ASP.NET MVC コントロール

2022.04.19 更新

グレープシティ株式会社

ASP.NET MVC コントロール

目次

ComponentOne for ASP.NET MVC 16
はじめに 17
システム要件 17
インストール 17-19
MVC のアンインストール 19-20
ライセンス 20-21
MVC アプリケーションの設定 22
ComponentOne テンプレートの使用 22-23
Visual Studio テンプレートの使用 23-27
リソースの登録 27
遅延スクリプト 27-28
コントロールの追加 29
Razor 構文の使用 29-32
スキャフォールディングの使用 32-42
クライアント側のサポート 43
クライアント側のIntellisenseサポート 43-47
クライアント側のメソッドおよびイベントの処理 47-50
コントロールの使用 51
Accordion 51
主な特長 52
クイックスタート 52-55
Accordion の使用 55
ペインの展開/折り畳み 55-60
BarCode 60
主な特長 60-61
バーコードエンコーディング 61-63
クイックスタート 63-65
CollectionView 65-66
主な特長 66
クイックスタート 66-69
CollectionViewの使用 69

Copyright © GrapeCity, Inc. All rights reserved.

サーバー側処理 69
一括更新 69-72
作成 72-74
削除 74-76
サーバー読み取りの無効化 76-80
編集 80-82
複数コントロールのデータ連結 82-86
読み取り 86-88
クライアント側処理 88
現在のレコードの管理 88-90
フィルタ処理 90-93
グループ化 93-96
ソート 96-102
変更の追跡 102-105
計算フィールド 105-107
DateTime処理 107-117
ビューの移動 117-119
検証 119-123
DashboardLayout 123-124
主な特長 124-125
クイックスタート 125-129
DashboardLayout の使用 129
ドラッグアンドドロップ 129-131
ヘッダー 131-132
レイアウト 132
フロー レイアウト 132-134
自動グリッドレイアウト 134-137
手動グリッドレイアウト 137-139
スプリットレイアウト 139-141
サイズ変更 141-143
ツールアイコン 143-144
XMLのシリアル化 144-145
チュートリアル 145-146

Copyright © GrapeCity, Inc. All rights reserved.

カスタムタイル 146-150
カスタムツールバー 150-155
カスタムツールバーのオプション 155-159
Excel 159
主な特長 159-160
クイックスタート 160-163
File Manager 163-164
主な特長 164
UI要素 164-165
表示 165-166
コンテキストメニュー 166
クラウドストレージ内のファイルの管理 166-167
ローカルストレージ内のファイルの管理 167-168
FinancialChart 168-169
主な特長 169-170
財務チャートタイプ 170-180
クイックスタート：FinancialChartへのデータの追加 180-185
財務チャートの使用 185
フィボナッチツール 185-189
Financial Chart インジケータ 189
移動平均収束発散法 189-197
ストキャスティクス 197-205
Average True Range 205-213
商品チャンネル指数 213-219
相対力指数 219-226
Williams %R 226-233
ラインマーカー 233-235
移動平均 235-236
Financial Chart オーバーレイ 236
ボリンジャーバンドオーバーレイ 236-243
エンベロープオーバーレイ 243-250
範囲セレクタ 250-253
近似曲線 253-255

Copyright © GrapeCity, Inc. All rights reserved.

FlexChart 255
- 主な特長 255-256
- チャートの要素 256
- チャートタイプ 256-264
- クイックスタート：FlexChart へのデータの追加 264-271
- MVCチャートの比較 271-278
- FlexChartの使用 278-279
  - 注釈 279-284
  - チャートのアニメーション 284-285
  - チャートのジェスチャ 285-286
  - グラデーションチャート 286-287
  - 外観のカスタマイズ化 287-290
  - 軸のカスタマイズ 290-291
  - 関数系列 291-293
  - ヘッダーとフッター 293
  - ヒットテスト 293-295
  - ラベル 295-296
  - 凡例 296-297
  - マーカー 297-299
  - 複合チャート 299-300
  - 複数の軸 300-302
  - プロット領域 302-303
  - 範囲セレクタ 303-305
  - 選択 305-306
  - 系列 306
    - 箱ひげ図チャート 306-309
    - 損益分岐点チャート 309-310
    - 誤差範囲チャート 310-313
    - ウォータフォール 313-315
  - スタイル設定とCSS 315-317
    - スタイルの適用 317-318
  - ツールチップ 318-319

Copyright © GrapeCity, Inc. All rights reserved.

近似曲線 319-321

FlexGrid 321

主な特長 321-322

クイックスタート：FlexGrid へのデータの追加 322-327

FlexGridの使用 327-328

列 328-329

列フッターパネル 329-330

列グループ 331-334

列ヘッダーのグループ化 334-336

列ピッカー 336-339

列の固定 339

ヘッダにツールチップの表示 339-341

列レイアウトの保存とロード 342-343

クライアント側でグリッド列を表示/非表示 343-346

非連結列 346-348

カスタムセルテンプレート 348-350

テンプレート関数 350-354

カスタムエディタ 354-360

データ連結 360

AJAX連結 360-365

FlexGridのマスター詳細 365-368

モデル 連結 368-372

OData連結 372-374

オンデマンド読み込み 374-375

リモートデータ連結 375-376

DataMap 376-379

DataMapエディター 379-384

詳細行 384-386

サーバー読み取りの無効化 386-388

編集 388

一括編集 388-391

遅延一括更新 392-393

Excel 形式の編集 393-397

Copyright © GrapeCity, Inc. All rights reserved.

Excelのインポートまたはエクスポート 397-400
PDF へのエクスポート 400-402
排他値の検索 402
フィルタ処理 402-404
FlexGridセレクター 404-405
フルテキスト検索 405-407
大文字と小文字を区別する検索 407-409
グループ化 409-411
実行時にグループ化 411-413
キーボードハンドリング 413-417
マージ 417-418
ページング 418-419
右から左へのレンダリング 419-421
選択モード 421-422
スターサイズ 422-423
スタイル設定とCSS 423-426
スタイルの適用 426-428
TreeGrid 428-432
編集可能なTreeGrid 432-433
非連結のTreeGrid 433-435
XMLへの連結 435-437
非連結のFlexGrid 437
AJAX呼び出しを使用した非連結のFlexGrid 437-442
CollectionViewServiceを使用した非連結のFlexGrid 442-443
控えめな検証 443-446
仮想スクロール 446-447
TransposedGrid 447-453
FlexMap 453
主な特長 453-454
クイックスタート 454-455
FlexMapの使用 455-456
カラースケール 456-457
レイヤー 457-462

Copyright © GrapeCity, Inc. All rights reserved.

凡例 462-463
スタイル設定 463-464
サポートされているマップの種類 464-466
ツールチップ 466-467
FlexPie 467-468
主な特長 468
クイックスタート：FlexPie へのデータの追加 468-472
FlexPieの使用 472
FlexPie のカスタマイズ 472-473
ドーナツ円グラフ 474
分割円グラフ 474-475
柔軟なレイアウト 475-477
ヘッダーとフッター 477-478
内部テキスト 478
凡例 478-479
複数の円グラフ 479-480
選択 480-481
FlexRadar 481-482
主な特長 482
クイックスタート：FlexRadar へのデータの追加 482-485
FlexRadarの使用 485
ヘッダーとフッター 485-486
凡例 486-487
FlexReport 487
主な特長 487-488
FlexReport Web API の設定 488
ComponentOne WebAPI テンプレートの使用 488-489
標準のVisual for Web APIテンプレートの使用 490-493
FlexSheet 493
主な特長 493-494
クイックスタート：FlexSheet 494-498
FlexSheetの使用 498-499

Copyright © GrapeCity, Inc. All rights reserved.

セルの結合 499
実行時にセルの結合 499-501
コードでセルの結合 501-502
クライアント側のExcelファイルのロードおよび保存 502-505
コンテキストメニューの使用 505
データの連結 505-509
ドラッグアンドドロップ 509-513
フィルタリング 513-517
セルの書式 517-519
FlexSheetの数式 519
FlexSheetでサポートされた関数 519-522
実行時に数式の使用 522-523
コードで数式の使用 523-525
固定のセル 525-528
クライアント側でのJSONのロードと保存 528-529
複数のヘッダー 529-531
リモートでロードおよび保存 531-534
ソート 534-537
スタイル設定とCSS 537-538
表 538-540
非連結 シート 540-541
FlexViewer 541-542
ReportViewer 542
ReportViewerの要素 542-545
C1MVC ReportViewerテンプレートでレポートの表示 545-546
現在のプロジェクトのレポート 546-548
他のレポートサービスのレポート 548-550
SSRSサーバーのレポート 550-552
ActiveReportsの表示 552-553
ActiveReportsサービスの構成 553-555
FlexViewerを手動で設定 555-558
MobileView 558-559
MobileViewerの要素 559-562

Copyright © GrapeCity, Inc. All rights reserved.

PDFViewer 562-563
PdfViewer の使用 563-564
FlexViewerのスタイル設定 564-568
Gauge 568
主な特長 568-569
ゲージのタイプ 569-570
クイックスタート 570
BulletGraph クイックスタート 570-571
LinearGauge クイックスタート 571-573
RadialGauge クイックスタート 573-575
Gaugeの使用 575
外観のカスタマイズ 575-576
方向 576-577
値の表示 577-578
範囲 578-579
スタイル設定とCSS 579
Gaugeのスタイル設定 579-581
Input 581-583
AutoComplete 583
主な特長 583
クイックスタート 583-587
AutoCompleteの使用 587
カスタムデータソースを使用する AutoComplete 587-588
カスタムアクション 588-589
Calendar 589
主な特長 589-590
クイックスタート 590-591
Calendarの使用 591
日付検証 591-592
ColorPicker 592-593
主な特長 593
クイックスタート 593-595
ComboBox 595

Copyright © GrapeCity, Inc. All rights reserved.

主な特長 595-596
クイックスタート 596-598
ComboBox の使用 598
大文字と小文字を区別する検索 598-599
InputColor 599
主な特長 599
クイックスタート 599-601
InputColor の使用 601-602
色の文字列表現の表示 602-603
InputDate 603
主な特長 603-604
クイックスタート 604-605
InputDateの使用 605
日付検証 605-607
InputDateRange 607
主な特長 607
要素 607-608
クイックスタート 608-609
日時書式 609-610
日範囲 610-611
カレンダーの最小日と最大日の設定 611-612
事前定義された範囲 612-614
表示モード 614-615
日付の検証 615
レイアウトおよび外観 615-616
InputMask 617-618
主な特長 618
クイックスタート 618-619
InputMaskの使用 619
入力マスクのタイプ 619-620
InputNumber 620-621
主な特長 621

Copyright © GrapeCity, Inc. All rights reserved.

クイックスタート 621-624
InputTime 624
主な特長 624
クイックスタート 624-626
InputTimeの使用 626
データ連結 626-627
InputDateTime 627-628
主な特長 628
クイックスタート 628-630
ListBox 630
主な特長 630
クイックスタート 630-632
ListBoxの使用 632
カスタムの ListBox 632-634
大文字と小文字を区別する検索 634-635
Menu 635
主な特長 635-636
クイックスタート 636-637
Menuの使用 637
コンテキストメニュー 637-640
InputNumber を使用した Menu 640-641
MultiAutoComplete 641-642
主な特長 642
クイックスタート：MultiAutoComplete へのデータの追加 642-645
MultiAutoCompleteの使用 645
複合型 645-647
インクリメンタル検索 647-649
Multi-select 649-650
主な特長 650
クイックスタート 650-653
MultiSelectの使用 653
オブジェクトのコレクションに連結するMultiSelect 653-654
MultiSelectフィルター 654-656

Copyright © GrapeCity, Inc. All rights reserved.

MultiSelectListBox 656-658
Popup 658-659
主な特長 659
クイックスタート 659-662
Popupの使用 662
ダイアログ：Owner要素のないポップアップ 662-664
共通の機能 664
AutoExpandSelection 664-665
DropDownCssClass 665-666
スタイル設定とCSS 666-669
スタイルの適用 669-673
控えめな検証 673-676
仮想化 676-677
タブオーダー 677-679
MultiRow 679-680
主な特長 680
レイアウト定義 680-681
クイックスタート：MultiRow へのデータの追加 681-686
MultiRowの使用 686
一括編集 686-690
折りたたみ列ヘッダー 690-691
データ連結 691
モデル連結 691-696
リモートデータ連結 696-698
エクスポート 698
Excel エクスポート 698-702
PDF エクスポート 702-706
フィルタ 706-708
グループ化 708-710
グループパネル 710-713
ページング 713-715
固定行や列 715-716
選択モード 716-717

Copyright © GrapeCity, Inc. All rights reserved.

スタイル設定とCSS 717-719
レコード、グループ、セルのスタイル設定 719-722
仮想スクロール 722-723
TransposedMultiRow 723-728
OLAP 728-729
主な特長 729-730
OLAP の概要 730-731
OLAP のアーキテクチャ 731
Pivot Chart 731-732
Pivot Grid 732
Pivot Panel 732-733
Slicer 733-734
クイックスタート: OLAP へのデータの追加 734-738
OLAPの使用 738
データ連結 738-739
キューブデータ連結 739-740
DataEngineを使用したキューブデータ連結 740-743
データエンジンサービス 743-748
モデル連結 748-752
リモートデータ連結 752-755
Pivotフィールドの定義 755-756
エクスポート 756-759
フィルタリング 759
フィルタ処理 759-761
Slicerを使用してフィルタ処理 761-763
グループ化 763-764
ビューの保存とロード 764-765
ソート 765
スタイル設定とCSS 765-767
スタイルの適用 767-769
実行時にフィールドのプロパティを更新する 769-770
PDF 770

Copyright © GrapeCity, Inc. All rights reserved.

主な特長 770-771
クイックスタート 771-776
SunburstChart 776
主な特長 776-777
クイックスタート：Sunburst へのデータの追加 777-780
Sunburstの使用 780
ヘッダーとフッター 780-781
凡例 781-782
選択 782-783
テーマ 783-785
TabPanel 785
主な特長 785-786
基本クラス 786
クイックスタート：TabPanelへのデータの追加 786-788
TabPanel の使用 788-789
ユーザー補助 789-790
分離されたパネル 790-791
無効と非表示タブ 791-793
MVC コントロールのホスティング 793-797
右から左 797-798
スタイル設定とCSS 798
スタイル設定 798-801
TreeMap 801
主な特長 801
要素 801-802
レイアウト 802-804
クイックスタート：TreeMapへのデータの追加 804-806
TreeMapの使用 806
グループコレクション 806-807
MaxDepth 807-810
テーマ 810-811
TreeView 811-812
主な特長 812

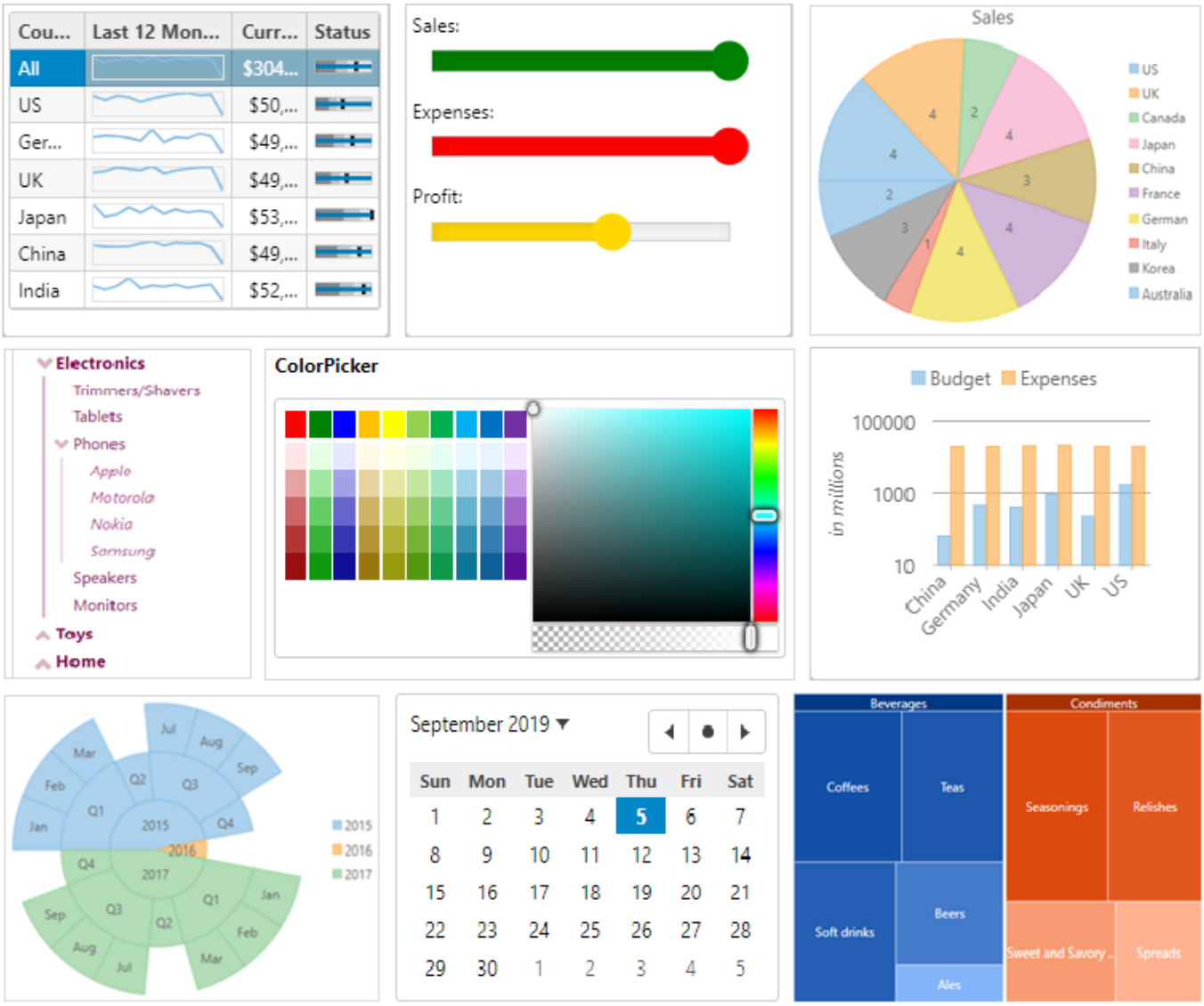
Copyright © GrapeCity, Inc. All rights reserved.

クイックスタート：TreeView へのデータの追加 812-815
TreeViewの使用 815-816
カスタムノード 816-817
データ連結 817
モデル連結 817-820
リモート連結 820-821
ドラッグアンドドロップ 821-822
ノードの編集 822-823
ノードの展開と折りたたみ 823-824
画像 824-826
遅延ロード 826-828
ノード間の移動 828-830
スタイル設定とCSS 830-831
スタイルの適用 831-833
チェックボックスの使用 833-835
Globalization 836
スタイル設定 837
CSS スタイル 837
テーマ 837-840

Copyright © GrapeCity, Inc. All rights reserved.

ComponentOne for ASP.NET MVC

ComponentOne for ASP.NET MVC は、従来のブラウザのサポートについて妥協することなく、HTML5、CSS、ECMA5 などの最先端技術の上に構築された最新の UI コントロールのコレクションです。ComponentOne ASP.NET MVC には、データ管理を初め、データの視覚化、プロジェクトテンプレート、洗練されたデザインのテーマなどの機能を備えた、高速かつ軽量なコントロールが含まれます。



Copyright © GrapeCity, Inc. All rights reserved.

はじめに

このセクションは、ASP.NET MVC コントロールの使用を開始するにあたって役立つ情報の提供を目的としています。システム要件、ASP.NET MVC Editionのインストール、ライセンス、テクニカルサポートなどに関する情報が含まれています。コントロールの使用方法の詳細については、「コントロールの追加」および「コントロールの使用」セクションを参照してください。

システム要件

ComponentOne for ASP.NET MVC をインストールするには、以下がシステムにインストールされている必要があります。

環境	ブラウザのサポート
• .NET 4 以上。 • ASP.NET MVC 3 以上。 • Visual Studio 2012 以上。	• Internet Explorer 9 以上 • Microsoft Edge • Mozilla Firefox • Safari • Chrome

インストール

You can install ASP.NET MVC Edition using ComponentOne ControlPanel. To do so, download the **ComponentOneControlPanel** tool from GrapeCity website and install it. After installing ComponentOneControlPanel, it's shortcut will be available at the following location on your system: **C:\ProgramData\Microsoft\Windows\Start Menu\Programs** or else you can find it using the **Search** box from the taskbar.

For detailed information on ComponentOne ControlPanel and its functionalities, see ComponentOne ControlPanel blog.

> **Note**: Your machine setup may require you to be logged in as an Administrator to install new software. If this is the case and you do not have Administrator privileges, consult your system administrator.

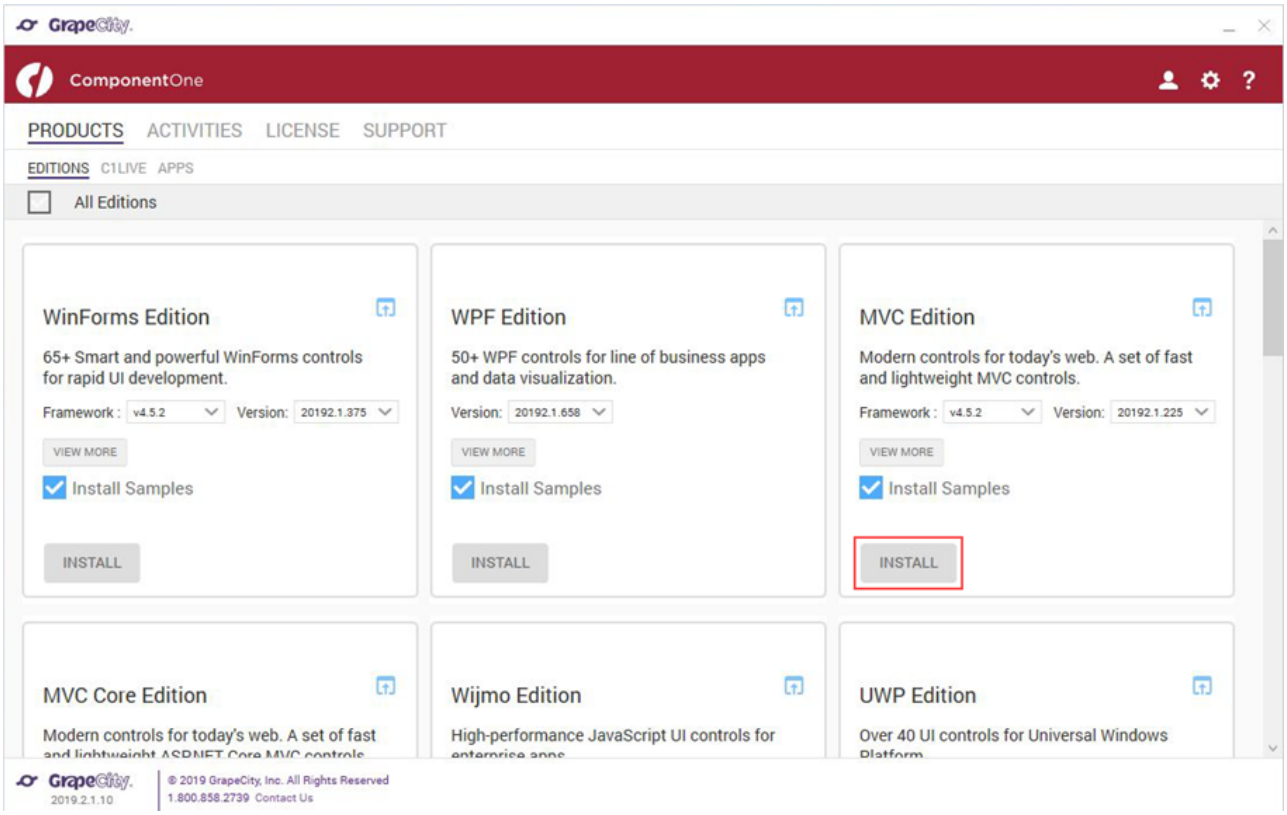
Installing MVC Edition

To install ComponentOne ASP.NET MVC Edition, follow these steps.

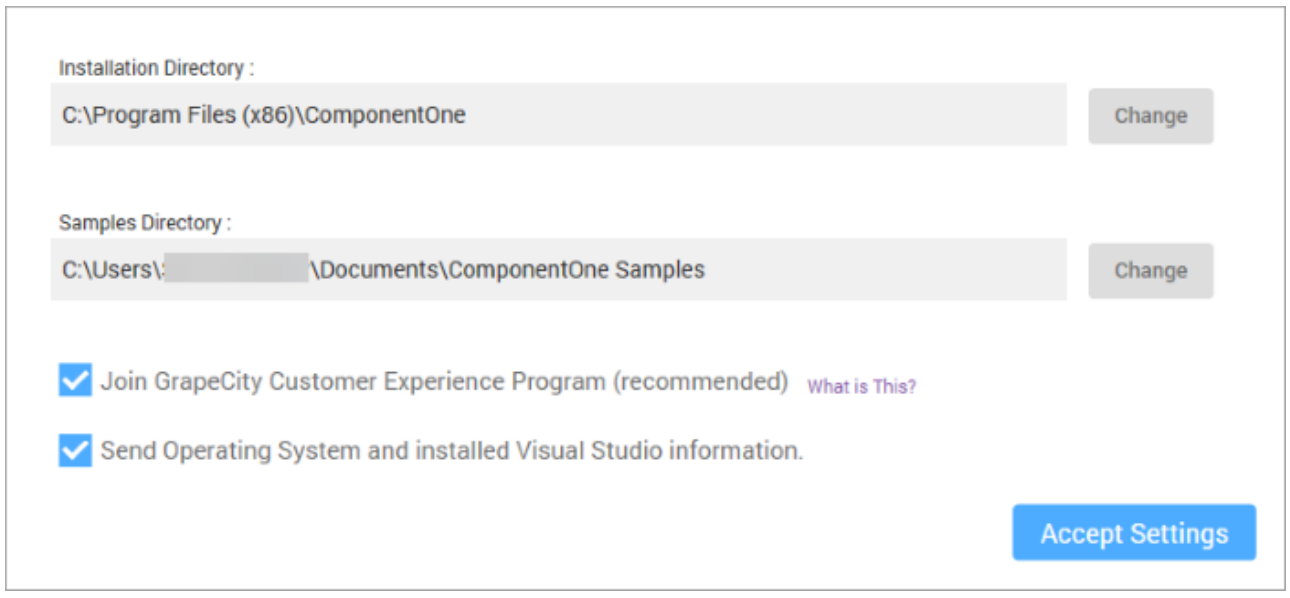
1. Open **ComponentOne ControlPanel** using the **ComponentOneControlPanel.exe** file**.** Note that any running instances of **Visual Studio** must be closed.
   In case you are an existing user, you can login using the registered email address and password. Alternatively, if you are a new user:
   - Register with ComponentOne and create an account by filling the required fields.
   - A verification mail will be sent to your email address.
   - Visit the activation link to verify your email address.

   In case you do not wish to login or register, you can proceed as an anonymous user.
2. In **ComponentOne ControlPanel**, click **Install** button from the **MVC Edition** tile.

Copyright © GrapeCity, Inc. All rights reserved.

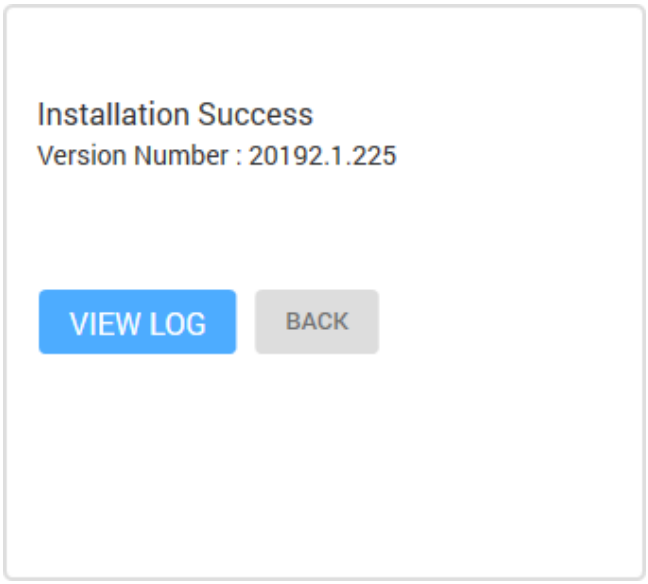


3. Review the license agreement, and then click the **Accept License Agreement** button.
4. Click **Accept Settings** button to confirm the directory path and start the installation process. In case you want to change the installation and samples directories location, click the **Change** button and select the location of your choice.



Once the installation is completed, the installer displays the **Installation Success** screen.

Copyright © GrapeCity, Inc. All rights reserved.



- All dependency packages, style sheets and scripts are already added to the project created using C1 ASP.NET MVC Web Application template. There is no need to explicitly add the packages to the project. But, you need to manually add the packages into your MVC project which is created using Visual Studio templates, in order to use these controls.
- By default, ControlPanel downloads the latest version of an edition. Note that you can select older versions by clicking on the Change/Repair Version button.

Back to Top

サンプル

この製品のサンプルは、デフォルトで ComponentOne Samples フォルダにインストールされます。

C:¥Users¥<username>¥Documents¥ComponentOne Samples¥ASP.NET MVC

Back to Top

ComponentOne Scaffolder Visual Studio拡張機能のインストール

- ComponentOne Scaffolder拡張機能は、ComponentOne と共に自動的にインストールされます。このVisual Studio拡張機能がインストールされているかどうかは、[**ツール**]→[**拡張機能と更新プログラム**]に移動して確認できます。

Back to Top

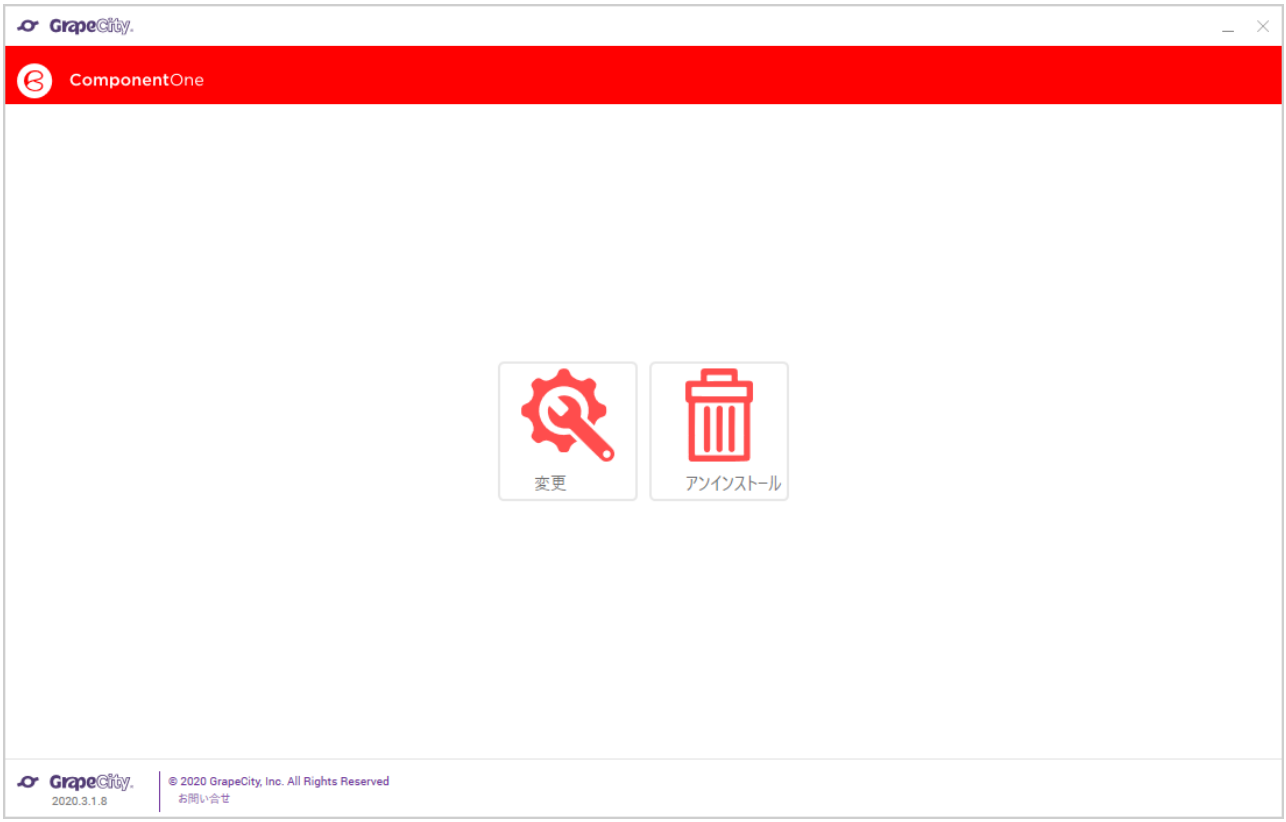
ASP.NET MVC をアンインストールする場合は、「MVC のアンインストール」を参考してください。

MVC のアンインストール

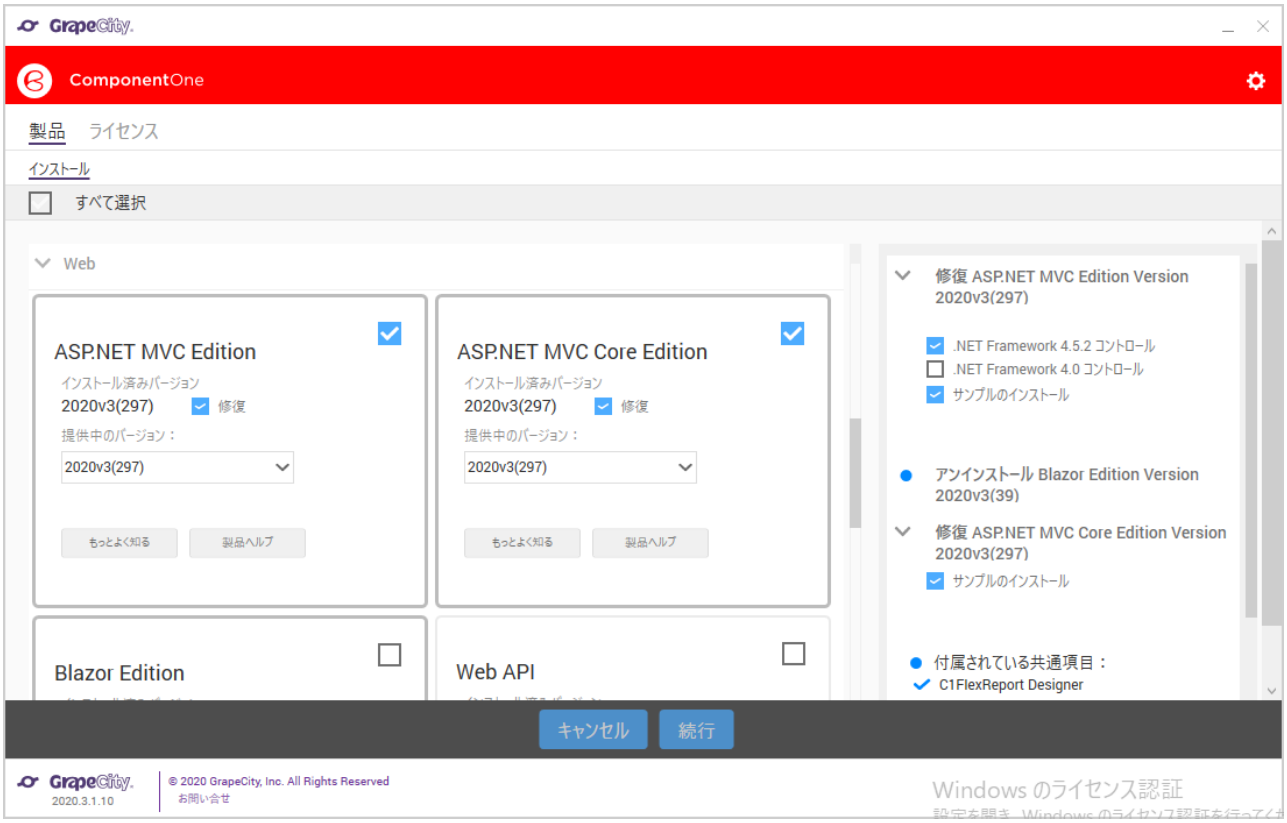
ASP.NET MVCをアンインストールするには

1. スタートメニューから**[コントロールパネル]**を開き、**[プログラム]**の**[プログラムのアンインストール]**を選択します。
2. リストから**[ComponentOne]**を選択し、**[アンインストール]**ボタンをクリックします。
3. インストーラウィンドウが表示されたら、**[変更]**をクリックします。

Copyright © GrapeCity, Inc. All rights reserved.



4. インストーラウィンドウで**[ASP.NET MVC]**のチェックマークをオフにし、**[次へ]**をクリックしてプログラムをアンインストールします。



ライセンス

Copyright © GrapeCity, Inc. All rights reserved.

ASP.NET MVC コントロール

ASP.NET MVC は、ComponentOne Enterpriseの一製品であり、他のすべてのComponentOne製品と同様に標準のサブスクリプションモデルに準拠しています。MVC 3、4、および5アプリケーションのライセンス情報は**Licenses.licx**ファイルに含まれていますが、ASP.NET Core MVCアプリケーションはアプリケーションベースのライセンスを使用します。

ライセンス認証

本製品をインストールしてご使用いただくためには、ライセンス認証を行う必要があります。詳細については、リリースノートを参照してください。

ライセンスの仕組み

ライセンスファイルをアプリケーションに手作業で追加するには

1. **ソリューションエクスプローラー**で、プロジェクト名を右クリックし、**[追加]**→**[新しい項目]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、**[C#]**→**[一般]**を選択し、右ペインで**[テキストファイル]**を選択します。
3. テキストファイルに「**licenses.licx**」という名前を付けます。
4. 次のコードをテキストファイルに貼り付けます。

licenses.licx

```
C1.Web.Mvc.LicenseDetector, C1.Web.Mvc
```

5. アプリケーションで**FinancialChart**、**FlexSheet**、**MultiRow**、**FlexViewer**、**OLAP**の各コントロールを使用するには、さらにいくつかの手順が必要な場合があります。**licenses.licx**ファイルで、**C1.Web.Mvc.LicenseDetector, C1.Web.Mvc**の後で[Enter]を押し、次のコードを貼り付けます。

licenses.licx

```
C1.Web.Mvc.Finance.LicenseDetector, C1.Web.Mvc.Finance
C1.Web.Mvc.Sheet.LicenseDetector, C1.Web.Mvc.FlexSheet
C1.Web.Mvc.Viewer.LicenseDetector, C1.Web.Mvc.FlexViewer
C1.Web.Mvc.OLAP.LicenseDetector, C1.Web.Mvc.OLAP
C1.Web.Mvc.MultiRow.LicenseDetector, C1.Web.Mvc.MultiRow
```


Copyright © GrapeCity, Inc. All rights reserved.

MVC アプリケーションの設定

このトピックでは、Visual Studio で、Visual Studio テンプレートおよび ComponentOne テンプレートを使用して、新しい MVC アプリケーションを作成する方法を示します。次の実装を行う前に、「システム要件」を参照してください。

- ComponentOne テンプレートの使用
- Visual Studio テンプレートの使用

標準の Visual Studio テンプレートに優る ComponentOne テンプレートのメリット

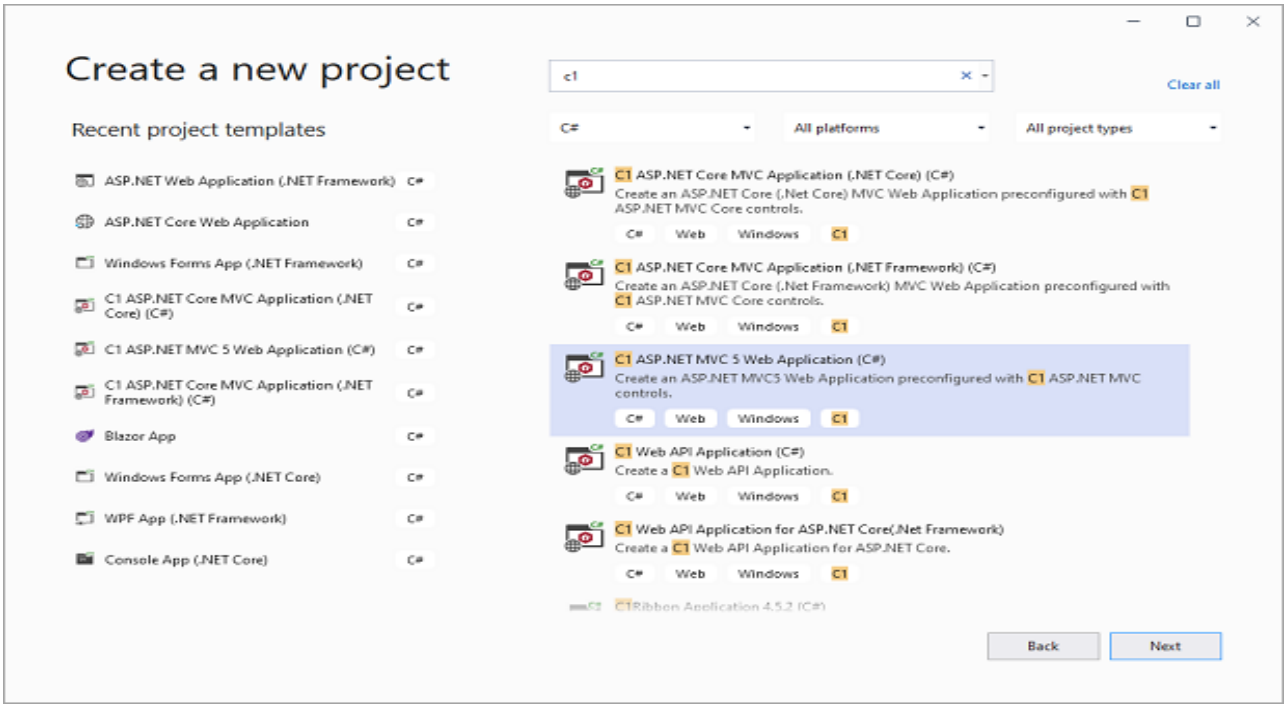
ComponentOne テンプレートは、以下のようなメリットがあります。

- MVC コントロールを実装するために必要な参照を登録します。
- web.config ファイルに自動的にエントリを追加します。
- プロジェクトに ComponentOne for ASP.NET MVC の参照を自動的に追加します。
- プロジェクトにlicense.licxファイルを自動的に追加します。

ComponentOne テンプレートの使用

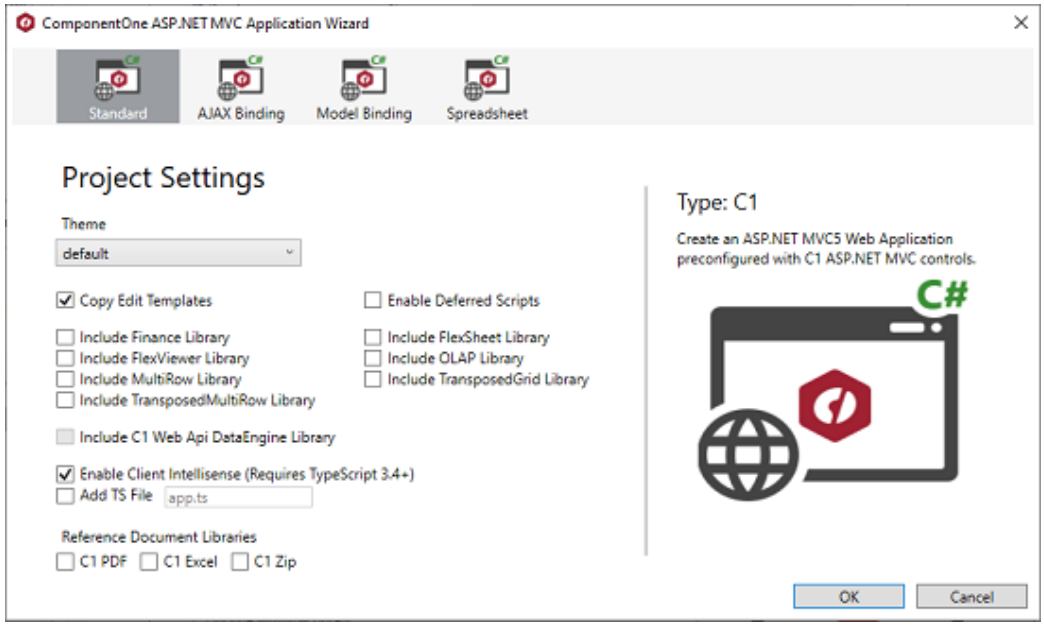
ComponentOne テンプレートを使用して、新規 MVC アプリケーションを作成するには、以下の手順を実行します。

1. **[ファイル]**→**[新規作成]**→**[新しいプロジェクト]**を選択します。
2. インストール済みテンプレートから、**[C1]**→**[Web]**→**[C1 ASP.NET MVC 5 アプリケーション]**を選択します。



3. アプリケーションの**名前**および**場所**を設定します。
4. **[OK]**をクリックします。「**ComponentOne ASP.NET MVC アプリケーションウィザード**」が表示されます。

Copyright © GrapeCity, Inc. All rights reserved.



5. **[テーマ]**を選択し、参照するドキュメントライブラリを選択します。
6. **[OK]**をクリックします。
7. ComponentOne for ASP.NET MVC を使用するように設定された新しいプロジェクトが作成されます。

アプリケーションに、**licenses.licx** ファイルが自動的に追加されます。アプリケーションにライセンスを追加する方法については、「ライセンス」を参照してください。

Visual Studio テンプレートの使用

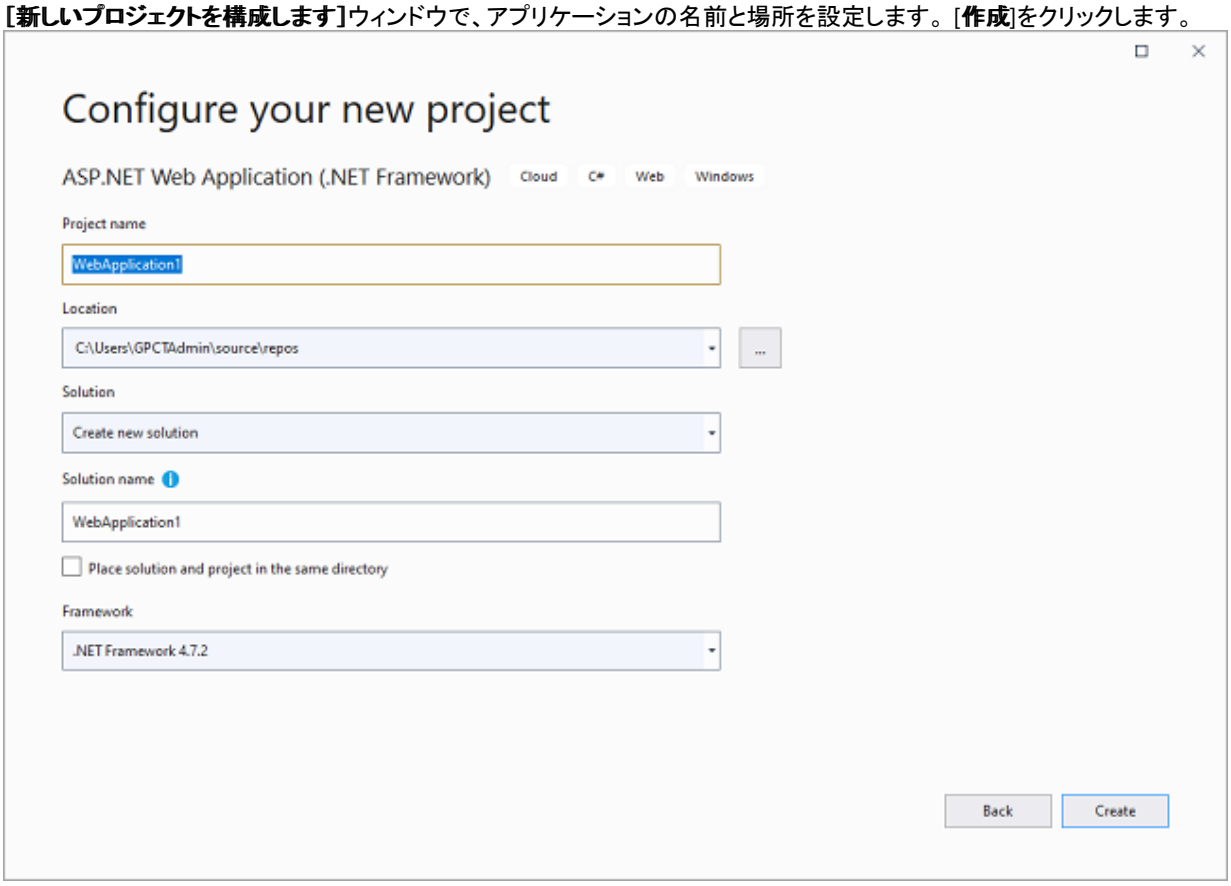
ComponentOne テンプレートを使用して、新規 MVC アプリケーションを作成するには、以下の手順を実行します。

1. **[ファイル]**→**[新規作成]**→**[プロジェクト]**を選択します。
2. **[新しいプロジェクトを作成する]**ウィンドウで、**[ASP.NET Web アプリケーション(.NET Framework)]**を選択し、[**次へ**]をクリックします。

Copyright © GrapeCity, Inc. All rights reserved.

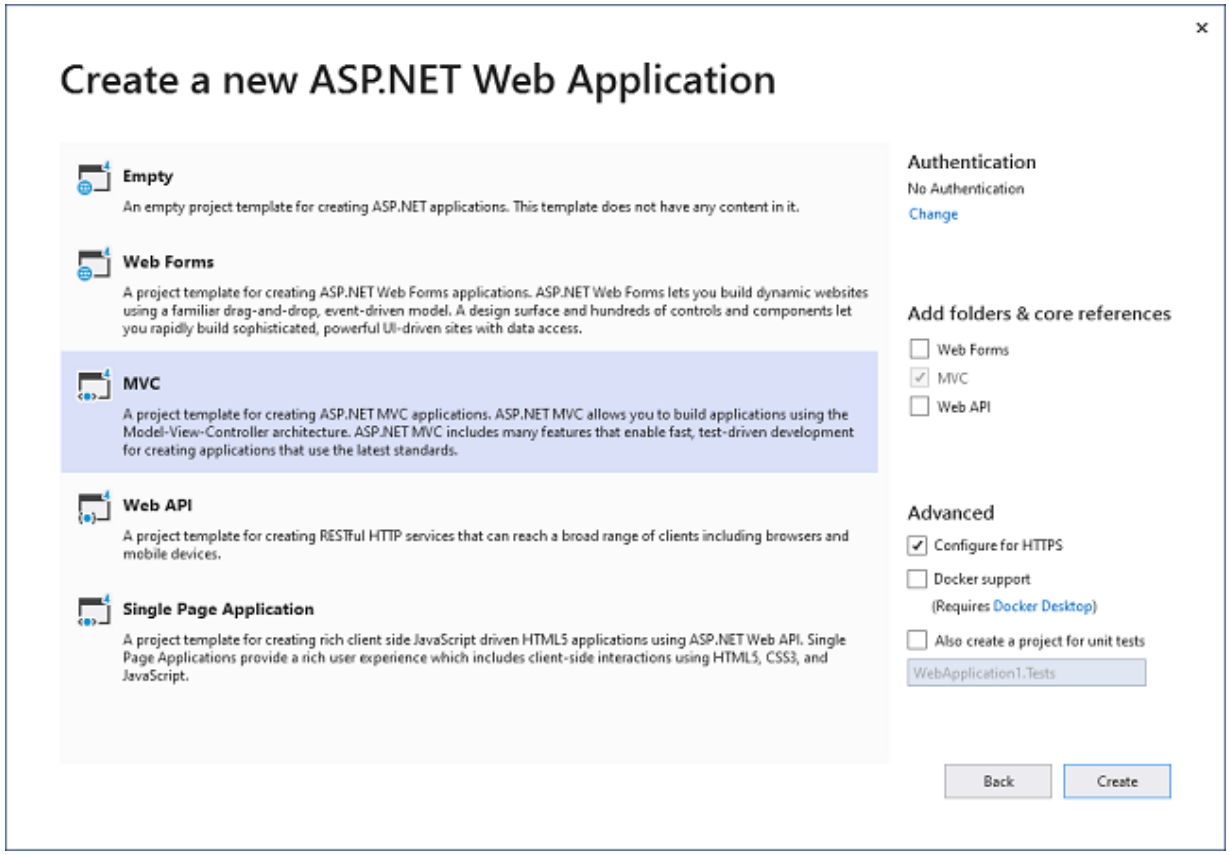


3. **[新しいプロジェクトを構成します]**ウィンドウで、アプリケーションの名前と場所を設定します。[**作成**]をクリックします。



4. [**新しいASP.NET Webアプリケーションを作成する**]ウィンドウで、[**MVC**]を選択し、[**作成**]をクリックします。

Copyright © GrapeCity, Inc. All rights reserved.



プロジェクトの設定

基本のVisual Studioテンプレートを使用して作成されたプロジェクトを設定するには、多少の追加手順が必要になります。

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開し、`web.config` ファイルをダブルクリックして開きます。
2. 次のマークアップを `<system.web.webPages.razor></system.web.webPages.razor>` マークアップ内に追加します。

```
HTML
<add namespace="C1.Web.Mvc" />
<add namespace="C1.Web.Mvc.Fluent" />
```

3. (オプション)アプリケーションに特定のコントロールを追加するには、次のマークアップをC1.Web.Mvcマークアップの下の<namespaces></namespaces>タグ間に追加する必要があります。たとえば、次の図では、株価チャートコントロールのマークアップを追加しています。

コントロール	マークアップ
Financial Chart	`<add namespace="C1.Web.Mvc.Finance" />` `<add namespace="C1.Web.Mvc.Finance.Fluent" />`
FlexSheet	`<add namespace="C1.Web.Mvc.Sheet" />` `<add namespace="C1.Web.Mvc.Sheet.Fluent" />`
OLAP	`<add namespace="C1.Web.Mvc.Olap" />` `<add namespace="C1.Web.Mvc.Olap.Fluent" />`
MultiRow	`<add namespace="C1.Web.Mvc.MultiRow" />` `<add namespace="C1.Web.Mvc.MultiRow.Fluent" />`
FlexViewer	`<add namespace="C1.Web.Mvc.Viewer" />` `<add namespace="C1.Web.Mvc.Viewer.Fluent" />`

Copyright © GrapeCity, Inc. All rights reserved.

TransposedGrid	<add namespace="C1.Web.Mvc.TransposedGrid" /> <add namespace="C1.Web.Mvc.TransposedGrid.Fluent" />
TransposedMultiRow	<add namespace="C1.Web.Mvc.TransposedMultiRow" /> <add namespace="C1.Web.Mvc.TransposedMultiRow.Fluent" />

```
<system.web.webPages.razor>
  <host factoryType="System.Web.Mvc.MvcWebRazorHostFactory, System.Web.Mvc, Version=5.1.0.0, Culture=neutral
  <pages pageBaseType="System.Web.Mvc.WebViewPage">
    <namespaces>
      <add namespace="System.Web.Mvc" />
      <add namespace="System.Web.Mvc.Ajax" />
      <add namespace="System.Web.Mvc.Html" />
      <add namespace="System.Web.Optimization"/>
      <add namespace="System.Web.Routing" />
      <add namespace="C1MvcWebApplication3" />
      <add namespace="C1.Web.Mvc" />
      <add namespace="C1.Web.Mvc.Fluent" />
      <add namespace="C1.Web.Mvc.Finance" />
      <add namespace="C1.Web.Mvc.Finance.Fluent" />
    </namespaces>
  </pages>
</system.web.webPages.razor>

<appSettings>
  <add key="webpages:Enabled" value="false" />
</appSettings>
```

4. Web.configファイルに加えられた変更を保存します。
5. **ソリューションエクスプローラー**で、プロジェクトを右クリックし、**[追加]**→**[新しい項目]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
6. **[新しい項目の追加]**ダイアログで、**[C#]**→**[一般]**を選択し、右ペインで**[テキストファイル]**を選択します。
7. テキストファイルに「**licenses.licx**」という名前を付けます。
8. 次のコードをテキストファイルに貼り付けます。

Licenses.licx
`C1.Web.Mvc.LicenseDetector, C1.Web.Mvc`

9. アプリケーションで特定のコントロールを使用するには、そのコントロールに応じてLicenses.licxに次のコードを追加します。詳細については、「ライセンス」を参照してください。

コントロール	**マークアップ**
Financial Chart	`C1.Web.Mvc.Finance.LicenseDetector, C1.Web.Mvc.Finance`
FlexSheet	`C1.Web.Mvc.Sheet.LicenseDetector, C1.Web.Mvc.FlexSheet`
OLAP	`C1.Web.Mvc.Olap.LicenseDetector, C1.Web.Mvc.Olap`
MultiRow	`C1.Web.Mvc.MultiRow.LicenseDetector, C1.Web.Mvc.MultiRow`
FlexViewer	`C1.Web.Mvc.Viewer.LicenseDetector, C1.Web.Mvc.FlexViewer`

10. プロジェクトに ASP.NET MVC の参照を追加します。ソリューションエクスプローラーで、参照を右クリックして「参照の追加」を選択します。「**C:\Program Files (x86)\ComponentOne\ASP.NET MVC\bin**」パスへ移動して、**C1.Web.Mvc.dll** を選択し「追加」をクリックします。C1.Web.Mvc.dllの「ローカルコピー」プロパティをTrueに設定します。
11. (オプション)MVCアプリケーションで特定のコントロールを使用するには、そのコントロールに応じた参照を追加する必要があります。

コントロール	**アセンブリ(場所 - C:\Program Files (x86)\ComponentOne\ASP.NET MVC\bin)**

Copyright © GrapeCity, Inc. All rights reserved.

Financial Chart	C1.Web.Mvc.Finance.dll
FlexSheet	C1.Web.Mvc.FlexSheet.dll
OLAP	C1.Web.Mvc.Olap.dll
MultiRow	C1.Web.Mvc.MultiRow.dll
FlexViewer	C1.Web.Mvc.FlexViewer.dll

12. 上記の手順が完了したら、アプリケーションで使用するコントロールのリソースを登録します。詳細については、「リソースの登録」を参照してください。

リソースの登録

デフォルトでは、ASP.NET MVC テンプレートを使用して新しいアプリケーションを作成すると、提供されているコントロールを使用するために必要なすべてのリソースが登録されます。ただし、標準の Visual Studio テンプレートを使用してプロジェクトを作成した場合は、リソースを手作業で登録する必要があります。

ASP.NET MVC コントロールを使用するために必要なリソースを登録するには、次の手順を実行します。

1. **ソリューションエクスプローラー**で、**[ビュー]**→**[共有]**フォルダを開きます。
2. **_Layout.cshtml** をダブルクリックして開きます。
3. 次のコードを <head></head> タグの間に追加します。この手順では、**FinancialChart**、**FlexSheet**、**FlexViewer**、**MultiRow**、**OLAP**、**TransposedGrido**や**TransposedMultiRow**コントロールを除く、アプリケーションで使用されるすべてのMVCコントロールを登録します。

```
Razor
@Html.C1().Styles()@Html.C1().Scripts().Basic()
```

4. （任意）アプリケーションで**FinancialChart**、**FlexSheet**、**FlexViewer**、**MultiRow**、**OLAP**、**TransposedGrido**や**TransposedMultiRow**コントロールなどの特定のコントロールを使用する場合、_Layout.cshtmlで上記のコードを次のコードに置き換えます。プロジェクトの要件に応じて、コントロールを追加または削除することができます。

```
Razor
@Html.C1().Styles()@Html.C1().Scripts().Basic().Finance().FlexSheet().FlexViewer().Olap().MultiRow().TransposedGrid().TransposedMultiRow()
```

ASP.NET MVC は、**条件付きリソース登録**をサポートします。これにより、アプリケーションで使用するリソースだけを登録でき、アプリケーションを軽量にすることができます。条件付きリソース登録をサポートするコントロールは次のとおりです。

- Grid
- Chart
- CollectionView
- Input
- Gauge

たとえば、次のコードは、FlexGrid および FlexChart を使用するために必要なリソースを登録します。

```
Razor
@Html.C1().Scripts().Basic(b=>b.Grid().Chart())
```

同様に、CollectionView、Input、およびGaugeを使用するために必要なリソースを登録することもできます。

ビューで_Layout.cshtmlが使用されていない場合は、リソースを登録するには、ビューページの先頭に上記のコードを追加してください。

スタイル

MVCアプリケーションでスタイルを使用するには、テーマやカルチャのリソースが含まれるスタイル（CSS）ファイルを登録する必要があります。また、Themeメソッドを使用してさまざまなテーマを適用することもできます。サポートされるテーマは、**C1.Web.Mvc.Themes** 静的クラスで定義されています。

```
_Layout.cshtml
@Html.C1().Styles().Theme("cocoa")
```

遅延スクリプト

HTMLスクリプト要素（<script>）は、HTMLまたはXHTMLドキュメント内で実行可能スクリプトを埋め込みまたは参照するために使用されます。非同期または遅延属性なしのスクリプトは、インラインスクリプトと同様に、ブラウザがページの解析を続行する前に即座に取得および実行されます。業界標準に従い、すべてのスクリプトはHTMLページの下部に、スタイルシートは上部に置くことをお勧めします。C1スクリプトを下部で登録する場合は、コントロールの起動スクリプトも下部に登録します。

_Layout.cshtmlページに遅延スクリプトを追加するには、次の手順に従います。以下の図を参考に、遅延スクリプトを追加する手順を理解してください。

1. headタグ内でスタイルを登録します。
2. 終了<body>タグの前でスクリプトファイルを登録します。
3. 終了<body>タグの前で起動スクリプトを登録します。

Copyright © GrapeCity, Inc. All rights reserved.

```
<body>
    @Html.C1().FlexGrid()
    @Html.C1().InputDate()
    @Html.C1().FinancialChart()

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")

    @Html.C1().Scripts()
    @RenderSection("scripts", required: false)
    @Html.C1().DeferredScripts()
</body>
</html>
```

MVCコントロールのスクリプトには、必要なスクリプトファイルと起動スクリプトという2つの部分があります。MVC 3/4/5およびASP.NET Coreアプリケーションで遅延スクリプトを有効にするためのさまざまな方法が提供されています。遅延スクリプトは、アプリケーションレベルとページレベルという2つのスコープで適用できます。名前が示すように、アプリケーションレベルは、アプリケーション内のすべてのComponentOneコントロールが遅延スクリプトを使用することを意味し、ページレベルは、ページで定義されているComponentOneコントロールだけが遅延スクリプトを使用することを意味します。

Web.config

```
@Html.C1().DeferredScripts()
```

遅延スクリプト - スイッチャー

ユーザーがスイッチャーをオンにすると、コントロールの起動スクリプトが一時停止されます。

アプリケーションレベルでは、現在のRazorページが最初にレンダリングされ、その後、_layoutページがレンダリングされます。この場合、ユーザーは、遅延スクリプトのアプリケーションレベルのスイッチャーをweb.configファイルまたはstartup.csファイルに移動する必要があります。

スイッチャーレベルでは、コントローラーのActionメソッドでEnableDeferredScriptsをtrueに設定すると、ページ内のComponentOne MVCコントロールのすべての起動スクリプトが一時停止されます。

Web.config

```
<configuration>
<appSettings>
<add key="C1:EnableDeferredScripts" value="true" />
```


Copyright © GrapeCity, Inc. All rights reserved.

コントロールの追加

This topic will help you understand how you can add an ASP.NET MVC control in your application using the following ways.

Razor 構文の使用
: Learn about adding controls in your application using Razor Syntax.

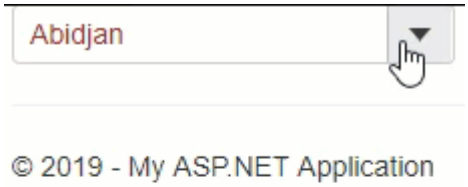
スキャフォールディングの使用
: Learn about adding controls in your application using C1 Scaffolder.

Razor 構文の使用

このトピックでは、アプリケーションに ComponentOne ASP.NET MVC コントロールを追加する方法を示します。また、MVCアプリケーションにモデル、ビュー、およびコントローラーを追加する方法についても説明します。WebアプリケーションでMVCコントロールを使用すると、コードの複雑さを低減し、柔軟性を高めることができ、コードの再利用に役立ちます。MVCコントロールの機能の詳細については、「**MVCの基本**」を参照してください。

必要な参照がプロジェクトに含まれていることを確認します。詳細については、「インストール」を参照してください。

次の図は、上記の手順を実行した後の**ComboBox**を示しています。



モデルの追加

コントロールのデータを取得するにはモデルが必要です。詳細については、「**MVC の基本**」を参照してください。必要によりモデルを追加します。この例では、モデルが使用されていないが、アプリケーションにモデルを追加する方法の手順が示しています。詳細については、「**MVC の基本**」を参照してください。

1. **ソリューションエクスプローラー**で、**[モデル]**フォルダを右クリックし、**[追加]**→**[クラス]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、クラスの名前（例：`Cities.cs`）を設定します。
3. **Cities.cs**モデルで次のコードを追加します。Citiesクラスを使用して国のリストを表しています。

C#

```
public class Cities
    {
        public static List<string> GetCities()
        {
            return new List<string>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            {
                "Abidjan", "Accra", "Ahmedabad", "Alexandria", "Ankara",
"Atlanta", "Baghdad",
                "Bandung", "Bangkok", "Barcelona", "Beijing", "Belo Horizonte",
"Bengaluru",
                "Bogota", "Boston", "Buenos Aires", "Cairo", "Calcutta",
"Chengdu", "Chennai",
                "Chicago", "Chongqung", "Dalian", "Dallas", "Delhi", "Detroit",
"Dhaka",
                "Dongguan", "Essen", "Fuzhou", "Guadalajara", "Guangzhou",
"Hangzhou", "Harbin",
                "Ho Chi Minh City", "Hong Kong", "Houston", "Hyderabad",
"Istanbul", "Jakarta",
                "Johannesburg", "Karachi", "Khartoum", "Kinshasa", "Kuala Lumpur",
"Lagos",
                "Lahore", "Lima", "London", "Los Angeles", "Luanda", "Madrid",
"Manila",
                "Medellin", "Mexico City", "Miami", "Milan", "Monterrey",
"Moscow", "Mumbai",
                "Nagoya", "Nanjing", "Naples", "New York", "Osaka", "Paris",
"Pheonix",
                "Philadelphia", "Porto Alegre", "Pune", "Qingdao", "Quanzhou",
"Recife",
                "Rio de Janeiro", "Riyadh", "Rome", "Saint Petersburg",
"Salvador", "San Francisco",
                "Santiago", "Sao Paulo", "Seoul", "Shanghair", "Shenyang",
"Shenzhen", "Tianjin",
                "Singapore", "Surabaya", "Surat", "Suzhou", "Sydney", "Taipei",
"Tehran",
                "Toronto", "Washington", "Wuhan", "Xi'an-Xianyang", "Yangoon",
"Zhengzhou", "Tokyo"
            };
        }
    }
```

アプリケーションに新しいクラスが追加されます。

先頭に戻る

コントローラーの追加

コントローラーは単純なクラスファイルです。このファイルは、アプリケーションへの着信要求、データを取得し、およびクライアント に応答を返すビューテンプレートを指定します。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. [MVC 5 コントローラー -空]テンプレートを選択します。
    2. コントローラーの名前を設定します（例：ComboBoxController)
    3. **[追加]**をクリックします。
4. Index() メソッドのコードを次のように置き換えます。

```
C#
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@using <ApplicationName>.Models

public ActionResult Index()
{
    ViewBag.Cities = Cities.GetCities();
    return View();
}
```

アプリケーションに新しいコントローラーが追加されます。

先頭に戻る

コントロールの追加

ビューを使用して、モデルを視覚的に表すことができます。ビューは、通常、モデルに関連付けられ、コントローラーを利用して必要なデータを取得します。ここでは、Index.cshtmlにコードを追加して、ブラウザに**ComboBox**コントロールを表示します。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：ComboBoxController）をダブルクリックして開きます。
2. メソッド Index() 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。

コントローラーにビューが追加されます。

- **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
- Index.cshtml をダブルクリックして開きます。
- Index.cshtml ファイルのデフォルトのコードを次のコードに置き換えて、**ComboBox** コントロールを初期化します。

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. Index.cshtml をダブルクリックして開きます。
3. Index.cshtml ファイルのデフォルトのコードを次のコードに置き換えて、**ComboBox** コントロールを初期化します。

Razor

```
@{List<string> cities = ViewBag.Cities;}
<div>
@(Html.C1().ComboBox().Bind(cities).SelectedIndex(0))
</div>
```

先頭に戻る

カスタムCSSの適用

CSSクラスを定義することで、コントロールの外観を定義できます。これらのクラスは、カスタムCSSルールを使用してさらにカスタマイズして、コントロールの外観を変更できます。次の例では、カスタムのCSSをComboBoxコントロールに適用して、その外観をカスタマイズする方法を示しています。カスタムCSSの使用の詳細については、スタイル設定とCSSを参照してください。

C#

```
<style>
    .wj-combobox {
        color: brown;
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .wj-listbox-item {
        font-size: 11pt;
        font-weight: bold;
        font-family:Calibri;
        background-color:burlywood;
    }
</style>
```

プロジェクトのビルドおよび実行

1. [**ビルド**]→[**ソリューションのビルド**]をクリックして、プロジェクトをビルドします。
2. [**F5**]キーを押してプロジェクトを実行します。

ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**ComboBox/index**）、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

スキャフォールディングの使用

ASP.NETスキャフォールディングは、ASP.NET MVCアプリケーション向けのコード生成フレームワークです。プロジェクトにスキャフォールディングを追加することで、データモデルとやり取りするコードをすばやく追加できます。スキャフォールディングを使用すると、プロジェクトで標準的なデータ操作を開発するためにかかる時間を削減できます。

ComponentOne ASP.NET MVC では、**FlexGrid**、**MultiRow**、**FlexSheet**、**FlexChart**、**FlexPie**、**Sunburst**、**FlexRadar**、**Input**の各コントロール用のスキャフォールディングが提供されています。

スキャフォールディングでC1 MVCコントロールの追加

ASP.NET MVCのコントロールをスキャフォールディングする手順は、次のとおりです。

データソースの設定

アプリケーションのデータソースを設定するために、C1NWindデータベース（C1NWind.mdfファイル）を使用することにします。このファイルは、システムのC:\Users\<ユーザー名>\Documents\ComponentOne Samples\ASP.NET MVC\MVC\MvcExplorer\App_Dataに用意されています。

アプリケーションのデータベースを生成する手順は、次のとおりです。

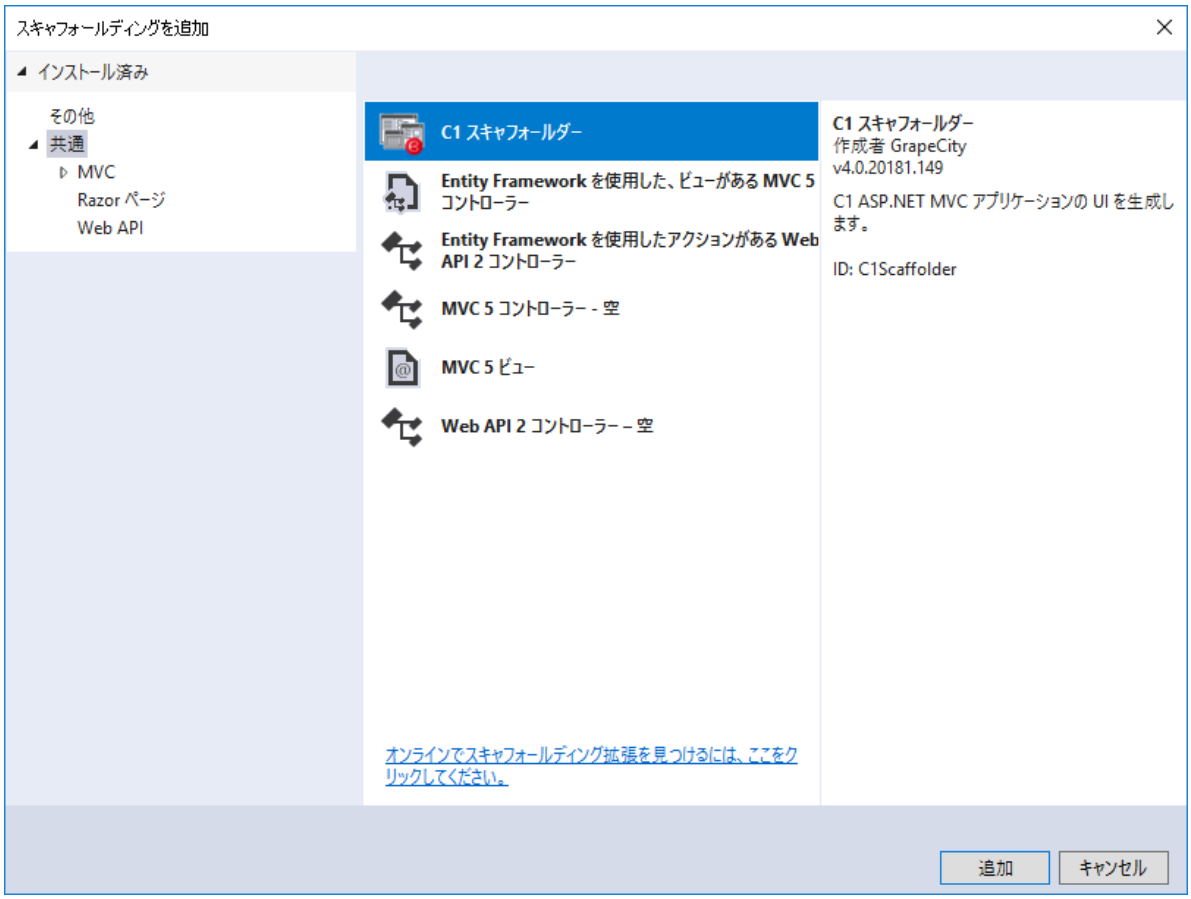
1. **C1NWind.mdf** ファイルをソリューションエクスプローラーのApp_Dataフォルダに追加します。
2. ソリューションエクスプローラーで右クリックして[**モデル**]→[**新しい項目の追加**]→[**データ**]を選択し、ADO.NETエンティティデータモデルを選択します。
3. モデルに**C1NWind**と名前を付け、[**追加**]をクリックします。
4. エンティティデータモデルウィザードで、[**データベースからEF Designer**]を選択し、[**次へ**]をクリックします。**C1NWind.mdf**データベースがデータ接続ドロップダウンに追加されます。
5. [次へ]をクリックし、**Entity** Frameworkのバージョンを選択し、[**次へ**]をクリックします。
6. [**データベースオブジェクトと設定を選択**]で、テーブルを選択し、[**完了**]をクリックします。
   アプリケーションのデータソースを正常に構成された場合、**C1NWind.edmx**ファイルが**Models**フォルダーの下のプロジェクトに追加されます。

先頭に戻る

スキャフォールダーの追加

1. ソリューションエクスプローラーで、プロジェクト名を右クリックし、[追加]→[新規スキャフォールディングアイテム]を選択します。[スキャフォールディングを追加]ウィザードが表示されます。
2. [スキャフォールディングを追加]ウィザードで、[共通]を選択し、右ペインからC1 スキャフォールダーを選択します。[共通]→[MVC]→[コントローラー]、または[共通]→[MVC 5 ビュー]を選択し、C1 スキャフォールダーを選択して、コントローラーまたはビューだけを追加することもできます。

Copyright © GrapeCity, Inc. All rights reserved.



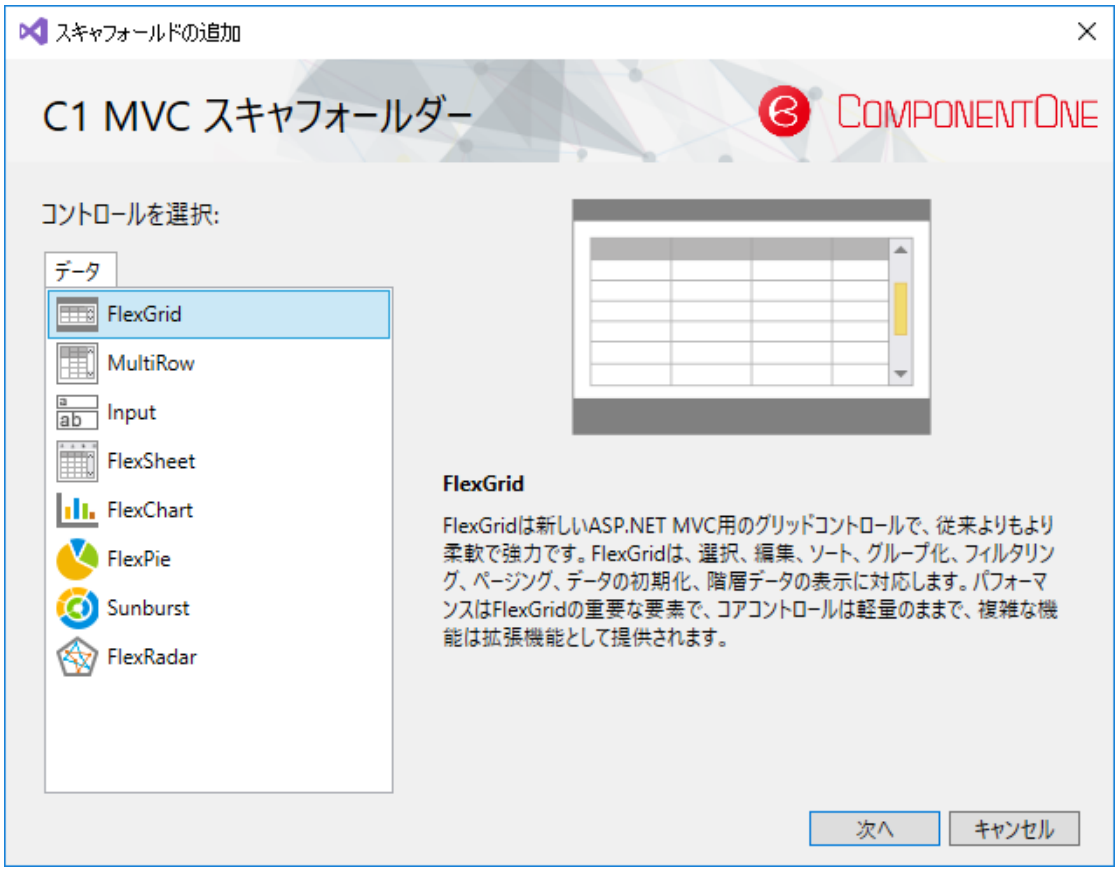
3. [追加]をクリックします。

先頭に戻る

コントロールの追加

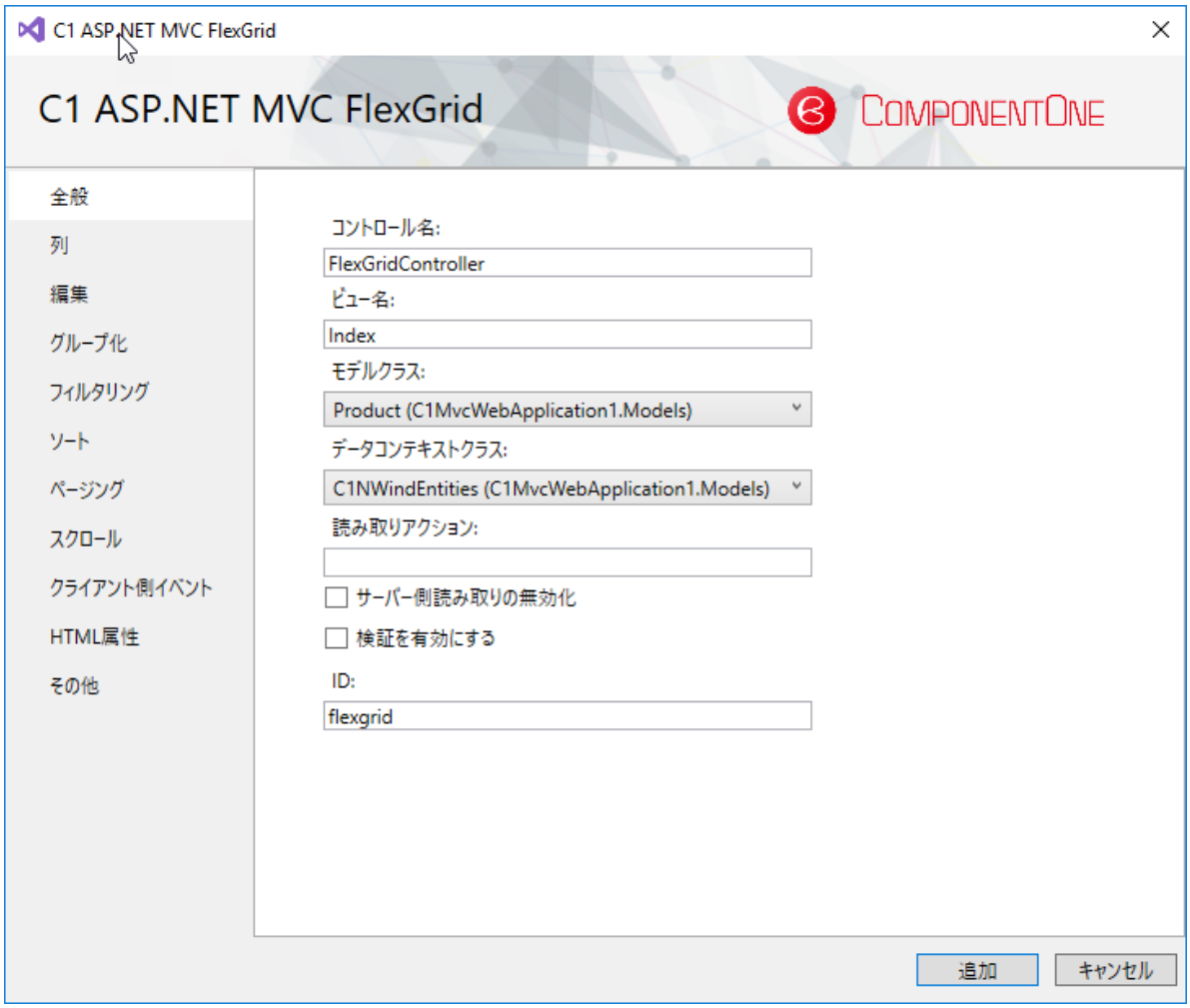
1. 必要に応じてコントロールを選択し、[次へ]をクリックします。この例では、**FlexGrid**コントロールを選択します。

Copyright © GrapeCity, Inc. All rights reserved.



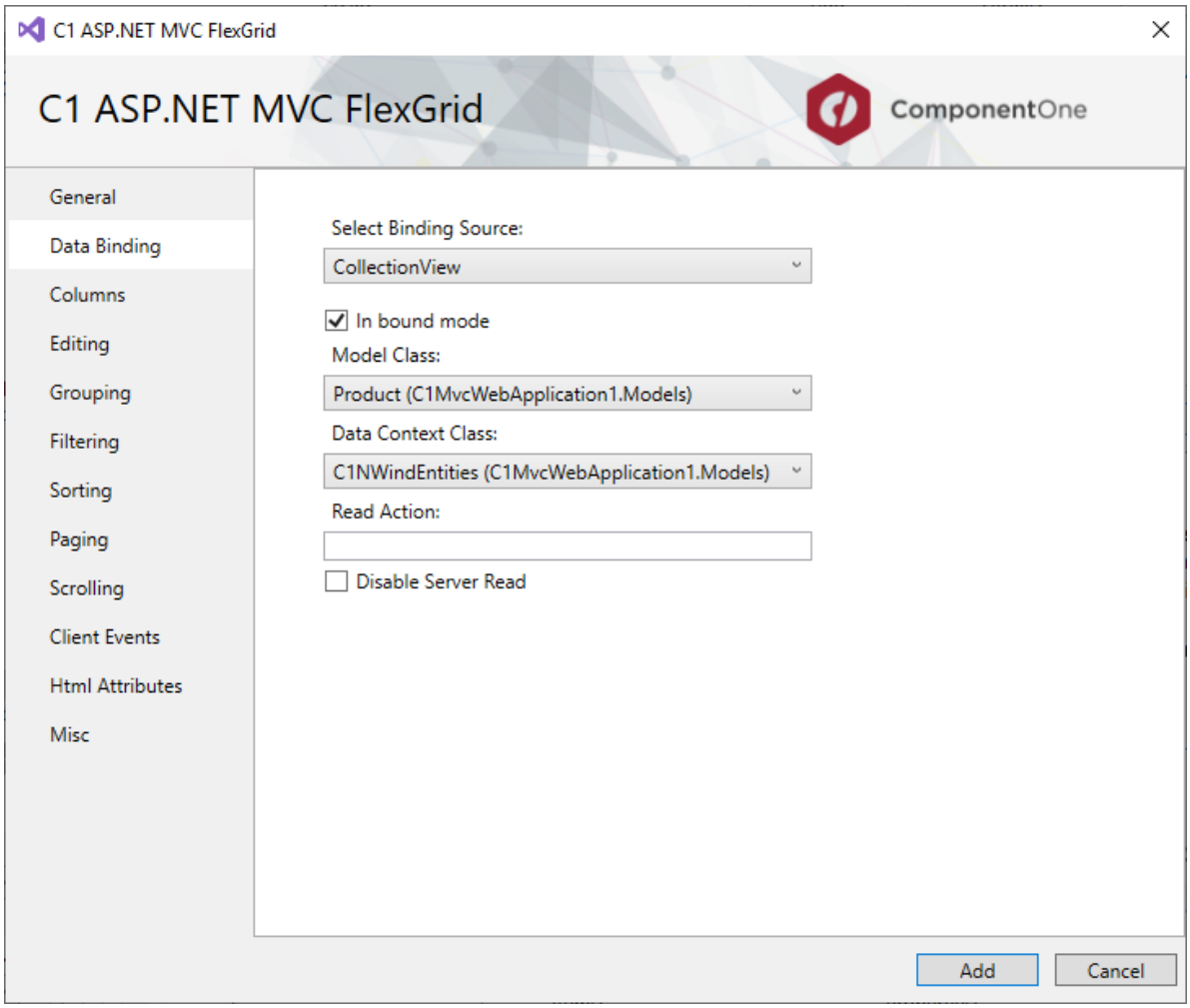
[**C1 ASP.NET MVC FlexGrid**]ウィザードが表示され、デフォルトで[**全般**]タブが選択されます。

Copyright © GrapeCity, Inc. All rights reserved.



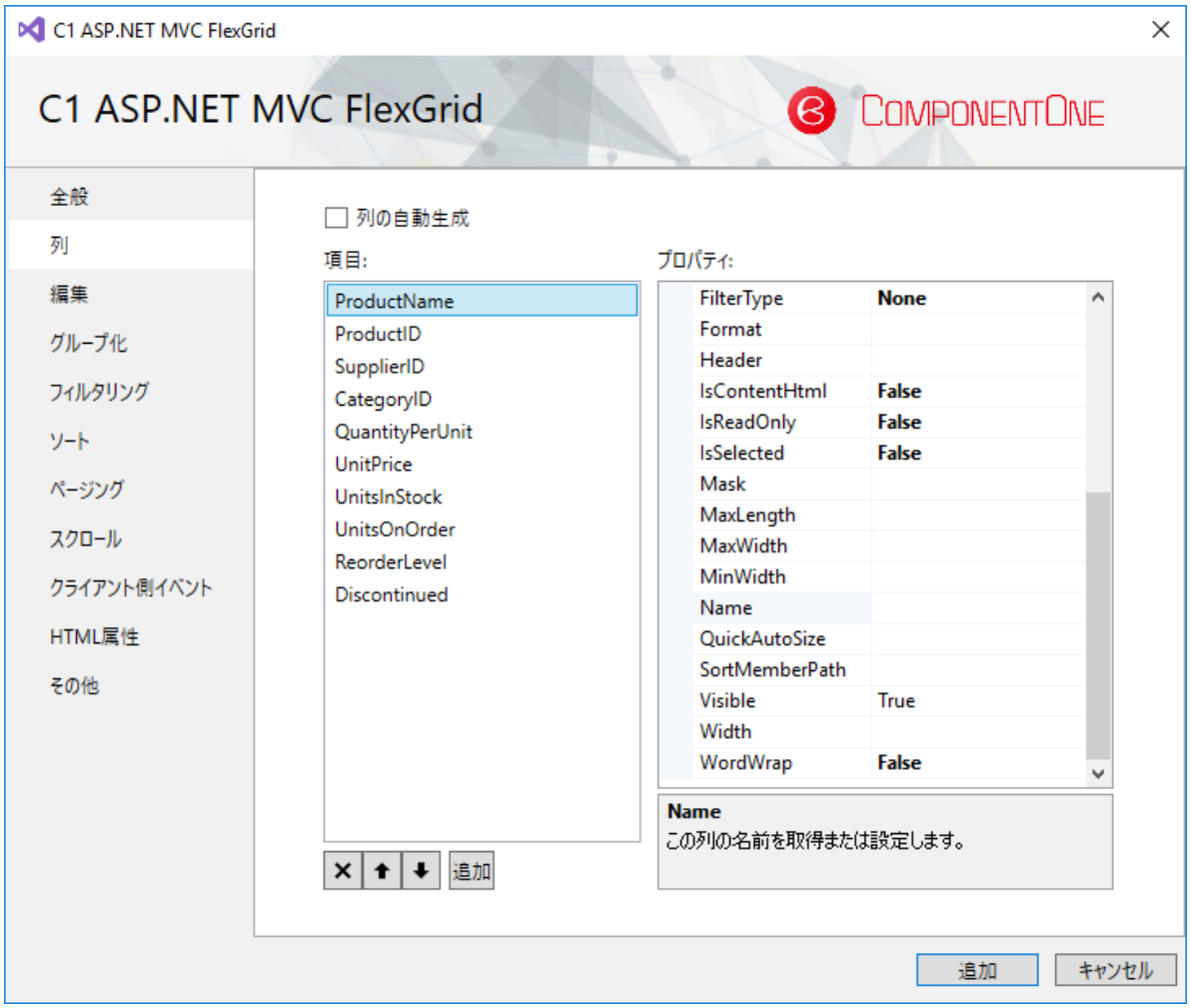
2. [**データ連結**]タブで、次を指定します。
    1. [**連結ソースの選択**]ドロップダウンから**CollectionView**を選択します。
    2. [**連結モード**]チェックボックスを選択して、**モデルクラス**と**データコンテキストクラス**を選択できるようにします。
    3. ドロップダウンリストから[**モデルクラス**]を選択します。リストには、手順1で追加したC1NWind.edmxモデルに加えて、アプリケーションで使用できるすべてのモデルタイプが表示されます。この例では、Productを選択して、FlexGridに製品を挿入します。
    4. ドロップダウンリストから[**データコンテキストクラス**]を選択します。この例では、**C1NWindEntities**を選択しています。

Copyright © GrapeCity, Inc. All rights reserved.



3. [**列**]タブに移動し、FlexGridコントロールの列を指定します。デフォルトでは、[**列の自動**]がオンになっています。オフにすると、列を追加および削除したり、最終的なビューに表示する順序になるように上下に移動することができます。この例では、次の図に示すように列を選択します。

Copyright © GrapeCity, Inc. All rights reserved.



4. [**編集**]タブに移動し、[**編集の許可**]チェックボックスと[**行削除の許可**]チェックボックスをオンにします。
5. [**グループ化**]タブに移動します。[**グループ化の設定**]で、[グループ化の設定]から[グループ行の表示]チェックボックスと[グループヘッダの書式]の[グループ化の条件]から[CategoryID]チェックボックスをオンにし、「Group by Category ID」という名前を付けます。
6. [**フィルタリング**]タブに移動し、[**フィルタリングの許可**]チェックボックスをオンにします。他の設定は、デフォルトのままにします。
7. [**ソート**]タブに移動し、設定はデフォルトのままにします。[**ソートの許可**]チェックボックスと[**ソート記号の表示**]チェックボックスはどちらもオンにします。
8. [**クライアント側イベント**]タブに移動し、[**BeginningEdit**]チェックボックスをオンにします。
9. [**追加**]をクリックします。
選択したモデルのコントローラーとビューがプロジェクトに追加されます。コントローラーとビューに対して生成されるコードは、次のようになります。

FlexGridController.cs

FlexGrid1Controller.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using C1.Web.Mvc;
using C1.Web.Mvc.Serialization;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Validation;
using System.Web;
using System.Web.Mvc;

//このコードはC1 スキャフォールダーによって生成されました。
namespace C1MvcWebApplication1.Controllers
{
    public class FlexGridController : Controller
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    {
        C1MvcWebApplication1.Models.C1NWindEntities db = new
C1MvcWebApplication1.Models.C1NWindEntities();

        public ActionResult Index()
        {

            return View(db.Products);
        }


        public ActionResult
Flexgrid_Update([C1JsonRequest]CollectionViewEditRequest<C1MvcWebApplication1.Models.Product>
requestData)
        {
            return Update(requestData, db.Products);
        }

        private ActionResult Update<T>(CollectionViewEditRequest<T> requestData, DbSet<T>
data) where T : class
        {
            return this.C1Json(CollectionViewHelper.Edit<T>(requestData, item =>
            {
                string error = string.Empty;
                bool success = true;
                try
                {
                    db.Entry(item as object).State = EntityState.Modified;
                    db.SaveChanges();
                }
                catch (DbEntityValidationException e)
                {
                    error = string.Join(",", e.EntityValidationErrors.Select(result =>
                    {
                        return string.Join(",", result.ValidationErrors.Select(err =>
err.ErrorMessage));
                    }));
                    success = false;
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }
                return new CollectionViewItemResult<T>
                {
                    Error = error,
                    Success = success && ModelState.IsValid,
                    Data = item
                };
            }, () => data.ToList<T>()));
        }

        public ActionResult
Flexgrid_Delete([C1JsonRequest]CollectionViewEditRequest<C1MvcWebApplication1.Models.Product>
requestData)
        {
            return Delete(requestData, db.Products, item => new object[] { item.ProductID });
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        private ActionResult Delete<T>(CollectionViewEditRequest<T> requestData, DbSet<T>
data, Func<T, object[]> getKeys) where T : class
        {
            return this.C1Json(CollectionViewHelper.Edit<T>(requestData, item =>
            {
                string error = string.Empty;
                bool success = true;
                try
                {
                    var resultItem = data.Find(getKeys(item));
                    data.Remove(resultItem);
                    db.SaveChanges();
                }
                catch (DbEntityValidationException e)
                {
                    error = string.Join(",", e.EntityValidationErrors.Select(result =>
                    {
                        return string.Join(",", result.ValidationErrors.Select(err =>
err.ErrorMessage));
                    }));
                    success = false;
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }
                return new CollectionViewItemResult<T>
                {
                    Error = error,
                    Success = success && ModelState.IsValid,
                    Data = item
                };
            }, () => data.ToList<T>()));
        }
    }
}
```

Index

Razor (Index.cshtml)

```
@using C1.Web.Mvc
@using C1.Web.Mvc.Fluent
@using C1.Web.Mvc.Grid
@model IEnumerable<C1MvcWebApplication1.Models.Product>


<script>
function flexgrid_BeginningEdit(sender, e){
    // beginningEditのイベントハンドラを実装します。
}

</script>

@(Html.C1().FlexGrid<C1MvcWebApplication1.Models.Product>()
    .AllowDelete(true).AutoGenerateColumns(false).GroupHeaderFormat("Group by Category
ID").OnClientBeginningEdit("flexgrid_BeginningEdit")
    .Columns(csb=>
    {
        csb.Add().Binding("ProductName");
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            csb.Add().Binding("CategoryID");
            csb.Add().Binding("QuantityPerUnit");
            csb.Add().Binding("UnitPrice");
            csb.Add().Binding("UnitsInStock");
            csb.Add().Binding("UnitsOnOrder");
            csb.Add().Binding("Discontinued");
            })
        .Filterable()
        .Bind(cvsb=>cvsb
            .GroupBy("CategoryID")

.Update(Url.Action("Flexgrid_Update")).Delete(Url.Action("Flexgrid_Delete")).Bind(Model))
    .Height("800px").Id("flexgrid"))
```

10. プロジェクトを実行します。

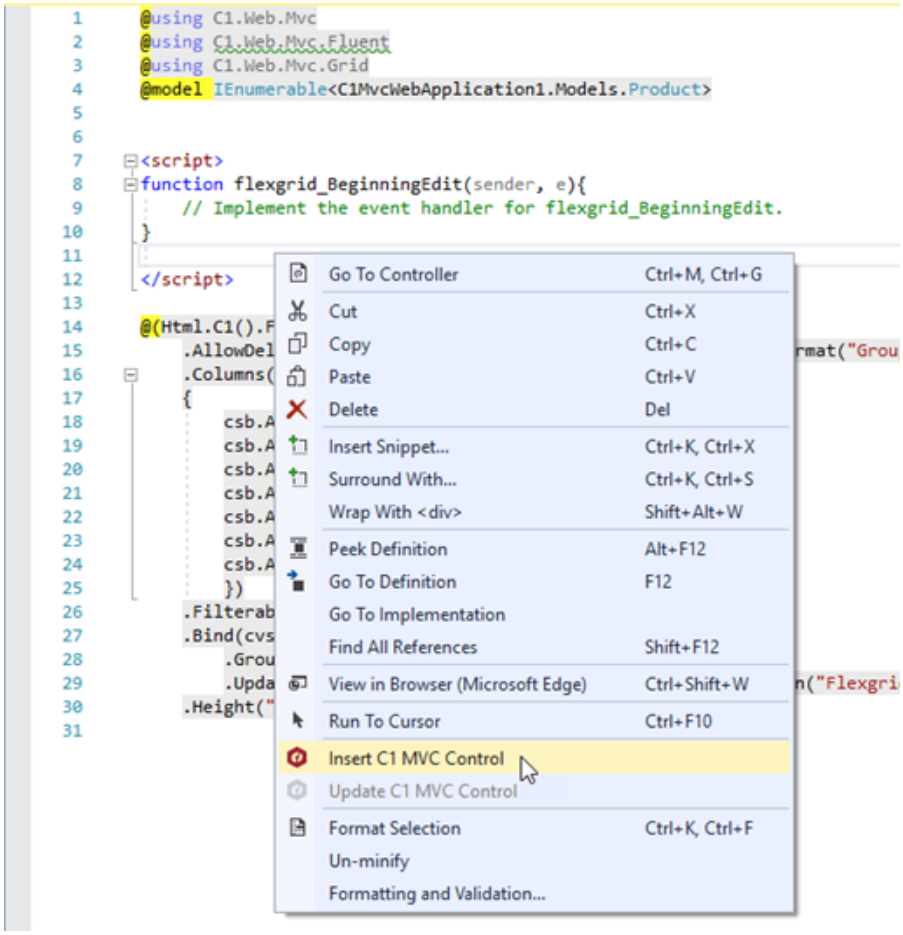
ProductNa...	CategoryID	QuantityP...	UnitPrice	UnitsInSto...	UnitsOnOr...	Discontin...
◢ Group by Category ID						
Chai	1	10 boxes x 20...	18	39	0	☑
Guaraná Fa...	1	12 - 355 ml c...	4.50	20	0	☑
Sasquatch Ale	1	24 - 12 oz bot...	14	111	0	☐
Steeleye Stout	1	24 - 12 oz bot...	18	20	0	☐
Côte de Blaye	1	12 - 75 cl bottles	263.50	17	0	☐
Chartreuse ...	1	750 cc per bo...	18	69	0	☐
Ipoh Coffee	1	16 - 500 g tins	46	17	10	☐
Laughing Lu...	1	24 - 12 oz bot...	14	52	0	☐
Outback Lager	1	24 - 355 ml b...	15	15	10	☐
Rhönbräu Kl...	1	24 - 0.5 l bottles	7.75	125	0	☐
◢ Group by Category ID						
Uncle Bob's ...	7	12 - 1 lb pkgs.	30	15	0	☑
Tofu	7	40 - 100 g pkgs.	23.25	35	0	☐
Rössle Sau...	7	25 - 825 g cans	45.60	26	0	☑
Manjimup D...	7	50 - 300 g pkgs.	53	20	0	☐
Longlife Tofu	7	5 kg pkg.	10	4	20	☐
◢ Group by Category ID						
Mishi Kobe ...	6	18 - 500 g pkgs.	97	29	0	☑
Alice Mutton	6	20 - 1 kg tins	39	0	0	☑
Thüringer R...	6	50 bags x 30 ...	123.79	0	0	☑
Perth Pasties	6	48 pieces	32.80	0	0	☑

先頭に戻る

Insert C1 MVC Control

In addition to adding C1 MVC controls at the time of application creation, you can also insert C1 MVC controls in an existing application using the **Insert C1 MVC Control** option. The **Insert C1 MVC Control** option allows you to easily add a fully functional control in the view code of an application. This option can be accessed through the context menu displayed in the design area of a View file in ASP.NET MVC application.

Copyright © GrapeCity, Inc. All rights reserved.



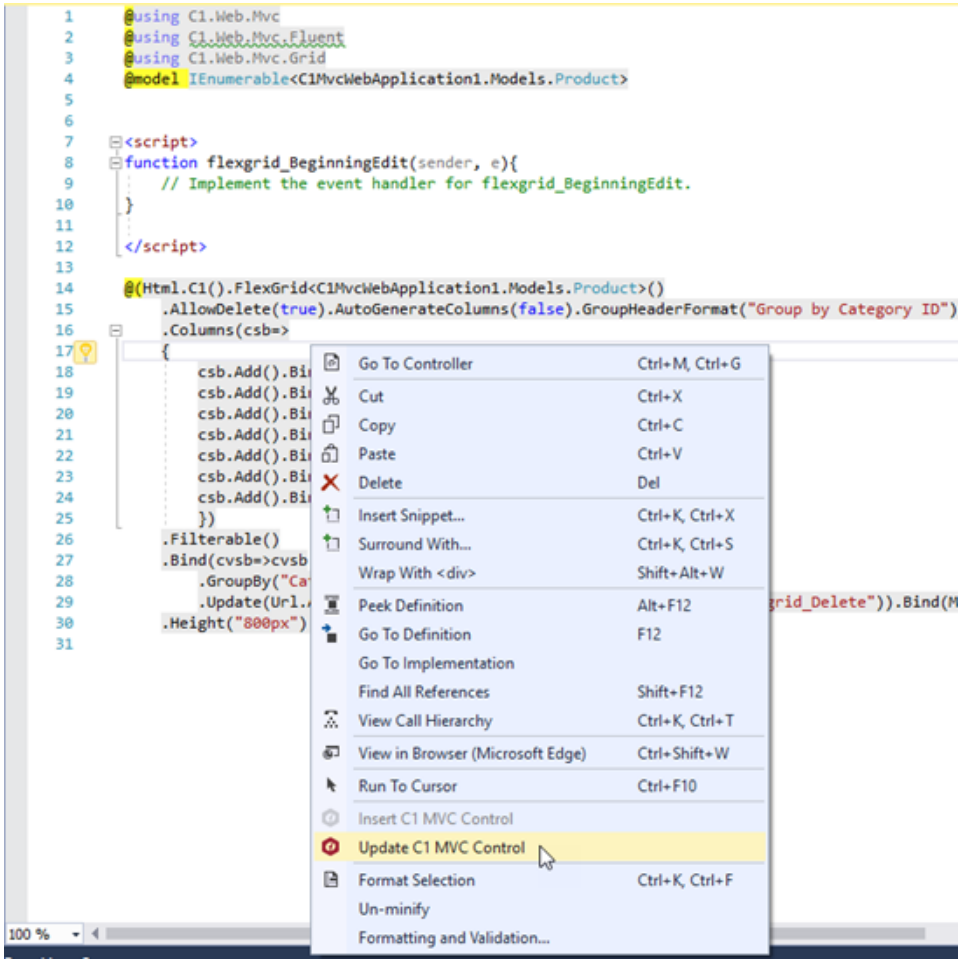
Selecting the **Insert C1 MVC Control** option from the menu invokes the **Add Scaffold** dialog, which can further be used the same way as described in step 2 and 3 of the above section.

先頭に戻る

Update C1 MVC Control

Besides inserting C1 MVC Controls, you can also update C1 MVC controls in an existing application using the **Update C1 MVC Control** option. The **Update C1 MVC Control** option allows you to update the properties of a control. This option can be accessed through the context menu displayed in the design area of a View file in ASP.NET MVC application.

Copyright © GrapeCity, Inc. All rights reserved.



Selecting the **Update C1 MVC Control** option from the menu invokes the **Add Scaffold** dialog, which can further be used the same way as described in step 2 and 3 of the using C1 Scaffolder section above .

先頭に戻る

Copyright © GrapeCity, Inc. All rights reserved.

クライアント側のサポート

ComponentOne ASP.NET MVC Edition provides client-side API for client side support. The following topics will help you understand how to use client-side API for client-side support in MVC applications.

クライアント側のIntellisenseサポート
Learn about using Client, JavaScript and TypeScript IntelliSense.
クライアント側のメソッドおよびイベントの処理
Learn about using client-side methods and events to update or modify controls at client-side.

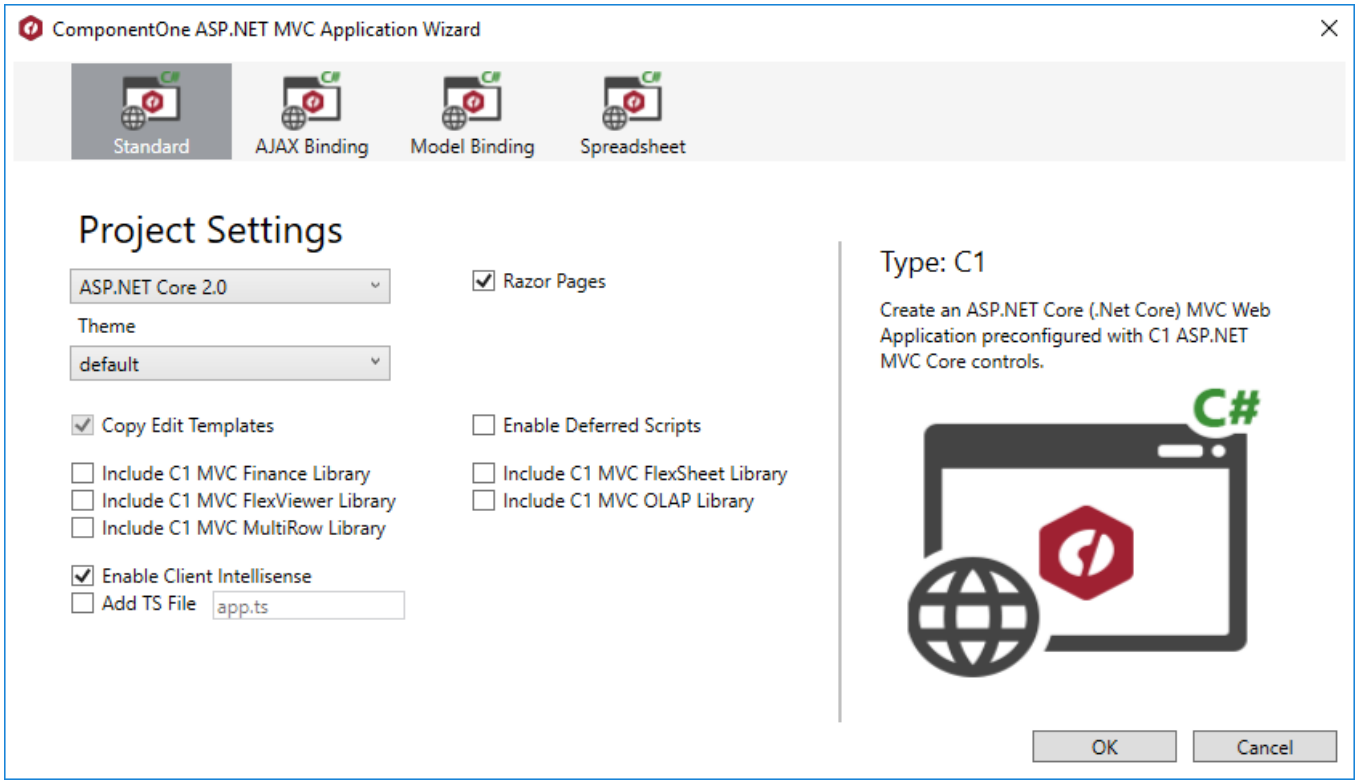
クライアント側のIntellisenseサポート

IntelliSenseを使用すると、開発者は、入力ミスやよくある間違いを減らすことで、アプリケーションのコーディングプロセスを迅速化できます。TypeScriptエディタでクライアントスクリプトの作業を行う際、IntelliSenseは、現在のコンテキストに基づいてオブジェクト、関数、プロパティなどを表示して作業を支援します。MVC IntelliSense によって提供されるポップアップリストからオプションを選択して、コードを完成できます。

- **TypeScript インテリセンス**
- **JavaScript インテリセンス**
- **Castメソッドの使用**
- **VisualStudio 2017のJavaScriptインテリセンス**

TypeScript インテリセンス

Visual StudioアプリケーションでMVCコントロールのIntelliSenseを使用するには、Visual Studioテンプレートウィザードで**[クライアント側のインテリセンスを有効にする]**チェックボックスをオンにします。



[クライアント側のインテリセンスを有効にする]オプションをオンにすると、**c1.mvc.core.lib.d.ts**が**Scripts\Typings**フォルダに自動的に追加されます。また、株価チャートまたはFlexSheetの使用時に、**ComponentOne ASP.NET MVCアプリケーション**

Copyright © GrapeCity, Inc. All rights reserved.

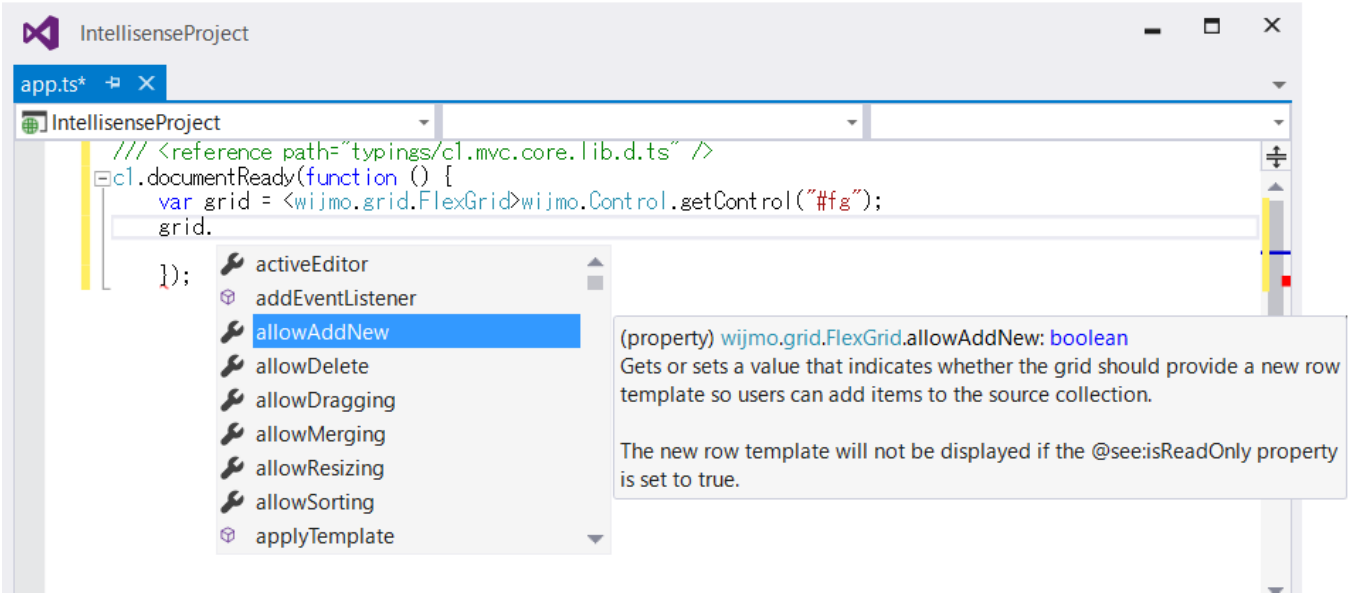
ウィザードで**[C1 MVC Financeライブラリの参照]**、**[C1 MVC FlexSheetライブラリの参照]**、**[C1 MVC FlexViewerライブラリの参照]**、**[C1 MVC MultiRowライブラリの参照]**、または**[C1 MVC OLAPライブラリの参照]**チェックボックスをオンにした場合は、**c1.mvc.finance.lib.d.ts**ファイル、**c1.mvc.flexsheet.lib.d.ts**ファイル、**c1.mvc.flexviewer.lib.d.ts**ファイル、**c1.mvc.multirow.lib.d.ts**ファイル、または**c1.mvc.OLAP.lib.d.ts**ファイルも同じフォルダに自動的に追加されます。

> **メモ**:標準のVisualStudioテンプレートを使用してアプリケーションを作成している場合は、アプリケーションでTSインテリセンスを有効にすることができます。これには、**C:\Program Files (x86)\ComponentOne\ASP.NET MVC Edition\TsTypings**にある型定義ファイルを追加して参照します。

[TSファイルを追加する]オプションを使用すると、TypeScriptファイルで作業を行う際にComponentOne MVC IntelliSenseを使用できます。[TSファイルを追加]オプションを選択すると、関連する**.d.ts**ファイルを参照する**.ts**ファイルがScriptsフォルダに追加されます。

```
@using IntellisenseProject.Models
@model IEnumerable<Sale>
<script src="~/Scripts/app.js"></script>
@{
    ViewBag.Title = "ホーム ページ";
}

