().Tooltip("tooltip7").Content("JavaScript is the programming
language of HTML and the Web.").Collision(Collision.Fit)
```

How To

Use AJAX to generate the Tooltip's content

Use jQuery's built-in AJAX functionality to retrieve remote content and set the content to the Tooltip as follows.

Define three employee's JSON Array as given below(an array of 3 employees records)

JS

```
[
{
"EmployeeID": 1,
"LastName": "Davolio",
"Title": "Sales Representative",
"TitleOfCourtesy": "Dr.",
"Address": "507 - 20th Ave. E.Apt. 2A",
"City": "Seattle",
"Region": "WA",
"PostalCode": "98122",
},
{
"EmployeeID": 2,
"LastName": "Buchanan",
"Title": "Sales Representative",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
"TitleOfCourtesy": "Dr.",
"Address": "908 W. Capital Way",
"City": "Tacoma",
"Region": "WA",
"PostalCode": "98401",
},
{
"EmployeeID": 3,
"LastName": "Leverling",
"Title": "Sales Representative",
"TitleOfCourtesy": "Dr.",
"Address": "722 Moss Bay Blvd.",
"City": "Kirkland",
"Region": "WA",
"PostalCode": "98033",
}
]
```

Render the employees table and create the tooltip. Once the tooltip created, the data will be fetched using AJAX method as follows.

ASPX-CS

```
<div class="frame">
<table id="details">
<tr>
<th>EmployeeID</th>
<th>Name</th>
<th>Designation</th>
</tr>
<tr>
<td>SF6089</td>
<td><a href="#" class="e-info" title="Davolio">Davolio</a></td>
<td>Software Engineer</td>
</tr>
<tr>
<td>SF6073</td>
<td> <a href="#" class="e-info" title="Leverling">Leverling </a> </td>
<td>Tester</td>
</tr>
<tr>
<td>SF6073</td>
<td> <a href="#" class="e-info" title="Suyama"> Suyama </a> </td>
<td>Content Writer</td>
</tr>
<tr>
<td>SF7896</td>
<td> <a href="#" class="e-info" title="Buchanan"> Buchanan </a> </td>
<td>Graphics Designer</td>
</tr>
</table>
</div>
@Html.EJ().Tooltip("details").Target(".e-info").ClientSideEvents(events
=>
events.BeforeOpen("onOpen")).Width("350px").Height("128px").Position(pos
=> pos.Target(target =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
target.Horizontal("right").Vertical("bottom")).Stem(stem =>
stem.Horizontal("left").Vertical("top"))).Content(" ")
<script type="text/javascript">
var content;
var target = $("#details").data("ejTooltip");
function onOpen(args) {
proxy = args;
$.ajax({
dataType: "json",
url: "tooltipData.js",
success: function (result) {
var employee = [
{ photo:
"http://mvc.syncfusion.com/demos/web/content/images/Employees/2.png" },
{ photo:
"http://mvc.syncfusion.com/demos/web/content/images/Employees/4.png" },
{ photo:
"http://mvc.syncfusion.com/demos/web/content/images/Employees/8.png" },
{ photo:
"http://mvc.syncfusion.com/demos/web/content/images/Employees/3.png" },
{ photo:
"http://mvc.syncfusion.com/demos/web/content/images/Employees/1.png" },
{ photo:
"http://mvc.syncfusion.com/demos/web/content/images/Employees/6.png" },
];
for (i = 0; i < result.length ; i++) {
if (result[i].LastName == $(proxy.event.target).attr("data-content"))
content = '<div class="main"> <img src=' + employee[i].photo + '
class="logo"/><div class="des"><table> <tr> <th> ' +
result[i].TitleOfCourtesy + ' ' + result[i].LastName + '</th> </tr> <tr>
<td> Title </td> <td>  : ' + result[i].Title + '</td> </tr>
<tr><td> Address</td><td>  : ' + result[i].Address + '</td>
</tr><tr> <td> City </td> <td>  : ' + result[i].City + '</td>
</tr> <tr> <td> PostalCode </td> <td>  : ' +
result[i].PostalCode + '</td> </tr> </table></div></div>';
}
target.setModel({ content: content });
}
});
target.show(args.event.target);
args.cancel = true;
}
</script>
```

Defines the style for the tooltip layout and table as follows.

CSS

```
<style>
.frame {
width: 100%;
box-sizing: border-box;
}
.frame table {
border-collapse: collapse;
width: 100%;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
.frame th, .frame td {
text-align: left;
padding: 8px;
}
.logo {
float: left;
width: 100px;
height: 114px;
}
.frame tr:nth-child(even) {
background-color: #f2f2f2;
}
.frame th {
background-color: #4CAF50;
color: white;
}
.des {
width: 230px;
float: right;
line-height: 24px;
}
</style>
```



Integration with the Slider control

Tooltip can also be integrated with various other jQuery plugin. Tooltip shows the slider value above the handle.

Render the slider control and finds its handle to render the Tooltip as follows

ASPX-CS

```
<div class="frame">
<div id="loanheading">
Details of Loan
</div>
<span class="ColumnLeft">
<span class="loan">Loan Amount</span>
</span>
<span class="ColumnRight">
```

```
<span class="value">1000</span><span id="loantext">$ </span>
</span>
@(Html.EJ().Slider("loanSlider").Height("16px").Value("100").MinValue(10)
.MaxValue(1000).IncrementStep(10)
.ClientSideEvents(evt =>
evt.Change("onSlide").Slide("onSlide").Stop("onStop")))
</div>
@Html.EJ().Tooltip("loanSlider").Target(".e-
handle").ShowRoundedCorner(true).IsBalloon(true)
<script>
$(function () {
var loanObj = $('#loanSlider').data("ejSlider");
var loadTip = $('#loanSlider').data("ejTooltip");
loadTip.setModel({content : "$ " + loanObj.getValue().toString() });
});
</script>
```

Once the handler in the slider, start its sliding the below function will be triggered

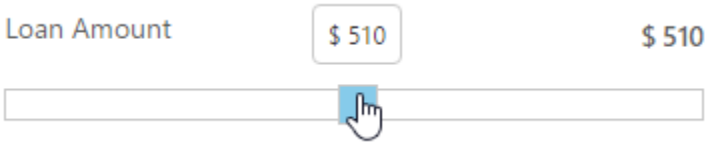
ASPX-CS

```
function onSlide(args) {
this.wrapper.prev().children('span.value').html(args.value);
var  value = args.value.toString();
if(args.id == "loanSlider")
value =  "$ " +value;
var tipWidth = (value < 10) ? "13px" : (10 <= value >= 100) ? "25px" :
"auto";
var tipObj = $('#loanSlider').data("ejTooltip");
tipObj.setModel({content : value, width : tipWidth});
tipObj.show($("div" + target).find(".e-handle"));
}
```

Once the sliding is stopped, the Tooltip will be shown for the particular period of time then it will be closed.

ASPX-CS

```
function onStop(args){
setTimeout( function(){
var tipObj = $('#loanSlider').data("ejTooltip");
tipObj.hide();
}, 1000);
}
```



Tip(arrow) customization

Styling the Tip's background and border colors is done using "cssClass" API of Tooltip. Change the size of the tip using the property name called "Tip" and CSS as follows.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
<div class="frame">
<div class="img" id="custom">
<img
src="http://mvc.syncfusion.com/demos/web/content/images/tooltip/template-
04.png" alt="Roslyn Succinctly" >
<div class="desc">Roslyn Succinctly</div>
</div>
</div>
@Html.EJ().Tooltip("custom").Content("Learn to use the Python language to
create programs of all kinds to creating practical
applications.").Width("200px").Position(pos => pos.Target(target =>
target.Horizontal("right").Vertical("bottom")).Stem(stem =>
stem.Horizontal("left").Vertical("top"))).Tip(tip => tip.Size(size =>
size.Height(12).Width(25)))
<script type="text/javascript">
$(function () {
target = $("#custom").data("ejTooltip");
});
</script>
```

Defines the style for the tip as follow as

CSS

```
<style>
div.img {
border: 1px solid #ccc;
width: 159px;
height: 213px;
left: 35%;
position: relative;
top: 20%;
}
div.img img{
width: 159px;
height: 179px;
}
div.desc {
padding: 8px;
text-align: center;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Initialize Tooltip for the target element

Tooltips are also useful for input elements, to show hint in the context of each field.

Rather than using multiple Tooltips, a single Tooltip can be used to show the information of each and every field of the form/table element. This can be achieved using “target” API. Target property Specifies a selector for elements, within the container, for which the Tooltip will be displayed.

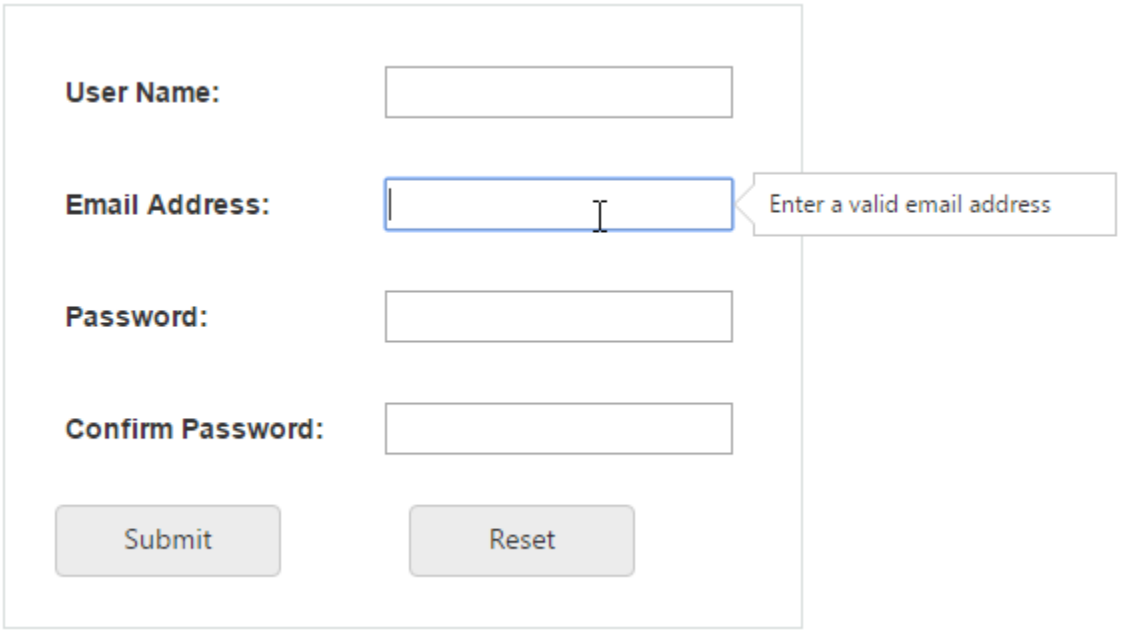
The example below demonstrates how to create a Tooltip for multiple targets element.

ASPX-CS

```
<form id="details" role="form">
<fieldset>
<table>
<tr>
<td class="info"> User Name:</td>
<td> <input type="text" class="e-info" name="firstname" title="Please
enter your name"> </td>
</tr>
<tr>
<td class="info"> Email Address:</td>
<td> <input type="text" class="e-info" name="email" title="Enter a valid
email address"></td>
</tr>
<tr>
<td class="info"> Password:</td>
<td> <input type="password" class="e-info" name="password" title="Be at
least 8 characters length"></td>
</tr>
<tr>
<td class="info"> Confirm Password:</td>
<td> <input type="password" class="e-info" name="Cpwd" title="Re-enter
your password"></td>
</tr>
</table>
@Html.EJ().Button("sample").Type(ButtonType.Submit).Text("Submit").ShowRo
undedCorner(true).Size(ButtonSize.Medium)
@Html.EJ().Button("clear").Type(ButtonType.Reset).Text("Reset").ShowRound
edCorner(true).Size(ButtonSize.Medium)
</fieldset>
</form>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Tooltip("details").Target(".e-
info").Height("30px").Width("180px").Position(pos => pos.Target(target =>
target.Horizontal("right").Vertical("center")).Stem(stem =>
stem.Horizontal("left").Vertical("center"))).ClientSideEvents(events =>
events.Create("onCreate").BeforeOpen("onOpen"))
```



Apply the following styles to the form element.

CSS

```
<style>
th, td {
padding: 15px;
text-align: left;
}
#form {
position: absolute;
left: 200px;
}
fieldset {
border: 1px solid #D9DEDD;
padding: 15px;
width: 400px;
}
.info {
font-weight: bold;
}
#sample {
margin-right: 50px;
}
.e-button {
margin: 10px;
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Interact with the Tooltip

Give users, the possibility to interact with the tooltip. If the tooltip is interactive and activated by a hover event, set the amount of time (milliseconds)allowed for a user to hover off of the tooltip activator on to the tooltip itself – keeping the tooltip from closing using “autoCloseTimeout” property.

Using this property, Links can be provided in Tooltip content where user can navigate to some other page. This is handy for situations where you need to grab the user attention.

ASPX-CS

```
<div class="control">
TypeScript lets you write <a id="test"><u> JavaScript</u> </a>the way you
really want to.
</div>
@Html.EJ().Tooltip("test").Content("JavaScript is the programming
language of HTML and the
Web.").CloseMode(CloseMode.Auto).AutoCloseTimeout(2000)
```

Render Tooltip from code behind

Tooltip can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of Tooltip using Tooltip properties from code behind.

RAZOR

```
<div class="control">
Alongside HTML and CSS, JavaScript is one of the three essential
technologies of <a id="link1"><u> World Wide Web </u>
</a>content production; the majority of websites employ it and it is
supported by all modern Web browsers without plug-ins.
</div>
@{
Html.EJ().Tooltip("link1",
(Syncfusion.JavaScript.Models.TooltipProperties)ViewData["tooltip"]).Rend
er();
}
```

C#

```
public ActionResult Index()
{
TooltipProperties obj = new TooltipProperties();
obj.Content = "The World Wide Web (WWW) is an information space where
documents and other web resources are identified by URLs, interlinked by
hypertext links, and can be accessed via the Internet.";
ViewData["tooltip"] = obj;
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

TreeMap

TreeMap

Essential TreeMap for **JavaScript** is ideal for visualizing large amounts of data. **TreeMap** holds a set of nested nodes to display hierarchical data. Each nested node may contain sub-nodes containing area proportional to a specified data value bounded. Color and annotation can be used to provide extra information about the leaf nodes. **EssentialTreeMap** is a perfect solution for developers looking to add advanced, feature rich **Treemap** to their applications.

Key Features

- **Levels** - TreeMap levels are used to define levels of various flat data and hierarchical data collection.
- **Layout** - Layouts such as Squarified, SliceAndDiceAuto, SliceAndDiceHorizontal and SliceAndDiceVertical determines the visual representation of nodes belonging to all the treemap levels in TreeMap.
- **Visualization for Colors** - Easy customization options is provided to customize the colors of the leaf nodes of TreeMap.

Use Case Scenarios

TreeMap is used to represent large or complex data sets in various applications, such as:

- Stock market analysis where the weight of each stock in the index is represented by the size of the rectangle and its range of loss or gain is represented by the color of the rectangle.
- Visualizing Internet usage in certain categories, such as retail, social networks, and search.
- Categorizing the news aggregated by Google News where the colors represent different sections, such as business or politics, and the size of the boxes represent how many similar stories also appear in Google News.
- Indicating weather report analysis around the world. The opacity of each rectangle can differ based on its humidity.

Getting Started

This section explains briefly about how to create a **TreeMap** in your application with ASP.NET MVC .

Configure a TreeMap

You can configure an **MVC TreeMap** in simple steps. This manual provides instructions on how to configure TreeMap with grouping of populated datum based on population growth in population in each continent. It also provides a walk-through on some of the customization feature available in TreeMap control.

Copyright © 2001 - 2020 Syncfusion Inc.



Create a simple MVC Application

You can create a new ASP.NET MVC project Razor application by adding the necessary dll's and scripts.

Add Data

You can populate the TreeMap data inside the controller. For example, you can populate data of few countries from the following location.

~/Controller/TreeMapController.cs

C#

```
public class TreeMapPopulationData
{
public string Continent { get; set; }
public string Region { get; set; }
public int Growth { get; set; }
public long Population { get; set; }
public static List<TreeMapPopulationData> GetData()
{
List<TreeMapPopulationData> population = new
List<TreeMapPopulationData>();
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Southern Asia", Growth = 1.32, Population = 1749046000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Eastern Asia", Growth = 0.57, Population = 1620807000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"South-Eastern Asia", Growth = 1.20, Population = 618793000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Western Asia", Growth = 1.98, Population = 245707000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Central Asia", Growth = 1.43, Population = 64370000});
population.Add(new TreeMapPopulationData() { Continent = "Europe", Region
= "Europe", Growth = 0.10, Population = 742452000});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
population.Add(new TreeMapPopulationData() { Continent = "America",
Region = "South America", Growth = 1.06, Population = 406740000});
population.Add(new TreeMapPopulationData() { Continent = "America",
Region = "Northern America", Growth = 0.85, Population = 355361000});
population.Add(new TreeMapPopulationData() { Continent = "America",
Region = "Central America", Growth = 1.40, Population = 167387000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Eastern Africa", Growth = 2.89, Population = 373202000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Western Africa", Growth = 2.78, Population = 331255000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Northern Africa", Growth = 1.70, Population = 210002000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Middle Africa", Growth = 2.79, Population = 135750000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Southern Africa", Growth = 0.91, Population = 60425000});
return population;
}
}
```

Note: Population data is referred from List of continents by population

Initialize TreeMap

1. Create a tag with a specific id and set the height and width to determine the TreeMap size to be rendered in “TreeMap.cshtml” file

ASPX-CS

```
<div style="width:700px; height:400px;">
</div>
```

2. Add the following code in “TreeMap.cshtml” file to create the TreeMap control in a View page.

ASPX-CS

```
<div style="width:700px; height:400px;">
@(Html.EJ().TreeMap("treemap")
.Render())
</div>
```

3. Add the TreeMap() action in “TreeMapController.cs” as illustrated in the following code sample.

C#

```
public ActionResult TreeMap()
{
ViewData["datasource"] = TreeMapPopulationData.GetData();
return View();
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Data Source

The DataSource property accepts the collection values as input. For example, you can provide the list of objects as input.

Weight Value Path

You can calculate the size of the object using WeightValuePath of TreeMap.

1. Populate the TreeMap by using the above properties.

ASPX-CS

```
@{
var datasource = ViewData["datasource"];
}
<html xmlns="http://www.w3.org/1999/xhtml">
<head> // ...
</head>
<body>
<div style="height:400px;width:700px;">
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Render())
</div>
// ...
</body>
</html>
```

2. Final TreeMap.cshtml file is illustrated in the following code sample.

ASPX-CS

```
@{
var datasource = ViewData["datasource"];
}
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
<script src="http://cdn.syncfusion.com/13.1.0.21/js/
ej.widgets.all.min.js"></script>
<script src="http://cdn.jsdelivr.net/jsrender/1.0pre35/jsrender.min.js"
type="text/javascript"></script>
<script
src="http://cdn.syncfusion.com/13.1.0.21/js/web/ej.unobtrusive.min.js"
type="text/javascript"></script>
<script src="http://cdn.syncfusion.com/13.1.0.21/js/web/flat-
azure/ej.web.all.min.css" type="text/javascript"></script><body>
<div style="height:400px;width:700px;">
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Render())
</div>
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</body>
</html>
```

The following image displays a TreeMap with default properties using the above code.

GroupTreeMap Items using Levels

You can group TreeMap Items using Levels in TreeMap.

Group Path

You can use GroupPath property for every flat level of the TreeMap control. It is a path to a field on the source object that serves as the “Group” for the level specified. You can group the data based on the GroupPath in the TreeMap control. When the GroupPath is not specified, then the items are not grouped and the data is displayed in the order specified in the DataSource.

Group Gap

You can use GroupGap property to separate the items from every flat level and to differentiate the levels mentioned in the TreeMap control.

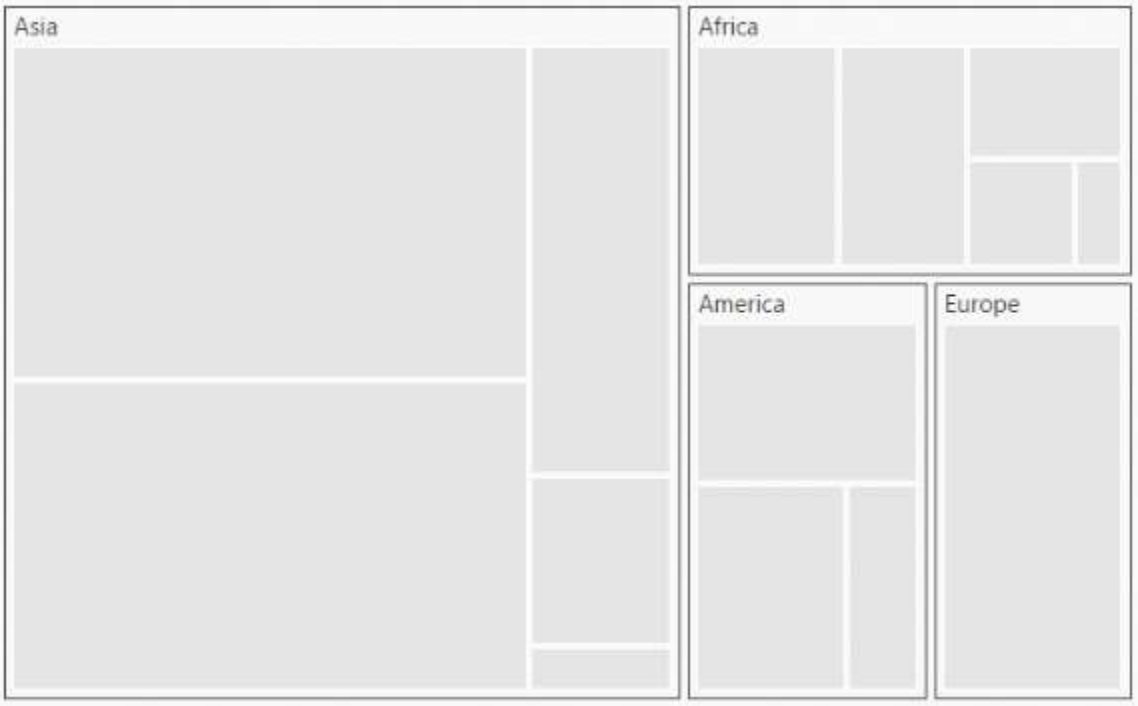
The following code sample explains how to group TreeMap Items using ‘Levels’

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5).Add();
})
.Render())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays grouping of TreeMapItems using Levels.



Customize TreeMap Appearance by Range

You can differentiate the nodes based on its value and color ranges using RangeColorMapping . You can also define the color value range using From and To properties.

Color Value Path

The ColorValuePath of TreeMap is a path to a field on the source object. You can determine the color for the object using ColorValuePath of TreeMap.

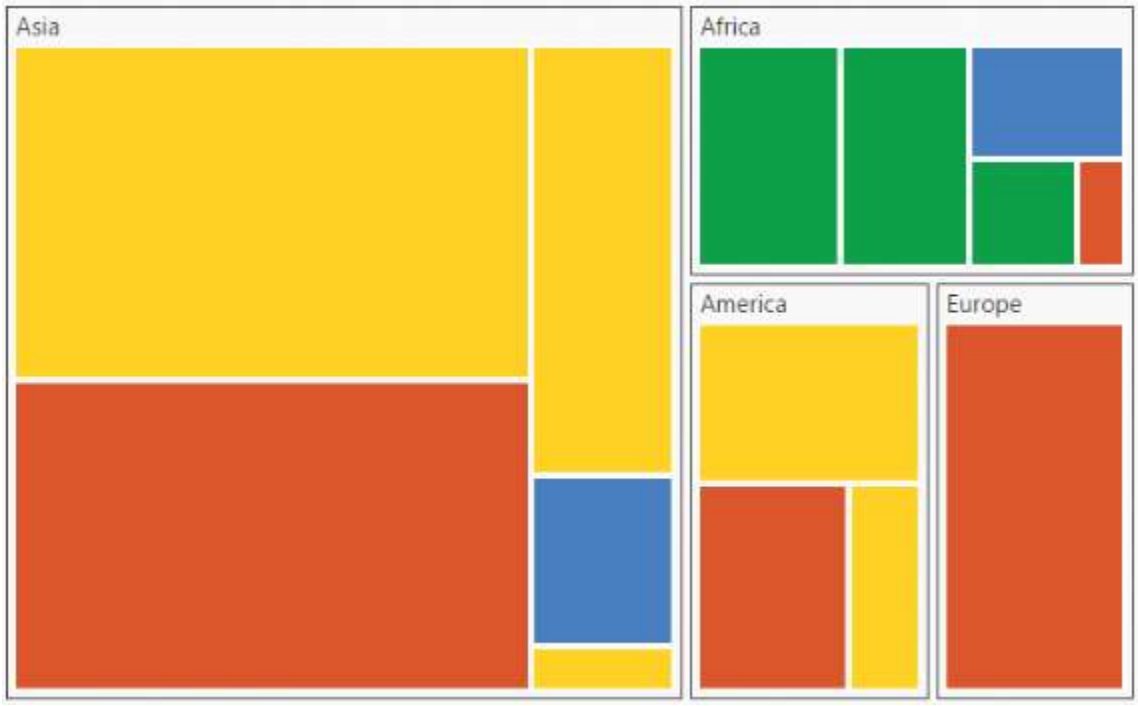
The following code sample explains how to customize TreeMap Appearance by Range.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5).Add();
})
.ColorValuePath("Growth")
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#DC562D").Add();
cm.To(1.5).From(1).Color("#FED124").Add();
cm.To(2).From(1.5).Color("#487FC1").Add();
cm.To(3).From(2).Color("#0E9F49").Add();
})
.Render())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays customized TreeMap Appearance by Range



Enable Tooltip

You can enable the tooltip by setting ShowTooltip property to 'true'. By default, it takes the property of the bound object that is referred in the WeightValuePath and displays its content when the corresponding node is hovered. You can customize the template for tooltip using TooltipTemplate property.

Leaf Item Settings

You can customize the Leaf level TreeMap items using LeafItemSettings. The Label and tooltip values take the property of bound object that is referred in the LabelPath when defined.

The following code sample displays how the tooltip is enabled.

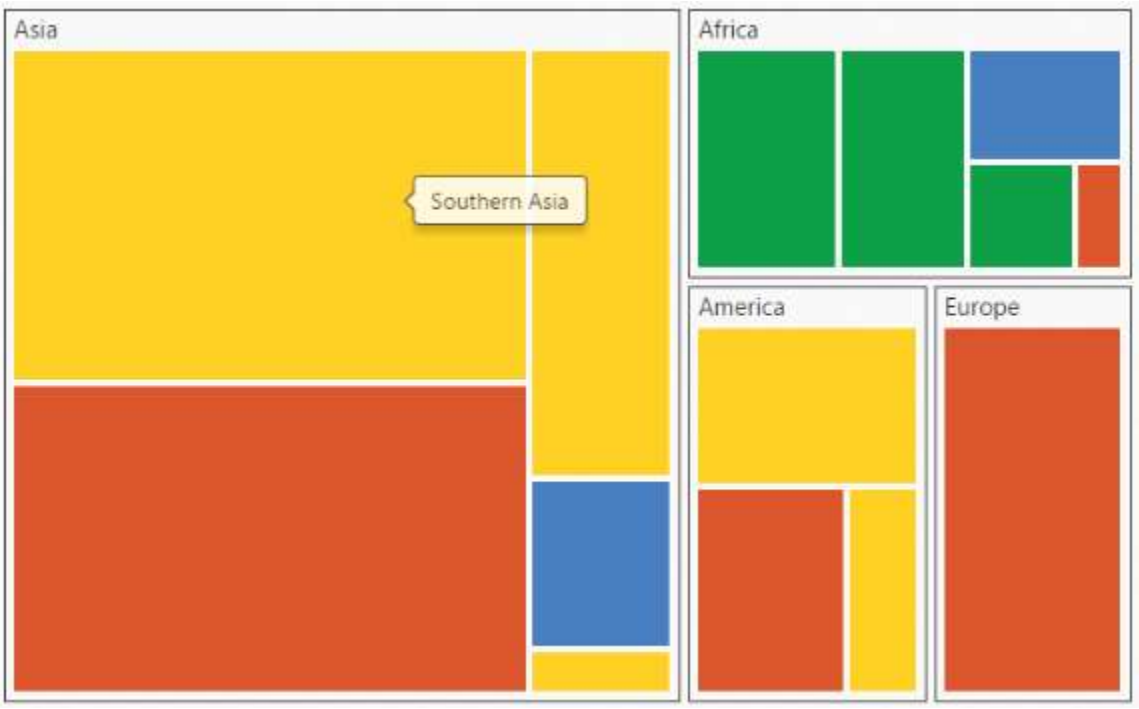
ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5).Add();
})
.ColorValuePath("Growth")
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#DC562D").Add();
cm.To(1.5).From(1).Color("#FED124").Add();
cm.To(2).From(1.5).Color("#487FC1").Add();
cm.To(3).From(2).Color("#0E9F49").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
.ShowTooltip(true)
.LeafItemsSettings(leaf =>
{
leaf.LabelPath("Region");
})
.Render())
```

The following screenshot displays the TreeMap when the Tooltip is enabled.



Legend

You can set the color value of leaf nodes using TreeMap LegendSettings . This legend is appropriate only for the TreeMap whose leaf nodes are colored using RangeColorMapping.

You can set ShowLegend property value to '*true*' to make a Legend visible.

Label for Legend

You can customize the labels of the legend item using LegendLabel property of RangeColorMapping.

The following code sample displays how to add labels for legend in a TreeMap.

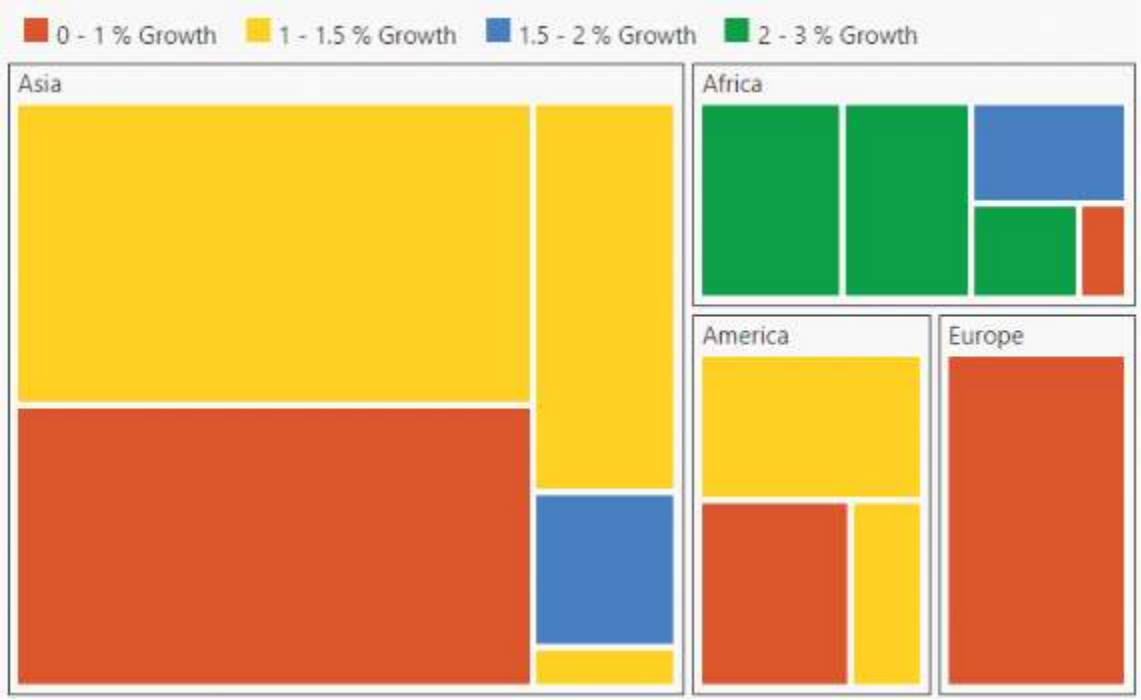
ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5).Add();
})
.ColorValuePath("Growth")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#DC562D").LegendLabel("0 - 1 % Growth").Add();
cm.To(1.5).From(1).Color("#FED124").LegendLabel("1 - 1.5 %
Growth").Add();
cm.To(2).From(1.5).Color("#487FC1").LegendLabel("1.5 - 2 %
Growth").Add();
cm.To(3).From(2).Color("#0E9F49").LegendLabel("2 - 3 % Growth").Add();
})
.ShowTooltip(true)
.LeafItemsSettings(leaf =>
{
leaf.LabelPath("Region");
})
.ShowLegend(true)
.Render())
```

The following screenshot displays the TreeMap when Labels are enabled.



DataBinding

TreeMap control supports Data Binding and it can be achieved using DataSource property.

The DataSource property accepts the collection values as input. For example, you can provide the list of objects as input. The following code illustrates you on how to bind a flat collection as datasource for **TreeMap**.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.ColorValuePath("Population")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Render())
```

C#

```
public class TreeMapPopulationData
{
public string Continent { get; set; }
public string Region { get; set; }
public double Growth { get; set; }
public long Population { get; set; }
public static List<TreeMapPopulationData> GetData()
{
List<TreeMapPopulationData> population = new
List<TreeMapPopulationData>();
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Southern Asia", Growth = 1.32, Population = 1749046000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Eastern Asia", Growth = 0.57, Population = 1620807000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"South-Eastern Asia", Growth = 1.20, Population = 618793000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Western Asia", Growth = 1.98, Population = 245707000});
population.Add(new TreeMapPopulationData() { Continent = "Asia", Region =
"Central Asia", Growth = 1.43, Population = 64370000});
population.Add(new TreeMapPopulationData() { Continent = "Europe", Region
= "Europe", Growth = 0.10, Population = 742452000});
population.Add(new TreeMapPopulationData() { Continent = "America",
Region = "South America", Growth = 1.06, Population = 406740000});
population.Add(new TreeMapPopulationData() { Continent = "America",
Region = "Northern America", Growth = 0.85, Population = 355361000});
population.Add(new TreeMapPopulationData() { Continent = "America",
Region = "Central America", Growth = 1.40, Population = 167387000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Eastern Africa", Growth = 2.89, Population = 373202000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Western Africa", Growth = 2.78, Population = 331255000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Northern Africa", Growth = 1.70, Population = 210002000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Middle Africa", Growth = 2.79, Population = 135750000});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Region
= "Southern Africa", Growth = 0.91, Population = 60425000});
return population;
}
}
```

TreeMapLevels

The Levels of TreeMap can be categorized into two types as,

- FlatLevel
- Hierarchical Level

Following customization options are available to customize the treemap level as per your requirements.

Copyright © 2001 - 2020 Syncfusion Inc.

- To specify the background color for the group, you can use GroupBackground property.
- To specify the border color for the group, you can use GroupBorderColor property.
- To maintain the border thickness for the group, you can use GroupBorderThickness property.
- You can specify the gaps between groups using GroupGap property.
- You can specify the padding using GroupPadding property.
- For specifying the header height, you can use HeaderHeight property.
- You can customize the header template using HeaderTemplate property.
- To specify the label position, you can use LabelPosition property.
- To specify the label template for treemap, you can use LabelTemplate property.
- You can specify the label visibility using LabelVisibilityMode property.
- You can control the label visibility using ShowLabels property.
- For controlling text overflow, you can use TextOverflow property.

Flat Level

Group Path

You can use GroupPath property for every flat level of the **TreeMap** control. It is a path to a field on the source object that serves as the **"Group"** for the level specified. You can group the data based on the GroupPath in the **TreeMap** control. When the GroupPath is not specified, then the items are not grouped and the data is displayed in the order specified in the DataSource.

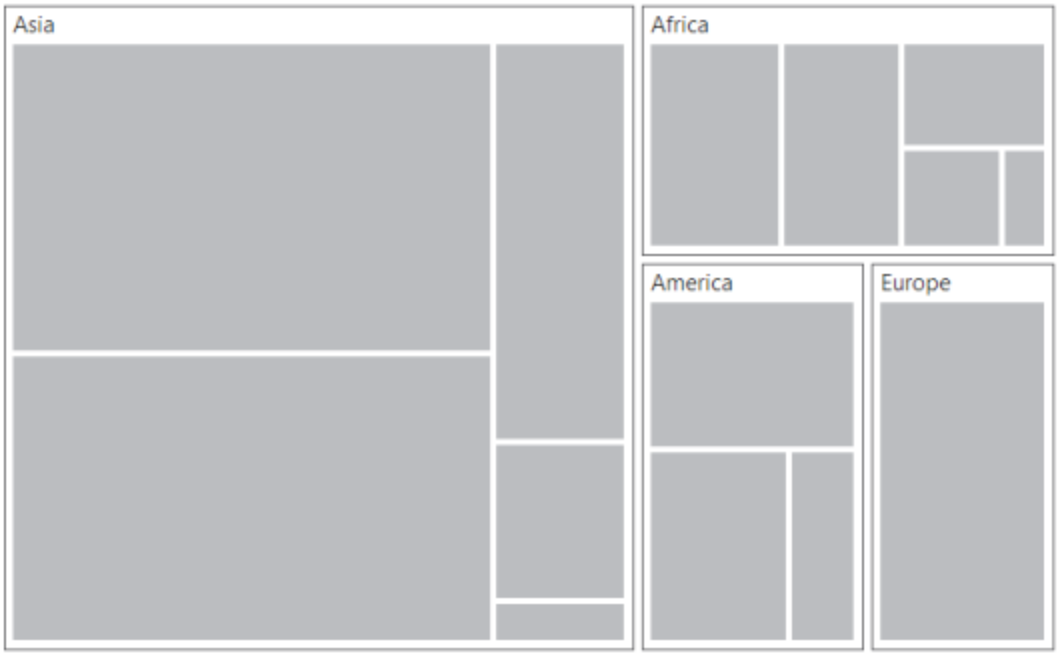
Group Gap

You can use GroupGap property to separate the items from every flat level and to differentiate the levels mentioned in the **TreeMap** control.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.ColorValuePath("Growth")
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5).Add();
})
.Render())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Hierarchical Level

TreeMap Hierarchical level is used to define levels for hierarchical data collection that contains tree-structured data.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.WeightValuePath("Population")
.Render())
```

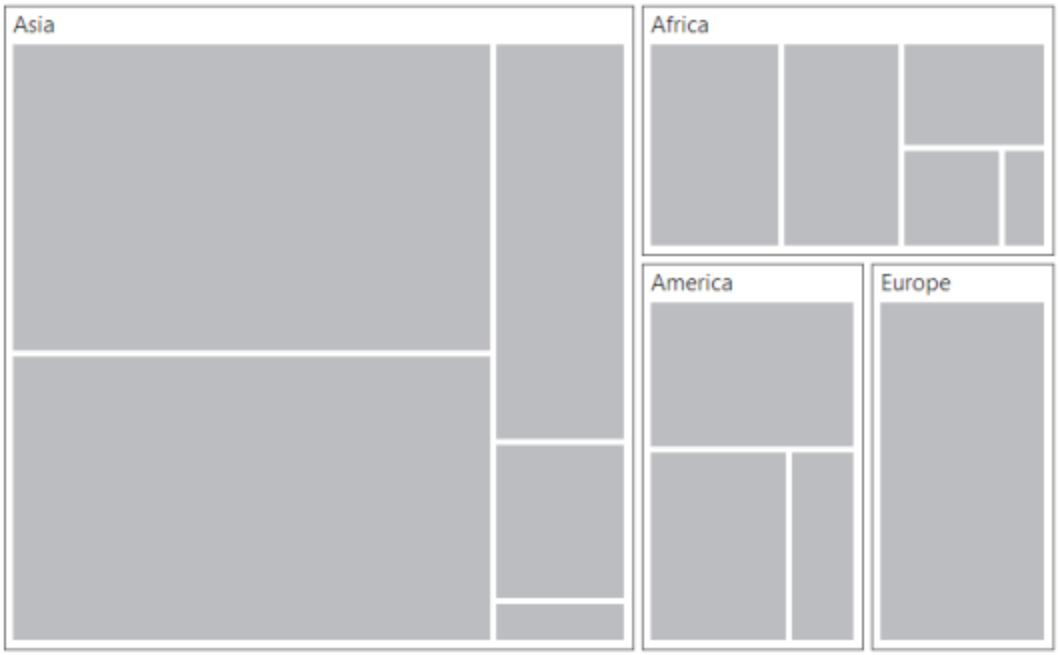
C#

```
public class ContinentData
{
public string Region { get; set; }
public double Growth { get; set; }
public long Population { get; set; }
public static List<ContinentData> GetAsiaData()
{
List<ContinentData> Asia = new List<ContinentData>();
Asia.Add(new ContinentData() { Region = "Southern Asia", Growth = 1.32,
Population = 1749046000 });
Asia.Add(new ContinentData() { Region = "Eastern Asia", Growth = 0.57,
Population = 1620807000 });
Asia.Add(new ContinentData() { Region = "South-Eastern Asia", Growth =
1.20, Population = 618793000 });
Asia.Add(new ContinentData() { Region = "Western Asia", Growth = 1.98,
Population = 245707000 });
Asia.Add(new ContinentData() { Region = "Central Asia", Growth = 1.43,
Population = 64370000 });
return Asia;
}
public static List<ContinentData> GetEuropeData()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
List<ContinentData> Europe = new List<ContinentData>();
Europe.Add(new ContinentData() { Region = "Europe", Growth = 0.10,
Population = 742452000 });
return Europe;
}
public static List<ContinentData> GetAmericaData()
{
List<ContinentData> America = new List<ContinentData>();
America.Add(new ContinentData() {  Region = "South America", Growth =
1.06, Population = 406740000 });
America.Add(new ContinentData() {  Region = "Northern America", Growth =
0.85, Population = 355361000 });
America.Add(new ContinentData() {  Region = "Central America", Growth =
1.40, Population = 167387000 });
return America;
}
public static List<ContinentData> GetAfricaData()
{
List<ContinentData> Africa = new List<ContinentData>();
Africa.Add(new ContinentData() { Region = "Eastern Africa", Growth =
2.89, Population = 373202000 });
Africa.Add(new ContinentData() { Region = "Western Africa", Growth =
2.78, Population = 331255000 });
Africa.Add(new ContinentData() { Region = "Northern Africa", Growth =
1.70, Population = 210002000 });
Africa.Add(new ContinentData() { Region = "Middle Africa", Growth = 2.79,
Population = 135750000 });
Africa.Add(new ContinentData() { Region = "Southern Africa", Growth =
0.91, Population = 60425000 });
return Africa;
}
}
public class TreeMapPopulationData
{
public string Continent { get; set; }
public List<ContinentData> Data { get; set; }
public static List<TreeMapPopulationData> GetData()
{
List<TreeMapPopulationData> population = new
List<TreeMapPopulationData>();
population.Add(new TreeMapPopulationData() { Continent = "Asia", Data =
ContinentData.GetAsiaData()});
population.Add(new TreeMapPopulationData() { Continent = "Europe", Data =
ContinentData.GetEuropeData()});
population.Add(new TreeMapPopulationData() { Continent = "America", Data
= ContinentData.GetAmericaData()});
population.Add(new TreeMapPopulationData() { Continent = "Africa", Data =
ContinentData.GetAfricaData() });
return population;
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Layout

You can decide on the visual representation of nodes belonging to all the treemap levels using the ItemsLayoutMode property of the TreeMap.

There are four different **TreeMap Layouts** such as

- Squarified Layout
- SliceAndDiceAuto Layout
- SliceAndDiceHorizontal Layout
- SliceAndDiceVertical Layout

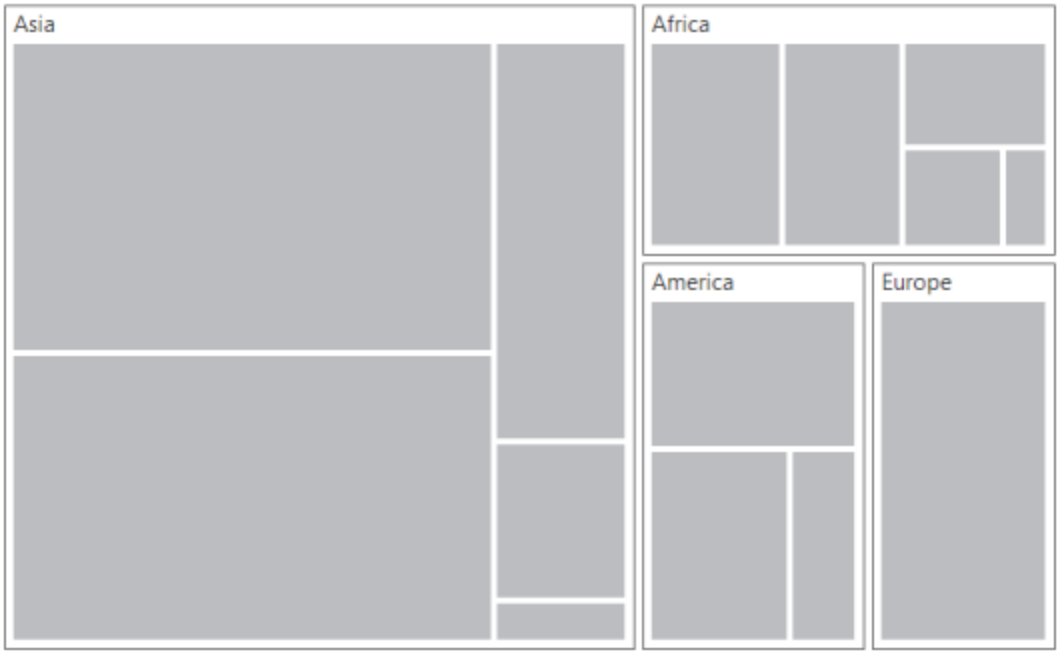
Squarified Layout

Squarified layout creates rectangles with best aspect ratio.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.ColorValuePath("Growth")
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5).Add();
})
.ItemsLayoutMode("Squarified")
.Render())
```

Copyright © 2001 - 2020 Syncfusion Inc.

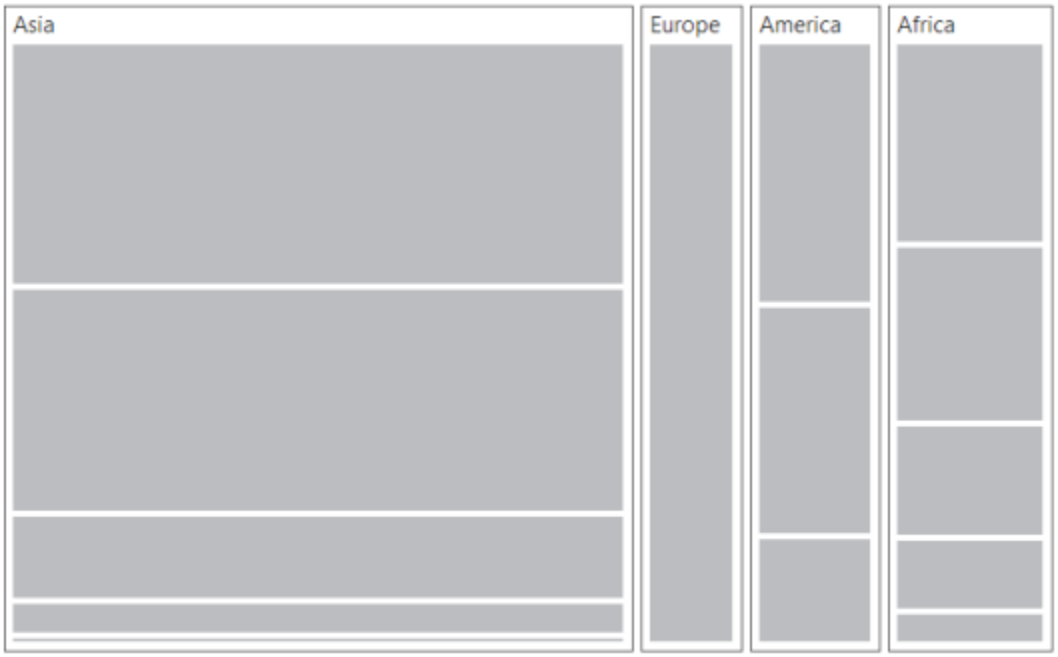


SliceAndDiceAuto Layout

SliceAndDiceAuto layout creates rectangles with high aspect ratio and displays them sorted both horizontally and vertically.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.ItemsLayoutMode("SliceAndDiceAuto ")
.Render())
```



Copyright © 2001 - 2020 Syncfusion Inc.

SliceAndDiceHorizontal Layout

SliceAndDiceHorizontal layout creates rectangles with high aspect ratio and displays them sorted horizontally.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.ItemsLayoutMode("SliceAndDiceHorizontal ")
.Render())
```



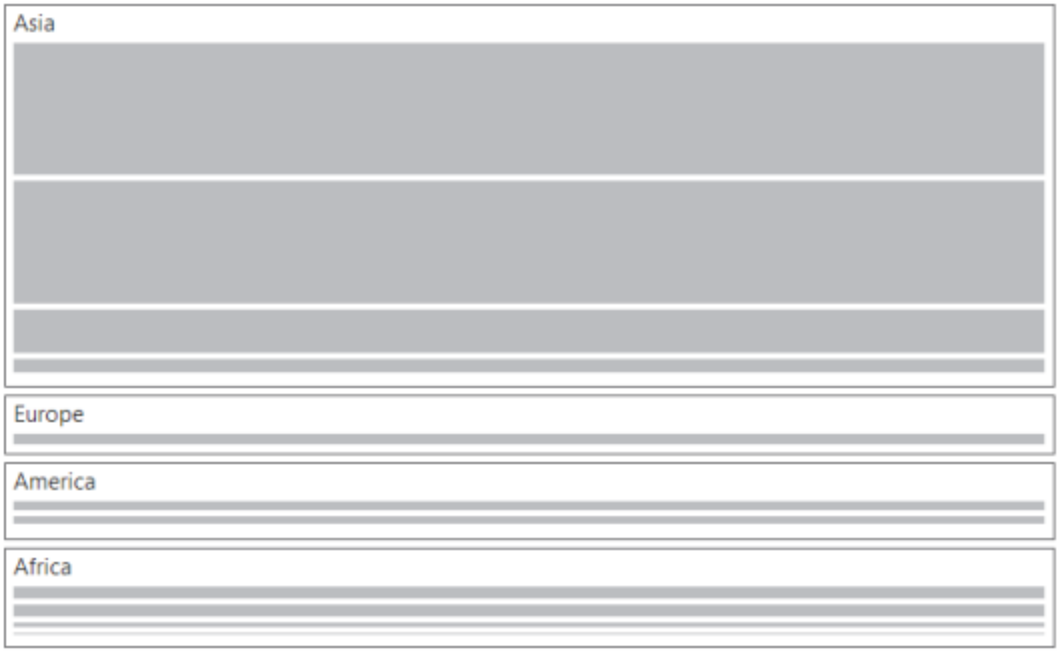
SliceAndDiceVertical Layout

SliceAndDiceVertical layout creates rectangles with high aspect ratio and displays them sorted vertical.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.ItemsLayoutMode("SliceAndDiceVertical ")
.Render())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Customization

TreeMap control supports color customization to determine the exact combination of colors for tree nodes displayed in TreeMap and tooltip support to display additional information of treemap data.

Color

You can customize the color of leaf nodes in **Treemap** either using ColorMapping support of the **Treemap** or mapping a field in the datasource.

Binding color from the datasource

You can set color for each leaf items from data source by using `ColorPath` property.

Note: While setting color, do not set any other color mapping for treemap because color mapping has higher priority than `ColorPath` property. And also, if `ColorPath` is set, the legend will be generated for each leaf item in treemap.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.ColorPath("fill")
.Render())
```



Copyright © 2001 - 2020 Syncfusion Inc.

ColorMapping is categorized into three different types such as,

- UniColorMapping
- RangeBrushColorMapping
- DesaturationColorMapping

UniColorMapping

You can color, all the leaf nodes with the same color by setting the Color value of the UniColorMapping property of the TreeMap.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.TreeMapUniColorMapping(cm =>
{
cm.Color("crimson");
})
.Render())
```



Range Color Mapping

You can group the leaf nodes based on the range of the data's color values. You can set a unique color for every ranges. To achieve this, specify the To and From values as range bound and Color or GradientColors values to fill the leaf nodes of the particular range, through the RangeColorMapping property of the **TreeMap**.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#77D8D8").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
cm.To(2).From(0).Color("#AED960").Add();
cm.To(3).From(0).Color("#FFAF51").Add();
cm.To(4).From(0).Color("#F3D240").Add();
})
.Render())
```



Desaturation Color Mapping

You can differentiate all the leaf nodes using the DesaturationColorMapping property of the TreeMap. Differentiation is achieved, even though same color is applied for all the leaf nodes by varying the opacity of the leaf nodes based on the color value specified in the color value range using RangeMinimum and RangeMaximum value of the data collection. You can also bound the opacity range by setting From and To property of the DesaturationColorMapping.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.TreeMapDesaturationColorMapping(cm =>
{
cm.To(0.2).From(1).Color("DeepSkyBlue")
.RangeMinimum(0).RangeMaximum(4);
})
.Render())
```

Copyright © 2001 - 2020 Syncfusion Inc.



Tooltip

You can enable the tooltip support for the TreeMap by setting the ShowTooltip property to true. By default, it takes the property of the bound object that is referred to in the GroupPath and displays its content when the corresponding node is tapped. The TooltipTemplate is a HTML element that is used to expose the custom template for the tooltip.

Leaf Item Setting

You can customize the **Leaf level TreeMap** items using LeafItemsSetting. In LeftItemSetting following customization options are available.

- You can specify the border color using BorderBrush property.
- For customizing border thickness, you can use BorderThickness property.
- To customize the gap between the leaf items, you can use Gap property.
- You can specify the label template for the leaf item using ItemTemplate property.
- The Label and tooltip values take the property of bound object that is referred in the LabelPath when defined.
- You can specify the position of the leaf labels using LabelPosition property.
- You can control the mode of label visibility of the labels using LabelVisibilityMode property.
- To show or hide the visibility of the leaf item labels you can use ShowLabels property.
- For specifying over flow action of left item labels you can use TextOverflow property.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.ColorValuePath("Growth")
.WeightValuePath("Population")
.Levels(lv =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
lv.GroupPath("Continent")
.GroupGap(5)
.Add();
})
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#77D8D8").Add();
cm.To(2).From(0).Color("#AED960").Add();
cm.To(3).From(0).Color("#FFAF51").Add();
cm.To(4).From(0).Color("#F3D240").Add();
})
.LeafItemsSetting(li =>
{
li.LabelPath("Region")
.ShowLabels(true)
})
.ShowTooltip(true)
.TooltipTemplate("template")
.Render())
</div>
<script  id="template" type="application/jsrender">
<div  style="margin-left:17px;margin-top:-45px;">
<div style="height:auto;width:auto;background:black;border-
radius:3px;opacity:0.6">
<div style="margin-top:-20px;margin-left:9px;padding-top:3px;margin-
right:9px;">
<label style="margin-top:-20px;font-weight:normal;font-
size:12px;color:white;font-family:Segoe UI;">{{:Region}}</label>
</div>
<div style="height:10px;"></div>
<div style="margin-top:-10px;margin-left:9px;margin-right:9px;padding-
bottom:3px;">
<label style="margin-top:-10px;font-weight:normal;font-
size:14px;color:white;font-family:segoe ui
light;">{{:Population}}</label>
</div>
</div>
</div>
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Border Brush

You can able to customize the border color of the treemap using the property BorderBrush.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set borderBrush API value during initialization
.BorderBrush('white')
)
```

Border Thickness

For customizing the border thickness of the treemap, you can use the BorderThickness property.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set borderThickness API value during initialization
.BorderThickness(1)
)
```

Dock Position

You can position the legend at top, bottom, left and right side of the treemap as per your requirement. For changing the position as per your requirement, you can use DockPosition property.

<ts name="ej.datavisualization.TreeMap.DockPosition"/>

Specifies the dockPosition for legend

Name	Description
top	specifies the top position

Copyright © 2001 - 2020 Syncfusion Inc.

Name	Description
bottom	specifies the bottom position
right	specifies the bottom position
left	specifies the left position

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set dockPosition API value during initialization
.TreeMapLegend(legend=>legend.DockPosition(TreeMapDockPosition.Top))
)
```

Clicking and Dragging

You can select the single treemap element on click and drag. To click and drag treemap items, you have to enable the DraggingOnSelection property.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set draggingOnSelection API value during initialization
.DraggingOnSelection(false)
)
```

For selecting the group element of treemap while clicking and dragging, you can use DraggingGroupOnSelection property.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set draggingGroupOnSelectionAPI value during initialization
.DraggingGroupOnSelection(false)
)
```

Fill with Gradient

You can customize that whether gradient color have to be applied for treemap or not. This can be customized using the property EnableGradient.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set enableGradient API value during initialization
.EnableGradient(true)
)
```

Responsive Treemap

You can customize whether treemap have to be responsive or not while resizing the container. For making treemap responsive you can use EnableResize or IsResponsive property.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().TreeMap("treemap")
//To set enableResize API value during initialization
.EnableResize(false)
```

GroupColorMapping

You can customize the color of the each group using GroupColorMapping property. To use group color mapping, kindly specify GroupID and RangeColorMapping inside the GroupColorMapping.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set groupColorMapping API value during initialization
.TreeMapGroupColorMapping( )
```

GroupSelectionMode

You can specifies the selection mode of the treemap using GroupSelectionMode property. You can set either group selection mode value as Default or Multiple.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
// Set the selection mode during initialization.
.GroupSelectionMode(GroupSelectionMode.Default)
```

Header

You can specify the header for the parent item using the property Header. This is applicable only for hierarchical data source.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set header API value during initialization
.Header("Country")
```

Specifying HierarchicalDatasource

You can specify whether data source bound for the treemap is hierarchical or not using the property IsHierarchicalDatasource.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
//To set isHierarchicalDatasource API value during initialization
.IsHierarchicalDatasource(true)
```

Treemap Items

You can specify the treemap items which you want to display in the treemap using the property TreeMapItems.

TreeMap Elements

TreeMap contains various elements such as,

Copyright © 2001 - 2020 Syncfusion Inc.

- Legend
- Headers
- Labels

Legend

You can set the color value of **leaf nodes** using TreeMapLegend. This legend is appropriate only for the TreeMap whose leaf nodes are colored using RangeColorMapping.

You can set ShowLegend property value to “true” to enable or disable legend visibility.

TreeMap Legend

You can customize the treemap legend using following properties

- For customizing the alignment of legend, you can use Alignment property.
- You can set the legend column count using ColumnCount property.
- To set the dock position of the legend text, you can use DockPosition.
- You can specify the size of the legend by setting Height and Width of the TreeMapLegend.
- You can decide the size of the legend icons by setting IconWidth and IconHeight properties of the TreeMapLegend property avail in TreeMap.
- You can customize the left and right label text for the legend using LeftLabel and RightLabel properties.
- To set the legend mode as default or interactive, you can use Mode property.
- Using the property Template, you can specify template for legend settings.
- To set the legend title, you can use the Title property of the LegendSettings.

Label for Legend

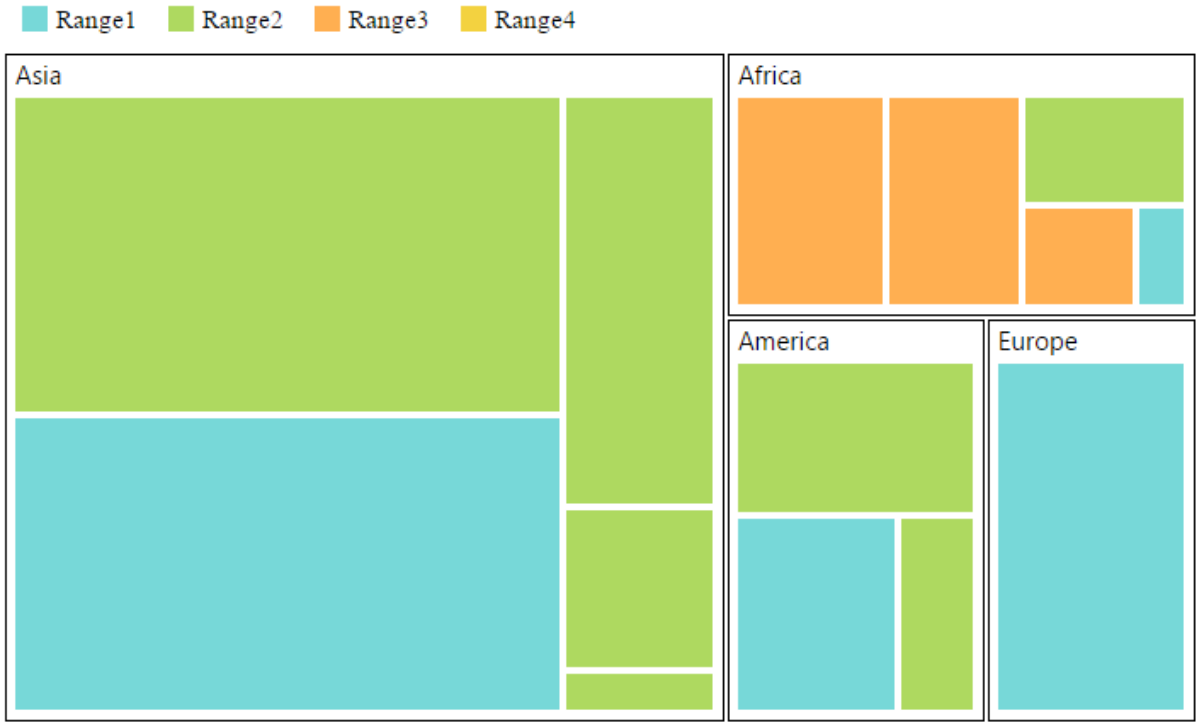
You can customize the labels of the legend item using LegendLabel property of RangeColorMapping.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.ColorValuePath("Growth")
.WeightValuePath("Population")
.ShowLegend(true)
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5)
.Add();
})
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#77D8D8").LegendLabel("Range1").Add();
cm.To(2).From(0).Color("#AED960").LegendLabel("Range2").Add();
cm.To(3).From(0).Color("#FFAF51").LegendLabel("Range3").Add();
cm.To(4).From(0).Color("#F3D240").LegendLabel("Range4").Add();
})
.TreeMapLegend(legend =>
{
legend. IconWidth(20)
. IconHeight(20);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
.Render())
```



Interactive Legend

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding treemap items. You can enable this option by setting Mode property in Legend settings value as “interactive” and default value of Mode property is “default” to enable the normal legend.

Title for Interactive Legend

You can provide the title for interactive legend by using Title property in LegendSettings.

Label for Interactive Legend

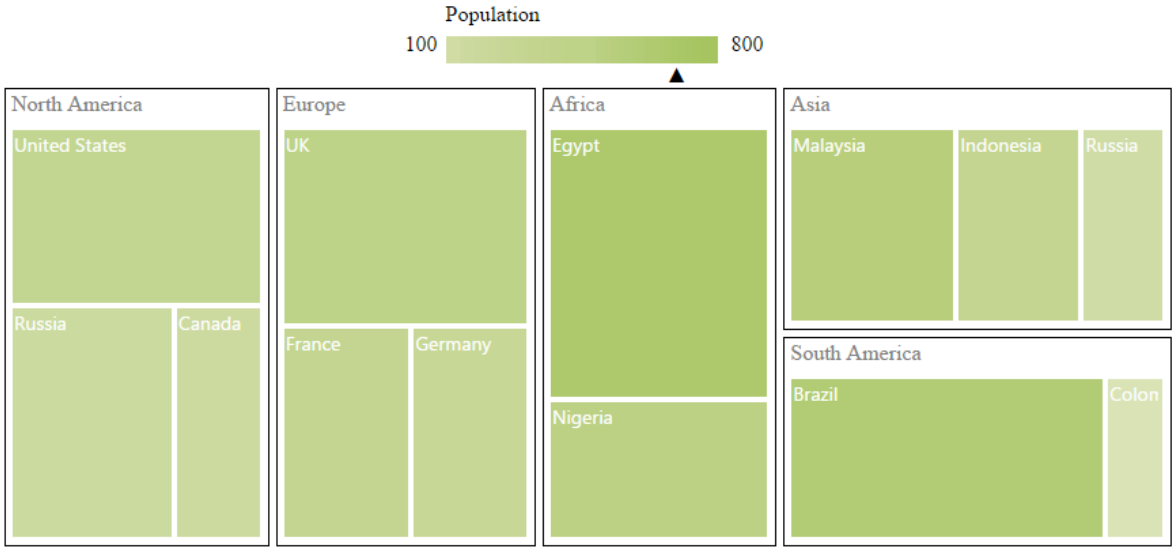
You can provide the left and right labels to interactive legend by using LeftLabel and RightLabel properties in LegendSettings.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
// ...
.ShowLegend(true)
.TreeMapLegend(legend =>
{
legend.DockPosition(TreeMapDockPosition.Top)
.Height(20)
.Width(150)
.Mode(LegendMode.Interactive)
.Title("population")
.LeftLabel("20000000")
.RightLabel("300000000");
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
```



Header

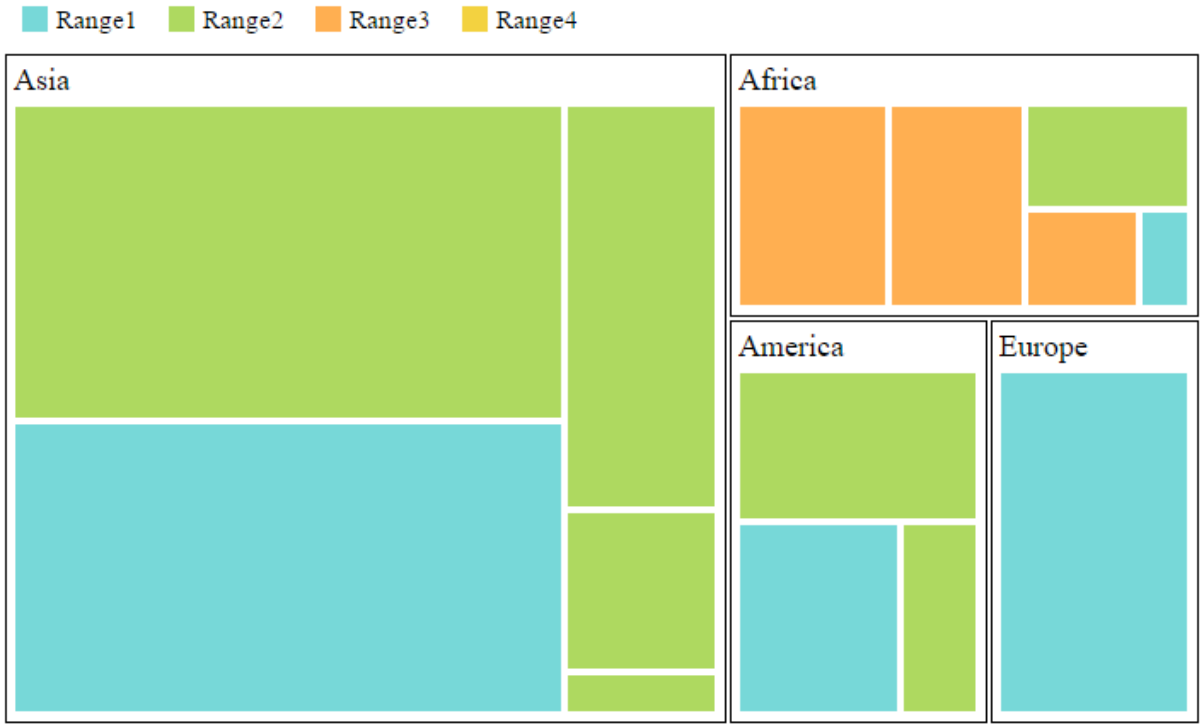
You can set headers for each level by setting the ShowHeader property of the each TreeMap levels. The HeaderHeight property helps to set the height of the header and Group path value determines the header value. You can customize the default header appearance by setting the HeaderTemplate of the TreeMap levels.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.ColorValuePath("Growth")
.WeightValuePath("Population")
.ShowLegend(true)
.TreeMapLegend(legend =>
{
legend.IconWidth(17)
.IconHeight(17);
})
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#77D8D8").LegendLabel("Range1").Add();
cm.To(2).From(0).Color("#AED960").LegendLabel("Range2").Add();
cm.To(3).From(0).Color("#FFAF51").LegendLabel("Range3").Add();
cm.To(4).From(0).Color("#F3D240").LegendLabel("Range4").Add();
})
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5)
.HeaderHeight(30)
.HeaderTemplate("template").Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Render())
<script  id="template" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:medium;" >{{:header}}</label><br />
</div>
</script>
```



Customizing the header

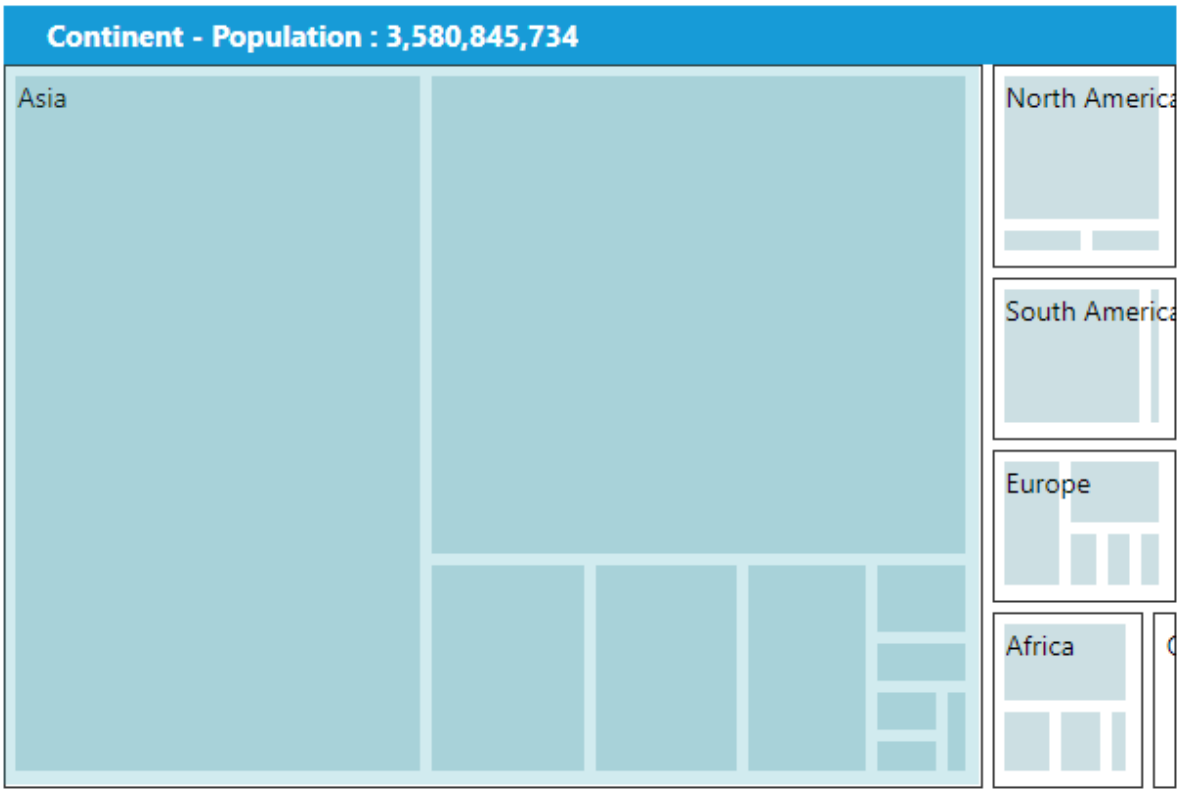
The text in the header can be customized by triggering the event HeaderTemplateRendering of the **TreeMap**. This event is triggered before rendering the header template.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.Levels(lv =>
{
lv.GroupPath("Continent")
.HeaderHeight(25)
.HeaderTemplate('Template')
.ShowHeader(false)
.GroupGap(5)
.LabelPosition(DockPosition.TopLeft)
.ShowLabels(true).Add();
})
.HeaderTemplateRendering("loadTemplate")
)
<script  id="Template" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:large;" >{{:header}}</label><br />
</div>
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<script type="text/javascript">
function loadTemplate(sender) {
//...
}
</script>
```



Label

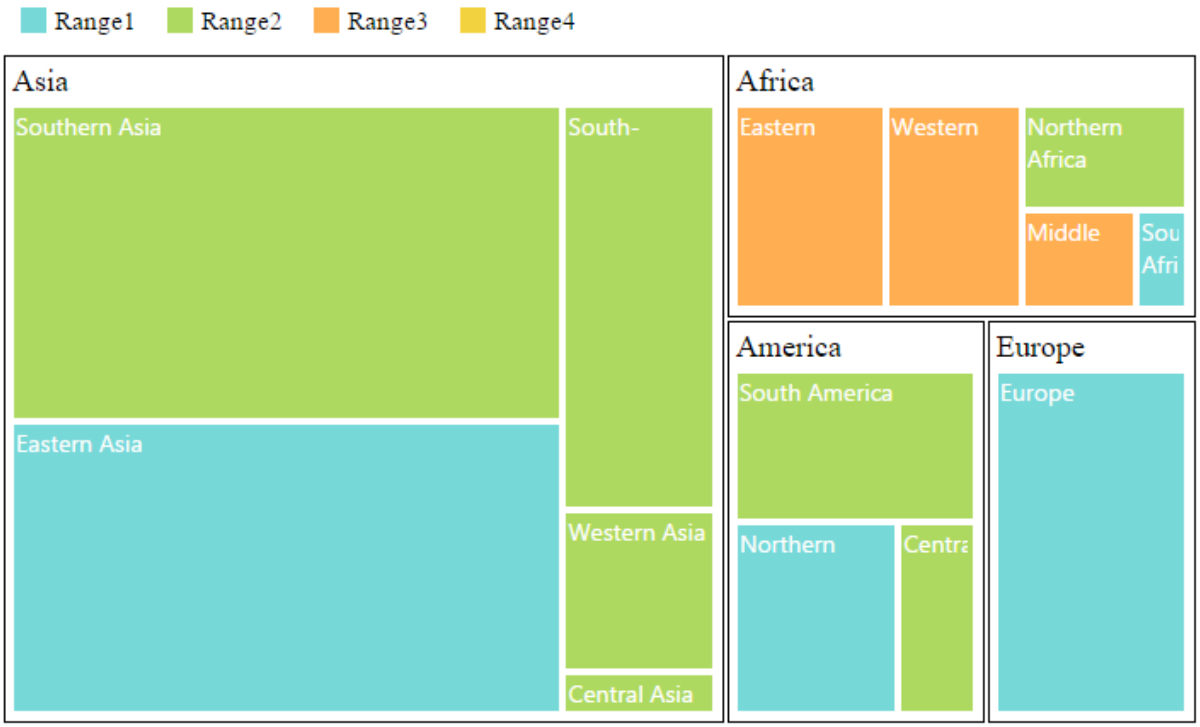
You can also set labels for the leaf nodes by setting the ShowLabels property as true. GroupPath value is displayed as a label for leaf nodes. You can customize the default label appearance by setting the LabelTemplate of the TreeMap levels.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.ColorValuePath("Growth")
.WeightValuePath("Population")
.ShowLegend(true)
.TreeMapLegend(legend =>
{
legend. IconWidth(17)
. IconHeight(17);
})
.Levels(lv =>
{
lv.GroupPath("Continent")
.GroupGap(5)
.HeaderHeight(30)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.HeaderTemplate("headerTemplate").Add();
})
.TreeMapRangeColorMappings(cm =>
{
cm.To(1).From(0).Color("#77D8D8").LegendLabel("Range1").Add();
cm.To(2).From(0).Color("#AED960").LegendLabel("Range2").Add();
cm.To(3).From(0).Color("#FFAF51").LegendLabel("Range3").Add();
cm.To(4).From(0).Color("#F3D240").LegendLabel("Range4").Add();
})
.LeafItemsSetting(li =>
{
li.LabelPath("Region")
.ShowLabels(true)
.LabelTemplate("labelTemplate");
})
.Render())
<script  id="headerTemplate" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:medium;" >{{:header}}</label><br />
</div>
</script>
<script  id="labelTemplate" type="application/jsrender">
<div style="background-color: transparent;">
<label style="color:white;font-size:small;margin:5px;"
>{{:label}}</label><br/>
</div>
</script>
```



Customizing the Overflow labels

You can handle the label overflow, by specifying any one of the following values to the property TextOverflowas

None - It displays the default label text.

Hide - You can hide the label, when it exceeds the header width.

Wrap - You can wrap the label text by letter.

WrapByWord - You can wrap the label text by word.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.Levels(level =>
{
level.GroupPath("Continent")
.HeaderHeight(25)
.HeaderTemplate('Template')
.ShowHeader(false)
.GroupGap(5)
.LabelPosition(DockPosition.TopLeft)
.ShowLabels(true).Add();
})
.HeaderTemplateRendering("loadTemplate")
.LeafItemsSetting(leaf =>
{
leaf.ShowLabels(true)
.TextOverflow(TextOverflow.Wrap);
})
)
<script  id="headertemplate" type="application/jsrender">
<div style="background-color: white; margin:5px">
<label style="color:black;font-size:medium;" >{{:header}}</label><br />
</div>
</script>
```

Palette Color Mapping

Treemap is having support for PaletteColorMapping. You can set the color for palette color mapping using the property Color in palette color mapping.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.TreeMapPaletteColorMapping(
//..
)
```

Drill Down Support

Treemap enables drill down to expose the hierarchy achieved by clicking on a node and this results in enabling the Treemap to move to the next level or sub level and can return back to the normal Treemap view by clicking on the node header. Only a single level of the Treemap is visible at once.

Copyright © 2001 - 2020 Syncfusion Inc.

Enabling Drill Down

Treemap elements can be drilled down by setting the EnableDrillDown property to true. You can view the hierarchy of the Treemap by clicking on the treemap items and can move to the previous level by clicking on the drill down header. The header color can be customized by changing the values in the property DrillDownHeaderColor and the selection color can be done by changing the DrillDownSelectionColor property.

Drill down Properties

Property	Type	Description
enableDrillDown	bool	Gets or sets a value that indicates whether the drill down feature is enabled or not
drillDownHeaderColor	string	Gets or sets a color for header during drill down
drillDownSelectionColor	string	Gets or sets a color for highlighting tree map item during drill down.

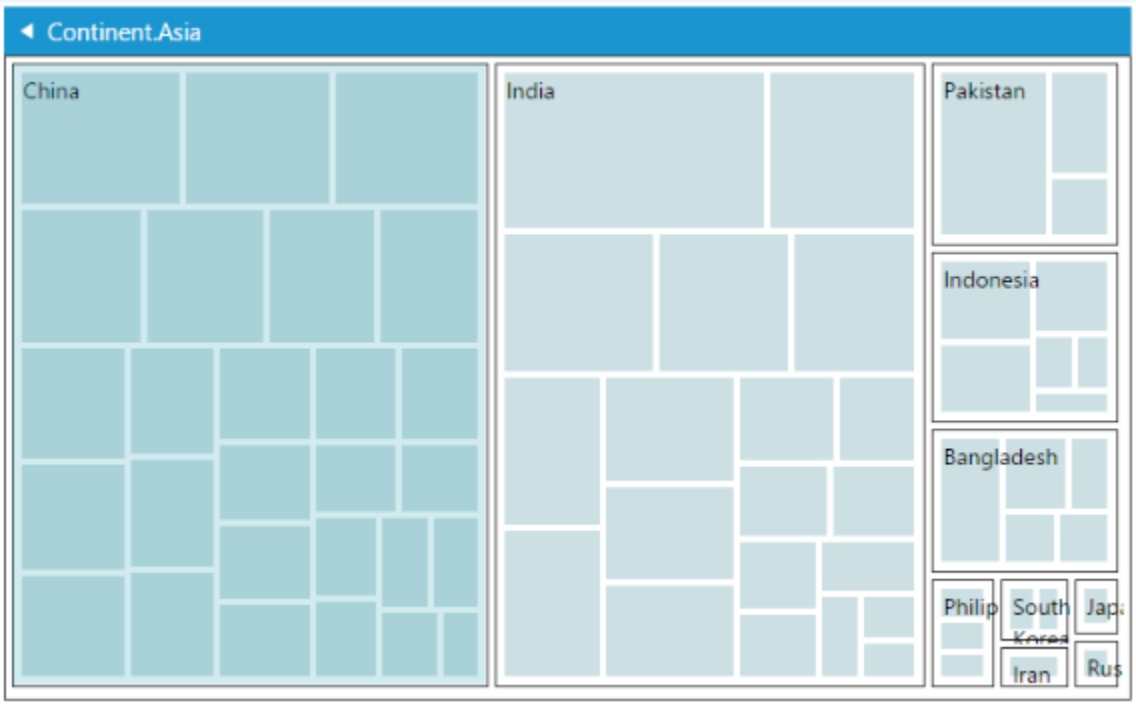
ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DataSource(datasource)
.EnableDrillDown(true)
.DrillDownHeaderColor("#199DAF")
.DrillDownSelectionColor("#199DAF")
.WeightValuePath("Population")
.Levels(lv =>
{
lv.GroupPath("Continent")
.HeaderHeight(25)
.GroupGap(5)
.ShowLabels(true).Add();
lv.GroupPath("Country")
.HeaderHeight(25)
.GroupGap(0)
.ShowLabels(true).Add();
lv.GroupPath("Name")
.HeaderHeight(25)
.GroupGap(0)
.ShowLabels(true).Add();
})
)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Before Drill Down



After Drill Down

Methods and Events

Method

Refresh()

refresh() method is used to reload the treemap with updated values.

Copyright © 2001 - 2020 Syncfusion Inc.

Returns: void

ASPX-CS

```
@(Html.EJ().TreeMap("treemap"))
<script type="text/javascript">
$("#treemap").ejTreeMap();
var treemapObj = $("#treemap").data("ejTreeMap");
treemapObj.refresh();
</script>
```

Events

TreeMapItemSelected

TreeMapItemSelected event will trigger when treemap item is selected.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns selected treeMapItem object.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.TreeMapItemSelected("itemSelected")
)
function itemSelected(sender) {
// do something
}
```

DrillStarted

DrillStarted event will fire, when the drilldown action is started in the treemap control.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns selected drilled treeMap object.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DrillStarted("onDrillStarted")
)
function onDrillStarted(sender) {
// do something
}
```

DrillDownItemSelected

If the treemap drilldown item is selected, then DrillDownItemSelected event will fire.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns selected drilldown treeMap object.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DrillDownItemSelected("drillDownItemSelected")
)
function drillDownItemSelected(sender) {
// do something
}
```

HeaderTemplateRendering

HeaderTemplateRendering event will fire before rendering the treemap drilldown header template.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns drilldown header.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.HeaderTemplateRendering("loadTemplate")
)
function loadTemplate(sender) {
// do something
}
```

Refreshed

Refreshed event will trigger after refreshing the treemap control with updated values.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Refresh and load the treemap.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.Refreshed("onRefreshed")
)
function onRefreshed(sender) {
// do something
}
```

TreeMapGroupSelected

TreeMapGroupSelected event will fired when the group selection is performed on treemap items.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns the selected group of treeMapItems as object.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.TreeMapGroupSelected("groupSelected")
)
function groupSelected(sender) {
// do something
}
```

ItemRendering

ItemRendering event will trigger while rendering each item in treemap.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns each leaf items in treemap

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.ItemRendering("itemRendering")
)
function itemRendering(sender) {
// do something
}
```

LegendItemRendering

LegendItemRendering event will trigger while rendering each legend item in treemap.

Name	Type	Description
{% highlight html %}originalEvent{% endhighlight %}	object	Returns each legend item in treemap

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.LegendItemRendering("legendItemRendering")
)
function legendItemRendering(sender) {
// do something
}
```

Click

Click event will trigger while clicking an item in the treemap.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.Click("click")
)
function click(){
// Do Something
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

DoubleClick

DoubleClick event will trigger while double clicking an item in the treemap

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.DoubleClick("doubleClick")
)
function doubleClick(){
// Do Something
}
```

RightClick

RightClick event will trigger while right clicking an item in the treemap.

ASPX-CS

```
@(Html.EJ().TreeMap("treemap")
.RightClick("rightClick")
)
function rightClick(){
// Do Something
}
```

AngularJS Support

AngularJS is a JavaScript framework added to a **HTML** page with a <script> tag. It extends HTML attributes with directives and binds data to HTML with expressions. AngularJS directives allow you to specify custom and reusable HTML tags that moderate the behavior of certain elements. **Angular binding** uses directives to plug its action into the page. Directives, all prefaced with ng-, are placed in HTML attributes. To know more about Angular binding refer to: http://help.syncfusion.com/aspnetmvc/treemap/angularjs-support

Apply the plugin and property assigning the **Treemap** element through the directive that starts with the letter “e-“. The following code illustrates how to bind data to the Treemap component through Angular support.

ASPX-CS

```
<script>
var population_data = [
{ Continent: "Asia", Country: "Indonesia", Growth: 3, Population:
237641326 },
{ Continent: "Asia", Country: "Russia", Growth: 2, Population: 152518015
},
{ Continent: "Asia", Country: "Malaysia", Growth: 1, Population: 29672000
},
{ Continent: "North America", Country: "United States", Growth: 4,
Population: 315645000 },
{ Continent: "North America", Country: "Mexico", Growth: 2, Population:
112336538 },
{ Continent: "North America", Country: "Canada", Growth: 1, Population:
39056064 },
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{ Continent: "South America", Country: "Colombia", Growth: 1, Population:
47000000 },
{ Continent: "South America", Country: "Brazil", Growth: 3, Population:
193946886 },
{ Continent: "Africa", Country: "Nigeria", Growth: 2, Population:
170901000 },
{ Continent: "Africa", Country: "Egypt", Growth: 1, Population: 83661000
},
{ Continent: "Europe", Country: "Germany", Growth: 1, Population:
81993000 },
{ Continent: "Europe", Country: "France", Growth: 1, Population: 65605000
},
{ Continent: "Europe", Country: "UK", Growth: 1, Population: 63181775 },
];
</script>
//References to be added for AngularJS support.
<script src="@Url.Content("~/Scripts/angular.min.js")"></script>
<script
src="@Url.Content("~/Scripts/ej/ej.widget.angular.min.js")"></script>
//Initializes controller
<div ng-app="SyncApp">
//Initializes controller
<div ng-controller="TreeMapcontroller">
//Initializes TreeMap
<div id="treemap" ej-treemap e-datasource="ndatasource" e-
uniColorMapping-color="ncolor" e-weightvaluepath="nweightValuePath" e-
colorvaluepath="ncolorValuePath" e-leafitemsettings-
labelpath="nlabelPath" style="width: 700px;height:370px;">
<e-levels>
<e-level e-grouppath="ngroupPath" e-groupgap="ngroupGap" e-
showheader="nshowHeader"></e-level>
</e-levels>
</div>
//Renders a checkbox to change the header visibility
<div>
Show Header:  <input type="checkbox" ng-model="nshowHeader"
style="outline: none;"/>
</div>
//Renders a textbox to change the groupGap value
<div>
Group Gap:  <input type="text" id="Text11" ng-model="ngroupGap"
style="width: 110px" />
</div>
<script>
angular.module('syncApp', ['ejangular'])
.controller('TreeMapController', function ($scope) {
$scope.datasource = population_data;
$scope.colorValuePath = "Growth";
$scope.weightValuePath = "Growth";
$scope.labelPath = "Country";
$scope.groupPath = "Continent";
$scope.groupGap = 5;
$scope.showHeader = true;
$scope.color = "#2380BB";
});
</script>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
```

C#

```
public partial class TreeMapController : Controller
{
public ActionResult AngularTreeMap()
{
return View();
}
}
```



TreeGrid

TreeGrid

The TreeGrid is an efficient control designed for representing the hierarchical data in a tabular format, combining the visual representation of Grid and TreeView controls; it represents the data from datasource such as array of JSON objects, ej.DataManager or self-referential datasource.

Key Features

- **Editing** - Offers cell and row editing modes for updating the records.
- **Sorting** - Supports *n* levels of sorting.
- **Filtering** - Supports filtering records with filterbar,menu and excel filtering modes.
- **Virtualization** - Supports rendering large number of hierarchical data at once.
- **Column Template** - To display custom columns along with customized expand-collapse icon.
- **Row drag and drop** - Allows you to drag any row and drop it at any desired position.
- **Paging** - Paginated view of records.

Copyright © 2001 - 2020 Syncfusion Inc.

- **Summary rows** - To summarize the rows at each hierarchical levels.
- **Details template** - To display the detailed view of a record.
- **Frozen columns** - To freeze the columns from scrolling.
- **Exporting** - Exporting the TreeGrid contents to PDF and Excel files.
- **Column Menu** - Supports sorting, column visibility and filtering for the columns.
- **User Interaction** - Supports cell tooltip, expand collapse at ease, single and multiple row selection, columns resizing, contextual menu.
- **Column Template** - To render custom columns along with customized expand-collapse icon.
- **Command Column** - Allows you to perform custom action or default CRUD actions using TreeGrid column.
- **Row Template** - To customize the TreeGrid rows.
- **Column Reordering** - Provides support to re-arrange the column order.
- **Resize columns** - Provides option for resizing the columns.

Getting Started

Create your first TreeGrid in MVC

This section explains briefly about the control structure and how to create a TreeGrid.

1. First, create a new MVC Razor project, please refer MVC-Getting Started documentation to create new project and add necessary DLL's and script files.
2. Create a data source in Controllers/HomeController.cs file, We can bind datasource to TreeGrid using Datasource property.
3. Add the following code example in the Index.cshtml file to create a TreeGrid control in the View page.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
.Columns(co =>
{
co.Field("TaskId").HeaderText("Task
Id").EditType(TreeGridEditingType.Numeric).Add();
co.Field("TaskName").HeaderText("Task
Name").EditType(TreeGridEditingType.String).Add();
co.Field("StartDate").HeaderText("Start
Date").EditType(TreeGridEditingType.Datepicker).Add();
co.Field("Duration").HeaderText("Duration").EditType(TreeGridEditingType.
Numeric).Add();
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

C#

```
public class TreeGridController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = this.GetData();
```

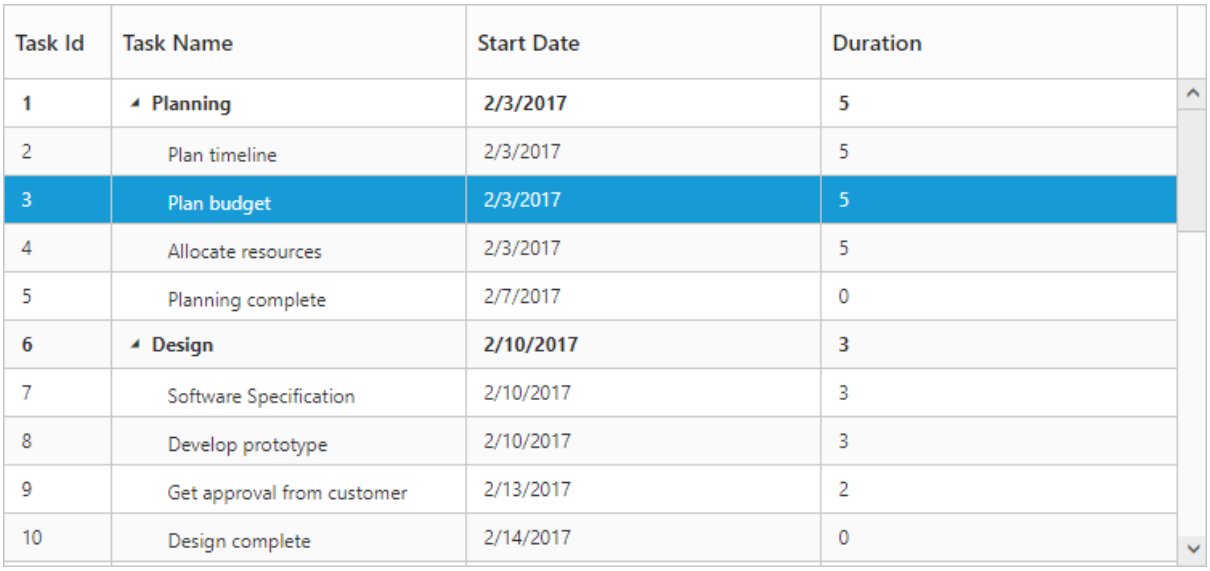
Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
private List<BusinessObject> GetData()
{
List<BusinessObject> dataCollection = new List<BusinessObject>();
BusinessObject Record1 = new BusinessObject()
{
TaskId = 1,
TaskName = "Planning",
StartDate = "02/03/2017",
Duration = 5,
Children = new List<BusinessObject>(),
};
BusinessObject Child1 = new BusinessObject()
{
TaskId = 2,
TaskName = "Plan timeline",
StartDate = "02/03/2017",
Duration = 5
};
BusinessObject Child2 = new BusinessObject()
{
TaskId = 3,
TaskName = "Plan budget",
StartDate = "02/03/2017",
Duration = 5
};
BusinessObject Child3 = new BusinessObject()
{
TaskId = 4,
TaskName = "Allocate resources",
StartDate = "02/03/2017",
Duration = 5
};
BusinessObject Child4 = new BusinessObject()
{
TaskId = 5,
TaskName = "Planning complete",
StartDate = "02/07/2017",
Duration = 0
};
Record1.Children.Add(Child1);
Record1.Children.Add(Child2);
Record1.Children.Add(Child3);
Record1.Children.Add(Child4);
BusinessObject Record2 = new BusinessObject()
{
TaskId = 6,
TaskName = "Design",
StartDate = "02/10/2017",
Duration = 3,
Children = new List<BusinessObject>(),
};
BusinessObject Child5 = new BusinessObject()
{
TaskId = 7,
TaskName = "Software Specification",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
StartDate = "02/10/2017",
Duration = 3
};
BusinessObject Child6 = new BusinessObject()
{
TaskId = 8,
TaskName = "Develop prototype",
StartDate = "02/10/2017",
Duration = 3
};
BusinessObject Child7 = new BusinessObject()
{
TaskId = 9,
TaskName = "Get approval from customer",
StartDate = "02/13/2017",
Duration = 2
};
BusinessObject Child8 = new BusinessObject()
{
TaskId = 10,
TaskName = "Design complete",
StartDate = "02/14/2017",
Duration = 0
};
Record2.Children.Add(Child5);
Record2.Children.Add(Child6);
Record2.Children.Add(Child7);
Record2.Children.Add(Child8);
dataCollection.Add(Record1);
dataCollection.Add(Record2);
return dataCollection;
}
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
public int Duration { get; set; }
public List<BusinessObject> Children { get; set; }
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	Duration
1	Planning	2/3/2017	5
2	Plan timeline	2/3/2017	5
3	Plan budget	2/3/2017	5
4	Allocate resources	2/3/2017	5
5	Planning complete	2/7/2017	0
6	Design	2/10/2017	3
7	Software Specification	2/10/2017	3
8	Develop prototype	2/10/2017	3
9	Get approval from customer	2/13/2017	2
10	Design complete	2/14/2017	0

Simple TreeGrid

Enable Toolbar

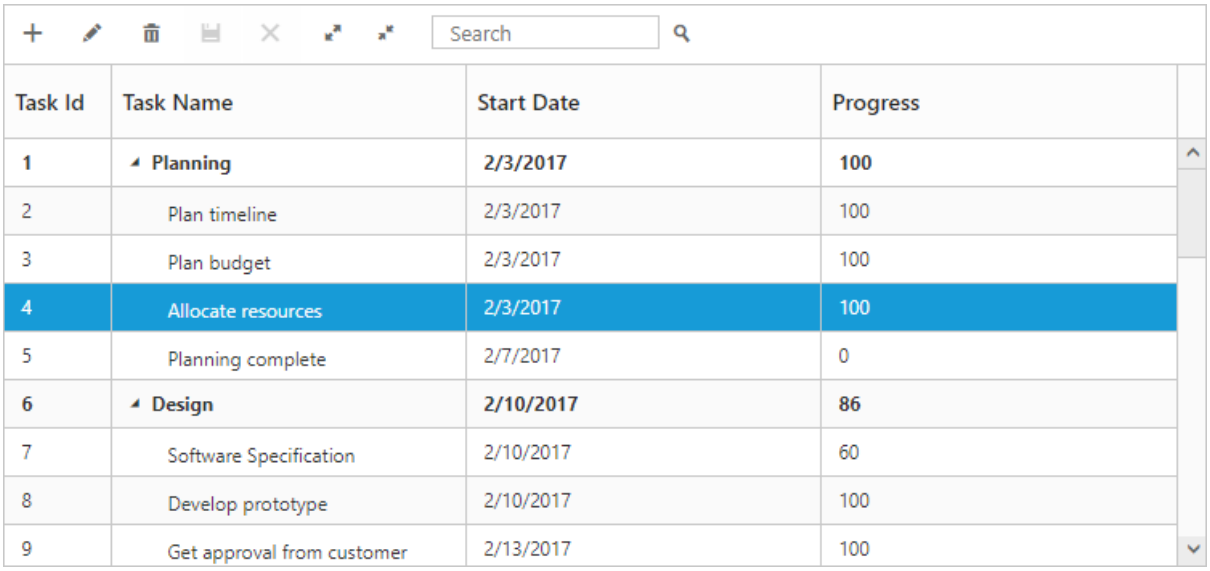
TreeGrid control contains the toolbar options to Add, Edit, Delete, Cancel, Update, Search, ExpandAll and CollapseAll operations, you can enable the toolbar by using ToolbarSettings property.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.Add,
TreeGridToolBarItems.Edit,
TreeGridToolBarItems.Delete,
TreeGridToolBarItems.Update,
TreeGridToolBarItems.Cancel,
TreeGridToolBarItems.ExpandAll,
TreeGridToolBarItems.CollapseAll,
TreeGridToolBarItems.Search
});
})
)
@(Html.EJ().ScriptManager())
```

The following screen shot displays a Toolbar in TreeGrid.

Copyright © 2001 - 2020 Syncfusion Inc.



Toolbar in TreeGrid

Note: Add, Edit, Delete options are enabled by `AllowEditing`, `AllowAdding`, `AllowDeleting` in the `EditSettings`.

Click here to refer more details for TreeGrid ToolbarSettings.

Enable Sorting

The TreeGrid control contains sorting functionality to arrange the data in **ascending** or **descending** order based on a particular column.

Multicolumn Sorting

You can enable the multicolumn sorting in TreeGrid by setting `AllowMultiSorting` as `true` .You can sort multiple columns in TreeGrid by selecting the desired column header when holding the **CTRL** key.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowSorting(true)
.AllowMultiSorting(true)
)
@(Html.EJ().ScriptManager())
```


Click here to refer more details for TreeGrid Sorting.

Enable Editing

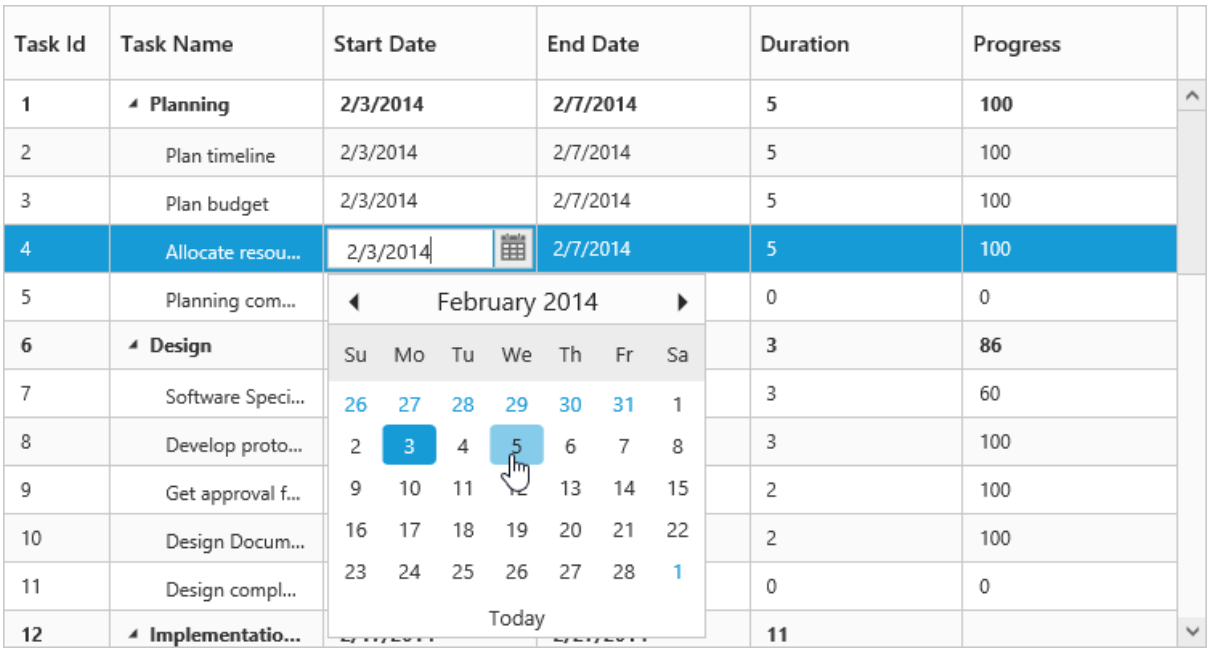
You can enable editing in TreeGrid by using `EditSettings` property and it is illustrated in the following code example.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit =>
{
edit.AllowEditing(true);
edit.AllowAdding(true);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
edit.AllowDeleting(true);
edit.EditMode(TreeGridEditMode.CellEditing);
})
)
@(Html.EJ().ScriptManager())
```



Click here to refer more details for TreeGrid Editing.

Enable Context Menu

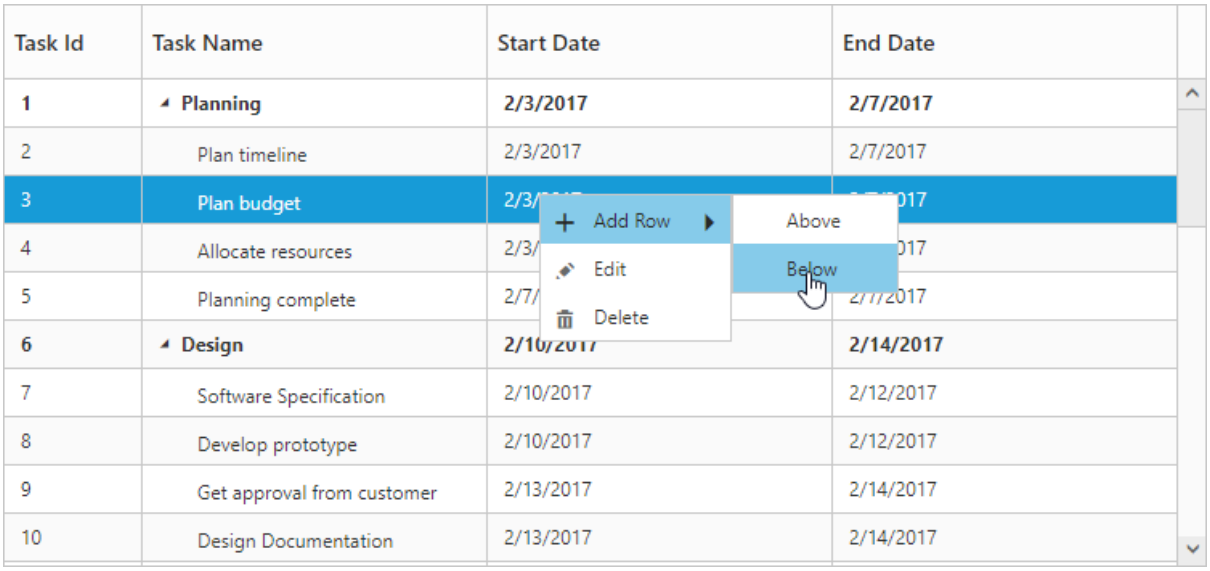
The **Context menu** in TreeGrid control is used to manipulate (add, edit and delete) the TreeGrid rows.

In TreeGrid, context menu can be enabled by ContextMenuSettings property. The ContextMenuSettings property contains two inner properties ShowContextMenu and ContextMenuItems.

ASPX-CS

```
@(Html.EJ().TreeGrid("treegrid1")
.ContextMenuSettings(context =>
{
context.ShowContextMenu(true);
context.ContextMenuItems(new List<TreeGridContextMenuItems>(){
TreeGridContextMenuItems.Add,
TreeGridContextMenuItems.Edit,
TreeGridContextMenuItems.Delete
});
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot displays the context menu in TreeGrid control.



TreeGrid with context menu

Click here to refer more details for TreeGrid context menu.

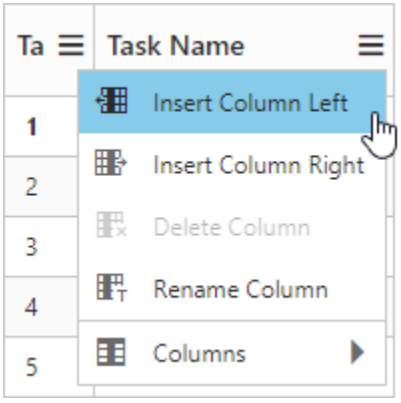
Enable Column Menu

You can enable the column menu in TreeGrid, by setting the ShowColumnChooser as true.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowColumnChooser(true)
.AllowSorting(true)
.AllowMultiSorting(true)
)
@(Html.EJ().ScriptManager())
```

The following screenshot displays the column chooser in TreeGrid control.



TreeGrid with column chooser

Define dimension of TreeGrid

By default TreeGrid control was rendered with 100% width and 450px height, we can define the dimension of TreeGrid control by using SizeSettings property. TreeGrid control width and height can be

Copyright © 2001 - 2020 Syncfusion Inc.

defined by either Height and Width properties or by defining inline style in TreeGrid container element. The below code example shows how to define width and height for TreeGrid control.

ASPX-CS

```
<!--<div id="TreeGridContainer" style="width:700px;height:350px"></div>--
>
<div id="TreeGridContainer"></div>
<script>
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SizeSettings(size=>size.Height("350px").Width("700px"))
)
@(Html.EJ().ScriptManager())
</script>
```

Note: 1.TreeGrid control will automatically update the width and height value based on container element on window resize action, this can be enabled by setting IsResponsive property as true for this Height and Width value will be defined in percentage.

2.We can also render TreeGrid with auto height by setting SizeSettings.Height as auto.

Data Binding

Data Binding is the process that establishes a connection between the application and different kinds of datasources such as business objects.

Local Data Binding

In Local Data Binding, datasource for rendering the TreeGrid control is retrieved from the same application locally.

Two types of Data Binding are possible with TreeGrid control,

- Hierarchical Datasource Binding
- Self-Referential Data Binding (Flat Data)

Hierarchical Datasource Binding

The ChildMapping property is used to map the child records in hierarchical datasource.

The following code example shows you how to bind the hierarchical local data into the TreeGrid control.

C#

```
public class TreeGridController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = this.GetData();
return View();
}
private List<BusinessObject> GetData()
{
List<BusinessObject> dataCollection = new List<BusinessObject>();
BusinessObject Record1 = new BusinessObject()
{
TaskId = 1,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
TaskName = "Planning",
StartDate = "02/03/2017",
Duration = 5,
Children = new List<BusinessObject>(),
};
BusinessObject Child1 = new BusinessObject()
{
TaskId = 2,
TaskName = "Plan timeline",
StartDate = "02/03/2017",
Duration = 5
};
BusinessObject Child2 = new BusinessObject()
{
TaskId = 3,
TaskName = "Plan budget",
StartDate = "02/03/2017",
Duration = 5
};
BusinessObject Child3 = new BusinessObject()
{
TaskId = 4,
TaskName = "Allocate resources",
StartDate = "02/03/2017",
Duration = 5
};
BusinessObject Child4 = new BusinessObject()
{
TaskId = 5,
TaskName = "Planning complete",
StartDate = "02/07/2017",
Duration = 0
};
Record1.Children.Add(Child1);
Record1.Children.Add(Child2);
Record1.Children.Add(Child3);
Record1.Children.Add(Child4);
BusinessObject Record2 = new BusinessObject()
{
TaskId = 6,
TaskName = "Design",
StartDate = "02/10/2017",
Duration = 3,
Children = new List<BusinessObject>(),
};
BusinessObject Child5 = new BusinessObject()
{
TaskId = 7,
TaskName = "Software Specification",
StartDate = "02/10/2017",
Duration = 3
};
BusinessObject Child6 = new BusinessObject()
{
TaskId = 8,
TaskName = "Develop prototype",
StartDate = "02/10/2017",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Duration = 3
};
BusinessObject Child7 = new BusinessObject()
{
TaskId = 9,
TaskName = "Get approval from customer",
StartDate = "02/13/2017",
Duration = 2
};
BusinessObject Child8 = new BusinessObject()
{
TaskId = 10,
TaskName = "Design complete",
StartDate = "02/14/2017",
Duration = 0
};
Record2.Children.Add(Child5);
Record2.Children.Add(Child6);
Record2.Children.Add(Child7);
Record2.Children.Add(Child8);
dataCollection.Add(Record1);
dataCollection.Add(Record2);
return dataCollection;
}
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
public int Duration { get; set; }
public List<BusinessObject> Children { get; set; }
}
}
```

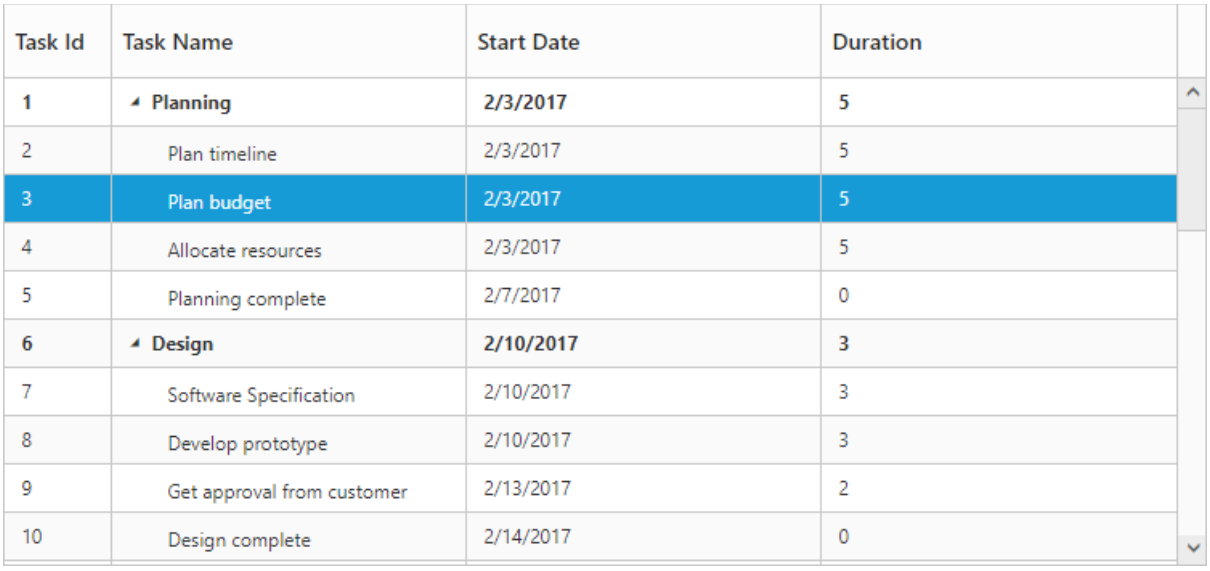
ASPX-CS

```
@using Syncfusion.JavaScript
@using Syncfusion.MVC.EJ
@using Syncfusion.JavaScript.Models
<!DOCTYPE html>
<html>
<head>
@*Add script reference and style reference here*@
</head>
<body>
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
.Columns(co =>
{
co.Field("TaskId").HeaderText("Task
Id").EditType(TreeGridEditingType.Numeric).Add();
co.Field("TaskName").HeaderText("Task
Name").EditType(TreeGridEditingType.String).Add();
co.Field("StartDate").HeaderText("Start
Date").EditType(TreeGridEditingType.Datepicker).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
co.Field("Duration").HeaderText("Duration").EditType(TreeGridEditingType.
Numeric).Add();
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
</body>
</html>
```

The output of the above steps is as follows:



Task Id	Task Name	Start Date	Duration
1	Planning	2/3/2017	5
2	Plan timeline	2/3/2017	5
3	Plan budget	2/3/2017	5
4	Allocate resources	2/3/2017	5
5	Planning complete	2/7/2017	0
6	Design	2/10/2017	3
7	Software Specification	2/10/2017	3
8	Develop prototype	2/10/2017	3
9	Get approval from customer	2/13/2017	2
10	Design complete	2/14/2017	0

Click here to view the online demo sample for hierarchical databinding.

Self-Referential Data Binding (Flat Data)

TreeGrid is rendered from Self-Referential data structures by providing two fields, **ID** field and **Parent ID** field.

- **ID Field**: This field contains unique values used to identify nodes. Its name is assigned to the IdMapping property.
- **Parent ID Field**: This field contains values that indicate parent nodes. Its name is assigned to the ParentIdMapping property.

C#

```
public class TreeGridController : Controller
{
public ActionResult Index()
{
ViewBag.datasource = this.GetSelfDataSource();
return View();
}
private List<BusinessObject> GetSelfDataSource()
{
List<BusinessObject> BusinessObjectCollection = new
List<BusinessObject>();
BusinessObjectCollection.Add(new BusinessObject()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
TaskId = 1,
TaskName = "Parent Task 1",
StartDate = "10/23/2017",
Duration = 10,
ParentId = null
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 2,
TaskName = "Child task 1",
StartDate = "10/23/2017",
Duration = 4,
ParentId = 1
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 3,
TaskName = "Child Task 2",
StartDate = "10/24/2017",
Duration = 5,
ParentId = 1
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 4,
TaskName = "Child task 3",
StartDate = "10/25/2017",
Duration = 6,
ParentId = 1
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 5,
TaskName = "Parent Task 2",
StartDate = "10/23/2017",
Duration = 10,
ParentId = null
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 6,
TaskName = "Child task 1",
StartDate = "10/23/2017",
Duration = 4,
ParentId = 5
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 7,
TaskName = "Child Task 2",
StartDate = "10/24/2017",
Duration = 5,
ParentId = 5
});
BusinessObjectCollection.Add(new BusinessObject()
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
TaskId = 8,
TaskName = "Child task 3",
StartDate = "10/25/2017",
Duration = 6,
ParentId = 5
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 9,
TaskName = "Child task 4",
StartDate = "10/25/2017",
Duration = 6,
ParentId = 5
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 10,
TaskName = "Parent Task 3",
StartDate = "10/23/2017",
Duration = 10,
ParentId = null
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 11,
TaskName = "Child task 1",
StartDate = "10/23/2017",
Duration = 4,
ParentId = 10
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 12,
TaskName = "Child Task 2",
StartDate = "10/24/2017",
Duration = 5,
ParentId = 10
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 13,
TaskName = "Child task 3",
StartDate = "10/25/2017",
Duration = 6,
ParentId = 10
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 14,
TaskName = "Child task 4",
StartDate = "10/25/2017",
Duration = 6,
ParentId = 10
});
BusinessObjectCollection.Add(new BusinessObject()
{
TaskId = 15,
```

Copyright © 2001 - 2020 Syncfusion Inc.

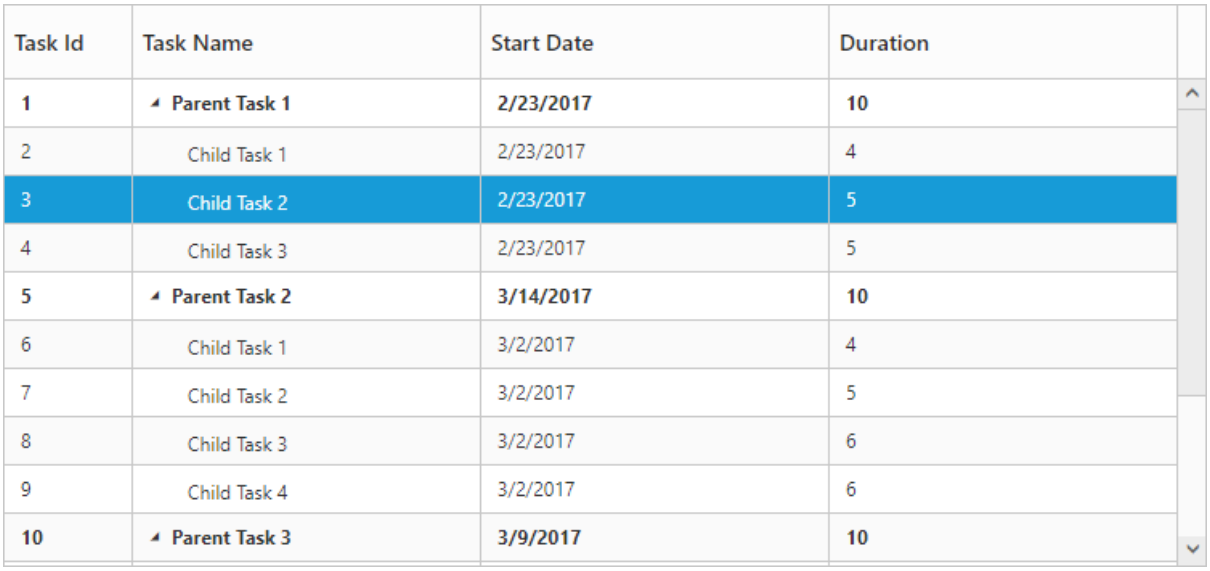
```
TaskName = "Child task 5",
StartDate = "10/25/2017",
Duration = 6,
ParentId = 10
});
return BusinessObjectCollection;
}
public class BusinessObject
{
public int TaskId { get; set; }
public string TaskName { get; set; }
public string StartDate { get; set; }
public int Duration { get; set; }
public int? ParentId { get; set; }
}
}
```

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.IdMapping("TaskId")
.ParentIdMapping("ParentId")
.Columns(co =>
{
co.Field("TaskId").HeaderText("Task
Id").EditType(TreeGridEditingType.Numeric).Add();
co.Field("TaskName").HeaderText("Task
Name").EditType(TreeGridEditingType.String).Add();
co.Field("StartDate").HeaderText("Start
Date").EditType(TreeGridEditingType.Datepicker).Add();
co.Field("Duration").HeaderText("Duration").EditType(TreeGridEditingType.
Numeric).Add();
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of the above steps,

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	Duration
1	Parent Task 1	2/23/2017	10
2	Child Task 1	2/23/2017	4
3	Child Task 2	2/23/2017	5
4	Child Task 3	2/23/2017	5
5	Parent Task 2	3/14/2017	10
6	Child Task 1	3/2/2017	4
7	Child Task 2	3/2/2017	5
8	Child Task 3	3/2/2017	6
9	Child Task 4	3/2/2017	6
10	Parent Task 3	3/9/2017	10

Click here to view the online demo sample for self-referential databinding.

Remote data binding

Load on demand

TreeGrid provides Load on Demand support for rendering remote data. Load on demand is considered in TreeGrid for the following actions,

- Expanding root nodes.
- Navigating pages, with paging enabled in TreeGrid.

When load on demand is enabled all the root nodes are rendered in collapsed state at initial load.

When load on demand support is enabled in TreeGrid with paging, the current or active page's root node alone will be rendered in collapsed state. On expanding the root node the child nodes will be loaded from the remote server.

When the root node is expanded its child nodes are rendered and are cached locally, such that on consecutive expand/collapse actions on root node, the child nodes are loaded from the cache instead from the remote server.

Similarly if the user navigates to a new page, the root nodes of that specific page will be rendered with request to the remote server.

Note: 1. Load on demand support in TreeGrid can be enabled only for remote data.

2. For better initial load time performance, we need to define the HasChildMapping property.

Load on demand support in TreeGrid can be enabled by using the property EnableLoadOnDemand.

The following code explains how to use Load on Demand in TreeGrid Control,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.IdMapping("TaskID")
.ParentIdMapping("ParentID")
.HasChildMapping("isParent")
```

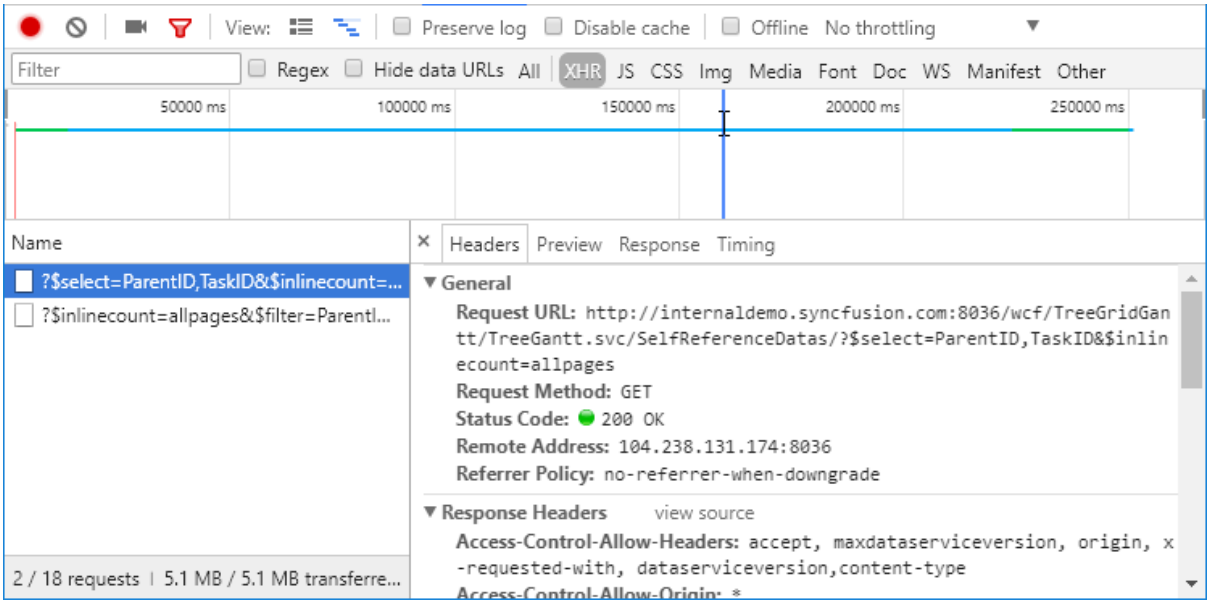
Copyright © 2001 - 2020 Syncfusion Inc.

```
.EnableLoadOnDemand(true)
.EnableVirtualization(true)
.Columns(co =>
{
co.Field("TaskID").HeaderText("Task Id").Width(45).Add();
co.Field("TaskName").HeaderText("Task Name").Add();
co.Field("StartDate").HeaderText("Start Date").Add();
co.Field("EndDate").HeaderText("End Date").Add();
co.Field("Progress").HeaderText("Progress").Add();
})
.Datasource(dataSource =>
dataSource.URL("http://js.syncfusion.com/demos/ejServices/Wcf/TreeGridGantt/TreeGantt.svc/SelfReferenceDatas").CrossDomain(true))
)
@(Html.EJ().ScriptManager())
```

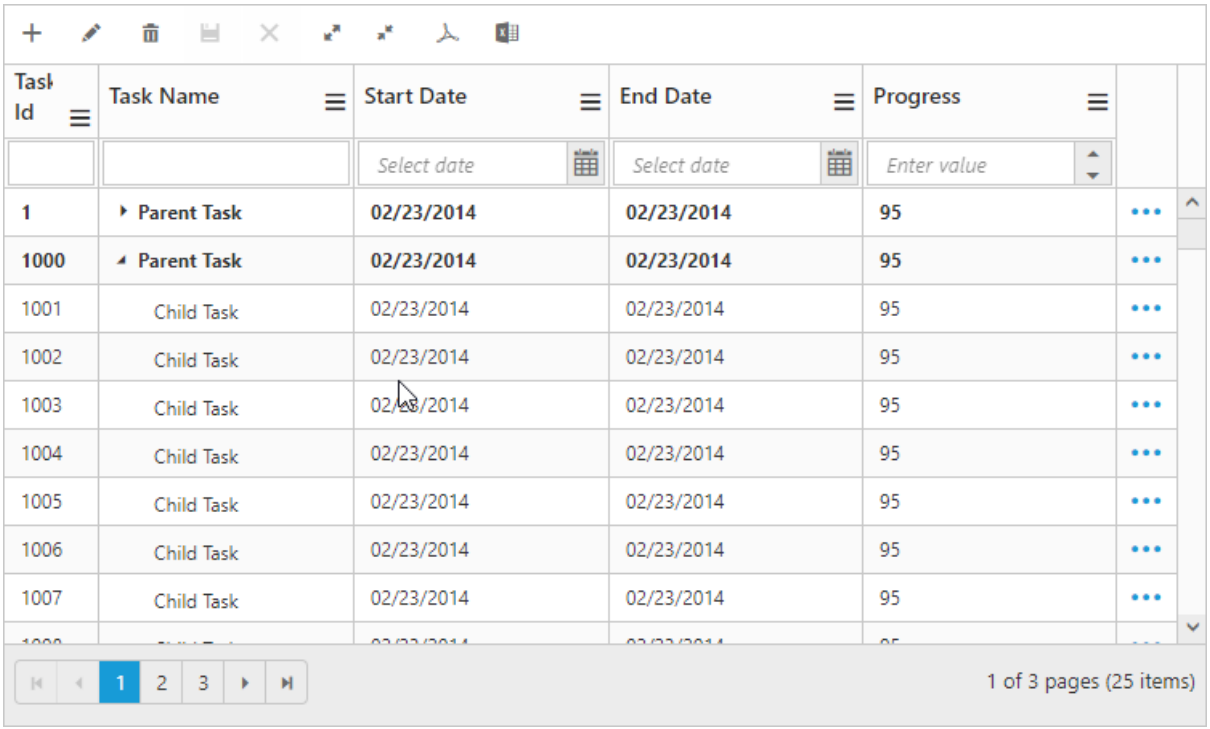
The output for load on demand support in TreeGrid:

Task Id	Task Name	Start Date	End Date	Progress
1	Parent Task	02/23/2014	02/23/2014	95
1000	Parent Task	02/23/2014	02/23/2014	95
2000	Parent Task	02/23/2014	02/23/2014	95
3000	Parent Task	02/23/2014	02/23/2014	95
4000	Parent Task	02/23/2014	02/23/2014	95
5000	Parent Task	02/23/2014	02/23/2014	95
6000	Parent Task	02/23/2014	02/23/2014	95
7000	Parent Task	02/23/2014	02/23/2014	95
8000	Parent Task	02/23/2014	02/23/2014	95
9000	Parent Task	02/23/2014	02/23/2014	95
10000	Parent Task	02/23/2014	02/23/2014	95
11000	Parent Task	02/23/2014	02/23/2014	95
12000	Parent Task	02/23/2014	02/23/2014	95

Copyright © 2001 - 2020 Syncfusion Inc.



The following output shows how load on demand works for expanding action



Click here to view the online demo sample for Load on demand.

Load at once:

On remote data binding, for every action such as paging, sorting, filtering the data will be fetched from the remote server each time. To avoid requesting the data from the remote server for each action, we can set TreeGrid to load all the data on initialization and make all the data operations in client side. To enable this, we have to set `Offline` property of `ej.DataManager` as `true`, the following code example explains this.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.IdMapping("TaskID")
.ParentIdMapping("ParentID")
.HasChildMapping("isParent")
.EnableVirtualization(true)
.Columns(co =>
{
co.Field("TaskID").HeaderText("Task Id").Width(45).Add();
co.Field("TaskName").HeaderText("Task Name").Add();
co.Field("StartDate").HeaderText("Start Date").Add();
co.Field("EndDate").HeaderText("End Date").Add();
co.Field("Progress").HeaderText("Progress").Add();
})
.Datasource(dataSource =>
dataSource.URL("http://js.syncfusion.com/demos/ejServices/Wcf/TreeGridGan
tt/TreeGantt.svc/SelfReferenceDatas").Offline(true))
)
@(Html.EJ().ScriptManager())
```

Please refer the link for further reference on offline property

Limitations:

1. Mapping the expand state of a record using ExpandStateMapping property is not supported in load on demand feature.
2. If a root or parent node is in collapsed state (child nodes not yet loaded), then that parent node will not be expanded while inserting new child to that parent node using toolbar icon or drag and drop actions.

Virtual rendering

Virtualization support is used to render large number of records in TreeGrid with effective performance. In this mode all the records are fetched from data source initially, but only few records will be displayed in the document object model (DOM) which should be visible to the user. While scrolling, the visible records are updated in DOM as per the scrolled position. This mode can be enabled by setting EnableVirtualization property as true.

The below code example shows how to use this property.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.EnableVirtualization(true)
)
@(Html.EJ().ScriptManager())
```

Click here to view the online demo sample for virtualization.

Columns

Column definitions specified in the Columns option defines how the data in the **DataSource** have to be displayed, formatted and edited in TreeGrid. The values in the **DataSource** can be mapped to the appropriate column using the Field property of the corresponding column object.

Copyright © 2001 - 2020 Syncfusion Inc.

Editing type

The edit type of a column can be defined using the EditType property of the column object.

The following example shows how to define the EditType in a column,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col=>
{
col.EditType(TreeGridEditingType.Boolean).Add();
col.EditType(TreeGridEditingType.Numeric).Add();
})
@(Html.EJ().ScriptManager())
```

The column editors can be further customized by using the EditOptions property of column object.

The following example shows how to define additional properties to the date picker control,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("StartDate").EditType(TreeGridEditingType.Datepicker).DateEditO
ptions(new Syncfusion.JavaScript.Models.DatePickerProperties
{
HighlightWeekend = true
}).Add();
})
)
@(Html.EJ().ScriptManager())
```

Formatting

The values in each column can be formatted using the Format property of the column object.

The following code example shows how to format the numeric and date values in TreeGrid column,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("Percentage").HeaderText("Percentage").Format("{0:P0}").Add();
col.Field("Currency").HeaderText("Currency").Format("{0:C2}").Add();
col.Field("startDate").HeaderText("Start
Date").Format("{0:MM/dd/yyyy}").Add();
col.Field("endDate").HeaderText("End Date").Format("{0:MM/dd/yyyy
hh:mm:ss}").Add();
})
)
@(Html.EJ().ScriptManager())
```

Note: 1.For more numeric format strings, please refer this link.

2.For more date format strings, please refer this link.

Copyright © 2001 - 2020 Syncfusion Inc.

Defining column width

In TreeGrid, it is possible to define width for a specific column by setting Width property to the column object.

The below code snippet shows how to set width for specific column,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Width(50).Add();
col.Field("taskName").HeaderText("Task Name").Width(150).Add();
col.Field("startDate").HeaderText("Start Date").Width(100).Add();
col.Field("endDate").HeaderText("End Date").Width(100).Add();
col.Field("duration").HeaderText("Duration").Width(100).Add();
})
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the TreeGrid rendered with specific column width values.

Task Id	Task Name	Start Date	End Date	Duration
1	Planning	2/3/2014	2/7/2014	5
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	Design	2/10/2014	2/14/2014	3
12	Implementation Phase	2/17/2014	2/27/2014	11
13	Phase 1	2/17/2014	2/27/2014	11
14	Implementation Modul	2/17/2014	2/27/2014	11
15	Development Task 1	2/17/2014	2/19/2014	3
16	Development Task 2	2/17/2014	2/19/2014	3
17	Testing	2/20/2014	2/21/2014	2

Defining common width for the columns

The TreeGrid control provides the support to set same width for all the available columns and this can be done by using CommonWidth property.

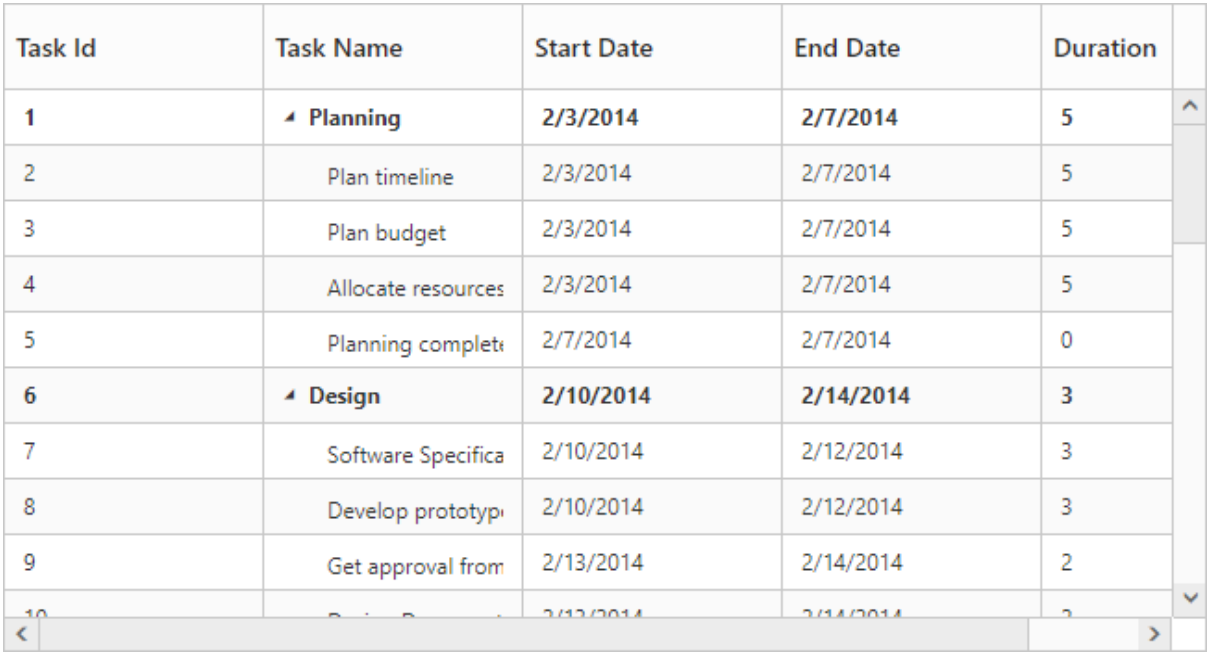
ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.CommonWidth(140)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
```

The below screenshot shows TreeGrid render with common width.



Headers

Header text

Using the HeaderText property, you can provide the title for a specific column. The below code snippet shows how to set HeaderText for TreeGrid columns.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Add();
col.Field("taskName").HeaderText("Task Name").Add();
col.Field("startDate").HeaderText("Start Date").Add();
col.Field("endDate").HeaderText("End Date").Add();
})
)
@(Html.EJ().ScriptManager())
```

Text wrapping

It is possible to wrap the header text or the title of the column when the content exceeds the column width using HeaderTextOverflow property. By default this property is set to **None**.

To enable wrapping of header text, you have to set the HeaderTextOverflow property as **Wrap**.

The below code snippet demonstrates this.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.HeaderTextOverflow(TreeGridHeaderTextOverflow.Wrap)
)
@(Html.EJ().ScriptManager())
```

Click here to view the online demo sample for text wrapping.

Header Template

TreeGrid column's header template was defined by using HeaderTemplateID property. The value of this property should be a valid JsRender template.

The following code snippet shows how to set the header template for TreeGrid columns,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskName").EditType(TreeGridEditingType.String).HeaderTemplate
ID("#projectName").Add();
col.Field("startDate").EditType(TreeGridEditingType.Datepicker).Add();
col.Field("resourceId").EditType(TreeGridEditingType.Dropdown).DropDownDa
ta((IEnumerable<object>)ViewBag.projectResources).HeaderTemplateID("#reso
urce").Add();
col.Field("progress").EditType(TreeGridEditingType.Numeric).Add();
})
)
@(Html.EJ().ScriptManager())
<script type="text/x-jsrender" id="resource">
<div>
<div class="name">
<img src="13.4.0.53/themes/web/images/treegrid/icon-03.png" width="20"
height="20" />
</div>
<div style="position:relative; top:3px;">
Resources
</div>
</div>
</script>
<script type="text/x-jsrender" id="projectName">
<div>
<div>
<img src="13.4.0.53/themes/web/images/treegrid/icon-01.png" width="20"
height="20" />
</div>
<div style="position:relative; top:3px;">
Task Name
</div>
</div>
</script>
```

The below screenshot depicts column headers with custom templates.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Name	startDate	Resources	progress
Planning	02/03/2014		
Plan timeline	02/03/2014	Software Analyst	100
Plan budget	02/03/2014	Project Manager	100
Allocate resources	02/03/2014	Project Manager	100

Click here to view the online demo sample for header template.

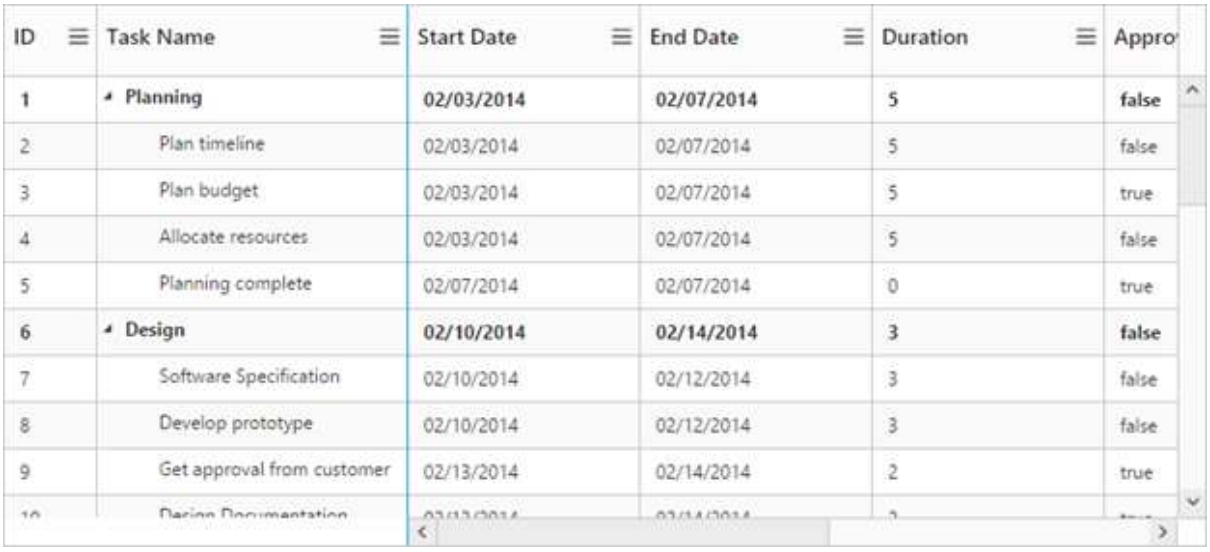
Frozen Columns

Specific columns can be frozen by enabling the IsFrozen property of the respective column object. The columns which are frozen remain static while scrolling the content horizontally. You can also freeze or unfreeze a column during runtime, by selecting Freeze or Unfreeze menu item in the column menu. These set of menu options will be displayed in all the columns when the IsFrozen property is enabled in any of the columns. However you can control the visibility of these menu options in a particular column by enabling/disabling the AllowFreezing property of that specific column.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowColumnChooser(true)
.Columns(co =>
{
co.Field("TaskId").HeaderText("ID").Width(100).IsFrozen(true).AllowFreezi
ng(false).Add();
co.Field("TaskName").HeaderText("Task
Name").Width(200).IsFrozen(true).Add();
co.Field("StartDate").HeaderText("Start Date").Add();
co.Field("EndDate").HeaderText("End Date").Add();
co.Field("Duration").HeaderText("Duration").Add();
})
)
@(Html.EJ().ScriptManager())
```

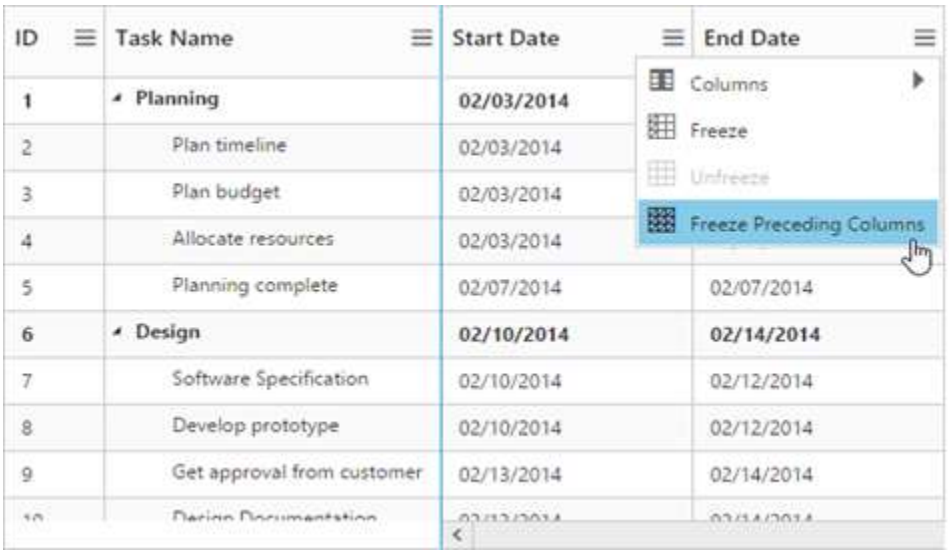
The below screenshot depicts TreeGrid with frozen columns,



ID	Task Name	Start Date	End Date	Duration	Appro
1	Planning	02/03/2014	02/07/2014	5	false
2	Plan timeline	02/03/2014	02/07/2014	5	false
3	Plan budget	02/03/2014	02/07/2014	5	true
4	Allocate resources	02/03/2014	02/07/2014	5	false
5	Planning complete	02/07/2014	02/07/2014	0	true
6	Design	02/10/2014	02/14/2014	3	false
7	Software Specification	02/10/2014	02/12/2014	3	false
8	Develop prototype	02/10/2014	02/12/2014	3	false
9	Get approval from customer	02/13/2014	02/14/2014	2	true

Copyright © 2001 - 2020 Syncfusion Inc.

It is also possible to freeze all the preceding columns at run-time by choosing **Freeze Preceding Columns** option in the column menu or by using the freezePrecedingColumns method, the column field name, for which the columns preceding it to be frozen should be passed as the method parameter.



Freeze columns using method

Columns can also be frozen or unfrozen on any custom actions using the freezeColumn method.

The column's field name which is to be frozen/unfrozen should be passed as the method parameter, along with the freeze state.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//.
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
var treegridObj = $("#TreeGridContainer").data("ejTreeGrid");
treegridObj.freezeColumn(field, true);
</script>
```

Click here to view the online demo sample for frozen column.

Resizing

You can resize the column width to view the hidden text of the cell. This feature can be enabled by setting the AllowColumnResize property to true.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowColumnResize(true)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Column resize mode

In TreeGrid, it is possible to provide different column resizing mode by using ResizeMode property of ColumnResizeSettings property.

The below are the types of column resize modes available in TreeGrid,

- Normal - Columns are stretched with control width at load time. When resizing the column, the current column width is updated based on next column.
- Next column - Columns are stretched with control width at load time. When resize the column the current column width is updated based on stretching columns in control width.
- Fixed Columns - Column are rendered with given width value at load time. Only the current column width is changed while resizing the column.

The following code snippet explains how to set column resize mode in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ColumnResizeSettings(cr=>
{
cr.ResizeMode(TreeGridColumnResizeMode.FixedColumns);
})
)
@(Html.EJ().ScriptManager())
```

Task Id	Task Name	Start Date	End Date	Duration
1	**Planning**	**2/3/2014**	**2/7/2014**	**5**
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	**Design**	**2/10/2014**	**2/14/2014**	**3**
7	Software Specificati	2/10/2014	2/12/2014	3
8	Develop prototype	2/10/2014	2/12/2014	3
9	Get approval from c	2/13/2014	2/14/2014	2
10	Design Documentat	2/13/2014	2/14/2014	2

The above screenshot shows the TreeGrid rendered with FixedColumns resize mode.

Click here to view the online demo sample for column resizing.

Checkbox column

It is possible to display a column value as checkbox in TreeGrid by enabling the DisplayAsCheckbox property and by setting the EditType property as Boolean for that column.

Copyright © 2001 - 2020 Syncfusion Inc.

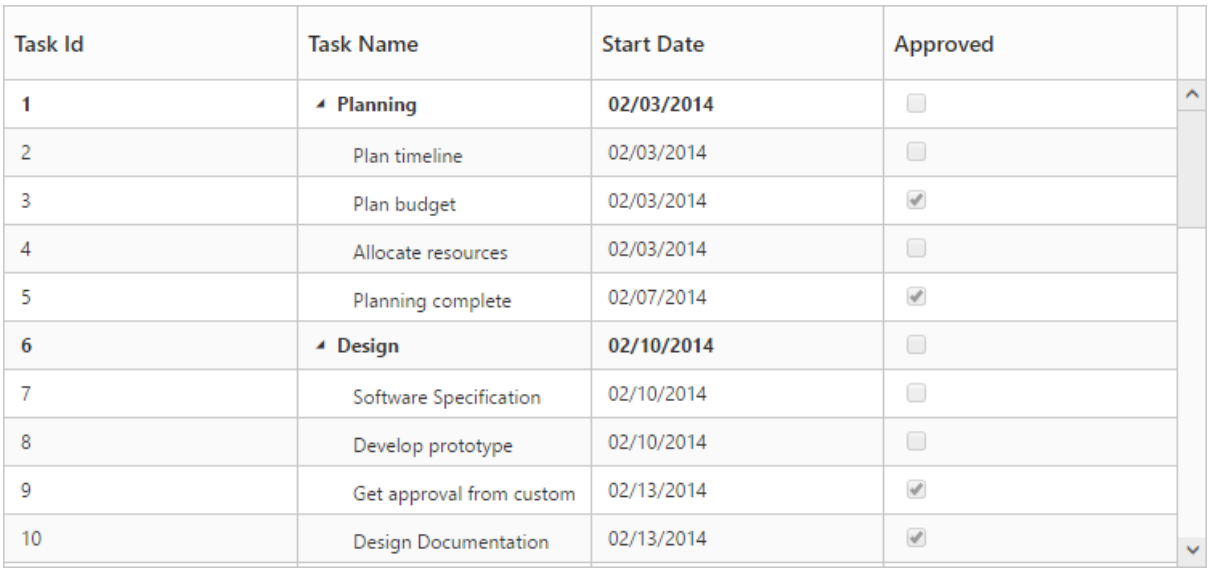
If the DisplayAsCheckbox property was set as false, then the column value will be displayed as string with the value mapped from the data source.

The following code snippet explains how to display a checkbox column in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("approved").HeaderText("Approved").EditType(TreeGridEditingType
.Boolean).DisplayAsCheckbox(true).Add();
})
)
@(Html.EJ().ScriptManager())
```

The below screen shot depicts the Approved column in TreeGrid displayed as a checkbox column.



Task Id	Task Name	Start Date	Approved
1	Planning	02/03/2014	
2	Plan timeline	02/03/2014	
3	Plan budget	02/03/2014	
4	Allocate resources	02/03/2014	
5	Planning complete	02/07/2014	
6	Design	02/10/2014	
7	Software Specification	02/10/2014	
8	Develop prototype	02/10/2014	
9	Get approval from custom	02/13/2014	
10	Design Documentation	02/13/2014	

Column Template

Column Template is used to customize the column's look and feel based on requirement.

The following code example shows you how to display the icon in the TreeGrid column,

- TemplateID - Using the TemplateID property, you can specify the Id of the script element, which contains the template for the column.
- Template - HTML templates can be specified in the Template property of the particular column as a string (HTML element).

Column template support was enabled for that column by setting IsTemplateColumn as true.

The following code example show how to define template for the column,

ASPX-CS

```
// JsRender template definition.
<script type="text/x-jsrender" id="columnTemplate">
{{if #data['resourceNames']}}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div style="display:inline-block;position:relative;left:10px;top:1px">
<img src="../images/gantt/{{:#data['resourceNames']}}.png" height="40" />
</div>
{{/if}}
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.RowHeight(50)
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Width(45).Add();
col.Field("taskName").HeaderText("Task Name").Add();
col.HeaderText("Resource").IsTemplateColumn(true).TemplateID("columnTempl
ate").TextAlign(TextAlign.Center).Add();
col.Field("resourceNames").HeaderText("Resource Name").Add();
})
)
@(Html.EJ().ScriptManager())
```



Click here to view the online demo sample for frozen column.

Column Menu

Column menu can be displayed in column header by setting ShowColumnChooser property as true.

The following are the items displayed in the column menu,

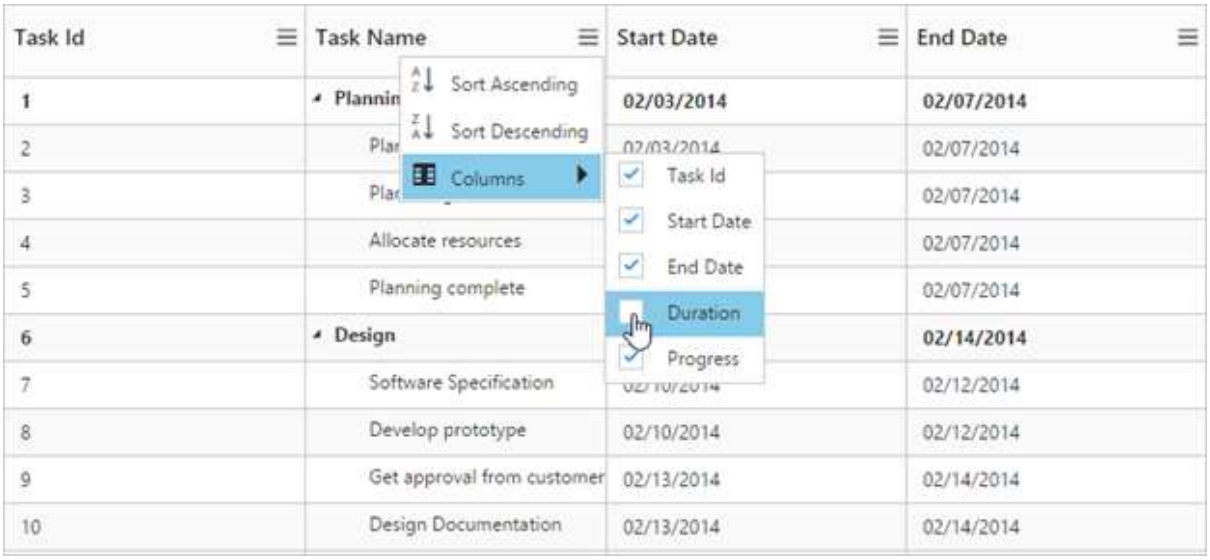
- **Column Chooser** – Displays all the column names, you can enable or disable a column by check or uncheck the respective column name in the column chooser menu.
- **Sort Ascending & Sort Descending** – Used to sort the items in the column. These menu options will be displayed only when you set the AllowSorting property as true. To perform multilevel sorting, the AllowMultiSorting property should be enabled.

Copyright © 2001 - 2020 Syncfusion Inc.

- **Freeze, Unfreeze & Freeze Preceding Columns** – Used to freeze or unfreeze the columns. These set of menu options will be displayed in all the columns when the IsFrozen property is enabled in any of the columns. However, you can control the visibility of these menu options in a particular column by enabling/disabling the AllowFreezing property of that specific column.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowColumnChooser(true)
.AllowSorting(true)
.AllowMultiSorting(true)
.Columns(col =>
{
col.Field("duration").HeaderText("Duration").Visible(false).Add();
})
)
@(Html.EJ().ScriptManager())
```



The column menu also provides support for some of the additional column options such as,

- Insert column left
- Insert column right
- Delete column
- Rename column

The column options can be enabled or disabled with the ShowColumnOptions property, default value of this property is false.

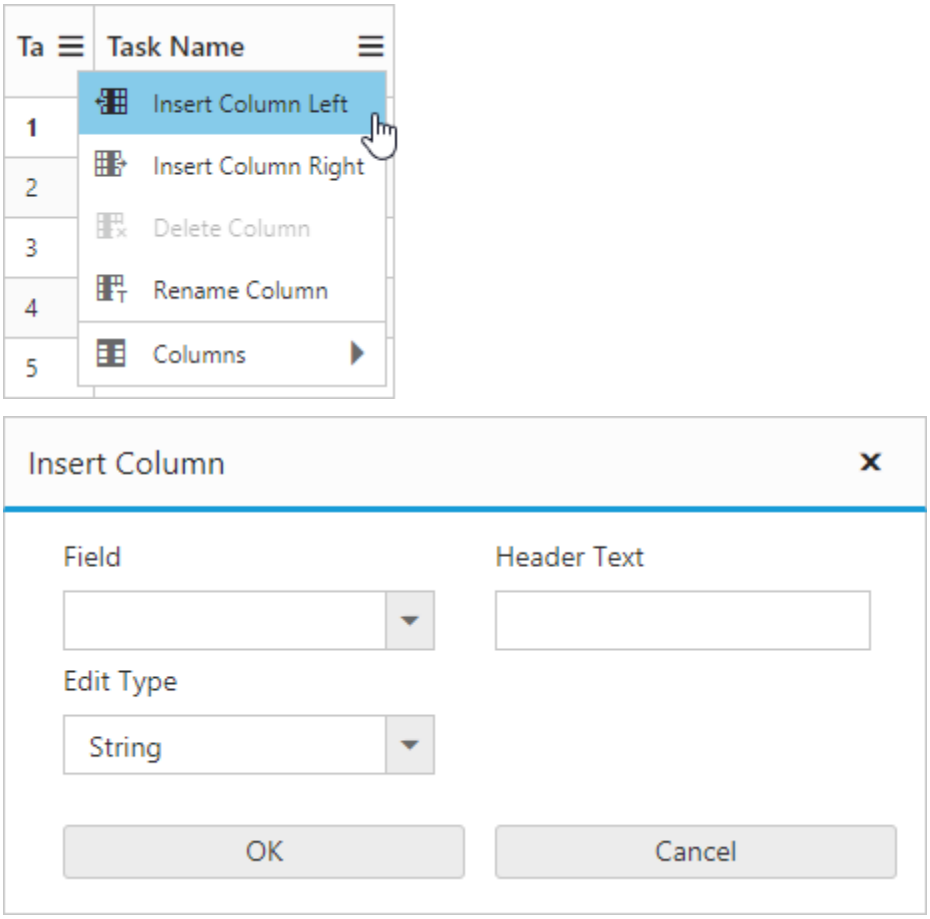
The following code example shows how to enable the column option in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowColumnOptions(true)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
```



The above screenshot shows insert column dialog in TreeGrid

The TreeGrid columns can also be renamed or deleted at run-time on any custom actions using the renameColumn and deleteColumn methods.

Customize insert column dialog fields

It is possible to add or remove the Columns properties in insert column dialog using ColumnDialogFields property. In insert column dialog Field, HeaderText and EditType properties are necessary to create a new column, so these fields are unable to remove from insert column option.

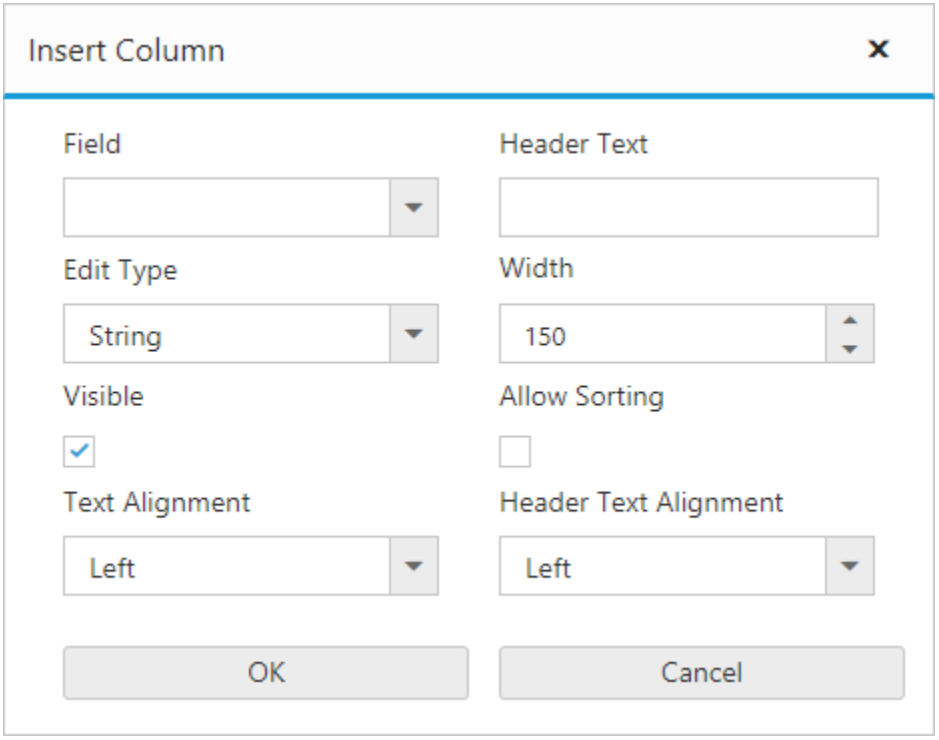
The following code example shows how to customize the insert column option in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowSorting(true)
.ShowColumnOptions(true)
.ColumnDialogFields(new List<TreeGridColumnDialogFields>()
{
TreeGridColumnDialogFields.Field,
TreeGridColumnDialogFields.HeaderText,
TreeGridColumnDialogFields.EditType,
TreeGridColumnDialogFields.Width,
TreeGridColumnDialogFields.Visible,
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
TreeGridColumnDialogFields.AllowSorting,
TreeGridColumnDialogFields.TextAlign,
TreeGridColumnDialogFields.HeaderTextAlign
})
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows customized insert column dialog in TreeGrid.

Hide specific column in column chooser list

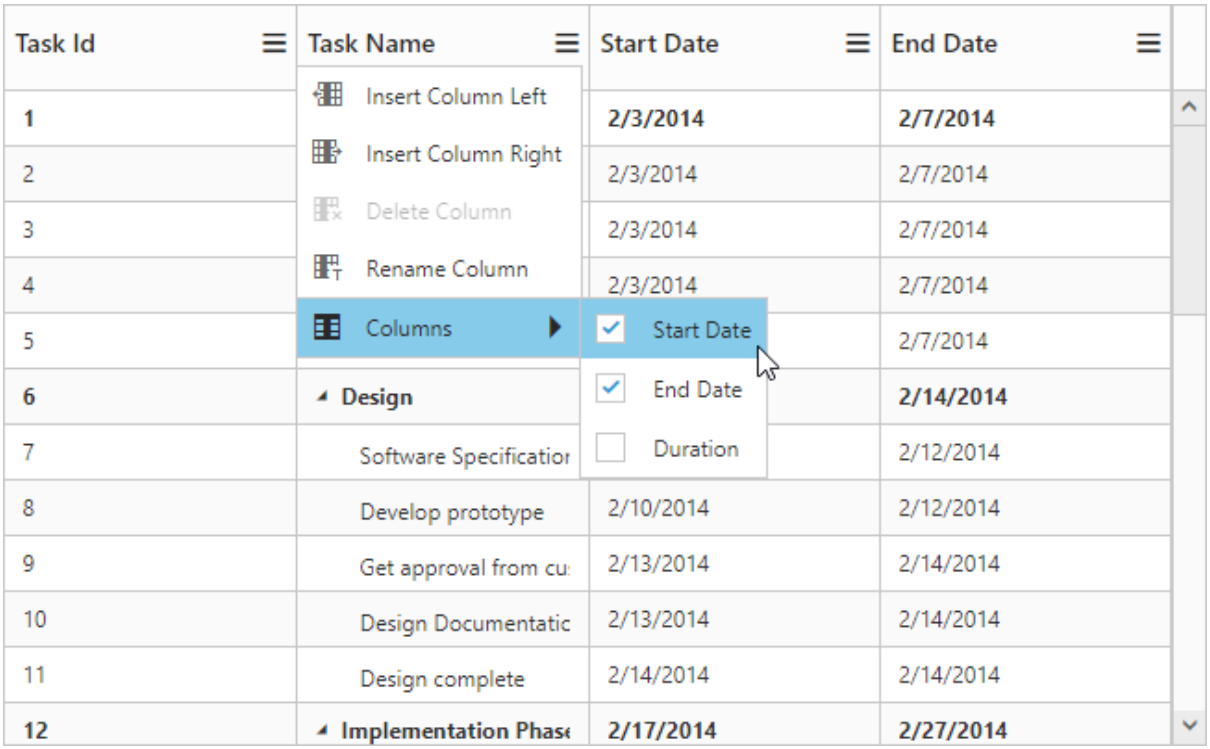
It is possible to hide the specific column in column chooser list by settings ShowInColumnChooser as false in the column definition.