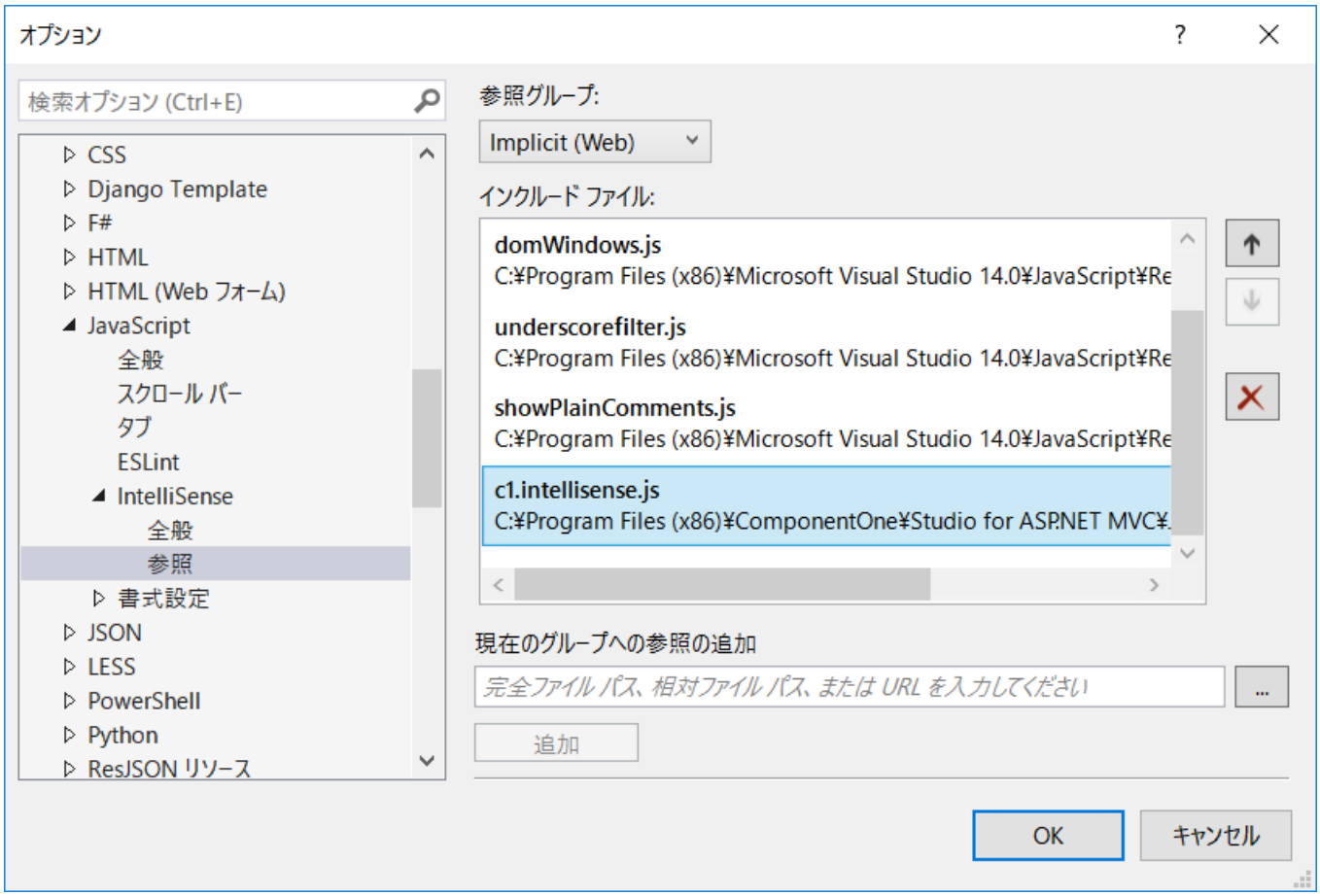
@(Html.C1().FlexGrid<Sale>().Id("fg").Bind(Model).AutoGenerateColumns(true))
```



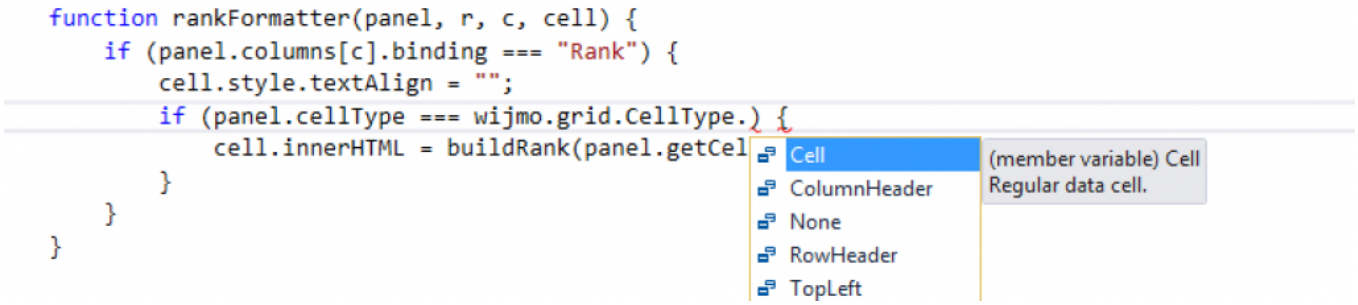
JavaScript インテリセンス

ComponentOne MVC では、JavaScript インテリセンスが導入されており、開発者はコードをすばやく記述して開発を行うことができます。ComponentOne インストーラを使用してComponentOne MVC をインストールすると、MVC用のJS インテリセンスファイルがVisual Studioに自動的に追加されます。

Copyright © GrapeCity, Inc. All rights reserved.



JavaScript インテリセンスサポートがVisual Studioにインストールされると、スクリプトコードを記述するときは常にクライアント側のインテリセンスが有効になります。

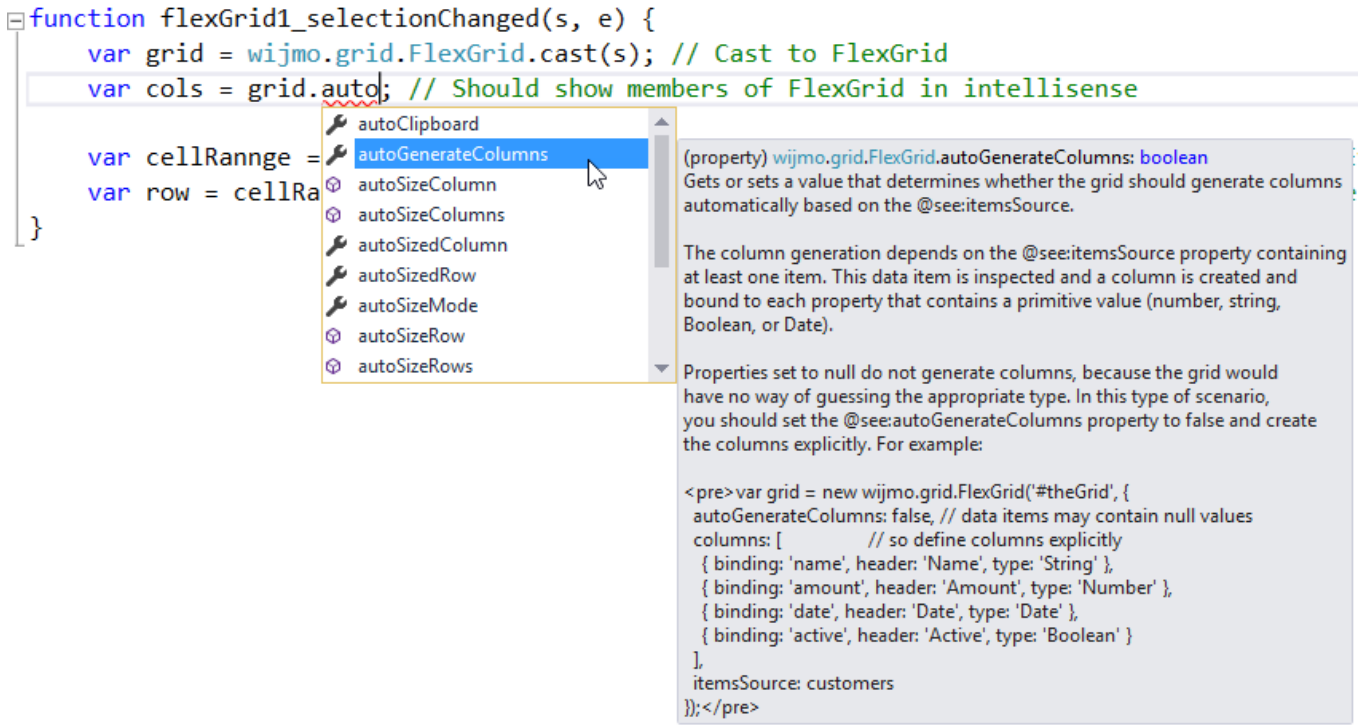


Castメソッドの使用

Castメソッドは、ComponentOne MVC ASP.NETコントロールのJavaScriptインテリセンスに含まれます。TypeScriptまたはJavaScriptファイルで、あるオブジェクトを別の型にキャストし、新しく作成された型のメンバをインテリセンスで利用できます。クライアント側APIを使用してJavaScriptコードを記述する際にインテリセンスを使用するには、**Cast**メソッドを使用すると便利です。

Castメソッドの使用例を以下に示します。

Copyright © GrapeCity, Inc. All rights reserved.

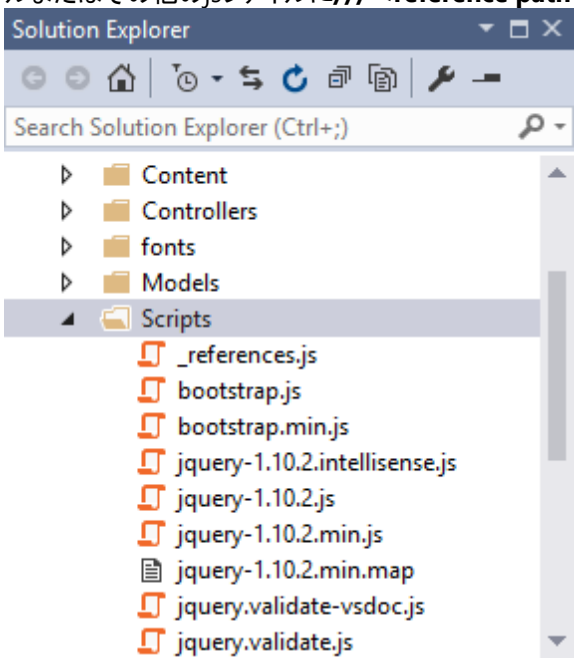


VisualStudio 2017のJavaScriptインテリセンス

Studio 2017でJavaScript IntelliSenseを有効にするには、MVCアプリケーションに**c1.mvc.basic.lib.d.ts** TypeScriptファイルを追加する必要があります。

ASP.NET

1. **ソリューションエクスプローラ**で、[**Scripts**]フォルダを右クリックし、**追加| 既存の項目**を選択します。
2. [**既存項目の追加**]ダイアログで、**C:\Program Files (x86)\ComponentOne\ASP.NET MVC Edition\TsTypings**から **c1.mvc.basic.lib.d.ts** を選択します。
3. JavaScriptファイルまたはTypeScriptファイルでJavaScript IntelliSenseを使用するには、**Scripts_references.js**ファイルまたはその他のjsファイルに**/// <reference path="c1.mvc.basic.lib.d.ts" />**参照を追加します。



Copyright © GrapeCity, Inc. All rights reserved.

同様
に、**c1.mvc.finance.lib.d.ts**、**c1.mvc.flexsheet.lib.d.ts**、**c1.mvc.flexviewer.lib.d.ts**、**c1.mvc.multirow.lib.d.ts**または**c1.mvc.OLAP.lib.d.ts**ファイルを使用して、アプリケーション固有のIntelliSenseを使用できます。

クライアント側のメソッドおよびイベントの処理

MVCコントロールには、高度なクライアント側APIが用意されています。これには、クライアント側で使用しているコントロールを更新または変更するためのメソッド、イベント、およびプロパティが含まれます。このため、サーバー側の処理とクライアント側の処理を組み合わせて、一貫したWebアプリケーションを開発することができます。イベント、メソッド、およびプロパティは、**.NET Framework**がイベント、メソッド、およびプロパティを処理する方法と同様に機能します。付属するクライアント側APIは、JavaScriptを使用して実装されています。クライアントAPIを実装するJavaScriptコードは、ビューファイルに埋め込まれるか、リソースファイルとして埋め込まれます。

- クライアント側メソッド
- クライアント側イベント

クライアント側メソッド

クライアント側APIには、コントロールの動作と外観を制御するさまざまなクライアント側メソッドが用意されています。アプリケーションでクライアント側APIメソッドを使用するには、コントロール宣言の後に、JavaScriptコードを追加する必要があります。このJavaScriptコードで、特定のコントロールのメソッド、イベント、およびプロパティを定義または設定することができます。

シナリオ

FlexGridコントロールを使用して、売上データ（Country、Product、Amount）を表示するとします。さらに、FlexGridで選択されている現在の行に基づいて、対応する列（Amount 2、Active、Discount）の値をラベル内に表示します。このシナリオは、クライアント側メソッドを使用して実装することができます。クライアント側メソッドの詳細については、「**クライアント側APIリファレンス**」を参照してください。

このシナリオを実装する前に、「**FlexGridクライアント側API**」セクションで最適なメソッドを特定する必要があります。**FlexGrid**クラスに、選択の変更の処理に役立つ**OnSelectionChanged**メソッドがあります。OnSelectionChangedメソッドを選択すると、引数タイプの詳細が提供されるようになります。この場合、引数タイプは**CellRangeEventArgs**です。CellRangeEventArgsは、そのメソッドで使用される**row**、**col**、**data**、**range**、**panel**の各引数を提供します。このシナリオに適切なメソッドを選択したので、FlexGridコントロールで実装を行うことができます。

次のコード例では、**selectionChanged**イベントを発生させるRazorコードで**OnSelectionChanged()**メソッドを使用しています。JavaScript関数**gridSelectionChange(s, e)**を作成し、**CollectionView**クラスを使用して各行の詳細を取得します。gridSelectionChange関数の**s**は送信元（FlexGrid）を表し、**e**は**CellRangeEventArgs**クラス型です。その下では、3つのラベル項目（Amount2、Active、Discount）の参照変数をそれぞれ宣言しています。CollectionViewクラスには、グリッドで現在選択されている項目を返す**currentItem**プロパティがあります。最後に、このプロパティを使用して3つの列の値を取得し、それらをラベルに割り当てています。

次のコード例は、クライアント側でOnClientSelectionChanged()メソッドを使用して各行の詳細を取得し、それをラベルに表示します。この例では、「FlexGrid：クイックスタート」トピックで作成した**Sale.cs**モデルを使用しています。

HTML Helpers

Razor

```
@using FlexGridClientSide.Models
@using C1.Web.Mvc.Grid

@model IEnumerable<Sale>
<script>
    function gridSelectionChange(s, e)
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            var cv =s.collectionView;

            var amt2 = document.getElementById('lblAmt2');
            var active = document.getElementById('lblActive');
            var discount = document.getElementById('lblDisc');
            if (cv.currentItem)
            {
                amt2.innerText = cv.currentItem.Amount2;
                active.innerText = cv.currentItem.Active;
                discount.innerText = cv.currentItem.Discount;
            }
    }
</script>

@(Html.C1().FlexGrid().Id("fg").Bind(Model).AutoGenerateColumns(false).Height("440px")
        .OnClientSelectionChanged("gridSelectionChange")
        .Columns(c => c.Add(cb => cb.Header("国名").Binding("Country"))
        .Add(cb => cb.Binding("Product").Header("製品名"))
        .Add(cb => cb.Binding("Amount").Header("金額")))
)
<div class="col-sm-4">
    <label>金額2:</label>
    <label id="lblAmt2"></label>
    <br />
    <label>アクティブ:</label>
    <label id="lblActive"></label>
    <br />
    <label>割引:</label>
    <label id="lblDisc"></label>
</div>
```

クライアント側イベント

クライアント側のイベントハンドラとコールバックは、JavaScriptコードを使用して実装されます。クライアント側イベントを処理するために、このコントロールには、イベント処理コードを直接受け取るか、処理関数の名前を受け取るプロパティが用意されています。この関数は、ビューコード内または個々のjsファイル内で初期化する必要があります。

シナリオ

売上データ(Country、Product、Amount)を表示し、**Amount**値が1,000ドル未満の場合は金額列データを赤色で表示するとします。このシナリオは、クライアント側イベントを使用して実装することができます。クライアント側イベントの詳細については、「**クライアント側APIリファレンス**」を参照してください。

このシナリオを実装する前に、「**FlexGridクライアント側API**」セクションで最適なイベントを特定する必要があります。次のシナリオでは、グリッドのセルを書式設定する必要があるため、**FlexGrid**クラスの**formatItem**イベントを使用します。formatItemイベントを選択すると、引数タイプの詳細が提供されるようになります。この場合、引数タイプは**FormatItemEventArgs**です。FormatItemEventArgsは、そのイベントで使用される**行、列、データ、範囲、パネル**の各引数を提供します。このシナリオに適切なイベントを選択したので、FlexGridコントロールで実装を行うことができます。

次のコード例では、Razorコード内でJavaScript関数名を割り当てることで、**OnClientFormatItem**イベントをサブスクライブしています。<Script>セクションでは、**formatItem**関数を宣言して、書式設定するセルがセルタイプかどうか、および書式設定する列が**Amount**かどうかをチェックしています。その下では、**getCellData**メソッドを使用してセルデータを取得してから、Amount列の値をチェックして、**cell.style.color**プロパティを使用して色を付けています。このコードを実行すると、1,000ドル未満の金額を含むセルは赤色で表示されます。

次のコード例は、FlexGridコントロールでOnClientFormatItemイベントを使用する方法を示します。この例では、「FlexGrid:ク

Copyright © GrapeCity, Inc. All rights reserved.

イックスタート」トピックで作成したSale.csモデルを使用しています。

HTML Helpers

Razor

```
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>
<script>
    function formatItem(s, e) {
        if (e.panel.cellType == wijmo.grid.CellType.Cell &&
e.panel.columns[e.col].binding == "金額") {
            var val = e.panel.grid.getCellData(e.row, e.col, false);
            if (val < 1000) {
                e.cell.style.color = 'red';
            }
        }
    }
</script>

@(Html.C1().FlexGrid().Id("fg").Bind(Model).AutoGenerateColumns(false).Height("440px")
        .OnClientFormatItem("formatItem")
        .Columns(c => c.Add(cb => cb.Header("国名").Binding("Country"))
        .Add(cb => cb.Binding("Product").Header("製品名"))
        .Add(cb => cb.Binding("Amount").Header("金額")))
)
```

addHandler()を使用するイベント処理

イベントを処理する方法として、.NETイベントと同様に、**addHandler**メソッドを使用することもできます。イベントをサブスクライブするにはaddHandlerメソッドを使用し、アンサブスクライブするにはremoveHandlerメソッドを使用します。addHandler文を使用して、イベントハンドラを指定することができます。さらに、addHandlerをremoveHandlerと組み合わせて使用すると、イベントに関連付けられたイベントハンドラを動的に追加、削除、および変更することができ、柔軟性が高まります。

次のコード例では、**addHandler**メソッドを使用して、**formatItem**イベントの機能を定義しています。

HTML Helpers

Razor

```
@model IEnumerable<Sale>
<script>
    var fg;
    c1.documentReady(function () {
        fg = wijmo.Control.getControl("#fg");
        fg.formatItem.addHandler(function (s,e) {
            if (e.panel.cellType == wijmo.grid.CellType.Cell &&
e.panel.columns[e.col].binding == "金額") {
                var val = e.panel.grid.getCellData(e.row, e.col, false);
                if (val < 1000)
                    e.cell.style.color = 'red';
            }
        })
    });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
</script>

@(Html.C1().FlexGrid().Id("fg").Bind(Model).AutoGenerateColumns(false).Height("440px")
        .Columns(c => c.Add(cb => cb.Header("国名").Binding("Country"))
        .Add(cb => cb.Binding("Product").Header("製品名"))
        .Add(cb => cb.Binding("Amount").Header("金額"))))
```


Copyright © GrapeCity, Inc. All rights reserved.

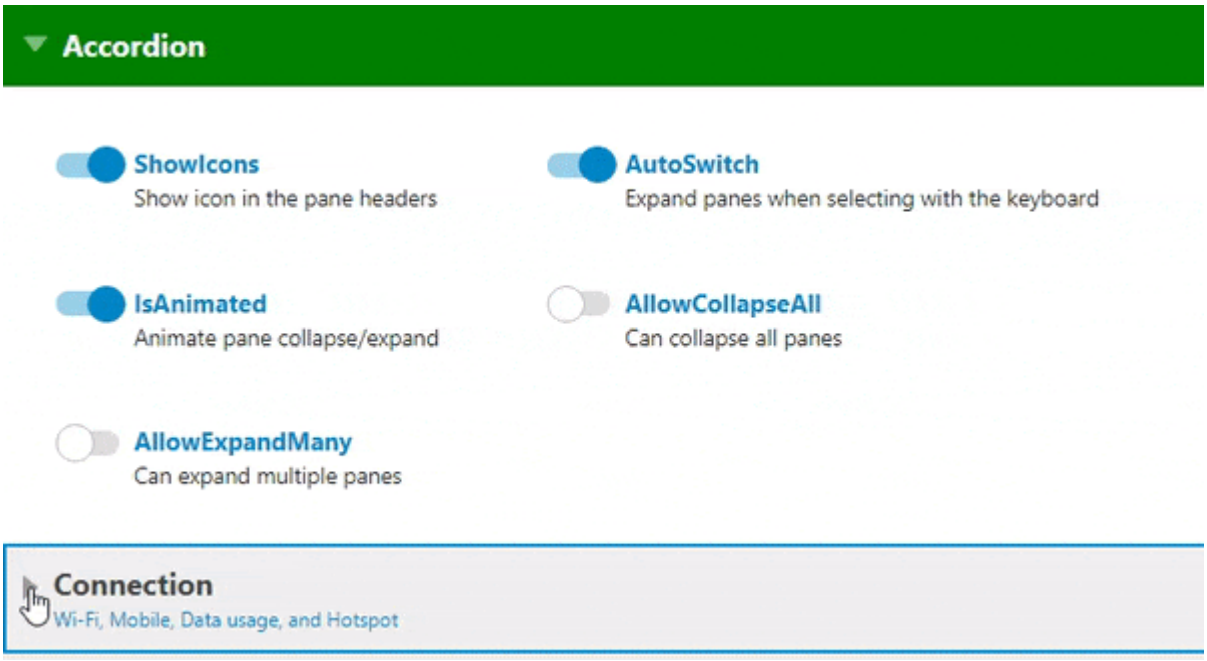
コントロールの使用

ComponentOne ASP.NET MVC Editionには、MVC Webアプリケーションを作成するために次のコントロールが提供します。

- CollectionView
- DashboardLayout
- Excel
- Financial Charts
- FlexChart
- FlexGrid
- FlexPie
- FlexRadar
- FlexReport
- FlexSheet
- FlexViewer
- Gauge
- Input
- MultiRow
- OLAP
- PDF
- Sunburst Chart
- TabPanel
- TreeMap
- TreeView

Accordion

Display a list of expandable items with ComponentOne Accordion for ASP.NET MVC. The Accordion control helps organize your UI and optimize the use of screen real estate. Accordion is used for navigation and it's optimized with responsive rendering and animation. The main advantage of using accordion is that it reduces page scrolling and allows the user to hide content that makes it appear less complicated. When one item is expanded, all others collapse. The Accordion control is optimized for intelligent rendering on devices of different screen sizes. Complete with smooth animations, the accordion provides a visually compelling interface for a menu or a navigation bar in your ASP.NET MVC application.



Copyright © GrapeCity, Inc. All rights reserved.

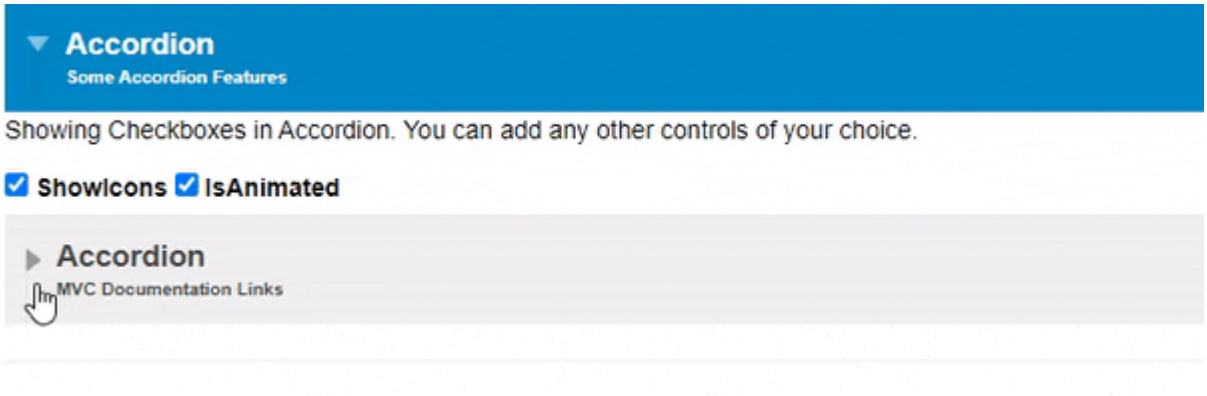
主な特長

Following are some of the main features of Accordion that you may find useful:

- **View Collapsible Data in Your MVC App**
  The Accordion control helps organize your UI and optimize the use of screen real estate by displaying a list of expandable items. When you expand one item, all the other items collapse.
- **Customize the Accordion Header**
  Customize Accordion pane's header with both text and controls. For example, display an icon and text in each header.
- **Add Objects of Any Data Type**
  Accordion can show any HTML content and you can add objects of any data type to its panes to create a visual representation of the items.
- **Collapse and Expand Animation**
  Accordion provides a visually compelling interface for a menu or a navigation bar in your application.

クイックスタート

The quick start guides you through the steps of adding the Accordion control to your MVC web application and customize the accordion. In the following example, we have used the **ShowIcons** and **IsAnimated** properties of the **Accordion** class to customize the Accordion control. The **ShowIcons** property shows collapsed/expanded icons in the pane headers and the **IsAnimated** property determines whether collapsing or expanding panes should be animated. You can also use other properties of the Accordion class, such as **AutoSwitch**, **AllowCollapseAll**, **AllowExpandMany**, etc., for further customization of the Accordion control.



To accomplish this, follow these steps:

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add Accordion

To add the Accordion control to your application, add **C1.Web.MVC** reference and follow these steps:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:

Copyright © GrapeCity, Inc. All rights reserved.

1. Select the **MVC 5 Controller - Empty** template.
2. Set name of the controller (for example: `AccordionController`).
3. Click **Add**.

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `AccordionController`.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller. Copy the following code and paste it inside **Index.cshtml**.

Index.cshtml copyCode

```
<style>
    /* styling to reduce font-size for Accordion's description */
    .wj-accordion .wj-header .tab-desc {
        font-size: 10px;
    }
    .wj-accordion .main-pane.wj-state-active {
        background: green;
    }
    .wj-accordion > .wj-header:focus {
        outline: 2px solid #0085c7;
        outline-offset: -2px;
    }
    .wj-accordion > .wj-header.wj-state-active {
        background: #0085c7;
        color: white;
    }
</style>

<!-- Accordion Host Element -->
<div id="accordion">
    <!-- Accordion Pane Wrapper-->
    <div>
        <!-- Accordion Pane Header-->
        <div>
            Accordion
            <!-- Short  description for Accordion Pane Header-->
            <div class="tab-desc">
                Some Accordion Features
            </div>
        </div>
        <!-- Accordion Pane Content-->
        <div>
            <p>Showing Checkboxes in Accordion. You can add any other controls
of your choice.</p>
            <input id="showIcons" type="checkbox" checked />
            <label for="showIcons">
                ShowIcons
            </label>
            <input id="isAnimated" type="checkbox" checked />
            <label for="isAnimated">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                IsAnimated
            </label>
        </div>
    </div>
    <div>
        <!-- Accordion Pane Header-->
        <div>
            Accordion
            <!-- Short  description for Accordion Pane Header-->
            <div class="tab-desc">
                MVC Documentation Links
            </div>
        </div>
        <!-- Accordion Pane Content-->
        <div>
            Click through to read the documentation for ComponentOne MVC
products.
            <ul>
                <li><a
href="https://www.grapecity.com/componentone/docs/mvc/online-mvc/overview.html"
target="_blank">MVC Documentation</a></li>
                <li><a
href="https://www.grapecity.com/componentone/docs/mvc/online-mvc-
core/overview.html" target="_blank">MVC Core Documentation</a></li>
            </ul>
        </div>

    </div>
</div>

@Html.C1().Accordion("#accordion")

@section Scripts{
    <script>
        c1.documentReady(function () {
            let acc = wijmo.Control.getControl("#accordion");
            link(acc, 'showIcons');
            link(acc, 'isAnimated');
        });
        function link(acc, id) {
            let cb = document.getElementById(id);
            acc[id] = cb.checked;
            cb.addEventListener('click', function(e) {
                acc[id] = cb.checked;
            });
        }
    </script>
}
```

Back to Top

Build and Run the Project

Copyright © GrapeCity, Inc. All rights reserved.

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/**BarCode/Index**) in the address bar of the browser to see the view.

Back to Top

Accordion の使用

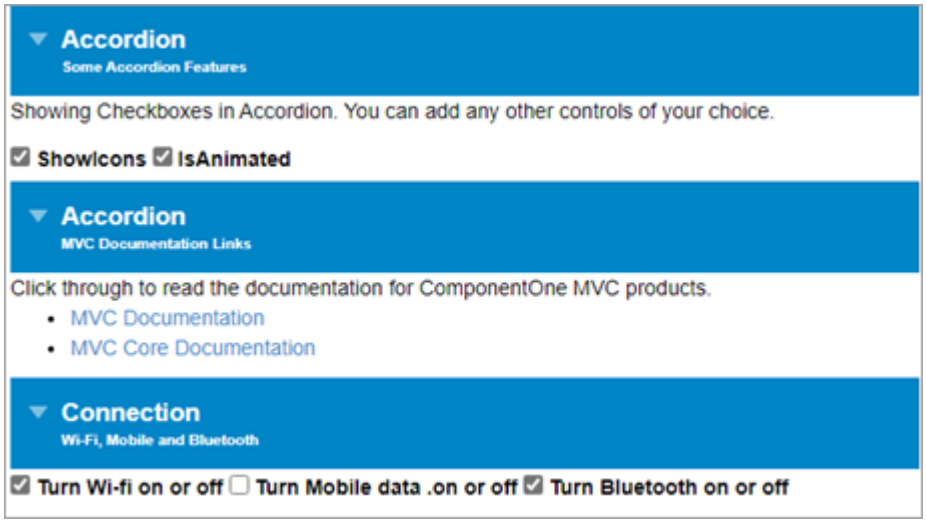
Learn about concepts that help you to understand how best to use the Accordion control.

ペインの展開/折り畳み
Learn how to expand and collapse panes in the Accordion control.

ペインの展開/折り畳み

The Accordion control allows you to organize content into collapsible panes so that you can choose to show or hide the content in the panes according to your requirements. By expanding the accordion panes, you can access their content and work with them. By collapsing the accordion panes, you can temporarily deny access to their content, save space and avoid scrolling. Let us explore how you can expand or collapse accordion pane(s) programmatically.

By default, the Accordion control expands only one pane at a time and collapses the other panes. However, you can change this behavior of the accordion panes by setting **AllowExpandMany** property of the Accordion class to **True**.



The following code demonstrates how you can allow a user to expand multiple panes at a time in the Accordion control programmatically.

Index.cshtml

```
<style>
        /* Accordionの説明のフォントサイズを小さくします。*/
        .wj-accordion .wj-header .tab-desc {
            font-size: 10px;
        }

        .wj-accordion .wj-header.wj-state-active {
            background: green;
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .wj-accordion > .wj-header:focus {
            outline: 2px solid #0085c7;
            outline-offset: -2px;
        }

        .wj-accordion > .wj-header.wj-state-active {
            background: #0085c7;
            color: white;
        }
</style>

<!-- Accordionのホスト要素 -->
<div id="accordion">
    <!-- Accordionのペインラッパー-->
    <div>
        <div>
            Accordion
            <div class="tab-desc">
                Some Accordion Features
            </div>
        </div>
        <div>
            <p>Showing Checkboxes in Accordion. You can add any other controls of
your choice.</p>
            <input id="showIcons" type="checkbox" checked />
            <label for="showIcons">
                ShowIcons
            </label>
            <input id="isAnimated" type="checkbox" checked />
            <label for="isAnimated">
                IsAnimated
            </label>

        </div>
    </div>
    <div>
        <!-- Accordion のペインヘッダ-->
        <div>
            Accordion
            <!-- Short  description for Accordion Pane Header-->
            <div class="tab-desc">
                MVC Documentation Links
            </div>
        </div>
        <!-- Accordion のペインコンテンツ-->
        <div>
            Click through to read the documentation for ComponentOne MVC products.
            <ul>
                <li><a href="https://www.grapecity.com/componentone/docs/mvc/online-
mvc/overview.html" target="_blank">MVC Documentation</a></li>
                <li><a href="https://www.grapecity.com/componentone/docs/mvc/online-
mvc-core/overview.html" target="_blank">MVC Core Documentation</a></li>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            </ul>
        </div>

    </div>
    <div>
        <!-- ペイン -->
        <div>
            <!-- ヘッダ -->
            Connection
            <div class="tab-desc">
                Wi-Fi, Mobile and Bluetooth
            </div>
        </div>
        <div>
            <!-- コンテンツ -->
                <input id="wifi" type="checkbox" checked />
                <label for="wifi">
                    <span>Turn Wi-fi on or off</span>
                </label>


                <input id="mobiledata" type="checkbox" />
                <label for="mobiledata">
                    <span>Turn Mobile data .on or off</span>
                </label>

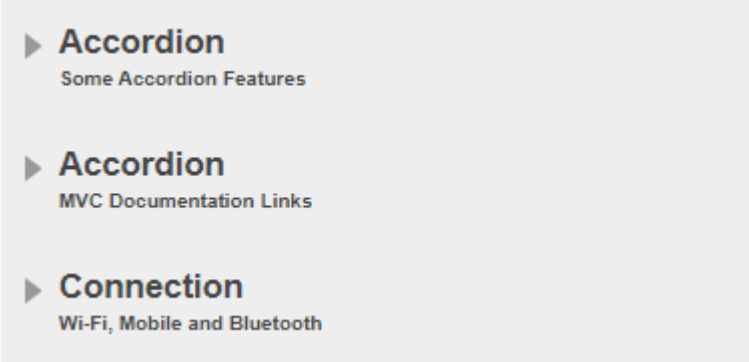

                <input id="bluetooth" type="checkbox" checked />
                <label for="bluetooth">
                    <span>Turn Bluetooth on or off</span>
                </label>
        </div>
    </div>
</div>

@Html.C1().Accordion("#accordion").AllowExpandMany(true)

    <script>
        c1.documentReady(function () {
            let acc = wijmo.Control.getControl("#accordion");
            link(acc, 'showIcons');
            link(acc, 'isAnimated');
        });
        function link(acc, id) {
            let cb = document.getElementById(id);
            acc[id] = cb.checked;
            cb.addEventListener('click', function(e) {
                acc[id] = cb.checked;
            });
        }
    </script>
```


Copyright © GrapeCity, Inc. All rights reserved.

As the default behavior of the Accordion control is to keep at least one pane open at all times, it prohibits you to fully collapse the Accordion. However, you can set **AllowCollapseAll** property of the Accordion class to **True** to allow a user to collapse all the accordion panes at once. This is useful in cases where you want to conserve screen real estate so that the user is free of unnecessary distractions.



The following code demonstrates how you can allow a user to collapse all the panes in the Accordion control programmatically.

Index.cshtml

```
<style>
        /* Accordionの説明のフォントサイズを小さくします。 */
        .wj-accordion .wj-header .tab-desc {
            font-size: 10px;
        }

        .wj-accordion .wj-header.wj-state-active {
            background: green;
        }

        .wj-accordion > .wj-header:focus {
            outline: 2px solid #0085c7;
            outline-offset: -2px;
        }

        .wj-accordion > .wj-header.wj-state-active {
            background: #0085c7;
            color: white;
        }
    </style>

<!-- Accordionのホスト要素 -->
<div id="accordion">
    <!-- Accordionペインのラッパー-->
    <div>
        <div>
            Accordion
            <div class="tab-desc">
                Some Accordion Features
            </div>
        </div>
        <div>
            <p>Showing Checkboxes in Accordion. You can add any other controls of
your choice.</p>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <input id="showIcons" type="checkbox" checked />
            <label for="showIcons">
                ShowIcons
            </label>
            <input id="isAnimated" type="checkbox" checked />
            <label for="isAnimated">
                IsAnimated
            </label>

        </div>
    </div>
    <div>
        <!-- Accordion ペインのヘッダ-->
        <div>
            Accordion
            <!-- Short  description for Accordion Pane Header-->
            <div class="tab-desc">
                MVC Documentation Links
            </div>
        </div>
        <!-- Accordion ペインのコンテンツ-->
        <div>
            Click through to read the documentation for ComponentOne MVC products.
            <ul>
                <li><a href="https://www.grapecity.com/componentone/docs/mvc/online-
mvc/overview.html" target="_blank">MVC Documentation</a></li>
                <li><a href="https://www.grapecity.com/componentone/docs/mvc/online-
mvc-core/overview.html" target="_blank">MVC Core Documentation</a></li>
            </ul>
        </div>

    </div>
    <div>
        <!-- ペイン -->
        <div>
            <!-- ヘッダ -->
            Connection
            <div class="tab-desc">
                Wi-Fi, Mobile and Bluetooth
            </div>
        </div>
        <div>
            <!-- コンテンツ -->
                <input id="wifi" type="checkbox" checked />
                <label for="wifi">
                    <span>Turn Wi-fi on or off</span>
                </label>


                <input id="mobiledata" type="checkbox" />
                <label for="mobiledata">
                    <span>Turn Mobile data .on or off</span>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                </label>

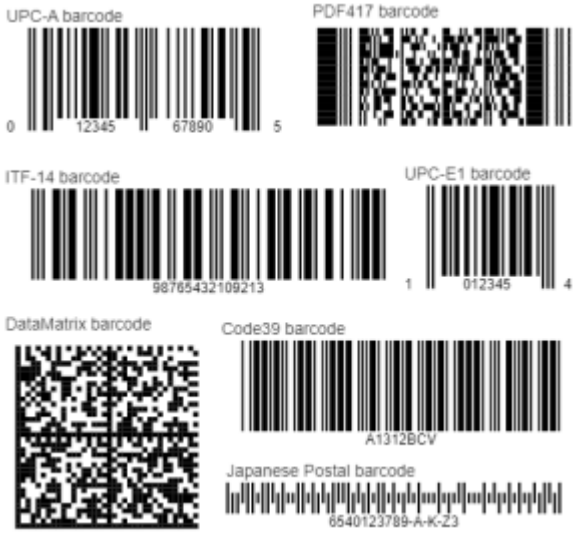

                <input id="bluetooth" type="checkbox" checked />
                <label for="bluetooth">
                    <span>Turn Bluetooth on or off</span>
                </label>
        </div>
    </div>
</div>

@Html.C1().Accordion("#accordion").AllowCollapseAll(true)

    <script>
        c1.documentReady(function () {
            let acc = wijmo.Control.getControl("#accordion");
            link(acc, 'showIcons');
            link(acc, 'isAnimated');
        });
        function link(acc, id) {
            let cb = document.getElementById(id);
            acc[id] = cb.checked;
            cb.addEventListener('click', function(e) {
                acc[id] = cb.checked;
            });
        }
    </script>
```

BarCode

Generate or retrieve barcodes on the server and render them entirely on the client using **Barcode for ASP.NET MVC**. Barcodes represent data in a visual, but machine-readable way. They are used around the world to share and track information. You can embed barcodes in your ASP.NET MVC web apps and reports to transfer information between apps or devices. BarCode for ASP.NET MVC supports 26 different types of encodings from QR codes to UPC barcodes which can be easily generated using a single component without having to use multiple libraries and different APIs.



Copyright © GrapeCity, Inc. All rights reserved.

主な特長

Barcode offers various features as described below:

- **26 Barcode Encodings in One Component**
  With support for 26 encoding types in a single component, you can easily generate different barcodes without having to use multiple libraries and different APIs.
- **Generate Barcodes on the Fly with a Web API**
  Generate scannable barcodes from text that can be obtained from a web service and saved as an image or stream.
- **Generate Barcodes for Manufacturing, Warehouse and Distribution Industries**
  Use Interleaved 2 of 5 codes among others for applications targeting manufacturing, warehouse and distribution.
- **Use 2D QRCodes for Marketing and Advertising**
  Use the QRCode encoding to display 2D barcodes in your MVC web apps and reports, or use it to serve up barcodes from a web service.
- **Maintain Data Integrity with CheckSums**
  Barcode automatically adds necessary control symbols and checksums to the value being encoded to guarantee a good read on your barcodes depending on the encoding being used.

バーコードエンコーディング

Barcode provides a list of 26 available barcodes suitable for your application. The following table lists all the 26 barcode types supported by the MVC BarCode control.

Encoding Type	Image	Description
Codabar		Codabar uses A B C D + - : . / $ and numbers.
Code39		Code 39 uses numbers, % * $ /. , - +, and upper case.
Code49		Code 49 is a 2D high-density stacked barcode containing two to eight rows of eight characters each. Each row has a start code and a stop code. Encodes the complete ASCII character set.
Code93		Code 93 uses uppercase, % $ * / , + -, and numbers.
Code128		linear barcode symbology defined in ISO/IEC 15417:2007.
DataMatrix ECC 200		Data Matrix is a high density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern. ECC 200 is a new version of Data Matrix.
DataMatrix ECC-000 to ECC-140		Data Matrix is a high density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern. ECC 000–140 code uses a convolution-based error correction
EAN-8		EAN-8 uses only numbers (7 numbers and a check digit).
EAN-13		EAN-13 uses only numbers (12 numbers and a check digit). It takes only 12 numbers as a string to calculate a check digit (CheckSum) and add it to the thirteenth

Copyright © GrapeCity, Inc. All rights reserved.

		position. The check digit is an additional digit used to verify that a bar code has been scanned correctly. The check digit is added automatically when the CheckSum property is set to True.
GS1-128		GS1-128 is an application standard of GS1 implementation and the subset of Code 128 barcode.
GS1DataBarExpanded		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is used for retail point-of-sale.
GS1DataBarExpandedStacked		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is used for retail point-of-sale.
GS1DataBarLimited		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is not used for retail point-of-sale.
GS1DataBarOmnidirectional		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is used for retail point-of-sale.
GS1DataBarStacked		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is not used for retail point-of-sale.
GS1DataBarStackedOmnidirectional		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is used for retail point-of-sale.
GS1DataBarTruncated		DataBar barcodes are usually used to label fresh foods and can hold information like item's batch number or expiry date. This GS1 Databar symbol is not used for retail point-of-sale.
Interleaved2of5		Interleaved 2 of 5 (ITF) barcode is a continuous two-width barcode symbology encoding digits.
ITF14		ITF14 barcode is the GS1 implementation of an Interleaved 2 of 5 bar code to encode a Global Trade Item Number. It is continuous, self-checking, bidirectionally decodable and it will always encode 14 digits. ITF14 is used on packaging levels of a product in general.
JapanesePostal		This is the barcode used by the Japanese Postal system. Encodes alpha and numeric characters consisting of 20 digits including a 7-digit postal code number, optionally followed by block and house number information. The data to be encoded can include hyphens.

Copyright © GrapeCity, Inc. All rights reserved.

MicroPDF417		MicroPDF417 is two-dimensional (2D), multi-row symbology, derived from PDF417. Micro-PDF417 is designed for applications that need to encode data in a two-dimensional (2D) symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits) with the minimal symbol size. MicroPDF417 allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs).
Pdf417		Pdf417 is a popular high-density 2-dimensional symbology that encodes up to 1108 bytes of information. This barcode consists of a stacked set of smaller barcodes. Encodes the full ASCII character set. It has ten error correction levels and three data compaction modes: Text, Byte, and Numeric. This symbology can encode up to 1,850 alphanumeric characters or 2,710 numeric characters.
QRCode		QRCode is a 2D symbology that is capable of handling numeric, alphanumeric and byte data as well as Japanese kanji and kana characters. This symbology can encode up to 7,366 characters.
UPC-A		UPC-A uses only numbers (11 numbers and a check digit).
UPC-E0		UPC-E0 uses only numbers. Used for zero-compression UPC symbols. For the Caption property, you may enter either a six-digit UPC-E code or a complete 11-digit (includes code type, which must bezero) UPC-A code. If an 11-digit code is entered, the Barcode control will convert it to a six-digit UPC-E code, if possible. If it is not possible to convert from the 11-digit code to the six-digit code, nothing is displayed.
UPC-E1		UPC-E1 uses only numbers. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters.

クイックスタート

The quick start guides you through the steps of adding different barcodes to your MVC web application and display them as the output. In the following example, we have added three barcodes, namely Codabar, Code39 and Gs1-128, and these barcodes can be displayed using the **Codabar**, **Code39** and **Gs1_128** barcode classes respectively. Similarly, you can use any of the 26 barcodes supported by the Barcode control. For the complete list of supported barcodes, see Barcode Encodings.

Copyright © GrapeCity, Inc. All rights reserved.



To accomplish this, follow these steps:

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add Barcodes

To add the Barcode control to your application, add **C1.Web.MVC** reference and follow these steps:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `BarcodeController`).
    3. Click **Add**.
4. Replace the method **Index()** with the following method.

Sale.cs copyCode

```
public ActionResult Index()
{
    return View();
}
```

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `BarcodeController`.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**

Copyright © GrapeCity, Inc. All rights reserved.

5. Click **Add** to add a view for the controller. Copy the following code and paste it inside **Index.cshtml**.

Index.cshtml copyCode

```
@model object[]

<table class="table">
    <thead>
        <tr>
            <th>BarCodes</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>
                <div>
                    <div style="font-weight:bold">Codabar</div>

<div>@Html.C1().Codabar().Value("A15126893B").AutoWidthZoom(2)</div>
                </div>
                <br />
                <div>
                    <div style="font-weight:bold">Code39</div>
                    <div>@Html.C1().Code39().Value("A1312BCV").AutoWidthZoom(2)
</div>
                </div>
                <br />
                <div>
                    <div style="font-weight:bold">Gs1_128</div>

<div>@Html.C1().Gs1_128().Value("GS1128Demo").AutoWidthZoom(2)</div>
                </div>
            </td>
        </tr>

    </tbody>
</table>
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/Barcode/Index) in the address bar of the browser to see the view.

Back to Top

CollectionView

ComponentOneのMVC **CollectionView**は、**ICollectionView**インタフェースを実装して**FlexGrid**などのデータ連結コントロールにデータを表示するサービスです。CollectionViewは、実際のデータをラップするレイヤを提供し、そのデータに影響を

Copyright © GrapeCity, Inc. All rights reserved.

与えることなくソート、フィルタ処理、グループ化などのさまざまな操作を行うという、たいへん強力な概念です。簡単に言えば、データベースの**DefaultView**と考えることができます。

CollectionViewクラスには、データに加えられた変更を記録し、現在のレコードを管理する機能があります。項目の編集、追加、削除、ページングなどを管理できます。これは**INotifyCollectionChanged**インタフェースを実装し、項目が追加または削除されたり、コレクションがソート、フィルタ処理、またはグループ化されるなどの動的な変更をリスナーに通知します。

CollectionViewクラスは、次のインタフェースを実装します。

- **ICollectionView**:現在のレコードの管理、カスタムソート、フィルタ処理、およびグループ化を提供します。
- **IEditableCollectionView**:項目を編集、追加、および削除するためのメソッドを提供します。
- **IPagedCollectionView**:ページングを提供します。

CollectionViewは、**サーバー側**操作と**クライアント側**操作をサポートします。

サーバー側の操作

サーバー側の **CollectionViewHelper**サービスにより、コレクションは読み込み、編集、フィルタ処理、グループ化、およびソートが可能になります。これは.Net ICollectionViewに似ています。サーバー上で、CollectionViewはデータ連結コントロールによるソート、ページング、フィルタ処理要求を内部的に処理します。このセクションでは、**Create**、**Read**、**Update**、**Delete**、およびCRUD操作の**BatchEdit**という各種アクションの使用方法を説明します。

- `Read()`:コレクションからデータを取得します。
- `Edit()`:データ連結コントロールからのデータ更新要求を処理します。詳細については、「編集」を参照してください。
- `BatchEdit()`:複数の項目を一度に更新できます。詳細については、「一括更新」を参照してください。

クライアント側の操作

CollectionViewには、強力なクライアントAPIが含まれています。**CollectionViewHelper**は、データ連結コントロール(FlexGrid、FlexChart、その他のInputコントロール)のデータに対するサーバー側の操作(**ソート**、**フィルタ処理**、**ページング**)を内部的に行います。ただし、これらの操作をクライアント側で明示的に行うことも可能です。クライアント側の操作には、**現在のレコードの管理**、**ソート**、**フィルタ処理**、**グループ化**、**変更の追跡**などがあります。

> CollectionViewのクライアント側APIの詳細については、「Wijmo CollectionView APIドキュメント」を参照してください。

主な特長

CollectionView offers many advanced features beyond simple data management. These features are listed below:

- **Powerful Library**
  CollectionView is a powerful and flexible view that is designed to perform common data transformations by implementing the ICollectionView, IEditableCollectionView and IPagedCollectionView interface.

- **Optimized Performance**
  CollectionView is highly optimized in performing operations such as, sorting, filtering, and grouping on large data sets efficiently.

- **Manipulate Data with Sorting, Filtering, Grouping and Record Management**
  CollectionView class and ICollectionView interface provides support for data manipulation in the form of sorting, filtering, record management, and grouping collections.

- **Easy to use with Any Control**
  CollectionView is a compatible data source that can be used with any data-aware control, such as DataGridView, DataFilter and FlexGrid.

Copyright © GrapeCity, Inc. All rights reserved.

クイックスタート

このセクションでは、MVC WebアプリケーションでCollectionViewをFlexGridコントロールと一緒に使用する方法を説明します。

このトピックは4つの手順で構成されます。

- **手順1：MVC アプリケーションの作成**
- **手順2：アプリケーションのデータソースを設定**
- **手順3：FlexGridコントロールの追加**
- **手順4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexGridを示しています。

	ID	カテゴリ名	詳細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish
*			

手順1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2：アプリケーションのデータソースを設定

例では、**C1NWind**データベースを使用します。**C1NWind.mdf**ファイルは、システムの*C:\Users\<username>\Documents\ComponentOne Samples\ASP.NET MVC\MVC\MvcExplorer\App_Data* に用意されています。

1. **C1NWind.mdf**ファイルをソリューションエクスプローラーの`AppData`フォルダに追加します。
2. ソリューションエクスプローラーで右クリックして[モデル]→[新しい項目の追加]→[データ]を選択し、**ADO.NET Entity Data Model**を選択します。
3. モデルに**C1NWind**と名前を付け、**[追加]**をクリックします。
4. エンティティデータモデルウィザードで、**データベースからEF Designer**を選択し、**[次へ]**をクリックします。
5. **C1NWind.mdf**データベースがデータ接続ドロップダウンに追加されます。
6. **[次へ]**をクリックし、Entity Frameworkのバージョンを選択し、**[次へ]**をクリックします。
7. **[データベースオブジェクトと設定を選択]**で、テーブルを選択し、**[完了]**をクリックします。

プロジェクトの**[モデル]**フォルダの下に**C1NWind.edmx**が追加されていれば、アプリケーションに対するデータソースの設定は成功です。

先頭に戻る

手順3：FlexGridコントロールの追加

FlexGridコントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

Copyright © GrapeCity, Inc. All rights reserved.

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`QuickStartController`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C# copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using C1.Web.Mvc.Grid;
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using System.Data.Entity.Validation;
using System.Data.Entity;
using System.Data;
using MVC_DataBinding.Models;
```

5. メソッドIndex()を次のメソッドに置き換えます。

IndexController.cs

C# copyCode

```
//FlexGrid に対してデータソースを定義します
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()
{
    return View();
}
//JSONのインスタンスを作成します
public ActionResult GridReadCategory([C1JsonRequest]
CollectionViewRequest<Categories> requestData)
{
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Categories));
}
```

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されました。
6. ソリューションエクスプローラーで、`Index.cshtml`をダブルクリックして開きます。
7. **Views\Index.cshtml**ファイルのデフォルトコードを次のコードに置き換えて、**FlexGrid**コントロールを初期化します。

この例では、コントローラーの**GridReadCategory**アクションはFlexGridの**ItemSource**の**Bind**プロパティに割り当てられ、データを挿入します。

Razor (Index.cshtml) copyCode

Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().FlexGrid().Id("fGRCView").AutoGenerateColumns(false).AllowAddNew(true)
    .AllowSorting(true).CssClass("grid")
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID"))
        .Add(c => c.Binding("CategoryName"))
        .Add(c => c.Binding("Description").Width("500")))
    .Bind(
        ib => ib.Bind(Url.Action("GridReadCategory"))
    )
)
```

手順4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

先頭に戻る

CollectionViewの使用

This section comprises all the features and auxiliary functionality offered by CollectionView.

サーバー側処理
: Learn about the server-side operations that can be performed using CollectionView.

クライアント側処理
: Learn about the client-side operations that can be performed using CollectionView.

DateTime処理
: Learn how to apply an explicit transformation on the dates in both client and server to keep time in the UTC format.

サーバー側処理

On the server, CollectionView internally handles sorting, paging, filtering requests by data bound controls. In addition, it provides the server-side **CollectionViewHelper** class, which is a service that enables collections to have reading, editing, filtering, grouping and sorting ability, similar to .Net ICollectionView. Furthermore, it allows you to perform CRUD operations and BatchEdit actions as well.

このセクションは、**ASP.NET MVC** コントロールの使用を開始するにあたって役立つ情報の提供を目的としています。

一括更新
: Learn how to define BatchEdit request in controller to update multiple items at a time.

作成
: Learn how to perform create operation in a databound control.

削除
: Learn how to perform delete operation in a databound control.

サーバー読み取りの無効化
: Learn how to disable server reading for Paging operation in a databound control.

編集
: Learn how to enable editing and updating records in a databound control.

複数コントロールのデータ連結
: Learn how to bind multiple databound control using CollectionView.

読み取り
: Learn how to use a Read request to retrieve data from the collection.

Copyright © GrapeCity, Inc. All rights reserved.

一括更新

CollectionViewHelperには、ユーザーが複数の変更をデータベースにコミットできる**BatchEdit**要求があります。この例は、コントローラーでBatchEdit要求を定義し、**CollectionViewHelper**のBatchEdit要求を使用してCollectionViewでデータベースを更新する方法を示します。

次の図は、CollectionViewHelperのBatchEditing要求を設定した後のFlexGridを示しています。ユーザーは、**[更新]**ボタンをクリックすることにより、変更をサーバーにコミットできます。

> FlexGridで**BatchEdit**を実行中に、グリッドでソート、フィルタ処理、またはページングを行うと、サーバーに対して更新要求が行われ、それによって自動的にソースデータベースのデータが更新されます。そのため、**ItemSource**の**DisableServerRead**プロパティを必ず**true**に設定してください。
>
> このプロパティの値をtrueに設定すると、一括更新は、**[更新]**ボタンのCollectionViewのCommitメソッドを明示的に呼び出すことによってのみ行うことができます。次に例を示します。

更新

	ID	カテゴリ名	詳細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish
*			

次のコード例は、**BatchEdit**プロパティと**CollectionViewEditRequest**を使用して、FlexGridで同時に複数のレコードを更新できるようにする方法を示します。

コードの場合

BatchUpdateController.cs

C# copyCode

```
private C1NWindEntities db = new C1NWindEntities();


public ActionResult Index()
{
    return View(db);
}

public ActionResult GridReadCategory([C1JsonRequest]
    CollectionViewRequest<Categories> requestData)
{
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Categories));
}


public ActionResult GridBatchEdit([C1JsonRequest]
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    CollectionViewBatchEditRequest<Categories> requestData)
{
    return this.C1Json(CollectionViewHelper.BatchEdit(requestData, batchData =>
    {
        var itemresults = new List<CollectionViewItemResult<Categories>>();
        string error = string.Empty;
        bool success = true;
        try
        {
            if (batchData.ItemsCreated != null)
            {
                batchData.ItemsCreated.ToList().ForEach(st =>
                {
                    db.Categories.Add(st);
                    itemresults.Add(new CollectionViewItemResult<Categories>
                    {
                        Error = "",
                        Success = ModelState.IsValid,
                        Data = st
                    });
                });
            }
            if (batchData.ItemsDeleted != null)
            {
                batchData.ItemsDeleted.ToList().ForEach(category =>
                {
                    var fCategory = db.Categories.Find(category.CategoryID);
                    db.Categories.Remove(fCategory);
                    itemresults.Add(new CollectionViewItemResult<Categories>
                    {
                        Error = "",
                        Success = ModelState.IsValid,
                        Data = category
                    });
                });
            }
            if (batchData.ItemsUpdated != null)
            {
                batchData.ItemsUpdated.ToList().ForEach(category =>
                {
                    db.Entry(category).State = EntityState.Modified;
                    itemresults.Add(new CollectionViewItemResult<Categories>
                    {
                        Error = "",
                        Success = ModelState.IsValid,
                        Data = category
                    });
                });
            }
            db.SaveChanges();
        }
        catch (DbEntityValidationException e)
        {
            error = string.Join(",", e.EntityValidationErrors
            .SelectMany(i => i.ValidationErrors).Select(i => i.ErrorMessage));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            success = false;
        }
        catch (Exception e)
        {
            error = e.Message;
            success = false;
        }

        return new CollectionViewResponse<Categories>
        {
            Error = error,
            Success = success,
            OperatedItemResults = itemresults
        };
    }, () => db.Categories.ToList()));
}
```

この例では、**BatchEdit**要求はFlexGridの**ItemsSource**のBatchEditプロパティに割り当てられます。

BatchUpdate.cshtml

Razor copyCode

```
<input type="button" value="更新" class="btn" onclick="batchUpdate()" />
@(Html.C1().FlexGrid().Id("fGBECView").AutoGenerateColumns(false)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID"))
        .Add(c => c.Binding("CategoryName"))
        .Add(c => c.Binding("Description").Width("500")))
    .Bind(ib =>
ib.Bind(Model.Categories).DisableServerRead(true).BatchEdit(Url.Action("GridBatchEdit")))
    .AllowAddNew(true)
    .AllowDelete(true)
    .CssClass("grid")
)
```

JS copyCode

```
<script>
    //一括更新
    function batchUpdate() {
        var batchEditGrid = wijmo.Control.getControl('#fGBECView'),
        cv = batchEditGrid.collectionView;
        cv.commit();
    };
</script>
```

作成

CollectionViewは、FlexGridなどのデータ連結コントロール内のフィールドの作成、読み込み、更新、削除、編集など、サーバー側のさまざまな操作をサポートします。

この例は、**CollectionViewHelper**の**Edit**要求の使用方法を示します。この要求を使用すると、FlexGridなどのデータ連結コントロールでレコードをソースデータベースに追加できます。ここで、コントローラーの**GridCreateCategory**アクションは、FlexGridの**ItemSource**のCreateプロパティに割り当てられます。

Copyright © GrapeCity, Inc. All rights reserved.

次の図は、**AllowAddNew**プロパティがtrueに設定され、**CollectionViewEditRequest**がコントローラーで使用された後のFlexGridを示しています。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

	ID	カテゴリ名	詳細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish
*			

次のコード例は、FlexGridの**ItemSource**の**Create**プロパティと**CollectionViewEditRequest**を使用してFlexGrid内でレコードを作成できるようにする方法を示します。

コードの場合

CreateController.cs

C# copyCode

```
  //FlexGrid に対してデータソースを定義します
  private C1NWindEntities db = new C1NWindEntities();
  public ActionResult Index()
  {
      return View(db);
  }
  //JSONのインスタンスを作成します
  public ActionResult GridReadCategory([C1JsonRequest]
      CollectionViewRequest<Categories> requestData)
  {
return this.C1Json(CollectionViewHelper.Read(requestData, db.Categories));
  }
  public ActionResult GridCreateCategory([C1JsonRequest]
      CollectionViewEditRequest<Categories> requestData)
  {
      var category = requestData.OperatingItems.First();
      if (category.CategoryName == null)
      {
          category.CategoryName = "";
      }
      return Create(requestData, db.Categories);
  }

  private ActionResult Create<T>(CollectionViewEditRequest<T>
      requestData, DbSet<T> data) where T : class
```


Copyright © GrapeCity, Inc. All rights reserved.

```
  {
   return this.C1Json(CollectionViewHelper.Edit<T>(requestData, item =>
      {
        string error = string.Empty;
        bool success = true;
         try
         {
             data.Add(item);
             db.SaveChanges();
       }
  catch (DbEntityValidationException e)
    {
   error = string.Join(",", e.EntityValidationErrors.Select(result =>
           {
             return string.Join(",", result.ValidationErrors
             .Select(err => err.ErrorMessage));
              }));
              success = false;
          }
          catch (Exception e)
          {
              error = e.Message;
              success = false;
          }
          return new CollectionViewItemResult<T>
          {
              Error = error,
              Success = success && ModelState.IsValid,
              Data = item
          };
      }, () => data.ToList<T>()));
  }
```

この例では、**GridCreateCategory**アクションはFlexGridの**ItemSource**のCreateプロパティに割り当てられています。

Create.cshtml

Razor copyCode

```
@(Html.C1().FlexGrid().Id("fGRCView")
.AutoGenerateColumns(false).AllowAddNew(true)
  .AllowSorting(true).CssClass("grid")
  .Columns(columns => columns
    .Add(c => c.Binding("CategoryID").Header("ID"))
    .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
    .Add(c => c.Binding("Description").Header("詳細").Width("500")))
   .Bind(ib=>ib.Bind(Url.Action("GridReadCategory"))
   .Create(Url.Action("GridCreateCategory"))
        )
)
```

削除

Copyright © GrapeCity, Inc. All rights reserved.

この例では、コントローラー内のアクションを定義して**CollectionViewHelper**のEdit要求を処理することにより、データベースから列を削除する方法を示します。

次の図は、**CollectionViewEditRequest**がコントローラーで使用された後のFlexGridを示しています。

この例では**C1NWind**データソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。行を**削除**するには、行ヘッダーをクリックして行を選択し、キーボードの**[Delete]**ボタンをクリックします。選択された行が削除され、それに合わせてソースデータベースが更新されます。

	ID	カテゴリ名 ▲	詳細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish

次のコードは、**CollectionViewEditRequest**を処理してFlexGridでレコードを削除できるようにする方法を示したものです。

コードの場合

DeleteController.cs

C# copyCode

```
//FlexGrid に対してデータソースを定義します
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()
{
    return View(db);
}
//JSONのインスタンスを作成します
public ActionResult GridReadCategory([C1JsonRequest]
    CollectionViewRequest<Categories> requestData)
{
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Categories));
}

public ActionResult GridDeleteCategory([C1JsonRequest]
    CollectionViewEditRequest<Categories> requestData)
{
    return Delete(requestData, db.Categories, item => item.CategoryID);
}
private ActionResult Delete<T>(CollectionViewEditRequest<T>
    requestData, DbSet<T> data, Func<T, object> getKey) where T : class
{
    return this.C1Json(CollectionViewHelper.Edit<T>(requestData, item =>
    {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        string error = string.Empty;
        bool success = true;
        try
        {
            var resultItem = data.Find(getKey(item));
            data.Remove(resultItem);
            db.SaveChanges();
        }
        catch (DbEntityValidationException e)
        {
            error = string.Join(",", e.EntityValidationErrors.Select(result =>
            {
                return string.Join(",", result.ValidationErrors
                .Select(err => err.ErrorMessage));
            }));
            success = false;
        }
        catch (Exception e)
        {
            error = e.Message;
            success = false;
        }
        return new CollectionViewItemResult<T>
        {
            Error = error,
            Success = success && ModelState.IsValid,
            Data = item
        };
    }, () => data.ToList<T>()));
}
```

この例では、**Delete**アクションはFlexGridの**ItemSource**のDeleteプロパティに割り当てられています。

Delete.cshtml

Razor copyCode

```
@(Html.C1().FlexGrid().Id("fGDelCView").AutoGenerateColumns(false).IsReadOnly(false)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID"))
        .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
        .Add(c => c.Binding("Description").Header("詳細").Width("*")))
    .AllowDelete(true)
    .Bind(
        ib => ib.Bind(Url.Action("GridReadCategory"))
        .Delete(Url.Action("GridDeleteCategory"))
    )
)
```

サーバー読み取りの無効化

CollectionViewを使用して、FlexGridなどのデータ連結コントロールにおけるサーバー側のソート、ページング、フィルタ処理、またはスクロールを無効にします。これは、FlexGridの**ItemSource**の**DisableServerRead**プロパティの値を**True**に設定する

Copyright © GrapeCity, Inc. All rights reserved.

ことによって行います。デフォルトでは、この値はfalseに設定されています。任意の操作に対してこのプロパティ値をtrueに設定すると、すべての現在の項目がクライアント側に転送され、サーバー側イベントは無効になります。ネットワーク呼び出しを実行せずにクライアント側の操作を直接実行できます。

この例は、FlexGridのPaging操作でのサーバー読み取りを無効化する方法を示します。

次の図は、DisableServerReadプロパティをtrue、ページサイズを**10**に設定した後のFlexGridを示しています。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。



ID	会社名	連絡先名	都市	国名	電話番号
ALFKI	Alfreds Futter...	Maria Anders	Berlin	Germany	030-0074321
ANATR	Ana Trujillo E...	Ana Trujillo	México D.F.	Mexico	(5) 555-4729
ANTON	Antonio More...	Antonio Moreno	México D.F.	Mexico	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	London	UK	(171) 555-7788
BERGS	Berglunds sn...	Christina Berg...	Luleå	Sweden	0921-12 34 65
BLAUS	Blauer See D...	Hanna Moos	Mannheim	Germany	0621-08460
BLONP	Blondel père ...	Frédérique Ci...	Strasbourg	France	88.60.15.31
BOLID	Bólido Comid...	Martín Sommer	Madrid	Spain	(91) 555 22 82
BONAP	Bon app'	Laurence Leb...	Marseille	France	91.24.45.40
BOTTM	Bottom-Dollar...	Elizabeth Linc...	Tsawassen	Canada	(604) 555-4729

次のコード例は、**DisableServerRead**プロパティを設定してサーバー側のPagingをFlexGridで無効にする方法を示します。

コードの場合

Pagingを有効にするには、`_Pager.cshtml`ページを**[ビュー]|[共有]**フォルダに追加し、その内容を次のものに置き換えます。

_Pager.cshtml

_Pager.cshtml

```
@model string
<div class="wj-control wj-content wj-pager">
    <div class="wj-input-group">
        <span class="wj-input-group-btn">
            <button class="wj-btn wj-btn-default demo-grid-firstpage" type="button">
                <span class="wj-glyph-left"></span>
                <span class="wj-glyph-left"></span>
            </button>
        </span>
        <span class="wj-input-group-btn">
            <button class="wj-btn wj-btn-default demo-grid-previouspage"
type="button">
                <span class="wj-glyph-left"></span>
            </button>
        </span>
        <input type="text" class="wj-form-control demo-grid-currentpage" disabled="">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        <span class="wj-input-group-btn">
            <button class="wj-btn wj-btn-default demo-grid-nextpage" type="button">
                <span class="wj-glyph-right"></span>
            </button>
        </span>
        <span class="wj-input-group-btn">
            <button class="wj-btn wj-btn-default demo-grid-lastpage" type="button">
                <span class="wj-glyph-right"></span>
                <span class="wj-glyph-right"></span>
            </button>
        </span>
    </div>
</div>
<script>
    var pagingGrid = wijmo.Control.getControl('#@(Model)'),
        cv,
        pagerButtons = Array.prototype.slice.call
           document.querySelectorAll('.wj-pager button'));
    if (pagingGrid) {
        cv = pagingGrid.collectionView;
    }
    if (!cv) {
        throw "the collectionview is null";
    }
    // ユーザーがボタンをクリックするとページャを更新します
    pagerButtons.forEach(function (el) {
        el.addEventListener('click', function () {
            var className = el.className;
            if (className.indexOf("firstpage") > -1) {
                cv.moveToFirstPage();
            } else if (className.indexOf("lastpage") > -1) {
                cv.moveToLastPage();
            } else if (className.indexOf("previouspage") > -1) {
                cv.moveToPreviousPage();
            } else if (className.indexOf("nextpage") > -1) {
                cv.moveToNextPage();
            }
        });
    });
    cv.collectionChanged.addHandler(function () {
        updatePager();
    });
    updatePager();
    // ボタンを無効/有効にし、ページャの表示テキストを更新します
    function updatePager() {
        // IDでボタンを取得します
        var display = document.querySelector(".wj-pager .demo-grid-currentpage"),
            next = document.querySelector(".wj-pager .demo-grid-nextpage"),
            previous = document.querySelector(".wj-pager .demo-grid-previouspage"),
            first = document.querySelector(".wj-pager .demo-grid-firstpage"),
            last = document.querySelector(".wj-pager .demo-grid-lastpage"),
            enableBackwards,
            enableForwards;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        // ページャテキストを更新します
        display.value = (cv.pageIndex + 1) + ' / ' + (cv.pageCount);
        // どのページャボタンを有効/無効にするかを決定します
        enableBackwards = cv.pageIndex <= 0;
        enableForwards = cv.pageIndex >= cv.pageCount - 1;
        // ページャボタンを有効/無効にします
        previous.disabled = enableBackwards;
        first.disabled = enableBackwards;
        next.disabled = enableForwards;
        last.disabled = enableForwards;
    }
</script>
```

DisableServerReadController.cs

C# copyCode

```
     //FlexGrid に対してデータソースを定義します
     private C1NWindEntities db = new C1NWindEntities();
     public ActionResult Index()
     {
         return View(db);
     }
     //JSONのインスタンスを作成します
     public ActionResult GridReadCategory([C1JsonRequest]
         CollectionViewRequest<Customers> requestData)
 {
return this.C1Json(CollectionViewHelper.Read(requestData, db.Customers));
 }
```

DisableServerRead.cshtml

Razor copyCode

```
@(Html.C1().FlexGrid().Id("fGDisableServerView").IsReadOnly(true)
.AllowSorting(true).AutoGenerateColumns(false)
    .Bind(b => b.DisableServerRead(true).PageSize(10)
.Bind(Url.Action("GridReadCategory"))).CssStyle("height", "100%")
    .Columns(columns => columns
        .Add(c => c.Binding("CustomerID").Header("ID"))
        .Add(c => c.Binding("CompanyName").Header("会社名"))
        .Add(c => c.Binding("ContactName").Header("連絡先名"))
        .Add(c => c.Binding("City").Header("都市"))
        .Add(c => c.Binding("Country").Header("国名"))
        .Add(c => c.Binding("Phone").Header("電話番号"))
        )
)
@Html.Partial("_Pager", "fGDisableServerView")
```

JS

Script copyCode

Copyright © GrapeCity, Inc. All rights reserved.

```
<script>
$(document).ready(function () {
    //Server Read を無効にします
    fGDisableServerView = wijmo.Control
.getControl('#fGDisableServerView');
});

//Server Read を無効にします
var fGDisableServerView = null;
</script>
```

編集

サーバー側の**CollectionViewHelper**クラスがEdit要求を定義して更新を処理します。このサンプルでは、更新アクションの定義方法を示します。このアクションは、CollectionViewHelperの**Edit**要求を処理することで、ソースデータベース内のレコードを更新可能にします。これによりユーザーは、FlexGridでデータを編集し、ソースデータベース内のそのデータを更新できます。

次の図は、**CollectionViewEditRequest**がコントローラーで使用された後のFlexGridを示しています。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

	ID	カテゴリ名	詳細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
✎	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish

次のコード例は、**CollectionViewEditRequest**を処理してFlexGridでレコードを編集および更新できるようにする方法を示します。

コードの場合

EditController.cs

C# copyCode

```
//FlexGrid に対してデータソースを定義します
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()
{
    return View(db);
}
//JSONのインスタンスを作成します
public ActionResult GridReadCategory([C1JsonRequest]
    CollectionViewRequest<Categories> requestData)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{
return this.C1Json(CollectionViewHelper.Read(requestData, db.Categories));
}

public ActionResult GridUpdateCategory([C1JsonRequest]
    CollectionViewEditRequest<Categories> requestData)
{
    return Update(requestData, db.Categories);
}
private ActionResult Update<T>(CollectionViewEditRequest<T>
    requestData, DbSet<T> data) where T : class
{
    return this.C1Json(CollectionViewHelper.Edit<T>(requestData, item =>
    {
        string error = string.Empty;
        bool success = true;
        try
        {
            db.Entry(item as object).State = EntityState.Modified;
            db.SaveChanges();
        }
        catch (DbEntityValidationException e)
        {
            error = string.Join(",", e.EntityValidationErrors.Select(result =>
            {
                return string.Join(",", result.ValidationErrors
                .Select(err => err.ErrorMessage));
            }));
            success = false;
        }
        catch (Exception e)
        {
            error = e.Message;
            success = false;
        }
        return new CollectionViewItemResult<T>
        {
            Error = error,
            Success = success && ModelState.IsValid,
            Data = item
        };
    }, () => data.ToList<T>()));
}
```

この例では、**Update**アクションはFlexGridの**ItemSource**のUpdateプロパティに割り当てられています。

Edit.cshtml

Razor copyCode

```
@(Html.C1().FlexGrid().Id("fGUCView").AutoGenerateColumns(false)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID").IsReadOnly(true))
        .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .Add(c => c.Binding("Description").Header("詳細").Width("500")))
    .Bind(ib =>
        ib.Bind(Url.Action("GridReadCategory"))
        .Update(Url.Action("GridUpdateCategory"))
        )
)
```

複数コントロールのデータ連結

CollectionViewは、**CollectionViewService**を使用して、MVCアプリケーションで連結している複数のコントロールをサポートします。FlexGridやFlexChartなどのさまざまなデータ連結コントロールを使用でき、CollectionViewを使用して連結できます。

最初に、CollectionViewを作成し、そのIdを設定します。FlexGridやFlexChartなどのデータ連結コントロールを初期化し、そのItemsSourceIdにCollectionViewのIdを割り当てることで、共通のCollectionViewを使用してコントロールを連結します。

CollectionViewは複数の項目連結コントロールに対して使用できます。それらのコントロールは同じitemsSourceを持ち、1つのコントロールのデータを変更すると他のコントロールも即座に変更されます。つまり、1つのCollectionViewを作成すれば、それを異なる複数のコントロールに使用できるということです。

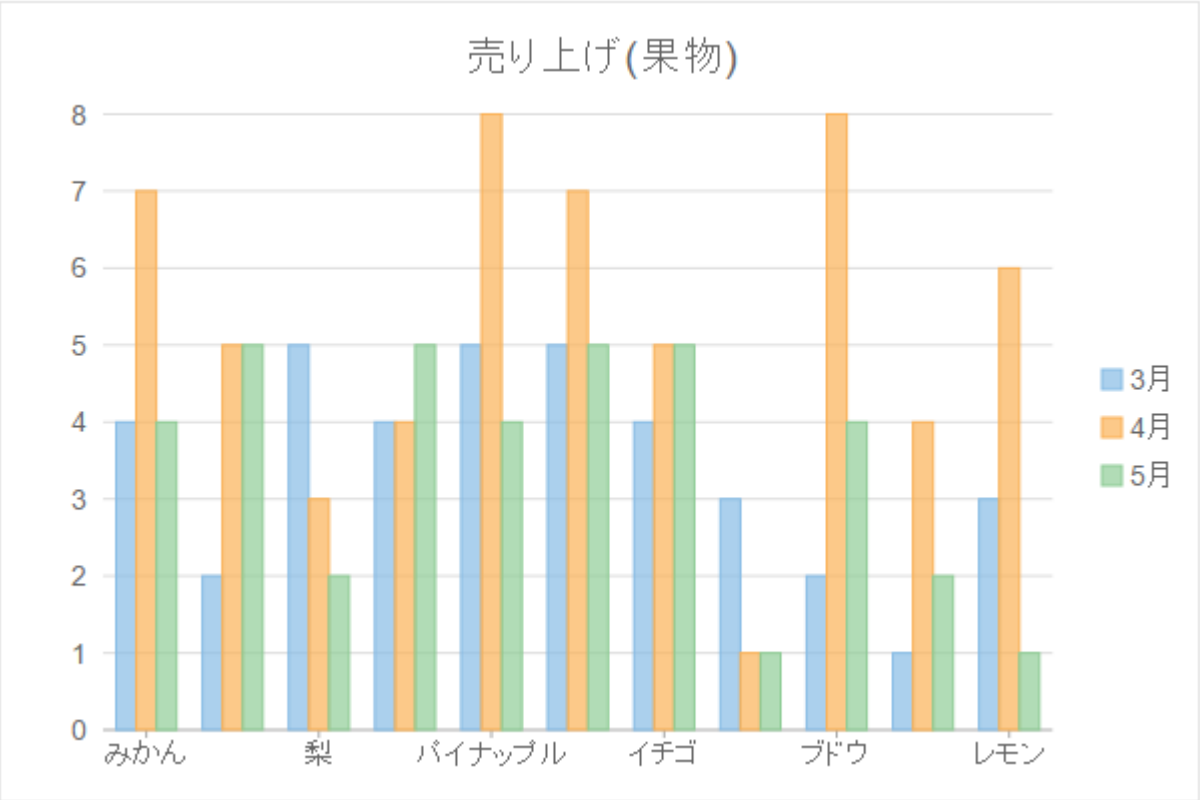
このセクションでは、MVC WebformでCollectionViewServiceをFlexGridおよびFlexChartコントロールと一緒に使用する方法を説明します。

次の図は、CollectionViewを介して連結された後のFlexGridとFlexChartが**CollectionViewService**を使用してどのように表示されるかを示しています。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

	ID	名前	国名	3月の価格	4月の価格	5月の価格
	1	Oranges	France	4	7	4
	2	Apples	Japan	2	5	5
	3	Pears	China	5	3	2
	4	Bananas	US	4	4	5
	5	Pineapples	Canada	5	8	4
	6	Peaches	US	5	7	5
	7	Strawberries	German	4	5	5
	8	Cherries	China	3	1	1
	9	Grapes	German	2	8	4
	10	Mangos	Korea	1	4	2
	11	Lemons	Canada	3	6	1
*						

Copyright © GrapeCity, Inc. All rights reserved.



アプリケーションのデータソースを設定

1. 新しいクラス**Fruit.cs**をModelsフォルダに追加します。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 新しいモデル内で次のコードを置き換えて、FlexGridコントロールとFlexChartコントロールのデータソースになるクラスを定義します。

Fruit.cs copyCode

```
public class Fruit
{
    public int ID { get; set; }
    public string Name { get; set; }
    public int MarPrice { get; set; }
    public int AprPrice { get; set; }
    public int MayPrice { get; set; }

    public string Country { get; set; }

    private static List<string> COUNTRIES = new List<string> { "US", "UK",
"Canada", "Japan", "China", "France", "German", "Italy", "Korea", "Australia" };
    public static IEnumerable<Fruit> GetFruitsSales()
    {
        var rand = new Random(0);
        var fruits = new[] { "みかん", "りんご", "梨", "バナナー", "パイナップル",
        "モモ", "イチゴ", "チェリー", "ブドウ", "マンゴー", "レモン"};
        var list = fruits.Select((f, i) =>
        {
            int id = i + 1;
            int mar = rand.Next(1, 6);
            int apr = rand.Next(1, 9);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            int may = rand.Next(1, 6);
            var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
            return new Fruit { ID = id, Country = country, Name = f, MarPrice =
mar, AprPrice = apr, MayPrice = may };
        });

        return list;
    }
}
```

先頭に戻る

次のコード例は、**CollectionViewService**を使用して、チャートコントロールとグリッドコントロールで複数のコントロールを連結できるようにする方法を示します。

コードの場合

MultipleControlsBindController.cs

C# copyCode

```
        public static List<Fruit> fruitsSales = Fruit.GetFruitsSales().ToList();
        public ActionResult Index()
        {
            return View(fruitsSales);
        }

        public ActionResult Update([C1JsonRequest]CollectionViewEditRequest<Fruit>
requestData)
        {
            return this.C1Json(CollectionViewHelper.Edit<Fruit>(requestData, fruit =>
            {
                string error = string.Empty;
                bool success = true;
                var fSale = fruitsSales.Find(item => item.ID == fruit.ID);
                fSale.Name = fruit.Name;
                fSale.Country = fruit.Country;
                fSale.MayPrice = fruit.MayPrice;
                fSale.MarPrice = fruit.MarPrice;
                fSale.AprPrice = fruit.AprPrice;
                return new CollectionViewItemResult<Fruit>
                {
                    Error = error,
                    Success = success && ModelState.IsValid,
                    Data = fSale
                };
            }, () => fruitsSales));
        }

        public ActionResult Create([C1JsonRequest]CollectionViewEditRequest<Fruit>
requestData)
        {
            return this.C1Json(CollectionViewHelper.Edit<Fruit>(requestData, fruit =>
            {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                string error = string.Empty;
                bool success = true;
                fruit.ID = getId();
                fruitsSales.Add(fruit);
                return new CollectionViewItemResult<Fruit>
                {
                    Error = error,
                    Success = success && ModelState.IsValid,
                    Data = fruit
                };
            }, () => fruitsSales));
        }

        public ActionResult Delete([C1JsonRequest]CollectionViewEditRequest<Fruit>
requestData)
        {
            return this.C1Json(CollectionViewHelper.Edit<Fruit>(requestData, fruit =>
            {
                string error = string.Empty;
                bool success = true;
                var fSale = fruitsSales.Find(item => item.Name == fruit.Name);
                fruitsSales.Remove(fSale);
                return new CollectionViewItemResult<Fruit>
                {
                    Error = error,
                    Success = success && ModelState.IsValid,
                    Data = fruit
                };
            }, () => fruitsSales));
        }

        private int getId()
        {
            int id = 1;
            while (fruitsSales.Find(item => item.ID == id) != null)
            {
                id++;
            }
            return id;
        }
    }
}
```

MultipleControlsBind.cshtml

Razor copyCode

```
@using C1.Web.Mvc;
@using System.Collections.Generic;
@using C1.Web.Mvc.Fluent;
@using MVC_DataBinding.Models;

@model List<Fruit>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().CollectionViewService().Id("fruitsSales")
                                   .Bind(Model).Update(Url.Action("Update", "Home"))
                                   .Delete(Url.Action("Delete", "Home"))
                                   .Create(Url.Action("Create", "Home")))

@(Html.C1().FlexGrid<Fruit>().Id("flexGrid")
                            .Columns(cls =>
                            {
                                cls.Add(col =>
col.Binding("ID").Header("ID").Visible(true).IsReadOnly(true));
                                cls.Add(col => col.Binding("Name").Header("名
前").Width("*"));
                                cls.Add(col => col.Binding("Country").Header("国
名").Width("*"));
                                cls.Add(col => col.Binding("MarPrice").Header("3月の価
格").Width("*"));
                                cls.Add(col => col.Binding("AprPrice").Header("4月の価
格").Width("*"));
                                cls.Add(col => col.Binding("MayPrice").Header("5月の価
格").Width("*"));
                            })
                        .IsReadOnly(true)
                        .AllowSorting(true)
                        .ItemsSourceId("fruitsSales")
                        .AutoGenerateColumns(false)
                        .IsReadOnly(false)
                        .AllowAddNew(true)
                        .AllowDelete(true)
                        .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)
                        .Width(600)
                        .Height(400)
)
@(Html.C1().FlexChart().Id("flexChart").Header("売り上げ(果物)")

.ItemsSourceId("fruitsSales").BindingX("Name").Series(sers =>
                        {
                            sers.Add().Binding("MarPrice").Name("3月");
                            sers.Add().Binding("AprPrice").Name("4月");
                            sers.Add().Binding("MayPrice").Name("5月");
                        }).SelectionMode(C1.Web.Mvc.Chart.SelectionMode.Point)
                        .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
                        .Rotated(false)
                        .Height(400)
                        .Width(600)
)
```

読み取り

サーバー側の**CollectionViewHelper**クラスは**Read**要求を定義してデータを取得します。これにより、**CollectionViewHelper**の**Read**要求を処理することでソースデータベースからのレコードを表示できます。

Copyright © GrapeCity, Inc. All rights reserved.

次の図は、**Read**要求を定義した後のFlexGridを示しています。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

	ID	カテゴリ名	詳細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish

次のコード例は、**CollectionViewHelper**の**Read**要求を使用してデータソースからのレコードをFlexGridで表示する方法を示します。

コードの場合

ReadController.cs

C# copyCode

```
//FlexGrid に対してデータソースを定義します
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()
{
    return View();
}
//JSONのインスタンスを作成します
public ActionResult GridReadCategory([C1JsonRequest]
CollectionViewRequest<Categories> requestData)
{
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Categories));
}
```

この例では、コントローラーの**GridReadCategory**アクションはFlexGridの**ItemSource**のBindプロパティに割り当てられ、データを挿入します。

Read.cshtml

Razor copyCode

```
@(Html.C1().FlexGrid().Id("fGRCView").AutoGenerateColumns(false).IsReadOnly(true)
    .CssClass("grid")
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID"))
        .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
        .Add(c => c.Binding("Description").Header("詳細").Width("500")))
    .Bind(
        ib => ib.Bind(Url.Action("GridReadCategory"))
    )
```


Copyright © GrapeCity, Inc. All rights reserved.

```
)
```

クライアント側処理

CollectionViewHelper internally performs server-side operations like **sorting**, **filtering**, **paging** on data for databound controls like FlexGrid, FlexChart and other Input controls. However, it is possible to explicitly perform these operations on client-side too. CollectionView has a powerful client-side API. The Client-side operations include **Current Record Management**, **Sorting**, **Filtering**, **Grouping** and **Tracking Changes**.

See Wijmo CollectionView API Documentation for more information on CollectionView's client-side API.

In this section, you learn about the following client-side operations that can be performed using CollectionView.

現在のレコードの管理
: Learn how to manage current record using CollectionView.

フィルタ処理
: Learn how to perform filtering in databound control using CollectionView.

グループ化
: Learn how to perform grouping in databound control using CollectionView.

ソート
: Learn how to perform sorting in databound control using CollectionView.

変更の追跡
: Learn how to keep track of changes made to the data in databound control using CollectionView.

現在のレコードの管理

CollectionViewは、**ICollectionView**インタフェースを使用して現在のレコードを管理できます。

コレクションのレコードの現在位置を取得するには、**currentPosition**プロパティを使用します。また、**moveCurrentTo(item)**、**moveCurrentToFirst()**、**moveCurrentToLast()**、**moveCurrentToNext()**、**moveCurrentToPosition(index)**、および**moveCurrentToPrevious()**メソッドを使用して現在位置を変更します。現在位置が変更されると、**currentChanging**イベントと**currentChanged**イベントを使用して追跡されます。**currentChanging**イベント内の現在の変更はキャンセルできます。

クライアント側のみで使用できるデータに対してフィルタ処理、ページング、ソートを行う場合は、**ItemSource**の**DisableServerRead**プロパティを必ず**true**に設定してください。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexGridでCollectionViewを介して現在のレコードを管理する方法を示します。

コードの場合

CurrentRecordManagementController.cs

C# copyCode

```
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()

{
    return View(db);
}
public ActionResult GridReadCategory([C1JsonRequest]
    CollectionViewRequest<Customers> requestData)
{
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Customers));
}
```

CurrentRecordManagement.cshtml

ASP.NET

Razor copyCode

```
<div>
    <button class="btn btn-default" id="btnCRMMoveNext">次へ</button>
    <button class="btn btn-default" id="btnCRMMovePre">前へ</button>
    <button class="btn btn-default" id="btnCRMStop4">4行目で停止</button>
    <button class="btn btn-default" id="btnCRMReset">停止をクリア</button>
</div>
@(Html.C1().FlexGrid().Id("crmGrid").IsReadOnly(true)
.SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)
.Height(300).Width(800).AutoGenerateColumns(true)
.Bind(b => b.DisableServerRead(true).Bind(Url.Action("GridReadCategory")))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
)
```

JS

Script copyCode

```
<script>
    $(document).ready(function () {
        //現在レコードを管理します
        crmGrid = wijmo.Control.getControl('#crmGrid');
        cvCRM = crmGrid.itemsSource;
        //new wijmo.collections.CollectionView(getData(10)),

        // ボタンクリックに対して処理を追加します
        // 次の項目へ移動します
   document.getElementById('btnCRMMoveNext').addEventListener('click', function () {
            cvCRM.moveCurrentToNext();
        });

        // 前の項目へ移動します
   document.getElementById('btnCRMMovePre').addEventListener('click', function () {
            cvCRM.moveCurrentToPrevious();
        });

        // 4番目の項目が現在の項目になる時は、変更を制限します
  document.getElementById('btnCRMStop4').addEventListener('click', function () {
            cvCRM.currentChanging.addHandler(stopCurrentIn4th);
        });

        // 変更できるように復元します
  document.getElementById('btnCRMReset').addEventListener('click', function () {
            cvCRM.currentChanging.removeHandler(stopCurrentIn4th);
        });

        // 現在の移動の動作を停止する関数を定義します
        function stopCurrentIn4th(sender, e) {
            // 4番目の項目現在の項目になる時、移動を停止します
            if (sender.currentPosition === 3) {
                e.cancel = true;
            }
        };
    });

    // collectionview および グリッドを作成します
    var crmGrid = null
        , cvCRM = null;

</script>
```

フィルタ処理

Copyright © GrapeCity, Inc. All rights reserved.

.NETと同様に、**CollectionView**クラスは**ICollectionViewインタフェース**を介してフィルタ処理をサポートします。フィルタ処理を有効にするには、ビューに入れるオブジェクトを決定する関数を**CollectionView.filter**プロパティに設定します。

次の図は、FlexGridコントロールにフィルタ処理が適用された後のFlexGridを示しています。

> クライアント側のみで使用できるデータに対してフィルタ処理、ページング、ソートを行う場合は、**ItemSource**の**DisableServerRead**プロパティを必ず**true**に設定してください。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

CustomerIDによるフィルタする文字を入力し

ID	会社名	連絡先名	都市	国名	電話番号
ALFKI	Alfreds Futter...	Maria Anders	Berlin	Germany	030-0074321
ANATR	Ana Trujillo E...	Ana Trujillo	México D.F.	Mexico	(5) 555-4729
ANTON	Antonio More...	Antonio Moreno	México D.F.	Mexico	(5) 555-3932
AROUT	Around the Horn	Thomas Hardy	London	UK	(171) 555-7788
BERGS	Berglunds sn...	Christina Berg...	Luleå	Sweden	0921-12 34 65
BLAUS	Blauer See D...	Hanna Moos	Mannheim	Germany	0621-08460
BLONP	Blondel père ...	Frédérique Ci...	Strasbourg	France	88.60.15.31
BOLID	Bólido Comid...	Martín Sommer	Madrid	Spain	(91) 555 22 82
BONAP	Bon app'	Laurence Leb...	Marseille	France	91.24.45.40
BOTTM	Bottom-Dollar...	Elizabeth Linc...	Tsawassen	Canada	(604) 555-4729
BSBEV	B's Beverages	Victoria Ashw...	London	UK	(171) 555-1212
CACTU	Cactus Comid...	Patricio Simps...	Buenos Aires	Argentina	(1) 135-5555
CENTC	Centro comer...	Francisco Ch...	México D.F.	Mexico	(5) 555-3392
CHOPS	Chop-suey C...	Yang Wang	Bern	Switzerland	0452-076545
COMMI	Comércio Min...	Pedro Afonso	São Paulo	Brazil	(11) 555-7647

任意の文字を入力してグリッドデータをフィルタ処理します。たとえば、次の図は**eA**を入力したときのFlexGridです。

eA

ID	会社名	連絡先名	都市	国名	電話番号
EASTC	Eastern Conn...	Ann Devon	London	UK	(171) 555-0297
GREAL	Great Lakes ...	Howard Snyder	Eugene	USA	(503) 555-7555
OCEAN	Océano Atlánt...	Yvonne Monc...	Buenos Aires	Argentina	(1) 135-5333
SAVEA	Save-a-lot Ma...	Jose Pavarotti	Boise	USA	(208) 555-8097

次のコード例は、CollectionViewを使用してFlexGridでフィルタ処理を適用する方法を示します。

コードの場合

FilteringController.cs

Copyright © GrapeCity, Inc. All rights reserved.

C# copyCode

```
 private C1NWindEntities db = new C1NWindEntities();

  public ActionResult Index()
  {
     return View(db);
   }
public ActionResult GridReadCategory([C1JsonRequest]
        CollectionViewRequest<Customers> requestData)
 {
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Customers));
 }
```

Filtering.cshtml

ASP.NET

Razor copyCode

```
<div>
    <input id="filteringInput" type="text" class="form-control app-pad"
           placeholder="CustomerIDによるフィルタする文字を入力してください"/>
</div>
<br/>
@(Html.C1().FlexGrid().Id("filteringGrid").IsReadOnly(true).
AllowSorting(false).AutoGenerateColumns(false)
    .Bind(b => b.Bind(Url.Action("GridReadCategory")).
DisableServerRead(true))
    .Columns(columns => columns
        .Add(c => c.Binding("CustomerID").Header("ID"))
        .Add(c => c.Binding("CompanyName").Header("会社名"))
        .Add(c => c.Binding("ContactName").Header("連絡先名"))
        .Add(c => c.Binding("City").Header("都市"))
        .Add(c => c.Binding("Country").Header("国名"))
        .Add(c => c.Binding("Phone").Header("電話番号"))
        )
)
```

JS

スクリプト copyCode

```
<script>

    $(document).ready(function () {
  //フィルタ処理
  // collectionview、 グリッド、タイムアウトが含むフィルタ、フィルタを入力
  // するフィルタを作成します
    filteringGrid = wijmo.Control.getControl('#filteringGrid');
    cvFiltering = filteringGrid.itemsSource;
    filteringInput = document.getElementById('filteringInput');
```


Copyright © GrapeCity, Inc. All rights reserved.

```
  // 入力時フィルタを適用します
        filteringInput.addEventListener('input', filterGrid);
    });

  //フィルタ処理
  // collectionview、 グリッド、タイムアウトが含むフィルタ、フィルタを入力する
  // フィルタを作成します
    var cvFiltering = null,
        filteringGrid = null,
        toFilter,
        filteringInput = null;

    // collection view に対してフィルタ処理を定義します
    function filterFunction(item) {
        var filter = filteringInput.value.toLowerCase();
        if (!filter) {
            return true;
        }

        return item.CustomerID.toLowerCase().indexOf(filter) > -1;
    };

    // フィルタを適用します(500 ms のタイムアウトに適用された)
    function filterGrid() {
        if (toFilter) {
            clearTimeout(toFilter);
        }

        toFilter = setTimeout(function () {
            toFilter = null;
            if (cvFiltering.filter === filterFunction) {
                cvFiltering.refresh();
            }
            else {
                cvFiltering.filter = filterFunction;
            }
        }, 500);
    };
</script>
```

グループ化

CollectionViewクラスは、.NETと類似の**ICollectionView**インタフェースを介してグループ化をサポートしています。グループ化を有効にするには、1つ以上の**GroupDescription**オブジェクトを**CollectionView.groupDescriptions**プロパティに追加し、グリッドインスタンス作成時にグリッドのShowGroupsプロパティを必ずtrueに設定します(デフォルト値はfalse)。

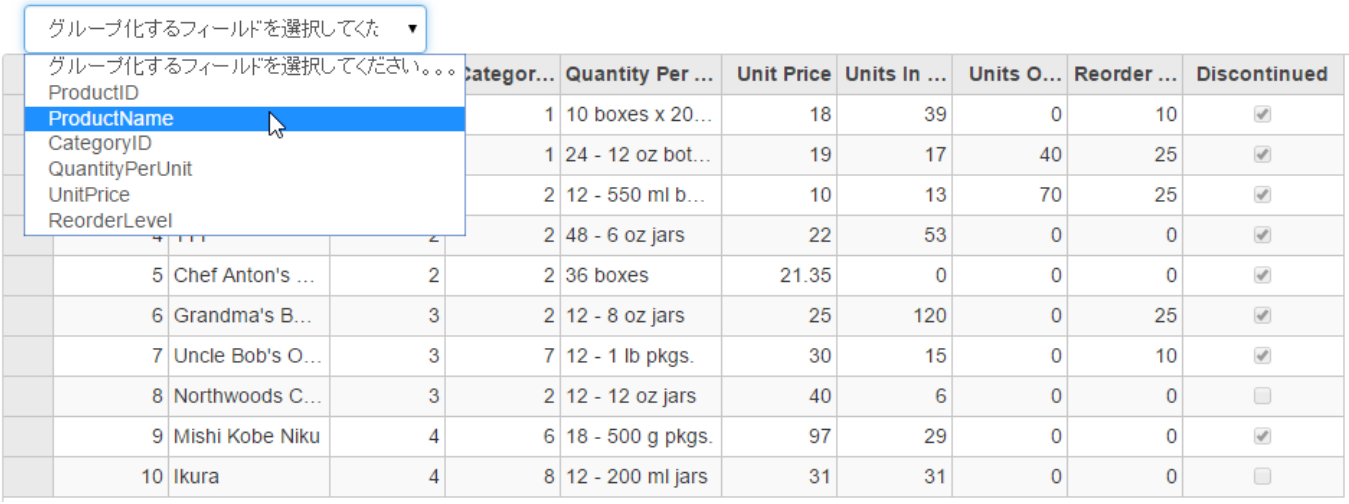
GroupDescriptionオブジェクトは柔軟であり、値またはグループ化関数に基づいてデータをグループ化できます。

次の例では、リストから選択するフィールドによってコレクションをグループ化します。グリッドには、項目のコンテンツだけでなく、グループ情報(グループ名とグループ内の金額の平均値)も表示されます。

> クライアント側のみで使用できるデータに対してフィルタ処理、ページング、ソートを行う場合は、**ItemSource**の**DisableServerRead**プロパティを必ず**true**に設定してください。

Copyright © GrapeCity, Inc. All rights reserved.

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。次の図は、グループ化が適用された後のFlexGridを示しています。



	5	Chef Anton's ...	2	2	36 boxes	21.35	0	0	0
	6	Grandma's B...	3	2	12 - 8 oz jars	25	120	0	25
	7	Uncle Bob's O...	3	7	12 - 1 lb pkgs.	30	15	0	10
	8	Northwoods C...	3	2	12 - 12 oz jars	40	6	0	0
	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97	29	0	0
	10	Ikura	4	8	12 - 200 ml jars	31	31	0	0

FlexGridをグループ化する際のキーになるフィールドをドロップダウンから選択できます。



	Product ID	Product Name	Supplier...	Categor...	Quantity Per ...	Unit Price	Units In ...	Units O...	Reorder ...	Discontinued
	◢ Product Name: **Chai** (1 項目)									
	1	Chai	1	1	10 boxes x 20...	18	39	0	10	
	◢ Product Name: **42** (1 項目)									
	2	42	1	1	24 - 12 oz bot...	19	17	40	25	
	◢ Product Name: **111** (2 項目)									
	3	111	1	2	12 - 550 ml b...	10	13	70	25	
	4	111	2	2	48 - 6 oz jars	22	53	0	0	
	◢ Product Name: **Chef Anton's Gumbo Mix** (1 項目)									
	5	Chef Anton's ...	2	2	36 boxes	21.35	0	0	0	
	◢ Product Name: **Grandma's Boysenberry Spread** (1 項目)									
	6	Grandma's B...	3	2	12 - 8 oz jars	25	120	0	25	

次のコード例は、FlexGridでCollectionViewを介してグループ化を適用する方法を示します。

コードの場合

GroupingController.cs

C# copyCode

```
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()
{
    return View(db);
}

public ActionResult Index1()
{
    return View(db);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}

public ActionResult GridReadCategory([C1JsonRequest] CollectionViewRequest<Products>
requestData)
{
    return this.C1Json(CollectionViewHelper.Read(requestData, db.Products));
}
```

Grouping.cshtml

ASP.NET

Razor copyCode

```
<div class="col-md-6">
    <select id="groupingFieldNameList" class="form-control"></select>
</div>

@(Html.C1().FlexGrid().Id("groupingGrid").IsReadOnly(true).AllowSorting(true)
    .AutoGenerateColumns(true).PageSize(10).Height(500)
    .Bind(b => b.DisableServerRead(true).Bind(Url.Action("GridReadCategory")))
)
```

JS

スクリプト copyCode

```
<script>

    function getGroupNames() {
        return ['ProductID', 'ProductName', 'CategoryID', 'QuantityPerUnit',
'UnitPrice', 'ReorderLevel'];
    };
    $(document).ready(function () {
        //グループ化します

        groupingGrid = wijmo.Control.getControl('#groupingGrid');
        cvGrouping = groupingGrid.itemsSource;
        groupingFieldNameList = document.getElementById('groupingFieldNameList');
        groupingFieldNameList.addEventListener('change', groupGrid);

        // リストを初期化してリストで選択の変更に対応します
        groupingFieldNameList.innerHTML += '<option value="" selected="selected">グ
ループ化するフィールドを選択してください。。。</option>';
        for (var i = 0; i < groupingNames.length; i++) {
            groupingFieldNameList.innerHTML += '<option value="' + groupingNames[i] +
'">' + groupingNames[i] + '</option>';
        }
    });
    // FlexGrid にてグループします
    // CollectionView、グリッド、select要素、と NameListを作成します
    var cvGrouping = null,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        groupingGrid = null,
        groupingFieldNameList = null,
        groupingNames = getGroupNames();

    // グループする設定を更新します
    function groupGrid() {
        var gd,
            fieldName = groupingFieldNameList.value;
        gd = cvGrouping.groupDescriptions;
        if (!fieldName) {
            // グループする設定をすべてクリアします
            gd.splice(0, gd.length);
            return;
        }
        if (findGroup(fieldName) >= 0) {
            return;
        }
        if (fieldName === 'UnitPrice') {
            // 価格によるグループする場合、特定の値の代わりに範囲を指定します
            gd.push(new wijmo.collections.PropertyGroupDescription(fieldName,
function (item, propName) {
                var value = item[propName]; // UnitPrice
                if (value > 100) return 'Large Amounts';
                if (value > 50) return 'Medium Amounts';
                if (value > 0) return 'Small Amounts';
                return 'Negative Amounts';
            }));
        }
        else {
            //特定のプロパティ値によるグループします
            gd.push(new wijmo.collections.PropertyGroupDescription(fieldName));
        }
    };
    // 指定したプロパティ名のグループが既に存在しているかを確認します
    function findGroup(propName) {
        var gd = cvGrouping.groupDescriptions;
        for (var i = 0; i < gd.length; i++) {
            if (gd[i].propertyName === propName) {
                return i;
            }
        }
        return -1;
    };
</script>
```

ソート

.NETと同様に、**CollectionView**クラスは**ICollectionView**インタフェースを介してソートをサポートします。ソートを有効にするには、1つ以上の**sortDescriptions**オブジェクトを**CollectionView.sortDescriptions**プロパティに追加します。ソート結果は**CollectionView.items**プロパティから取得できます。

SortDescriptionオブジェクトは柔軟であり、値に基づいて昇順または降順でデータをソートできます。次のサンプルでは、最

Copyright © GrapeCity, Inc. All rights reserved.

初のリストで選択されたフィールド値に基づいてコレクションをソートすることができます。また、2番目のリストでソート順を指定することもできます。

> クライアント側のみで使用できるデータに対してフィルタ処理、ページング、ソートを行う場合は、**ItemSource**の**DisableServerRead**プロパティを必ず**true**に設定してください。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。



次のコード例は、FlexGridでCollectionViewを介してソートを適用する方法を示します。

コードの場合

SortingController.cs

C# copyCode

```
    private C1NWindEntities db = new C1NWindEntities();
    public ActionResult Index()
    {
        return View(db);
    }
    public ActionResult GridReadCategory([C1JsonRequest]
        CollectionViewRequest<Customers> requestData)
    {
        return this.C1Json(CollectionViewHelper.Read(requestData, db.Customers));
    }
}
```

Sorting.cshtml

ASP.NET

Razor copyCode

Copyright © GrapeCity, Inc. All rights reserved.

```
<div class="row-fluid well row">
    <div class="col-md-8">
        <select id="sortingFieldNameList" class="form-control"></select>
    </div>
    <div class="col-md-4">
        <select id="sortingOrderList" class="form-control">
            <option value="true"selected="selected">上昇</option>
            <option value="false">下降</option>
        </select>
    </div>
</div>

@(Html.C1().FlexGrid().Id("sortingGrid").IsReadOnly(true)
.AllowSorting(false).AutoGenerateColumns(false)
    .Bind(b => b.DisableServerRead(true).Bind(Url.Action("GridReadCategory")))
    .Columns(columns => columns
        .Add(c => c.Binding("CustomerID").Header("ID"))
        .Add(c => c.Binding("CompanyName").Header("会社名"))
        .Add(c => c.Binding("ContactName").Header("連絡先"))
        .Add(c => c.Binding("City").Header("都市"))
        .Add(c => c.Binding("Country").Header("国名"))
        .Add(c => c.Binding("Phone").Header("電話番号"))
        )
)
```

JS

スクリプト copyCode

```
<script>
    function getNames() {
        return ['CustomerID', 'CompanyName', 'ContactName',
'City', 'Country', 'Phone'];
    };

    $(document).ready(function () {
        //並び替え処理
        sortingGrid = wijmo.Control.getControl('#sortingGrid');
        cvSorting = sortingGrid.itemsSource;
        sortingFieldNameList = document.getElementById('sortingFieldNameList');
        sortingOrderList = document.getElementById('sortingOrderList');
        // FieldNameとOrder のリストを初期化します
        sortingFieldNameList.innerHTML += '<option value=""
selected="selected">並び替えするフィールドを選択してください。。。</option>';
        for (var i = 0; i < sortingNames.length; i++) {
            sortingFieldNameList.innerHTML += '<option value="'
+ sortingNames[i] + '">' + sortingNames[i] + '</option>';
        }

        // sortDescriptions プロパティを更新するようにリストの変更を追跡します
        sortingFieldNameList.addEventListener('change', sortGrid);
        sortingOrderList.addEventListener('change', sortGrid);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        //並び替え処理
        // collectionview, グリッド, jQuery の要素とフィールド名リストを作成します
        var cvSorting = null,
            sortingGrid = null,
            sortingFieldNameList = null,
            sortingOrderList = null,
            sortingNames = getNames();

        function sortGrid() {
            var fieldName = sortingFieldNameList.value,
                ascending = sortingOrderList.value,
                sd, sdNew;

            if (!fieldName) {
                return;
            }

            ascending = ascending === 'true';
            sd = cvSorting.sortDescriptions;
            sdNew = new wijmo.collections
.SortDescription(fieldName, ascending);

            // 並び替えに関する既存の記述子を削除して新規のを追加します
            sd.splice(0, sd.length, sdNew);
        };

    })
</script>
```

Sort Null Values

CollectionView allows you to sort null values using the **SortNulls** property, which determines how the null values are sorted. The **SortNulls** property accepts values from the **CollectionViewSortNulls** enumeration that lets you choose the sorting order from the following values:

- **Last** : This is set as the default value which causes null values to appear last on the sorted collection, regardless of sort direction.
- **First** : This value causes null values to appear first on the sorted collection, regardless of sort direction.
- **Natural** : This value sorts null values in natural order, i.e. first in ascending and then in descending order.

The following example illustrates how to sort null values so that they appear first on the sorted collection. This example uses a class named Sale which contains sales data for various countries.

Copyright © GrapeCity, Inc. All rights reserved.

ID	Start	End	Country	Product	Color	Amount	Amount2	Discount	Active
1	Jan 25 20	00:00	German	Gadget	Green	$581.61	$1,030.17	14%	☑
2	Feb 25 20	01:30	Canada	Gadget	Green	$4,919.02	$163.13	17%	☐
3	Mar 25 20	02:00	Japan	Gadget	Red	$2,159.73	$594.79	7%	☐
4	Apr 25 20	03:30	German	Gadget	Red	$1,248.66	$1,092.03	22%	☐
5	May 25 20	04:00	German	Gadget	Black	$4,051.76	$945.62	12%	☑
6	Jun 25 20	05:30	Canada	Gadget	Black	($3,131.28)	$342.59	16%	☐
7	Jul 25 20	06:00	China	Widget	Red	$698.62	$1,127.02	15%	☐
8	Aug 25 20	07:30	US	Widget	White	$3,464.15	$3,065.79	3%	☐
9	Sep 25 20	08:00	Korea	Gadget	Black	($2,363.16)	$4,212.80	14%	☑
10	Oct 25 20	09:30	US	Widget	White	($2,836.94)	$358.45	6%	☐
0	Jun 19 20	13:01				$0.00	$0.00	0%	☐

Sale.cs

Sale.cs copyCode

```
public class Sale
{
    public Sale()
    {
        Start = End = DateTime.Now;
    }
    public int ID { get; set; }
    public DateTime Start { get; set; }
    public DateTime End { get; set; }
    public string Country { get; set; }
    public string Product { get; set; }
    public string Color { get; set; }
    public double Amount { get; set; }
    public double Amount2 { get; set; }
    public double Discount { get; set; }
    public bool Active { get; set; }

    public MonthData[] Trends { get; set; }
    public int Rank { get; set; }

    private static List<string> COUNTRIES = new List<string> { "US", "UK", "Canada",
"Japan", "China", "France", "German", "Italy", "Korea", "Australia" };
    private static List<string> PRODUCTS = new List<string> { "Widget", "Gadget",
"Doohickey" };

    /// <summary>
    /// データを取得します。
    /// </summary>
    /// <param name="total"></param>
    /// <returns></returns>
    public static IEnumerable<Sale> GetData(int total)
    {
        var colors = new[] { "Black", "White", "Red", "Green", "Blue" };
        var rand = new Random(0);
        var dt = DateTime.Now;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        var list = Enumerable.Range(0, total).Select(i =>
        {
            var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
            var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
            var color = colors[rand.Next(0, colors.Length - 1)];
            var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
            var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, (i % 2) * 30,
0);

            return new Sale
            {
                ID = i + 1,
                Start = startDate,
                End = endDate,
                Country = country,
                Product = product,
                Color = color,
                Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                Amount2 = Math.Round(rand.NextDouble() * 5000, 2),
                Discount = Math.Round(rand.NextDouble() / 4, 2),
                Active = (i % 4 == 0),
                Trends = Enumerable.Range(0, 12).Select(x => new MonthData { Month =
x + 1, Data = rand.Next(0, 100) }).ToArray(),
                Rank = rand.Next(1, 6)
            };
        });
        return list;
    }
}

public class MonthData
{
    public int Month { get; set; }
    public double Data { get; set; }
}
```

Controller Code

SortingController.cs copyCode

```
private C1NWindEntities db = new C1NWindEntities();

//ソート処理
public ActionResult Index(FormCollection data)
{
    var model = Sale.GetData(10).ToList();
    var nullObj = new Sale { };
    model.Add(nullObj);
    return View(model);
}
```

View Code

Copyright © GrapeCity, Inc. All rights reserved.

Index.cshtml copyCode

```
@using C1.Web.Mvc.Grid
@using CVSorting_MVC.Models

@model IEnumerable<Sale>

@(Html.C1().CollectionViewService()
    .Id("CVService")
    .Bind(Model)
    .SortNulls(CollectionViewSortNulls.First)
)
@(Html.C1().FlexGrid<Sale>()
    .Id("FlexGridCV")
    .ItemsSourceId("CVService")
    .AutoGenerateColumns(false)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("Start").Format("MMM d yy"));
        bl.Add(cb => cb.Binding("End").Format("HH:mm"));
        bl.Add(cb => cb.Binding("Country"));
        bl.Add(cb => cb.Binding("Product"));
        bl.Add(cb => cb.Binding("Color"));
        bl.Add(cb => cb.Binding("Amount").Format("c"));
        bl.Add(cb => cb.Binding("Amount2").Format("c"));
        bl.Add(cb => cb.Binding("Discount").Format("p0"));
        bl.Add(cb => cb.Binding("Active"));
    })
    .Height(500)
)
```

変更の追跡

CollectionViewクラスは、データに加えられた変更を追跡できます。これは、変更をサーバーに送信する必要がある場合に便利です。変更追跡をオンにするには、**TrackChanges**プロパティをtrueに設定します。これを行うと、**CollectionView**はデータに加えられた変更の追跡を開始し、以下を使用して公開します。

- **ItemsEdited**：このリストには、**BeginEdit**メソッドと**CommitEdit**メソッドを使用して編集された項目が含まれます。
- **ItemsAdded**：このリストには、**AddNew**メソッドと**CommitNew**メソッドを使用して追加された項目が含まれます。
- **ItemsRemoved**：このリストには、**Remove**メソッドを使用して削除された項目が含まれます。

> クライアント側のみで使用できるデータに対してフィルタ処理、ページング、ソートを行う場合は、**ItemSource**の**DisableServerRead**プロパティを必ず**true**に設定してください。

この例ではC1NWindデータソースを使用しますが、これは「クイックスタート」で示されているようにアプリケーション内で設定されたものです。

次の図は、**TrackChanges**プロパティを**true**に設定した後のFlexGridを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexGridを初期化し、CollectionViewの**TrackChanges**プロパティを使用して変更を記録する方法を示します。

コードの場合

TrackChangesController.cs

C# copyCode

```
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index()
{

    return View(db);
}
```

TrackChanges.cshtml

Copyright © GrapeCity, Inc. All rights reserved.

ASP.NET

Razor copyCode

```
<h5>データを以下で変更してください</h5>
@(Html.C1().FlexGrid().Id("tcMainGrid").AutoGenerateColumns(false)
    .AllowAddNew(true).AllowDelete(true)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID"))
        .Add(c => c.Binding("CategoryName"))
        .Add(c => c.Binding("Description").Header("詳細").Width("*")))
    .Bind(ib =>
        ib.Bind(Url.Action("GridReadCategory")).DisableServerRead(true)
        )
)
<h5>変更をここで参照できます</h5>
<h6>編集された項目:</h6>
@(Html.C1().FlexGrid().Id("tcEditedGrid").AutoGenerateColumns(false)
    .IsReadOnly(true).Height(100)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID").IsReadOnly(true))
        .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
        .Add(c => c.Binding("Description").Header("詳細").Width("*")))
)
<h6>追加された項目:</h6>
@(Html.C1().FlexGrid().Id("tcAddedGrid").AutoGenerateColumns(false)
    .IsReadOnly(true).Height(100)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID").IsReadOnly(true))
        .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
        .Add(c => c.Binding("Description").Header("詳細").Width("*")))
)
<h6>削除された項目:</h6>
@(Html.C1().FlexGrid().Id("tcRemovedGrid").AutoGenerateColumns(false)
    .IsReadOnly(true).Height(100)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID").IsReadOnly(true))
        .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
        .Add(c => c.Binding("Description").Header("詳細").Width("*")))
)
```

JS

スクリプト copyCode

```
<script>

    $(document).ready(function () {
        //変更の追跡処理
        tcMainGrid = wijmo.Control.getControl('#tcMainGrid');
        // データを変更する FlexGrid
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        tcEditedGrid = wijmo.Control.getControl('#tcEditedGrid');
        // 編集された項目を保存する FlexGrid
        tcAddedGrid = wijmo.Control.getControl('#tcAddedGrid');
        // 追加された項目を保存する FlexGrid
        tcRemovedGrid = wijmo.Control.getControl('#tcRemovedGrid');
        // 削除された項目を保存する FlexGrid
        cvTrackingChanges = tcMainGrid.itemsSource;

        tcEditedGrid.itemsSource = cvTrackingChanges.itemsEdited;
        tcAddedGrid.itemsSource = cvTrackingChanges.itemsAdded;
        tcRemovedGrid.itemsSource = cvTrackingChanges.itemsRemoved;

        // collectionviewすべての変更を追跡します
        cvTrackingChanges.trackChanges = true;
    });

    //変更の追跡処理
    var tcMainGrid = null,
        tcEditedGrid = null,
        tcAddedGrid = null,
        tcRemovedGrid = null,
        cvTrackingChanges = null;
</script>
```

計算フィールド

Calculated fields are new columns created by the CollectionView and placed inside FlexGrid that use custom expressions to populate their cells. CollectionView allows you to add calculated fields to your data sources. Calculated fields offer dynamic values that are updated automatically as data changes. You can add calculated fields to CollectionView by setting **CalculatedFields** property of the CollectionView to a value that contains the names of the calculated fields and functions used to obtain the field values. The functions may be specified as regular JavaScript functions that take the current data item as an argument or as string expressions with an "$" value representing the current data item.

Calculated fields are read-only and are automatically updated based on their dependent fields change. The following code demonstrates the use of calculated fields in a grid to populate the cells.

Copyright © GrapeCity, Inc. All rights reserved.

ID	Start	End	Country	Product	Color	Amount2	Discount	Description	Net Revenue	Tax	Active
1	Jan 25 21	00:00	German	Gadget	Green	$1,030.17	14%	Green Gadget	$885.95	$88.59	☑
2	Feb 25 21	01:30	Canada	Gadget	Green	$163.13	17%	Green Gadget	$135.40	$13.54	☐
3	Mar 25 21	02:00	Japan	Gadget	Red	$594.79	7%	Red Gadget	$553.15	$55.32	☐
4	Apr 25 21	03:30	German	Gadget	Red	$1,092.03	22%	Red Gadget	$851.78	$85.18	☐
5	May 25 21	04:00	German	Gadget	Black	$945.62	12%	Black Gadget	$832.15	$83.21	☑
6	Jun 25 21	05:30	Canada	Gadget	Black	$342.59	16%	Black Gadget	$287.78	$28.78	☐
7	Jul 25 21	06:00	China	Widget	Red	$1,127.02	15%	Red Widget	$957.97	$95.80	☐
8	Aug 25 21	07:30	US	Widget	White	$3,065.79	3%	White Widget	$2,973.82	$297.38	☐
9	Sep 25 21	08:00	Korea	Gadget	Black	$4,212.80	14%	Black Gadget	$3,623.01	$362.30	☑
10	Oct 25 21	09:30	US	Widget	White	$358.45	6%	White Widget	$336.94	$33.69	☐
11	Nov 25 21	10:00	France	Widget	Green	$3,861.01	8%	Green Widget	$3,552.13	$355.21	☐
12	Dec 25 21	11:30	Korea	Widget	Red	$974.55	12%	Red Widget	$857.60	$85.76	☐
13	Jan 25 21	12:00	German	Gadget	Red	$1,324.17	12%	Red Gadget	$1,165.27	$116.53	☑
14	Feb 25 21	13:30	German	Widget	Black	$550.31	18%	Black Widget	$451.25	$45.13	☐
15	Mar 25 21	14:00	Japan	Gadget	Green	$448.52	16%	Green Gadget	$376.76	$37.68	☐
16	Apr 25 21	15:30	China	Gadget	Black	$2,766.17	8%	Black Gadget	$2,544.88	$254.49	☐
17	May 25 21	16:00	US	Widget	Black	$2,811.90	18%	Black Widget	$2,305.76	$230.58	☑

The following code demonstrates how you can add calculated fields to your data source. The example uses Sale.cs model added in the Custom Editors topic.

Controller Code

CalculatedFieldController.cs copyCode

```
public ActionResult Index()
{
    var model = Sale.GetData(500);
    return View(model);
}
```

View Code

Index.cshtml copyCode

```
@model IEnumerable<Sale>

<style>
    .calculated {
        background-color: honeydew;
    }

    .wj-alt.calculated {
        background-color: #DCF8C6;
    }
</style>

@(Html.C1().CollectionViewService()
    .Id("CVService")
    .Bind(Model)
    .CalculatedFields(cfs =>
    {
        cfs.Add(cf => cf.Name("Description").Expression("[$.Color,$.Product].join('
```


Copyright © GrapeCity, Inc. All rights reserved.

```
')"));
        cfs.Add(cf => cf.Name("NetRevenue").Expression("$.Amount2*(1-$.Discount)"));
        cfs.Add(cf => cf.Name("Tax").Expression("$.NetRevenue*0.1"));
    })
)
@(Html.C1().FlexGrid<Sale>()
    .Id("FlexGridCV")
    .ItemsSourceId("CVService")
    .AutoGenerateColumns(false)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID").Width("60"));
        bl.Add(cb => cb.Binding("Start").Format("MMM d yy"));
        bl.Add(cb => cb.Binding("End").Format("HH:mm"));
        bl.Add(cb => cb.Binding("Country"));
        bl.Add(cb => cb.Binding("Product"));
        bl.Add(cb => cb.Binding("Color"));
        bl.Add(cb => cb.Binding("Amount2").Format("c"));
        bl.Add(cb => cb.Binding("Discount").Format("p0"));
        bl.Add(cb => cb.Binding("Description").CssClass("calculated"));
        bl.Add(cb => cb.Binding("NetRevenue").Header("Net
Revenue").Format("c").CssClass("calculated"));
        bl.Add(cb => cb.Binding("Tax").Format("c").CssClass("calculated"));
        bl.Add(cb => cb.Binding("Active").Width("90"));
    })
    .SortingType(AllowSorting.SingleColumn)
    .CssClass("grid")
    .Height(500)
)
```

Note: To use calculated fields in IE11, you must include a proxy polyfill such as https://www.npmjs.com/package/proxy-polyfill.