The following code example shows how to hide specific column in column chooser list,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowColumnChooser(true)
.Columns(col =>
{
col.Field("taskID").HeaderText("Task
Id").ShowInColumnChooser(false).Add();
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows TreeGrid column chooser rendered without Task Id column.

Show/Hide columns using method

It is possible to toggle the visibility of the columns using the hideColumn and showColumn methods. The column's header text should be passed as the method parameter which is to be shown/hidden.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//.
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
var treegridObj = $("#TreeGridContainer").data("ejTreeGrid");
treegridObj.hideColumn("Task Name");
treegridObj.showColumn("Order ID");
</script>
```

Click here to view the online demo sample for column menu.

Command Column

Default action buttons

Using command columns in TreeGrid, we can display a separate column to perform CRUD operations. It is also possible to perform any custom actions by using custom command buttons. Command column can be defined in TreeGrid by using the Commands property.

A command column can be customized by using the Type and ButtonOptions properties.

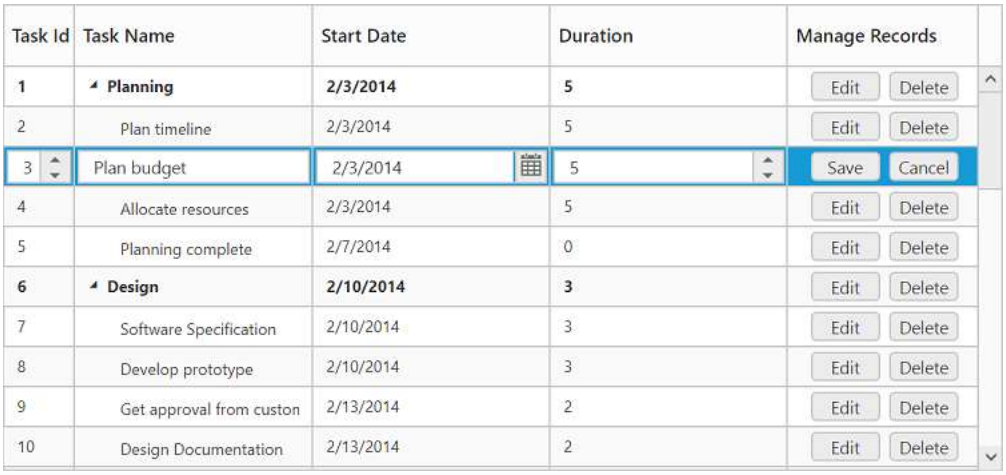
- **Type**: Using this property we can add required action buttons in TreeGrid command column such as edit,delete,save and cancel.

Copyright © 2001 - 2020 Syncfusion Inc.

- **ButtonOptions**: Using this property we can customize the button in the command column with the properties available in Button.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(co =>
{
co.HeaderText("Manage Records").Commands(command =>
{
command.Type(TreeGridUnboundType.Edit)
.ButtonOptions(new ButtonProperties()
{
Text = "Edit",
Width="58"
}).Add();
command.Type(TreeGridUnboundType.Delete)
.ButtonOptions(new ButtonProperties()
{
Text = "Delete",
Width = "58"
}).Add();
command.Type(TreeGridUnboundType.Save)
.ButtonOptions(new ButtonProperties()
{
Text = "Save",
Width = "58"
}).Add();
command.Type(TreeGridUnboundType.Cancel)
.ButtonOptions(new ButtonProperties()
{
Text = "Cancel",
Width = "58"
}).Add();
}).Add();
})
)
@(Html.EJ().ScriptManager())
```



Copyright © 2001 - 2020 Syncfusion Inc.

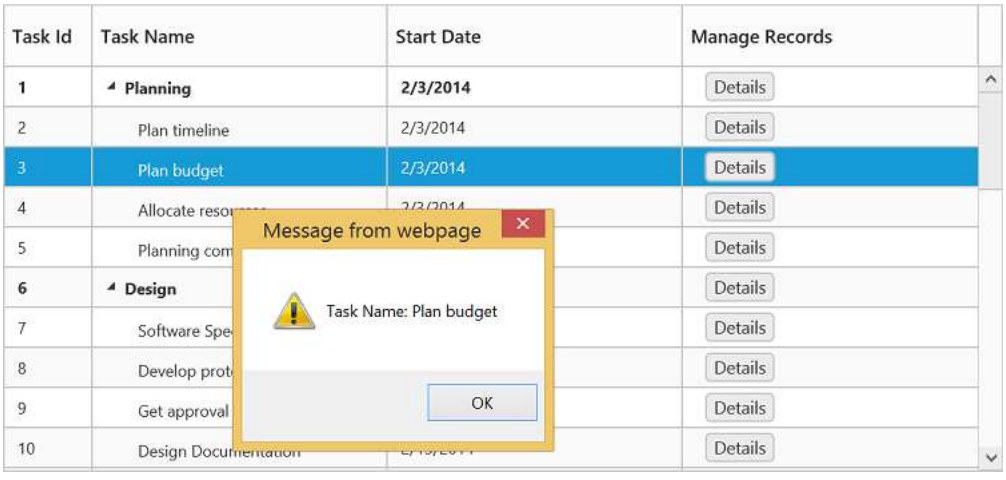
Click here to view the online demo sample for command column.

Custom buttons

We can also add custom buttons to the command column by specifying text value other than default buttons to the type property. We can also bind actions to the custom button using the Click client-side event of Button.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(co =>
{
co.HeaderText("Manage Records").Commands(command =>
{
command.Type("Details")
.ButtonOptions(new ButtonProperties()
{
Text = "Details",
Width = "58",
Click = "onClick"
}).Add();
}).Add();
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function onClick(args) {
var $tr = $(args.e.target).closest('tr'),
treeObj = $("#TreeGridContainer").data("ejTreeGrid"),
rowIndex = treeObj.getIndexByRow($tr),
record = treeObj.model.currentViewData[rowIndex];
alert("Task Name: " + record.item.TaskName);
}
</script>
```



Tree column/ Expander column

The position of the expander column/tree column can be changed by using the TreeColumnIndex property.

The following code example shows how to change the position of the expander column,

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.TreeColumnIndex(2)
)
@(Html.EJ().ScriptManager())
```

The TreeColumnIndex can also be changed at run-time by using the columnIndex method.

Visibility

The visibility of TreeGrid column can be customized by usingVisible property. TreeGrid columns column be hidden by setting Visible property as false.

The following code example explains how to hide the fourth column,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Width(45).Add();
col.Field("taskName").HeaderText("Task Name").Add();
col.Field("startDate").HeaderText("Start Date").Add();
col.Field("endDate").HeaderText("End Date").Visible(false).Add();
col.Field("progress").HeaderText("Progress").Add();
})
)
@(Html.EJ().ScriptManager())
```

Task Id	Task Name	Start Date	Progress
1	Planning	02/03/2014	100
2	Plan timeline	02/03/2014	100
3	Plan budget	02/03/2014	100
4	Allocate resources	02/03/2014	100
5	Planning complete	02/07/2014	0
6	Design	02/10/2014	86
7	Software Specification	02/10/2014	60
8	Develop prototype	02/10/2014	100
9	Get approval from customer	02/13/2014	100
10	Design Documentation	02/13/2014	100

Read-only

A column can be made read-only by setting the AllowEditing property as false.

Note: By setting Columns.AllowEditing as false that specific column alone is made as read-only, and by setting the EditSettings.AllowEditing as false the entire TreeGrid is made read-only.

The below code snippet demonstrates this,

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Width(45).Add();
col.Field("taskName").HeaderText("Task Name").AllowEditing(false).Add();
col.Field("startDate").HeaderText("Start Date").Add();
col.Field("endDate").HeaderText("End Date").Add();
})
)
@(Html.EJ().ScriptManager())
```

Validation Rules

At some occasions, we will need to validate the data before updating it to the database. In TreeGrid it is possible to validate the data while performing adding and editing actions. The validation rules must be provided in the column definition using ValidationRules property. TreeGrid has built-in support for the below validation rules.

- **maxlength** – Makes the value require a given maximum text length.
- **minlength** – Makes the value require a given minimum text length.
- **required** – Makes the value required.
- **number** – Makes the value require a decimal number.
- **range** – Makes the value require a given value range.

The below code example explains defining the validation rules for the column.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Width(45).ValidationRules(v=>
v.AddRule("required",true)).Add();
col.Field("taskName").HeaderText("Task Name").ValidationRules(v =>
v.AddRule("maxlength",5).AddRule("minlength",2)).Add();
col.Field("progress").HeaderText("Progress").ValidationRules(v =>
v.AddRule("range", new List<object>() { -1, 101 })).Add();
})
)
@(Html.EJ().ScriptManager())
```

Custom validation error messages can also be defined in the column object. The below code example explains defining the custom error message.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Width(45).ValidationRules(v=>
v.AddRule("required",true).AddRule("number",true).AddRule("messages", new
Dictionary<string, object>() { { "required", "This field was must" }
})).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
@(Html.EJ().ScriptManager())
```

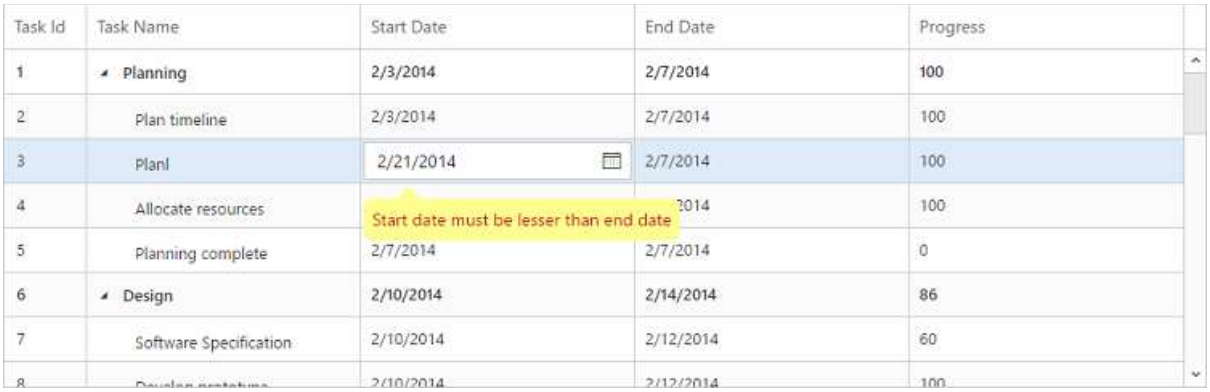
Custom Validation rules

Apart by the in-built validation rules, any custom validation rules can also be defined for the column. The below code example explains defining custom validation rule for a column.

ASPX-CS

```
<script type="text/javascript">
// Custom validation rule definition
$.validator.addMethod("customCompare", function(value, element, params) {
return element.value > params[0] && element.value < params[1]
}, "progress value must be between 0 and 100");
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("Progress").HeaderText("Progress").EditType(TreeGridEditingType
.Numeric).ValidationRules(v => v.AddRule("customCompare", new
List<object>() { -1, 101 })).Add();
})
)
@(Html.EJ().ScriptManager())
```

The below image displays the TreeGrid with validation rule applied for a date column.



Click here to view the online demo sample for TreeGrid validation support.

Column Reorder

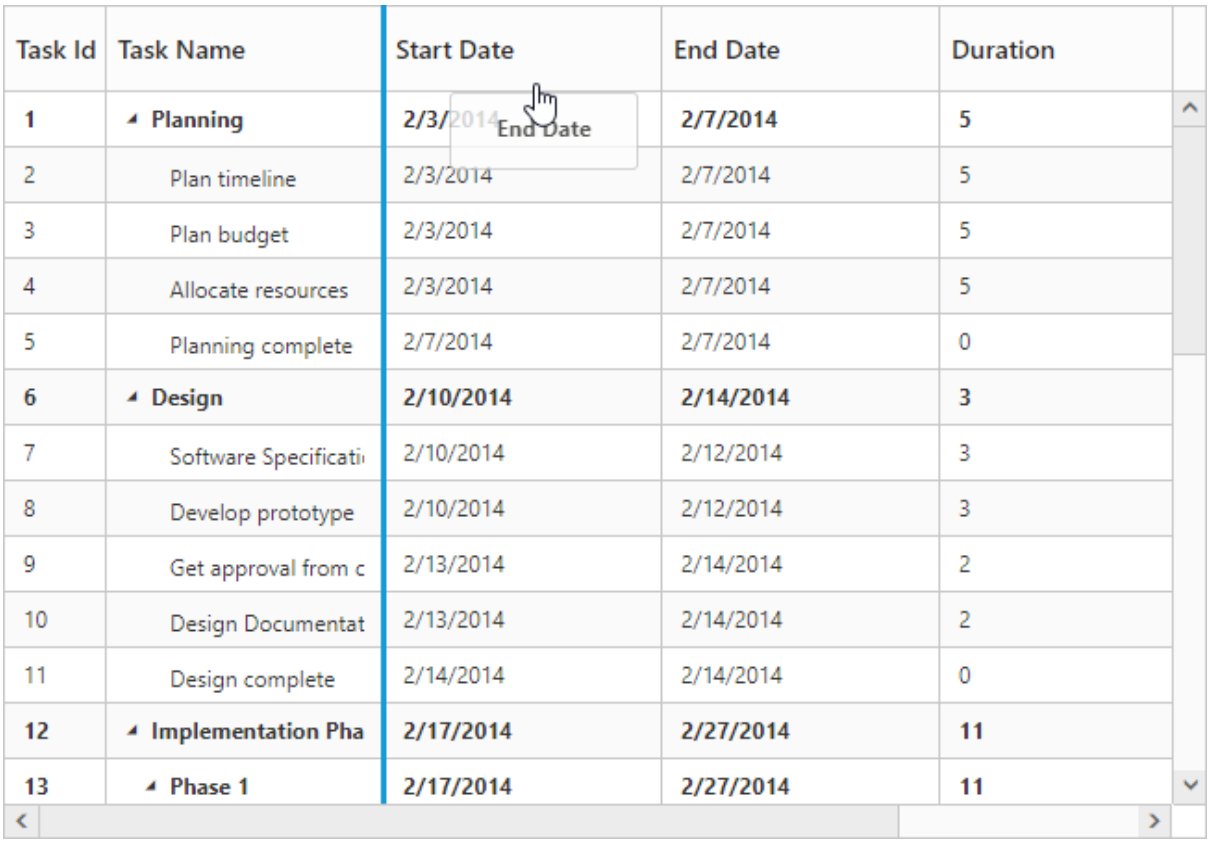
Column reorder support was used to change the order of the column by mouse interactions. In TreeGrid, AllowColumnReordering property was used to enable this support and the default value of this property was false.

The following code example explains how to enable column reorder in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowColumnReordering(true)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows the column reorder in TreeGrid.

The TreeGrid columns can also be reordered by using the reorderColumn method, where the column field name and the target index should be passed as the method parameters.

Click here to view the online demo sample for column re-ordering.

Text Alignment

In TreeGrid, it is possible to align both content and header text of particular column by using the `TextAlign` and `HeaderTextAlign` property of columns.

There are four possible ways to align content and header text of column, they are

1. Left
2. Right
3. Center
4. Justify

Note: The `TextAlign` property will affect both content and header text of the TreeGrid, when `HeaderTextAlign` is not set in the column definition.

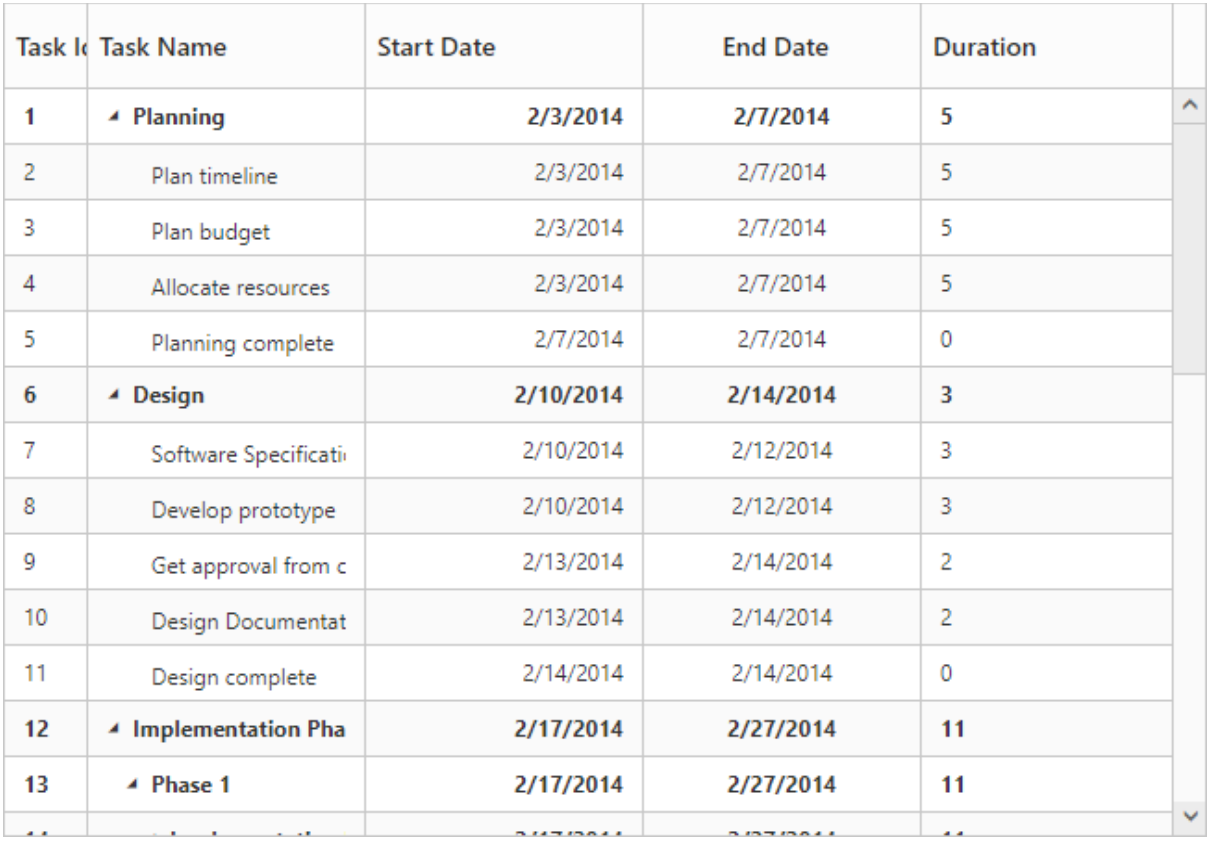
The following code example explains how to set text alignment for content and header text in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Add();
col.Field("taskName").HeaderText("Task Name").Add();
col.Field("startDate").HeaderText("Start
Date").TextAlign(TextAlign.Right).HeaderTextAlign(TextAlign.Left).Add();
col.Field("endDate").HeaderText("End
Date").TextAlign(TextAlign.Center).Add();
col.Field("duration").HeaderText("Duration").Add();
})
)
@(Html.EJ().ScriptManager())
```



Task Id	Task Name	Start Date	End Date	Duration
1	**Planning**	**2/3/2014**	**2/7/2014**	**5**
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	**Design**	**2/10/2014**	**2/14/2014**	**3**
7	Software Specificati	2/10/2014	2/12/2014	3
8	Develop prototype	2/10/2014	2/12/2014	3
9	Get approval from c	2/13/2014	2/14/2014	2
10	Design Documentat	2/13/2014	2/14/2014	2
11	Design complete	2/14/2014	2/14/2014	0
12	**Implementation Pha**	**2/17/2014**	**2/27/2014**	**11**
13	**Phase 1**	**2/17/2014**	**2/27/2014**	**11**

The above screenshot shows TreeGrid render with text alignment and header text alignment.

Customize the column at initial load

In TreeGrid, it is possible to customize the column at load time using Load event.

The following code examples shows how to customize the column at load time,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(col =>
{
col.Field("taskID").HeaderText("Task Id").Add();
col.Field("taskName").HeaderText("Task Name").Add();
col.Field("startDate").HeaderText("Start Date").Add();
col.Field("endDate").HeaderText("End Date").Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Field("duration").HeaderText("Duration").Add();
})
.ClientSideEvents(eve=>
{
eve.Load("load");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function load(args) {
var columns = args.model.columns;
columns[0].isFrozen = true;
columns[3].textAlign = "center";
columns[4].visible = false;
}
</script>
```

Task Id	Task Name	Start Date	End Date
1	Planning	2/3/2017	2/7/2017
2	Plan timeline	2/3/2017	2/7/2017
3	Plan budget	2/3/2017	2/7/2017
4	Allocate resources	2/3/2017	2/7/2017
5	Planning complete	2/7/2017	2/7/2017
6	Design	2/10/2017	2/14/2017
7	Software Specification	2/10/2017	2/12/2017
8	Develop prototype	2/10/2017	2/12/2017
9	Get approval from customer	2/13/2017	2/14/2017
10	Design Documentation	2/13/2017	2/14/2017

The above screenshot shows TreeGrid rendered with customized column.

Get Column Object

The column object which consists the list of columns available in TreeGrid and we can get the particular column details by using the getColumnByHeaderText and getColumnByField methods.

Similarly we can get the column index by using the getColumnIndexByField method. To use this method we have to pass the field name as parameter. And we can get the field name of the column by using its `HeaderText` value by using getFieldNameByHeaderText method.

Rows

The TreeGrid rows displays the information of each row from the bounded data source.

Row Template

Row template is used to customize the TreeGrid rows based on requirements. In TreeGrid, `RowTemplateId` and `AltRowTemplateId` properties are used for customizing the row.

Copyright © 2001 - 2020 Syncfusion Inc.

- **RowTemplateId** is used to customize all the rows in TreeGrid. For this property, ID of the row template is to be provided.
- **AltRowTemplateId** is used to customize the alternative rows in TreeGrid. For this property, ID of the alternative row template is to be provided.

ASPX-CS

```
<style>
.border {
border-color: #BDBDBD;
border-width: 1px;
border-style: solid;
}
</style>
<script id="rowTemplateScript" type="text/x-jsrender">
<tr>
<td class="border" style='height:99px;'>
<div>{{"{{"}}:#data['EmployeeID']{{}}}}</div>
</td>
<td class="border" style='height:99px;'>
<div style="font-size:14px;">
{{"{{"}}:#data['Name']{{}}}}
<p style="font-size:8px;">{{"{{"}}:#data['Designation']{{}}}}</p>
</div>
</td>
<td class="border" style='height:99px;'>
<div>
<div style="position:relative;top:-25px;display:inline-block;">
<img src="../Images/treegrid/{{:#data['FullName']}}.png" />
</div>
<div style="display:inline-block;">
<div style="padding:5px;">{{"{{"}}:#data['Address']{{}}}}</div>
<div style="padding:5px;">{{"{{"}}:#data['Country']{{}}}}</div>
<div style="padding:5px;font-
size:12px;">{{"{{"}}:#data['Contact']{{}}}}</div>
</div>
</div>
</td>
<td class="border" style='height:99px;'>
<div>{{"{{"}}:#data['DOB']{{}}}}</div>
</td>
</tr>
</script>
<script id="altRowTemplateScript" type="text/x-jsrender">
<tr>
<td class="border" style='height:99px;'>
<div>{{"{{"}}:#data['EmployeeID']{{}}}}</div>
</td>
<td class="border" style='height:99px;'>
<div style="font-size:14px;">
{{"{{"}}:#data['Name']}}
<p style="font-size:8px;">{{"{{"}}:#data['Designation']{{}}}}</p>
</div>
</td>
<td class="border" style='height:99px;'>
<div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div style="position:relative;top:-25px;display:inline-block;">
<img src="../Images/treegrid/{{:#data['FullName']}}.png" />
</div>
<div style="display:inline-block;">
<div style="padding:5px;">{{"{{"}}:#data['Address']{{}}}}</div>
<div style="padding:5px;">{{"{{"}}:#data['Country']{{}}}}</div>
<div style="padding:5px;font-
size:12px;">{{"{{"}}:#data['Contact']{{}}}}</div>
</div>
</div>
</td>
<td class="border" style='height:99px;'>
<div>{{"{{"}}:#data['DOB']{{}}}}</div>
</td>
</tr>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.AltRowTemplateId("altRowTemplateScript")
.RowTemplateId("rowTemplateScript")
.RowHeight(99)
.Columns(co =>
{
co.Field("EmployeeID").HeaderText("Employee ID").Add();
co.Field("Name").HeaderText("Employee Name").Add();
co.Field("Address").HeaderText("Employee Details").Add();
co.Field("DOB").HeaderText("DOB").EditType(TreeGridEditingType.Datepicker
).Add();
}
)
.SizeSettings(ss => ss.Width("100%").Height("350px"))
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

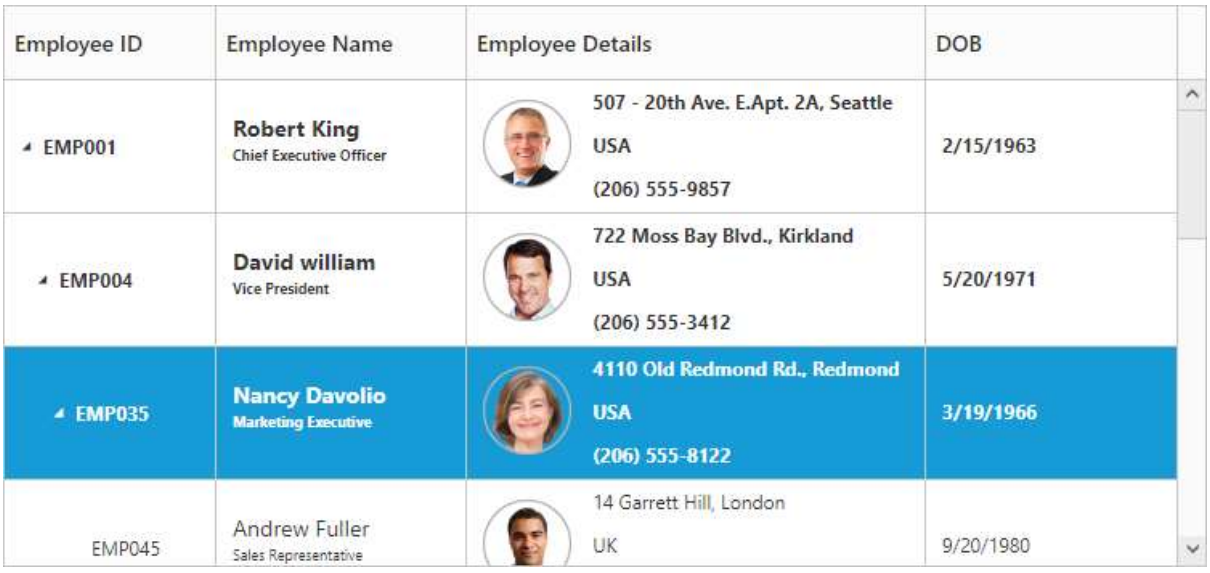
C#

```
public ActionResult TreeGridRowTemplate()
{
var data = this.getRowData();
ViewBag.datasource = data;
return View();
}
private List<RowData> getRowData()
{
List<RowData> DataCollection = new List<RowData>();
RowData Record1 = new RowData()
{
Name = "Robert King",
FullName = "Robert King",
Designation = "Chief Executive Officer",
EmployeeID = "EMP001",
Address = "507 - 20th Ave. E.Apt. 2A, Seattle",
Contact = "(206) 555-9857",
Country = "USA",
DOB = "2/15/1963",
DOJ = "5/1/1983",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Children = new List<RowData>()
};
RowData parent = new RowData()
{
Name = "David William",
FullName = "David William",
Designation = "Vice President",
EmployeeID = "EMP004",
Address = "722 Moss Bay Blvd., Kirkland",
Country = "USA",
Contact = "(206) 555-3412",
DOB = "5/20/1971",
DOJ = "1/5/1991",
Children = new List<RowData>()
};
//..
Record1.Children.Add(parent);
//..
return DataCollection;
}
public class RowData
{
public string Name { get; set; }
public string FullName { get; set; }
public string Designation { get; set; }
public string EmployeeID { get; set; }
public string Address { get; set; }
public string Contact { get; set; }
public string DOB { get; set; }
public string DOJ { get; set; }
public string Country { get; set; }
public List<RowData> Children { get; set; }
}
```

The output of TreeGrid with Row Template is as follows.



Click here to view online sample for TreeGrid Row Template

Copyright © 2001 - 2020 Syncfusion Inc.

Note: In TreeGrid, the given row template is parsed for default row functionality like row selection, alt row and other default row customization. Using ParseRowTemplate property we can disable the row template parsing. If we disable that property, TreeGrid is rendered with given row template.

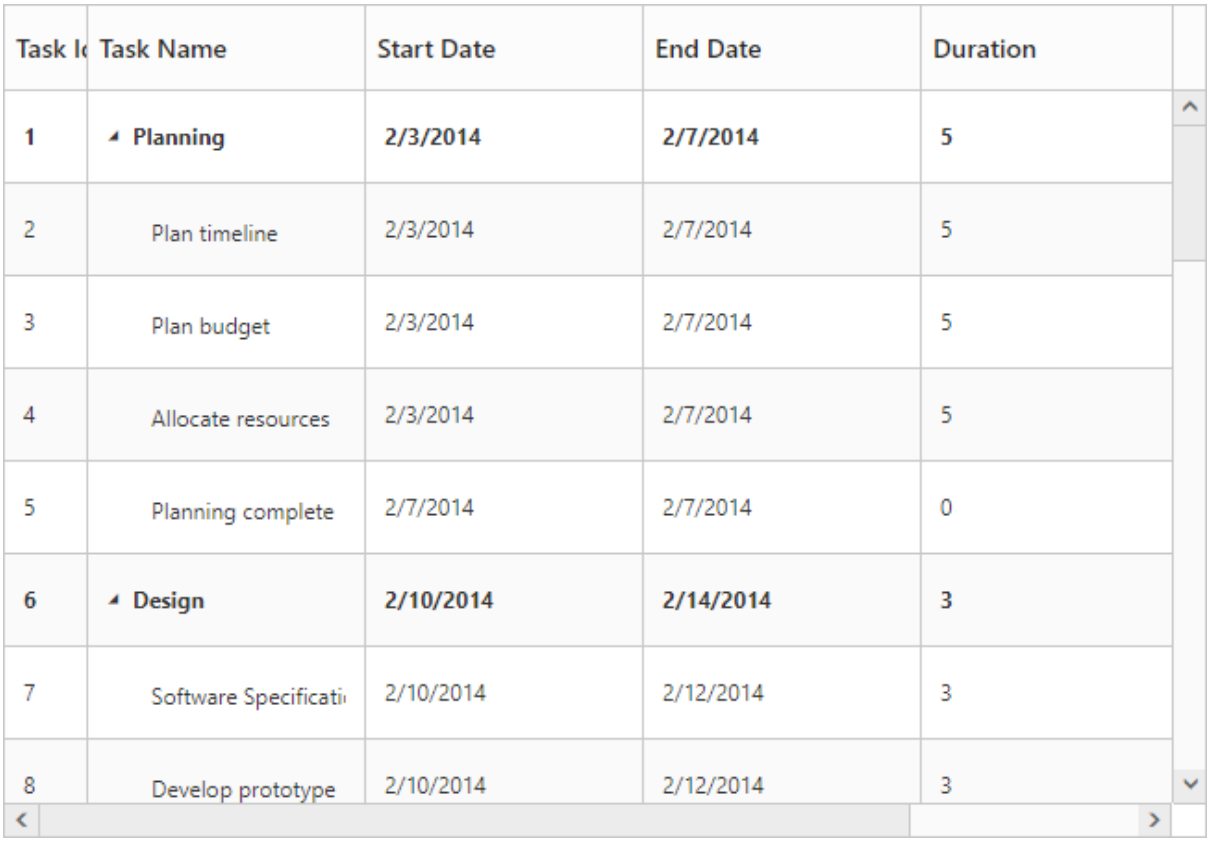
Row Height

The RowHeight property is used to change the height of row in TreeGrid, default value of this property is 30.

The following code example explains how to change the row height in TreeGrid

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.RowHeight(50)
//.
)
@(Html.EJ().ScriptManager())
```



Task Id	Task Name	Start Date	End Date	Duration
1	**Planning**	**2/3/2014**	**2/7/2014**	**5**
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	**Design**	**2/10/2014**	**2/14/2014**	**3**
7	Software Specificati	2/10/2014	2/12/2014	3
8	Develop prototype	2/10/2014	2/12/2014	3

The above screenshot shows TreeGrid render with row height of 50.

Alternate row styling

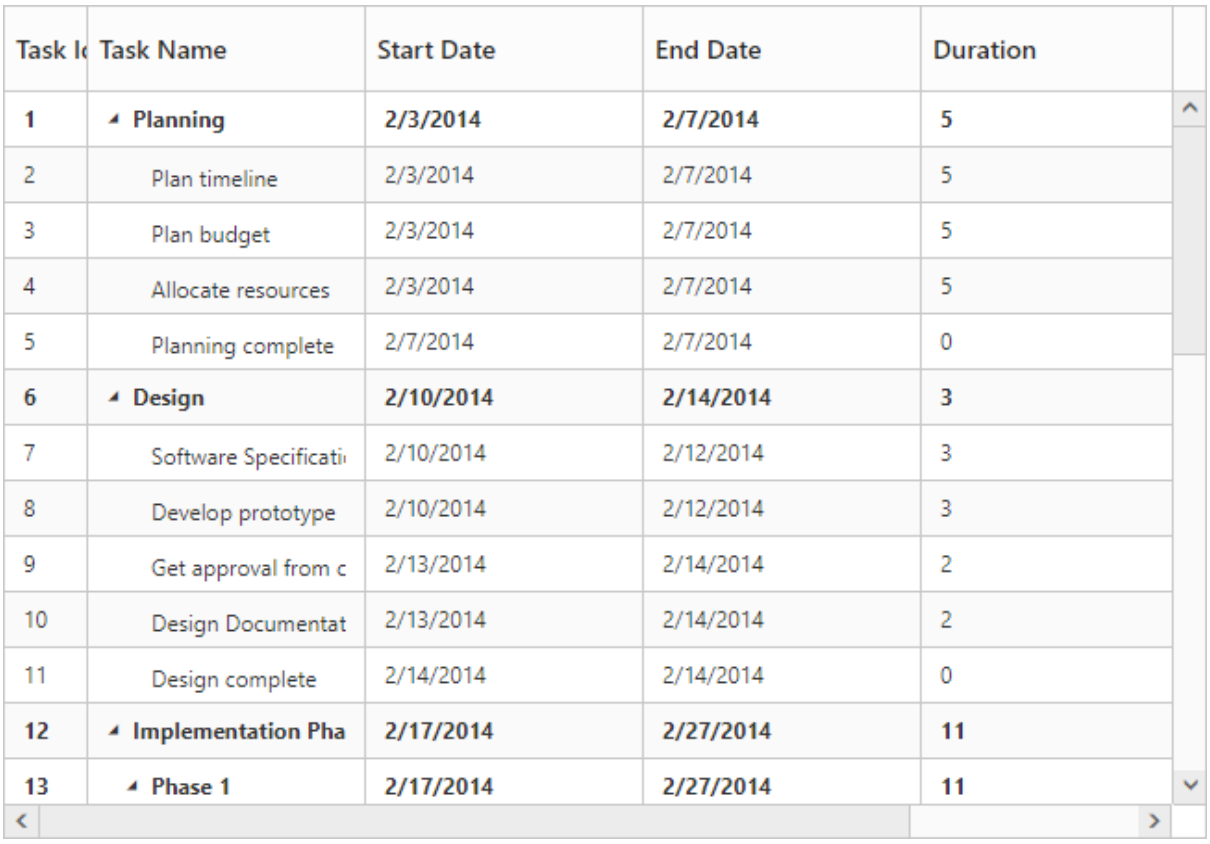
Alternate row style is used to enable the different background color for every alternate row. The EnableAltRow property is used to enable the alternate row style in TreeGrid, default value of this property is true.

The following code explains about enabling the alternate row style in TreeGrid

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EnableAltRow(true)
//.
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows TreeGrid with alternate row style.

Row Drag and Drop

It is possible to dynamically re-arrange the rows in the TreeGrid control by using the AllowDragAndDrop property. Using this property row drag option can be enabled or disabled. Rows can be inserted above, below or as a sibling or as a child to the existing row with the help of this feature. A default tooltip is rendered while dragging the TreeGrid row and this tooltip can be customized by the DragTooltip property. This property has inner properties such as ShowTooltip, TooltipItems and TooltipTemplate.

The ShowTooltip property is used to enable or disable the tooltip and the default value of this property is false.

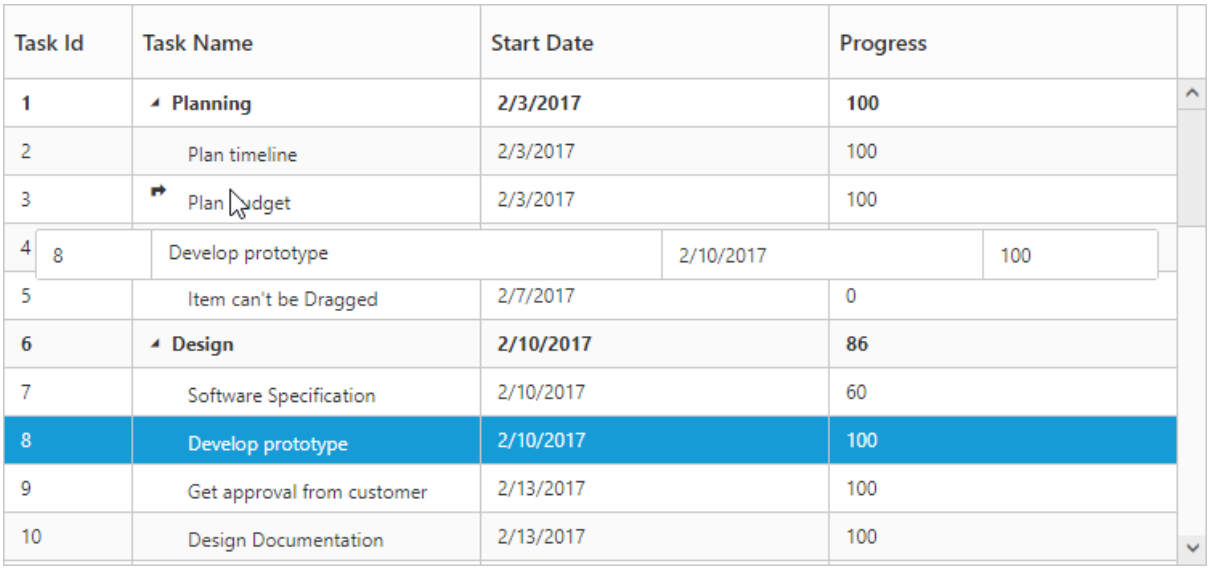
The following code explains about enabling the row drag and drop with the default tooltip in the TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(co =>
{
```

```
co.Field("TaskId").headerText("Task Id"). Add();
co.Field("TaskName").headerText("Task Name").Add();
})
.AllowDragAndDrop(true)
.DragTooltip(tooltip =>
{
tooltip.ShowTooltip(true);
}))
@(Html.EJ().ScriptManager())
```

The following screenshot depicts a row drag and drop in the TreeGrid.



Customizing Drag tooltip

The default drag tooltip in TreeGrid can be customized by using TooltipTemplate property. We can provide JsRender template or HTML formatted string as the value for this property.

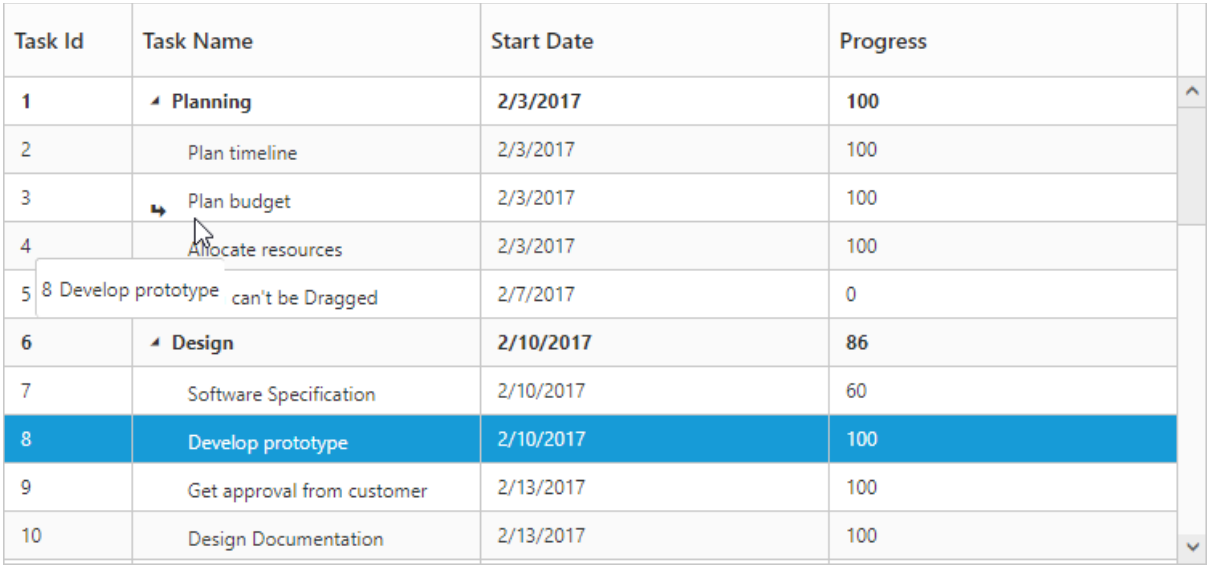
The following code shows how to render row drag tooltip with JsRender template.

ASPX-CS

```
<script id="customTooltip" type="text/x-jsrender">
<tr>
<td style='height:30px'>
<div>{{"{{"}}:#data['TaskId']{{}}}}</div>
</td>
<td style='height:30px'>
<div>{{"{{"}}:#data['TaskName']{{}}}}</div>
</td>
</tr>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.DragTooltip(tt =>
{
tt.ShowTooltip(true);
tt.TooltipTemplate("#customTooltip");
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
@(Html.EJ().ScriptManager())
```



Multiple row drag and drop

TreeGrid provides support for multiple row reordering with mouse drag and drop interaction. The selected rows can be dropped above and below as siblings or as child records similar to single row reordering.

In TreeGrid we can enable the multiple row drag and drop by setting SelectionSettings.SelectionType as Multiple or Checkbox and also we should enable the AllowDragAndDrop.

Please find the code example below to enable multiple drag and drop in TreeGrid.

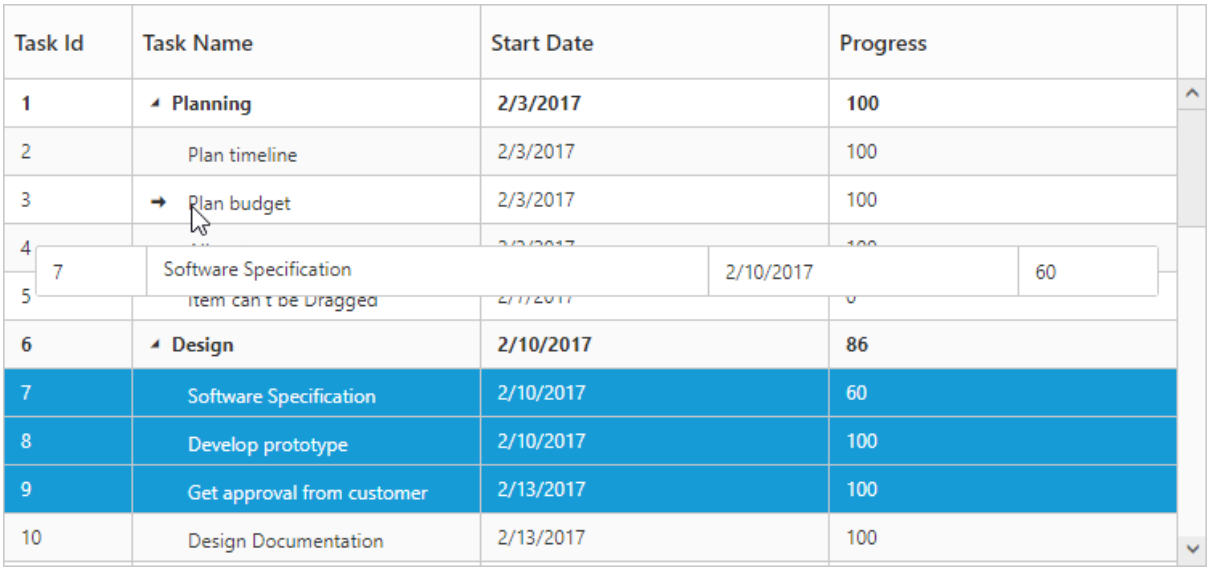
ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowDragAndDrop(true)
.SelectionSettings(ss=>
ss.SelectionType(TreeGridSelectionType.Multiple))
)
```

We can also customize row drag and drop actions by using below properties

- canDrag – It is used to enable/disable the row drag action for draggedRecords collection in RowDragStart client side event.
- canDrop – It is used to enable/disable the row drop action for draggedRecords collection in RowDropActionBegin client side event.

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the demo sample for multiple drag and drop action.

Details row

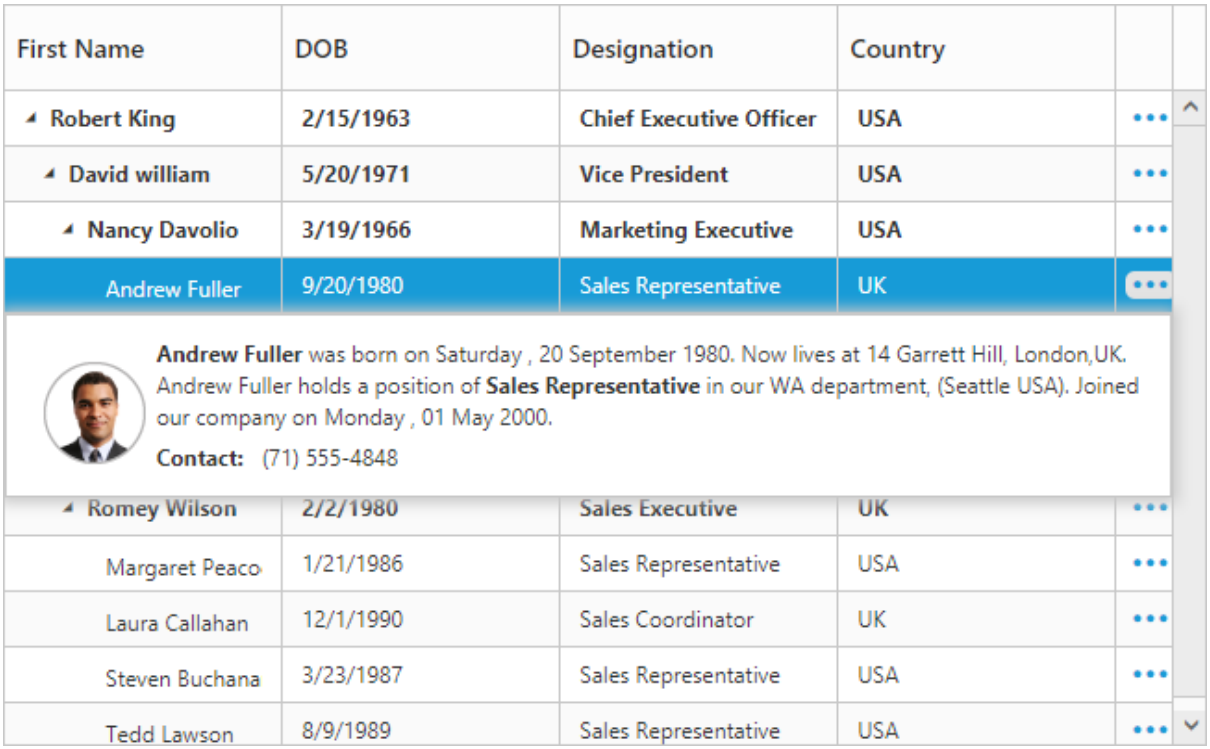
Details row is used to provide a additional information about each row of TreeGrid. You can specify the detail row JsRender template id or HTML element as string to `DetailsTemplate` property. However you need to enable the details template by setting `ShowDetailsRow` property as `true`.

The following code example shows how to enable details tow in TreeGrid.

ASPX-CS

```
<script id="descriptionTemplate" type="text/x-jsrender">
<div style="position: relative; display: inline-block; float: left; font-weight: bold; width: 10%">
<img src="../images/treegrid/{{"{{"}}:Name{{}}}}.png" style="margin-left: 10px;margin-top:15px;" />
</div>
<div style="padding-left: 10px; display: inline-block; width: 88%; text-wrap: normal;font-size:12px;font-family:'Segoe UI';margin-top:2px;">
<div class="e-description" style="word-wrap: break-word;">
<b>{{"{{"}}:Name{{}}}}</b> was born on
{{"{{"}}:~_treeGridDateFormatter("DOB"){{}}}}. Now lives at
{{"{{"}}:Address{{}}}},{{"{{"}}:Country{{}}}}. {{"{{"}}:Name{{}}}} holds
a position of <b>{{"{{"}}:Designation{{}}}}</b> in our WA department,
(Seattle USA). Joined our company on
{{"{{"}}:~_treeGridDateFormatter("DOJ"){{}}}}.
</div>
<div class="e-description" style="word-wrap: break-word;margin-top:5px;">
<b style="margin-right:10px;">Contact:</b>{{"{{"}}:Contact{{}}}}
</div>
</div>
</script>
@(Html.EJ().TreeGrid("TreeGridRowDetails")
.DetailsTemplate("descriptionTemplate")
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows details row in TreeGrid.

The visibility of the details view of a record can also be toggled with any custom actions by using the method ShowHideDetailsRow.

Click here to view the demo sample for details row template.

Disable details row info column

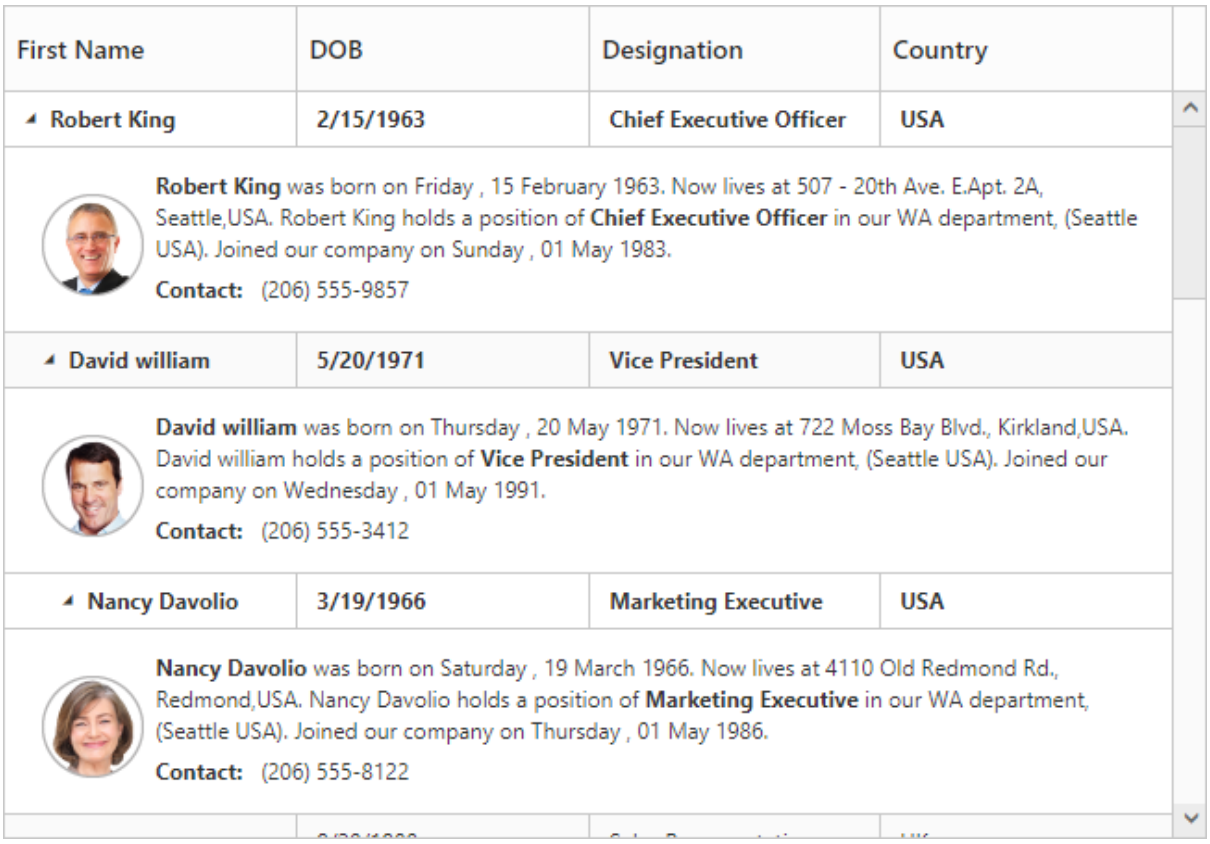
On enabling details template, details row info column will be added in TreeGrid. It is used for show or hide the detail row of respective row.

You can disable that column while enabling details template using ShowDetailsRowInfoColumn property. If you disable details row info column, then the details row will render next to the respective row.

The following code example shows how to hide detail info column in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridRowDetails")
.DetailsTemplate("descriptionTemplate")
.ShowDetailsRowInfoColumn(false)
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows details row rendered next to the respective row.

Defining row height for detail template

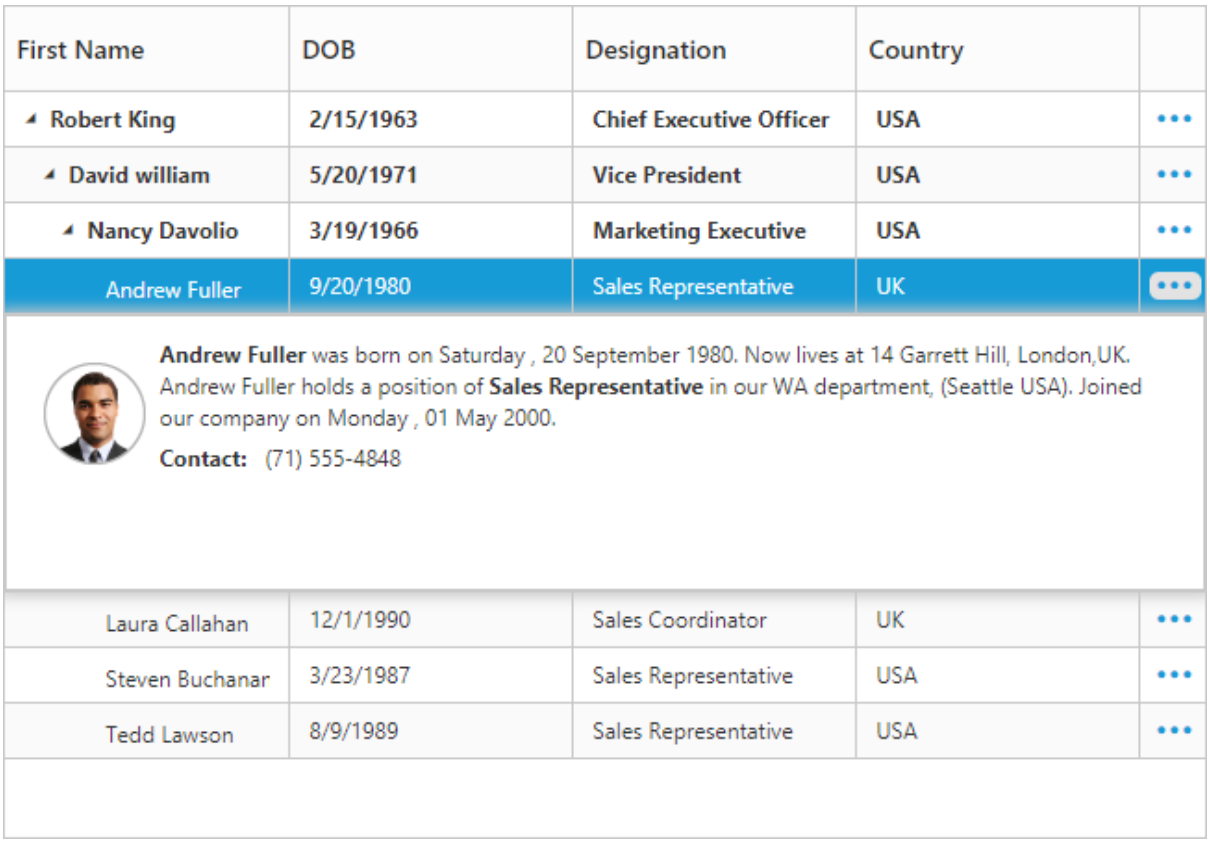
In TreeGrid, it is provide a support to change the detail template height using DetailsRowHeight property.

The following code example shows how to set details row height in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridRowDetails")
.DetailsTemplate("descriptionTemplate")
.DetailsRowHeight(150)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows details row rendered with height of 150px.

Customize detail row

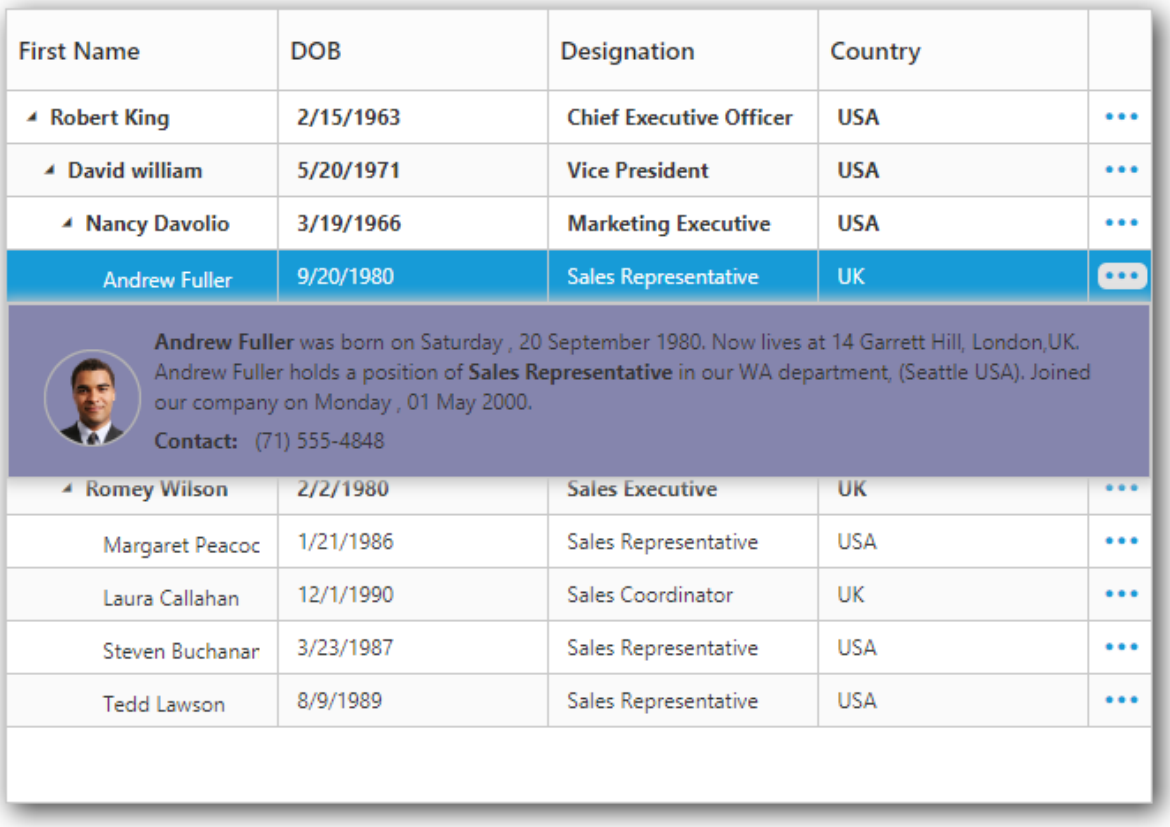
In TreeGrid, while rendering the details row the DetailsDataBound event will be triggered. Using this event we can customize the detail template for specific row.

The below code example shows how to customize details row for specific row.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.DetailsTemplate("descriptionTemplate")
.ClientSideEvents(cs =>
{
cs.DetailsDataBound("detailsDataBound");
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function detailsDataBound(args) {
if (args.data.Name == "Andrew Fuller")
$(args.detailsElement).css("background-color", "rgba(133, 133, 173,1)");
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows details row customization for specific row.

While opening and closing the details row, the DetailsShown and DetailsHidden events are triggered. Using this event we can prevent the details row show and hide action for specific row.

The below code example shows how to prevent details row show action for specific row.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.DetailsTemplate("descriptionTemplate")
.ClientSideEvents(cs =>
{
cs.DetailsShown("detailsShown");
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function detailsShown(args) {
if (args.data.Name == "Andrew Fuller")
args.cancel = true;
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Expand/Collapse Row

In TreeGrid, parent rows are expanded/collapsed by using expand/collapse icons, expand all/collapse all toolbar items and by using the expandCollapseRow method. By default all records in TreeGrid will be rendered in expanded state.

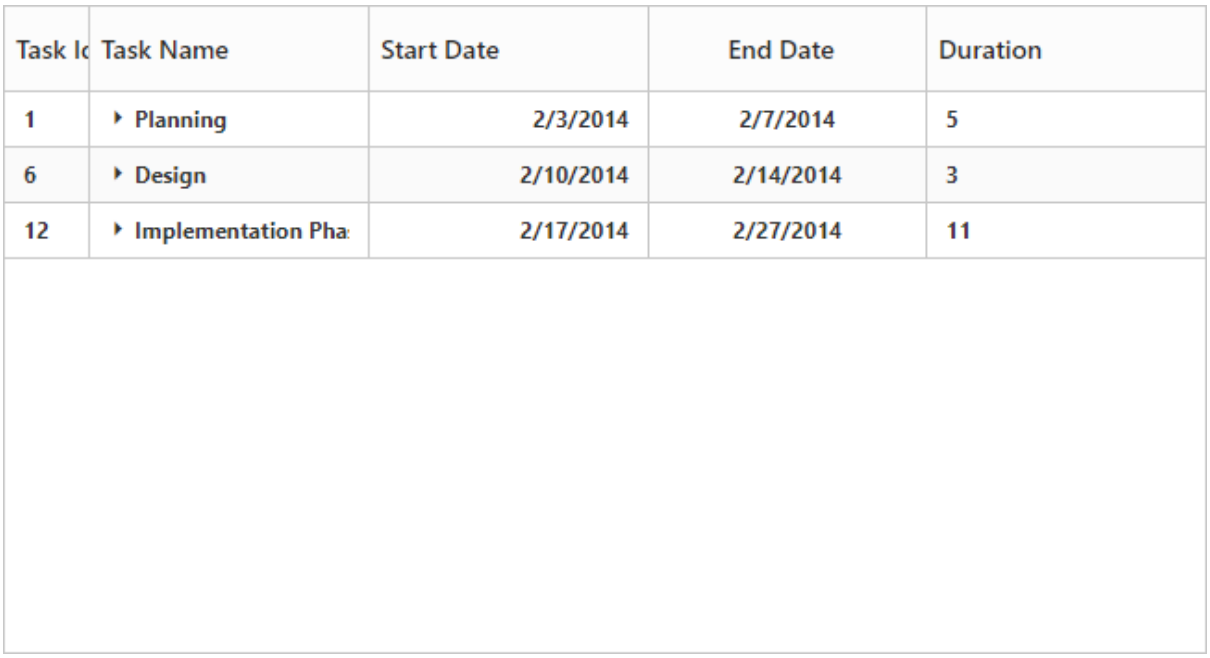
Collapse parent records at initial load

It is possible to display all the records in collapsed state by setting EnableCollapseAll property as true.

The following code example shows how to use this property.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.EnableCollapseAll(true)
)
@(Html.EJ().ScriptManager())
```



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	2/3/2014	2/7/2014	5
6	Design	2/10/2014	2/14/2014	3
12	Implementation Pha	2/17/2014	2/27/2014	11

The above screenshot shows TreeGrid render with collapsed state.

Define expand/collapse state of every record

In TreeGrid, it is possible to render records either in collapsed state or in expanded state, and this can done by mapping the expand state to the records from the data source by using ExpandStateMapping property.

The following code example shows how to use this property.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.ExpandStateMapping("expandState")
.Datasource(ViewBag.datasource)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

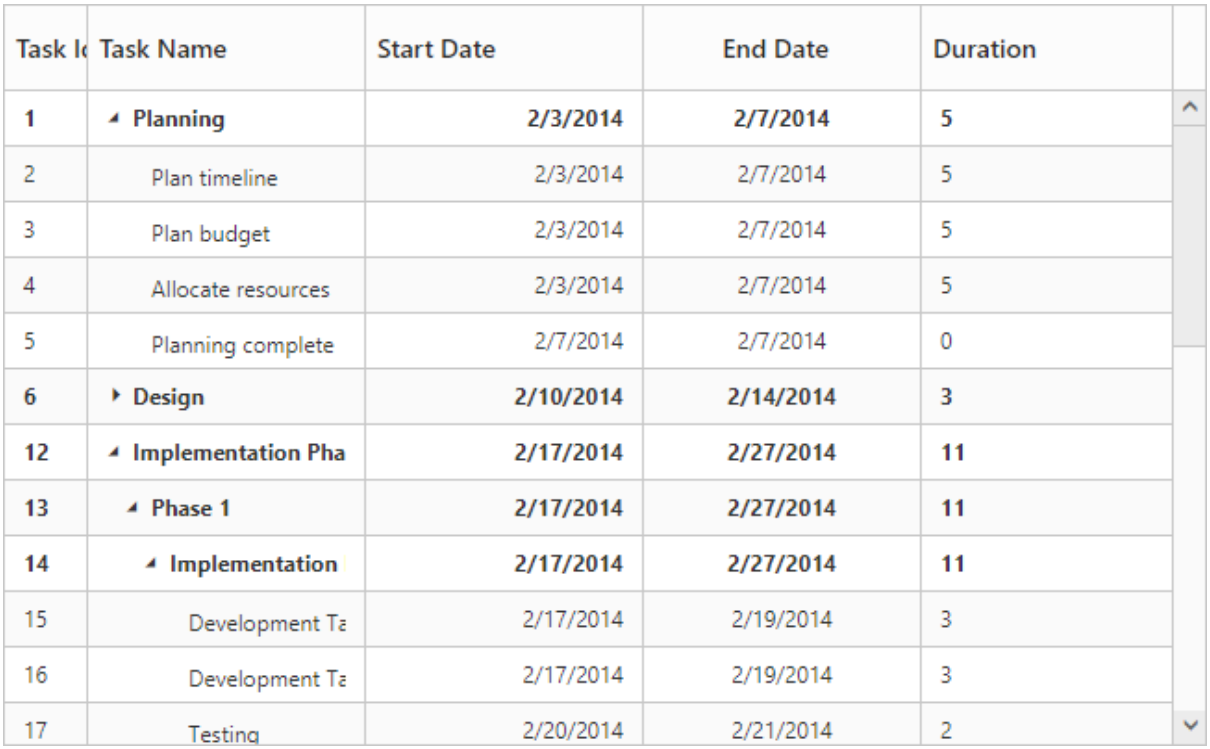
```
@(Html.EJ().ScriptManager())
```

C#

```
public class TreeGridController : Controller
{
//
// GET: /Default/
public ActionResult Default()
{
var DataSource = TreeGridDefaultData.GetData();
ViewBag.datasource = DataSource;
return View();
}
public class TreeGridDefaultData
{
public static List<DefaultData> GetData()
{
List<DefaultData> list = new List<DefaultData>();
list.Add(new DefaultData()
{
Id = 6,
Name = "Design",
ExpandState = false,
Children = (new List<DefaultData>()
{
new DefaultData()
{
Id = 7,
Name = "Software Specification",
ExpandState = false,
},
//..
})
});
return list;
}
}
}
```

The below screenshot shows the output of above code example..

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	2/3/2014	2/7/2014	5
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	Design	2/10/2014	2/14/2014	3
12	Implementation Pha	2/17/2014	2/27/2014	11
13	Phase 1	2/17/2014	2/27/2014	11
14	Implementation	2/17/2014	2/27/2014	11
15	Development Ta	2/17/2014	2/19/2014	3
16	Development Ta	2/17/2014	2/19/2014	3
17	Testing	2/20/2014	2/21/2014	2

Expand/Collapse all the rows dynamically

All the rows in TreeGrid will be expanded/collapsed by clicking ExpandAll and CollapseAll toolbar items or by using expandAll and collapseAll methods. We can invoke this methods

dynamically on any action like external button click.

The below code example shows how to use this methods.

ASPX-CS

```
<button id="expandAll">Expand All</button>
<button id="collapseAll">Collapse All</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.ExpandAll,
TreeGridToolBarItems.CollapseAll,
});
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#expandAll").click(function () {
var treegridObj = $("#TreeGridContainer").ejTreeGrid("instance");
treegridObj.expandAll();
});
$("#collapseAll").click(function () {
var treegridObj = $("#TreeGridContainer").ejTreeGrid("instance");
treegridObj.collapseAll();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
});
</script>
```

Dynamically expand/Collapse the specific level row

The TreeGrid control provides the support to dynamically expand/collapse the specific level of rows by using expandAtLevel and collapseAtLevel methods. This methods are used to expand/collapse the rows which are in specific level.

The below code example shows how to use this methods.

ASPX-CS

```
<button id="expandAtLevel">Expand At Level</button>
<button id="collapseAtLevel">Collapse At Level</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#expandAtLevel").click(function () {
var treegridObj = $("#TreeGridContainer").ejTreeGrid("instance");
treegridObj.expandAtLevel(1);
});
$("#collapseAtLevel").click(function () {
var treegridObj = $("#TreeGridContainer").ejTreeGrid("instance");
treegridObj.collapseAtLevel(1);
});
</script>
```

Customize expand/collapse action

In TreeGrid, while expanding the parent row Expanding and Expanded event will be triggered with current expanding row detail. Similarly Collapsing and Collapsed event will be triggered while collapsing the parent row. Using this event and its arguments we can customize the expand/collapse action.

The following code example shows how to prevent the particular row from expand/collapse action using Expanding and Collapsing event.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.DetailsTemplate("descriptionTemplate")
.ClientSideEvents(cs =>
{
cs.Collapsing("collapsing");
cs.Expanding("expanding");
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function expanding(args) {
if(args.data.progress < 50) // we can't collapse the record which has
progress is less than 50
args.cancel = true;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function collapsing(args) {
if(args.data.duration > 5) // we can't expand the record which has
duration is greater than 5
args.cancel = true;
}
</script>
```

Cell

Tooltip

In TreeGrid tooltip can be enabled using the ShowGridCellTooltip property. Using this property tooltip can be enabled for cells both header and content.

Please find the example describes the above behavior.

ASPX-CS

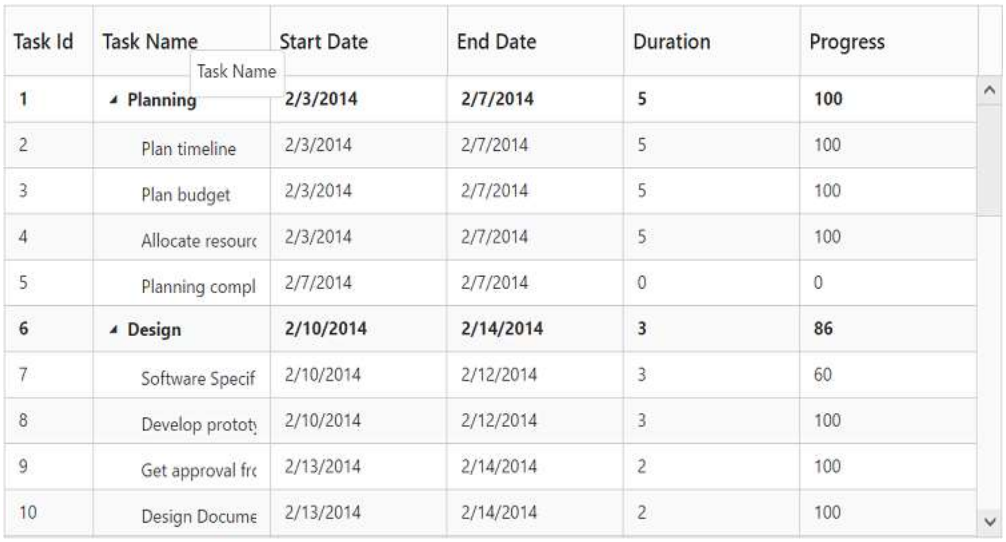
```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowGridCellTooltip(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output shows the result of above code example.

Task Id	Task Name	Start Date	End Date	Duration	Progress
1	Planning	2/3/2014	2/7/2014	5	100
2	Plan timeline	2/3/2014	2/7/2014	5	100
3	Plan budget	2/3/2014	2/7/2014	5	100
4	Allocate resou...	2/3/2014	2/7/2014	5	100
5	Planning com...	2/7/2014	2/7/2014	0	0
6	Design	2/10/2014	/14/2014	3	86
7	Software Speci...	2/10/2014	2/12/2014	3	60
8	Develop proto...	2/10/2014	2/12/2014	3	100
9	Get approval f...	2/13/2014	2/14/2014	2	100
10	Design Docum...	2/13/2014	2/14/2014	2	100
11	Design compl...	2/14/2014	2/14/2014	0	0
12	Implementatio...	2/17/2014	2/27/2014	11	

Cell Tooltip

Copyright © 2001 - 2020 Syncfusion Inc.



Header Tooltip

Tooltip Template

It is possible to display a custom tooltip across all the TreeGrid cells using the property CellTooltipTemplate with the property ShowGridCellTooltip enabled. We need to set the template of the custom tooltip to this property.

Please find code example that describes the cell tooltip template support below.

ASPX-CS

```
<script type="text/x-jsrender" id="cellTooltipTemplate">
<table>
<tr>
<td style='padding:5px;font-weight: bold;'>
Task ID
</td>
<td style='padding:5px;'>
: {{"{{"}}:#data['record']['taskID']{{}}}}
</td>
</tr>
<tr>
<td style='padding:5px;font-weight: bold;'>
Task Name
</td>
<td style='padding:5px;'>
: {{"{{"}}:#data['record']['taskName']{{}}}}
</td>
</tr>
<tr>
<td style='padding:5px;font-weight: bold;'>
Start Date
</td>
<td style='padding:5px;'>
: {{"{{"}}:#data['record']['startDate']{{}}}}
</td>
</tr>
<tr>
<td style='padding:5px;font-weight: bold;'>
```

```
End Date
</td>
<td style='padding:5px;'>
: {{"{{"}}:#data['record']['endDate']{{}}}}
</td>
</tr>
<tr>
<td style='padding:5px;font-weight: bold;'>
Duration
</td>
<td style='padding:5px;'>
: {{"{{"}}:#data['record']['duration']{{}}}}
</td>
</tr>
<tr>
<td style='padding:5px;font-weight: bold;'>
Progress
</td>
<td style='padding:5px;'>
: {{"{{"}}:#data['record']['progress']{{}}}}
</td>
</tr>
</table>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.ShowGridCellTooltip(true)
.CellTooltipTemplate("cellTooltipTemplate")
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output shows the result of above code example.

Task Id	Task Name	Start Date	End Date	Duration	Progress
1	**Planning**	**2/3/2014**	**2/7/2014**	**5**	**100**
2	Plan timeline	2/3/2014	2/7/2014	5	100
3	Plan budget	2/3/2014	2/7/2014	5	100
4	Allocate resou...	2/3/2014		5	100
5	Planning com...	2/7/2014		0	0
6	**Design**	**2/10/2014**		**3**	**86**
7	Software Speci...	2/10/2014		3	60
8	Develop proto...	2/10/2014		3	100
9	Get approval f...	2/13/2014		2	100
10	Design Docum...	2/13/2014	2/14/2014	2	100
11	Design compl...	2/14/2014	2/14/2014	0	0
12	**Implementatio...**	**2/17/2014**	**2/27/2014**	**11**	

Task ID : 3
Task Name : Plan budget
Start Date : 02/03/2014
End Date : 02/03/2014
Duration : 5
Progress : 100

Copyright © 2001 - 2020 Syncfusion Inc.

Column tooltip

By using the property the Columns.Tooltip it is possible to display a custom tooltip for a specific column. The ID of the script template must be set to the Columns.Tooltip property.

Please refer to the following code example for setting a custom tooltip for a specific column.

ASPX-CS

```
<script type="text/x-jsrender" id="template">
<div style='padding:10px;color:red;font-weight: bold;'>
{{"{{"}}:#data['record']['taskName']{{}}}}
</div>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
.Columns(co =>
{
//..
co.Field("taskName").headerText("Task Name").Tooltip("template").Add();
//..
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output shows the output of above code snippets.

Task Id	Task Name	Start Date	End Date	Duration	Progress
1	Planning	2/3/2014	2/7/2014	5	100
2	Plan timeline	2/3/2014	2/7/2014	5	100
3	Plan budget	2/3/2014	2/7/2014	5	100
4	Allocate	014	2/7/2014	5	100
5	Planning com...	2/7/2014	2/7/2014	0	0
6	Design	2/10/2014	2/14/2014	3	86
7	Software Speci...	2/10/2014	2/12/2014	3	60
8	Develop proto...	2/10/2014	2/12/2014	3	100
9	Get approval f...	2/13/2014	2/14/2014	2	100
10	Design Docum...	2/13/2014	2/14/2014	2	100
11	Design compl...	2/14/2014	2/14/2014	0	0
12	Implementatio...	2/17/2014	2/27/2014	11	

Plan budget

Note: Template element should be enclosed with <script> tag with type as “text/x-jsrender”.

Header tooltip

By using the property the Columns.HeaderTooltip it is possible to display a custom tooltip for a specific column header. The ID of the script template must be set to the Columns.HeaderTooltip property.

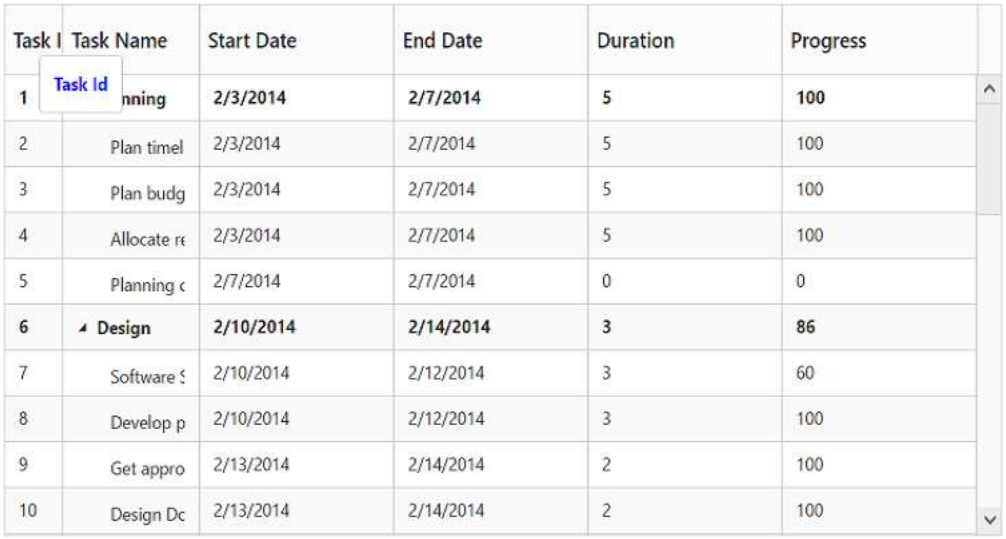
Copyright © 2001 - 2020 Syncfusion Inc.

Please refer to the following code example for setting a custom tooltip for a specific column header.

ASPX-CS

```
<script type="text/x-jsrender" id="template">
<div style='padding:10px;color:blue;font-weight: bold;'>
{{"{{"}}:#data['column']['headerText']{{}}}}
</div>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
.Columns(co =>
{
//..
co.Field("taskID").headerText("Task Id").HeaderTooltip("template").Add();
//..
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output shows the result of above code example.



Clip Mode

Clip mode enables the TreeGrid to clip cell content and header content when the content exceeds the boundary of the cell width.

We can specify the type of clip mode using the Columns.ClipMode property, clip mode will be enabled for both TreeGrid content and header of that specific column.

The below are the available clipping modes in TreeGrid.

1. Clip
2. Ellipsis

Copyright © 2001 - 2020 Syncfusion Inc.

Clip

When ClipMode of Columns property set as the TreeGridColumnClipMode.Clip, then it truncates the overflown text in the cell.

Note: 1. By default the ClipMode will be set as theTreeGridColumnClipMode.Clip.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
.Columns(co =>
{
//..
co.Field("taskID").headerText("Task
Id").ClipMode(TreeGridColumnClipMode.Clip).Add();
//..
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output shows the result of above code example.

Task Id	Task Name	Start Date	End Date	Duration	Progress
1	**Planning**	**2/3/2014**	**2/7/2014**	**5**	**100**
2	Plan timeline	2/3/2014	2/7/2014	5	100
3	Plan budget	2/3/2014	2/7/2014	5	100
4	Allocate resource	2/3/2014	2/7/2014	5	100
5	Planning comple	2/7/2014	2/7/2014	0	0
6	**Design**	**2/10/2014**	**2/14/2014**	**3**	**86**
7	Software Specific	2/10/2014	2/12/2014	3	60
8	Develop prototy	2/10/2014	2/12/2014	3	100
9	Get approval fro	2/13/2014	2/14/2014	2	100
10	Design Documer	2/13/2014	2/14/2014	2	100
11	Design complete	2/14/2014	2/14/2014	0	0
12	**Implementation F**	**2/17/2014**	**2/27/2014**	**11**	

Ellipsis

When the Columns.ClipMode property is set as TreeGridColumnClipMode.Ellipsis then it shows ellipsis for the overflown cell.

The following code example describes the above behavior.

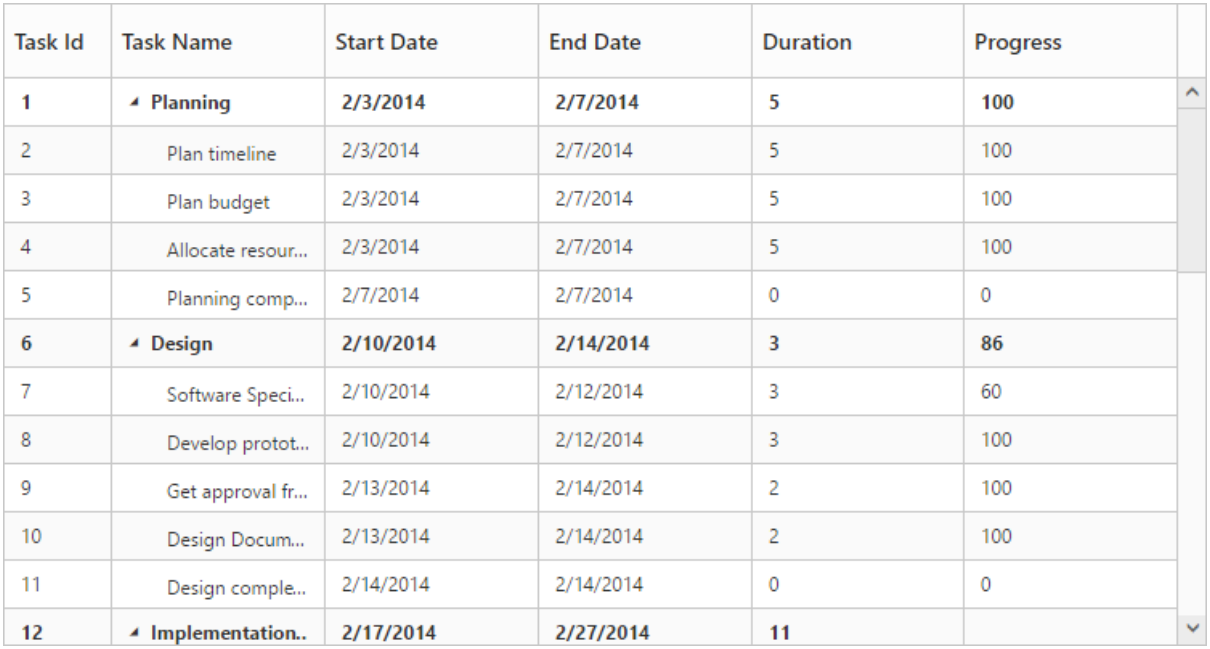
ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ChildMapping("Children")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Columns(co =>
{
//..
co.Field("taskID").headerText("Task
Id").ClipMode(TreeGridColumnClipMode.Ellipsis).Add();
//..
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output is shows the result of the above code example.



Task Id	Task Name	Start Date	End Date	Duration	Progress
1	**Planning**	**2/3/2014**	**2/7/2014**	**5**	**100**
2	Plan timeline	2/3/2014	2/7/2014	5	100
3	Plan budget	2/3/2014	2/7/2014	5	100
4	Allocate resour...	2/3/2014	2/7/2014	5	100
5	Planning comp...	2/7/2014	2/7/2014	0	0
6	**Design**	**2/10/2014**	**2/14/2014**	**3**	**86**
7	Software Speci...	2/10/2014	2/12/2014	3	60
8	Develop protot...	2/10/2014	2/12/2014	3	100
9	Get approval fr...	2/13/2014	2/14/2014	2	100
10	Design Docum...	2/13/2014	2/14/2014	2	100
11	Design comple...	2/14/2014	2/14/2014	0	0
12	**Implementation..**	**2/17/2014**	**2/27/2014**	**11**	

Text Wrap

Text wrap enables the TreeGrid to wrap cell content or header content to next line when the content exceeds the boundary of the cell width.

Header Text Wrap

To enable header cell text wrap, set the HeaderTextOverflow property as TreeGridHeaderTextOverflow.Wrap.

Note: By default the HeaderTextOverflow will be set as TreeGridHeaderTextOverflow.None.

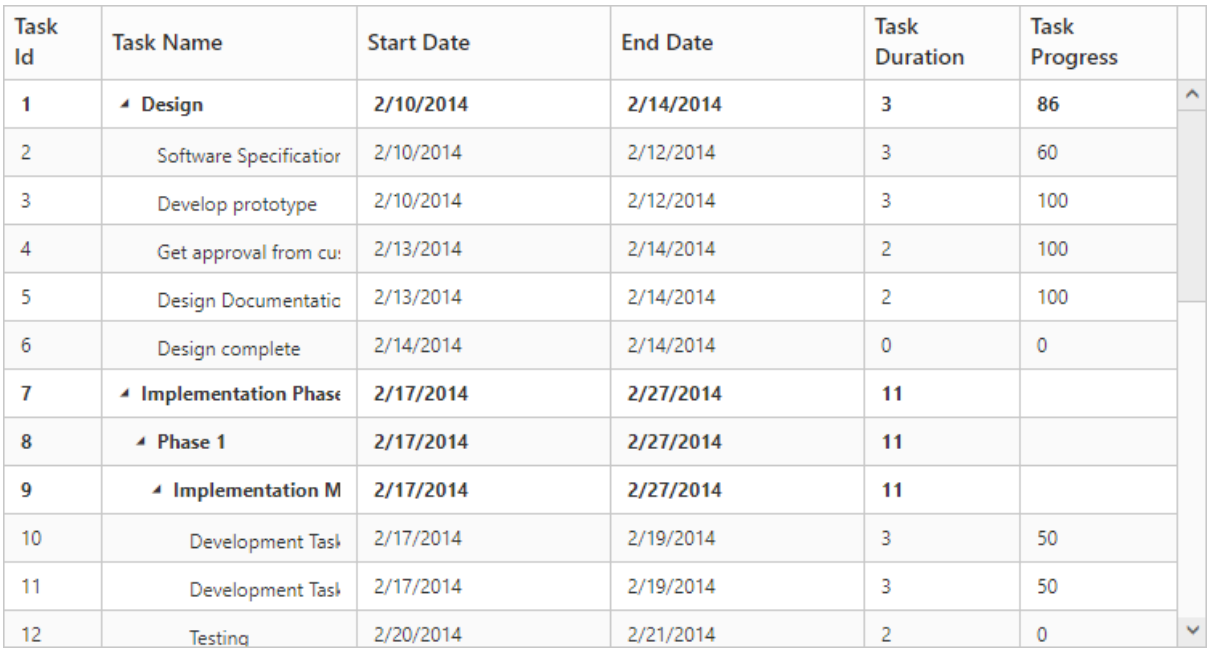
The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.HeaderTextOverflow(TreeGridHeaderTextOverflow.Wrap)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following output shows the result of above code example.



Task Id	Task Name	Start Date	End Date	Task Duration	Task Progress
1	**Design**	**2/10/2014**	**2/14/2014**	**3**	**86**
2	Software Specificatior	2/10/2014	2/12/2014	3	60
3	Develop prototype	2/10/2014	2/12/2014	3	100
4	Get approval from cu:	2/13/2014	2/14/2014	2	100
5	Design Documentatic	2/13/2014	2/14/2014	2	100
6	Design complete	2/14/2014	2/14/2014	0	0
7	**Implementation Phase**	**2/17/2014**	**2/27/2014**	**11**	
8	**Phase 1**	**2/17/2014**	**2/27/2014**	**11**	
9	**Implementation M**	**2/17/2014**	**2/27/2014**	**11**	
10	Development Task	2/17/2014	2/19/2014	3	50
11	Development Task	2/17/2014	2/19/2014	3	50
12	Testing	2/20/2014	2/21/2014	2	0

Content Text Wrap

To enable cell text wrap, set the `AllowTextWrap` property as `true`.

Note: 1.By default the `AllowTextWrap` will be set as `false`.

Note: 2.The text wrap feature provide only limited support in virtualization mode.

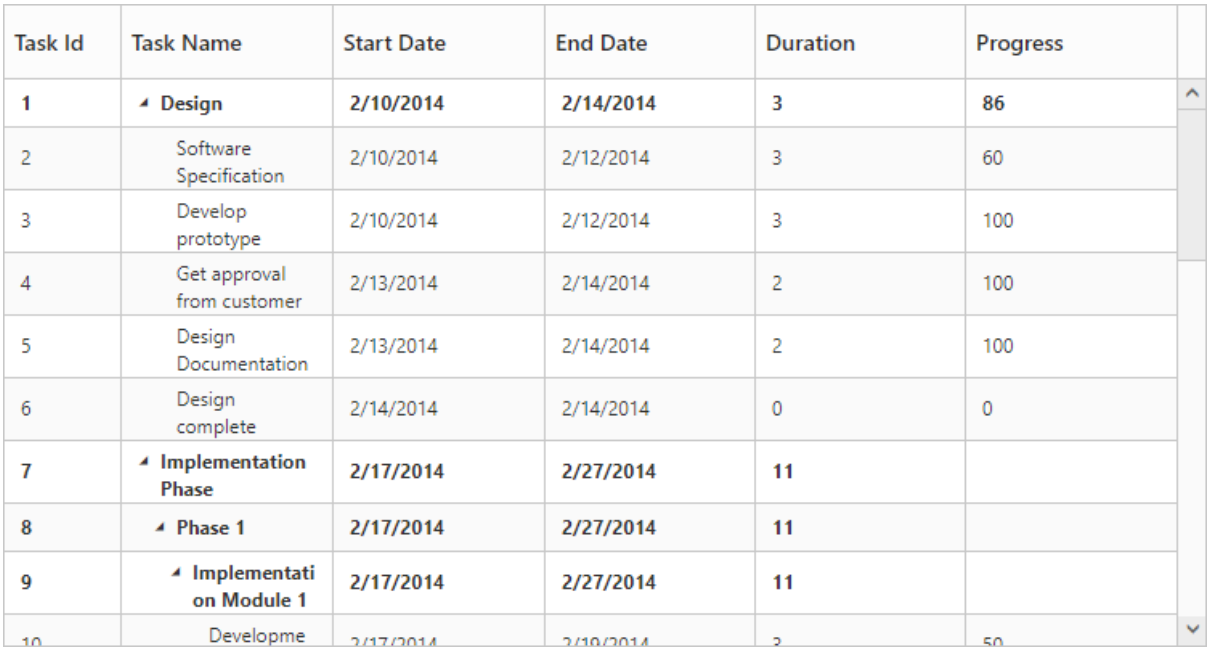
The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.AllowTextWrap(true)
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The following output shows the result of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration	Progress
1	Design	2/10/2014	2/14/2014	3	86
2	Software Specification	2/10/2014	2/12/2014	3	60
3	Develop prototype	2/10/2014	2/12/2014	3	100
4	Get approval from customer	2/13/2014	2/14/2014	2	100
5	Design Documentation	2/13/2014	2/14/2014	2	100
6	Design complete	2/14/2014	2/14/2014	0	0
7	Implementation Phase	2/17/2014	2/27/2014	11	
8	Phase 1	2/17/2014	2/27/2014	11	
9	Implementation Module 1	2/17/2014	2/27/2014	11	
10	Developme	2/17/2014	2/19/2014	3	50

Editing

TreeGrid provides support to add, edit and delete the records dynamically using EditSettings property.

Add new record

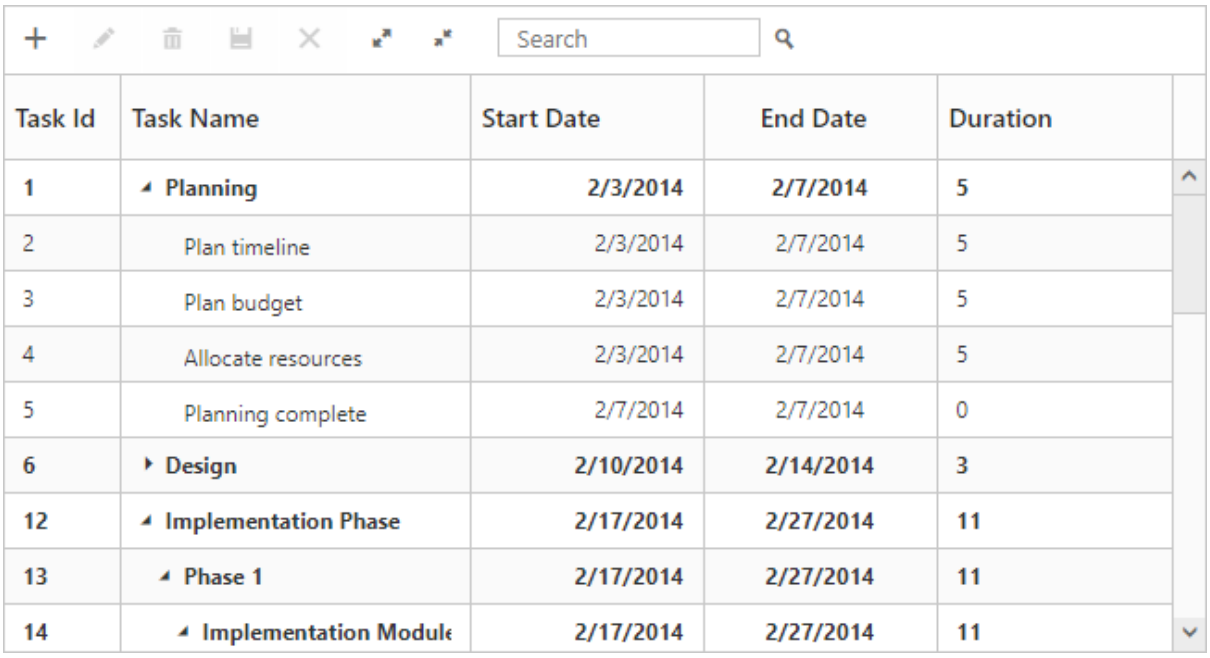
TreeGrid provides support for adding a new record by setting AllowAdding property as true. You can add new record by toolbar add item click or context menu.

The below code example shows how to enable add option in TreeGrid.

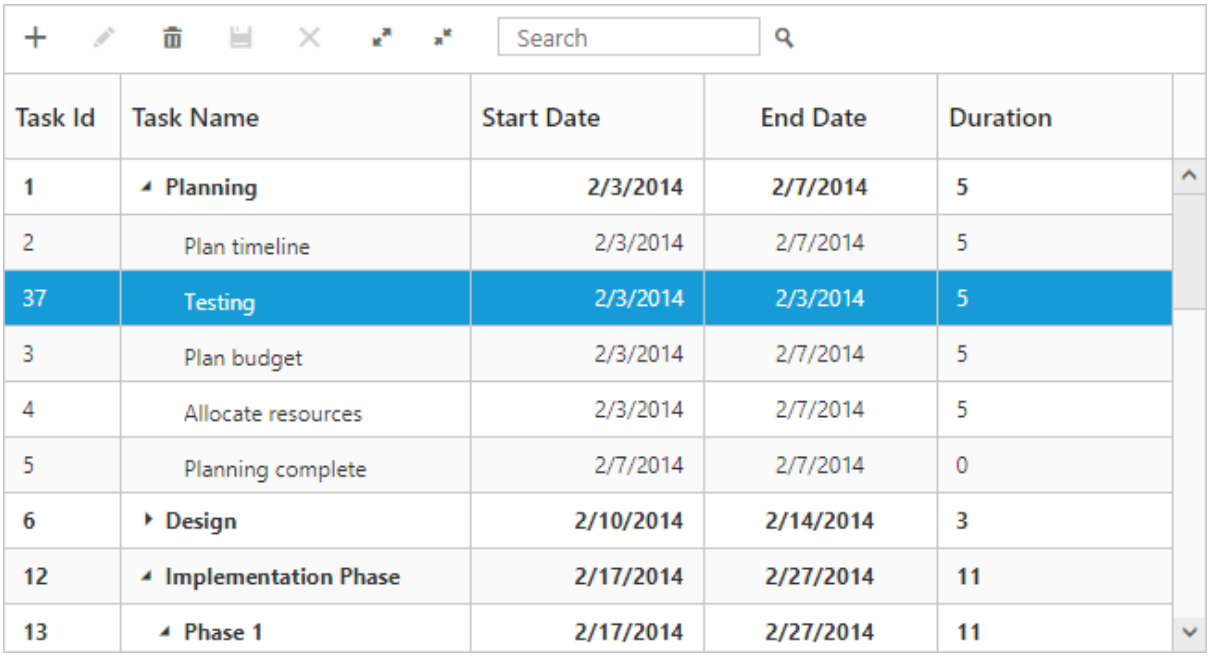
ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowAdding(true);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows before add a new record in TreeGrid.



The above screenshot shows after add a new record in TreeGrid.

Add row position

Using the EditSettings.RowPosition parameter, user can able to insert the record at any desired index at run-time. The user can insert a record dynamically in the following positions

- Top: Top to all the existing records
- Bottom: Bottom to all the existing records
- AboveSelectedRow: Above to the selected row
- BelowSelectedRow: Below to the selected row

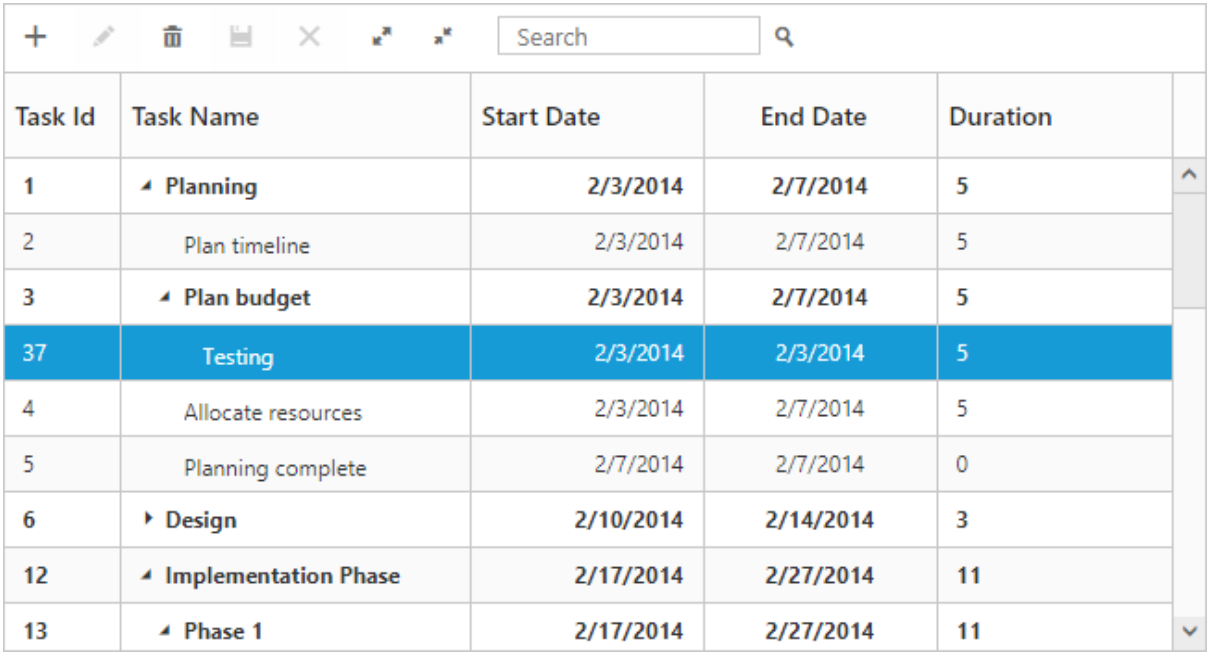
Copyright © 2001 - 2020 Syncfusion Inc.

- Child: As a child to the selected row

The below code example shows how to set row position for new record add in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowAdding(true);
edit.RowPosition(TreeGridRowPosition.Child);
})
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows new record added in row position of Child.

Adding records using method

Using addRow method records can be added dynamically to the TreeGrid. Before calling this method, you should enable the AllowAdding property.

The below code snippet explains dynamically inserting a record in TreeGrid. The record will be inserted as a child node to the current selected record.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowAdding(true);
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
```

```
$("#add").click(function () {
var treeGridObj = $("#TreeGridContainer").data("ejTreeGrid");
var data = {
taskId: "40",
taskName: "New Task 40",
startDate: "2/20/2014",
startDate: "2/25/2014"
};
treeGridObj.addRow(data, ej.TreeGrid.RowPosition.Child); // To add a task
})
</script>
```

Edit modes

TreeGrid supports the below editModes,

- Cell Editing
- Row Editing
- Dialog Editing
- Batch Editing

You can enable editing in TreeGrid by enabling the property AllowEditing.

Cell Editing

Update the record through editing a cell by setting EditMode as the CellEditing.

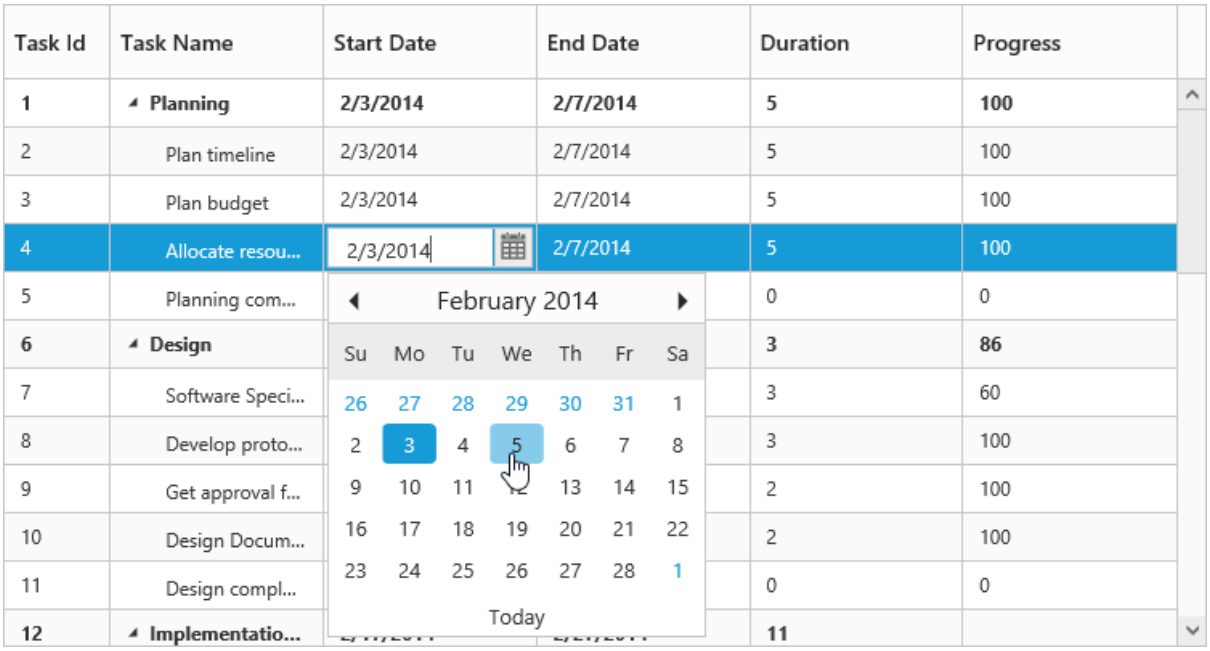
The following code example shows you how to enable the CellEditing in TreeGrid control.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.CellEditing);
})
)
@(Html.EJ().ScriptManager())
```

The output of TreeGrid with CellEditing is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Prevent cell editing

In cell edit action BeginEdit and EndEdit events are triggered before and after the editing action. Cell editing for specific cell can be prevented by using BeginEdit event.

The following code example show, how to prevent cell editing in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.CellEditing);
})
.ClientSideEvents(eve=>
{
eve.BeginEdit("beginEdit");
eve.EndEdit("endEdit");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function beginEdit(args) {
if (args.columnIndex == 1)
args.cancel = true;
}
function endEdit(args) {
//To perform actions after editing
}
</script>
```

Row Editing

It is possible to make the entire row to editable state and to update a record by setting EditMode as RowEditing.

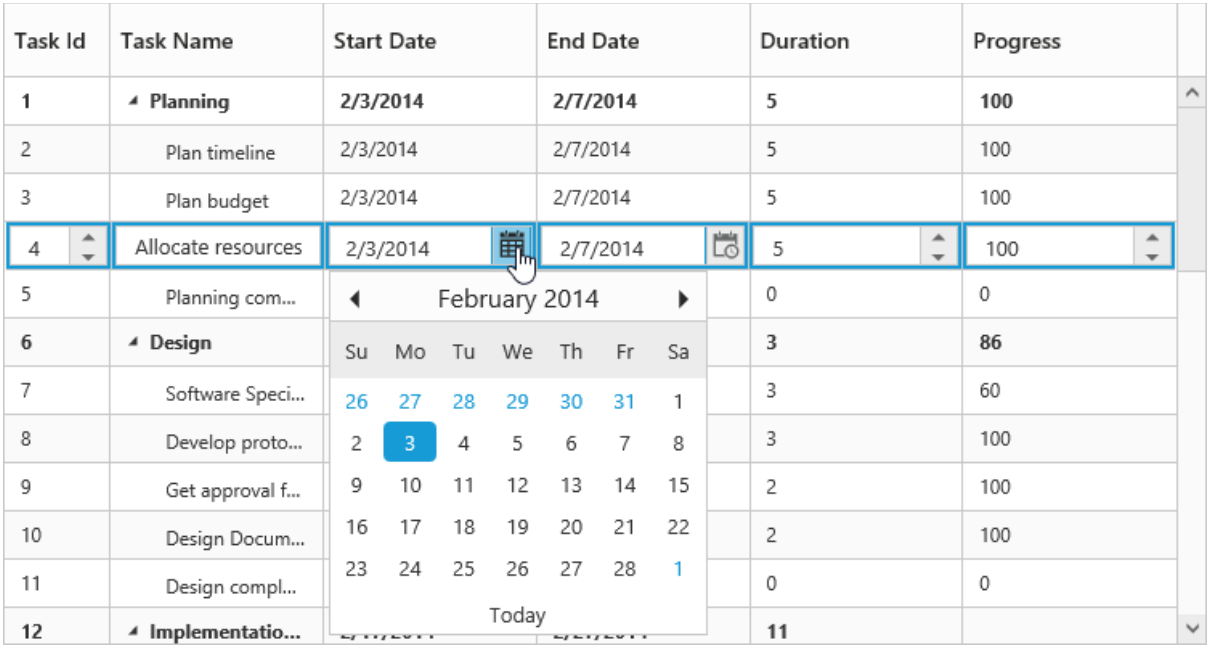
Copyright © 2001 - 2020 Syncfusion Inc.

The following code example shows you how to enable RowEditing in TreeGrid control.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.RowEditing);
})
)
@(Html.EJ().ScriptManager())
```

The following output is displayed as a result of the above code example.



Dialog Editing

Set the EditMode as DialogEditing to edit/add a record using dialog.

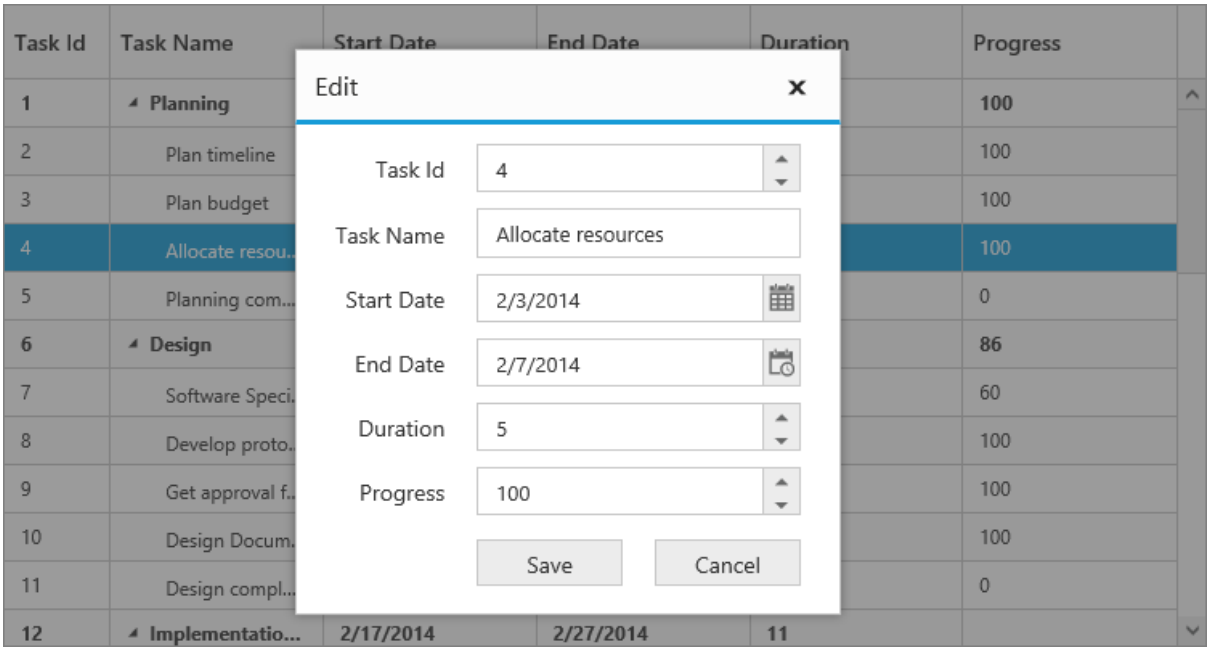
The following code example shows you how to enable the DialogEditing in TreeGrid control.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.DialogEditing);
})
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with DialogEditing is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



The add and edit dialogs can be opened on custom actions instead of toolbar icons using the methods showAddDialog and showEditDialog.

Prevent dialog editing

In dialog editing action ActionBegin and ActionComplete client side events are triggered before and after the edit action. Dialog editing for specific row can be prevented by using ActionBegin event.

The following code example show, how to prevent dialog editing in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.DialogEditing);
})
.ClientSideEvents(eve=>
{
eve.ActionBegin("actionBegin");
eve.ActionComplete("actionComplete");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function actionBegin(args) {
if(args.requestType == "beforeOpenEditDialog")
{
if(args.data.taskID == 4)
args.cancel = true;
}
}
function actionComplete(args) {
//To perform actions on action complete
}
```

```
</script>
```

Note: While saving the edited record ActionComplete event will be triggered with updated record value in data argument and requestType as recordUpdate. Using this event we can update the database.

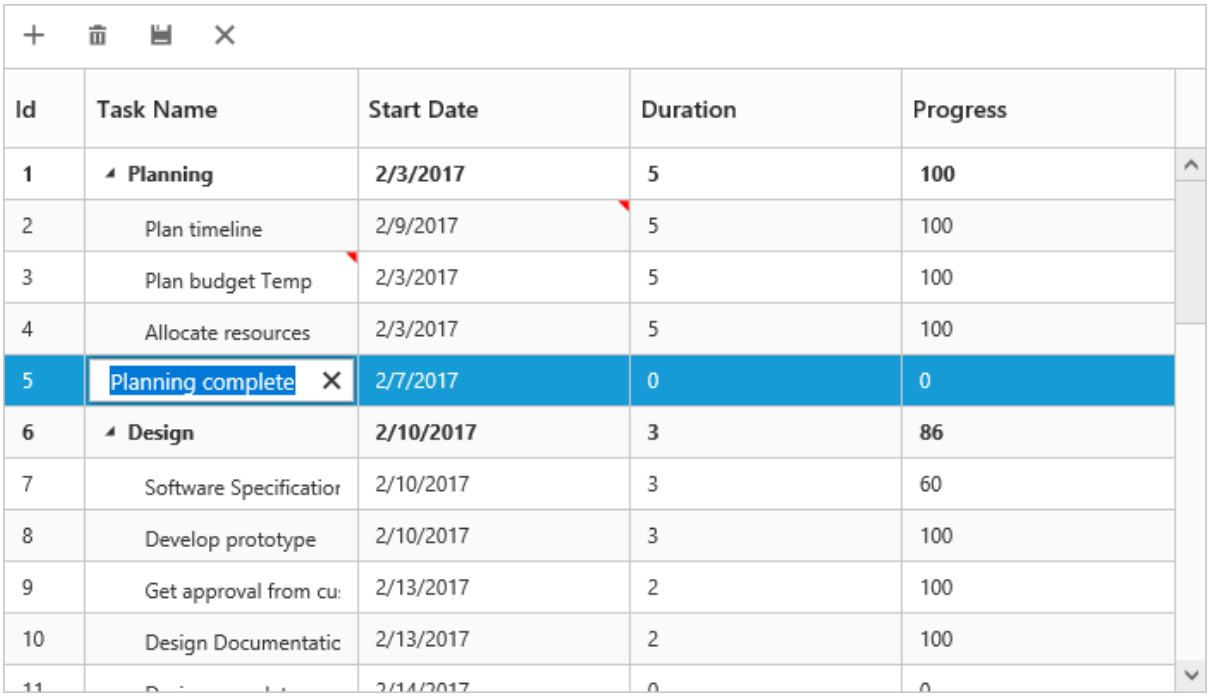
Batch Editing

The batch editing support in the tree grid is used to save all added, edited, and deleted changes to the database with a single action. This can be enabled by setting the EditMode property to BatchEditing. The following code example shows how to enable the BatchEditing in the tree grid control.

HTML

```
@(Html.EJ().TreeGrid("TreeGrid")
.EditSettings(ed =>
.EditMode(TreeGridEditMode.BatchEditing))
)
```

The output of a tree grid with BatchEditing is as follows.



In batch editing, the edit mode can be changed to **Cell** or **Row** or **Dialog** with the BatchEditSettings.EditMode property. The following code snippet shows how to set the editMode to row in the BatchEditSettings property.

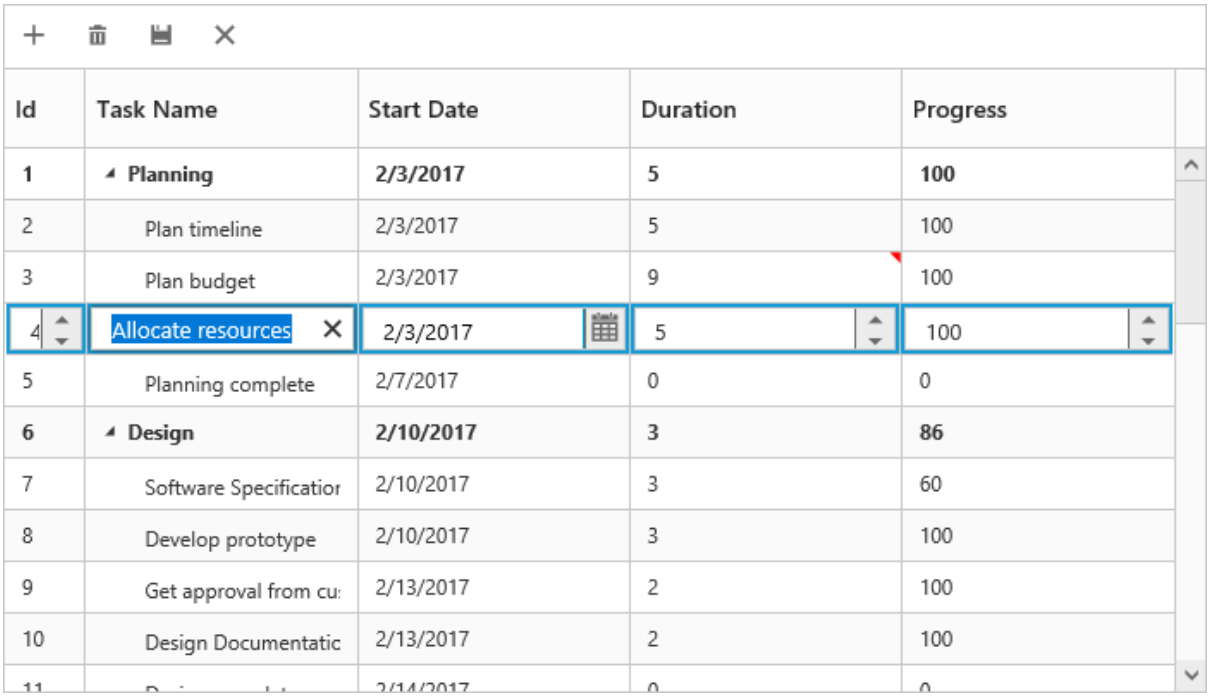
JS

```
@(Html.EJ().TreeGrid("TreeGrid")
.EditSettings(ed =>
ed.BatchEditSettings(ba =>
ba.EditMode(TreeGridBatchEditMode.Row))
.EditMode(TreeGridEditMode.BatchEditing))
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
```

The output of a tree grid with BatchEditSettings and EditMode set as Row is as follows.



Note: After modifying all changes in the tree grid, click the save button in the toolbar. The ActionComplete event will be triggered with updated records in the batchChanges argument with requestType argument as batchSave. Using this event, you can update all the modified records to the database.

Edit type and its params

The edit type of columns can be customized using EditType property of Columns. The following controls are supported built-in by EditType. You can set the EditType based on specific data type of the column.

- **CheckBox** control for boolean data type.
- **NumericTextBox** control for integers, double, and decimal data types.
- **InputTextBox** control for string data type.
- **DatePicker** control for date data type.
- **DateTimePicker** control for date-time data type.
- **DropDownList** control for list of data type.

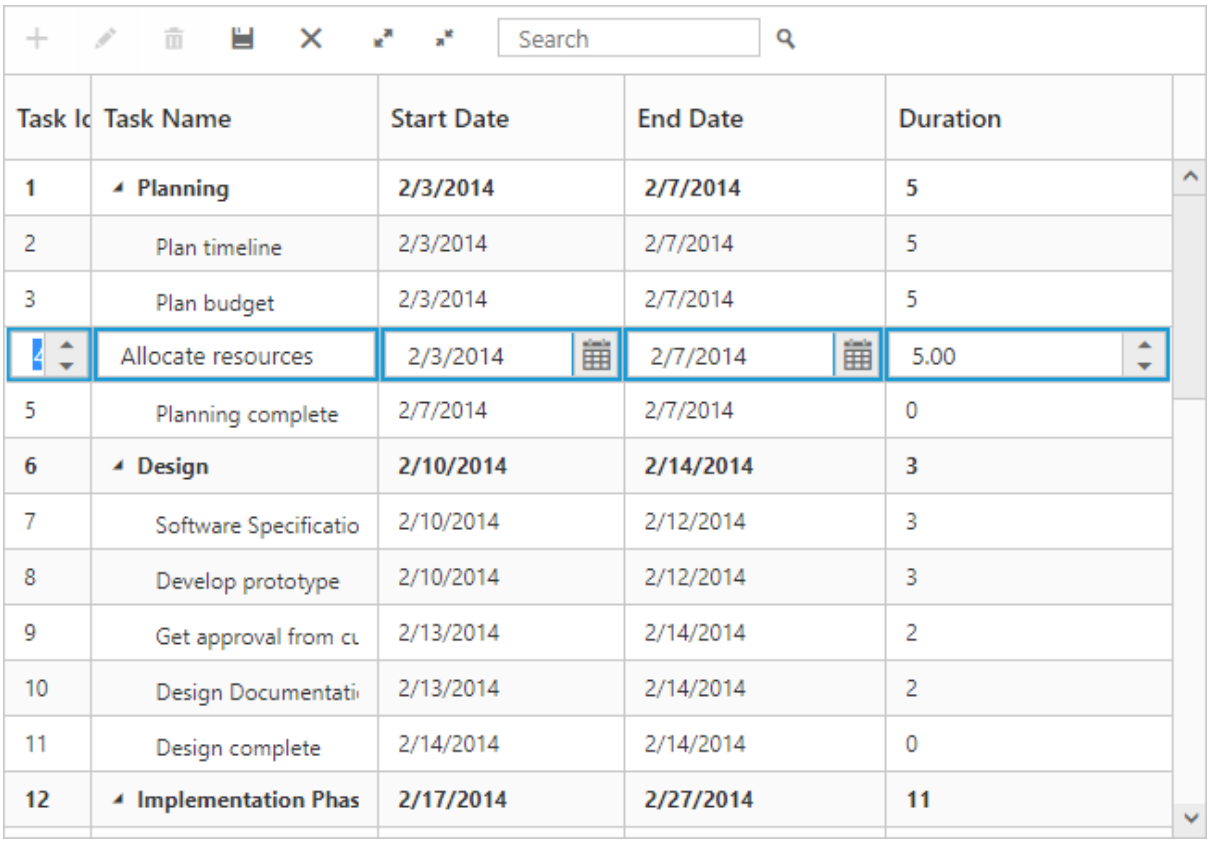
And also you can define the model for all the EditTypes controls while editing through EditOptions property of Columns.

The following code example describes the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(co =>
```

```
{
co.Field("taskID").HeaderText("Task
Id").EditType(TreeGridEditingType.Numeric).Add();
co.Field("taskName").HeaderText("Task
Name").EditType(TreeGridEditingType.String).Add();
co.Field("startDate").HeaderText("Start
Date").EditType(TreeGridEditingType.Datepicker).Add();
co.Field("endDate").HeaderText("End
Date").EditType(TreeGridEditingType.Datepicker).Add();
co.Field("duration").HeaderText("Duration").EditType(TreeGridEditingType.
Numeric)
.NumericEditOptions(new EditorProperties { DecimalPlaces= 2 }).Add();
})
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows TreeGrid with different Edit type.

Assign data source for drop down edit type

In TreeGrid, we need to assign the data source for drop down list control to populate the suggestion list while editing the column in TreeGrid. The DropdownData property is used to set the drop down list data source in TreeGrid control.

The following code example shows how to set data source for drop down edit type.

C#

```
public class TreeGridController : Controller
```

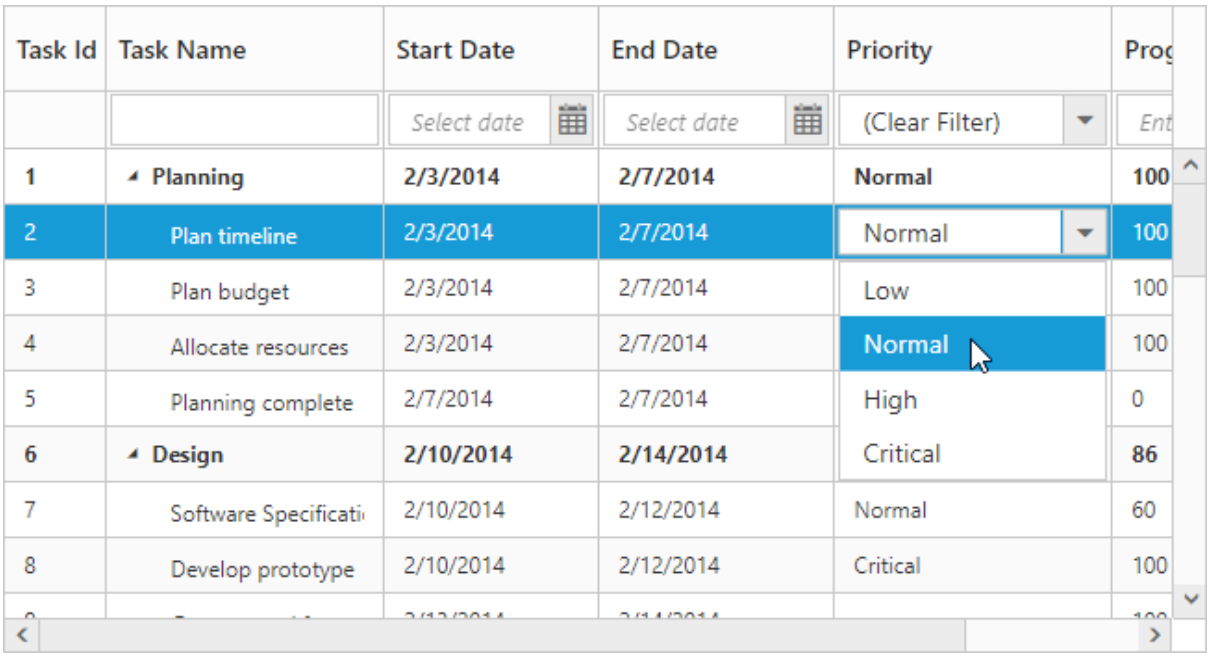
Copyright © 2001 - 2020 Syncfusion Inc.

```
{
//
// GET: /TreeGridColumn/
public ActionResult TreeGridColumn()
{
//..
StageDetails stages = new StageDetails();
ViewBag.dropData = stages.GetStageCollection();
return View();
}
public class Stage
{
public int id { get; set; }
public string text { get; set; }
public string value { get; set; }
}
public class StageDetails
{
public List<Stage> GetStageCollection()
{
List<Stage> stageCollection = new List<Stage>();
stageCollection.Add(new Stage() { id = 1, text = "Low", value = "Low" });
stageCollection.Add(new Stage() { id = 2, text = "Normal", value =
"Normal" });
stageCollection.Add(new Stage() { id = 3, text = "High", value = "High"
});
stageCollection.Add(new Stage() { id = 4, text = "Critical", value =
"Critical" });
return stageCollection;
}
}
}
```

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(co =>
{
co.Field("Priority").HeaderText("Priority").EditType(TreeGridEditingType.
Dropdown).DropDownData((IEnumerable<object>)ViewBag.dropData).Add();
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows drop down edit type in TreeGrid.

Dialog Template

You can edit any of the fields pertaining to a single record of data and apply it to a template so that the same format is applied to all the other records that you may edit later.

Using this template support, you can edit/add the fields that are not bound to TreeGrid columns.

To edit/add the records using dialog template form, set `EditMode` as `DialogEditing` and specify the template id to `DialogEditorTemplateID` property of `EditSettings`.

Note: The `value` attribute is used to bind the corresponding field value while editing.

The `name` attribute is used to get the changed field values while saving the edited record.

The `id` attribute must to be set in the format of (treegrid control id + fieldname).

The following code example describes the above behavior.

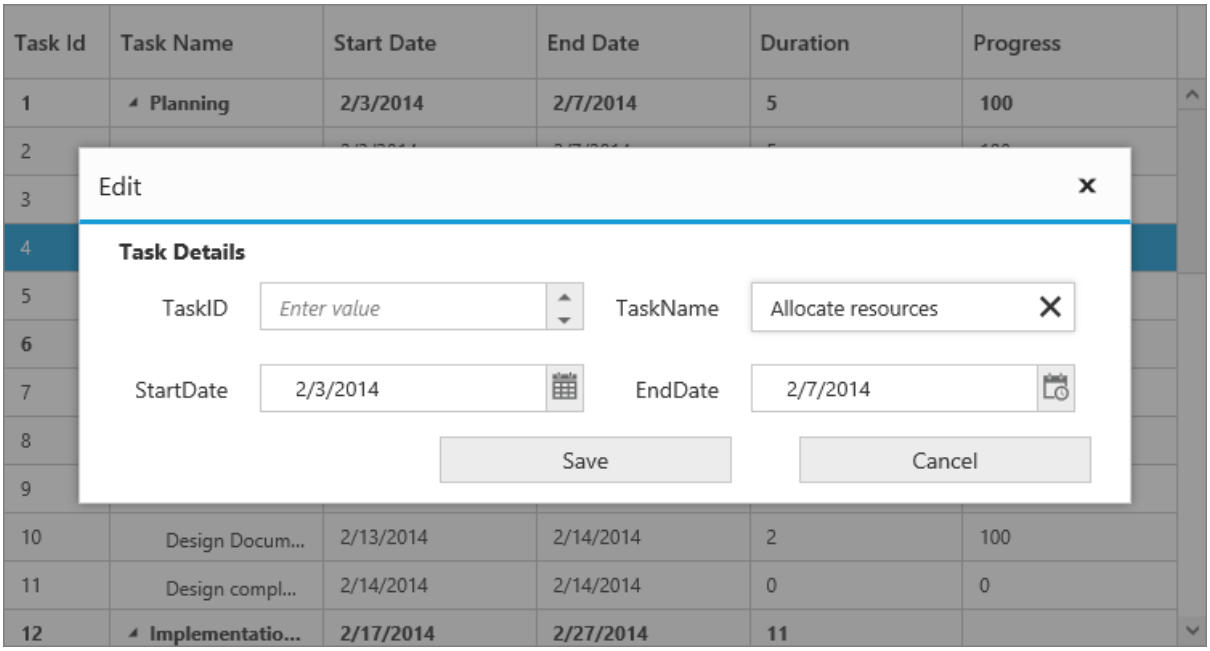
ASPX-CS

```
<script type="text/x-jsrender" id="template">
<div>
<b>Task Details</b>
<table cellspacing="10" class="beta">
<tr>
<td style="text-align:right;padding: 10px;">
TaskID
</td>
<td style="text-align: left;padding: 10px;">
<input id="TreeGridContainertaskID" type="number" name="taskID"
value="{{'{{'}}:taskID{{}}}}" disabled="disabled" class="e-field e-
ejinputtext valid e-disable"/>
</td>
<td style="text-align: right;padding: 10px;">
TaskName
```

```
</td>
<td style="text-align: left;padding: 10px;">
<input id="TreeGridContainertaskName" name="taskName"
value="{{'{{'}}:taskName{{}}}}" class="e-field e-ejinputtext valid"/>
</td>
</tr>
<tr>
<td style="text-align: right;padding: 10px;">
StartDate
</td>
<td style="text-align: left;padding: 10px;">
<input type="text" id="TreeGridContainerstartDate" name="startDate"
value="{{'{{'}}:startDate{{}}}}" class="e-field e-ejinputtext valid" />
</td>
<td style="text-align: right;padding: 10px;">
EndDate
</td>
<td style="text-align: left;padding: 10px;">
<input id="TreeGridContainerendDate" type="text" name="endDate"
value="{{'{{'}}:endDate{{}}}}" class="e-field e-ejinputtext valid"  />
</td>
</tr>
</table>
</div>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.DialogEditing);
edit.DialogEditorTemplateID("#template");
})
)
@(Html.EJ().ScriptManager())
```

The following output is displayed as a result of the above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the online demo sample for Dialog Template.

Using methods to open dialog

It is possible to open the add dialog dynamically using showAddDialog method.

Similarly, open the edit dialog dynamically using the method showEditDialog(index), with the index of the row to be edited as parameter.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
$("#add").click(function () {
treegridObj = $("#TreeGridContainer").data("ejTreeGrid");
treegridObj.showAddDialog();
})
$("#edit").click(function () {
treegridObj = $("#TreeGridContainer").data("ejTreeGrid");
treegridObj.showEditDialog(3);
})
</script>
```

BeginEditAction - edit cell/row by single/double click

In TreeGrid we can perform edit action by single or double click using `BeginEditAction` , default value of this property is `DblClick`.

The following code example shows how to enable single click edit in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
//..
edit.BeginEditAction(TreeGridBeginEditAction.Click);
})
)
@(Html.EJ().ScriptManager())
```

Cell Edit Template

Edit template is used to create custom editor for editing the column values. This can be achieved by using Columns.EditTemplate.

Following functions required to render edit template in TreeGrid,

- Create - It is used to create the control on initialization.
- Read - It is used to read the input value on save.
- Write - It is used to assign the value to control on editing.

The following code example describes edit template behavior

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.Columns(co =>
{
co.Field("TaskName").HeaderText("Task Name").EditTemplate(new
TreeGridEditTemplate() { Create="create",Write="write",Read="read"
}).Add();
})
)
@(Html.EJ().ScriptManager())
```

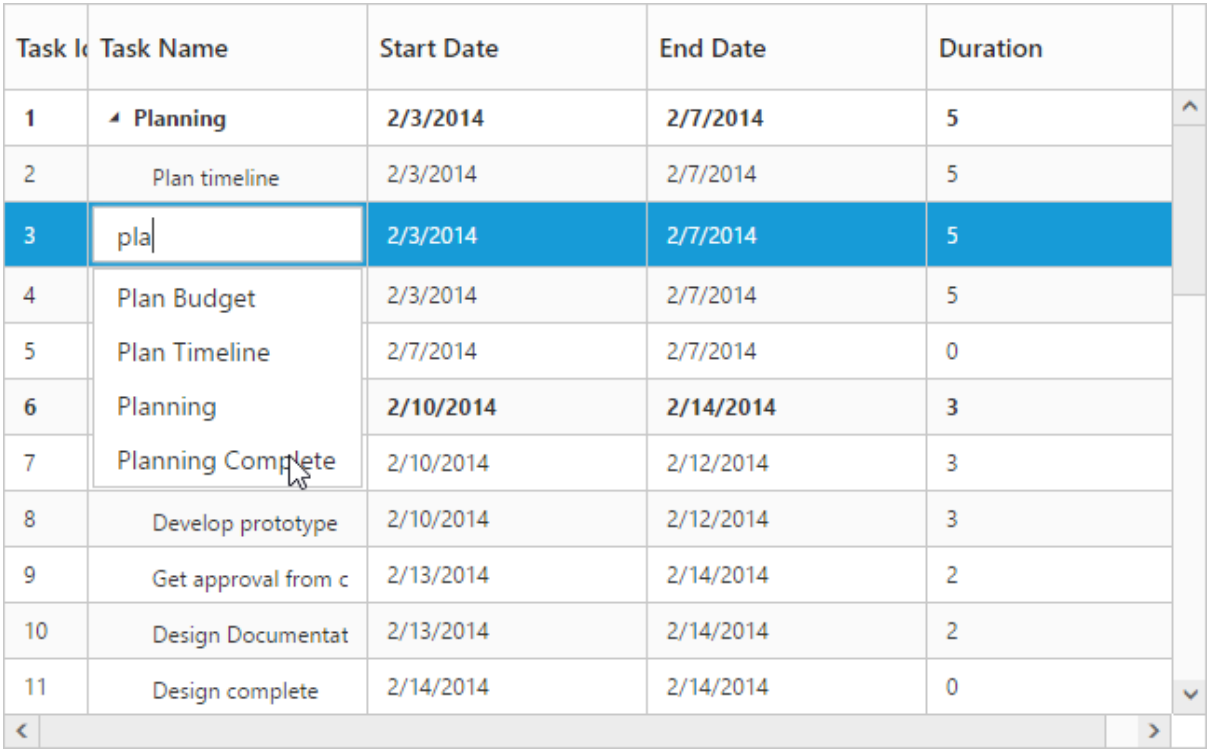
JS

```
<script>
var autocompleteData = ["Planning", "Plan Timeline", "Plan Budget",
"Allocate Resources", "Planning Complete"];
function create()
{
return "<input>";
}
function write(args)
{
args.element.ejAutocomplete({
width: "100%",
dataSource: autocompleteData,
enableDistinct: true,
value: args.rowData !== undefined ? args.rowData["taskName"] : ""
});
}
function read(args)
{
args.ejAutocomplete('suggestionList').css('display', 'none');
return args.ejAutocomplete("getValue");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</script>
```

The output of the TreeGrid width EditTemplate as follows



The updated record values are maintained in collection in TreeGrid, and the user can retrieve the updated record collection at any time by using the getUpdatedRecords method.

Click here to view the online demo sample for Edit Template.

Delete record

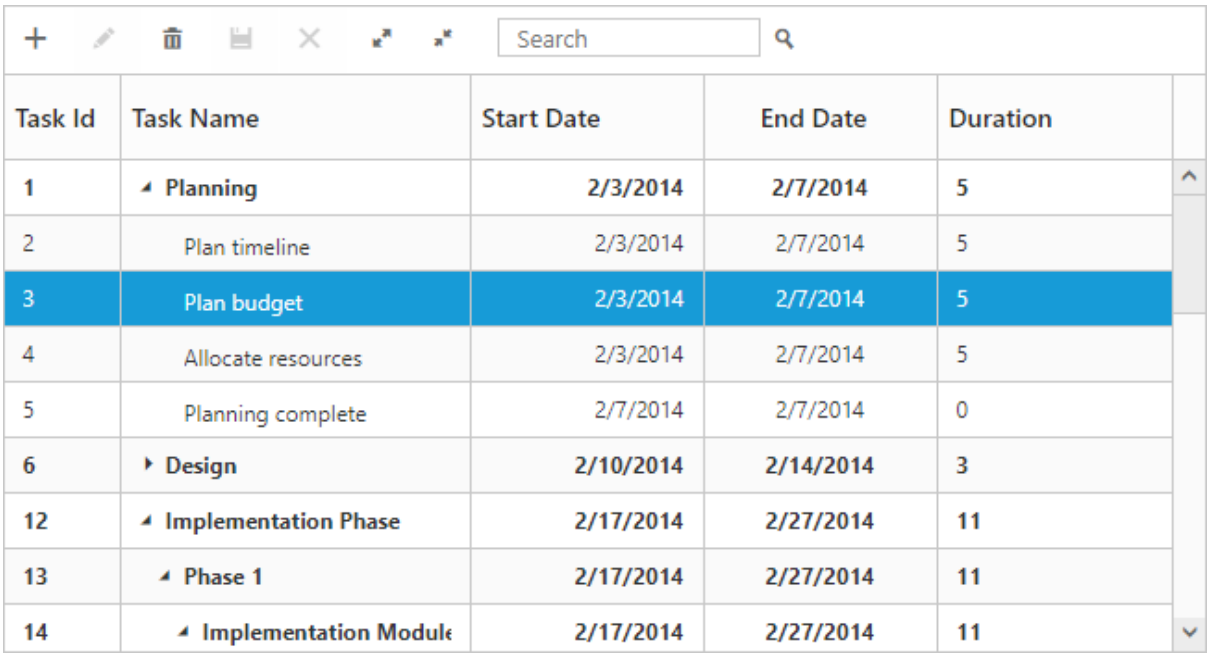
TreeGrid provides support to delete a record by enabling AllowDeleting property.

The below code example shows how to enable delete option in TreeGrid.

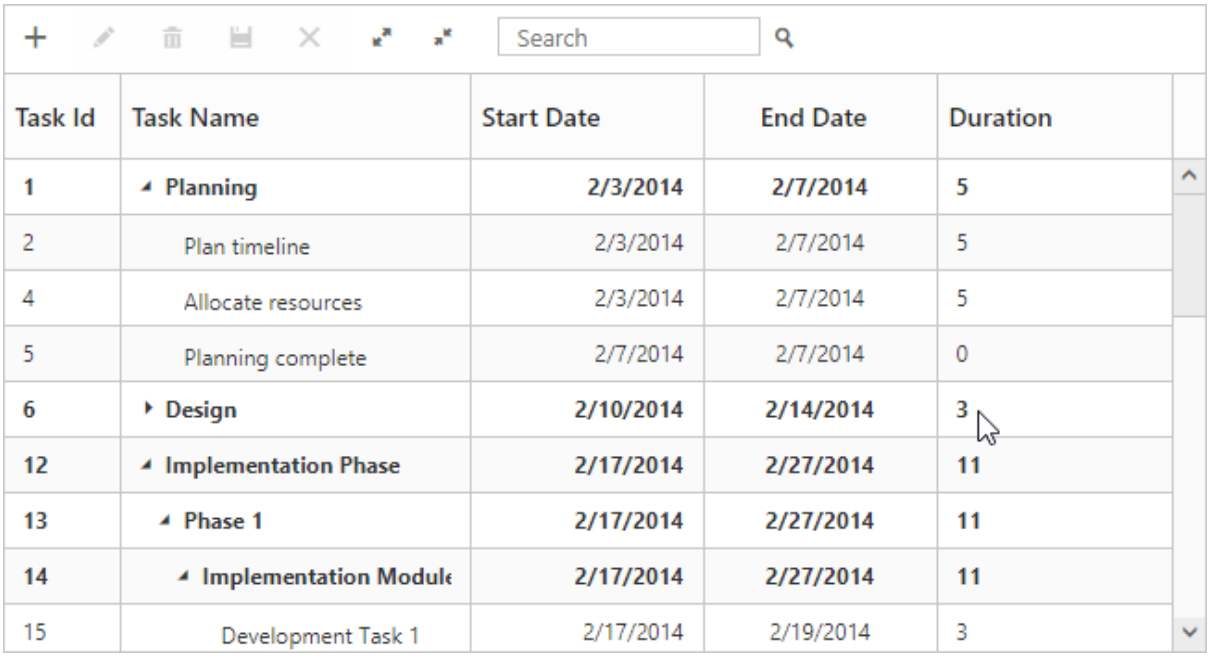
ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.EditSettings(edit=>
{
edit.AllowDeleting(true);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screenshot shows before delete a record in TreeGrid.



The above screenshot shows after delete a record in TreeGrid.

Delete confirmation message

Delete confirmation message is used to get the confirmation from the user before deleting action. This confirmation message can be enabled by setting `ShowDeleteConfirmDialog` property as `true`.

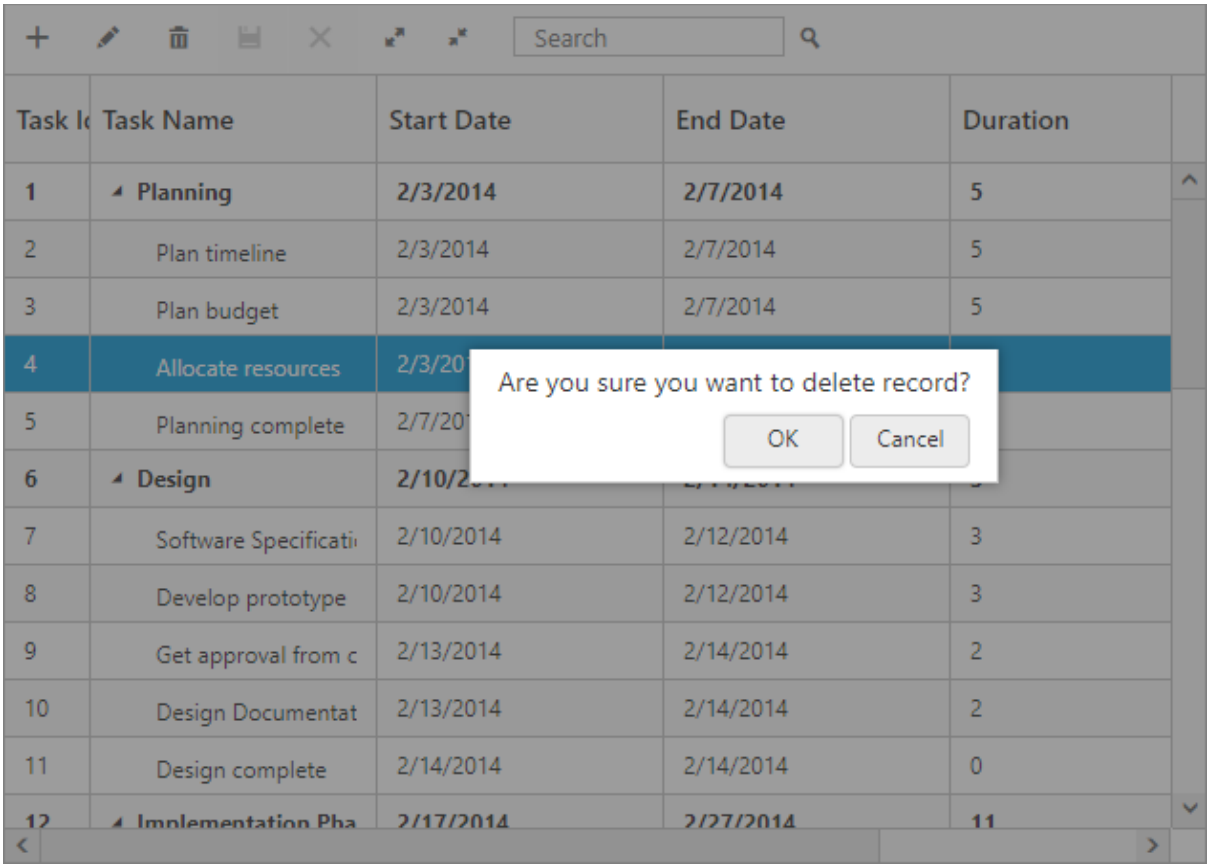
The following code snippet explains how to enable delete confirmation message in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.EditSettings(edit=>
{
edit.AllowDeleting(true);
edit.ShowDeleteConfirmDialog(true);
})
)
@(Html.EJ().ScriptManager())
```



Click here to view the online demo sample for Editing.

Filtering

Filtering helps to view specific or related records from data source which meets a given filtering criteria. To enable filtering in TreeGrid, set the `AllowFiltering` as `true`.

The `FilterSettings` property is used to customize the filtering in TreeGrid.

TreeGrid provides support for the following filtering modes.

- FilterBar
- Menu
- Excel

Also, the following filtering types are available in TreeGrid.

- String

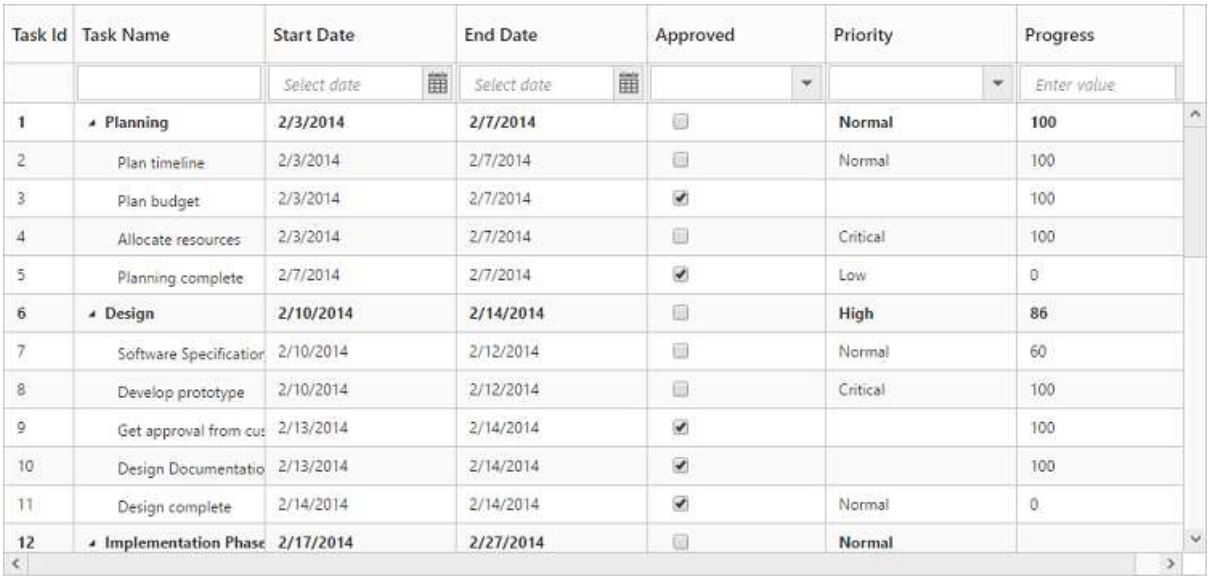
- Boolean
- Date
- Numeric
- Dropdown

The below code explains how to enable filtering in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with filtering enabled is as follows.



Task Id	Task Name	Start Date	End Date	Approved	Priority	Progress
		Select date	Select date			Enter value
1	Planning	2/3/2014	2/7/2014	☐	Normal	100
2	Plan timeline	2/3/2014	2/7/2014	☐	Normal	100
3	Plan budget	2/3/2014	2/7/2014	☑		100
4	Allocate resources	2/3/2014	2/7/2014	☐	Critical	100
5	Planning complete	2/7/2014	2/7/2014	☑	Low	0
6	Design	2/10/2014	2/14/2014	☐	High	86
7	Software Specificatior	2/10/2014	2/12/2014	☐	Normal	60
8	Develop prototype	2/10/2014	2/12/2014	☐	Critical	100
9	Get approval from cus	2/13/2014	2/14/2014	☑		100
10	Design Documentatio	2/13/2014	2/14/2014	☑		100
11	Design complete	2/14/2014	2/14/2014	☑	Normal	0
12	Implementation Phase	2/17/2014	2/27/2014	☐	Normal	

The TreeGrid allows the user to filter the columns with custom actions by using the method filterColumn. And it is possible to clear the filter for a specific column by using the method clearFilter.

Filtering Modes

Filter Bar

This is the default filtering mode in TreeGrid. It can also be enabled by setting the `FilterSettings.FilterType` as `FilterBar`. When this filtering mode is enabled, a filter row will be displayed below the column header, in which we can provide the filter query.

There are two types of actions available to initiate the filtering process in the filter bar mode, we can set the filter bar mode using `FilterSettings.FilterBarMode` property.

`Immediate`: Filtering action will be initiated immediately on key press, for each character being typed in the filter bar.

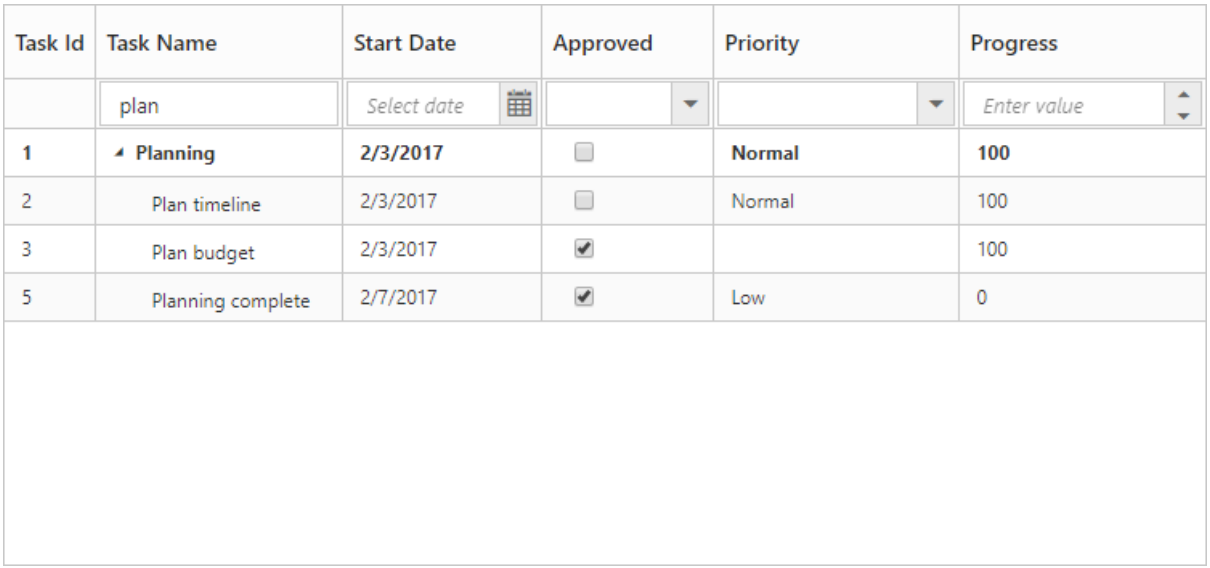
`OnEnter`: Filtering action will be initiated only on enter key press in the filter bar.

The below code snippet explains the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.FilterSettings(filtered=>
{
filtered.FilterType(TreeGridFilterType.FilterBar);
filtered.FilterBarMode(TreeGridFilterBarMode.OnEnter);
})
)
@(Html.EJ().ScriptManager())
```

The output of the filtering with filter bar enabled is as follows.



Menu filtering

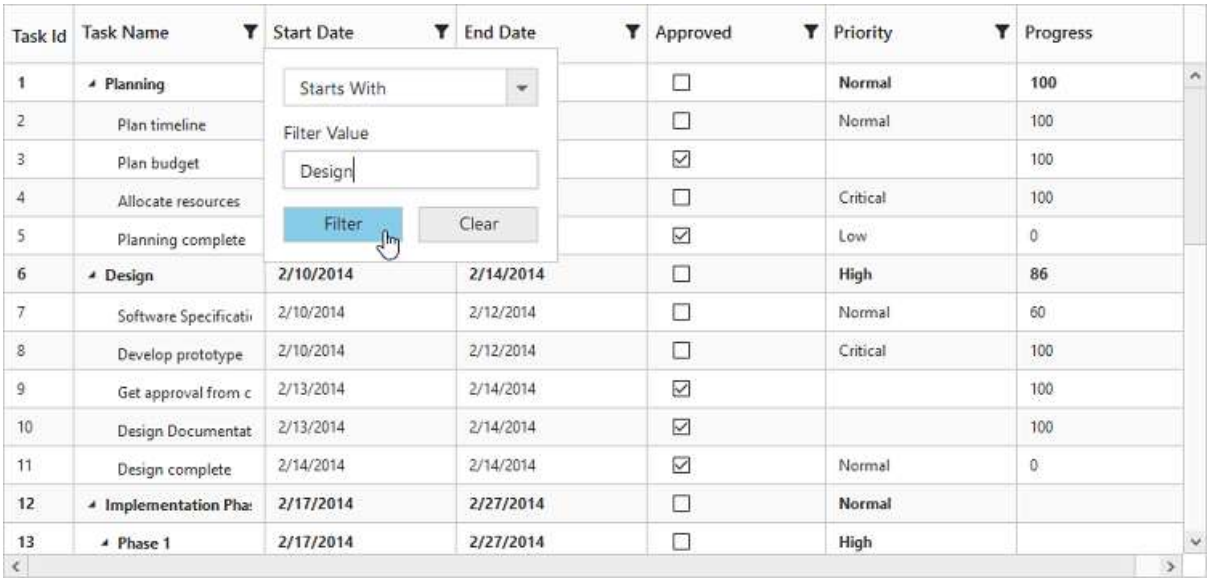
Menu filtering can be enabled by setting the FilterSettings.FilterType property as Menu. The below code snippet explains how to enable menu filtering in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.FilterSettings(filtered=>
{
filtered.FilterType(TreeGridFilterType.Menu);
})
)
@(Html.EJ().ScriptManager())
```

The output of the filtering with filter menu enabled is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Excel Filtering

Excel filtering can be enabled by setting the FilterSettings.FilterType property as Excel. The excel menu contains options such as sorting, clear filter and advance filtering options.