DateTime処理

このトピックでは、FlexGridとAjax連結またはAjax編集を使用する場合に、サーバーとクライアントの両方でUTC形式のDateTimeプロパティを維持する方法を示します。それには、まずクライアント側とサーバー側でDateTime値がどのように処理されるかを理解する必要があります。

サーバー側では、**DateTime**クラスを使用するたびに、**Kind**プロパティを指定する必要があります。このプロパティは、そのインスタンスで表される時刻がローカル時間か、協定世界時（UTC）か、またはそのどちらでもないかを示します。一方、DateTimeオブジェクトをクライアント側で使用する場合は、日付が**Number**から**Date**オブジェクトに解析されるときに、ブラウザによってすべての日付が暗黙にローカル時間に変換されます。

たとえば、サーバー側で**new DateTime(2017, 0, 25, 7, 0, 0, DateTimeKind.Utc);**のようにDateTimeインスタンスを作成したとします。この値をサーバーからクライアントに転送すると、異なるTimeZoneシステム設定を使用するさまざまなマシンのブラウザごとに、異なる文字列表現が表示されます。

UTC形式の時刻を維持するには、クライアントとサーバーの両方で日付に明示的な変換を適用する必要があります。

サーバー

サーバーでは、すべてのDateTimeオブジェクトをUnspecifiedに変換し、必要に応じてCREATE、UPDATE、またはDELETE操作を使用してこれを元に戻します。次の2つの手順を実装する必要があります。

Copyright © GrapeCity, Inc. All rights reserved.

1. データを読み取るときにUnspecified形式に変換する。
2. CREATE、UPDATE、DELETEの各操作でUnspecified形式を元に戻す。

この変換を理解するために、Utc、Local、Unspecifiedなどの形式の**DateTime**フィールドを含むFlexGridコントロールを例にして、このグリッドをデータ値と連結します。

新しいモデルの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：DatesData.cs）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 新しいモデルの次のコードを書き換えて、FlexGridコントロールのデータソースになるクラスを定義します。

DatesData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace DateTimeFields.Models
{
    public class DatesData
    {

        /// 主キー。
        public int Id { get; set; }

        /// KindがUtcのDateTimeフィールド。
        public DateTime UtcDateTime { get; set; }

        /// KindがUnspecifiedのDateTimeフィールド。
        public DateTime UnspecifiedDateTime { get; set; }

        /// KindがLocalのDateTimeフィールド。
        public DateTime LocalDateTime { get; set; }

        /// データを取得します。
        /// <param name="total"></param>
        /// <returns></returns>
        public static IEnumerable<DatesData> GetData(int total)
        {
            var rand = new Random(0);
            var dt = DateTime.Now;
            var list = Enumerable.Range(0, total).Select(i =>
            {
                return new DatesData
                {
                    Id = i + 1,
                    UtcDateTime = new DateTime(dt.Year, i % 12 + 1, 25, 7, 0, 0,
DateTimeKind.Utc),
                    UnspecifiedDateTime = new DateTime(dt.Year, i % 12 + 1, 25,
7, 0, 0, DateTimeKind.Unspecified),
                    LocalDateTime = new DateTime(dt.Year, i % 12 + 1, 25, 7, 0,
0, DateTimeKind.Local)
                };
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            });

            return list;
        }
    }
}
```

コントローラー

コード内 - HomeController.cs

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serialization;
using <ApplicationName>.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace DateTimeFields.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }

        private static List<DatesData> convertedData = DatesData.GetData(3).ToList();
        private static List<DatesData>
ConvertToUnspecifiedData(IEnumerable<DatesData> sourceData)
        {
            return sourceData.Select(item => new DatesData
            {
                Id = item.Id,
                UnspecifiedDateTime = item.UnspecifiedDateTime,
                UtcDateTime = new DateTime(item.UtcDateTime.Ticks),
                LocalDateTime = new DateTime(item.LocalDateTime.Ticks)
            }).ToList();
        }

        private static void
ConvertUpspecifiedBack(CollectionViewEditRequest<DatesData> requestData)
        {
            // UnspecifiedのDateTimeを元に戻します。
            foreach (var item in requestData.OperatingItems)
            {
                item.LocalDateTime = new DateTime(item.LocalDateTime.Ticks,
DateTimeKind.Local);
                item.UtcDateTime = new DateTime(item.UtcDateTime.Ticks,
DateTimeKind.Utc);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            }
        }
        public ActionResult Converted_ReadDatesData([C1JsonRequest]
CollectionViewRequest<DatesData> requestData)
        {
            return this.C1Json(CollectionViewHelper.Read(requestData,
ConvertToUnspecifiedData(convertedData)));
        }
        public ActionResult
Converted_UpdateDatesData([C1JsonRequest]CollectionViewEditRequest<DatesData>
requestData)
        {
            ConvertUpspecifiedBack(requestData);
            return Update(requestData, convertedData, ConvertToUnspecifiedData);
        }
        public ActionResult
Converted_CreateDatesData([C1JsonRequest]CollectionViewEditRequest<DatesData>
requestData)
        {
            ConvertUpspecifiedBack(requestData);
            return Create(requestData, convertedData, ConvertToUnspecifiedData);
        }
        public ActionResult
Converted_DeleteDatesData([C1JsonRequest]CollectionViewEditRequest<DatesData>
requestData)
        {
            ConvertUpspecifiedBack(requestData);
            return Delete(requestData, convertedData, ConvertToUnspecifiedData);
        }

        public ActionResult Update(CollectionViewEditRequest<DatesData> requestData,
List<DatesData> sourceData, Func<IEnumerable<DatesData>, List<DatesData>> converter =
null)
        {
            return this.C1Json(CollectionViewHelper.Edit(requestData, item =>
            {
                var error = string.Empty;
                var success = true;
                try
                {
                    var index = sourceData.FindIndex(u => u.Id == item.Id);
                    sourceData.RemoveAt(index);
                    sourceData.Insert(index, item);
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }
                return new CollectionViewItemResult<DatesData>
                {
                    Error = error,
                    Success = success,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                    Data = item
                };
            }, () => converter != null ? converter(sourceData) : sourceData));
        }

        public ActionResult Create(CollectionViewEditRequest<DatesData> requestData,
List<DatesData> sourceData, Func<IEnumerable<DatesData>, List<DatesData>> converter =
null)
        {
            return this.C1Json(CollectionViewHelper.Edit(requestData, item =>
            {
                var error = string.Empty;
                var success = true;
                try
                {
                    sourceData.Add(item);
                    item.Id = sourceData.Max(u => u.Id) + 1;
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }
                return new CollectionViewItemResult<DatesData>
                {
                    Error = error,
                    Success = success,
                    Data = item
                };
            }, () => converter != null ? converter(sourceData) : sourceData));
        }

        public ActionResult Delete(CollectionViewEditRequest<DatesData> requestData,
List<DatesData> sourceData, Func<IEnumerable<DatesData>, List<DatesData>> converter =
null)
        {
            return this.C1Json(CollectionViewHelper.Edit(requestData, item =>
            {
                var error = string.Empty;
                var success = true;
                try
                {
                    var index = sourceData.FindIndex(u => u.Id == item.Id);
                    sourceData.RemoveAt(index);
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }
                return new CollectionViewItemResult<DatesData>
                {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                    Error = error,
                    Success = success,
                    Data = item
                };
            }, () => converter != null ? converter(sourceData) : sourceData));
        }
    }
}
```

ビューの追加

コード内 - Index.cshtml

DatesData.cs

```
@(Html.C1().FlexGrid()
    .Id("convertedGrid")
    .AllowAddNew(true)
    .AllowDelete(true)
    .AutoGenerateColumns(false)
    .Bind(cvb => cvb.Bind(Url.Action("Converted_ReadDatesData"))
                    .Create(Url.Action("Converted_CreateDatesData"))
                    .Update(Url.Action("Converted_UpdateDatesData"))
                    .Delete(Url.Action("Converted_DeleteDatesData")))
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("Id").IsReadOnly(true).Visible(false));
        columns.Add(column => column.Binding("UtcDateTime").Format("dd/MM/yyyy
HH:mm:ss").Width("*"));
        columns.Add(column => column.Binding("LocalDateTime").Format("dd/MM/yyyy
HH:mm:ss").Width("*"));
        columns.Add(column =>
column.Binding("UnspecifiedDateTime").Format("dd/MM/yyyy HH:mm:ss").Width("*"));
    })
    .Filterable(f => f.DefaultFilterType(FilterType.Both)))
```

クライアント

クライアントでは、明示的な変換を行うために、**OnClientReponseTextParsing**と**OnClientRequestDataStringifying**の2つのクライアントイベントがComponentOne MVCコンポーネントに用意されています。

- **OnClientReponseTextParsing** - サーバーからクライアントにデータが取得されるときに、すべてのデータがJSONテキストにシリアライズされ、そのテキストがクライアントに取得されます。クライアントでは、そのテキストがJavaScriptオブジェクトに解析されます。解析中に変換を実行する必要があります。次のイベント引数をサポートします。
  - Key - 解析される項目テキストの名前。
  - Value：解析される項目のテキスト。
  - Result：テキストの解析後の値を指定します。
  - Cancel：デフォルトの解析が必要ない場合は、この値をtrueに設定します。
- **OnClientRequestDataStringifying** - サーバーからクライアントにデータが戻される場合。データがテキストにシリアライズされ、サーバーに送信されます。サーバーで、テキストがシリアライズ解除されてオブジェクトに変換されます。JavaScript Dateオブジェクトは、常にUTC形式のテキストにシリアライズされます。シリアライズをカスタマイズしない場合は、シリアライズ解除後にUTC形式のDateTimeオブジェクトが取得されます。これは、正しい実装方法ではありません。解析中にOnClientRequestDataStringifyingイベントを使用して変換を行う必要があります。
  - Key - シリアライズされる項目テキストの名前。

Copyright © GrapeCity, Inc. All rights reserved.

- Value：シリアライズされる項目の名前。
- Result：オブジェクトのシリアライズ後のテキストを指定します。
- Cancel：デフォルトのシリアライズが必要ない場合は、これをtrueに設定します。

この変換を理解するために、Utc、Local、Unspecifiedなどの形式の**DateTime**フィールドを含むFlexGridコントロールを例にして、このグリッドをデータ値と連結します。

新しいモデルの追加

1. 新しいデータクラスを**[モデル]**フォルダに追加します。この例では、サーバーセクションで作成した同じ**DatesData.cs**モデルを使用します。

新しいコントローラーの追加

コード内 - HomeController.cs

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serialization;
using <ApplicationName>.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;

namespace DateTimeFields.Controllers {
 public class HomeController: Controller {
  public ActionResult Index() {
   return View();
  }

  private static List <DatesData> customData = DatesData.GetData(3).ToList();
  public ActionResult Custom_ReadDatesData([C1JsonRequest] CollectionViewRequest
<DatesData> requestData) {
   return this.C1Json(CollectionViewHelper.Read(requestData, customData));
  }

  public ActionResult Custom_UpdateDatesData([C1JsonRequest]
CollectionViewEditRequest <DatesData> requestData) {
   return Update(requestData, customData);
  }

  public ActionResult Custom_CreateDatesData([C1JsonRequest]
CollectionViewEditRequest <DatesData> requestData) {
   return Create(requestData, customData);
  }

  public ActionResult Custom_DeleteDatesData([C1JsonRequest]
CollectionViewEditRequest <DatesData> requestData) {
   return Delete(requestData, customData);
  }

  public ActionResult Update(CollectionViewEditRequest <DatesData> requestData, List
<DatesData> sourceData, Func <IEnumerable <DatesData> , List <DatesData>> converter =
```


Copyright © GrapeCity, Inc. All rights reserved.

```
null) {
   return this.C1Json(CollectionViewHelper.Edit(requestData, item => {
    var error = string.Empty;
    var success = true;
    try {
     var index = sourceData.FindIndex(u => u.Id == item.Id);
     sourceData.RemoveAt(index);
     sourceData.Insert(index, item);
    } catch (Exception e) {
     error = e.Message;
     success = false;
    }
    return new CollectionViewItemResult <DatesData> {
     Error = error,
     Success = success,
     Data = item
    };
   }, () => converter != null ? converter(sourceData) : sourceData));
  }

  public ActionResult Create(CollectionViewEditRequest <DatesData> requestData, List
<DatesData> sourceData, Func <IEnumerable <DatesData> , List <DatesData>> converter =
null) {
   return this.C1Json(CollectionViewHelper.Edit(requestData, item => {
    var error = string.Empty;
    var success = true;
    try {
     sourceData.Add(item);
     item.Id = sourceData.Max(u => u.Id) + 1;
    } catch (Exception e) {
     error = e.Message;
     success = false;
    }
    return new CollectionViewItemResult <DatesData> {
     Error = error,
     Success = success,
     Data = item
    };
   }, () => converter != null ? converter(sourceData) : sourceData));
  }

  public ActionResult Delete(CollectionViewEditRequest <DatesData> requestData, List
<DatesData> sourceData, Func <IEnumerable <DatesData> , List <DatesData>> converter =
null) {
   return this.C1Json(CollectionViewHelper.Edit(requestData, item => {
    var error = string.Empty;
    var success = true;
    try {
     var index = sourceData.FindIndex(u => u.Id == item.Id);
     sourceData.RemoveAt(index);
    } catch (Exception e) {
     error = e.Message;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
     success = false;
    }
    return new CollectionViewItemResult <DatesData> {
     Error = error,
     Success = success,
     Data = item
    };
   }, () => converter != null ? converter(sourceData) : sourceData));
  }
 }
}
```

JavaScriptファイルを追加します。

responseTextParsing関数と**requestDataStringfying**関数を定義するJavaScriptファイル（例：app.js）を追加する必要があります。

app.js

```
// DateTimeテキストの判別に使用されるRegExpオブジェクト。
var dateJsonRegx = /^\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d*)?(Z|[\+\-
]\d{2}:\d{2}|)$/;
function reponseTextParsing(sender, args) {
    var dateText = args.value;
    // これが有効なDateTimeテキストかどうかをチェックします。
    var matched = dateJsonRegx.exec(dateText);
    if (!matched) {
        return;
    }
    var timeZoneText = matched[2];
    var dateKind = getDateKind(timeZoneText);
    // UtcまたはLocal形式のDateオブジェクトの解析のみをカスタマイズします。
    if (dateKind == c1.mvc.DateKind.Unspecified) {
        return;
    }
    if (typeof dateText === 'string' && matched) {
        var index = dateText.indexOf(timeZoneText);
        // タイムゾーンのテキストを削除し、Dateオブジェクトを作成します。
        var date = new Date(dateText.substr(0, index));
        // 解析されたDateオブジェクトのdateKindを忘れずに設定します。
        // これがOnClientRequestDataStringifyingで使用されます。
        date.dateKind = dateKind;
        args.result = date;
        args.cancel = true;
    }
}

function getDateKind(timeZoneText) {
    if (!timeZoneText) {
        return c1.mvc.DateKind.Unspecified;
    }

    if (timeZoneText.toLowerCase() === 'z') {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        return c1.mvc.DateKind.Utc;
    }

    return c1.mvc.DateKind.Local;
}

function requestDataStringifying(sender, args) {
    if (args.value instanceof Date || args.parent[args.key] instanceof Date) {
        var date = args.value instanceof Date ? args.value : args.parent[args.key];
        // Utc形式のDateオブジェクトのシリアライズのみをカスタマイズします。
        if (!date.dateKind || date.dateKind == c1.mvc.DateKind.Unspecified) {
            return;
        }

        args.result = c1.mvc.Utils.formatNumber(date.getFullYear(), 4) + '-' +
                c1.mvc.Utils.formatNumber(date.getMonth() + 1, 2) + '-' +
                c1.mvc.Utils.formatNumber(date.getDate(), 2) + 'T' +
                c1.mvc.Utils.formatNumber(date.getHours(), 2) + ':' +
                c1.mvc.Utils.formatNumber(date.getMinutes(), 2) + ':' +
                c1.mvc.Utils.formatNumber(date.getSeconds(), 2) + '.' +
                c1.mvc.Utils.formatNumber(date.getMilliseconds(), 3)
                + (date.dateKind == c1.mvc.DateKind.Utc ? 'Z' :
getLocalTimeZoneText());
        args.cancel = true;
    }
}

function getLocalTimeZoneText() {
    var date = new Date();
    var timeoffset = date.getTimezoneOffset();
    var result = '';
    if (timeoffset > 0) {
        result += '-';
    } else {
        result += '+';
        timeoffset *= -1;
    }
    var hour = Math.floor(timeoffset / 60);
    result += formatNumber(hour, 2);
    result += ":";
    result += formatNumber(timeoffset - hour * 60, 2);
    return result;
}

function formatNumber(n, k) {
    // 少なくともk桁になるように整数を書式設定します。
    var text = n.toString();
    while (text.length < k) {
        text = '0' + text;
    }
    return text;
}
```


Copyright © GrapeCity, Inc. All rights reserved.

ビューの追加

コード内 - Index.cshtml

DatesData.cs

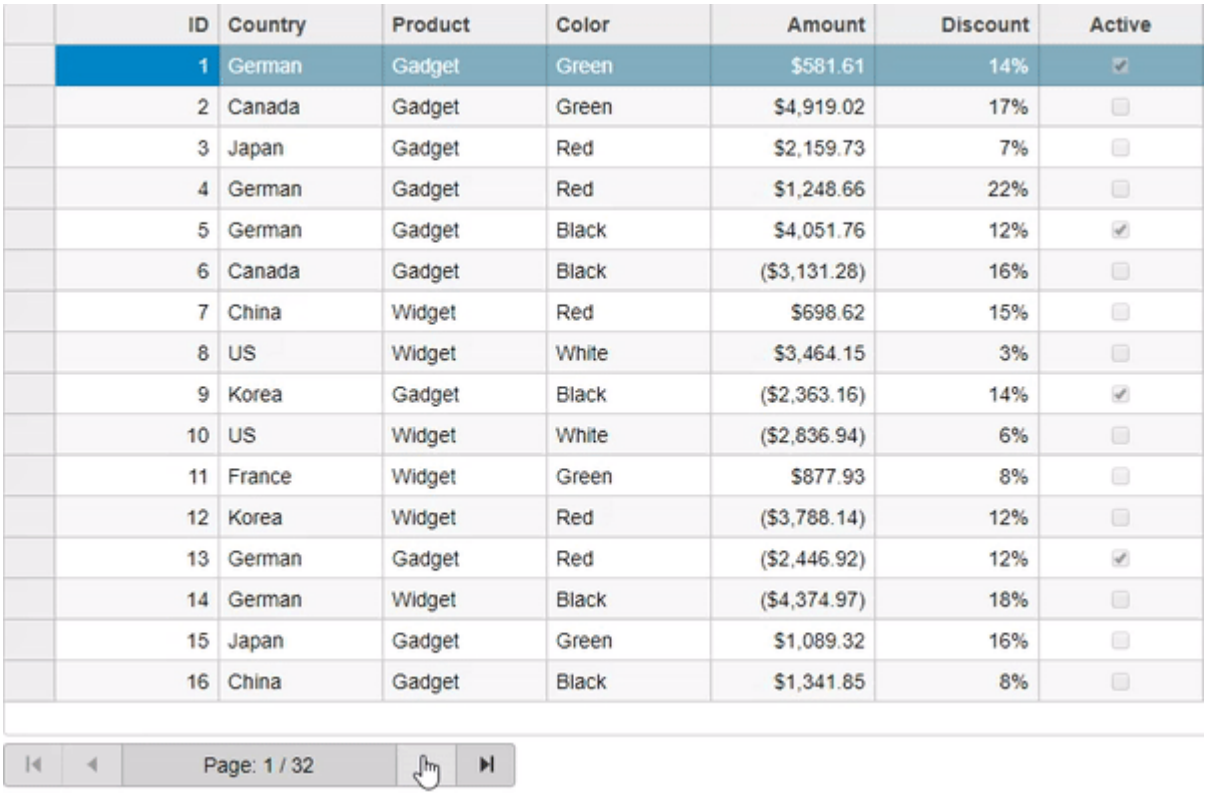
```
@(Html.C1().FlexGrid()
      .Id("customGrid")
      .AllowAddNew(true)
      .AllowDelete(true)
      .AutoGenerateColumns(false)
      .Bind(cvb => cvb.Bind(Url.Action("Custom_ReadDatesData"))
                  .Create(Url.Action("Custom_CreateDatesData"))
                  .Update(Url.Action("Custom_UpdateDatesData"))
                  .Delete(Url.Action("Custom_DeleteDatesData"))
                  .OnClientReponseTextParsing("reponseTextParsing")
                  .OnClientRequestDataStringifying("requestDataStringifying"))
      .Columns(columns =>
          {
            columns.Add(column =>
column.Binding("Id").IsReadOnly(true).Visible(false));
            columns.Add(column => column.Binding("UtcDateTime").Format("dd/MM/yyyy
HH:mm:ss").Width("*"));
            columns.Add(column => column.Binding("LocalDateTime").Format("dd/MM/yyyy
HH:mm:ss").Width("*"));
            columns.Add(column =>
column.Binding("UnspecifiedDateTime").Format("dd/MM/yyyy HH:mm:ss").Width("*"));
          })
       .Filterable(f => f.DefaultFilterType(FilterType.Both)))
```

ビューの移動

ASP.NET MVC Edition allows you to navigate through the records in CollectionView using the **CollectionViewNavigator** control. This control can be accessed via the **CollectionViewNavigator** class which allows you to navigate through the records in CollectionView. This class allows you to navigate the pages by setting the **ByPage** property to **true** and navigate the items by setting the **ByPage** property to **false**. In addition, the class also allows you to format the text displayed in CollectionViewNavigator control header using the **HeaderFormat** property.

The following GIF shows the CollectionViewNavigator control and how it is used.

Copyright © GrapeCity, Inc. All rights reserved.



ID	Country	Product	Color	Amount	Discount	Active
1	German	Gadget	Green	$581.61	14%	☑
2	Canada	Gadget	Green	$4,919.02	17%	☐
3	Japan	Gadget	Red	$2,159.73	7%	☐
4	German	Gadget	Red	$1,248.66	22%	☐
5	German	Gadget	Black	$4,051.76	12%	☑
6	Canada	Gadget	Black	($3,131.28)	16%	☐
7	China	Widget	Red	$698.62	15%	☐
8	US	Widget	White	$3,464.15	3%	☐
9	Korea	Gadget	Black	($2,363.16)	14%	☑
10	US	Widget	White	($2,836.94)	6%	☐
11	France	Widget	Green	$877.93	8%	☐
12	Korea	Widget	Red	($3,788.14)	12%	☐
13	German	Gadget	Red	($2,446.92)	12%	☑
14	German	Widget	Black	($4,374.97)	18%	☐
15	Japan	Gadget	Green	$1,089.32	16%	☐
16	China	Gadget	Black	$1,341.85	8%	☐

The following example shows how the CollectionViewNavigator control can be used for navigating the records by page. This code uses data from the Sale class which is bound to the FlexGrid control and available with the MvcExplorer sample.

Controller code

CVNavigationController copyCode

```
//FlexGridのデータソースを定義します。
public ActionResult Index(FormCollection collection)
{
    IValueProvider data = collection;
    var model = Sale.GetData(500);
    return View(model);
}
```

View code

Index copyCode

```
@using C1.Web.Mvc.Grid
@using CVNavigator.Models

@model IEnumerable<Sale>

@(Html.C1().CollectionViewService()
      .Id("CVService").Bind(Model)
      .InitialItemsCount(500)
      .PageSize(16))

@(Html.C1().FlexGrid<Sale>()
      .Id("FlexGridCVN")
      .ItemsSourceId("CVService")
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .AllowSorting(true)
        .SelectionMode(SelectionMode.Row)
        .CssClass("grid")
        .IsReadOnly(true)
        .Height(500)
        .AutoGenerateColumns(false)
        .Columns(bl =>
          {
             bl.Add(cb => cb.Binding("ID"));
             bl.Add(cb => cb.Binding("Country"));
             bl.Add(cb => cb.Binding("Product"));
             bl.Add(cb => cb.Binding("Color"));
             bl.Add(cb => cb.Binding("Amount").Format("c"));
             bl.Add(cb => cb.Binding("Discount").Format("p0"));
             bl.Add(cb => cb.Binding("Active"));
          })
)


@(Html.C1().CollectionViewNavigator()
        .Id("CVNavigatorToGridCV")
        .ItemsSourceId("CVService")
        .ByPage(true)
        .HeaderFormat("Page: {currentPage} / {pageCount}") //'{currentItem}',
'{itemCount}'
        .RepeatButtons(true)
        .CssStyle("background-color", "lightgray")
)
```

検証

CollectionView provides validation support to show custom or multiple errors in the FleGrid row header cells. The CollectionView provides **GetError** property to offer validation support. When you use the **CollectionView.GetError** method for validation, the FlexGrid shows errors in data cells and also in the row header cells. All the errors in the entire data item are included in the error message, by default. This message makes the information more useful and our data grid more user-friendly.

By default, the row header cells shows all the errors in the item. To customize that behavior, return the error message you want to show for the row header when the **GetError** method is called without a specific property (prop == null). You can also perform your own validation using a custom function when items are edited.

Copyright © GrapeCity, Inc. All rights reserved.



The following sample uses the GetError function to only allow certain countries and positive numbers to be committed. The example uses Sale.cs model added in the Custom Editors topic.

Controller Code

ValidationController.cs copyCode

```
public class ValidationController : Controller
{
    private static List<Sale> ValidationSales = Sale.GetValidationData().ToList();
    // GET: Validation
    public ActionResult Index()
    {
        return View(ValidationSales);
    }

    public ActionResult GridUpdateSale([C1JsonRequest]
CollectionViewEditRequest<Sale> requestData)
    {
        return this.C1Json(CollectionViewHelper.Edit<Sale>(requestData, item =>
        {
            string error = string.Empty;
            bool success = true;
            try
            {
                var resultItem = ValidationSales.Find(s => s.ID == item.ID);
                var index = ValidationSales.IndexOf(resultItem);
                ValidationSales.Remove(resultItem);
                ValidationSales.Insert(index, item);
            }
            catch (Exception e)
            {
                error = e.Message;
                success = false;
            }
            return new CollectionViewItemResult<Sale>
            {
                Error = error,
                Success = success,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                Data = item
            };
        }, () => ValidationSales));
    }

    public ActionResult GridCreateSale([C1JsonRequest]
CollectionViewEditRequest<Sale> requestData)
    {
        return this.C1Json(CollectionViewHelper.Edit<Sale>(requestData, item =>
        {
            string error = string.Empty;
            bool success = true;
            try
            {
                ValidationSales.Add(item);
                item.ID = ValidationSales.Max(s => s.ID) + 1;
            }
            catch (Exception e)
            {
                error = e.Message;
                success = false;
            }
            return new CollectionViewItemResult<Sale>
            {
                Error = error,
                Success = success,
                Data = item
            };
        }, () => ValidationSales));
    }

    public ActionResult GridDeleteSale([C1JsonRequest]
CollectionViewEditRequest<Sale> requestData)
    {
        return this.C1Json(CollectionViewHelper.Edit<Sale>(requestData, item =>
        {
            string error = string.Empty;
            bool success = true;
            try
            {
                var resultItem = ValidationSales.Find(u => u.ID == item.ID);
                ValidationSales.Remove(resultItem);
            }
            catch (Exception e)
            {
                error = e.Message;
                success = false;
            }
            return new CollectionViewItemResult<Sale>
            {
                Error = error,
                Success = success,
                Data = item
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            };
        }, () => ValidationSales));
    }
}
```

View Code

Index.cshtml copyCode

```
@model List<Sale>

<script type="text/javascript">
    var validCountries = ["US", "UK", "Canada", "Japan", "China", "France", "German",
"Italy", "Korea", "Australia"];
    var validProducts = ["Widget", "Gadget", "Doohickey"];
    function getError(item, property) {
        switch (property) {
            case 'Country':
                return validCountries.indexOf(item[property]) < 0
                    ? '@Html.Raw("Invalid Country, Please use the country in the
list[US, UK, Canada, Japan, China, France, German, Italy, Korea, Australia]")'
                    : null;
            case 'Product':
                return validProducts.indexOf(item[property]) < 0
                    ? '@Html.Raw("Invalid Product, Please use the product in the list
[Widget, Gadget, Doohickey]")'
                    : null;
            case 'Amount2':
                return item[property] < 0 || item[property] >= 5000
                    ? '@Html.Raw("The Amount cannot be less than 0 or greater than or
equal to 5000!")'
                    : null;
            case 'Active':
                return item[property] && item['Country'] &&
item['Country'].match(/US|UK/)
                    ? '@Html.Raw("Active items are not allowed in the US or UK!")'
                    : null;
        }
        return null;
    }
</script>

<style>
    .text-style-errortip {
        background-color: lightyellow;
        color: red;
        font-style: oblique;
        text-shadow: 2px 2px 5px orangered;
    }
</style>

@(Html.C1().CollectionViewService().Id("validationCV")
    .GetError("getError")
```


Copyright © GrapeCity, Inc. All rights reserved.

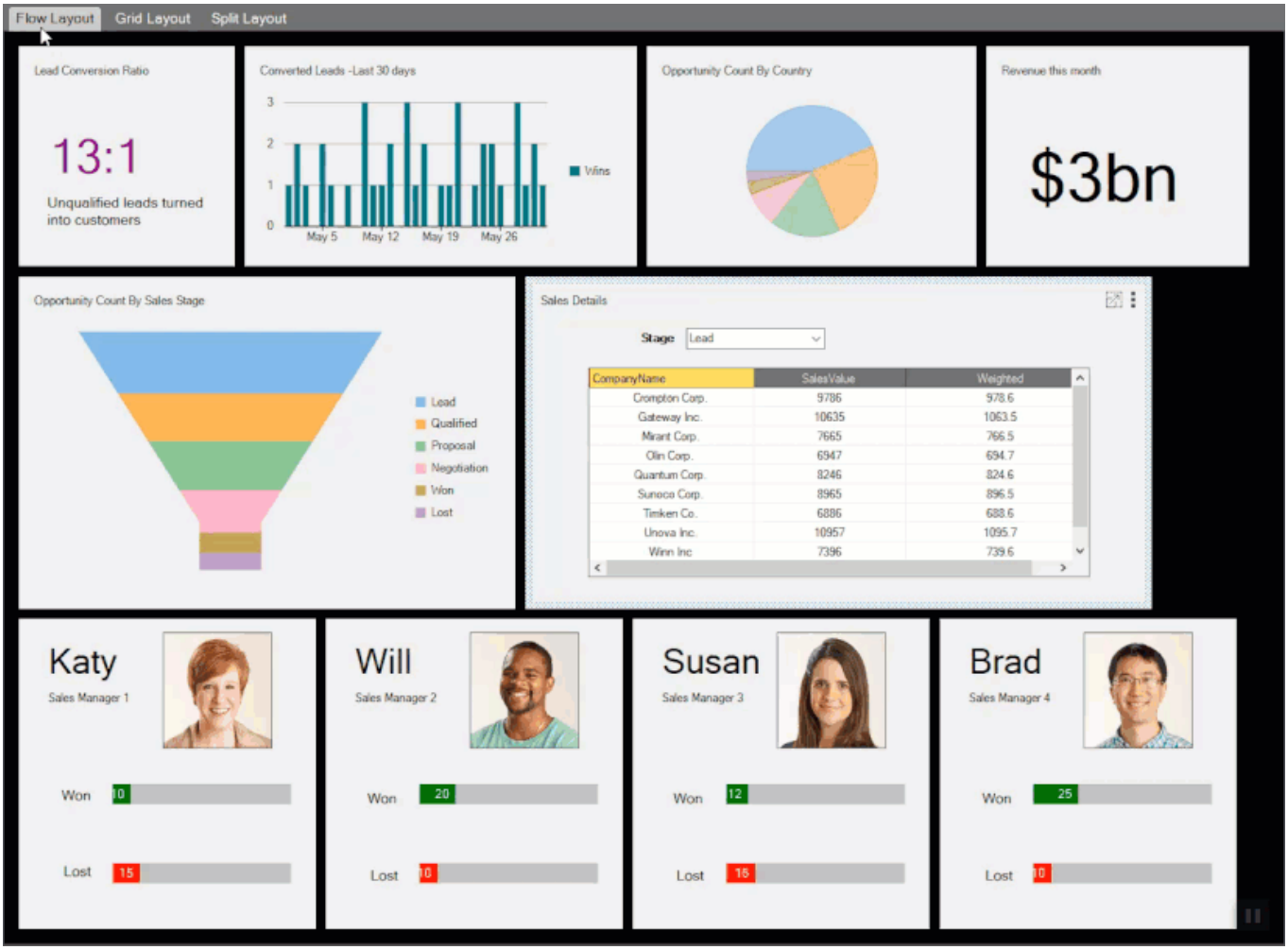
```
    .Bind(Model)
    .Update(Url.Action("GridUpdateSale"))
    .Create(Url.Action("GridCreateSale"))
    .Delete(Url.Action("GridDeleteSale"))
)
@(Html.C1().FlexGrid<Sale>()
    .Id("grid")
    .AutoGenerateColumns(false)
    .Columns(columns => columns
        .Add(c => c.Binding("ID").IsReadOnly(true))
        .Add(c => c.Binding("Product"))
        .Add(c => c.Binding("Country"))
        .Add(c => c.Binding("Color"))
        .Add(c => c.Binding("Amount2"))
        .Add(c => c.Binding("Discount"))
        .Add(c => c.Binding("Active"))
    ).ItemsSourceId("validationCV")
    .AllowAddNew(true)
    .AllowDelete(true)
    .CssStyle("height", "400px")
    .ErrorTip(tip =>
    {
        tip.CssClass("text-style-errortip");
        tip.Position(PopupPosition.Above);
    })
    .ValidateEdits(true)
)
```

DashboardLayout

DashboardLayout is a layout control that allows you to create dynamic dashboards for interactive data visualization. It allows you to organize and present data in a consolidated form with the help of images, grids, charts, maps, etc in different layouts. This makes it easy for you to monitor the presented information.

The DashboardLayout control acts as a container which lets you dynamically place controls within tiles also called child containers. These child containers can be arranged in four different types of layouts supported by the DashboardLayout control i.e. Flow, AutoGrid, ManualGrid and Split. The appropriate layout is attached to the DashboardLayout control to achieve the specific layout. The control lets you resize and rearrange these child containers at runtime to create an ideal workspace.

Copyright © GrapeCity, Inc. All rights reserved.



主な特長

DashboardLayout provides many different features that enable the developers to build intuitive and professional-looking dashboards. The main features for DashboardLayout are as follows:

- **Use different layouts**

  DashboardLayout provides four types of layouts, namely Flow, AutoGrid, ManualGrid and Split. These layouts specify the arrangement of the tiles in different ways on the DashboardLayout control.

- **Tile headers**

  DashboardLayout control provides a header for each tile, which displays a caption and provides options to maximize/restore or show/hide the tiles.

- **Drag and drop tiles**

  DashboardLayout allows you to drag and drop a tile using two methods; crosshair icon and tool-icon. The crosshair appears when you hover the mouse over the tile header. As an alternative, you can also use the tool-icon to perform the drag drop operation by holding the cursor on the tool-icon.

- **Maximize/Restore Tiles**

  DashboardLayout control's header provides a maximize/restore icon, which is used to maximize or restore the

Copyright © GrapeCity, Inc. All rights reserved.

display state of the tile.

- **Perform multiple functions with ToolIcon**

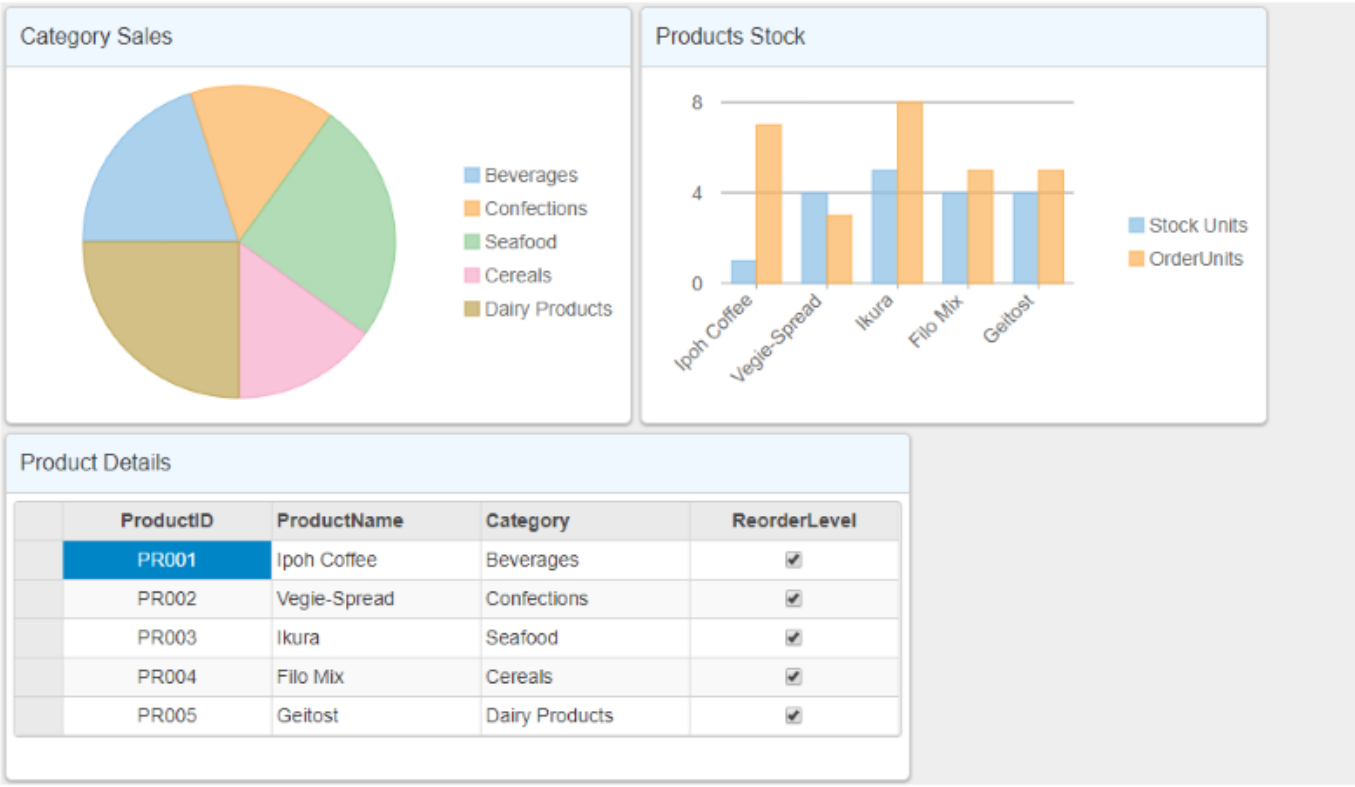
  DashboardLayout provides a multi functional tool called Tool-icon, which appears as an icon in the header or on the surface of the tile on hovering the mouse over it. It can be used to drag and drop the tiles. Moreover, tool-icon provides a context menu with options to hide a tile or show all the tiles.

- **Serialization/Deserialization**

  DashboardLayout offers XML serialization by providing the functionality to save the layout properties of the control to an XML file or stream and load them from another XML file or stream.

クイックスタート

The quick start guides you through the steps of adding DashboardLayout control to your MVC web application for creating a simple dashboard application. Follow the steps given below to get started:



Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Add Data to the Application

1. Add a new class to the **Models** folder (Name: `ProductDashboardData.cs`). For more information on how to add a new model, see Adding Controls.
2. Add the following code to `ProductDashboardData.cs` model. We are using **`ProductDashboardData`** class to represent data.

Copyright © GrapeCity, Inc. All rights reserved.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Dashboard_Quickstart.Models
{
    public class ProductDashboardData
    {
        private IEnumerable<ProductData> _productDetails = null;
        public IEnumerable<ProductData> ProductDetails
        {
            get
            {
                if (_productDetails == null)
                {
                    _productDetails = GetProductData();
                }
                return _productDetails;
            }
        }
        public IEnumerable<ProductData> GetProductData()
        {
            var rand = new Random(0);
            var productID = new[] { "PR001", "PR002", "PR003", "PR004", "PR005"
};
            var products = new[] { "Ipoh Coffee", "Vegie-Spread", "Ikura", "Filo
Mix", "Geitost" };
            var categories = new[] { "Beverages", "Confections", "Seafood",
"Cereals", "Dairy Products" };

            var list = products.Select((p, i) =>
            {
                int stockunits = rand.Next(1, 6);
                int orderunits = rand.Next(1, 9);
                int sales = rand.Next(1, 6);
                return new ProductData { ProductID = productID[i], ProductName =
p, Category = categories[i], UnitsInStock = stockunits, UnitsOnOrder =
orderunits, Sales = sales, ReorderLevel = true };
            });

            return list;
        }
    }
    public class ProductData
    {
        public string ProductID { get; set; }
        public string ProductName { get; set; }
        public string Category { get; set; }
        public int UnitsInStock { get; set; }
        public int UnitsOnOrder { get; set; }
        public int Sales { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public bool ReorderLevel { get; set; }
    }
}
```

Back to Top

Add a DashboardLayout Control

Steps to add a DashboardLayout control to the application, are as follows:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template, and then click **Add**.
    2. Set name of the controller (for example: `DashboardLayoutController`).
    3. Click **Add**.
4. Include the following references as shown below.

C#

```
using <ApplicationName>.Models;
```

5. Replace the **Index()** method with the following method.

DashboardLayoutController.cs

```
public ActionResult Index()
        {
            ProductDashboardData data = new ProductDashboardData();
            return View(data.ProductDetails);
        }
```

Add a View for the Controller

In the view, we create an instance of **DashboardLayout** and attach the flow layout to the control by using the **AttachFlowLayout** method provided by the **DashboardLayoutBuilder** class.

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `DashboardLayoutController.`
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller, and then copy the following code and paste it inside **Index.cshtml**.

Index.cshtml

```
@using ApplicationName.Models
@model IEnumerable<ProductData>

<style>
    .wj-dashboard .wj-flexchart {
        margin: 0px;
        padding: 4px;
        border: none;
        height: 240px;
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
</style>
<br />
<div>
    @(Html.C1().FlexPie<ProductData>().Id("CategorySales")
      .Bind("Category", "Sales", Model)
    )

    @(Html.C1().FlexChart().Id("ProductsStock")
     .Bind("ProductName", Model)
     .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsInStock")
        .Name("Stock Units");
     })
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsOnOrder")
        .Name("OrderUnits");
     })
    )

    @(Html.C1().FlexGrid<ProductData>().Id("ProductDetails")
     .AutoGenerateColumns(false)
     .Bind(Model)
     .Columns(bl =>
     {
         bl.Add(cb => cb.Binding("ProductID").Width("150").Align("Center"));
         bl.Add(cb => cb.Binding("ProductName").Width("150"));
         bl.Add(cb => cb.Binding("Category").Width("150"));
         bl.Add(cb => cb.Binding("ReorderLevel").Width("150"));
     })
    )
</div>

@(Html.C1().DashboardLayout().Id("SampleDashboard")
        .AttachFlowLayout(flb => flb.Direction(FlowDirection.LeftToRight)
        .Items(isb =>
        {
            isb.Add(ftb => ftb.HeaderText("Category Sales")
                .Content("#CategorySales").Width(450).Height(300));
            isb.Add(ftb => ftb.HeaderText("Products Stock")
                .Content("#ProductsStock").Width(450).Height(300));
            isb.Add(ftb => ftb.HeaderText("Product Details")
                .Content("#ProductDetails").Width(650).Height(250));
        })))
```

Build and Run the Project

1. Click **Build | Build Solution** to build the project.

Copyright © GrapeCity, Inc. All rights reserved.

2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/**DashboardLayout/Index**) in the address bar of the browser to see the view.

Back to Top

DashboardLayout の使用

This section comprises all the features and auxiliary functionality offered by the DashboardLayout control.

スキャフォールディング
Learn about how to work with DashboardLayout control using Scaffolding.

レイアウト
Learn about different layouts supported by DashboardLayout.

ヘッダー
Learn about the tile header.

ドラッグアンドドロップ
Learn how drag and drop works in DashboardLayout.

ツールアイコン
Learn about the tool-icon and how to set its location.

サイズ変更
Learn about the resizing behavior of child containers in different layouts.

XMLのシリアル化
Learn how DashboardLayout supports serialization.

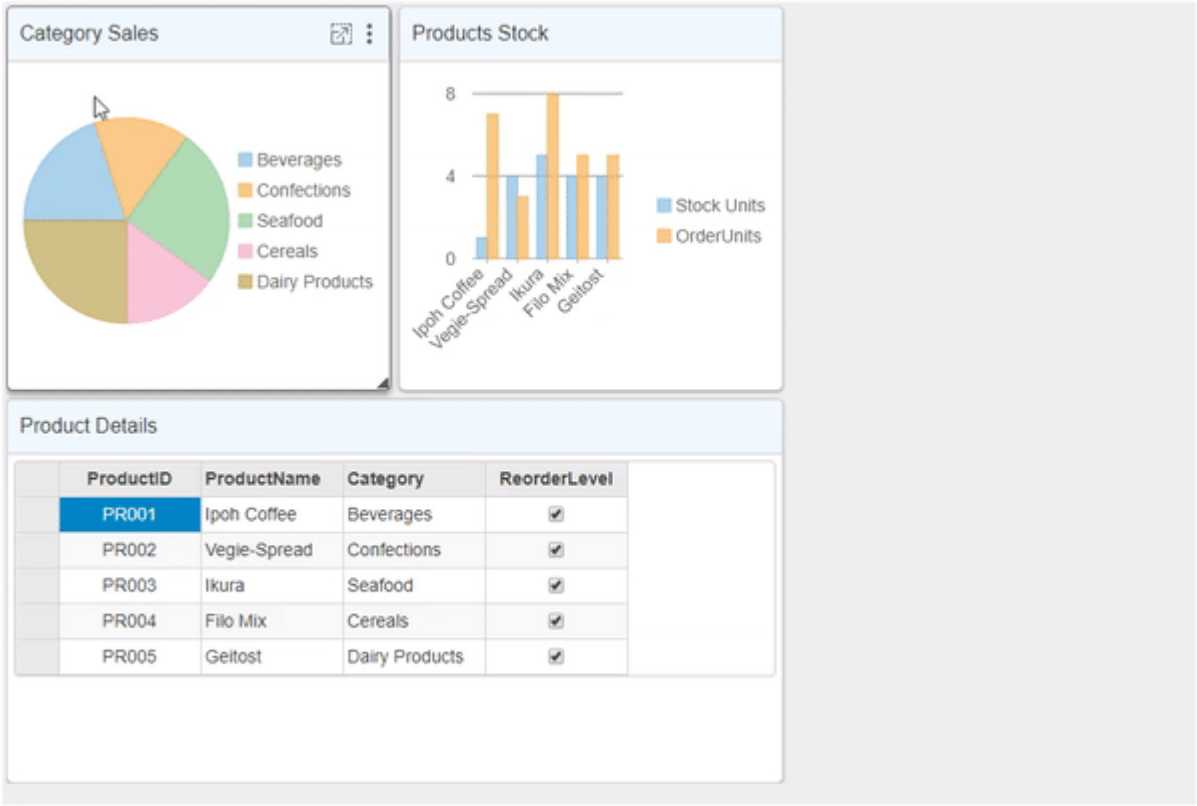
ドラッグアンドドロップ

DashboardLayout supports drag drop operation of the tiles. On performing drag drop operation, the manner in which tiles reposition themselves differ for every layout. You can drag and drop a tile using two methods, using crosshair icon and tool-icon. The crosshair appears when you hover the mouse over the tile header. This tool lets you drag and drop a tile to another position on the DashboardLayout control. As an alternative, you can use the tool-icon to perform the drag drop operation by holding the cursor on the tool-icon.

Drag and Drop in Flow Layout

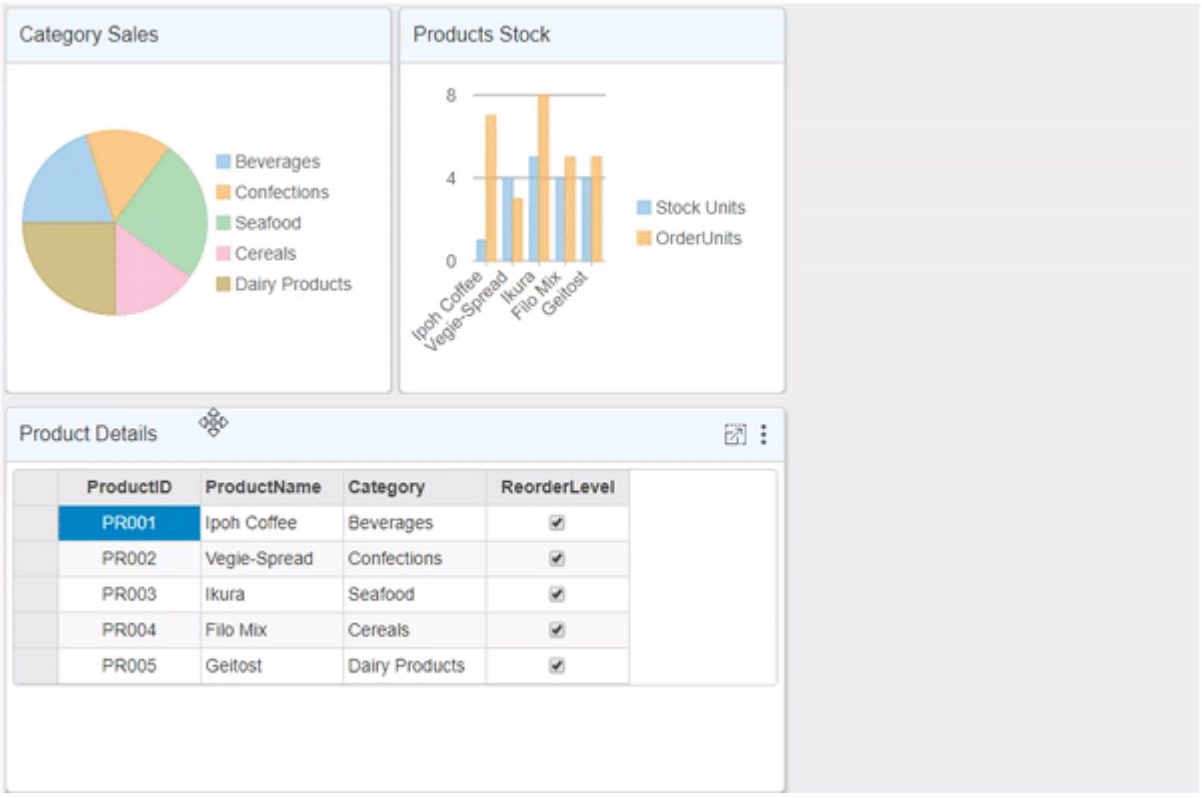
In flow layout, when a tile is dragged, all the remaining tiles reposition themselves in the flow direction set for the layout.

Copyright © GrapeCity, Inc. All rights reserved.



Drag and Drop in Auto Grid Layout

In auto grid layout, when you drag a tile, the dragged tile replaces its position with another tile in its cell.

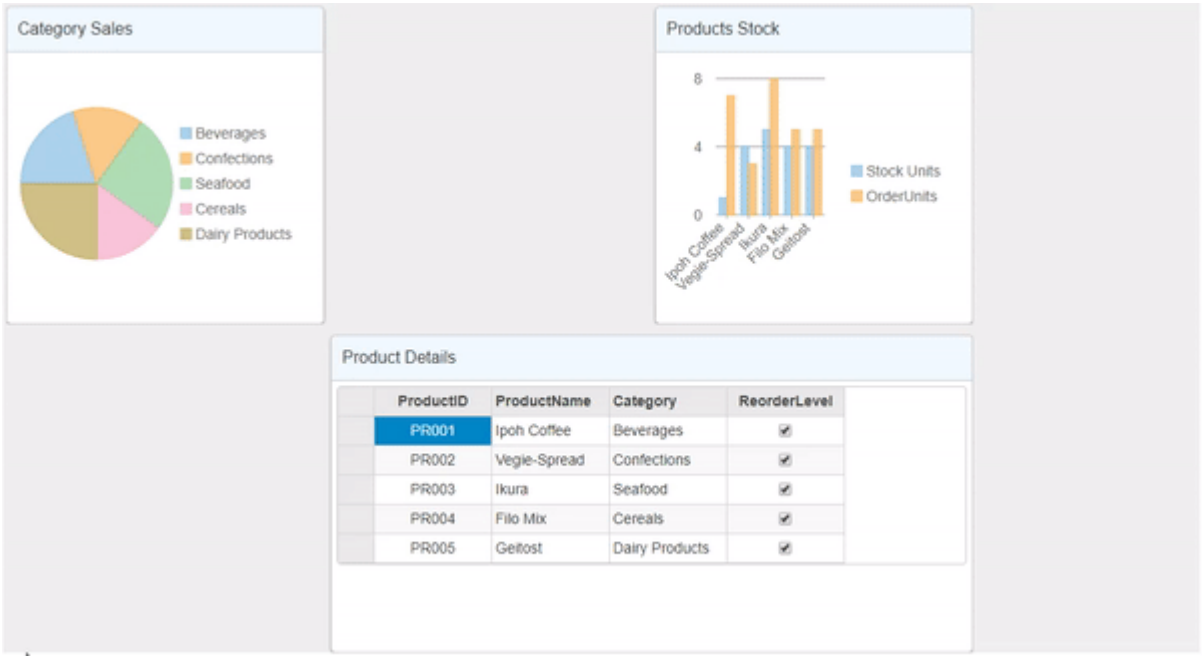


Drag and Drop in Manual Grid Layout

In manual grid layout, when you drag a tile, the dragged tile replaces its position with another tile in the same row or

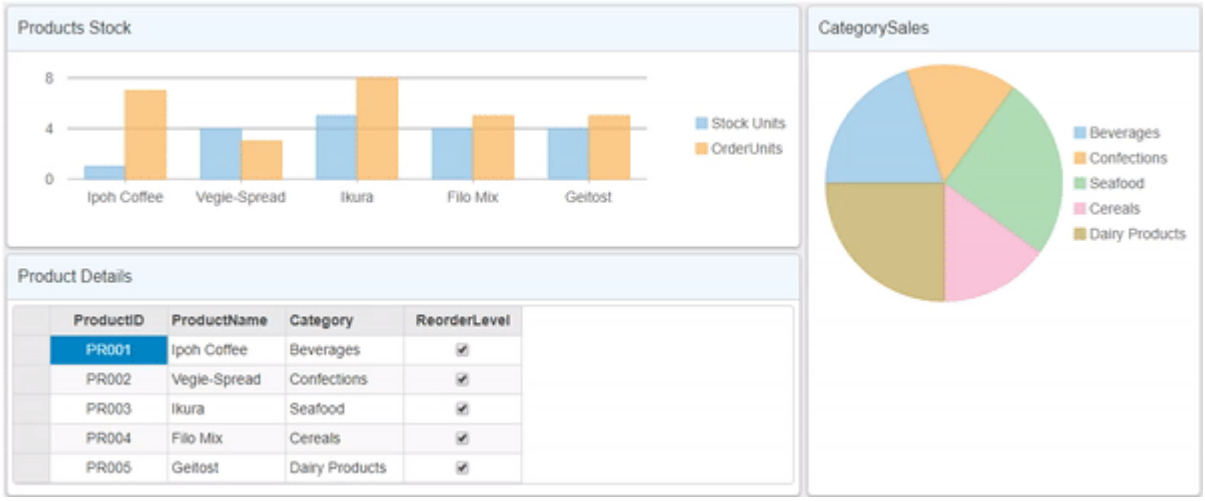
Copyright © GrapeCity, Inc. All rights reserved.

column.



Drag and Drop in Split Layout

In split layout, when you drag a tile, it shifts to the other panel in the mouse navigation path and all the other tiles change their positions as well.



By default, in the DashboardLayout control drag and drop operation is enabled. However, in case you want to disable drag and operations, you can make use of **AllowDrag** property provided by the **DashboardLayout** class.

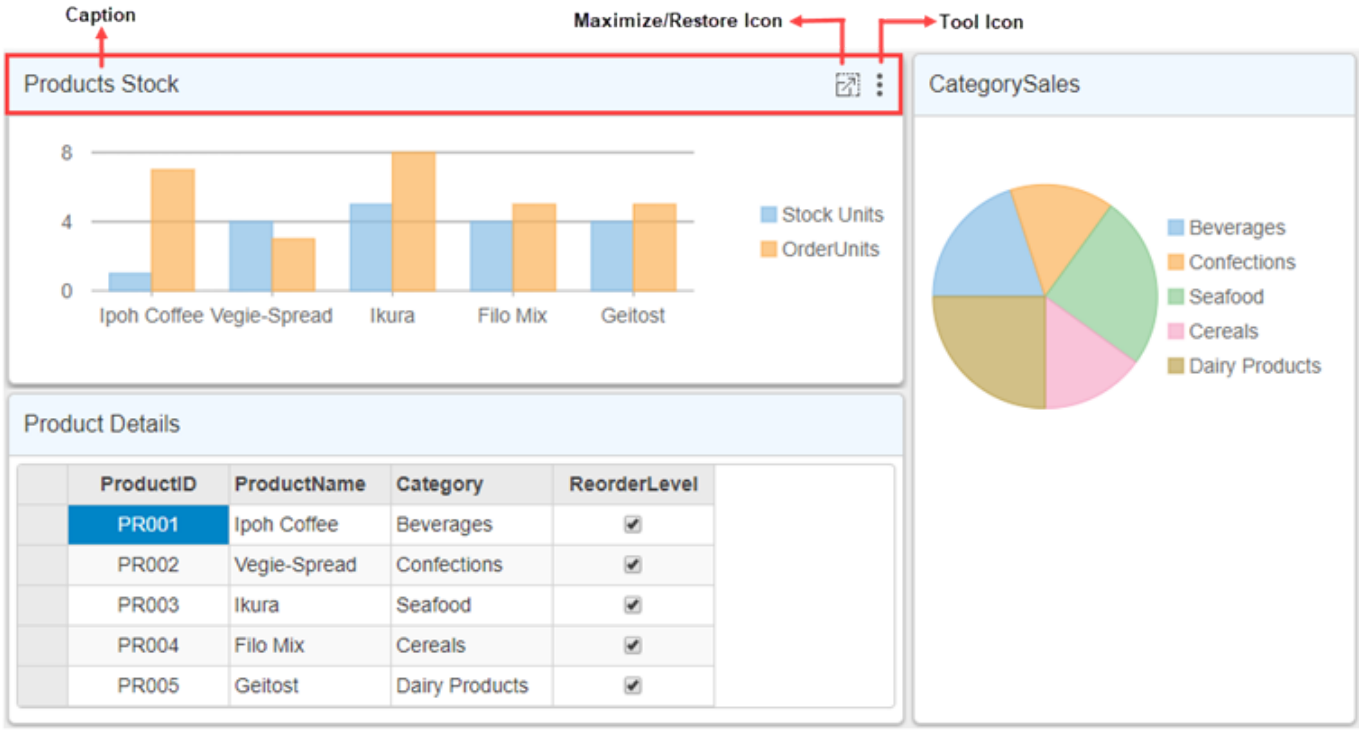
Razor

```
//タイルのドラッグアンドドロップ操作を無効にします
.AllowDrag(false)
```

ヘッダー

DashboardLayout allows you to display a caption for each tile with the help of **HeaderText** method of the **TileBuilder** class. This method sets caption for the tile. The header consists of caption, a maximize/restore icon and tool-icon.

Copyright © GrapeCity, Inc. All rights reserved.



The following code snippet shows how to set header text.

Razor

```
.Children(hgcb => hgcb.AddTile().HeaderText("Products Stock"))
```

レイアウト

This section contains information about

フロー レイアウト
: Learn how to attach flow layout in the DashboardLayout control.

自動グリッドレイアウト
: Learn how to attach auto grid layout in the DashboardLayout control.

手動グリッドレイアウト
: Learn how to attach manual grid layout in the DashboardLayout control.

スプリットレイアウト
: Learn how to attach split layout in the DashboardLayout control.

フロー レイアウト

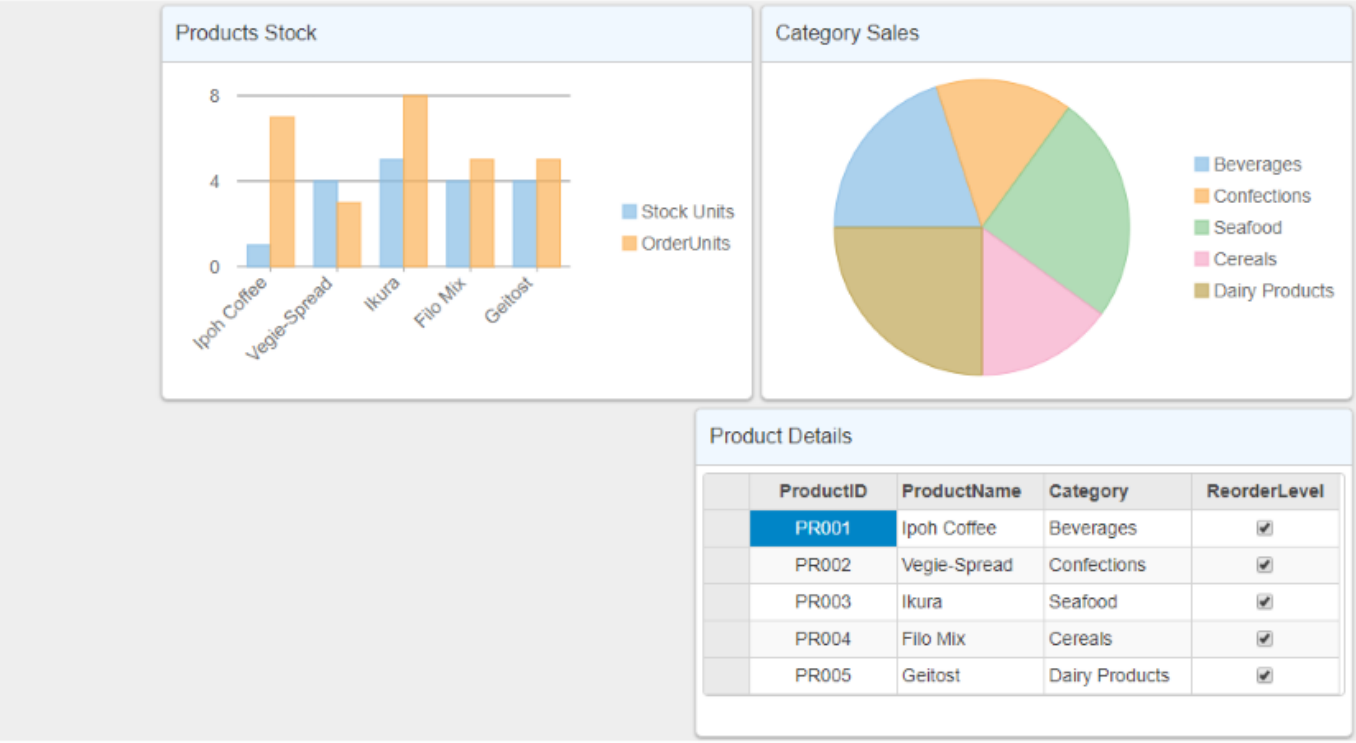
DashboardLayout allows you to set the flow layout using its **AttachFlowLayout** method of the **DashboardLayoutBuilder** class. This layout arranges the child containers in a horizontal or vertical direction depending on the flow direction set for the layout control. The flow direction of the contents can be specified by using the **FlowDirection** property which accepts the values from **FlowDirection** enumeration. The possible values for this enumeration are as follows:

- LeftToRight
- TopDown
- RightToLeft
- BottomUp

Copyright © GrapeCity, Inc. All rights reserved.

By default, the **FlowDirection** property is set to **LeftToRight**. Therefore, when the first control is dragged on the flow layout, the newly created child container positions itself on the upper left corner. When you add more controls to the layout, child containers get created and placed in left to right direction maintaining the flow direction. As soon as the width of DashboardLayout has exhausted, the child containers automatically get wrapped and shift to the next row. Moreover, DashboardLayout provides **FlowTile** class that represents the tiles in the flow layout.

The following image shows controls arranged in flow layout with FlowDirection property set to **RightToLeft**. This example uses the sample created in the QuickStart section.



The following code example demonstrates how to set flow layout for the DashboardLayout control.

Razor

```
@using ApplicationName.Models
@model IEnumerable<ProductData>

<style>
    .wj-dashboard .wj-flexchart {
        margin: 0px;
        padding: 4px;
        border: none;
        height: 240px;
    }
</style>

<div>
    @(Html.C1().FlexPie<ProductData>().Id("CategorySales")
      .Bind("Category", "Sales", Model)
    )

    @(Html.C1().FlexChart().Id("ProductsStock")
     .Bind("ProductName", Model)
     .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
      .Series(sers =>
      {
          sers.Add()
         .Binding("UnitsInStock")
         .Name("Stock Units");
      })
      .Series(sers =>
      {
          sers.Add()
         .Binding("UnitsOnOrder")
         .Name("OrderUnits");
      })
     )

     @(Html.C1().FlexGrid<ProductData>().Id("ProductDetails")
      .AutoGenerateColumns(false)
      .Bind(Model)
      .Columns(bl =>
      {
          bl.Add(cb => cb.Binding("ProductID").Align("Center"));
          bl.Add(cb => cb.Binding("ProductName"));
          bl.Add(cb => cb.Binding("Category"));
          bl.Add(cb => cb.Binding("ReorderLevel"));
      })
     )
</div>
<br />
@(Html.C1().DashboardLayout().Id("SampleDashboard")
         .AttachFlowLayout(flb => flb.Direction(FlowDirection.RightToLeft)
         .Items(isb =>
         {
             isb.Add(ftb => ftb.HeaderText("Category Sales")
                 .Content("#CategorySales").Width(450).Height(300));
             isb.Add(ftb => ftb.HeaderText("Products Stock")
                 .Content("#ProductsStock").Width(450).Height(300));
             isb.Add(ftb => ftb.HeaderText("Product Details")
                 .Content("#ProductDetails").Width(500).Height(250));
         }))
)
```

自動グリッドレイアウト

Auto grid layout, as the name suggests, arranges the tiles in the tabular form automatically. An auto grid layout consists of groups which are rendered in the direction specified by the **Orientation** property. The property also defines the direction in which each group expands. You can also define the maximum number of rows or columns that each group can have in horizontal or vertical orientation respectively using **MaxRowsOrCols** property.

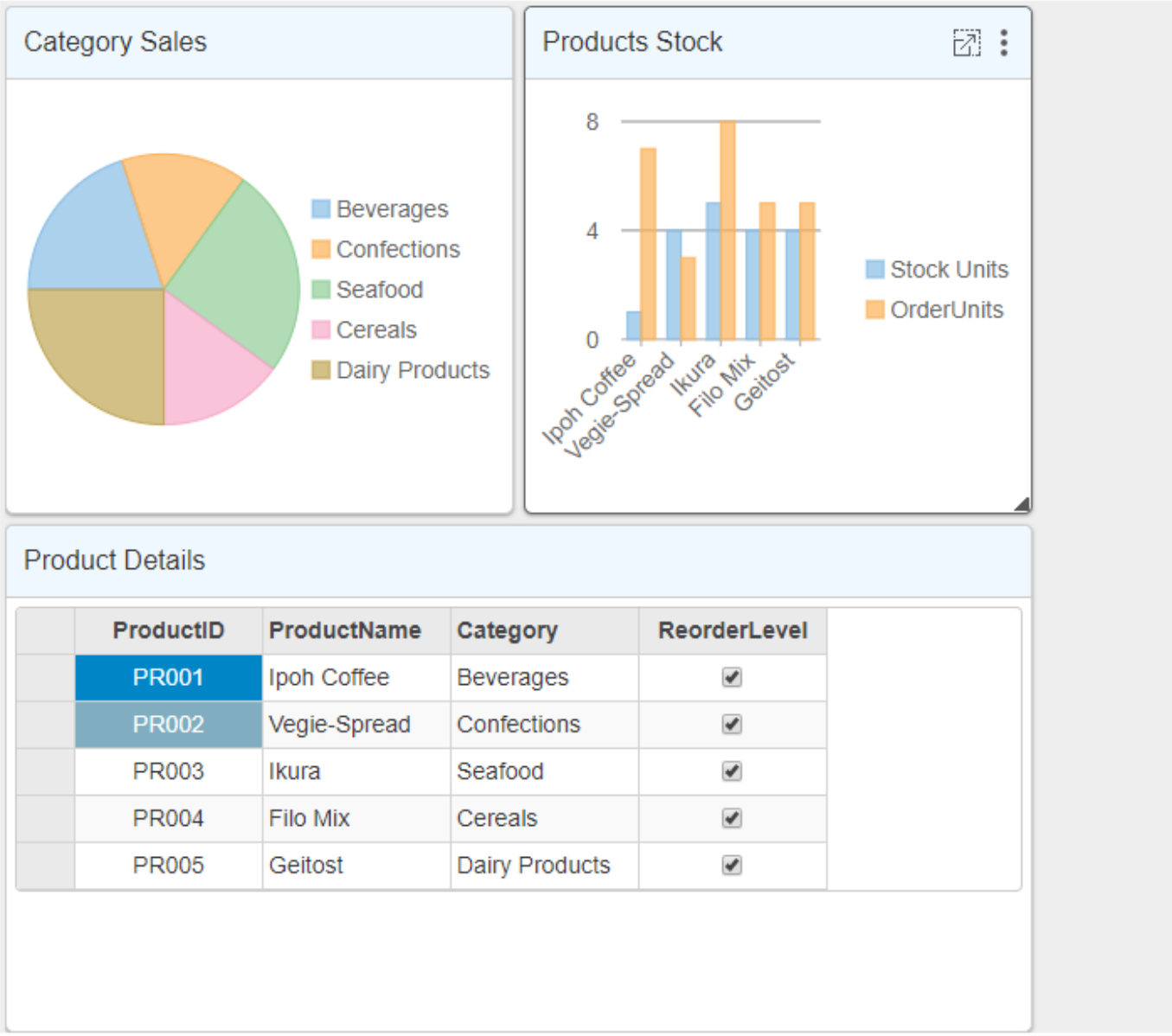
For instance, when orientation is set to horizontal, tiles are added row-wise until the maximum row count is reached. Once that limit is reached, the layout starts expanding horizontally by adding new columns one after the other. There is no limit to the number of columns that can be added in the horizontal orientation.

The auto grid layout also supports cell merging by spanning rows or columns through **RowSpan** and **ColumnSpan** properties of the tile available in the designer.

Copyright © GrapeCity, Inc. All rights reserved.

You can set the auto grid layout using the **AttachAutoGridLayout** method provided by the **DashboardLayoutBuilder** class. The layout automatically positions the tiles in the DashboardLayout control. Each cell in the table can contain multiple controls, and these controls can be grouped together with the help of the **AutoGridGroup** class. Also, it provides **AutoGridTile** class that represents the tiles in the auto grid layout.

The following image shows how DashboardLayout control appears after adding the auto grid layout.



The following code example demonstrates how to set auto grid layout for the DashboardLayout control. This example uses the sample created in the QuickStart section.

Razor

```
@using ApplicationName.Models
@model IEnumerable<ProductData>

<style>
    .wj-dashboard .wj-flexchart {
        margin: 0px;
        padding: 4px;
        border: none;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        height: 240px;
    }
</style>

<div id="DemoName">
    <p style="font-size:large;">Product Dashboard Demo</p>

    @(Html.C1().FlexPie<ProductData>().Id("CategorySales")
      .Bind("Category", "Sales", Model))

    @(Html.C1().FlexChart().Id("ProductsStock")
     .Bind("ProductName", Model)
     .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsInStock")
        .Name("Stock Units");
     })
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsOnOrder")
        .Name("OrderUnits");
     }))

    @(Html.C1().FlexGrid<ProductData>().Id("ProductDetails")
     .AutoGenerateColumns(false)
     .Bind(Model)
     .Columns(bl =>
     {
         bl.Add(cb => cb.Binding("ProductID").Align("Center"));
         bl.Add(cb => cb.Binding("ProductName"));
         bl.Add(cb => cb.Binding("Category"));
         bl.Add(cb => cb.Binding("ReorderLevel"));
     }))
</div>

<h2>Auto Grid Layout</h2>
<br />
@(Html.C1().DashboardLayout()
    .AttachAutoGridLayout(mglb => mglb.Orientation(LayoutOrientation.Vertical)
        .MaxRowsOrColumns(3)
        .CellSize(303)
        .Items(isb =>
        {
            isb.Add().Children(cb =>
            {
                cb.Add().HeaderText("Category Sales")
                    .Content("#CategorySales")
                    .RowSpan(1).ColumnSpan(1);
                cb.Add().HeaderText("Products Stock")
                    .Content("#ProductsStock")
```


Copyright © GrapeCity, Inc. All rights reserved.

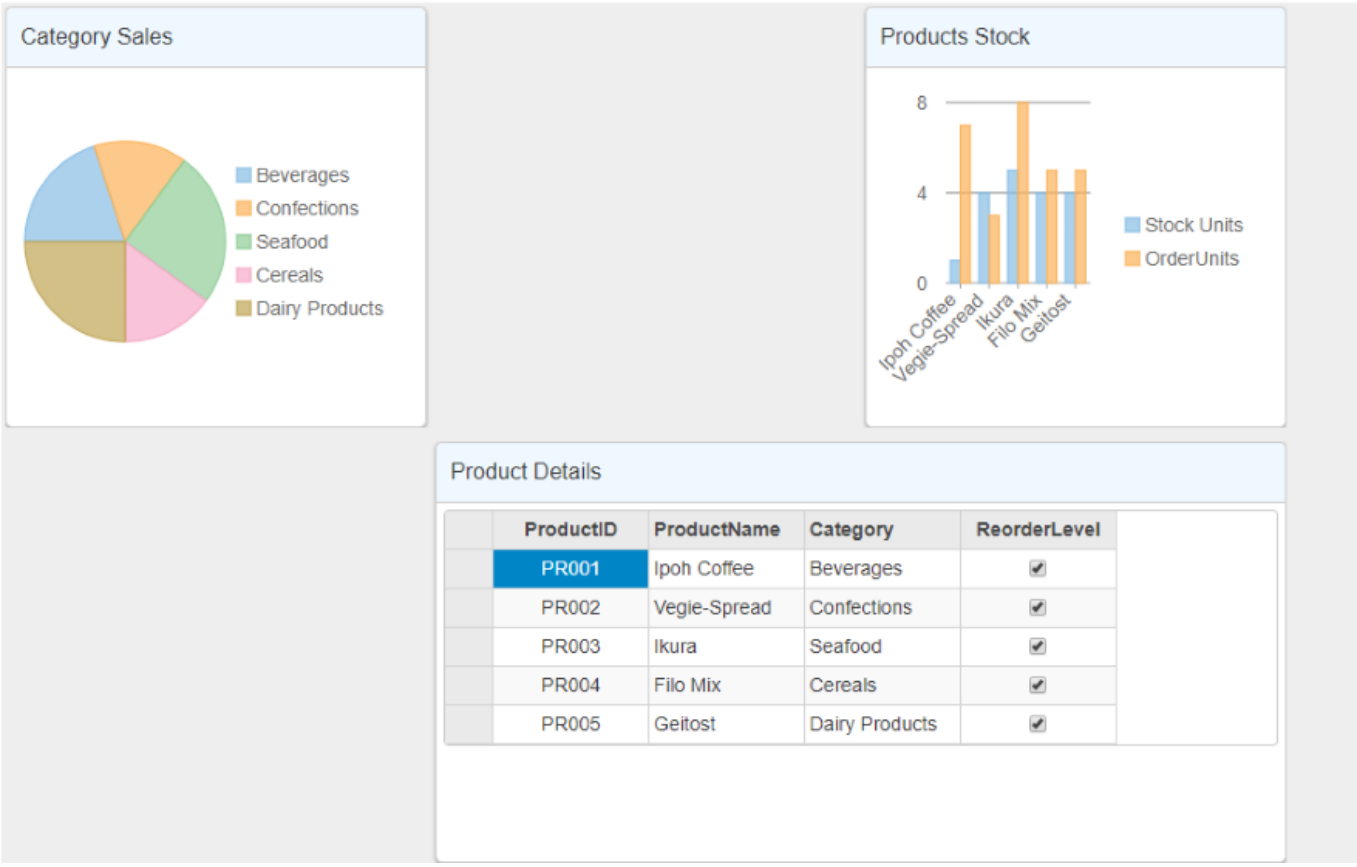
```
                .RowSpan(1).ColumnSpan(1);
        });
        isb.Add().Children(cb =>
        {
            cb.Add().HeaderText("Product Details")
                .Content("#ProductDetails")
                .RowSpan(1).ColumnSpan(2);
        });
    }))
```

手動グリッドレイアウト

Manual grid layout, as the name suggests, arranges the tiles in the tabular form in the specified manner. The layout is similar to auto grid layout except the fact that you can specify the row and column numbers where a particular tile should be positioned.

You can set the manual grid layout using the **AttachManualGridLayout** method provided by the **DashboardLayoutBuilder** class. Position of the tiles in the manual grid layout is specified using the **Row** and **Column** properties of the **ManualGridTileBuilder** class. Each cell in the manual grid layout can contain multiple controls, and these controls can be grouped together with the help of the **ManualGridGroup** class. Also, it provides the **ManualGridTile** class that represents the tiles in the manual grid layout.

The following image shows how DashboardLayout control appears after adding the manual grid layout. This example uses the sample created in the QuickStart section.



The following code example demonstrates how to set manual grid layout for the **DashboardLayout** control.

Razor

Copyright © GrapeCity, Inc. All rights reserved.

```
@using ApplicationName.Models
@model IEnumerable<ProductData>

<style>
    .wj-dashboard .wj-flexchart {
        margin: 0px;
        padding: 4px;
        border: none;
        height: 240px;
    }
</style>

<div id="DemoName">
    <p style="font-size:large;">Product Dashboard Demo</p>

    @(Html.C1().FlexPie<ProductData>().Id("CategorySales")
      .Bind("Category", "Sales", Model))

    @(Html.C1().FlexChart().Id("ProductsStock")
     .Bind("ProductName", Model)
     .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsInStock")
        .Name("Stock Units");
     })
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsOnOrder")
        .Name("OrderUnits");
     }))

    @(Html.C1().FlexGrid<ProductData>().Id("ProductDetails")
     .AutoGenerateColumns(false)
     .Bind(Model)
     .Columns(bl =>
     {
         bl.Add(cb => cb.Binding("ProductID").Align("Center"));
         bl.Add(cb => cb.Binding("ProductName"));
         bl.Add(cb => cb.Binding("Category"));
         bl.Add(cb => cb.Binding("ReorderLevel"));
     }))
</div>

<h2>Manual Grid Layout</h2>
<br />
@(Html.C1().DashboardLayout()
    .AttachManualGridLayout(mglb => mglb.Orientation(LayoutOrientation.Vertical)
        .MaxRowsOrColumns(3)
        .CellSize(303)
        .Items(isb =>
```


Copyright © GrapeCity, Inc. All rights reserved.

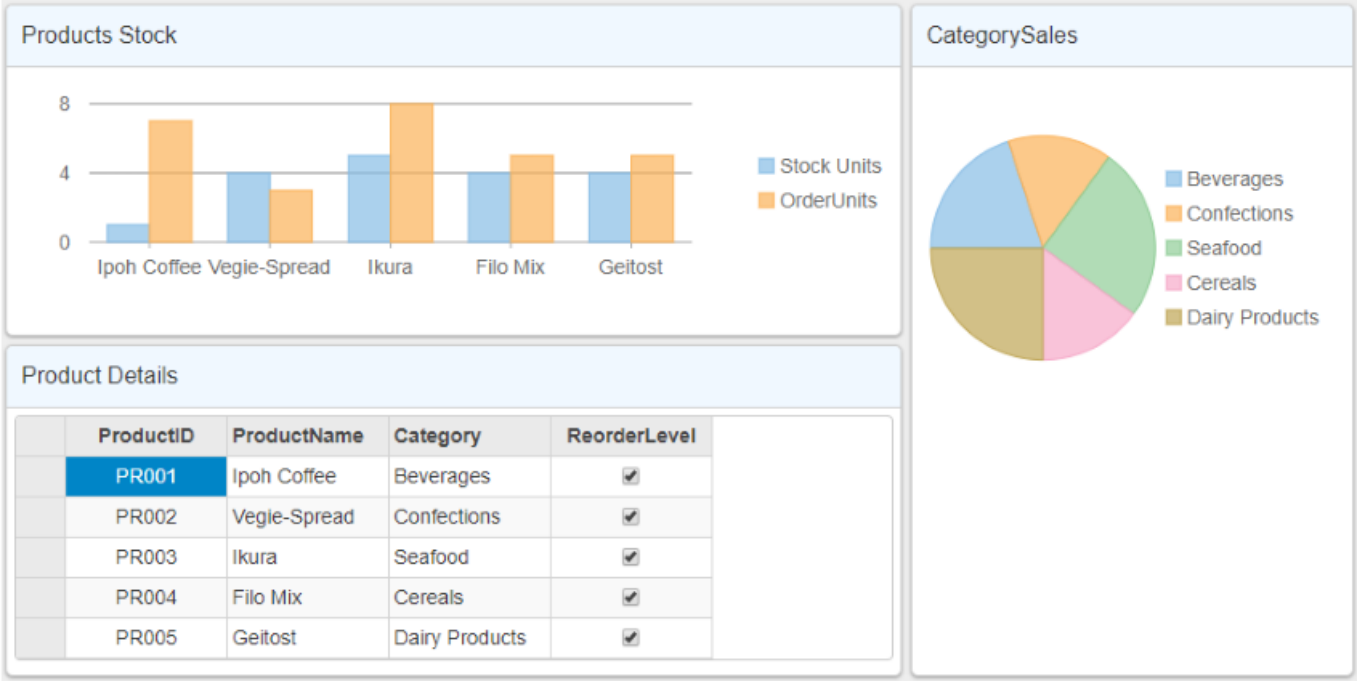
```
        {
            isb.Add().Children(cb =>
            {
                cb.Add().HeaderText("Category Sales")
                    .Content("#CategorySales")
                    .RowSpan(1).ColumnSpan(1);
                cb.Add().HeaderText("Products Stock")
                    .Content("#ProductsStock")
                    .RowSpan(1).ColumnSpan(1)
                    .Row(1).Column(3);
            });
            isb.Add().Children(cb =>
            {
                cb.Add().HeaderText("Product Details")
                    .Content("#ProductDetails")
                    .RowSpan(1).ColumnSpan(2)
                    .Row(1).Column(2);
            });
        }))
```

スプリットレイアウト

The split layout is helpful when we want to add the tiles in different orientations for different groups. It enables the users to have nested groups where in each group can have a different orientation. The group can either consist of another or a tile. Each tile can render any MVC or HTML control. The order in which the groups are rendered is determined by the **Orientation** property. When the orientation is set to Horizontal, the groups or tiles are rendered horizontally and when the orientation is set to Vertical, then the groups or tiles are rendered vertically.

DashboardLayout allows you to set the split layout using its **AttachSplitLayout** method of the DashboardLayoutBuilder class. The split layout divides the dashboard into multiple section where in you can add multiple tiles to the layout. The **SplitGroup** class represents the group for the split layout and on the other hand **SplitTile** class represents the tiles in the split layout.

The following image shows how DashboardLayout control appears after adding a split layout.



Copyright © GrapeCity, Inc. All rights reserved.

The following code example demonstrates how to set split layout for the DashboardLayout control. This example uses the sample created in the QuickStart section.

Razor

```
@using ApplicationName.Models
@model IEnumerable<ProductData>

<style>
    .wj-dashboard .wj-flexchart {
        margin: 0px;
        padding: 4px;
        border: none;
        height: 240px;
    }
</style>

<div>
    @(Html.C1().FlexPie<ProductData>().Id("CategorySales")
      .Bind("Category", "Sales", Model))

    @(Html.C1().FlexChart().Id("ProductsStock")
     .Bind("ProductName", Model)
     .Height(150)
     .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsInStock")
        .Name("Stock Units");
     })
     .Series(sers =>
     {
         sers.Add()
        .Binding("UnitsOnOrder")
        .Name("OrderUnits");
     }))

    @(Html.C1().FlexGrid<ProductData>().Id("ProductDetails")
     .AutoGenerateColumns(false)
     .Bind(Model)
     .Columns(bl =>
     {
         bl.Add(cb => cb.Binding("ProductID").Align("Center"));
         bl.Add(cb => cb.Binding("ProductName"));
         bl.Add(cb => cb.Binding("Category"));
         bl.Add(cb => cb.Binding("ReorderLevel"));
     }))
</div>
<br />
@(Html.C1().DashboardLayout()
    .AttachSplitLayout(slb => slb.Orientation(LayoutOrientation.Horizontal)
        .Items(isb =>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
            isb.AddGroup()
                .Size("2*")
                .Orientation(LayoutOrientation.Vertical)
                .Children(vgcb =>
                {
                    vgcb.AddGroup()
                        .Size("*")
                        .Orientation(LayoutOrientation.Horizontal)
                        .Children(hgcb =>
                        {
                            hgcb.AddTile()
                        .Size("*")
                        .HeaderText("Products Stock")
                        .Content("#ProductsStock");
                        });
                    vgcb.AddTile()
                        .Size("*")
                        .HeaderText("Product Details")
                        .Content("#ProductDetails");
                });

            isb.AddGroup()
                .Size("*")
                .Children(hgcb =>
                {
                    hgcb.AddTile()
                                    .Size("*")
                                    .HeaderText("CategorySales")
                                    .Content("#CategorySales");
                });
        }))
```

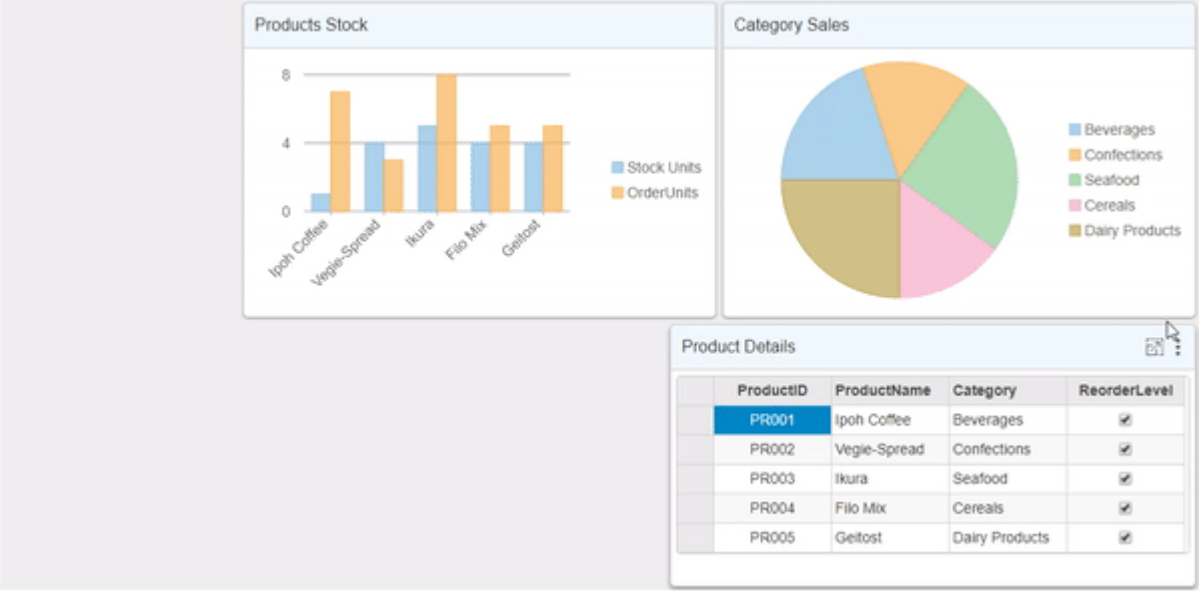
サイズ変更

DashboardLayout allows you to resize the tiles at runtime. The resizing behavior for the tiles differ for different layouts. You can resize any tile with the help of double-header arrow, which appears when you select the tile and place your cursor at the bottom right corner of the tile. The different behaviors for different layouts are discussed below.

Resizing in Flow Layout

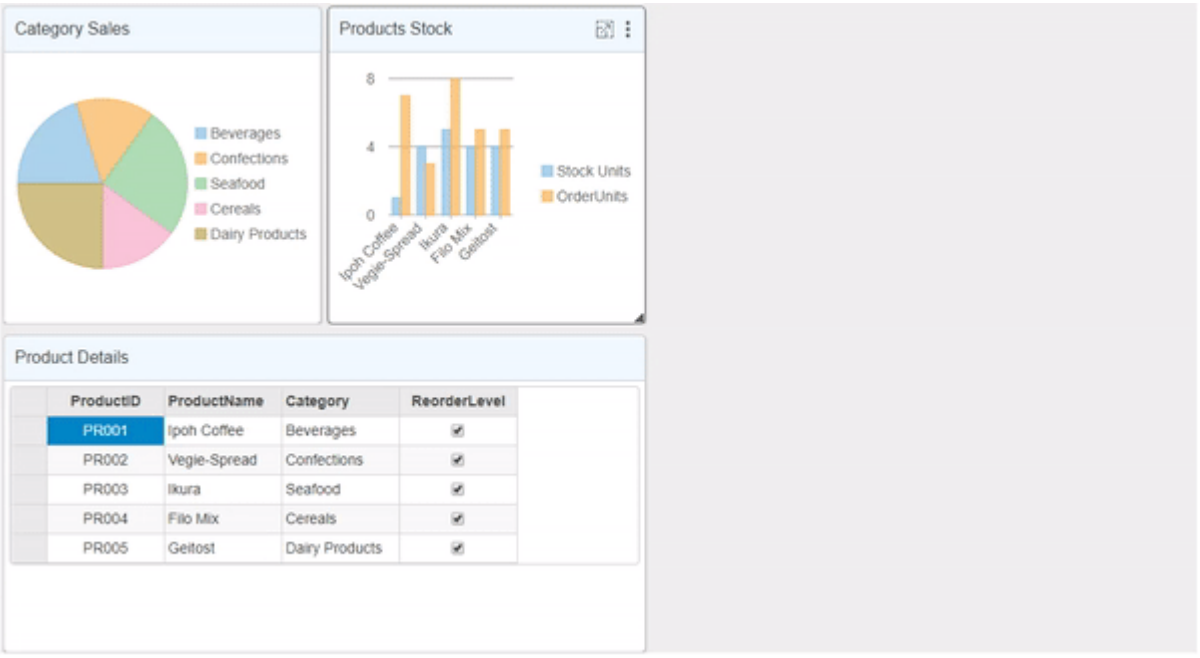
On resizing a specific tile, other tiles rearrange themselves maintaining the flow direction.

Copyright © GrapeCity, Inc. All rights reserved.



Resizing in Auto Grid Layout

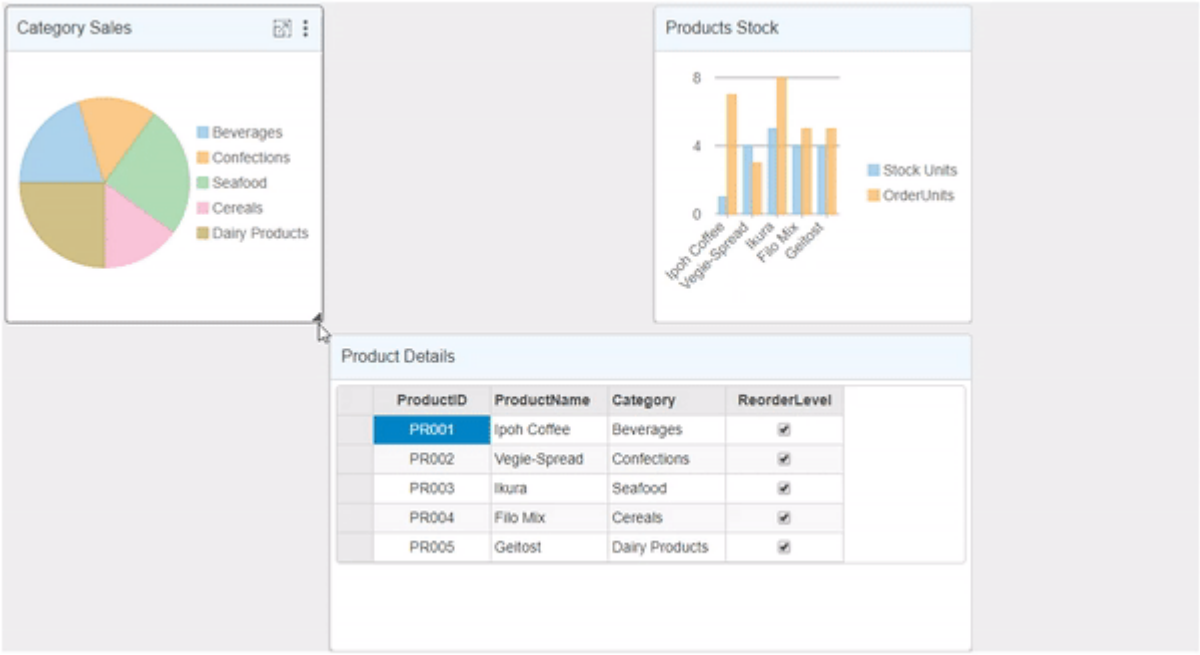
On resizing a specific tile, it spans the adjacent column or row depending on whether the width/height of the tiles are increased respectively.



Resizing in Manual Grid Layout

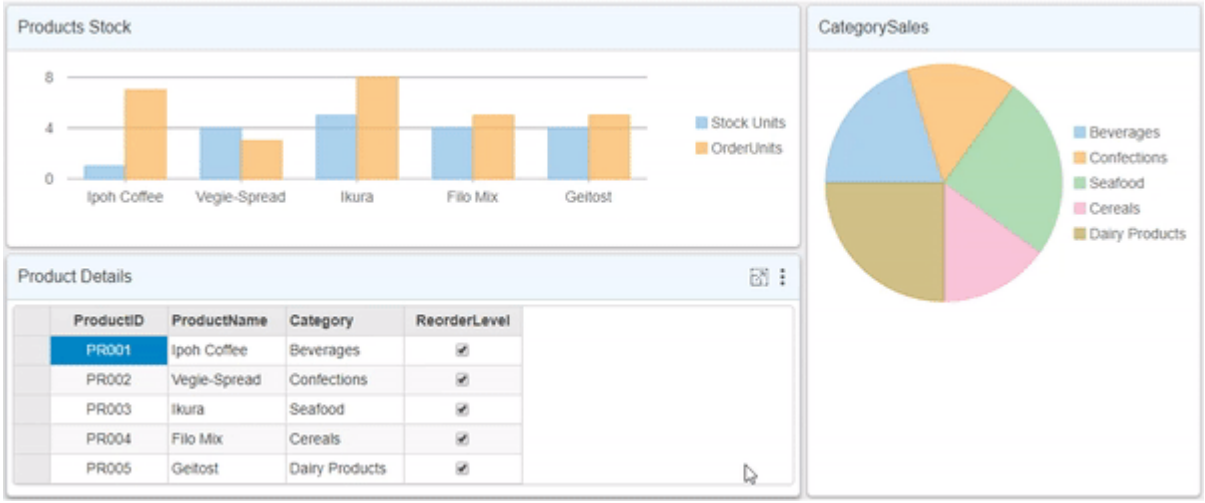
On resizing a specific tile, it spans the adjacent column or row depending on whether the width/height of the tiles are increased respectively.

Copyright © GrapeCity, Inc. All rights reserved.



Resizing in Split Layout

On hovering mouse over the split line, a resizing handle appears over the bar which can be used to resize the tiles at runtime.



By default, in the DashboardLayout control resizing is enabled. However, in case you want to disable resizing of tiles, you can make use of **AllowResize** property provided by the **DashboardLayout** class.

Razor

```
// タイルのサイズ変更を制限します
.AllowResize(false)
```

ツールアイコン

Tool-icon is a multi functional tool which appears as an icon either on the in the header of the tile when it is selected with mouse hover. It appears as three dots positioned vertically.

The tool-icon can be used to drag and drop a tile by hovering the mouse over it and dragging the child container to another position. On clicking, the tool-icon opens a context menu which by default, displays the options for hiding a

Copyright © GrapeCity, Inc. All rights reserved.

tile or show all tiles.

The following GIF image shows the tool-icon operations for each tile.

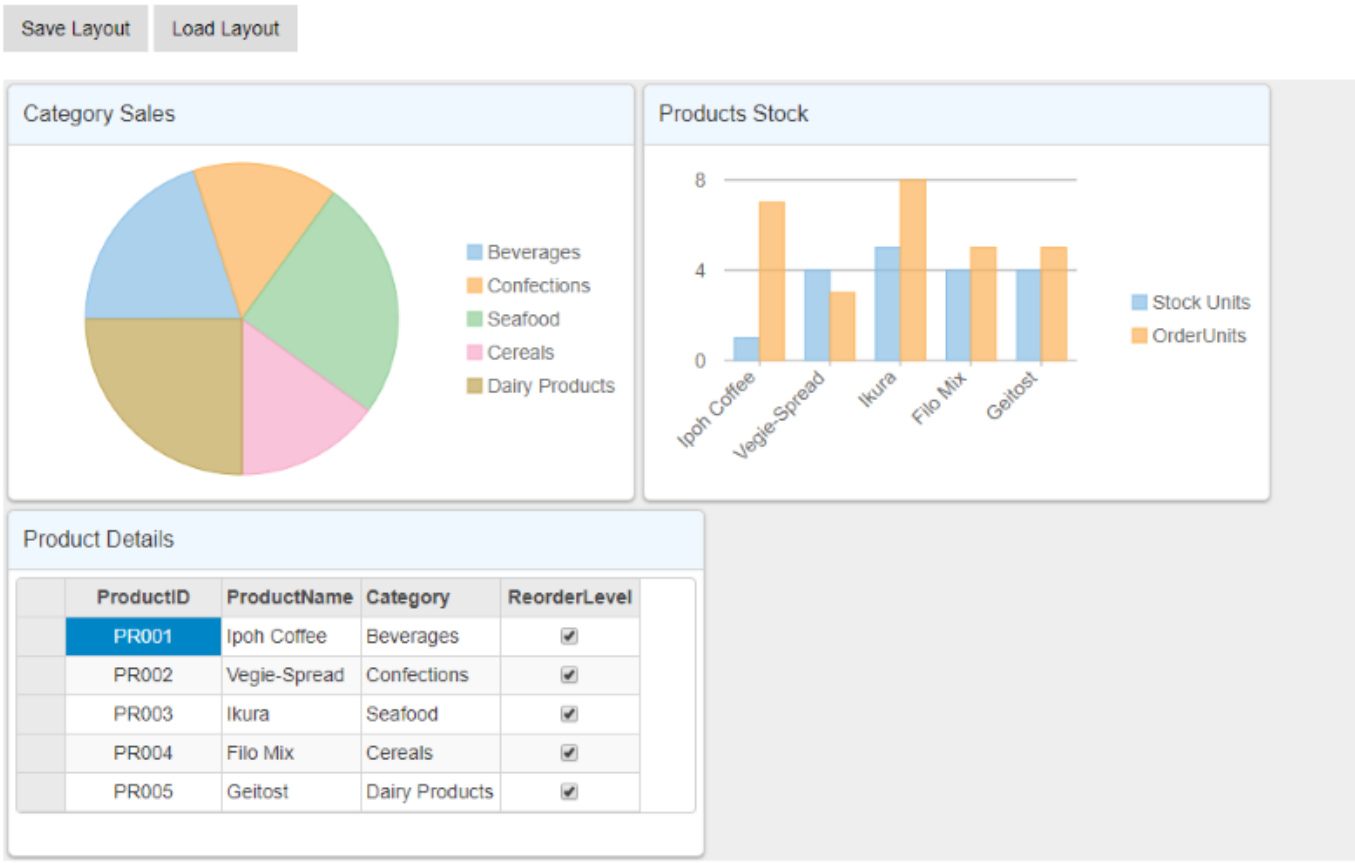


XMLのシリアル化

Serialization refers to converting an object into a stream of bytes. The purpose of serialization is to save the current state of an object so that it can be recreated when required. DashboardLayout supports serialization through **saveLayout** and **loadLayout** client-side methods of the DashboardLayout control. The **saveLayout** method saves the DashboardLayout properties, i.e., order and bounds of tiles, to an XML stream or file. The **loadLayout** method loads the DashboardLayout layout properties from an XML stream or file.

The following image shows DashboardLayout control with load and save layout options.

Copyright © GrapeCity, Inc. All rights reserved.



The following code shows the example of saving and loading the DashboardLayout to and from an XML file.

Razor

```
<input type="button" value="Save Layout" class="btn" onclick="saveLayout()" />
<input type="button" value="Load Layout" class="btn" onclick="loadLayout()" />

<script>
    // localStorageにレイアウト定義を保存または復元します
    function saveLayout() {
        var dashboard = wijmo.Control.getControl('#SampleDashboard');
        localStorage['layout'] = dashboard.saveLayout();
    }
    function loadLayout() {
        var dashboard = wijmo.Control.getControl('#SampleDashboard'),
            layoutDef = localStorage['layout'];
        if (layoutDef) {
            dashboard.loadLayout(layoutDef);
        }
    }
</script>
```

チュートリアル

This section contains information about

カスタムタイル

Copyright © GrapeCity, Inc. All rights reserved.

Learn how to add custom tiles in the DashboardLayout control.

カスタムツールバー
Learn how to add custom toolbar options in the DashboardLayout control.

カスタムツールバーのオプション
Learn how to add custom context menu options in the DashboardLayout control.

カスタムタイル

This walkthrough depicts how you can customize the tile header and the tile content. The walkthrough lets you accomplish the following customizations.

1. Hide the default toolbar and the default header.
2. Modify the tile header to display a custom title.
3. Update the tile content.

The topic comprises the following steps.



Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add Data to the Application

1. Add a new class to the **Models** folder (Name: `DashboardData.cs`). For more information on how to add a new model, see Adding Controls.
2. Replace the following code in the `DashboardData.cs` model to define the classes that serve as a data source for the different controls rendered in the DashboardLayout control.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace CustomTile.Models
{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public class DashboardData
    {
        private IEnumerable<CountryData> _countryDetails = null;
        public IEnumerable<CountryData> CountryDetails
        {
            get
            {
                if (_countryDetails == null)
                {
                    _countryDetails = GetCountryData();
                }
                return _countryDetails;
            }
        }

        public IEnumerable<CountryData> GetCountryData()
        {
            var rand = new Random(0);
            var countryID = new[] { "CR001", "CR002", "CR003", "CR004", "CR005",
"CR006" };
            var countries = new[] { "US", "Germany", "UK", "Japan", "China",
"India" };

            var list = countries.Select((c, i) =>
            {
                double sales = rand.Next(1, 6);
                double budget = rand.Next(1, 9);
                double expenses = rand.Next(1, 6);
                return new CountryData { ID = countryID[i], Country = c, Sales =
sales, Budget = budget, Expenses = expenses };
            });
            return list;
        }
    }
    public class CountryData
    {
        public string ID { get; set; }
        public string Country { get; set; }
        public double Sales { get; set; }
        public double Budget { get; set; }
        public double Expenses { get; set; }
    }
}
```

Back to Top

Add Controller to the Application

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template, and then click **Add**.

Copyright © GrapeCity, Inc. All rights reserved.

2. Set name of the controller (for example: `DashboardController`).
3. Click **Add**.

4. Include the following references as shown below.

C#

```
using <ApplicationName>.Models;
```

5. Replace the **Index()** method with the following method.

DashboardController.cs

```
public ActionResult Index()
        {
            DashboardData data = new DashboardData();
            return View(data.CountryDetails);
        }
```

Back to Top

Add View to the Application

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `DashboardController.`
2. Place the cursor inside the method `Index().`
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller, and then copy the following codes and paste it inside **Index.cshtml**.

The code snippet provided below shows how to handle the **OnClientFormatTile** event provided by **DashboardLayout** class to create custom tiles for the DashboardLayout control.

The event argument for this event is of type **TileFormattedEventArgs** and provides access to different elements of the tile. In the sample code below, the tile header text is accessed by using the “headerText” property, the header element is accessed using “headerElement” property and the “contentElement” property is used to access the content of the tile. These properties have been used to update the header content and the tile content.

Index.cshtml

```
<script type="text/javascript">
    function formatTile(sender, e) {
        var dashboard = sender, // DashboardLayoutコントロールを取得します
            tile = e.tile; // 書式設定されたタイルを取得します

        switch (tile.headerText) {
            case 'Sales Dashboard': UpdateHeader(e.headerElement);
                break;
            case 'Country': UpdateTileContent(e.tile, e.contentElement);
                break;
        }
    }
    // ヘッダーのタイトルを変更します
    function UpdateHeader(header) {
        var headerText = 'Sales Dashboard for 2018';
        header.querySelector('span.title').innerText = headerText;
    }
    // タイルの内容を更新します
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    function UpdateTileContent(tile, contentElement) {
        var grid = wijmo.Control.getControl('#salesDashboardFGrid');

        if (grid && grid.selectedItems && grid.selectedItems.length) {
            var selectedRowData = grid.selectedItems[0];
            tile.hostElement.style.backgroundColor = '#009ccc';
            var htmlContent = '<div style="color:white;">Country</div>' +
                '<div style="font-size:72px; text-align: center;
color:white;overflow:hidden; text-overflow:ellipsis">' + selectedRowData.Country
+ '</div>';
            contentElement.innerHTML = htmlContent;
        }
    }
    function gridSelectionChanged(sender, e) {
        // グリッドでselectionChangedが発生した後、DashboardLayoutコントロールを更新します
        var dashboard = wijmo.Control.getControl('#custom');
        dashboard.refresh();
    }
</script>
```

The code snippet provided below initializes a **DashboardLayout** control and the controls that are to be rendered inside the DashboardLayout control.

Index.cshtml

```
@using <ApplicationName>.Models
@using C1.Web.Mvc.Grid;
@model IEnumerable<CountryData>

<div style="position:absolute;left:-10000px; top:-10000px; visibility:hidden">
    @(Html.C1().FlexGrid().Id("salesDashboardFGrid")
    .IsReadOnly(true).AutoGenerateColumns(false)
    .HeadersVisibility(HeadersVisibility.Column)
    .AllowResizing(AllowResizing.None)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)
    .Bind(Model)
    .Columns(clsb =>
    {
        clsb.Add(cb => cb.Header("Country").Binding("Country").Width("*"));
        clsb.Add(cb => cb.Header("Current Year(mil.)").Binding("Sales")
        .Format("c0").Width("*"));
    })
    .OnClientSelectionChanged("gridSelectionChanged"))
</div>
<br />
@(Html.C1().DashboardLayout().Id("custom")
    .AttachAutoGridLayout(aglb =>
        aglb.Orientation(LayoutOrientation.Vertical)
            .MaxRowsOrColumns(6)
            .CellSize(152)
            .Items(isb =>
            {
                isb.Add().Children(cb =>
                {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                    cb.Add().HeaderText("Sales Dashboard")
                        .Content("#salesDashboardFGrid")
                        .ColumnSpan(2).RowSpan(2)
                        .ShowToolbar(false);
                    cb.Add().HeaderText("Country")
                        .ColumnSpan(2).RowSpan(1)
                        .ShowHeader(false).ShowToolbar(false);
                });
            })
    )
    .OnClientFormatTile("formatTile"))
```

Back to Top

Build and Run the Project

1. Configure the **RouteConfig.cs** file to set the default navigation path by setting the controller to **Dashboard** and action to **Index**.
2. Click **Build | Build Solution** to build the project.
3. Press **F5** to run the project.

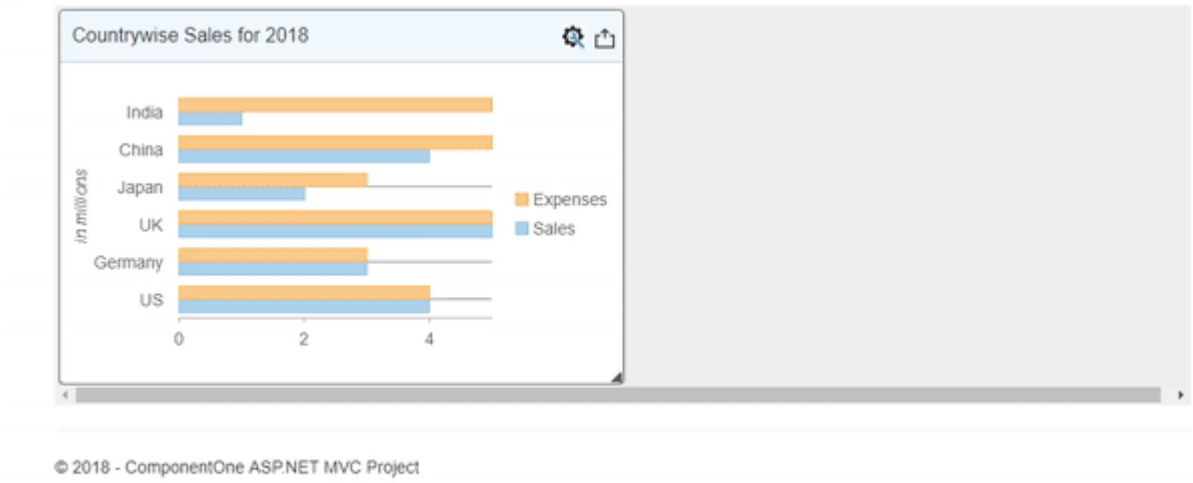
Back to Top

カスタムツールバー

This walkthrough depicts how you can customize the default options displayed in the toolbar of the tile header. The default options have been removed and the following two options are added in the toolbar.

1. To change the chart type of the chart displayed in the tile.
2. To export the chart to an image.

The topic comprises the following steps.



Create an MVC Application

Copyright © GrapeCity, Inc. All rights reserved.

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add Data to the Application

1. Add a new class to the **Models** folder (Name: `DashboardData.cs`). For more information on how to add a new model, see Adding Controls.
2. Replace the following code in the `DashboardData.cs` model to define the classes that serve as a datasource for the different controls rendered in the DashboardLayout control.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace CustomToolBar.Models
{
    public class DashboardData
    {
        private IEnumerable<CountryData> _countryDetails = null;
        public IEnumerable<CountryData> CountryDetails
        {
            get
            {
                if (_countryDetails == null)
                {
                    _countryDetails = GetCountryData();
                }
                return _countryDetails;
            }
        }

        public IEnumerable<CountryData> GetCountryData()
        {
            var rand = new Random(0);
            var countryID = new[] { "CR001", "CR002", "CR003", "CR004", "CR005",
"CR006" };
            var countries = new[] { "US", "Germany", "UK", "Japan", "China",
"India" };

            var list = countries.Select((c, i) =>
            {
                double sales = rand.Next(1, 6);
                double budget = rand.Next(1, 9);
                double expenses = rand.Next(1, 6);
                return new CountryData { ID = countryID[i], Country = c, Sales =
sales, Budget = budget, Expenses = expenses };
            });

            return list;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
    }
    public class CountryData
    {
        public string ID { get; set; }
        public string Country { get; set; }
        public double Sales { get; set; }
        public double Budget { get; set; }
        public double Expenses { get; set; }
    }
}
```

Back to Top

Add Controller to the Application

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template, and then click **Add**.
    2. Set name of the controller (for example: `DashboardController`).
    3. Click **Add**.
4. Include the following references as shown below.

C#

```
using <ApplicationName>.Models;
```

5. Replace the **Index()** method with the following method.

DashboardController.cs

```
public ActionResult Index()
        {
            DashboardData data = new DashboardData();
            return View(data.CountryDetails);
        }
```

Back to Top

Add View to the Application

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `DashboardController.`
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller, and then copy the following code and paste it inside **Index.cshtml**.

   The code snippet provided below shows how to handle the **OnClientFormatTile** client event provided by **DashboardLayout** class to create a custom toolbar for the DashboardLayout control. The event argument for this event is of type **TileFormattedEventArgs** and provides access to different elements of the tile.

   In the sample code below, the tile toolbar is accessed by using the **toolbar** property. This property returns an instance of the toolbar which provides different methods such as **insertBefore** and **insertToolbarItem**, to add new toolbar item. Note that we have also added two images in the sample to display the custom icons for the

Copyright © GrapeCity, Inc. All rights reserved.

toolbar options.

The code snippet provided below initializes a **DashboardLayout** control and the controls that are to be rendered inside the DashboardLayout control.

Index.cshtml

```
@using <ApplicationName>.Models
@using C1.Web.Mvc.Chart;
@model IEnumerable<CountryData>

<script type="text/javascript">
    var popup, cmbChartType;

    function formatTile(sender, e) {
        var dashboard = sender, // DashboardLayoutコントロールを取得します
            tile = e.tile; // 書式設定されたタイルを取得します

        if (tile.headerText == 'Countrywise Sales') {
                // ヘッダーのタイトルを変更します
                var headerText = "Countrywise Sales for 2018";
                e.headerElement.querySelector('span.title').innerText =
headerText;

                // 内部の項目をすべてクリアします
                e.toolbar.clear();

                AddChartTypeOption(e.toolbar);// DOMでチャートオプションを設定するため、
ツールバーに「Settings」項目を追加します。
                AddExportOption(e.toolbar, e.tile);// ツールバーAPIを使用して内容をエク
スポートするために、ツールバーにカスタム項目を追加します。
        }
    }

    function AddChartTypeOption(toolbar) {
        var iconClose = document.createElement('img');
        iconClose.title = 'Settings';
        iconClose.alt = 'Settings';
        iconClose.style.marginRight = '6px';
        iconClose.style.cursor = 'default';
        iconClose.src = '@Href("~/Images/th.png")';

        // 最初の位置に項目を挿入します
        var eleToolbar = toolbar.hostElement;
        eleToolbar.insertBefore(iconClose, eleToolbar.firstChild);
        iconClose.addEventListener('click', iconCloseClick);
    }

    function iconCloseClick() {
        // この項目をクリックすると、チャートの種類を指定するダイアログが表示されます
        if (!popup) {
            popup = new wijmo.input.Popup('#popup');
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        if (!cmbChartType) {
            cmbChartType = new wijmo.input.ComboBox('#chartType', { itemsSource:
["Column", "Bar", "Scatter", "Line", "LineSymbols", "Area", "Spline",
"SplineSymbols", "SplineArea"] });
        }

        var countrywiseSalesChart =
wijmo.Control.getControl('#countrywiseSalesChart');
        if (countrywiseSalesChart) {
            cmbChartType.text =
wijmo.chart.ChartType[countrywiseSalesChart.chartType];
        }

        popup.show(true, function (e) {
            if (e.dialogResult == 'wj-hide-ok') {
                // チャートの種類を適用します
                var chart = wijmo.Control.getControl('#countrywiseSalesChart');
                chart.chartType = wijmo.chart.ChartType[cmbChartType.text];
            }
        });
    }

    function AddExportOption(toolbar, tile) {
        var strExportIcon = '<img style="vertical-align:middle"
src="@Href("~/Images/icon_export.png")" alt="Export" title="Export" />';
                toolbar.insertToolbarItem({
                    icon: strExportIcon,
                    title: 'Export',
                    command: function () {
                        var selector = e.tile.content,
                        chart = wijmo.Control.getControl(selector);
                        chart.saveImageToFile(selector.substr(1) + '.png');
                    }
                }, 0);
    }
</script>
<div style="position:absolute;left:-10000px; top:-10000px; visibility:hidden">
    @(Html.C1().FlexChart().Id("countrywiseSalesChart")
        .BindingX("Country").ChartType(ChartType.Bar)
        .Legend(Position.Right)
        .Bind(Model)
        .AxisY(ayb => ayb.Title("in millions").Format("g4,,"))
        .Series(ssb =>
        {
            ssb.Add().Name("Sales").Binding("Sales");
            ssb.Add().Name("Expenses").Binding("Expenses");
        })
        .Tooltip(ttb => ttb.Content("<b>{seriesName}</b><br/>{x} {y:c0}"))
        .ShowAnimation(ab => ab.AnimationMode(AnimationMode.All)
            .Easing(Easing.Swing).Duration(400)))

    <div id="popup" class="modal-content">
        <div class="modal-header">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <button type="button" tabindex="-1" class="close wj-hide">
                <span>×</span>
            </button>
            <h4 class="modal-title">ChartType</h4>
        </div>
        <div class="modal-body">
            <div class="wj-labeled-input">
                <input id="chartType" />
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-primary wj-hide-
ok">OK</button>
                <button type="button" class="btn btn-default wj-
hide">Cancel</button>
            </div>
        </div>
    </div>
</div>

@(Html.C1().DashboardLayout().Id("custom")
    .AttachAutoGridLayout(aglb =>
        aglb.Orientation(LayoutOrientation.Vertical)
            .MaxRowsOrColumns(6)
            .CellSize(152)
            .Items(isb =>
            {
                isb.Add().Children(cb =>
                {
                    cb.Add().HeaderText("Countrywise Sales")
                        .Content("#countrywiseSalesChart")
                        .RowSpan(2).ColumnSpan(3);
                });
            })
    )
    .OnClientFormatTile("formatTile"))
```

Back to Top

Build and Run the Project

1. Configure the **RouteConfig.cs** file to set the default navigation path by setting the controller to **Dashboard** and action to **Index**.
2. Click **Build | Build Solution** to build the project.
3. Press **F5** to run the project.