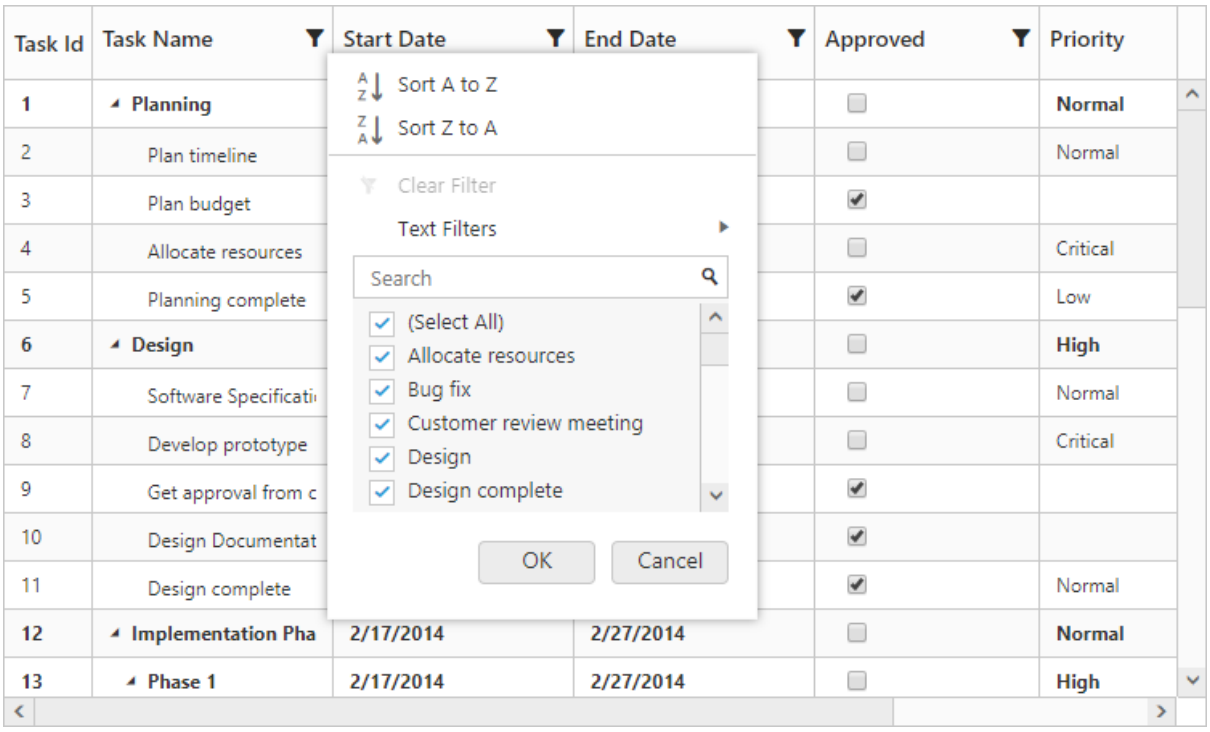
The below code snippet explains how to enable excel filtering in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.FilterSettings(filtered=>
{
filtered.FilterType(TreeGridFilterType.Excel);
})
)
@(Html.EJ().ScriptManager())
```

The output of the filtering with excel filter enabled is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Checkbox list generation

By default, the checkbox list is generated from distinct values of the filter column from `Datasource` which gives an option to search and select the required items.

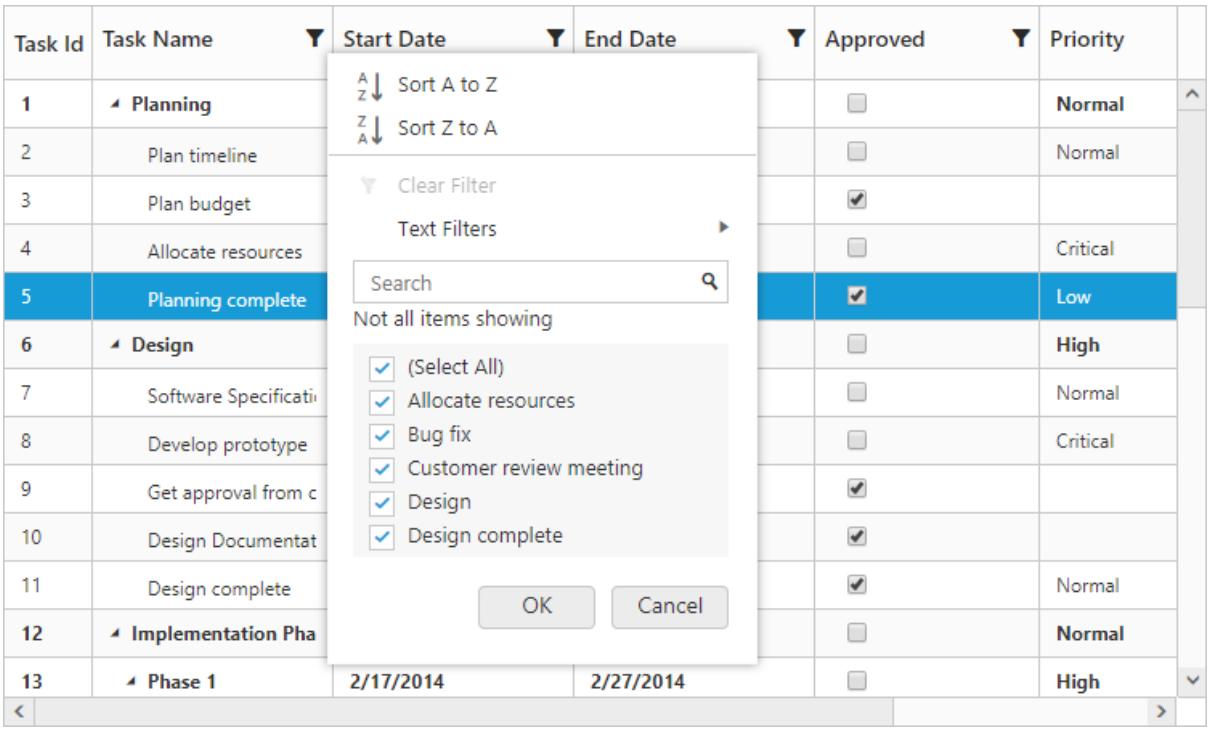
If the number of distinct values are greater than 1000, then the excel filter will display only first 1000 values and show "Not all items shown" label to ensure the best performance on rendering and searching. However, this limit has been customized according to your requirement by setting the `MaxFilterChoices` with required limit in integer.

The below code snippet explains how to change the max filter choices value.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.FilterSettings(filtered=>
{
filtered.FilterType(TreeGridFilterType.Excel);
filtered.MaxFilterChoices(5);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screen shot shows TreeGrid with MaxFilterChoices as 5.

Case Sensitivity

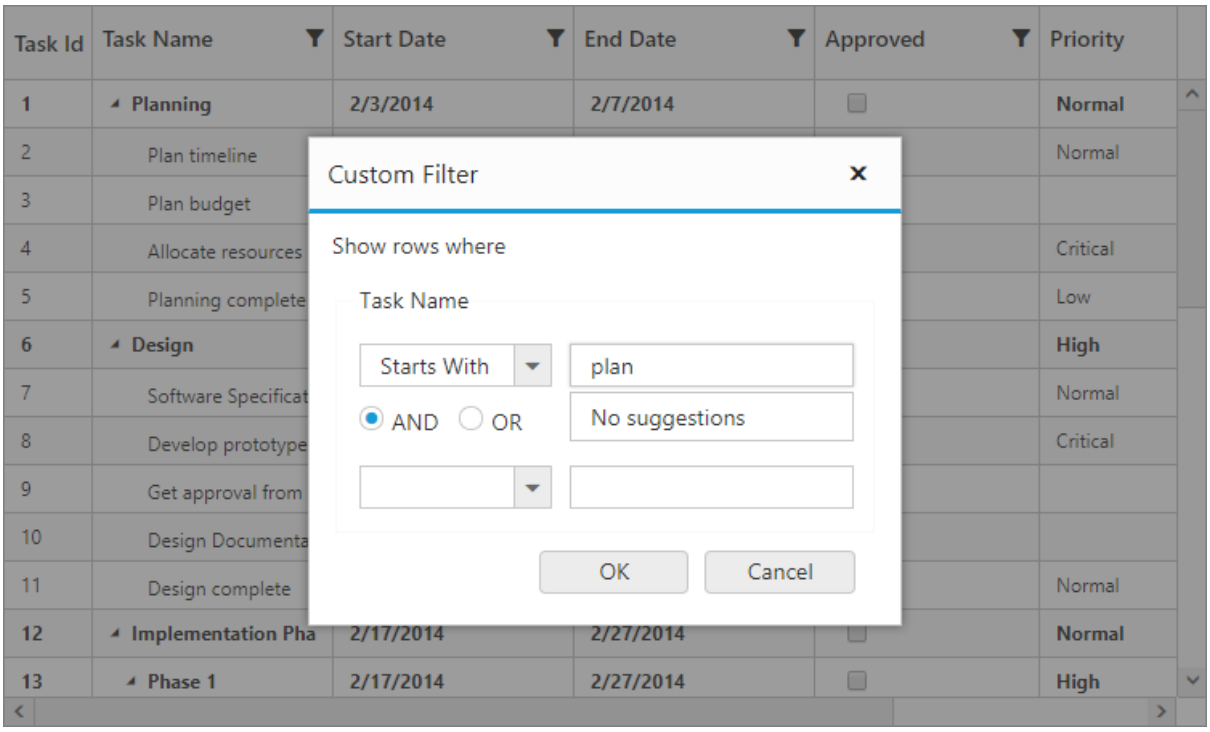
To perform filter operation with case sensitive in excel styled filter menu mode by setting the EnableCaseSensitivity as true.

The below code snippet explains how to enable the case sensitivity in excel filter.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.FilterSettings(filtered=>
{
filtered.FilterType(TreeGridFilterType.Excel);
filtered.EnableCaseSensitivity(true);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screen shot shows TreeGrid with EnableCaseSensitivity set as false in search action.

Change filter mode for specific column

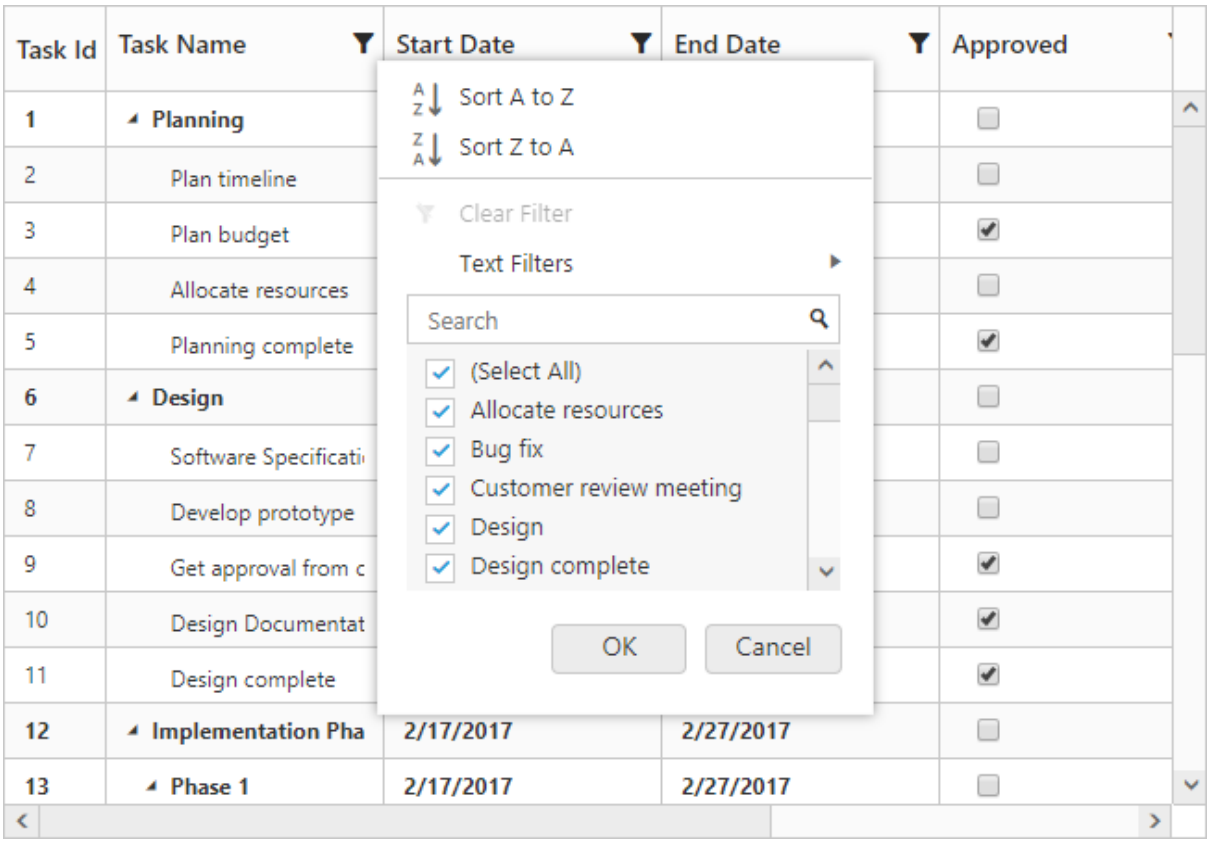
TreeGrid provides option for changing the filter mode for specific column by using Columns.FilterType property. Using this property we can either set Menu or Excel filter mode when FilterSettings.FilterType property as Menu or Excel.

The following code example show, how to change filter mode for specific column.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.FilterSettings(filtered=>
{
filtered.FilterType(TreeGridFilterType.Menu);
})
.Columns(co =>
{
co.Field("taskName").HeaderText("Task
Name").EditType(TreeGridEditingType.String).FilterType(TreeGridFilterOpti
on.Excel).Add();
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.



The above screen shot shows TreeGrid with excel filter mode for Task Name column only.

Filtering types

By default, the filtering type for a column is inherited from the Columns.EditType property. You can also define a specific filtering type for a column using the Columns.FilterEditType property.

The below code snippet explains on how to set a filtering type for a column.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.Columns(co =>
{
co.Field("taskID").AllowFiltering(false).Add();
co.Field("taskName").EditType(TreeGridEditingType.String).FilterEditType(
TreeGridFilterEditType.String).Add();
co.Field("startDate").EditType(TreeGridEditingType.Datepicker).Format("{0
:MM/dd/yyyy}").FilterEditType(TreeGridFilterEditType.Datepicker).Add();
co.Field("priority").EditType(TreeGridEditingType.Dropdown).DropDownData(
(IEnumerable<object>)ViewBag.dropData).FilterEditType(TreeGridFilterEditT
ype.Dropdown).Add();
co.Field("progress").EditType(TreeGridEditingType.Numeric).FilterEditType
(TreeGridFilterEditType.Numeric).Add();
})
)
@(Html.EJ().ScriptManager())
```

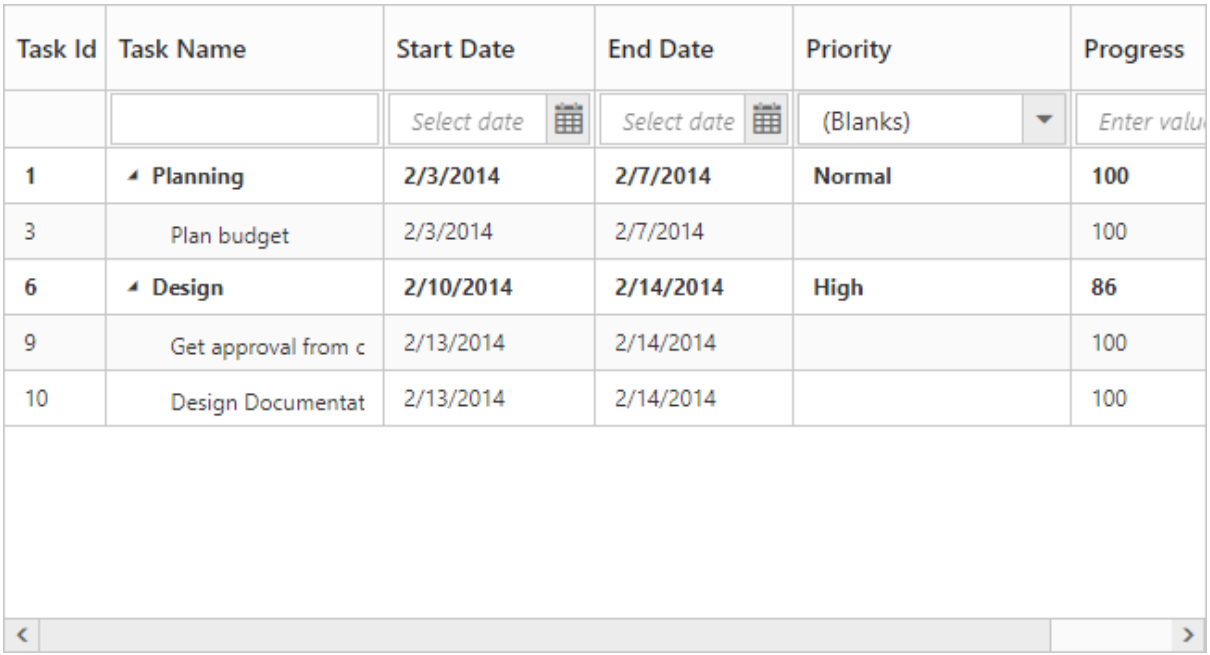
Copyright © 2001 - 2020 Syncfusion Inc.

Filtering blank values in drop down filtering

In TreeGrid, it is possible to filter the null or empty string value in drop down filter type by setting `AllowFilteringBlankContent` as `true` in the column definition.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.Columns(co =>
{
co.Field("priority").HeaderText("Priority").FilterEditType(TreeGridFilter
EditType.Dropdown).AllowFilteringBlankContent(true).Add();
})
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows TreeGrid with filtering blank content.

Filter Hierarchy Modes

TreeGrid provides support for a set of filtering modes with `FilterSettings.FilterHierarchyMode` property.

The below are the type of filter mode available in TreeGrid.

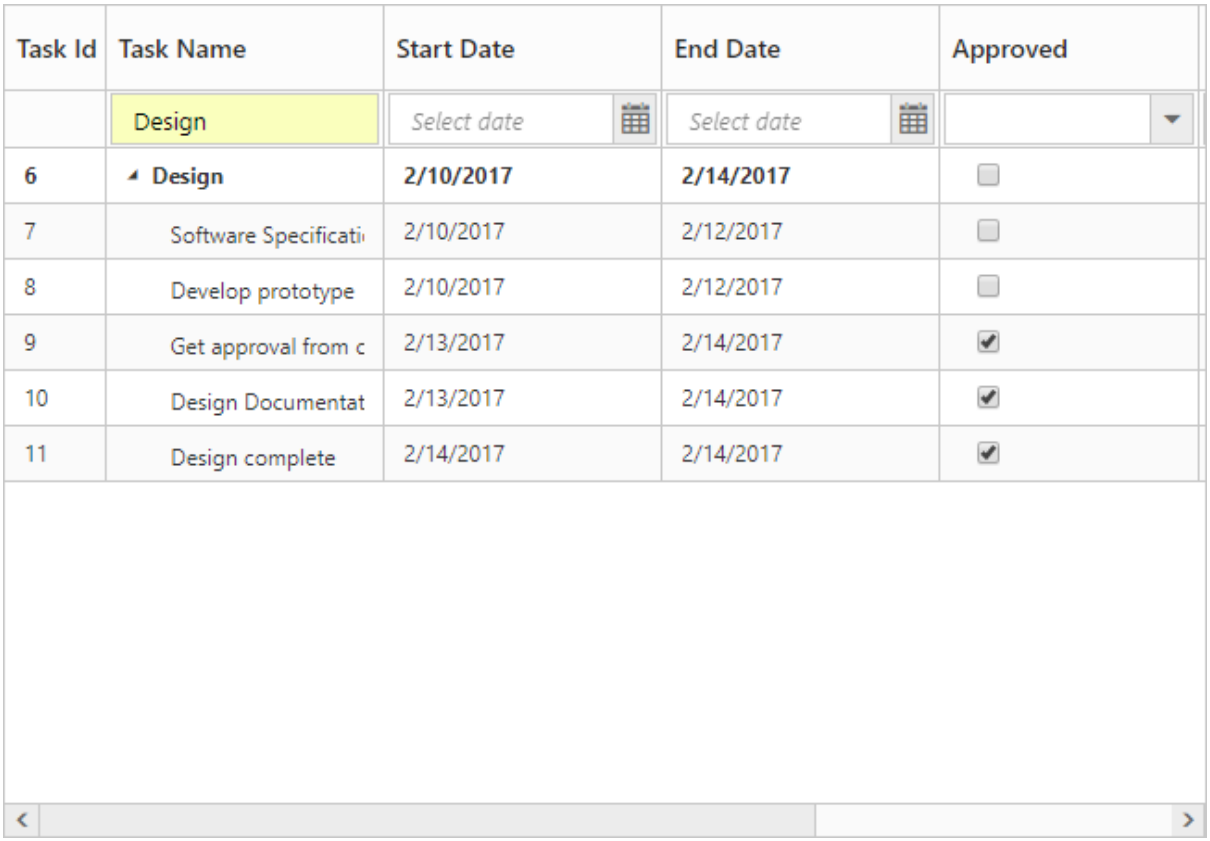
- **Parent**: This is the default filter hierarchy mode in TreeGrid. The filtered record displayed with its parent records, if the filtered records not have any parent record then the filtered record only displayed.
- **Child** : The filtered record displayed with its child record, if the filtered records not have any child record then the filtered record only displayed.
- **Both** : The filtered record displayed with its both parent and child record, if the filtered records not have any parent and child record then the filtered record only displayed.
- **None** : The filtered record only displayed.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example shows how to set filter mode in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.FilterSettings(filtered =>
{
filtered.FilterHierarchyMode(TreeGridFilterHierarchyMode.Child);
})
)
@(Html.EJ().ScriptManager())
```



The above screenshot shows TreeGrid filter with Child filter mode.

Filter columns at initial load

It is also possible to filter one or more columns at load time by providing the FilterSettings.FilteredColumns.Field, FilterSettings.FilteredColumns.Value, FilterSettings.FilteredColumns.Predicate and FilterSettings.FilteredColumns.Operator to the FilterSettings.FilteredColumns property. The following code example explains how to filter a column on initial load.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.FilterSettings(filtered =>
{
filtered.FilteredColumns(filtered =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
filtered.Field("taskName").Value("plan").Predicate("and").Operator(Filter
OperatorType.StartsWith).Add();
});
})
)
@(Html.EJ().ScriptManager())
```

Disabling filtering for a specific column

It is possible to disable filtering for a specific column by setting the Columns.AllowFiltering as false in the column definition.

The below code snippet explains the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowFiltering(true)
.Columns(co =>
{
co.Field("taskID").AllowFiltering(false).Add();
})
)
@(Html.EJ().ScriptManager())
```

The output of the filtering enabled for only one column is as follows.

Task Id	Task Name	Start Date	End Date	Approved	Priority
1	Planning	2/3/2014	2/7/2014	☐	Normal
2	Plan timeline	2/3/2014	2/7/2014	☐	Normal
3	Plan budget	2/3/2014	2/7/2014	☑	
4	Allocate resources	2/3/2014	2/7/2014	☐	Critical
5	Planning complete	2/7/2014	2/7/2014	☑	Low
6	Design	2/10/2014	2/14/2014	☐	High
7	Software Specificati	2/10/2014	2/12/2014	☐	Normal
8	Develop prototype	2/10/2014	2/12/2014	☐	Critical
9	Get approval from c	2/13/2014	2/14/2014	☑	
10	Design Documentat	2/13/2014	2/14/2014	☑	
11	Design complete	2/14/2014	2/14/2014	☑	Normal
12	Implementation Pha:	2/17/2014	2/27/2014	☐	Normal

Click to find the demo sample for filtering in TreeGrid

Toolbox searching

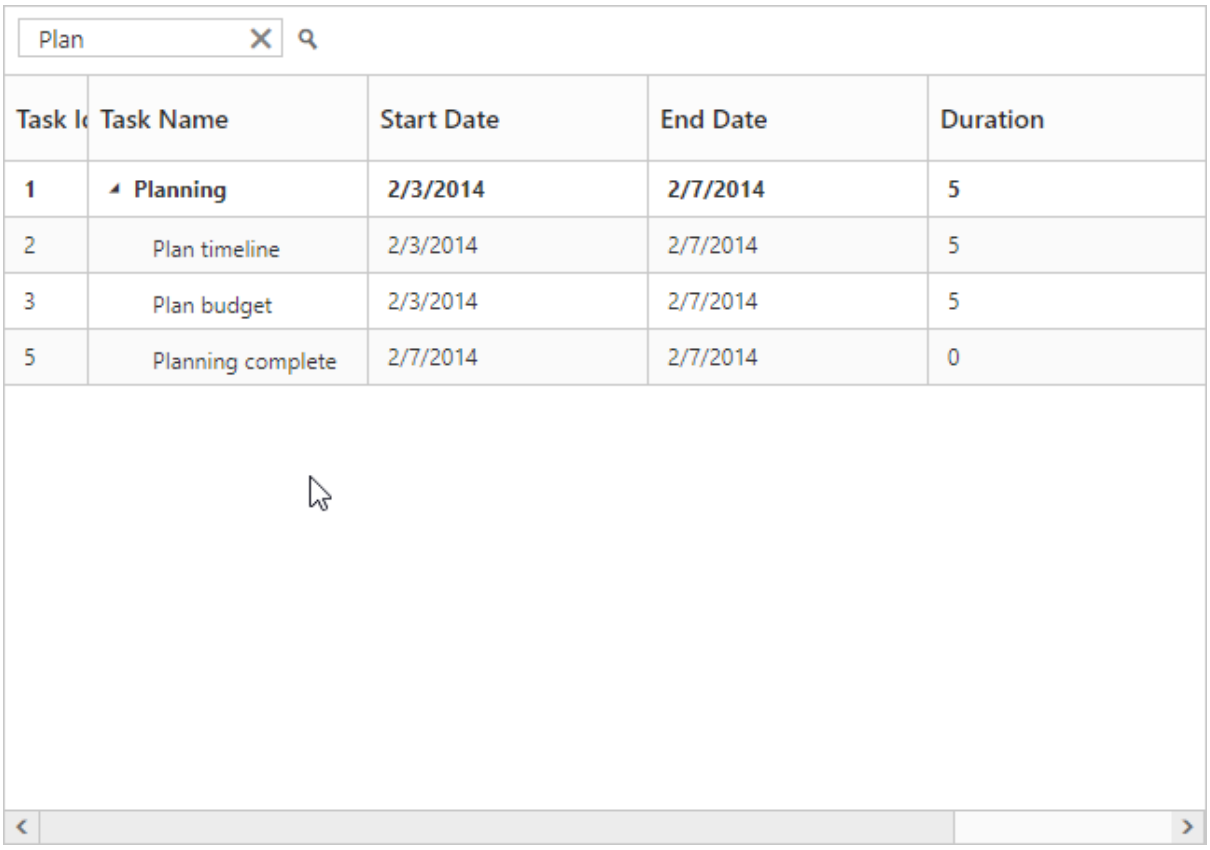
The TreeGrid control has an option to search its content using toolbar search box. The toolbar search box can be enabled by using the ToolbarSettings.ToolbarItems property.

Copyright © 2001 - 2020 Syncfusion Inc.

The following code example explains how to integrate search text box in toolbar.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.Search
});
})
)
@(Html.EJ().ScriptManager())
```



The above screen shot shows TreeGrid search with **Plan** key word.

The toolbox searching can be customized using the SearchSettings property. It is possible to search the TreeGrid content with the specific column values as the query for searching, using the property SearchSettings.Field.

The below code example explains searching the TreeGrid content across the "TaskId" and "TaskName" columns.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.SearchSettings(ss =>
{
ss.Fields(new List<string> { "TaskId", "TaskName" });
})
)
@(Html.EJ().ScriptManager())
```

It is possible to filter the TreeGrid contents at initial load using the toolbar search, with the SearchSettings.Key property.

The below code example explains filtering the TreeGrid contents at initial load with a search key, which will be applied across all the columns.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.SearchSettings(ss =>
{
ss.Key("task 1");
})
)
@(Html.EJ().ScriptManager())
```

TreeGrid provides support for conditional searching with operators in toolbar search using the property SearchSettings.Operator. And case sensitivity for the search query can be ignored using the property SearchSettings.IgnoreCase.

The below code example explains filtering the TreeGrid content using toolbar search with operators

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.SearchSettings(ss =>
{
ss.Fields(new List<string> { "TaskName" }).Key("Task
1").Operator(TreeGridSearchOperator.StartsWith).IgnoreCase(false);
})
)
@(Html.EJ().ScriptManager())
```

If the toolbar search textbox is not enabled in TreeGrid, and still if the contents need to be filtered at initial load using the SearchSettings property, then the user should enable the AllowSearching property along with search settings.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowSearching(true)
.SearchSettings(ss =>
{
ss.Key("Plan");
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

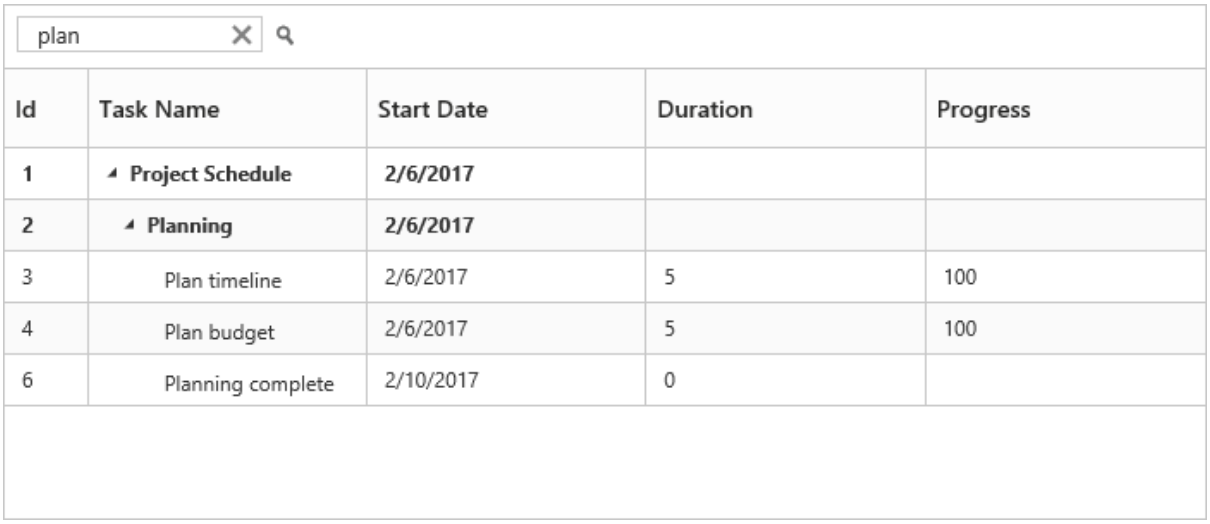
Searching

The TreeGrid control has an option to search its content using toolbar search box. The toolbar search box can be enabled by using the ToolbarSettings.ToolbarItems property. The following code example explains how to integrate search textbox in toolbar.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.Search,
});
})
)
```

The below screenshot shows TreeGrid search with plan key word.



Search Hierarchy Modes

The tree grid supports different types of search mode through the SearchSettings.SearchHierarchyMode property.

The following are the types of search modes available in the tree grid.

Parent: This is the default search hierarchy mode in the tree grid. It displays a searched record with its parent records. If the searched records do not have any parent record, it displays only the searched record.

Child: Displays the searched record with its child record. If the searched records do not have any child record, it displays only the searched record.

Both: Displays the searched record with both its parent and child records. If the searched records do not have any parent and child records, it displays only the searched record.

None: Displays only the searched record.

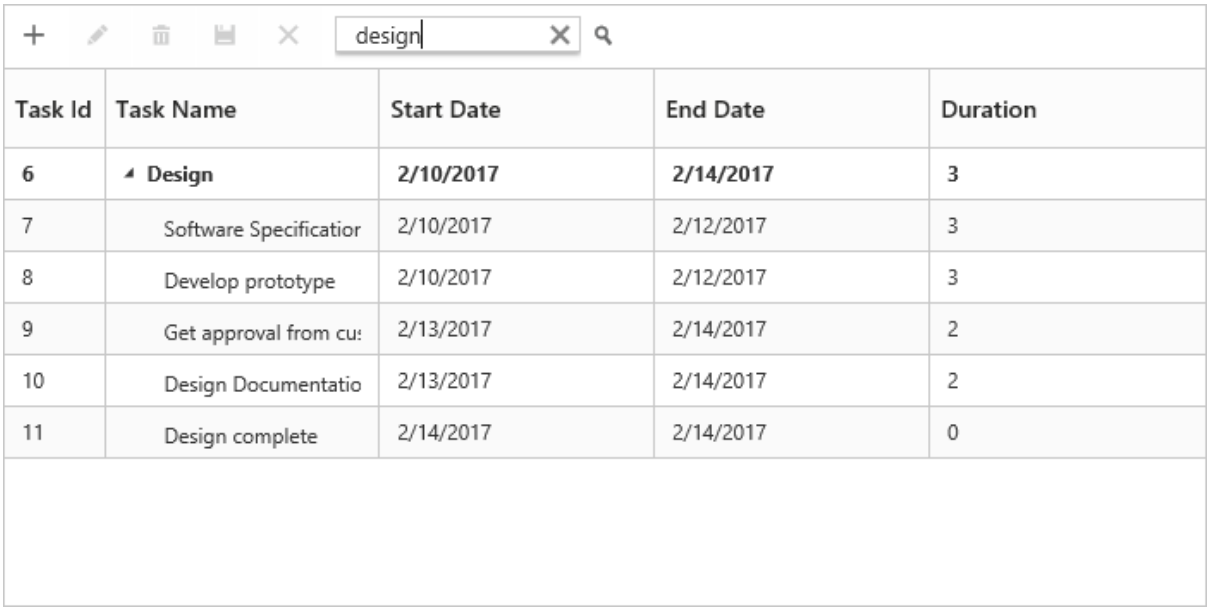
Copyright © 2001 - 2020 Syncfusion Inc.

The following code example shows how to set the SearchHierarchyMode in the tree grid.

HTML

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SearchSettings(se => {
se.SearchHierarchyMode(TreeGridSearchHierarchyMode.Child);
})
.Datasource(ViewBag.datasource)
)
```

The following image depicts the output of the previous code example.



The above screenshot shows Tree Grid with Child search mode.

Summary Row

Summary rows in TreeGrid are used to summarize every hierarchy with the set of predefined summary types using the column values. Using the SummaryRows property, user can define the summary rows in TreeGrid and the title for each summary row can be defined using the SummaryRows.Title property. And summary for the columns in TreeGrid can be defined by using the SummaryRows.SummaryColumns property. Summary rows are enabled by setting the ShowSummaryRow property as true. The total summary row is the overall summary row displayed for all the rows in the TreeGrid and its visibility can be defined by using the ShowTotalSummary property.

Defining summary columns

Using the **SummaryType** property, user can define the type of summary to be displayed in a column.

The **DataMember** property is used the map the field values which is used for summary calculations.

The **DisplayColumn** property is used to specify the column in which the summary to be displayed.

The **Prefix** and **Suffix** properties are used to define the text should be displayed along with the summary column value.

Copyright © 2001 - 2020 Syncfusion Inc.

The **Format** property is used for formatting the summary column value.

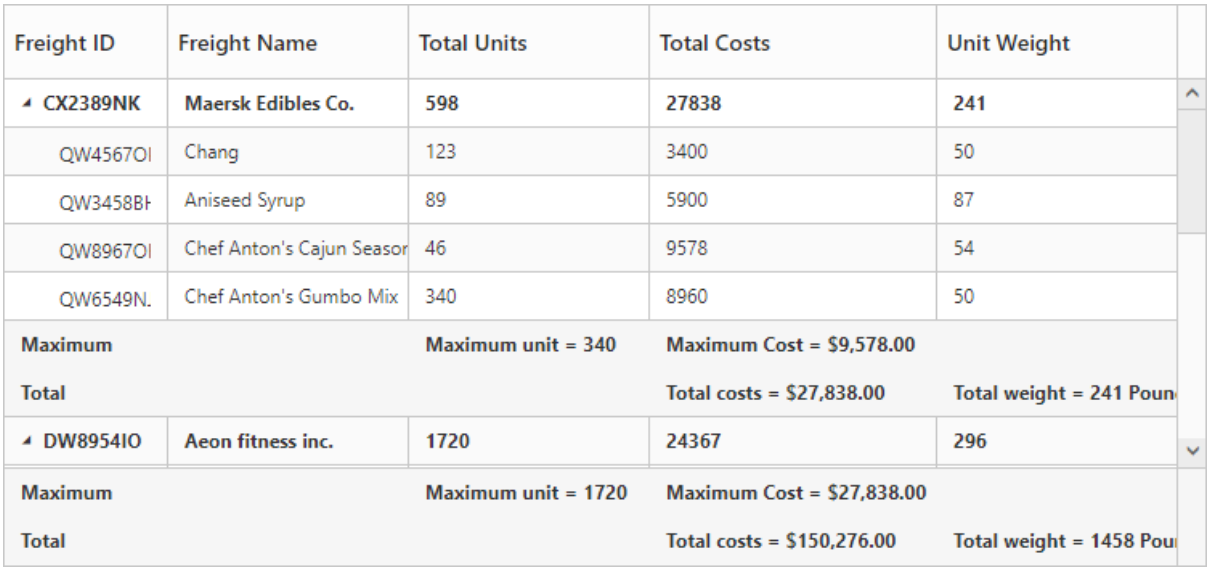
The below code snippet explains defining a summary row in TreeGrid,

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowSummaryRow(true)
.ShowTotalSummary(true)
.SummaryRows(rows =>
{
rows.Title("Maximum").SummaryColumns(col =>
{
col.SummaryType(TreeGridSummaryType.Maximum)
.DataMember("TotalUnits")
.Prefix("Maximum unit = ")
.DisplayColumn("TotalUnits").Add();
col.SummaryType(TreeGridSummaryType.Maximum)
.DataMember("TotalCosts")
.Prefix("Maximum Cost = ")
.DisplayColumn("TotalCosts")
.Format("{0:C}").Add();
}).Add();
rows.Title("Total").SummaryColumns(summary =>
{
summary.SummaryType(TreeGridSummaryType.Sum)
.DataMember("TotalCosts")
.Prefix("Total costs = ")
.DisplayColumn("TotalCosts")
.Format("{0:C}").Add();
summary.SummaryType(TreeGridSummaryType.Sum)
.DataMember("UnitWeight")
.Prefix("Total weight = ")
.DisplayColumn("UnitWeight")
.Suffix("Pounds").Add();
}).Add();
})
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Freight ID	Freight Name	Total Units	Total Costs	Unit Weight
CX2389NK	Maersk Edibles Co.	598	27838	241
QW4567OI	Chang	123	3400	50
QW3458BH	Aniseed Syrup	89	5900	87
QW8967OI	Chef Anton's Cajun Seasor	46	9578	54
QW6549N.	Chef Anton's Gumbo Mix	340	8960	50
Maximum		Maximum unit = 340	Maximum Cost = $9,578.00	
Total			Total costs = $27,838.00	Total weight = 241 Poun
DW8954IO	Aeon fitness inc.	1720	24367	296
Maximum		Maximum unit = 1720	Maximum Cost = $27,838.00	
Total			Total costs = $150,276.00	Total weight = 1458 Pou

Customize height of total summary

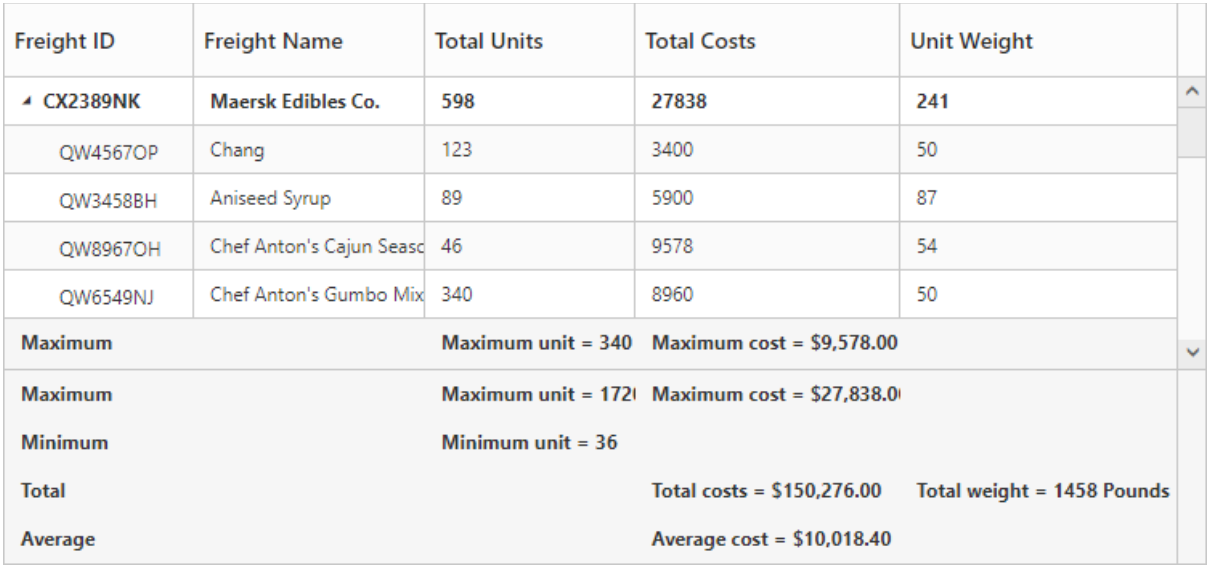
Using TotalSummaryHeight property we can customize the height of the total summary container.

The below code example shows how to update the footer summary container height.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowSummaryRow(true)
.ShowTotalSummary(true)
.TotalSummaryHeight(120)
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.



Freight ID	Freight Name	Total Units	Total Costs	Unit Weight
CX2389NK	Maersk Edibles Co.	598	27838	241
QW4567OP	Chang	123	3400	50
QW3458BH	Aniseed Syrup	89	5900	87
QW8967OH	Chef Anton's Cajun Seasc	46	9578	54
QW6549NJ	Chef Anton's Gumbo Mix	340	8960	50
Maximum		Maximum unit = 340	Maximum cost = $9,578.00	
Maximum		Maximum unit = 172	Maximum cost = $27,838.0	
Minimum		Minimum unit = 36		
Total			Total costs = $150,276.00	Total weight = 1458 Pounds
Average			Average cost = $10,018.40	

Expand/collapse total summary row

We can expand/collapse the total summary rows in TreeGrid using following methods.

Copyright © 2001 - 2020 Syncfusion Inc.

- Using Expander Icon
- Using Method

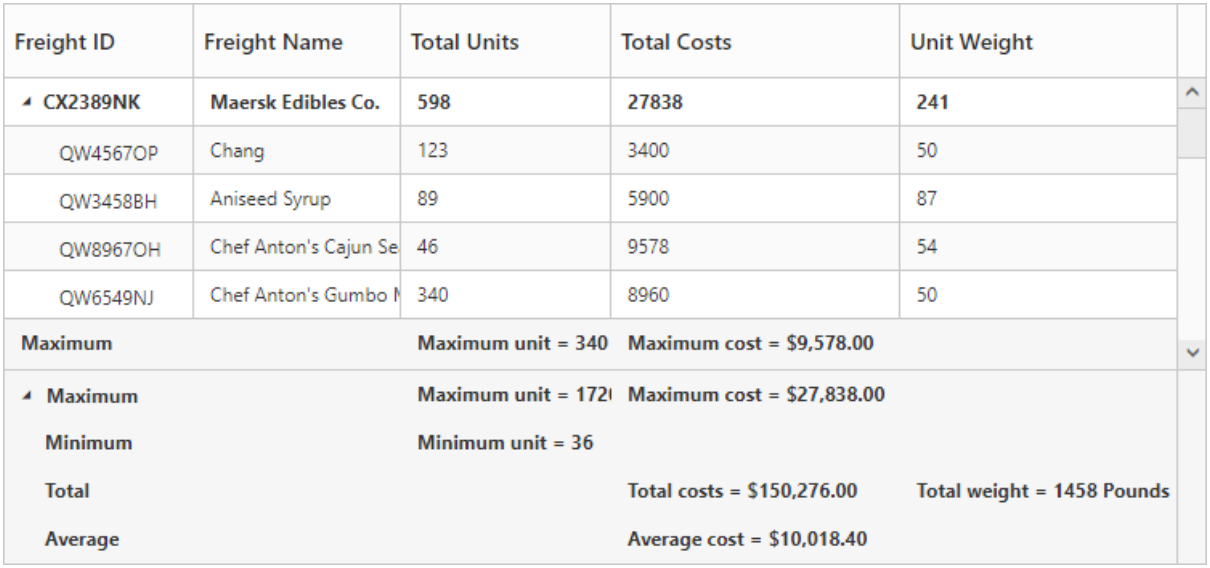
Using Expander Icon

We can enable expander icon in total summary row by using CollapsibleTotalSummary property. By default expander icon will be rendered in first row of 0th column in total summary rows.

Please find the below code example to enable collapsible total summary row in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowSummaryRow(true)
.ShowTotalSummary(true)
.TotalSummaryHeight(120)
.CollapsibleTotalSummary(true)
)
@(Html.EJ().ScriptManager())
```



Freight ID	Freight Name	Total Units	Total Costs	Unit Weight
CX2389NK	Maersk Edibles Co.	598	27838	241
QW4567OP	Chang	123	3400	50
QW3458BH	Aniseed Syrup	89	5900	87
QW8967OH	Chef Anton's Cajun Se	46	9578	54
QW6549NJ	Chef Anton's Gumbo M	340	8960	50
Maximum		Maximum unit = 340	Maximum cost = $9,578.00	
Maximum		Maximum unit = 172	Maximum cost = $27,838.00	
Minimum		Minimum unit = 36		
Total			Total costs = $150,276.00	Total weight = 1458 Pounds
Average			Average cost = $10,018.40	

Note: We can also customize the expander icon column in total summary row by using _summaryColumnIndex property and Load event.

Using Method

Total summary rows in TreeGrid can be expanded/collapsed by using expandCollapseTotalSummary method.

Please find the code example to collapse the total summary rows below.

ASPX-CS

```
<button onclick="expandCollapse()">expandCollapse</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowSummaryRow(true)
.ShowTotalSummary(true)
.TotalSummaryHeight(120)
.CollapsibleTotalSummary(true)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
function expandCollapse() {
var treeObj = $("#TreeGridContainer").data("ejTreeGrid");
treeObj.expandCollapseTotalSummary(false);
}
```

Click here to view the demo sample for summary row in TreeGrid.

Custom Summary

Custom summary can be used to create summary values based on your required custom logic and calculations. To enable the custom summary, the SummaryType should be set to 'Custom' and the CustomSummaryValue property should be defined as function. After the custom calculation, the returned value will be displayed in the corresponding summary cell.

HTML

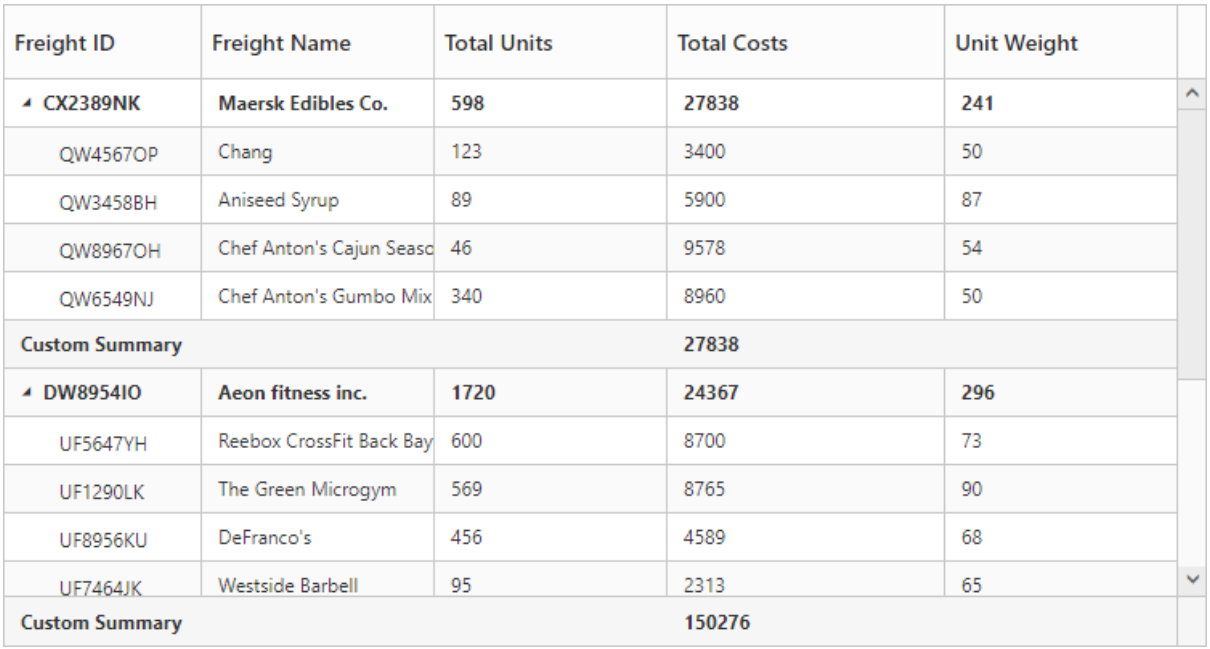
```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ShowSummaryRow(true)
.SummaryRows(su =>
{
su.Title("Custom Summary").
SummaryColumns(
summary =>
{
summary.SummaryType(TreeGridSummaryType.Custom)
.CustomSummaryValue("sum")
.DisplayColumn("Progress").Add();
}).Add();
})
)
```

JS

```
<script>
function sum(args, data) {
//ej.sum is aggregate to add data of total costs from datasource
return ej.sum(data, "TotalCosts");
}
</script>
```

The output of the tree grid with custom summary value is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Freight ID	Freight Name	Total Units	Total Costs	Unit Weight
CX2389NK	**Maersk Edibles Co.**	**598**	**27838**	**241**
QW4567OP	Chang	123	3400	50
QW3458BH	Aniseed Syrup	89	5900	87
QW8967OH	Chef Anton's Cajun Seaso	46	9578	54
QW6549NJ	Chef Anton's Gumbo Mix	340	8960	50
Custom Summary			**27838**	
DW8954IO	**Aeon fitness inc.**	**1720**	**24367**	**296**
UF5647YH	Reebox CrossFit Back Bay	600	8700	73
UF1290LK	The Green Microgym	569	8765	90
UF8956KU	DeFranco's	456	4589	68
UF7464JK	Westside Barbell	95	2313	65
Custom Summary			**150276**	

Responsive

The TreeGrid control has support for responsive behavior based on client browser's width and height. To enable responsive support in TreeGrid, IsResponsive property must be enabled.

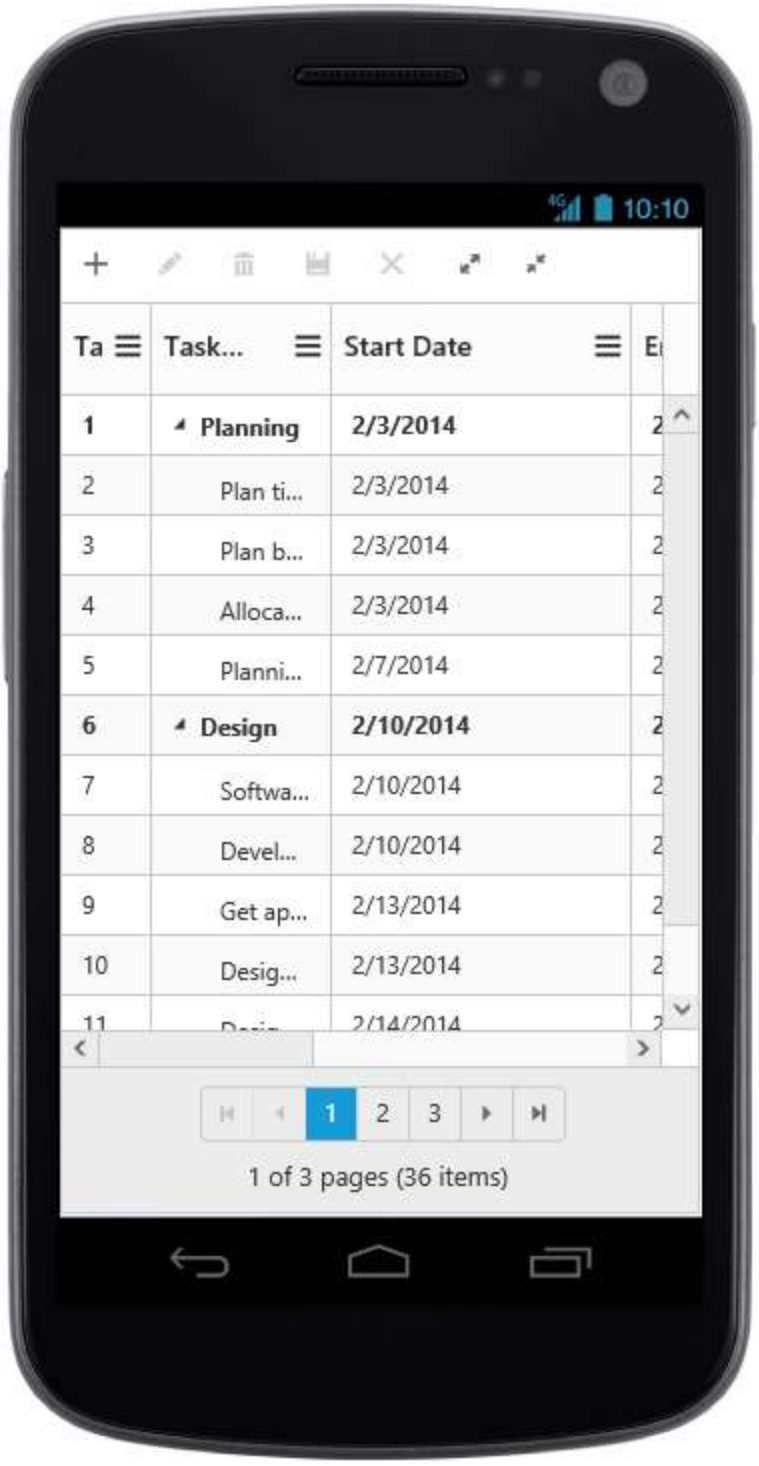
Please find the example describes the above behavior.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.IsResponsive(true)
)
@(Html.EJ().ScriptManager())
```

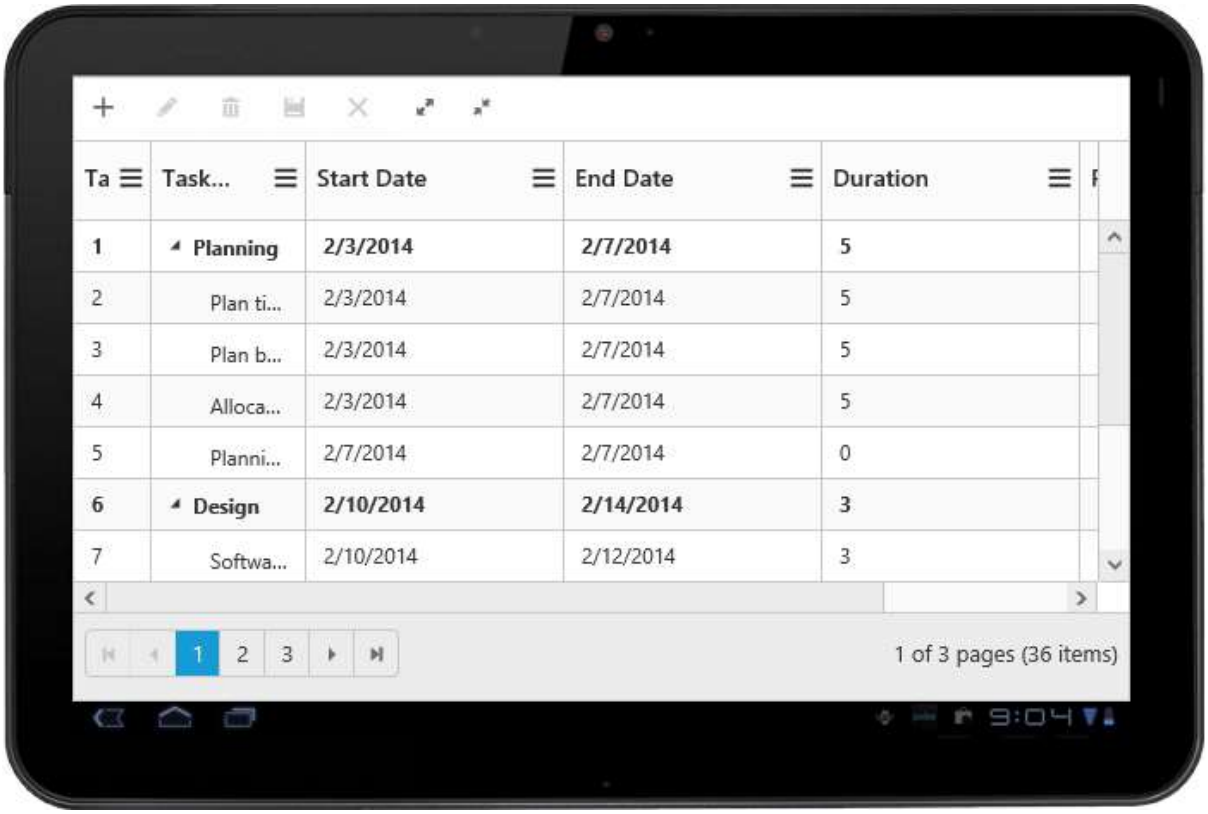
The following output is displayed as a result of the above code example

Copyright © 2001 - 2020 Syncfusion Inc.



Responsive TreeGrid in Mobile layout.

Copyright © 2001 - 2020 Syncfusion Inc.



Responsive TreeGrid in tablet layout.

Layout

Below are the modes of responsive layout available in TreeGrid based on client width,

- Mobile(<321px)
- Tablet(321px to 768px)
- Desktop(>768px)

Mobile Layout

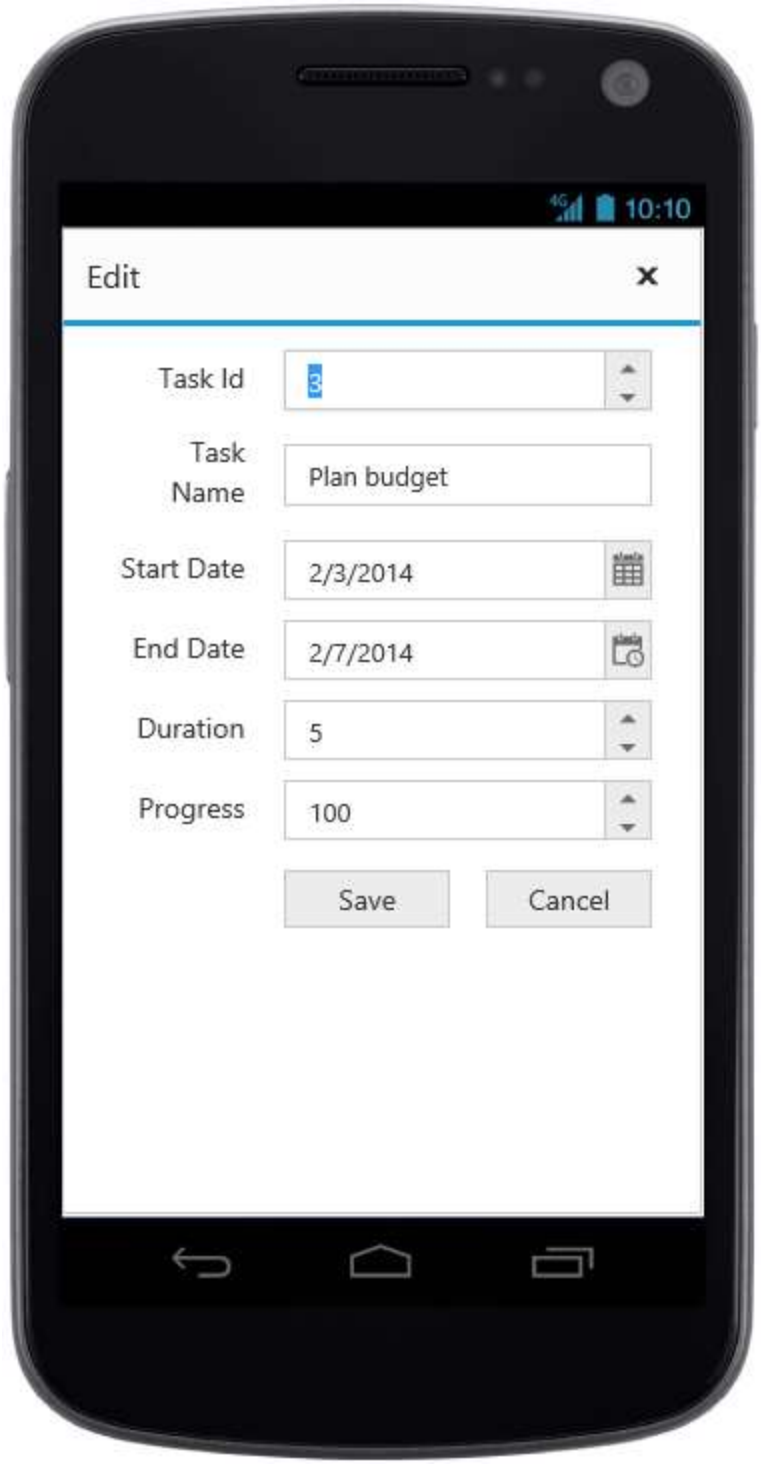
The customized layout for filtering, column chooser operations and add/edit operations in mobile device can be seen in the following screen shots.

Copyright © 2001 - 2020 Syncfusion Inc.



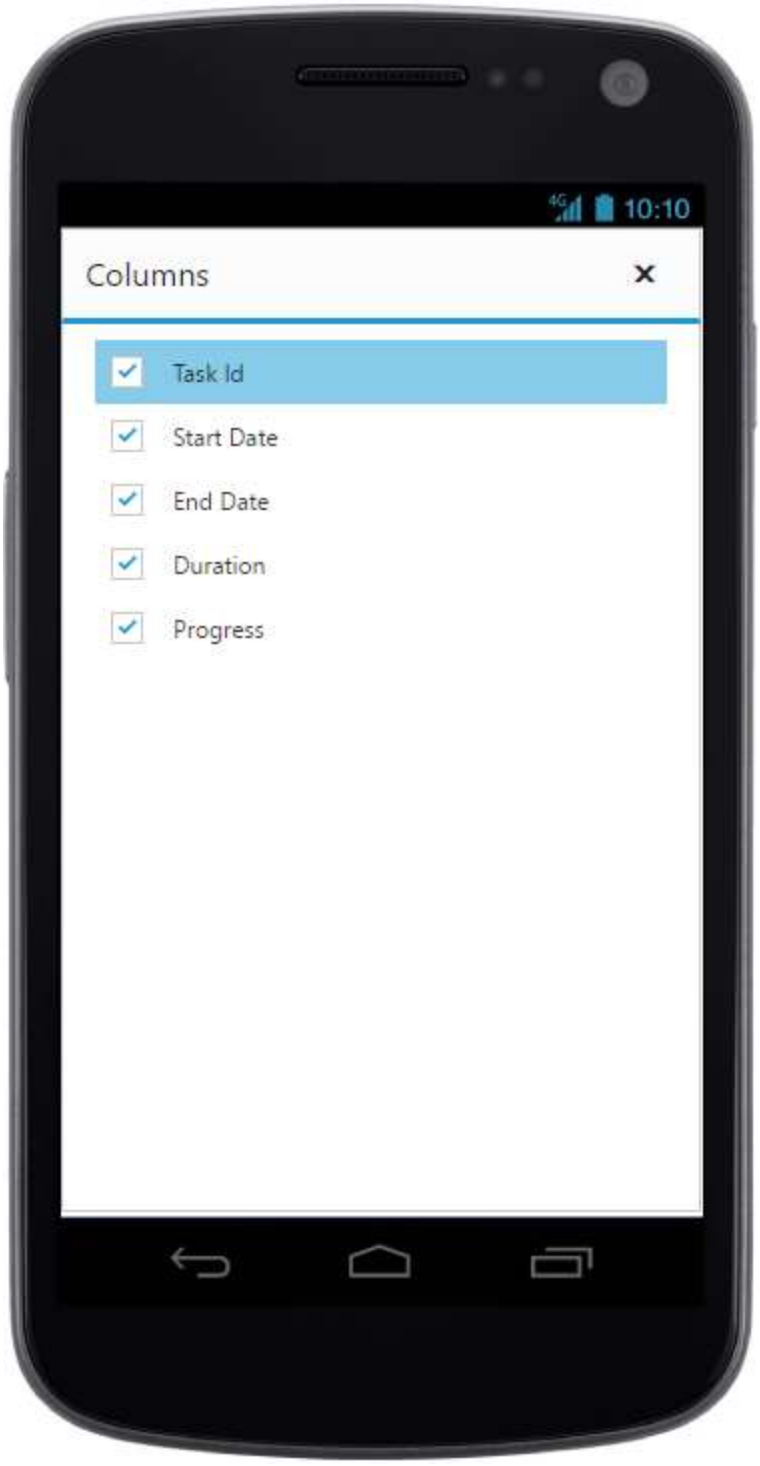
filtering in mobile layout.

Copyright © 2001 - 2020 Syncfusion Inc.



Add/edit in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.



Column chooser in mobile layout

Copyright © 2001 - 2020 Syncfusion Inc.

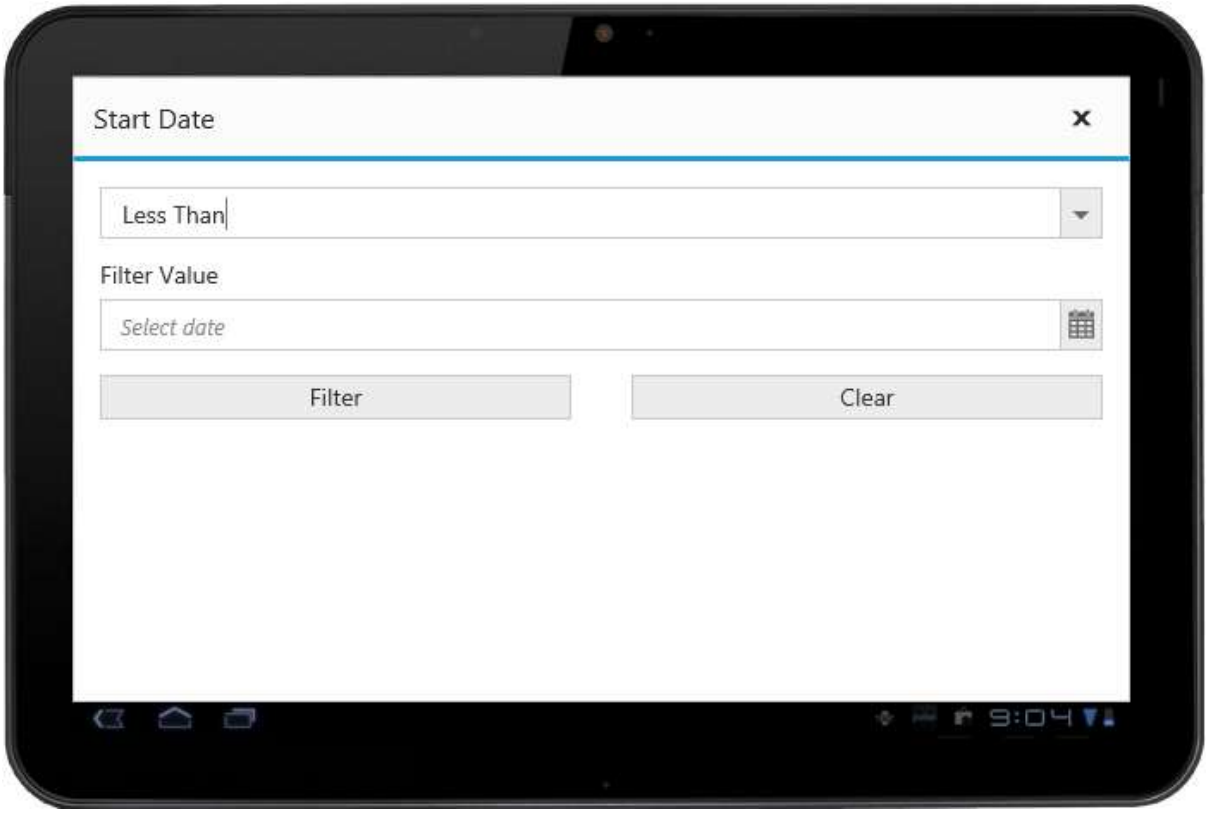


Insert column options in mobile layout

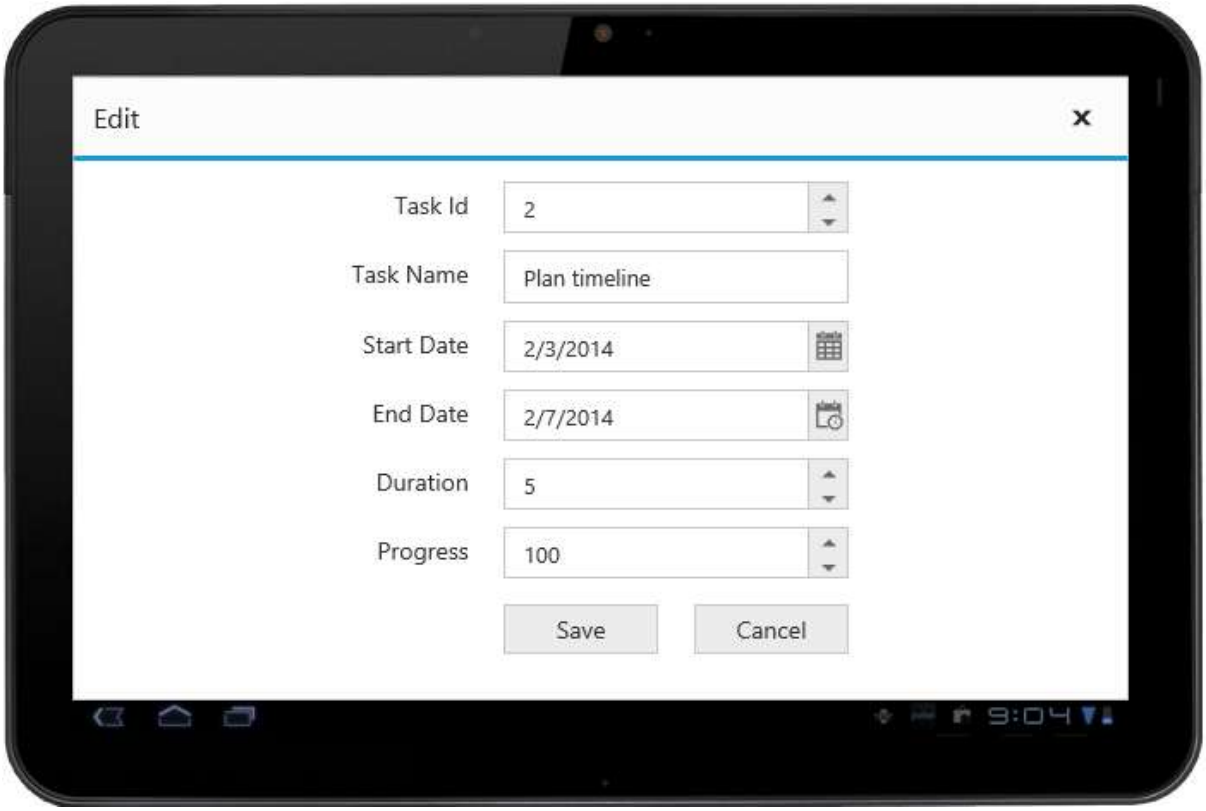
Tablet Layout

The customized layout for filtering, column chooser operations and add/edit operations in tablet device can be seen in the following screen shots.

Copyright © 2001 - 2020 Syncfusion Inc.

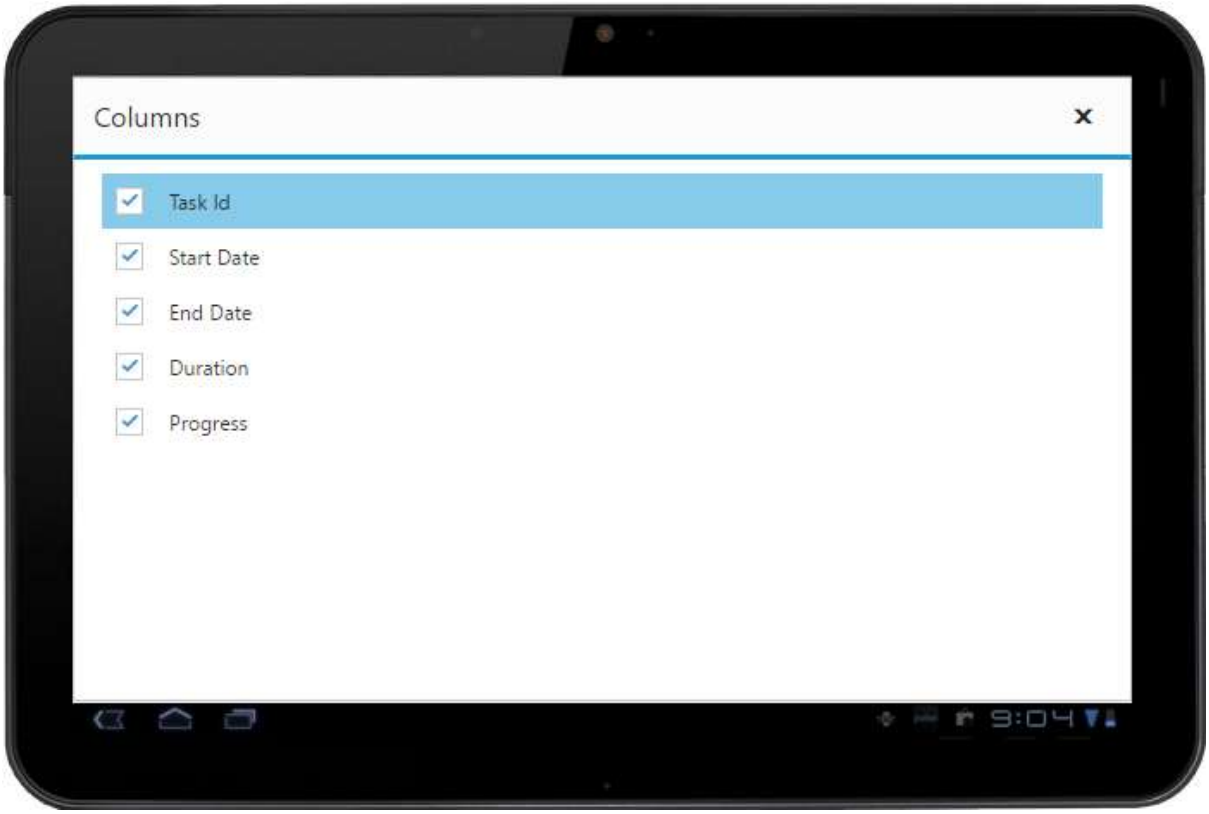


filtering in tablet layout.



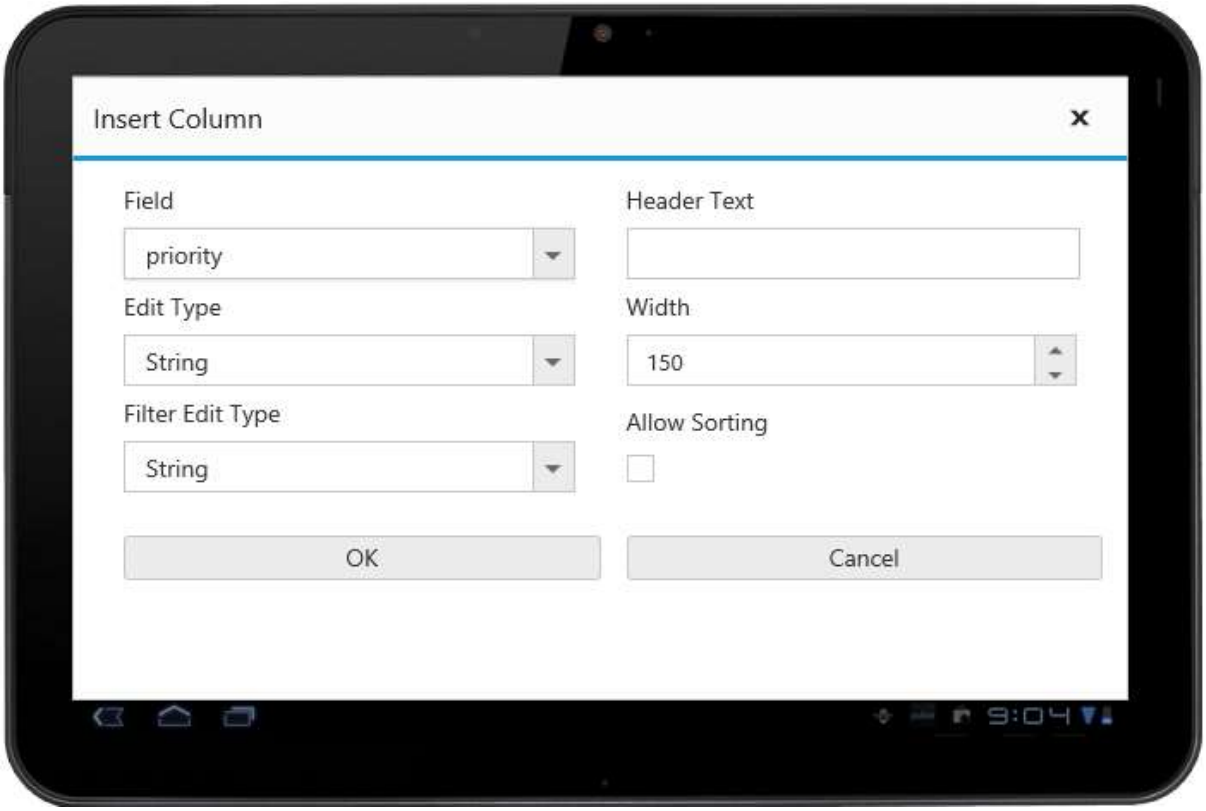
Copyright © 2001 - 2020 Syncfusion Inc.

Add/edit in tablet layout



Column chooser in tablet layout

Copyright © 2001 - 2020 Syncfusion Inc.



Insert column options in tablet layout

Changing responsive width using method

You can change the minimum responsive width dynamically by using public method updateResponsiveMinWidth(width) by passing the width as an argument.

The TreeGrid control get works in responsive mode only when the window width is below the minimum responsive width.

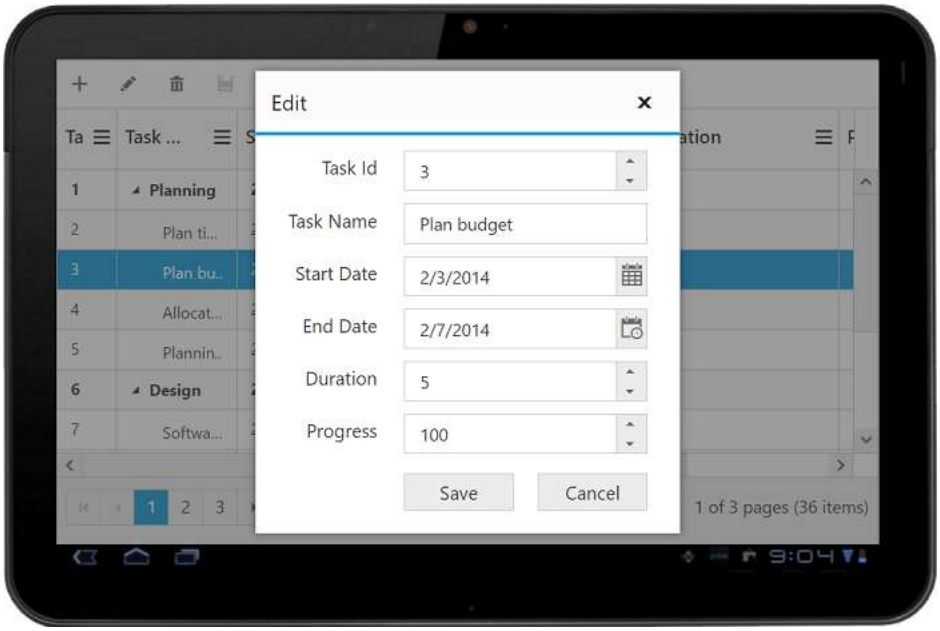
Please find the example describes the above behavior.

ASPX-CS

```
<button id="minResponsiveWidth">minResponsiveWidth</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
.IsResponsive(true)
)
@(Html.EJ().ScriptManager())
$("#minResponsiveWidth").click(function (args) {
treegridObj = $("#TreeGridContainer ").data("ejTreeGrid");
treegridObj.updateResponsiveMinWidth(600);
});
```

The following output is displayed as a result of the above code example

Copyright © 2001 - 2020 Syncfusion Inc.



Priority for columns

Column priorities are used to show or hide the columns based on priority values and browser width to best accommodate the possible columns. Priority value of the column is set by Columns.Priority property and the value ranges from one to six.

The following code example explains how to set priority for column

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.IsResponsive(true)
.Columns(co =>
{
co.Field("TaskId").HeaderText("Task Id").Width(45).Add();
co.Field("TaskName").HeaderText("Task Name").Add();
co.Field("StartDate").HeaderText("Start Date").Add();
co.Field("EndDate").HeaderText("End Date").Priority(5).Add();
co.Field("Duration").HeaderText("Duration").Priority(6).Add();
co.Field("Progress").HeaderText("Progress").Priority(6).Add();
})
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Task Id	Task Name	Start Date
1	**Planning**	**2/3/2014**
2	Plan timeline	2/3/2014
3	Plan budget	2/3/2014
4	Allocate resources	2/3/2014
5	Planning complete	2/7/2014
6	**Design**	**2/10/2014**
7	Software Specification	2/10/2014
8	Develop prototype	2/10/2014
9	Get approval from customer	2/13/2014
10	Design Documentation	2/13/2014
11	Design complete	2/14/2014
12	**Implementation Phase**	**2/17/2014**

The above screen shot show TreeGrid rendered in browser width of 640 pixels.

Selection

The TreeGrid control provides support for row and cell selections using SelectionSettings property.

Row selection

You can enable or disable the row selection in TreeGrid, using the AllowSelection property. By default row selection is enabled in TreeGrid.

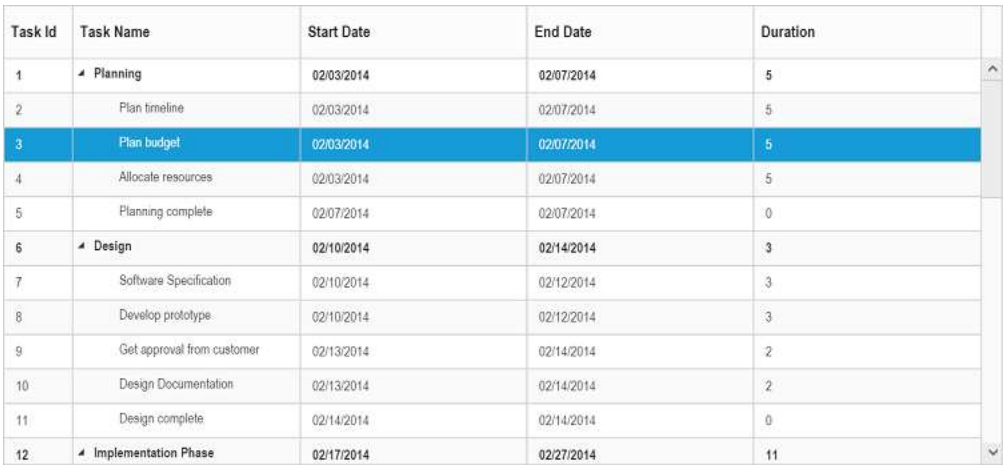
The following code example shows, how to disable the row selection in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.AllowSelection(true)
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with row selection is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2
10	Design Documentation	02/13/2014	02/14/2014	2
11	Design complete	02/14/2014	02/14/2014	0
12	Implementation Phase	02/17/2014	02/27/2014	11

Selecting a row at initial load

You can select a row at initial load by setting the index of the row to the SelectedRowIndex property.

Find the following code for selecting a row at initial load.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SelectedRowIndex(3)
)
@(Html.EJ().ScriptManager())
```

Multiple row selection

It is also possible to select multiple rows by setting the SelectionType as Multiple. You can select more than one row by holding down **CTRL** key and to click on the rows.

The following code example explains how to enable multiple selection in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Row);
ss.SelectionType(TreeGridSelectionType.Multiple);
}))
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with multiple row selection is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2
10	Design Documentation	02/13/2014	02/14/2014	2
11	Design complete	02/14/2014	02/14/2014	0
12	Implementation Phase	02/17/2014	02/27/2014	11

Selecting a row programmatically

You can select a row programmatically by setting the row index value to the SelectedRowIndex property.

The following code shows on how to select a row programmatically with button click action.

ASPX-CS

```
<button id="selectRow">SelectRow</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
)
@(Html.EJ().ScriptManager())
$("#selectRow").click(function (args) {
$("#TreeGridContainer").ejTreeGrid("option", "selectedRowIndex", 4);
})
```

Customize row selection action

While selecting a row in TreeGrid, RowSelecting and RowSelected event will be triggered. Row selecting event will be triggered on initialization of row selection action. In RowSelecting event we can get the previously selected row and current selecting row's information, using this information we can prevent selection of particular row. The RowSelected event will be triggered on completion of row selection action, in this event we can get the current selected row's information.