Back to Top

カスタムツールバーのオプション

This walkthrough depicts how you can add more options to the toolbar menu which is displayed when clicking on the tool icon in the toolbar. In this walkthrough, we have added an **Export** option to the menu, which when clicked exports/saves the chart displayed in the tile to a PNG image.

Copyright © GrapeCity, Inc. All rights reserved.

Follow the given steps to get started:



Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add Data to the Application

1. Add a new class to the **Models** folder (Name: `DashboardData.cs`). For more information on how to add a new model, see Adding Controls.
2. Replace the following code in the `DashboardData.cs` model to define the classes that serve as a datasource for the different controls rendered in the DashboardLayout control.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace CustomContextMenu.Models
{
    public class DashboardData
    {
        private IEnumerable<CountryData> _countryDetails = null;
        public IEnumerable<CountryData> CountryDetails
        {
            get
            {
                if (_countryDetails == null)
                {
                    _countryDetails = GetCountryData();
                }
                return _countryDetails;
            }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }

        public IEnumerable<CountryData> GetCountryData()
        {
            var rand = new Random(0);
            var countryID = new[] { "CR001", "CR002", "CR003", "CR004", "CR005",
"CR006" };
            var countries = new[] { "US", "Germany", "UK", "Japan", "China",
"India" };

            var list = countries.Select((c, i) =>
            {
                double sales = rand.Next(1, 6);
                double budget = rand.Next(1, 9);
                double expenses = rand.Next(1, 6);
                return new CountryData { ID = countryID[i], Country = c, Sales =
sales, Budget = budget, Expenses = expenses };
            });
            return list;
        }
    }
    public class CountryData
    {
        public string ID { get; set; }
        public string Country { get; set; }
        public double Sales { get; set; }
        public double Budget { get; set; }
        public double Expenses { get; set; }
    }
}
```

Back to Top

Add Controller to the Application

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template, and then click **Add**.
    2. Set name of the controller (for example: `DashboardController`).
    3. Click **Add**.
4. Include the following references as shown below.

C#

```
using <ApplicationName>.Models;
```

5. Replace the **Index()** method with the following method.

DashboardController.cs

```
public ActionResult Index()
        {
            DashboardData data = new DashboardData();
            return View(data.CountryDetails);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
```

Back to Top

Add View to the Application

The code snippet provided below shows how to handle the **OnClientFormatTile** client event provided by **DashboardLayout** class to create a custom toolbar for the DashboardLayout control. The event argument for this event is of type **TileFormattedEventArgs** and provides access to different elements of the tile.
In this sample code, the tile toolbar is accessed by using the **toolbar** property. This property returns an instance of the toolbar which provides different methods such as **insertBefore** and **insertToolbarItem**, to add new toolbar item.
Note that we have also added two images in the sample to display the custom icons for the toolbar options.
The code snippet provided below initializes a **DashboardLayout** control and the controls that are to be rendered inside the DashboardLayout control. Follow the given steps to accomplish the above:

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `DashboardController`.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller, and then copy the following code and paste it inside **Index.cshtml**.

Index.cshtml

```
@using <ApplicationName>.Models
@using C1.Web.Mvc.Chart;
@model IEnumerable<CountryData>

<script type="text/javascript">
    function formatTile(dashboardLayout, e) {
            var tile = e.tile,
                toolbar = e.toolbar;
            if(tile.headerText == ('Cost Budgeting for 2018')){
                var strExportIcon = '<img style="vertical-align:middle"
src="@Href("Images/icon_export.png")" alt="Export" title="Export" />';
                toolbar.insertToolbarItem({
                    icon: strExportIcon,
                    title: 'Export',
                    command: function () {
                        var selector = e.tile.content,
                        chart = wijmo.Control.getControl(selector);
                        chart.saveImageToFile(selector.substr(1) + '.png');
                    }
                });
            }
        }
</script>
<div style="position:absolute;left:-10000px; top:-10000px; visibility:hidden">
    @(Html.C1().FlexChart().Id("costBudgetingChart")
        .BindingX("Country").ChartType(ChartType.Column)
        .Legend(Position.Right)
        .Bind(Model)
        .AxisY(ayb => ayb.Title("in millions"))
        .Series(ssb =>
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            ssb.Add().Name("Budget").Binding("Budget");
            ssb.Add().Name("Expenses").Binding("Expenses");
        })
        .Tooltip(ttb => ttb.Content("<b>{seriesName}</b><br/>{x} {y:c0}"))
        .ShowAnimation(ab => ab.AnimationMode(AnimationMode.All)
        .Easing(Easing.Swing).Duration(400)))
</div>
<br />
@(Html.C1().DashboardLayout()
        .Id("custom")
        .AttachFlowLayout(flb => flb.Direction(FlowDirection.LeftToRight)
        .Items(isb =>
        {
            isb.Add(ftb => ftb.HeaderText("Cost Budgeting for 2018")
            .Content("#costBudgetingChart").Width(549).Height(322));
        }))
        .OnClientFormatTile("formatTile"))
```

Back to Top

Build and Run the Project

1. Configure the **RouteConfig.cs** file to set the default navigation path by setting the controller to **Dashboard** and action to **Index**.
2. Click **Build | Build Solution** to build the project.
3. Press **F5** to run the project.

Back to Top

Excel

Excelコントロールでは、1つのコマンドでワークブックをロードまたは保存できます。Microsoft Excelがインストールされている必要もありません。XLSファイルを作成またはロードできます。Excelコントロールは、OpenXml形式をサポートします。この形式を使用すると、小さく圧縮されたXLSXファイルを保存できます。

Excel for .NETでは、アプリケーションのコードから直接Microsoft Excelファイルを作成できます。コードで公開される操作しやすいExcelオブジェクトモデルの可能性は無限です。グリッド、スケジュール、チャートなど、Excelエクスポートを直接サポートしないデータからExcelファイルを作成できます。

詳細については、Excel for .NET を参照してください。

主な特長

Excelのいくつかの主な特長が次のとおりです。

- **単一コマンドによるワークブックの保存またはロード**
  1つのコマンドを使用して、ワークブックをロードまたは保存したり、シートを操作することができます。

- **Microsoft Excelが不要**
  Microsoft Excelに依存することなく、XLS（Excel 97以上）ファイルとXLSX（OpenXml形式）ファイルを読み書きすることができます。

- **完全なMicrosoft Excel互換**

Copyright © GrapeCity, Inc. All rights reserved.

>1つのプロパティC1XLBook.CompatibilityModeを設定するだけで、ワークブックをどのバージョンのMicrosoft Excel互換とするかを指定できます。Excel2003（最大64k行×256列）、Excel2007（最大100万行×18k列）、およびNoLimits（制限なし）の3つのオプションがあります。

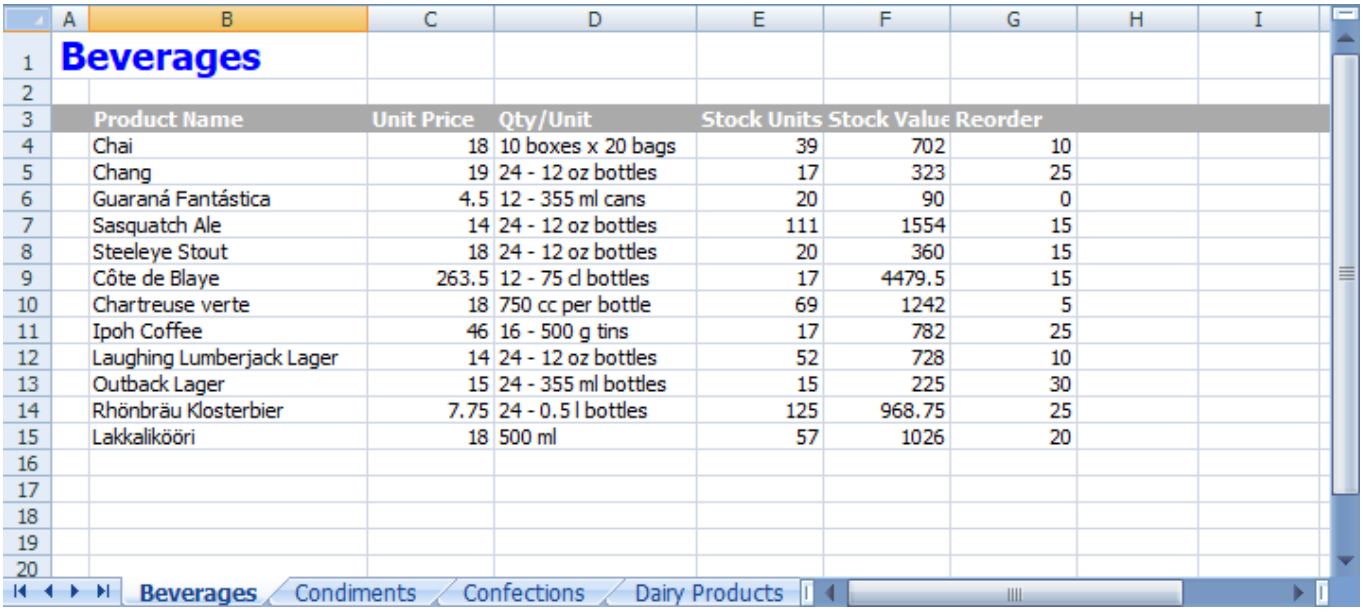
- **データの書式設定**
  エンドユーザーから正しいデータが入力されるように、セルを書式設定できます。各セルに関連付けられた書式には、セルに格納されているデータと同様に簡単にアクセスできます。

- **セルの数式**
  XLCell.Formulaプロパティを使用して、セルの数式を指定できます。

- **グループ化と小計**
  行と列の小計を計算できます。コードでアウトラインレベルのグループ化を宣言して、合計や小計を最適に表示できます。

クイックスタート

The quick start guides you through the steps of adding C1XLBook control in your MVC web application and adding data to it. Complete the following steps to see how the C1XLBook control appears after data binding:



Back to Top

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Configure the Data Source for Application

> The example uses **C1NWind** database. The **C1NWind.mdf** file is available on your system at the following location:
>
> *Documents\ComponentOne Samples\ASP.NET MVC\MVC\MvcExplorer\App_Data*

Copyright © GrapeCity, Inc. All rights reserved.

1. Add **C1NWind.mdf** file to the `AppData` folder in the Solution Explorer.
2. In the Solution Explorer, right-click `Models|Add New Item|Data`, and select **ADO.NET Entity Data Model**.
3. Name the model as **C1NWind**, and click **Add**.
4. In the Entity Data Model Wizard, select **EF Designer from database**, click **Next**. **C1NWind.mdf** database is added to the data connection dropdown.
5. Click **Next** to choose Entity Framework version, and click **Next**.
6. In the **Choose Your Database Objects and Settings**, select `Products` table and click **Finish**.

If you can see **C1NWind.edmx** added to your project under the **Models** folder, you have successfully configured the data source for your application.

Back to Top

Add a C1XLBook control

Complete the following steps to initialize a FlexGrid control.

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `ExcelController`).
    3. Click **Add**.
4. Include the MVC references as shown below.

C#

```
using <ApplicationName>.Models;
```

5. In ExcelController, create an Excel file using CreateExcelFile method as shown in the following code:

C# copyCode

```
private string CreateExcelFile()
{
    //Excelブックをクリアし、1つの空白シートを削除します
    _c1xl.Clear();
    _c1xl.Sheets.Clear();
    _c1xl.DefaultFont = new Font("Tahoma", 8);

    //Excelスタイルを作成します
    _styTitle = new XLStyle(_c1xl);
    _styHeader = new XLStyle(_c1xl);

    //スタイルを設定します
    _styTitle.Font = new Font(_c1xl.DefaultFont.Name, 15, FontStyle.Bold);
    _styTitle.ForeColor = Color.Blue;
    _styHeader.Font = new Font(_c1xl.DefaultFont, FontStyle.Bold);
    _styHeader.ForeColor = Color.White;
    _styHeader.BackColor = Color.DarkGray;

    //カテゴリごとに1つのシートでレポートを作成します
    List<Category> categories = db.Categories.ToList<Category>();
    foreach (Category category in categories)
    {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        CreateSheet(category);
    }

    //XLSファイルへ保存します
    string uid = System.Guid.NewGuid().ToString();
    string filename = Server.MapPath("~") + uid + ".xlsx";
    _c1xl.Save(filename);

    return filename;
}
```

6. Add new worksheets to the Excel workbook for each product category using the following code.

C# copyCode

```
private void CreateSheet(Category category)
{
    //現在のカテゴリ名を取得します
    string catName = category.CategoryName;

    //ブックに新しいワークシートを追加します
    //(「/」はシート名では無効なので、「+」に置き換えます)
    string sheetName = catName.Replace("/", " + ");
    XLSheet sheet = _c1xl.Sheets.Add(sheetName);

    //ワークシートにタイトルを追加します
    sheet[0, 0].Value = catName;
    sheet.Rows[0].Style = _styTitle;

    // 列幅(twip単位)を設定します
    sheet.Columns[0].Width = 300;
    sheet.Columns[1].Width = 2200;
    sheet.Columns[2].Width = 1000;
    sheet.Columns[3].Width = 1600;
    sheet.Columns[4].Width = 1000;
    sheet.Columns[5].Width = 1000;
    sheet.Columns[6].Width = 1000;

    //列ヘッダーを追加します
    int row = 2;
    sheet.Rows[row].Style = _styHeader;
    sheet[row, 1].Value = "Product Name";
    sheet[row, 2].Value = "Unit Price";
    sheet[row, 3].Value = "Qty/Unit";
    sheet[row, 4].Value = "Stock Units";
    sheet[row, 5].Value = "Stock Value";
    sheet[row, 6].Value = "Reorder";

    //このカテゴリの製品をループします
    List<Product> products = db.Products.Where(pro => pro.CategoryID ==
category.CategoryID).ToList<Product>();
    foreach (Product product in products)
    {
        //次の行に移動します
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            row++;

            //データを含む行を追加します
            sheet[row, 1].Value = product.ProductName;
            sheet[row, 2].Value = product.UnitPrice;
            sheet[row, 3].Value = product.QuantityPerUnit;
            sheet[row, 4].Value = product.UnitsInStock;

            //在庫の値を計算します
            double valueInStock = Convert.ToDouble(product.UnitPrice) *
Convert.ToInt32(product.UnitsInStock);
            sheet[row, 5].Value = valueInStock;
            sheet[row, 6].Value = product.ReorderLevel;
        }
    }
```

Add a View for the controller:

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the controller (for example: `ExcelController`) to open it.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add**. A view is added for the controller.
6. In the Solution Explorer, double click `Index.cshtml` to open it.
7. Replace the default code in the **Views\Index.cshtml** file with the following code to display a message about the Excel file being saved to server.

C# copyCode

```
@{
    Layout = null;
}

<br />
<br />
@Html.ActionLink("GenerateExcel", "GenerateExcel")
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

This creates a workbook with NorthWind product information using the C1XLBook control where each product category is placed on a separate worksheet.

Back to Top

File Manager

File Manager is a UI control which allows you to manage your files efficiently. It handles all your files, whether stored in local server or cloud storage accounts. The control integrates with Cloud Storage APIs for listing, uploading,

Copyright © GrapeCity, Inc. All rights reserved.

downloading, moving, and deleting files. It supports integration with the AWS, Azure, DropBox, GoogleDrive, and OneDrive clouds.



主な特長

File Manager provides the following key features:

- **Intuitive and user-friendly UI**
  File Manager has simple, intuitive and easy to use UI which supports intuitive features like search, navigation, context menus to manage files and navigate through different folders.

- **Support for major cloud storage solutions**
  File Manager, in association with Cloud Web API supports Azure, AWS, DropBox, Google Drive, OneDrive storages. It makes HTTP calls to the ComponentOne Cloud Web API's to perform operations which in turn communicate with the native file systems of the respective storages.

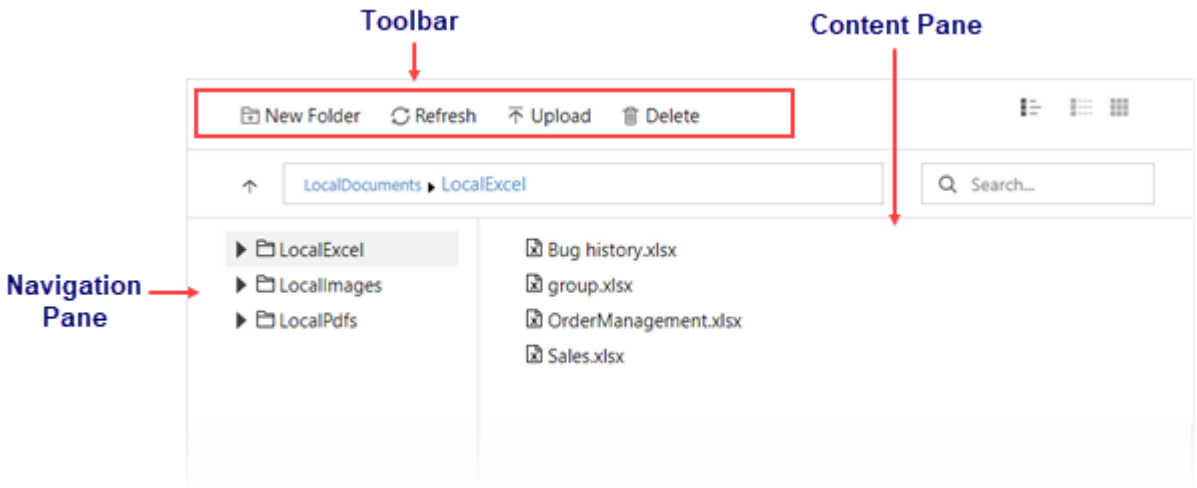
- **Fully features file management**
  File Manager allows you to perform various data operations, such as upload, download and delete files from cloud and local storage.

- **Theme Support**
  FileManager supports over two dozen available themes of the MVC studio which can be used to style the control.

UI要素

FileManager UI consists of toolbar, navigation pane, and content pane. The toolbar contains the list of options for performing various common operations like adding a new folder, refreshing the view, uploading file and deleting file. The navigation pane shows the folder hierarchy and the content pane displays the files and folders in the file system.

Copyright © GrapeCity, Inc. All rights reserved.

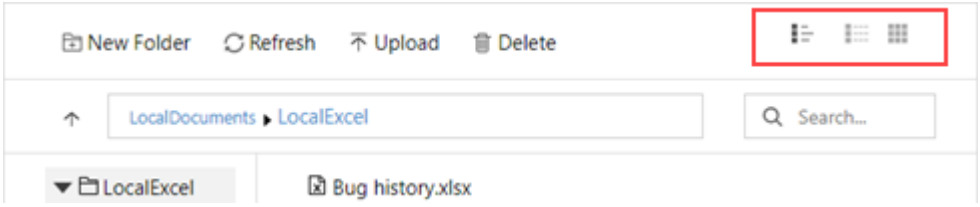


The toolbar consists of the following options:

Options	Description
New Folder	Creates a new folder in the file system.
Refresh	Refreshes the view.
Upload	Uploads files to the cloud or local storage file system.
Delete	Deletes files from the file system.

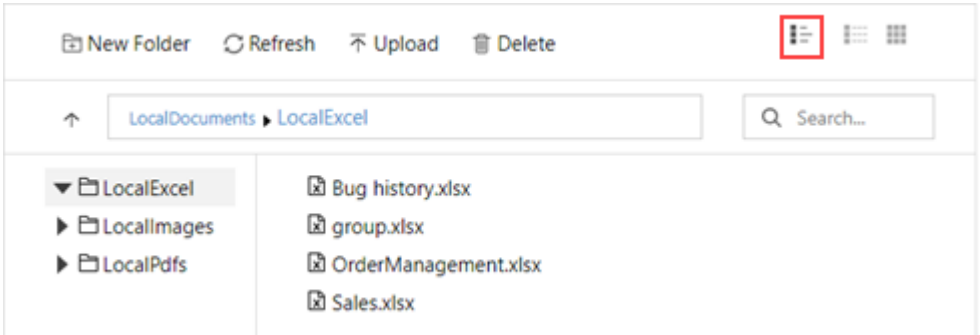
表示

File Manager supports three views, namely list, detail and thumbnail view, for content visualization. You can switch between the views from the top-right of the File Manager UI.



List View

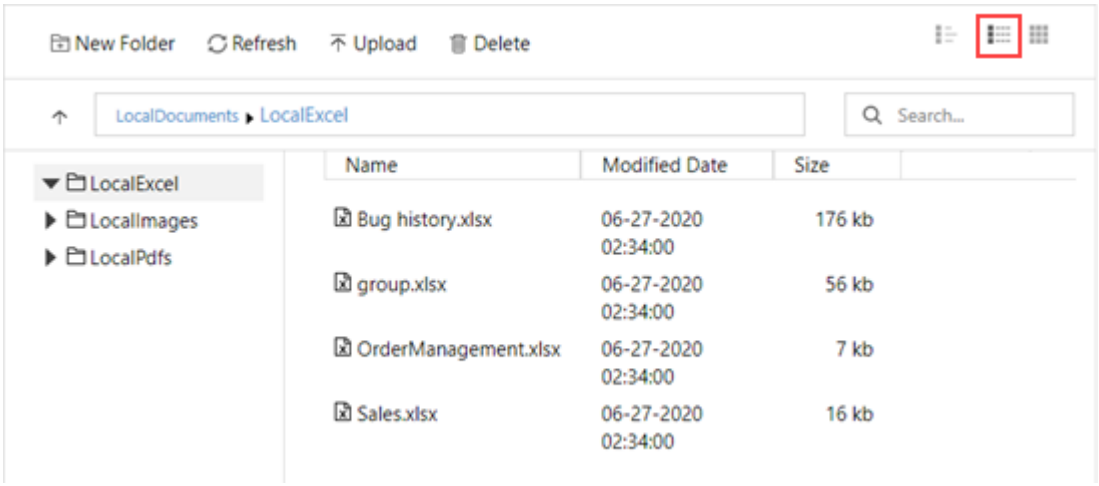
The list view shows the list of files with icons based on file type.



Detail View (Grid View)

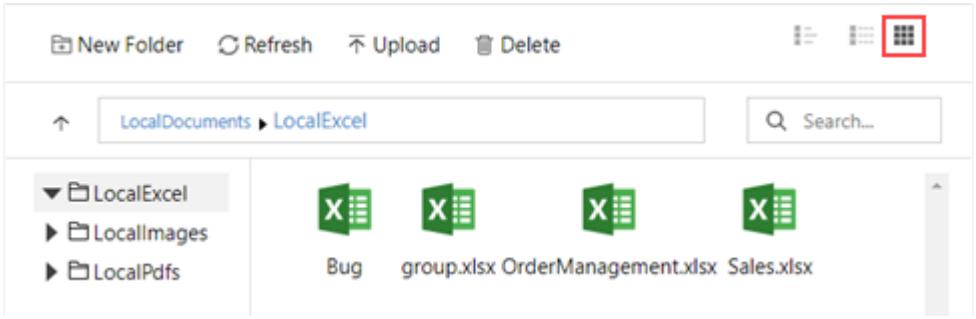
The detail view or grid view displays files in a list along with the columns of information about various properties, such as modified date and file size.

Copyright © GrapeCity, Inc. All rights reserved.



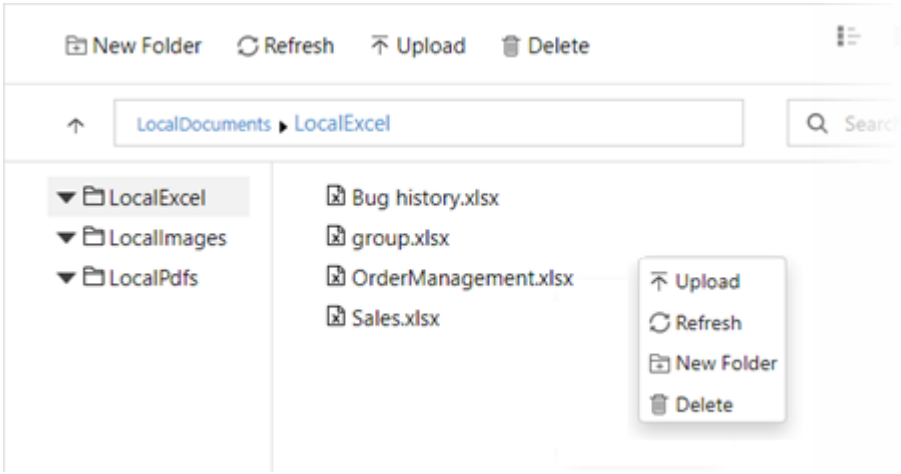
Thumbnail View

The thumbnail view presents the list of thumbnails which represent files.



コンテキストメニュー

A context menu is a pop-up menu that appears on right-click mouse operation. File Manager provides the shortcuts to all the toolbar options in the context menu. You can use the context menu to directly perform operations on a selected folder.

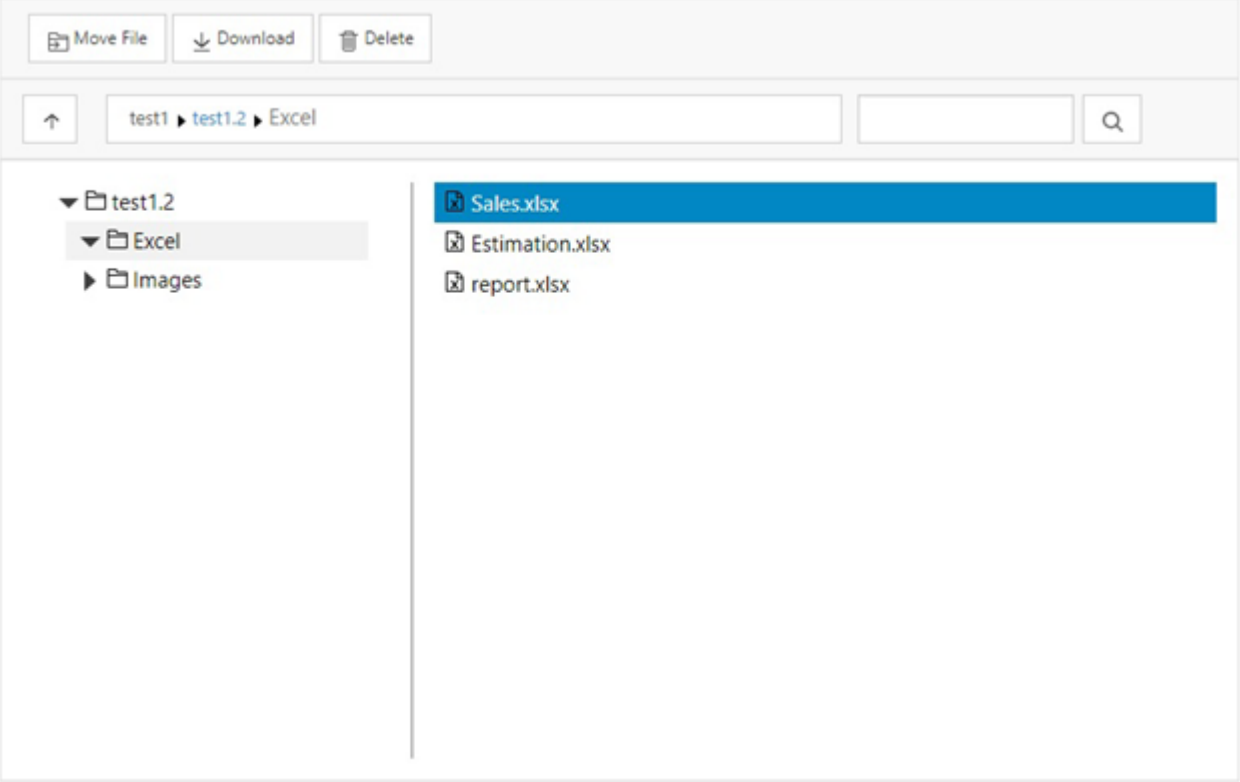


クラウドストレージ内のファイルの管理

File Manager control is represented by the **FileManager** class. The **FileManager** class allows you to host the Url of the machine which hosts the server using the **HostUrl** property and define the initial path of the root folder from cloud container folder using the **InitPath** property. As it supports multiple clouds, it also allows you to choose the type of

Copyright © GrapeCity, Inc. All rights reserved.

cloud service to support according to your requirements using the **CloudType** property which accepts the cloud name from the **CloudTypes** enumeration.



The following example demonstrates the File Manager control integrating with the DropBox cloud and accessing a folder from our DropBox account. Hence, the InitialPath key is set as "C1WebApi/test1" here. If you have already configured a Cloud WebAPI service, use the host url, folder path and type of your cloud service in the following code. In case, you have not configured any Cloud WebAPI service, see Cloud Services to configure a Cloud API.

Note: In order to connect to the cloud services, you need to configure the connection string and include all the other related information in the Web.config file.

View Code

CSHTML

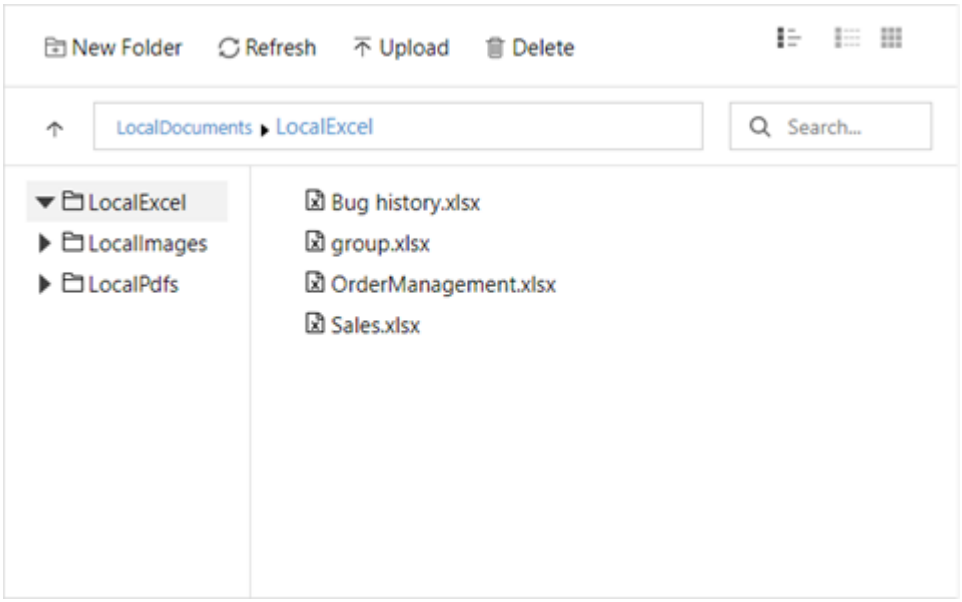
```
@(Html.C1().FileManager().Id("FileManager")
  .HostUrl("http://localhost:61712")
  .InitPath("C1WebApi/test1")
  .CloudType(CloudTypes.DropBox)
)
```

Note: In order to work with FileManager, you require Bootstrap library.

ローカルストレージ内のファイルの管理

In addition to cloud storage, FileManager supports the local storage. It uses the storage APIs of the C1WebAPI to integrate with the local storages and handles all your files stored in the local server. File Manager allows you to choose the local storage using **CloudType** property of the **FileManager** class, which accepts the cloud name from the **CloudTypes** enumeration. Additionally, it also allows you to host the Url of the machine which hosts the server using the **HostUrl** property and define the initial path of the root folder from cloud container folder using the **InitPath** property.

Copyright © GrapeCity, Inc. All rights reserved.



The following example demonstrates the File Manager control integrating with the local storage and accessing a folder from the file system in local storage. Hence, the InitialPath key is set as "ExcelRoot/LocalDocuments" in the example.

View Code

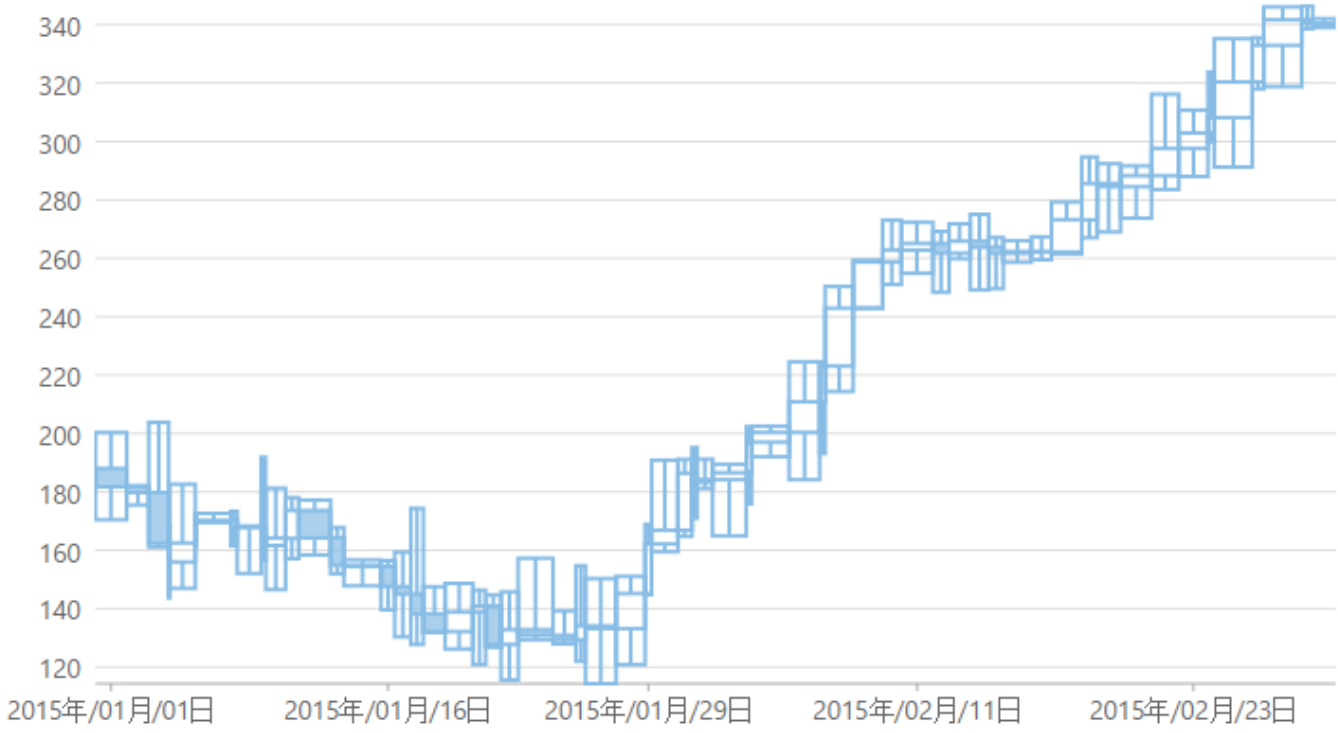
Index.cshtml

```
@(Html.C1().FileManager().Id("FileManager")
.HostUrl("http://localhost:61712")
.InitPath("ExcelRoot/LocalDocuments")
.CloudType(CloudTypes.LocalStorage)
)
```

FinancialChart

ComponentOneのMVC は、最新の分野として複数のFinancial **Charts** を提供し、株価動向の高度な可視化を実現します。これらのチャートで近似曲線、フィルタ、範囲セレクタ、および注釈を使い、最小限のコーディングでデータのトレンドを分析します。さまざまな組み込み機能と多様なチャートタイプにより、**FinancialChart**コントロールはその基礎から金融業界向けに最適化されています。Furthermore, It can be used for performing technical analysis in financial applications using predefined financial indicators and special chart types.

Copyright © GrapeCity, Inc. All rights reserved.



主な特長

FinancialChartコントロールの主な特長は以下のとおりです。

- **チャートタイプ**
  1つのプロパティを設定するだけで、折れ線グラフを横棒グラフなどの他のチャートタイプに変更できます。16種類のチャートタイプから選択できます。

- **複数の領域**
  1つのチャートに複数の系列と領域をデータが重なることなく追加できます。

- **マーカー**
  チャートのテキスト領域にマーカーを表示して、データポイントを表示できます。

- **範囲セレクタ**
  FinancialChartの実行時のデータ表示範囲を調整します。

- **近似曲線**
  データのトレンドを可視化し、予測の問題点を分析します。

- **ツールチップ**
  ツールチップを使用してチャートの値を表示できます。

- **ヘッダーとフッター**
  単純なプロパティを使用して、タイトルやフッターテキストを設定できます。

- **凡例**
  必要に応じて凡例の位置を変更できます。

- **注釈**

Copyright © GrapeCity, Inc. All rights reserved.

重要なニュースまたはイベントをマークするために、FinancialChart上の特定のデータポイントにアタッチできます。

- **Fibonacci Series**
  FinancialChartでさまざまな警告レベルを計算やプロットして株式市場分析で使用できます。
- **移動平均**
  データセット全体の複数サブセットの平均を計算して、その平均の系列でデータを解析します。
- **関数系列**
  Y値を計算するためのいろいろな関数を使用して解析的な財務チャートを作成できます。
- **パラメータ系列**
  XとY値を計算するためのいろいろな関数を使用して解析的な財務チャートを作成できます。
- **ヒットテスト**
  チャート上にホバーされたデータポイントの座標とインデックスを判定できます。
- **インジケータ**
  取引商品の価格と量モメンタムの動向を解析して予想できます。

財務チャートタイプ

要件に応じて、FinancialChartコントロールのタイプを変更できます。チャートタイプを変更するには、FinancialChartコントロールのChartTypeプロパティを設定します。この場合、FinancialChartに複数の系列を追加すると、そのすべての系列が同じチャートタイプになります。

サポートされたチャートタイプ:

面グラフ	**ArmsCandleVolumeチャート**	**ローソク足チャート**	**CandleVolumeチャート**
縦棒グラフ	**ColumnVolumeチャート**	**EquiVolumeチャート**	**平均足チャート**
HighLowOpenCloseチャート	**カギ足チャート**	**折れ線グラフ**	**新値足チャート**
折れ線グラフ(シンボル付き)	**練行足チャート**	**散布図**	**ポイントアンドフィギュアチャート**

コードの場合

Razor

```
.ChartType(C1.Web.Mvc.Finance.Chart.ChartType.Renko)
```

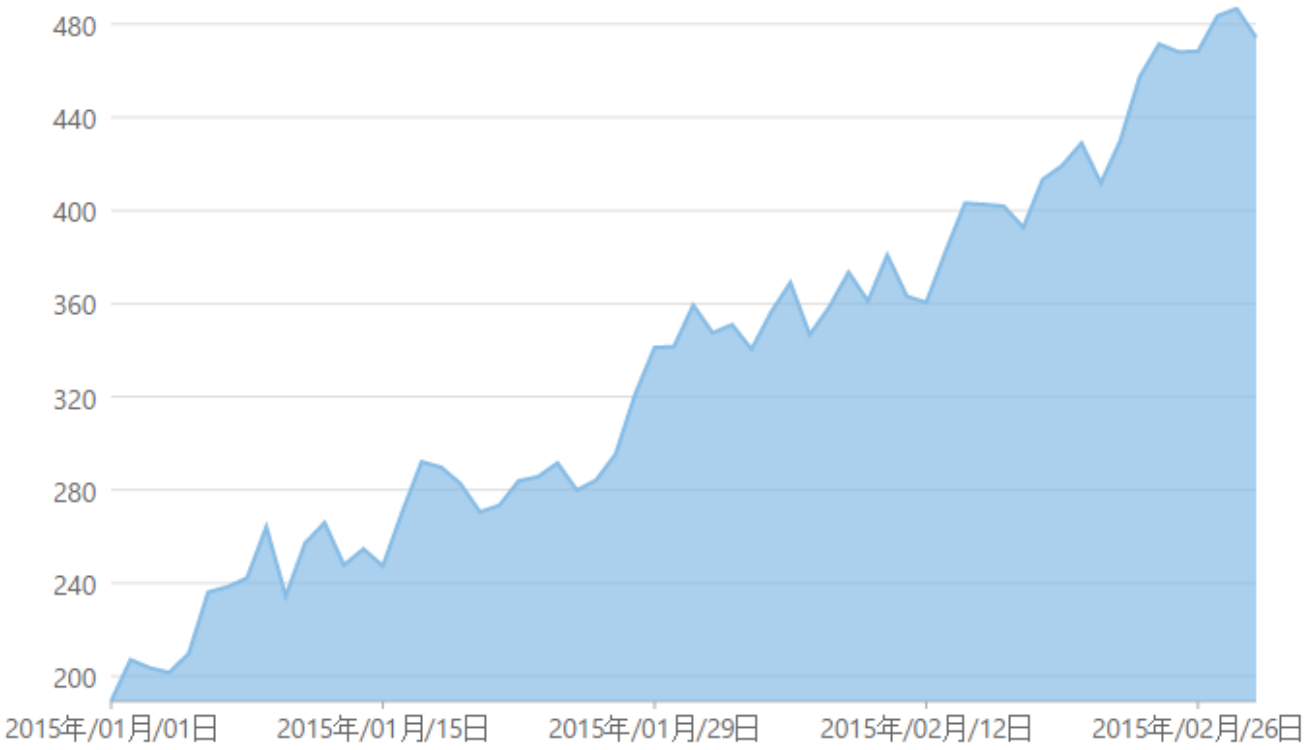
FinancialChartは16種類のチャートタイプをサポートしています。

面グラフ

面グラフは、同じ系列のデータポイントどうしを線で結び、この線の下の領域を塗りつぶして量を示します。新しい系列は、それぞれ前の系列の上に表示されます。系列は、個別に描画することも、積層にして描画することもできます。

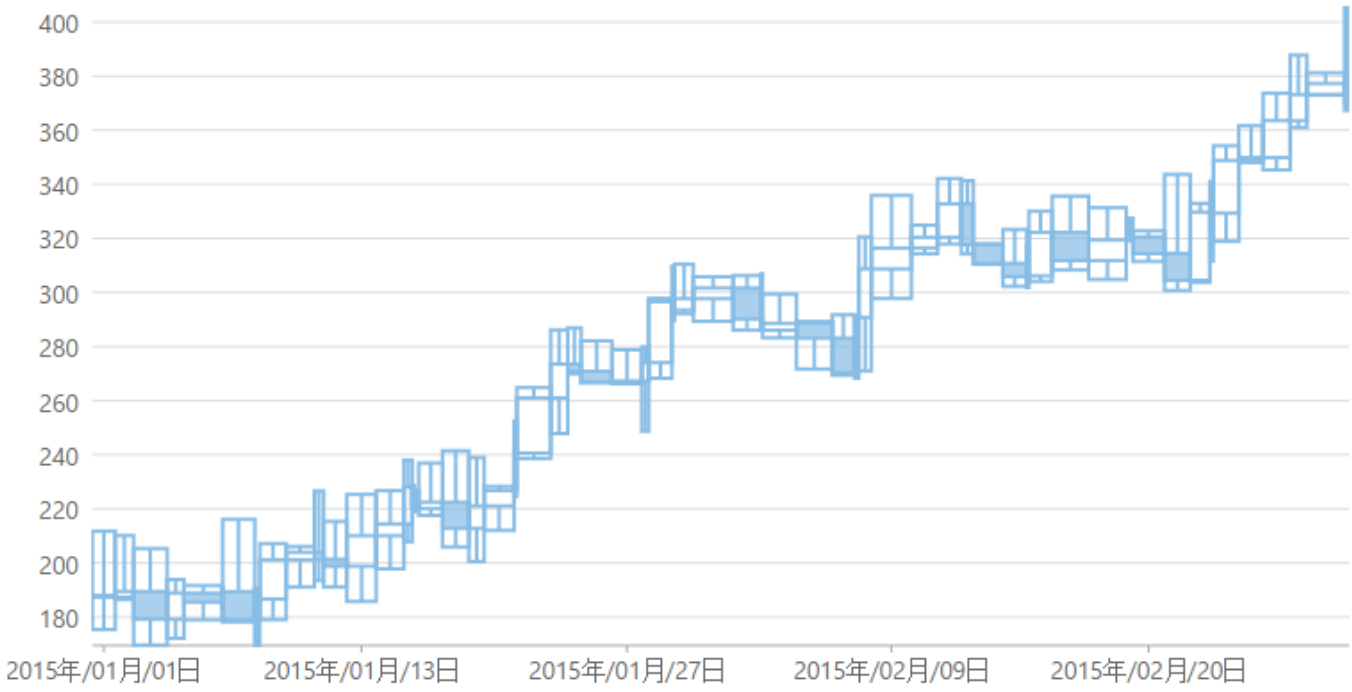
これらのグラフは、関連する属性間の長期にわたるトレンドを示す場合によく使用されます。

Copyright © GrapeCity, Inc. All rights reserved.



ArmsCandleVolumeチャート

Arms CandleVolumeチャートは、EquiVolumeチャートとCandleVolumeチャートを組み合わせたチャートです。



ローソク足チャート

ローソク足チャートは、特定の株の始値、終値、高値、安値を示す株価チャートです。このチャートは、特殊なタイプのHighLowOpenCloseチャートで、始値と終値の関係、および高値と安値の関係を示す場合に使用されます。ローソク足チャートは、価格データ(高値、安値、始値、終値)を使用します。また、始値と終値の関係を追加情報として提供するために、太い

Copyright © GrapeCity, Inc. All rights reserved.

ローソク部分の色とサイズを使用します。たとえば、長い白抜きのローソクは買い圧力を示し、長い塗りつぶされたローソクは売り圧力を示します。

ローソク足チャートの要素

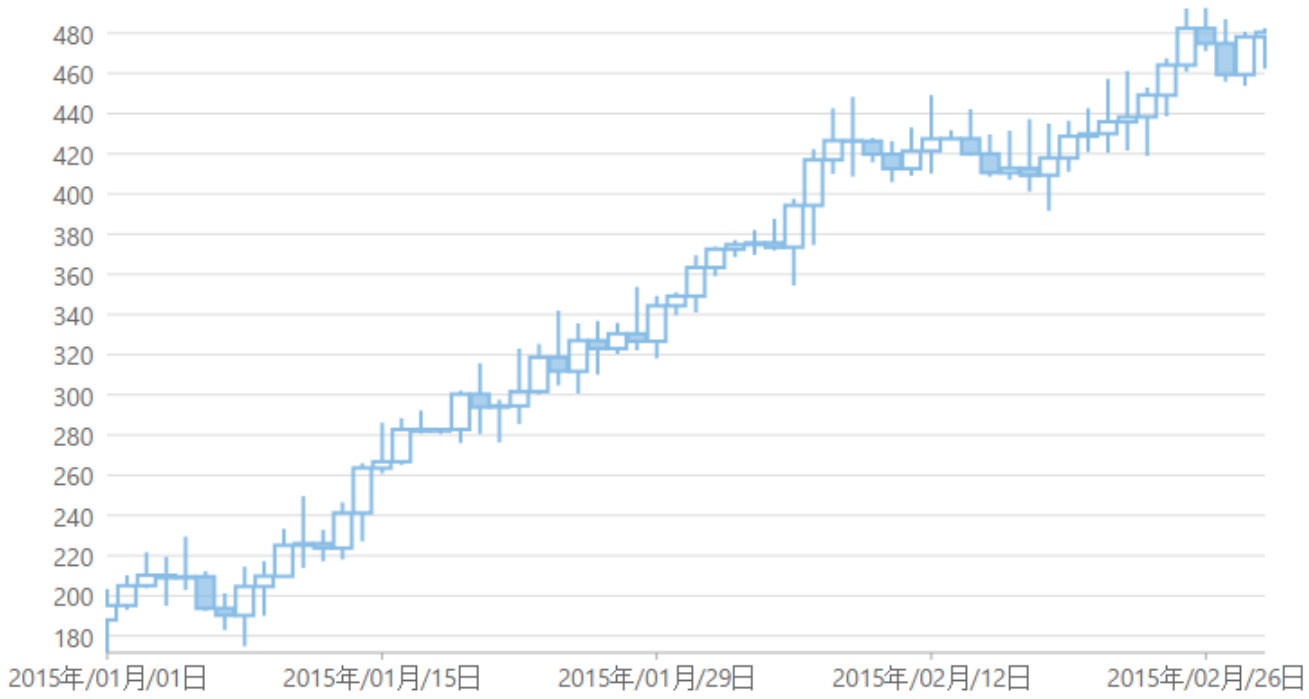
ローソク足チャートは、**ローソク**、**上ヒゲ**、および**下ヒゲ**で構成されています。

- **ローソク**:ローソクの胴体部分(始値と終値の間のバー)は、株価の始値と終値を示しています。
- **上ヒゲと下ヒゲ**:ローソクの上下にある細い線(上ヒゲおよび下ヒゲ)は、株価の高値/安値の範囲を表現しています。
- **空洞の本体**:空洞のローソクは、株価が上昇したこと(終値が始値より高い)を示しています。白抜きのローソクでは、胴体部分の下端が始値を示し、上端が終値を示します。
- **塗りつぶされた本体**:塗りつぶされたローソクは、株価が下降したこと(始値が終値より高い)ことを示しています。塗りつぶされたローソクでは、胴体部分の上端が始値を示し、下端が終値を示します。

ローソク足では、系列内のデータポイントごとに5つの値があります。

- **x**:X軸に沿って日付の位置を決定します。
- **high**:その日の最高値を決定し、Y軸に沿ってローソクの上端としてプロットします。
- **low**:その日の最低値を決定し、Y軸に沿ってローソクの下端としてプロットします。
- **open**:その日の始値を決定します。
- **close**:その日の終値を決定します。

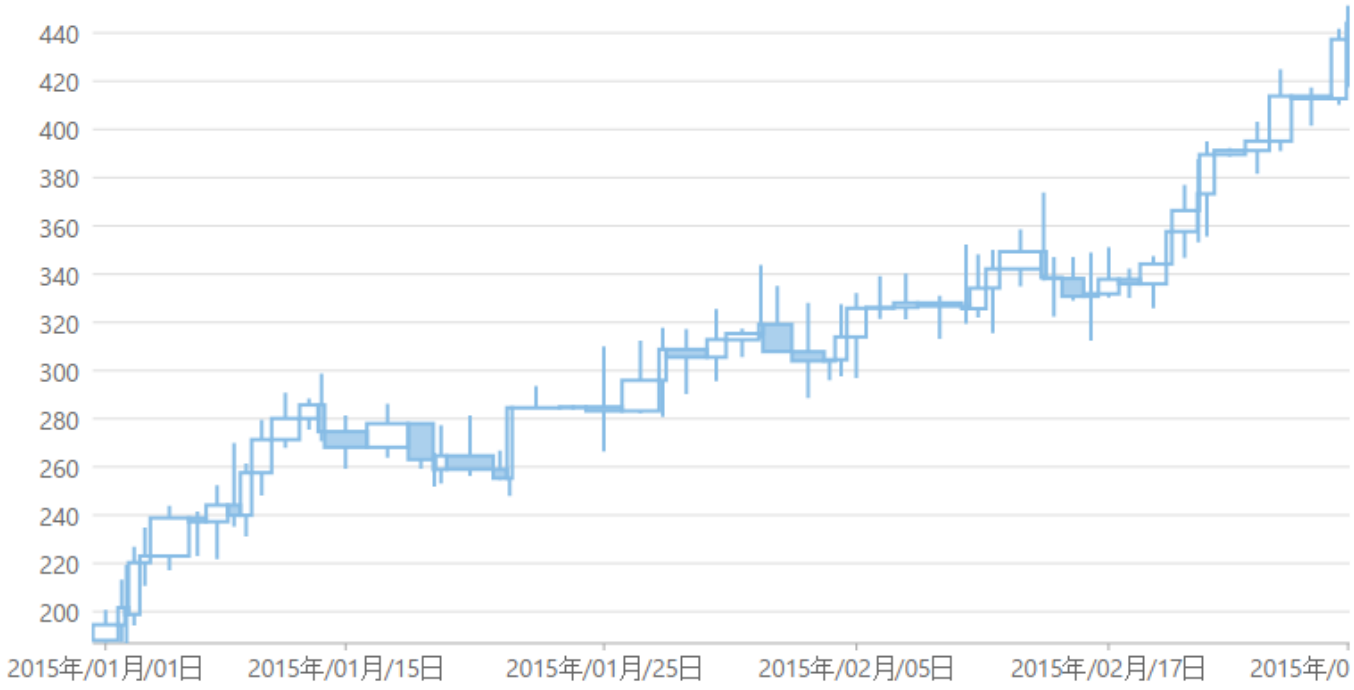
次の図は、株価を表示するローソク足チャートを示しています。



CandleVolumeチャート

CandleVolumeチャートは標準のローソク足チャートと同じです。1つだけ異なる点は、CandleVolumeチャートの各棒の幅がVolumeによって決定されることです。

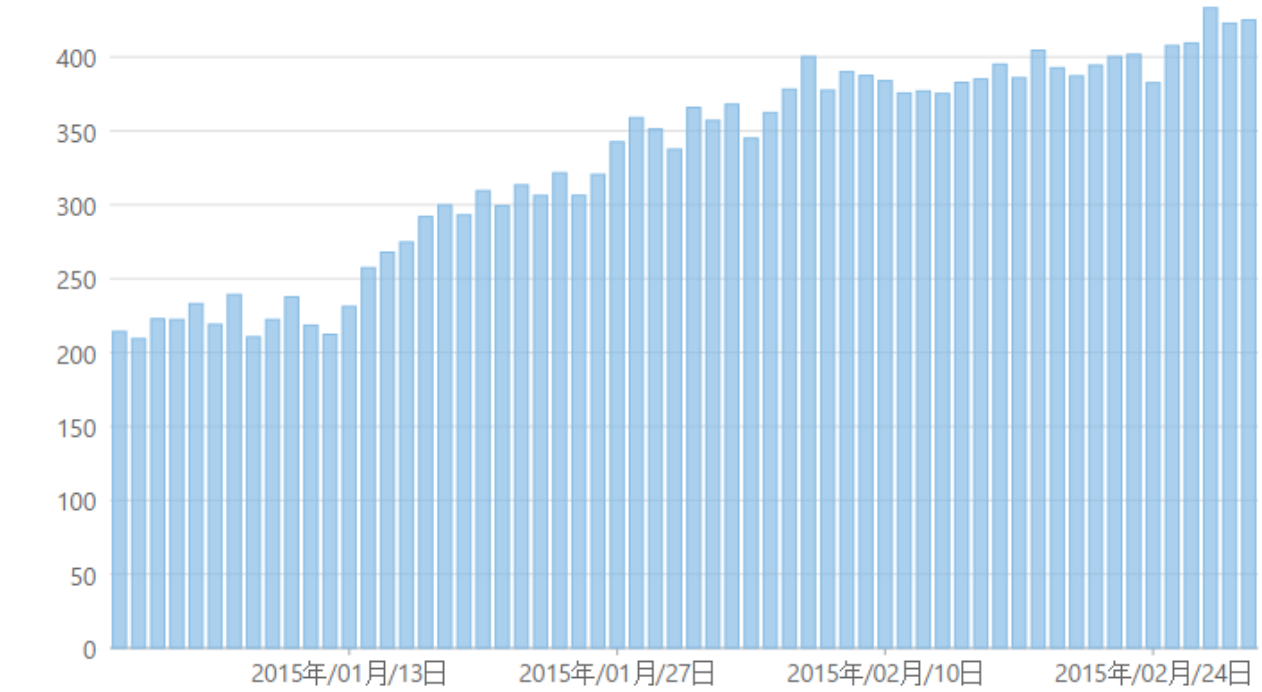
Copyright © GrapeCity, Inc. All rights reserved.



縦棒グラフ

縦棒グラフは、各系列を同じ色と幅の棒形式の集合として表し、各棒の長さは値によって決定されます。新しい系列は、それぞれ前の系列の棒の集合の隣に棒の集合としてプロットされます。縦棒グラフでは、棒は縦向きに置かれます。縦棒グラフは、グループ化するか、積層にすることができます。

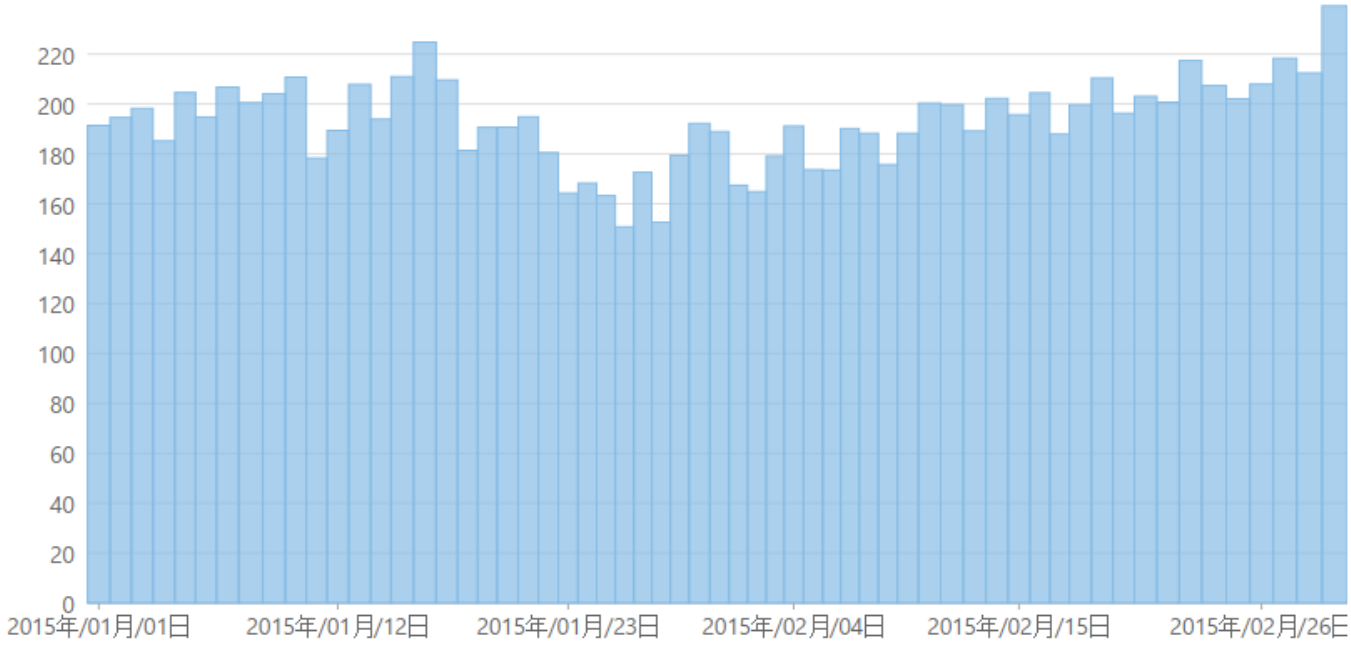
これらのグラフは、年代別や月別などの個別カテゴリにグループ化されたデータを視覚的に表す場合によく使用されます。



ColumnVolumeチャート

縦棒グラフと似ていますが、ColumnVolumeチャートの幅は**Volume**値によって決定されます。

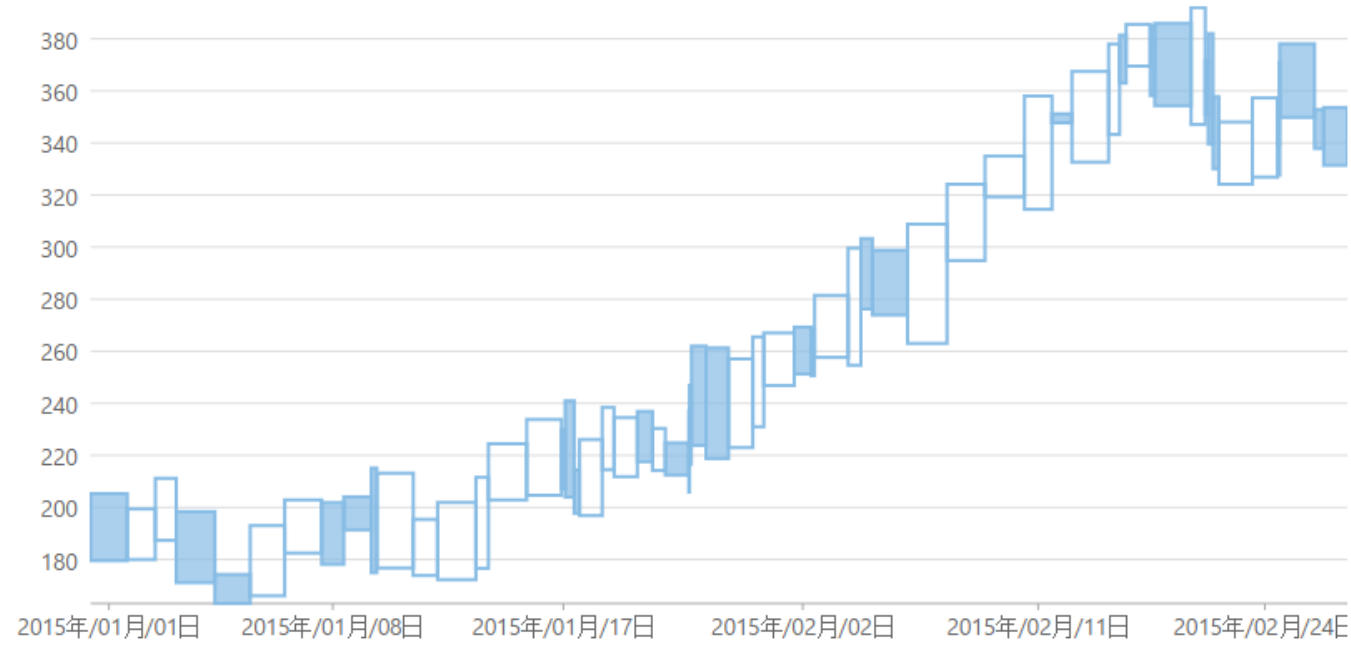
Copyright © GrapeCity, Inc. All rights reserved.



EquiVolumeチャート

EquiVolumeチャートはローソク足チャートと似ていますが、これらのチャートではローソク足がさまざまな幅の四角形のボックス（上ヒゲなし）に置き換えられます。EquiVolumeボックスには、高値および安値の成分と、3つ目の次元として各ボックスの幅を決定する**Volume**が含まれます。色は、終値の数字が直前のボックスの終値よりも上か下かを表します。

EquiVolumeチャートの詳細については、「StockCharts.com」を参照してください。



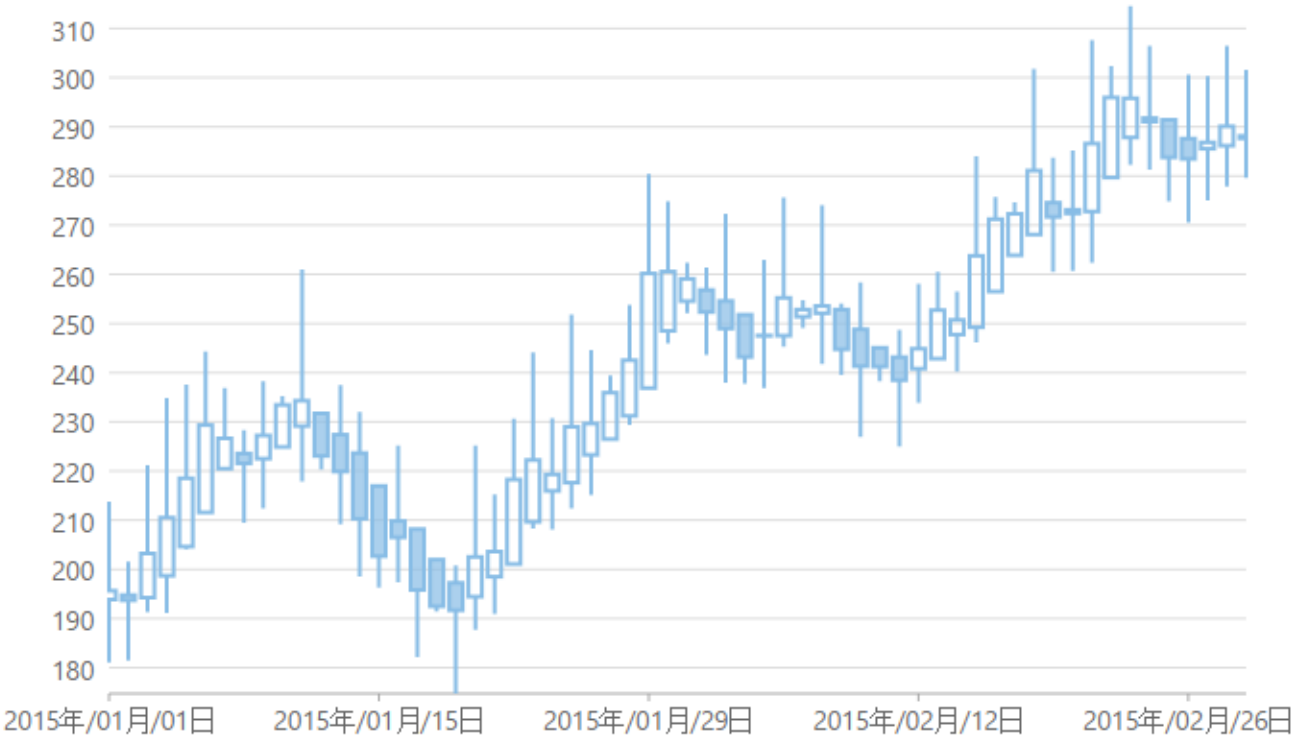
平均足チャート

平均足チャートは日本のローソク足チャートのバリエーションで、ローソク足チャートからノイズを取り除き、移動平均に似た動作をするように設計されています。平均足チャートはローソク足チャートに似ていますが、チャート上のローソクを計算してプロットする方法が違います。平均足チャートの各ローソクは、直前のローソクが示している情報を使用します。これらのチャートでは、データの読み取りや解釈がさらに容易です。

Copyright © GrapeCity, Inc. All rights reserved.

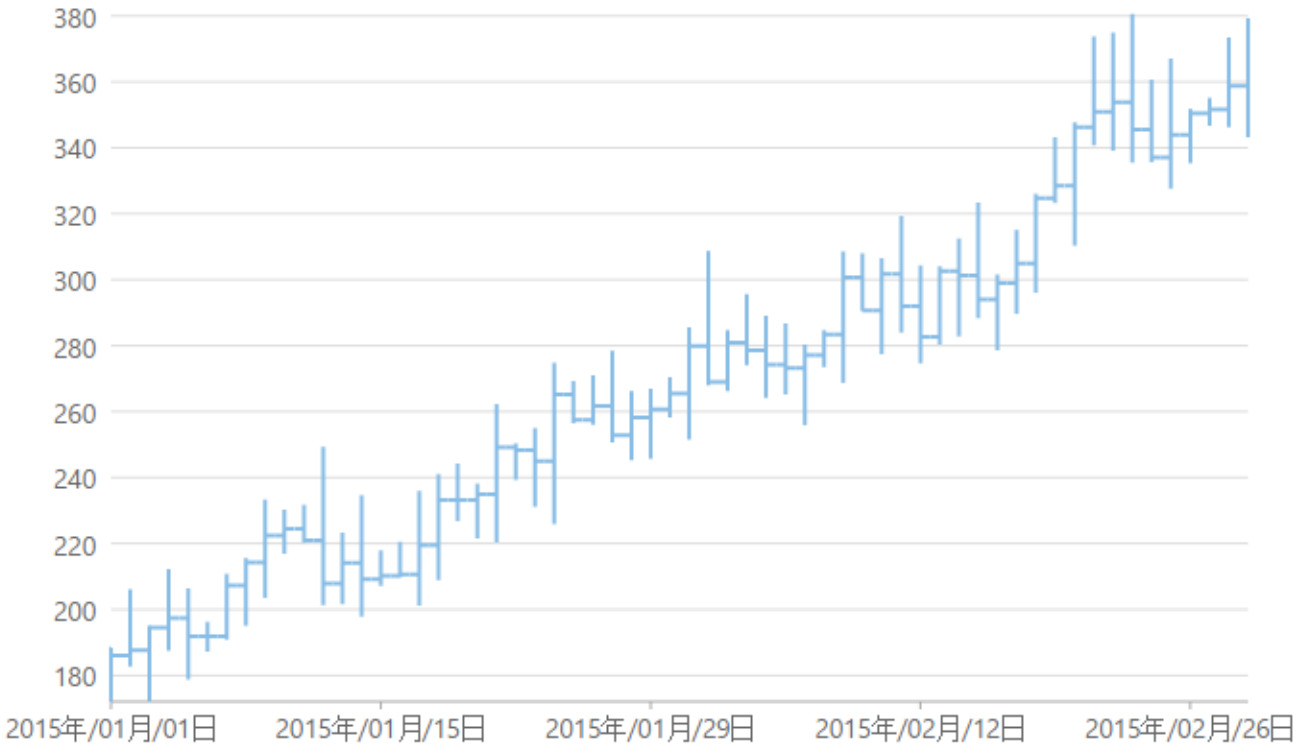
このチャートを使用すると、トレンド、潜在的反転ポイントなどのテクニカル分析パターンを見極めることができます。

平均足チャートの詳細については、「平均足の役割とそれを使用した取引方法」を参照してください。



HighLowOpenCloseチャート

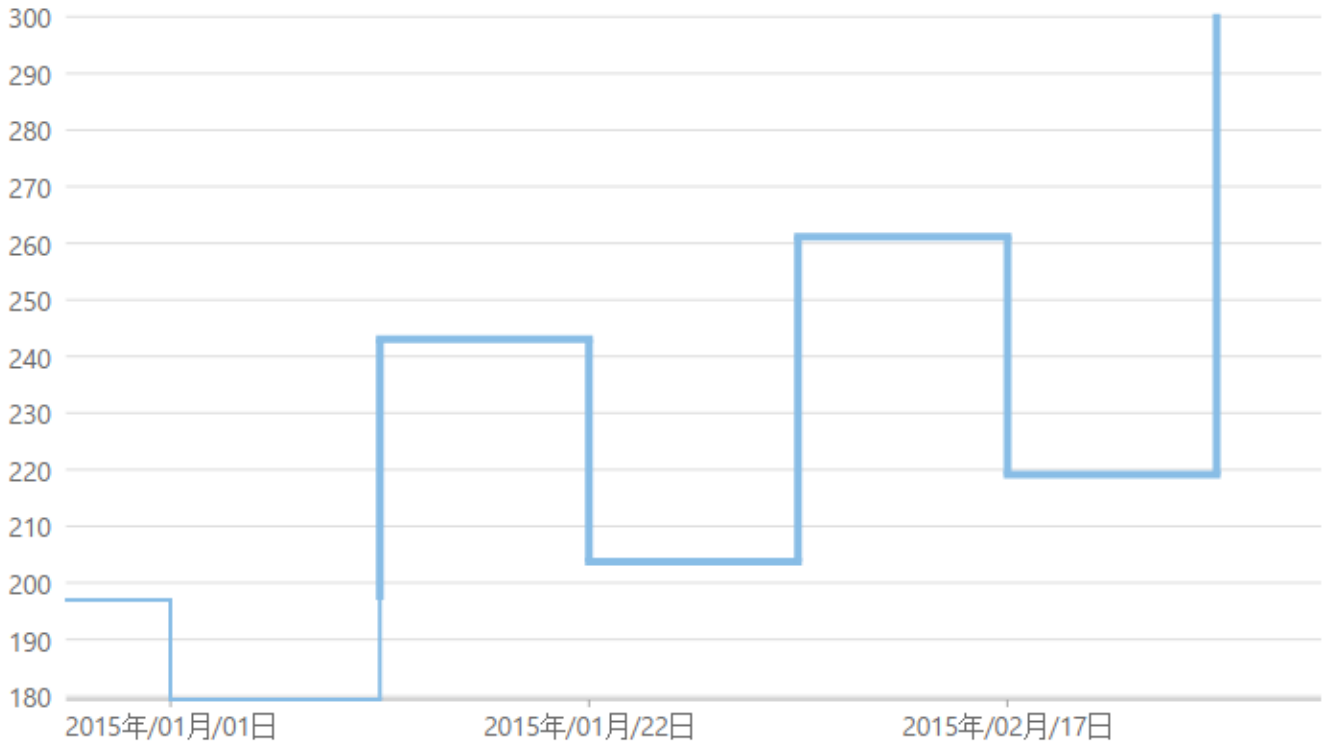
HiLoOpenCloseは、4つの独立した値を組み合わせた株価チャートで、系列内のポイントの高値、安値、始値、および終値を提供します。株の高値および安値を表示するほか、Y2およびY3配列要素がそれぞれ株の始値および終値を表します。



Copyright © GrapeCity, Inc. All rights reserved.

カギ足チャート

カギ足チャートは、一連の垂直線を連結して、需要と供給のトレンドを示します。線の太さと方向は、株価の動きによって決まります。終値が直前の終値と同じ方向に進む場合は、そのカギ足ラインが延長されます。しかし、終値があらかじめ設定した反転幅以上に反転した場合は、次の列に、新しいカギ足ラインが反対方向に引かれます。細い線は、価格が直前の底値を下回ったこと(売り)を、太い線は、価格が直前の高値を上回ったこと(買い)を示します。



折れ線グラフ

折れ線グラフは、データポイントどうしを線で結んで各系列を描画します。面グラフと似ていますが、データポイントどうしを結んだ線の下の領域は塗りつぶされません。系列は、個別に描画することも、積層にして描画することもできます。

これらのグラフは、長期にわたるトレンドやパフォーマンスを示す場合によく使用されます。

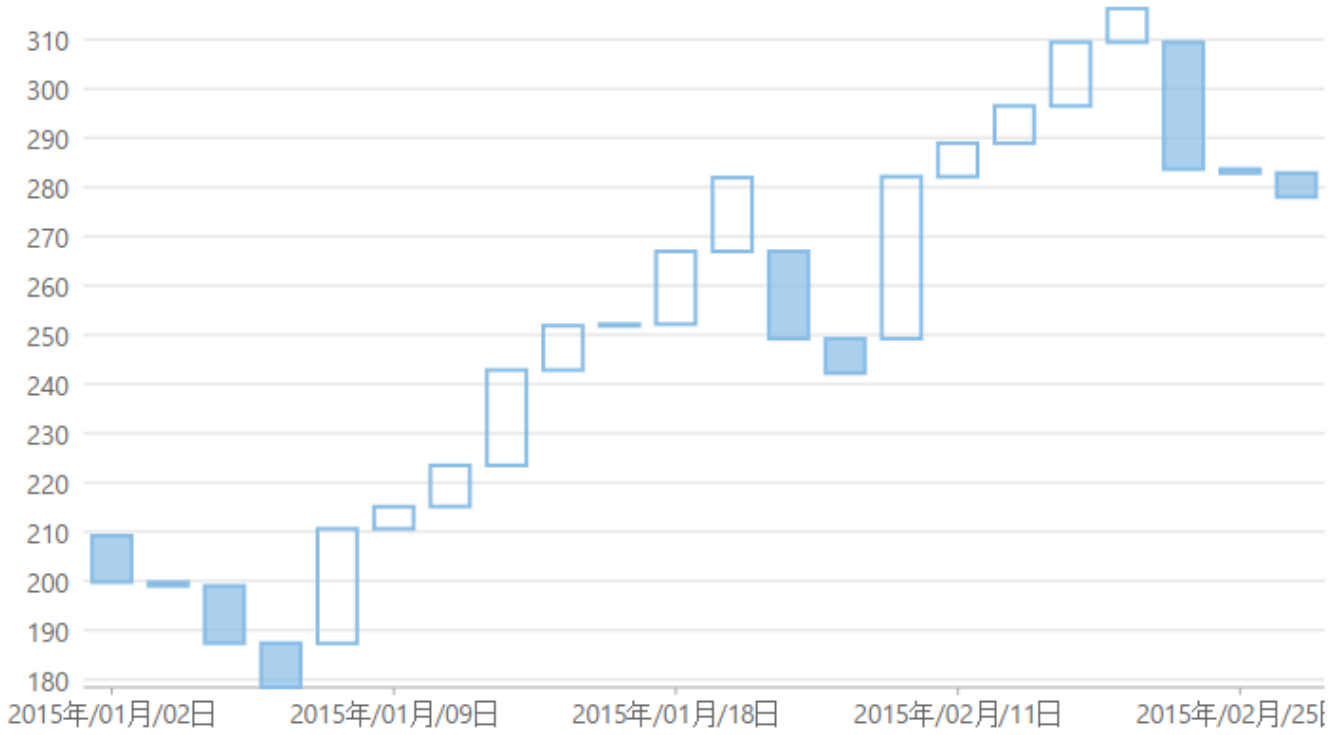
Copyright © GrapeCity, Inc. All rights reserved.



新値足チャート

新値足チャートまたは新値三本足チャートは、垂直のボックスまたはラインを使用して、資産や市場の価格変動を示します。

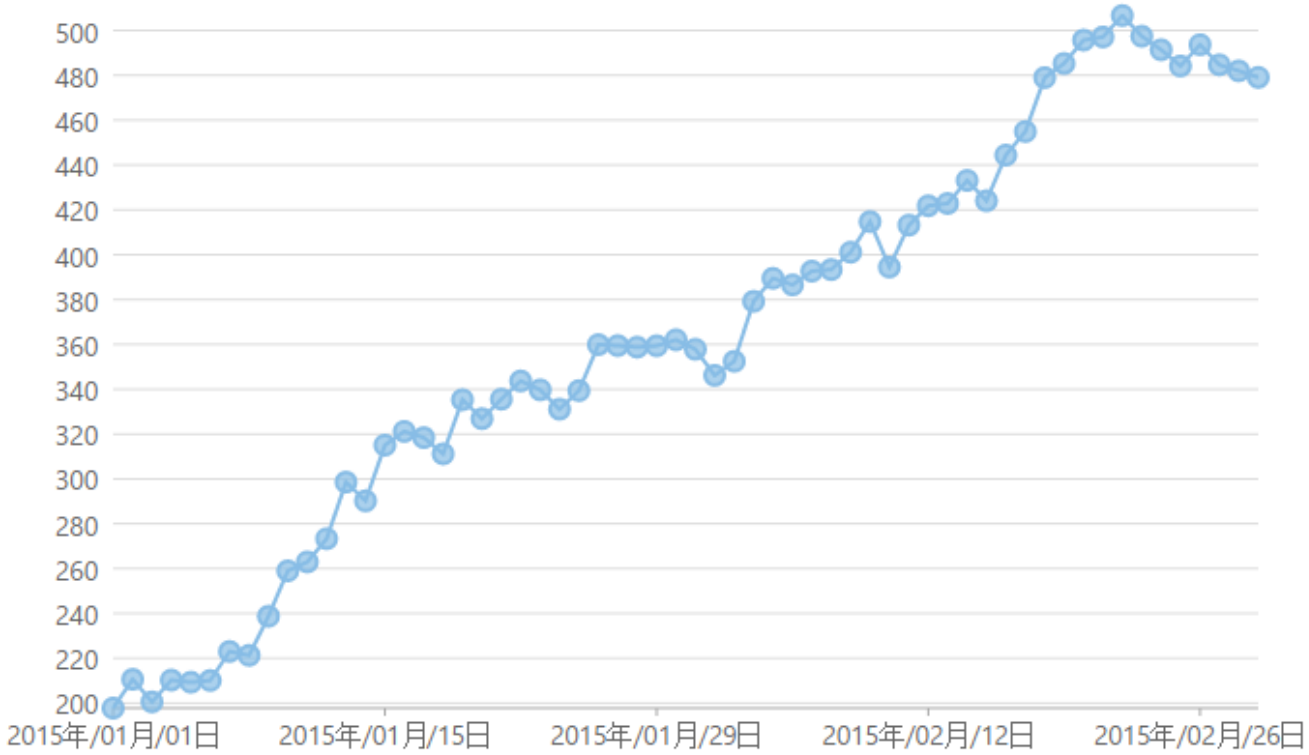
値動きは、ボックスの色とスタイルで表されます。直前のボックスのトレンドが続く値動きは同じ色で表され、反対方向のトレンドを示す値動きは異なる色/スタイルで示されます。反対方向のトレンドは、値が直前n個のボックスまたはラインの最大/最小値を超えた場合にのみ描画されます。この個数は、newLineBreaksオプションで決定されます。



Copyright © GrapeCity, Inc. All rights reserved.

折れ線グラフ(シンボル付き)

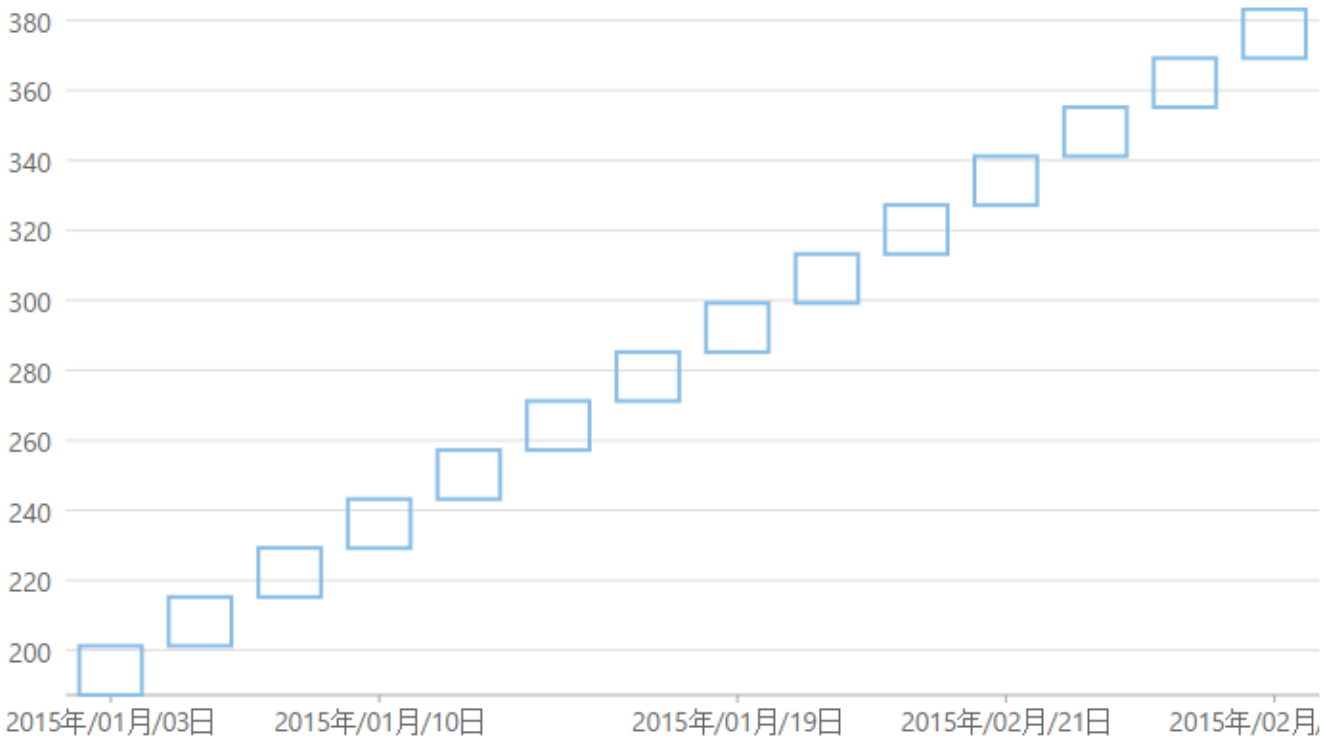
折れ線シンボルグラフは、折れ線グラフに似ていますが、シンボルを使用してデータポイントを表します。これは、さまざまなデータグループ間の値の変化を最も効率よく示す方法です。



練行足チャート

練行足チャートは、一様なサイズのブロックを使用して株価の動きをグラフ化します。価格が、新しいブロックの描画に必要なあらかじめ設定されたboxSizeオプションより大きい値または小さい値に変化すると、次の列に新しいブロックが描画されます。ボックスの色と方向の変化は、トレンドの反転を示します。
練行足チャートは時間を無視し、価格変動だけに着目します。

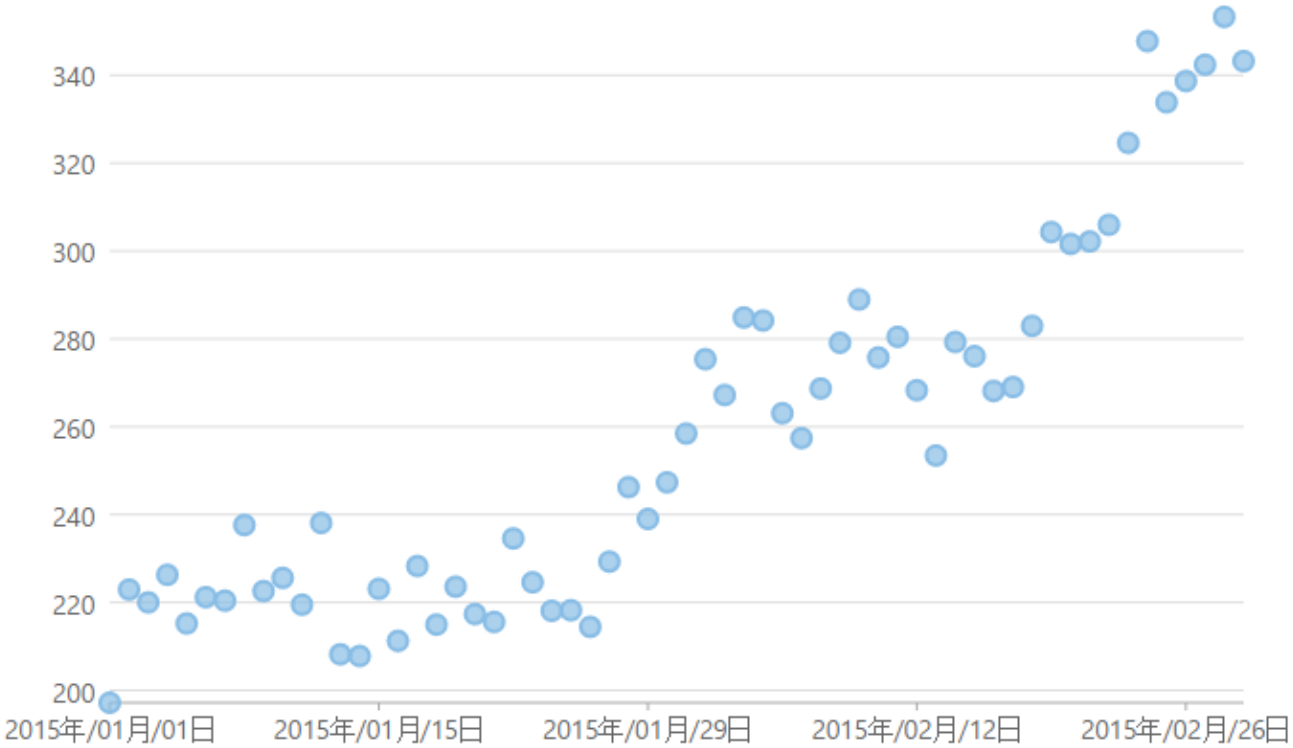
Copyright © GrapeCity, Inc. All rights reserved.



散布図

散布図は、XおよびY軸座標を使用してプロットされた点で系列を表します。X軸とY軸を組み合わせて複数のデータポイントにし、それらを不均一な点の集まりとして表示します。

これらのチャートは、さまざまなXおよびY座標値に対するデータポイント密度のばらつきを判断する場合によく使用されます。



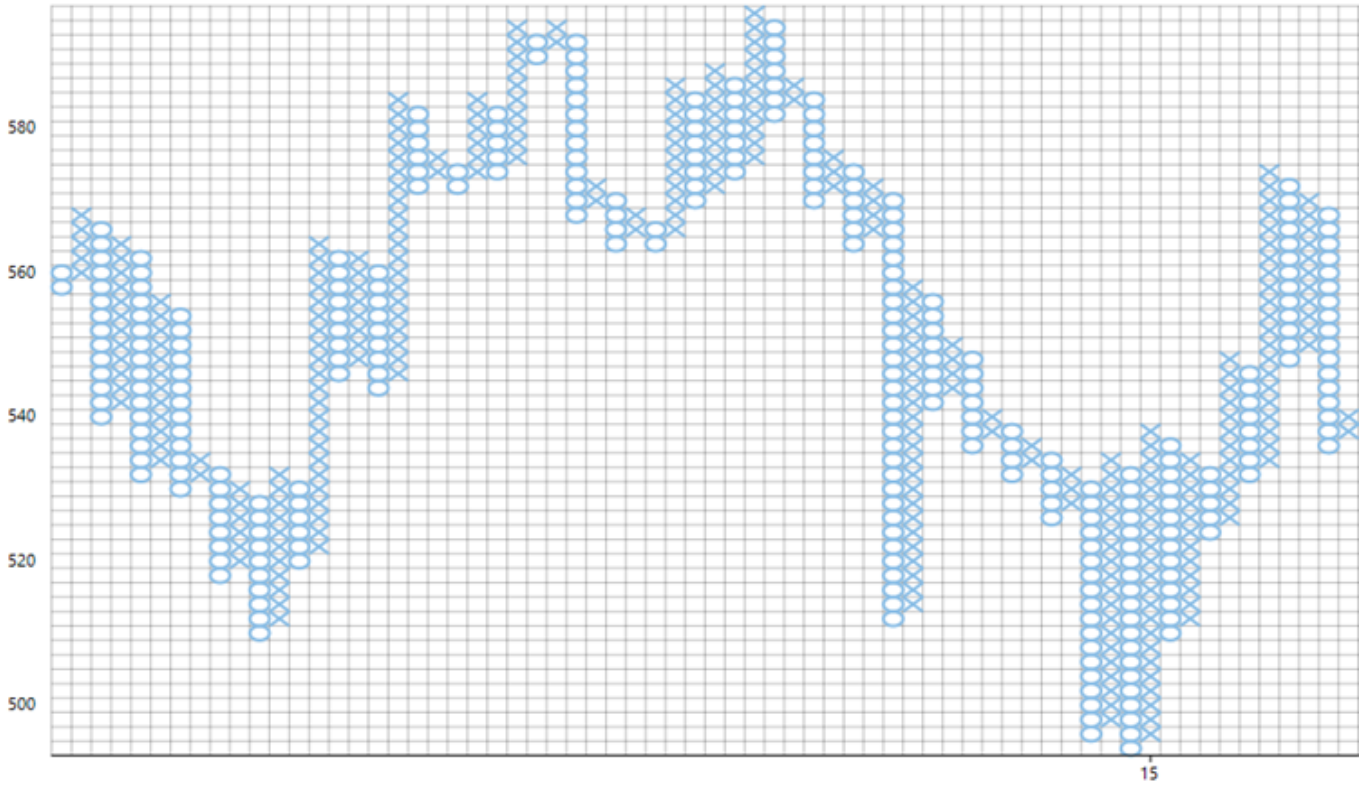
ポイントアンドフィギュアチャート

Copyright © GrapeCity, Inc. All rights reserved.

ポイントアンドフィギュアチャートは、積み重ねた X または O を列にして並べて株価のトレンドを明示します。時間ベースのチャートとは異なり、このチャートは、時間を考慮せずに株価の変動だけに注目します。X が株価の上昇を表し、O が株価の下落を表します。

ボックスサイズを指定できます。ボックスサイズは、1 つの X と O がそれぞれ表す額です。ボックスサイズの額だけ株価が上昇すると、X が記録されます。さらに上昇すると、その上にもう 1 つ X が積み重ねて記録されます。一方、逆の傾向を示すと、ボックスサイズの額だけ株価が下落したところで、次の列に O が記録されます。すべての株価の変動を記録しても冗長なだけなので、反転幅を設定して、株価の反転を記録する 1 列内の最小の X または O の数を指定できます。指定した反転幅になるまでは、株価の反転が発生しても無視されます。

FinancialChart は、ポイントアンドフィギュアチャートの標準スケーリング、固定スケーリング、動的スケーリングをサポートします。標準スケーリングは、ChartCraft よって事前定義された株価範囲テーブルを使用してボックスのサイズを定義します。固定スケーリングでは、エンドユーザーがボックスサイズを指定でき、動的スケーリングでは、計算された ATR 値をボックスサイズとして使用します。



クイックスタート：FinancialChartへのデータの追加

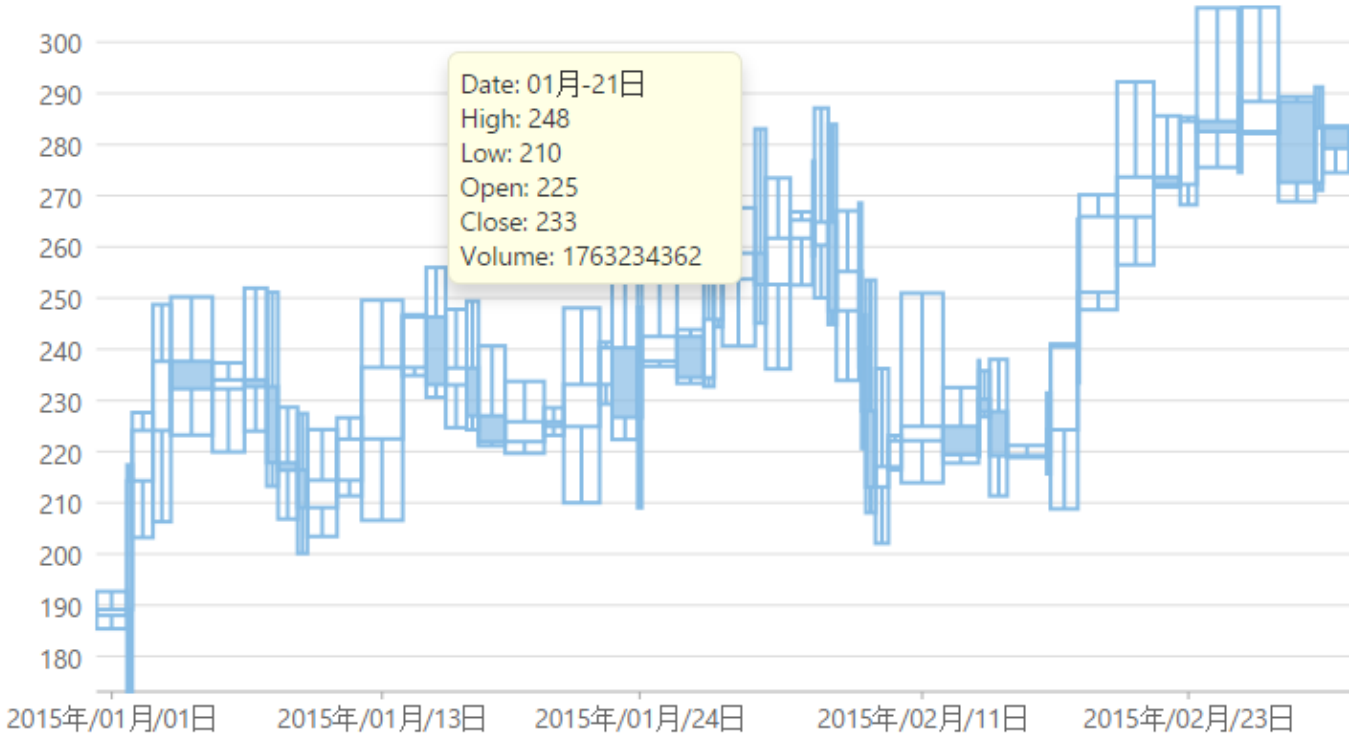
このセクションでは、MVC Webアプリケーションに**FinancialChart**コントロールを追加し、そこにデータを追加する方法について説明します。

このトピックは5つの手順で構成されます。

- **手順1：FinancialChartを使用する許諾をアプリに与える**
- **手順2：アプリケーションへの関連する参照の追加**
- **手順3：FinancialChartコントロールを使用するためのアプリケーションの設定**
- **手順4：FinancialChartのリソースを登録**
- **手順5：FinancialChartのデータソースの作成**
- **手順6：FinancialChartコントロールの追加**
- **手順7：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFinancialChartを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



手順1：FinancialChartを使用する許諾をアプリに与える

1. **ソリューションエクスプローラー**で、プロジェクト名（**MVCFinancialChart**など）をダブルクリックし、**Properties**ノードを展開します。
2. **licenses.licx**ファイルをダブルクリックして開きます。
3. **licenses.licx**ファイルで、'C1.Web.Mvc.LicenseDetector, C1.Web.Mvc' の後で[ENTER]を押し、次のコードを貼り付けます。

licenses.licx

```
C1.Web.Mvc.LicenseDetector, C1.Web.Mvc
C1.Web.Mvc.Finance.LicenseDetector, C1.Web.Mvc.Finance
```

アプリケーションにライセンスを追加する方法については、「ライセンス」を参照してください。

先頭に戻る

手順2：アプリケーションへの関連する参照の追加

ASP.NET

次の手順を実行して、ASP.NET MVC の参照とFinancialChartの参照をプロジェクトに追加します。

1. **ソリューションエクスプローラー**で、**[参照]**を右クリックし、**[参照の追加]**を選択します。
2. **C:\Program Files (x86)\ComponentOne\ASP.NET MVC \bin**を参照します。
3. **C1.Web.Mvc.dll**と**C1.Web.Mvc.Financial.dll**を選択し、**[追加]**をクリックします。
4. **C1.Web.Mvc.dll**と**C1.Web.Mvc.Financial.dll**の**[ローカルコピー]**プロパティを**True**に設定します。

先頭に戻る

手順3：FinancialChartコントロールを使用するためのアプリケーションの設定

Copyright © GrapeCity, Inc. All rights reserved.

ASP.NET

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開し、web.configファイルをダブルクリックして開きます。
2. <system.web.webPages.razor></system.web.webPages.razor>タグ内の<namespaces></namespaces>タグに、以下のマークアップを追加します。

HTML

```
<add namespace="C1.Web.Mvc" />
<add namespace="C1.Web.Mvc.Finance" />
<add namespace="C1.Web.Mvc.Finance.Fluent" />
```

先頭に戻る

手順4：FinancialChartのリソースを登録

ASP.NET MVC FinancialChartコントロールを使用するために必要なリソースを登録するには、次の手順を実行します。

1. **ソリューションエクスプローラー**で、**[ビュー]**→**[共有]**フォルダを開きます。
2. _Layout.cshtmlをダブルクリックして開きます。
3. <head></head>タグの間に次のコードを追加します。

_Layout.cshtml

_Layout.cshtml

```
@Html.C1().Styles()
@Html.C1().Scripts().Basic().Finance()
```

FinancialChartのリソースを登録する方法については、「リソースの登録」を参照してください。

手順5：FinancialChartのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：FData.cs）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FinancialChartコントロールのデータソースになるクラスを定義します。

C# copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MVCFinancialChart.Models
{
    public class FinanceData
    {
        public DateTime X { get; set; }
        public double High { get; set; }
        public double Low { get; set; }
        public double Open { get; set; }
        public double Close { get; set; }
        public double Volume { get; set; }
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

先頭に戻る

手順6：FinancialChartコントロールの追加

FinancialChartコントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`QuickStartController`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
```

5. メソッド`Index()`を次のメソッドに置き換えます。

C# copyCode

```
public ActionResult Index()
{
    var model = GenerateFinanceData();
    return View(model);
}

private List<FData> GenerateFinanceData(int count = 60)
{
    List<FData> financeDatas = new List<FData>() { };

    DateTime startTime = new DateTime(2015, 1, 1);
    var rand = new Random();
    double high, low, open, close, volume;
    for (int i = 0; i < count; i++)
    {
        DateTime dt = startTime.AddDays(i);

        if (i > 0)
            open = financeDatas[i - 1].Close;
        else
            open = 188;

        high = open + rand.NextDouble() * 30;
        low = open - rand.NextDouble() * 20;

        close = low + rand.NextDouble() * (high - low);
        volume = rand.Next();

        financeDatas.Add(new FData { X = dt, High = high, Low = low, Open =
```


Copyright © GrapeCity, Inc. All rights reserved.

```
open, Close = close, Volume = volume });
    }

    return financeDatas;
}
```

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラーQuickStartControllerをダブルクリックして開きます。
2. メソッドQuickStart()内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。
6. QuickStartビューのFinancialChartコントロールを次のようにインスタンス化します。

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FData>

<script type="text/javascript">
    var tooltipContent = function (ht) {
        var item = ht.series.collectionView.items[ht.pointIndex];
        if (item) {
            return 'Date: ' + wijmo.Globalize.format(ht.x, 'MM月-dd日') + '<br/>'
+
                                'High: ' + item.High.toFixed() + '<br/>' +
                                'Low: ' + item.Low.toFixed() + '<br/>' +
                                'Open: ' + item.Open.toFixed() + '<br/>' +
                                'Close: ' + item.Close.toFixed() + '<br/>' +
                                'Volume: ' + item.Volume.toFixed();
        }
    };
</script>

@*FinancialChart コントロールを初期化します。*@
@(Html.C1().FinancialChart()
.Bind(Model)
//グラフの高さと幅を設定します。
.Width(1000)
.BindingX("X")
//グラウのタイプを設定します。
.ChartType(C1.Web.Mvc.Finance.ChartType.ArmsCandleVolume)
.Series(sers =>
{
    sers.Add().Binding("High,Low,Open,Close,Volume");
}).AxisX(x => x.Format("yyyy年MM月dd日"))
.Tooltip(t => t.Content("tooltipContent")))
```

先頭に戻る

手順7:プロジェクトのビルドおよび実行

Copyright © GrapeCity, Inc. All rights reserved.

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**QuickStart/Index**）。

先頭に戻る

財務チャートの使用

This section comprises all the features and auxiliary functionalities offered by Financial Charts.

フィボナッチツール
: Learn how to use Fibonacci tool for performing stock analysis.

Financial Chart インジケータ
: Learn about various indicators supported by MVC Edition.

ラインマーカー
: Learn how to markers to display data points.

移動平均
: Learn how to use MovingAverage to analyze data in FinancialChart.

Financial Chart オーバーレイ
: Learn about different overlays supported in MVC Edition.

範囲セレクタ
: Learn how to display specific range of data on the chart.

近似曲線
: Learn how to add trend lines to the charts to display data trends.

フィボナッチツール

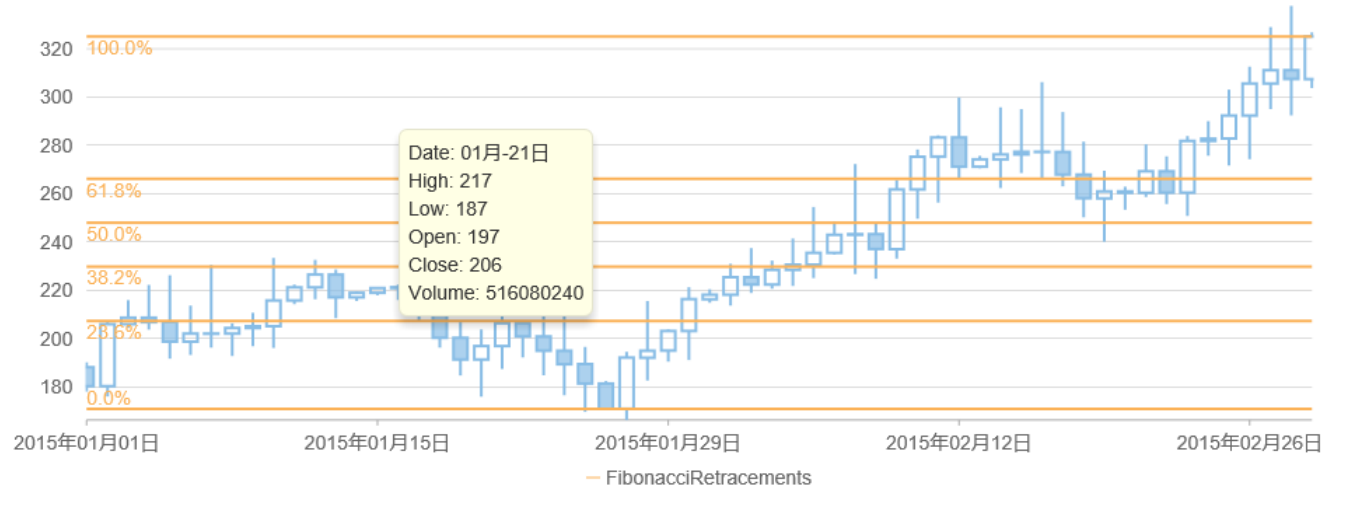
フィボナッチツールにより、財務チャートで便利なさまざまな警告レベルの計算やプロットが可能になります。このトピックでは、FinancialChartでフィボナッチツールを初期化し、株価分析に使用する方法について説明します。

フィボナッチ数列を使用して上昇トレンドや下降トレンドを表示するには、**Uptrend**プロパティの値をtrue（デフォルト）またはfalseに設定します。trueに設定するとUptrend値は上昇トレンド値をプロットし、falseに設定すると下降トレンド値をプロットします。ラベルの位置を設定するには、**LabelPosition**プロパティの値を次のように設定します。

- **Bottom**：ラベルをデータポイントの下に表示します。
- **Center**：ラベルをデータポイントの中央に表示します。
- **Left**：ラベルをデータポイントの左に表示します。
- **None**：データラベルを表示しません。
- **Right**：ラベルをデータポイントの右に表示します。
- **Top**：ラベルをデータポイントの上に表示します。

次の図は、フィボナッチ数列がチャートに追加され、**Uptrend**値がtrue、**LabelPosition**がLeftに設定されたときのFinancialChartを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、フィボナッチ数列とツールチップコンテンツをFinancialChartに追加する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FData>

<script type="text/javascript">
    var tooltipContent = function (ht) {
        var item = ht.series.collectionView.items[ht.pointIndex];
        if (item) {
            return 'Date: ' + wijmo.Globalize
     .format(ht.x, 'MM月-dd日') + '<br/>' +
              'High: ' + item.High.toFixed() + '<br/>' +
              'Low: ' + item.Low.toFixed() + '<br/>' +
              'Open: ' + item.Open.toFixed() + '<br/>' +
              'Close: ' + item.Close.toFixed() + '<br/>'
        }
    };
</script>

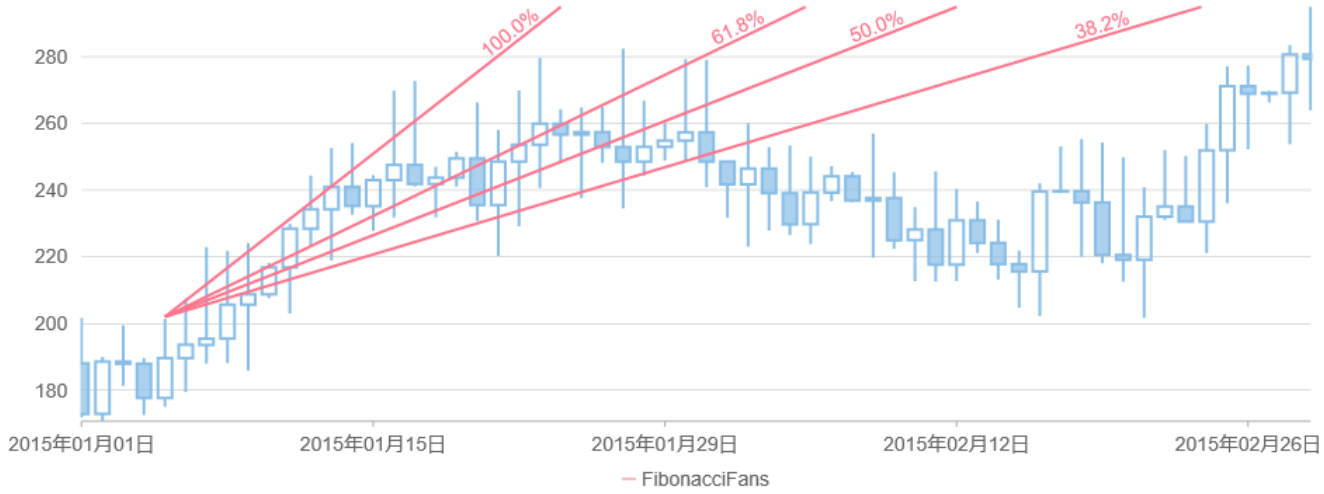
@(Html.C1().FinancialChart()
.Width(1000)
.Bind(Model)
.BindingX("X")
    .ChartType(C1.Web.Mvc.Finance.ChartType.Candlestick)
.Series(sers =>
{
    sers.Add().Binding("High,Low,Open,Close");
    sers.AddFibonacci().Binding("Close").Uptrend(true)
.LabelPosition(C1.Web.Mvc.Chart.LabelPosition.Left)
.Name("Fibonacci Retracements");
})
.AxisX(x => x.Format("yyyy年MM月dd日"))
.Tooltip(t => t.Content("tooltipContent"))
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
)
```

Fibonacci Fans

MVC版のFinancialChartはフィボナッチファン線、フィボナッチリトレースメントポイントに基づくトレンド線、を対応します。上昇するフィボナッチファン線がサポートレベルまたは、逆転ゾーンを予想するには使用されますが、下降ファン線がレジスタンスレベルまたは、逆転を予想します。



コードの場合

Razor

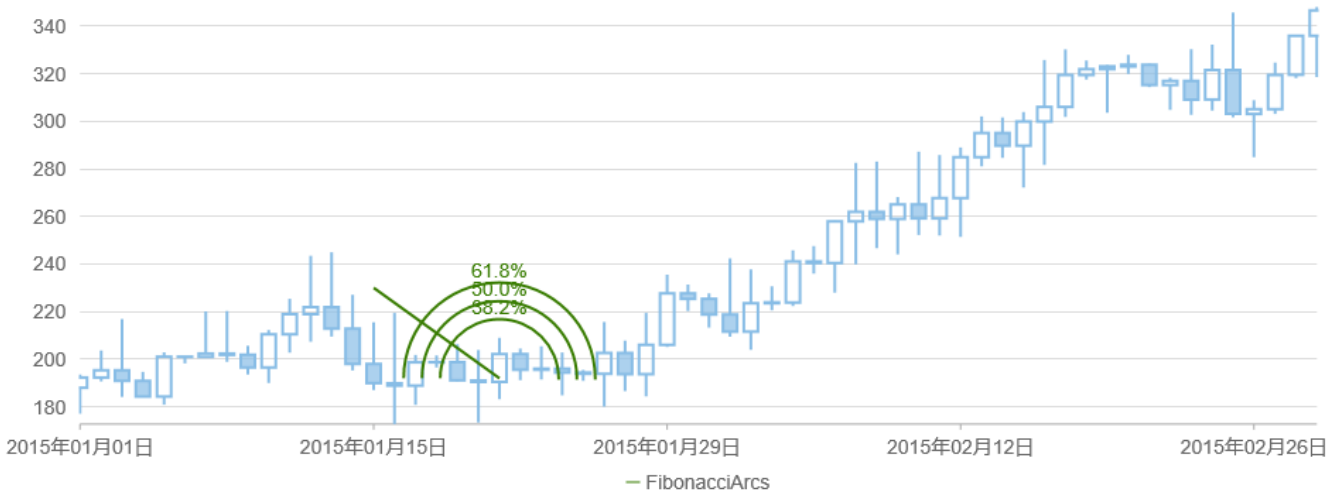
```
@(Html.C1().FinancialChart()
.Bind(Model)
.BindingX("X")
.ChartType(C1.Web.Mvc.Finance.ChartType.Candlestick)
.Series(sers =>
    {
        sers.Add().Binding("High,Low,Open,Close").Name("BOX");
        sers.AddFibonacciFans().Binding("Close")
        .Start(new DataPoint(10, 18.12)).End(new DataPoint(32, 20.53))
        .LabelPosition(C1.Web.Mvc.Chart.LabelPosition.Top);

    })
```

Fibonacci Arcs

フィボナッチアークは、トレンド線から外延する半円を示します。逆転ゾーンまたは、低下後のカウンタートレンドバウンス状態を予想します。

Copyright © GrapeCity, Inc. All rights reserved.



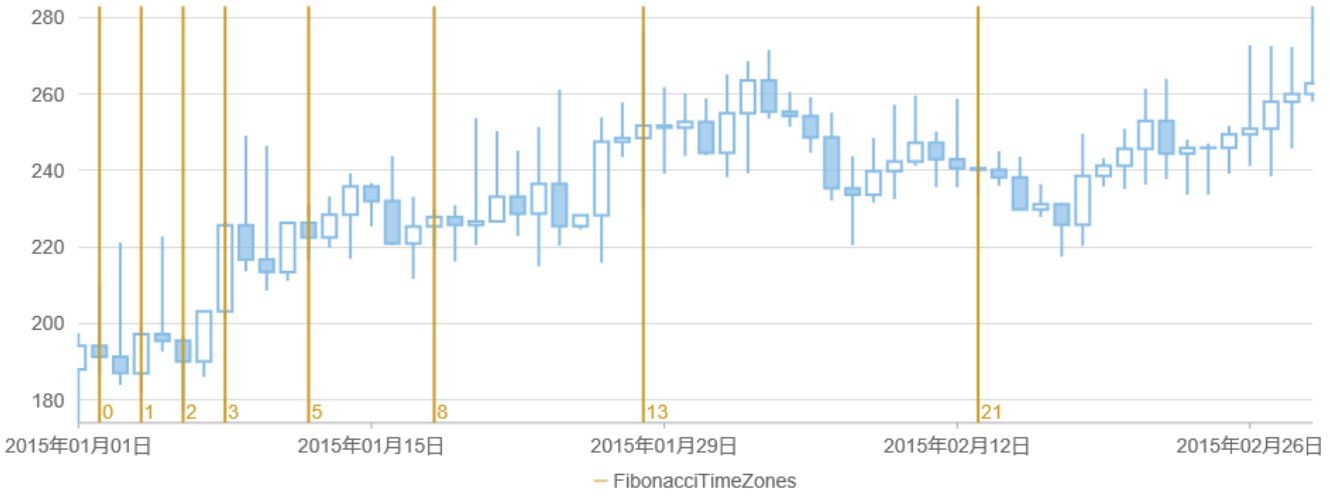
コードの場合

Razor

```
@(Html.C1().FinancialChart()
.Id(demoSettingsModel.ControlId)
.Bind(Model)
.BindingX("X")
.SymbolSize(6)
.ChartType(C1.Web.Mvc.Finance.ChartType.Candlestick)
.Series(sers =>
    {
        sers.Add().Binding("High,Low,Open,Close").Name("BOX");
        sers.AddFibonacciArcs().Binding("Close")
        .Start(new DataPoint(46, 19.75)).End(new DataPoint(54, 17.10))
        .LabelPosition(C1.Web.Mvc.Chart.LabelPosition.Left);
    })
```

Fibonacci Time Zones

フィボナッチタイムゾーンは、フィボナッチ数列に基づき、垂直線として表し、未来の逆転ポイントを予想しに使用されます。MVC版がFinancialChartのフィボナッチタイムゾーンを対応します。



Copyright © GrapeCity, Inc. All rights reserved.

コードの場合

Razor

```
@(Html.C1().FinancialChart()
.Bind(Model)
.BindingX("X")
.ChartType(C1.Web.Mvc.Finance.ChartType.Candlestick)
.Series(sers =>
    {
        sers.Add().Binding("High,Low,Open,Close").Name("BOX");
        sers.AddFibonacciTimeZones().Binding("Close").StartX(0).EndX(3);
    })
```

Financial Chart インジケータ

インジケータおよびテクニカルインジケータは、取引商品の過去から現在までの株価推移と出来高量に基づいて行われた数学計算の結果であるデータポイントのセットです。これらのデータポイントは、金融商品の価格にいくつかの数式を適用することで導出されます。インジケータは、金融市場の方向を予測し、取引決定の基礎を形成することを目的としています。

MVC Editionは、財務アプリケーションで簡単に使用できる、株価チャートコントロール用のテクニカルインジケータをサポートしています。これらの財務インジケータは、チャートパターンとしてプロットされ、テクニカル分析の基礎を形成します。テクニカルインジケータのY軸のスケールは株価または出来高チャートのデータのスケールとは異なるため、一般にインジケータは、元の株価または出来高のデータとは別にプロットされることに注意してください。以下のセクションでは、MVC Editionでサポートされているさまざまな財務チャートインジケータについて説明します。

移動平均収束発散法

FinancialChartのMACD（移動平均収束発散法）インジケータは、トレンドフォロー型のモメンタムインジケータです。資産価格の強さ、方向、期間、およびモメンタムの変化を明らかにします。MACDを使用すると、短期的な価格モメンタムを効率よく見極めることができます。

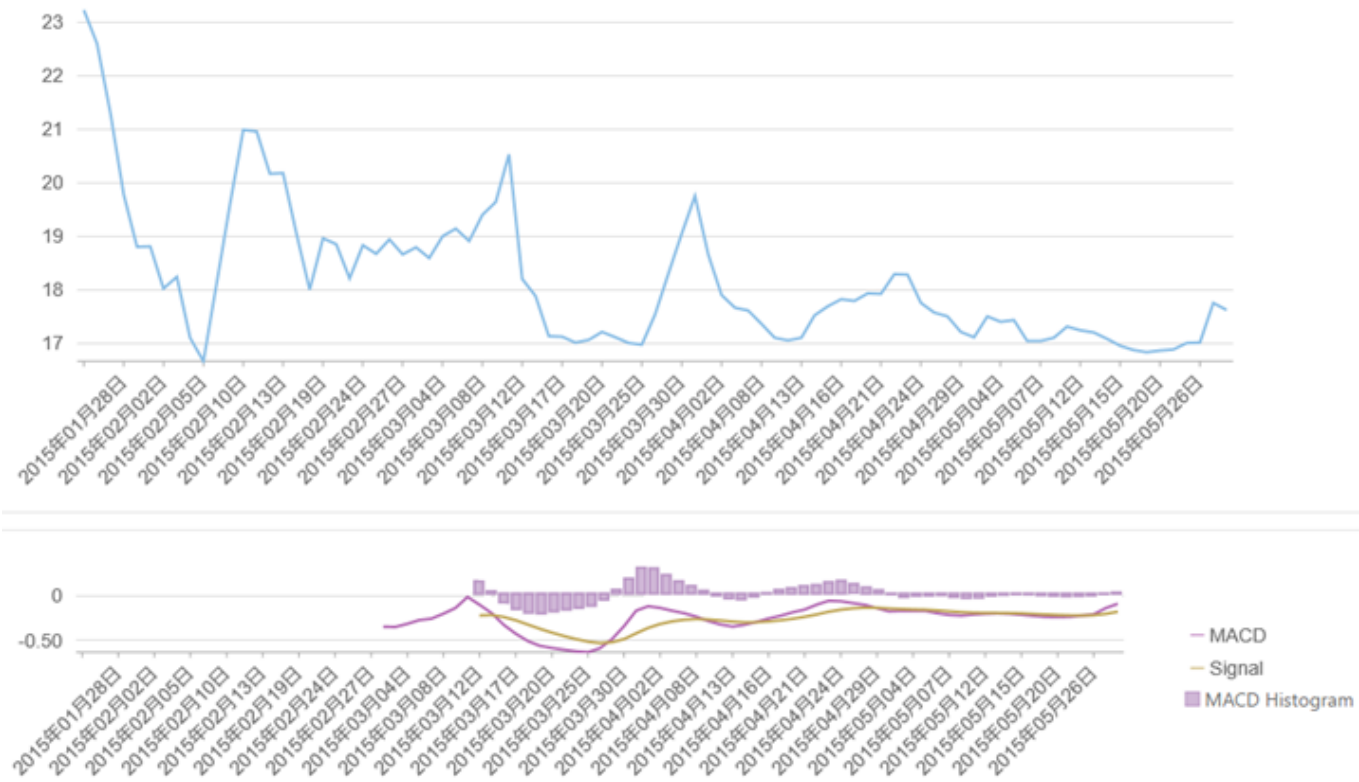
MACDモメンタムオシレータは、26日間の指数移動平均と12日間の指数移動平均の関係を表示します。2つの移動平均が収束、交差、発散するに伴って、MACDオシレータはゼロラインの上下に変動します。オシレータに重ねて「シグナルライン」がプロットされます。このラインはMACDの9日間の指数移動平均を示し、買いシグナルと売りシグナルのトリガとして機能します。

MACDヒストグラム

MACDヒストグラムは、MACDモメンタムオシレータとシグナルラインの差を示すオシレータです。MACDインジケータと同様に、ヒストグラムもゼロラインの上下に変動します。正のヒストグラムは、MACDがシグナルラインより上にあることを示し、MACDがシグナルラインより下になると、負のヒストグラムが生成されます。

次の図は、短期期間を12、長期期間を26、シグナルスムージング期間を9に設定して、MACDインジケータとMACDヒストグラムオシレータを使用したときのFinancialChartを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、MACDインジケータとMACDヒストグラムオシレータをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            double open = Convert.ToDouble(obj["open"].ToString());
            double close = Convert.ToDouble(obj["close"].ToString());
            double volume = Convert.ToDouble(obj["volume"].ToString());
            list.Add(new FinanceData { X = date, High = high,
            Low = low, Open = open, Close = close, Volume = volume });
        }
        _jsonData = list;
        return list;
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
```


Copyright © GrapeCity, Inc. All rights reserved.

```
         their rating on Hercules Technology Growth Capital, Inc.
        (NYSE: HTGX), with a price target of $14. The analysts believe
         that the company's dividend ...",
   "&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
        Enterprise File Sharing&lt;/b&gt;"+
   "&lt;br&gt;Box today announced that it is working with the U.S.
        Department of Justice to deliver file sharing and information
        management to the agency’s workforce. After rigorous assessment,
        Box will receive an agency Authorization to Operate this week,
        allowing the DOJ to leverage the platform across all its component
        agencies.",
   "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
        the ASPCA&lt;/b&gt;"+
   "&lt;br&gt;Box today announced that the ASPCA® , the first humane
        society established in North America, is bringing Box’s cloud
        content sharing and collaboration platform to its employees."
   };

          return _annotationToolTips;
      }
  }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json

```
[
{ "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16,
        "close": 23.23, "volume": 42593223 },
{ "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5,
        "close": 22.6, "volume": 8677164 },
{ "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17,
        "close": 21.3, "volume": 3272512 },
{ "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6,
        "close": 19.78, "volume": 5047364 },
{ "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51,
        "close": 18.8, "volume": 3419482 },
{ "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22,
        "close": 18.81, "volume": 2266439 },
{ "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01,
        "close": 18.02, "volume": 2071168 },
{ "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12,
        "close": 18.24, "volume": 1587435 },
{ "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17,
        "close": 17.1, "volume": 2912224 },
{ "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41,
        "close": 16.66, "volume": 2682187 },
{ "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21,
        "close": 18.12, "volume": 3929164 },
{ "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45,
        "close": 19.6, "volume": 3226650 },
{ "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        "close": 20.99, "volume": 2804409 },
{ "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2,
        "close": 20.96, "volume": 1698365 },
{ "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75,
        "close": 20.17, "volume": 1370320 },
{ "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20,
        "close": 20.18, "volume": 711951 },
{ "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8,
        "close": 19.05, "volume": 2093602 },
{ "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96,
        "close": 18, "volume": 1849490 },
{ "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91,
        "close": 18.96, "volume": 1311518 },
{ "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65,
        "close": 18.85, "volume": 1001692 },
{ "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11,
        "close": 18.21, "volume": 670087 },
{ "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27,
        "close": 18.83, "volume": 759263 },
{ "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47,
        "close": 18.67, "volume": 915580 },
{ "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64,
        "close": 18.94, "volume": 461283 },
{ "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55,
        "close": 18.66, "volume": 617199 },
{ "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65,
        "close": 18.79, "volume": 519605 },
{ "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45,
        "close": 18.59, "volume": 832415 },
{ "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32,
        "close": 19, "volume": 539688 },
{ "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8,
        "close": 19.14, "volume": 486149 },
{ "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7,
        "close": 18.91, "volume": 685659 },
{ "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96,
        "close": 19.4, "volume": 1321363 },
{ "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85,
        "close": 19.64, "volume": 615743 },
{ "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24,
        "close": 20.53, "volume": 2167167 },
{ "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76,
        "close": 18.2, "volume": 6837638 },
{ "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3,
        "close": 17.88, "volume": 1715629 },
{ "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01,
        "close": 17.13, "volume": 1321313 },
{ "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6,
        "close": 17.12, "volume": 1272242 },
{ "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91,
        "close": 17.01, "volume": 530063 },
{ "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17,
        "close": 17.06, "volume": 536427 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{ "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88,
        "close": 17.21, "volume": 1320237 },
{ "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01,
        "close": 17.11, "volume": 509798 },
{ "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82,
        "close": 17, "volume": 962149 },
{ "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82,
        "close": 16.97, "volume": 565673 },
{ "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83,
        "close": 17.54, "volume": 884523 },
{ "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11,
        "close": 18.3, "volume": 705626 },
{ "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4,
        "close": 19.05, "volume": 1151620 },
{ "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4,
        "close": 19.75, "volume": 2020679 },
{ "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55,
        "close": 18.65, "volume": 961078 },
{ "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85,
        "close": 17.9, "volume": 884233 },
{ "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51,
        "close": 17.66, "volume": 605252 },
{ "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53,
        "close": 17.61, "volume": 591988 },
{ "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32,
        "close": 17.36, "volume": 618855 },
{ "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1,
        "close": 17.1, "volume": 761855 },
{ "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17,
        "close": 17.05, "volume": 568373 },
{ "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81,
        "close": 17.1, "volume": 667142 },
{ "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02,
        "close": 17.52, "volume": 870138 },
{ "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5,
        "close": 17.69, "volume": 530456 },
{ "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6,
        "close": 17.82, "volume": 548730 },
{ "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5,
        "close": 17.79, "volume": 446373 },
{ "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52,
        "close": 17.93, "volume": 487017 },
{ "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71,
        "close": 17.92, "volume": 320302 },
{ "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57,
        "close": 18.29, "volume": 644812 },
{ "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18,
        "close": 18.28, "volume": 563879 },
{ "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61,
        "close": 17.75, "volume": 650762 },
{ "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45,
        "close": 17.57, "volume": 437294 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{ "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39,
        "close": 17.5, "volume": 224519 },
{ "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1,
        "close": 17.21, "volume": 495706 },
{ "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17,
        "close": 17.11, "volume": 391040 },
{ "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85,
        "close": 17.5, "volume": 563075 },
{ "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3,
        "close": 17.4, "volume": 253138 },
{ "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09,
        "close": 17.43, "volume": 290935 },
{ "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17,
        "close": 17.04, "volume": 313662 },
{ "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92,
        "close": 17.04, "volume": 360284 },
{ "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91,
        "close": 17.1, "volume": 297653 },
{ "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13,
        "close": 17.31, "volume": 268504 },
{ "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99,
        "close": 17.24, "volume": 376961 },
{ "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06,
        "close": 17.2, "volume": 244617 },
{ "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02,
        "close": 17.08, "volume": 252526 },
{ "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95,
        "close": 16.95, "volume": 274783 },
{ "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76,
        "close": 16.87, "volume": 418513 },
{ "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6,
        "close": 16.83, "volume": 367660 },
{ "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65,
        "close": 16.86, "volume": 297914 },
{ "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79,
        "close": 16.88, "volume": 229346 },
{ "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85,
        "close": 17, "volume": 253279 },
{ "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86,
        "close": 17.01, "volume": 212640 },
{ "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87,
        "close": 17.75, "volume": 857109 },
{ "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44,
        "close": 17.62, "volume": 338482 }
]
```


MACDController.cs

C# copyCode

```
// GET: /MACD/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models;

@model List&lt;FinanceData&gt;

&lt;script&gt;
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
        date = wijmo.Globalize.formatDate(date, 'yyyy年MM月dd日');
        }

        content =
                '&lt;b&gt;' + ht.name + '&lt;/b&gt;&lt;br/&gt;' +
                'Date: ' + date + '&lt;br/&gt;' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
            '&lt;br/&gt;' +
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }

    function chartRendered(sender, args) {
// メインチャートに一致するように指標チャートの最小/最大を設定します。
   var indicatorChart = wijmo.Control.getControl('#indicator'),
        chart = wijmo.Control.getControl('#financeChart');

     if (chart && indicatorChart) {
     indicatorChart.axisX.min = chart.axisX.actualMin;
     indicatorChart.axisX.max = chart.axisX.actualMax;
        }
    };
&lt;/script&gt;

@(Html.C1().FinancialChart()
.Id("financeChart")
.Height(400)
.Width(1000)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
.Bind(Model)
.BindingX("X")
.Series(sers =&gt;
{
    sers.Add().Binding("Close").Name("BOX")
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t =&gt; t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Top)
)

@(Html.C1().FinancialChart()
.Id("indicator")
.Height(200)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =&gt;
{
    sers.AddMacd().Binding("Close").FastPeriod(12).SlowPeriod(26)
.SmoothingPeriod(9).SetMacdLineStyle("#b266b2")
.SetSignalLineStyle("#bfa554").Name("MACD,Signal")
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
    sers.AddMacdHistogram().Binding("Close").FastPeriod(12)
.SlowPeriod(26).SmoothingPeriod(9).Name("MACD Histogram")
.Style(svgb =&gt; svgb.Fill("#CC99CC").Stroke("#CC99CC"))
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t =&gt; t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

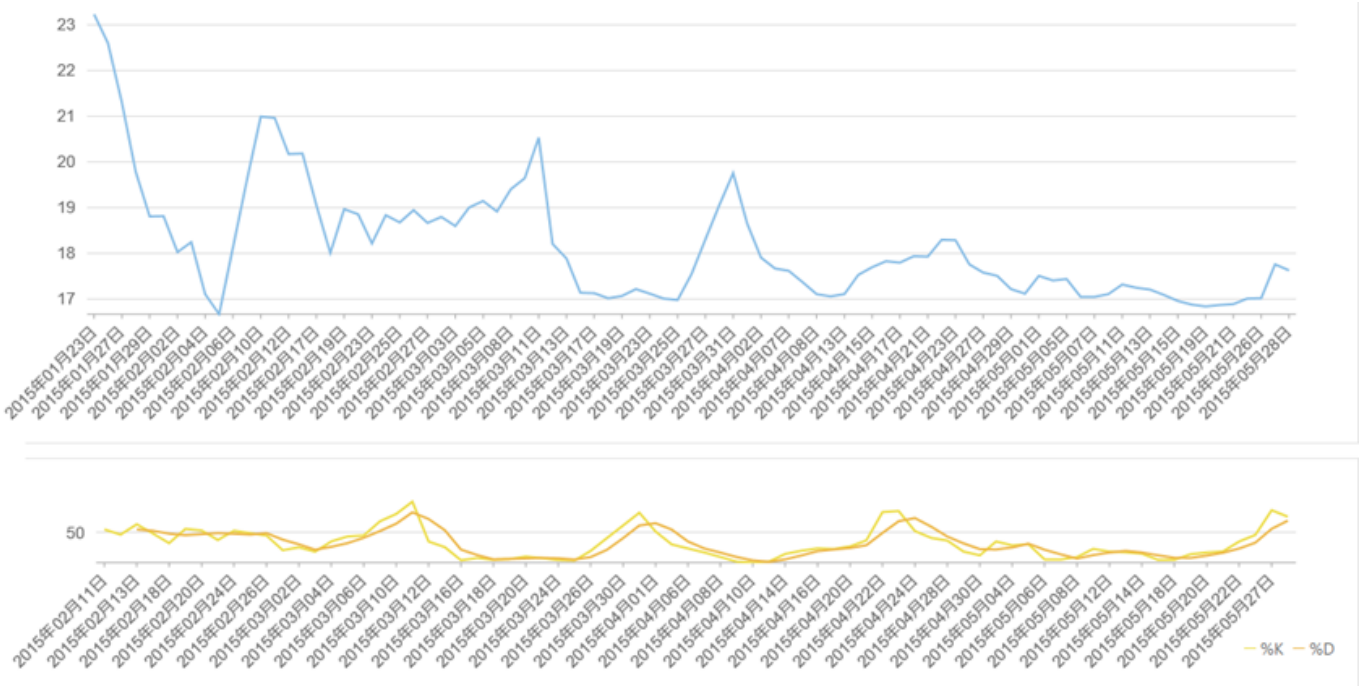
ストキャスティクス

株価チャートのストキャスティクスインジケータは、株価のターニングポイントの前兆となるモメンタムインジケータです。株価のモメンタムを追いかけ、ブルベアセットアップを特定することで将来の反転を予測するために使用できます。

ストキャスティクスインジケータ系列は、短期（デフォルト）、長期、およびフルストキャスティクスに使用できます。長期またはフルストキャスティクスを作成するには、SmoothingPeriodを1より大きな整数値に設定します。また、長期ストキャスティクスでは、通常、SmoothingPeriodには固定値3を使用します。短期ストキャスティクスを作成するには、SmoothingPeriodを整数値1に設定します。

次の図は、K期間を14、D期間を3、平滑期間を1に設定して、ストキャスティクスインジケータをチャートに追加したときのFinancialChartを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、ストキャスティクスインジケータをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
public class BoxData
 {
     private static List&lt;FinanceData&gt; _jsonData;
     public static List&lt;FinanceData&gt; GetDataFromJson()
     {
         if (_jsonData != null)
         {
             return _jsonData;
         }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
              double open = Convert.ToDouble(obj["open"].ToString());
              double close = Convert.ToDouble(obj["close"].ToString());
              double volume = Convert.ToDouble(obj["volume"].ToString());
              list.Add(new FinanceData { X = date, High = high,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            Low = low, Open = open, Close = close, Volume = volume });
        }
        _jsonData = list;
        return list;
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
      their rating on Hercules Technology Growth Capital, Inc.
     (NYSE: HTGX), with a price target of $14. The analysts believe
      that the company's dividend ...",
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    "&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
         Enterprise File Sharing&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that it is working with the U.S.
         Department of Justice to deliver file sharing and information
         management to the agency's workforce. After rigorous assessment,
         Box will receive an agency Authorization to Operate this week,
         allowing the DOJ to leverage the platform across all its component
         agencies.",
    "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
         the ASPCA&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that the ASPCA® , the first humane
         society established in North America, is bringing Box's cloud
         content sharing and collaboration platform to its employees."
    };

            return _annotationToolTips;
        }
    }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json

```
[
{ "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23,
"volume": 42593223 },
{ "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6,
"volume": 8677164 },
{ "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3,
"volume": 3272512 },
{ "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78,
"volume": 5047364 },
{ "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8,
"volume": 3419482 },
{ "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close":
18.81, "volume": 2266439 },
{ "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02,
"volume": 2071168 },
{ "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close":
18.24, "volume": 1587435 },
{ "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1,
"volume": 2912224 },
{ "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66,
"volume": 2682187 },
{ "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close":
18.12, "volume": 3929164 },
{ "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6,
"volume": 3226650 },
{ "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99,
"volume": 2804409 },
{ "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96,
"volume": 1698365 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{ "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close":
20.17, "volume": 1370320 },
{ "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18,
"volume": 711951 },
{ "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05,
"volume": 2093602 },
{ "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18,
"volume": 1849490 },
{ "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close":
18.96, "volume": 1311518 },
{ "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85,
"volume": 1001692 },
{ "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21,
"volume": 670087 },
{ "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83,
"volume": 759263 },
{ "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close":
18.67, "volume": 915580 },
{ "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94,
"volume": 461283 },
{ "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66,
"volume": 617199 },
{ "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close":
18.79, "volume": 519605 },
{ "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close":
18.59, "volume": 832415 },
{ "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19,
"volume": 539688 },
{ "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14,
"volume": 486149 },
{ "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91,
"volume": 685659 },
{ "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4,
"volume": 1321363 },
{ "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64,
"volume": 615743 },
{ "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close":
20.53, "volume": 2167167 },
{ "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2,
"volume": 6837638 },
{ "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88,
"volume": 1715629 },
{ "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13,
"volume": 1321313 },
{ "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12,
"volume": 1272242 },
{ "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01,
"volume": 530063 },
{ "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06,
"volume": 536427 },
{ "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close":
17.21, "volume": 1320237 },
{ "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close":
```


Copyright © GrapeCity, Inc. All rights reserved.

```
17.11, "volume": 509798 },
{ "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17,
"volume": 962149 },
{ "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close":
16.97, "volume": 565673 },
{ "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close":
17.54, "volume": 884523 },
{ "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3,
"volume": 705626 },
{ "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05,
"volume": 1151620 },
{ "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75,
"volume": 2020679 },
{ "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close":
18.65, "volume": 961078 },
{ "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9,
"volume": 884233 },
{ "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close":
17.66, "volume": 605252 },
{ "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61,
"volume": 591988 },
{ "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close":
17.36, "volume": 618855 },
{ "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1,
"volume": 761855 },
{ "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05,
"volume": 568373 },
{ "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1,
"volume": 667142 },
{ "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52,
"volume": 870138 },
{ "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69,
"volume": 530456 },
{ "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82,
"volume": 548730 },
{ "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79,
"volume": 446373 },
{ "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close":
17.93, "volume": 487017 },
{ "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close":
17.92, "volume": 320302 },
{ "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close":
18.29, "volume": 644812 },
{ "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close":
18.28, "volume": 563879 },
{ "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75,
"volume": 650762 },
{ "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close":
17.57, "volume": 437294 },
{ "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5,
"volume": 224519 },
{ "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 495706 },
{ "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11,
"volume": 391040 },
{ "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5,
"volume": 563075 },
{ "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4,
"volume": 253138 },
{ "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close":
17.43, "volume": 290935 },
{ "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04,
"volume": 313662 },
{ "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close":
17.04, "volume": 360284 },
{ "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1,
"volume": 297653 },
{ "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close":
17.31, "volume": 268504 },
{ "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close":
17.24, "volume": 376961 },
{ "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2,
"volume": 244617 },
{ "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close":
17.08, "volume": 252526 },
{ "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close":
16.95, "volume": 274783 },
{ "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close":
16.87, "volume": 418513 },
{ "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83,
"volume": 367660 },
{ "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86,
"volume": 297914 },
{ "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88,
"volume": 229346 },
{ "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17,
"volume": 253279 },
{ "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close":
17.01, "volume": 212640 },
{ "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close":
17.75, "volume": 857109 },
{ "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close":
17.62, "volume": 338482 }
]
```


StochasticOscillatorController.cs

C# copyCode

```
// GET: /StochasticOscillator/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```


Copyright © GrapeCity, Inc. All rights reserved.

Razor copyCode

```
@using MVCFinancialChart.Models

@model List&lt;FinanceData&gt;

&lt;script&gt;
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
            date = wijmo.Globalize.formatDate(date, 'yyyy年MM月dd日');
        }

        content =
                '&lt;b&gt;' + ht.name + '&lt;/b&gt;&lt;br/&gt;' +
                'Date: ' + date + '&lt;br/&gt;' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
            '&lt;br/&gt;' +
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }

    function chartRendered(sender, args) {
        // メインチャートに一致するように指標チャートの最小/最大を設定します。
        var indicatorChart = wijmo.Control.getControl('#indicator'),
            chart = wijmo.Control.getControl('#financeChart');

        if (chart && indicatorChart) {
            indicatorChart.axisX.min = chart.axisX.actualMin;
            indicatorChart.axisX.max = chart.axisX.actualMax;
        }
    };
&lt;/script&gt;

@(Html.C1().FinancialChart()
.Id("financeChart")
.Height(400)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =&gt;
```


Copyright © GrapeCity, Inc. All rights reserved.

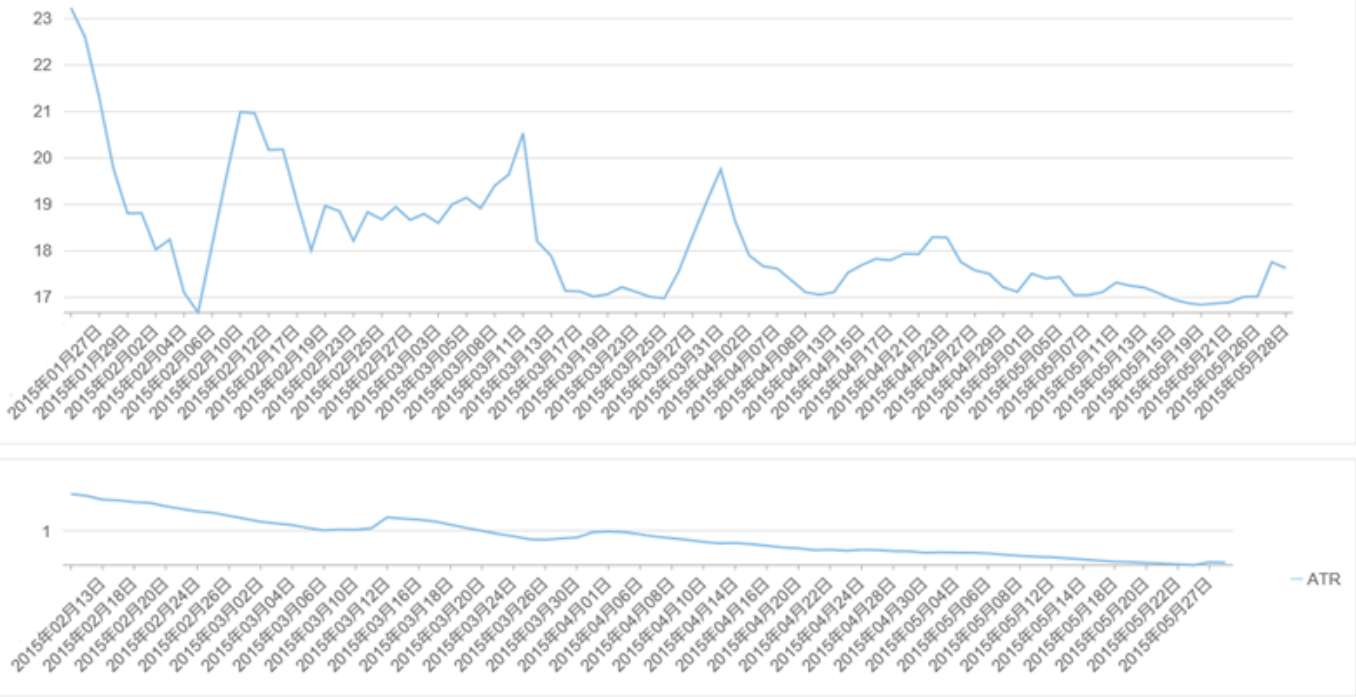
```
{
    sers.Add().Binding("Close")
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible).Name("BOX");
})
.Tooltip(t =&gt; t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Top)
)

@(Html.C1().FinancialChart()
.Id("indicator")
.Height(200)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =&gt;
{
    sers.AddStochastic().Binding("High,Low,Open,Close").KPeriod(14)
.DPeriod(3).SmoothingPeriod(1).SetKLineStyle("#eddd46")
.SetDLineStyle("#edb747").Name("%K, %D")
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t =&gt; t.Content("tooltip"))
```

Average True Range

FinancialChartのATR（Average True Range）インジケータは、資産価格のボラティリティを示すテクニカルインジケータです。ATRは、通常、14期間に基づいて、日中、日、週、または月単位で計算できます。株価のボラティリティの程度に関する情報を提供しますが、株価のトレンドは示しません。ボラティリティが高い株はATRが高くなる一方、ボラティリティが低い株はATRが低くなります。

次の図は、ATR（Average True Range）インジケータをチャートに追加し、期間を14に設定したときのFinancialChartを示しています。



Copyright © GrapeCity, Inc. All rights reserved.

次のコード例は、ATR（Average True Range）指数インジケータをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
              double open = Convert.ToDouble(obj["open"].ToString());
              double close = Convert.ToDouble(obj["close"].ToString());
              double volume = Convert.ToDouble(obj["volume"].ToString());
              list.Add(new FinanceData { X = date, High = high,
              Low = low, Open = open, Close = close, Volume = volume });
          }
          _jsonData = list;
          return list;
      }

      private static List&lt;string&gt; _annotationToolTips;
      public static List&lt;string&gt; GetAnnotationTooltips()
      {
          if (_annotationToolTips != null)
          {
              return _annotationToolTips;
          }
          _annotationToolTips = new List&lt;string&gt;{
   "&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
   "&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
   "&lt;b&gt;Center for Sustainable Energy Leverages Box to
        Power California Clean Vehicle Rebate Project&lt;/b&gt;"
   +"&lt;br&gt;Box today announced that the Center for Sustainable
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        Energy, a nonprofit organization committed to accelerating the
        transition to a sustainable world powered by clean energy, is
        leveraging the Box platform to power a cloud-based document
        submission and review process as part of the Clean Vehicle
        Rebate Project.",
    "&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
        Telecom Conference&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that Aaron Levie, co-founder and
        CEO, will participate in the J.P. Morgan Technology, Media
        and Telecom Conference in Boston on Monday, May 18, 2015.",
    "&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
         BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
    "&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
         Insulin Delivery and Fewer Flow Interruptions including Silent
         Occlusions Infusion Set Developed in Collaboration with JDRF and
        Helmsley Charitable ...",
    "&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
    "&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
    "&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
        Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
        cents per common share payable on June ...",
    "&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
        Financial Results&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that it will report financial results
        for its first quarter which ended April 30, 2015, following the
        close of the market on Wednesday, June 10, 2015.",
    "&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
    "&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
    "&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
    "&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
         their rating on Hercules Technology Growth Capital, Inc.
        (NYSE: HTGX), with a price target of $14. The analysts believe
         that the company's dividend ...",
    "&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
        Enterprise File Sharing&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that it is working with the U.S.
        Department of Justice to deliver file sharing and information
        management to the agency’s workforce. After rigorous assessment,
        Box will receive an agency Authorization to Operate this week,
        allowing the DOJ to leverage the platform across all its component
        agencies.",
    "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
        the ASPCA&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that the ASPCA® , the first humane
        society established in North America, is bringing Box’s cloud
        content sharing and collaboration platform to its employees."
    };

            return _annotationToolTips;
        }
    }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

Copyright © GrapeCity, Inc. All rights reserved.

box.json

box.json


```
[
{ "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23,
"volume": 42593223 },
{ "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6,
"volume": 8677164 },
{ "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3,
"volume": 3272512 },
{ "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78,
"volume": 5047364 },
{ "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8,
"volume": 3419482 },
{ "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close":
18.81, "volume": 2266439 },
{ "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02,
"volume": 2071168 },
{ "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close":
18.24, "volume": 1587435 },
{ "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1,
"volume": 2912224 },
{ "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66,
"volume": 2682187 },
{ "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close":
18.12, "volume": 3929164 },
{ "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6,
"volume": 3226650 },
{ "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99,
"volume": 2804409 },
{ "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96,
"volume": 1698365 },
{ "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close":
20.17, "volume": 1370320 },
{ "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18,
"volume": 711951 },
{ "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05,
"volume": 2093602 },
{ "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18,
"volume": 1849490 },
{ "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close":
18.96, "volume": 1311518 },
{ "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85,
"volume": 1001692 },
{ "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21,
"volume": 670087 },
{ "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83,
"volume": 759263 },
{ "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close":
18.67, "volume": 915580 },
{ "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94,
"volume": 461283 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{ "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66,
"volume": 617199 },
{ "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close":
18.79, "volume": 519605 },
{ "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close":
18.59, "volume": 832415 },
{ "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19,
"volume": 539688 },
{ "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14,
"volume": 486149 },
{ "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91,
"volume": 685659 },
{ "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4,
"volume": 1321363 },
{ "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64,
"volume": 615743 },
{ "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close":
20.53, "volume": 2167167 },
{ "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2,
"volume": 6837638 },
{ "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88,
"volume": 1715629 },
{ "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13,
"volume": 1321313 },
{ "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12,
"volume": 1272242 },
{ "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01,
"volume": 530063 },
{ "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06,
"volume": 536427 },
{ "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close":
17.21, "volume": 1320237 },
{ "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close":
17.11, "volume": 509798 },
{ "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17,
"volume": 962149 },
{ "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close":
16.97, "volume": 565673 },
{ "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close":
17.54, "volume": 884523 },
{ "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3,
"volume": 705626 },
{ "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05,
"volume": 1151620 },
{ "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75,
"volume": 2020679 },
{ "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close":
18.65, "volume": 961078 },
{ "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9,
"volume": 884233 },
{ "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close":
17.66, "volume": 605252 },
{ "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 591988 },
{ "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close":
17.36, "volume": 618855 },
{ "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1,
"volume": 761855 },
{ "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05,
"volume": 568373 },
{ "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1,
"volume": 667142 },
{ "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52,
"volume": 870138 },
{ "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69,
"volume": 530456 },
{ "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82,
"volume": 548730 },
{ "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79,
"volume": 446373 },
{ "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close":
17.93, "volume": 487017 },
{ "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close":
17.92, "volume": 320302 },
{ "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close":
18.29, "volume": 644812 },
{ "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close":
18.28, "volume": 563879 },
{ "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75,
"volume": 650762 },
{ "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close":
17.57, "volume": 437294 },
{ "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5,
"volume": 224519 },
{ "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21,
"volume": 495706 },
{ "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11,
"volume": 391040 },
{ "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5,
"volume": 563075 },
{ "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4,
"volume": 253138 },
{ "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close":
17.43, "volume": 290935 },
{ "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04,
"volume": 313662 },
{ "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close":
17.04, "volume": 360284 },
{ "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1,
"volume": 297653 },
{ "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close":
17.31, "volume": 268504 },
{ "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close":
17.24, "volume": 376961 },
{ "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 244617 },
{ "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close":
17.08, "volume": 252526 },
{ "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close":
16.95, "volume": 274783 },
{ "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close":
16.87, "volume": 418513 },
{ "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83,
"volume": 367660 },
{ "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86,
"volume": 297914 },
{ "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88,
"volume": 229346 },
{ "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17,
"volume": 253279 },
{ "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close":
17.01, "volume": 212640 },
{ "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close":
17.75, "volume": 857109 },
{ "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close":
17.62, "volume": 338482 }
]
```

ATRController.cs

C# copyCode

```
// GET: /ATR/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FinanceData>

<script>
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
            date = wijmo.Globalize.formatDate(date, 'yyyy年MM月dd日');
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        content =
                '<b>' + ht.name + '</b><br/>' +
                'Date: ' + date + '<br/>' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
            '<br/>' +
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }

    function chartRendered(sender, args) {
        // メインチャートに一致するように指標チャートの最小/最大を設定します。
        var indicatorChart = wijmo.Control.getControl('#indicator'),
            chart = wijmo.Control.getControl('#financeChart');

        if (chart && indicatorChart) {
            indicatorChart.axisX.min = chart.axisX.actualMin;
            indicatorChart.axisX.max = chart.axisX.actualMax;
        }
    };
</script>

@(Html.C1().FinancialChart()
.Id("financeChart")
.Height(400)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =>
{
    sers.Add().Binding("Close").Name("BOX")
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Top)
)

@(Html.C1().FinancialChart()
.Id("indicator")
.Height(200)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =>
{
    sers.AddATR().Binding("High,Low,Open,Close")
.Period(14).Name("ATR")
.Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
```

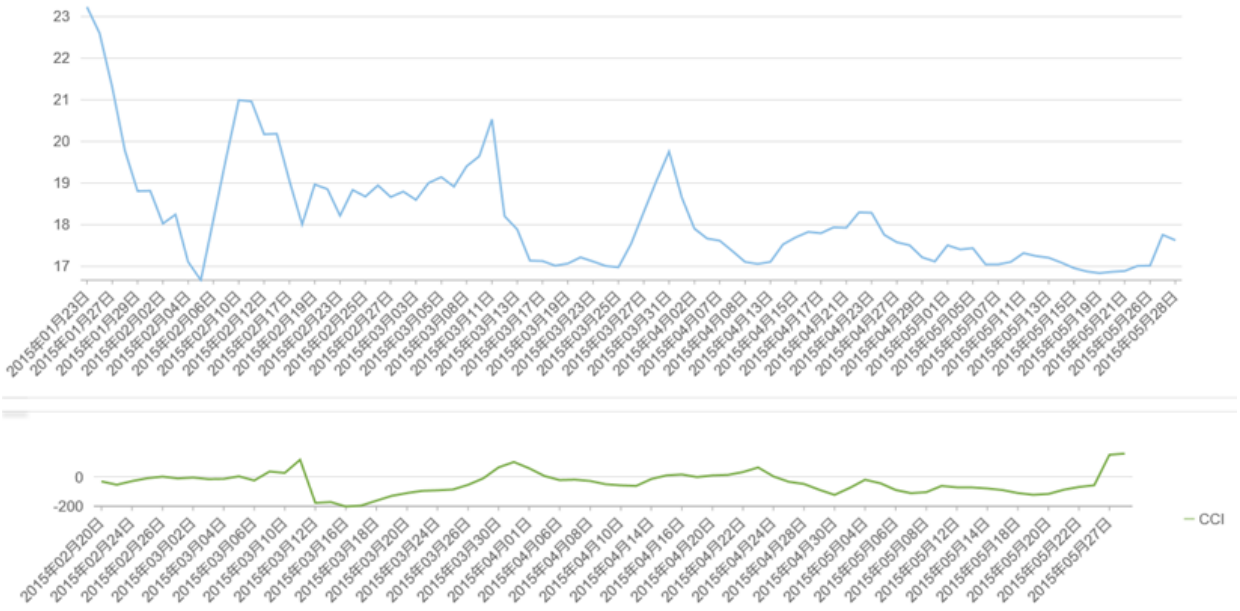

Copyright © GrapeCity, Inc. All rights reserved.

```
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

商品チャンネル指数

FinancialChartのCCI（商品チャンネル指数）インジケータは、平均株価から株価の変動を表示するオシレータです。新しいトレンドを判断したり、極端な状況について警告するために使用されます。通常、特定の期間の平均株価に対して資産の現在の価格レベルを測定します。

次の図は、CCI（商品チャンネル指数）インジケータをチャートに追加し、期間を20に設定したときのFinancialChartを示しています。



次のコード例は、CCI（商品チャンネル指数）インジケータをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
          string date = obj["date"].ToString();
            // DateTime date = Convert.ToDateTime(obj["date"]);
            double high = Convert.ToDouble(obj["high"].ToString());
            double low = Convert.ToDouble(obj["low"].ToString());
            double open = Convert.ToDouble(obj["open"].ToString());
            double close = Convert.ToDouble(obj["close"].ToString());
            double volume = Convert.ToDouble(obj["volume"].ToString());
            list.Add(new FinanceData { X = date, High = high,
            Low = low, Open = open, Close = close, Volume = volume });
        }
        _jsonData = list;
        return list;
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        their rating on Hercules Technology Growth Capital, Inc.
       (NYSE: HTGX), with a price target of $14. The analysts believe
        that the company's dividend ...",
  "&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
       Enterprise File Sharing&lt;/b&gt;"+
  "&lt;br&gt;Box today announced that it is working with the U.S.
       Department of Justice to deliver file sharing and information
       management to the agency's workforce. After rigorous assessment,
       Box will receive an agency Authorization to Operate this week,
       allowing the DOJ to leverage the platform across all its component
       agencies.",
  "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
       the ASPCA&lt;/b&gt;"+
  "&lt;br&gt;Box today announced that the ASPCA® , the first humane
       society established in North America, is bringing Box's cloud
       content sharing and collaboration platform to its employees."
  };

        return _annotationToolTips;
    }
}
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json

```
[
    { "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23,
"volume": 42593223 },
    { "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6,
"volume": 8677164 },
    { "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3,
"volume": 3272512 },
    { "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78,
"volume": 5047364 },
    { "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8,
"volume": 3419482 },
    { "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81,
"volume": 2266439 },
    { "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02,
"volume": 2071168 },
    { "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24,
"volume": 1587435 },
    { "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1,
"volume": 2912224 },
    { "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66,
"volume": 2682187 },
    { "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12,
"volume": 3929164 },
    { "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6,
"volume": 3226650 },
    { "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99,
"volume": 2804409 },
    { "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96,
"volume": 1698365 },
    { "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17,
"volume": 1370320 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18,
"volume": 711951 },
    { "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05,
"volume": 2093602 },
    { "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18,
"volume": 1849490 },
    { "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96,
"volume": 1311518 },
    { "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85,
"volume": 1001692 },
    { "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21,
"volume": 670087 },
    { "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83,
"volume": 759263 },
    { "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67,
"volume": 915580 },
    { "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94,
"volume": 461283 },
    { "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66,
"volume": 617199 },
    { "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79,
"volume": 519605 },
    { "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59,
"volume": 832415 },
    { "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19,
"volume": 539688 },
    { "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14,
"volume": 486149 },
    { "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91,
"volume": 685659 },
    { "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4,
"volume": 1321363 },
    { "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64,
"volume": 615743 },
    { "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53,
"volume": 2167167 },
    { "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2,
"volume": 6837638 },
    { "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88,
"volume": 1715629 },
    { "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13,
"volume": 1321313 },
    { "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12,
"volume": 1272242 },
    { "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01,
"volume": 530063 },
    { "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume":
536427 },
    { "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21,
"volume": 1320237 },
    { "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11,
"volume": 509798 },
    { "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17,
"volume": 962149 },
    { "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97,
"volume": 565673 },
    { "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54,
"volume": 884523 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3,
"volume": 705626 },
    { "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05,
"volume": 1151620 },
    { "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75,
"volume": 2020679 },
    { "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65,
"volume": 961078 },
    { "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9,
"volume": 884233 },
    { "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66,
"volume": 605252 },
    { "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61,
"volume": 591988 },
    { "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36,
"volume": 618855 },
    { "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1,
"volume": 761855 },
    { "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05,
"volume": 568373 },
    { "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1,
"volume": 667142 },
    { "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52,
"volume": 870138 },
    { "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69,
"volume": 530456 },
    { "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82,
"volume": 548730 },
    { "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79,
"volume": 446373 },
    { "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93,
"volume": 487017 },
    { "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92,
"volume": 320302 },
    { "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29,
"volume": 644812 },
    { "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28,
"volume": 563879 },
    { "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75,
"volume": 650762 },
    { "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57,
"volume": 437294 },
    { "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5,
"volume": 224519 },
    { "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21,
"volume": 495706 },
    { "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11,
"volume": 391040 },
    { "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5,
"volume": 563075 },
    { "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4,
"volume": 253138 },
    { "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43,
"volume": 290935 },
    { "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04,
"volume": 313662 },
    { "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04,
"volume": 360284 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1,
"volume": 297653 },
    { "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31,
"volume": 268504 },
    { "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24,
"volume": 376961 },
    { "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2,
"volume": 244617 },
    { "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08,
"volume": 252526 },
    { "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95,
"volume": 274783 },
    { "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87,
"volume": 418513 },
    { "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83,
"volume": 367660 },
    { "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86,
"volume": 297914 },
    { "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88,
"volume": 229346 },
    { "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17,
"volume": 253279 },
    { "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01,
"volume": 212640 },
    { "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75,
"volume": 857109 },
    { "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62,
"volume": 338482 }
]
```

CCIController.cs

C# copyCode

```
// GET: /CCI/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models
@model List<FinanceData>

<script>
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
            date = wijmo.Globalize.formatDate(date, 'yyyy年MM月dd日');
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }

        content =
                '<b>' + ht.name + '</b><br/>' +
                'Date: ' + date + '<br/>' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
            '<br/>' +
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }

    function chartRendered(sender, args) {
        // メインチャートに一致するように指標チャートの最小/最大を設定します。
        var indicatorChart = wijmo.Control.getControl('#indicator'),
            chart = wijmo.Control.getControl('#financeChart');

        if (chart && indicatorChart) {
            indicatorChart.axisX.min = chart.axisX.actualMin;
            indicatorChart.axisX.max = chart.axisX.actualMax;
        }
    };
</script>

@(Html.C1().FinancialChart().Height(400).Width(1000)
.Id("financeChart")
.Bind(Model)
.BindingX("X")
.Series(sers =>
{

sers.Add().Binding("Close").Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible).Name("BOX");
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Top)
)

@(Html.C1().FinancialChart().Height(200).Width(1000)
.Id("indicator")
.Bind(Model)
.BindingX("X")
.Series(sers =>
{
    sers.AddCCI().Binding("High,Low,Open,Close").Period(20).Name("CCI").Style(svgb =>
svgb.Stroke("#78AB46")).Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

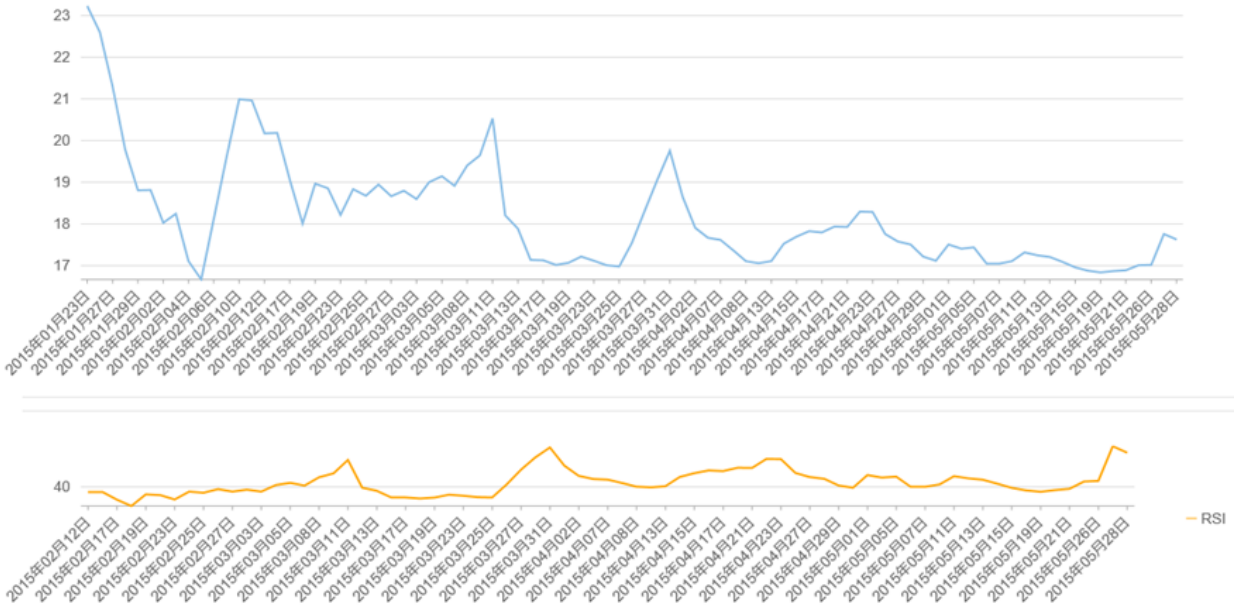
相対力指数

Copyright © GrapeCity, Inc. All rights reserved.

FinancialChartのRSI（相対力指数）インジケータは、株価の動きの速さと大きさを測定するモメンタムオシレータです。資産の終値の上向きの動きを取引期間中の下向きの動きと比較して、株の強さまたは弱さを判断します。値は0～100の間で変動します。プラスの変化が強い株の方が、マイナスの変化が強い株よりRSIが高くなります。

最新の評価益の大きさを最新の評価損と比較する際にRSIを応用することで、資産の買われ過ぎ/売られ過ぎの状況を判断できます。株は、RSIが70を超えると買われ過ぎ、30を下回ると売られ過ぎと見なされます。

次の図は、RSI（相対力指数）インジケータをチャートに追加し、期間を14に設定したときのFinancialChartを示しています。



次のコード例は、RSI（相対力指数）インジケータをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
              double open = Convert.ToDouble(obj["open"].ToString());
              double close = Convert.ToDouble(obj["close"].ToString());
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            double volume = Convert.ToDouble(obj["volume"].ToString());
            list.Add(new FinanceData { X = date, High = high,
            Low = low, Open = open, Close = close, Volume = volume });
        }
        _jsonData = list;
        return list;
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
      their rating on Hercules Technology Growth Capital, Inc.
     (NYSE: HTGX), with a price target of $14. The analysts believe
      that the company's dividend ...",
"&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
     Enterprise File Sharing&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it is working with the U.S.
     Department of Justice to deliver file sharing and information
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        management to the agency's workforce. After rigorous assessment,
        Box will receive an agency Authorization to Operate this week,
        allowing the DOJ to leverage the platform across all its component
        agencies.",
    "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
        the ASPCA&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that the ASPCA® , the first humane
        society established in North America, is bringing Box's cloud
        content sharing and collaboration platform to its employees."
    };

            return _annotationToolTips;
        }
    }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json

```
[
    { "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23,
"volume": 42593223 },
    { "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6,
"volume": 8677164 },
    { "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3,
"volume": 3272512 },
    { "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78,
"volume": 5047364 },
    { "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8,
"volume": 3419482 },
    { "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81,
"volume": 2266439 },
    { "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02,
"volume": 2071168 },
    { "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24,
"volume": 1587435 },
    { "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1,
"volume": 2912224 },
    { "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66,
"volume": 2682187 },
    { "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12,
"volume": 3929164 },
    { "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6,
"volume": 3226650 },
    { "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99,
"volume": 2804409 },
    { "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96,
"volume": 1698365 },
    { "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17,
"volume": 1370320 },
    { "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18,
"volume": 711951 },
    { "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05,
"volume": 2093602 },
    { "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18,
"volume": 1849490 },
    { "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 1311518 },
    { "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85,
"volume": 1001692 },
    { "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21,
"volume": 670087 },
    { "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83,
"volume": 759263 },
    { "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67,
"volume": 915580 },
    { "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94,
"volume": 461283 },
    { "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66,
"volume": 617199 },
    { "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79,
"volume": 519605 },
    { "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59,
"volume": 832415 },
    { "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19,
"volume": 539688 },
    { "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14,
"volume": 486149 },
    { "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91,
"volume": 685659 },
    { "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4,
"volume": 1321363 },
    { "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64,
"volume": 615743 },
    { "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53,
"volume": 2167167 },
    { "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2,
"volume": 6837638 },
    { "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88,
"volume": 1715629 },
    { "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13,
"volume": 1321313 },
    { "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12,
"volume": 1272242 },
    { "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01,
"volume": 530063 },
    { "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume":
536427 },
    { "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21,
"volume": 1320237 },
    { "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11,
"volume": 509798 },
    { "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17,
"volume": 962149 },
    { "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97,
"volume": 565673 },
    { "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54,
"volume": 884523 },
    { "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3,
"volume": 705626 },
    { "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05,
"volume": 1151620 },
    { "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75,
"volume": 2020679 },
    { "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 961078 },
    { "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9,
"volume": 884233 },
    { "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66,
"volume": 605252 },
    { "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61,
"volume": 591988 },
    { "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36,
"volume": 618855 },
    { "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1,
"volume": 761855 },
    { "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05,
"volume": 568373 },
    { "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1,
"volume": 667142 },
    { "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52,
"volume": 870138 },
    { "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69,
"volume": 530456 },
    { "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82,
"volume": 548730 },
    { "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79,
"volume": 446373 },
    { "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93,
"volume": 487017 },
    { "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92,
"volume": 320302 },
    { "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29,
"volume": 644812 },
    { "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28,
"volume": 563879 },
    { "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75,
"volume": 650762 },
    { "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57,
"volume": 437294 },
    { "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5,
"volume": 224519 },
    { "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21,
"volume": 495706 },
    { "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11,
"volume": 391040 },
    { "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5,
"volume": 563075 },
    { "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4,
"volume": 253138 },
    { "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43,
"volume": 290935 },
    { "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04,
"volume": 313662 },
    { "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04,
"volume": 360284 },
    { "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1,
"volume": 297653 },
    { "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31,
"volume": 268504 },
    { "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24,
"volume": 376961 },
    { "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 244617 },
    { "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08,
"volume": 252526 },
    { "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95,
"volume": 274783 },
    { "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87,
"volume": 418513 },
    { "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83,
"volume": 367660 },
    { "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86,
"volume": 297914 },
    { "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88,
"volume": 229346 },
    { "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17,
"volume": 253279 },
    { "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01,
"volume": 212640 },
    { "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75,
"volume": 857109 },
    { "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62,
"volume": 338482 }
]
```

RSIController.cs

C# copyCode

```
// GET: /RSI/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FinanceData>

<script>
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
            date = wijmo.Globalize.formatDate(date, 'MM/dd/yy');
        }

        content =
                '<b>' + ht.name + '</b><br/>' +
                'Date: ' + date + '<br/>' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        if (ht.item.Volume) {
            content +=
            '<br/>' +
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }

    function chartRendered(sender, args) {
        // メインチャートに一致するように指標チャートの最小/最大を設定します。
        var indicatorChart = wijmo.Control.getControl('#indicator'),
            chart = wijmo.Control.getControl('#financeChart');

        if (chart && indicatorChart) {
            indicatorChart.axisX.min = chart.axisX.actualMin;
            indicatorChart.axisX.max = chart.axisX.actualMax;
        }
    };
</script>

@(Html.C1().FinancialChart()
.Id("financeChart")
.Height(400)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =>
{

sers.Add().Binding("Close").Name("BOX").Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Top)
)

@(Html.C1().FinancialChart()
.Id("indicator")
.Height(200)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =>
{
    sers.AddRSI().Binding("Close").Period(14).Name("RSI").Style(svgb => svgb.Stroke("#ffa500
")).Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

Williams %R

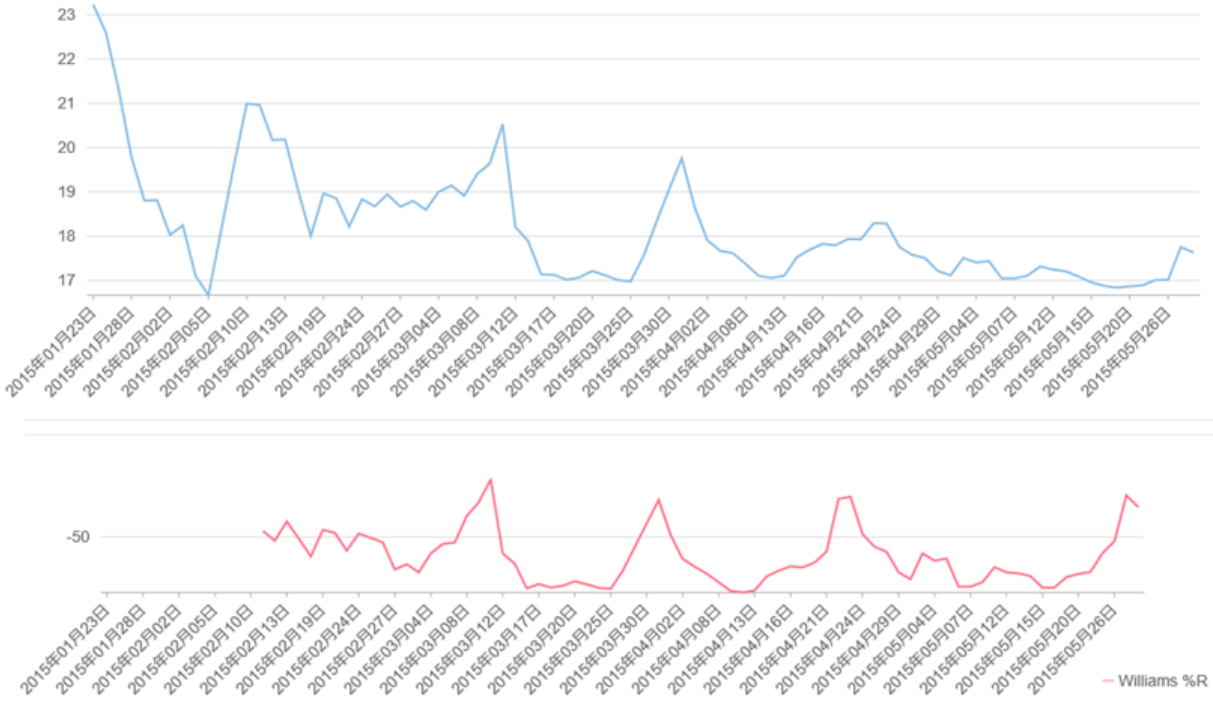
FinancialChartのWilliams %Rインジケータは、現在の資産価格を過去の一定の期間中の最高値と比較するモメンタムインジケータです。通常、比較対象の過去の期間は14期間です。このインジケータは、0～-100の間で変動します。これは、短期ストキャスティクスの逆になりま

Copyright © GrapeCity, Inc. All rights reserved.

す。Williams %Rが過去の期間の最高値と比較した株の終値のレベルを表示するのに対して、ストキャスティクスは最安値と比較した株の終値のレベルを表示します。どちらのインジケータも同じ線を示しますが、スケーリングが異なります。

Williams %Rを買われ過ぎ/売られ過ぎのレベルの判定に応用すると、買いと売りのシグナルを提供したり、モメンタムを確認することができます。

次の図は、Williams %Rインジケータをチャートに追加し、期間を14に設定したときのFinancialChartを示しています。



次のコード例は、Williams %RインジケータをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
              double open = Convert.ToDouble(obj["open"].ToString());
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            double close = Convert.ToDouble(obj["close"].ToString());
            double volume = Convert.ToDouble(obj["volume"].ToString());
            list.Add(new FinanceData { X = date, High = high,
            Low = low, Open = open, Close = close, Volume = volume });
        }
        _jsonData = list;
        return list;
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
      their rating on Hercules Technology Growth Capital, Inc.
     (NYSE: HTGX), with a price target of $14. The analysts believe
      that the company's dividend ...",
"&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
     Enterprise File Sharing&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it is working with the U.S.
     Department of Justice to deliver file sharing and information
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        management to the agency’s workforce. After rigorous assessment,
        Box will receive an agency Authorization to Operate this week,
        allowing the DOJ to leverage the platform across all its component
        agencies.",
    "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
        the ASPCA&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that the ASPCA® , the first humane
        society established in North America, is bringing Box’s cloud
        content sharing and collaboration platform to its employees."
    };

            return _annotationToolTips;
        }
    }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json

```
[
    { "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23,
"volume": 42593223 },
    { "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6,
"volume": 8677164 },
    { "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3, "volume":
3272512 },
    { "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78,
"volume": 5047364 },
    { "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8,
"volume": 3419482 },
    { "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close": 18.81,
"volume": 2266439 },
    { "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02,
"volume": 2071168 },
    { "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close": 18.24,
"volume": 1587435 },
    { "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1, "volume":
2912224 },
    { "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66,
"volume": 2682187 },
    { "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close": 18.12,
"volume": 3929164 },
    { "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6,
"volume": 3226650 },
    { "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99, "volume":
2804409 },
    { "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96, "volume":
1698365 },
    { "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close": 20.17,
"volume": 1370320 },
    { "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18,
"volume": 711951 },
    { "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05,
"volume": 2093602 },
    { "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18, "volume":
1849490 },
    { "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close": 18.96,
"volume": 1311518 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85,
"volume": 1001692 },
    { "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21,
"volume": 670087 },
    { "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83,
"volume": 759263 },
    { "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close": 18.67,
"volume": 915580 },
    { "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94,
"volume": 461283 },
    { "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66,
"volume": 617199 },
    { "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close": 18.79,
"volume": 519605 },
    { "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close": 18.59,
"volume": 832415 },
    { "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19,
"volume": 539688 },
    { "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14,
"volume": 486149 },
    { "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91,
"volume": 685659 },
    { "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4,
"volume": 1321363 },
    { "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64,
"volume": 615743 },
    { "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close": 20.53,
"volume": 2167167 },
    { "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2,
"volume": 6837638 },
    { "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88,
"volume": 1715629 },
    { "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13,
"volume": 1321313 },
    { "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12,
"volume": 1272242 },
    { "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01,
"volume": 530063 },
    { "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06, "volume":
536427 },
    { "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close": 17.21,
"volume": 1320237 },
    { "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close": 17.11,
"volume": 509798 },
    { "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17,
"volume": 962149 },
    { "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close": 16.97,
"volume": 565673 },
    { "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close": 17.54,
"volume": 884523 },
    { "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3,
"volume": 705626 },
    { "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05,
"volume": 1151620 },
    { "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75,
"volume": 2020679 },
    { "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close": 18.65,
"volume": 961078 },
    { "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 884233 },
    { "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close": 17.66,
"volume": 605252 },
    { "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61,
"volume": 591988 },
    { "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close": 17.36,
"volume": 618855 },
    { "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1,
"volume": 761855 },
    { "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05,
"volume": 568373 },
    { "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1,
"volume": 667142 },
    { "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52,
"volume": 870138 },
    { "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69,
"volume": 530456 },
    { "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82, "volume":
548730 },
    { "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79,
"volume": 446373 },
    { "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close": 17.93,
"volume": 487017 },
    { "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close": 17.92,
"volume": 320302 },
    { "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close": 18.29,
"volume": 644812 },
    { "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close": 18.28,
"volume": 563879 },
    { "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75,
"volume": 650762 },
    { "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close": 17.57,
"volume": 437294 },
    { "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5,
"volume": 224519 },
    { "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21,
"volume": 495706 },
    { "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11, "volume":
391040 },
    { "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5,
"volume": 563075 },
    { "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4,
"volume": 253138 },
    { "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close": 17.43,
"volume": 290935 },
    { "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04,
"volume": 313662 },
    { "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close": 17.04,
"volume": 360284 },
    { "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1,
"volume": 297653 },
    { "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close": 17.31,
"volume": 268504 },
    { "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close": 17.24,
"volume": 376961 },
    { "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2,
"volume": 244617 },
    { "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close": 17.08,
"volume": 252526 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close": 16.95,
"volume": 274783 },
    { "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close": 16.87,
"volume": 418513 },
    { "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83,
"volume": 367660 },
    { "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86,
"volume": 297914 },
    { "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88,
"volume": 229346 },
    { "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17, "volume":
253279 },
    { "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close": 17.01,
"volume": 212640 },
    { "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close": 17.75,
"volume": 857109 },
    { "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close": 17.62,
"volume": 338482 }
]
```

WilliamsRController.cs

C# copyCode

```
// GET: /Williams/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FinanceData>

<script>
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {

            date = wijmo.Globalize.formatDate(date, 'yyyy年MM月dd日');
        }

        content =
                '<b>' + ht.name + '</b><br/>' +
                'Date: ' + date + '<br/>' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
            '<br/>' +
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }

    function chartRendered(sender, args) {
        // メインチャートに一致するように指標チャートの最小/最大を設定します。
        var indicatorChart = wijmo.Control.getControl('#indicator'),
            chart = wijmo.Control.getControl('#financeChart');

        if (chart && indicatorChart) {
            indicatorChart.axisX.min = chart.axisX.actualMin;
            indicatorChart.axisX.max = chart.axisX.actualMax;
        }
    };
</script>

@(Html.C1().FinancialChart()
.Id("financeChart")
.Height(400)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =>
{

sers.Add().Binding("Close").Name("BOX").Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Top)
)

@(Html.C1().FinancialChart()
.Id("indicator")
.Height(200)
.Width(1000)
.Bind(Model)
.BindingX("X")
.Series(sers =>
{
    sers.AddWilliamsR().Binding("High,Low,Open,Close").Period(14).Style(svgb =>
svgb.Stroke("#ffb6c1")).Name("WilliamsR").Visibility(C1.Web.Mvc.Chart.SeriesVisibility.Visible);
})
.Tooltip(t => t.Content("tooltip"))
.OnClientRendered("chartRendered")
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

ラインマーカー

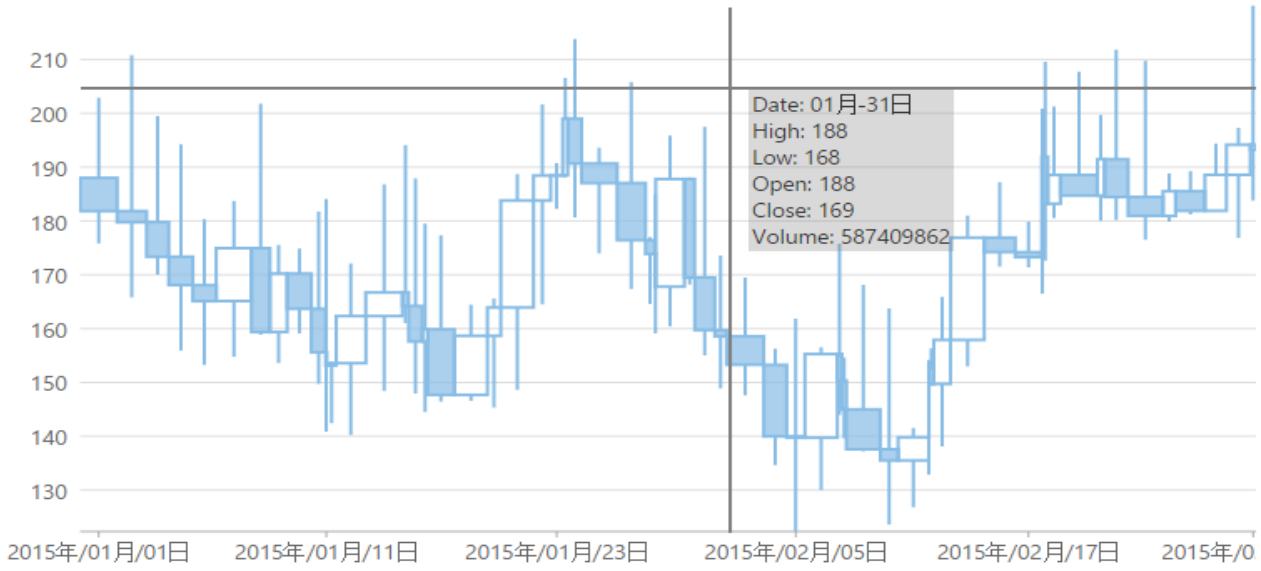
マーカーは、マウスがデータ系列の上に置かれたときにデータポイントを表示するために使用されるシンボルです。FinancialChartでは、**AddLineMarker**メソッドを使用してラインマーカーを追加できます。**LineMarkerLines**プロパティを使用すると、次のようにラインマーカーを設定できます。

- **Vertical**：縦方向のラインマーカーを表示します。

Copyright © GrapeCity, Inc. All rights reserved.

- **Horizontal**：横方向のラインマーカーを表示します。
- **Both**：縦方向および横方向のラインマーカーを設定します。
- **None**：ラインを表示しません。

次の図は、FinancialChartのデータポイントの値を表示するためにLineMarkerLinesプロパティを**Both（十字線効果）**に設定したときのFinancialChartを示しています。



次のコード例は、ラインマーカーとマーカーコンテンツをFinancialChartに追加する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FData>

<script type="text/javascript">
    function lineMarkerContent(ht, pt) {
        var item = ht.series.collectionView.items[ht.pointIndex];
        if (item) {
            return 'Date: ' + wijmo.Globalize.format(ht.x, 'MM月-dd日') + '<br/>' +
                              'High: ' + item.High.toFixed() + '<br/>' +
                              'Low: ' + item.Low.toFixed() + '<br/>' +
                              'Open: ' + item.Open.toFixed() + '<br/>' +
                              'Close: ' + item.Close.toFixed() + '<br/>' +
                              'Volume: ' + item.Volume.toFixed();
        }
    }
</script>
@*FinancialChart コントロールを初期化します。*@
@(Html.C1().FinancialChart().Width(1000)
.Bind(Model)
.BindingX("X")
        //グラウのタイプを設定します。
    .ChartType(C1.Web.Mvc.Finance.ChartType.CandleVolume)
.Series(sers =>
    {
        sers.Add().Binding("High,Low,Open,Close,Volume");
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    }).AxisX(x => x.Format("yyyy年MM月dd日"))
.Tooltip(tp => tp.Content(""))
//FinancialChartにラインマーカーを追加します。
.AddLineMarker(lm => lm
        .Alignment(C1.Web.Mvc.Chart.LineMarkerAlignment.Auto)
        .Lines(C1.Web.Mvc.Chart.LineMarkerLines.Both)
        .DragContent(true)

.Interaction(C1.Web.Mvc.Chart.LineMarkerInteraction.Move).Content("lineMarkerContent")))
```

移動平均

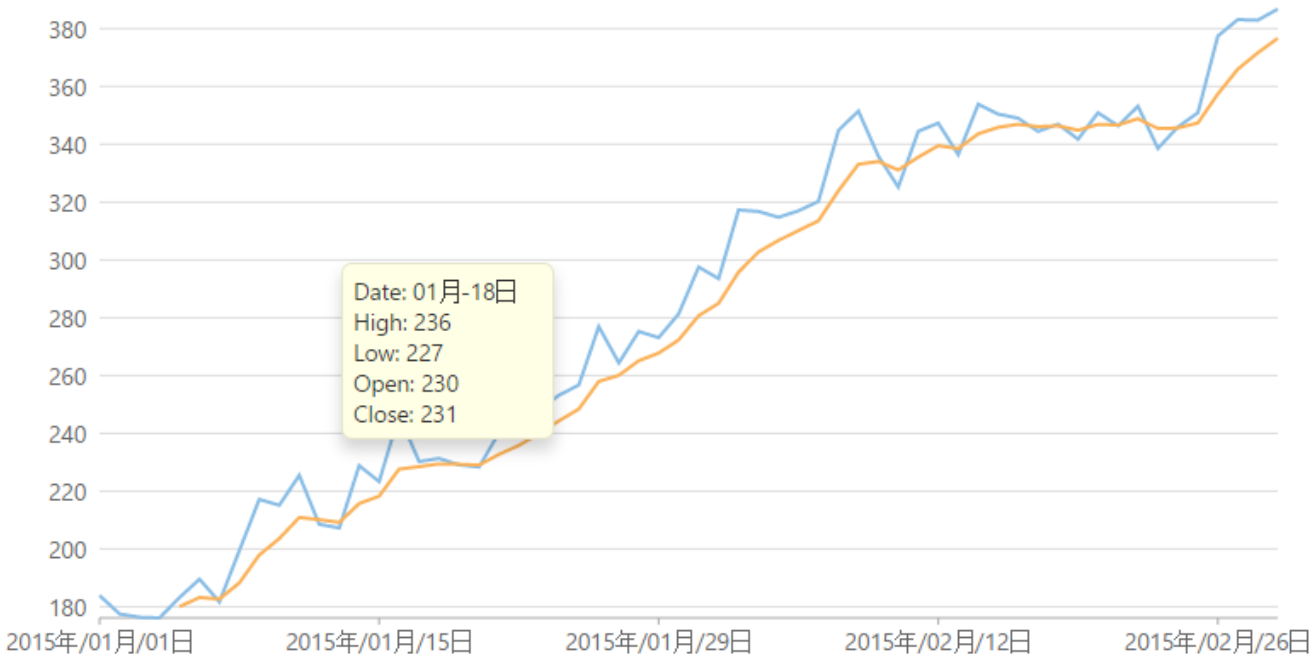
FinancialChartの**MovingAverage**は、財務チャートの移動平均近似曲線を表します。これは、データセット全体のさまざまなサブセットから一連の平均値を求めることでデータポイントを分析する計算です。各**MovingAverage**オブジェクトに対して異なるタイプを定義できます。それには、**MovingAverageType**プロパティを次のように設定します。

- **Exponential**：直近n個の値の加重平均で、重み付けは1つ前の値と比較して指数関数的に減っていきます。
- **Simple**：直近n個の値の平均。
- **Triangular**：直近n個の値の加重平均で、結果は2回の平滑化を行った単純移動平均と同じです。
- **Weighted**：直近n個の値の加重平均で、重み付けは1つ前の値と比較して1ずつ減っていきます。

この例では、MovingAverageTypeプロパティの値は**Exponential**に設定されています。

MovingAverageクラスの**Period**プロパティを使用して、平均値を計算する期間の数を設定します。この値は、**1より大きな**整数値に設定する必要があります。この例では、**Period**プロパティの値は整数値5に設定されています。

次の図は、MovingAverageを使用して計算し、その平均値をチャートに表示したFinancialChartを示しています。**MovingAverageType**値は**Exponential**に、**Period**プロパティの値は**5**に設定されています。



次のコード例は、MovingAverageを使用してFinancialChartのデータを分析する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor	copyCode

```
@using MVCFinancialChart.Models
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@model List<FData>

<script type="text/javascript">
    var tooltipContent = function (ht) {
        var item = ht.series.collectionView.items[ht.pointIndex];
        if (item) {
            return 'Date: ' + wijmo.Globalize
             .format(ht.x, 'MM月-dd日') + '<br/>' +
             'High: ' + item.High.toFixed() + '<br/>' +
             'Low: ' + item.Low.toFixed() + '<br/>' +
             'Open: ' + item.Open.toFixed() + '<br/>' +
             'Close: ' + item.Close.toFixed() + '<br/>'
        }
    };

</script>
@(Html.C1().FinancialChart()
.Width(1000)
.Bind(Model)
.BindingX("X")
    .ChartType(C1.Web.Mvc.Finance.ChartType.Line)
.Series(sers =>
    {
        sers.Add().Binding("Close");
        sers.AddMovingAverage().Binding("Close").Period(5)
.Type(C1.Web.Mvc.Chart.MovingAverageType.Exponential);
    }).AxisX(x => x.Format("yyyy年MM月dd日"))
.Tooltip(t => t.Content("tooltipContent")))
```

Financial Chart オーバーレイ

オーバーレイは、株価チャートまたは出来高チャートのデータと同じスケールを使用するインジケータです。したがって、株価の棒グラフと共にプロットされます。MVC Editionでは、株価チャートコントロールで、金融市場のモメンタム予測のためのオーバーレイがサポートされています。以下のセクションでは、株価チャートでテクニカル分析のためにオーバーレイを使用する方法を示します。

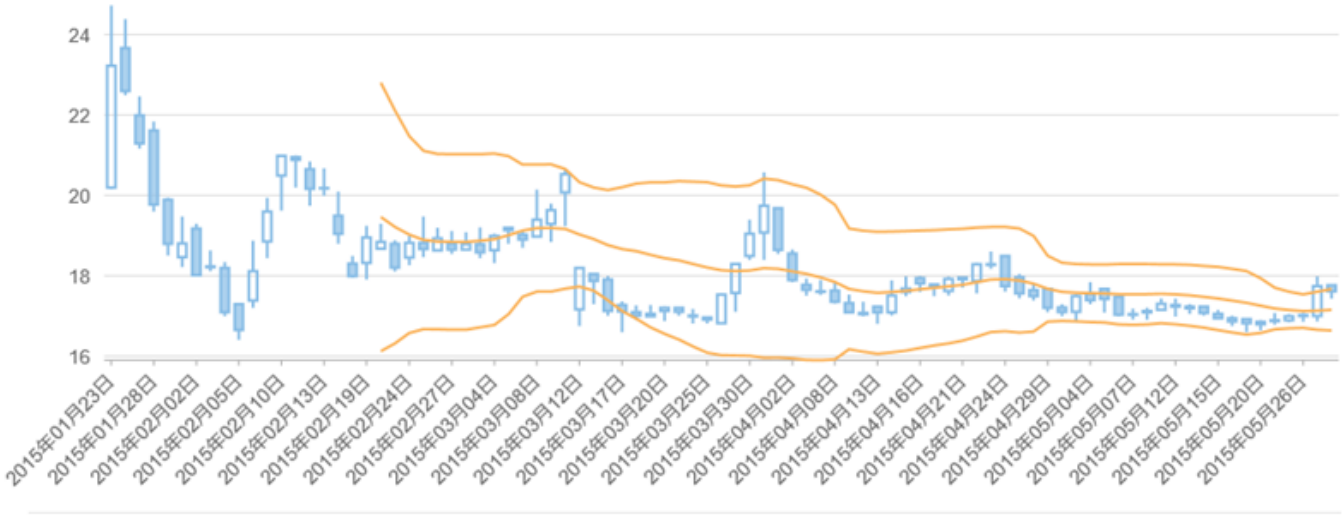
ボリンジャーバンドオーバーレイ

FinancialChartのボリンジャーバンドオーバーレイは、正常な株価の動きの上限と下限を表示するボラティリティバンドです。このバンドは、株価系列の上と下に配置されます。ボラティリティは、株価の標準偏差に基づくため、このバンドは、ボラティリティが増大すると広くなり、低下すると狭くなります。

ボリンジャーバンドは、株価の高い安いを相対的に定義するためにたいへん役立ちます。上のバンド近くにプロットされた株価は高いと見なされ、下のバンド近くにプロットされた価格は低いと見なされます。

次の図は、期間を20に設定し、乗数を2に設定して、フィボナッチ数列ボリンジャーバンドオーバーレイをチャートに追加したときのFinancialChartを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、ボリンジャーバンドオーバーレイをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
              double open = Convert.ToDouble(obj["open"].ToString());
              double close = Convert.ToDouble(obj["close"].ToString());
              double volume = Convert.ToDouble(obj["volume"].ToString());
              list.Add(new FinanceData { X = date, High = high,
              Low = low, Open = open, Close = close, Volume = volume });
          }
          _jsonData = list;
          return list;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
      their rating on Hercules Technology Growth Capital, Inc.
     (NYSE: HTGX), with a price target of $14. The analysts believe
      that the company's dividend ...",
"&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
     Enterprise File Sharing&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it is working with the U.S.
     Department of Justice to deliver file sharing and information
     management to the agency’s workforce. After rigorous assessment,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        Box will receive an agency Authorization to Operate this week,
        allowing the DOJ to leverage the platform across all its component
        agencies.",
    "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
        the ASPCA&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that the ASPCA® , the first humane
        society established in North America, is bringing Box's cloud
        content sharing and collaboration platform to its employees."
    };

            return _annotationToolTips;
        }
    }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json


```
[
    { "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close":
23.23, "volume": 42593223 },
    { "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close":
22.6, "volume": 8677164 },
    { "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close":
21.3, "volume": 3272512 },
    { "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close":
19.78, "volume": 5047364 },
    { "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close":
18.8, "volume": 3419482 },
    { "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close":
18.81, "volume": 2266439 },
    { "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close":
18.02, "volume": 2071168 },
    { "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close":
18.24, "volume": 1587435 },
    { "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1,
"volume": 2912224 },
    { "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close":
16.66, "volume": 2682187 },
    { "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close":
18.12, "volume": 3929164 },
    { "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close":
19.6, "volume": 3226650 },
    { "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close":
20.99, "volume": 2804409 },
    { "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close":
20.96, "volume": 1698365 },
    { "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close":
20.17, "volume": 1370320 },
    { "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close":
20.18, "volume": 711951 },
    { "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close":
```


Copyright © GrapeCity, Inc. All rights reserved.

```
19.05, "volume": 2093602 },
    { "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close":
18, "volume": 1849490 },
    { "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close":
18.96, "volume": 1311518 },
    { "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close":
18.85, "volume": 1001692 },
    { "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close":
18.21, "volume": 670087 },
    { "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close":
18.83, "volume": 759263 },
    { "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close":
18.67, "volume": 915580 },
    { "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close":
18.94, "volume": 461283 },
    { "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close":
18.66, "volume": 617199 },
    { "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close":
18.79, "volume": 519605 },
    { "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close":
18.59, "volume": 832415 },
    { "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close":
19, "volume": 539688 },
    { "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close":
19.14, "volume": 486149 },
    { "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close":
18.91, "volume": 685659 },
    { "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close":
19.4, "volume": 1321363 },
    { "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close":
19.64, "volume": 615743 },
    { "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close":
20.53, "volume": 2167167 },
    { "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close":
18.2, "volume": 6837638 },
    { "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close":
17.88, "volume": 1715629 },
    { "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close":
17.13, "volume": 1321313 },
    { "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close":
17.12, "volume": 1272242 },
    { "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close":
17.01, "volume": 530063 },
    { "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06,
"volume": 536427 },
    { "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close":
17.21, "volume": 1320237 },
    { "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close":
17.11, "volume": 509798 },
    { "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close":
17, "volume": 962149 },
    { "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close":
16.97, "volume": 565673 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close":
17.54, "volume": 884523 },
    { "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close":
18.3, "volume": 705626 },
    { "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close":
19.05, "volume": 1151620 },
    { "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close":
19.75, "volume": 2020679 },
    { "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close":
18.65, "volume": 961078 },
    { "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close":
17.9, "volume": 884233 },
    { "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close":
17.66, "volume": 605252 },
    { "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close":
17.61, "volume": 591988 },
    { "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close":
17.36, "volume": 618855 },
    { "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close":
17.1, "volume": 761855 },
    { "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close":
17.05, "volume": 568373 },
    { "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close":
17.1, "volume": 667142 },
    { "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close":
17.52, "volume": 870138 },
    { "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close":
17.69, "volume": 530456 },
    { "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close":
17.82, "volume": 548730 },
    { "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close":
17.79, "volume": 446373 },
    { "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close":
17.93, "volume": 487017 },
    { "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close":
17.92, "volume": 320302 },
    { "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close":
18.29, "volume": 644812 },
    { "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close":
18.28, "volume": 563879 },
    { "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close":
17.75, "volume": 650762 },
    { "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close":
17.57, "volume": 437294 },
    { "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close":
17.5, "volume": 224519 },
    { "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close":
17.21, "volume": 495706 },
    { "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close":
17.11, "volume": 391040 },
    { "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close":
17.5, "volume": 563075 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    { "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close":
17.4, "volume": 253138 },
    { "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close":
17.43, "volume": 290935 },
    { "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close":
17.04, "volume": 313662 },
    { "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close":
17.04, "volume": 360284 },
    { "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close":
17.1, "volume": 297653 },
    { "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close":
17.31, "volume": 268504 },
    { "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close":
17.24, "volume": 376961 },
    { "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close":
17.2, "volume": 244617 },
    { "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close":
17.08, "volume": 252526 },
    { "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close":
16.95, "volume": 274783 },
    { "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close":
16.87, "volume": 418513 },
    { "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close":
16.83, "volume": 367660 },
    { "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close":
16.86, "volume": 297914 },
    { "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close":
16.88, "volume": 229346 },
    { "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close":
17, "volume": 253279 },
    { "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close":
17.01, "volume": 212640 },
    { "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close":
17.75, "volume": 857109 },
    { "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close":
17.62, "volume": 338482 }
]
```

BollingerBandsController.cs

C# copyCode

```
// GET: /BollingerBands/

public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@model List<FinanceData>

<script>
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
            date = wijmo.Globalize.formatDate(date, 'yyyy年MM月dd日');
        }

        content =
                '<b>' + ht.name + '</b><br/>' +
                'Date: ' + date + '<br/>' +
                'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
            '<br/>' +
            'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }
</script>

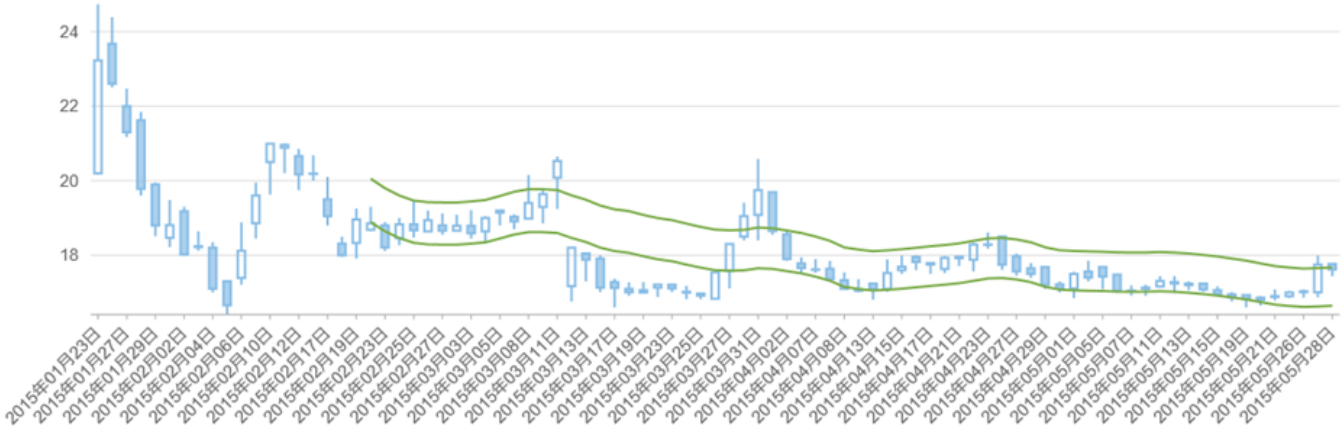
@(Html.C1().FinancialChart()
.Height(400)
.Width(1000)
.Bind(Model)
.BindingX("X")
.SymbolSize(6)
.ChartType(C1.Web.Mvc.Finance.ChartType.Candlestick)
.Series(sers =>
{
    sers.Add().Binding("High,Low,Open,Close");
    sers.AddBollingerBands().Binding("Close").Period(20).Multiplier(2);
})
.Tooltip(t => t.Content("tooltip"))
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

エンベロープオーバーレイ

エンベロープオーバーレイは、FinancialChartの移動平均エンベロープオーバーレイ系列を表します。この移動平均エンベロープは、標準移動平均の上および下に設定されるパーセントベースのエンベロープです。移動平均には、単純移動平均または指数移動平均を使用できます。

Copyright © GrapeCity, Inc. All rights reserved.

次の図は、期間を20に設定し、サイズを3%に設定して、単純移動平均エンベロープオーバーレイをチャートに追加したときのFinancialChartを示しています。



次のコード例は、単純移動平均エンベロープをFinancialChartに追加する方法を示します。この例では、BoxData.csモデルのデータとbox.jsonファイルを使用します。

コードの場合

BoxData.csクラスを[モデル]フォルダに追加します。

Model

BoxData.cs copyCode

```
 public class BoxData
  {
      private static List&lt;FinanceData&gt; _jsonData;
      public static List&lt;FinanceData&gt; GetDataFromJson()
      {
          if (_jsonData != null)
          {
              return _jsonData;
          }

 string path = HttpContext.Current.Server.MapPath("~/Content/box.json");
 string jsonText = new StreamReader(path, System.Text.Encoding.Default).ReadToEnd();
JArray ja = (JArray)JsonConvert.DeserializeObject(jsonText);
          List&lt;FinanceData&gt; list = new List&lt;FinanceData&gt;();
          foreach (var obj in ja)
          {
            string date = obj["date"].ToString();
              // DateTime date = Convert.ToDateTime(obj["date"]);
              double high = Convert.ToDouble(obj["high"].ToString());
              double low = Convert.ToDouble(obj["low"].ToString());
              double open = Convert.ToDouble(obj["open"].ToString());
              double close = Convert.ToDouble(obj["close"].ToString());
              double volume = Convert.ToDouble(obj["volume"].ToString());
              list.Add(new FinanceData { X = date, High = high,
              Low = low, Open = open, Close = close, Volume = volume });
          }
          _jsonData = list;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        return list;
    }

    private static List&lt;string&gt; _annotationToolTips;
    public static List&lt;string&gt; GetAnnotationTooltips()
    {
        if (_annotationToolTips != null)
        {
            return _annotationToolTips;
        }
        _annotationToolTips = new List&lt;string&gt;{
"&lt;b&gt;Why the hot IPO market is cooling off?&lt;/b&gt;",
"&lt;b&gt;Tech IPO market healthy?&lt;/b&gt;",
"&lt;b&gt;Center for Sustainable Energy Leverages Box to
     Power California Clean Vehicle Rebate Project&lt;/b&gt;"
+"&lt;br&gt;Box today announced that the Center for Sustainable
     Energy, a nonprofit organization committed to accelerating the
     transition to a sustainable world powered by clean energy, is
     leveraging the Box platform to power a cloud-based document
     submission and review process as part of the Clean Vehicle
     Rebate Project.",
"&lt;b&gt;Box to Present at the J.P. Morgan Technology, Media and
     Telecom Conference&lt;/b&gt;"+
"&lt;br&gt;Box today announced that Aaron Levie, co-founder and
     CEO, will participate in the J.P. Morgan Technology, Media
     and Telecom Conference in Boston on Monday, May 18, 2015.",
"&lt;b&gt;BD Receives FDA Clearance for a Novel Infusion Set with
      BD FlowSmart™ Technology to Enhance the Use of Insulin Pumps&lt;/b&gt;"+
"&lt;br&gt;Unique Side-Ported Catheter Designed to Offer Consistent
      Insulin Delivery and Fewer Flow Interruptions including Silent
      Occlusions Infusion Set Developed in Collaboration with JDRF and
     Helmsley Charitable ...",
"&lt;b&gt;Wall Street tech debutant Box dismisses doubters&lt;/b&gt;",
"&lt;b&gt;BD Board Declares Dividend&lt;/b&gt;"+
"&lt;br&gt;The Board of Directors of BD (Becton, Dickinson and
     Company) (NYSE: BDX) has declared a quarterly dividend of 60.0
     cents per common share payable on June ...",
"&lt;b&gt;Box Sets Date to Announce First Quarter Fiscal 2016
     Financial Results&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it will report financial results
     for its first quarter which ended April 30, 2015, following the
     close of the market on Wednesday, June 10, 2015.",
"&lt;b&gt;Making Money With Charles Payne: 05/21/15&lt;/b&gt;",
"&lt;b&gt;Best Buy to move sideways: Analyst&lt;/b&gt;",
"&lt;b&gt;Macquarie Upgrades Hercules Technology, Likes Dividend&lt;/b&gt;"+
"&lt;br&gt;In a report published Tuesday, Macquarie analysts upgraded
      their rating on Hercules Technology Growth Capital, Inc.
     (NYSE: HTGX), with a price target of $14. The analysts believe
      that the company's dividend ...",
"&lt;b&gt;The U.S. Department of Justice Selects Box to Enable
     Enterprise File Sharing&lt;/b&gt;"+
"&lt;br&gt;Box today announced that it is working with the U.S.
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        Department of Justice to deliver file sharing and information
        management to the agency's workforce. After rigorous assessment,
        Box will receive an agency Authorization to Operate this week,
        allowing the DOJ to leverage the platform across all its component
        agencies.",
    "&lt;b&gt;Box Enables Improved Collaboration and Content Sharing at
        the ASPCA&lt;/b&gt;"+
    "&lt;br&gt;Box today announced that the ASPCA® , the first humane
        society established in North America, is bringing Box's cloud
        content sharing and collaboration platform to its employees."
    };

            return _annotationToolTips;
        }
    }
```

また、box.jsonファイルをプロジェクトに追加し、次の内容をファイルに貼り付けます。

box.json

box.json

```
[
{ "date": "2015年01月23日", "open": 20.2, "high": 24.73, "low": 20.16, "close": 23.23,
"volume": 42593223 },
{ "date": "2015年01月26日", "open": 23.67, "high": 24.39, "low": 22.5, "close": 22.6,
"volume": 8677164 },
{ "date": "2015年01月27日", "open": 22, "high": 22.47, "low": 21.17, "close": 21.3,
"volume": 3272512 },
{ "date": "2015年01月28日", "open": 21.62, "high": 21.84, "low": 19.6, "close": 19.78,
"volume": 5047364 },
{ "date": "2015年01月29日", "open": 19.9, "high": 19.95, "low": 18.51, "close": 18.8,
"volume": 3419482 },
{ "date": "2015年01月30日", "open": 18.47, "high": 19.48, "low": 18.22, "close":
18.81, "volume": 2266439 },
{ "date": "2015年02月02日", "open": 19.18, "high": 19.3, "low": 18.01, "close": 18.02,
"volume": 2071168 },
{ "date": "2015年02月03日", "open": 18.22, "high": 18.64, "low": 18.12, "close":
18.24, "volume": 1587435 },
{ "date": "2015年02月04日", "open": 18.2, "high": 18.35, "low": 17, "close": 17.1,
"volume": 2912224 },
{ "date": "2015年02月05日", "open": 17.3, "high": 17.31, "low": 16.41, "close": 16.66,
"volume": 2682187 },
{ "date": "2015年02月06日", "open": 17.39, "high": 18.88, "low": 17.21, "close":
18.12, "volume": 3929164 },
{ "date": "2015年02月08日", "open": 18.86, "high": 19.95, "low": 18.45, "close": 19.6,
"volume": 3226650 },
{ "date": "2015年02月10日", "open": 20.5, "high": 21, "low": 19.63, "close": 20.99,
"volume": 2804409 },
{ "date": "2015年02月11日", "open": 20.89, "high": 21, "low": 20.2, "close": 20.96,
"volume": 1698365 },
{ "date": "2015年02月12日", "open": 20.66, "high": 20.85, "low": 19.75, "close":
20.17, "volume": 1370320 },
{ "date": "2015年02月13日", "open": 20.19, "high": 20.68, "low": 20, "close": 20.18,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"volume": 711951 },
{ "date": "2015年02月17日", "open": 19.5, "high": 20.1, "low": 18.8, "close": 19.05,
"volume": 2093602 },
{ "date": "2015年02月18日", "open": 18.31, "high": 18.5, "low": 17.96, "close": 18,
"volume": 1849490 },
{ "date": "2015年02月19日", "open": 18.33, "high": 19.25, "low": 17.91, "close":
18.96, "volume": 1311518 },
{ "date": "2015年02月20日", "open": 18.68, "high": 19.3, "low": 18.65, "close": 18.85,
"volume": 1001692 },
{ "date": "2015年02月23日", "open": 18.8, "high": 18.89, "low": 18.11, "close": 18.21,
"volume": 670087 },
{ "date": "2015年02月24日", "open": 18.46, "high": 19, "low": 18.27, "close": 18.83,
"volume": 759263 },
{ "date": "2015年02月25日", "open": 18.83, "high": 19.48, "low": 18.47, "close":
18.67, "volume": 915580 },
{ "date": "2015年02月26日", "open": 18.64, "high": 19.2, "low": 18.64, "close": 18.94,
"volume": 461283 },
{ "date": "2015年02月27日", "open": 18.8, "high": 19.12, "low": 18.55, "close": 18.66,
"volume": 617199 },
{ "date": "2015年03月02日", "open": 18.66, "high": 19.09, "low": 18.65, "close":
18.79, "volume": 519605 },
{ "date": "2015年03月03日", "open": 18.79, "high": 19.21, "low": 18.45, "close":
18.59, "volume": 832415 },
{ "date": "2015年03月04日", "open": 18.64, "high": 19.05, "low": 18.32, "close": 19,
"volume": 539688 },
{ "date": "2015年03月05日", "open": 19.2, "high": 19.2, "low": 18.8, "close": 19.14,
"volume": 486149 },
{ "date": "2015年03月06日", "open": 19.03, "high": 19.1, "low": 18.7, "close": 18.91,
"volume": 685659 },
{ "date": "2015年03月08日", "open": 18.98, "high": 20.15, "low": 18.96, "close": 19.4,
"volume": 1321363 },
{ "date": "2015年03月10日", "open": 19.3, "high": 19.8, "low": 18.85, "close": 19.64,
"volume": 615743 },
{ "date": "2015年03月11日", "open": 20.08, "high": 20.65, "low": 19.24, "close":
20.53, "volume": 2167167 },
{ "date": "2015年03月12日", "open": 17.17, "high": 18.2, "low": 16.76, "close": 18.2,
"volume": 6837638 },
{ "date": "2015年03月13日", "open": 18.05, "high": 18.05, "low": 17.3, "close": 17.88,
"volume": 1715629 },
{ "date": "2015年03月16日", "open": 17.91, "high": 18, "low": 17.01, "close": 17.13,
"volume": 1321313 },
{ "date": "2015年03月17日", "open": 17.28, "high": 17.37, "low": 16.6, "close": 17.12,
"volume": 1272242 },
{ "date": "2015年03月18日", "open": 17.1, "high": 17.27, "low": 16.91, "close": 17.01,
"volume": 530063 },
{ "date": "2015年03月19日", "open": 17, "high": 17.28, "low": 17, "close": 17.06,
"volume": 536427 },
{ "date": "2015年03月20日", "open": 17.13, "high": 17.24, "low": 16.88, "close":
17.21, "volume": 1320237 },
{ "date": "2015年03月23日", "open": 17.21, "high": 17.23, "low": 17.01, "close":
17.11, "volume": 509798 },
{ "date": "2015年03月24日", "open": 17.02, "high": 17.18, "low": 16.82, "close": 17,
"volume": 962149 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{ "date": "2015年03月25日", "open": 16.92, "high": 16.99, "low": 16.82, "close":
16.97, "volume": 565673 },
{ "date": "2015年03月26日", "open": 16.83, "high": 17.56, "low": 16.83, "close":
17.54, "volume": 884523 },
{ "date": "2015年03月27日", "open": 17.58, "high": 18.3, "low": 17.11, "close": 18.3,
"volume": 705626 },
{ "date": "2015年03月30日", "open": 18.5, "high": 19.4, "low": 18.4, "close": 19.05,
"volume": 1151620 },
{ "date": "2015年03月31日", "open": 19.08, "high": 20.58, "low": 18.4, "close": 19.75,
"volume": 2020679 },
{ "date": "2015年04月01日", "open": 19.69, "high": 19.69, "low": 18.55, "close":
18.65, "volume": 961078 },
{ "date": "2015年04月02日", "open": 18.56, "high": 18.66, "low": 17.85, "close": 17.9,
"volume": 884233 },
{ "date": "2015年04月06日", "open": 17.78, "high": 17.94, "low": 17.51, "close":
17.66, "volume": 605252 },
{ "date": "2015年04月07日", "open": 17.62, "high": 17.9, "low": 17.53, "close": 17.61,
"volume": 591988 },
{ "date": "2015年04月08日", "open": 17.64, "high": 17.85, "low": 17.32, "close":
17.36, "volume": 618855 },
{ "date": "2015年04月08日", "open": 17.33, "high": 17.54, "low": 17.1, "close": 17.1,
"volume": 761855 },
{ "date": "2015年04月10日", "open": 17.08, "high": 17.36, "low": 17, "close": 17.05,
"volume": 568373 },
{ "date": "2015年04月13日", "open": 17.24, "high": 17.26, "low": 16.81, "close": 17.1,
"volume": 667142 },
{ "date": "2015年04月14日", "open": 17.1, "high": 17.89, "low": 17.02, "close": 17.52,
"volume": 870138 },
{ "date": "2015年04月15日", "open": 17.6, "high": 17.99, "low": 17.5, "close": 17.69,
"volume": 530456 },
{ "date": "2015年04月16日", "open": 17.95, "high": 18, "low": 17.6, "close": 17.82,
"volume": 548730 },
{ "date": "2015年04月17日", "open": 17.75, "high": 17.79, "low": 17.5, "close": 17.79,
"volume": 446373 },
{ "date": "2015年04月20日", "open": 17.63, "high": 17.98, "low": 17.52, "close":
17.93, "volume": 487017 },
{ "date": "2015年04月21日", "open": 17.96, "high": 17.98, "low": 17.71, "close":
17.92, "volume": 320302 },
{ "date": "2015年04月22日", "open": 17.88, "high": 18.33, "low": 17.57, "close":
18.29, "volume": 644812 },
{ "date": "2015年04月23日", "open": 18.29, "high": 18.61, "low": 18.18, "close":
18.28, "volume": 563879 },
{ "date": "2015年04月24日", "open": 18.5, "high": 18.5, "low": 17.61, "close": 17.75,
"volume": 650762 },
{ "date": "2015年04月27日", "open": 17.97, "high": 18.05, "low": 17.45, "close":
17.57, "volume": 437294 },
{ "date": "2015年04月28日", "open": 17.65, "high": 17.79, "low": 17.39, "close": 17.5,
"volume": 224519 },
{ "date": "2015年04月29日", "open": 17.68, "high": 17.68, "low": 17.1, "close": 17.21,
"volume": 495706 },
{ "date": "2015年04月30日", "open": 17.22, "high": 17.3, "low": 17, "close": 17.11,
"volume": 391040 },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{ "date": "2015年05月01日", "open": 17.11, "high": 17.55, "low": 16.85, "close": 17.5,
"volume": 563075 },
{ "date": "2015年05月04日", "open": 17.56, "high": 17.85, "low": 17.3, "close": 17.4,
"volume": 253138 },
{ "date": "2015年05月05日", "open": 17.68, "high": 17.68, "low": 17.09, "close":
17.43, "volume": 290935 },
{ "date": "2015年05月06日", "open": 17.48, "high": 17.48, "low": 17, "close": 17.04,
"volume": 313662 },
{ "date": "2015年05月07日", "open": 17.05, "high": 17.19, "low": 16.92, "close":
17.04, "volume": 360284 },
{ "date": "2015年05月08日", "open": 17.13, "high": 17.21, "low": 16.91, "close": 17.1,
"volume": 297653 },
{ "date": "2015年05月11日", "open": 17.16, "high": 17.44, "low": 17.13, "close":
17.31, "volume": 268504 },
{ "date": "2015年05月12日", "open": 17.28, "high": 17.44, "low": 16.99, "close":
17.24, "volume": 376961 },
{ "date": "2015年05月13日", "open": 17.24, "high": 17.3, "low": 17.06, "close": 17.2,
"volume": 244617 },
{ "date": "2015年05月14日", "open": 17.24, "high": 17.25, "low": 17.02, "close":
17.08, "volume": 252526 },
{ "date": "2015年05月15日", "open": 17.06, "high": 17.16, "low": 16.95, "close":
16.95, "volume": 274783 },
{ "date": "2015年05月18日", "open": 16.95, "high": 17.01, "low": 16.76, "close":
16.87, "volume": 418513 },
{ "date": "2015年05月19日", "open": 16.93, "high": 16.94, "low": 16.6, "close": 16.83,
"volume": 367660 },
{ "date": "2015年05月20日", "open": 16.8, "high": 16.9, "low": 16.65, "close": 16.86,
"volume": 297914 },
{ "date": "2015年05月21日", "open": 16.9, "high": 17.08, "low": 16.79, "close": 16.88,
"volume": 229346 },
{ "date": "2015年05月22日", "open": 16.9, "high": 17.05, "low": 16.85, "close": 17,
"volume": 253279 },
{ "date": "2015年05月26日", "open": 17.03, "high": 17.08, "low": 16.86, "close":
17.01, "volume": 212640 },
{ "date": "2015年05月27日", "open": 17.01, "high": 17.99, "low": 16.87, "close":
17.75, "volume": 857109 },
{ "date": "2015年05月28日", "open": 17.77, "high": 17.77, "low": 17.44, "close":
17.62, "volume": 338482 }
]
```


EnvelopesController.cs

C# copyCode

```
// GET: /Envelopes/
public ActionResult Index()
{
    var model = BoxData.GetDataFromJson();
    return View(model);
}
```

Razor copyCode

```
@using MVCFinancialChart.Models
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@model List<FinanceData>

<script>
    // XYVデータをツールチップに表示するために使用する関数。
    function tooltip(ht) {
        var date, content;

        if (!ht || !ht.item) {
            return '';
        }

        date = ht.item.X ? ht.item.X : null;

        if (wijmo.isDate(date)) {
            date = wijmo.Globalize.formatDate(date, 'MM/dd/yy');
        }

        content =
           '<b>' + ht.name + '</b><br/>' +
           'Date: ' + date + '<br/>' +
           'Y: ' + wijmo.Globalize.format(ht.y, 'n2');
        if (ht.item.Volume) {
            content +=
         '<br/>' +
         'Volume: ' + wijmo.Globalize.format(ht.item.Volume, 'n0');
        }

        return content;
    }
</script>

@(Html.C1().FinancialChart()
.Height(400)
.Width(1000)
.Bind(Model)
.BindingX("X")
.SymbolSize(6)
.ChartType(C1.Web.Mvc.Finance.ChartType.Candlestick)
.Series(sers =>
{
    sers.Add().Binding("High,Low,Open,Close");
    sers.AddEnvelopes().Binding("Close").Period(20).Size(0.03)
.Type(C1.Web.Mvc.Finance.MovingAverageType.Simple)
.Style(svgb => svgb.Stroke("#78AB46"));
})
.Tooltip(t => t.Content("tooltip"))
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
)
```

範囲セレクタ

Copyright © GrapeCity, Inc. All rights reserved.

FinancialChartの**RangeSelector**を使用すると、ユーザーはチャートに表示する特定の範囲のデータを選択できます。ユーザーは、さまざまなタイプの財務チャートに簡単にRangeSelectorを連結できます。これは主に金融業界で、さまざまなデータ範囲に対して株価の分析を実行するために使用されています。

RangeSelectorには左サム（最小値用）と右サム（最大値用）があり、チャートの特定の時間をスクロールできます。ユーザーはRangeSelectorの最小値と最大値を変更し、これらのサムを左側や右側にドラッグして、チャート上でのデータの表示範囲を調整できます。範囲バーでサムを左にドラッグすると値が減少し、右にドラッグする範囲バーの値が増加します。

コードで最小値と最大値の範囲を設定するには、RangeSelectorの**Min**プロパティと**Max**プロパティを使用します。

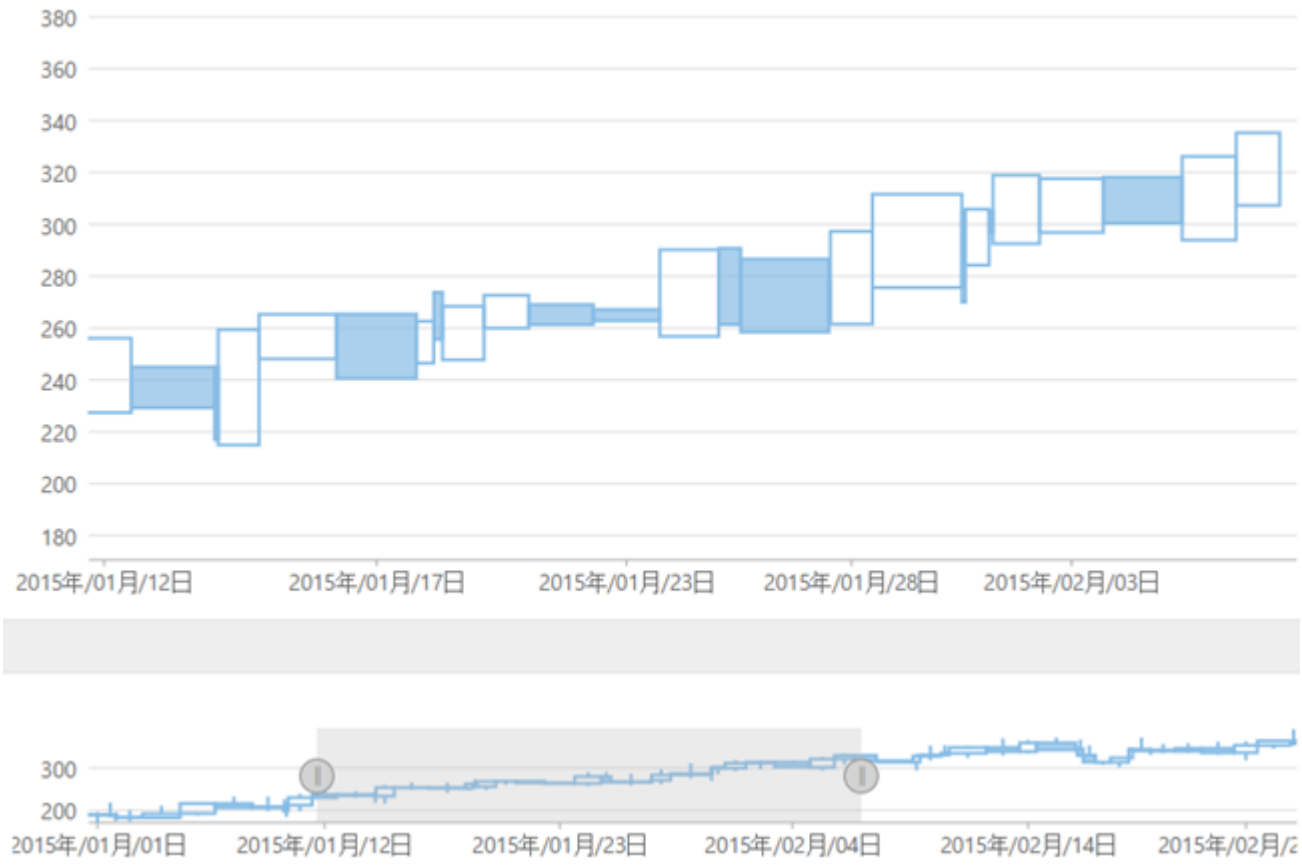
RangeSelectorの**Orientation**プロパティを**X**（デフォルト）または**Y**に設定することで、サムの位置を変更できます。

C#

```
.Orientation(C1.Web.Mvc.Chart.Orientation.Y)
```

向きを**Y**に設定すると、左サムと右サムはRangeSelectorで下サムおよび上サムとして位置が決定されます。これを上下にドラッグすることで値を増減でき、チャートに表示されるデータの範囲を指定できます。

次の図は、RangeSelectorが追加された後のFinancialChartを示しています。



次のコード例は、範囲セレクタを使用してFinancialChartにサム値を表示する方法を示します。

コードの場合

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FData>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
<script type="text/javascript">
    var tooltipContent = function (ht) {
        var item = ht.series.collectionView.items[ht.pointIndex];
        if (item) {
            return 'Date: ' + wijmo.Globalize.format(ht.x, 'MMM-dd') + '<br/>' +
             'High: ' + item.High.toFixed() + '<br/>' +
             'Low: ' + item.Low.toFixed() + '<br/>' +
             'Open: ' + item.Open.toFixed() + '<br/>' +
             'Close: ' + item.Close.toFixed() + '<br/>' +
             'Volume: ' + item.Volume.toFixed() + '<br/>'
        }
    };

    function rangeChanedHandler(sender, e) {
        var stChart = wijmo.Control.getControl("#FinancialChart"),
            rs = wijmo.Control.getControl("#rs").extenders[0];
        if (stChart && rs) {
            stChart.axisX.min = rs.min;
            stChart.axisX.max = rs.max;
            stChart.invalidate();
        }
    }
</script>
@*FinancialChart コントロールを初期化します。*@
@(Html.C1().FinancialChart()
.Id("FinancialChart")
.Width(1000)
.Bind(Model)
.BindingX("X")
    .ChartType(C1.Web.Mvc.Finance.ChartType.EquiVolume)
.Series(sers =>
    {
        sers.Add().Binding("High,Low,Open,Close,Volume");
    }).AxisX(x => x.Format("yyyy年MM月dd日"))
.Tooltip(t => t.Content("tooltipContent")))
<p></p>
<br />
@(Html.C1().FinancialChart()
.Id("rs")
.Width(1000)
.Bind(Model)
.BindingX("X")
    .ChartType(C1.Web.Mvc.Finance.ChartType.CandleVolume)
.Series(sers =>
    {
        sers.Add().Binding("High,Low,Open,Close,Volume");
    })
    //範囲セレクターを追加します。
.AddRangeSelector(rs => rs.OnClientRangeChanged("rangeChanedHandler")
.Orientation(C1.Web.Mvc.Chart.Orientation.X))
    .AxisX(x => x.Format("yyyy年MM月dd日"))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .Width(800)
.Tooltip(t => t.Content("")))
```

近似曲線

近似曲線は、データの傾向を表し、予測の問題点を検討するために使用されます。近似曲線は、通常は価格チャートや財務チャートに対して使用されますが、さまざまなテクニカル分析チャート使用できます。たとえば、MACD（移動平均収束発散法）は株価のテクニカル分析に使用される取引インジケータです。RSI（相対力指数）は金融市場の分析に使用されるテクニカルインジケータです。

近似曲線の種類

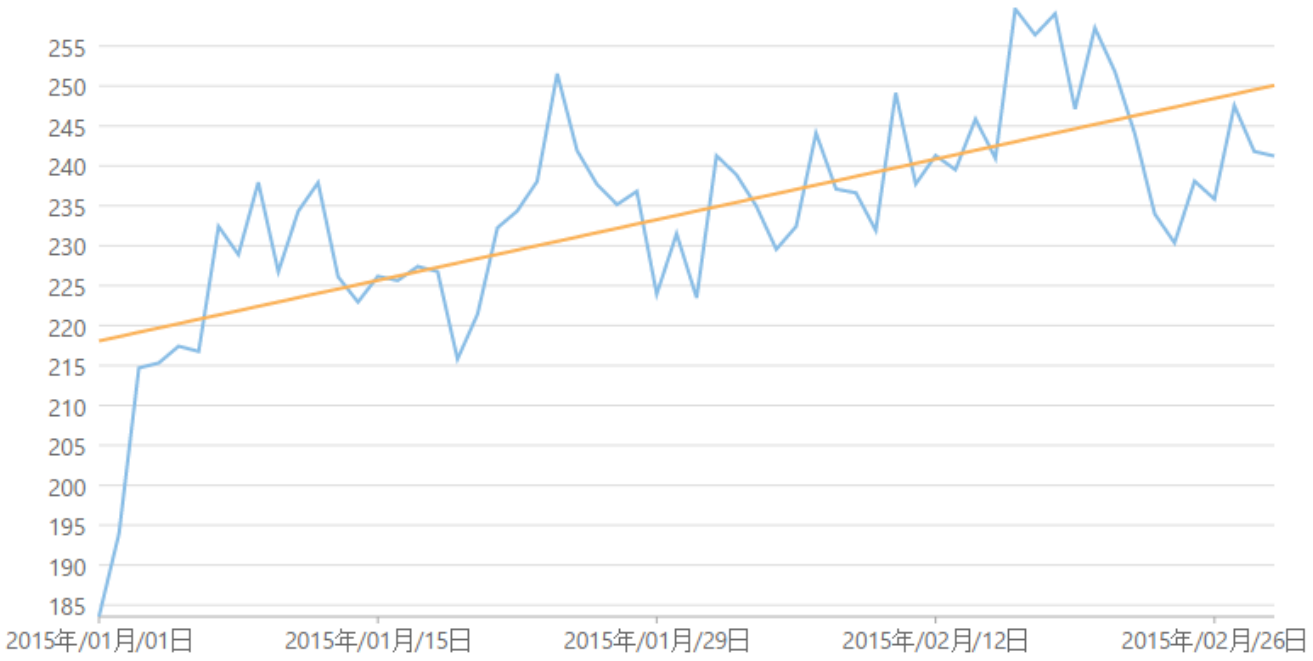
サポートされているFitTypeを次の表に示します。各トレンドタイプは、そのタイプの計算式に基づいて描画されます。

タイプ	説明
Average X	チャートデータからXの平均値を計算し、近似曲線を描画します。
Average Y	チャートデータからYの平均値を計算し、近似曲線を描画します。
Exponential	データ値の増減が徐々に大きくなっていく場合に使用すると便利な曲線です。データにゼロまたは負の値がある場合は、指数近似曲線は作成できません。
Fourier	波のような関数を単純な正弦波の組み合わせとして表示する方法です。フーリエ級数式を使用して作成されます。
Linear	線形近似曲線は、最高の適合度を持つ直線です。データポイントパターンが直線状であれば、データは線形です。R-2乗値が1またはそれに近い場合は線形近似曲線がフィットします。
Logarithmic	最高の適合度を持つ曲線で、データの視覚効果を高めるために使用されます。データの変化率が急速に増加または減少してから平坦化する場合に使用されます。正の値も負の値も使用できます。
Max X	チャートからXの最大値を取り、それを使用して近似曲線を描画します。
Max Y	チャートからYの最大値を取り、それを使用して近似曲線を描画します。
Min X	チャートからXの最小値を取り、それを使用して近似曲線を描画します。
Min Y	チャートからYの最小値を取り、それを使用して近似曲線を描画します
Polynomial	データが振幅の形状を表す場合に使用されるねじれた線です。大きなデータセットに対して損益を分析する際に便利です。
Power	一定の率で増加する計算式に沿ったデータセットでの使用に適しています。たとえば、1秒ごとの車両の加速量などです。

近似曲線をチャートに追加することにより、データのトレンドを表示できます。これを使用すると、Xデータに対するYデータの増加や減少を簡単に分析できます。

次の図は、**FitType**をLineに設定すると近似曲線が**FinancialChart**でどう表示されるかを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、近似曲線を**FinancialChart**に追加する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor copyCode

```
@using MVCFinancialChart.Models

@model List<FData>

<script type="text/javascript">
    var tooltipContent = function (ht) {
        var item = ht.series.collectionView.items[ht.pointIndex];
        if (item) {
            return 'Date: ' + wijmo.Globalize.format(ht.x, 'MMM-dd') + '<br/>' +
             'High: ' + item.High.toFixed() + '<br/>' +
             'Low: ' + item.Low.toFixed() + '<br/>' +
             'Open: ' + item.Open.toFixed() + '<br/>' +
             'Close: ' + item.Close.toFixed() + '<br/>'
        }
    };

</script>

@(Html.C1().FinancialChart()
.Width(1000)
.Bind(Model)
.BindingX("X")
    .ChartType(C1.Web.Mvc.Finance.ChartType.Line)
.Series(sers =>
    {
        sers.Add().Binding("Close");
```


Copyright © GrapeCity, Inc. All rights reserved.