The following code example shows how to prevent the selection of particular row using RowSelecting event.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.ClientSideEvents(cs =>
{
cs.RowSelecting("rowSelecting");
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function rowSelecting(args) {
if(args.data.taskID == 5) // prevent selection of Task id 5
args.cancel = true;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
</script>
```

Get record details

In TreeGrid, It is possible to get the record detail when **click** and **dblClick** the row using RecordClick and RecordDoubleClick event, using this method we can get the record details even AllowSelection is false.

The below code example show, how to get record details when **click** the row.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.ClientSideEvents(cs =>
{
cs.RecordClick("recordClick");
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function recordClick(args) {
// Perform action for recordClick
}
</script>
```

Cell selection

You can select cells in TreeGrid by setting the SelectionMode property as Cell.

And you can able to get the selected cell information using the SelectedCellIndexes property from the TreeGrid object.

The SelectedCellIndexes is an object collection, which has the CellIndex and RowIndex information of the selected cells.

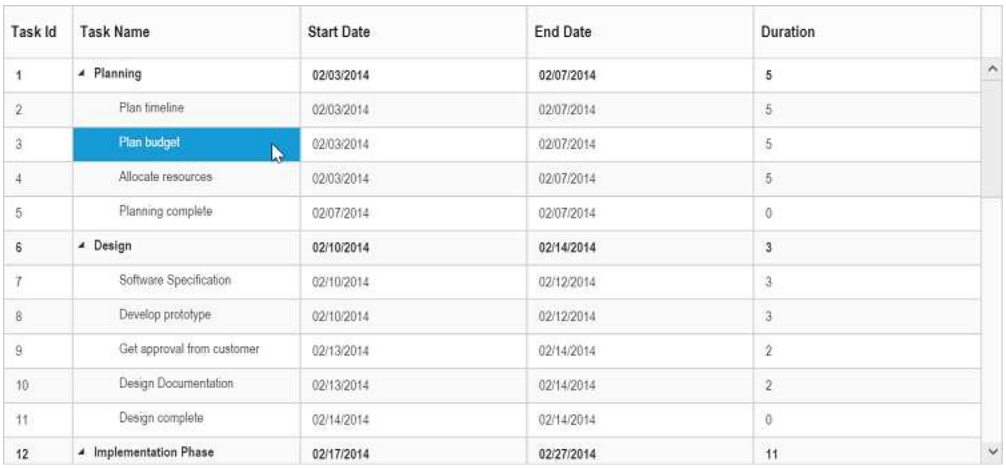
Find the code example below to enable the cell selection in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Cell);
}))
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with cell selection is as follows.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2
10	Design Documentation	02/13/2014	02/14/2014	2
11	Design complete	02/14/2014	02/14/2014	0
12	Implementation Phase	02/17/2014	02/27/2014	11

It is possible to get the list of HTML elements of the selected cells at run-time using the getSelectedCells method.

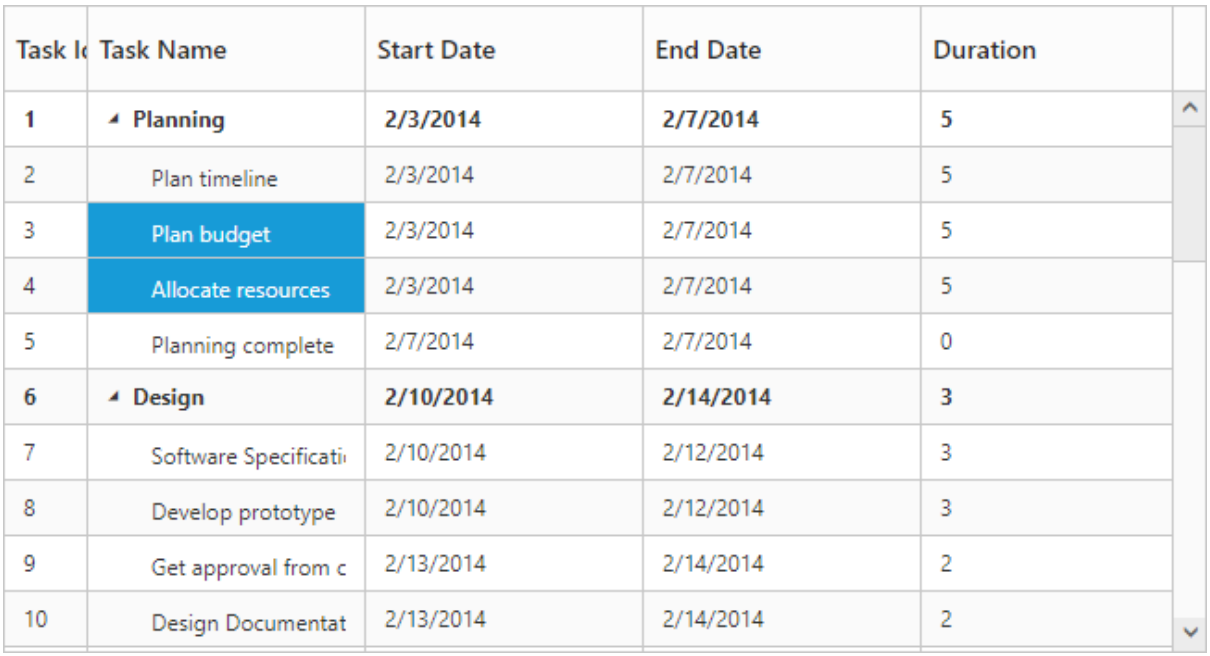
Select cells dynamically

You can select the cells programmatically using the selectCells public method. Find the code example below for details.

ASPX-CS

```
<button id="selectCells">Select Cells</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Cell);
ss.SelectionType(TreeGridSelectionType.Multiple);
}))
@(Html.EJ().ScriptManager())
$("#selectCells").click(function () {
var treegridObj = $("#TreeGridContainer").ejTreeGrid("instance");
var cellIndex = [{
rowIndex: 2,
cellIndex: 1
}, {
rowIndex: 3,
cellIndex: 1
}];
treegridObj.selectCells(cellIndex);
})
```

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	2/3/2014	2/7/2014	5
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	Design	2/10/2014	2/14/2014	3
7	Software Specificati	2/10/2014	2/12/2014	3
8	Develop prototype	2/10/2014	2/12/2014	3
9	Get approval from c	2/13/2014	2/14/2014	2
10	Design Documentat	2/13/2014	2/14/2014	2

Disabling cell selection for specific column

It is possible to disable cell selection for a specific column by setting `AllowCellSelection` as `false` in the column definition.

The below code snippet explains how to disable cell selection for specific column in TreeGrid

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.Columns(co =>
{
//..
co.Field("TaskId").AllowCellSelection(false).Add();
}))
@(Html.EJ().ScriptManager())
```

Multiple cell selection

You can also select multiple cell by setting the `SelectionType` property as `Multiple`.

Multiple selection can be done by holding the **CTRL** key and to click the required cells.

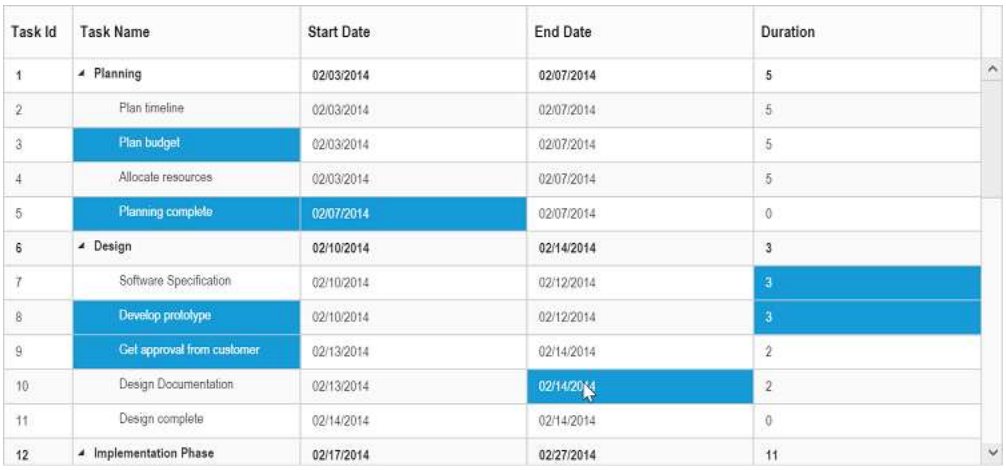
The following code example shows you to select multiple cells.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Cell);
ss.SelectionType(TreeGridSelectionType.Multiple);
}))
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The output of the TreeGrid with multiple cell selection is as follows.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2
10	Design Documentation	02/13/2014	02/14/2014	2
11	Design complete	02/14/2014	02/14/2014	0
12	Implementation Phase	02/17/2014	02/27/2014	11

Customize cell selection action

While selecting a cell in TreeGrid, CellSelecting and CellSelected event will be triggered. Cell selecting event will be triggered on initialization of cell selection action. In CellSelecting event we can get the current selecting cell information, using this information we can prevent selection of particular cell in particular row. The CellSelected event will be triggered on completion of cell selection action, in this event we can get the current selected cell's information.

The following code example shows how to prevent the selection of particular cell using CellSelecting event.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.ClientSideEvents(cs =>
{
cs.CellSelecting("cellSelecting");
}))
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function cellSelecting(args) {
if(args.data.taskID == 5 && args.cellIndex == 1) // prevent selection of
Task Name cell of Task id 5
args.cancel = true;
}
</script>
```

Checkbox selection

TreeGrid supports checkbox selection and to enable the checkbox selection, you need to set the SelectionType property to Checkbox and the SelectionMode property as Row. By default, checkbox column will be displayed as the left most column, on enabling the checkbox selection in TreeGrid.

Column header checkbox

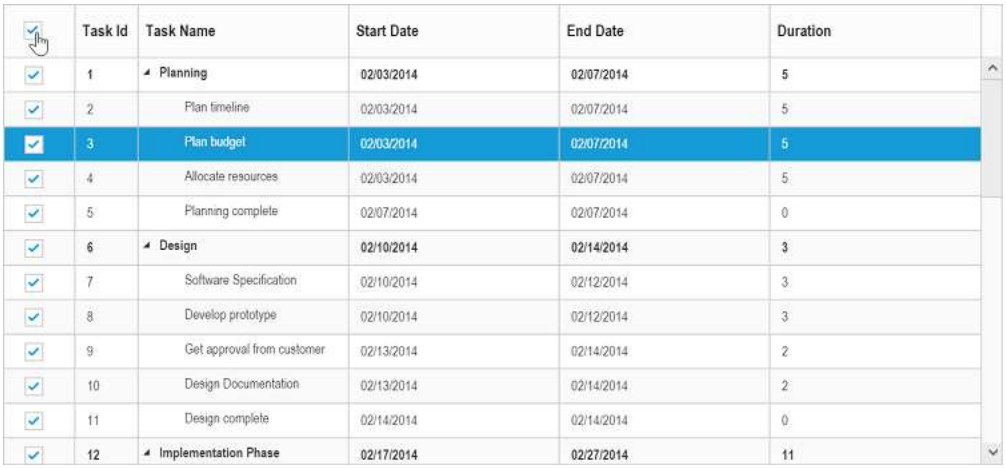
It is possible to select/unselect all the TreeGrid rows using column header checkbox. To enable this you need to set the EnableSelectAll property as true. The following code snippet explains how to enable the column header checkbox.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Row);
ss.SelectionType(TreeGridSelectionType.Checkbox);
ss.EnableSelectAll(true);
})
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with checkbox enabled in column header.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2
10	Design Documentation	02/13/2014	02/14/2014	2
11	Design complete	02/14/2014	02/14/2014	0
12	Implementation Phase	02/17/2014	02/27/2014	11

Hierarchy selection

It is possible to select the rows hierarchically using checkboxes in TreeGrid by enabling the EnableHierarchySelection property.

In this selection the hierarchy between the records will be retained, where the child records will get selected on selecting its parent record's checkbox and parent record checkbox will get selected on checking all of its child items.

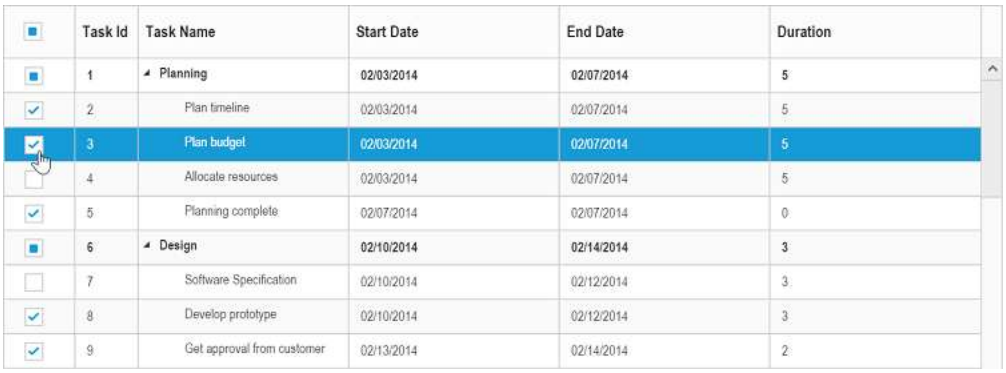
Following code snippet explains on enabling hierarchy selection in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Row);
ss.SelectionType(TreeGridSelectionType.Checkbox);
ss.EnableHierarchySelection(true);
})
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with hierarchy selection enabled.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2

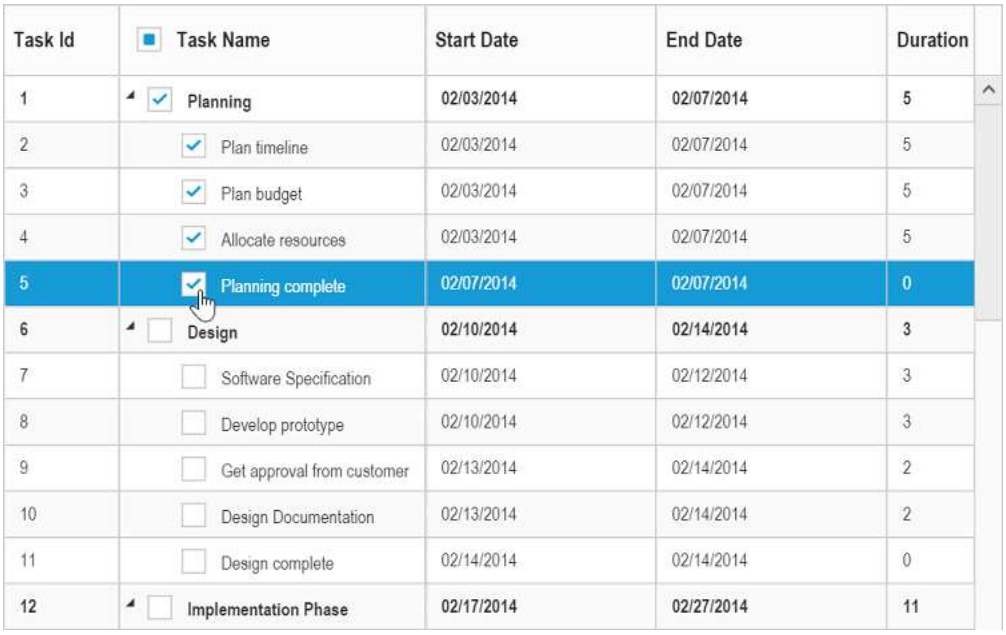
Checkbox column

It is possible to change the default index of the checkbox column and we can display the checkboxes in any of the existing column. And to enable the checkbox in any of the column, we need to set `Columns.ShowCheckbox` property as true in the column object.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//..
.SelectionSettings(ss=>
{
ss.SelectionMode(TreeGridSelectionMode.Row);
ss.SelectionType(TreeGridSelectionType.Checkbox);
})
.Columns(co =>
{
co.Field("taskName").HeaderText("Task Name").ShowCheckbox(true).Add();
})
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with checkbox enabled in task name column.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from customer	02/13/2014	02/14/2014	2
10	Design Documentation	02/13/2014	02/14/2014	2
11	Design complete	02/14/2014	02/14/2014	0
12	Implementation Phase	02/17/2014	02/27/2014	11

Copyright © 2001 - 2020 Syncfusion Inc.

The index of the checkbox column can be changed at run-time using the updateCheckboxColumn method. The index of the column in which the checkbox should be displayed is passed as the method parameter.

MultiSelection – Touch Option

It is possible to select cells using touch action in TreeGrid. TreeGrid provides support for both single selection and multiple cell selection using touch action. For multiple cell selection, when we tap on a cell a helper icon will be displayed using that we can select multiple cells.

The following code example describes how to enable multiple selection in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.SelectionSettings(ss=>
{
ss.SelectionType(TreeGridSelectionType.Multiple);
}))
@(Html.EJ().ScriptManager())
```

The following output is displayed the result of multiple selection in touch device environment.

Task Id	Task Name	Start Date	End Date	Duration	Progress
1	Planning	2/3/201	2/7/2014	5	100
2	Plan timeline	2/3/201	2/7/2014	5	100
3	Plan budget	2/3/2014	2/7/2014	5	100
4	Allocate resour...	2/3/2014	2/7/2014	5	100
5	Planning comp...	2/7/2014	2/7/2014	0	0
6	Design	2/10/2014	2/14/2014	3	86
7	Software Speci...	2/10/2014	2/12/2014	3	60
8	Develop protot...	2/10/2014	2/12/2014	3	100
9	Get approval fr...	2/13/2014	2/14/2014	2	100
10	Design Docum...	2/13/2014	2/14/2014	2	100
11	Design comple...	2/14/2014	2/14/2014	0	0
12	Implementation..	2/17/2014	2/27/2014	11	

Clear selection using method

It is possible to clear the selection in TreeGrid at run-time using the clearSelection method.

The specific row will be deselected when the row index is passed as the method parameter. If the index is not passed, then all the selected rows in TreeGrid will be deselected.

ASPX-CS

```
<button id="clearSelection">Clear Selection</button>
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@(Html.EJ().ScriptManager())
$("#clearSelection").click(function () {
var treegridObj = $("#TreeGridContainer").data("ejTreeGrid");
treegridObj.clearSelection(2);
})
```

Export

Exporting feature provides support to export TreeGrid content to excel and PDF files. To export the contents, the ExcelExport and PdfExport toolbar items must be added in the toolbar using the ToolbarItems property of ToolbarSettings. When you click, the toolbar exporting icons, it internally invokes the export public method of TreeGrid object to export.

The below code snippet explains the above behavior,

C#

```
public class TreeGridController  : Controller
{
public ActionResult TreeGridExporting()
{
ViewBag.datasource = this.GetEditingDataSource();
return View();
}
public void ExportToPdf(string TreeGridModel)
{
PdfExport exp = new PdfExport();
var DataSource = this.GetEditingDataSource();
TreeGridProperties obj = ConvertTreeGridObject(TreeGridModel);
exp.Export(obj, DataSource, new TreeGridExportSettings() { Theme =
ExportTheme.FlatSaffron },"Export");
}
public void ExportToExcel(string TreeGridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = this.GetEditingDataSource();
TreeGridProperties obj = ConvertTreeGridObject(TreeGridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, new
TreeGridExportSettings() { Theme = ExportTheme.FlatSaffron});
}
private TreeGridProperties ConvertTreeGridObject(string gridProperty)
{
JavaScriptSerializer serializer = new JavaScriptSerializer();
IEnumerable div = (IEnumerable)serializer.Deserialize(gridProperty,
typeof(IEnumerable));
TreeGridProperties gridProp = new TreeGridProperties();
foreach (KeyValuePair<string, object> ds in div)
{
var property = gridProp.GetType().GetProperty(ds.Key,
BindingFlags.Instance | BindingFlags.Public | BindingFlags.IgnoreCase);
if (property != null)
{
Type type = property.PropertyType;
string serialize = serializer.Serialize(ds.Value);
object value = serializer.Deserialize(serialize, type);
property.SetValue(gridProp, value, null);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
}
return gridProp;
}
}
```

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.PdfExport,
TreeGridToolBarItems.ExcelExport
});
})
.Datasource(ViewBag.datasource)
)
@(Html.EJ().ScriptManager())
```

The below screen shot shows TreeGrid with excel and PDF exporting enabled.

Task Id	Task Name	Start Date	End Date	Duration
1	Planning	02/03/2014	02/07/2014	5
2	Plan timeline	02/03/2014	02/07/2014	5
3	Plan budget	02/03/2014	02/07/2014	5
4	Allocate resources	02/03/2014	02/07/2014	5
5	Planning complete	02/07/2014	02/07/2014	0
6	Design	02/10/2014	02/14/2014	3
7	Software Specification	02/10/2014	02/12/2014	3
8	Develop prototype	02/10/2014	02/12/2014	3
9	Get approval from custor	02/13/2014	02/14/2014	2

Server dependencies

Export Helper functions are available in the Assembly Syncfusion.EJ.Export, which is available in the Essential Studio & Essential ASP.NET MVC builds. The list of assemblies needed for TreeGrid Export as follows

- Syncfusion.EJ
- Syncfusion.EJ.Export
- Syncfusion.Linq.Base
- Syncfusion.Compression.Base
- Syncfusion.XlsIO.Base

Copyright © 2001 - 2020 Syncfusion Inc.

- Syncfusion.PDF.Base

Supported Export Types

Currently server helper function allows following two types of exporting.

- Excel
- PDF

Export Mapper in MVC

Mappers is used to change the default routing path for exporting. By using Mappers you can change any action name in controller and the action can be in any controller (Need not to be in TreeGrid View Page Controller).

C#

```
public  class TreeGridController : Controller
{
public void PdfAction(string TreeGridModel)
{
PdfExport exp = new PdfExport();
var DataSource = this.GetEditingDataSource();
TreeGridProperties obj = ConvertTreeGridObject(TreeGridModel);
exp.Export(obj, DataSource, new TreeGridExportSettings() { Theme =
ExportTheme.FlatSaffron },"Export");
}
public void ExcelAction(string TreeGridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = this.GetEditingDataSource();
TreeGridProperties obj = ConvertTreeGridObject(TreeGridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, new
TreeGridExportSettings() { Theme = ExportTheme.FlatSaffron});
}
}
```

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.Mappers(map =>
map.ExportToPdfAction("PdfAction")).ExportToExcelAction("ExcelAction"))
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.PdfExport,
TreeGridToolBarItems.ExcelExport
});
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Click here to view the online demo sample for Exporting.

Multiple exporting

Multiple export is used for export more than one TreeGrid object in the same file. Multiple export in TreeGrid can be enabled by setting AllowMultipleExporting property to true. TreeGrid properties of all the TreeGrid which are available in current page are passed as string array parameter to controller action method.

The following code example describes exporting multiple TreeGrid in PDF format

C#

```
public class TreeGridController : Controller
{
public ActionResult TreeGridMultipleExporting()
{
ViewBag.datasource1 = this.GetPlanDataSource();
ViewBag.datasource2 = this.GetDesignDataSource();
ViewBag.datasource3 = this.GetImplementationDataSource();
return View();
}
public void MultipleExportToPdf(string[] TreeGridModel)
{
PdfExport exp = new PdfExport();
var PlanData = this.GetPlanDataSource();
var DesignData = this.GetDesignDataSource();
var ImplementationData = this.GetImplementationDataSource();
PdfDocument document = null;
int count = 1;
foreach (string gridProperty in TreeGridModel)
{
TreeGridProperties gridProp = this.ConvertTreeGridObject(gridProperty);
if (count == 1)
{
document = exp.Export(gridProp, PlanData, new TreeGridExportSettings() {
Theme = ExportTheme.FlatSaffron }, false, "Planning Phase");
}
else if(count == 2)
{
exp.Export(gridProp, DesignData, new TreeGridExportSettings() { Theme =
ExportTheme.FlatSaffron }, "Export", document, false, "Design Phase");
}
else
{
exp.Export(gridProp, ImplementationData, new TreeGridExportSettings() {
Theme = ExportTheme.FlatSaffron }, "Export", document, true,
"Implementation Phase");
}
count++;
}
}
public void MultipleExportToExcel(string[] TreeGridModel)
{
ExcelExport exp = new ExcelExport();
IWorkbook  workbook = null;
int count = 1;
foreach (string gridProperty in TreeGridModel)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
TreeGridProperties gridProp = this.ConvertTreeGridObject(gridProperty);
if (count == 1)
{
workbook = exp.Export(gridProp, this.GetPlanDataSource(), new
TreeGridExportSettings() { Theme = ExportTheme.FlatSaffron }, true,
"Planning Phase");
}
else if (count == 2)
{
exp.Export(gridProp, this.GetDesignDataSource(), new
TreeGridExportSettings() { Theme = ExportTheme.FlatSaffron }, workbook,
true, "Design Phase");
}
else
{
exp.Export(gridProp, this.GetImplementationDataSource(),
"ExcelExport.xlsx", ExcelVersion.Excel2010, new TreeGridExportSettings()
{ Theme = ExportTheme.FlatSaffron }, workbook, false, "Implementation
Phase");
}
count++;
}
}
}
```

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.AllowMultipleExporting(true)
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.PdfExport,
TreeGridToolBarItems.ExcelExport
});
})
.Mappers(map =>
map.ExportToPdfAction("MultipleExportToPdf").ExportToExcelAction("Multipl
eExportToExcel"))
.Datasource(ViewBag.datasource1)
)
@(Html.EJ().TreeGrid("TreeGridContainer1")
//...
.AllowMultipleExporting(true)
.Datasource(ViewBag.datasource2)
)
@(Html.EJ().TreeGrid("TreeGridContainer2")
//...
.AllowMultipleExporting(true)
.Datasource(ViewBag.datasource3)
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Click here to view the online demo sample for Multiple Exporting.

Export Theme

The TreeGrid export supports the below themes,

- flat-azure
- flat-azure-dark
- flat-lime
- flat-lime-dark
- flat-saffron
- flat-saffron-dark
- gradient-azure
- gradient-azure-dark
- gradient-lime
- gradient-lime-dark
- gradient-saffron
- gradient-saffron-dark
- bootstrap-theme

The desired theme should be passed as a parameter to the Export method and the code snippet for this as follows

C#

```
public void ExcelAction(string TreeGridModel)
{
ExcelExport exp = new ExcelExport();
var DataSource = this.GetEditingDataSource();
TreeGridProperties obj = ConvertTreeGridObject(TreeGridModel);
exp.Export(obj, DataSource, "Export.xlsx", ExcelVersion.Excel2010, new
TreeGridExportSettings() { Theme = ExportTheme.FlatSaffron});
}
```

Paging

TreeGrid provides support for displaying records in paginated view. Paging can be enabled in TreeGrid by setting the `AllowPaging` property as `true`.

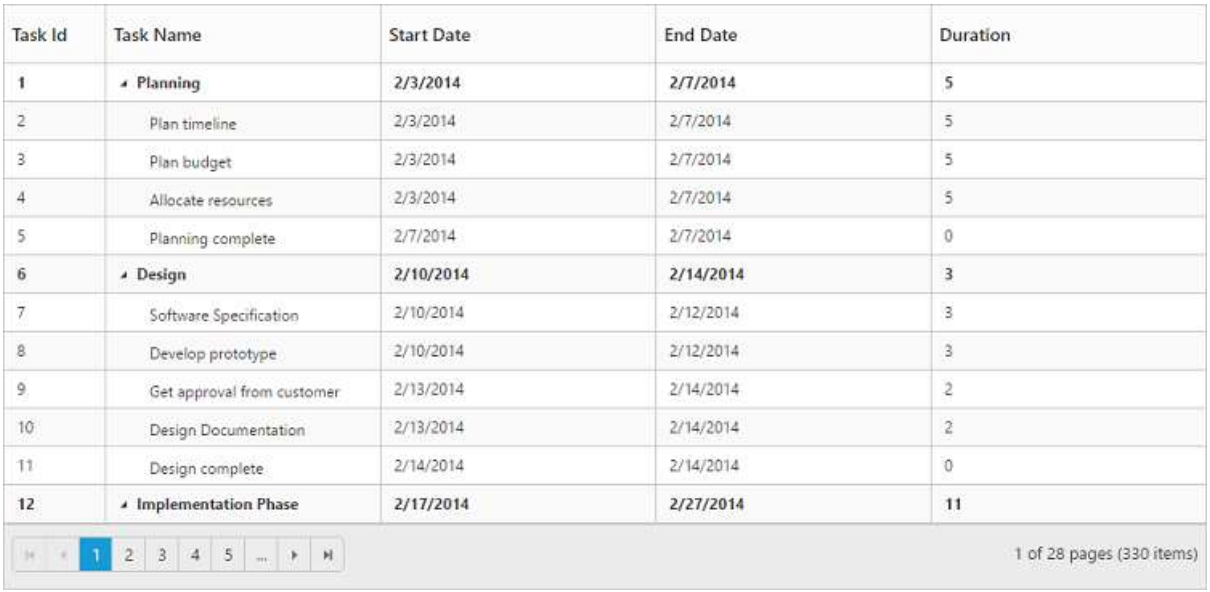
The below code snippet explains enable paging in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowPaging(true)
)
@(Html.EJ().ScriptManager())
```

The output of the TreeGrid with paging enabled is displayed below.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration
1	Planning	2/3/2014	2/7/2014	5
2	Plan timeline	2/3/2014	2/7/2014	5
3	Plan budget	2/3/2014	2/7/2014	5
4	Allocate resources	2/3/2014	2/7/2014	5
5	Planning complete	2/7/2014	2/7/2014	0
6	Design	2/10/2014	2/14/2014	3
7	Software Specification	2/10/2014	2/12/2014	3
8	Develop prototype	2/10/2014	2/12/2014	3
9	Get approval from customer	2/13/2014	2/14/2014	2
10	Design Documentation	2/13/2014	2/14/2014	2
11	Design complete	2/14/2014	2/14/2014	0
12	Implementation Phase	2/17/2014	2/27/2014	11

Paging settings

The paging in TreeGrid can be customized by using the PageSettings property.

- **PageSize** - Using this property we can limit the number of records to be displayed per page.
- **PageSizeMode** - By setting this property as Root we can limit the number of root nodes or the 0th level records to be displayed per page .

When the PageSizeMode property is set as Root the number of records to be displayed per page which is defined in the the PageSize property will be considered only for the root nodes or the 0th level records.

- **PageCount** - It is used to display the page number to be displayed in the pager.
- **CurrentPage** - This property is used to set the active page to be displayed initially.
- **TotalRecordsCount** - This property is used to limit the total number of records from the data source to be displayed in TreeGrid.

The following code example explains the properties in pageSettings.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowPaging(true)
.PageSettings(page=>
{
page.PageCount(5);
page.CurrentPage(3);
page.TotalRecordsCount(50);
page.PageSizeMode(TreeGridPageSizeMode.All);
page.PageSize(12);
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

Click here to find our online demo sample for Paging.

Pager Template

It is possible to customize the default pager in TreeGrid by using the PageSettings.Template property.

The below code snippet explains how to customize the default pager with template

ASPX-CS

```
<script type="text/x-jsrender" id="Template">
<div class="e-pagercontainer">
<div class="e-first e-icon e-mediaback e-firstpagedisabled e-disable"
title="Go to first page"></div><div class="e-prev e-icon e-arrowheadleft-
2x e-prevpagedisabled e-disable" style="border-right:none" title="Go to
previous page"></div>
</div>
<div class="e-pagercontainer e-currentPageContainer" style="border-
radius:0px">
<input id="currentPage" type="text" style="text-align:center;
margin:0px;border:none;width:32px;height:23px" />
</div>
<div id="totalPages" class="e-pagercontainer" style="margin-left:2px;
margin-bottom:5px;border:none">
<span></span>
</div>
<div class="e-pagercontainer">
<div class="e-nextpage e-icon e-arrowheadright-2x e-default" title="Go to
next page"></div><div class="e-lastpage e-icon e-mediaforward e-default"
title="Go to last page"></div>
</div>
</script>
@(Html.EJ().TreeGrid("TreeGridContainer")
.ClientSideEvents(eve => eve.ActionComplete("complete").Create("create"))
.AllowPaging(true)
.PageSettings(pg => pg.Template("#Template"))
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function create(args) {
//Code for navigating to the page
$("#currentPage").keydown(function (e) {
var obj = $("#TreeGridContainer").data("ejTreeGrid");
var val = parseInt($("#currentPage").val());
if (e.keyCode == 13 && !isNaN(val)) {
if (val > obj.model.pageSettings.totalPages)
val = obj.model.pageSettings.totalPages;
if (val <= 0)
val = 1;
obj.gotoPage(val);
return false;
}
});
}
function complete(args) {
$("#totalPages").find('span').text('of ' +
this.model.pageSettings.totalPages);
$(".e-pagercontainer:first").css('border-style', 'none');
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (args.requestType == 'paging' || args.requestType == 'refresh') {
if (this.model.pageSettings.currentPage ==
this.model.pageSettings.totalPages) {
this.element.find('.e-nextpage').addClass('e-
nextpagedisabled').removeClass('e-nextpage');
this.element.find('.e-lastpage').addClass('e-
lastpagedisabled').removeClass('e-lastpage');
}
else {
this.element.find('.e-nextpagedisabled').addClass('e-
nextpage').removeClass('e-nextpagedisabled');
this.element.find('.e-lastpagedisabled').addClass('e-
lastpage').removeClass('e-lastpagedisabled');
}
if (this.model.pageSettings.currentPage == 1) {
this.element.find('.e-prevpage').addClass('e-prevpagedisabled e-
disable').removeClass('e-prevpage');
this.element.find('.e-firstpage').addClass('e-firstpagedisabled e-
disable').removeClass('e-firstpage');
}
else {
this.element.find('.e-prevpagedisabled').addClass('e-
prevpage').removeClass('e-prevpagedisabled e-disable');
this.element.find('.e-firstpagedisabled').addClass('e-
firstpage').removeClass('e-firstpagedisabled e-disable');
}
$("#currentPage").val(this.model.pageSettings.currentPage);
}
}
</script>
```

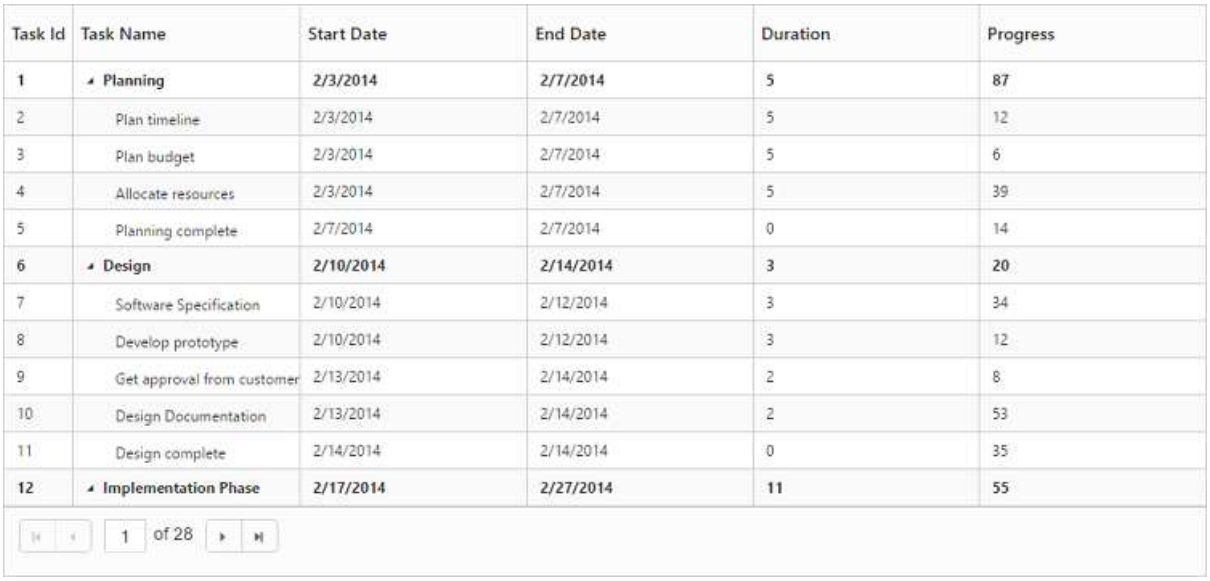
CSS

```
#currentPage {
background-color: white;
}
.darktheme #currentPage {
background-color: black;
}
.e-pagercontainer .e-icon {
display: inline-block;
height: 8px;
}
.e-pager .e-pagercontainer {
margin: 0px;
margin-left: 6px;
}
```

Click here to find our online sample for Pager Template

The below image displays TreeGrid with paging template.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	End Date	Duration	Progress
1	Planning	2/3/2014	2/7/2014	5	87
2	Plan timeline	2/3/2014	2/7/2014	5	12
3	Plan budget	2/3/2014	2/7/2014	5	6
4	Allocate resources	2/3/2014	2/7/2014	5	39
5	Planning complete	2/7/2014	2/7/2014	0	14
6	Design	2/10/2014	2/14/2014	3	20
7	Software Specification	2/10/2014	2/12/2014	3	34
8	Develop prototype	2/10/2014	2/12/2014	3	12
9	Get approval from customer	2/13/2014	2/14/2014	2	8
10	Design Documentation	2/13/2014	2/14/2014	2	53
11	Design complete	2/14/2014	2/14/2014	0	35
12	Implementation Phase	2/17/2014	2/27/2014	11	55

1 of 28

It is possible to navigate to a specific page with a custom action instead of clicking pager button by using the gotoPage method.

The below code snippet explains how to navigate to 3rd page in TreeGrid by using gotoPage method

ASPX-CS

```
var treeObject = $("#TreeGridContainer").data("ejTreeGrid");
treeObject.gotoPage(3);
```

Paging - Touch Option

With paging and responsive mode enabled in TreeGrid, it is possible to change the current page using swipe action.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.AllowPaging(true)
.IsResponsive(true)
)
@(Html.EJ().ScriptManager())
```

Toolbar

In TreeGrid we can show/hide the Toolbar by using ShowToolbar property.We can add default toolbar items by ToolbarItems. User can also create a custom toolbar items by using CustomToolbarItems.

Default Toolbar Items

Using TreeGrid default toolbar items we can perform below operations.

- **Add**- To add new task.
- **Edit**-To edit a selected task.
- **Delete**- To delete a selected task.
- **Cancel**- To cancel the edited changes in a task.
- **Update**- To save the edited changes in a task.

Copyright © 2001 - 2020 Syncfusion Inc.

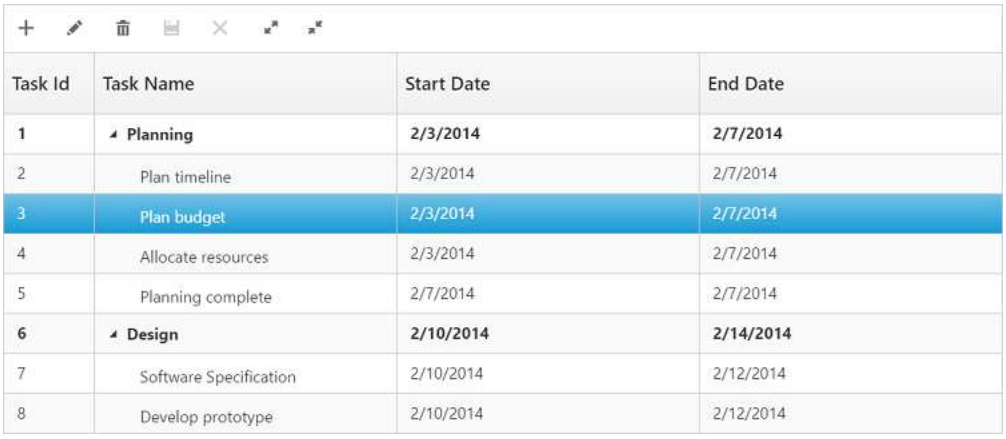
- **ExpandAll**- To expand all the TreeGrid rows.
- **CollapseAll**- To collapse all the TreeGrid rows.
- **PdfExport**- To export TreeGrid in PDF format.
- **ExcelExport**- To export TreeGrid in Excel format.

We can enable TreeGrid toolbar by using below code example:

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>()
{
TreeGridToolBarItems.Add,
TreeGridToolBarItems.Edit,
TreeGridToolBarItems.Delete,
TreeGridToolBarItems.Update,
TreeGridToolBarItems.Cancel,
TreeGridToolBarItems.ExpandAll,
TreeGridToolBarItems.CollapseAll,
TreeGridToolBarItems.ExcelExport,
TreeGridToolBarItems.PdfExport
});
})
)
```

The following screenshot displays the toolbar option in TreeGrid control.



Task Id	Task Name	Start Date	End Date
1	Planning	2/3/2014	2/7/2014
2	Plan timeline	2/3/2014	2/7/2014
3	Plan budget	2/3/2014	2/7/2014
4	Allocate resources	2/3/2014	2/7/2014
5	Planning complete	2/7/2014	2/7/2014
6	Design	2/10/2014	2/14/2014
7	Software Specification	2/10/2014	2/12/2014
8	Develop prototype	2/10/2014	2/12/2014

Note: To perform add,edit,delete,cancel,update using Toolbar items we need to enable add/edit/delete using EditSettings.

Custom Toolbar Items

CustomToolbarItems allows us to insert custom icons and custom template in TreeGrid toolbar. By using below properties we can customize TreeGrid toolbar as per our requirement.

- **Text**- To insert the custom icons in toolbar using CSS class name selector.

Copyright © 2001 - 2020 Syncfusion Inc.

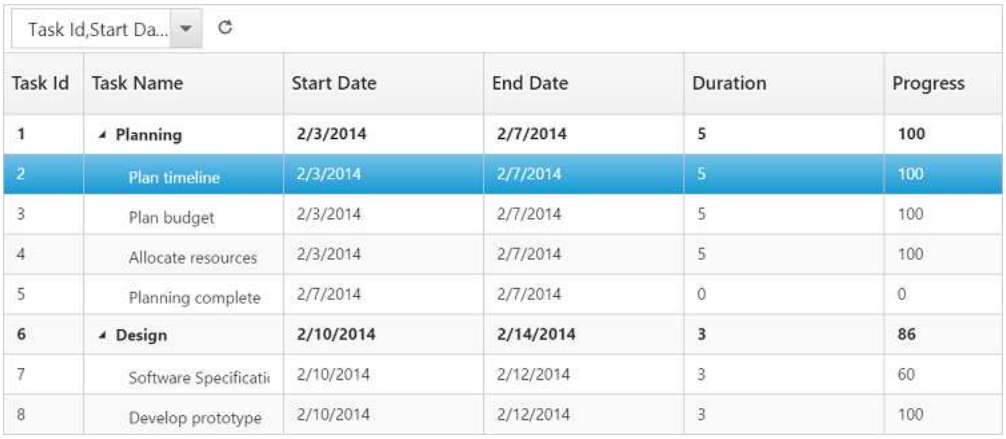
- **TemplateID**-To insert the custom icons in toolbar using script templates. Using this property we can bind HTML elements and other EJ controls to TreeGrid toolbar.
- **TooltipText**-Displays tooltip text for the custom icons.

To insert EJ Controls in TreeGrid toolbar we need to initiate the control in Create client side event.In ToolbarClick client side event we can bind actions to the custom toolbar items.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ToolbarSettings(tool =>
{
tool.ShowToolbar(true);
tool.CustomToolbarItems(ct =>
{
ct.TemplateID("#ColumnVisibility").TooltipText("Column
Visibility").Add();
ct.Text("Reset").TooltipText("Reset").Add();
});
})
.ClientSideEvents(cs =>
{
cs.Create("create");
cs.ToolbarClick("toolbarClick");
})
)
<script id="ColumnVisibility" type="text/x-jsrender">
<input id="dropdownContainer" />
</script>
<script type="text/javascript">
function toolbarClick(args) {
if (args.itemName == "Reset") {
//we can bind the custom actions here
}
}
//Here we can append custom EJ controls
function create(args) {
$("#dropdownContainer").ejDropDownList({});
}
</script>
<style type="text/css" class="cssStyles">
#TreeGridContainer_ColumnVisibility {
padding-top: 2px;
padding-bottom: 0px;
}
.Reset:before {
content: "\e677";
}
</style>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Click here to view the demo sample for custom toolbar item

Sorting

The TreeGrid control for JavaScript has built-in support for Sorting one or more columns.

Sorting Columns

TreeGrid allows the items to be sorted in ascending or descending order based on the selected column by enabling the AllowSorting option in TreeGrid control. The following code example shows you how to enable Sorting in TreeGrid control.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.AllowSorting(true)
)
@(Html.EJ().ScriptManager())
```

Multicolumn sorting

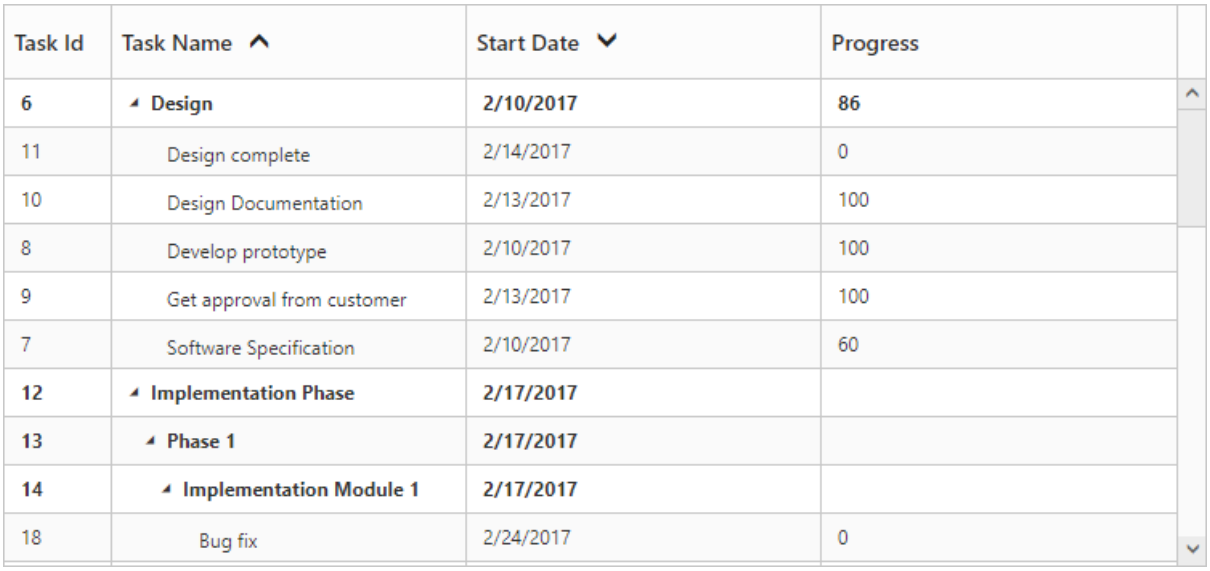
Gantt allows you to sort multiple columns by clicking the desired column headers while holding the **CTRL** key with AllowMultiSorting as true . The following code example shows you how to enable multicolumn sorting in Gantt control.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.AllowSorting(true)
.AllowMultiSorting(true)
)
@(Html.EJ().ScriptManager())
```

The following screenshot shows the output of Multicolumn sorting in TreeGrid control.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	Progress
6	Design	2/10/2017	86
11	Design complete	2/14/2017	0
10	Design Documentation	2/13/2017	100
8	Develop prototype	2/10/2017	100
9	Get approval from customer	2/13/2017	100
7	Software Specification	2/10/2017	60
12	Implementation Phase	2/17/2017	
13	Phase 1	2/17/2017	
14	Implementation Module 1	2/17/2017	
18	Bug fix	2/24/2017	0

Disable sorting for specific column

It is possible to disable sorting for a specific column by setting AllowSorting as false in the column definition.

The below code snippet demonstrates this.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.AllowSorting(true)
.Columns(co =>
{
//..
co.Field("taskName").HeaderText("Task Name").AllowSorting(false).Add();
})
)
@(Html.EJ().ScriptManager())
```

Sort column at initial load

In TreeGrid, It is possible to render the control with sorted columns, this can be achieve by using SortSettings property. We can add columns which are sorted initially in SortedColumns collection. SortedColumns collection was defined with Field and Direction properties.

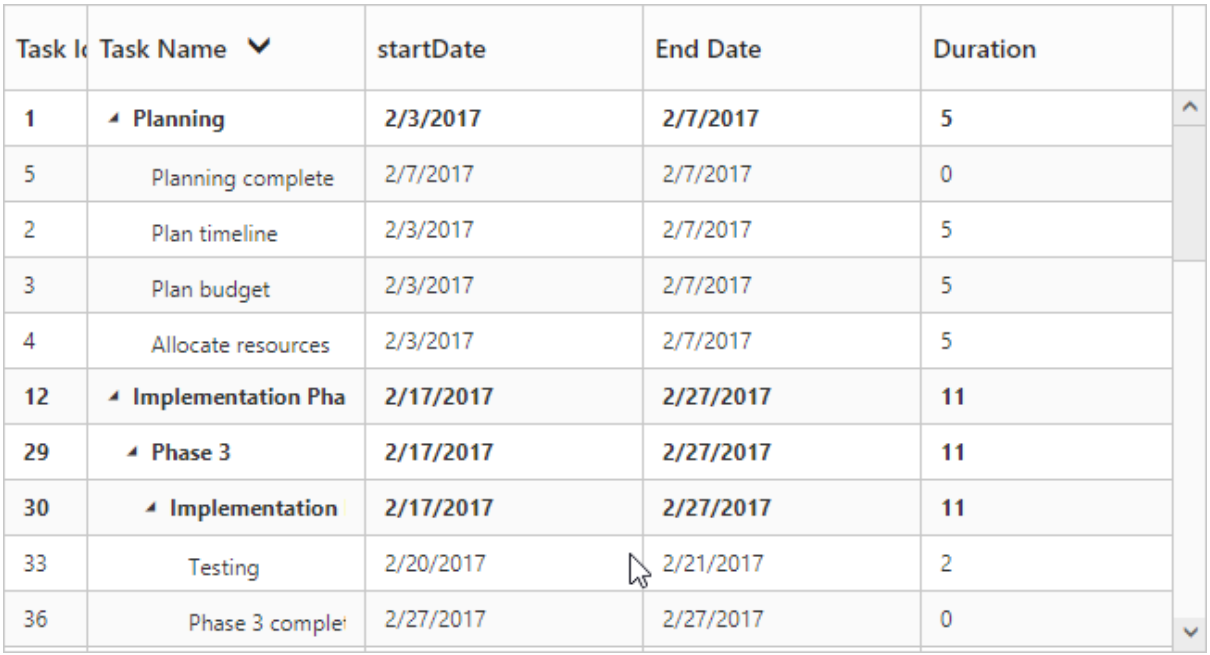
The following code shows how to add sorted column in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//...
.AllowSorting(true)
.SortSettings(ss =>
{
ss.SortedColumns(cs =>
{
cs.Field("taskName").Direction(TreeGridSortOrder.Descending).Add();
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
@(Html.EJ().ScriptManager())
```



Task Id	Task Name	startDate	End Date	Duration
1	Planning	2/3/2017	2/7/2017	5
5	Planning complete	2/7/2017	2/7/2017	0
2	Plan timeline	2/3/2017	2/7/2017	5
3	Plan budget	2/3/2017	2/7/2017	5
4	Allocate resources	2/3/2017	2/7/2017	5
12	Implementation Pha	2/17/2017	2/27/2017	11
29	Phase 3	2/17/2017	2/27/2017	11
30	Implementation	2/17/2017	2/27/2017	11
33	Testing	2/20/2017	2/21/2017	2
36	Phase 3 comple	2/27/2017	2/27/2017	0

The above screenshot shows TreeGrid rendered with descending order of **Task Name** column.

Stacked Headers

The stacked headers helps you to group the logical columns in treegrid. It can be shown by setting `ShowStackedHeader` as `true` and by defining `StackedHeaderRows`.

Adding Stacked header columns

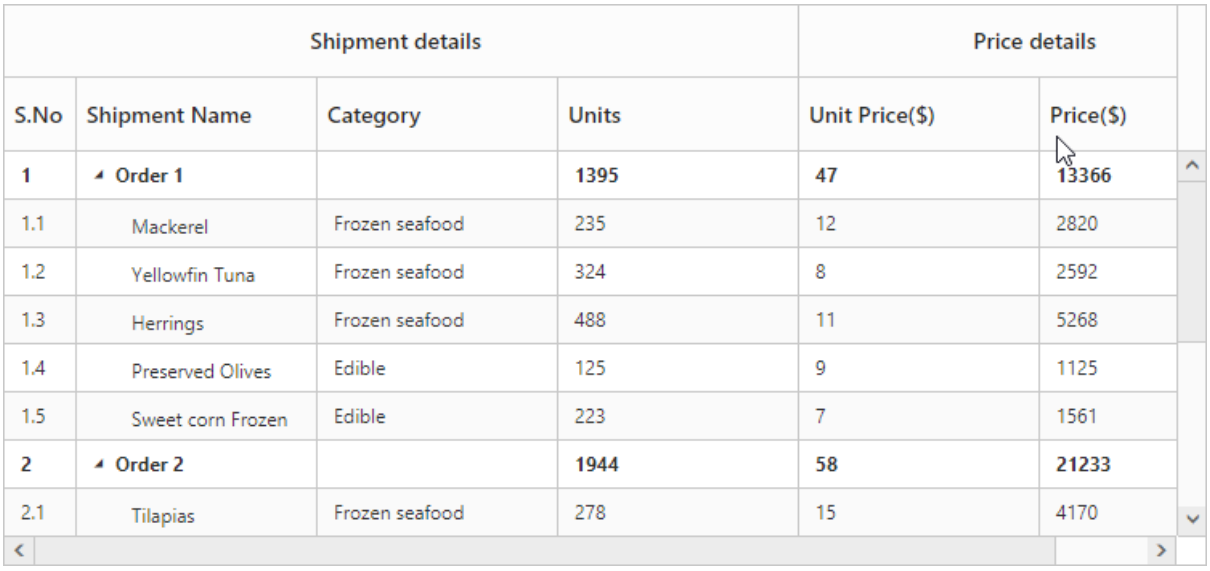
To stack columns in stacked header, you need to define `Column` property in `StackedHeaderColumns` with field names of visible columns.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//
.Columns(co =>
{
co.Field("ID").HeaderText("S.No").Width(45).Add();
co.Field("Name").HeaderText("Shipment Name").Add();
co.Field("Category").HeaderText("Category").Add();
co.Field("Units").HeaderText("Units").Add();
co.Field("UnitPrice").HeaderText("Unit Price($)").Add();
co.Field("Price").HeaderText("Price($)").Add();
}
)
.ShowStackedHeader(true)
.StackedHeaderRows(row => {
row.StackedHeaderColumns(stack =>
{
stack.HeaderText("Shipment Details").Column(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Add("ID");
col.Add("Name");
col.Add("Category");
col.Add("Units");
}).Add();
stack.HeaderText("Price Details").Column(col =>
{
col.Add("UnitPrice");
col.Add("Price");
}).Add();
}).Add();
})
)
```



Shipment details				Price details	
S.No	Shipment Name	Category	Units	Unit Price($)	Price($)
1	Order 1		1395	47	13366
1.1	Mackerel	Frozen seafood	235	12	2820
1.2	Yellowfin Tuna	Frozen seafood	324	8	2592
1.3	Herrings	Frozen seafood	488	11	5268
1.4	Preserved Olives	Edible	125	9	1125
1.5	Sweet corn Frozen	Edible	223	7	1561
2	Order 2		1944	58	21233
2.1	Tilapias	Frozen seafood	278	15	4170

Stacked header column customization

we can customize the stacked header cells with more options as described below.

CSS Class

You can provide external CSS styles to the stacked header with the help of CssClass property.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//
.ShowStackedHeader(true)
.StackedHeaderRows(row => {
row.StackedHeaderColumns(stack =>
{
stack.HeaderText("Shipment Details").Column(col =>
{
col.Add("ID");
col.Add("Name");
col.Add("Category");
col.Add("Units");
}).CssClass("grouphead").Add();
stack.HeaderText("Price Details").Column(col =>
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
col.Add("UnitPrice");
col.Add("Price");
}).Add();
}).Add();
})
)
<style>
.grouphead {
background-color: #ffb3b3;
}
</style>
```



Text Align

There is an option to align the stacked header text inside the header cell using TextAlign property as follows.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//
.ShowStackedHeader(true)
.StackedHeaderRows(row => {
row.StackedHeaderColumns(stack =>
{
stack.TextAlign(TextAlign.Right).HeaderText("Shipment
Details").Column(col =>
{
col.Add("ID");
col.Add("Name");
col.Add("Category");
col.Add("Units");
}).Add();
stack.TextAlign(TextAlign.Left).HeaderText("Price Details").Column(col =>
{
col.Add("UnitPrice");
col.Add("Price");
}).Add();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}).Add();
})
)
```



Shipment details		Price details	
S.No	Shipment Name	Unit Price($)	Price($)
1	Order 1	47	13366
1.1	Mackerel	12	2820
1.2	Yellowfin Tuna	8	2592
1.3	Herrings	11	5268
1.4	Preserved Olives	9	1125
1.5	Sweet corn Frozen	7	1561
2	Order 2	58	21233
2.1	Tilapias	15	4170
2.2	White Shrimp	7	3920

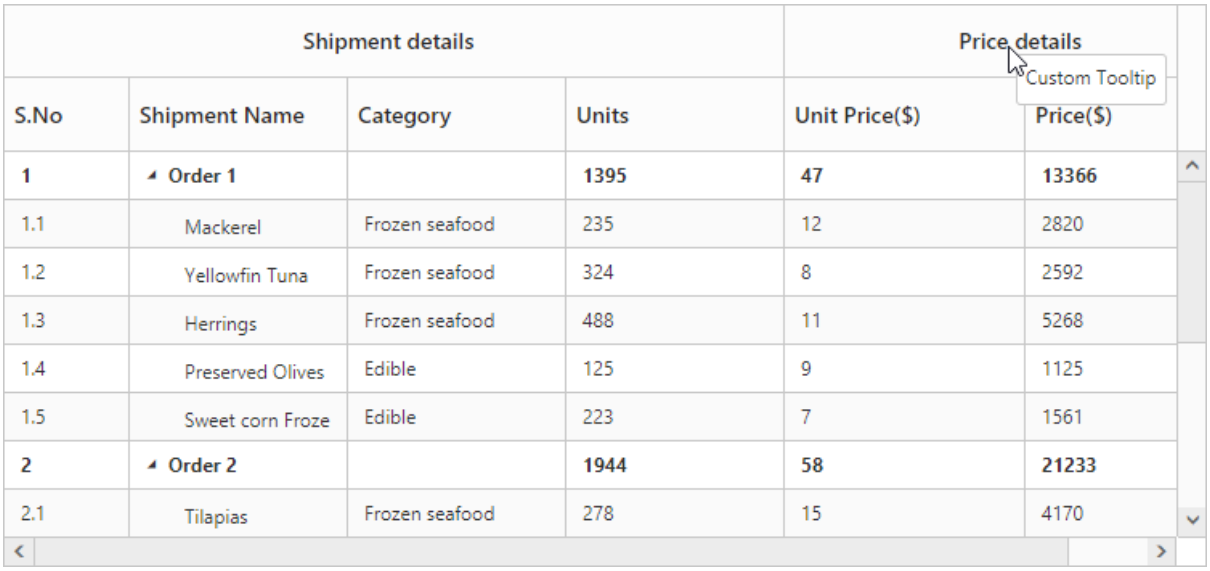
Tooltip

We can have the customized tooltip for the stacked header text with the help of ToolTip property. JsRender template script id can be assigned to this property to get tooltip.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
//
.ShowGridCellTooltip(true)
.ShowStackedHeader(true)
.StackedHeaderRows(row => {
row.StackedHeaderColumns(stack =>
{
stack.HeaderText("Shipment Details").Column(col =>
{
col.Add("ID");
col.Add("Name");
col.Add("Category");
col.Add("Units");
}).Add();
stack.HeaderText("Price Details").Column(col =>
{
col.Add("UnitPrice");
col.Add("Price");
}).Tooltip("#tooltip").Add();
}).Add();
})
)
<script id="tooltip" type="text/x-jsrender">
<div>Custom Tooltip</div>
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.



Note:

To enable stacked header tooltip we need to set ShowGridCellTooltip as true.

Click here to view the demo sample for stacked header.

Context Menu

The **Context menu** in TreeGrid control is used to manipulate (add, edit and delete) the TreeGrid rows. In TreeGrid, context menu can be enabled with ContextMenuSettings property. The ContextMenuSettings property contains two inner properties ShowContextMenu and ContextMenuItems.

The ShowContextMenu property is used to **enable or disable** the context menu, default value for this property is false.

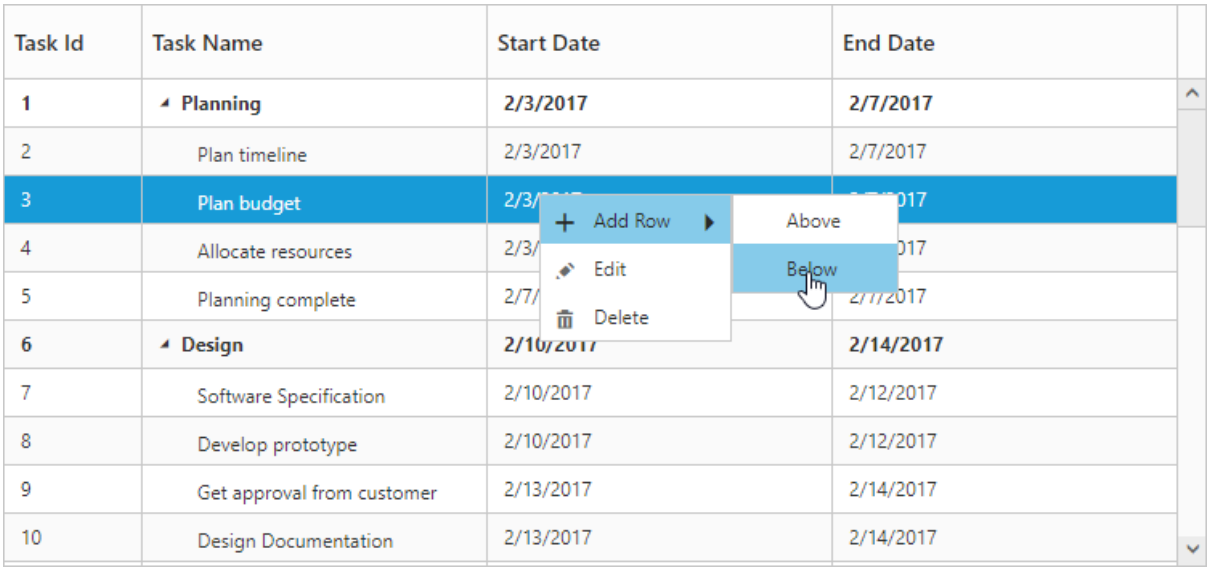
The ContextMenuItems property is used to add the menu items to context menu, this property renders Add and Delete by default when the menu items are not provided.

ASPX-CS

```
@(Html.EJ().TreeGrid("treegrid1")
.EditSettings(edit=>
{
edit.AllowEditing(true);
edit.EditMode(TreeGridEditMode.RowEditing);
})
.ContextMenuSettings(context =>
{
context.ShowContextMenu(true);
context.ContextMenuItems(new List<TreeGridContextMenuItems>(){
TreeGridContextMenuItems.Add,
TreeGridContextMenuItems.Edit,
TreeGridContextMenuItems.Delete
});
})
)
@(Html.EJ().ScriptManager())
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays the Context menu in TreeGrid control.



ContextMenu Customization

The Context menu can be customized by adding a new custom menu item to it. In TreeGrid, context menu can be customized using the ContextMenuOpen client-side event. This event is triggered when the context menu is rendered with mouse right click action. The following properties are available in the event.

- headerText: Display text for menu item.
- iconPath: Image location for menu item.
- eventHandler: Client-side event for menu item click.

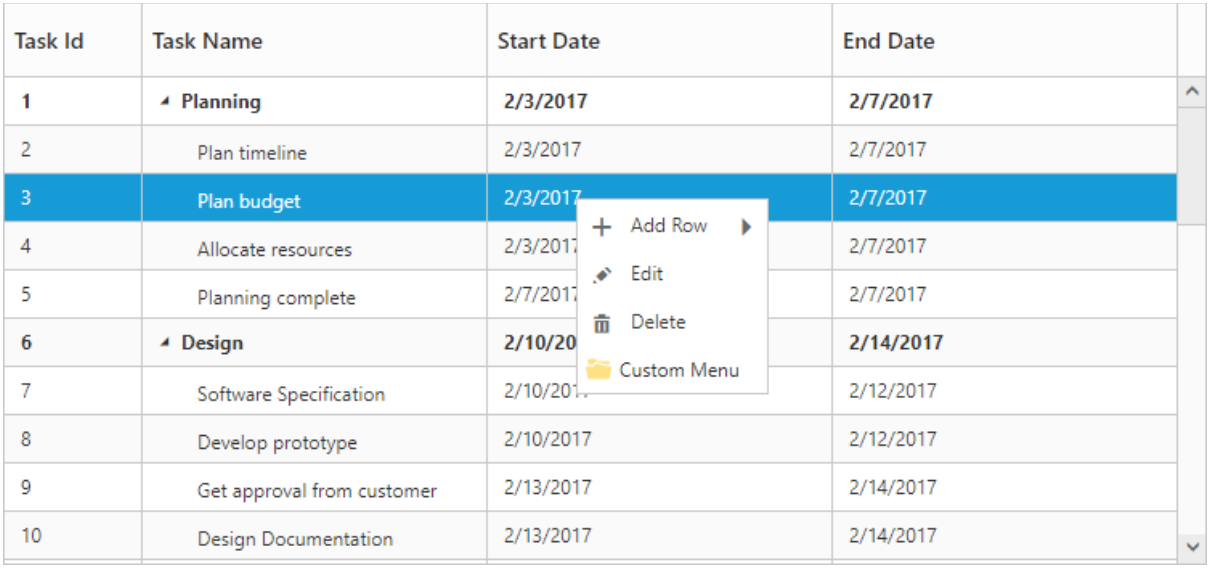
ASPX-CS

```
@(Html.EJ().TreeGrid("treegrid1")
.ContextMenuSettings(context =>
{
context.ShowContextMenu(true);
})
.ClientSideEvents(eve=>
{
eve.ContextMenuOpen("contextMenuOpen");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function contextMenuOpen(args) {
args.contextMenuItems.push({
headerText: "Custom Menu",
iconPath: "url(../content/images/treegrid/Folder.png)",
eventHandler: customMenuClick,
});
}
function customMenuClick(args) {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//Bind actions for custom menu here
}
</script>
```

The following screenshot displays the customization of Context menu in TreeGrid control.



Header Context Menu

Header context menu can be enabled by setting ShowContextMenu as true. The default value of the ShowContextMenu property is false.

Following options are shown in header context menu.

- **Column Chooser** - Display all the column names; you can enable or disable a column by select or deselect the respective column name in the column chooser menu. This option is default option of header context menu.
- **Sort Ascending & Sort Descending** - Used to sort the items in the column. These menu options will be displayed only when you enable the AllowSorting property. To perform multilevel sorting, the AllowMultiSorting property should be enabled.
- **Freeze, Unfreeze & Freeze Preceding Columns** - Used to freeze or unfreeze the columns. These set of menu options will be displayed in all the columns when IsFrozen property is enabled in any of the columns. However you can control the visibility of these menu options in a particular column by enabling/disabling the AllowFreezing property of that specific column.

The below code snippet explains how to enable header context menu in TreeGrid

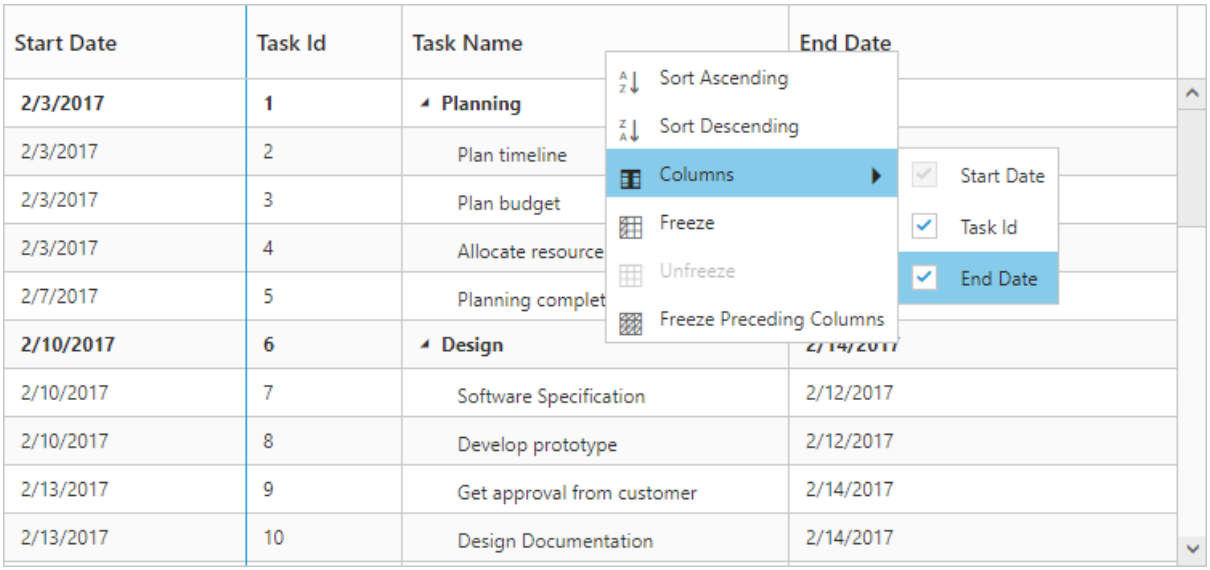
ASPX-CS

```
@(Html.EJ().TreeGrid("treegrid1")
.AllowSorting(true)
.Columns(co =>
{
co.Field("StartDate").HeaderText("Start
Date").IsFrozen(true).EditType(TreeGridEditingType.Datepicker).Add();
})
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ContextMenuSettings(context =>
{
context.ShowContextMenu(true);
})
)
@(Html.EJ().ScriptManager())
```

The following screenshot displays the Header context menu in TreeGrid control.



Printing

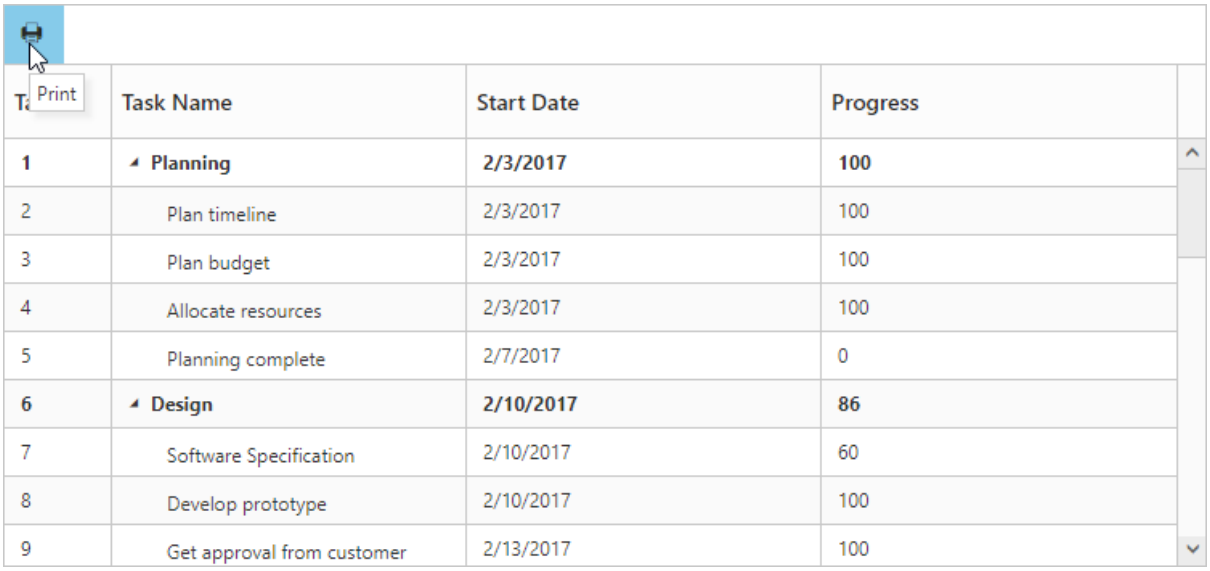
TreeGrid provides support to print the contents. To print the TreeGrid the print toolbar item must be added to ToolbarSettings.ToolbarItems property. The below code example shows how to enable print in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGrid")
.ToolbarSettings(tool => {
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>(){
TreeGridToolBarItems.Print
});
})
)
```

The below screen shot shows TreeGrid with printing enabled.

Copyright © 2001 - 2020 Syncfusion Inc.



The print preview window will be opened by clicking on this toolbar icon.

Print Mode

It is possible to set the printMode in PageSettings property, to give printing preference, as to print current page alone or all the pages in case of paging enabled in TreeGrid. The following code example explains this.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGrid")
.AllowPaging(true)
.PageSettings(pg=>pg.PrintMode(TreeGridPrintMode.CurrentPage))
.ToolbarSettings(tool => {
tool.ShowToolbar(true);
tool.ToolbarItems(new List<TreeGridToolBarItems>(){
TreeGridToolBarItems.Print
});
})
)
```

In this case only the visible records in the current page will be send to printing.

BeforePrint Event

BeforePrint event will be triggered once after printing initiated in TreeGrid. This event contains the treegrid element which is going to be printing. The following code explains this.

JS

```
@(Html.EJ().TreeGrid("TreeGrid")
.ClientSideEvents(cl => {
cl.BeforePrint("beforePrint");
})
.ToolbarSettings(tool => {
tool.ShowToolbar(true);
tool.ToolbarItems(new List < TreeGridToolBarItems > () {
TreeGridToolBarItems.Print
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
})
)
<script>
function beforePrint(args) {
//will be triggered before printing the TreeGrid
}
</script>
```

Click here to view the online demo sample Printing.

Appearance and Styling

The look and feel of the TreeGrid control can be customized by applying themes.

The following are the available themes in TreeGrid control.

1. Flat Azure
2. Flat Azure Dark
3. Flat Lime
4. Flat Lime Dark
5. Flat Saffron
6. Flat Saffron Dark
7. Gradient Azure
8. Gradient Azure Dark
9. Gradient Lime
10. Gradient Lime Dark
11. Gradient Saffron
12. Gradient Saffron Dark
13. Bootstrap
14. High Contrast 01
15. High Contrast 02
16. Material
17. Office-365

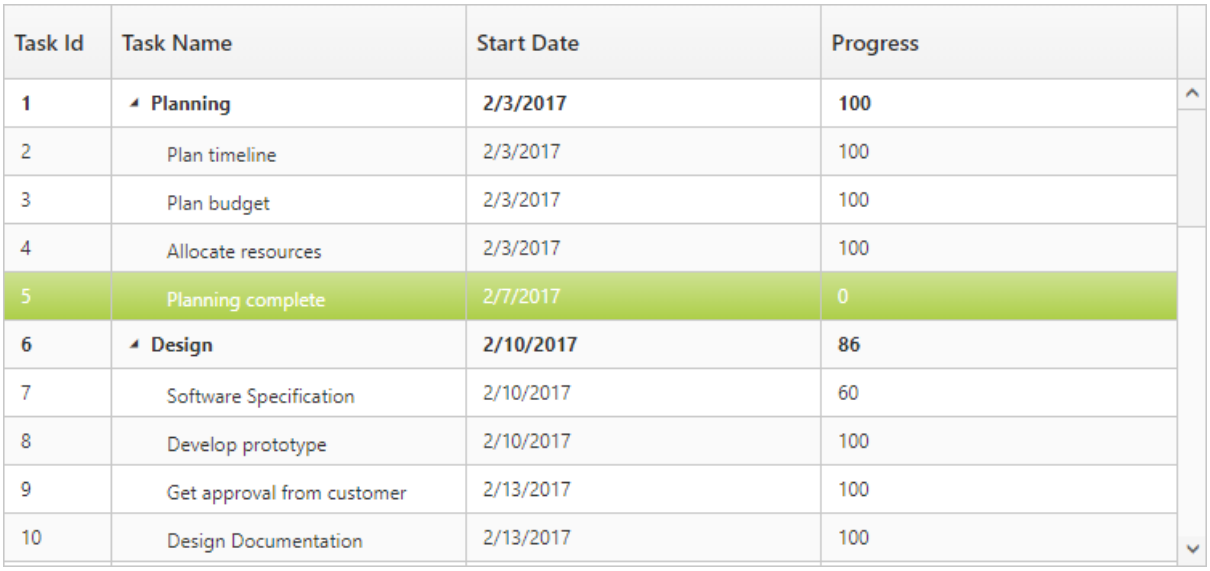
You can apply the theme (Gradient lime) to the TreeGrid control by using the style sheet from the online link as follows.

ASPX-CS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Getting Started with TreeGrid Control for JavaScript</title>
<!-- style sheet for default theme(gradient lime) -->
<link href=" http://cdn.syncfusion.com/{{ site.releaseversion
}}/js/web/flat-azure/ej.web.all.min.css" rel="stylesheet">
</html>
```

The following screenshot shows the TreeGrid control with Gradient-lime theme.

Copyright © 2001 - 2020 Syncfusion Inc.



Task Id	Task Name	Start Date	Progress
1	Planning	2/3/2017	100
2	Plan timeline	2/3/2017	100
3	Plan budget	2/3/2017	100
4	Allocate resources	2/3/2017	100
5	Planning complete	2/7/2017	0
6	Design	2/10/2017	86
7	Software Specification	2/10/2017	60
8	Develop prototype	2/10/2017	100
9	Get approval from customer	2/13/2017	100
10	Design Documentation	2/13/2017	100

Configuring CSS class

In TreeGrid CssClass property is used to apply different custom styles to multiple TreeGrid controls available in the webpage.

The following code example shows how to apply different background color for each TreeGrid control's toolbar element.