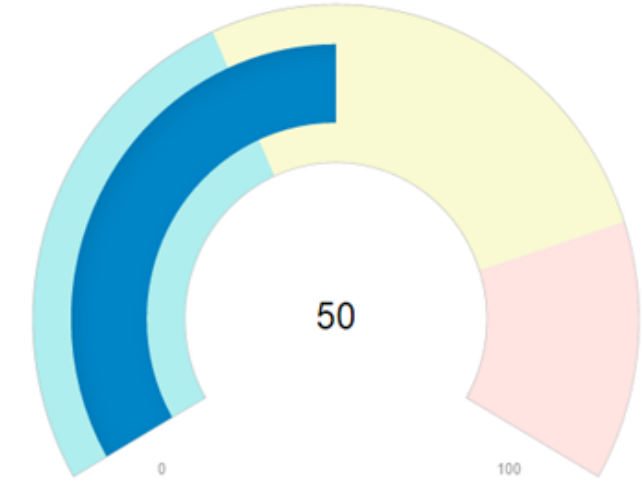
```
        sers.AddTrendLine().Binding("Close");
    }).AxisX(x => x.Format("yyyy年MM月dd日"))
.Tooltip(t => t.Content("tooltipContent")))
```

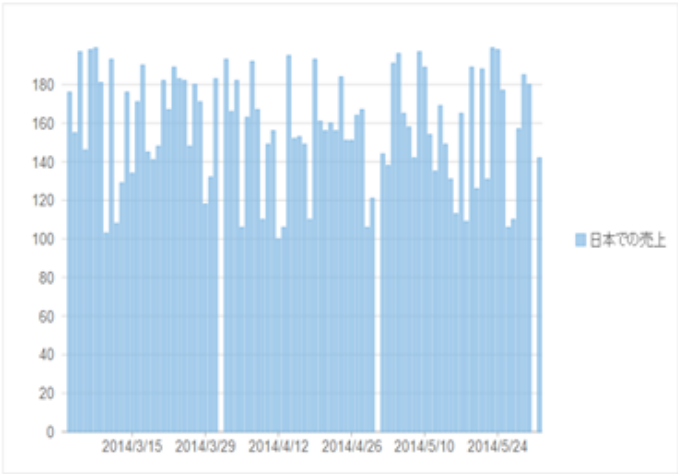
FlexChart

FlexChart コントロールを使用して、Web アプリケーションでデータを視覚的に表現できます。表示するデータのタイプに基づいて、データを縦棒、横棒、バブル、ローソク足、折れ線、散布図として表すことができます。また、データを複数のチャートタイプで表示することもできます。

チャートコントロールにはつきものの複雑な要素が FlexChart によって完全に管理されるため、開発者はアプリケーション固有の重要なタスクに集中できます。

	開始日	終了日	国名	製品名	金額	割引
	1 25 15	00:00	ドイツ	Gadget	¥582	14%
	2 25 15	01:01	カナダ	Gadget	¥4,919	17%
	3 25 15	02:02	日本	Gadget	¥2,160	7%
	4 25 15	03:03	ドイツ	Gadget	¥1,249	22%
	5 25 15	04:04	ドイツ	Gadget	¥4,052	12%
	6 25 15	05:05	カナダ	Gadget	-¥3,131	16%
	7 25 15	06:06	中国	Widget	¥699	15%
	8 25 15	07:07	米国	Widget	¥3,464	3%
	9 25 15	08:08	韓国	Gadget	-¥2,363	14%
	10 25 15	09:09	米国	Widget	-¥2,837	6%





主な特長

FlexChartコントロールには、次の主要機能が含まれています。

- **チャートタイプ**:
  1つのプロパティを設定するだけで、折れ線グラフを横棒グラフなどの他のチャートタイプに変更できます。FlexChart は、11 種類のチャートタイプをサポートします。
- **ツールチップ**:

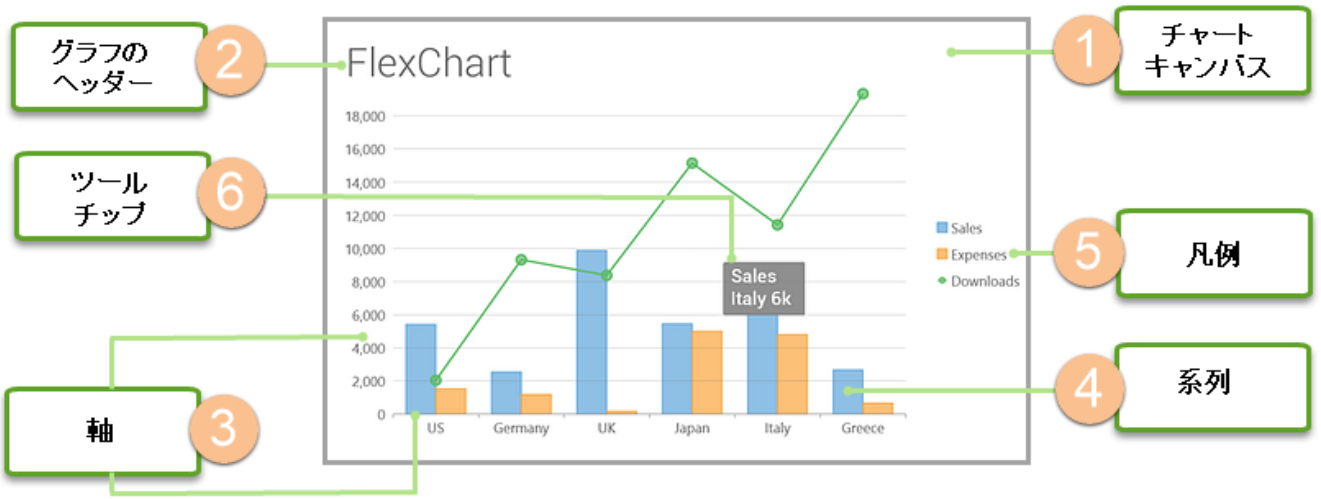
Copyright © GrapeCity, Inc. All rights reserved.

ツールチップを使用してチャートの値を表示できます。

- **複数の系列**:
  1つのチャートに複数の系列を追加できます。
- **ヘッダーとフッター**:
  単純なプロパティを使用して、タイトルやフッターテキストを設定できます。
- **凡例**:
  必要に応じて凡例の位置を変更できます。
- **WebGLレンダリング**
  FlexChartでは、パフォーマンスを犠牲にすることなく、最大数十万のデータポイントまでの非常に大きなデータセットをロードすることにより、FlexChartコントロールのパフォーマンスを向上させるWebGLレンダリングがサポートしています。

チャートの要素

FlexChart は、次に示すいくつかの要素で構成されます。



1	**チャートキャンバス**	系列、軸タイトル、凡例などのすべてのチャート要素が配置される領域。
2	**グラフのヘッダー**	チャートの上に表示するテキスト（基本的にチャートの見出しとなるタイトル）。
3	**軸**	2つの主軸 X および Y。ただし、場合によっては副軸を追加することもできます。
4	**系列**	チャートにプロットされるデータのコレクション。
5	**凡例**	チャートに追加される系列の名前と、その系列のデータをプロットするために使用する事前定義されたシンボルおよび色。
6	**ツールチップ**	系列の上にカーソルを置くと表示されるツールチップまたはラベル。

チャートタイプ

Copyright © GrapeCity, Inc. All rights reserved.

要件に応じて、FlexChart コントロールのタイプを変更できます。チャートタイプを変更するには、FlexChart の **ChartType** プロパティを設定します。この場合、FlexChart に複数の系列を追加すると、そのすべての系列が同じチャートタイプになります。複数の系列を追加し、系列ごとに異なる **ChartType** を設定する方法については、「複合チャート」を参照してください。

FlexChart は、10 種類のチャートタイプをサポートします。

面チャート	**横棒チャートと縦棒チャート**	**バブルチャート**	**ローソク足チャート**
ファンネルチャート	**HighLowOpenClose チャート**	**折れ線チャートと折れ線シンボルチャート**	**散布図**
スプラインおよびスプラインシンボルチャート	**スプライン面チャート**		

コードの場合

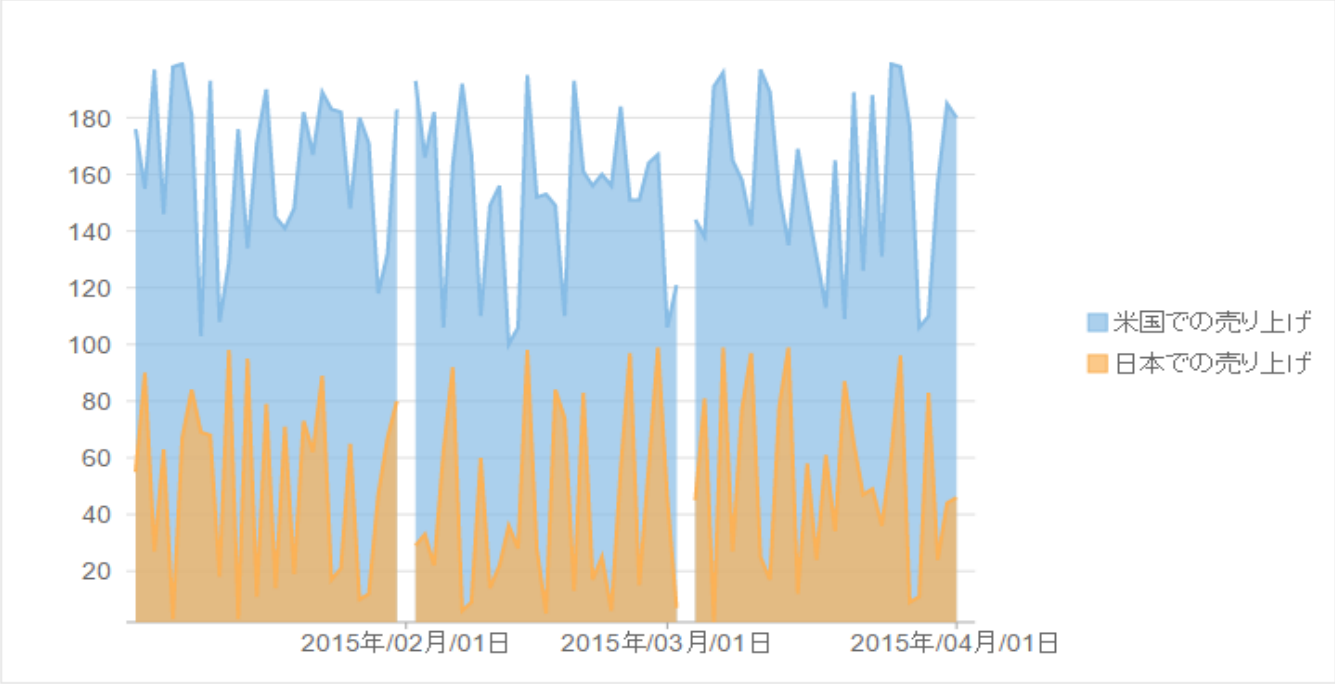
Razor copyCode

```
.ChartType(C1.Web.Mvc.Chart.ChartType.SplineSymbols)
```

面チャート

面チャートは、同じ系列のデータポイントどうしを線で結び、この線の下の領域を色で塗りつぶして量を示します。新しい系列は、それぞれ前の系列の上に描画されます。系列は、個別に描画することも、積み重ねて描画することもできます。

これらのチャートは、関連する属性間の長期にわたるトレンドを示す場合によく使用されます。

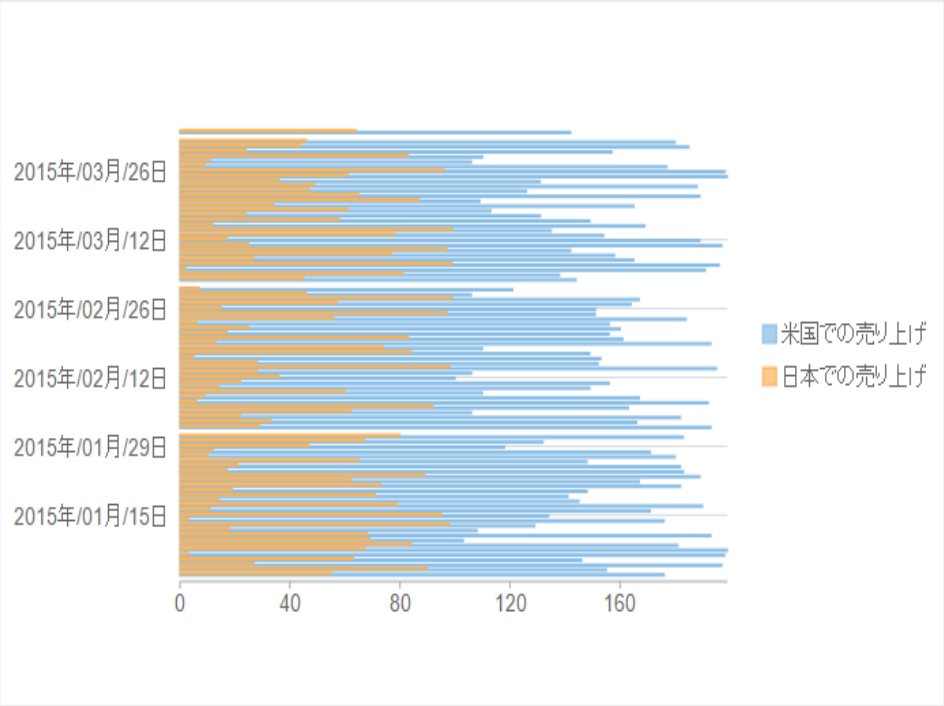
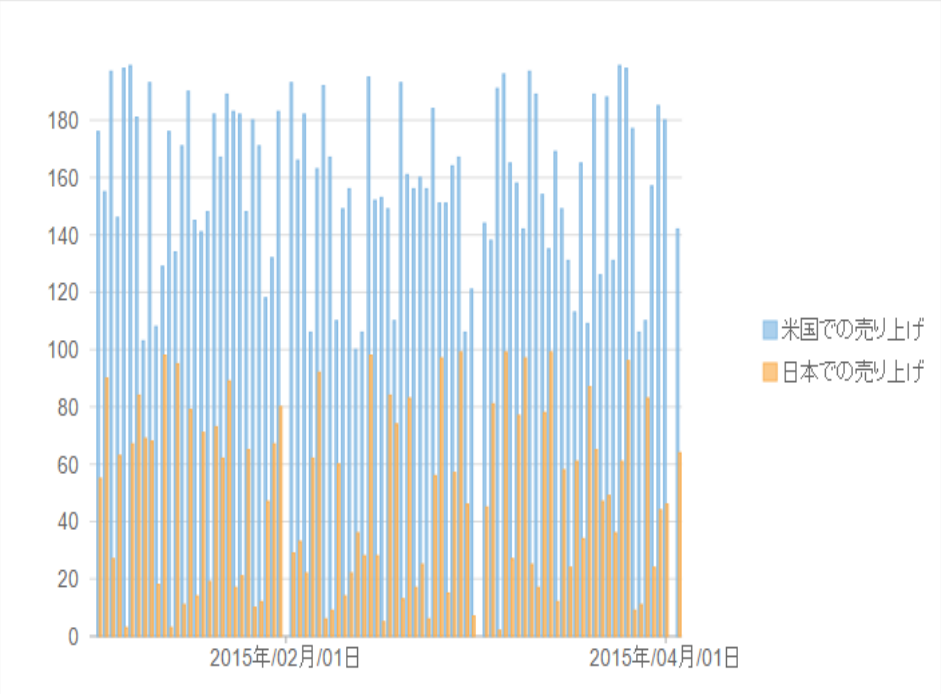


横棒チャートと縦棒チャート

横棒チャートまたは縦棒チャートは、各系列を同じ色と幅の棒の集合として表し、各棒の長さは値によって決定されます。新しい系列は、それぞれ前の系列の棒の集合の隣に棒の集合としてプロットされます。棒が横向きに置かれる場合は横棒チャートと呼ばれ、棒が縦向きに置かれる場合は縦棒チャートと呼ばれます。横棒チャートおよび縦棒チャートは、グループ化するか、積み重ねることができます。

これらのチャートは、年代別や月別などの個別カテゴリにグループ化されたデータを視覚的に表す場合によく使用されます。

Copyright © GrapeCity, Inc. All rights reserved.

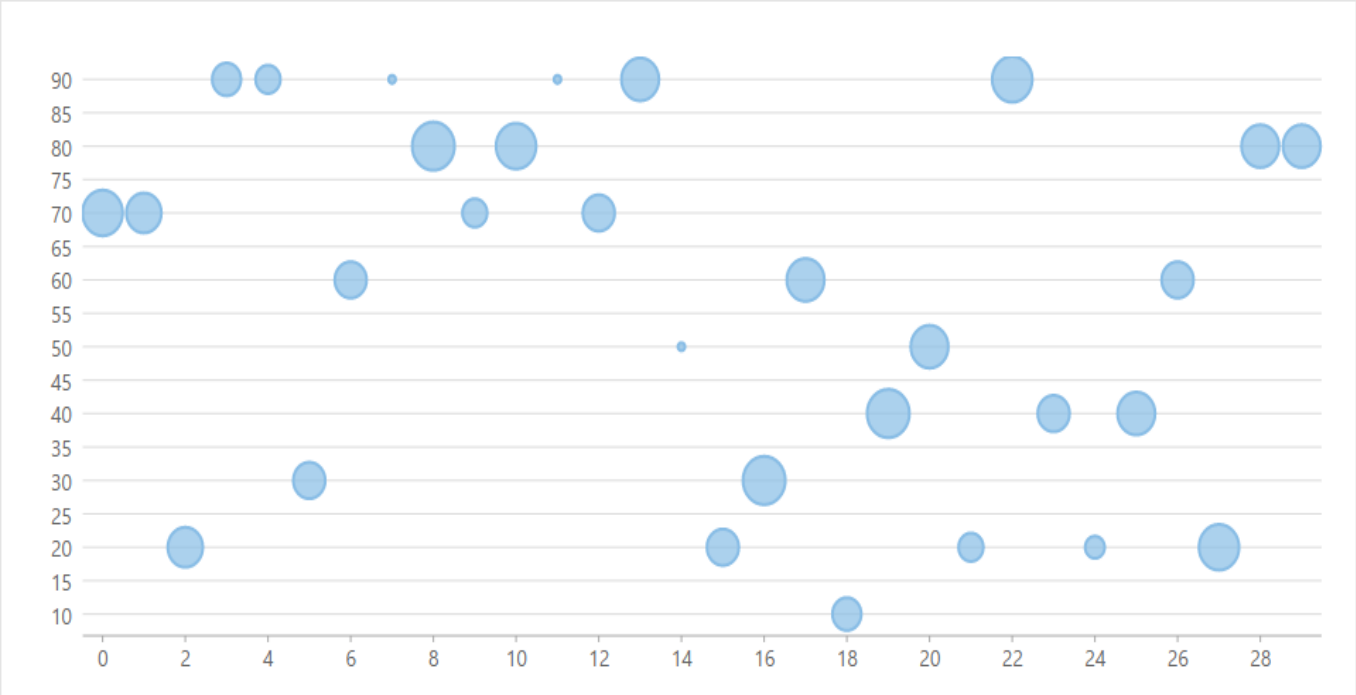
横棒チャート	
縦棒チャート	

バブルチャート

バブルチャートは3次元のデータを表します。X 値と Y 値で2つのデータ次元を示します。3つめの次元は、バブルのサイズによって示されます。

これらのチャートは、軸に対する相対位置とサイズに基づいてエンティティを比較する場合に使用されます。

Copyright © GrapeCity, Inc. All rights reserved.



ローソク足チャート

ローソク足チャートは、特定の株の始値、終値、高値、安値を示す株価チャートです。このチャートは、特殊なタイプの HiLoOpenClose チャートで、始値と終値の関係、および高値と安値の関係を示す場合に使用されます。ローソク足チャートは、価格データ(高値、安値、始値、終値)を使用します。また、始値と終値の関係を追加情報として提供するために、太いローソク部分の色とサイズを使用します。たとえば、長い白抜きのローソクは買い圧力を示し、長い塗りつぶされたローソクは売り圧力を示します。

ローソク足チャートの要素

ローソク足チャートは、**ローソク**、**上ヒゲ**、および**下ヒゲ**で構成されています。

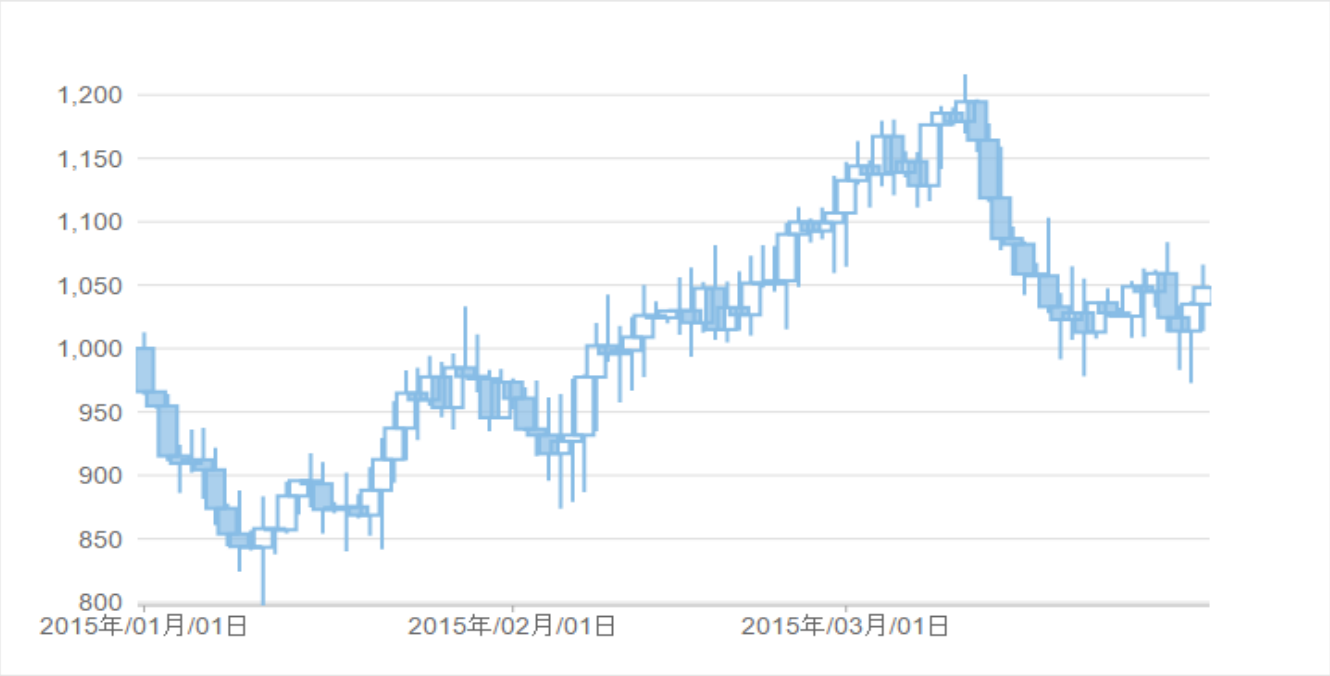
- **ローソク**:ローソクの本体部分(始値と終値の間のバー)は、株価の始値と終値を示しています。
- **上ヒゲと下ヒゲ**:ローソクの上下にある細い線(上ヒゲおよび下ヒゲ)は、株価の高値/安値の範囲を表現しています。
- **空洞の本体**:空洞のローソクは、株価が上昇したこと(終値が始値より高い)を示しています。白抜きのローソクでは、胴体部分の下端が始値を示し、上端が終値を示します。
- **塗りつぶされた本体**:塗りつぶされたローソクは、株価が下降したこと(始値が終値より高い)ことを示しています。塗りつぶされたローソクでは、胴体部分の上端が始値を示し、下端が終値を示します。

ローソク足では、系列内のデータポイントごとに5つの値があります。

- **x**:X 軸に沿ってデータの位置を決定します。
- **high**:その日の最高値を決定し、Y 軸に沿ってローソクの上端としてプロットします。
- **low**:その日の最安値を決定し、Y 軸に沿ってローソクの下端としてプロットします。
- **open**:その日の始値を決定します。
- **close**:その日の終値を決定します。

次の図は、株価を表示するローソク足チャートを示しています。

Copyright © GrapeCity, Inc. All rights reserved.

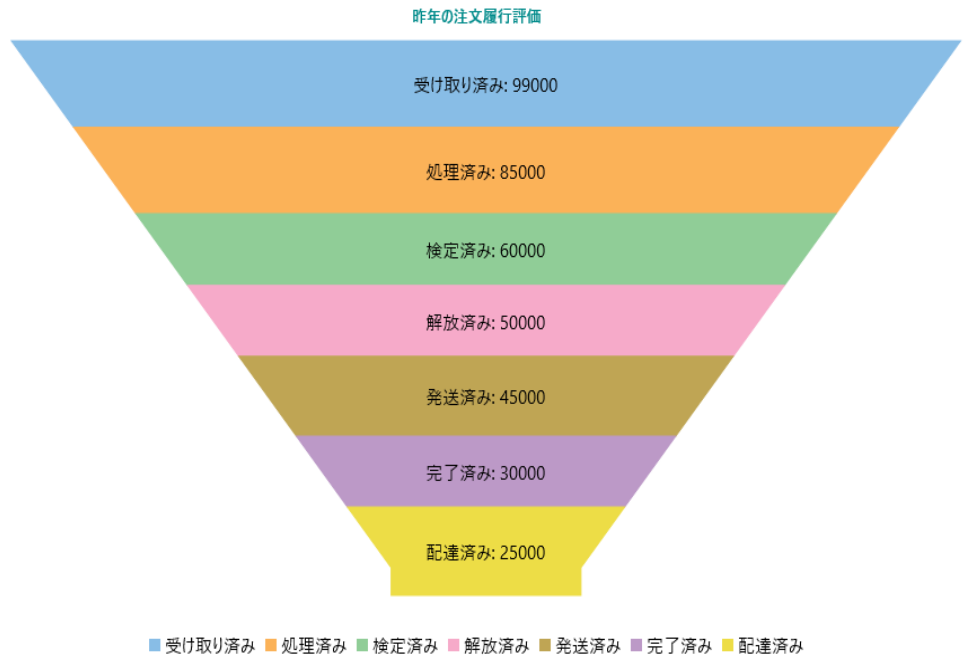


ファンネルチャート

ファンネルチャートを使用すると、1次元プロセスの連続的な段階を表現できます。たとえば、販売見込み客、有望見込み客、売買交渉、成約などの売買プロセスの段階を通して見込み客を追跡するとします。このプロセスの各段階で、全体に対する割合（パーセンテージ）を表します。そのため、チャートは、最初に最大の段階があり、段階が進むごとに小さくなる漏斗の形状になります。ファンネルチャートは、どの段階にどの程度の割合で値が減少しているかが顕著にわかる場所として、プロセス内に潜在的な問題がある領域を特定します。

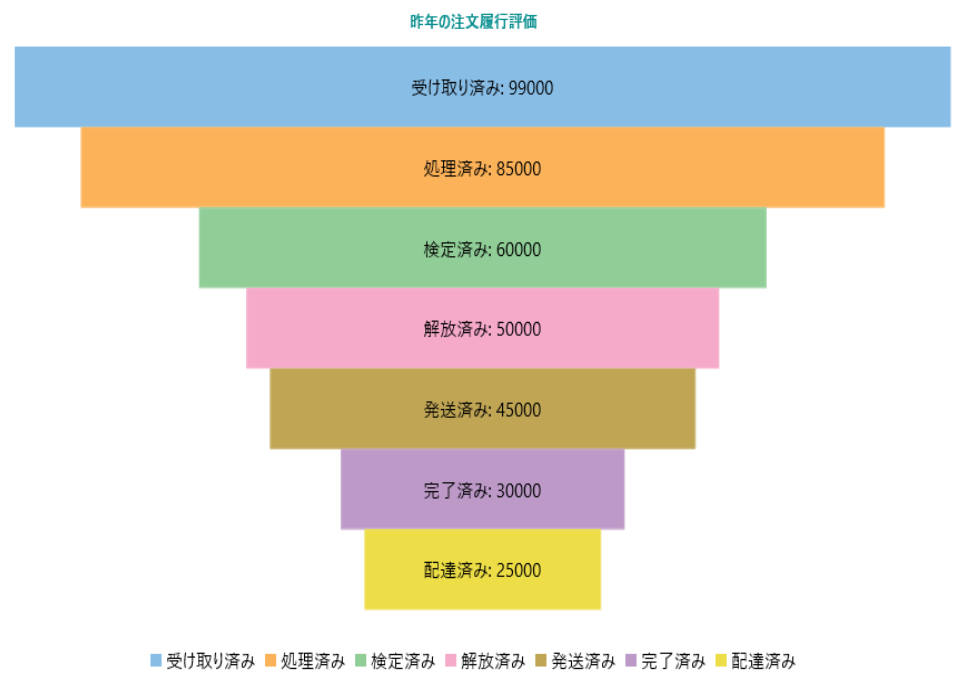
- **台形チャート**:平行な2辺を含みます。
- **積層横棒チャート**:水平の棒の形式で、関連する値を互いの値の上に配置します。

次の図は、受注処理評価プロセスの7つの段階の注文数を示す台形チャートと積層横棒チャートを示します。

台形チャート	

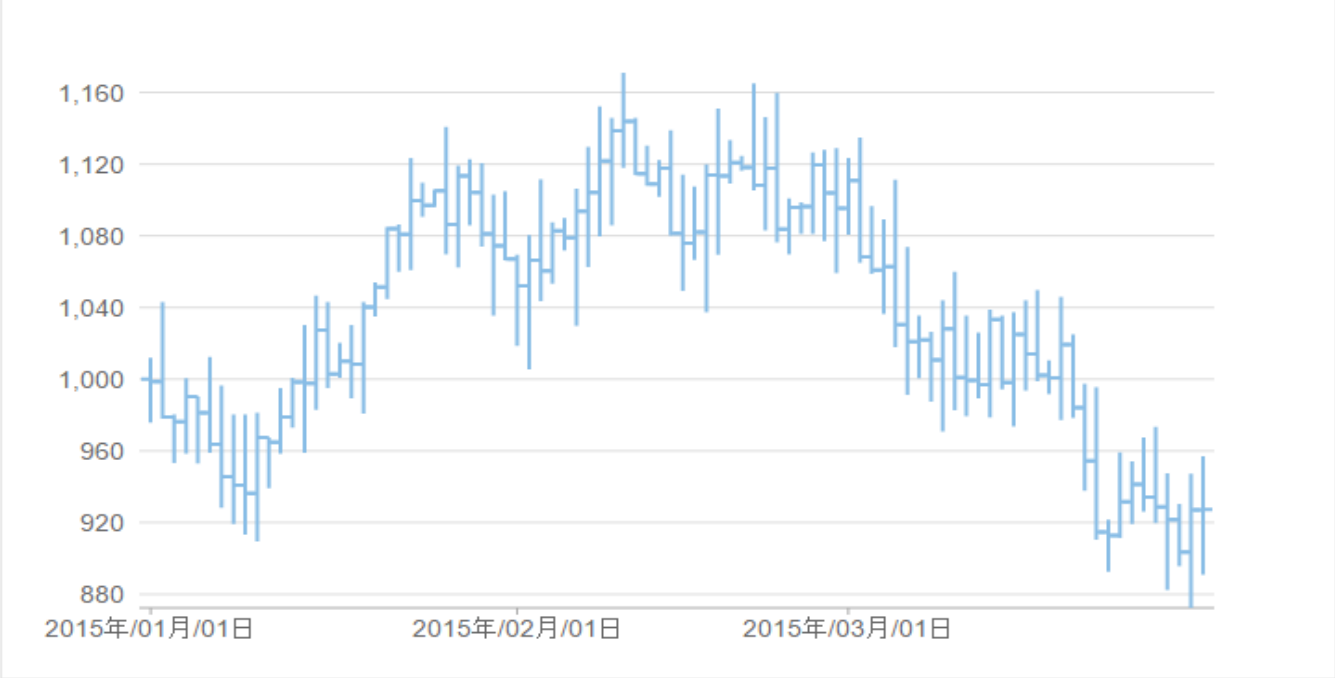
Copyright © GrapeCity, Inc. All rights reserved.

積層横棒チャート



HiLowOpenClose チャート

HiLowOpenClose は、4つの独立した値を組み合わせた株価チャートで、系列内のポイントの高値、安値、始値、および終値を提供します。株の高値および安値を表示するほか、Y2 および Y3 配列要素がそれぞれ株の始値および終値を表します。

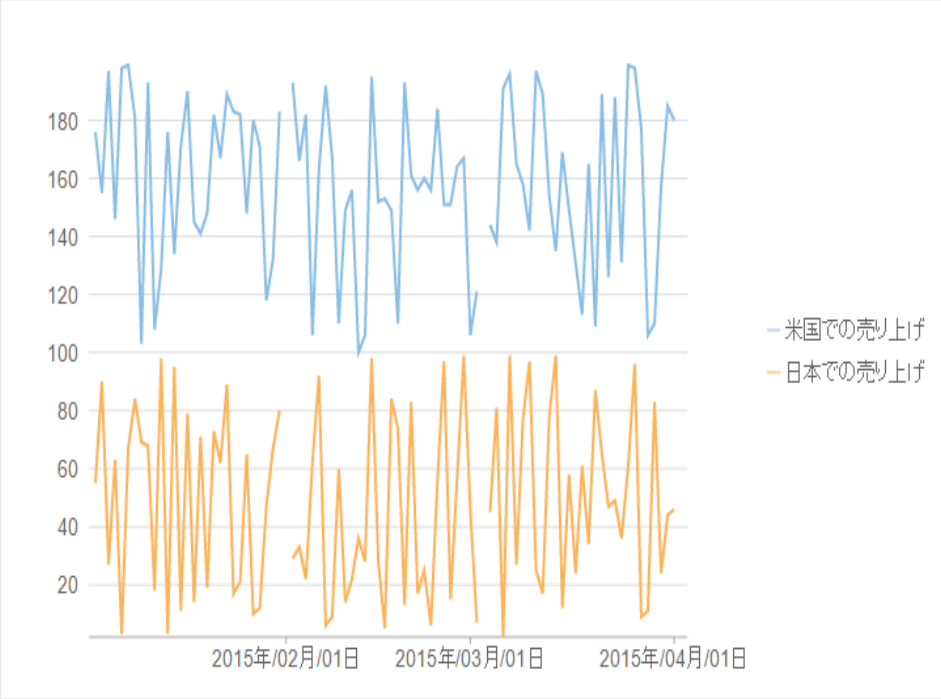
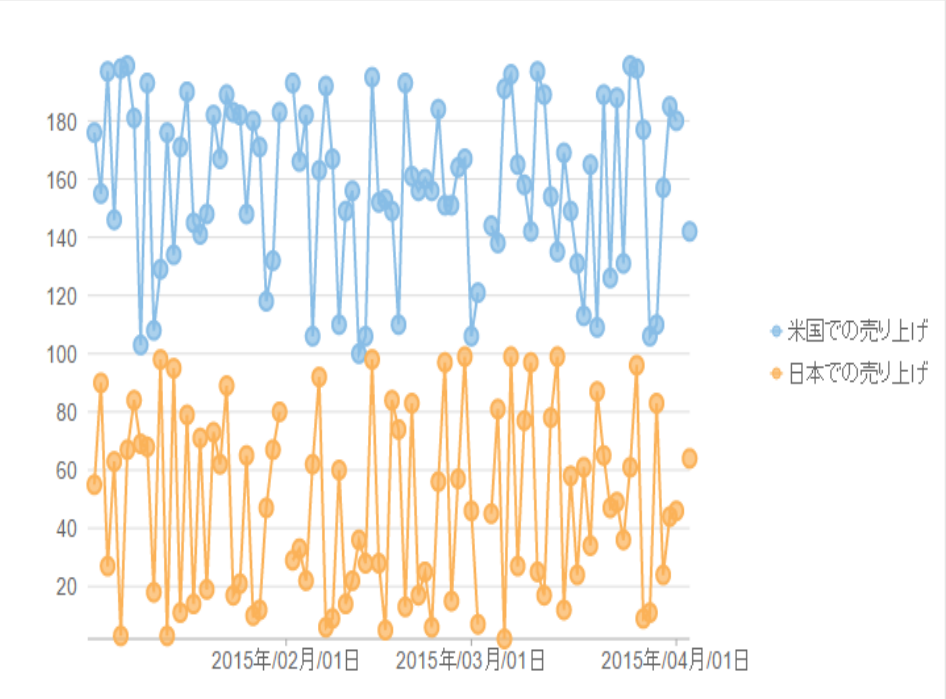


折れ線チャートと折れ線シンボルチャート

折れ線チャートは、データポイントどうしを線で結んで各系列を描画します。面チャートに似ていますが、データポイントどうしを結んだ線の下の領域は塗りつぶされません。系列は、個別に描画することも、積み重ねて描画することもできます。折れ線シンボルチャートは、折れ線チャートに似ていますが、シンボルを使用してデータポイントを表します。これは、さまざまなデータグループ間の値の変化を最も効率よく示す方法です。

これらのチャートは、長期にわたるトレンドやパフォーマンスを示す場合によく使用されます。

Copyright © GrapeCity, Inc. All rights reserved.

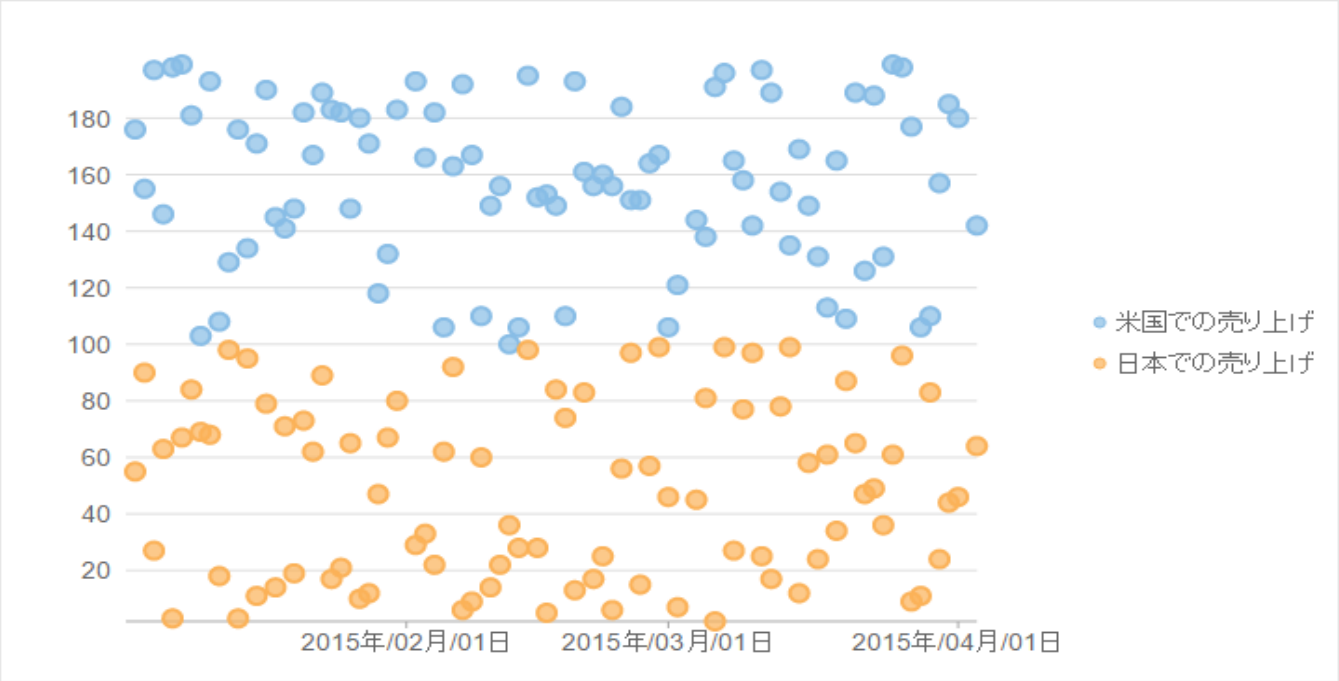
横棒チャート	
縦棒チャート	

散布図

散布図は、X および Y 軸座標を使用してプロットされた点で系列を表します。X 軸と Y 軸を組み合わせて複数のデータポイントにし、それらを不均一な点の集合として表示します。

これらのチャートは、さまざまな X および Y 座標値に対するデータポイント密度のばらつきを判断する場合によく使用されます。

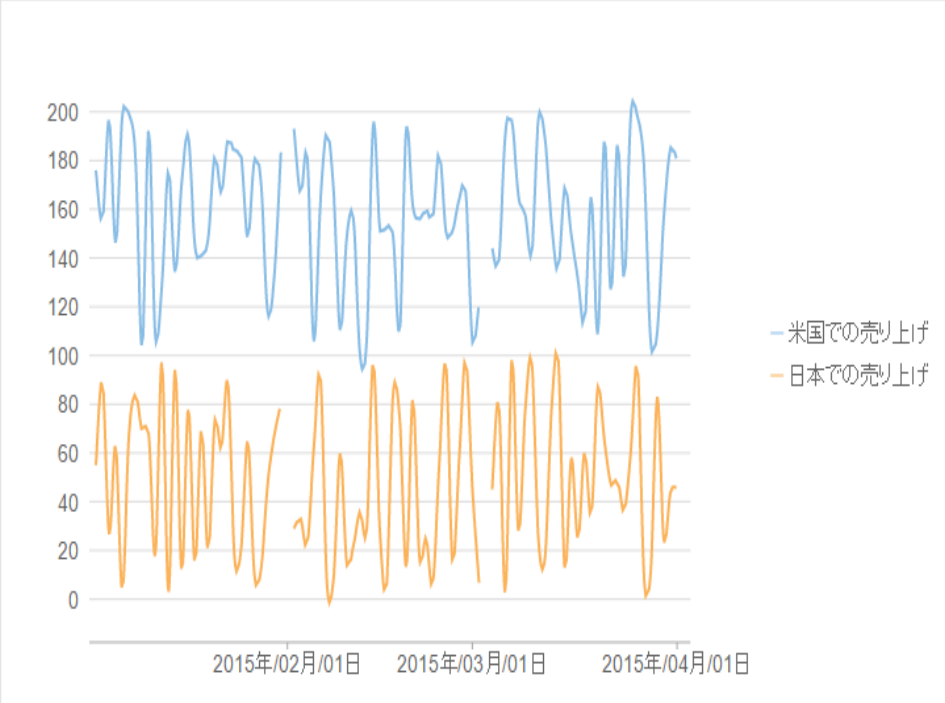
Copyright © GrapeCity, Inc. All rights reserved.



スプラインおよびスプラインシンボルチャート

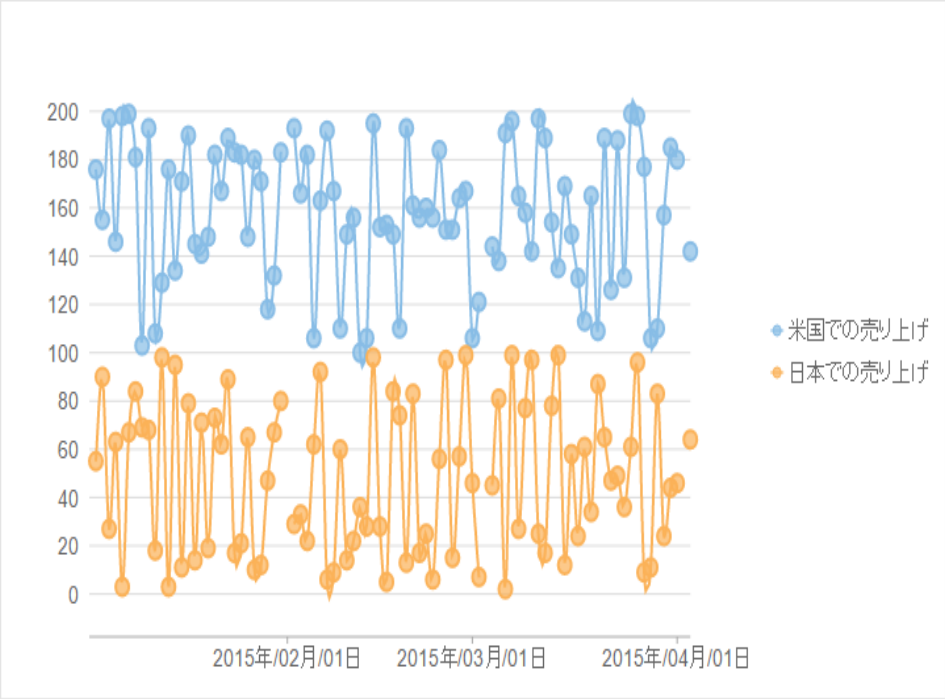
スプラインチャートは、同じ系列のデータポイントどうしを線で結びます。折れ線チャートに似ていますが、ポイント間は滑らかな曲線で結ばれる点が異なります。鋭い角や、カーブの強さに急激な変化はありません。スプラインシンボルチャートは、スプラインチャートに似ていますが、シンボルを使用してデータポイントを表します。これは、さまざまなデータグループ間の値の変化を最も効率よく示す方法です。

これらのチャートは、曲線によって表現する必要がある製品寿命などのデータをプロットする場合によく使用されます。

横棒チャート	

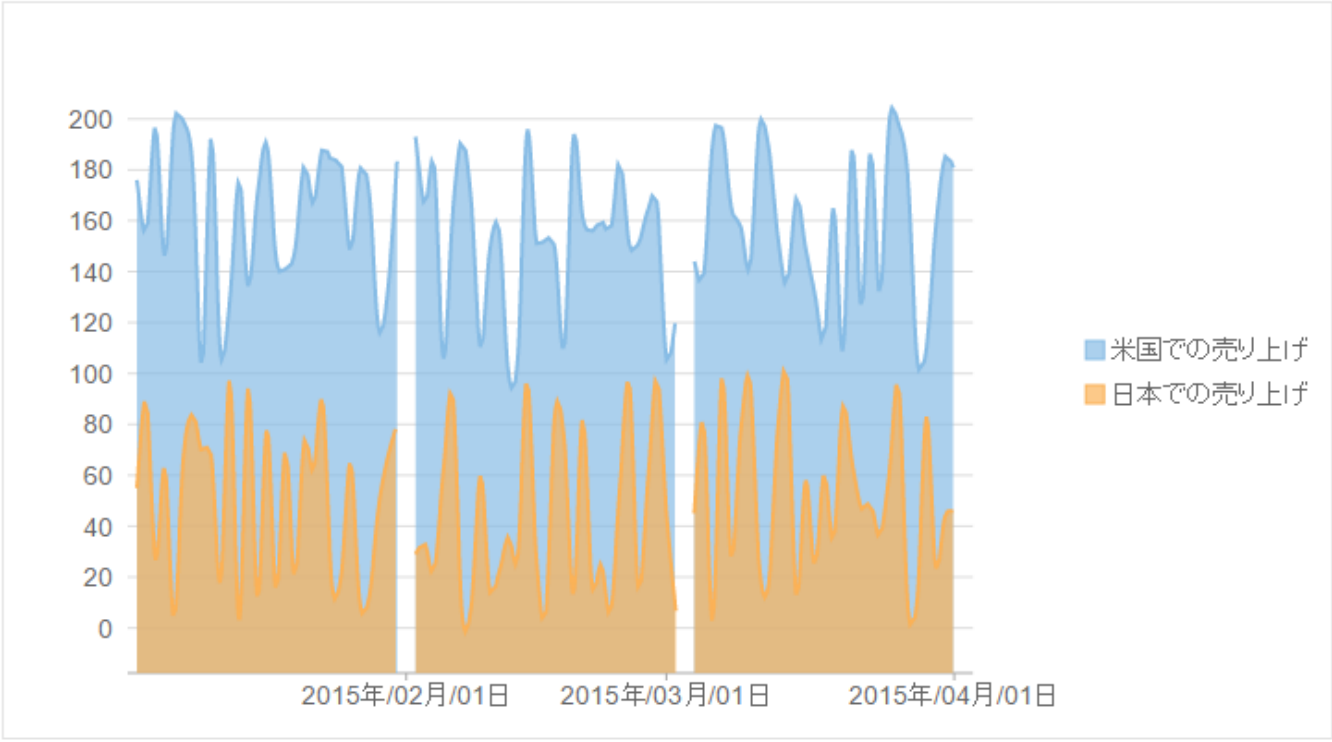
Copyright © GrapeCity, Inc. All rights reserved.

縦棒チャート



スプライン面チャート

スプライン面チャートは、線の下の領域が色で塗りつぶされたスプラインチャートです。スプライン面チャートは面チャートと同様に面を表示するに使用されていますが、スプライン面はデータ点を結ぶには、スプラインを使用すると面チャートは線を使用します。



クイックスタート：FlexChart へのデータの追加

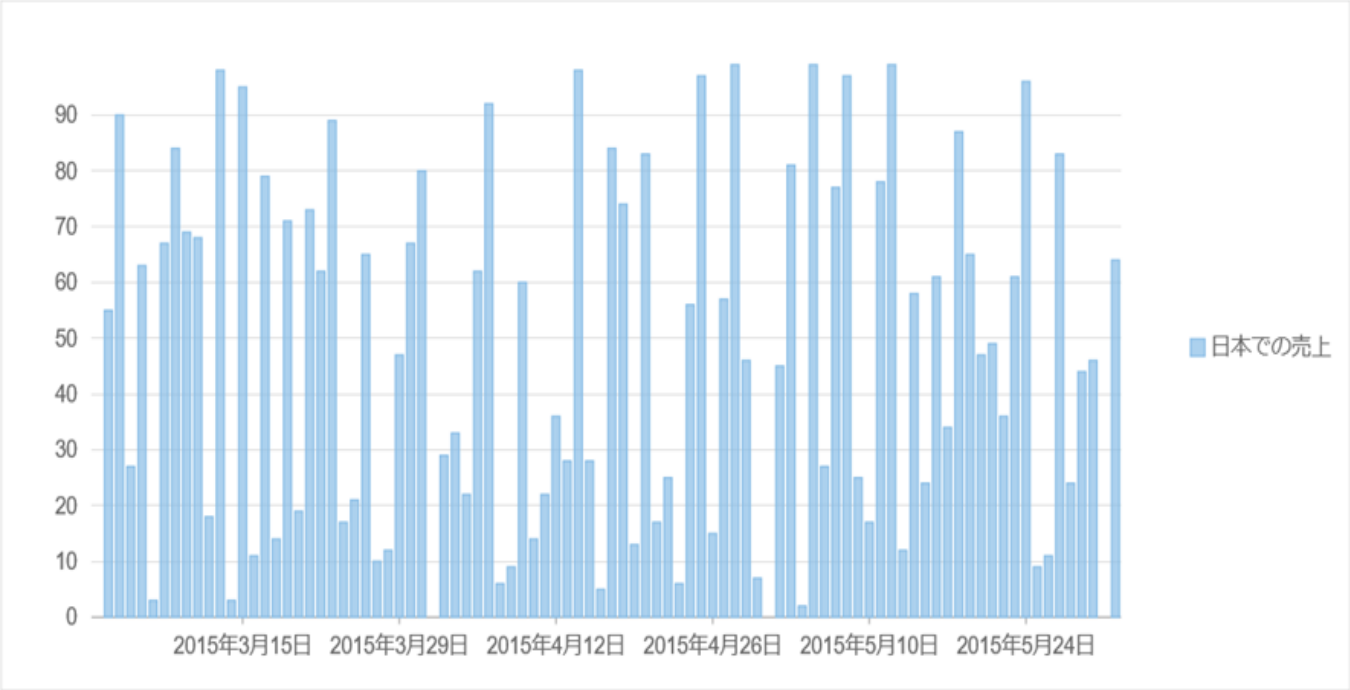
このセクションでは、MVC Web アプリケーションに **FlexChart** コントロールを追加し、そこにデータを追加する方法について説明します。

Copyright © GrapeCity, Inc. All rights reserved.

このトピックは3つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:FlexChart のデータソースの作成**
- **手順 3:FlexChart コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の FlexChart を示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:FlexChart のデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`FlexChartDataSource.cs`)。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FlexChart コントロールのデータソースになるクラスを定義します。

C#

Index.cshtml copyCode

```
public class FlexChartDataSource
{

public string Name { get; set; }

public int MarPrice { get; set; }
public int AprPrice { get; set; }
public int MayPrice { get; set; }

private IEnumerable<FruitSale> _sales = null;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public IEnumerable<FruitSale> Sales
    {
        get
        {
            if (_sales == null)
            {
                _sales = GetSales();
            }
            return _sales;
        }
    }


    public static IEnumerable<FlexChartDataSource> GetFruitsSales()
    {
        var rand = new Random(0);
        var fruits = new[] { "オレンジ", "リンゴ", "なし(梨)", "バナナ", "パイナップル" };
        var list = fruits.Select((f, i) =>
        {
            int mar = rand.Next(1, 6);
            int apr = rand.Next(1, 9);
            int may = rand.Next(1, 6);
            return new FlexChartDataSource { Name = f, MarPrice = mar, AprPrice
= apr, MayPrice = may };
        });

        return list;
    }

    private IEnumerable<FruitSale> GetSales()
    {
        var rand = new Random(0);
        var today = DateTime.Now.Date;
        var firstDay = new DateTime(today.Year-2013, 1, 1);
        var dataTimes = new List<DateTime>();
        for (int i = 0; i < 92; i++)


        {
            dataTimes.Add(firstDay.AddDays(i + 1));
        }
        var list = dataTimes.Select((date, i) =>



        {
            FruitSale sale = new FruitSale { Date = date };
            sale.SalesInChina = rand.Next(150, 250);
            if (i % 30 != 0)
            {
                sale.SalesInUSA = rand.Next(100, 200);
                sale.SalesInJapan = rand.Next(0, 100);
            }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            else
            {
                sale.SalesInUSA = null;
                sale.SalesInJapan = null;
            }

            return sale;
        });

        return list;
    }
}

public class FruitSale
{
    public DateTime Date { get; set; }
    public int? SalesInUSA { get; set; }
    public int? SalesInChina { get; set; }
    public int? SalesInJapan { get; set; }
}
```

VB

Index.vbhtml

```
Public Class FlexChartDataSource
    Public Property Name() As String
        Get
            Return m_Name
        End Get
        Set
            m_Name = Value
        End Set
    End Property
    Private m_Name As String

    Public Property MarPrice() As Integer
        Get
            Return m_MarPrice
        End Get
        Set
            m_MarPrice = Value
        End Set
    End Property
    Private m_MarPrice As Integer
    Public Property AprPrice() As Integer
        Get
            Return m_AprPrice
        End Get
        Set
            m_AprPrice = Value
        End Set
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    End Property
    Private m_AprPrice As Integer
    Public Property MayPrice() As Integer
        Get
            Return m_MayPrice
        End Get
        Set
            m_MayPrice = Value
        End Set
    End Property
    Private m_MayPrice As Integer

    Private _sales As IEnumerable(Of FruitSale) = Nothing
    Public ReadOnly Property Sales() As IEnumerable(Of FruitSale)
        Get
            If _sales Is Nothing Then
                _sales = GetSales()
            End If
            Return _sales
        End Get
    End Property

    Public Shared Function GetFruitsSales() As IEnumerable(Of
FlexChartDataSource)
        Dim rand = New Random(0)
        Dim fruits = New String() {"オレンジ", "リンゴ", "なし(梨)", "バナナ",

"パイナップル"}
        Dim list = fruits.[Select](Function(f, i)
           Dim mar As Integer = rand.[Next](1, 6)
           Dim apr As Integer = rand.[Next](1, 9)
           Dim may As Integer = rand.[Next](1, 6)
           Return New FlexChartDataSource() With {
               Key.Name = f,
               Key.MarPrice = mar,
               Key.AprPrice = apr,
               Key.MayPrice = may
           }

       End Function)

        Return list
    End Function

    Private Function GetSales() As IEnumerable(Of FruitSale)
        Dim rand = New Random(0)
        Dim today = DateTime.Now.[Date]
        Dim firstDay = New DateTime(today.Year - 1, 3, 1)
        Dim dataTimes = New List(Of DateTime)()
        For i As Integer = 0 To 91
            dataTimes.Add(firstDay.AddDays(i + 1))
        Next
        Dim list = dataTimes.[Select](Function([date], i)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            Dim sale As New FruitSale() With {
                Key.[Date] = [date]
            }
            sale.SalesInChina = rand.[Next](150, 250)
            If i Mod 30 <> 0 Then
                sale.SalesInUSA = rand.[Next](100, 200)
                sale.SalesInJapan = rand.[Next](0, 100)
            Else
                sale.SalesInUSA = Nothing
                sale.SalesInJapan = Nothing
            End If

            Return sale

        End Function)

        Return list
    End Function
End Class

Public Class FruitSale
    Public Property [Date]() As DateTime
        Get
            Return m_Date
        End Get
        Set
            m_Date = Value
        End Set
    End Property
    Private m_Date As DateTime
    Public Property SalesInUSA() As System.Nullable(Of Integer)
        Get
            Return m_SalesInUSA
        End Get
        Set
            m_SalesInUSA = Value
        End Set
    End Property
    Private m_SalesInUSA As System.Nullable(Of Integer)
    Public Property SalesInChina() As System.Nullable(Of Integer)
        Get
            Return m_SalesInChina
        End Get
        Set
            m_SalesInChina = Value
        End Set
    End Property
    Private m_SalesInChina As System.Nullable(Of Integer)
    Public Property SalesInJapan() As System.Nullable(Of Integer)
        Get
            Return m_SalesInJapan
        End Get
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        Set
            m_SalesInJapan = Value
        End Set
    End Property
    Private m_SalesInJapan As System.Nullable(Of Integer)
End Class
```

先頭に戻る

手順 3：FlexChart コントロールの追加

FlexChart コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`QuickStartController`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すように MVC 参照を追加します。

```
C#
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc.Chart;
```

5. メソッド `Index()` を次のメソッドに置き換えます。

C#

```
C#                                                                    copyCode
public ActionResult QuickStart()
{
    // データソースを設定します
    FlexChartDataSource ds = new FlexChartDataSource();
    return View(ds.Sales);
```

VB

```
VB
Public Function QuickStart() As ActionResult
        ' データソースを設定します
        Dim ds As New FlexChartDataSource()
        Return View(ds.Sales)
End Function
```

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー `QuickStartController`をダブルクリックして開きます。

Copyright © GrapeCity, Inc. All rights reserved.

2. メソッド `QuickStart()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **QuickStart** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。
6. FlexChart コントロールをビュー `QuickStart` で次のようにインスタンス化します。

Razor copyCode

```
@using MVCFlexChart_JP.Models
@model IEnumerable<FruitSale>
@(Html.C1().FlexChart()
.Bind("Date", Model)
.ChartType(C1.Web.Mvc.Chart.ChartType.SplineSymbols)
.Series(sers =>
  {
      sers.Add()
     .Binding("SalesInJapan")
     .Name("日本での売上");
  })
)
```

Index.vbhtml

```
@Imports C1.Web.Mvc
@Imports C1.Web.Mvc.Chart
@Imports System.Drawing

@ModelType IEnumerable(Of FruitSale)

@(Html.C1().FlexChart() _
    .Bind("Date", Model) _
    .ChartType(C1.Web.Mvc.Chart.ChartType.Column) _
    .Series(Sub(ser)
                ser.Add() _
                .Binding("SalesInJapan") _
                .Name("日本での売上")
        End Sub) _)
```

先頭に戻る

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**QuickStart/QuickStart**）、ビューを表示します。

先頭に戻る

MVCチャートの比較

チャートタイプ

Copyright © GrapeCity, Inc. All rights reserved.

チャート タイプ	FlexChart	MSチャート
面	✓	✓
積層面	✓	✓
100%積層面	✓	✓
スプライン面	✓	✓
積層スプライン面	✓	
100%積層スプライン面	✓	
横棒	✓	✓
積層横棒	✓	✓
100%積層横棒	✓	✓
バブル	✓	✓
ローソク足	✓	✓
縦棒	✓	✓
積層縦棒	✓	✓
100%積層縦棒	✓	✓
株価/High-Low-Open-Close	✓	✓
Sunburst	✓	✓
折れ線	✓	✓
積層折れ線	✓	
100%積層折れ線	✓	
スプライン	✓	✓
積層スプライン	✓	
100%積層スプライン	✓	
折れ線シンボル	✓	
積層折れ線シンボル	✓	
100%積層折れ線シンボル	✓	
スプラインシンボル	✓	
積層スプラインシンボル	✓	
100%積層スプラインシンボル	✓	
円	✓	✓
ドーナツ円	✓	✓
分割円	✓	✓
分割ドーナツ円	✓	✓
点/散布図	✓	✓

Copyright © GrapeCity, Inc. All rights reserved.

2D	✓	✓
平均足	*	
新値足/新値三本足	*	✓
練行足	*	✓
カギ足	*	✓
カラムボリューム	*	
エクイボリューム	*	
ローソクボリューム	*	
アームズローソクボリューム	*	
FastLine		✓
FastPoint		✓
ステップライン		✓
範囲横棒		✓
範囲縦棒		✓
スプライン範囲		✓
範囲		✓
ポイントアンドフィギュア		✓
極座標		✓
レーダー		✓
ボックスプロット		✓
ErrorBar		✓
じょうご		✓
ピラミッド		✓
3D		✓

* FinancialChartで使用可能

データ連結

データ連結	FlexChart	MSチャート
MS SQL Server	✓	✓
OLEDB	✓	✓
ODBC	✓	✓
オブジェクトデータソース	✓	✓
Oracle	✓	✓
SharePointオブジェクト	✓	✓
非連結データ		✓

Copyright © GrapeCity, Inc. All rights reserved.

コア機能

コア機能	FlexChart	MSチャート
空/Nullのデータポイントの処理	✓	✓
ヒットテスト	✓	✓
DirectXレンダリング	✓	
近似曲線	✓	
座標変換メソッド	✓	✓
シリアライズのサポート		✓
データポイントのフィルタ処理		✓
データポイントのソート		✓
グループ化		✓
データポイントの検索		✓
系列データのコピー/結合/分割		✓
データセットへのエクスポート		✓
定義済みの式を使用した統計分析		✓

外見

外見	FlexChart	MSチャート
テーマ	23	12
カスタムパレット	✓	✓
背景色	✓	✓
背景画像	✓	✓
背景グラデーション/ハッチスタイル		✓
境界線および境界線スタイル		✓

チャート領域

チャート領域	FlexChart	MSチャート
ヘッダー	✓	✓
フッター	✓	
マルチヘッダー		✓
ヘッダー/フッターの境界線	✓	✓
ヘッダー/フッターの配置	✓	✓
チャート領域の回転	✓	
複数のチャート領域		✓
表示/非表示の切り替え		✓

Copyright © GrapeCity, Inc. All rights reserved.

プロットエリア

プロットエリア	FlexChart	MSチャート
プロット位置の設定		✓
プロットマージン	✓	
ゾーン	✓	
項目の条件付き書式設定	✓	
ラインマーカー/十字線	✓	
プロット要素のマーカー	FlexChartでは、折れ線シンボル、スプラインシンボル、および散布図のチャートタイプでサポートされています。	✓
マーカーのサイズ	✓	✓
画像のマーカー		✓
マーカー：境界線および境界線スタイル設定		✓
プロット要素の背景画像/グラデーション/ハッチスタイル		✓
プロット要素の境界線および境界線スタイル設定		✓
空のポイントのスタイル設定		✓

データラベル

データラベル	FlexChart	MSチャート
オフセット	✓	
接続線	✓	✓
境界線および境界線スタイル設定	✓	✓
スタイル設定	✓	✓
書式文字列	✓	✓
カスタムコンテンツ	✓	
デカルト座標グラフの位置	下/中央/左/なし/右/上	横棒グラフにのみ適用。外側/左/右/中央
円グラフの位置	中央/内側/外側/なし	内側/外側/なし

軸

Copyright © GrapeCity, Inc. All rights reserved.

軸	FlexChart	MSチャート
軸:第1 X/Y	✓	✓
軸:第2 X/Y	✓	✓
軸:複数の第2 X/Y	✓	
軸ラベル:書式文字列	✓	
軸ラベル:非表示	✓	✓
軸ラベル:スタイル設定	✓	✓
軸範囲(Min/Max)の値	✓	
軸:非表示	✓	✓
軸:対数	✓	✓
軸:逆転	✓	✓
軸線のスタイル設定	✓	✓
ラベル:配置	✓	
ラベル:角度	✓	✓
ラベル:重なっているラベルを非表示にする	✓	
主/副グリッド線	✓	✓
大/小目盛りマーク	✓	✓
ラベル単位/データ単位	✓	✓
目盛りマークの長さ	✓	
目盛りマークのスタイル設定	✓	✓
タイトルおよびタイトルのスタイル設定	✓	✓
原点の設定	任意の値	自動/最小値/最大値
目盛りマークの位置	交差/内側/外側/なし	交差/内側/外側/なし
位置	上/下/左/右/自動/なし	
矢印のスタイル設定		✓
軸ラベル:カスタム		✓
軸ラベル:互い違い		✓
目盛りの区切り		✓

系列

系列	FlexChart	MSチャート
複数の系列	✓	✓
データ連結	✓	✓
系列レベルのチャートタイプ	✓	✓
スタイル設定	✓	✓

Copyright © GrapeCity, Inc. All rights reserved.

XおよびY値関数	✓	
表示/非表示	XおよびY値関数	プロット/プロットと凡例/非表示

凡例

凡例	FlexChart	MSチャート
タイトル		✓
タイトルのスタイル		✓
凡例での系列の表示/非表示の切り替え	✓	
方向	自動/垂直/水平	垂直/水平/テーブル
位置	左/上/右/下/自動/なし	左/上/右/下
カスタム凡例項目		✓
項目の順序付け		✓
複数の凡例		✓

マーカーシンボル

マーカーシンボル	FlexChart	MSチャート
ボックス	✓	✓
ドット	✓	✓
ひし形		✓
三角		✓
十字形		✓
星型4		✓
星型5		✓
星型6		✓
星型10		✓

ユーザー操作

ユーザー操作	FlexChart	MSチャート
ツールチップ	✓	
系列選択	✓	
ポイント選択	✓	
ズーム	✓	
スクロール	✓	
ドラッグ可能な注釈		

ツールチップ

Copyright © GrapeCity, Inc. All rights reserved.

ツールチップ	FlexChart	MSチャート
自動ツールチップ	✓	
カスタムコンテンツ	✓	文字列型のみ
遅延の表示	✓	
スタイル設定	✓	
さまざまなチャート要素のツールチップ		✓

円グラフ

円グラフ	FlexChart	MSチャート
分割されたスライス	✓	✓
内側半径	✓	✓
最初のスライスの開始角度	✓	✓
選択項目の位置	✓	
選択項目のオフセット	✓	

エクスポート/インポート

エクスポート/インポート	FlexChart	MSチャート
JPEG/JPGへのエクスポート	✓	✓
PNGへのエクスポート	✓	✓
SVGへのエクスポート		
BMPへのエクスポート		✓
EMFへのエクスポート		✓
EMfDualへのエクスポート		✓
EMfPlusへのエクスポート		✓
GIFへのエクスポート		✓
TIFFへのエクスポート		✓
バイナリファイルへの保存/ロード		✓
メモリストリームへの保存/ロード		✓
XMLへの保存/ロード		✓
印刷	✓	✓

FlexChartの使用

This section comprises all the features and auxiliary functionalities offered by FlexChart.

注釈

Learn to add annotations in FlexChart.

Copyright © GrapeCity, Inc. All rights reserved.

チャートのアニメーション
Learn how chart animation works in FlexChart.
チャートのジェスチャ
Learn how to enable chart gesture for zooming and panning modes.
グラデーションチャート
Learn how to add gradient colors in FlexChart
外観のカスタマイズ化
Learn how to customize the appearance of the data points in FlexChart.
軸のカスタマイズ
Learn how to customize X and Y axes of FlexChart.
関数系列
Learn how to use Y function series with FlexChart.
ヘッダーとフッター
Learn how to add header and footer in FlexChart and add header and footer styles as well.
ヒットテスト
Learn how to display the X, Y coordinate values and index of the hit point using simple labels.
ラベル
Learn how to add labels to the FlexChart.
凡例
Learn how to display legend in FlexChart.
マーカー
Learn how to add vertical line markers and marker content to the FlexChart.
複合チャート
Learn how to add multiple chart series with different ChartTypes to the FlexChart.
複数の軸
Learn how to add multiple Y axes to the FlexChart.
プロット領域
Learn how to create plot areas in FlexChart.
範囲セレクタ
Learn how to display thumb values using a range selector on FlexChart.
選択
Learn how to select elements of FlexChart.
系列
Learn about Waterfall, ErrorBar and BoxWhisker series and customizing their appearance.
ツールチップ
Learn how to add tooltip in FlexChart.
近似曲線
Learn how to add a trend line to FlexChart.

注釈

注釈は、重要なニュースまたはイベントをマークするために、FinancialChart上の特定のデータポイントにアタッチして使用されます。ユーザーがイベント上にマウスポインタを置くと、注釈の全文が表示されます。FinancialChartのwijbarchart の場合、ユーザーはテキスト、図形、画像などさまざまな種類の注釈をグラフに追加できます。画像とテキスト以外、Wijmoのグラフでは、**円**、**楕円**、**線**、**多角形**、**四角形**、**正方形**などさまざまな組み込み図形が対応されています。

FlexChart上の注釈の位置は、**AnnotationPosition**プロパティを**Bottom**、**Center**、**Left**、**Right**、または**Top**に設定することで指定できます。FlexChartで注釈の添付を指定するには、**AnnotationAttachment**プロパティを使用し、その値を次のように設定します。

- **Absolute**：注釈ポイントは、コントロールのピクセル座標で指定されます。
- **DataCoordinate**：注釈ポイントは、データ座標で指定されます。
- **DataIndex**：注釈ポイントの座標は、データ系列インデックスとデータポイントインデックスで定義されます。
- **Relative**：注釈ポイントは、コントロール内の相対位置で指定されます。(0,0)が左上隅、(1,1)が右下隅になります。

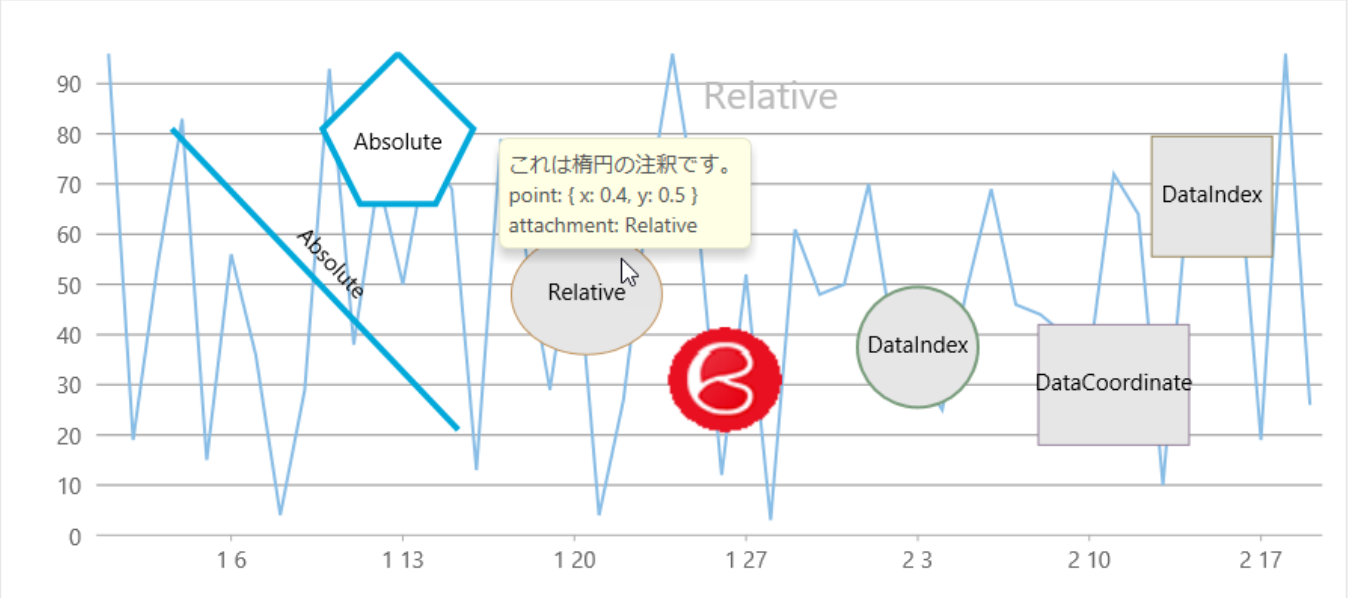
このセクションでは、MVC Webアプリケーションの**FlexChart**コントロールに注釈を追加する方法を説明します。

Copyright © GrapeCity, Inc. All rights reserved.

このトピックは3つの手順で構成されます。

- **手順1：FlexChartのデータソースの作成**
- **手順2：FlexChartコントロールの初期化と注釈の追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexChartを示しています。



手順1：FlexChartのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Sale.cs`）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FlexChartコントロールのデータソースになるクラスを定義します。

C# copyCode

```
public class Sale
{
    public int ID { get; set; }
    public DateTime Start { get; set; }
    public DateTime End { get; set; }
    public string Country { get; set; }
    public string Product { get; set; }
    public string Color { get; set; }
    public double Amount { get; set; }
    public double Amount2 { get; set; }
    public double Discount { get; set; }
    public bool Active { get; set; }

    public MonthData[] Trends { get; set; }
    public int Rank { get; set; }

    private static List<string> COUNTRIES = new List<string> { "US", "UK",
"Canada", "Japan", "China", "France", "German", "Italy", "Korea", "Australia" };
    private static List<string> PRODUCTS = new List<string> { "Widget",
"Gadget", "Doohickey" };

    /// <summary>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    /// データを取得します。
    /// </summary>
    /// <param name="total"></param>
    /// <returns></returns>
    public static IEnumerable<Sale> GetData(int total)
    {
        var colors = new[] { "Black", "White", "Red", "Green", "Blue" };
        var rand = new Random(0);
        var dt = DateTime.Now;
        var list = Enumerable.Range(0, total).Select(i =>
        {
            var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
            var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
            var color = colors[rand.Next(0, colors.Length - 1)];
            var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
            var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60,
i % 60);

            return new Sale
            {
                ID = i + 1,
                Start = startDate,
                End = endDate,
                Country = country,
                Product = product,
                Color = color,
                Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                Amount2 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                Discount = Math.Round(rand.NextDouble() / 4, 2),
                Active = (i % 4 == 0),
                Trends = Enumerable.Range(0, 12).Select(x => new MonthData {
Month = x + 1, Data = rand.Next(0, 100) }).ToArray(),
                Rank = rand.Next(1, 6)
            };
        });
        return list;
    }

    public static List<string> GetCountries()
    {
        var countries = new List<string>();
        countries.AddRange(COUNTRIES);
        return countries;
    }

    public static List<string> GetProducts()
    {
        List<string> products = new List<string>();
        products.AddRange(PRODUCTS);
        return products;
    }
}
```


Copyright © GrapeCity, Inc. All rights reserved.

```
public class BasicSale
{
    public int Sale { get; set; }
    public DateTime Date { get; set; }

    public BasicSale(int sale, DateTime date)
    {
        Sale = sale;
        Date = date;
    }
    public static List<BasicSale> GetBasicSales()
    {
        List<BasicSale> list = new List<BasicSale>();
        int[] sales = {
        96, 19, 54, 83, 15, 56, 36, 4, 29, 93,
        38, 71, 50, 77, 69, 13, 79, 57, 29, 62,
        4, 27, 66, 96, 65, 12, 52, 3, 61, 48, 50,
        70, 39, 33, 25, 49, 69, 46, 44, 40, 35,
        72, 64, 10, 66, 63, 78, 19, 96, 26};
        for (int i = 0; i < sales.Length; i++)
        {
            list.Add(new BasicSale(sales[i], new DateTime(2014, i / 31 + 1, i %
31 + 1)));
        }
        return list;
    }
}
```

先頭に戻る

手順2：FlexChartコントロールの初期化と注釈の追加

FlexChartコントロールを初期化して注釈を追加するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：Default1Controller）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. メソッド`Index()`を次のメソッドに置き換えます。

C# copyCode

```
public ActionResult Index()
{
    return View(BasicSale.GetBasicSales());
}
```

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー`AnnotationController`をダ

Copyright © GrapeCity, Inc. All rights reserved.

ブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。
6. QuickStartビューのFlexChartコントロールを次のようにインスタンス化します。

> 例では、C1logo画像を使用してFlexChartの画像注釈を表示します。画像は、システムの*C:\Users\<username>\Documents\ComponentOne Samples\ASP.NET MVC\MVC\MvcExplorer\Content\images\c1logo.png* に用意されています。

Razor copyCode

```
@using MVCFinancialChart.Models
@using C1.Web.Mvc
@using C1.Web.Mvc.Chart
@using System.Drawing
@model IEnumerable<BasicSale>

@*ツールチップ上で表示するデータを追加します。*@

@{
    var lineTooltip = "線注釈を表します。 </br> start: { x: 50, y: 150 } end: { x:
240, y: 350 } </br> attachment: Absolute";
    var textTooltip = "テキスト注釈を表します。</br> point: { x: 0.55, y: 0.15 }</br>
attachment: Relative";
    var ellipseTooltip = "楕円注釈を表します。</br> point: { x: 0.4, y: 0.5 }</br>
attachment: Relative";
    var rectTooltip = "四角形注釈を表します。</br> point:{ x: new DateTime(2014, 2,
11), y: 30 }</br> attachment: DataCoordinate";
    var imageTooltip = "イメージ注釈を表します。</br> point:{x: new DateTime(2014, 1,
26), y: 30}</br> attachment: DataCoordinate";
}

@*FlexChart を初期化します。*@

@(Html.C1().FlexChart().Width(1000).Bind(Model)
.BindingX("Date").Series(series =>

{
    series.Add().Binding("Sale").ChartType(C1.Web.Mvc.Chart.ChartType.Line);
}).AxisX(x => x.Format("yyyy年/MM月/dd日"))
//注釈レイヤーを追加します。Add Annotation layer
.AddAnnotationLayer(layer =>
{
    //線注釈を追加します。
    layer.AddLine(line => line.Attachment(AnnotationAttachment.Absolute)
        .Position(AnnotationPosition.Center)
        .Content("Absolute")
        .Start(new DataPoint(50, 150))
        .End(new DataPoint(240, 350))
        .Tooltip(lineTooltip)
        .Style(style => style.Stroke("#17EAA5").StrokeWidth(4)
        ));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    //テキスト注釈を追加します。
    layer.AddText(text => text.Attachment(AnnotationAttachment.Relative)
        .Content("Relative")
        .Position(AnnotationPosition.Center)
        .Point(new DataPoint(0.55, 0.15))
        .Tooltip(textTooltip)
        .Style(style => style.Fill("#EA4C17").FontSize(26))
        );
    //楕円注釈を追加します。
    layer.AddEllipse(ellipse =>
ellipse.Attachment(AnnotationAttachment.Relative)
        .Position(AnnotationPosition.Center)
        .Point(new DataPoint(0.4, 0.5))
        .Width(100).Height(80)
        .Content("Relative")
        .Tooltip(ellipseTooltip)
        .Style(style => style.Fill("#EAD617").Stroke("#A1840D"))
        );
    //四角形注釈を追加します。
    layer.AddRectangle(rectangle =>
rectangle.Attachment(AnnotationAttachment.DataCoordinate)
        .Point(new DataPoint(new DateTime(2014, 2, 11), 30))
        .Content("DataCoordinate")
        .Position(AnnotationPosition.Center)
        .Tooltip(rectTooltip)
        .Style(style => style.Fill("#F6C8F0").Stroke("#F947FB"))
        );
    //イメージ注釈を追加します。
    layer.AddImage(image =>
image.Attachment(AnnotationAttachment.DataCoordinate)
        .Href("/Content/images/c1logo.png")
        .Point(new DataPoint(new DateTime(2014, 1, 26), 30))
        .Width(80).Height(80)
        .Tooltip(imageTooltip)
        );
})
)
```

先頭に戻る

手順3：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**Annotation/Index**）。

先頭に戻る

チャートのアニメーション

ChartAnimation は、チャートをロードおよび更新する際に組み込みのアニメーションを提供します。アニメーションは、時間、イージング関数、アニメーションモードなどのプロパティを使用して設定できます。ChartAnimationプロパティは、FlexChart、FinancialChart、FlexPie

Copyright © GrapeCity, Inc. All rights reserved.

で使用できます。

次のコード例は、時間、イージング、アニメーションモードなどのプロパティを使用して、チャートアニメーションを有効にする方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor

```
@using ChartAnimation.Models
@model IEnumerable<FruitSale>
@(Html.C1().FlexChart()
.Bind("Date", Model)
.ChartType(C1.Web.Mvc.Chart.ChartType.Column)
.Series(sers =>
{       sers.Add()
   .Binding("SalesInJapan")
   .Name("日本での売上");
})
.ShowAnimation(ca =>
ca.AnimationMode(AnimationMode.All).AxisAnimation(true).Duration(1000).Easing(Easing.Linear)))
)
```

チャートアニメーションクラス

次のプロパティを使用して、チャートでアニメーションを有効にできます。

名前	タイプ	説明
AnimationMode	アニメーションモード	プロットポイントのアニメーション表示を一度に1つずつ行うか、系列ごとに行うか、または一度にすべて行うかを取得または設定します。アニメーション全体は時間内に完了します。
AxisAnimation	Boolean	アニメーションが軸に適用されるかどうかを示す値を取得または設定します。
Duration	Integer	アニメーション全体の長さ(ミリ秒)を取得または設定します。
Easing	イージング	アニメーションに適用されるイージング関数を取得または設定します。

AnimationMode列挙体

AnimationModeを使用すると、ポイント、系列、またはその両方にアニメーションを適用できます。

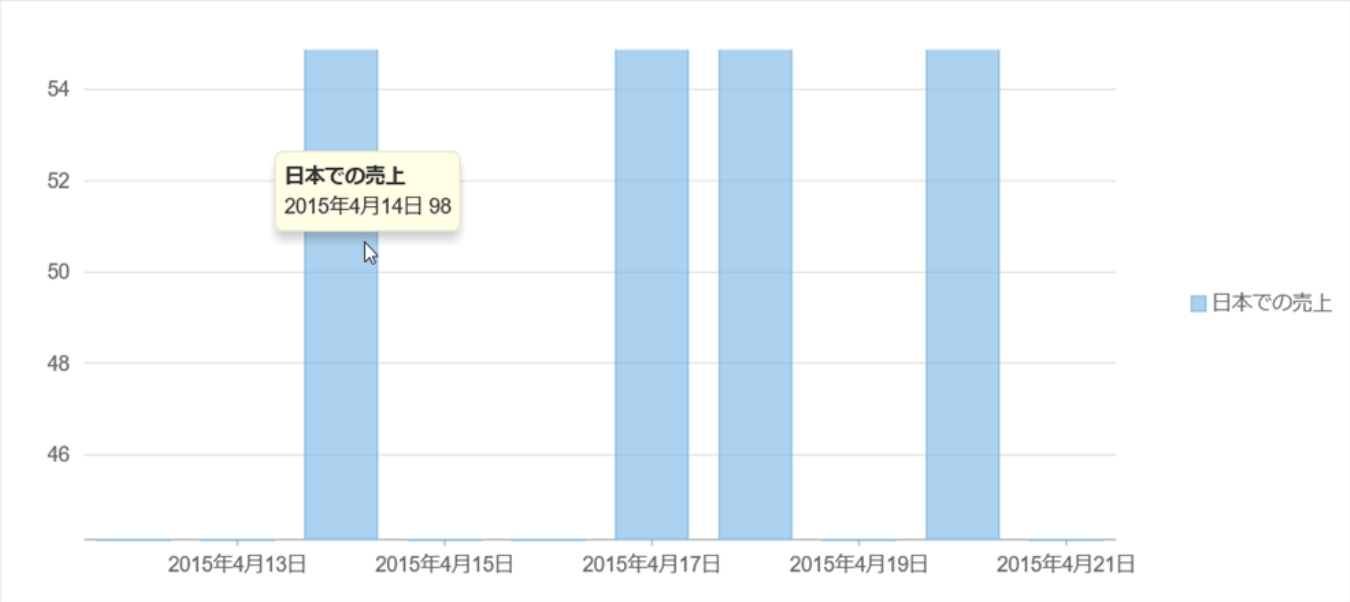
名前	説明
All	すべてのポイントと系列が一度にアニメーションします。
Point	アニメーションがポイントごとに実行されます。複数の系列が同時にアニメーションします。
Series	アニメーションが系列ごとに実行されます。"All"オプションと同様に系列全体が一度にアニメーションされますが、一度に1つの系列だけがアニメーションします。

チャートのジェスチャ

FlexChartでは、ズームとパンという2種類のチャート操作を行うことができます。ズーム機能とパン機能は、データ量が大きい場合に特に重要です。ズームを使用すると、選択したチャート領域を拡大または縮小して表示できます。また、パンを使用すると、チャート領域内をナビゲートできます。ズームとパンはジェスチャもサポートしているため、デバイス上でスムーズにズーム/パン操作を行うことができます。

次の図は、ブラウザでチャートジェスチャを使用してズームした後のFlexChartを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、ズームおよびパンモードに対してチャートジェスチャを有効にする方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor

```
@using ChartAnimation.Models
@model IEnumerable<FruitSale>
@(Html.C1().FlexChart()
.Bind("Date", Model)
.ChartType(C1.Web.Mvc.Chart.ChartType.Column)
.Series(sers =>
{    sers.Add()
   .Binding("SalesInJapan")
   .Name("日本での売上");
})
.SupportGestures(cg =>
cg.InteractiveAxes(InteractiveAxes.XY).MouseAction(MouseAction.Zoom)))
```

グラデーションチャート

グラデーションは、1つの色を別の色にブレンドすることで立体感のある色の表現を生み出すグラフィック効果です。複数の色を使用して、1つの色を徐々に薄くして別の色に変化させることができます。グラデーションはブラウザの機能を利用して生成されるため、グラデーション色を使用した要素やコントロールは綺麗に表示されます。FlexChartでは、スタイル設定にグラデーション色を適用してチャートの見栄えをよくすることができます。

グラデーションのディスクリプタは、次のように書式設定された式です。

HTML

```
<type>(<coords>)<colors>[:<offset>[:<opacity>]]
[-<colors>[:<offset>[:<opacity>]]]-<colors>[:<offset>[:<opacity>]]
```

グラデーションのディスクリプタには、次の要素が含まれます。

- **タイプ** - <type>は、linear(直線状)またはradial(放射状)のいずれかに設定できます。
  - 大文字のLまたはRは、SVGサーフェスからの絶対座標オフセットを示します。
  - 小文字のlまたはrは、グラデーションが適用される要素を基準にして計算される座標を示します。

Copyright © GrapeCity, Inc. All rights reserved.

- **座標**
  - 直線グラデーションベクトルは「x1, y1, x2, y2」で指定します。
  - 放射状グラデーションベクトルは「cx, cy, r」で指定し、オプションで、円の中心から離れた焦点を「fx, fy, fr」で指定します。

- **色** - <colors>は、ダッシュで区切られたCSSカラー値のリストとして指定します。各色の後に、カスタムのオフセットと不透明度の値をコロン文字で区切って指定することができます。

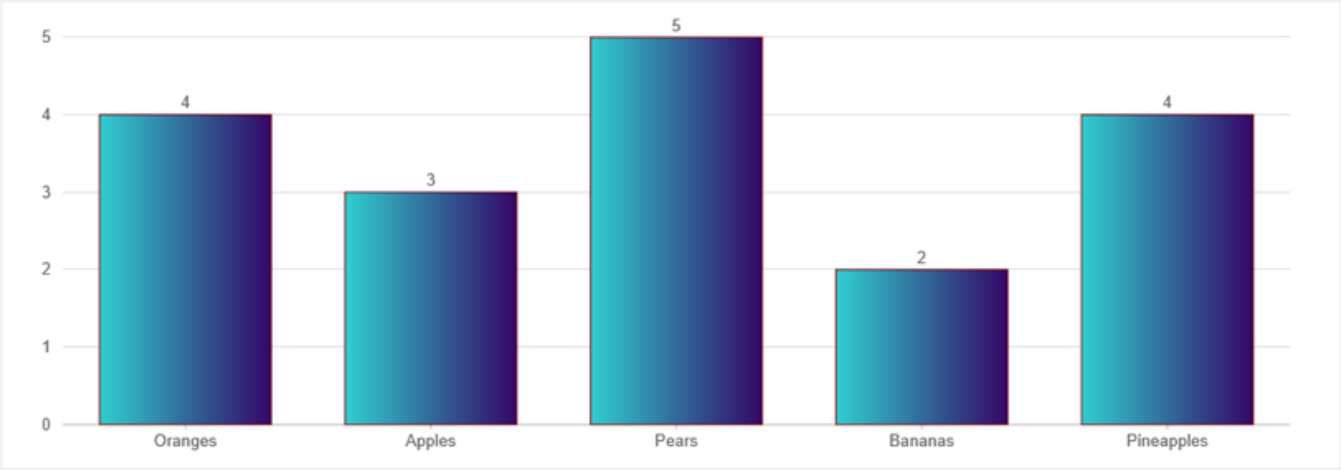
直線グラデーションと放射状グラデーションの例を次に示します。

Linear Gradient Format

```
'l(0,0,1,0)#ff0000-#00ff00-#0000ff', 'L(0,0,300,300)#ff0000:0.2:0.2-00ff00:0.8'
```

Radial Gradient Format

```
'r(0.5,0.5,1)#ff0000-#0000ff', 'R(100,100,100,200,200,200)#ff0000-#0000ff'
```



次のコード例は、FlexChartでグラデーション色を追加する方法を示します。この例では、「FlexChart: クイックスタート」セクションで作成した**Fruit.cs**モデルを使用しています。

Razor

```
@using ChartGradient.Models
@model IEnumerable<Fruit>
@using C1.Web.Mvc.Chart

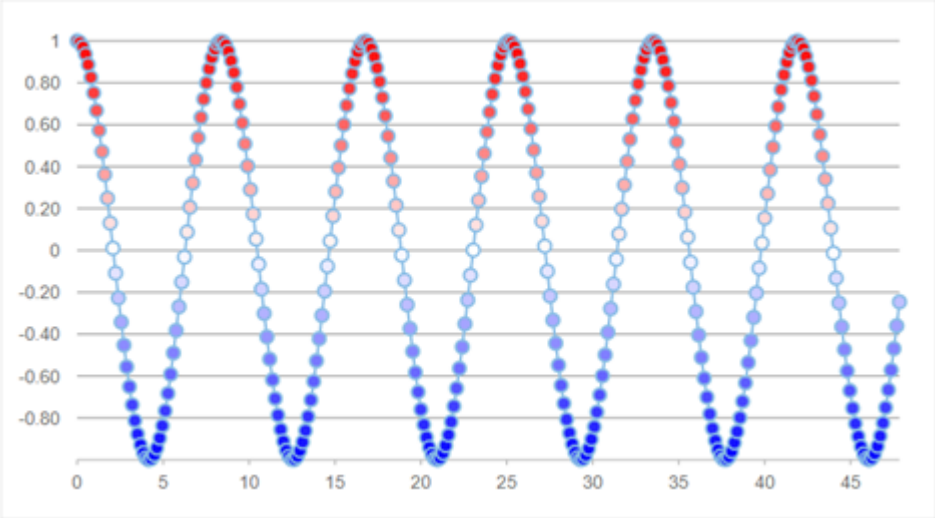
@(Html.C1().FlexChart().Id("chartGradients")
    .ChartType(ChartType.Column)
    .Bind(Model)
    .BindingX("Name").Legend(Position.None)
    .DataLabel(label => { label.Content("{y}"); })
    .Series(ser =>
    {
        ser.Add().Name("March").Binding("MarPrice").Style(s => s.Stroke("darkred")
        .StrokeWidth(1).Fill("l(0, 0, 1, 0)#30cfd0-#330867"));
    }))
```


Copyright © GrapeCity, Inc. All rights reserved.

外観のカスタマイズ化

FlexChart allows you to customize the appearance of the data points based on their values using **ItemFormatter** property of the FlexChart class. The **ItemFormatter** property lets you customize the UI of the control using a JavaScript function.

The following code examples demonstrate how to customize the appearance of the data points in FlexChart.



Create an MVC application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Create a data source for FlexChart

1. Add a new class to the folder **Models** (for example: `Dot.cs`). See Adding controls to know how to add a new model.
2. Add the following code to the new model to define the class that serve as a datasource for the FlexChart control.
    - **C#**

```
public class Dot
{
    public double X { get; set; }
    public double Y { get; set; }
    public double Size { get; set; }
}
```

Visual Basic

```
Public Class Dot
    Public Property X As Double
    Public Property Y As Double
    Public Property Size As Double
End Class
```

Back to Top

Copyright © GrapeCity, Inc. All rights reserved.

Add FlexChart and customize its appearance

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `FlexChartController`).
    3. Click **Add**.

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the controller `FlexChartController` to open it.
2. Place the cursor inside the `Index()` method.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the view name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add**. A view is added for the controller.
6. Replace the code in the Index.cshtml file with the following.

cshtml

```
@using ItemFormatter.Models;

@{
    List<Dot> cos = new List<Dot>();
    for (int i = 0; i < 300; i++)
    {
        cos.Add(new Dot { X = 0.16 * i, Y = Math.Cos(0.12 * i) });
    }
}

<script type="text/javascript">
    var itemFormatter = function (engine, hitTestInfo, defaultFormat) {
        if (hitTestInfo.chartElement === wijmo.chart.ChartElement.SeriesSymbol)
{
            var y = hitTestInfo.y;
            var r = y >= 0 ? 255 : (255 * (1 + y)).toFixed();
            var b = y < 0 ? 255 : (255 * (1 - y)).toFixed();
            var g = ((1 - Math.abs(y)) * 255).toFixed();
            engine.fill = 'rgb(' + r + ',' + g + ',' + b + ')';
            defaultFormat();
        }
    };
</script>


@(Html.C1().FlexChart()
        .ChartType(C1.Web.Mvc.Chart.ChartType.LineSymbols)
        .Series(sers =>
        {
        sers.Add().Bind(cos)
                .BindingX("X")
                .Binding("Y")
                .Name("cos(x)")
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                .TooltipContent("<b>{name}</b> : {value}");})
    .Legend(C1.Web.Mvc.Chart.Position.None).ItemFormatter("itemFormatter"))
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

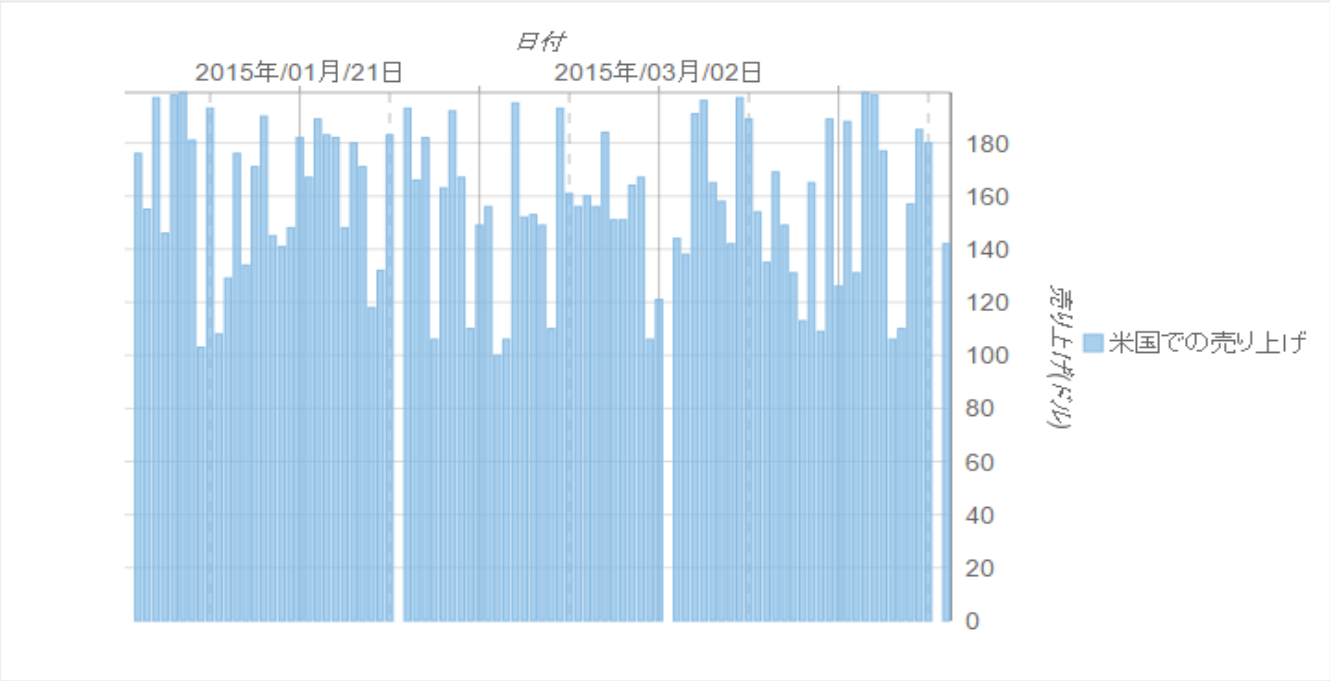
> Append the folder name and view name to the generated URL (for example: http://localhost:1234/**FlexChart/Index**) in the address bar of the browser to see the view.

Back to Top

軸のカスタマイズ

軸は、ラベル、線、目盛りマーク、タイトルなどのいくつかの要素で構成されます。FlexChart では、X 軸と Y 軸の両方に、これらの要素をカスタマイズするためのプロパティがいくつか用意されています。Position プロパティを使用して、軸の位置を変更することもできます。

次の図は、カスタマイズされた軸を持つ FlexChart を示しています。



次のコード例は、このプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor copyCode

```
.AxisY(axis => axis
      .Position(C1.Web.Mvc.Chart.Position.Right)
      .MajorGrid(true).MajorUnit(20)
      .AxisLine(true)
```


Copyright © GrapeCity, Inc. All rights reserved.

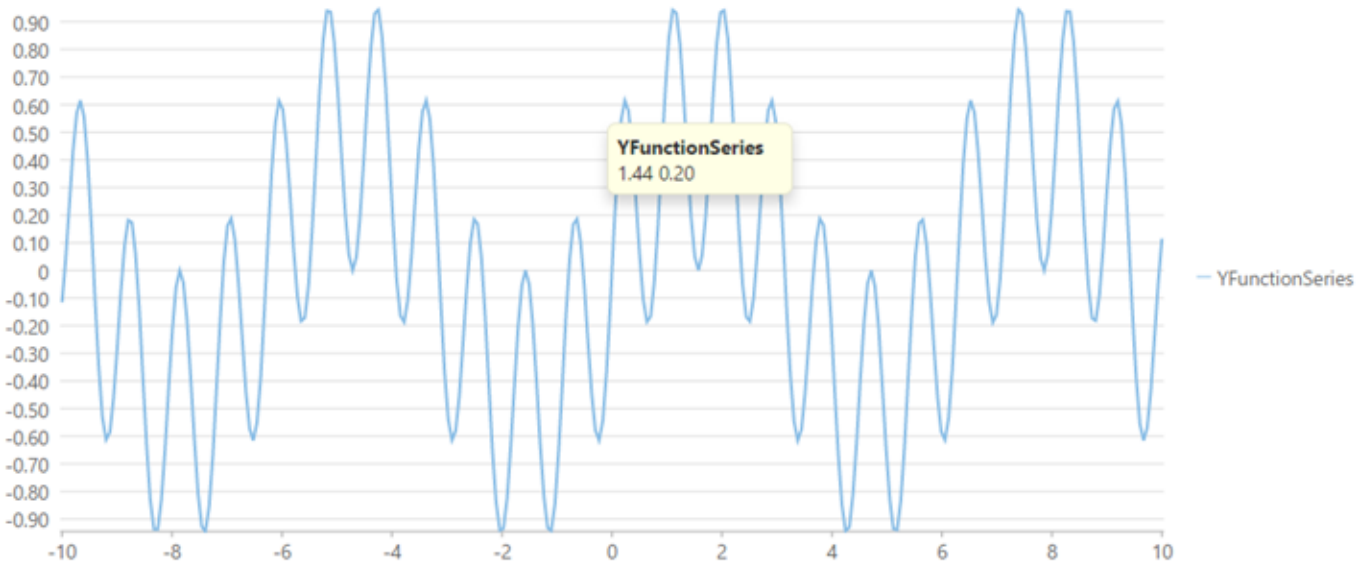
```
        .Labels(true)
        .Title("売り上げ(ドル)")
    )
.AxisX(axis => axis
        .Position(C1.Web.Mvc.Chart.Position.Top)
        .AxisLine(true)
        .MajorGrid(true)
        .MajorUnit(20)
        .MinorGrid(true)
        .Labels(true)
        .Title("日付")
        .Format("2015年/MM月/dd日")
    )
```

関数系列

FlexChartとFinancialChartを使用すると、ユーザーはさまざまな式を使用した関数系列を備えた分析チャートを作成できます。**AddYFunctionSeries**および**AddParametricFunctionSeries**を使用して、Y関数系列またはパラメータ関数系列を追加できます。YFunctionSeriesは与えられた関数によってyを計算し、PfunctionSeriesは与えられたx関数とy関数によってXとYを計算します。ユーザーは値と式に基づいて任意のタイプの曲線をプロットできます。これらの例では、さまざまな式を使用して正弦値と余弦値を計算し、チャートに表示する値を返します。

次の図は、Y関数系列を使用したときのFlexChartを示しています。

Y関数系列



次のコード例は、Y関数系列を使用してFinancialChartのデータを分析する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

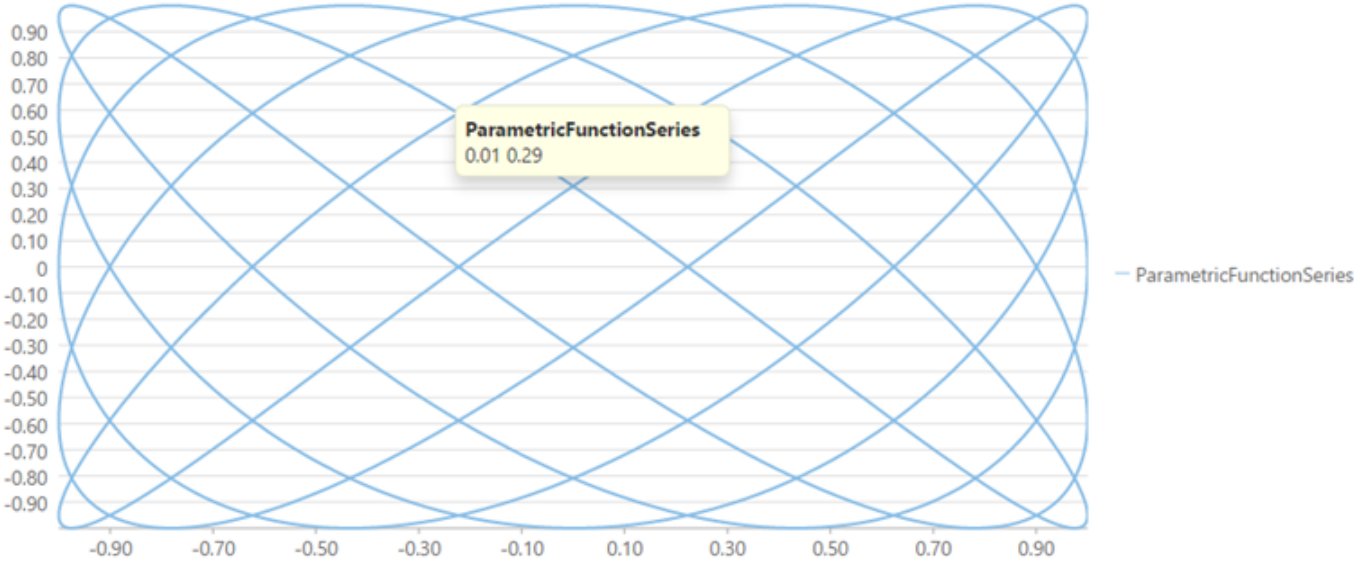
```
@using C1.Web.Mvc.Chart

<div>
    <div>
        <script type="text/javascript">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            function yFunc(value) {
                return Math.sin(4 * value) * Math.cos(3 * value);
            }
        </script>
        @(Html.C1().FlexChart().Legend(Position.Right)
            .Series(ses =>
            {
                ses.AddYFunctionSeries("YFunctionSeries").Min(-
10).Max(10).SampleCount(300).Func("yFunc");
            }))
    </div>
</div>
```

パラメータ関数系列



次のコード例は、パラメータ関数系列を使用してFinancialChartのデータを分析する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

```
@using C1.Web.Mvc.Chart

<div>
    <div>
        <script type="text/javascript">
            var xParam = 5, yParam = 7;
            function xFunc(value) {
                return Math.cos(value * xParam);
            }
            function yFunc(value) {
                return Math.sin(value * yParam);
            }
        </script>
        @(Html.C1().FlexChart().Legend(Position.Right)
            .Series(ses =>
            {
```

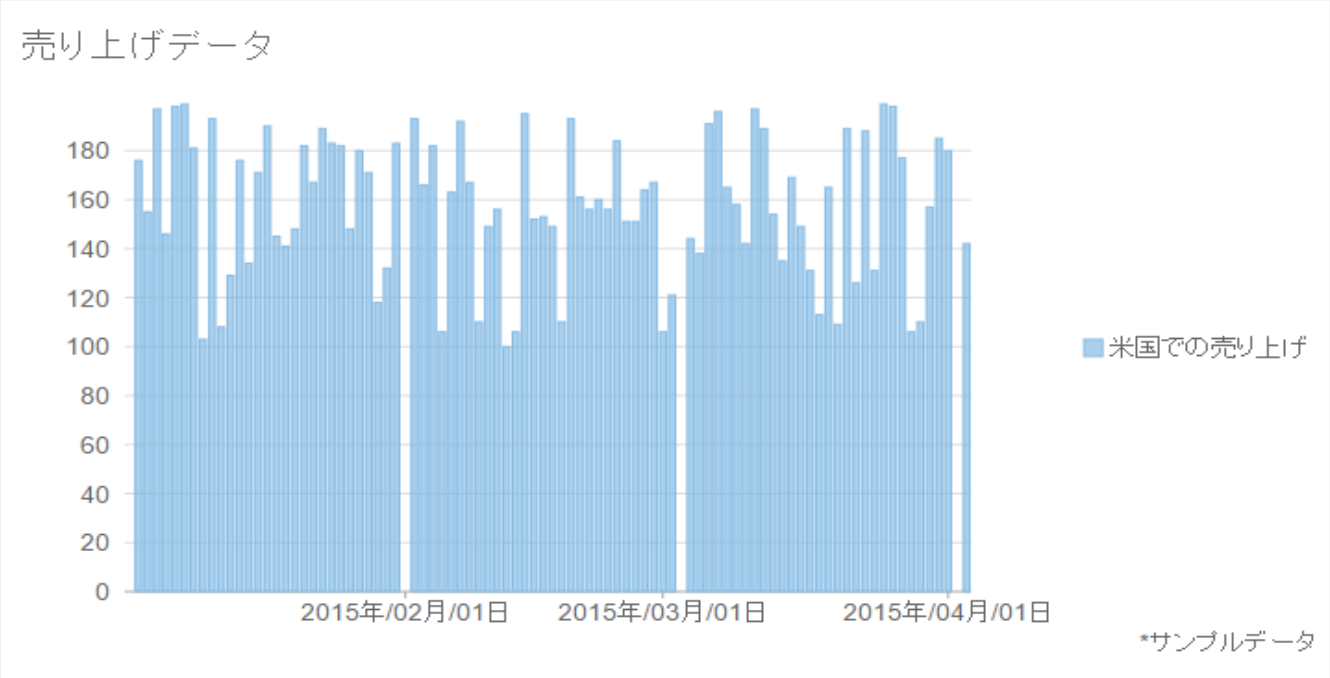

Copyright © GrapeCity, Inc. All rights reserved.