ASPX-CS

```
<style>
.c-class1.e-treegrid .e-toolbar {
background-color: rgba(169, 45, 45, 0.31);
}
.c-class2.e-treegrid .e-toolbar {
background-color: rgba(0, 128, 0, 0.2);
}
</style>
@(Html.EJ().TreeGrid("TreeGridContainer")
.CssClass("c-class1")
)
@(Html.EJ().TreeGrid("TreeGridContainer1")
.CssClass("c-class2")
)
@(Html.EJ().ScriptManager())
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



Customize rows and cells

In TreeGrid, while rendering rows `RowDataBound` event will be triggered for rows. Similarly `QueryCellInfo` event will be triggered for every cells. Using these events we can customize the TreeGrid rows and cells at initial load.

The below code example shows how to customize the rows and cells in TreeGrid.

ASPX-CS

```
@(Html.EJ().TreeGrid("TreeGridContainer")
.ClientSideEvents(eve=>
{
eve.QueryCellInfo("queryCellInfo");
```

```
eve.RowDataBound("rowDataBound");
})
)
@(Html.EJ().ScriptManager())
<script type="text/javascript">
function queryCellInfo(args) {
if (args.column.mappingName == "progress") {
if (args.data.item["progress"] < 75)
$(args.cellElement).css("background-color", "rgba(255, 0, 0, 0.12)");
else
$(args.cellElement).css("background-color", "rgba(86, 226, 86, 0.25)");
}
}
function rowDataBound(args) {
if (args.data.item["taskID"] == 5)
$(args.rowElement).css("background-color", "rgba(251, 255, 0, 0.24)");
}
</script>
```

The below screenshot shows the output of above code example.



Task Id	Task Name	startDate	End Date	Progress
1	Planning	2/3/2017	2/7/2017	100
2	Plan timeline	2/3/2017	2/7/2017	60
3	Plan budget	2/3/2017	2/7/2017	40
4	Allocate resources	2/3/2017	2/7/2017	70
5	Planning complete	2/7/2017	2/7/2017	40
6	Design	2/10/2017	2/14/2017	86
7	Software Specificati	2/10/2017	2/12/2017	60
8	Develop prototype	2/10/2017	2/12/2017	100
9	Get approval from c	2/13/2017	2/14/2017	20
10	Design Documentat	2/13/2017	2/14/2017	75

Keyboard navigation

TreeGrid control supports the following keyboard shortcuts for performing various actions and keyboard navigation can be enabled/disabled by AllowKeyboardNavigation property

Home : To select the first row.

End : To select the last row.

F2 : For editing the focused cell of the selected row.

Enter : To save the edited cell.

Delete : To delete the selected row.

Copyright © 2001 - 2020 Syncfusion Inc.

Ctrl + Down Arrow : To expand all the parent row.

Ctrl + Up Arrow : To collapse all the parent row.

Left Arrow : To collapse the selected parent row.

Right Arrow : To expand the selected parent row.

Click here to view the online demo sample for Keyboard Navigation.

How to

Rendering the TreeGrid control using TreeGridProperties in server-side

The TreeGridProperties helps to render the TreeGrid control in server-side.

The following code example explains how to render the TreeGrid control in server-side.

RAZOR

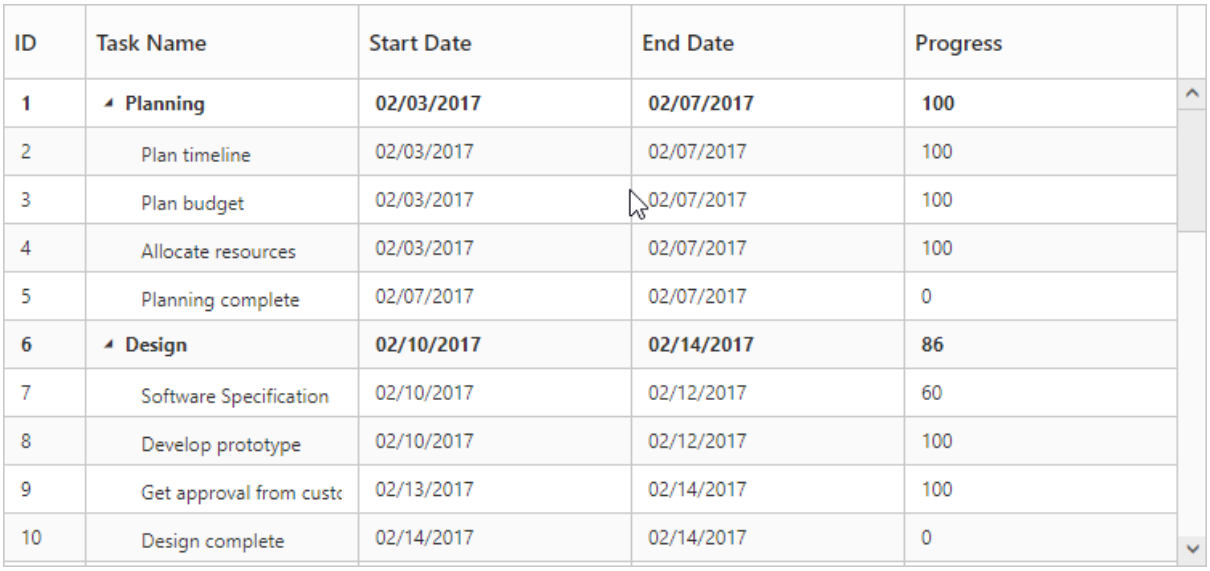
```
@(Html.EJ().TreeGrid("TreeGridControl", ViewBag.TreeGridModel))
```

C#

```
namespace TreeGrid_MVC.Controllers
{
public class TreegridController : Controller
{
public ActionResult TreeGrid()
{
TreeGridProperties treegrid = new TreeGridProperties();
List<TreeGridColumn> colList = new List<TreeGridColumn>();
colList.Add(new TreeGridColumn() { Field = "TaskId", HeaderText = "ID",
Width = 45 });
colList.Add(new TreeGridColumn() { Field = "TaskName", HeaderText = "Task
Name" });
colList.Add(new TreeGridColumn() { Field = "StartDate", HeaderText =
"Start Date" });
colList.Add(new TreeGridColumn() { Field = "EndDate", HeaderText = "End
Date" });
colList.Add(new TreeGridColumn() { Field = "Progress", HeaderText =
"Progress" });
treegrid.TreeColumnIndex = 1;
treegrid.Columns = colList;
treegrid.ChildMapping = "Children";
treegrid.SizeSettings.Width = "750px";
treegrid.SizeSettings.Height = "350px";
treegrid.DataSource = GetDefaultData();
ViewBag.TreeGridModel = treegrid;
return View();
}
}
}
```

The below screenshot shows the output of above code example.

Copyright © 2001 - 2020 Syncfusion Inc.



ID	Task Name	Start Date	End Date	Progress
1	Planning	02/03/2017	02/07/2017	100
2	Plan timeline	02/03/2017	02/07/2017	100
3	Plan budget	02/03/2017	02/07/2017	100
4	Allocate resources	02/03/2017	02/07/2017	100
5	Planning complete	02/07/2017	02/07/2017	0
6	Design	02/10/2017	02/14/2017	86
7	Software Specification	02/10/2017	02/12/2017	60
8	Develop prototype	02/10/2017	02/12/2017	100
9	Get approval from custc	02/13/2017	02/14/2017	100
10	Design complete	02/14/2017	02/14/2017	0

TreeView

Overview

The EJ MVC TreeView component provides all the functionalities that is available in EJ TreeView, since it is a server side wrapper of EJ TreeView widget. It represents hierarchical data in a tree-like structure and allows to edit, drag n drop, selection with checkboxes and more. TreeView can be generated from the specified item list, list of objects or using OData services.

Key features

- Data population
- Node manipulations
- Drag and drop
- Template
- Checkbox
- Accessibility
- Appearance and styling

Getting started

This section explains briefly about how to create a TreeView in ASP.NET MVC platform.

Create your first TreeView in MVC

Create a MVC Project and add necessary assemblies, CSS and scripts with the help of the given MVC-Getting Started documentation. After creating this project, you can create a TreeView in following ways.

TreeView using Helper element

You can create a tree using “Items” API of TreeView control. Here there is no necessary to use a data source for rendering TreeView.

In the view page, add TreeView helper as shown below. It will render the TreeView with specified items.

RAZOR

```
@Html.EJ().TreeView("tree").Items(items =>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
items.Add().Text("Item 1").Children(child =>
{
child.Add().Text("Item 1.1").Children(child1 =>
{
child1.Add().Text("Item 1.1.1");
});
});
items.Add().Text("Item 2").Children(child =>
{
child.Add().Text("Item 2.1");
child.Add().Text("Item 2.2");
});
items.Add().Text("Item 3").Children(child =>
{
child.Add().Text("Item 3.1");
});
})
```

TreeView using Data Binding

Another way of creating TreeView is binding with the data source, you can bind local data source to create a TreeView.

The beforeLoad event will be triggered before loading nodes into TreeView.

Render TreeView with local data source.

In the model page, specify the TreeView node properties as shown below.

C#

```
public class LoadData
{
public int Id { get; set; }
public int Parent { get; set; }
public string Text { get; set; }
}
```

In the controller page, create a data list which contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 4, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 6, Parent = 3, Text = "Item 3.1" });
ViewBag.datasource = load;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Above data can be directly assigned to DataSource property and mapping data fields with respect to the mapper field in order to form TreeView.

RAZOR

```
@(Html.EJ().TreeView("tree")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
)
```

Create Instance for TreeView

You can create an instance for existing TreeView in following ways. Once a reference has been established, you can use the API's of TreeView to control its behavior.

JAVASCRIPT

```
<script type="text/javascript">
//Specify this after your TreeView creation
//create instance for TreeView
var treeObj = $("#treeView").ejTreeView('instance');
//or
var treeObj1 = $("#treeView").data("ejTreeView");
</script>
```

Note: To configure the API settings after TreeView creation, please refer API configuration, Invoking Methods.

TreeView events

EJ MVC TreeView supports all the events which is available in EJ TreeView. Refer the following code example to specify an event using HTML helper of EJ MVC TreeView.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.ClientSideEvents(events =>
events.NodeExpand("onNodeExpanded")
.NodeCollapse("onNodeCollapsed")
.NodeSelect("onNodeSelected")
)
)
<script type="text/javascript">
function onNodeCollapsed(args) {
//Handle the node collapse event
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function onNodeExpanded(args) {
//Handle the node expand event
}
function onNodeSelected(args) {
//Handle the node select event
}
</script>
```

Populate Data

TreeView can be populated with local or remote data source using a property DataSource, which is the member of "**TreeViewFields**" property.

In TreeView, you should use "**TreeViewFields**" property to go with data source. It specifies the mapping fields for the data source to receive the data, query to process the data and field mappers to map the data members.

Fields

Below table list outs the field members with description.

Properties	**Description**
DataSource	The data source contains the list of data for generating the TreeView list. Also contains properties to load data from remote data source.
Query	It specifies the query to retrieve the data from the online server.
TableName	It specifies the name of the table from which data to be processed from given data source.
Id	It specifies the ID of the node.
ParentId	It specifies the parent id of the node
Text	It specifies the text content of the node.
HasChild	It specifies the node has child (which is the nested or inner level of nodes). Also itâ€™s used in load on demand of tree data.
Expanded	It specifies the tree node to be in expanded state
Selected	It specifies the select node at initialize. [note: only one node get selected]
IsChecked	It specifies the node to be in checked state, if tree node represented with checkboxes.
ImageUrl	It defines the image location.
ImageAttribute	It defines the image attributes such as height, width, styles, etc.
SpriteCssClass	It defines the sprite CSS for the image tag.

Copyright © 2001 - 2020 Syncfusion Inc.

HtmlAttribute	It defines the HTML attributes such as class and styles for a node ("li" tag).
LinkAttribute	It defines the HTML attributes such as class and styles for a link tag, which is child of node.
Child	It used to specify the properties of child nodes in TreeView

Mapping all fields with data source

In the model page, specify the TreeView node properties as shown below.

C#

```
public class LoadData
{
public int Id { get; set; }
public int Parent { get; set; }
public string Text { get; set; }
public string SpriteImage { get; set; }
public string ImageURL { get; set; }
public bool HasChild { get; set; }
public bool Expanded { get; set; }
public bool Selected { get; set; }
public bool NodeChecked { get; set; }
public object NodeProperty { get; set; }
public object LinkProperty { get; set; }
public object ImageProperty { get; set; }
}
```

In the controller page, create a data list which contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> treeData = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
treeData.Add(new LoadData
{
Id = 1,
Parent = 0,
Text = "Item 1",
Expanded = true,
NodeProperty = new Dictionary<string, string>() {
{ "class", "text-blue" },
{ "value", "Item 1" }
}
});
treeData.Add(new LoadData
{
Id = 2,
Parent = 0,
Text = "Item 2",
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
LinkProperty = new Dictionary<string, string>() {
{ "class", "text-underline" },
{ "href", "http://www.syncfusion.com" },
{ "target", "_blank"}
}
});
treeData.Add(new LoadData
{
Id = 3,
Parent = 0,
Text = "Item 3",
Selected = true,
SpriteImage = "mailicon sprite-calendar"
});
treeData.Add(new LoadData
{
Id = 4,
Parent = 0,
Text = "Item 4",
NodeChecked = true,
ImageProperty = new Dictionary<string, string>() {
{ "width", "20px" },
{ "height", "20px" }
},
ImageURL = "http://cdn.syncfusion.com/13.3.0.7/js/web/flat-
azure/images/ajax-loader.gif"
});
treeData.Add(new LoadData
{
Id = 5,
Parent = 1,
Text = "Item 1.1"
});
treeData.Add(new LoadData
{
Id = 6,
Parent = 1,
Text = "Item 1.2"
});
treeData.Add(new LoadData
{
Id = 7,
Parent = 1,
Text = "Item 1.3"
});
treeData.Add(new LoadData
{
Id = 8,
Parent = 3,
Text = "Item 3.1"
});
treeData.Add(new LoadData
{
Id = 9,
Parent = 3,
Text = "Item 3.2"
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
treeData.Add(new LoadData
{
Id = 10,
Parent = 5,
Text = "Item 1.1.1"
});
ViewBag.datasource = treeData;
return View();
}
}
```

In the view page, add TreeView helper and map the properties defined to the corresponding fields in data source.

RAZOR

```
@using TreeView_Doc.Models
<div style="width: 250px">
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
.IsChecked("NodeChecked")
.Selected("Selected")
.SpriteCssClass("SpriteImage")
.ImageUrl("ImageURL")
.LinkAttribute("LinkProperty")
.HtmlAttribute("NodeProperty")
.ImageAttribute("ImageProperty")
)
.ShowCheckbox(true)
)
</div>
```

Note: If you want to display nodes in root level, exclude parent attribute or specify “0” in corresponding value.

Local data

TreeView can be rendered from a self-referential data by providing the two required fields “**id**”and “**parent**”.

In the controller page, create a data list which contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
load.Add(new LoadData { Id = 4, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 6, Parent = 3, Text = "Item 3.1" });
ViewBag.datasource = load;
return View();
}
}
```

Above data can be directly assigned to DataSource property and mapping data fields with respect to the mapper field in order to form TreeView.

RAZOR

```
@(Html.EJ().TreeView("tree")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
)
```

Nested Object Support

The nested object support is provided for the TreeView component. Please find the following data list which contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = new InnerData {
nodeName = "Item 1" } });
load.Add(new LoadData { Id = 2, Parent = 0, Text = new InnerData {
nodeName = "Item 2" } });
load.Add(new LoadData { Id = 3, Parent = 0, Text = new InnerData {
nodeName = "Item 3" } });
load.Add(new LoadData { Id = 4, Parent = 1, Text = new InnerData {
nodeName = "Item 1.1" } });
load.Add(new LoadData { Id = 5, Parent = 1, Text = new InnerData {
nodeName = "Item 1.2" } });
load.Add(new LoadData { Id = 6, Parent = 3, Text = new InnerData {
nodeName = "Item 3.1" } });
ViewBag.datasource = load;
return View();
}
}
public class LoadData
{
public int Id { get; set; }
public int Parent { get; set; }
public InnerData Text { get; set; }
}
public class InnerData
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
public string nodeName { get; set; }
}
```

Above data can be directly assigned to DataSource property and mapping data fields with respect to the mapper field in order to form TreeView.

RAZOR

```
@(Html.EJ().TreeView("tree")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text.nodeName")
)
)
```

Remote Data

When using remote data binding, the adaptor of ej.DataManager plays vital role in processing queries to make them suitable to sends along with data request and also process the response data from the server.

The following steps explain how you can bind remote data to TreeView control.

- In the view page, add TreeView helper to configure TreeView.
- In the DataSource field assign remote data source. Here DataManager gets the remote web service and filters the data using Query API. The select property of ej.Query is used to retrieve the specified columns from the data source.
- Assign DataSource and Query property values to bind the remote data. Map the corresponding fields in TreeView control

OData

OData is a standardized protocol for creating and consuming data. You can bind OData service data to TreeView in two ways using DataSource API of “TreeView” control.

Using Datasource(DataSource) API

Create an object for DataSource class using OData service URL and then assign it to DataSource API of “TreeView”.

RAZOR

```
@{
DataSource treeData = new DataSource();
treeData.URL = "//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/";
}
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource(treeData)
.Query("ej.Query().from('Categories').select('CategoryID,CategoryName').t
ake(3)")
.Id("CategoryID")
.Text("CategoryName")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Child(childField =>
childField.Datasource(treeData)
.TableName("Products")
.Id("ProductID")
.ParentId("CategoryID")
.Text("ProductName")
)
)
)
```

Using Datasource(Action<DataSourceBuilder>) API

Here directly, you can specify the OData service URL using Action<DataSourceBuilder>

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/")
)
.Query("ej.Query().from('Categories').select('CategoryID,CategoryName').t
ake(3)")
.Id("CategoryID")
.Text("CategoryName")
.Child(childField =>
childField.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/")
)
.TableName("Products")
.Id("ProductID")
.ParentId("CategoryID")
.Text("ProductName")
)
)
)
```

In above methods, you may also specify the **adaptor** as ODataAdaptor and it is optional to specify.

You can provide adaptor value either as string value (“ODataAdaptor”) or Enum type (AdaptorType.ODataAdaptor)

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/")
.Adaptor(AdaptorType.ODataAdaptor)
)
.Query("ej.Query().from('Categories').select('CategoryID,CategoryName').t
ake(3)")
.Id("CategoryID")
.Text("CategoryName")
.Child(childField =>
childField.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Adaptor(AdaptorType.ODataAdaptor)
)
.TableName("Products")
.Id("ProductID")
.ParentId("CategoryID")
.Text("ProductName")
)
)
)
```

Note: You can use above code until OData service version 3. For OData Service version 4 End Point, we have created a separate adaptor ej.ODataV4Adaptor for data binding.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Categori
es")
.Adaptor(AdaptorType.ODataV4Adaptor)
)
.Query("ej.Query().from('Categories').select('CategoryID,CategoryName').t
ake(3)")
.Id("CategoryID")
.Text("CategoryName")
.Child(childField =>
childField.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Products
")
.Adaptor(AdaptorType.ODataV4Adaptor)
)
.TableName("Products")
.Id("ProductID")
.ParentId("CategoryID")
.Text("ProductName")
)
)
)
```

Web API

Using ej.WebApiAdaptor, you can bind Web API service data to TreeView as shown in below code example.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource(data =>
data.URL("http://mvc.syncfusion.com/OdataServices/treeView/TreeViewData/G
etAllData")
.Adaptor(AdaptorType.WebApiAdaptor)
.CrossDomain(true)
)
.Id("id")
.Text("name")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ParentId("pid")
)
)
```

Other Restful web services

The Custom Adaptor concept of ej.DataManager allow you to customize or generate your own adaptor which is used to process query and result data.

http://helpjs.syncfusion.com/js/datamanager/data-adaptors#custom-adaptor

When using remote data binding, the adaptor of ej.DataManager plays vital role in processing queries to make them suitable to sends along with data request and also process the response data from the server.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 4, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 6, Parent = 3, Text = "Item 3.1" });
ViewBag.datasource = load;
return View();
}
}
```

In the view page, add TreeView helper and specify the custom adaptor as shown below.

RAZOR

```
@(Html.EJ().TreeView("tree")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.ClientSideEvents(events =>
events.Create("onCreate")
)
)
<script type="text/javascript">
function onCreate() {
var treeObj = $("#tree").ejTreeView("instance");
var treeData = treeObj.model.fields.dataSource;
var customAdaptor = new ej.Adaptor().extend({
insert: function (dataManagerObj, data) {
return dataManagerObj.dataSource.json.push(data);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
},
processQuery: ej.JsonAdaptor.prototype.processQuery
});
var dataManager = new ej.DataManager(treeData);
dataManager.adaptor = new customAdaptor();
dataManager.insert({ Id: 7, Text: "CustomData", Parent: 0 });
treeObj.option("fields", { dataSource: dataManager });
}
</script>
```

Load on demand

Load on demand is a technique (Lazy load) that is used to reduce the bandwidth size of consuming huge data. You can load data on demand in TreeView by using "**LoadOnDemand**" property when you're going to use huge data.

For local data source, TreeView loads the first level nodes initially. While expand a parent node then TreeView loads it's their child nodes from the data source based on the parentId member. It reduces the time to render TreeView with huge data.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public class HomeController : Controller
{
List<loadOnDemand> data = new List<loadOnDemand>();
public ActionResult Index()
{
data.Add(new loadOnDemand { id = 1, name = "Local Disk(C:)", hasChild =
true });
data.Add(new loadOnDemand { id = 2, name = "Local Disk(D:)", hasChild =
true });
data.Add(new loadOnDemand { id = 3, name = "Local Disk(E:)", hasChild =
true });
data.Add(new loadOnDemand { id = 4, parentId = 1, name = "Folder 1",
hasChild = true });
data.Add(new loadOnDemand { id = 5, parentId = 1, name = "Folder 2" });
data.Add(new loadOnDemand { id = 6, parentId = 1, name = "Folder 3" });
data.Add(new loadOnDemand { id = 7, parentId = 2, name = "Folder 4" });
data.Add(new loadOnDemand { id = 8, parentId = 2, name = "Folder 5",
hasChild = true });
data.Add(new loadOnDemand { id = 9, parentId = 2, name = "Folder 6" });
data.Add(new loadOnDemand { id = 10, parentId = 3, name = "Folder 7" });
data.Add(new loadOnDemand { id = 11, parentId = 3, name = "Folder 8" });
data.Add(new loadOnDemand { id = 12, parentId = 3, name = "Folder 9",
hasChild = true });
data.Add(new loadOnDemand { id = 13, parentId = 4, name = "File 1" });
data.Add(new loadOnDemand { id = 14, parentId = 4, name = "File 2" });
data.Add(new loadOnDemand { id = 15, parentId = 4, name = "File 3" });
data.Add(new loadOnDemand { id = 16, parentId = 8, name = "File 4" });
data.Add(new loadOnDemand { id = 17, parentId = 8, name = "File 5" });
data.Add(new loadOnDemand { id = 18, parentId = 8, name = "File 6" });
data.Add(new loadOnDemand { id = 19, parentId = 12, name = "File 7" });
data.Add(new loadOnDemand { id = 20, parentId = 12, name = "File 8" });
data.Add(new loadOnDemand { id = 21, parentId = 12, name = "File 9" });
ViewBag.datasource = data;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
return View();
}
}
public class loadOnDemand
{
public int id { get; set; }
public int? parentId { get; set; }
public string name { get; set; }
public bool? hasChild { get; set; }
public bool? expanded { get; set; }
public bool? ischecked { get; set; }
public bool? selected { get; set; }
public string spriteCss { get; set; }
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in “Datasource” with that enable load on demand option.

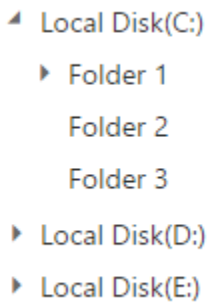
RAZOR

```
@Html.EJ().TreeView("tree").LoadOnDemand(true).TreeViewFields(field =>
field.Datasource((IEnumerable<loadOnDemand>)ViewBag.datasource)
.Id("id").Text("name").ParentId("parentId").HasChild("hasChild")
)
```

The following screenshot displays the load on demand for local data source in TreeView control.



While expanding the parent node



After expanding the parent node

For more details about load on demand for local data source, refer the sample here.

For remote data source, TreeView loads the first level nodes initially. While expand the node from TreeView, the data manager passes the query to the controller. Based on this query, you can filter the data from table and return to TreeView.

Copyright © 2001 - 2020 Syncfusion Inc.

Refer below code example to load data on demand from remote data source.

RAZOR

```
@(Html.EJ().TreeView("treeView").LoadOnDemand(true).TreeViewFields(field
=>
field.Datasource(data =>
data.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/"))
.Query("ej.Query().from('Categories').select('CategoryID,CategoryName').t
ake(3)")
.Id("CategoryID").Text("CategoryName").HasChild("CategoryName").Child(chi
ld =>
child.Datasource(childData =>
childData.URL("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/"))
.TableName("Products").ParentId("CategoryID").Text("ProductName")))
)
```

For URL Adaptor, TreeView loads the first level nodes initially. While expand the node from TreeView, the data manager passes the Data Manager instance to the controller. The Data Manager instance contains the parent Id and filter options. Based on the Data Manager instance, you can filter the data from data source and return to TreeView.

In the controller page, create a data list that contains the details about tree nodes and create method to handle client side request.

C#

```
public class HomeController : Controller
{
public ActionResult Index()
{
return View();
}
public JsonResult Data(DataManager dataManagerObj)
{
List<loadOnDemand> treeData = GetTreeData();
IEnumerable<loadOnDemand> results;
if (dataManagerObj.Where == null)
{
//return the first level nodes
results = treeData.Where(item => item.parentId == null);
}
else
{
//return the nodes which has parentId as you request
results = treeData.Where(s => s.parentId ==
Convert.ToInt32(dataManagerObj.Where[0].value));
}
return Json(results, JsonRequestBehavior.AllowGet);
}
List<loadOnDemand> data = new List<loadOnDemand>();
public List<loadOnDemand> GetTreeData()
{
data.Add(new loadOnDemand { id = 1, name = "Local Disk(C:)", hasChild =
true });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
data.Add(new loadOnDemand { id = 2, name = "Local Disk(D:)", hasChild =
true });
data.Add(new loadOnDemand { id = 3, name = "Local Disk(E:)", hasChild =
true });
data.Add(new loadOnDemand { id = 4, parentId = 1, name = "Folder 1",
hasChild = true });
data.Add(new loadOnDemand { id = 5, parentId = 1, name = "Folder 2" });
data.Add(new loadOnDemand { id = 6, parentId = 1, name = "Folder 3" });
data.Add(new loadOnDemand { id = 7, parentId = 2, name = "Folder 4" });
data.Add(new loadOnDemand { id = 8, parentId = 2, name = "Folder 5",
hasChild = true });
data.Add(new loadOnDemand { id = 9, parentId = 2, name = "Folder 6" });
data.Add(new loadOnDemand { id = 10, parentId = 3, name = "Folder 7" });
data.Add(new loadOnDemand { id = 11, parentId = 3, name = "Folder 8" });
data.Add(new loadOnDemand { id = 12, parentId = 3, name = "Folder 9",
hasChild = true });
data.Add(new loadOnDemand { id = 13, parentId = 4, name = "File 1" });
data.Add(new loadOnDemand { id = 14, parentId = 4, name = "File 2" });
data.Add(new loadOnDemand { id = 15, parentId = 4, name = "File 3" });
data.Add(new loadOnDemand { id = 16, parentId = 8, name = "File 4" });
data.Add(new loadOnDemand { id = 17, parentId = 8, name = "File 5" });
data.Add(new loadOnDemand { id = 18, parentId = 8, name = "File 6" });
data.Add(new loadOnDemand { id = 19, parentId = 12, name = "File 7" });
data.Add(new loadOnDemand { id = 20, parentId = 12, name = "File 8" });
data.Add(new loadOnDemand { id = 21, parentId = 12, name = "File 9" });
return data;
}
}
public class loadOnDemand
{
public int id { get; set; }
public int? parentId { get; set; }
public string name { get; set; }
public bool? hasChild { get; set; }
public bool? expanded { get; set; }
public bool? ischecked { get; set; }
public bool? selected { get; set; }
public string spriteCss { get; set; }
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in “Datasource” with that enable load on demand option.

RAZOR

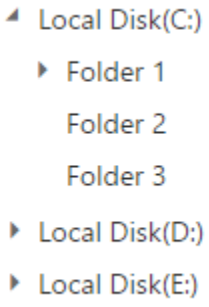
```
@Html.EJ().TreeView("tree").LoadOnDemand(true).TreeViewFields(field =>
field.Datasource(data =>
data.URL("/Home/Data").Adaptor(AdaptorType.UrlAdaptor)
).Text("name").Id("id").ParentId("parentId").HasChild("hasChild"))
```

The following screenshot displays the load on demand for URL adaptor in TreeView control.

Copyright © 2001 - 2020 Syncfusion Inc.



While expanding the parent node



After expanding the parent node

Tree Node

TreeView node is structured with expand/ collapse arrow, checkbox, image and text as shown in below.



Also TreeView node object holds the following properties.

Properties	Data Type	Description
checked	Boolean	Checked state of the node
count	Number	Number of child
expanded	Boolean	Expanded state of the node

Copyright © 2001 - 2020 Syncfusion Inc.

index	Number	Index of the node
level	Number	Level of the node
id	String	Node id
parentId	String	Parent id of the node
selected	Boolean	Selected state of the node

Get / Set Node Value

TreeView provides a set of options to configure all its properties by setting and getting values at initialization or dynamically.

To get the node value, you can use getNode method as shown in the below code example, in which on button click action the node value has retrieved.

Also you can get the text value of tree node by using getText method.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 4, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 6, Parent = 3, Text = "Item 3.1" });
ViewBag.datasource = load;
return View();
}
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
)
```

In the view page, specify following method in script tag to get node value

Copyright © 2001 - 2020 Syncfusion Inc.

RAZOR

```
<script type="text/javascript">
//bind below onClick action to button to get node at button click or else
in any action
function onClick() {
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//get node object using getNode method
var node = treeObj.getNode("1");
}
</script>
```

Note: Existing TreeView instance can be created by jQuery.data() and you can control the API's of TreeView behavior.

To edit the node text, you can use updateText method as shown below code example.

RAZOR

```
<script type="text/javascript">
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//get node object using getNode method
var node = treeObj.getNode("2");
//sets text to existing node
treeObj.updateText(node.id, "updated Item");
</script>
```

Get Parent Node

To get current parent node of a particular node, you can use the getParent method as shown in below code example

RAZOR

```
<script type="text/javascript">
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//get parent node using getParent method
var node = treeObj.getParent("4");
</script>
```

Get Node Index

To get node index, you can use the getNodeIndex as shown in below code example

You can use getNodeByIndex method to get TreeView node by using index position.

RAZOR

```
<script type="text/javascript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//get node object using getNode method
var node = treeObj.getNode("4");
//gets the index of node
treeObj.getNodeIndex(node.id);
</script>
```

Node Manipulations

You can perform following operation in tree nodes and the modified node values can be saved in database.

Add or Remove nodes

To add/remove nodes programmatically, use addNode and removeNode methods of the TreeView.

JAVASCRIPT

```
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
var newNode = { Id: 11, Text: "Item 2.1" };
//to add tree node
treeObj.addNode(newNode, "2");
//to remove node
treeObj.removeNode("4");
```

You can able to add a new node after or before some TreeView node by using insertAfter and insertBefore methods.

JAVASCRIPT

```
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
var newNode = { Id: 12, Text: "Item 2.2" };
//to add tree node after some element, which having id 2
treeObj.{{'**insertAfter(newNode, "2");**'| markdownify }}
var newNode = { Id: 13, Text: "Item 2.3" };
//to add tree node before some element, which having id 2
treeObj.{{'**insertBefore(newNode, "2");**'| markdownify }}
```

Move node

You can also achieve cut and paste operation by using moveNode method.

JAVASCRIPT

```
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
//to move tree node which having id 5, to child of node which having id
2.
treeObj.moveNode("5", "2");
```

Expand or Collapse node

Tree nodes can be expanded or collapsed by clicking the expand/collapse icon of the node or in code behind by using the following methods.

Methods	Description
expandNode	Expands the node with specified id
collapseNode	Collapse the node with specified id
expandAll	Expands all the node
collapseAll	Collapse all the node

Also you can get all the expanded nodes index in tree by using getExpandedNodesIndex method, which returns the array of expanded node indices.

Prevent multiple node expand

You can able to prevent multiple expand of TreeView nodes by specifying enableMultipleExpand as false.

For example, if you want to allow only one node to be expanded in TreeView at a time. Refer the below code block for more details.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.EnableMultipleExpand(false)
)
```

Get updated node collection

You can get the updated node values after manipulating or editing the node of TreeView using getTreeData method. It returns the JSON data represented as in tree with modified structure.

You can also get the updated data source for remote data binding after performing the operation like editing, selecting/unselecting, expanding/collapsing, checking/unchecking and removing node. You cannot get the updated data source when you perform operation like drag and drop, adding node for remote data binding.

The updated data source also contains custom attributes if you return these from server.

JAVASCRIPT

```
//create an instance from an existing TreeView.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//to get TreeView data
treeObj.getTreeData();
```

Editing

You can directly edit the tree node's text in-place by double-click on the tree node or select the tree node and press F2 key. The editing works only if the allowEditing property is true in TreeView control. When editing is completed by focus out or "enter" key press, the modified node's text is saved automatically. The nodeEdit event will be triggered whenever edited the TreeView node.

Also beforeEdit event will be triggered before the TreeView node change into editing mode.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.AllowEditing(true)
)
```

Selection

You can select a specific node by using selectedNode property or selectNode method. To get the selected status of a given TreeView node you have to use isSelected method.

The nodeClick event will be triggered whenever TreeView node is clicked. The beforeSelect event will be triggered before the TreeView node is selected.

The nodeSelect/nodeUnselect events will be triggered based on the TreeView node click operations.

JAVASCRIPT

```
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//to select node
treeObj.selectNode(2);
```

Ensure Visibility

To get the visibility status of the given TreeView node, you can use isVisible method.

You can ensure the specified tree node is visible by using ensureVisible method, which expands tree nodes and scrolls the TreeView control as necessary.

Also you can use getVisibleNodes method to get currently visible nodes in TreeView.

RAZOR

```
<script type="text/javascript">
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
function onClick() {
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//to ensure the visibility of node
treeObj.ensureVisible("4");
}
</script>
```

Checkboxes

TreeView consists of built-in checkbox option and it can be displayed to the left of the tree node by setting the **ShowCheckbox** property as true. It allows you to select more than one node at a time.

Indeterminate Checkboxes

TreeView supports tristate checkboxes in addition with standard two state checkboxes. By default Treeview has been enabled with tristate in checkboxes. You can enable 2-state checkboxes by using **AutoCheckParentNode** as false.

RAZOR

```
@*initialize and bind the TreeView with local data*@
@*shows each node with check boxes*@
@*enable 2-state check boxes*@
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.ShowCheckbox(true)
.AutoCheckParentNode(false)
)
```

Auto Checkable

By default checkbox state of child nodes depends up on parent node checkbox state and also parent node state gets updated based on child nodes state. You can turn off this option by setting **AutoCheck** as false to make independent parent and child nodes checkboxes.

RAZOR

```
@*initialize and bind the TreeView with local data*@
@*shows each node with check boxes*@
@*turns off auto check of parent and child nodes*@
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.ShowCheckbox(true)
.AutoCheck(false)
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

Check or Uncheck Node

Tree node can be checked or unchecked using checkNode and uncheckNode methods while showCheckbox property is enabled in TreeView. Also you can get or set the checked nodes of TreeView using checkedNodes property, which indicates the checked nodes index collection as array. The nodeCheck and nodeUncheck event occurs based on checkbox state.

You can use isNodeChecked method to check the particular TreeView node is checked or unchecked. Also you can use checkAll method to check all the nodes in TreeView.

JAVASCRIPT

```
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//to check node
treeObj.checkNode("2");
//to uncheck node
treeObj.uncheckNode("2");
```

Get Checked Nodes

To get checked nodes of TreeView, you can use getCheckedNodes method. It returns the collection of checked tree nodes. Also you can get currently checked nodes indexes in TreeView by using getCheckedNodesIndex method.

RAZOR

```
<script type="text/javascript">
function onClick() {
//create an instance from an existing TreeView.
// only after control creation you can get treeObj otherwise it throws
exception.
treeObj = $("#treeView").ejTreeView('instance');
//to get checked nodes
treeObj.getCheckedNodes();
}
</script>
```

Drag and Drop

To perform drag and drop operation in TreeView specify **AllowDragAndDrop** as true. It allows you to drag and drop node in all level of same TreeView.

The nodeDrag, nodeDragStart, nodeDragStop and

nodeDropped event occurs based on Treeview node drag and drop state.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
.AllowDragAndDrop(true)
)
```

Note: TreeView provides much easier option to drop the dragged nodes at any levels by indicator lines with icons.

Position Indicators

TreeView provides much easier option to drop the dragged nodes at any levels by indicator lines with icons such as line, plus/ minus and restrict icons while dragging and dropping the nodes. It represents exact position where the node to be dropped as sibling or child.

Indicators	description
Plus icon	represents the dragged node to be added as child of targeted node
Minus with restrict icon	represents the dragged node not to be dropped at the hovered region
In between icon	represents the dragged node to be dropped in between the nodes as siblings

Restriction

You can restrict the dragged nodes to be dropped at siblings or children's level by using **AllowDropSibling** and **AllowDropChild** properties.

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.AllowDragAndDrop(true)
.AllowDropSibling(true)
.AllowDropChild(false)
)
```

Drag and drop between Trees

You can drag and drop tree nodes between two TreeView by setting **AllowDragAndDrop** as true along with **AllowDragAndDropAcrossControl** as true.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> tree1 = new List<LoadData>();
List<LoadData> tree2 = new List<LoadData>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult TreeViewFeatures()
{
tree1.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
tree1.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
tree1.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
tree1.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
tree1.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
tree1.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
tree1.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
tree1.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
tree1.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
tree1.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource1 = tree1;
tree2.Add(new LoadData { Id = 11, Parent = 0, Text = "Item 5" });
tree2.Add(new LoadData { Id = 12, Parent = 0, Text = "Item 6" });
tree2.Add(new LoadData { Id = 13, Parent = 0, Text = "Item 7" });
tree2.Add(new LoadData { Id = 14, Parent = 0, Text = "Item 4" });
tree2.Add(new LoadData { Id = 15, Parent = 11, Text = "Item 5.1" });
tree2.Add(new LoadData { Id = 16, Parent = 11, Text = "Item 5.2" });
tree2.Add(new LoadData { Id = 17, Parent = 11, Text = "Item 5.3" });
tree2.Add(new LoadData { Id = 18, Parent = 13, Text = "Item 7.1" });
tree2.Add(new LoadData { Id = 19, Parent = 13, Text = "Item 7.2" });
tree2.Add(new LoadData { Id = 10, Parent = 15, Text = "Item 5.1.1" });
ViewBag.datasource2 = tree2;
return View();
}
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and specify the drag and drop settings.

RAZOR

```
@(Html.EJ().TreeView("treeViewDrag")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource1)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.AllowDragAndDrop(true)
.AllowDropSibling(true)
.AllowDropChild(true)
.AllowDragAndDropAcrossControl(true)
)
<br />
<br />
@(Html.EJ().TreeView("treeViewDrop")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource2)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.AllowDragAndDrop(true)
.AllowDropSibling(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.AllowDropChild(true)
.AllowDragAndDropAcrossControl(true)
)
```

Auto node structuring

You may not need to have two TreeView to be in same structured node while drag and drop the nodes between them. But after the node has been dropped, it should get structure of the TreeView node in which dropped. By default TreeView auto structure the node whenever you drop a node from different tree.

RAZOR

```
@(Html.EJ().TreeView("treeViewDrag")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource1)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.AllowDragAndDrop(true)
.AllowDropSibling(true)
.AllowDropChild(true)
.AllowDragAndDropAcrossControl(true)
.ShowCheckbox(true)
)
<br />
<br />
@(Html.EJ().TreeView("treeViewDrop")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource2)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.AllowDragAndDrop(true)
.AllowDropSibling(true)
.AllowDropChild(true)
.AllowDragAndDropAcrossControl(true)
)
```

Note: Auto node structure only applicable for well-structured node object.

Multiple Selection

TreeView supports to select the multiple nodes by specifying allowMultiSelection as true. It allows you to select more than one nodes in TreeView.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
// GET: Custom
List<LoadData> tree1 = new List<LoadData>();
public ActionResult Index()
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
tree1.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
tree1.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
tree1.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
tree1.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
tree1.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
tree1.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
tree1.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
tree1.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
tree1.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
tree1.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = tree1;
return View();
}
}
public class LoadData
{
public int Id { get; set; }
public int? Parent { get; set; }
public string Text { get; set; }
public bool? hasChild { get; set; }
public bool? expanded { get; set; }
public bool? ischecked { get; set; }
public bool? selected { get; set; }
public string spriteCss { get; set; }
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and specify the **AllowMultiSelection** property.

RAZOR

```
@Html.EJ().TreeView("treeMultiSelect").TreeViewFields(fields =>
fields.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
).AllowMultiSelection(true)
```

Select Nodes

To select more than one nodes of TreeView, you can use selectedNodes property. It will select the TreeView nodes from the given indexes.

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and we you can select the specific nodes in TreeView by using **selectedNodes** property.

RAZOR

```
@Html.EJ().TreeView("treeMultiSelect").TreeViewFields(fields =>
fields.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
).AllowMultiSelection(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

JS

```
$(function () {
// create instance for TreeView
var treeObj = $("#treeMultiSelect").data("ejTreeView");
treeObj.option("selectedNodes", [0, 5, 6]); //select the TreeView nodes
from the given indexes
});
```

Get Selected Nodes

To get the selected Nodes of TreeView, you can use getSelectedNodes method. It returns the collections of TreeView selected nodes.

You can use getSelectedNodesIndex method to get the index positions of currently selected nodes.

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and we you can get selected nodes from TreeView by using **getSelectedNodes** method.

RAZOR

```
@Html.EJ().TreeView("treeMultiSelect").TreeViewFields(fields =>
fields.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
).AllowMultiSelection(true)
```

JS

```
$(function () {
// create instance for TreeView
var treeObj = $("#treeMultiSelect").data("ejTreeView");
treeObj.getSelectedNodes(); //returns the collections of TreeView
selected nodes
});
```

Drag and Drop Multiple Nodes

TreeView supports to drag and drop multiple nodes by specifying allowMultiSelection as true along with allowDragAndDrop as true. It allows you to drag and drop multiple nodes in TreeView.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> tree1 = new List<LoadData>();
List<LoadData> tree2 = new List<LoadData>();
public ActionResult Index()
{
tree1.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
tree1.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
tree1.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
tree1.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
tree1.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
tree1.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
tree1.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
tree1.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
tree1.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
tree1.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource1 = tree1;
tree2.Add(new LoadData { Id = 11, Parent = 0, Text = "Item 5" });
tree2.Add(new LoadData { Id = 12, Parent = 0, Text = "Item 6" });
tree2.Add(new LoadData { Id = 13, Parent = 0, Text = "Item 7" });
tree2.Add(new LoadData { Id = 14, Parent = 0, Text = "Item 4" });
tree2.Add(new LoadData { Id = 15, Parent = 11, Text = "Item 5.1" });
tree2.Add(new LoadData { Id = 16, Parent = 11, Text = "Item 5.2" });
tree2.Add(new LoadData { Id = 17, Parent = 11, Text = "Item 5.3" });
tree2.Add(new LoadData { Id = 18, Parent = 13, Text = "Item 7.1" });
tree2.Add(new LoadData { Id = 19, Parent = 13, Text = "Item 7.2" });
tree2.Add(new LoadData { Id = 10, Parent = 15, Text = "Item 5.1.1" });
ViewBag.datasource2 = tree2;
return View();
}
}
public class LoadData
{
public int Id { get; set; }
public int? Parent { get; set; }
public string Text { get; set; }
public bool? hasChild { get; set; }
public bool? expanded { get; set; }
public bool? ischecked { get; set; }
public bool? selected { get; set; }
public string spriteCss { get; set; }
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and specify the multiple drag and drop settings.

RAZOR

```
@Html.EJ().TreeView("treeViewDrag").TreeViewFields(fields =>
fields.Datasource((IEnumerable<LoadData>)ViewBag.datasource1)
.Id("Id")
.ParentId("Parent")
.Text("Text")
).AllowMultiSelection(true).AllowDragAndDrop(true)
<br />
@Html.EJ().TreeView("treeViewDrop").TreeViewFields(fields =>
fields.Datasource((IEnumerable<LoadData>)ViewBag.datasource2)
.Id("Id")
.ParentId("Parent")
.Text("Text")
).AllowMultiSelection(true).AllowDragAndDrop(true)
```

Full Row Selection

The TreeView control provides the option to highlight a full row of tree view nodes. When **FullRowSelect** is **true**, a selection highlights the entire width of the tree view, display instead of the width of just the tree node text. It makes selection easier.

Copyright © 2001 - 2020 Syncfusion Inc.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
// GET: MultiSelection
List<loadOnDemand> fullRowData = new List<loadOnDemand>();
public ActionResult FullRowSelection()
{
fullRowData.Add(new loadOnDemand { id = 1, name = "Discover Music",
hasChild = true, expanded = true });
fullRowData.Add(new loadOnDemand { id = 2, parentId = 1, name = "Hot
Singles", selected = true });
fullRowData.Add(new loadOnDemand { id = 3, parentId = 1, name = "Rising
Artists" });
fullRowData.Add(new loadOnDemand { id = 4, parentId = 1, name = "Live
Music" });
fullRowData.Add(new loadOnDemand { id = 6, parentId = 1, name = "Best of
2013 So Far" });
fullRowData.Add(new loadOnDemand { id = 7, name = "Sales and Events",
hasChild = true, expanded = true });
fullRowData.Add(new loadOnDemand { id = 8, parentId = 7, name = "100
Albums - $5 Each" });
fullRowData.Add(new loadOnDemand { id = 9, parentId = 7, name = "Hip-Hop
and R&B Sale" });
fullRowData.Add(new loadOnDemand { id = 10, parentId = 7, name = "CD
Deals" });
fullRowData.Add(new loadOnDemand { id = 11, name = "Categories", hasChild
= true });
fullRowData.Add(new loadOnDemand { id = 12, parentId = 11, name = "Songs"
});
fullRowData.Add(new loadOnDemand { id = 13, parentId = 11, name =
"Bestselling Albums" });
fullRowData.Add(new loadOnDemand { id = 14, parentId = 11, name = "New
Releases" });
fullRowData.Add(new loadOnDemand { id = 15, parentId = 11, name =
"Bestselling Songs" });
fullRowData.Add(new loadOnDemand { id = 16, name = "MP3 Albums", hasChild
= true });
fullRowData.Add(new loadOnDemand { id = 17, parentId = 16, name = "Rock"
});
fullRowData.Add(new loadOnDemand { id = 18, parentId = 16, name =
"Gospel" });
fullRowData.Add(new loadOnDemand { id = 19, parentId = 16, name = "Latin
Music" });
fullRowData.Add(new loadOnDemand { id = 20, parentId = 16, name = "Jazz"
});
fullRowData.Add(new loadOnDemand { id = 21, name = "More in Music",
hasChild = true });
fullRowData.Add(new loadOnDemand { id = 22, parentId = 21, name = "Music
Trade-In" });
fullRowData.Add(new loadOnDemand { id = 23, parentId = 21, name = "Redeem
a Gift Card" });
fullRowData.Add(new loadOnDemand { id = 24, parentId = 21, name = "Band
T-Shirts" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
fullRowData.Add(new loadOnDemand { id = 25, parentId = 21, name = "Mobile
MVC" });
ViewBag.datasource = fullRowData;
return View();
}
}
public class loadOnDemand
{
public int id { get; set; }
public int? parentId { get; set; }
public string name { get; set; }
public bool? hasChild { get; set; }
public bool? expanded { get; set; }
public bool? ischecked { get; set; }
public bool? selected { get; set; }
public string spriteCss { get; set; }
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and specify the **FullRowSelect** property.

RAZOR

```
@Html.EJ().TreeView("fullRowTree").TreeViewFields(field =>
field.Datasource((IEnumerable<loadOnDemand>)ViewBag.datasource)
.Id("id").ParentId("parentId").Text("name")
.HasChild("hasChild").Expanded("expanded")
).FullRowSelect(true)
```

For more details about full row selection in TreeView, refer the sample here.

Customization

You can customize the TreeView full row selection for template support by using **CssClass** property. Based on this custom class, you can customize the height of the highlighting TreeView node.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
// GET: MultiSelection
List<loadOnDemand> fullRowData = new List<loadOnDemand>();
public ActionResult FullRowSelection()
{
fullRowData.Add(new loadOnDemand { id = 1, name = "Browsers", className =
"browser", hasChild = true, expanded = true });
fullRowData.Add(new loadOnDemand { id = 2, parentId = 1, name = "Internet
Explorer", className = "internet-explorer-browser", selected = true });
fullRowData.Add(new loadOnDemand { id = 3, parentId = 1, name = "Chrome",
className = "chrome-browser" });
fullRowData.Add(new loadOnDemand { id = 4, parentId = 1, name =
"Firefox", className = "firefox-browser" });
fullRowData.Add(new loadOnDemand { id = 6, parentId = 1, name = "Bitty",
className = "bitty-browser" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
fullRowData.Add(new loadOnDemand { id = 7, parentId = 1, name = "Opera",
className = "opera-browser" });
ViewBag.datasource = fullRowData;
return View();
}
}
public class loadOnDemand
{
public int id { get; set; }
public int? parentId { get; set; }
public string name { get; set; }
public bool? hasChild { get; set; }
public bool? expanded { get; set; }
public bool? ischecked { get; set; }
public bool? selected { get; set; }
public string spriteCss { get; set; }
public string className { get; set; }
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source and specify the **CssClass** property.

RAZOR

```
@Html.EJ().TreeView("fullRowTree").TreeViewFields(field =>
field.Datasource((IEnumerable<loadOnDemand>)ViewBag.datasource)
.Id("id").ParentId("parentId").Text("name")
.HasChild("hasChild").Expanded("expanded")
).FullRowSelect(true).CssClass("custom").Template("#treeTemplate")
<script id="treeTemplate" type="text/x-jsrender">
{{"{"}}if !hasChild{{}}}}
<span class="con-image {{"{"}}>className{{}}}}"></span>
{{"{"}}/if{{}}}}
{{"{"}}>name{{}}}}
</script>
```

CSS

```
<style>
.custom .con-image {
background-image:
url("http://mvc.syncfusion.com/demos/web/images/toolbar/browserl.png");
background-repeat: no-repeat;
height: 32px;
width: 32px;
display: inline-block;
overflow: hidden;
background-repeat: no-repeat;
text-align: center;
vertical-align: middle;
}
.custom .e-li-active > .e-text-wrap .con-image {
background-image:
url("http://mvc.syncfusion.com/demos/web/images/toolbar/browserh.png");
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.custom .e-li-hover > .e-text-wrap .con-image, .e-fullrow-wrap .e-li-
focus > .e-text-wrap .con-image {
background-image:
url("http://mvc.syncfusion.com/demos/web/images/toolbar/browserl.png");
}
.custom .internet-explorer-browser {
background-position: -84px 0px;
}
.custom .chrome-browser {
background-position: -42px 0px;
}
.custom .firefox-browser {
background-position: 0px 0px;
}
.custom .bitty-browser {
background-position: -126px 0px;
}
.custom .opera-browser {
background-position: -168px 0px;
}
/*customize the height of highlighting TreeView node*/
.custom.e-fullrow-wrap .e-item ul .e-fullrow {
margin-top: -36px;
height: 36px;
}
</style>
```

Template Option

Tree nodes can be customized by using **Template** property. Treeview template option requires addition JS library called **JsRender**, which helps to create the structured way of tree nodes with simple codes and increased performance. To know more about JsRender - http://www.jsviews.com/ .

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> treeData = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
treeData.Add(new LoadData {
Id = 1,
Name = "UK",
ClassName = "uk-style",
HasChild = true,
Expanded = true
});
treeData.Add(new LoadData {
Id = 2,
PId = 1,
ImgId = "1",
Name = "Steven John",
City = "London",
Phone = "555-5665-2323"
});
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
treeData.Add(new LoadData {
Id = 3,
Name = "USA",
ClassName = "usa-style",
HasChild = true,
Expanded = true
});
treeData.Add(new LoadData {
Id = 5,
PId = 3,
ImgId = "3",
Name = "Andrew",
City = "Capital way",
Phone = "934-8374-2455"
});
treeData.Add(new LoadData {
Id = 4,
PId = 3,
ImgId = "2",
Name = "Angelica",
City = "Dayton",
Phone = "988-4243-0806"
});
ViewBag.datasource = treeData;
return View();
}
}
```

In the view page, specify template format and add TreeView helper.

RAZOR

```
<script id="treeTemplate" type="text/x-jsrender">
{{"{{"}}if HasChild{{}}}}
<div class={{"{{"}}>ClassName{{}}}}>{{"{{"}}>Name{{}}}}</div>
{{"{{"}}else{{}}}}
<div class="cont-list">
<img class="con-image"
src="http://mvc.syncfusion.com/demos/web/images/treeview/template-image-
{{"{{"}}>ImgId{{}}}}.png" />
<div class="cont-delete"></div>
<b>{{"{{"}}>Name{{}}}}</b><br />
<span>{{"{{"}}>City{{}}}}</span>
<br />
<span>{{"{{"}}>Phone{{}}}}</span>
</div>
<div class="treeFooter"></div>
{{"{{"}}/if{{}}}}
</script>
<div style="width: 250px">
@(Html.EJ().TreeView("treeview")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("PId")
.Text("Name")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.HasChild("HasChild")
.Expanded("Expanded")
)
.Template("#treeTemplate")
.ClientSideEvents(events =>
events.Create("onCreate")
)
)
</div>
<style class="cssStyles">
.e-treeview .e-node-hover, .e-treeview .e-active {
background: transparent;
border-color: transparent;
}
.e-treeview .e-node-hover {
opacity: 0.8;
}
.con-image {
float: left;
}
.cont-list {
background: none repeat scroll 0 0 white;
border: 1px solid #BBBCBB;
height: 85px;
width: 200px;
color: #5c5c5c;
line-height: 17px;
}
.cont-details {
margin-top: 12px;
font-size: 13px;
}
.uk-style {
background-color: #74A247 !important;
color: #FFFFFF !important;
}
.usa-style {
background-color: #12A99A !important;
color: #FFFFFF !important;
}
.cont-delete {
background-image:
url("http://mvc.syncfusion.com/demos/web/images/treeview/remove-
icon.png");
background-position: -6px -10px;
background-repeat: no-repeat;
float: right;
height: 16px;
width: 16px;
cursor: pointer;
}
.cont-list .treeFooter {
height: 5px;
width: 100%;
background-color: gray;
margin-top: 17px;
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</style>
<script>
function onCreate(args) {
var treeObj = $("#treeview").data("ejTreeView");
$("#treeview").find(".cont-delete").bind("click", function (e) {
e.preventDefault();
treeObj.removeNode($(e.target).parents("li").first());
});
}
</script>
```

Custom action in nodes

You can able to perform custom action in TreeView template node. You can able to perform node delete operation while clicking delete icon by using removeNode method TreeView as shown in below code example.

JAVASCRIPT

```
function onCreate(args) {
var treeObj = $("#treeview").data("ejTreeView");
$("#treeview").find(".cont-delete").bind("click", function (e) {
e.preventDefault();
treeObj.removeNode($(e.target).parents("li").first());
});
}
```

State Persistence

TreeView state can be persisted by using '**EnablePersistence**'. In which user intractable model values only has been persisted in order to maintain performance.

The model values of below are maintained through id basis of tree node.

- selected
- checked
- expanded/ collapsed state

Note: In "UL" "li" template state has been persisted by index.

TreeView stores its model in local storage / cookies of browser before page refreshes and reinitialized with their stored model after refresh.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> treeData = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
treeData.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1" });
treeData.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
treeData.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
treeData.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
treeData.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
treeData.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
treeData.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
treeData.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
treeData.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
treeData.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = treeData;
return View();
}
}
```

In the view page, add TreeView helper with following properties.

RAZOR

```
@(Html.EJ().TreeView("treeViewDrag")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.EnablePersistence(true)
)
```

Right to Left Align

TreeView supports right to left align and it can be applied by specifying **EnableRTL** as true.

RAZOR

```
@(Html.EJ().TreeView("treeViewDrag")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
)
.EnableRTL(true)
)
```

How To

Update the modified data from tree to database.

TreeView allows us to get the updated tree data after performing such operation like node editing, drag and drop, add and remove node. Using getTreeData method you can get the updated tree data.

Refer the following code block to know how to get updated data from TreeView.

In the controller page, create a data list that contains the details about tree nodes and specify the method to store modified data.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult TreeViewFeatures()
{
var storedData = System.Web.HttpContext.Current.Session ["modifiedData"];
if (storedData == null)
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1", Expanded =
true });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
load.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
load.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
load.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = load;
}
else
ViewBag.datasource = storedData;
return View();
}
public ActionResult StoreData(List<LoadData> data)
{
System.Web.HttpContext.Current.Session ["modifiedData"] = data;
return View("TreeViewFeatures");
}
}
```

In the view page, add TreeView helper and specify the scripts for sending modified data through AJAX action with that also specify clear cache option.

RAZOR

```
@using TreeView_Doc.Models
<div style="width: 250px">
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
)
.ClientSideEvents(events =>
events.Create("onCreate")
)
)
@(Html.EJ().Button("move")
.Text("Move Node")
.Size(ButtonSize.Normal)
.ClientSideEvents(events =>
events.Click("moveNode")
)
)
@(Html.EJ().Button("clear")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Text("Clear Cache")
.Size(ButtonSize.Normal)
.ClientSideEvents(events =>
events.Click("clearCache")
)
)
</div>
<script type="text/javascript">
var treeObj;
function onCreate() {
treeObj = $("#treeView").data('ejTreeView');
}
function moveNode() {
treeObj.moveNode($('#4'), $('#1'), 11);
var updatedData = treeObj.getTreeData();
ajaxRequest(updatedData);
}
function ajaxRequest(updatedData) {
$.ajax({
url: "StoreData",
data: JSON.stringify({ data: updatedData }),
type: "POST",
contentType: "application/json",
dataType: "JSON"
});
}
function clearCache() {
treeObj.destroy();
ajaxRequest();
}
</script>
```

You can also get the updated data source for remote data binding after performing the operation like editing, selecting/unselecting, expanding/collapsing, checking/unchecking and removing node. You cannot get the updated data source when you perform operation like drag and drop, adding node for remote data binding.

The updated data source also contains custom attributes ("ContactTitle", "OrderID", "EmployeeID", "Freight") if you return these from server.

Refer the following code block to know more about how to get updated data with custom attributes from TreeView for remote data binding.

In the view page, add TreeView helper and specify the scripts for getting updated data source.

RAZOR

```
<button id="btn1"
onclick="getSelectedNodeObject()">GetSelectedNodeObject</button>
@Html.EJ().TreeView("tree1").TreeViewFields(s => s.Datasource(data =>
data.URL("//js.syncfusion.com/ejServices/Wcf/Northwind.svc/").CrossDomain
(true))
.Query("ej.Query().from('Orders').select('CustomerID,OrderID,EmployeeID,F
reight').take(3)").Id("CustomerID").Text("CustomerID")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Child(c => c.Datasource(childData =>
childData.URL("//js.syncfusion.com/ejServices/Wcf/Northwind.svc/").CrossD
omain(true))
.Query("ej.Query().from('Customers').select('CustomerID,ContactTitle,Cont
actName,Country')").Id("Country").ParentId("CustomerID").Text("ContactNam
e")))
<script>
function getSelectedNodeObject() {
var treeObj = $("#tree1").data("ejTreeView");
var nodeId = treeObj.getSelectedNode().attr("id"); //get selected node id
if (nodeId != null) {
var node = treeObj.getTreeData(nodeId); //get the given node object
if (node[0].OrderID != null)
alert("The OrderID of node '" + node[0].CustomerID + "' is: '" +
node[0].OrderID + "'");
else
alert("The ContactTitle of node '" + node[0].ContactName + "' is: '" +
node[0].ContactTitle + "'");
} else
alert("Select any node");
}
</script>
```

TreeView context menu to process node operations.

TreeView control is availed with the context menu options that open on right-click, over the node. Other than the default menu items available, you can add the new node dynamically in TreeView and also delete the item, enable and disable the item in TreeView. It is achieved by adding the Context Menu option to the TreeView.

Menu Item

By default, the context menu options are provided with four items namely: Add New Item, Delete Item, Enable Item and Disable Item. When you want to customize and use your own custom menu items, then you can customize the TreeView with the desired collections.

The following code example illustrates how to configure the context menu elements for the TreeView and in the following example, you have to specify the menu type as ej.MenuType.ContextMenu and in the menuClick function, you can check the cases with add, delete, remove or enable item in TreeView.

And each functionality in the context menu option is done by specific methods. For example, you have added the new item in TreeView by using the addNode() method, delete the item using removeNode() method, disable the item using disableNode() method and enable the item enableNode() method respectively.

The following steps explain how you can enable the **ShowCheckbox** property for TreeView.

In the view page, add TreeView helper and specify context menu.

RAZOR

```
<div style="width: 250px">
@Html.EJ().TreeView("treeview").Items(items =>
{
items.Add().Text("Favorites").Expanded(true).Children(child =>
{
child.Add().Text("Desktop");
```

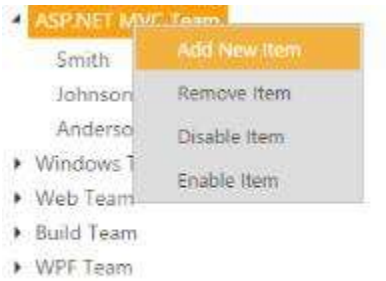
Copyright © 2001 - 2020 Syncfusion Inc.

```
child.Add().Text("Downloads");
child.Add().Text("Recent places");
});
items.Add().Text("Libraries").Expanded(true).Children(child =>
{
child.Add().Text("Documents").Children(child1 =>
{
child1.Add().Text("My Documents");
child1.Add().Text("Public Documents");
});
child.Add().Text("Pictures").Children(child1 =>
{
child1.Add().Text("My Pictures");
child1.Add().Text("Public Pictures");
});
child.Add().Text("Music").Children(child1 =>
{
child1.Add().Text("My Music");
child1.Add().Text("Public Music");
});
child.Add().Text("Subversion");
});
items.Add().Text("Computer").Children(child =>
{
child.Add().Text("Folder(C)");
child.Add().Text("Folder(D)");
child.Add().Text("Folder(E)");
});
})
</div>
<div>
@(Html.EJ().Menu("treeviewMenu")
.Items(items =>
{
items.Add().Text("Add New Item");
items.Add().Text("Remove Item");
items.Add().Text("Disable Item");
items.Add().Text("Enable Item");
})
.OpenOnClick(false)
.MenuType(MenuType.ContextMenu)
.ShowSubLevelArrows(true)
.ContextMenuTarget("#treeview")
.ClientSideEvents(events =>
events.Click("menuClick")
.BeforeOpen("beforeOpen")
)
)
</div>
<script type="text/javascript">
//Define the events for menu click in the script section as follows,
var tabIndex = 1, treeviewObj, selectedNode;
function beforeOpen(args) {
var treeviewObj = $("#treeview").data("ejTreeView");
if (!$(args.target).hasClass("e-text"))
args.cancel = true;
else {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
selectedNode = args.target;
treeviewObj.selectNode(selectedNode);
}
}
function menuClick(args) {
var treeviewObj = $("#treeview").data("ejTreeView");
if (args.events.text == "Add New Item") {
treeviewObj.addNode("Item" + tabIndex, selectedNode);
tabIndex++;
}
else if (args.events.text == "Remove Item") {
treeviewObj.removeNode(selectedNode);
}
else if (args.events.text == "Disable Item") {
treeviewObj.disableNode(selectedNode)
}
else if (args.events.text == "Enable Item") {
treeviewObj.enableNode(selectedNode)
}
}
</script>
```

The output for the context menu for TreeView control is as follows.



Sorted data using refresh method

TreeView allows you to refresh the entire tree data by using refresh method. Refer the below code block to know how to sort entire tree data using refresh method.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1", Expanded =
true });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
load.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
load.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
load.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = load;
return View();
}
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source with that specify the scripts for sorting and refreshing tree data.

RAZOR

```
<div style="width: 250px">
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
)
)
@(Html.EJ().Button("move")
.Text("Sort Countries")
.Size(ButtonSize.Normal)
.ClientSideEvents(events =>
events.Click("sortCountries")
)
)
</div>
<script type="text/javascript">
function sortCountries() {
window.tree = $("#treeView").data("ejTreeView");
TreeNodeData = sortResults(tree.model.fields.dataSource, "Text", true);
tree.model.fields.dataSource = TreeNodeData;
tree.refresh();
}
//Sorting the tree data based on ascending order
function sortResults(data, fieldName, ascOrder) {
return data.sort(function (a, b) {
if (ascOrder) return (a[fieldName] > b[fieldName]);
else return (b[fieldName] > a[fieldName]);
});
}
</script>
```

Persist updated data after edit, add and remove node

TreeView allows us to persist the updated data after performing some tree operations like node add and delete. Refer the following code block to know how to persist updated tree data after refresh.

The nodeAdd, nodeCut, nodeDelete and nodePaste events occurs based on Treeview node manipulation. The beforeAdd,

Copyright © 2001 - 2020 Syncfusion Inc.

beforeCut, beforeDelete and beforePaste events are triggered before the TreeView component node manipulation.

In the controller page, create a data list that contains the details about tree nodes and specify the method to store modified data.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
var storedData = System.Web.HttpContext.Current.Session["modifiedData"];
if (storedData == null)
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1", Expanded =
true });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
load.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
load.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
load.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = load;
}
else
ViewBag.datasource = storedData;
return View();
}
public ActionResult StoreData(List<LoadData> data)
{
System.Web.HttpContext.Current.Session["modifiedData"] = data;
return View("TreeViewFeatures");
}
}
```

In the view page, add TreeView helper and specify the scripts for sending modified data through AJAX action.

RAZOR

```
@using TreeView_Doc.Models
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
)
.ClientSideEvents(events =>
events.Create("onCreate")
.NodeAdd("updateData")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.NodeEdit("updateData")
.NodeDelete("updateData")
)
)
@(Html.EJ().Button("clear")
.Text("Clear Cache")
.Size(ButtonSize.Normal)
.ClientSideEvents(events =>
events.Click("clearCache")
)
)
@(Html.EJ().Button("add")
.Text("Add Node")
.Size(ButtonSize.Normal)
.ClientSideEvents(events =>
events.Click("addNode")
)
)
@(Html.EJ().Button("remove")
.Text("Remove Node")
.Size(ButtonSize.Normal)
.ClientSideEvents(events =>
events.Click("removeNode")
)
)
<script type="text/javascript">
var treeObj;
function onCreate() {
treeObj = $("#treeView").data('ejTreeView');
}
function updateData() {
var updatedData = treeObj.getTreeData();
ajaxRequest(updatedData);
}
function addNode() {
treeObj.addNode({ Id: 11, Text: "NewNode1" });
}
function removeNode() {
treeObj.removeNode();
}
function ajaxRequest(updatedData) {
$.ajax({
url: "StoreData",
data: JSON.stringify({ data: updatedData }),
type: "POST",
contentType: "application/json",
dataType: "JSON"
});
}
function clearCache() {
treeObj.destroy();
ajaxRequest();
}
</script>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Filtering nodes in TreeView

You can able to filter TreeView nodes based on node text. Refer the below code blocks to filter tree nodes based on the node text.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1", Expanded =
true });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
load.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
load.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
load.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = load;
return View();
}
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source with that specify the script for filtering tree nodes.

RAZOR

```
<div style="margin: 20px; padding: 20px; height: 2000px;">
<div class="heading">
Treeview with Filtering Nodes
</div>
<div class="treeviewfilter">
<!-- TextBox Element -->
@(Html.EJ().MaskEdit("inputBox")
.Width("100%")
.InputMode(InputMode.Text)
.MaskFormat("")
.ClientSideEvents(events =>
events.Change("searchNodes")
)
)
<!-- TreeView Element -->
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.Width("100%")
.ClientSideEvents(events =>
events.Create("onCreate")
)
)
</div>
</div>
<script type="text/javascript">
var treeObj, inputObj, count = 0;
window._temp = [];
function onCreate(args) {
// Creating Objects for TreeView & MaskEdit TextBox
treeObj = $("#treeView").data("ejTreeView");
inputObj = $("#inputBox").data("ejMaskEdit");
}
//Function triggers while change the TextBox input value and start new
search
function searchNodes(args) {
var treeElement = treeObj.element;
// Getting the Input String
var searchVal = inputObj.get_StrippedValue() ?
inputObj.get_StrippedValue().toLowerCase() : "";
// If the string length is greater than 0 then Do searching.
if (searchVal.length > 0) {
var matchedNode = [], otherNodes = [];
// Searching Anchor nodes
var linkNodes = treeElement.find('ul>li>div>a');
for (var p = 0; p < linkNodes.length; p++) {
if ($(linkNodes[p]).text().toLowerCase().indexOf(searchVal) != -1)
matchedNode.push(linkNodes[p]);     //  Stores the Matched nodes
else
otherNodes.push(linkNodes[p]);      // Stores the unMatched nodes
}
// Find the Matched nodes parent Li element and set Display "block"
var resultNodes =
treeElement.find(matchedNode).closest("li").css("display", "block");
// Find the unMatched nodes parent Li element and set Display "none"
treeElement.find(otherNodes).closest("li").css("display", "none");
for (var i = 0; i < resultNodes.length; i++) {
var currentNode = $(resultNodes[i]);
// Get the Parent Li element of current li element and Set Display
"block"
var parentNode = currentNode.parents('ul').closest('li').css("display",
"block");
if (parentNode.length > 0) {
// Call expandNode function If a parentNode has children
treeObj.expandNode(parentNode);
if
(treeObj.model.expandedNodes.indexOf($(treeObj._liList).index(parentNode)
) == -1)
window._temp.push(parentNode);
}
var childrenUl = currentNode.children('ul');
if (childrenUl.length > 0 && childrenUl.children('li:visible').length ==
0) {
currentNode.children('ul').children('li').css("display", "block");
// Call expandNode function If a resultNodes[i] has children
```

```
treeObj.expandNode(resultNodes[i]);
if
(treeObj.model.expandedNodes.indexOf($(treeObj._liList).index(resultNodes
[i])) == -1)
window._temp.push(resultNodes[i]);
}
}
}
else {
// window._temp contains Li elements, which is expanded while searching.
treeElement.find('ul>li').css("display", "block");
for (var i = 0; i < window._temp.length; i++) {
treeObj._collpaseNode(window._temp[i]);
}
window._temp = [];
}
}
</script>
<style type="text/css">
.treeviewfilter {
width: 200px;
min-height: 300px;
max-height: auto;
border: 1px solid #bbbbbb;
margin: 0 auto;
}
.e-mask .e-in-wrap {
border: none;
border-bottom: 1px solid #bbbbbb;
}
.heading {
font-size: 22px;
margin: 0 auto;
width: 300px;
margin-bottom: 10px;
}
</style>
```

AngularJS data binding to update data while add and remove node

TreeView allows us to bind and update tree data in mapped data component while add and removing node using AngularJS binding. Refer the below code block to know how to update data using AngularJS binding.

RAZOR

```
<div class="content-container-fluid" ng-controller="TreeCtrl">
<div class="row">
<div class="cols-sample-area">
<div style="width: 400px; float:left">
<button type="button" ng-click="add()">Add</button>
<button type="button" ng-click="remove()">Remove</button>
<div id="treeView" e-allowediting="true" ej-treeview e-fields-
datasource="dataList" e-fields-id="iconId" e-fields-parentid="parentId"
e-fields-text="name" e-fields-haschild="hasChild" e-fields-
expanded="expanded" e-showcheckbox="true" e-allowediting="true"></div>
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div style="width: 400px; float:left">
<table>
<tr>
<td>
<p>direct to $scope.dataList</p>
<div ng-repeat="icon in dataList">
<input type="text" ng-model="icon.name" class="sampleText" /><button
type="button" ng-click="delete(icon.iconId)">Delete</button><br />
</div>
</td>
</tr>
</table>
</div>
</div>
</div>
</div>
<script>
var phones = [
{ iconId: 1, name: "Fiction Book Lists", hasChild: true, expanded: true
},
{ iconId: 2, parentId: 1, name: "To Kill a Mockingbird " },
{ iconId: 3, parentId: 1, name: "Pride and Prejudice " },
{ iconId: 4, parentId: 1, name: "Harry Potter and the Sorcerer's Stone"
},
{ iconId: 5, parentId: 1, name: "The Hobbit " },
{ iconId: 6, name: "Mystery Book Lists", hasChild: true, expanded: true
},
{ iconId: 7, parentId: 6, name: "And Then There Were None " },
{ iconId: 8, parentId: 6, name: "Angels & Demons" },
{ iconId: 9, parentId: 6, name: "In Cold Blood " },
{ iconId: 10, parentId: 6, name: "The Name of the Rose " },
{ iconId: 11, name: "Horror Novels", hasChild: true },
{ iconId: 12, parentId: 11, name: "The Shining (The Shining, #1) " },
{ iconId: 13, parentId: 11, name: "The Haunting of Hill House " },
{ iconId: 14, parentId: 11, name: "The Silence of the Lambs (Hannibal
Lecter, #2) " },
{ iconId: 15, name: "Novel Lists", hasChild: true },
{ iconId: 16, parentId: 15, name: "Shadow Hills (Shadow Hills, #1) " },
{ iconId: 17, parentId: 15, name: "After Forever Ends " },
{ iconId: 18, parentId: 15, name: "Angel Star" },
{ iconId: 19, parentId: 15, name: "Raised by Wolves" },
{ iconId: 20, parentId: 15, name: "Falling From Grace" }];
var Nodes = ["1", "2"];
angular.module('treeApp', ['ejangular']).controller('TreeCtrl', function
($scope) {
$scope.dataList = phones;
$scope.add = function () {
var treeObj = $("#treeView").data("ejTreeView"), obj;
//obj = [{ iconId: 21, parentId: 1, name: "Newnode" }];
obj = "New TextNode";
treeObj.addNode(obj);    // Two way binding when add node
};
$scope.remove = function () {
var treeObj = $("#treeView").data("ejTreeView");
var id = treeObj.getSelectedNode()[0].id;
treeObj.removeNode(treeObj.getSelectedNode());  // Two way binding when
remove node
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
};
// Delete from outside tree
$scope.delete = function (iconId) {
for (var i = $scope.dataList.length - 1; i > -1; i--) {
if ($scope.dataList[i].iconId === iconId)
$scope.dataList.splice(i, 1); // Delete parent node
}
for (var i = $scope.dataList.length - 1; i > -1; i--) {
if ($scope.dataList[i].parentId === iconId)
$scope.dataList.splice(i, 1); // Delete child nodes
}
};
});
</script>
```

Set tooltip for TreeView nodes

TreeView allows you to set tooltip option to TreeView nodes using fields.linkAttribute property of TreeView. Refer the below code block to know how to set tooltip for TreeView nodes.

In the model page, add a class and define the properties as shown below.

C#

```
namespace TreeView_Doc.Models
{
public class LoadData
{
public int Id { get; set; }
public int Parent { get; set; }
public string Text { get; set; }
public bool Expanded { get; set; }
public object LinkAttribute { get; set; }
}
public class LinkAttribute
{
public string Title { get; set; }
}
}
```

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData
{
Id = 1,
Parent = 0,
Text = "Item 1",
Expanded = true,
LinkAttribute = new LinkAttribute()
{
```

```
Title = "First Item"
}
});
load.Add(new LoadData
{
Id = 2,
Parent = 0,
Text = "Item 2",
LinkAttribute = new LinkAttribute()
{
Title = "Second Item"
}
});
load.Add(new LoadData
{
Id = 3,
Parent = 0,
Text = "Item 3"
});
load.Add(new LoadData
{
Id = 4,
Parent = 0,
Text = "Item 4"
});
load.Add(new LoadData
{
Id = 5,
Parent = 1,
Text = "Item 1.1"
});
load.Add(new LoadData
{
Id = 6,
Parent = 1,
Text = "Item 1.2"
});
load.Add(new LoadData
{
Id = 7,
Parent = 1,
Text = "Item 1.3"
});
load.Add(new LoadData
{
Id = 8,
Parent = 3,
Text = "Item 3.1"
});
load.Add(new LoadData
{
Id = 9,
Parent = 3,
Text = "Item 3.2"
});
load.Add(new LoadData
{
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
Id = 10,
Parent = 5,
Text = "Item 1.1.1"
});
ViewBag.datasource = load;
return View();
}
}
```

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source

RAZOR

```
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
.LinkAttribute("LinkAttribute")
)
)
```

Auto hide/show the expand/collapse icon of TreeView

You can able to display expand icon on mouse entering into TreeView and hide while leaving from the TreeView. Refer the below code block to know how to hide/ show the expand/collapse icons automatically based on mouse position.

In the controller page, create a data list that contains the details about tree nodes.

C#

```
public partial class TreeViewController : Controller
{
List<LoadData> load = new List<LoadData>();
public ActionResult TreeViewFeatures()
{
load.Add(new LoadData { Id = 1, Parent = 0, Text = "Item 1", Expanded =
true });
load.Add(new LoadData { Id = 2, Parent = 0, Text = "Item 3" });
load.Add(new LoadData { Id = 3, Parent = 0, Text = "Item 2" });
load.Add(new LoadData { Id = 4, Parent = 0, Text = "Item 4" });
load.Add(new LoadData { Id = 5, Parent = 1, Text = "Item 1.1" });
load.Add(new LoadData { Id = 6, Parent = 1, Text = "Item 1.2" });
load.Add(new LoadData { Id = 7, Parent = 1, Text = "Item 1.3" });
load.Add(new LoadData { Id = 8, Parent = 3, Text = "Item 3.1" });
load.Add(new LoadData { Id = 9, Parent = 3, Text = "Item 3.2" });
load.Add(new LoadData { Id = 10, Parent = 5, Text = "Item 1.1.1" });
ViewBag.datasource = load;
return View();
}
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source with that specify the scripts to hide and show the icons of tree nodes.

RAZOR

```
<div style="width: 150px">
<!-- TreeView Element -->
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
)
.ClientSideEvents(events =>
events.Create("onCreate")
)
)
</div>
<script type="text/javascript">
function onCreate(args) {
this.element.find(".e-item > div > .e-plus, .e-minus").css("visibility",
"hidden");
var treeObj = $("#treeView").data("ejTreeView");
$("#treeView").on("mouseenter", function () {
treeObj.element.find(".e-item > div > .e-plus, .e-
minus").css("visibility", "");
});
$("#treeView").on("mouseleave", function () {
treeObj.element.find(".e-item > div > .e-plus, .e-
minus").css("visibility", "hidden");
});
}
</script>
```

Customize the expand/collapse icons of TreeView

You can able to customize the TreeView expand and collapse icon by using “cssClass” property of TreeView. Refer the below code block to know how to customize the expand/collapse icons.

In the view page, add TreeView helper and map the properties defined in to the corresponding fields in data source with that specify the styles for customizing expand/ collapse icon of TreeView.

RAZOR

```
<div style="width: 150px">
<!-- TreeView Element -->
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id")
.ParentId("Parent")
.Text("Text")
.Expanded("Expanded")
)
.CssClass("customize")
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
)
</div>
<style type="text/css">
.customize .e-icon {
width: 16px;
height: 16px;
background-repeat: no-repeat;
background-image:
url("http://jsplayground.syncfusion.com/13.3.0.7/themes/web/images/ui-
icons/ui-icons-default.png");
}
.customize .e-icon.e-plus:before, .customize .e-icon.e-minus:before {
content: "";
}
.customize .e-icon.e-plus {
background-position: -80px -46px;
}
.customize .e-icon.e-minus {
background-position: -132px -46px;
}
</style>
```

In form post, how to get all checked nodes details of TreeView in server end

You have to create a hidden element for storing custom details of checked nodes and add it to the form element. These details will be sent to server at the time of form submit.

Refer the following code block to know how to get checked nodes details in server end

In the view page, add TreeView helper and specify the scripts for adding checked node details in form element.

RAZOR

```
<div style="width: 250px">
<form action="\TreeView\CheckedNodes" method="post" id="myForm">
@Html.EJ().TreeView("tree").Items(items =>
{
items.Add().Text("Item 1").Children(child =>
{
child.Add().Text("Item 1.1").Children(child1 =>
{
child1.Add().Text("Item 1.1.1");
});
});
items.Add().Text("Item 2").Expanded(true).Children(child =>
{
child.Add().Text("Item 2.1");
child.Add().Text("Item 2.2");
});
items.Add().Text("Item 3").Children(child =>
{
child.Add().Text("Item 3.1");
});
}).ShowCheckbox(true)
<input type="submit" id="submit" value="GetCheckedNodes" />
</form>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</div>
<script>
$('#submit').click(function () {
tree = $("#tree").data("ejTreeView");
checkedNodes = tree.getCheckedNodes();
for (i = 0; i < checkedNodes.length ; i++) {
var checkedNode = checkedNodes[i];
var nodeData = tree.getNode(checkedNode);
var parentNode = tree.getParent(checkedNode);
if (parentNode.length > 0)
$('#myForm').append('<input type="hidden" name="checkedNodes" value="' +
parentNode[0].id + '/' + nodeData.id + '" />');
else
$('#myForm').append('<input type="hidden" name="checkedNodes" value=" No
Parent /' + nodeData.id + '" />');
}
});
</script>
```

In the controller page, configure following settings to get the checked node details in form submit.

C#

```
public partial class TreeViewController : Controller
{
public ActionResult TreeViewFeatures()
{
return View();
}
// on data all checkedNodes are received
public ActionResult CheckedNodes(List<string> checkedNodes)
{
ViewBag.CheckedNodes = checkedNodes;
return PartialView();
}
}
```

Using following partial view page, you can display the custom details of checked nodes.

RAZOR

```
<html>
<body>
<h3>Checked node ID</h3>
<ul>
@*using ViewBag, we can get the checked node details*@
@if (ViewBag.CheckedNodes != null)
{
foreach (var nodes in ViewBag.CheckedNodes)
{
<li>@nodes</li>
}
}
</ul>
</body>
</html>
```

Copyright © 2001 - 2020 Syncfusion Inc.

Populate data for TreeView using AJAX

TreeView provides AJAX action supports to populate the data for rendering TreeView nodes. Refer the following code blocks to achieve this behavior.

In the model page, specify the TreeView node properties as shown below.

C#

```
public class TreeViewData
{
public int id { get; set; }
public int parentId { get; set; }
public string name { get; set; }
public bool hasChild { get; set; }
public bool expanded { get; set; }
}
```

In the controller page, create a data list which contains the details about tree nodes and specify AJAX action handling method.

C#

```
public partial class TreeViewController : Controller
{
public ActionResult TreeViewFeatures()
{
return View();
}
List<TreeViewData> m_templateData;
public List<TreeViewData> TemplateData
{
get
{
if (m_templateData == null)
InitializeTemplateData();
return m_templateData;
}
}
private void InitializeTemplateData()
{
m_templateData = new List<TreeViewData>();
m_templateData.Add(new TreeViewData
{
id = 1,
parentId = 0,
name = "Local Disk(C:)",
hasChild = true
});
m_templateData.Add(new TreeViewData
{
id = 2,
parentId = 1,
name = "Folder 1"
});
m_templateData.Add(new TreeViewData
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
id = 3,
parentId = 1,
name = "Folder 2"
});
m_templateData.Add(new TreeViewData
{
id = 4,
parentId = 1,
name = "Folder 3",
hasChild = true
});
m_templateData.Add(new TreeViewData
{
id = 20,
parentId = 4,
name = "File 1"
});
m_templateData.Add(new TreeViewData
{
id = 21,
parentId = 4,
name = "File 2"
});
m_templateData.Add(new TreeViewData
{
id = 22,
parentId = 4,
name = "File 3"
});
m_templateData.Add(new TreeViewData
{
id = 5,
parentId = 0,
name = "Local Disk(D:)",
hasChild = true
});
m_templateData.Add(new TreeViewData
{
id = 6,
parentId = 5,
name = "Folder 4",
hasChild = true
});
m_templateData.Add(new TreeViewData
{
id = 7,
parentId = 6,
name = "File 4"
});
m_templateData.Add(new TreeViewData
{
id = 8,
parentId = 6,
name = "File 5"
});
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
public ActionResult GetAllData()
{
return Json(TemplateData, JsonRequestBehavior.AllowGet);
}
}
```

In the view page, add TreeView helper and map the properties defined to the corresponding fields in data source.

RAZOR

```
@using TreeView_Doc.Models
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource(data =>
data.URL(@Url.Content("GetAllData"))
)
.Id("id")
.ParentId("parentId")
.Text("name")
.HasChild("hasChild")
.Expanded("expanded")
)
)
```

Populate data for TreeView using XML data source

TreeView provides XML data binding support to populate the data for rendering TreeView nodes, so that values can be mapped to the TreeView fields from an existing XML data source as below.

RAZOR

```
@* In the View page, add TreeView helper and map the properties defined
to the corresponding fields in data source.*@
@(Html.EJ().TreeView("treeView")
.TreeViewFields(field =>
field.Datasource((IEnumerable<FileList>)ViewBag.datasource)
.Id("Name")
.ParentId("Parent")
.Text("Name")
)
)
```

C#

```
//In controller part, need to convert XML data into List of Objects for
specifying the list data in TreeView DataSource.
public partial class TreeViewController : Controller
{
public ActionResult TreeViewFeatures()
{
XElement element =
XElement.Load(Server.MapPath("~/App_Data/XMLFile1.xml"));
List<FileList> fileList = ToFileList(element);
ViewBag.datasource = fileList;
return View();
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
//Convert XML element to List of Objects
private List<FileList> ToFileList(XElement element)
{
if (element != null && element.HasElements)
{
List<FileList> employees = new List<FileList>();
AddChildElements(element, employees);
return employees;
}
return null;
}
private FileList ToFileItem(XElement element, string parentId)
{
return new FileList()
{
Name = element.Attribute("Text").Value,
Parent = parentId
};
}
private void AddChildElements(XElement element, List<FileList> employees)
{
foreach (var child in element.Elements("RootItem"))
{
var employee = ToFileItem(child, element.Attribute("Text").Value);
employees.Add(employee);
AddChildElements(child, employees);
}
foreach (var child in element.Elements("Item"))
{
var employee = ToFileItem(child, element.Attribute("Text").Value);
employees.Add(employee);
}
}
}
```

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<items Text="0">
<RootItem Text="Favorites" Url="#">
<Item Text="Desktop" Url="#"></Item>
<Item Text="Downloads" Url="#"></Item>
<Item Text="Recent places" Url="#"></Item>
</RootItem>
<RootItem Text="Libraries" Url="#">
<RootItem Text="Documents" Url="#">
<Item Text="My Documents" Url="#"></Item>
<Item Text="Public Documents" Url="#"></Item>
</RootItem>
<RootItem Text="Pictures" Url="#">
<Item Text="My Pictures" Url="#"></Item>
<Item Text="Public Pictures" Url="#"></Item>
</RootItem>
<RootItem Text="Music" Url="#">
<Item Text="My Music" Url="#"></Item>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<Item Text="Public Music" Url="#"></Item>
</RootItem>
</RootItem>
<RootItem Text="Computer" Expanded="True" Url="#">
<Item Text="Folder(C)" Url="#"></Item>
<Item Text="Folder(D)" Url="#"></Item>
<Item Text="Folder(F)" Url="#"></Item>
</RootItem>
</items>
```

UploadBox

UploadBox

The UploadBox control supports uploading files into the designated server, regardless of the file format and size. The UploadBox control helps you with the selection of files to upload to the server. The UploadBox control has theme support.