```
                ses.AddParametricFunctionSeries("ParametricFunctionSeries")
                    .Max(2 * Math.PI)
                    .SampleCount(1000)
                    .XFunc("xFunc")
                    .YFunc("yFunc");
            }))
    </div>
</div>
```

ヘッダーとフッター

FlexChart コントロールにタイトルを追加できます。それには、**Header** プロパティを設定します。タイトル以外に、チャートのフッターを設定することもできます。それには、**Footer** プロパティを設定します。ヘッダーおよびフッターテキストのスタイルを設定することもできます。それには、**HeaderStyle** および **FooterStyle** プロパティを使用します。

次の図は、これらのプロパティを設定した後の FlexChart を示しています。



次のコード例は、これらを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor	copyCode

```
.Header("売り上げデータ")
.HeaderStyle(hs => hs.Halign("left"))
.FooterStyle(fs => fs.Halign("right"))
.Footer("*サンプルデータ")
```

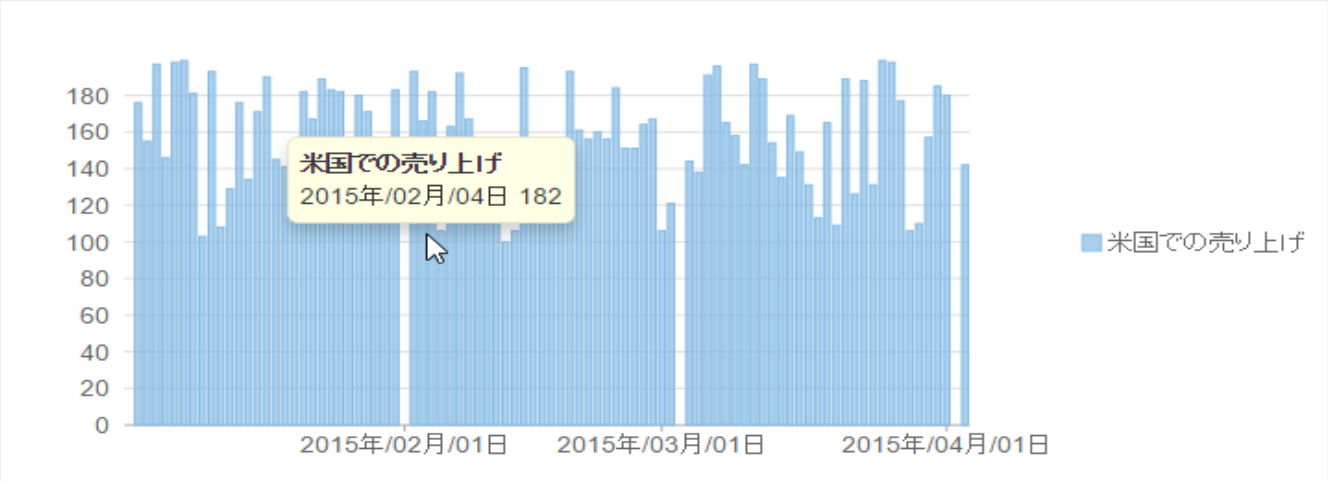
ヒットテスト

`HitTest()` メソッドを使用して、マウスが置かれたときに、X および Y 座標や FlexChart 内のポイントのインデックスを判定し

Copyright © GrapeCity, Inc. All rights reserved.

ます。

次の図は、X および Y 座標値と FlexChart 内のヒットポイントのインデックスがチャートの下のラベルに表示されているところです。



グラフ要素: PlotArea
系列名: 米国での売り上げ
ポイントインデックス: 33
x coordinate: -2143042046452.49
y coordinate: 103.90

次のコード例は、単純なラベルを使用して X および Y 座標値とヒットポイントのインデックスを表示する方法を示します。

コードの場合

1. 次のように FlexChart をインスタンス化します。

Razor copyCode

```
<div id="flexChart1" style="height: 300px"></div>
@(Html.C1().FlexChart("#flexChart1")
.Bind("Date", Model)
.ChartType(C1.Web.Mvc.Chart.ChartType.Column)
.Series(sers => sers
      .Add()
      .Binding("SalesInUSA")
      .Name("米国での売り上げ")
  )
       .AxisX(x => x.Format("2015年/MM月/dd日"))
)
```

2. 次のように、ヒットポイントの情報を表示するための タグを FlexChart の直後に追加します。

Razor copyCode

```
<div id="info"></div>
```

3. 次のように、X/Y 座標値とヒットポイントのインデックスを取得し、それらをグラフのすぐ下に追加した部分に表示するためのスクリプトを追加します。

JavaScript copyCode

```
<script type="text/javascript">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    c1.mvc.Utils.documentReady(function ()
    {
        var chart = wijmo.Control.getControl("#flexChart1"),
            formatHitInfo = function (hitInfo, pt)
            {
                var s = '<div><b>グラフ要素</b>: ' +
wijmo.chart.ChartElement[hitInfo.chartElement] + '</div>';
         if (hitInfo.series)
           {
              s += '<div><b>系列名</b>: ' + hitInfo.series.name;
              if (hitInfo.pointIndex !== null) {
              s += '<div><b>ポイントインデックス</b>: ' + hitInfo.pointIndex +
'</div>';
              if (hitInfo.chartElement == wijmo.chart.ChartElement.PlotArea)
                 {
              s += '<div><b>x coordinate</b>: ' + pt.x.toFixed(2) + '</div>';
              s += '<div><b>y coordinate</b>: ' + pt.y.toFixed(2) + '</div>';
                        }
                    }
                }
                return s;
            };
        chart.hostElement.onmousemove = function (e)


        {
            var hit = chart.hitTest(e);
            var info = document.getElementById("info");
            info.innerHTML = formatHitInfo(hit, chart.pointToData(e));

        };

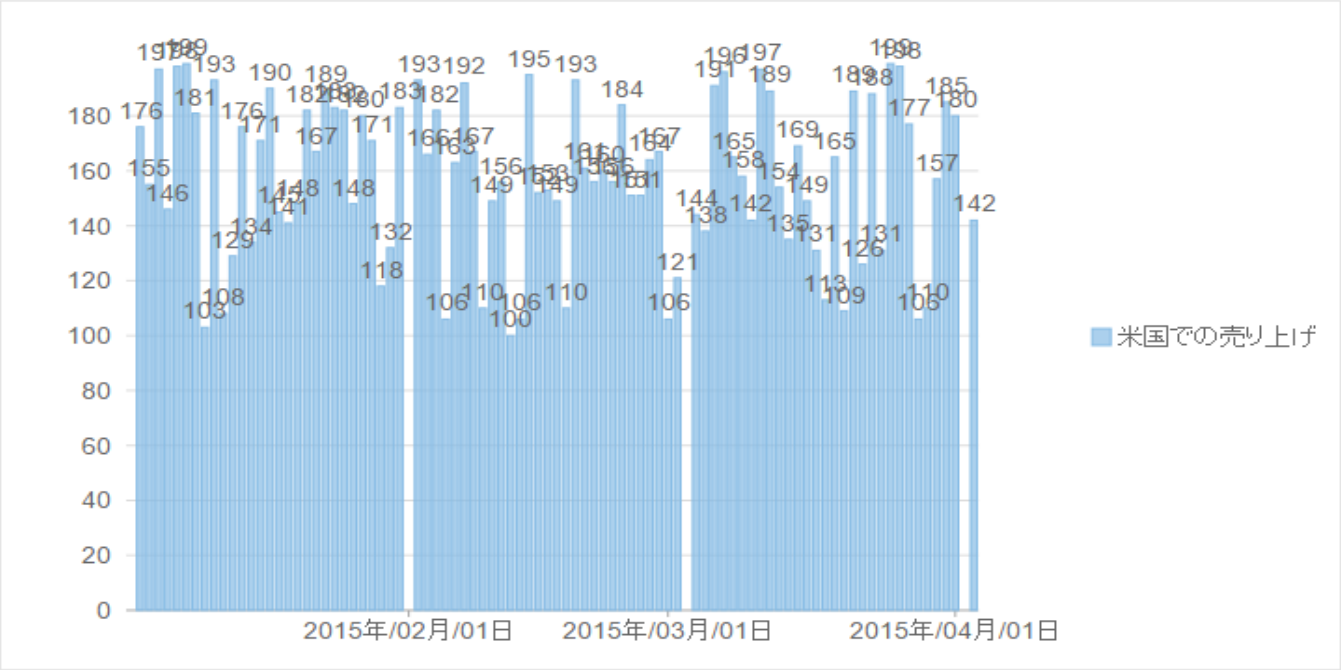
    });
</script>
```

ラベル

各データポイントの値を表示するためのラベルを追加できます。それには、**DataLabel** プロパティを設定します。ラベルを表示する位置を指定することもできます。それには、**LabelPosition** プロパティを設定します。

次の図は、FlexChart でデータポイントごとにラベルを表示したところです。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexChart にラベルを追加する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor	copyCode

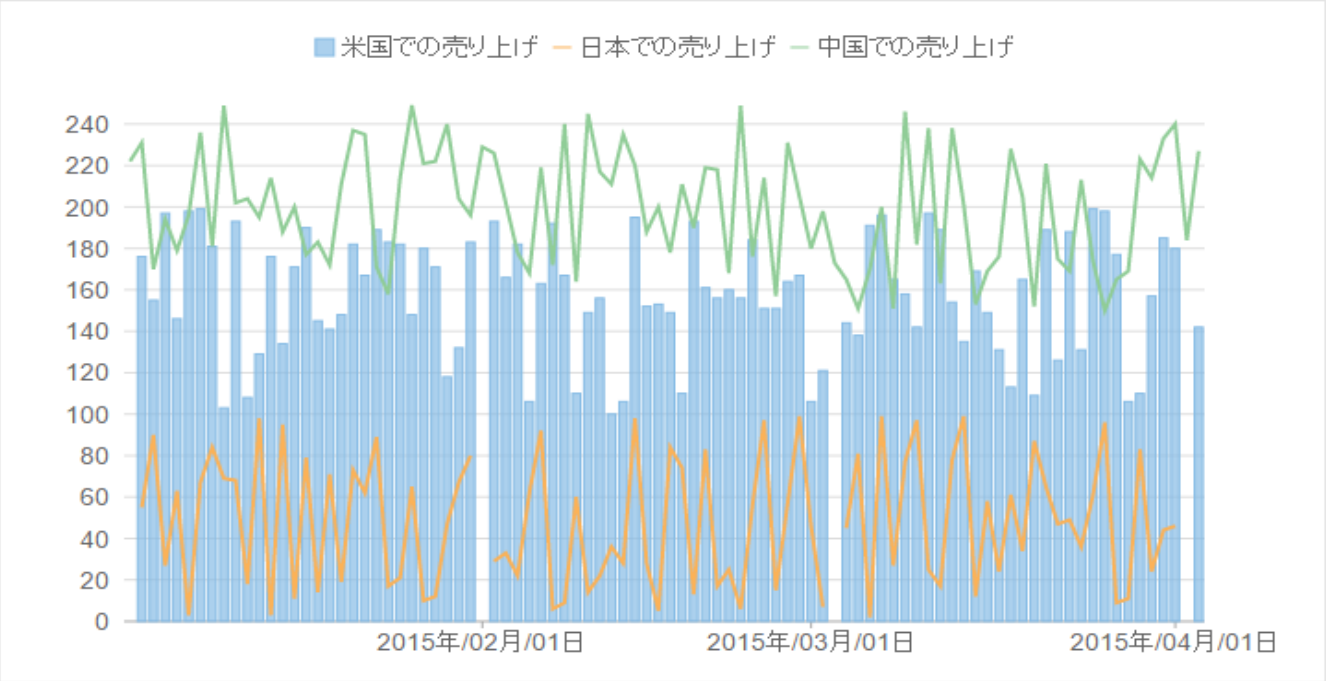
```
.DataLabel(dl => dl.Position(C1.Web.Mvc.Chart.LabelPosition.Top).Content("{y}"))
```

凡例

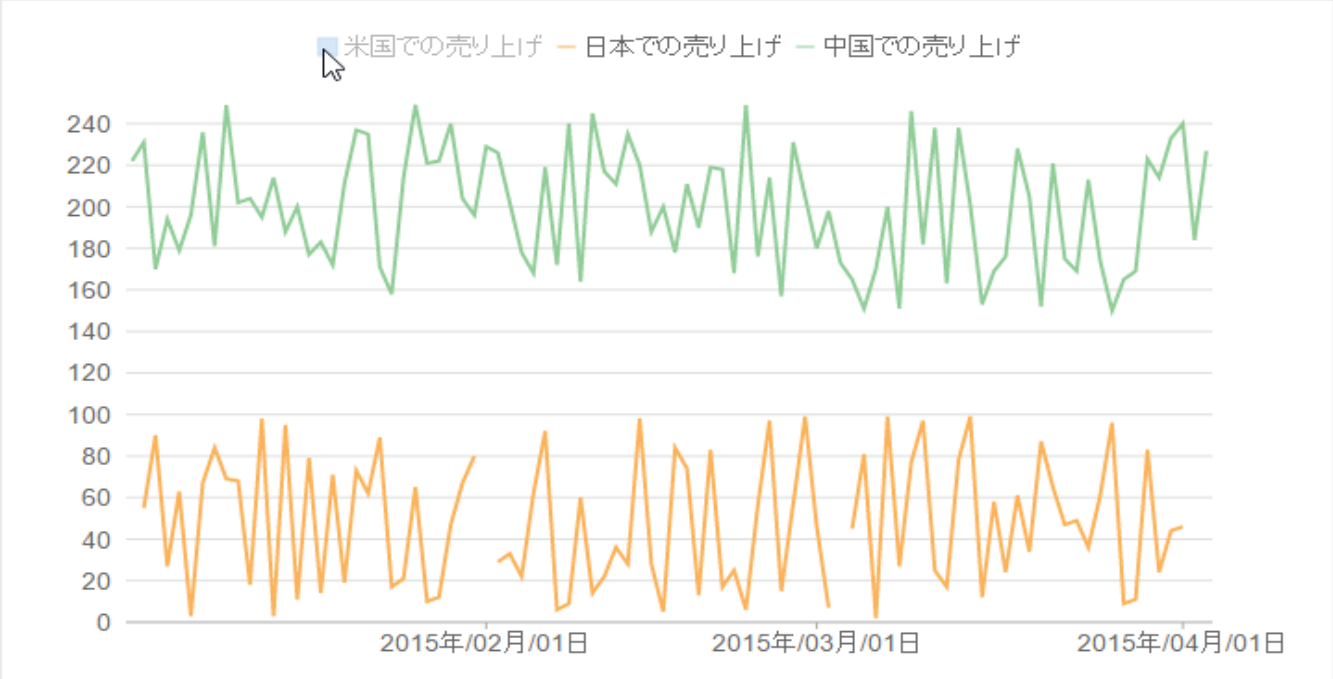
凡例を表示する位置を指定できます。それには、FlexChart の **Legend** プロパティを設定します。**LegendToggle** プロパティを true に設定すると、対応する凡例項目をクリックすることで、任意の系列の表示/非表示を切り替えることができます。

次の図は、これらのプロパティを設定した後の FlexChart を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



凡例項目をクリックすると、対応する系列が次のように非表示になります。



次のコード例は、これらのプロパティを設定する方法を示します。この例では、「複合チャート」セクションで作成したサンプルを使用します。

コードの場合

Razor	copyCode

```
.Legend(C1.Web.Mvc.Chart.Position.Top)
.LegendToggle(true)
```

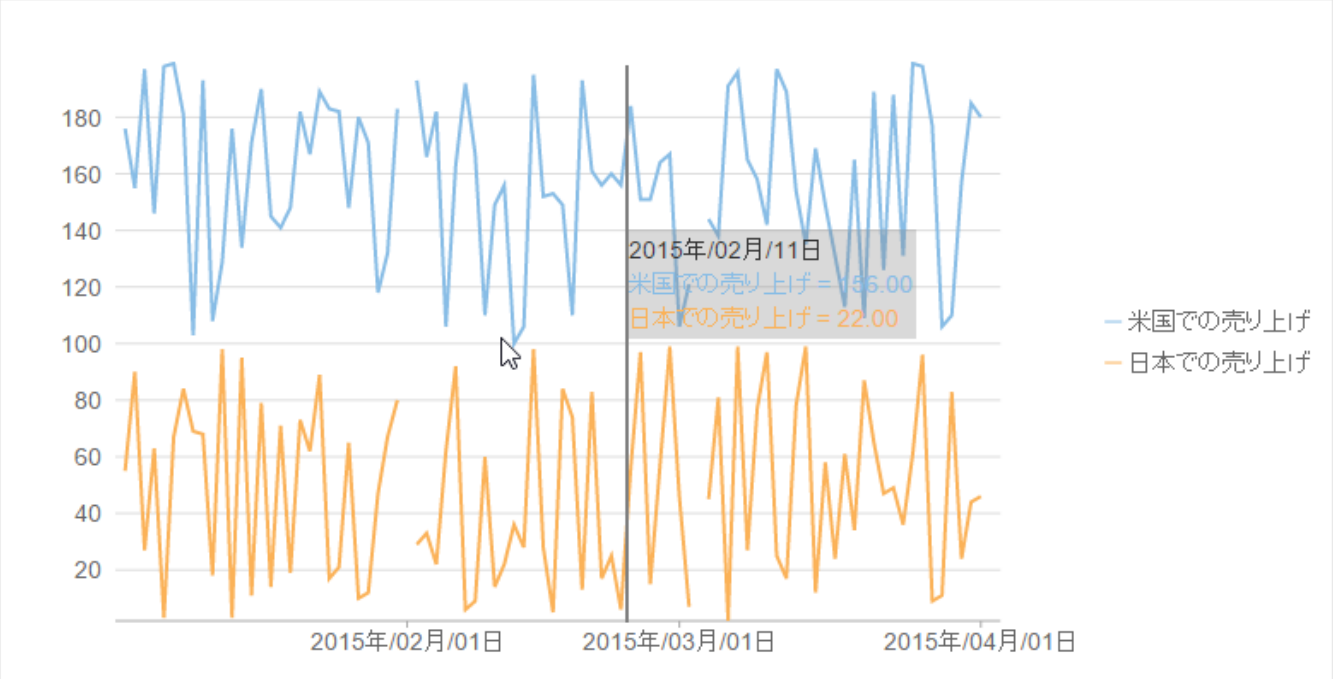
マーカー

Copyright © GrapeCity, Inc. All rights reserved.

マーカーは、データ系列上にマウスホバーした際、データポイントの値を示す記号です。FlexChart にて、**AddLineMarker** メソッドを使用してラインマーカーを追加することができます。また、**LineMarkersLines** プロパティを以下の値のいずれかに設定できます:

- **Vertical**:縦ラインマーカーが表示されます。
- **Horizontal**:横ラインマーカが表示されます。
- **Both**:縦および横ラインマーカーの両方が表示されます。
- **None**: ラインマーカが表示されません。

次の図は、グラフ上にデータポイントを表示するため、内容付き縦ラインマーカーを示しています。



次のコード例では、FlexChart に内容付き縦ラインマーカを追加する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

```
@using MVCFlexChart_JP.Models;
@using C1.Web.Mvc.Chart;
@model IEnumerable<FruitSale>

<br/>
<br/>

@*マーカーの内容を追加します*@
<script type="text/javascript">
    function lineMarkerContent(hitInfo, pt) {
        var html = '', chart = hitInfo.series ? hitInfo.series.chart : undefined;
        if (!chart || !chart.series) {
            return html;
        }
        chart.series.forEach(function (s, i) {
            var ht = s.hitTest(new wijmo.Point(pt.x, NaN)),
        hostEle = s.hostElement, polyline;

            polyline = s.hostElement ? s.hostElement
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                .getElementsByTagName("polyline")[0] : undefined;

            if (polyline && ht.x && ht.y !== null) {
                if (i == 0) {
                    html += wijmo.Globalize.formatDate(ht.x, '2015年/MM月/dd日');
                }
                html += '<div style="color:' + polyline.getAttribute('stroke')
 + '">' + ht.name + ' = ' + ht.y.toFixed(2) + '</div>';
            }
        });
        return html;
    }
</script>

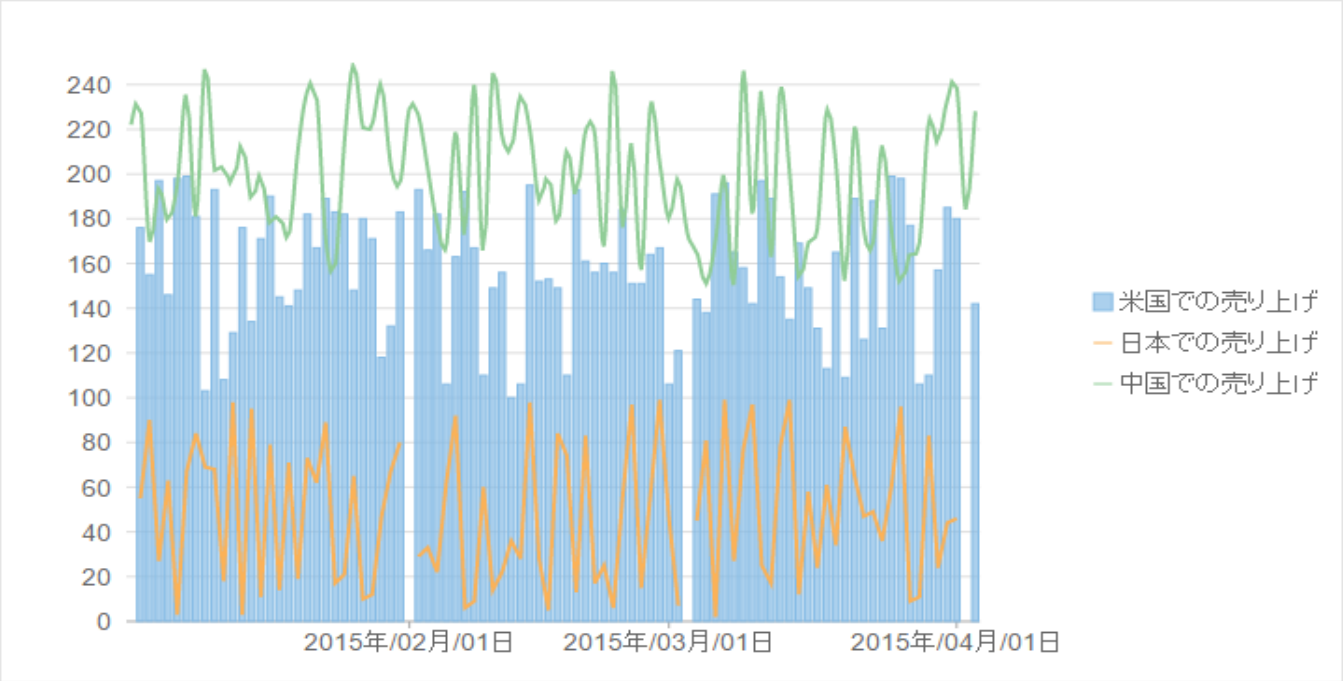
@*マーカーを追加します*@
<div style="width: 780px">
    @(Html.C1().FlexChart().Bind("Date", Model).ChartType(ChartType.Line).Series(sers
=>
{
    sers.Add().Binding("SalesInUSA").Name("米国での売り上げ");
    sers.Add().Binding("SalesInJapan").Name("日本での売り上げ");
})
            .AxisX(x => x.Format("2015年/MM月/dd日")).Tooltip(tp => tp.Content(""))
    .AddLineMarker(lm => lm
    .Alignment(LineMarkerAlignment.Auto)
    .Lines(LineMarkerLines.Vertical)
    .Interaction(LineMarkerInteraction.Move).Content("lineMarkerContent")))
</div>
```

複合チャート

チャートに複数の系列を追加し、すべての系列に異なる **ChartType** を設定できます。このようなチャートは、1つのキャンバスで複雑なグラフデータを分析する場合に役立ちます。同じデータを異なる方法で視覚化して使用したり、関連するデータを一緒に表示して傾向を示すことができます。

次の図は、複数の系列を持つ FlexChart を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、異なる **ChartType**を持つ複数のチャート系列を FlexChart に追加する方法を示します。

コードの場合

Razor copyCode

```
// FlexChartを初期化します
@(Html.C1().FlexChart()
.Bind("Date", Model)
    .Series(sers =>
      {
          // 最初の系列を追加します
          sers.Add()
              .Binding("SalesInUSA")
              .Name("米国での売り上げ")

.ChartType(C1.Web.Mvc.Chart.ChartType.Column);

          // 2番目の系列を追加します
          sers.Add()
          .Binding("SalesInJapan")
          .Name("日本での売り上げ")
.ChartType(C1.Web.Mvc.Chart.ChartType.Line);

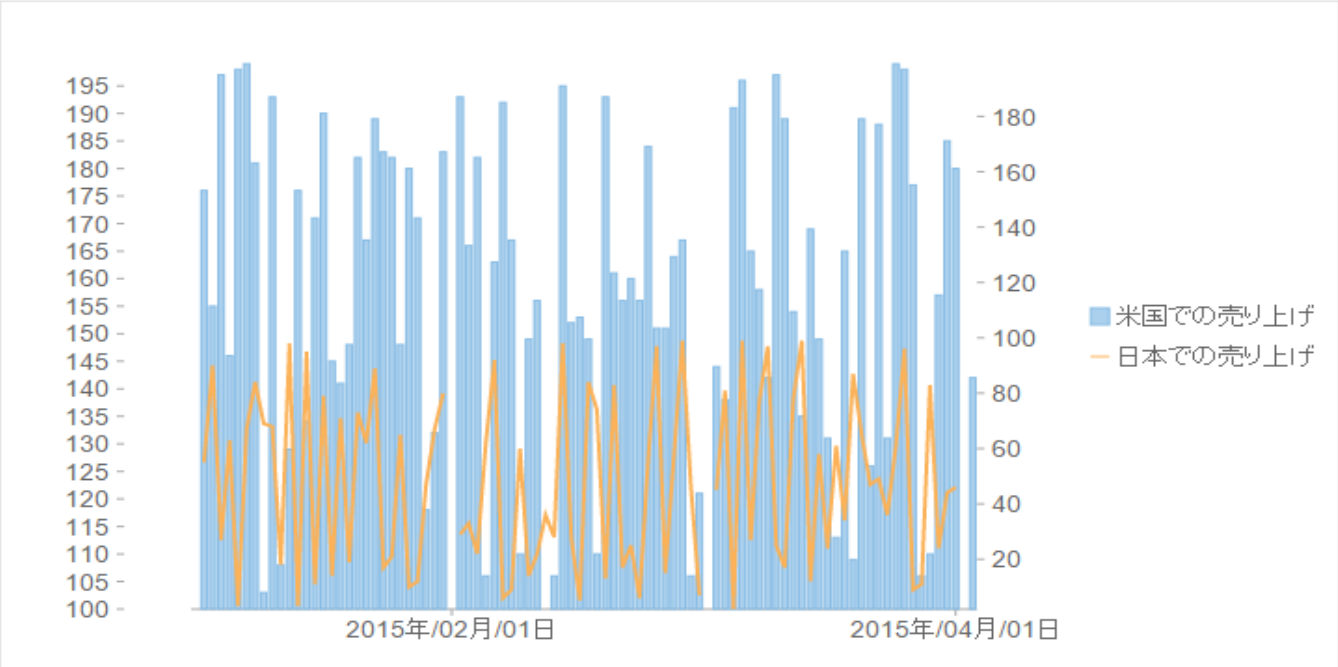
          // 3番目の系列を追加します
          sers.Add()
          .Binding("SalesInChina")
          .Name("中国での売り上げ")
          .ChartType(C1.Web.Mvc.Chart.ChartType.Spline);
      })
     .AxisX(x => x.Format("2015年/MM月/dd日"))
)
```


Copyright © GrapeCity, Inc. All rights reserved.

複数の軸

FlexChart に複数の系列を追加する場合には、副軸を追加できます。副軸の目盛りは、その軸に関連付けられている系列の値を表します。副軸にプロットされている値を簡単に識別できるように、関連付けられている系列の **ChartType** を変更することもできます。

次の図は、FlexChart に複数の Y 軸を表示したところです。



次のコード例は、FlexChart に複数の Y 軸を追加する方法を示します。この例では、「複合チャート」セクションで作成したサンプルを使用します。

コードの場合

Razor copyCode

```
//複数の軸を初期化します
@(Html.C1().FlexChart()
.Bind("Date", Model)

//チャートに系列を追加します
.ChartType(C1.Web.Mvc.Chart.ChartType.Column)
    .Series(sers =>
      {
          //一番目の系列を追加します
          sers.Add()
              .Binding("SalesInUSA")
              .Name("米国での売り上げ")

            //一番目の系列に対して Y軸の位置を設定します
              .AxisY(axis => axis.Position(C1.Web.Mvc.Chart.Position.Left));

        //二番目の系列を追加します
          sers.Add()
          .Binding("SalesInJapan")
          .Name("日本での売り上げ")
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            .ChartType(C1.Web.Mvc.Chart.ChartType.Line)

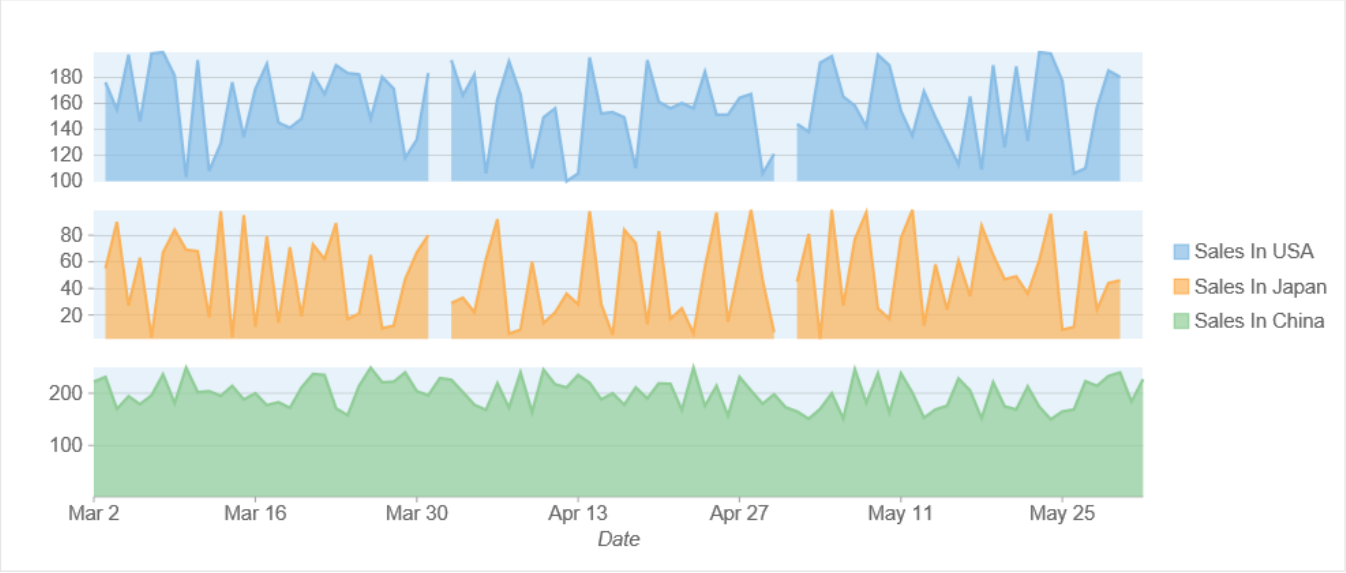
            //二番目の系列に対して Y軸の位置を設定します
            .AxisY(axis => axis.Position(C1.Web.Mvc.Chart.Position.Right));
        })

        .AxisX(x => x.Format("2015年/MM月/dd日"))
)
```

プロット領域

FlexChartでは、1つのチャート領域内に系列ごとに異なるプロット領域を作成できます。これにより、一方の軸を固定したまま、他方の軸で各系列を個別のプロット領域に表示することで、データの視認性を向上させることができます。

次の図は、Y軸の3つのプロット領域で構成されるチャートを示します。



次のコード例は、FlexChartでプロット領域を作成する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

```
@using C1MvcWebAppPlotAreas.Models
@model IEnumerable<FruitSale>

<script>
    var flexChart;
    c1.documentReady(function () {
        flexChart = wijmo.Control.getControl('#flexChart');

        addPlotArea(0, "plot1", { fill: 'rgba(136,189,230,0.2)' }, null);
        addPlotArea(1, "div1",null, 20);
        addPlotArea(2, "plot2", { fill: 'rgba(136,189,230,0.2)' }, null);
        addPlotArea(3, "div2", null, 20);
        addPlotArea(4, "plot3", { fill: 'rgba(136,189,230,0.2)' }, null);
        flexChart.series[1].axisY.plotArea =
flexChart.plotAreas.getPlotArea('plot2');
        flexChart.series[2].axisY.plotArea =
```


Copyright © GrapeCity, Inc. All rights reserved.

```
flexChart.plotAreas.getPlotArea('plot3');
    });

    function addPlotArea(row, name, style, height) {
        var plotArea = new wijmo.chart.PlotArea();
        plotArea.row = row;
        plotArea.name = name;
        if (style) {
            plotArea.style = style;
        }

        if (height) {
            plotArea.height = height;
        }

        flexChart.plotAreas.push(plotArea);
    }
</script>

@(Html.C1().FlexChart().Id("flexChart").ChartType(C1.Web.Mvc.Chart.ChartType.Area)
    .CssClass("chart").Bind(Model).BindingX("Date")
    .AxisX(axis => axis.Title("Date"))
    .Series(sers =>
    {
        sers.Add().Binding("SalesInUSA").Name("Sales In USA");
        sers.Add().Binding("SalesInJapan").Name("Sales In Japan").AxisY(axis =>
            axis.Position(C1.Web.Mvc.Chart.Position.Left).MajorGrid(true));
        sers.Add().Binding("SalesInChina").Name("Sales In China").AxisY(axis =>
            axis.Position(C1.Web.Mvc.Chart.Position.Left).MajorGrid(true));
    })
)
```

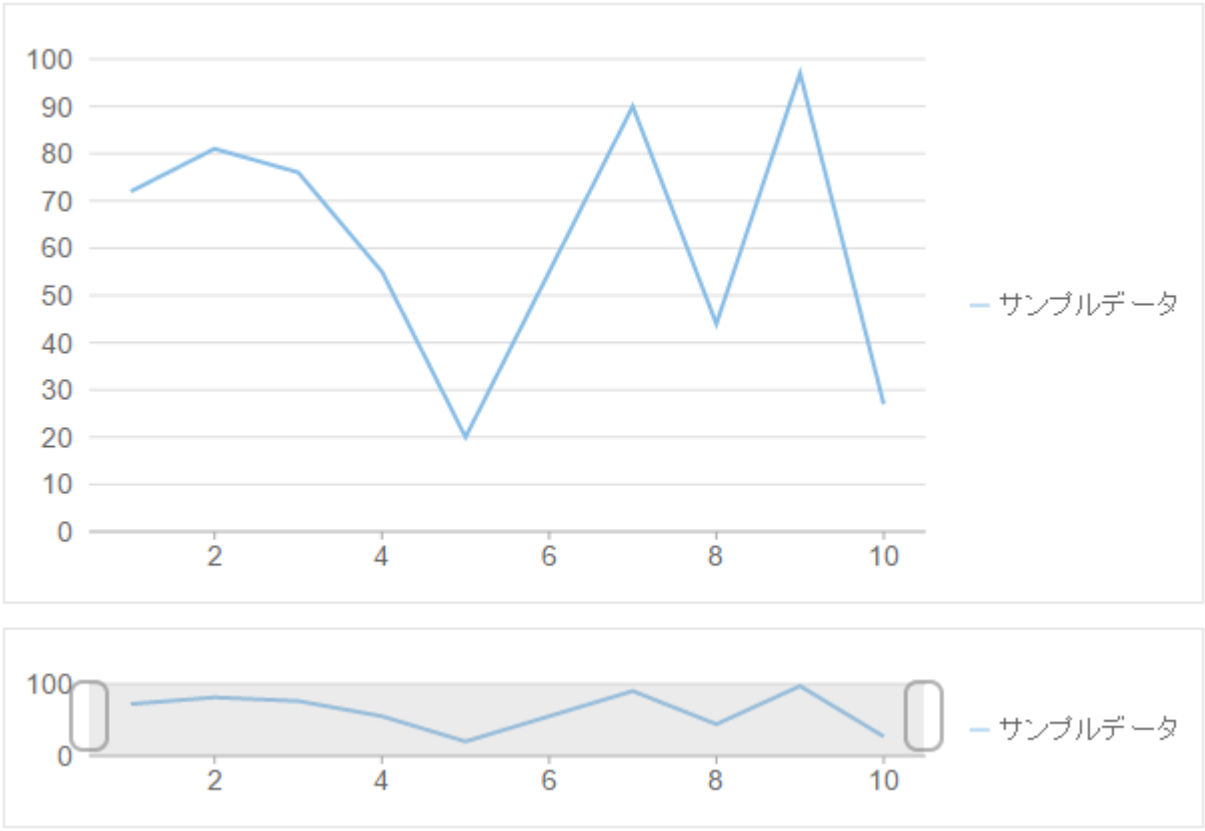
範囲セレクタ

FlexChart では、チャートに表示する数値データの範囲を指定するために**RangeSelector** が用意されています。RangeSelectorには、2つのつまみボタンが含まれています。左側の主つまみボタンと右側のつまみボタンはそれぞれ値範囲の開始値と終了値を定義します。レンジバー上でつまみボタンを左方向に移動すると、数値は減少され、右方向に移動すると数値は増加されます。RangeSelector の **Min** および **Max** プロパティを使用してつまみボタンの最小値と最大値を設定できます。

つまみボタンの値は、RangeSelector に設定した最小値より大きいと最大値より低い値に設定する必要があります。また、**Orientation** プロパティを使用して、RangeSelector の向きの変更を設定できます。

次の図はRangeSelectorのプロパティを設定した後の FlexChart を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexChart 上で RangeSelector を使用してつまみボタンの値を表示する方法を示します。

コードの場合

サンプルコード

```
@model List<MathPoint>
@(Html.C1().FlexChart().Id("trendChart")
    .Bind("X", Model)
    .AxisY(ay => ay.Min(0).Max(100))
    .Legend(Position.Right)
    .Width(600)
    .Height(300)
    .LegendToggle(true)
    .Series(ss =>
    {
        ss.Add(ChartType.Line, "サンプルデータ").Binding("Y");

    })
)
<script type="text/javascript">
    function rangeChanged(rs, ui) {
        updateStChartRange(rs.min, rs.max);
    }
</script>

@(Html.C1().FlexChart()

    .Bind("X", Model)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .AxisY(ay => ay.Min(0).Max(100))
    .Legend(Position.Right)
    .Width(600)
    .Height(100)
    .LegendToggle(true)
    .Series(ss =>
    {
        ss.Add(ChartType.Line, "Sample Data").Binding("Y");

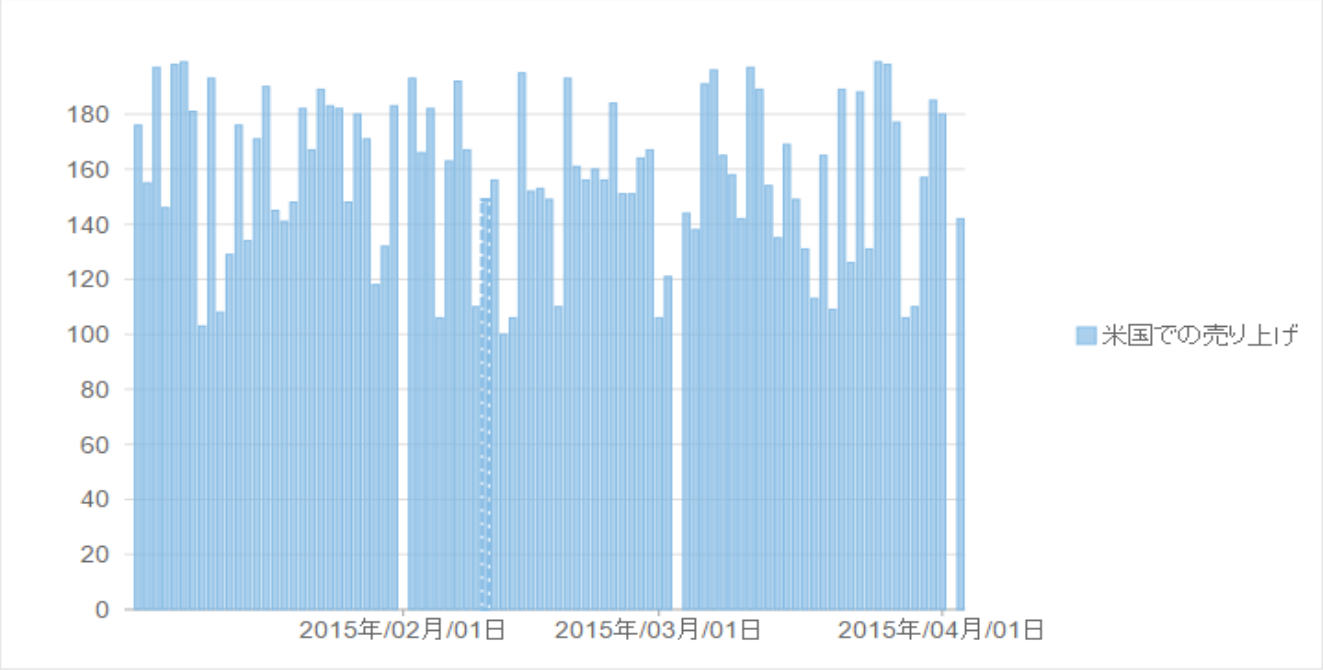
    })
     .AddRangeSelector(rs => rs.Min(0).Max(100).OnClientRangeChanged("rangeChanged"))
     )
```

選択

ユーザーが FlexChart 内の任意の領域をクリックしたときに FlexChart のどの要素を選択するかを選択できます。それには、**SelectionMode** プロパティを設定します。このプロパティは3つのオプションを提供します。

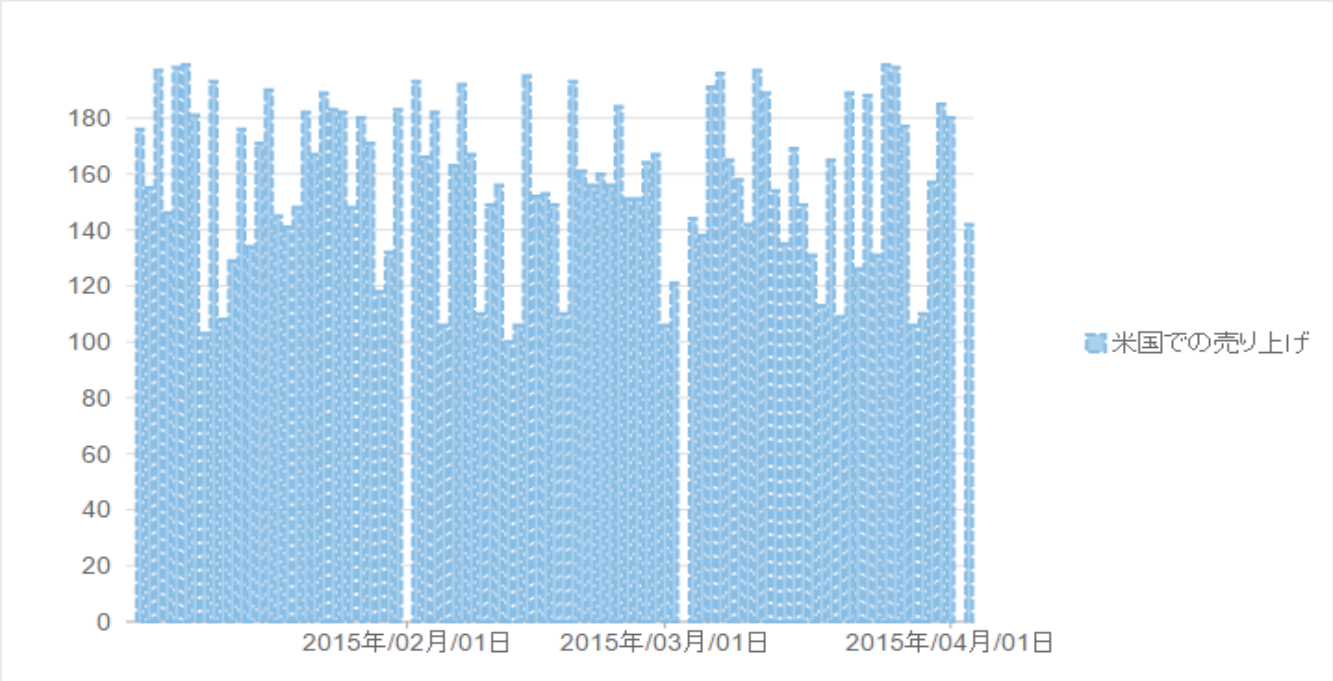
- **None**:どの要素も選択されません。
- **Point**:ユーザーがクリックした点が強調表示されます。
- **Series**:ユーザーがクリックした系列が強調表示されます。

次の図は、これらのプロパティを設定した後の FlexChart を示しています。



SelectionMode を **Point** に設定した場合

Copyright © GrapeCity, Inc. All rights reserved.



SelectionMode を **Series** に設定した場合

次のコード例は、このプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor	copyCode

```
.SelectionMode(C1.Web.Mvc.Chart.SelectionMode.Series)
```

系列

このセクションには、次の情報が含まれます。

ウォータフォール
　FlexChartにウォータフォール系列を追加する方法を説明します。
誤差範囲チャート
　FlexChartに誤差範囲系列を追加する方法を説明します。
箱ひげ図チャート
　FlexChartに箱ひげ図系列を追加する方法を説明します。

箱ひげ図チャート

箱ひげ図チャート系列は、データの分布を四分位数で示して、平均値や異常値を強調表示します。箱には、「ひげ」と呼ばれる垂直方向に伸びた線が付くことがあります。これらの線は、上位四分位数と下位四分位数の外側のばらつきを示し、それらの線、つまり「ひげ」の外側にあるポイントはすべて異常値と見なされます。

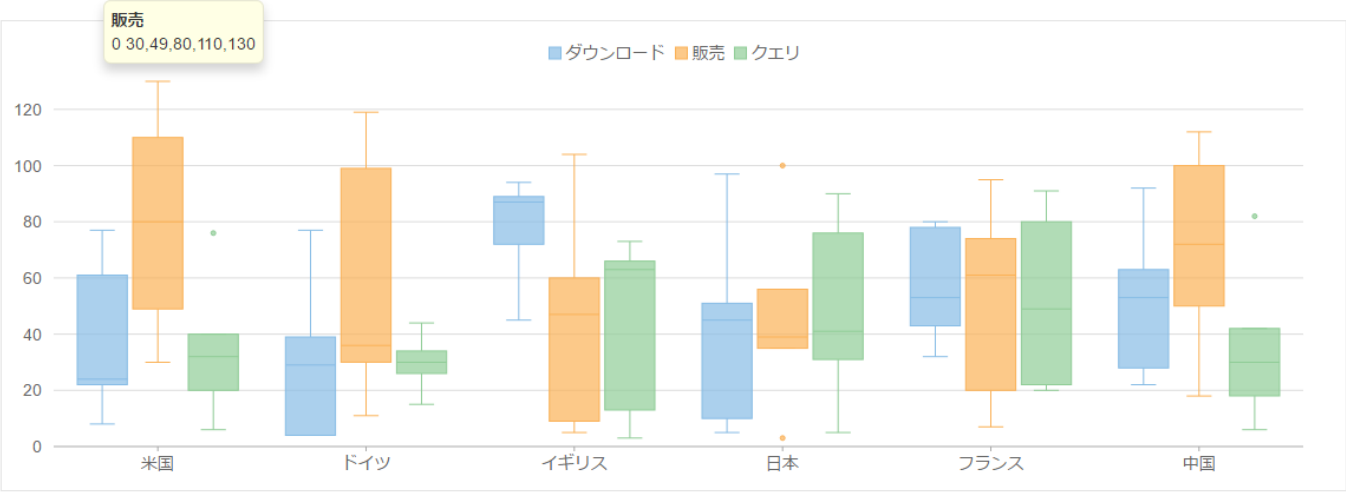
箱ひげ図は、統計分析で最もよく使用されます。たとえば、箱ひげ図を使用して、臨床試験の結果や学校のテストの点数を比較できます。

このトピックでは、FlexChartで箱ひげ図系列を使用して、データの分布を四分位数で示し、平均値や異常値を強調表示する方法を説明します。箱ひげ図系列を使用するには、**C1.Web.Mvc**名前空間に含まれる**FlexChart**クラスのインスタンスを作成する必要があります。

Copyright © GrapeCity, Inc. All rights reserved.

- **手順1：FlexChartのデータソースの作成**
- **手順2：FlexChartへの箱ひげ図系列の追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexChartを示しています。



手順1：FlexChartのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：ProductSales.cs）。
2. 次のコードを新しいモデルに追加して、FlexChartのデータソースになるクラスを定義します。

ProductSale.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BoxWhiskerSeries.Models
{
    public class ProductSales
    {
        public int Id { get; set; }
        public string Country { get; set; }
        public List<int> Sales { get; set; }
        public List<int> Downloads { get; set; }
        public List<int> Queries { get; set; }
        public ProductSales(string country, int[] downloads, int[] sales, int[]
queries)
        {
            Country = country;
            Sales = sales.ToList();
            Downloads = downloads.ToList();
            Queries = queries.ToList();
        }
        public static List<ProductSales> GetData()
        {
            int[][][] stats = {

new int[][]{ new int[] { 8, 22, 24, 61, 77 }, new int[] { 30, 49, 80, 110, 130
```


Copyright © GrapeCity, Inc. All rights reserved.

```
},
new int[] { 6, 20, 32, 40, 76 } },

new int[][]{ new int[] {4, 4, 29, 39, 77}, new int[] {11, 30, 36, 99, 119},
new int[] {15, 26, 30, 34, 44} },

new int[][]{ new int[] {45, 72, 87, 89, 94}, new int[] {5, 9, 47, 60, 104},
new int[] {3, 13, 63, 66, 73} },

new int[][]{ new int[] {5, 10, 45, 51, 97}, new int[] {3, 35, 39, 56, 100},
new int[] {5, 31, 41, 76, 90} },

new int[][]{ new int[] {32, 43, 53, 78, 80}, new int[] {7, 20, 61, 74, 95},
new int[] {20, 22, 49, 80, 91} },

new int[][]{ new int[] {22, 28, 53, 63, 92}, new int[] {18, 50, 72, 100, 112},
new int[] {6, 18, 30, 42, 82} }

};
            var countries = new string[] { "米国", "ドイツ", "イギリス", "日本", "フラ
ンス", "中国" };
            var data = new List<ProductSales>();
            for (var i = 0; i < countries.Length; i++)
            {
                data.Add(new ProductSales(countries[i], stats[i][0], stats[i]
[1], stats[i][2]));
            }
            return data;
        }
    }
}
```

先頭に戻る

手順2：FlexChartへの箱ひげ図系列の追加

アプリケーションにFlexChartを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`BoxWhiskerController`）。
    3. **[追加]**をクリックします。
4. メソッドIndex()を次のメソッドに置き換えます。

C#

```
public ActionResult Index()
        {
            return View(ProductSales.GetData());
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
```

先頭に戻る

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、BoxWhiskerControllerをダブルクリックして開きます。
2. メソッドIndex()内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

Index.cshtml copyCode

```
@using C1.Web.Mvc.Chart;
@using BoxWhiskerSeries.Models;

@model IEnumerable<ProductSales>

@(Html.C1().FlexChart()
.Bind("Country", "Downloads", Model)
.Series(ser =>
{
    ser.AddBoxWhisker().Name("ダウンロード");
    ser.AddBoxWhisker().Binding("Sales").Name("販売").ShowOutliers(true);
    ser.AddBoxWhisker().Binding("Queries").Name("クエリ").ShowOutliers(true);
})
.Legend(Position.Top))
```

手順3:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を追加して(例:http://localhost:1234/**ErrorBar/Index**)、ビューを表示します。

先頭に戻る

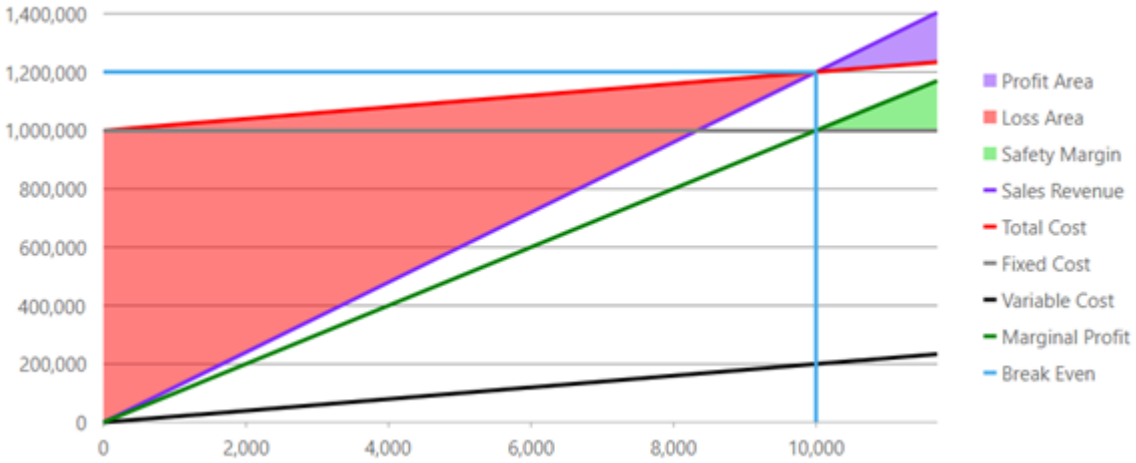
損益分岐点チャート

Break-even chart series is a graphical representation between cost, sales volume and profits. It shows the sales volume level at which total costs equal sales and helps in making profit planning. You can use this chart in the scenarios where you want to determine at what point your company, or a new product or service, will be profitable or say, the number of products/services your company needs to sell to at least cover your costs.

In FlexChart, the **BreakEven** class represents the Break-Even chart series. This series can be added to FlexChart controls using the **AddBreakEven** method to show the evolution of profits over time, and how long it takes to reach the break-even point where revenues surpass expenses.

The following example shows how you can create a BreakEven chart using the FlexChart control. In the example, based on the given data, the break-even point is 10,000 units sold. At this point, sales revenue and total cost, both are at 1,200,000 and cancel each other out. Also, the profit area is beyond this point.

Copyright © GrapeCity, Inc. All rights reserved.



Add Controller

BreakEvenController

```
public ActionResult Index()
{
     return View();
}
```

Add View for the Controller

Index.cshtml copyCode

```
@(Html.C1().FlexChart()
    .Series(ser =>
    {
        ser.AddBreakEven().FixedCost(1000000).VariableCost(20).SalesPrice(120)
        .Style(style => style.Fill("rgba(127,42,250,0.5)").StrokeWidth(0))
        .AltStyle(style => style.Fill("rgba(255,0,0,0.5)").StrokeWidth(0))
        .Styles(s =>
            s.SafetyMargin(sm => sm.Fill("lightgreen").StrokeWidth(0))
             .SalesRevenue(sr => sr.Stroke("rgba(127,42,250,1)").StrokeWidth(3))
             .FixedCost(fc => fc.Stroke("grey").StrokeWidth(3))
             .TotalCost(tc => tc.Stroke("red").StrokeWidth(3))
             .VariableCost(vc => vc.Stroke("black").StrokeWidth(3))
             .MarginalProfit(mp => mp.Stroke("green").StrokeWidth(3))
             .BreakEven(be => be.Stroke("rgba(69,171,235,1)").StrokeWidth(3))
             );
    })
)
```

誤差範囲チャート

誤差範囲チャートを使用すると、ユーザーは誤差の範囲や標準偏差を一目で見極めることができます。チャートは、標準誤差量、パーセント値、または標準偏差として表示することができます。ユーザーは、チャートに正確な誤差量を表示するために独自のカスタム値を設定することもできます。一般に、科学研究や科学実験の結果では、ErrorBarチャートを使用して、元の値からのデータのばらつきを表現します。

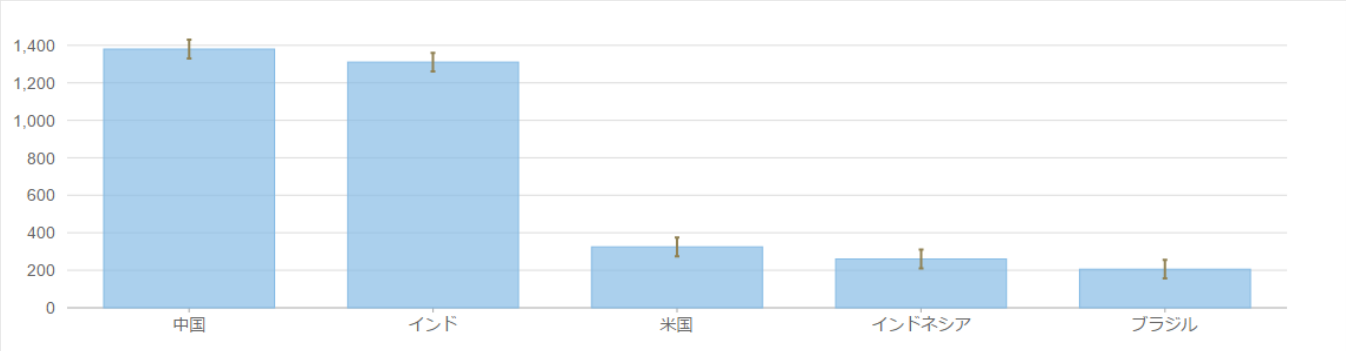
このトピックでは、FlexChartでErrorBar系列を使用して誤差や標準偏差を表す方法を説明します。ErrorBar系列を使用するに

Copyright © GrapeCity, Inc. All rights reserved.

は、**C1.Web.Mvc**名前空間に含まれる**FlexChart**クラスのインスタンスを作成する必要があります。

- **手順1：FlexChartのデータソースの作成**
- **手順2：FlexChartへのErrorBar系列の追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexChartを示しています。



手順1：FlexChartのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：PopulationByCountry.cs）。
2. 次のコードを新しいモデルに追加して、FlexChartのデータソースになるクラスを定義します。

PopulationByCountry.cs copyCode

```
namespace ErrorBar.Models
{
    public class PopulationByCountry
    {
        public string Country { get; set; }
        public int Population { get; set; }
        public static List<PopulationByCountry> GetData()
        {
            string[] countries = "中国,インド,米国,インドネシア,ブラジル".Split(new
char[] { ',' });
            int[] population = new int[] { 1380, 1310, 325, 260, 206 };
            var data = new List<PopulationByCountry>();

            for (var i = 0; i < countries.Length; i++)
            {
                data.Add(new PopulationByCountry()
                {
                    Country = countries[i],
                    Population = population[i]
                });
            }
            return data;
        }
    }
}
```

先頭に戻る

手順2：FlexChartへのErrorBar系列の追加

Copyright © GrapeCity, Inc. All rights reserved.

アプリケーションにFlexChartを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します(例:`ErrorBarController`)。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc.Chart;
```

5. メソッドIndex()を次のメソッドに置き換えます。

C#

```
public ActionResult Index()
        {
            return View(PopulationByCountry.GetData());
        }
```

先頭に戻る

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`ErrorBarController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

Index.cshtml copyCode

```
@using C1.Web.Mvc;

@model IEnumerable<PopulationByCountry>
@using ErrorBar.Models;
<br />
<br />
@(Html.C1().FlexChart()
.Bind("Country", "Population", Model)
.Series(ser =>
{
    ser.AddErrorBar().Value(50)
    .ErrorBarStyle(errorBarStyle =>
errorBarStyle.Fill("#e6e6e6").Stroke("#918254").StrokeWidth(2));
})
.Height("300px")
)
```


Copyright © GrapeCity, Inc. All rights reserved.

手順3：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を追加して（例：http://localhost:1234/**ErrorBar/Index**）、ビューを表示します。

先頭に戻る

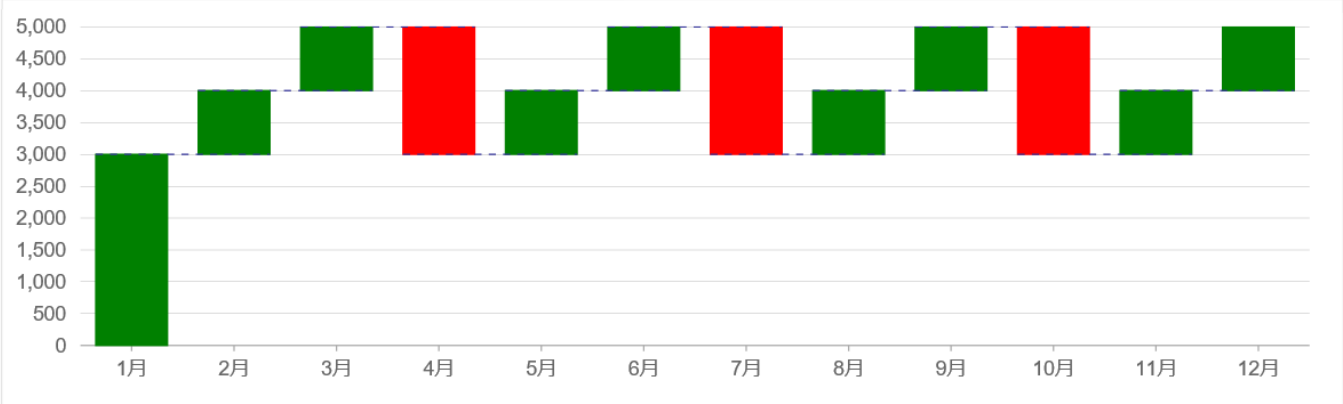
ウォータフォール

ウォータフォール系列は、FlexChart内で正と負の値の進行的な影響を理解するために役立つデータ仮想化形式です。ウォータフォール系列は、主に、連続的な値の推移すなわち増減を理解または説明する分析的な目的で使用されます。

このトピックでは、FlexChartのウォータフォール系列を使用して、エンティティの正と負の値を表す方法について説明します。

- **手順1：FlexChartのデータソースの作成**
- **手順2：FlexChartの追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexChartを示しています。



手順1：FlexChartのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Waterfall.cs`）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FlexChartのデータソースになるクラスを定義します。

C#

```
public class Waterfall
    {
        public int WaterfallId { get; set; }
        public string Name { get; set; }
        public int Value { get; set; }
        public Waterfall()
        {
        }
        public Waterfall(string name, int value)
        {
            Name = name;
            Value = value;
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public static List<Waterfall> GetData()
        {
            var names = new string[] { "1月", "2月", "3月", "4月", "5月", "6月",
"7月", "8月", "9月", "10月", "11月", "12月" };
            var data = new List<Waterfall>();
            for (int i = 0, len = names.Length; i < len; i++)
            {
                data.Add(new Waterfall(names[i], (((i % 3) + 3) * 1000)));
            };
            return data;
        }
}
```

先頭に戻る

手順2：FlexChartの追加

FlexChartを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順を実行します。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`FlexChartController`）。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc.Chart;
```

5. メソッドIndex()を次のメソッドに置き換えます。

C#

```
public ActionResult Index()
        {
            return View(Waterfall.GetData());
        }
```

先頭に戻る

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`FlexChartController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

Razor

```
@(Html.C1().FlexChart()
.Bind("Name", "Value", Model)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
.Series(ser =>
{
    ser.AddWaterfall()
    .RelativeData(false)
    .ConnectorLines(true)
    .Styles(s =>
    s.ConnectorLines(cnt => cnt.StrokeDasharray("5 5").Stroke("#339")));
}).Height("300px"))
```

手順3：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**FlexChart/Index**）。

先頭に戻る

スタイル設定とCSS

CSS Classes for the FlexChart control are as follows:

Class	Description
.wj-flexchart	Styles the div host element of the FlexChart control.
.wj-header .wj-title	Styles the header element.
.wj-footer .wj-title	Styles the footer element.
.wj-label	Styles the text for axis labels and legend labels.
Axis	
.wj-axis-x .wj-line	Styles the x-axis line.
.wj-axis-y .wj-line	Styles the y-axis line.
.wj-axis-x .wj-title	Styles the x-axis title.
.wj-axis-y .wj-title	Styles the y-axis title.
.wj-axis-x .wj-label	Styles the x-axis labels.
.wj-axis-y .wj-label	Styles the y-axis labels.
.wj-tick	Styles the tick marks
.wj-axis-x .wj-tick	Styles the x-axis tick marks.
.wj-axis-y .wj-tick	Styles the y-axis tick marks.
.wj-tick-minor	Styles the minor tick marks.
.wj-axis-x .wj-tick-minor	Styles the minor tick marks on x-axis, which are displayed when the MinorTickMarks property is set to true.
.wj-axis-y .wj-tick-minor	Styles the minor tick marks on y-axis, which are displayed when the MinorTickMarks property is set to true.
Plot Area	

Copyright © GrapeCity, Inc. All rights reserved.

.wj-plot-area>rect	Styles the plot area of the chart.
.wj-gridline	Styles the major gridlines, which is displayed when the MajorGrid property is set to true.
.wj-gridline-minor	Styles the minor gridlines, which is displayed when the MinorGrid property is set to true.
Legends	
.wj-legend > rect	Styles the legends of the chart.
.wj-legend .wj-label	Styles the labels for each chart legend entries.
Data Labels	
.wj-data-label	Styles all the data labels for each data point.
.wj-data-label-border	Styles the data label border
.wj-data-label-line	Styles the data label line.
Selection	
.wj-flexchart .wj-state-selected	Styles the selected items in the FlexChart control, when SelectionMode property is set to series or point.
Interaction	
.wj-flexchart .wj-chart-linemarker .wj-chart-linemarker-hline	Styles the horizontal line marker.
.wj-flexchart .wj-chart-linemarker .wj-chart-linemarker-vline	Styles the vertical line marker.
.wj-chart-linemarker-container	Styles the div host element that includes the line marker content.
.wj-flexchart .wj-chart-linemarker-draggable	Styles the cursor that appears while dragging the line marker.
.wj-chart-rangeslider .wj-chart-hrangeslider	Styles the range slider.
.wj-flexchart .wj-chart-rangeslider .wj-rangeslider-rangehandle	Styles the range slider bar.
.wj-chart-rangeslider button	Styles the button of range slider.
.wj-chart-rangeslider .wj-rangeslider-content	Styles the content of range slider.
.wj-chart-hrangeslider	Styles the horizontal range slider.
.wj-chart-hrangeslider .wj-rangeslider-minhandle	Styles the min handle of horizontal range slider.
.wj-chart-hrangeslider .wj-rangeslider-maxhandle	Styles the max handle of horizontal range slider.
.wj-chart-vrangeslider	Styles the vertical range slider.
.wj-chart-vrangeslider .wj-rangeslider-minhandle	Styles the min handle of vertical range slider.
.wj-chart-vrangeslider .wj-rangeslider-	Styles the max handle of vertical range slider.

Copyright © GrapeCity, Inc. All rights reserved.

maxhandle	
.wj-rangeslider-maxhandle .wj-rangeslider-handle-active	Styles the currently selected max range slider handler.
.wj-rangeslider-minhandle .wj-rangeslider-handle-active	Styles the currently selected min range slider handler.
.wj-panable	Styles the cursor that appears while performing the panning operation on the FlexChart control.
.wj-zoom-overlay	Styles the highlighting zooming area on the chart.

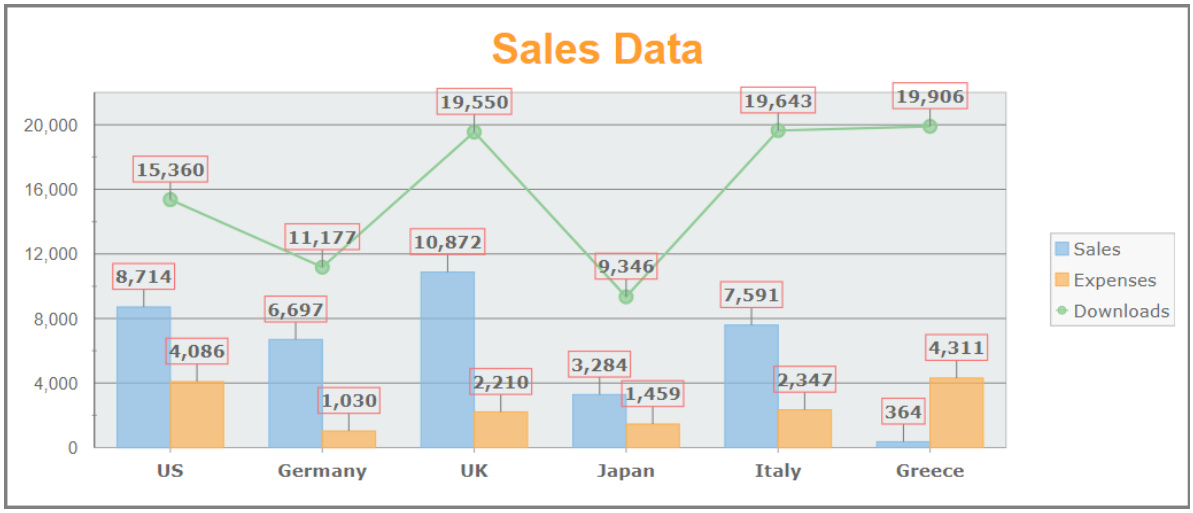
スタイルの適用

The appearance of the FlexChart control is defined using CSS classes which represents different elements, such as plot area, legends, axis, header, series group etc. You can customize these classes to change the appearance of the FlexChart control. The custom CSS rules can be applied to the FlexChart control, by applying a CSS to the FlexChart control using the CssClass property.

In this example, we customize the default CSS rules to make the following customizations to the FlexChart control.

1. Apply a solid border to the FlexChart control.
2. Change the background color for the plot area of the chart.
3. Change the font-size, font-weight and fill color to the header element.
4. Create a rectangle shape around the legends entries and change the background color.
5. Modify the font-weight and font-family for each data label entry.
6. Add a colored border for each data label entry in the chart.
7. Change the font-family and font-weight for the entries on the x-axis.

The following image shows how FlexChart appears after applying styles using custom CSS.



Add Custom Stylesheet

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

To add a custom style sheet in your application, follow these steps:

1. In the Solution Explorer, right-click the **Content** folder.
2. From the context menu, select **Add | Style Sheet**. The **Specify Name for Item** dialog appears.
3. Set name of the style sheet (for example: `app.css`) and click **OK**.
4. Replace the default code of **app.css** file with the code given below.

app.css

```
.custom.wj-flexchart {
            border: grey 3px solid !important;
            border-radius: 0;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            margin-top: 15px;
        }
        .custom .wj-legend > rect {
            fill: #f8f8f8;
            stroke: #c0c0c0;
        }
        .custom .wj-legend {
            font-family:Verdana, Geneva, Tahoma, sans-serif;
        }
        .custom .wj-data-label {
            font-weight: bold;
            font-family: Verdana, Geneva, Tahoma, sans-serif;
        }
        .custom .wj-data-label-border {
            stroke: rgb(246, 115, 115);
        }
        .custom .wj-header .wj-title {
            font-size: 25pt;
            fill: #FF9E2C;
            font-weight: bold;
        }
        .custom .wj-axis-x .wj-label {
            font-weight: bold;
            font-family: Verdana, Geneva, Tahoma, sans-serif;
            transform: rotate(45,100,);
        }
        .wj-plot-area > rect {
            fill: #EAEDED;
            stroke: #c0c0c0;
        }
```

Styling.cshtml

Razor

```
@using <ApplicationName>.Models
@model IEnumerable<FlexChartData.Sale>
@(Html.C1().FlexChart<FlexChartData.Sale>().Id("theChart")
            .Bind("Country", Model)
            .Header("Sales Data")
            .AxisY(ay => ay
            .AxisLine(true).Max(22000)
            .MinorTickMarks(C1.Web.Mvc.Chart.AxisTickMark.Cross))
            .DataLabel(dl =>
dl.Position(C1.Web.Mvc.Chart.LabelPosition.Top).ConnectingLine(true).Content("{y}").Border(true))
            .CssClass("custom")
            .Series(sb =>
            {
                sb.Add().Binding("Sales").Name("Sales");
                sb.Add().Binding("Expenses").Name("Expenses");

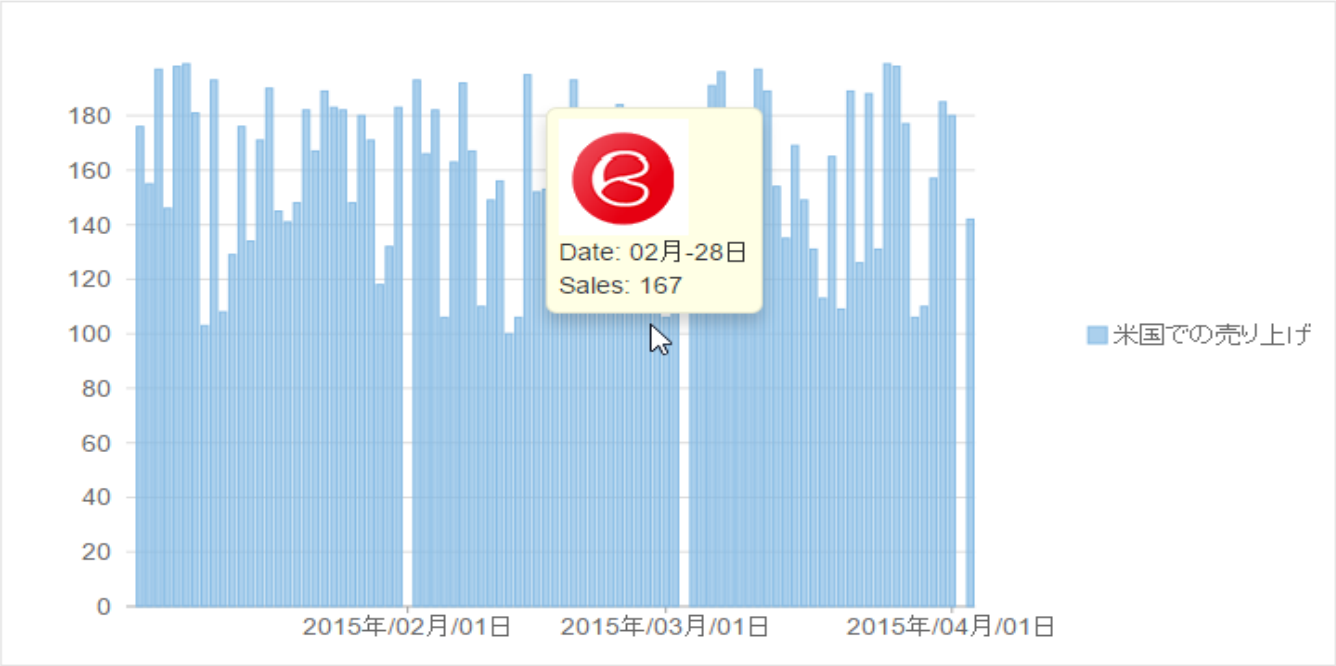
sb.Add().Binding("Downloads").Name("Downloads").ChartType(C1.Web.Mvc.Chart.ChartType.LineSymbols);
            }))
```

ツールチップ

デフォルトのツールチップには、通常、選択された点の X 値および Y 値と共に凡例の名前が表示されます。FlexChart を使用すると、ラベルや画像を含むカスタマイズされたツールチップを表示できます。それには、**Content** プロパティを設定します。

Copyright © GrapeCity, Inc. All rights reserved.

次の図は、カスタマイズされたツールチップが FlexChart に表示されたところです。



次のコード例は、ツールチップをカスタマイズする方法を示します。

コードの場合

それには、最初に、ツールチップに追加するコンテンツを作成する必要があります。ツールチップには、ラベル、画像などの要素を含めることができます。

JavaScript copyCode

```
<script type="text/javascript">
    var tooltipContent = function (ht)
    {
        return "<img src='../C1.png' alt='Others'  />"
            + '<br/>' + "Date: " + wijmo.Globalize.format(ht.x, 'MM月-dd日')
            + '<br/>' + "Sales: " + ht.y;
    };
</script>
```

次に示すように、チャートツールチップにビューを割り当てます。次の例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor copyCode

```
.Tooltip(tooltip => tooltip.Content("tooltipContent"))
```

近似曲線

近似曲線は、データの傾向を示し、予測問題の検討に利用されるツールです。価格チャートや金融チャートで使用されるのが多いですが、株価のテクニカル分析で使用される取引指標、MACD(Moving Average Convergence Divergence、移動平均収束拡散法)や金融市場の解析に使用されるテクニカル指標、RSI(Relative Strength Index、相対力指数)のようなテクニカル分析グラフでも使用されます。

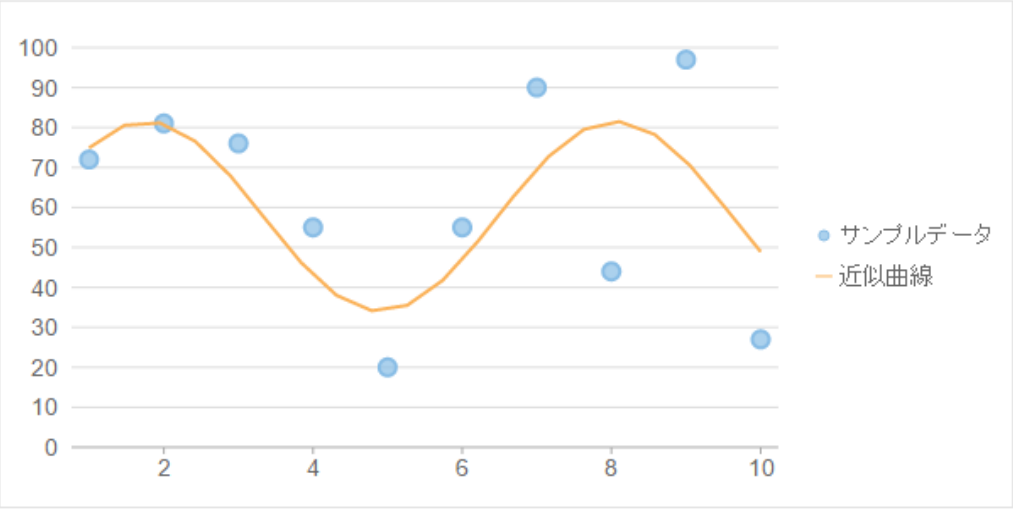
近似曲線の種類

FitType	説明

Copyright © GrapeCity, Inc. All rights reserved.

Average X	グラフからXの平均値を取得して近似曲線を作成します。
Average Y	グラフからYの平均値を取得して近似曲線を作成します。
Exponential	データ値の増減率が高くなっていく場合に適します。0 または負の値が含まれているデータの場合、指数近似曲線を作成できません。
Fourier	フーリエ級数式で作成され、波動関数を簡単な正弦波の組み合わせとして表示する方法です。
Linear	線形近似曲線は、単純な線形の関係を持つデータに適した直線です。データ要素のパターンが直線に近い場合、データは線形です。R-2 乗値は1の近くにある場合この線形近似曲線は適しています。
Logarithmic	データの可視化に最適する曲線です。データの変化率が急速に増加または減少してから平になる場合に適します。この曲線では負の値と正の値を使用できます。
Max X	グラフからXの最大値を取得して近似曲線を作成します。
Max Y	グラフからYの最大値を取得して近似曲線を作成します。
Min X	グラフからXの最小値を取得して近似曲線を作成します。
Min Y	グラフからYの最小値を取得して近似曲線を作成します。
Polynomial	データが変動する場合に使用される曲線です。大量のデータで利益や損失を分析するのに採用されます。
Power	特定な比率で増加する測定を比較するデータセットの場合に適します。たとえば、超特急列車の加速を 1 秒間隔で計測した値など。

チャートに近似曲線を追加してデータの傾向を表示できます。それで、Y軸のデータは、X軸のデータより増加または、減少することを簡単に解析できます。
次の図は、FlexChart 上で FitType を Fourier に設定した近似曲線を示しています。



次のコード例は、近似曲線を FlexChart に追加する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Example Title

```
@using C1.Web.Mvc.Chart
@model List<MathPoint>

@(Html.C1().FlexChart().Id("trendChart")
    .Bind("X", Model)
    .AxisY(ay => ay.Min(0).Max(100))
    .Legend(Position.Right)
    .Width(600)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .Height(300)
    .LegendToggle(true)
    .Series(ss =>
    {
ss.Add(ChartType.Scatter, "サンプルデータ").Binding("Y");
     ss.AddTrendLine("Trend Line").Binding("Y").SampleCount(20)
         .TrendLineOrder(3).FitType(TrendLineFitType.Fourier);
    }))
```

FlexGrid

FlexGrid コントロールは、データソースから取得したデータを表形式で表示する強力で柔軟な方法を提供します。**FlexGrid** コントロールは完全な機能を備えたグリッドです。複数の選択モード、ソート、列の並べ替え、グループ化、フィルタ処理、編集、カスタムセル、XAML スタイルのスターサイズの列、行および列の仮想化など、さまざまな機能を提供します。

FlexGrid は、非連結列、条件付き書式設定、およびセルレベルのカスタマイズによる柔軟性の高い設計を提供します。このコントロールを使用して、データベースに連結されたデータを編集および更新できる複雑なグリッドベースのアプリケーションを作成できます。

	ID	国名	製品名	傾向	ランク
	1	ドイツ	Gadget		★★★★☆
	2	カナダ	Gadget		★★☆☆☆
	3	日本	Gadget		★★★★★
	4	ドイツ	Gadget		★☆☆☆☆
	5	ドイツ	Gadget		★★☆☆☆
	6	カナダ	Gadget		★★★★★
	7	中国	Widget		★★★☆☆
	8	米国	Widget		★★☆☆☆
	9	韓国	Gadget		★★☆☆☆
	10	米国	Widget		★☆☆☆☆
	11	フランス	Widget		★★☆☆☆
	12	韓国	Widget		★★★★★
	13	ドイツ	Gadget		★☆☆☆☆
	14	ドイツ	Widget		★★★★☆
	15	日本	Gadget		★★★★★

主な特長

FlexGrid offers many advanced features beyond simple data management. These features are listed below:

- **セルの結合**
  AllowMerging を true に設定して、同じコンテンツを含むセルをマージします。

- **ページング**
  PageSizeproperty プロパティの値を1ページに入れる項目の数に設定します。

Copyright © GrapeCity, Inc. All rights reserved.

- **ソート**
  FlexGrid enables you to sort the grid data simply by tapping column's header and allows you to choose if you want to sort the grid by a single column or multiple columns at a time.

- **Show data as a tree**
  ChildItemsPath プロパティの値を設定して、グリッドをツリービュー形式で表します。

- **サーバー側のスクロールを無効にする**
  DisableServerRead プロパティの値を True に設定して、サーバー側のグリッドのスクロールを無効化します。

- **ヘッダーの表示/非表示**
  HeadersVisibility の値を None、All、Column、または Row に設定して、グリッド内のヘッダーの表示/非表示を指定します。

- **IME サポート**
  ImeEnabled プロパティの値を True に設定して、グリッドがIME（Input Method Editor）をサポートするようにします。このプロパティは、日本語、中国語、韓国語など、IMEサポートを必要とする言語のサイト/アプリケーションにのみ関係します。

- **Automatic row resizing**
  Row height of FlexGrid can get updated when the cells are edited or columns are resized. This can be done using the **AutoRowHeights** property of FlexGrid.

- **Lazy loading**
  FlexGrid provides the LazyRender property that allows you to skip rendering the grid cells that were already updated in the last render cycle. This can decrease the rendering time.

- **Easy to click check boxes**
  FlexGrid allows you to toggle between the big and the regular HTML check boxes using the **BigCheckboxes** property. You can use the big check boxes if you need them to cover the whole width of the cell they are used in.

- **Preserve White Spaces**
  FlexGrid lets you keep the whitespace in cells as they appear in the data using the **PreserveWhiteSpace** property.

クイックスタート：FlexGrid へのデータの追加

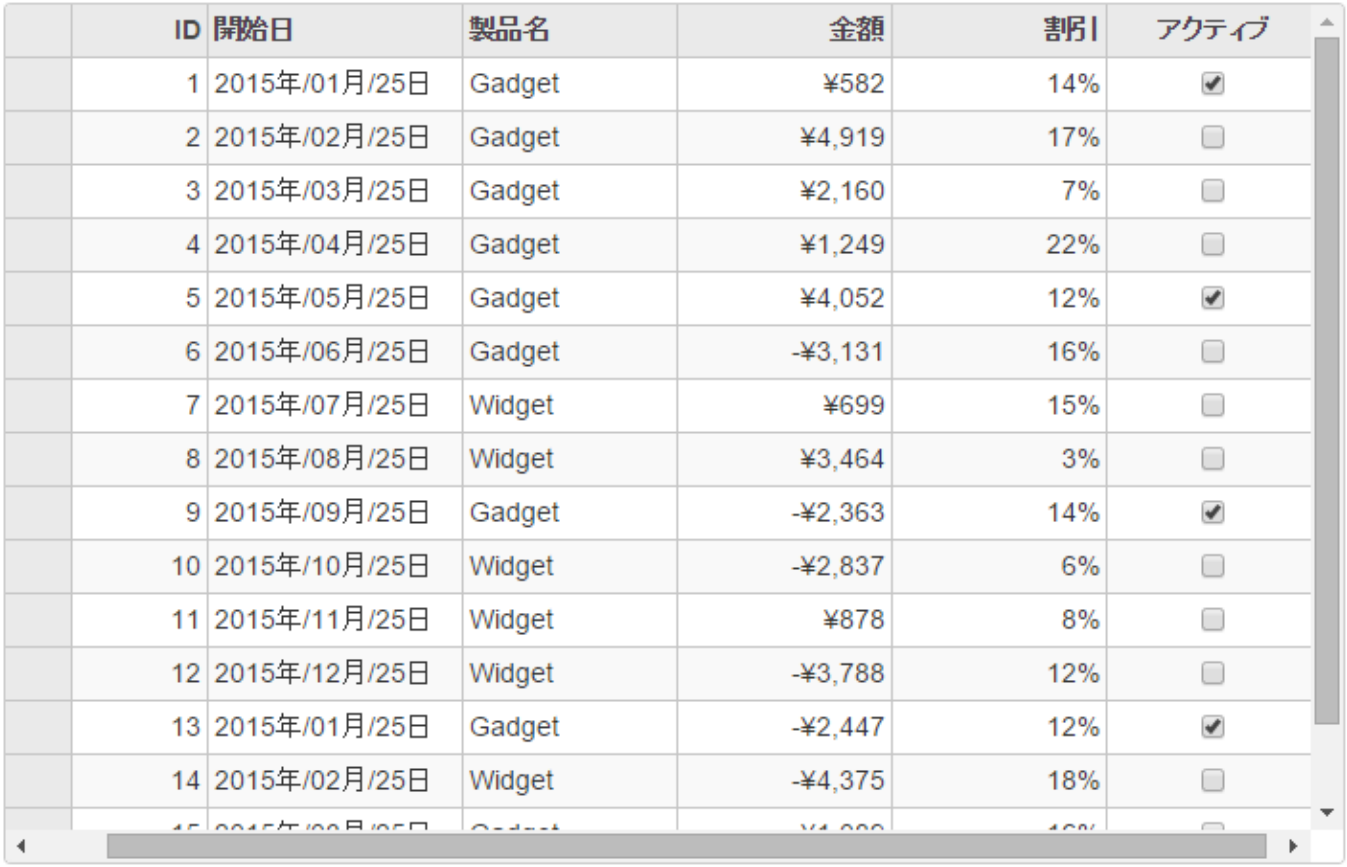
このセクションでは、MVC Web フォームに FlexGrid コントロールを追加し、コントロールにデータを挿入する方法について説明します。以下の例は、FlexGridコントロールにローカルモデルのバインドを示しています。FlexGridでリモート連結については、「データ連結」を参照してください。

このトピックは3つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：FlexGrid のデータソースの作成**
- **手順 3：FlexGrid コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の FlexGrid を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



ID	開始日	製品名	金額	割引	アクティブ
1	2015年/01月/25日	Gadget	¥582	14%	☑
2	2015年/02月/25日	Gadget	¥4,919	17%	☐
3	2015年/03月/25日	Gadget	¥2,160	7%	☐
4	2015年/04月/25日	Gadget	¥1,249	22%	☐
5	2015年/05月/25日	Gadget	¥4,052	12%	☑
6	2015年/06月/25日	Gadget	-¥3,131	16%	☐
7	2015年/07月/25日	Widget	¥699	15%	☐
8	2015年/08月/25日	Widget	¥3,464	3%	☐
9	2015年/09月/25日	Gadget	-¥2,363	14%	☑
10	2015年/10月/25日	Widget	-¥2,837	6%	☐
11	2015年/11月/25日	Widget	¥878	8%	☐
12	2015年/12月/25日	Widget	-¥3,788	12%	☐
13	2015年/01月/25日	Gadget	-¥2,447	12%	☑
14	2015年/02月/25日	Widget	-¥4,375	18%	☐

手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：FlexGrid のデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`FlexGridDataSource.cs`）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FlexGrid コントロールのデータソースになるクラスを定義します。

Sale.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using C1.Web.Mvc;
using C1.Web.Mvc.Serialization;
using MVCFlexGrid_JP.Models;



namespace MVCFlexGrid_JP.Models
{
    public class Sale
    {
        public int ID { get; set; }
    public DateTime 開始日 { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public DateTime 終了日 { get; set; }
    public string 国名 { get; set; }
    public string 製品名 { get; set; }
    public string 色 { get; set; }
    public double 金額 { get; set; }
    public double 金額2 { get; set; }
    public double 割引 { get; set; }
    public bool アクティブ { get; set; }

    public MonthData[] 傾向 { get; set; }
    public int ランク { get; set; }

    /// <summary>
    /// データを取得
    /// </summary>
    /// <param name="total"></param>
    /// <returns></returns>
    public static IEnumerable<Sale> GetData(int total)
    {
        var countries = new[] { "米国", "イギリス", "カナダ", "日本", "中国",
    "フランス", "ドイツ", "イタリア", "韓国", "オーストラリア" };
        var products = new[] { "Widget", "Gadget", "Doohickey" };
        var colors = new[] { "黒色", "白色", "赤色", "緑色", "青い色" };
        var rand = new Random(0);
        var dt = DateTime.Now;
        var list = Enumerable.Range(0, total).Select(i =>
        {
            var country = countries[rand.Next(0, countries.Length - 1)];
            var product = products[rand.Next(0, products.Length - 1)];
            var color = colors[rand.Next(0, colors.Length - 1)];
            var date = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60, i %
60);

            return new Sale
            {
                ID = i + 1,
                開始日 = date,
                終了日 = date,
                国名 = country,
                製品名 = product,
                色 = color,
                金額 = rand.NextDouble() * 10000 - 5000,
                金額2 = rand.NextDouble() * 10000 - 5000,
                割引 = rand.NextDouble() / 4,
                アクティブ = (i % 4 == 0),
                傾向 = Enumerable.Range(0, 12).Select(x =>
                new MonthData { Month = x + 1,
               Data = rand.Next(0, 100) }).ToArray(),
                ランク = rand.Next(1, 6)
            };
        });
        return list;
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    internal static dynamic GetCountries()
    {
        throw new NotImplementedException();
    }

    internal static dynamic GetProducts()
    {
        throw new NotImplementedException();
    }
}

public class MonthData
{
    public int Month { get; set; }
    public double Data { get; set; }
}
    }
```

先頭に戻る

手順 3：FlexGrid コントロールの追加

FlexGrid コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すように MVC 参照を追加します。

```
C#                                                                    copyCode
using System.Collections;
using System.Globalization;
using System.Linq;
using System.Web.Mvc;
using C1.Web.Mvc;
using MVCFlexGrid_JP.Models;
using System.Collections.Generic;
using System;
```

5. メソッド Index() を次のメソッドに置き換えます。

QuickStartController.cs

C#

```
C#                                                                    copyCode
public ActionResult Index()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{
    return View(Sale.GetData(15));
}
```

VB

VB

```
Public Function Index() As ActionResult
        Return View(Sale.GetData(15))
End Function
```

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されました。
6. ソリューションエクスプローラーで、`Index.cshtml` をダブルクリックします。
7. **Views\Index.cshtml** ファイルのデフォルトのコードを次のコードに置き換えて、**FlexGrid** コントロールを初期化します。

HTML Helpers

Razor（Index.cshtml） copyCode

```
@using C1.Web.Mvc;
@using MVCFlexGrid_JP.Models;
@using C1.Web.Mvc.Serializition;
@using C1.Web.Mvc.Grid;
@using C1.Web.Mvc.Fluent;
@using C1.Web.Mvc.CollectionView;


@model IEnumerable<Sale>


  // FlexGridのインスタンスを作成し、プロパティを設定します
@(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .Height(450)
    .Width(700)
    .AllowAddNew(true)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .CssClass("grid")
    .Bind(Model)
        // FlexGridに列データを連結します
     .Columns(bl =>
       {
           bl.Add(cb => cb.Binding("ID"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            bl.Add(cb => cb.Binding("開始日").Format("yyyy年/MM月/dd日"));
            bl.Add(cb => cb.Binding("製品名"));
            bl.Add(cb => cb.Binding("金額").Format("c"));
            bl.Add(cb => cb.Binding("割引").Format("p0"));
            bl.Add(cb => cb.Binding("アクティブ"));




        })

)
```

Index.vbhtml

```
@Imports C1.Web.Mvc
@Imports C1.Web.Mvc.Fluent
@Imports C1.Web.Mvc.Grid
@ModelType IEnumerable(Of Sale)

@(Html.C1().FlexGrid(Of Sale)() _
.CssClass("grid") _
.Bind(Model) _
.AutoGenerateColumns(False) _
.Width(700) _
.Height("800px") _
.SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell) _
.Columns(Sub(bl)
                bl.Add(Sub(cb) cb.Binding("ProductName"))
                bl.Add(Sub(cb) cb.Binding("SupplierID"))
                bl.Add(Sub(cb) cb.Binding("QuantityPerUnit"))
                bl.Add(Sub(cb) cb.Binding("UnitPrice"))
                bl.Add(Sub(cb) cb.Binding("UnitsInStock"))
                bl.Add(Sub(cb) cb.Binding("UnitsOnOrder"))
                bl.Add(Sub(cb) cb.Binding("Discontinued"))
                End Sub) _)
```

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

先頭に戻る

FlexGridの使用

This section comprises all the features and auxiliary functionalities offered by FlexGrid.

列

Learn how to work with columns in FlexGrid.

Copyright © GrapeCity, Inc. All rights reserved.

カスタムセルテンプレート
Learn how to set the custom cell templates in a column of FlexGrid.
カスタムエディタ
Learn how to add custom editors in FlexGrid.
データ連結
Learn about different types of data binding in FlexGrid.
DataMap
Learn how to create data mapping in FlexGrid.
詳細行
Learn how to enable Detail Row in FlexGrid.
サーバー読み取りの無効化
Learn how to disable server reading in FlexGrid.
編集
Learn batch and excel style editing in FlexGrid.
Excelのインポートまたはエクスポート
Learn how to export or import FlexGrid content to/from an Excel xlsx file.
PDF へのエクスポート
Learn how to export FlexGrid to PDF file.
排他値の検索
Learn how to include exclusive value search in FlexGrid.
フィルタ処理
Learn how to enable Filtering in FlexGrid.
FlexGridセレクター
Learn how to use FlexGrid Selector.
フルテキスト検索
Learn how to perform full text search in FlexGrid.
大文字と小文字を区別する検索
Learn how to perform case-sensitive search in FlexGrid.
グループ化
Learn how to group data in FlexGrid at runtime and through code.
キーボードハンドリング
Learn ways to provide keyboard support in MVC applications using FlexGrid.
マージ
Learn how to enable Merging in the FlexGrid.
ページング
Learn how to enable Paging using Pager control in FlexGrid.
右から左へのレンダリング
Learn how to use in-built functionality of rendering in FlexGrid.
選択モード
Learn how to set the selection mode in FlexGrid.
スターサイズ
Learn how to apply star sizing to the columns of FlexGrid.
TreeGrid
Learn how to enable TreeView in FlexGrid.
非連結のFlexGrid
Learn how to set the data by adding code to the client side using JavaScript and AJAX.
控えめな検証
Learn how to validate data using unobtrusive validation
仮想スクロール
Learn how to enable Virtual Scrolling in FlexGrid.

列

To understand how you can work with FlexGrid columns, see the following topics.

Copyright © GrapeCity, Inc. All rights reserved.

列フッターパネル
Learn how to show the column footer panel in FlexGrid.
列グループ
Learn how to create column groups in FlexGrid.
列ヘッダーのグループ化
Learn how to implement the column header grouping in FlexGrid.
列の固定
Learn how to pin the columns.
列レイアウトの保存とロード
Learn how to set Save and Load Column layout in FlexGrid.
クライアント側でグリッド列を表示/非表示
Learn how to hide or show any columns on a button click.
非連結列
Learn how to add an unbound column in FlexGrid.

列フッターパネル

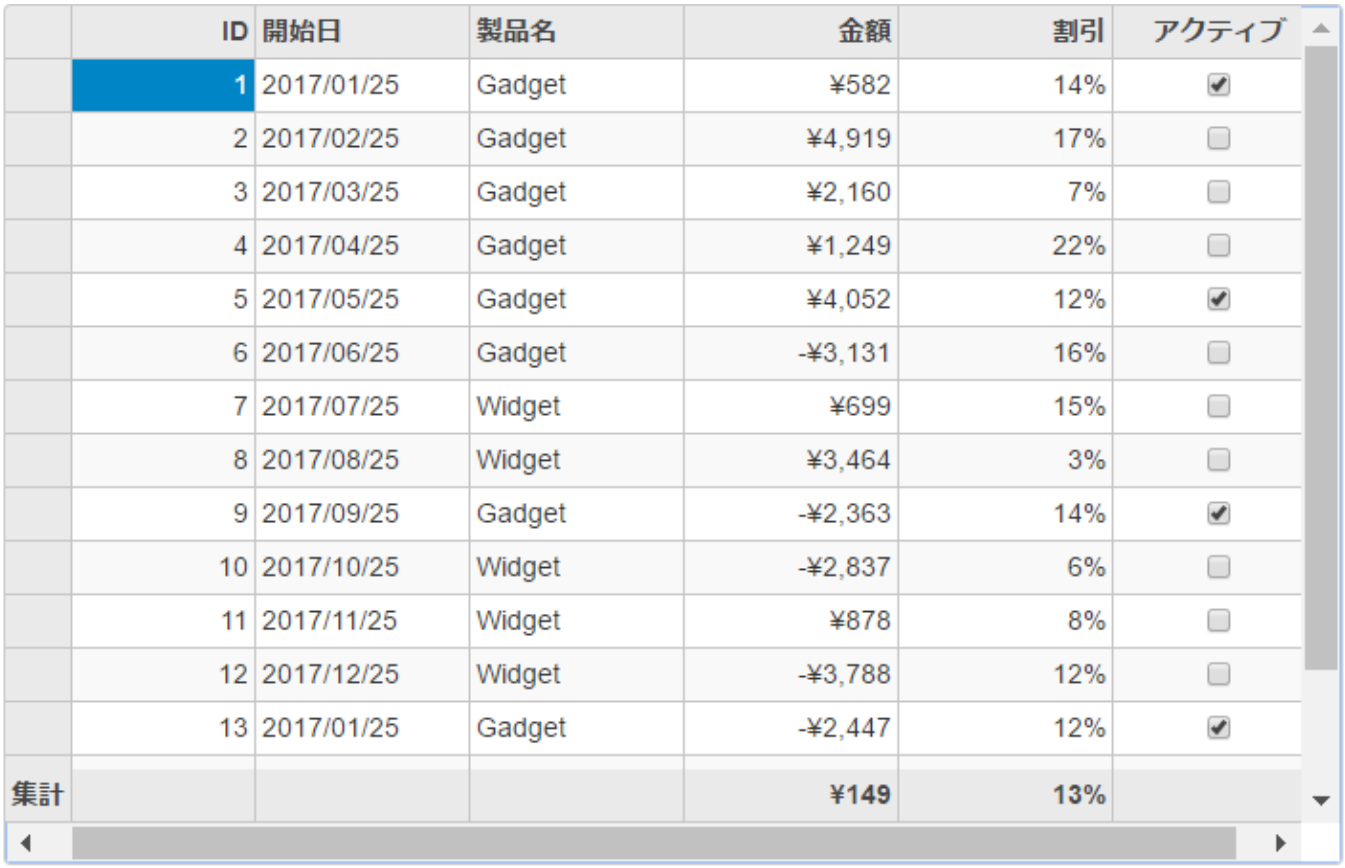
FlexGridは列フッターをサポートしており、ここに、集計データを表示するグループ行を表示できます。集計データを計算し、その結果を列フッターパネルに表示できます。FlexGridに列フッターパネルを表示するには、**ShowColumnFooters**プロパティを**true**に設定する必要があります。また、グループ行の行ヘッダーテキストを設定することもできます。設定しない場合は、デフォルト値のシグマ文字（Σ）が使用されます。

ShowColumnFooterプロパティは、現在の列のすべての値の集計値を表示します。

1. DisableServerReadが**true**の場合は、すべての行データを取得し、すべての行の集計を計算します。
2. DisableServerReadが**false**で、かつInitialItemsCountが設定されている場合、グリッドは仮想スクロールを使用してクライアントに送信されます。すべての行データが送信されるわけではありません。したがって、一部の行だけが集計対象となり、集計値が正しくなくなる可能性があります。

次の図は、ShowColumnFootersプロパティを設定した後のFlexGridを示しています。この例では、「クイックスタート」トピックで追加したSale.csモデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.



	ID	開始日	製品名	金額	割引	アクティブ
	1	2017/01/25	Gadget	¥582	14%	☑
	2	2017/02/25	Gadget	¥4,919	17%	☐
	3	2017/03/25	Gadget	¥2,160	7%	☐
	4	2017/04/25	Gadget	¥1,249	22%	☐
	5	2017/05/25	Gadget	¥4,052	12%	☑
	6	2017/06/25	Gadget	-¥3,131	16%	☐
	7	2017/07/25	Widget	¥699	15%	☐
	8	2017/08/25	Widget	¥3,464	3%	☐
	9	2017/09/25	Gadget	-¥2,363	14%	☑
	10	2017/10/25	Widget	-¥2,837	6%	☐
	11	2017/11/25	Widget	¥878	8%	☐
	12	2017/12/25	Widget	-¥3,788	12%	☐
	13	2017/01/25	Gadget	-¥2,447	12%	☑
集計				¥149	13%	

コードの場合

Razor

```
@using VirtualScrolling.Models
    @using C1.Web.Mvc.Grid

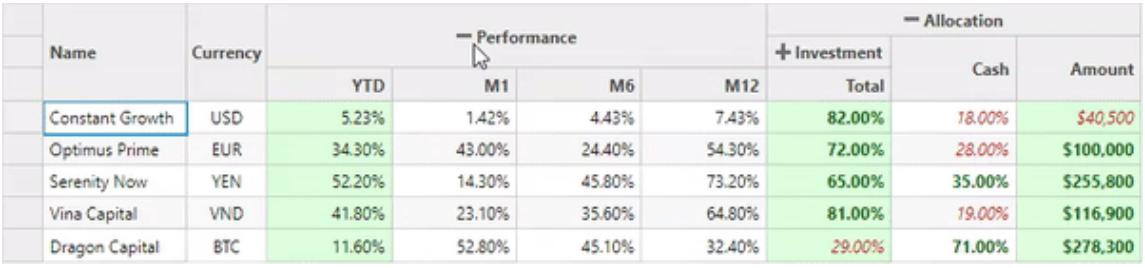
    @model IEnumerable<Sale>

    @(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .Height(450)
    .Width(700)
    .AllowAddNew(true)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .CssClass("grid")
    .ShowColumnFooters(true, "集計")
    .Bind(Model)
     .Columns(bl =>
     {
         bl.Add(cb => cb.Binding("ID"));
         bl.Add(cb => cb.Binding("開始日"));
         bl.Add(cb => cb.Binding("製品名"));
         bl.Add(cb => cb.Binding("金額").Format("c").Aggregate(Aggregate.Sum));
         bl.Add(cb => cb.Binding("割引").Format("p0").Aggregate(Aggregate.Avg));
         bl.Add(cb => cb.Binding("アクティブ"));
     })
    )
```


Copyright © GrapeCity, Inc. All rights reserved.

列グループ

In addition to hierarchical column headers, FlexGrid enables grouping the related fields together. This provides you the ability to create collapsible column groups. This is done by setting the columns property (or the columnGroups property) to an array containing column definitions where the definitions contain a 'columns' collection of child columns. The column groups may be collapsible and configured to show any of the child columns when the group is collapsed by setting ColapseTo and IsColapsed properties of the column.



Name	Currency	− Performance				− Allocation		
		YTD	M1	M6	M12	+ Investment Total	Cash	Amount
Constant Growth	USD	5.23%	1.42%	4.43%	7.43%	82.00%	18.00%	$40,500
Optimus Prime	EUR	34.30%	43.00%	24.40%	54.30%	72.00%	28.00%	$100,000
Serenity Now	YEN	52.20%	14.30%	45.80%	73.20%	65.00%	35.00%	$255,800
Vina Capital	VND	41.80%	23.10%	35.60%	64.80%	81.00%	19.00%	$116,900
Dragon Capital	BTC	11.60%	52.80%	45.10%	32.40%	29.00%	71.00%	$278,300

The following example demonstrates how you can create column groups by initializing the grid's column set as hierarchical columns.

Controller Code

C#

```
using System.Web.Mvc;
using C1.Web.Mvc;
using ApplicationName.Models;

namespace ApplicationName.Controllers
{
    public class Portfolio
    {
        public Portfolio(string name, string currency, params double[] args)
        {
            Name = name;
            Currency = currency;
            YTD = args[0];
            M1 = args[1];
            M6 = args[2];
            M12 = args[3];
            Stock = args[4];
            Bond = args[5];
            Cash = args[6];
            Other = args[7];
            Amount = args[8];
        }

        public string Name;
        public string Currency;
        public double YTD;
        public double M1;
        public double M6;
        public double M12;
        public double Stock;
        public double Bond;
        public double Cash;
        public double Other;
        public double Amount;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    }

    public partial class FlexGridController : Controller
    {
        private static List<Portfolio> _portfolios = new List<Portfolio>
        {
            new Portfolio("Constant Growth", "USD",   0.0523, 0.0142, 0.0443, 0.0743,
0.17, 0.50, 0.18, 0.15, 40500),
            new Portfolio("Optimus Prime",  "EUR",  0.343,  0.430,  0.244,  0.543,  0.33,
0.17,   0.28,   0.22, 100000),
            new Portfolio("Serenity Now",   "YEN",  0.522,  0.143,  0.458,  0.732,  0.15,
0.11,   0.35,   0.39, 255800),
            new Portfolio("Vina Capital",   "VND",  0.418,  0.231,  0.356,  0.648,  0.25,
0.31,   0.19,   0.25, 116900),
            new Portfolio("Dragon Capital",   "BTC",  0.116,  0.528,  0.451,  0.324,
0.15,   0.14,   0.71,   0, 278300)
        };

        public ActionResult ColumnGroups(FormCollection collection)
        {
            return View(_portfolios);
        }
    }
}
```

View Code

C#

```
@using C1.Web.Mvc.Grid
@using C1.Web.Mvc
@model  IEnumerable<ApplicationName.Controllers.Portfolio>

@{
    const string tpl1 = "<span class=${value> .3 ? 'big-val' : 'small-val'}>${text}
</span>";
    const string tpl2 = "<span class=${value > 50000 ? 'big-val' : 'small-val'}>${text}
</span>";
}

@(Html.C1().FlexGrid().Id("colGroupsFlexGrid")
    .Bind(Model)
    .AutoGenerateColumns(false)
    .SortingType(AllowSorting.None)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("Name").Width("130"));
        bl.Add(cb => cb.Binding("Currency").Align("center").Width("73"));
        bl.Add(cb =>
cb.Header("Performance").Align("center").CollapseTo("YTD").Columns(bl1 =>
        {
            bl1.Add(cb1 => cb1.Binding("YTD").Format("p2").CssClass("main-column"));
            bl1.Add(cb1 => cb1.Binding("M1").Format("p2"));
            bl1.Add(cb1 => cb1.Binding("M6").Format("p2"));
            bl1.Add(cb1 => cb1.Binding("M12").Format("p2"));
        })));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        bl.Add(cb =>
cb.Header("Allocation").Align("center").CollapseTo("Amount").Columns(bl2 =>
        {
        bl2.Add(cb2 =>
cb2.Header("Investment").Align("center").CollapseTo("Total").IsCollapsed(true).Columns(bl3
=>
        {
            bl3.Add(cb3 => cb3.Binding("Stock").Format("p2").Template(tpl1));
            bl3.Add(cb3 => cb3.Binding("Bond").Format("p2").Template(tpl1));
            bl3.Add(cb3 => cb3.Binding("Other").Format("p2").Template(tpl1));
            bl3.Add(cb3 =>
cb3.Name("#Total").Header("Total").Format("p2").Align("right").CssClass("main-column"));
        }));
        bl2.Add(cb2 => cb2.Binding("Cash").Format("p2").Template(tpl1));
        bl2.Add(cb2 => cb2.Binding("Amount").Format("c0").CssClass("main-
column").Template(tpl2));
    }));
    })
    .ShowMarquee(true)
    .DefaultColumnSize(113)
    .CssClass("grid animated")
    .ItemFormatter("formatItem")
)

<style type="text/css">
    .wj-flexgrid {
        margin: 10px 0;
    }

        /* グループのメイン列を強調表示します */
        .wj-flexgrid .wj-cells .wj-cell.main-column {
            background: #ddffdd;
        }

    /* セルのテンプレートを書式設定します */
    .big-val {
        font-weight: bold;
        color: darkgreen;
    }

    .small-val {
        font-style: italic;
        color: rgb(202, 0, 0);
    }

    /* グループを折りたたむ/展開するためのアニメーション */
    .wj-flexgrid.animated .wj-colheaders .wj-header.wj-cell.wj-colgroup {
        transition: all .2s;
    }
</style>

<script>
    function formatItem(panel, r, c, cell) {
        if (panel.cellType == 1) {
            switch (panel.columns[c].name) {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                case '#Total':
                    var item = panel.rows[r].dataItem,
                        value = item.Stock + item.Bond + item.Other,
                        cls = value > .3 ? 'big-val' : 'small-val';
                    txt = wijmo.Globalize.format(value, panel.columns[c].format);
                    cell.innerHTML = '<span class="' + cls + '">' + txt + '</span>'
                    break;
            }
        }
    }
</script>
```

列ヘッダーのグループ化

Column header grouping that allows you to customize the layout of the column header. This is very helpful in scenarios, where you need to group similar information in the columns or to display some addtional information in the header section. The columns have multi-level and merged headers that reflect the structure of the data. The whole column group can be selected by clicking the header cells.

FlexGrid allows you to implement the column header grouping by using the **HeaderTemplate** property provided by the **FlexGrid** class. Moreover, the **HeaderTemplate** class provides **RowCount** and **Cells** properties for creating groups in the column header as shown in the code below.

Note: When the **HeaderTemplate** property is applied to the FlexGrid control, column dragging is disabled.

	Name	Currency	Performance				Allocation			
			YTD	1 M	6 M	12 M	Stocks	Bonds	Cash	Others
	Constant Growth IXTR	USD	5.23%	1.42%	4.43%	7.43%	17%	32%	36%	15%
	Optimus Prime MMCT	EUR	3.43%	4.30%	2.44%	5.43%	61%	80%	90%	22%
	Serenity Now ZTZZZ	YEN	5.22%	1.43%	4.58%	7.32%	66%	9%	19%	6%

Model - DataRepresentation.cs

C#

```
public class DataRepresentation
    {
        public DataRepresentation(params string[] args)
        {
            name = args[0];
            currency = args[1];
            ytd = args[2];
            m1 = args[3];
            m6 = args[4];
            m12 = args[5];
            stock = args[6];
            bond = args[7];
            cash = args[8];
            other = args[9];
        }
        public string name;
        public string currency;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public string ytd;
        public string m1;
        public string m6;
        public string m12;
        public string stock;
        public string bond;
        public string cash;
        public string other;

        //データを取得します
        public static IEnumerable<DataRepresentation> GetData()
        {
            List<DataRepresentation> list = new List<DataRepresentation>();
            list.Add(new DataRepresentation("Constant Growth IXTR", "USD", "5.23%",
"1.42%", "4.43%", "7.43%", "17%", "32%", "36%", "15%"));
            list.Add(new DataRepresentation("Optimus Prime MMCT", "EUR", "3.43%",
"4.30%", "2.44%", "5.43%", "61%", "80%", "90%", "22%"));
            list.Add(new DataRepresentation("Serenity Now ZTZZZ", "YEN", "5.22%",
"1.43%", "4.58%", "7.32%", "66%", "9%", "19%", "6%"));
            return list;
        }
    }
```

Controller - ColumnHeaderController.cs

C#

```
public class HeaderTemplateController : Controller
    {
        // GET: HeaderTemplate
        public ActionResult Index()
        {
            return View(DataRepresentation.GetData());
        }
    }
```

View - ColumnHeader.cshtml

HTML copyCode

```
@using <ApplicationName>.Models;
@model IEnumerable<DataRepresentation>

<style>
  .wj-colheaders .wj-cell {text-align: center !important;}
</style>

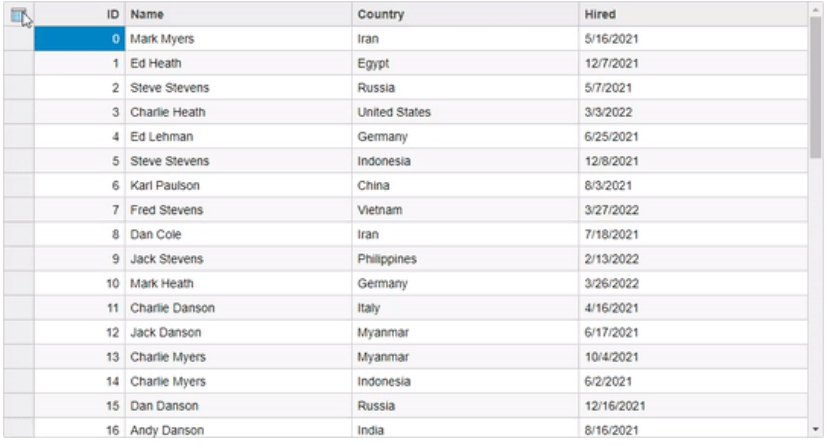
@(Html.C1().FlexGrid().Id("fnFlexGrid")
                    .Bind(Model)
                    .Width(1600)
                    .AutoGenerateColumns(false)
                    .AllowSorting(false)
                        .Columns(bl =>
                        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            bl.Add(cb => cb.Binding("name").Header("Name"));
            bl.Add(cb => cb.Binding("currency").Header("Currency"));
            bl.Add(cb => cb.Binding("ytd").Header("YTD"));
            bl.Add(cb => cb.Binding("m1").Header("1 M"));
            bl.Add(cb => cb.Binding("m6").Header("6 M"));
            bl.Add(cb => cb.Binding("m12").Header("12 M"));
            bl.Add(cb => cb.Binding("stock").Header("Stocks"));
            bl.Add(cb => cb.Binding("bond").Header("Bonds"));
            bl.Add(cb => cb.Binding("cash").Header("Cash"));
            bl.Add(cb => cb.Binding("other").Header("Others"));
        })
        .HeaderTemplate(builder =>
        {
            builder.RowCount(2);
            builder.Cells(c =>
            {
                c.Set(0, 0, 2, 1, "Name");
                c.Set(0, 1, 2, 1, "Currency");
                c.Set(0, 2, 1, 4, "Performance");
                c.Set(0, 6, 1, 4, "Allocation");
            });
        }))
```

列ピッカー

FlexGrid supports column picker to add or remove a visible column through the column-picker icon. This icon appears at the top-left cell of the grid to show the column-picker dropdown list where you can select the columns you want to display. In column picker, you can use the check boxes against the column names to show or hide the corresponding columns in the grid. You can also use filter box to filter columns from the dropdown list. Additionally, you can further customize the column picker to allow column drag and drop in the Column Picker's dropdown list to change the order of columns displayed in FlexGrid. To see the advanced column picker in action, see FlexGrid column picker product sample.



ID	Name	Country	Hired
0	Mark Myers	Iran	5/16/2021
1	Ed Heath	Egypt	12/7/2021
2	Steve Stevens	Russia	5/7/2021
3	Charlie Heath	United States	3/3/2022
4	Ed Lehman	Germany	6/25/2021
5	Steve Stevens	Indonesia	12/8/2021
6	Karl Paulson	China	8/3/2021
7	Fred Stevens	Vietnam	3/27/2022
8	Dan Cole	Iran	7/18/2021
9	Jack Stevens	Philippines	2/13/2022
10	Mark Heath	Germany	3/26/2022
11	Charlie Danson	Italy	4/16/2021
12	Jack Danson	Myanmar	6/17/2021
13	Charlie Myers	Myanmar	10/4/2021
14	Charlie Myers	Indonesia	6/2/2021
15	Dan Danson	Russia	12/16/2021
16	Andy Danson	India	8/16/2021

You can easily implement a column-picker UI using the grid's **columns** property, a ListBox control, and **showPopup** and **hidePopup** methods as showcased in the following example.

Add Model

Person.cs copyCode

```
public class Person
{
    internal static string[] Countries = "China|India|United
States|Indonesia|Brazil|Nigeria|Russia|Japan|Mexico|Philippines|Vietnam|Germany|Egypt|Iran|France|Thailand|United
Kingdom|Italy|Myanmar".Split('|');

    //ctor
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public Person(int id)
    {
        ID = id;
    }

    //オブジェクトモデル

    public int ID { get; set; }
    public string Name { get { return string.Format("{0} {1}", First, Last); } }
    public string Country { get { return Countries[CountryID]; } }
    public int CountryID { get; set; }
    public bool Active { get; set; }
    public string First { get; set; }
    public string Last { get; set; }
    public DateTime Hired { get; set; }
}

public class SampleData
{
    private static Random _rnd = new Random();
    private static string[] _firstNames =
"Andy|Ben|Charlie|Dan|Ed|Fred|Jack|Karl|Larry|Mark|Noah|Paul|Steve|Ted".Split('|');
    private static string[] _lastNames =
"Ambers|Cole|Danson|Griswold|Heath|Lehman|Myers|Paulson|Richards|Stevens".Split('|');

    private static string GetString(string[] arr)
    {
        return arr[_rnd.Next(arr.Length)];
    }

    public static IEnumerable<Person> GetData()
    {
        var list = Enumerable.Range(0, 50).Select(i =>
        {
            return new Person(i)
            {
                First = GetString(_firstNames),
                Last = GetString(_lastNames),
                CountryID = _rnd.Next() % Person.Countries.Length,
                Active = _rnd.NextDouble() >= .5,
                Hired = DateTime.Today.AddDays(-_rnd.Next(1, 365))
            };
        });
        return list;
    }
}
```

Add Controller

HomeController.cshtml

```
public static List<Person> Persons = SampleData.GetData().ToList();
public ActionResult Index()
{
     return View(Persons);
}
```

Add View for the Controller

Index.razor copyCode

```
@model IEnumerable<Person>

<script>
    c1.documentReady(function () {
        // 列ピッカーのドロップダウンリストを作成します。
        var grid = wijmo.Control.getControl('#gridColumnPicker');
        let theColumnPicker = new wijmo.input.MultiSelectListBox('#theColumnPicker', {
            itemsSource: grid.columns,
            checkedMemberPath: 'visible',
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            displayMemberPath: 'header',
            showFilterInput: true,
            showSelectAllCheckbox: true,
            lostFocus: function () { wijmo.hidePopup(theColumnPicker.hostElement) },
            checkedItemsChanged: function (sender) {
                grid.columns.forEach(function (item) { item.visible = false; });
                sender.checkedItems.forEach(function (item) {
                    grid.columns.getColumn(item.binding).visible = true;
                });
            }
        });
        // 列ピッカーボタンの準備をします。
        var tempBtnColPicker = document.getElementById('column-picker-icon');
        tempBtnColPicker._columnPicker = theColumnPicker;
        tempBtnColPicker._grid = grid;
    });

    function formatItem(panel, r, c, cell) {
        if (panel.cellType == wijmo.grid.CellType.TopLeft) {
            var tempBtnColPicker = document.getElementById('column-picker-icon'),
                btnColPicker = tempBtnColPicker.cloneNode(true),
                theColumnPicker = tempBtnColPicker._columnPicker,
                grid = tempBtnColPicker._grid;

            // ユーザーが左上のセルをクリックしたときに列ピッカーを表示します。
            btnColPicker.addEventListener("click", function (e) {
                let host = theColumnPicker.hostElement;
                if (host.offsetHeight) {
                    wijmo.hidePopup(host, true, true);
                    grid.focus();
                }
                else {
                    wijmo.showPopup(host, e.target, false, true, false);
                    theColumnPicker.focus();
                }
                e.preventDefault();
            });

            btnColPicker.addEventListener("mousedown", function (e) {
                e.preventDefault();
                e.stopPropagation();
            });

            cell.appendChild(btnColPicker);

            if (theColumnPicker.hostElement.style.display != 'none') {
                wijmo.showPopup(theColumnPicker.hostElement, btnColPicker, false, true, false);
            }
        }
    }
</script>

@(Html.C1().FlexGrid<Person>().Id("gridColumnPicker")
                    .IsReadOnly(true)
                    .Bind(Model)
                    .AutoGenerateColumns(false)
                    .CssClass("grid")
                    .Columns(bl =>
                    {
                        bl.Add(cb => cb.Binding("ID").Header("ID").Width("0.4*").IsReadOnly(true));
                        bl.Add(cb => cb.Binding("Name").Header("Name").Width("*").Name("Name"));
                        bl.Add(cb => cb.Binding("Country").Header("Country").Width("*").Name("Country"));
                        bl.Add(cb => cb.Binding("Hired").Header("Hired").Width("*").Name("Hired"));
                    })
                    .ItemFormatter("formatItem")
)

<div style="display:none">
    <img id="column-picker-icon" class="column-picker-icon" src="~/Content/Image/Columns.png" width="20"
```


Copyright © GrapeCity, Inc. All rights reserved.