Key Features

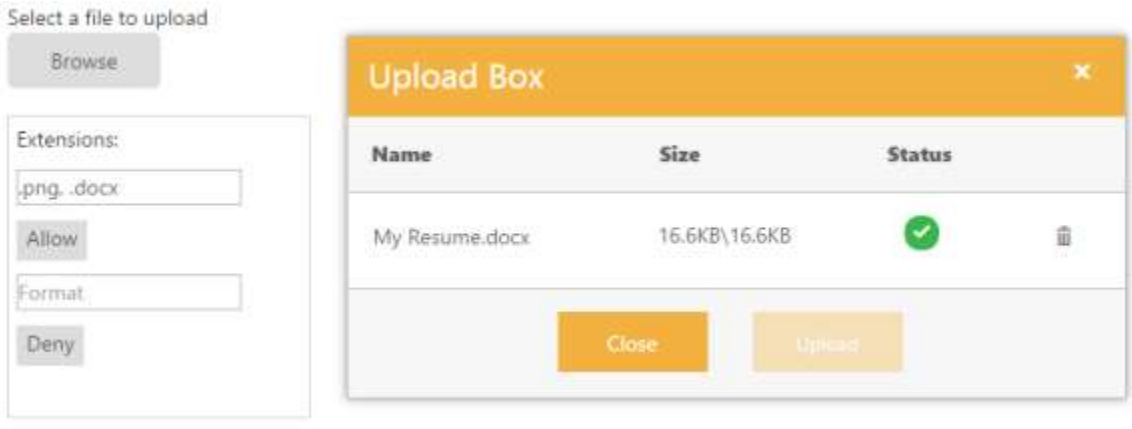
- Modern look: Rich appearance with theme support.
- RTL: Supports right-to-left alignment.
- Auto Upload: Support for manual and automatic uploading of files are included.
- Sync/Async Upload: Supports synchronous and asynchronous uploading of files.
- Multiple files: Supports uploading multiple files.

Getting Started

This section explains briefly about how to create an UploadBox in ASP.NET MVC application .

Create your first UploadBox in MVC

ASP.NET MVC UploadBox provides support to upload the files or photos within your web page. From the following guidelines, you can learn how to upload the file that is used in a Resume Upload scenario. This helps you to restrict some file extensions, while uploading the resume in the server by using UploadBox control. The following screenshot demonstrates the functionality of UploadBox with file extension.



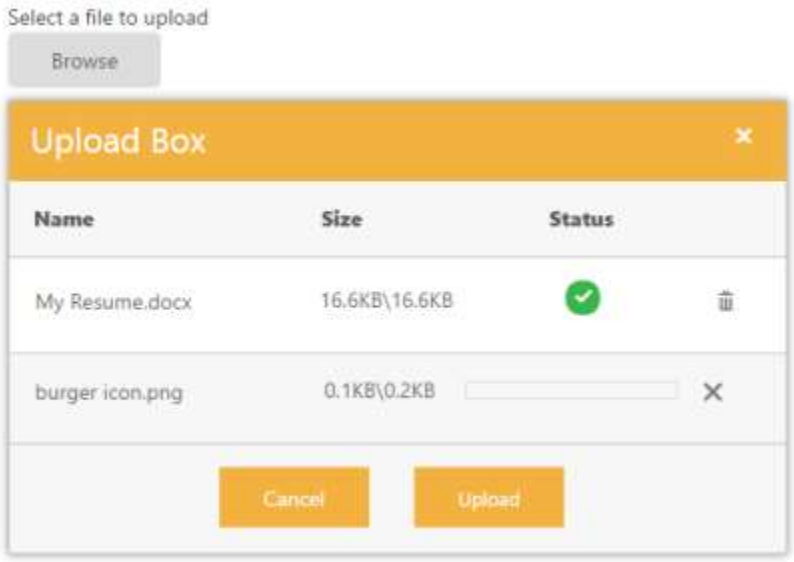
In the above screenshot, you can upload a resume. It allows .png and .docx file extension. This enables you to avoid unsupported resume formats to be uploaded in the server.

Copyright © 2001 - 2020 Syncfusion Inc.

Create UploadBox widgets

ASP.NET MVC UploadBox basically renders built-in features like upload multiple files, and deletes the files from Uploadbox. You can know the status of the file whether it is complete or fail and also can retry the upload if it failed. You can easily create the UploadBox widget by using the following steps.

You can create an MVC project and add necessary assemblies, styles, and scripts with the help of the given MVC-Getting Started Documentation.



Add the following code example to the corresponding view page to render the UploadBox.

ASPX-CS

```
@Html.EJ().Uploadbox("UploadDefault").SaveUrl(@Url.Action("SaveDefault"))
.RemoveUrl(@Url.Action("RemoveDefault"))
```

Add the following ActionResult in your controller.

C#

```
public ActionResult SaveDefault(IEnumerable<HttpPostedFileBase>
UploadDefault)
{
foreach (var file in UploadDefault)
{
var fileName = Path.GetFileName(file.FileName);
var destinationPath = Path.Combine(Server.MapPath("~/App_Data"),
fileName);
file.SaveAs(destinationPath);
}
return Content("");
}
```

Add the following Remove ActionResult in your controller page.

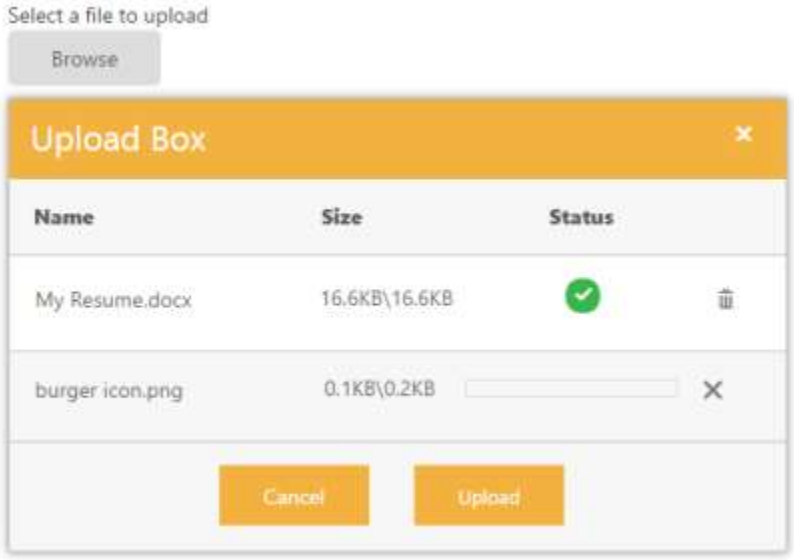
C#

```
public ActionResult RemoveDefault(string[] fileNames)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
foreach (var fullName in fileNames)
{
var fileName = Path.GetFileName(fullName);
var physicalPath = Path.Combine(Server.MapPath("~/App_Data"), fileName);
if (System.IO.File.Exists(physicalPath))
{
System.IO.File.Delete(physicalPath);
}
}
return Content("");
}
```

Execute the project to render the following output for the given steps. The file is being uploaded.



UploadBox after uploading

Set Restriction for File Extension

In a real-time scenario, some file extensions are restricted. You can either allow files or restrict files by using **extensionsAllow** and **extensionsDeny** respectively in **Uploadbox**.

Note: The SaveUrl and RemoveUrl are the same as above

Add the following elements in the view page.

ASPX-CS

```
<div id="targetElement">
<table id="uploadTable">
<tr>
<td>
Extensions:
</td>
<td></td>
</tr>
<tr>
<td>
```

```
<input type="text" id="Allow" class="ejinputtext" placeholder="Format" />
@(Html.EJ().Button("upbutton1").Text("Allow").ClientSideEvents(c =>
c.Click("allow")))
</td>
<td></td>
</tr>
<tr>
<td>
<input type="text" id="Deny" class="ejinputtext" placeholder="Format" />
@(Html.EJ().Button("upbutton2").Text("Deny").ClientSideEvents(c =>
c.Click("deny")))
</td>
<td>
@Html.EJ().Uploadbox("UploadDefault").SaveUrl(@Url.Action("SaveDefault"))
.RemoveUrl(@Url.Action("RemoveDefault")))
</td>
</tr>
</table>
</div>
```

Add the following code example in script section.

JS

```
var object;
$(function () {
object = $("#UploadDefault").data("ejUploadbox");
$("#upbutton1").ejButton({
click: "allow",
});
$("#upbutton2").ejButton({
click: "deny",
});
});
function allow() {
object.option('extensionsAllow', $("#Allow").val());
object.option('extensionsDeny', "");
}
function deny() {
object.option('extensionsAllow', "");
object.option('extensionsDeny', $("#Deny").val());
}
```

Add the given styles to display the **Uploadbox** with margin alignments.

CSS

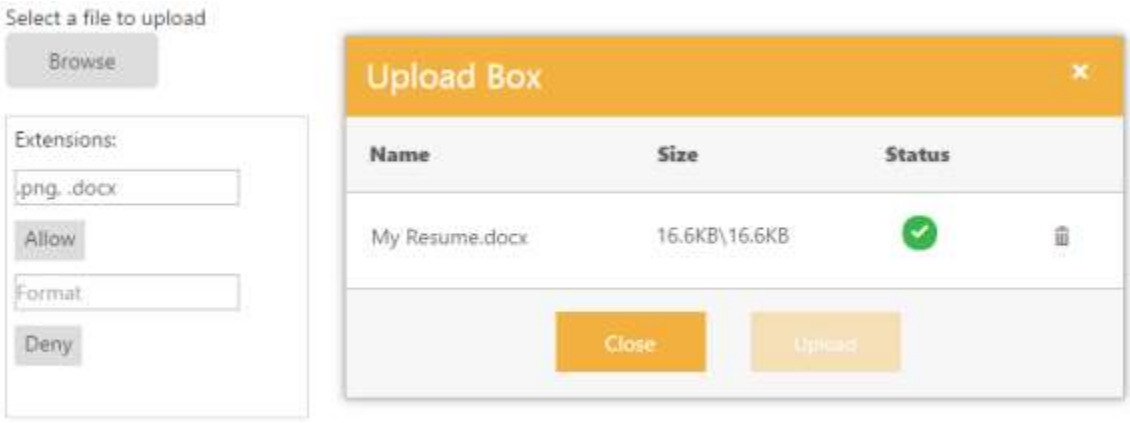
```
#targetElement {
width: 520px;
height: 500px;
margin: 0 auto;
}
#UploadDefault {
float: right;
}
#uploadTable {
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
width: 100%;
}
#Allow, #Deny {
width: 150px;
height: 20px;
padding: 5px;
}
```

Note: You can restrict one or more different extension files by giving file extension separated by comma (,)

The following screenshot displays an **Uploadbox** control with the file extension.



The above screenshot shows the **Uploadbox** that allows ".**docx**" and "**.png"** files.

Upload Multiple Files

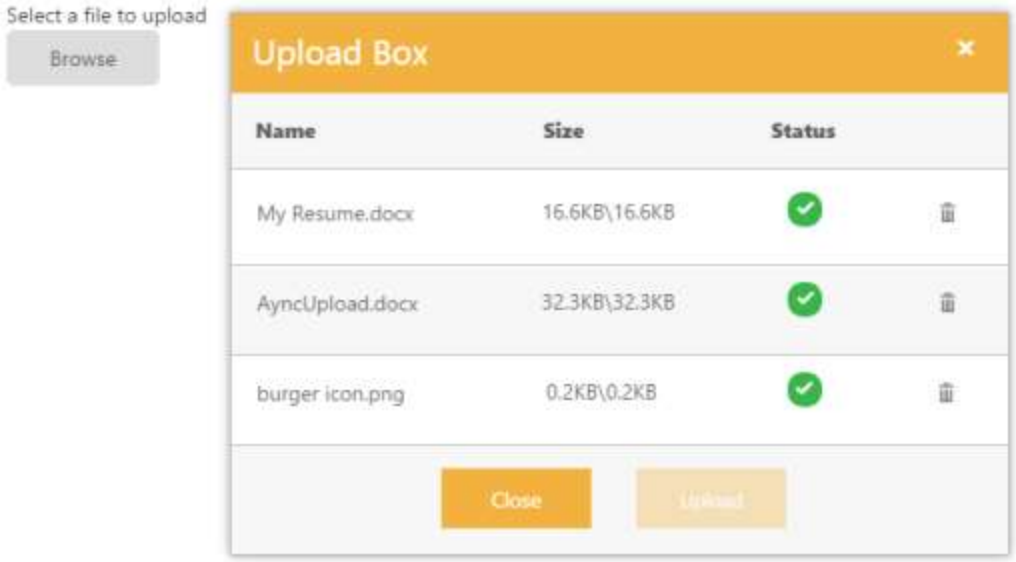
To upload multiple files in UploadBox control, click the Browse button and select multiple files from file explorer window

The following screenshot displays an UploadBox control with multiple files selected.

ASPX-CS

```
@Html.EJ().Uploadbox("UploadDefault").SaveUrl(@Url.Action("Save")).Remove
Url(@Url.Action("Remove")).MultipleFilesSelection(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.



Display uploaded File Details

To display the details of the uploaded files below the control, add an container element to display the details below the uploadbox control. Using the ClientSide success event we can access the details of the uploaded file and display it in the container element. Refer the following code.

ASPX-CS

```
<div class="frame">
<div class="control">
Select a file to upload
<div class="posupload">
@Html.EJ().Uploadbox("UploadDefault").SaveUrl("SaveDefault").RemoveUrl("R
emoveDefault").AutoUpload(true).UploadBoxDialogAction(act =>
act.CloseOnComplete(true)).ClientSideEvents(e => e.Success("onSuccess"))
</div>
</div>
<br />
<ul id="uploaded" ></ul>
</div>
```

JS

```
function onSuccess(args) {
var msg = "<li class='green'><span style='display:inline-block;'><span
class='e-icon e-file-empty'></span></span><span style='display:inline-
block;margin-left:20px;'><span style='display:block'>" + args.files.name
+ "</span><span style='display:block;font-size:
10px;'>"+(args.files.size/1024).toFixed(2)+"KB</span></li>"
$("#uploaded").append(msg);
}
```

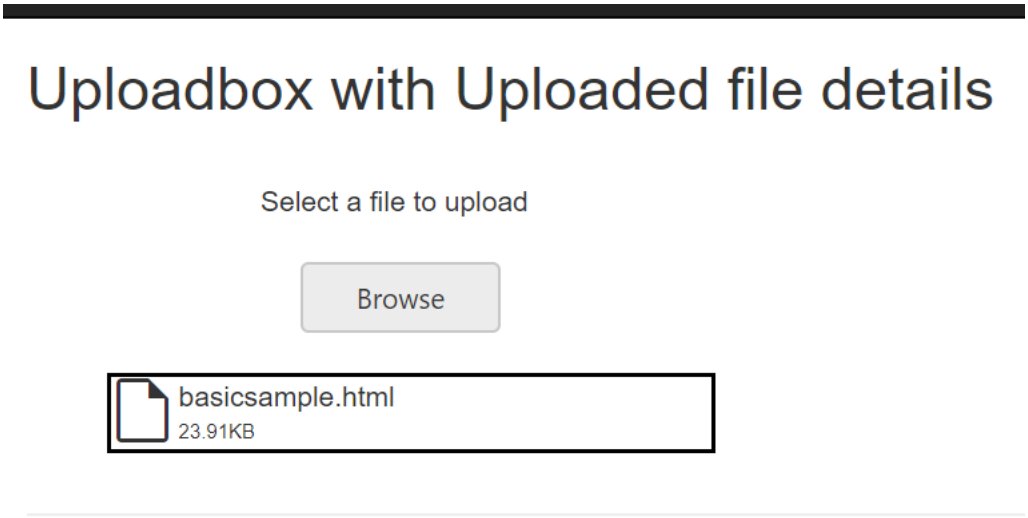
CSS

```
#uploaded li {
list-style: none;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
border: 2px solid black;
width:300px;;
}
.e-icon.e-file-empty:before {
font-size: 32px;
}
```

The following screenshot displays an UploadBox control with file details of the uploaded file.



[Sample link]

File Actions

Save File Action

To save the uploaded file in MVC, create an ActionResult and trigger the same in SaveUrl property. In ActionResult, save and specify the target location as App_Data folder for the uploaded files. The data type is string.

The following steps explain the configuration of SaveUrl property in the UploadBox.

1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

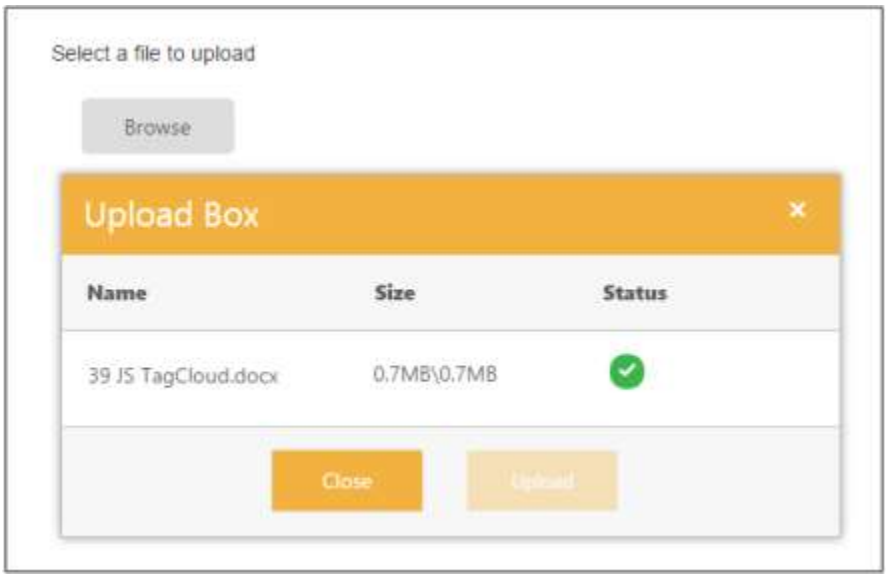
```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox")
.SaveUrl("Uploadbox/Save")
.RemoveUrl("Uploadbox/Remove")
```

CSHARP

```
// Configure the ActionResult to save the file in App_Data location. In
the following code example, the uploaded files is saved in App_Data
folder.
public ActionResult Save(IEnumerable<HttpPostedFileBase> uploadbox)
{
foreach (var file in uploadbox)
```

```
{
var fileName = Path.GetFileName(file.FileName);
var destinationPath = Path.Combine(Server.MapPath("~/App_Data"),
fileName);
file.SaveAs(destinationPath);
}
return Content("");
}
```

2. The following screenshot displays the output.



Remove File Action

To remove the uploaded file in MVC, create ActionResult and trigger the same in RemoveUrl property. This action removes the uploaded file from where it is saved. This is achieved by clicking remove button on upload dialog. The data type is string.

The following steps explain the configuration of RemoveUrl property in UploadBox.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox")
.SaveUrl("Uploadbox/Save")
.RemoveUrl("Uploadbox/Remove")
```

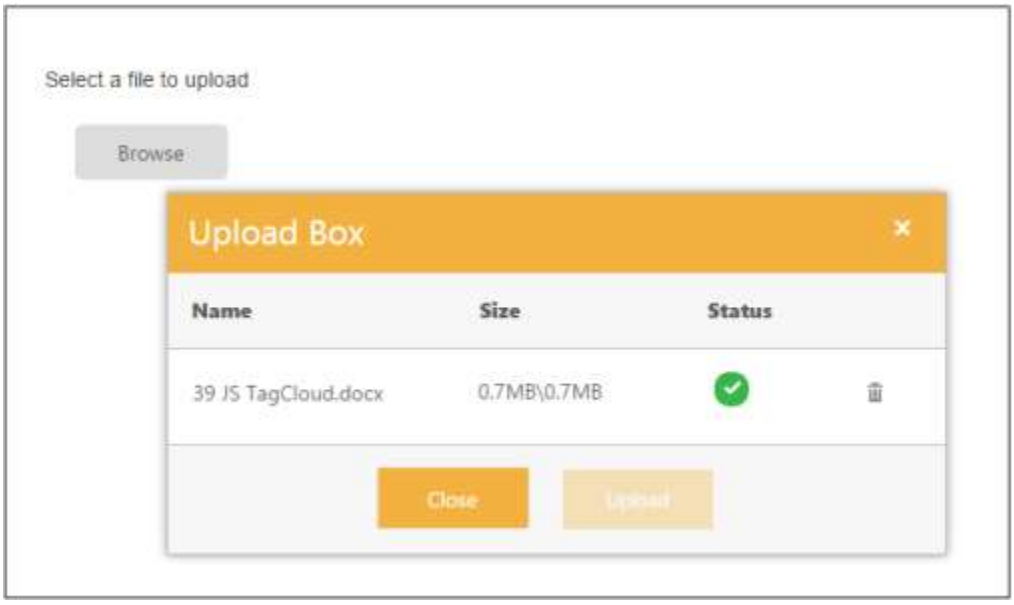
C#

```
// Configure the action to remove the file in target location. From that
location, the file is searched and removed from the App_Data folder.
public ActionResult Remove(string[] fileNames)
{
foreach (var fullName in fileNames)
{
var fileName = Path.GetFileName(fullName);
var physicalPath = Path.Combine(Server.MapPath("~/App_Data"), fileName);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
if (System.IO.File.Exists(physicalPath))
{
System.IO.File.Delete(physicalPath);
}
}
return Content("");
}
```

The following screenshot displays the output.



AutoUpload

The UploadBox widget provides support to upload the file automatically once file is selected by using Browse button, that is, without clicking upload button. To achieve this, set the AutoUpload property to ‘true’. The data type is Boolean. By default, the value is set to ‘false’, so Upload button is clicked to upload the files.

The following steps explain the configuration of AutoUpload property in UploadBox

1. In the VIEW page, add the below script to configure the UploadBox element.

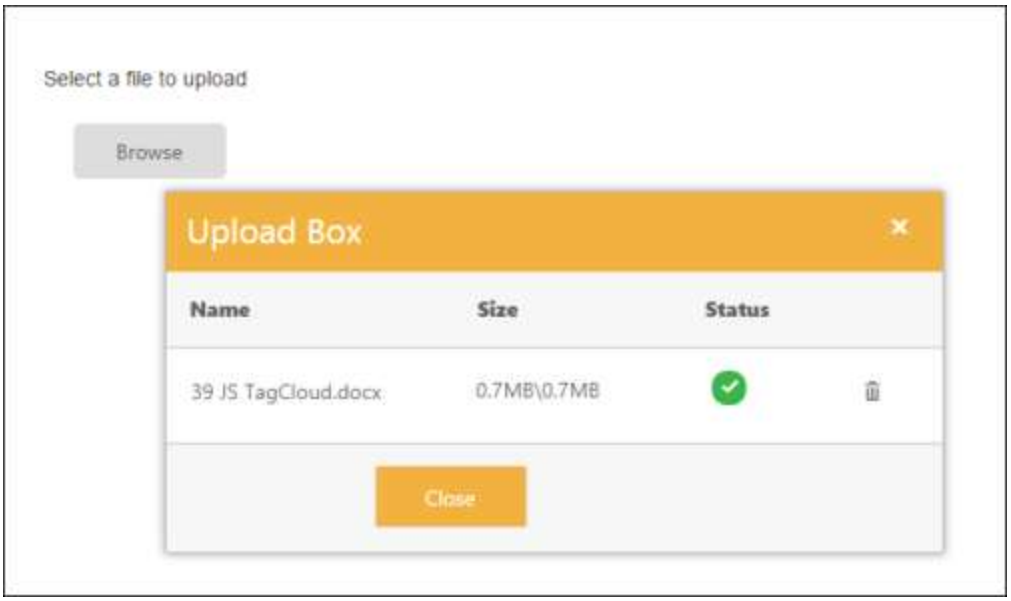
ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").AutoUpload(true)
```

2. For MVC, configure SaveActionResult and RemoveActionResult files as mentioned in Save file action and Remove file action respectively.

The following screenshot displays the output.

Copyright © 2001 - 2020 Syncfusion Inc.



Restricting uploading files based on its extension

Allow Extension

Files are filtered before they are uploaded. You can select the files to be filtered by using browse button. The ExtensionsAllow property allows upload of the selected extensions only. You can give multiple extensions by using comma (,). The data type is string.

Note: Prepend dot (.) symbol with extension like ".pdf".

The following steps explain the configuration of ExtensionsAllow property in UploadBox.

1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").ExtensionsAllow(".docx,.pdf")
```

Deny Extension

Files are filtered before they are uploaded. You can select the files to be filtered by using browse button. The ExtensionsDeny property denies upload of the selected extensions. You can give multiple extensions by using comma (,). The data type is string.

Note: Prepend dot (.) symbol with extension like ".pdf".

The following steps explain the configuration of ExtensionsDeny property in UploadBox

1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add UploadBox ox element.
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").ExtensionsDeny(".docx,.pdf")
```

Note: When **ExtensionsDeny** and **ExtensionsAllow** properties have same file extension, the extension will be allowed.

File Size

Maximum File Size for UploadBox

In the UploadBox control, you can browse files with the size going up to gigabytes. You can restrict the files from being browsed using the FileSize property. When you do not use this property, it takes a default size, 31457280B, that is, 31MB. When this size exceeds, we cannot browse the file.

1. Add the following code example to the corresponding View page to render the UploadBox control with the customized file size.

ASPX-CS

```
<div class="control">
@Html.EJ().Uploadbox("UploadBox")
.SaveUrl("Uploadbox/Save")
.RemoveUrl("UploadBox/Remove")
.FileSize(848576)
.ClientSideEvents(e => e.Error("error"))
</div>
<script type="text/javascript">
function error(e)
{
alert(e.error);
}
</script>
```

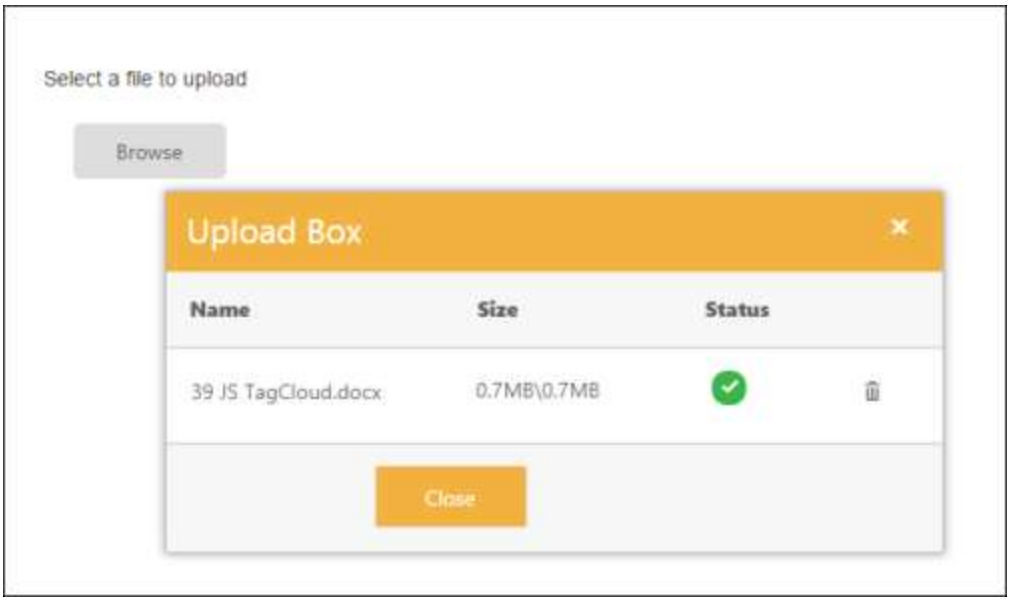
To know about file action, we need to refer link:

http://help.syncfusion.com/aspnetmvc/uploadbox/file-size

The following screenshot displays UploadBox control with customized file size.

When you want to browse the file within the fileSize, you can browse and upload the files.

Copyright © 2001 - 2020 Syncfusion Inc.

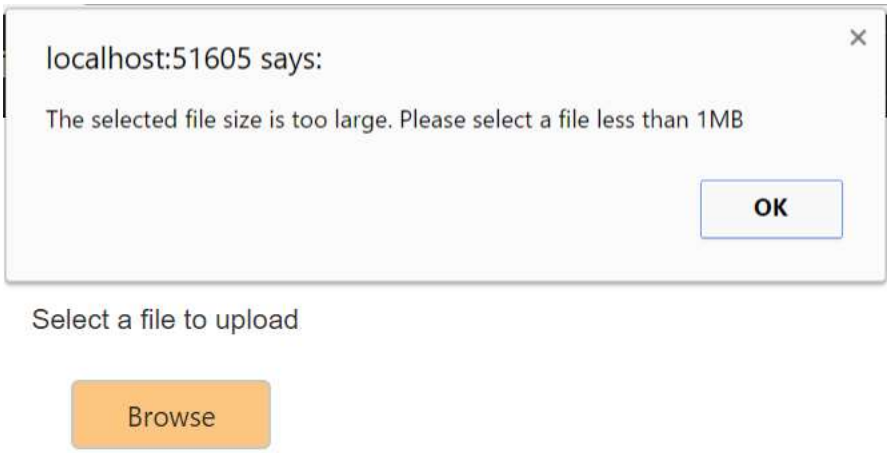


When you try to browse the file with exceeded FileSize, an error message will be displayed as an alert.You can customize the error message by converting the file size into mb and display the corresponding value in error message.

ASPX-CS

```
<script type="text/javascript">
function fileUploadError(e)
{
alert("The selected file size is too large. Please select a file less
than " + Math.round(e.model.fileSize / (1024 * 1024)) + "MB");
}
</script>
```

Now you will get the error message as shown in the below image



Note: By default, UploadBox will display the file size as bytes value in error message.

Copyright © 2001 - 2020 Syncfusion Inc.

Enable or Disable the UploadBox

The Enabled property is Boolean type, which allow us to set the enable or disable the UploadBox. By default value of Enabled property is true.

The following steps explain the configuration of Enabled property in UploadBox.

1. In the VIEW page, add below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").Enabled(false)
```

2. The following screenshot displays the output.

Asynchronous Upload

The AsyncUpload property is Boolean type, which allow us to upload and remove files asynchronously.When multiple files are chosen in Asynchronous upload,files will be uploaded one by one to the server.User interaction with the page will not be interrupted at the time of upload.User can also remove the file even after uploading.This is the best way of file upload when compared to synchronous so by default UploadBox works with asynchronous upload option only.To achieve this, set the AsyncUpload property to ‘true’. The default value of AsyncUpload property is ‘true’.

The following steps guide you in uploading the file asynchronously.

1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the view page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").AsyncUpload(true)
```

Synchronous Upload

This features allow you to upload and remove the files synchronously.When multiple files are chosen in Synchronous upload,all files will be uploaded only on form submission.Multitasking is not possible here.To achieve this, set the AsyncUpload property to ‘false’. The data type is Boolean.

Note: By default, UploadBox widget works with asynchronous upload option only.

The following steps guide you in uploading the file synchronously.

1. In the VIEW page, create a form with action and post method and then add the script into the form to configure the UploadBox element.

Copyright © 2001 - 2020 Syncfusion Inc.

ASPX-CS

```
// In the CSHTML page, create a form with action and post method and then
add the UploadBox element.
@using (Html.BeginForm("Index","Home",FormMethod.Post))
{
@Html.EJ().Uploadbox("uploadbox").AsyncUpload(false)
<input type="submit" value="Submit" />
}
```

Once the form is submitted by using Submit button, it triggers the post method and the files can be saved by using the below code.

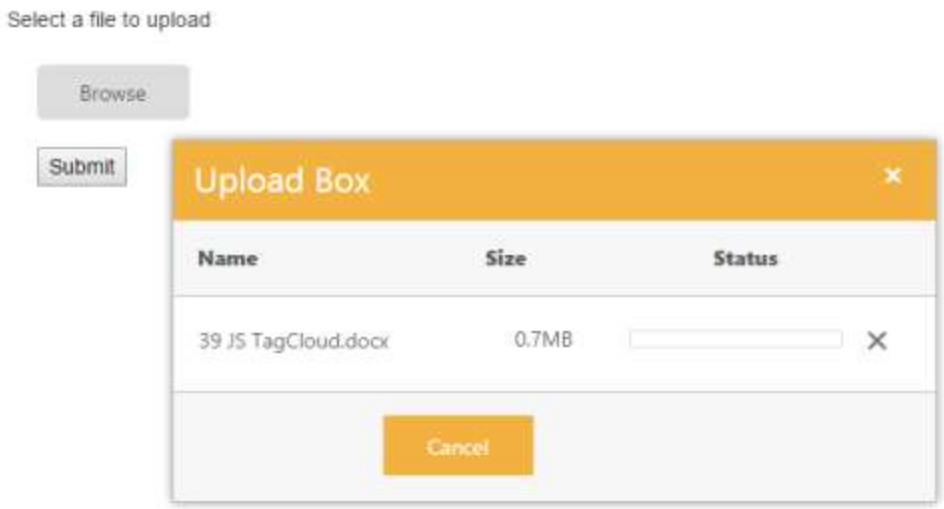
C#

```
public ActionResult Index()
{
if (string.IsNullOrEmpty((string)TempData["status"]))
ViewBag.status = string.Empty;
else
ViewBag.status = (string)TempData["status"];
return View();
}
[HttpPost]
public ActionResult Index(IEnumerable<HttpPostedFileBase> uploadbox)
{
if (uploadbox != null)
{
if (uploadbox != null)
{
foreach (var file in uploadbox)
{
var fileName = Path.GetFileName(file.FileName);
var destinationPath = Path.Combine(Server.MapPath("~/App_Data"),
fileName);
file.SaveAs(destinationPath);
}
TempData["status"] = "Successfully Uploaded";
return RedirectToAction("FileUploadFeatures");
}
else
{
TempData["status"] = "Select a file to upload";
return RedirectToAction("FileUploadFeatures");
}
}
return View();
}
```

Refer here for online demo

The following screenshot displays the output.

Copyright © 2001 - 2020 Syncfusion Inc.



Multiple files upload

The UploadBox widget provides support to upload multiple files spontaneously. The MultipleFilesSelection property enables you to select multiple files while browsing. To achieve this, set the MultipleFilesSelection property to 'true'. The data type is Boolean.

Note: The Multiple file selection supports all the latest versions of browser except Internet Explorer 9 and its previous versions.

The following steps explain the configuration of MultipleFilesSelection property in UploadBox.

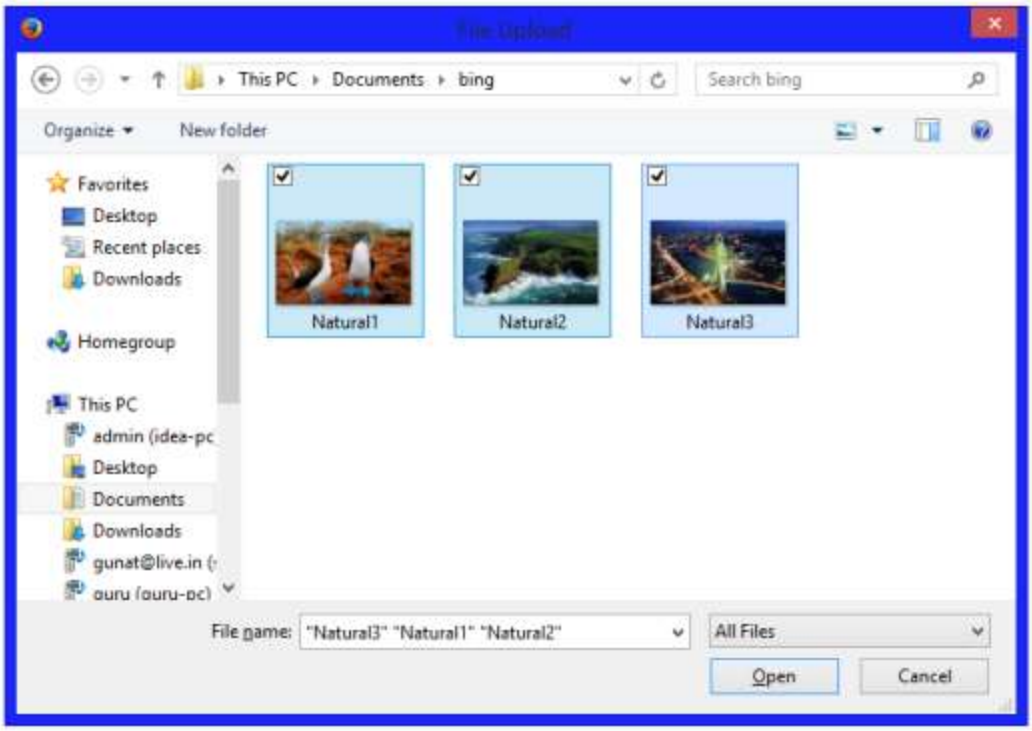
1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").MultipleFilesSelection(true)
```

The following screenshot displays the output.

Copyright © 2001 - 2020 Syncfusion Inc.





Drag and Drop Support

The UploadBox control provides the drag and drop support. You can simply drag-and-drop files, directly from the computer and can be dropped into the droppable area. A list of files can be dragged and dropped when you enable the MultipleFilesSelection property.

The following screenshot displays the drag and drop support.

Copyright © 2001 - 2020 Syncfusion Inc.



Enable drag and drop

AllowDragAndDrop property is Boolean type which allow us to enable or disable the Drag and Drop. When you want to drag and drop multiple files, you can enable multiple file selection by setting MultipleFilesSelection as true in theUploadBox control. By default the AllowDragAndDrop property is set as false in the UploadBox control.

The following steps explain how to enable the drag and drop in the UploadBox control.

1. In the VIEW page, add the below script to enable the drag and drop in UploadBox control.

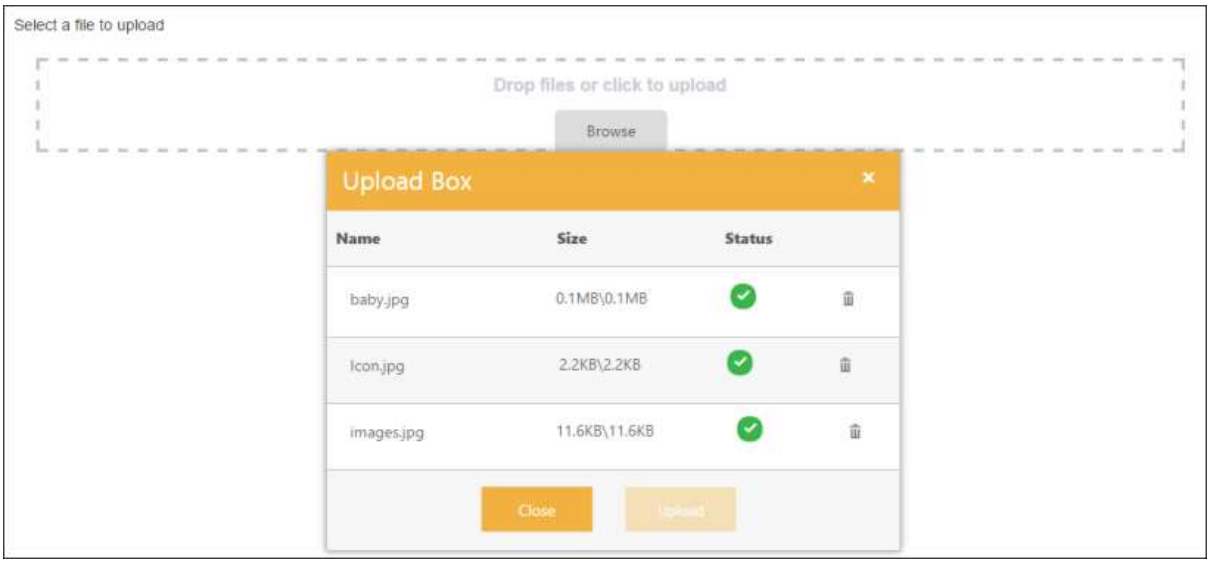
ASPX-CS

```
//Add the following code example to the corresponding CSHTML page to
render UploadBox with drag and drop support
<div class="frame">
<div class="control">
@Html.EJ().Uploadbox("uploadbox")
.SaveUrl("Uploadbox/Save")
.RemoveUrl("UploadBox/Remove")
.AllowDragAndDrop(true).MultipleFilesSelection(true)
</div>
</div>
```

To know about file action, refer to the following link:
http://help.syncfusion.com/aspnetmvc/uploadbox/drag-and-drop-support

2. The following screenshot displays the output for the above code.

Copyright © 2001 - 2020 Syncfusion Inc.



Drag Area text

You can change the drag area text by using the DragAreaText property. By default, the DragAreaText (string) property is Drop files or click to upload in the UploadBox control.

In the VIEW page, add the below script to enable the drag and drop in the UploadBox control.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render UploadBox with drag and drop support
<div class="frame">
<div class="control">
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save")
.RemoveUrl("Uploadbox/Remove")
.AllowDragAndDrop(true)
.MultipleFilesSelection(true)
.DragAreaText("Drop files here")
</div>
</div>
```

3. The following screenshot displays the output for the above code.



Copyright © 2001 - 2020 Syncfusion Inc.

Adjust Drop area size

The UploadBox control provides the ability to change or adjust the drop area size. The DropAreaHeight andDropAreaWidth properties in the UploadBox control allows you to set the maximum height and maximum width for the drop area. The value set to this property is string or number type.

The following steps explain you on how to adjust the Drop Area Size.

In the VIEW page, add the below script to enable the drag and drop in UploadBox control.

ASPX-CS

```
// Add the following code example to the corresponding CSHTML page to
render UploadBox with drag and drop support.
<div class="frame">
<div class="control">
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").AllowDragAndDrop(true).MultipleFilesSelection(true).Drop
AreaHeight("300px").DropAreaWidth("600px")
</div>
</div>
```

4. The following screenshot displays the output for the above code.



Drop area with Browse button behavior

You can click anywhere in the droppable area to browse and upload the files. The droppable area behaves like a browse button.

Enable the AllowDragAndDrop property to achieve this feature. Next, set the ShowBrowseButton as false in UploadBox Control.

The following steps explains the droppable area containing the browse button behavior

In the VIEW page, add the below script to enable drag and drop in the UploadBox control.

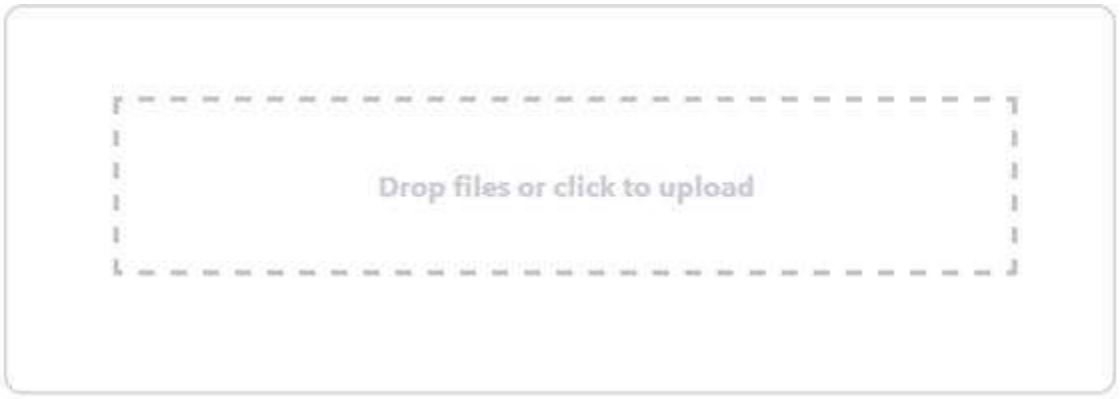
ASPX-CS

```
//Add the following code example to the corresponding CSHTML page to
render UploadBox with drag and drop support.
<div class="control">
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save")
.RemoveUrl("Uploadbox/Remove")
.AllowDragAndDrop(true)
.MultipleFilesSelection(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
.ShowBrowseButton(false)
</div>
```

5. The following screenshot displays the output for the above code.



Appearance and styling

The UploadBox widget provides support to customize the dialog box text and button text.

Customizing Button Text

The following table contains the sub properties available under UploadBox ButtonText property. To customize the text, pass the alternate text with corresponding sub properties.

Sub-properties under buttonText property

Name	Description	Data Type
Browse	Sets the alternative text for browse button.	String
Upload	Sets the alternative text for upload button.	String
Cancel	Sets the alternative text for cancel button.	String

The following steps explain the configuration of UploadBoxButtonText property in UploadBox.

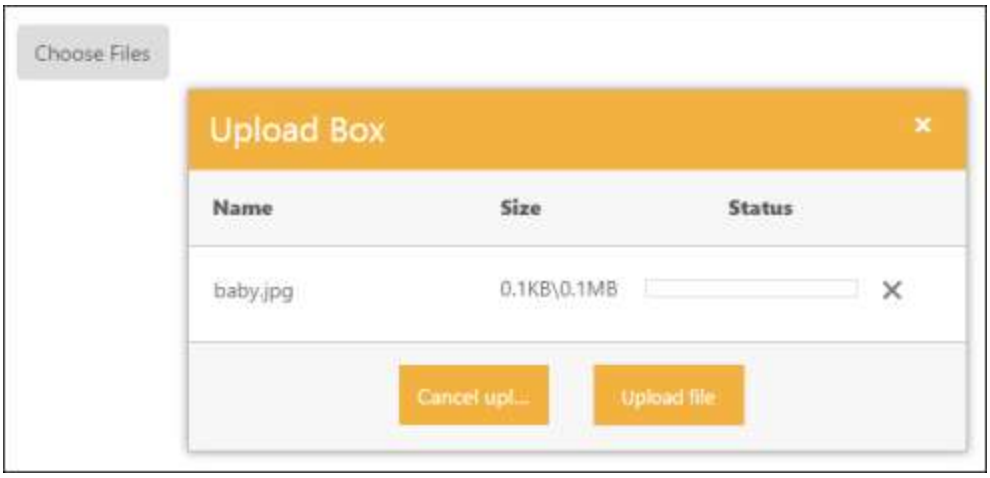
1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").UploadBoxButtonText(text=>text.Browse("Choose
Files").Cancel("Cancel upload").Upload("Upload file"))
```

The following screenshot displays the output.

Copyright © 2001 - 2020 Syncfusion Inc.



Customizing Upload Dialog

The following table contains the sub properties available under UploadBox DialogText property. To customize the text, pass the alternate text with corresponding sub properties.

Sub properties under dialogText property

Name	Description
Title	Sets the alternative text for Title of UploadBox dialog.
Name	Sets the alternative text for Name column.
Size	Sets the alternative text for Size column.
Status	Sets the alternative text for status column.

The following steps explain the configuration of UploadBoxDialogText property in UploadBox.

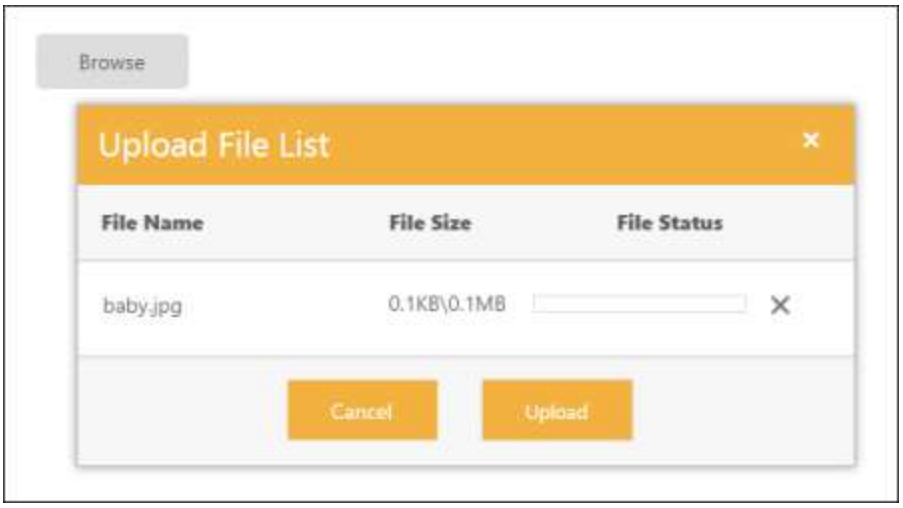
1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").UploadBoxDialogText(text=>text.Title("Upload File
List").Name("File Name").Size("File Size").Status("File Status"))
```

The following screenshot displays the output.

Copyright © 2001 - 2020 Syncfusion Inc.



Show or Hide File details

The ShowFileDetails property is Boolean type, which allow us to show or hide file details in the uploaded file list dialog. To hide the uploaded file details, set ShowFileDetails property is set to ‘false’. By default value of ShowFileDetails property is set to ‘true’.

The following steps explains the configuration of ShowFileDetails property in UploadBox.

1. In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").ShowFileDetails(false)
```

Theme

UploadBox control’s style and appearance are controlled based on CSS classes. In order to apply styles to the UploadBox control, you can refer to two files namely, ej.widgets.core.min.css and ej.theme.min.css. When the file ej.widgets.all.min.css is referred, then it is not necessary to include the files ej.widgets.core.min.css and ej.theme.min.css in your project, as ej.widgets.all.min.css is the combination of these both files.

By default, there are 12-theme support available for UploadBox control namely,

- Default-theme
- Flat-azure-dark
- Fat-lime
- Flat-lime-dark
- Flat-saffron
- Flat-saffron-dark
- Gradient-azure
- Gradient-azure-dark
- Gradient-lime
- Gradient-lime-dark
- Gradient-saffron

Copyright © 2001 - 2020 Syncfusion Inc.

- Gradient-saffron-dark

Custom CSS

CssClass customizes the UploadBox control's appearance. Define a CssClass as per the requirement and assign the class name to CssClass property. The data type is string.

The following steps explain the configuration of CssClass property in UploadBox.

1. In the View page, add the below script to configure the UploadBox element.

ASPX-CS

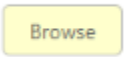
```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").CssClass("custom")
```

2. In CSS, configure Custom Styles for the UploadBox.

CSS

```
<style class="cssStyles">
.custom.e-uploadbox.e-widget .e-selectpart.e-select
{
background-color: #FFFFCC;
font-weight: bold;
font-family: sans-serif;
}
.custom.e-uploadbox.e-widget .e-selectpart.e-select:hover .e-inputbtn
{
background-color: #1e90ff;
}
</style>
```

The following screenshot displays the output.



RTL Support

The EnableRTL property is Boolean type, which allow us to change of left-to-right alignment of the UploadBox widget to right-to-left (RTL). That is, it sets the UploadBox to right-to-left actions. Default value of EnableRTL property is false.

The following steps explain the configuration of EnableRTL property in UploadBox.

In the VIEW page, add the below script to configure the UploadBox element.

ASPX-CS

```
// In the CSHTML page, add the UploadBox element.
@Html.EJ().Uploadbox("uploadbox").SaveUrl("Uploadbox/Save").RemoveUrl("Up
loadbox/Remove").EnableRTL(true)
```

Copyright © 2001 - 2020 Syncfusion Inc.

The following screenshot displays the output.



How To

Render UploadBox from code behind

UploadBox can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of UploadBox using UploadBox properties from code behind.

RAZOR

```
@{
Html.EJ().Uploadbox("UploadDefault",
(Syncfusion.JavaScript.Models.UploadboxProperties)ViewData["upload"]).Ren
der();
}
```

C#

```
public ActionResult Index()
{
UploadboxProperties upload = new UploadboxProperties();
upload.ExtensionsAllow = ".jpg,.png";
upload.SaveUrl = "SaveDefault";
upload.RemoveUrl = "RemoveDefault";
ViewData["upload"] = upload;
return View();
}
public ActionResult SaveDefault(IEnumerable<HttpPostedFileBase>
UploadDefault)
{
foreach (var file in UploadDefault)
{
var fileName = Path.GetFileName(file.FileName);
var destinationPath = Path.Combine(Server.MapPath("~/App_Data"),
fileName);
file.SaveAs(destinationPath);
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
}
return Content("");
}
public ActionResult RemoveDefault(string[] fileNames)
{
foreach (var fullName in fileNames)
{
var fileName = Path.GetFileName(fullName);
var physicalPath = Path.Combine(Server.MapPath("~/App_Data"), fileName);
if (System.IO.File.Exists(physicalPath))
{
System.IO.File.Delete(physicalPath);
}
}
return Content("");
}
```

Upload a file using pushFile property

Raw file details sent from client side will be updated as file details in web method. So, pass the raw file details through push file property and upload the file to server.

The following code illustrates the way to use pushFile property for Upload.

RAZOR

```
<div id = "ControlRegion">
<div class="frame">
<div class="control">
<table>
<tbody>
<tr>
<td>
<h3>Choose a file here to update file details.</h3>
</td>
<td>
<input type="file" id="select" />
</td>
</tr>
<tr>
<td  style="padding-right:30px">
<h3>Click push button to push selected files</h3>
</td>
<td>
<button id="push" onclick="push()">Push selected file to
ejUploadBox</button>
</td>
</tr>
<tr>
<td>
<h3>Syncfusion UploadBox</h3>
</td>
<td>
<div class="posupload">
@Html.EJ().Uploadbox("UploadDefault").SaveUrl("SaveDefault").RemoveUrl("R
emoveDefault"))
</div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
</td>
</tr>
</tbody>
</table>
</div>
</div>
<script>
var details;
var fileName;
$("#select").on("change", function () {
details = this.files[0]; //get file details selected through input
fileName = $('input[type=file]').val().replace(/C:\\fakepath\\/i, '');
//get selected filename
});
function push() {
if (fileName && details != null) {
var Files = { name: fileName, rawFile: details } //Files structure
var push = $("#UploadDefault").data("ejUploadbox");
push.setModel({ pushFile: { Files } });//push files to UploadBox
}
else
alert("Please select a file to push");
}
</script>
```

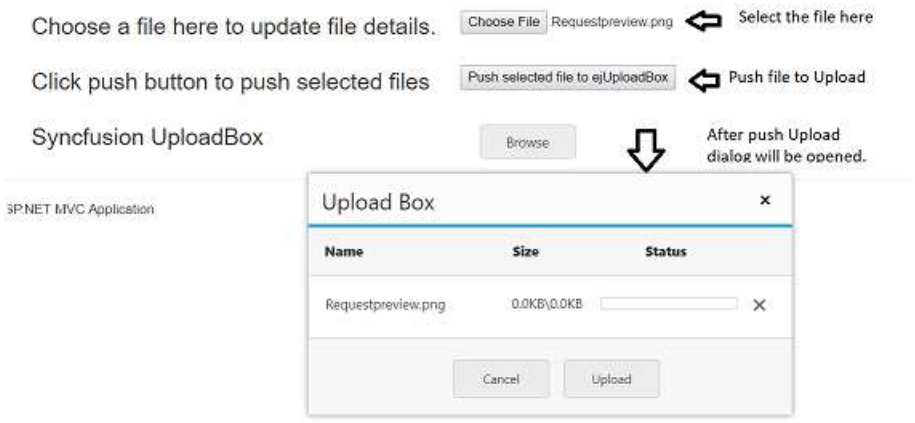
C#

```
public ActionResult Index()
{
return View();
}
public ActionResult SaveDefault(IEnumerable<HttpPostedFileBase>
UploadDefault)
{
var path="";
foreach (var file in UploadDefault)
{
var fileName = Path.GetFileName(file.FileName);
var destinationPath = Path.Combine(Server.MapPath("~/App_Data"),
fileName);
path = destinationPath;
file.SaveAs(destinationPath);
}
return Content(path);
}
public ActionResult RemoveDefault(string[] fileNames)
{
foreach (var fullName in fileNames)
{
var fileName = Path.GetFileName(fullName);
var physicalPath = Path.Combine(Server.MapPath("~/App_Data"), fileName);
if (System.IO.File.Exists(physicalPath))
{
System.IO.File.Delete(physicalPath);
}
}
```

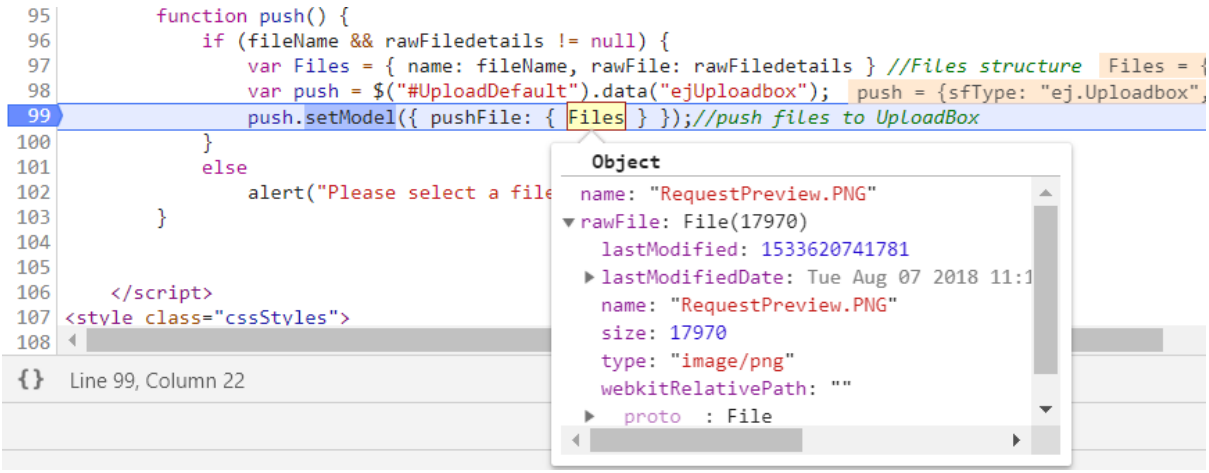
Copyright © 2001 - 2020 Syncfusion Inc.

```
return Content("");
}
```

Choose the file through input and push this file using pushFile property



Pass the file details from client side in the below format



Now the file details can be received in the controller as shown below



Use Synchronous upload with drag and drop

Drag and Drop is supported with Synchronous upload from 16.1.0.32 version. Refer to the below given code for uploading a file through drag and drop with Synchronous upload.

RAZOR

```
@using (Html.BeginForm("SaveSynchronous", "FileUpload", FormMethod.Post,
null))
{
<div class="control">
<div class="element">Select a file to upload</div>
<div class="posupload">
@Html.EJ().Uploadbox("Synchronous").AllowDragAndDrop(true).AsyncUpload(fa
lse)@*Uploading files will be done only on clicking submit button after
selecting the files*@
@Html.EJ().Button("post").Type(ButtonType.Submit).Text("Submit")
</div>
<div id="successmsg">@ViewBag.status</div>
</div>
}
```

C#

```
public ActionResult FileUploadFeatures()
{
if (string.IsNullOrEmpty((string)TempData["status"]))
ViewBag.status = string.Empty;
else
ViewBag.status = (string)TempData["status"];
return View();
}
public ActionResult SaveSynchronous(IEnumerable<HttpPostedFileBase>
Synchronous)
{
if (Synchronous != null)
{
foreach (var file in Synchronous)
{
var fileName = Path.GetFileName(file.FileName);
var destinationPath = Path.Combine(Server.MapPath("~/App_Data"),
fileName);
file.SaveAs(destinationPath);
}
TempData["status"] = "Successfully Uploaded";
return RedirectToAction("FileUploadFeatures");
}
else
{
TempData["status"] = "Select a file to upload";
return RedirectToAction("FileUploadFeatures");
}
}
```

WaitingPopup

WaitingPopup

The WaitingPopup control for MVC is a visual element that provides support for displaying a pop-up indicator over a target area and preventing the end user's interaction with the target area while loading.

Copyright © 2001 - 2020 Syncfusion Inc.

Key Features

- Custom text: Supports custom text inside the pop-up panel.
- Template: Supports including HTML content instead of the default image.
- Transparency: Supports customizing the transparency and opacity level.
- Themes: JavaScript controls include 12 built-in themes (6 flat and 6 gradient effects) and also support the custom skin option to set user-defined themes.

Getting Started

This section explains briefly about how to create a WaitingPopup in your application with ASP.NET MVC.

Create your first Waiting Popup in MVC

ASP.NET MVC Waiting Popup provides support to display Waiting Popup within your web page. From the following guidelines, you can learn how to create a Waiting Popup in a real-time Login page scenario. This helps you in authentication model. The following screenshot illustrates the functionality of a Waiting Popup in a Login page scenario.



In the above screenshot, you can give the Username and Password. When you click the Login button, the Waiting Popup appears. After loading, the alert box appears with a message “Signed in successfully”.

Create Waiting Popup

ASP.NET MVC Waiting Popup widget has a built-in feature to block all other actions until the page is loaded. You can easily create the Waiting Popup control by using simple HTML Helper element as follows

1. You can create an MVC Project and add necessary assemblies, styles, and scripts with the help of MVC-Getting Started Documentation.

Create Login Page

In a real-time Login page scenario, when you click on the Login button, the Waiting Popup is displayed. This is achieved by using the button Click.

1. Add the following code example to the corresponding view page to create Login page with username and password.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div class="content-container-fluid">
<div class="row">
<div class="cols-sample-area">
<table id="target">
<tr>
<td>Username</td>
<td><input type="text"></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password"></td>
</tr>
<tr>
<td></td>
<td>
@Html.EJ().Button("button").Text("Login").Size(ButtonSize.Large).ClientSi
deEvents(e =>e.Create("load").Click("Click"))
</td>
@Html.EJ().WaitingPopup("target").ShowOnInit(false)
</tr>
</table>
</div>
</div>
</div>
```

2. Add the following styles in the view page to show the Waiting Popup.

CSS

```
<style type="text/css" class="cssStyles">
#target
{
margin: 0 auto;
}
#target_WaitingPopup .e-image
{
display: block;
height: 70px;
}
#popup
{
height: auto;
width: auto;
margin-top: 100px;
}
</style>
```

3. Add the following script in the view page.

JS

```
<script>
function Click(e)
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
var popup = $("#target").data("ejWaitingPopup");
popup.show();
setTimeout(success, 5000);
}
function success()
{
alert("Signed in successfully");
var popup = $("#target").ejWaitingPopup("hide");
}
</script>
```

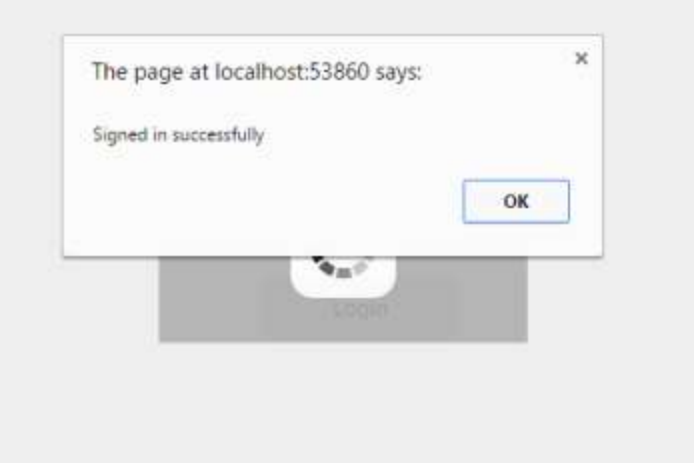
4. The following screenshot displays the User login.



5. The following screenshot shows the Waiting Popup



The following screenshot displays an alert box displayed with the message “Signed in successfully” after loading.



Behavior and Settings

Automatic Initializing WaitingPopup widget

WaitingPopup widget contains ShowOnInit property that allows the popup to display over a target on page load automatically. By default, ShowOnInit property is set as false.

The following steps explains you on how to display the WaitingPopup on page load.

1. In an VIEW page, add a helper element to render WaitingPopup widget.

ASPX-CS

```
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true)
</div>
```

2. Add the following styles to render WaitingPopup widget.

CSS

```
<style type="text/css" class="cssStyles">
#control
{
height: 320px;
width: 600px;
}
</style>
```

The following screenshot illustrates the WaitingPopup when ShowOnInit is set to “true”.

Copyright © 2001 - 2020 Syncfusion Inc.



WaitingPopup with enabled showOnInit property

Enable / Disable Popup Indicator

You can show or hide the popup indicator of WaitingPopup widget using ShowImage property. By default, ShowImage property is set as true.

The following steps explains you to enable / disable popup indicator in WaitingPopup widget.

1. In the VIEW page, add a helper element to render WaitingPopup widget.

ASPX-CS

```
Enable popup indicator:
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true).ShowImage(true).Text("
Loading... Please wait...")
</div>
Disable popup indicator:
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true).ShowImage(false).Text(
"Loading... Please wait...")
</div>
```

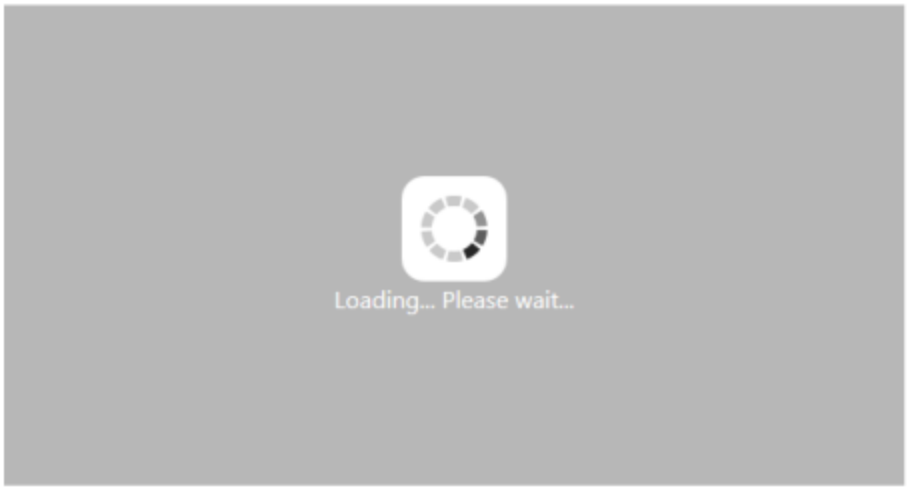
2. Add the following styles to render WaitingPopup widget.

CSS

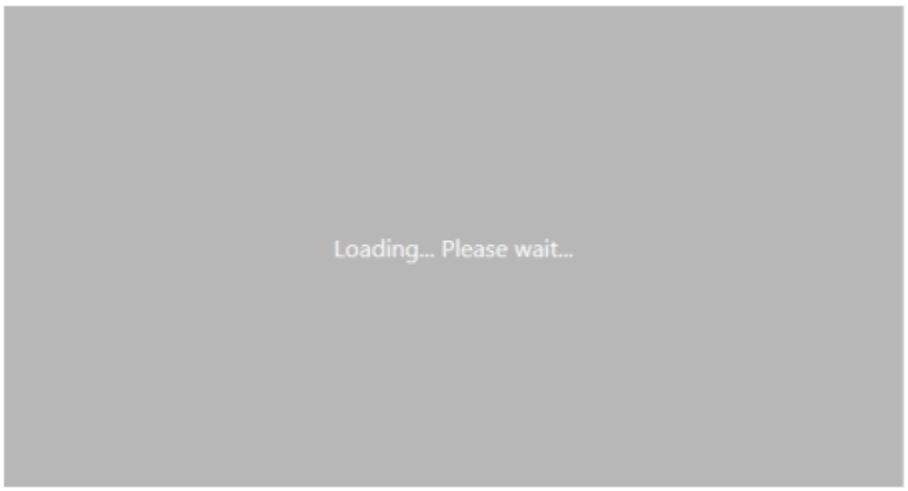
```
<style type="text/css" class="cssStyles">
#control
{
height: 320px;
width: 600px;
}
</style>
```

Execute the above code to render the following output.

Copyright © 2001 - 2020 Syncfusion Inc.



Enabled popup indicator WaitingPopup widget



Disabled popup indicator WaitingPopup widget

Show / Hide WaitingPopup

Using Show() and Hide() methods, you can display or hide the WaitingPopup widget over the target area.

The following steps explains you to show / hide the WaitingPopup widget.

1. In the VIEW page, add a helper element to render WaitingPopup widget.

ASPX-CS

```
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true)
</div>
Show WaitingPopup:
<script type="text/javascript">
var popUpObj;
$(function () {
$("#target").ejWaitingPopup();
popUpObj = $("#target").data("ejWaitingPopup");
```

```
popUpObj.show();
});</script>Hide WaitingPopup:<script type="text/javascript">
var popUpObj    $(function () {
$("#target").ejWaitingPopup();
popUpObj = $("#target").data("ejWaitingPopup");
popUpObj.hide();
});
</script>
```

2. Add the following styles to render WaitingPopup widget.

CSS

```
<style type="text/css" class="cssStyles">
#control
{
height: 320px;
width: 600px;
}
</style>
```

The following screenshot illustrates a WaitingPopup when Show() method is invoked.



WaitingPopup with Show() method

Appearance and Styling

Custom Text

WaitingPopup control provides support for Custom Text to mention any message inside the pop-up panel. You can specify a custom text through the option Text that displays when the Waiting Popup is loading.

The following steps explains you the configuration of the custom text for WaitingPopup control.

1. In the VIEW page, add a helper element to render WaitingPopup widget.

ASPX-CS

Copyright © 2001 - 2020 Syncfusion Inc.

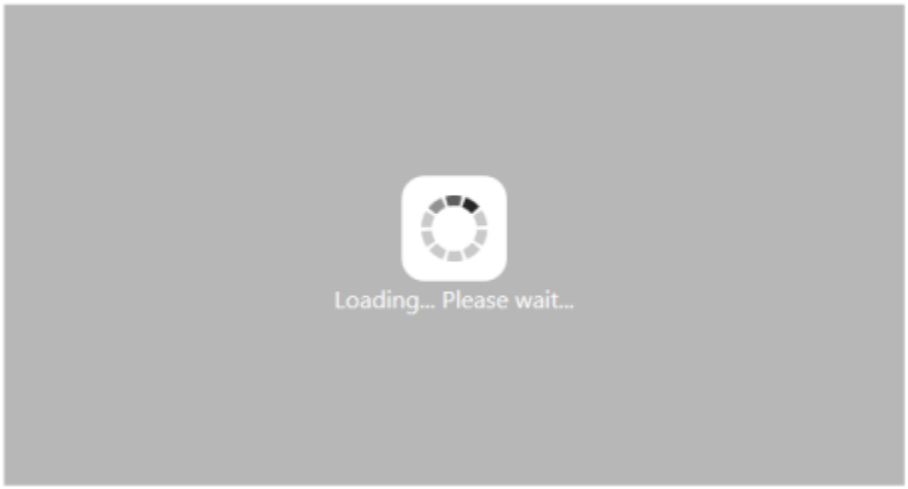
```
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true).Text("Loading...
Please wait...")
</div>
```

2. Add the following styles to render WaitingPopup widget.

CSS

```
<style type="text/css" class="cssStyles">
#control
{
height: 320px;
width: 600px;
}
</style>
```

Execute the above code to render the following output.



Custom Text in WaitingPopup

Template

WaitingPopup widget provides support for Template to customize the appearance of it by including HTML content instead of the default image.

The following steps explains you on how to define template to display a text and image for WaitingPopup control.

1. In the VIEW page, add a helper element to configure WaitingPopup widget.

ASPX-CS

```
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true).Template("#content")
<div id="content">
<div class="block">
<div class="logo"></div>
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
<div class="txt">
<p>Create cutting edge </p>
<p><b>HTML5</b> web application </p>
<p>with ease </p>
</div>
</div>
<div class="loader"></div>
</div>
</div>
```

2. In CSS, you can configure the custom styles for WaitingPopup.

Note: Images for this sample are available ‘installed location /Content/images’ and we need to define images in mentioned CSS. Henceforth the images will display._

CSS

```
<style type="text/css" class="cssStyles">
#waitingPopUp
{
height: 320px;
width: 600px;
margin: 0 auto;
}
.block
{
height: 76px;
}
.logo
{
background-image: url("../Image/js_logo.png");
float: left;
height: 100%;
width: 77px;
margin-right: 15px;
}
.txt
{
float: left;
font-size: 17px;
height: 100%;
text-align: left;
}
.txt p
{
margin: 0;
}
.loader
{
background: url("../Image/load dark.gif") no-repeat scroll -5px 18px
transparent;
height: 40px;
width: 100%;
}
#content
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
{
cursor: default;
height: 112px;
width: 275px;
}
</style>
```

Execute the above code to render the following output.



WaitingPopup with Custom Template

CSS Class

You can use the CSS class to customize the WaitingPopup control appearance. Define a CSS class as per requirement and assign the class name to CssClass property.

The following steps allows you to configure CSS class for an auto-complete textbox.

1. In the VIEW page, add a helper element to configure WaitingPopup widget.

ASPX-CS

```
<div id="target">
@Html.EJ().WaitingPopup("target").ShowOnInit(true).CssClass("custom").Tex
t("Loading... Please wait...")</div>
```

2. Define CSS class for customizing the WaitingPopup widget.

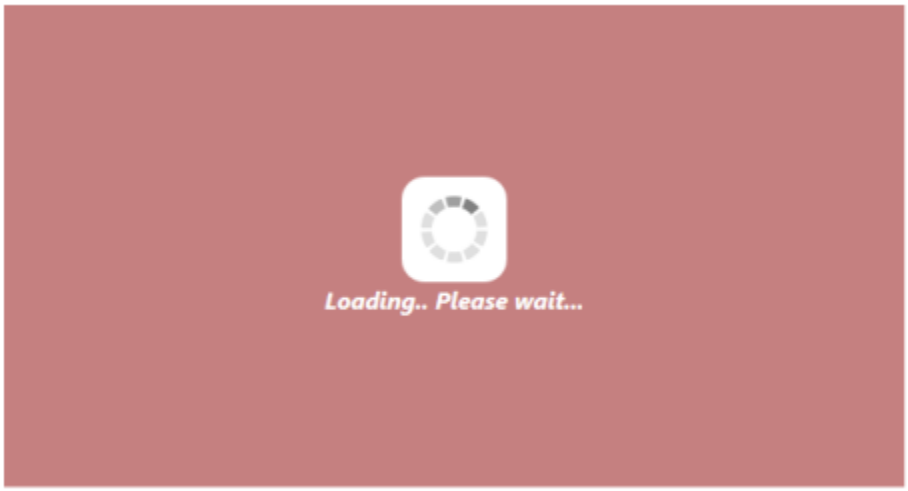
CSS

```
<style type="text/css" class="cssStyles">
/*Customize the panel property*/
#waitingPopUp
{
height: 320px;
width: 600px;
```

Copyright © 2001 - 2020 Syncfusion Inc.

```
margin: 0 auto;
}
/* Customize the WaitingPopup */
.customStyle
{
background-color:darkred;
font-style:italic;
font-weight:bolder;
opacity:0.5;
}
</style>
```

The following screenshot displays the output for the above code.



WaitingPopup with customized CSS

How To

Render WaitingPopup from code behind

WaitingPopup can be rendered from the code behind by initializing the required properties in controller and passing those properties via ViewData or Model to the client side

The following code illustrates the rendering of WaitingPopup using WaitingPopup properties from code behind.

RAZOR

```
<div id="target">
@{Html.EJ().WaitingPopup("target", Model).Render();}
</div>
```

C#

```
public ActionResult Index()
{
WaitingPopupProperties Obj = new WaitingPopupProperties();
Obj.ShowOnInit = true;
return View(Obj);
}
```

Copyright © 2001 - 2020 Syncfusion Inc.

Copyright © 2001 - 2020 Syncfusion Inc.