```
height="20" />
    <div id="theColumnPicker" class="column-picker" style="display:none"></div>
</div>
```

列の固定

FlexGrid allows you to pin columns to the left of the grid so they remain visible as you scroll the grid horizontally. Pinning the columns adds pin glyphs to the column headers and clicking the pin glyph freezes (or unfreezes) the columns. To pin columns in FlexGrid, you can use **PinningType** property of the FlexGrid class. This property accepts value from the **PinningType** enumeration, which are described as follows:

- **None** : This values does not allow pinning columns
- **SingleColumn** : Allows pinning and unpinning one column at a time
- **ColumnRange** : Allows pinning and unpinning column ranges
- **Both** : Allows pinning and unpinning single columns or column ranges. This can also be achieved using the Shift key.

ID	Start	Product	Amount	Discount	Active
1	1/25/2020	Gadget	$581.61	14%	☑
2	2/25/2020	Gadget	$4,919.02	17%	☐
3	3/25/2020	Gadget	$2,159.73	7%	☐
4	4/25/2020	Gadget	$1,248.66	22%	☐
5	5/25/2020	Gadget	$4,051.76	12%	☑
6	6/25/2020	Gadget	($3,131.28)	16%	☐
7	7/25/2020	Widget	$698.62	15%	☐
8	8/25/2020	Widget	$3,464.15	3%	☐
9	9/25/2020	Gadget	($2,363.16)	14%	☑
10	10/25/2020	Widget	($2,836.94)	6%	☐
11	11/25/2020	Widget	$877.93	8%	☐
12	12/25/2020	Widget	($3,788.14)	12%	☐
13	1/25/2020	Gadget	($2,446.92)	12%	☑
14	2/25/2020	Widget	($4,374.97)	18%	☐
15	3/25/2020	Gadget	$1,089.32	16%	☐

To pin a column range in FlexGrid, use the following code. This example uses the sample code used in Quick Start.

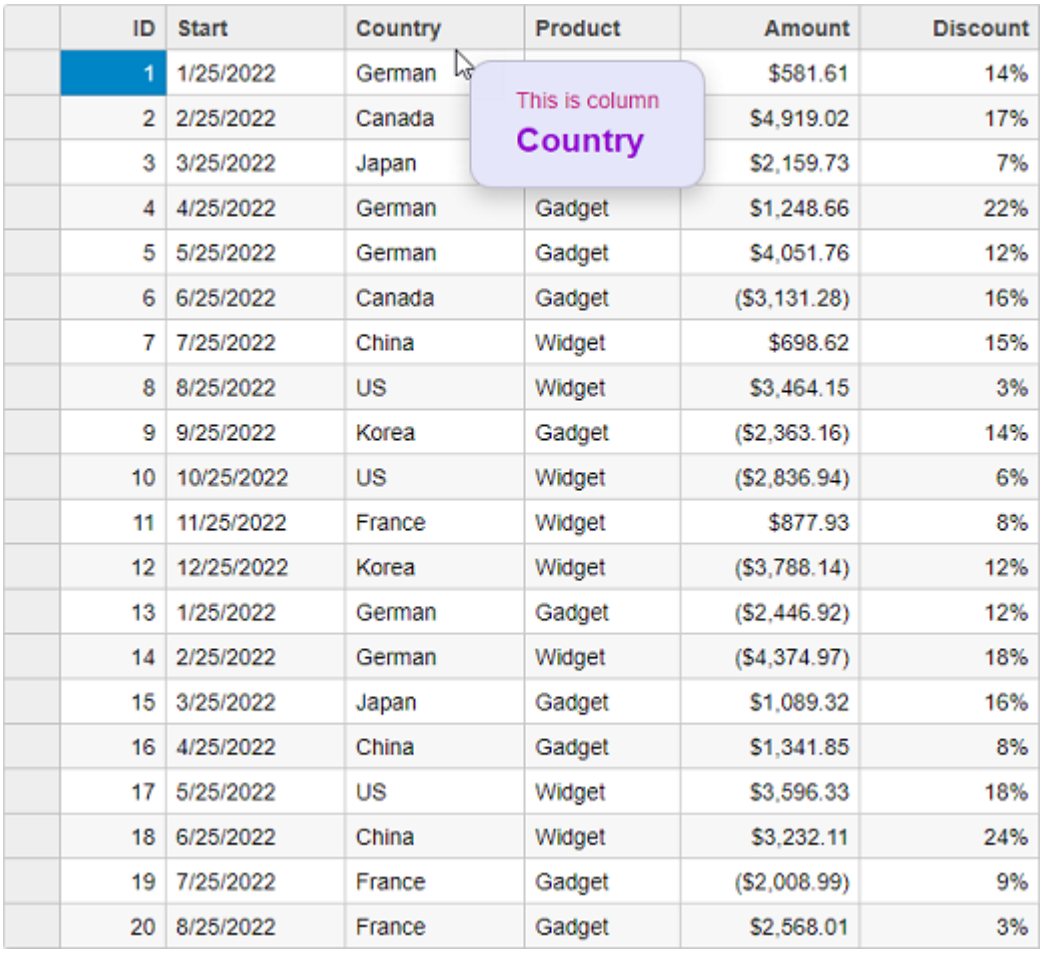
C#

```
.PinningType(PinningType.ColumnRange)
```

ヘッダにツールチップの表示

You can add tooltips to the FlexGrid column headers. FlexGrid can be used in conjunction with Wijmo's Tooltip control to display data when you hover the mouse cursor over column headers in the FlexGrid control.

Copyright © GrapeCity, Inc. All rights reserved.



	ID	Start	Country	Product	Amount	Discount
	1	1/25/2022	German		$581.61	14%
	2	2/25/2022	Canada		$4,919.02	17%
	3	3/25/2022	Japan		$2,159.73	7%
	4	4/25/2022	German	Gadget	$1,248.66	22%
	5	5/25/2022	German	Gadget	$4,051.76	12%
	6	6/25/2022	Canada	Gadget	($3,131.28)	16%
	7	7/25/2022	China	Widget	$698.62	15%
	8	8/25/2022	US	Widget	$3,464.15	3%
	9	9/25/2022	Korea	Gadget	($2,363.16)	14%
	10	10/25/2022	US	Widget	($2,836.94)	6%
	11	11/25/2022	France	Widget	$877.93	8%
	12	12/25/2022	Korea	Widget	($3,788.14)	12%
	13	1/25/2022	German	Gadget	($2,446.92)	12%
	14	2/25/2022	German	Widget	($4,374.97)	18%
	15	3/25/2022	Japan	Gadget	$1,089.32	16%
	16	4/25/2022	China	Gadget	$1,341.85	8%
	17	5/25/2022	US	Widget	$3,596.33	18%
	18	6/25/2022	China	Widget	$3,232.11	24%
	19	7/25/2022	France	Gadget	($2,008.99)	9%
	20	8/25/2022	France	Gadget	$2,568.01	3%

Following example shows how you can bind Tooltips to the column headers of FlexGrid control. The example uses **Sale.cs** model which was added to the application in the Quick Start topic:

Controller code

HeaderTooltipController.cshtml copyCode

```
public ActionResult Index()
{
    var model = Sale.GetData(20);
    return View(model);
}
```

View code

HeaderTooltip.razor copyCode

```
@model IEnumerable<Sale>

<style>
    .hdr-tip {
        background: lavender;
        color: mediumvioletred;
        padding: 1em 2em;
        margin: .5em;
        border-radius: 1em;
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .hdr-tip .col-header {
            color: darkviolet;
            font-weight: bold;
            font-size: 150%;
        }
</style>

<script type="text/javascript">
    var hdrTips;
    c1.documentReady(function () {
        // 列ヘッダにツールチップを表示します。
        hdrTips = new wijmo.Tooltip({
            position: wijmo.PopupPosition.RightTop,
            showAtMouse: true,
            showDelay: 600,
            cssClass: 'hdr-tip'
        });
    });

    function formatItem(panel, r, c, cell) {
        if (panel.cellType == wijmo.grid.CellType.ColumnHeader) {
            hdrTips.setTooltip(cell, 'This is column<br/>' +
                '<span class="col-header">' + panel.columns[c].header + '</span>');
        }
    }
    function loadingRows() {
        if (hdrTips) {
            hdrTips.dispose();
        }
    }
</script>

@(Html.C1().FlexGrid<Sale>()
            .Id("gridHeaderTooltips")
            .AutoGenerateColumns(false)
            .CssClass("grid")
            .IsReadOnly(true)
            .Bind(Model)
            //FlexGridに列データを連結します。
            .Columns(bl =>
            {
                bl.Add(cb => cb.Binding("ID"));
                bl.Add(cb => cb.Binding("Start"));
                bl.Add(cb => cb.Binding("Country"));
                bl.Add(cb => cb.Binding("Product"));
                bl.Add(cb => cb.Binding("Amount").Format("c"));
                bl.Add(cb => cb.Binding("Discount").Format("p0"));
            })
            .ItemFormatter("formatItem")
            .OnClientLoadingRows("loadingRows")
)
```


Copyright © GrapeCity, Inc. All rights reserved.

列レイアウトの保存とロード

FlexGridはクライアントの**ColumnLayout**プロパティを備えており、これによって列レイアウトをブラウザのlocalStorageに保存します。**ColumnLayout**は、グリッド列とそのプロパティの一覧を含むJSON文字列を取得または設定します。FlexGridでは、datamap列はサポートされていません。

この動作を確認するには、次の操作を実行します。

1. いくつかの列をサイズ変更し、新しい位置にドラッグします。
2. [列レイアウトの保存]ボタンをクリックし、変更をローカルストレージに保存します。
3. ページをリフレッシュして元のレイアウトを復元します。
4. [列レイアウトのロード]ボタンをクリックしてレイアウトをローカルストレージから復元します。

FlexGridは、サーバー側での列レイアウトの保存/ロードをサポートします。これは、ユーザーごとに異なるビューを表示するために列レイアウトをデータベースに保存する必要がある場合に便利です。

次の図は、**ColumnLayout**プロパティを使用してFlexGridのレイアウトを保存したときのFlexGridを示しています。この例では、「クイックスタート」セクションで追加された**Sale.cs**モデルをデータソースとして使用します。

列のレイアウトを保存　列のレイアウトをロード

	ID	開始日	終了日	国名	製品名	色
	1	2015/01/25	2015/01/25	ドイツ	Gadget	緑色
	2	2015/02/25	2015/02/25	カナダ	Gadget	緑色
	3	2015/03/25	2015/03/25	日本	Gadget	赤色
	4	2015/04/25	2015/04/25	ドイツ	Gadget	赤色
	5	2015/05/25	2015/05/25	ドイツ	Gadget	黒色
	6	2015/06/25	2015/06/25	カナダ	Gadget	黒色
	7	2015/07/25	2015/07/25	中国	Widget	赤色
	8	2015/08/25	2015/08/25	米国	Widget	白色
	9	2015/09/25	2015/09/25	韓国	Gadget	黒色
	10	2015/10/25	2015/10/25	米国	Widget	白色
	11	2015/11/25	2015/11/25	フランス	Widget	緑色
	12	2015/12/25	2015/12/25	韓国	Widget	赤色
	13	2015/01/25	2015/01/25	ドイツ	Gadget	赤色
	14	2015/02/25	2015/02/25	ドイツ	Widget	黒色
	15	2015/03/25	2015/03/25	日本	Gadget	緑色
	16	2015/04/25	2015/04/25	中国	Gadget	黒色
	17	2015/05/25	2015/05/25	米国	Widget	黒色

次のコード例は、FlexGridで列レイアウトの保存とロードを設定する方法を示します。

コードの場合

Copyright © GrapeCity, Inc. All rights reserved.

SaveColumnLayoutController.cs

C# copyCode

```
public ActionResult Index()
{
    var model = Sale.GetData(500);
    return View(model);
}
```

SaveColumnLayout.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>

<script>
    // localStorageに列のレイアウトを保存または復元します。
    function saveColumnLayout() {
        var grid = wijmo.Control.getControl("#ovFlexGrid");
        localStorage['columns'] = grid.columnLayout;
    }
    function loadColumnLayout() {
        var grid = wijmo.Control.getControl("#ovFlexGrid"),
            columnLayout = localStorage['columns'];
        if (columnLayout) {
            grid.columnLayout = columnLayout;
        }
    }
</script>
<input type="button" value="列のレイアウトを保存" class="btn" onclick="saveColumnLayout()"
/>
<input type="button" value="列のレイアウトをロード" class="btn"
onclick="loadColumnLayout()" />

@(Html.C1().FlexGrid<Sale>()
    .Id("ovFlexGrid")
    .AutoGenerateColumns(true)
    .AllowSorting(true)
    .SelectionMode(SelectionMode.Cell)
    .Bind(bl => bl.InitialItemsCount(100).Bind(Model))
    .CssClass("grid")
    .IsReadOnly(true)
)
```

クライアント側でグリッド列を表示/非表示

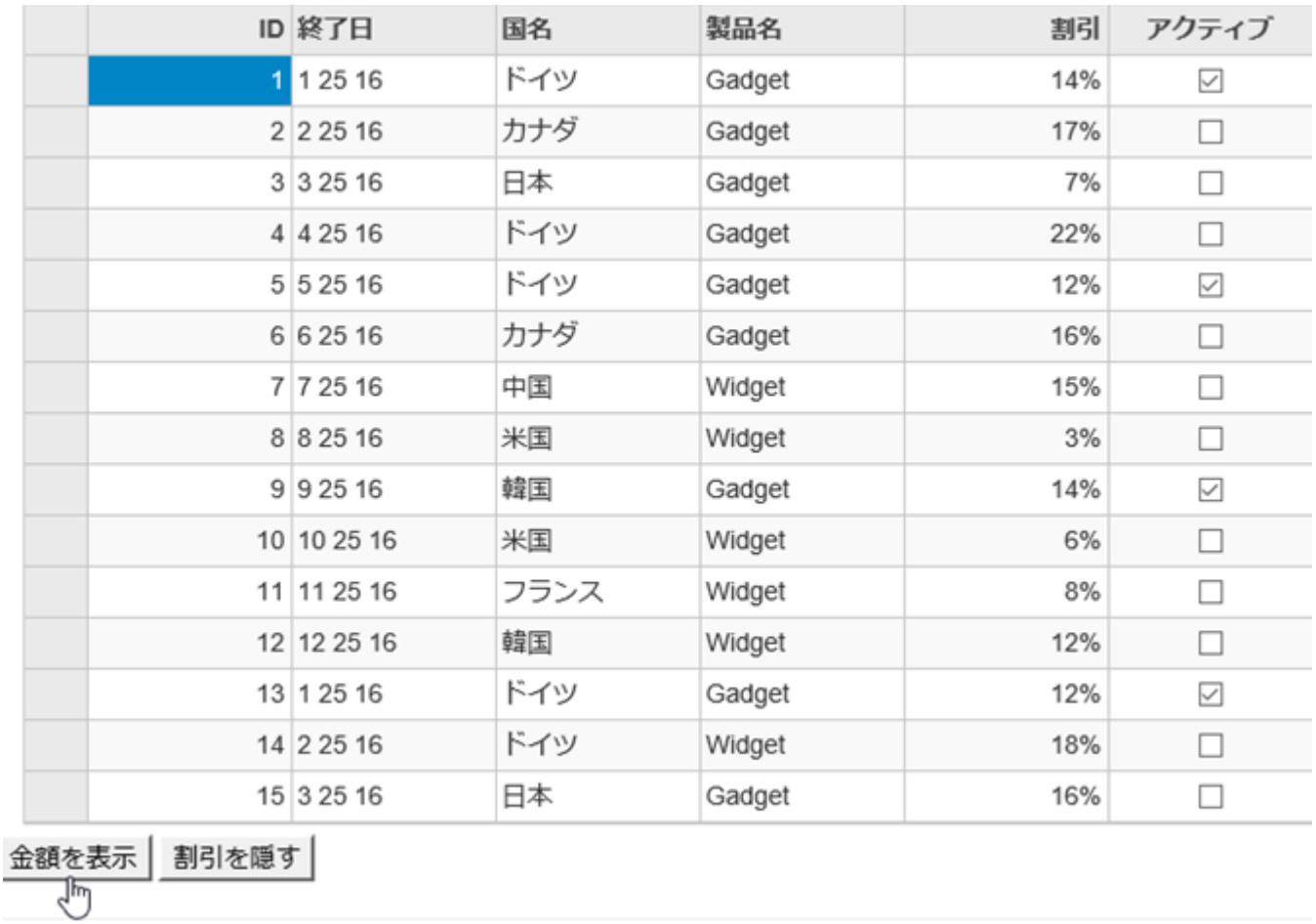
FlexGridコントロールでは、サーバー側で、FlexGridの**Visible**プロパティを使用して特定のグリッド列を表示または非表示にすることができます。しかし、この操作は、クライアント側で実行することもできます。MVCアプリケーションで、グリッドの非表示

Copyright © GrapeCity, Inc. All rights reserved.

の列をエンドユーザーのボタンクリックで表示できた方がよい場合もあります。

たとえば、FlexGridを使用して会社の売上データを分析する場合、最初の段階では、限られた列だけを表示し、月ごとの売上高などの追加情報は、必要なときにのみ表示するようにしたいとします。JavaScriptを使用すると、これをクライアント側で実現できます。

次の例は、このシナリオを示しています。この例では、ボタンクリックでAmount列を表示できます。



Copyright © GrapeCity, Inc. All rights reserved.



同様に、**[Hide Discount]**ボタンのクリックで、グリッドからDiscount列を非表示にできます。

次に示すように、MVCアプリケーションのビューでJavaScriptを使用して、ボタンクリックで任意の列を表示/非表示にできます。

以下のコードは、「クイックスタート」トピックで作成したアプリケーションを拡張しています。

Razor

```
<script>
function showAmount() {
                var grid = wijmo.Control.getControl("#hFlexGrid");
                //列インデックスによって
                //grid.columns[6].visible = true;
                //または、列の名によって
                grid.columns.getColumn("Amount").visible = true;
                }
                </script>
<script>
function hideDiscount() {
                var grid = wijmo.Control.getControl("#hFlexGrid");
                //列インデックスによって
                //grid.columns[4].visible = false;
                //または、列の名によって
                grid.columns.getColumn("Discount").visible = false;
                }
</script>
<div class="container">
<div>
@(Html.C1().FlexGrid<Sale>().Id("hFlexGrid")
                .AutoGenerateColumns(false)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                .Bind(Model)
                .Columns(bl =>{
                bl.Add(cb => cb.Binding("ID").Header("ID"));
                bl.Add(cb => cb.Binding("End").Header("End").Format("MMM d yy"));
                bl.Add(cb => cb.Binding("Country").Header("Country"));
                bl.Add(cb => cb.Binding("Product").Header("Product"));
                bl.Add(cb => cb.Binding("Discount").Format("p0"));
                bl.Add(cb => cb.Binding("Active"));
                bl.Add(cb => cb.Binding("Amount").Format("n0").Visible(false));
                })
                )
</div>
</div>
                <button title="SetValue" onclick="showAmount()">Show Amount</button>
        <button title="SetValue" onclick="hideDiscount()">Hide Discount</button>
```

先頭に戻る

非連結列

FlexGridコントロールでは、特定の列のみを連結することができます。グリッドでは連結列および非連結列の両方が含まれるように、コントロールがカスタマイズできます。連結列ではデータソースから取得されているデータが表示でき、非連結列ではカスタムデータが表示できます。このカスタマ化を実現するためには、itemFormatter関数を使用します。

以下の例では、データソースとの連結列のセル値を使用して、非連結列のセルに表示する値の計算方法を示します。以下のコード例では、itemFormatter関数を使用してFlexGrid のセルをカスタマイズするために、リッチなクライアント側アセンブリを使用します。

次の図は、非連結列「対象値」を含む FlexGrid を示していますが、列値は「価格」列の値を100で割られて計算されています。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	国名	製品名	金額	対象値
	◢国名: **ドイツ** (5 項目)			-¥940	-9.40
	1	ドイツ	Gadget	¥582	5.82
	4	ドイツ	Gadget	¥1,249	12.49
	5	ドイツ	Gadget	¥4,052	40.52
	13	ドイツ	Gadget	-¥2,447	-24.47
	14	ドイツ	Widget	-¥4,375	-43.75
	◢国名: **カナダ** (2 項目)			¥1,788	17.88
	2	カナダ	Gadget	¥4,919	49.19
	6	カナダ	Gadget	-¥3,131	-31.31
	◢国名: **日本** (2 項目)			¥3,249	32.49
	3	日本	Gadget	¥2,160	21.60
	15	日本	Gadget	¥1,089	10.89
	◢国名: **中国** (1 項目)			¥699	6.99
	7	中国	Widget	¥699	6.99
	◢国名: **米国** (2 項目)			¥627	6.27
	8	米国	Widget	¥3,464	34.64
	10	米国	Widget	-¥2,837	-28.37
	◢国名: **韓国** (2 項目)			-¥6,151	-61.51
	9	韓国	Gadget	-¥2,363	-23.63

次のコード例は、FlexGrid コントロールに非連結列を追加して、計算された値を列のセルに割り当てる方法を示します。この例では、「クイックスタート」でアプリケーションに追加した **Sale.cs** モデルを使用します。

コードの場合

UnboundcolumnController.cs

C# copyCode

```
    public ActionResult Index()
    {
        return View(Sale.GetData(15));
    }
}
```

Unboundcolumn.cshtml

以下のように、MVCへの参照を追加します。

C# copyCode

```
@using MVCFlexGrid_JP.Models
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>
```

非連結列のセルをカスタマイズするためにitemFormatter関数用のクライアントスクリプトを追加します。

JavaScript copyCode

```
<script>
    function itemFormatter(panel, r, c, cell) {
        var flex = panel.grid;
        var col = panel.columns[c];
        //計算された値を表示している列を非連結します。
        if (col.name == "TargetValue") {
            var calculatedData = parseFloat(flex.getCellData(r, 3, false) / 100);
            flex.setCellData(r, col.index, calculatedData, true);
        } //計算した値を表示する非連結列
    }
</script>
```

次に、以下のコードを追加して FlexGrid コントロールを初期化します。

Razor copyCode

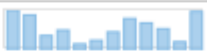
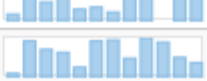
```
@(Html.C1().FlexGrid<Sale>().Id("Sales")
.AutoGenerateColumns(false)
.Bind(Model)
.AllowResizing(AllowResizing.None)
.ItemFormatter("itemFormatter")
.GroupBy("国名")
.Columns(columns =>
    {
        columns.Add(column => column.Binding("ID"));
        columns.Add(column => column.Binding("国名"));
        columns.Add(column => column.Binding("製品名"));
        columns.Add(column => column.Binding("金
額").Format("c").Width("*").Align("center").Aggregate(Aggregate.Sum));
        columns.Add(column => column.Header("対象値").Name("TargetValue"));
    }))
```

カスタムセルテンプレート

FlexGrid には、セルのコンテンツを完全に制御できる **itemFormatter** プロパティがあります。AngularJS ディレクティブは、インラインセルテンプレートをサポートします。これにより、プレーン HTML を使用してセルの外観を定義できます。列にセルテンプレートを定義するには、各セルに表示する HTML を列定義に追加します。

次の図は、トレンドを表すためのセルテンプレートとして **FlexChart** を設定し、ランクを表すためのテンプレートとして **Stars** を設定した FlexGrid を示しています。この例では、「クイックスタート」セクションで追加した Sale.cs モデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	国名	製品名	傾向	ランク
	1	ドイツ	Gadget		★★★★☆
	2	カナダ	Gadget		★★☆☆☆
	3	日本	Gadget		★★★★★
	4	ドイツ	Gadget		★☆☆☆☆
	5	ドイツ	Gadget		★★☆☆☆
	6	カナダ	Gadget		★★★★★
	7	中国	Widget		★★★☆☆
	8	米国	Widget		★★☆☆☆
	9	韓国	Gadget		★★☆☆☆
	10	米国	Widget		★☆☆☆☆
	11	フランス	Widget		★★☆☆☆
	12	韓国	Widget		★★★★★
	13	ドイツ	Gadget		★☆☆☆☆
	14	ドイツ	Widget		★★★★☆
	15	日本	Gadget		★★★★★

次のコード例は、FlexGrid の列の1つにカスタムセルテンプレートを設定する方法を示します。

コードの場合

CustomCellTemplateController.cs

C# copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(15));
}
```

CustomCellTemplate.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@model IEnumerable<Sale>

<script id="template1" type="text/html">
    @(Html.C1().FlexChart()
        .Width(100).Height(20).CssStyle("padding", "0")
        .TemplateBind("ItemsSource", "傾向")
        .BindingX("月")
        .Series(s => s.Add().Binding("Data"))
        .AxisX(C1.Web.Mvc.Chart.Position.None)
        .AxisY(C1.Web.Mvc.Chart.Position.None)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .ToTemplate()
    )
</script>

<style>
    .star-highlighted {
        color: orange;
    }

    .star-dimmed {
        color: lightgray;
    }
</style>

<script type="text/javascript">
    function rankFormatter(panel, r, c, cell) {
        if (panel.columns[c].binding === "ランク") {
            cell.style.textAlign = "";
            if (panel.cellType === wijmo.grid.CellType.Cell) {
                cell.innerHTML = buildRank(panel.getCellData(r, c));
            }
        }
    }

    function buildRank(rank) {
        var markup = "", j, starType;
        for (j = 0; j < 5; j++) {
            starType = j < rank ? "star star-highlighted" : "star star-dimmed";
            markup += "<span class='" + starType + "'>\u2605</span>";
        }
        return markup;
    }
</script>

@(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .IsReadOnly(true)
    .Bind(Model)
    .CssClass("grid")
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("ID"));
        columns.Add(column => column.Binding("国名"));
        columns.Add(column => column.Binding("製品名"));
        columns.Add(column => column.Header("傾向").CellTemplate(b =>
b.Template("template1")));
        columns.Add(column => column.Binding("ランク"));
    })
    .ItemFormatter("rankFormatter")
)
```


Copyright © GrapeCity, Inc. All rights reserved.

テンプレート関数

FlexGrid supports template functions in columns which allows you to call functions at runtime to create custom cells. It can be used to create several types of simple and useful cells, such as button, hyperlink, image, rating and sparkline columns using the client side CellMaker. For example, you can create a rating cell by creating a FlexGrid column and assign a template-function name of a client side function that uses the CellMaker.makeRating client side method.



ID	Country	Product	Amount	Trends	Rank	CellMaker Button	CellMaker Hyperlink	CellMake
1	German	Gadget	581.61		★★★★☆	German Button	Visit German	
2	Italy	Gadget	-4,673.75		★★☆☆☆	Italy Button	Visit Italy	
3	China	Gadget	-2,265.49		★★★★★	China Button	Visit China	
4	France	Widget	3,964.40		★☆☆☆☆	France Button	Visit France	
5	UK	Widget	-1,744.99		★☆☆☆☆	UK Button	Visit UK	
6	France	Gadget	4,276.37		★★☆☆☆	France Button	Visit France	
7	US	Widget	-4,376.92		★☆☆☆☆	US Button	Visit US	
8	Japan	Gadget	1,996.52		★★★★☆	Japan Button	Visit Japan	
9	Korea	Widget	-3,442.35		★★★☆☆	Korea Button	Visit Korea	
10	US	Widget	-2,973.96		★★★★★	US Button	Visit US	
11	Canada	Widget	-3,217.79		★★☆☆☆	Canada Button	Visit Canada	
12	UK	Gadget	1,556.19		★★☆☆☆	UK Button	Visit UK	
13	Canada	Gadget	-4,981.75		★★★★★	Canada Button	Visit Canada	
14	China	Gadget	-397.56		★★★★☆	China Button	Visit China	
15	Italy	Widget	199.60		★★★★☆	Italy Button	Visit Italy	

The following example showcases the creation of all the types of custom cells in FlexGrid columns, which includes button, hyperlink, image, rating and sparkline. The example uses **Sale.cs** model added in the Quick Start Section.

Controller

CellMarkerController.cs copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(50));
}
```

View for the Controller

Index.cshtml copyCode

```
@model IEnumerable<Sale>

@section Scripts{
    <script type="text/javascript">
        function rankFormatter(panel, r, c, cell) {
            if (panel.columns[c].header === "Rank") {
                cell.style.textAlign = "";
                if (panel.cellType === wijmo.grid.CellType.Cell) {
                    cell.innerHTML = buildRank(panel.getCellData(r, c));
                }
            }
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        function buildRank(rank) {
            var markup = "", j, starType;
            for (j = 0; j < 5; j++) {
                starType = j < rank ? "star star-highlighted" : "star star-dimmed";
                markup += "<span class='" + starType + "'>\u2605</span>";
            }
            return markup;
        }

        function createButton(CellMaker) {
            return CellMaker.makeButton({
                text: '<b>${item.Country}</b> Button',
                click: function (e, ctx) {
                    alert('Clicked Button ** ' + ctx.item.Country + ' **');
                }
            });
        }

        function createHyperlink(CellMaker) {
            return CellMaker.makeLink({
                text: 'Visit <b>${item.Country}</b>',
                href: '${item.Url}',
                attributes: {
                    target: '_blank',
                    rel: 'noopener noreferrer',
                    tabIndex: -1
                }
            });
        }

        function createImage(CellMaker) {
            return CellMaker.makeImage({
                label: 'image for ${item.Country}',
                click: function (e, ctx) {
                    alert('Clicked image for ' + ctx.item.Country);
                }
            });
        }

        function createRating(CellMaker) {
            return CellMaker.makeRating({
                range: [0, 5],
                label: 'Edit Product Rating'
            });
        }

        function createSparkline(CellMaker) {
            return CellMaker.makeSparkline({
                markers: wijmo.grid.cellmaker.SparklineMarkers.High |
wijmo.grid.cellmaker.SparklineMarkers.Low,
                label: '${item.Country} sales history line chart',
            });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
    </script>
}

@section Styles{
    <style>
        .star-highlighted {
            color: orange;
        }

        .star-dimmed {
            color: lightgray;
        }

        .wj-cell.wj-cell-maker label {
            padding: 0;
        }
        /* 内部IMG要素ではなく、セルに書式設定を適用する画像 */
        .wj-flexgrid .wj-cell.cell-img {
            padding: 0;
            text-align: center;
        }

            .wj-flexgrid .wj-cell.cell-img:hover {
                z-index: 1;
                overflow: visible;
            }

                .wj-flexgrid .wj-cell.cell-img:hover img {
                    transform: scale(3);
                    transition: all 600ms;
                }

        .wj-flexgrid .wj-cell img.wj-cell-maker {
            width: auto;
        }

        /* 評価記号/色/サイズをカスタマイズします */
        .wj-flexgrid .wj-cell.wj-cell-maker.wj-radio-map.custom-rating label {
            width: .25em;
            color: green;
        }

            .wj-flexgrid .wj-cell.wj-cell-maker.wj-radio-map.custom-rating
label:after {
                transform: scale(12);
                content: '\2b24';
            }
    </style>
}

<script id="template1" type="text/template">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    @(Html.C1().FlexChart()
        .Width(100).Height(20).CssStyle("padding", "0")
        .TemplateBind("ItemsSource", "Trends")
        .BindingX("Month")
        .Series(s => s.Add().Binding("Data"))
        .AxisX(C1.Web.Mvc.Chart.Position.None)
        .AxisY(C1.Web.Mvc.Chart.Position.None)
        .ToTemplate()
    )
</script>

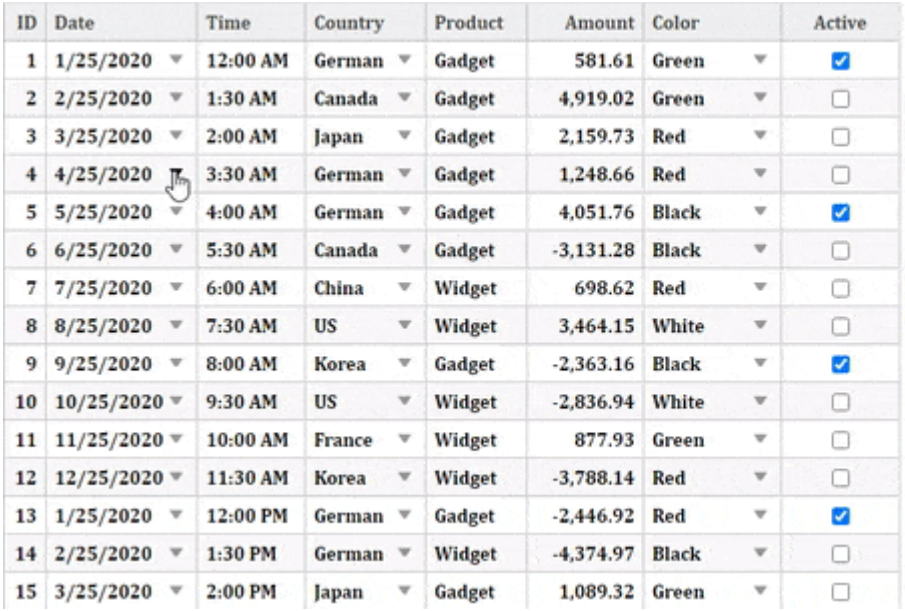

@(Html.C1().FlexGrid<Sale>()
    .Id("customCellFlexGrid")
    .AutoGenerateColumns(false)
    .IsReadOnly(false)
    .Bind(Model)
    .CssClass("grid")
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("ID").IsReadOnly(true).Width("50"));
        columns.Add(column => column.Binding("Country").Width("80"));
        columns.Add(column => column.Binding("Product").Width("80"));
        columns.Add(column => column.Binding("Amount").Width("100").Template("<span
class=${item.Amount > 1000 ? 'bg-info' : 'bg-danger'}>${text}</span>"));
        columns.Add(column =>
column.Header("Trends").IsReadOnly(true).Align("center").CellTemplate(b =>
b.TemplateId("template1")));
        columns.Add(column =>
column.Binding("Rank").IsReadOnly(true).Width("90").Align("center"));
        columns.Add(column => column.Binding("Country").Header("CellMaker
Button").Width("150").Align("center").TemplateFunction("createButton"));
        columns.Add(column => column.Binding("Country").Header("CellMaker
Hyperlink").Width("180").TemplateFunction("createHyperlink"));
        columns.Add(column => column.Binding("Img").Header("CellMaker
Image").Width("140").CssClass("cell-img").TemplateFunction("createImage"));
        columns.Add(column => column.Binding("Rank").Header("CellMaker
Rating").Width("150").Align("center").CssClass("custom-
rating").TemplateFunction("createRating"));
        columns.Add(column => column.Binding("History").Header("CellMaker
Sparkline").Width("180").TemplateFunction("createSparkline"));
    })
    .ItemFormatter("rankFormatter")
)
```

カスタムエディタ

FlexGrid provides in-built support for Excel like fast editing, however, in certain cases you may need advance editing experience. To enable such editing, you can customize your editing experience by using custom editors and specify custom editor controls using the **Editor** property on the Column. The **Editor** property simply sets a reference to an MVC input control to be used as a custom cell editor. For this, you can use any input control that you want, including Date Picker, Numeric Textbox, AutoComplete, Color Picker, and more.

Copyright © GrapeCity, Inc. All rights reserved.

The following example demonstrates how you can use MVC InputDate, InputTime, ComboBox, InputNumber, and InputColor controls as custom editors. To use an input control as a custom editor, all you have to do is associate an instance of the control with a grid column or a style using its **Editor** property. This example also showcases how **CellTemplate** can be used for customizing the editors further in some of the columns.



ID	Date	Time	Country	Product	Amount	Color	Active
1	1/25/2020	12:00 AM	German	Gadget	581.61	Green	☑
2	2/25/2020	1:30 AM	Canada	Gadget	4,919.02	Green	☐
3	3/25/2020	2:00 AM	Japan	Gadget	2,159.73	Red	☐
4	4/25/2020	3:30 AM	German	Gadget	1,248.66	Red	☐
5	5/25/2020	4:00 AM	German	Gadget	4,051.76	Black	☑
6	6/25/2020	5:30 AM	Canada	Gadget	-3,131.28	Black	☐
7	7/25/2020	6:00 AM	China	Widget	698.62	Red	☐
8	8/25/2020	7:30 AM	US	Widget	3,464.15	White	☐
9	9/25/2020	8:00 AM	Korea	Gadget	-2,363.16	Black	☑
10	10/25/2020	9:30 AM	US	Widget	-2,836.94	White	☐
11	11/25/2020	10:00 AM	France	Widget	877.93	Green	☐
12	12/25/2020	11:30 AM	Korea	Widget	-3,788.14	Red	☐
13	1/25/2020	12:00 PM	German	Gadget	-2,446.92	Red	☑
14	2/25/2020	1:30 PM	German	Widget	-4,374.97	Black	☐
15	3/25/2020	2:00 PM	Japan	Gadget	1,089.32	Green	☐

To accomplish this, follow these steps:

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Create a Datasource for FlexGrid

1. Add a new class to the folder **Models** (for example: `Sale.cs`). See Adding controls to know how to add a new model.
2. Replace the following code in the new model to define the classes that serve as a datasource for the FlexGrid control.

Sale.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace FlexGrid_MVC.Models
{
    public class Sale
    {
        public int ID { get; set; }
        public DateTime Start { get; set; }
        public DateTime End { get; set; }
        public string Country { get; set; }
        public string Product { get; set; }
        public string Color { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public double Amount { get; set; }
        public double Amount2 { get; set; }
        public double Discount { get; set; }
        public bool Active { get; set; }

        public MonthData[] Trends { get; set; }
        public int Rank { get; set; }

        private static List<string> COUNTRIES = new List<string> { "US", "UK",
"Canada", "Japan", "China", "France", "German", "Italy", "Korea", "Australia" };
        private static List<string> PRODUCTS = new List<string> { "Widget",
"Gadget", "Doohickey" };
        private static List<string> COLORS = new List<string> { "Black",
"White", "Red", "Green", "Blue" };
        /// <summary>
        /// Get the data.
        /// </summary>
        /// <param name="total"></param>
        /// <returns></returns>
        public static IEnumerable<Sale> GetData(int total)
        {
            var rand = new Random(0);
            var dt = DateTime.Now;
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var color = COLORS[rand.Next(0, COLORS.Count - 1)];
                var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
                var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, (i %
2) * 30, 0);

                return new Sale
                {
                    ID = i + 1,
                    Start = startDate,
                    End = endDate,
                    Country = country,
                    Product = product,
                    Color = color,
                    Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Amount2 = Math.Round(rand.NextDouble() * 5000, 2),
                    Discount = Math.Round(rand.NextDouble() / 4, 2),
                    Active = (i % 4 == 0),
                    Trends = Enumerable.Range(0, 12).Select(x => new MonthData {
Month = x + 1, Data = rand.Next(0, 100) }).ToArray(),
                    Rank = rand.Next(1, 6)
                };
            });
            return list;
        }
        public static List<string> GetCountries()
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }

        public static List<string> GetProducts()
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
    }


    public class MonthData
    {
        public int Month { get; set; }
        public double Data { get; set; }
    }
}
```

Back to Top

Add Custom Editors to the FlexGrid control

Complete the following steps to initialize a FlexGrid control.

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `CustomEditorController`).
    3. Click **Add**.
4. Replace the method Index() with the following method.

IndexController.cs

C# copyCode

```
using FlexGrid_MVC.Models;

namespace FlexGrid_MVC.Controllers
{
    public class CustomEditorController : Controller
    {
        // GET: CustomEditor
        public ActionResult Index()
        {
            return View(Sale.GetData(15));
        }
    }
}
```


Copyright © GrapeCity, Inc. All rights reserved.

Note: Replace **FlexGrid_MVC.Models;** with **<*YourMVCApplicationName*>.Models;** in the references.

Add a View for the controller:

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the controller (for example: `Default1Controller`) to open it.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add**. A view has been added for the controller.
6. In the Solution Explorer, double click `Index.cshtml` to open it.
7. Replace the default code of the **Views\Index.cshtml** file with the one given below to initialize a **FlexGrid** control.

Razor copyCode

```
@using FlexGrid_MVC.Models
@model IEnumerable<Sale>

@{
    List<string> countries = Sale.GetCountries();
    List<string> products = Sale.GetProducts();
}

<style type="text/css">
    .grid {
        height: 500px;
        border: 2px solid #e0e0e0;
        font-family: Cambria;
        font-weight: bold;
    }
</style>

<script id="edtDate" type="text/template">
    @(Html.C1().InputDate()
        .Id("dateEditor")
        .Format("d")
        .IsRequired(false) // add this for new row
        .CssStyle("width", "100%") // full with the cell
        .TemplateBind("Value", "Start")
        .ToTemplate()
    )
</script>
<script id="edtTime" type="text/template">
    @(Html.C1().InputTime()
        .Id("timeEditor")
        .Step(30)
        .Format("t")
        .IsRequired(false) // add this for new row
        .CssStyle("width", "100%") // full with the cell
        .TemplateBind("Value", "End").ToTemplate()
    )
</script>
<script id="edtAmount" type="text/template">
    @(Html.C1().InputNumber()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .Id("amountEditor")
        .Format("c2")
        .IsRequired(false) // add this for new row
        .CssStyle("width", "100%") // full with the cell
        .Step(10)
        .TemplateBind("Value", "Amount").ToTemplate()
    )
</script>
<script id="edtCountry" type="text/template">
    @(Html.C1().ComboBox()
        .Id("countryEditor")
        .IsEditable(false)
        .Bind(countries)
        .CssStyle("width", "100%") // full with the cell
        .TemplateBind("Text", "Country").ToTemplate()
    )
</script>
<script id="edtProduct" type="text/template">
    <input type="text" id="{{uid}}" onfocus="productEditorFocus(event)"
onblur="productEditorBlur(event)" style="width:100%;height:100%" value="
{{Product}}" />
</script>
<script id="edtColor" type="text/template">
    @(Html.C1().InputColor()
        .Id("colorEditor")
        .CssStyle("width", "100%") // full with the cell
        .TemplateBind("Text", "Color").ToTemplate()
    )
</script>

@(Html.C1().InputDate()
        .Id("dateEditor1")
        .Format("d")
        .IsRequired(false) // add this for new row
        .CssStyle("width", "100%") // full with the cell
)
@(Html.C1().ComboBox()
        .Id("countryEditor1")
        .IsEditable(false)
        .Bind(countries)
        .CssStyle("width", "100%") // full with the cell
)
@(Html.C1().InputColor()
        .Id("colorEditor1")
        .CssStyle("width", "100%") // full with the cell
)

@*Instantiate FlexGrid and set its properties*@
@(Html.C1().FlexGrid<Sale>()
            .AutoGenerateColumns(false)
            .Width(700)
            .AllowAddNew(true)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
            .CssClass("grid")
            .Bind(Model)

            //Binding columns data to FlexGrid
            .Columns(bl =>
            {
                  bl.Add(cb => cb.Binding("ID").Width("0.4*"));
                  bl.Add(cb =>
cb.Binding("Start").Header("Date").Format("d").Editor("dateEditor1"));
                  bl.Add(cb =>
cb.Binding("End").Header("Time").Format("t").CellTemplate(ctb =>
ctb.EditTemplateId("edtTime")));
                  bl.Add(cb => cb.Binding("Country").Editor("countryEditor1"));
                  bl.Add(cb => cb.Binding("Product").CellTemplate(ctb =>
ctb.EditTemplateId("edtProduct")));
                  bl.Add(cb =>
cb.Binding("Amount").Format("c").Format("n2").Width("1.0*").CellTemplate(ctb =>
ctb.EditTemplateId("edtAmount")));
                  bl.Add(cb => cb.Binding("Color").Editor("colorEditor1"));
                  bl.Add(cb => cb.Binding("Active"));
            }))
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

Back to Top

データ連結

FlexGridとその機能を使用する前に、FlexGridにデータを連結する必要があります。FlexGridにデータを連結したら、フィルタ処理、ソート、選択モード、詳細行などのさまざまな機能を使用できます。

このセクションには、次の情報が含まれます。

リモートデータ連結
　C1JSONRequestを使用して、リモートでFlexGridにデータを連結する方法を学習します。
モデル連結
　基本のモデル連結を使用して、FlexGridにデータを追加する方法を学習します。
AJAX 連結
　基本のモデル連結を使用して、FlexGridにデータを追加する方法を学習します。
OData 連結
　基本のOData連結を使用して、FlexGridにデータを追加する方法を学習します。
FlexGridのマスター/詳細
　FlexGridをカスタマイズして、マスター/詳細データを表示する方法を学習します。

AJAX連結

Copyright © GrapeCity, Inc. All rights reserved.

FlexGridでは、複数のデータ連結オプションが提供されています。AJAXを使用すると、バックグラウンドでデータをロードしてWebページに表示することができ、Webページ全体を再ロードする必要がありません。スムーズなユーザーエクスペリエンスを提供しながら、同時にFlexGridでページング、ソート、フィルタ処理などの機能を使用できます。ここでは、AJAX呼び出しを使用してクライアント側でFlexGridを連結する方法を説明します。

このトピックは次の手順で構成されます。

- **手順1：モデルの作成**
- **手順2：コントローラーアクションの追加**
- **手順3：ビューへのFlexGridの追加**
- **手順4：JavaScriptを使用したAJAX呼び出し**
- **手順5：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexGridを示しています。

GetData

	ID	Product	Country	Amount
	1	Gadget	German	581.61
	2	Gadget	Canada	4,919.02
	3	Gadget	Japan	2,159.73
	4	Gadget	German	1,248.66
	5	Gadget	German	4,051.76
	6	Gadget	Canada	-3,131.28
	7	Widget	China	698.62
	8	Widget	US	3,464.15
	9	Gadget	Korea	-2,363.16
	10	Widget	US	-2,836.94

先頭に戻る

手順1：モデルの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Sale.cs`）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. `Sale.cs`モデル内の次のコードを置き換えます。ここでは、**Sale**クラスを使用して、データベース内の受注データを表しています。**Sale**オブジェクトの各インスタンスがFlexGridコントロール内の1つのレコードに対応します。

Sale.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using MVCFlexGrid_JP.Models;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
namespace MVCFlexGrid_JP.Models
{
    public class Sale
    {
        public int ID { get; set; }
    public DateTime 開始日 { get; set; }
    public DateTime 終了日 { get; set; }
    public string 国名 { get; set; }
    public string 製品名 { get; set; }
    public string 色 { get; set; }
    public double 金額 { get; set; }
    public double 金額2 { get; set; }
    public double 割引 { get; set; }
    public bool アクティブ { get; set; }

    public MonthData[] 傾向 { get; set; }
    public int ランク { get; set; }

    /// <summary>
    /// データを取得
    /// </summary>
    /// <param name="total"></param>
    /// <returns></returns>
    public static IEnumerable<Sale> GetData(int total)
    {
        var countries = new[] { "米国", "イギリス", "カナダ", "日本", "中国",
    "フランス", "ドイツ", "イタリア", "韓国", "オーストラリア" };
        var products = new[] { "Widget", "Gadget", "Doohickey" };
        var colors = new[] { "黒色", "白色", "赤色", "緑色", "青い色" };
        var rand = new Random(0);
        var dt = DateTime.Now;
        var list = Enumerable.Range(0, total).Select(i =>
        {
            var country = countries[rand.Next(0, countries.Length - 1)];
            var product = products[rand.Next(0, products.Length - 1)];
            var color = colors[rand.Next(0, colors.Length - 1)];
            var date = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60, i %
60);

            return new Sale
            {
                ID = i + 1,
                開始日 = date,
                終了日 = date,
                国名 = country,
                製品名 = product,
                色 = color,
                金額 = rand.NextDouble() * 10000 - 5000,
                金額2 = rand.NextDouble() * 10000 - 5000,
                割引 = rand.NextDouble() / 4,
                アクティブ = (i % 4 == 0),
                傾向 = Enumerable.Range(0, 12).Select(x =>
                new MonthData { Month = x + 1,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                Data = rand.Next(0, 100) }).ToArray(),
                 ランク = rand.Next(1, 6)
            };
        });
        return list;
    }

    internal static dynamic GetCountries()
    {
        throw new NotImplementedException();
    }

    internal static dynamic GetProducts()
    {
        throw new NotImplementedException();
    }
}

public class MonthData
{
    public int Month { get; set; }
    public double Data { get; set; }
}
    }
```

先頭に戻る

手順2：コントローラーアクションの追加

1. [コントローラー]フォルダからHomeController.csを開きます。
2. JSONデータを返す次のコードを追加します。

C#

```
[HttpPost]
        public JsonResult GetData()
        {
            List<Sale> saleList = Sale.GetData(10).ToList<Sale>();
            return Json(saleList);
        }
```

先頭に戻る

手順3：ビューへのFlexGridの追加

1. [View/Home]フォルダからIndex.htmlを開きます。
2. このファイルの内容を、FlexGridコントロールを宣言する次のコードに置き換えます。以下のコードは、FlexGridのさまざまな列やプロパティを宣言していますが、データをコントロールには連結していません。

Razor

```
@(Html.C1().FlexGrid()
        .Id("fg")
        .AutoGenerateColumns(false)
        .IsReadOnly(false)
        .AutoClipboard(true)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .AllowSorting(true)
        .AllowAddNew(false)
        .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)

        .CssClass("grid")
        .Columns(bl =>
        {
            bl.Add(cb => cb.Binding("ID").Width("0.2*").IsReadOnly(true));
            bl.Add(cb => cb.Binding("製品名").Header("製品名").Name("Product"));
            bl.Add(cb => cb.Binding("国名").Header("国名").Name("Country"));
            bl.Add(cb => cb.Binding("金額").Header("金額").Name("Amount"));

        })
)
```

先頭に戻る

手順4：JavaScriptを使用したAJAX呼び出し

ここでは、クライアント側でデータをコントロールに連結するために、JavaScriptを使用して、**HomeController.cs**ファイルに作成したアクションへのAJAX呼び出しを行います。AJAX呼び出しの結果データをFlexGridの**itemSource**プロパティに割り当てるのではなく、以下のLoad関数でクライアント側でグリッドを連結する場合は、collectionViewのサーバー側の機能を維持するために、FlexGridの**collectionView**のsourceCollectionをクライアント側で更新する必要があります。また、JSONデータの日付値のエラーをチェックするコードも追加しています。

1. **ソリューションエクスプローラー**で、**[Views]**フォルダをクリックします。
2. **[Home]**フォルダ内の`Index.html`ファイルを開き、次のJavaScriptコードをコピーします。

Index.html

```
<script type="text/javascript">
   function parseDate(strDate) {
        var date = strDate.match(/\d+/g);
        return new Date(parseInt(date));
    }

    function Load() {
        $.ajax({
            type: "POST",
            url: "/Home/GetData",
            dataType: "json",
            success: function (result) {
                var flex = wijmo.Control.getControl("#fg"),
                    cv = flex.collectionView;
                //flex.itemsSource = result;
                try {
                    cv._isFillingData = true;
                    cv.deferUpdate(function () {
                        cv.sourceCollection.clear();
                        result.forEach(function (item) {
                            item.Start = parseDate(item.Start);
                            item.End = parseDate(item.End);
                            cv.sourceCollection.push(item);
                        });
                    })
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                } finally {
                    cv._isFillingData = false;
                }
            },
            error: function (err) {

            }
        });
    }
</script>
```

3. ボタンがクリックされると上記のスクリプトを呼び出す次のコードを追加します。

Index.html

```
<input type="button" id="btload" value="Get Data" onclick="Load()" />
```

先頭に戻る

手順5：プロジェクトのビルドおよび実行

1. **[ビルド]→[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行した後、**[Get Data]**ボタンをクリックして、FlexGridにデータをロードして表示します。

先頭に戻る

FlexGridのマスター詳細

FlexGridをカスタマイズしてマスター/詳細データを表示できます。このインタフェースはデータを2つのグリッドで表示します。マスターデータを含むグリッドにはオブジェクトのコレクションが表示され、詳細グリッドにはユーザーがマスターグリッドで選択したオブジェクトの固有の詳細が表示されます。マスターグリッドで選択を変更すると、詳細グリッドはそれに合わせて更新されます。

このトピックでは、関連する2つのデータテーブルを使用してマスター詳細データを表示する方法について説明します。次の例ではEntity Frameworkモデルを使用して、ローカルNorthWindデータベースファイル**C1NWind.mdf**からのCustomersおよびOrdersデータにFlexGridを連結します。

次の図は、マスター/詳細リレーションでデータを表示するFlexGridコントロールを示しています。マスターグリッドには、集計されたCustomersデータが4つの列に表示されます。しかし詳細グリッドには、マスターグリッドで選択された特定の顧客が発注した注文の情報の詳細が示されます。2つのテーブルの間には、CustomerID列を介して関連性が存在します。

Copyright © GrapeCity, Inc. All rights reserved.

Customers のデータ:

お客様ID	会社名	国名	都市
ALFKI	Alfreds Futterkiste	Germany	Berlin
ANATR	Ana Trujillo Emparedados y helados	Mexico	México D.F.
ANTON	Antonio Moreno Taquería	Mexico	México D.F.
AROUT	Around the Horn	UK	London
BERGS	Berglunds snabbköp	Sweden	Luleå
BLAUS	Blauer See Delikatessen	Germany	Mannheim
BLONP	Blondel père et fils	France	Strasbourg
BOLID	Bólido Comidas preparadas	Spain	Madrid
BONAP	Bon app'	France	Marseille
BOTTM	Bottom-Dollar Markets	Canada	Tsawassen

選択された客様のすべての受注:

受注ID	社員ID	受注日付	出荷日付	配達予定日	貨物	経由で	船名	配送先の国名	配送先の市名
10,643	6	1995/09/25	1995/10/03	1995/10/23	29.46	1	Alfreds Futter...	Germany	Berlin
10,692	4	1995/11/03	1995/11/13	1995/12/01	61.02	2	Alfred's Futter...	Germany	Berlin
10,702	4	1995/11/13	1995/11/21	1995/12/25	23.94	1	Alfred's Futter...	Germany	Berlin
10,835	1	1996/02/15	1996/02/21	1996/03/14	69.53	3	Alfred's Futter...	Germany	Berlin
10,952	1	1996/04/15	1996/04/23	1996/05/27	40.42	1	Alfred's Futter...	Germany	Berlin
11,011	3	1996/05/09	1996/05/13	1996/06/06	1.21	1	Alfred's Futter...	Germany	Berlin

次のコード例は、FlexGridコントロールでマスター/詳細データを表示する方法を示します。

コードの場合

MasterDetailController.cs

C# copyCode

```
{
    public class MasterDetailController : Controller
    {
        private C1NWindEntities db = new C1NWindEntities();
        public ActionResult MasterDet()
        {
            return View(db.Customers.Take(10).ToList());
        }
        public ActionResult DetailData([C1JsonRequest]
            CollectionViewRequest<Orders> requestData)
        {
          string customerID = requestData.ExtraRequestData["CustomerID"].ToString();
          return this.C1Json(CollectionViewHelper.Read(requestData,
          db.Orders.Where(s => s.CustomerID == customerID).ToList()));
        }
    }
```

MasterDetail.cshtml

FlexGridを初期化してマスターインタフェースでCustomerデータを表示します。

Razor copyCode

```
@using MVC_DataBinding.Models
@using C1.Web.Mvc.Grid
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@using C1.Web.Mvc
@model IEnumerable<Customers>


<h4>Customers のデータ:</h4>
@(Html.C1().FlexGrid().Id("Customer").AutoGenerateColumns(false).
IsReadOnly(true).SelectionMode(SelectionMode.Row)
.Bind(Model)
.Columns(cs =>
    cs.Add(c => c.Binding("CustomerID").Header("お客様ID").Width("*"))
    .Add(c => c.Binding("CompanyName").Header("会社名").Width("2*"))
    .Add(c => c.Binding("Country").Header("国名").Width("2*"))
    .Add(c => c.Binding("City").Header("都市").Width("2*"))
))
```

FlexGridを初期化して詳細インタフェースでOrderデータを表示します。

Razor copyCode

```
@(Html.C1().FlexGrid().Id("Orders").AutoGenerateColumns(false).
IsReadOnly(true)
.Bind(i => i.Bind(Url.Action("DetailData")).
  OnClientQueryData("getCustomer"))
.Columns(cs =>
    cs.Add(c => c.Binding("OrderID").Header("受注ID"))
    .Add(c => c.Binding("EmployeeID").Header("社員ID"))
    .Add(c => c.Binding("OrderDate").Header("受注日付"))
.Add(c => c.Binding("ShippedDate").Header("出荷日付").Format("yyyy/MM/dd"))
        .Add(c => c.Binding("RequiredDate").Header("配達予定日"))
  .Add(c => c.Binding("Freight").Header("貨物"))
    .Add(c => c.Binding("ShipVia").Header("経由で"))
    .Add(c => c.Binding("ShipName").Header("船名"))
        .Add(c => c.Binding("ShipCountry").Header("配送先の国名"))
        .Add(c => c.Binding("ShipCity").Header("配送先の市名"))
))
```

マスターFlexGridで選択されたCustomerIDに対応する注文データを表示するJavascript。

JavaScript copyCode

```
<script type="text/javascript">
    var grid, cv, detail, cvDetail, currentItem;
    c1.mvc.Utils.documentReady(function () {
        grid = wijmo.Control.getControl("#Customer");
        cv = grid.collectionView;
        cv.currentChanged.addHandler(getOrders);
        cv.moveCurrentToFirst();
    });

    function getOrders() {
        detail = wijmo.Control.getControl("#Orders");
        cvDetail = detail.collectionView;
        cvDetail.refresh();
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    function getCustomer(sender, e) {
        if (e.extraRequestData == null) {
            e.extraRequestData = {};
        }

        grid = wijmo.Control.getControl("#Customer");
        if (grid) {
            currentItem = grid.collectionView.currentItem;
            e.extraRequestData["CustomerID"] = currentItem ?
       currentItem.CustomerID : "";
        }
    }
</script>
```

モデル 連結

このセクションでは、MVC Web フォームに FlexGrid コントロールを追加し、コントロールにデータを挿入する方法について説明します。以下の例は、FlexGridコントロールにローカルモデルのバインドを示しています。FlexGridでリモート連結については、「リモートデータ連結」を参照してください。

このトピックは3つの手順で構成されます。

- **手順 1：FlexGrid のデータソースの作成**
- **手順 2：FlexGrid コントロールの追加**
- **手順 3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の FlexGrid を示しています。

	ID	開始日	製品名	金額	割引	アクティブ
	1	2015年/01月/25日	Gadget	¥582	14%	☑
	2	2015年/02月/25日	Gadget	¥4,919	17%	☐
	3	2015年/03月/25日	Gadget	¥2,160	7%	☐
	4	2015年/04月/25日	Gadget	¥1,249	22%	☐
	5	2015年/05月/25日	Gadget	¥4,052	12%	☑
	6	2015年/06月/25日	Gadget	-¥3,131	16%	☐
	7	2015年/07月/25日	Widget	¥699	15%	☐
	8	2015年/08月/25日	Widget	¥3,464	3%	☐
	9	2015年/09月/25日	Gadget	-¥2,363	14%	☑
	10	2015年/10月/25日	Widget	-¥2,837	6%	☐
	11	2015年/11月/25日	Widget	¥878	8%	☐
	12	2015年/12月/25日	Widget	-¥3,788	12%	☐
	13	2015年/01月/25日	Gadget	-¥2,447	12%	☑
	14	2015年/02月/25日	Widget	-¥4,375	18%	☐

Copyright © GrapeCity, Inc. All rights reserved.

先頭に戻る

手順 1：FlexGrid のデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：FlexGridDataSource.cs）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FlexGrid コントロールのデータソースになるクラスを定義します。

Sale.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using MVCFlexGrid_JP.Models;



namespace MVCFlexGrid_JP.Models
{
    public class Sale
    {
        public int ID { get; set; }
    public DateTime 開始日 { get; set; }
    public DateTime 終了日 { get; set; }
    public string 国名 { get; set; }
    public string 製品名 { get; set; }
    public string 色 { get; set; }
    public double 金額 { get; set; }
    public double 金額2 { get; set; }
    public double 割引 { get; set; }
    public bool アクティブ { get; set; }

    public MonthData[] 傾向 { get; set; }
    public int ランク { get; set; }

    /// <summary>
    /// データを取得
    /// </summary>
    /// <param name="total"></param>
    /// <returns></returns>
    public static IEnumerable<Sale> GetData(int total)
    {
        var countries = new[] { "米国", "イギリス", "カナダ", "日本", "中国",
    "フランス", "ドイツ", "イタリア", "韓国", "オーストラリア" };
        var products = new[] { "Widget", "Gadget", "Doohickey" };
        var colors = new[] { "黒色", "白色", "赤色", "緑色", "青い色" };
        var rand = new Random(0);
        var dt = DateTime.Now;
        var list = Enumerable.Range(0, total).Select(i =>
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            var country = countries[rand.Next(0, countries.Length - 1)];
            var product = products[rand.Next(0, products.Length - 1)];
            var color = colors[rand.Next(0, colors.Length - 1)];
            var date = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60, i %
60);

            return new Sale
            {
                ID = i + 1,
                開始日 = date,
                終了日 = date,
                国名 = country,
                製品名 = product,
                色 = color,
                金額 = rand.NextDouble() * 10000 - 5000,
                金額2 = rand.NextDouble() * 10000 - 5000,
                割引 = rand.NextDouble() / 4,
                アクティブ = (i % 4 == 0),
                傾向 = Enumerable.Range(0, 12).Select(x =>
                new MonthData { Month = x + 1,
               Data = rand.Next(0, 100) }).ToArray(),
                ランク = rand.Next(1, 6)
            };
        });
        return list;
    }

    internal static dynamic GetCountries()
    {
        throw new NotImplementedException();
    }

    internal static dynamic GetProducts()
    {
        throw new NotImplementedException();
    }
}

public class MonthData
{
    public int Month { get; set; }
    public double Data { get; set; }
}
    }
```

先頭に戻る

手順 2:FlexGrid コントロールの追加

FlexGrid コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。

Copyright © GrapeCity, Inc. All rights reserved.

2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すように MVC 参照を追加します。

C# copyCode

```
using System.Collections;
using System.Globalization;
using System.Linq;
using System.Web.Mvc;
using C1.Web.Mvc;
using MVCFlexGrid_JP.Models;
using System.Collections.Generic;
using System;
```

5. メソッド Index() を次のメソッドに置き換えます。

QuickStartController.cs

C# copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(15));
}
```

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されました。
6. ソリューションエクスプローラーで、`Index.cshtml` をダブルクリックします。
7. **Views\Index.cshtml** ファイルのデフォルトのコードを次のコードに置き換えて、**FlexGrid** コントロールを初期化します。

Razor (Index.cshtml) copyCode

```
@using C1.Web.Mvc;
@using MVCFlexGrid_JP.Models;
@using C1.Web.Mvc.Serializition;
@using C1.Web.Mvc.Grid;
@using C1.Web.Mvc.Fluent;
@using C1.Web.Mvc.CollectionView;


@model IEnumerable<Sale>


  // FlexGridのインスタンスを作成し、プロパティを設定します
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .Height(450)
    .Width(700)
    .AllowAddNew(true)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .CssClass("grid")
    .Bind(Model)
        // FlexGridに列データを連結します
     .Columns(bl =>
       {
           bl.Add(cb => cb.Binding("ID"));
           bl.Add(cb => cb.Binding("開始日").Format("yyyy年/MM月/dd日"));
           bl.Add(cb => cb.Binding("製品名"));
           bl.Add(cb => cb.Binding("金額").Format("c"));
           bl.Add(cb => cb.Binding("割引").Format("p0"));
           bl.Add(cb => cb.Binding("アクティブ"));




       })

)
```

手順 3:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

先頭に戻る

OData連結

OData model is a server-side model, so entire data is available at the server side. To apply OData binding in FlexGrid, you need to specify the service url, table name, and keys property. This topic describes how to add data to FlexGrid using OData binding.

The following image shows how the FlexGrid control appears after implementing the code below.

Copyright © GrapeCity, Inc. All rights reserved.



ProductID	ProductName	QuantityPerUnit	UnitsInStock
1	Chai	10 boxes x 20 bags	39
2	Chang	24 - 12 oz bottles	17
3	Aniseed Syrup	12 - 550 ml bottles	13
4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	53
5	Chef Anton's Gumbo Mix	36 boxes	0
6	Grandma's Boysenberry Spread	12 - 8 oz jars	120
7	Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	15
8	Northwoods Cranberry Sauce	12 - 12 oz jars	6
9	Mishi Kobe Niku	18 - 500 g pkgs.	29
10	Ikura	12 - 200 ml jars	31

In Code

FlexGridController

C#

```
public class FlexGridController : Controller
    {
        // GET: FlexGrid
        public ActionResult Index()
        {
            return View();
        }
    }
```

Index.cshtml

Razor

```
@(Html.C1().FlexGrid()
        .Id("ODataBinding")
        .IsReadOnly(true)
        .AllowAddNew(false)
        .AllowDelete(false)
        .CssStyle("height", "300px")
        .AutoGenerateColumns(false)
        .BindODataSource(odsb =>

odsb.ServiceUrl("http://services.odata.org/Northwind/Northwind.svc")
                            .TableName("Products")
                            .Keys("ProductID"))
        .Columns(csb =>
        {
            csb.Add(cb => cb.Binding("ProductID").Width("*"));
            csb.Add(cb => cb.Binding("ProductName").Width("*"));
            csb.Add(cb => cb.Binding("QuantityPerUnit").Width("*"));
            csb.Add(cb => cb.Binding("UnitsInStock").Width("*"));
        })
        .Filterable(fb => fb.DefaultFilterType(FilterType.Both)))
```


Copyright © GrapeCity, Inc. All rights reserved.

オンデマンド読み込み

FlexGrid supports loading data on demand through **ODataVirtualCollectionView**. The **ODataVirtualCollectionView** extends **ODataCollectionView** to provide on-demand loading of data. To perform on demand loading in FlexGrid, you need to access the **ODataVirtualCollectionViewService** class which supports loading data from the OData service on demand. The **ODataVirtualCollectionView** does not load the data from the server automatically. Instead, it relies on the **setWindow** method to load data fragments (windows) on demand.

The following example illustrates how the data is loaded on demand in the FlexGrid control using the **ODataVirtualCollectionViewService** class. In this example, we specify the service url, table name, and keys property to perform binding.

Controller Code

C#

```
public class FlexGridController : Controller
    {
        // GET: FlexGrid
        public ActionResult Index()
        {
            return View();
        }
    }
```

View Code

CSHTML

```
@(Html.C1().FlexGrid()
                .Id("ODataVirtualCollectionView")
                .CssStyle("height", "400px")
                .AllowAddNew(true)
                .AllowDelete(true)
                .BindODataVirtualSource(odsb =>
                                        odsb.ServiceUrl("~/MyNorthWind")
                                                .TableName("Orders")
                                                .Keys("OrderID")
                                        )
                .Filterable(fb => fb.DefaultFilterType(FilterType.Both))
                .OnClientScrollPositionChanged("scrollPositionChanged")
                .OnClientLoadedRows("oDataVCVLoaded")
)

@section Scripts{
<script type="text/javascript">
      function scrollPositionChanged(grid) {
        var rng = grid.viewRange;
        grid.collectionView.setWindow(rng.row, rng.row2);
      }
      function oDataVCVLoaded(grid, e) {
              var el = document.getElementById('totalItemCount');
              el.innerHTML = wijmo.format('{length:n0} items', grid.rows);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
</script>
}
```

リモートデータ連結

FlexGrid では、**C1JSONRequest** を使用してデータを直接取得できます。これは、サーバー、テーブル、および列を含むリモートデータの URL を指定します。返される配列は、**CollectionView** オブジェクトのデータソースとして使用されます。

`CollectionViewHelper` は、コレクションに編集、フィルタ処理、グループ化、ソートなどのサービスを提供する静的クラスです。このクラスには次のメソッドも含まれます。

- `Read()`:コレクションからデータを取得します。
- `Edit()`:FlexGrid で Excel 形式の編集を有効にします。詳細については、「Excel 形式の編集」を参照してください。
- `BatchEdit()`:一度に複数の項目を編集できます。詳細については、「一括編集」を参照してください。

`Bind` プロパティは、特定の操作を実行するためのアクション URL メソッドを渡すことで、FlexGrid をコレクションに連結するために使用されます。

このトピックでは、既存のデータソースからデータを取得する方法について説明します。これは、データをダッシュボードとして表すデータ集約型のアプリケーションやシナリオを開発する際に役立ちます。ローカルモデル連結については、「クイックスタート:FlexGrid へのデータの追加」を参照してください。

次の図は、「クイックスタート」でアプリケーションに追加したモデル Sale.cs から、C1JSON 要求によってデータをフェッチした後の FlexGrid を示しています。

	開始日	終了日	国名	製品名	金額	割引
	1 25 15	00:00	ドイツ	Gadget	¥582	14%
	2 25 15	01:01	カナダ	Gadget	¥4,919	17%
	3 25 15	02:02	日本	Gadget	¥2,160	7%
	4 25 15	03:03	ドイツ	Gadget	¥1,249	22%
	5 25 15	04:04	ドイツ	Gadget	¥4,052	12%
	6 25 15	05:05	カナダ	Gadget	-¥3,131	16%
	7 25 15	06:06	中国	Widget	¥699	15%
	8 25 15	07:07	米国	Widget	¥3,464	3%
	9 25 15	08:08	韓国	Gadget	-¥2,363	14%
	10 25 15	09:09	米国	Widget	-¥2,837	6%

次のコード例は、リモートデータソースからデータをフェッチするために FlexGrid を連結する方法を示します。

コードの場合

RemoteBindController.cs

C# copyCode

```
    }
        public partial class FlexGridController : Controller
{
    public ActionResult RemoteBind_Read([C1JsonRequest] CollectionViewRequest<Sale>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
requestData)
    {
        return this.C1Json(CollectionViewHelper.Read(requestData, Sale.GetData(10)));
    }

    public ActionResult RemoteBind()
    {
        return View();
    }
```

RemoteBind.cshtml

Razor

```
@using C1.Web.Mvc.Grid
@(Html.C1().FlexGrid()
    .AutoGenerateColumns(false)
    .AllowSorting(true)
    .Bind(Url.Action("RemoteBind_Read"))
    .CssClass("grid")
    .SelectionMode(SelectionMode.Row)
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("ID").Visible(false));
        columns.Add(column => column.Binding("開始日").Format("MMM d yy"));
        columns.Add(column => column.Binding("終了日").Format("HH:mm"));
        columns.Add(column => column.Binding("国名"));
        columns.Add(column => column.Binding("製品名"));
        columns.Add(column => column.Binding("金額").Format("c,ja-jp"));
        columns.Add(column => column.Binding("割引").Format("p0"));
    })
)
```

DataMap

FlexGrid コントロールは、グリッドに自動検索機能を提供するデータマッピングをサポートします。グリッドにデータベースからのデータを表示する際、基本のデータ構造を変更せずにクライアント側に表示される値を編集する場合に自動検索機能が役立ちます。例えば、製品名を「ID」列の代わりに「製品名」列に表示でき、「色」列に色を参照する16進コードの代わりに色の名前を表示できます。

次の図は、**DisplayMemberPath**プロパティと**SelectedValuePath**プロパティを使用して「製品名」列と「色」列に**DataMap**を割り当てた後のFlexGridコントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	開始日	終了日	国名	製品名	色	価格
	1	2015年 01月 25日	00:00	ドイツ	Widget	緑色	¥582
	2	2015年 02月 25日	01:01	カナダ	Widget	緑色	¥4,919
	3	2015年 03月 25日	02:02	日本		赤色	¥2,160
	4	2015年 04月 25日	03:03	ドイツ		赤色	¥1,249
	5	2015年 05月 25日	04:04	ドイツ		黒色	¥4,052
	6	2015年 06月 25日	05:05	カナダ	Gadget	黒色	-¥3,131
	7	2015年 07月 25日	06:06	中国	Widget	赤色	¥699
	8	2015年 08月 25日	07:07	米国	Widget	白色	¥3,464
	9	2015年 09月 25日	08:08	韓国	Gadget	黒色	-¥2,363
	10	2015年 10月 25日	09:09	米国	Widget	白色	-¥2,837
	11	2015年 11月 25日	10:10	フランス	Widget	緑色	¥878
	12	2015年 12月 25日	11:11	韓国	Widget	赤色	-¥3,788
	13	2015年 01月 25日	12:12	ドイツ	Gadget	赤色	-¥2,447
	14	2015年 02月 25日	13:13	ドイツ	Widget	黒色	-¥4,375
	15	2015年 03月 25日	14:14	日本	Gadget	緑色	¥1,089

Widget
Gadget
Doohickey

グリッド内の「製品名」列と「色」列にDataMapを割り当てない場合、これらの列ではデータソースから取得された製品IDおよび16進コードが表示されます。

次のコード例は、FlexGrid コントロールでデータマッピングを作成する方法を示します。この例では、「クイックスタート」でアプリケーションに追加した **Sale.cs**モデル、と**CustomerSale.cs**モデルを使用します。

コードの場合

CustomerSale.cs

C# copyCode

```
Type your example code here. It will be automatically colorized when you switch to
Preview or build the help system.
```

コントローラコード

以下のように、コントローラーへの参照を追加します。

C#

```
using System.Collections.Generic;using System.Linq;using System.Web.Mvc;using
<ApplicationName>.Models;using C1.Web.Mvc;using C1.Web.Mvc.Serializition;
```

Razor copyCode

```
public partial class DataMapController : Controller
{
    public static List<Sale> SALES = CustomerSale.GetData(15).ToList();
    public ActionResult Index()
    {
        ViewBag.Products = CustomerSale.PRODUCTS;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        ViewBag.Colors = CustomerSale.COLORS;
        return View(SALES);
    }
    public ActionResult ProductsUpdate([C1JsonRequest]
        CollectionViewEditRequest<Sale> requestData)
    {
        return this.C1Json(CollectionViewHelper.Edit<Sale>(requestData, sale =>
        {
            string error = string.Empty;
            bool success = true;
            var fSale = SALES.Find(item => item.ID == sale.ID);
            fSale.アクティブ = sale.アクティブ;
            fSale.金額 = sale.金額;
            fSale.色 = sale.色;
            fSale.国名 = sale.国名;
            fSale.割引 = sale.割引;
            fSale.終了日 = sale.終了日;
            fSale.金額2 = sale.金額2;
            fSale.開始日 = sale.開始日;
            fSale.製品名 = sale.製品名;
            return new CollectionViewItemResult<Sale>
            {
                Error = error,
                Success = success && ModelState.IsValid,
                Data = fSale
            };
        }, () => SALES));
    }
}
```

Viewコード

Razor copyCode

```
@model IEnumerable<Sale>
@using C1.Web.Mvc.Grid
@{
    var products = ViewBag.Products;
    var colors = ViewBag.Colors;
}

@(Html.C1().FlexGrid<Sale>()
    .Id("ovFlexGrid")
    .AutoGenerateColumns(false)
    .Bind(bl => bl.InitialItemsCount(15).Bind(Model)
    .Update(Url.Action("ProductsUpdate")))
    .CssClass("grid")
    .IsReadOnly(false)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("開始日").Format("MMM d yy"));
        bl.Add(cb => cb.Binding("終了日").Format("HH:mm"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        bl.Add(cb => cb.Binding("国名"));
        bl.Add(cb => cb.Binding("製品名")
            .DataMap(dm => dm.DisplayMemberPath("Name")
                .SelectedValuePath("Id")
                .Bind(products)));
        bl.Add(cb => cb.Binding("色")
            .DataMap(dm => dm.DisplayMemberPath("Name")
                .SelectedValuePath("Value")
                .Bind(colors)));
        bl.Add(cb => cb.Binding("金額").Format("c"));
    })
)
```

DataMapエディター

DataMap have been an essential feature of the FlexGrid control. It provides FlexGrid with automatic look up capabilities. When you assign a DataMap to a FlexGrid column, it performs the following:

- Automatic mapping between the raw data values and the values displayed by the grid.
- Enhanced editing with drop-down lists, auto-complete, and validation.

For example, let's say, you have a list of data items with a country field that contains the country name. If you bind that to a grid, users might type invalid or inconsistent information (say, "US" vs. "USA" vs. "United States"). This can be prevented by creating a data map containing a list of valid countries and assigning that list to the Grid's "Country" column.

Country	Product	Color	Amount	Active	Rank
Canada	Gadget	White	$581.61	☑	1 2 3 4 5
France	Gadget	White	$4,775.50	☐	1 2 3 4 5
Canada	Widget	Red	($304.88)	☐	1 2 3 4 5
US	Gadget	Green	$1,771.81	☐	1 2 3 4 5
Italy	Gadget	Green	($4,673.75)	☑	1 2 3 4 5
China	Gadget	Red	$468.15	☐	1 2 3 4 5
UK	Widget	White	$4,885.44	☐	1 2 3 4 5
German	Widget	White	($1,568.58)	☐	1 2 3 4 5
China	Gadget	Black	($2,265.49)	☑	1 2 3 4 5
Italy	Widget	White	($3,531.75)	☐	1 2 3 4 5

By default, FlexGrid provides a drop-down list for editing data-mapped cells. However, you can change the type of editor used for editing data-mapped cells or column using **DataMapEditor** property of the **Column** class. The **DataMapEditor** property allows you to select the type of editor by accepting values from the **DataMapEditor** enumeration, which are listed as follows:

- **DropDownList**: This is the default setting which shows an input element with auto-complete, validation, and a drop-down list.
- **AutoComplete**: Shows an input element with auto-complete and validation, but no drop-down.
- **RadioButtons**: Shows radio buttons with mouse and keyboard support. This option displays all available choices at once and provides single-click editing.

The following example demonstrates the use of all types of editor for editing different data-mapped columns.

Add Model

CustomerSale copyCode

Copyright © GrapeCity, Inc. All rights reserved.

```
public class CustomerSale
{
    public int ID { get; set; }
    public string Country { get; set; }
    public string Product { get; set; }
    public string Color { get; set; }
    public double Amount { get; set; }
    public bool Active { get; set; }
    public int Rank { get; set; }


    public static List<string> COUNTRIES = new List<string> { "US", "UK", "Canada",
"Japan", "China", "France", "German", "Italy", "Korea", "Australia" };
    public static List<NamedProduct> PRODUCTS = new List<NamedProduct> {
        new NamedProduct {Id = "1", Name = "Widget"},
        new NamedProduct {Id = "2", Name = "Gadget"},
        new NamedProduct {Id = "3", Name = "Doohickey"}
    };

    public static List<NamedColor> COLORS = new List<NamedColor> {
        new NamedColor {Name = "Black", Value = "#000000"},
        new NamedColor {Name = "White", Value = "#FFFFFF"},
        new NamedColor {Name = "Red", Value = "#FF0000"},
        new NamedColor {Name = "Green", Value = "#008000"},
        new NamedColor {Name = "Cyan", Value = "#00FFFF"},
        new NamedColor {Name = "DarkBlue", Value = "#0000A0"},
        new NamedColor {Name = "LightBlue", Value = "#ADD8E6"},
        new NamedColor {Name = "Purple", Value = "#800080"},
        new NamedColor {Name = "Yellow", Value = "#FFFF00"},
        new NamedColor {Name = "Lime", Value = "#00FF00"},
        new NamedColor {Name = "Magenta", Value = "#FF00FF"},
        new NamedColor {Name = "Olive", Value = "#808000"},
        new NamedColor {Name = "Maroon", Value = "#800000"},
        new NamedColor {Name = "Brown", Value = "#A52A2A"},
        new NamedColor {Name = "Orange", Value = "#FFA500"},
        new NamedColor {Name = "Gray", Value = "#808080"},
        new NamedColor {Name = "Silver", Value = "#C0C0C0"},
        new NamedColor {Name = "Night", Value = "#0C090A"},
        new NamedColor {Name = "Gunmetal", Value = "#2C3539"},
        new NamedColor {Name = "Midnight", Value = "#2B1B17"},
        new NamedColor {Name = "Charcoal", Value = "#34282C"},
        new NamedColor {Name = "Oil", Value = "#3B3131"},
        new NamedColor {Name = "Black Cat", Value = "#413839"},
        new NamedColor {Name = "Iridium", Value = "#3D3C3A"},
        new NamedColor {Name = "Columbia Blue", Value = "#87AFC7"},
        new NamedColor {Name = "Teal", Value = "#008080"},
        new NamedColor {Name = "Venom Green", Value = "#728C00"},
        new NamedColor {Name = "Blue", Value = "#0000FF"}
    };

    public static List<NamedCountry> NAMEDCOUNTRIES = COUNTRIES.Select(country =>
    {
        return new NamedCountry { Name = country };
    }).ToList();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public static List<NamedRank> RANKS = Enumerable.Range(1, 5).Select(i =>
    {
        return new NamedRank { Name = i };
    }).ToList();

    public static IEnumerable<CustomerSale> GetData(int total)
    {
        var colors = new[] { "Black", "White", "Red", "Green", "Blue" };

        var rand = new Random(0);
        var dt = DateTime.Now;
        var list = Enumerable.Range(0, total).Select(i =>
        {
            var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
            var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)].Id;
            var color = colors[rand.Next(0, colors.Length - 1)];

            return new CustomerSale
            {
                ID = i + 1,
                Country = country,
                Product = product,
                Color = color,
                Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                Active = (i % 4 == 0),
                Rank = rand.Next(1, 6)
            };
        });
        return list;
    }

    public class NamedCountry
    {
        public string Name { get; set; }
    }

    public class NamedProduct
    {
        public string Id { get; set; }

        public string Name { get; set; }
    }

    public class NamedRank
    {
        public int Name { get; set; }
    }


}

public class NamedColor
{
    public string Name { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public string Value { get; set; }
}
```

Add Controller

DataMapEditorController.cs copyCode

```
public static List<CustomerSale> sales = CustomerSale.GetData(10).ToList();
// GET: DataMapEditor
public ActionResult Index()
{
    return View(sales);
}

public ActionResult ProductsUpdate([C1JsonRequest]
CollectionViewEditRequest<CustomerSale> requestData)
{
    return this.C1Json(CollectionViewHelper.Edit<CustomerSale>(requestData, csale =>
    {
        string error = string.Empty;
        bool success = true;
        var fSale = sales.Find(item => item.ID == csale.ID);
        fSale.Active = csale.Active;
        fSale.Amount = csale.Amount;
        fSale.Color = csale.Color;
        fSale.Country = csale.Country;
        fSale.Product = csale.Product;
        fSale.Rank = csale.Rank;
        return new CollectionViewItemResult<CustomerSale>
        {
            Error = error,
            Success = success && ModelState.IsValid,
            Data = fSale
        };
    }, () => sales));
}
```

Add View for the Controller

Index.cshtml copyCode

```
@using C1.Web.Mvc.Grid
@using DataMapEditor_MVC.Models

@model IEnumerable<CustomerSale>
@{
    List<CustomerSale.NamedCountry> countries = CustomerSale.NAMEDCOUNTRIES;
    List<CustomerSale.NamedProduct> products = CustomerSale.PRODUCTS;
   List<NamedColor> colors= CustomerSale.COLORS;
    List<CustomerSale.NamedRank> ranks = CustomerSale.RANKS;
}


@(Html.C1().FlexGrid<CustomerSale>()
    .Id("ovFlexGrid")
    .AutoGenerateColumns(false)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .Bind(bl => bl.Bind(Model).Update(Url.Action("ProductsUpdate")))
    .CssClass("grid")
    .IsReadOnly(false)
    .Columns(bl =>
    {
    bl.Add(cb => cb.Binding("ID").Visible(false));
    bl.Add(cb => cb.Binding("Country").DataMapEditor(DataMapEditor.AutoComplete)
        .DataMap(dm => dm.DisplayMemberPath("Name")
            .SelectedValuePath("Name")
            .SortByDisplayValues(true)
            .Bind(countries)));
        bl.Add(cb => cb.Binding("Product").DataMapEditor(DataMapEditor.DropDownList)
        .DataMap(dm => dm.DisplayMemberPath("Name")
            .SelectedValuePath("Id")
            .SortByDisplayValues(true)
            .Bind(products)));
        bl.Add(cb => cb.Binding("Color").DataMapEditor(DataMapEditor.DropDownList)
            .DataMap(dm => dm.DisplayMemberPath("Name")
                .SelectedValuePath("Value")
                .SortByDisplayValues(true)
                .Bind(colors)));
        bl.Add(cb => cb.Binding("Amount").Format("c"));
        bl.Add(cb => cb.Binding("Active"));
        bl.Add(cb =>
cb.Binding("Rank").Width("250").Align("center").DataMapEditor(DataMapEditor.RadioButtons)
            .DataMap(dm => dm.DisplayMemberPath("Name")
                .SelectedValuePath("Name")
                .SortByDisplayValues(true)
                .Bind(ranks)));
    })
)

@(Html.C1().FlexGrid<CustomerSale>()
    .Id("multiColumns")
    .AutoGenerateColumns(false)
    .Bind(bl =>
bl.InitialItemsCount(10).Bind(Model).Update(Url.Action("ProductsUpdate")))
    .CssClass("grid")
    .IsReadOnly(false)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID").Visible(false));
        bl.Add(cb => cb.Binding("Country"));
        bl.Add(cb => cb.Binding("Product")
            .DataMap(dm => dm.DisplayMemberPath("Name")
                .SelectedValuePath("Id")
                .SortByDisplayValues(true)
                .Bind(products)));
        bl.Add(cb => cb.Binding("Color")
            .DropDownCssClass("multi-column")
            .DataMap(dm => dm.DisplayMemberPath("Name")
                .SelectedValuePath("Value")
                .SortByDisplayValues(true)
                .Bind(colors)));
        bl.Add(cb => cb.Binding("Amount").Format("c"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        bl.Add(cb => cb.Binding("Active"));
    })
)

<style>
    .wj-radio-map label {
        padding: 0 0 0 0;
    }
</style>
```

詳細行

FlexGridは詳細行機能をサポートしており、テンプレート化された展開可能な行をコントロールに追加できます。FlexGridの任意の行に対して詳細セクションを追加でき、データをテンプレートにグループ化してオプションで表示できます。これによりエンドユーザーは、単に行を選択するだけで、その行に関連する追加データを表示できます。

組み込みの展開/折りたたみボタンも用意されており、展開可能な行に含まれるデータの表示/非表示を制御します。詳細行の内部にグリッド内グリッドを追加して、階層グリッドインタフェースを作成できます。

次の図は、**Detail Row** を持つFlexGridを示しています。例では、ローカルNorthWindデータベースファイル**C1NWind.mdf**からCustomerデータを使用しています。

	ID	会社名	国名	都市
−	ALFKI	Alfreds Futterkiste	Germany	Berlin

	ShippedDate	Freight	ShipVia
	1995/10/03	29.46	1
	1995/11/13	61.02	2
	1995/11/21	23.94	1
	1996/02/21	69.53	3
	1996/04/23	40.42	1
	1996/05/13	1.21	1

+	ANATR	Ana Trujillo Emparedados y helados	Mexico	México D.F.
+	ANTON	Antonio Moreno Taquería	Mexico	México D.F.
	AROUT	Around the Horn	UK	London
+	BERGS	Berglunds snabbköp	Sweden	Luleå
+	BLAUS	Blauer See Delikatessen	Germany	Mannheim
+	BLONP	Blondel père et fils	France	Strasbourg
	BOLID	Bólido Comidas preparadas	Spain	Madrid
+	BONAP	Bon app'	France	Marseille
+	BOTTM	Bottom-Dollar Markets	Canada	Tsawassen

次のコード例は、FlexGridで詳細行を有効にする方法を示します。

コードの場合

DetailRowController.cs

C# copyCode

```
{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public class DetailRowController : Controller
    {
    private C1NWindEntities db = new C1NWindEntities();
        public ActionResult DetailRow()
        {
            return View(db.Customers.Take(10).ToList());
        }

        public ActionResult
DetailRow_Read([C1JsonRequest]CollectionViewRequest<Orders> requestData)
        {
            string customerID =
requestData.ExtraRequestData["CustomerID"].ToString();
            return this.C1Json(CollectionViewHelper.Read(requestData,
db.Orders.Where(s => s.CustomerID == customerID).ToList()));
        }
    }
```

DetailRow.cshtml

Razor copyCode

```
<script>
    function hasDetail(row) {
        return row.dataItem.Country.length > 5;
    }

    //extra query パラメーターを追加します
    function addQueryData(sender, e) {
        var grid, currentItem;

        if (e.extraRequestData == null) {
            e.extraRequestData = {};
        }
        grid = wijmo.Control.getControl("#detailRowFlexGrid");
        if (grid) {
   currentItem = grid.collectionView.currentItem;
   e.extraRequestData["CustomerID"] = currentItem ? currentItem.CustomerID : "";
        }
    }
</script>

<script id="detailTemplate" type="text/template">
    @(Html.C1().FlexGrid()
        .Height("200px")
        .AutoGenerateColumns(false)
        .Bind(bl => bl.Bind(Url.Action("DetailRow_Read"))
        .OnClientQueryData("addQueryData"))
        .Columns(columns =>
     {
       columns.Add(column => column.Binding("ShippedDate").Width("*"));
       columns.Add(column => column.Binding("Freight").Width("*").Align("Center"));
       columns.Add(column => column.Binding("ShipVia").Width("*").Align("Center"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
      })
          .ToTemplate()
     )
</script>

@(Html.C1().FlexGrid()
     .ShowDetailRow(d => d.DetailRowTemplateId("detailTemplate")
     .RowHasDetail("hasDetail")
     .DetailVisibilityMode(C1.Web.Mvc.Grid.DetailVisibilityMode.ExpandSingle))
     .Id("detailRowFlexGrid")
     .AutoGenerateColumns(false)
     .IsReadOnly(true)
     .Bind(Model)
     .Columns(columns =>
     {
          columns.Add(column => column.Binding("CustomerID").Header("ID").Width("*"));
          columns.Add(column => column.Binding("CompanyName")
          .Header("会社名").Width("2*").Align("Center"));
          columns.Add(column => column.Binding("Country")
          .Header("国名").Width("2*").Align("Center"));
          columns.Add(column => column.Binding("City")
           .Header("都市").Width("2*").Align("Center"));
     })
)
```

パブリッククラス**CustomerWithOrders**は**Customer**クラスを継承しており、CustomerWithOrdersという名前のモデルで次のように定義できることに注意してください。

Example Title copyCode

```
public class CustomerWithOrders : Customer
{
    public List<Order> Orders { get; set; }
}
```

サーバー読み取りの無効化

FlexGrid では、DisableServerRead プロパティの値を True に設定することで、サーバー側のソート、ページング、フィルタ処理、またはスクロールを無効にできます。デフォルトでは、この値は False に設定されています。任意の操作に対してこのプロパティ値を true に設定すると、すべての現在の項目がクライアント側に転送され、サーバー側イベントは無効になります。ネットワーク呼び出しを実行せずにクライアント側の操作を直接実行できます。

次の図は、**DisableServerRead** プロパティを設定してページサイズを **10** に設定した FlexGrid を示しています。この例では、「クイックスタート」セクションで追加した Sale.cs モデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.



	開始日	製品名	金額	金額2
	2015年/01月/25日	Gadget	¥582	-¥2,940
	2015年/02月/25日	Gadget	¥4,919	-¥4,674
	2015年/03月/25日	Gadget	¥2,160	-¥3,810
	2015年/04月/25日	Gadget	¥1,249	-¥2,816
	2015年/05月/25日	Gadget	¥4,052	-¥3,109
	2015年/06月/25日	Gadget	-¥3,131	-¥4,315
	2015年/07月/25日	Widget	¥699	-¥2,746
	2015年/08月/25日	Widget	¥3,464	¥1,132
	2015年/09月/25日	Gadget	-¥2,363	¥3,426
	2015年/10月/25日	Widget	-¥2,837	-¥4,283

1 / 2

次のコード例は、FlexGrid でサーバー読み取りを無効にする方法を示します。この例では、「ページング」トピックで追加した _Pager.cshtml ページを使用します。

コードの場合

DisableServerReadingController.cs

C# copyCode

```
        public ActionResult DisableServerRead(FormCollection collection)
{
    IValueProvider data = collection;
    if (CallbackManager.CurrentIsCallback)
    {
        var request =
CallbackManager.GetCurrentCallbackData<CollectionViewRequest<object>>();
        if (request != null && request.ExtraRequestData != null)
        {
            var extraData = request.ExtraRequestData.Cast<DictionaryEntry>()
                .ToDictionary(kvp => (string)kvp.Key, kvp => kvp.Value.ToString());
            data = new DictionaryValueProvider<string>(extraData,
CultureInfo.CurrentCulture);
        }
    }
    return View(Sale.GetData(20));
}
```

DisableServerRead.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@using C1.Web.Mvc.Grid
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@model IEnumerable<Sale>

@(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
      .Columns(columns =>
        {
            columns.Add(column => column.Binding("開始日").Format("yyyy年/MM月/dd日"));
            columns.Add(column => column.Binding("製品名").AllowResizing(false));
            columns.Add(column => column.Binding("金額").Format("c"));
            columns.Add(column => column.Binding("金額2").Format("c"));
        })
    .CssStyle("height", "auto")
    .Id("dsrPagingGrid")
    .CssClass("grid")
    .IsReadOnly(true)
    .Filterable<Sale>()
    .Bind(b => b.DisableServerRead(true)
    .PageSize(10)
    .Bind(Model))
)

@Html.Partial("_Pager", "dsrPagingGrid")
```

編集

FlexGrid supports the following types of editing.

一括編集
: FlexGridで一括編集を有効にする方法を学びます。

Excel 形式の編集
: 組み込みのExcelのような編集の使用方法を学びます。

一括編集

FlexGrid は一括編集をサポートしています。一括編集を使用すると、複数の項目を更新、作成、または削除し、それをデータソースにコミットできます。これにより、ソート、ページング、またはフィルタ処理を使用しながら、グリッドで複数の変更を実行できます。FlexGrid を **BatchEditing** モードにするには、BatchEditing アクション URL を提供する必要があります。

> **メモ:** ソート／フィルタ／ページングの操作中にデータ更新を無効にするには、ItemsSourceの**DisableServerRead**プロパティをTrueに設定します。この設定により、クライアント側のソート、フィルタ、ページングが有効になり、データを有効にするには、CollectionViewのcommitメソッドがクライアント側から明示的に呼ばれたときだけデータが送信されます。

次の図は、**BatchEditing アクション URL** プロパティを設定した後の FlexGrid を示しています。この例では、サーバーにデータをコミットする方法を2つ提供します。1つめは、セルのコンテンツを更新した後に[Enter]キーをクリックします。2つめは、[更新]ボタンをクリックします。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexGrid でバッチ編集を有効にする方法を示します。

コードの場合

BatchEditingController.cs

C# copyCode

```
using System.Data.Entity.Validation;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc;
using System.Data;
using MVC_DataBinding.Models;
using System.Data.Entity;

namespace MVC_DataBinding.Controllers.FlexGrid
{
    public class BatchEditFlexGridController : Controller
    {

        //NWind データベースと連結します
        private C1NWindEntities db = new C1NWindEntities();

        public ActionResult BatchEditing(CollectionViewRequest<Categories>
requestData)
        {
            return View(db.Categories.ToList());
        }
        public ActionResult GridReadCategory([C1JsonRequest]
CollectionViewRequest<Categories> requestData)
        {
            return this.C1Json(CollectionViewHelper.Read(requestData,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
db.Categories));
        }

        public ActionResult GridBatchEdit([C1JsonRequest]
        CollectionViewBatchEditRequest<Categories> requestData)
        {
            return this.C1Json(CollectionViewHelper.BatchEdit(requestData, batchData
=>
            {
                var itemresults = new List<CollectionViewItemResult<Categories>>();
                string error = string.Empty;
                bool success = true;
                try
                {
                    if (batchData.ItemsCreated != null)
                    {
                        batchData.ItemsCreated.ToList().ForEach(st =>
                        {
                            db.Categories.Add(st);
                            itemresults.Add(new CollectionViewItemResult<Categories>
                            {
                                Error = "",
                                Success = ModelState.IsValid,
                                Data = st
                            });
                        });
                    }
                    if (batchData.ItemsDeleted != null)
                    {
                        batchData.ItemsDeleted.ToList().ForEach(category =>
                        {
                            var fCategory = db.Categories.Find(category.CategoryID);
                            db.Categories.Remove(fCategory);
                            itemresults.Add(new CollectionViewItemResult<Categories>
                            {
                                Error = "",
                                Success = ModelState.IsValid,
                                Data = category
                            });
                        });
                    }
                    if (batchData.ItemsUpdated != null)
                    {
                        batchData.ItemsUpdated.ToList().ForEach(category =>
                        {
                            db.Entry(category).State = EntityState.Modified;
                            itemresults.Add(new CollectionViewItemResult<Categories>
                            {
                                Error = "",
                                Success = ModelState.IsValid,
                                Data = category
                            });
                        });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                    }
                    db.SaveChanges();
                }
                catch (DbEntityValidationException e)
                {
                    error = string.Join(",", e.EntityValidationErrors
            .SelectMany(i => i.ValidationErrors).Select(i => i.ErrorMessage));
                    success = false;
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }

                return new CollectionViewResponse<Categories>
                {
                    Error = error,
                    Success = success,
                    OperatedItemResults = itemresults
                };
            }, () => db.Categories.ToList()));
        }
```

BatchEditing.cshtml

Razor copyCode

```
@using MVC_DataBinding.Models

<script type="text/javascript">
    function batchUpdate() {
        var batchEditGrid = wijmo.Control.getControl('#batchEditGrid'),
            cv = batchEditGrid.collectionView;
        cv.commit();
    }
</script>

<input type="button" value="Update" class="btn" onclick="batchUpdate()" />

@(
 Html.C1().FlexGrid<Categories>()
    .Id("batchEditGrid").AutoGenerateColumns(false)
    .Columns(columns => columns
        .Add(c => c.Binding("CategoryID").Header("ID").IsReadOnly(true))
            .Add(c => c.Binding("CategoryName").Header("カテゴリ名"))
            .Add(c => c.Binding("Description").Header("明細").Width("400")))
    .Bind(ib =>
ib.Bind(Url.Action("GridReadCategory")).BatchEdit(Url.Action("GridBatchEdit")))
    .AllowAddNew(true)
    .AllowDelete(true)
    .CssClass("grid")
)
```


Copyright © GrapeCity, Inc. All rights reserved.

遅延一括更新

FlexGrid provides support for deferred batch updates through ODataCollectionViewService class which allows MVC applications to use OData data sources. It supports CRUD operations by committing changes (edits, adds, and deletes) immediately to the database.

To defer batch updates, you can set the **DeferCommits** property to **true**. This causes the ODataCollectionView to track any changes made to the data and prevent any commits to the database until you click the Commit button to commit the changes to the database or click the Cancel button to discard the changes without committing them as shown in the following example where the code creates an ODataCollectionView with deferred commits:

Commit Cancel

OData Id	OData edit link	Airline Code	Name
http://services.odata....	http://services.odata....	AA	American Airl...
http://services.odata....	http://services.odata....	FM	Shanghai Airl...
http://services.odata....	http://services.odata....	MU	China Easter...
http://services.odata....	http://services.odata....	AF	Air France
http://services.odata....	http://services.odata....	AZ	Alitalia
http://services.odata....	http://services.odata....	AC	Air Canada
http://services.odata....	http://services.odata....	OS	Austrian Airli...
http://services.odata....	http://services.odata....	TK	Turkish Airlines

Add Model

Category.cs copyCode

```
public partial class Category
{
    public int CategoryID { get; set; }
    public string CategoryName { get; set; }
    public string Description { get; set; }
}
```

Add Controller

ODataDeferCommmitController.cs

```
public ActionResult Index()
{
     return View();
}
```

Add View for the Controller

Example Title copyCode

```
@using ODataDeferCommmit_MVC.Models

@model IEnumerable<Category>

@section Scripts{
    <script type="text/javascript">
        var cv, oDataDeferCommits, minHeight;

        c1.documentReady(function () {
            oDataDeferCommits = wijmo.Control.getControl('#oDataDeferCommits');
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            cv = oDataDeferCommits.collectionView;
        });

        function commitEdits() {
            cv.commitChanges(function (r) {
                setQueryMessage('Commit Done');
            });

            var isChanged = (cv.itemsEdited && cv.itemsEdited.length);
            if (isChanged) {
                setQueryMessage('Commit Updating');
            } else {
                setQueryMessage('Commit NoChange');
            }

        }

        function cancelChanges() {
            cv.cancelChanges();
            setQueryMessage('');//clear message
        }

        function setQueryMessage(message, className) {
            var element = document.getElementById('queryMessage');
            element.innerHTML = message;
            element.className = className;
        }
    </script>
}

<input type="button" value="Commit" class="btn" onclick="commitEdits()" />
<input type="button" value="Cancel" class="btn" onclick="cancelChanges()" />
<span id="queryMessage"></span>
@(
Html.C1().FlexGrid<Category>()
    .Id("oDataDeferCommits").AutoGenerateColumns(false)
    .Columns(columns => columns
        .Add(c => c.Binding("@odata.id").Header("OData Id").IsReadOnly(true))
        .Add(c => c.Binding("@odata.editLink").Header("OData edit link").IsReadOnly(true))
        .Add(c => c.Binding("AirlineCode").Header("Airline Code"))
        .Add(c => c.Binding("Name")))
    .BindODataSource(odsb =>

odsb.ServiceUrl("https://services.odata.org/V4/(S(3vn0jej2dr3ebcgodm1zxcys))/TripPinServiceRW/")
                        .TableName("Airlines")
                        .Keys("AirlineCode")
                        .DeferCommits(true))
)
```

Excel 形式の編集

FlexGrid には、Excel 形式の高速なセル内編集が組み込みでサポートされています。余分な列を追加して、そこに表示モードと編集モードを切り替えるための[編集]ボタンを置く必要はありません。

ユーザーは、任意のセルでキー入力することで編集を開始できます。これで、そのセルはクイック編集モードになります。このモードでは、カーソルキーを押すと編集が終了し、選択範囲が別のセルに移動します。[F2]キーを押すか、セルをダブルクリックすることで、編集を実行することもできます。これで、そのセルは完全編集モードになります。このモードでは、カーソルキー

Copyright © GrapeCity, Inc. All rights reserved.

を押すと、セルテキスト内のカレットが移動します。編集を終了し、別のセルに移動するには、[Enter]、[Tab]、または[Esc]キーを押す必要があります。編集が終了すると、データは自動的に更新されます。

Grid、Column、または Row オブジェクトの **IsReadOnly** プロパティを使用して、Grid、Column、または Row レベルで編集を無効にすることができます。この例では、ID 列を読み取り専用にします。

次の図は、**CollectionView** を使用して編集を有効にした後の FlexGrid を示しています。

	ID	カテゴリ名	明細
	1	Beverages	Soft drinks, coffees, teas, beers, and ales
	2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
	3	Confections	Desserts, candies, and sweet breads
	4	Dairy Products	Cheeses
	5	Grains/Cereals	Breads, crackers, pasta, and cereal
	6	Meat/Poultry	Prepared meats
	7	Produce	Dried fruit and bean curd
	8	Seafood	Seaweed and fish
*			

次の例は、編集モードにあるグリッドの **CRUD**（Create、Read、Update、Delete）操作を示します。次のコードを使用すると、FlexGrid で Excel 形式の編集が有効になります。この例では、Column Pickerでアプリケーションに追加されたPerson.csモデルを使用しています。

Controllerのコード

コントローラーに次の参照を追加します。

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc;
using <ApplicationName>.Models;
```

次のコードを使用して、作成、更新、および削除機能を追加します。

C# copyCode

```
public class HomeController : Controller
{
    private static List<Person> People = SampleData.GetData(20).ToList<Person>();

    public ActionResult Index()
    {
        return View(People);
    }

    [HttpPost]
    public ActionResult GridCreate([C1JsonRequest] CollectionViewEditRequest<Person>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
requestData)
    {
        return this.C1Json(CollectionViewHelper.Edit<Person>(requestData, person =>
        {
            string error = string.Empty;
            bool success = true;
            try
            {
                person.ID = People.Count;
                People.Add(person);
            }
            catch (Exception e)
            {
                error = e.Message;
                success = false;
            }
            return new CollectionViewItemResult<Person>
            {
                Error = error,
                Success = success && ModelState.IsValid,
                Data = person
            };
        }, () => People));
    }
    [HttpPost]
    public ActionResult GridUpdateCategory([C1JsonRequest]
CollectionViewEditRequest<Person> requestData)
    {
        return Update(requestData, People);
    }
    private ActionResult Update<T>(CollectionViewEditRequest<T> requestData, List<T>
data) where T : class
    {
        return this.C1Json(CollectionViewHelper.Edit<T>(requestData, item =>
        {
            string error = string.Empty;
            bool success = true;
            try
            {
                Person _item = (Person)(object)item;
                List<Person> _persons = People;
                int index = People.FindIndex(x => x.ID == _item.ID);
                People[index] = _item;
            }

            catch (Exception e)
            {
                error = e.Message;
                success = false;
            }
            return new CollectionViewItemResult<T>
            {
                Error = error,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                Success = success && ModelState.IsValid,
                Data = item
            };
        }, () => data.ToList<T>()));
    }
    [HttpPost]
    public ActionResult GridDelete([C1JsonRequest] CollectionViewEditRequest<Person>
requestData)
    {
        return this.C1Json(CollectionViewHelper.Edit<Person>(requestData, person =>
        {
            string error = string.Empty;
            bool success = true;
            try
            {
                int index = People.FindIndex(x=>x.ID == person.ID);
                People.RemoveAt(index);
            }
            catch (Exception e)
            {
                error = e.Message;
                success = false;
            }
            return new CollectionViewItemResult<Person>
            {
                Error = error,
                Success = success && ModelState.IsValid,
                Data = person
            };
        }, () => People));
    }
    public ActionResult About()
    {
        ViewBag.Message = "Your application description page.";

        return View();
    }

    public ActionResult Contact()
    {
        ViewBag.Message = "Your contact page.";

        return View();
    }

    [HttpPost]
    public PartialViewResult Search()
    {
        Random _rnd = new Random();
        var val = _rnd.Next(0, 10) < 5 ? 0 : 20;
        return PartialView("_Grid", SampleData.GetData(val));
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

Viewのコード

Razor copyCode

```
@using C1.Web.Mvc.Fluent
@using C1.Web.Mvc
@model IEnumerable<Person>

@(Html.C1().FlexGrid<Person>()
    .Id("editGrid")
    .Bind(
        ib => ib.Bind(Model)
          //グリッド内のアイテムを更新するためのアクションURL。
        .Update(Url.Action("GridUpdateCategory"))
          //グリッドに新しいアイテムを追加するためのアクションURL。
        .Create(Url.Action("GridCreate"))
          //グリッドからアイテムを削除するためのアクションURL。
        .Delete(Url.Action("GridDelete"))
    ).AutoGenerateColumns(false)
    .Columns(columns =>
        {
        columns.Add(column => column.Binding("ID").IsReadOnly(true));
        columns.Add(column => column.Binding("First"));
        columns.Add(column => column.Binding("Last"));
        columns.Add(column => column.Binding("Hired"));
    })
    .AllowAddNew(true)
    .AllowDelete(true)
    .CssStyle("height", "400px")
)
```

Excelのインポートまたはエクスポート

このトピックでは、FlexGridコンテンツをExcelのxlsxファイルにエクスポートまたはインポートする方法を示します。FlexGridコンテンツをExcel形式にエクスポートするには、FlexGridインスタンスを**FlexGridXlsxConverter.save**メソッドに渡します。これにより、xlsxファイルコンテンツが生成され、このコンテンツをローカルファイルに保存したり、サーバーに送ることができます。Excel（xlsxファイル）コンテンツをFlexGridコントロールにインポートするには、FlexGridインスタンスとxlsxファイルコンテンツを**FlexGridXlsxConverter.load**メソッドに渡します。この例では、「クイックスタート」でアプリケーションに追加した**Sale.cs**モデルを使用します。

次のコード例は、FlexGridコンテンツとExcel xlsxファイル間でエクスポートまたはインポートする方法を示します。

参照の追加

以下のコードを実装する前に、FlexGridをExcel xlsxファイルとの間でインポートおよびエクスポートするには、**jszip.min.js**ライブラリの参照を追加する必要があります。

_Layout.cshtml

HTML

```
<!-- SheetJS library -->
```


Copyright © GrapeCity, Inc. All rights reserved.

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.2.1/jszip.min.js">
</script>
```

HomeController.cs

C#

```
public ActionResult Index()
{
         return View(Sale.GetData(500));
}
```

App.js

JavaScript

```
//コントロールの宣言
var gFlexGrid = null,
    IncludeHeadersExport = null,
    IncludeHeadersImport = null;

//コントロールの初期化
function InitialControls() {
    gFlexGrid = wijmo.Control.getControl("#gFlexGrid");
    IncludeHeadersImport = document.getElementById('IncludeHeadersImport');
    IncludeHeadersExport = document.getElementById('IncludeHeadersExport');
}

// エクスポート
function exportExcel() {
    if (gFlexGrid) {
        wijmo.grid.xlsx.FlexGridXlsxConverter.save(gFlexGrid, { includeColumnHeaders:
IncludeHeadersExport.checked }, 'FlexGrid.xlsx');
    }
};

// インポート
function importExcel() {
    if (gFlexGrid) {
        if ($('#importFile')[0].files[0]) {
            wijmo.grid.xlsx.FlexGridXlsxConverter.load(gFlexGrid, $('#importFile')
[0].files[0], { includeColumnHeaders: IncludeHeadersImport.checked });
        }
        else {
            alert('Select an Excel file to Import.');
        }
    }
};
```

Index.cshtml

Razor

```
@using C1MVCExcelImportExport.Models
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>

@{
    ViewBag.Title = "ComponentOne ASP.NET MVC Excel Import Export";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<!-- App scripts -->
<script src="@Url.Content("~/Scripts/app.js")" type="text/javascript"></script>

<div class="container">
    <div class="row">
        <div class="col-md-6 col-xs-12">
            <div class="form-inline well well-lg">
                <input type="file" class="form-control" id="importFile"
accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet" />
                <button class="btn btn-default" onclick="importExcel()">インポート
</button>
                <br />
                <div class="checkbox">
                    <label>
                        <input id="IncludeHeadersImport" type="checkbox"
checked="checked"> Include Column Headers
                    </label>
                </div>
            </div>
        </div>
        <div class="col-md-6 col-xs-12">
            <div class="form-inline well well-lg">
                <a href="#" class="btn btn-default" id="export"
onclick="exportExcel()">エクスポート</a>
                <div class="checkbox">
                    <label>
                        <input id="IncludeHeadersExport" type="checkbox"
checked="checked"> 列のヘッダを含む
                    </label>
                </div>
            </div>
        </div>
    </div>
    <div class="row">
@(Html.C1().FlexGrid().Id("gFlexGrid").AutoGenerateColumns(false).AllowSorting(true)
.("max-height", "400px").IsReadOnly(true)
Bind(bl => bl.DisableServerRead(true).Bind(Model)).CssClass("grid")
Columns(bl =>
        {
        bl.Add(cb => cb.Binding("Country").Header("Country").Width("*"));
        bl.Add(cb => cb.Binding("Product").Header("Product").Width("*"));
        bl.Add(cb => cb.Binding("Color").Header("Color").Width("*"));
        bl.Add(cb => cb.Binding("Start").Header("Start").Width("*"));
        bl.Add(cb => cb.Binding("Amount").Header("Amount").Format("n0").Width("*")
        .Aggregate(C1.Web.Mvc.Grid.Aggregate.Sum));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        })
)
    </div>
</div>

<script type="text/javascript">
    c1.documentReady(function () {
        if (window["InitialControls"]) {
            window["InitialControls"]();
        }
    });
</script>
```

PDF へのエクスポート

FlexGrid supports exporting the grid content to a PDF file. You can use the FlexGridPdfConverter, a PDFKit-based JavaScript library, to export FlexGrid to PDF (Portable Document Format) without using any server-side code. To export FlexGrid to a PDF, you need to use the **FlexGridPdfConverter.export** function that takes the following arguments:

- **embeddedFonts**: Provides information to the export library, about various custom fonts to be embedded, such as URL, name, style, weight.
- **styles**: It is used to set up the style for the grid elements and link them with the embedded fonts.
- Export settings.

You can also specify some security settings, such as a separate user password and owner password. In addition, you can also define permissions for the PDF document including Annotating, ContentAccessibility, Copying, DocumentAssembly, FillingForms, Modifying, and Printing. You can refer to the FlexGrid PDF Export product sample for experiencing how the MVC FlexGrid control supports security settings in PDF Export.

Moreover, you can also create and add tags in the FlexGrid cells when exporting the grid to a PDF file. You can use the tag method to create tags and addTag method to add tags to the logical document tree of your application. Note that tagged PDF file required document version 1.4 or higher. For more information, see the FlexGrid PDF Export product sample to understand how the MVC FlexGrid control supports tags in PDF Export.

In the following example, we have set the following export settings to show how you can export FlexGrid to PDF. The example uses Sale.cs model added in the Custom Editors topic.

- **scaleMode**: Determines how the FlexGrid content should be scaled in order to fit the page.
- **orientation**: Determines the orientation of pages.
- **version**: Determines the PDF version.
- **exportMode**: Determines which part of the FlexGrid should be exported (all of the data or only the current selection).

Controller

CellMarkerController.cs copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(20));
}
```

View for the Controller

Copyright © GrapeCity, Inc. All rights reserved.

Index.cshtml copyCode

```
@model IEnumerable<Sale>

<style type="text/css">
    .grid {
        height: 500px;
        border: 2px solid #e0e0e0;
        font-family: Cambria;
        font-weight: bold;
    }
</style>
<div>
    <button id="btnExportToPDF" class="btn btn-default">
        PDFへエクスポート
    </button>
    <br />
    <br />
    @*FlexGridをインスタンス化し、そのプロパティを設定します。*@
    @(Html.C1().FlexGrid<Sale>()
        .Id("fgrid")
                    .AutoGenerateColumns(false)
                    .Width(700)
                    .AllowAddNew(true)
                    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
                    .CssClass("grid")
                    .Bind(Model)

                     //FlexGridに列データを連結します。
                     .Columns(bl =>
                     {
                         bl.Add(cb => cb.Binding("ID"));
                         bl.Add(cb => cb.Binding("Start"));
                         bl.Add(cb => cb.Binding("Product"));
                         bl.Add(cb => cb.Binding("Amount").Format("c"));
                         bl.Add(cb => cb.Binding("Discount").Format("p0"));
                         bl.Add(cb => cb.Binding("Active"));
                     }))
</div>

@section Scripts{
    <script>
        c1.documentReady(function () {
            var grid = wijmo.Control.getControl('#fgrid');

            //Pdf
            document.getElementById("btnExportToPDF").addEventListener("click",
function () {
                wijmo.grid.pdf.FlexGridPdfConverter.export(grid,
'ExportedFlexGrid.pdf', {
                    exportMode: wijmo.grid.pdf.ExportMode.All,
                    scaleMode: wijmo.grid.pdf.ScaleMode.ActualSize,
                    orientation: wijmo.pdf.PdfPageOrientation.Portrait,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                    version: "v1_3"
                });
            });
        });
    </script>
}
```

排他値の検索

FlexGrid offers an exclusive value search feature which enables Excel-like behavior wherein searching excludes items from the filter. It provides **ExclusiveValueSearch** property in **ValueFilter** class, which sets a value that determines whether the filter should only include values selected by the **FilterText** property. This property is of type Boolean and is set to true, by default, allowing the search to exclude items from the filter. If set to false, searching only affects which items are displayed on the list and not which items are included in the filter.

To include exclusive value search in FlexGrid, use the following code. In this example, we have created a model with Person type data that includes ID, Name, Country, First, Last and Sales properties.

IndexController.cs

Example Title

```
public static List<Person> Persons = SampleData.GetData().ToList();
        public ActionResult Index()
        {
            return View(Persons);
        }
```

Index.cshtml

Example Title

```
@(Html.C1().FlexGrid<Person>().Id("flexGrid")
      .Filterable(f => f.DefaultFilterType(FilterType.Both)
           .ColumnFilters(cfsb =>
            {
                  cfsb.Add(cfb =>
cfb.Column("First").FilterType(FilterType.Value).ValueFilter(vfb =>
                  {
                       vfb.ExclusiveValueSearch(true);
                   }));
             })
    )
    .Bind(m => m.Bind(Model)
)
```

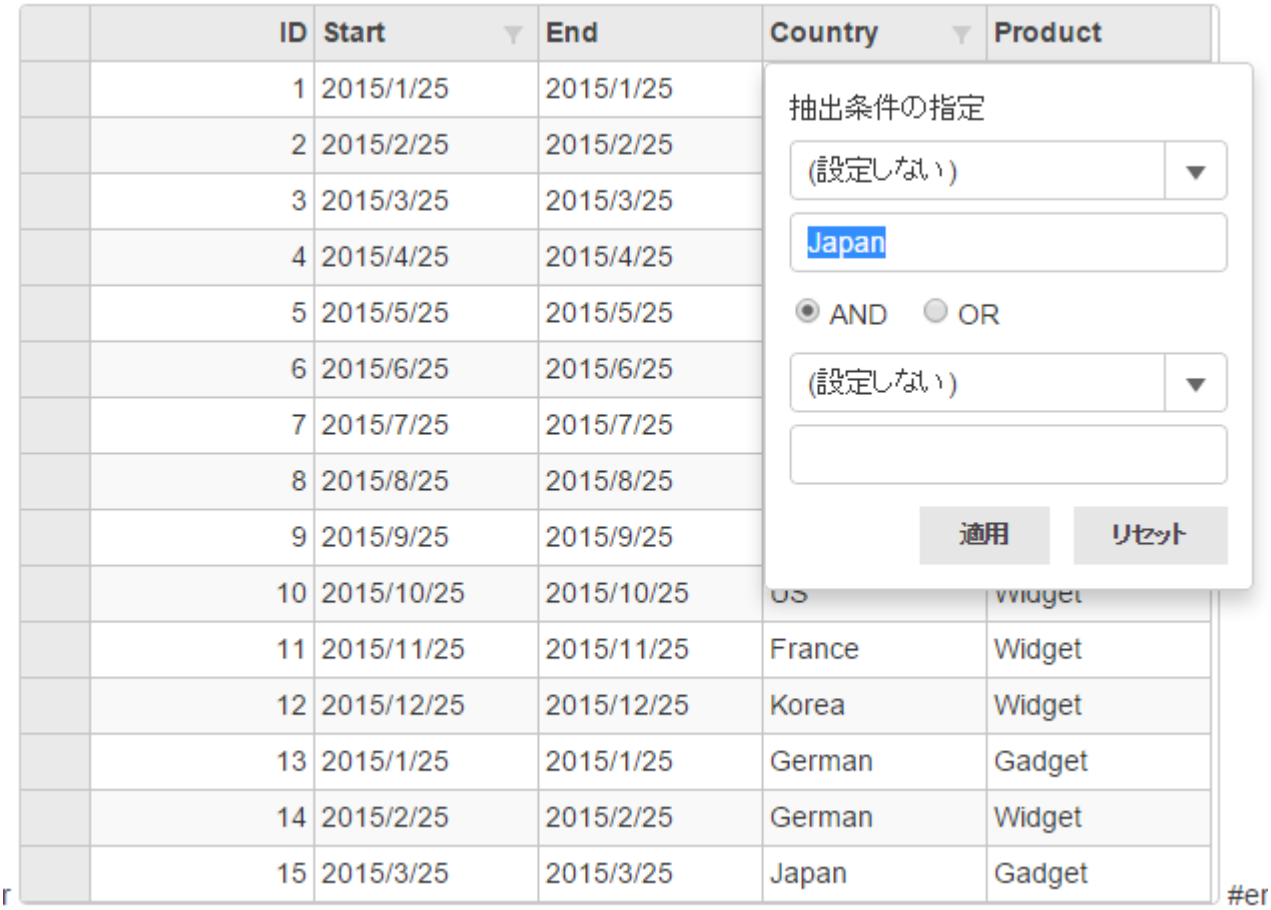
フィルタ処理

FlexGrid は、**ICollectionView** インタフェースを使用した**フィルタ処理**をサポートし、条件フィルタおよび値フィルタを適用することができます。したがって、グリッド内にあるデータから対象エントリを簡単に取得できます。

条件によるフィルタを適用することで検索結果を絞り込め、**値によるフィルタ**で対象値を入力して該当するデータを正確に検索できます。さらに、**Ascending** および **Descending** ボタンのそれぞれは、適用する列のデータを昇順や降順にソートします。

Copyright © GrapeCity, Inc. All rights reserved.

以下の例では、グリッドの各列にフィルタを適用し、検索する条件や値を指定して **Apply** ボタンをクリックすると、対象値を取得できます。この例で「クイックスタート」セクションで追加した **Sale.cs** モデルを使用します。



次のコード例は、FlexGrid でフィルタ処理を有効にする方法を示します。

コードの場合

FilteringController.cs

C# copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(15));
}
```

Filtering.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>

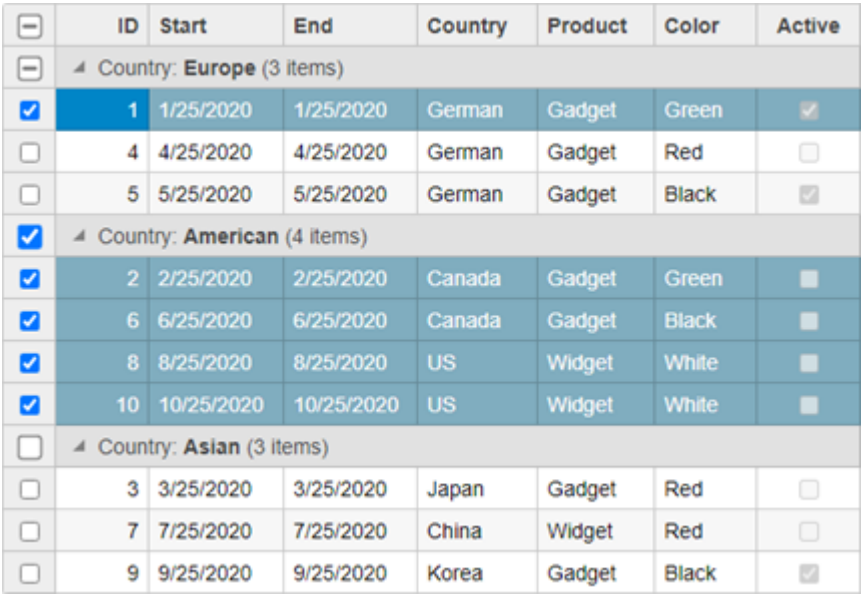
@(Html.C1().FlexGrid<Sale>()
    .Id("filteringGrid")
    .AutoGenerateColumns(false)
    .Bind(Model)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .IsReadOnly(true)
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("ID"));
        columns.Add(column => column.Binding("開始日").Format("yyyy年/MM月/dd日"));
        columns.Add(column => column.Binding("終了日").Format("t"));
        columns.Add(column => column.Binding("国名"));
        columns.Add(column => column.Binding("製品名"));
        columns.Add(column => column.Binding("色"));
    })
    .Filterable(f => f.DefaultFilterType(C1.Web.Mvc.FilterType.Both))
    .CssClass("grid")
)
```

FlexGridセレクター

FlexGrid selector adds check boxes to every grid row for multiple row selection. This selection scheme can be very useful on mobile devices, which have no keyboards with shift and control keys for extended selections. The selector is represented by the **Selector** class which defines an extender used to set the FlexGrid selector. The **Selector** class can be used on header columns as well as regular scrollable/data columns. It works with both, flat and hierarchical views, where you can toggle the selected state for a row in flat view and toggle the selected state for entire groups at once in hierarchical views.



	ID	Start	End	Country	Product	Color	Active
☐	Country: **Europe** (3 items)						
☑	1	1/25/2020	1/25/2020	German	Gadget	Green	☑
☐	4	4/25/2020	4/25/2020	German	Gadget	Red	☐
☐	5	5/25/2020	5/25/2020	German	Gadget	Black	☑
☑	Country: **American** (4 items)						
☑	2	2/25/2020	2/25/2020	Canada	Gadget	Green	☐
☑	6	6/25/2020	6/25/2020	Canada	Gadget	Black	☐
☑	8	8/25/2020	8/25/2020	US	Widget	White	☐
☑	10	10/25/2020	10/25/2020	US	Widget	White	☐
☐	Country: **Asian** (3 items)						
☐	3	3/25/2020	3/25/2020	Japan	Gadget	Red	☐
☐	7	7/25/2020	7/25/2020	China	Widget	Red	☐
☐	9	9/25/2020	9/25/2020	Korea	Gadget	Black	☑

The following example demonstrates how the check boxes show and toggle the selected state for all items in the group with an additional checkbox added to the column header to toggle the selection for all the rows.

Controller

CellMarkerController.cs copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(15));
}
```

View for the Controller

Copyright © GrapeCity, Inc. All rights reserved.

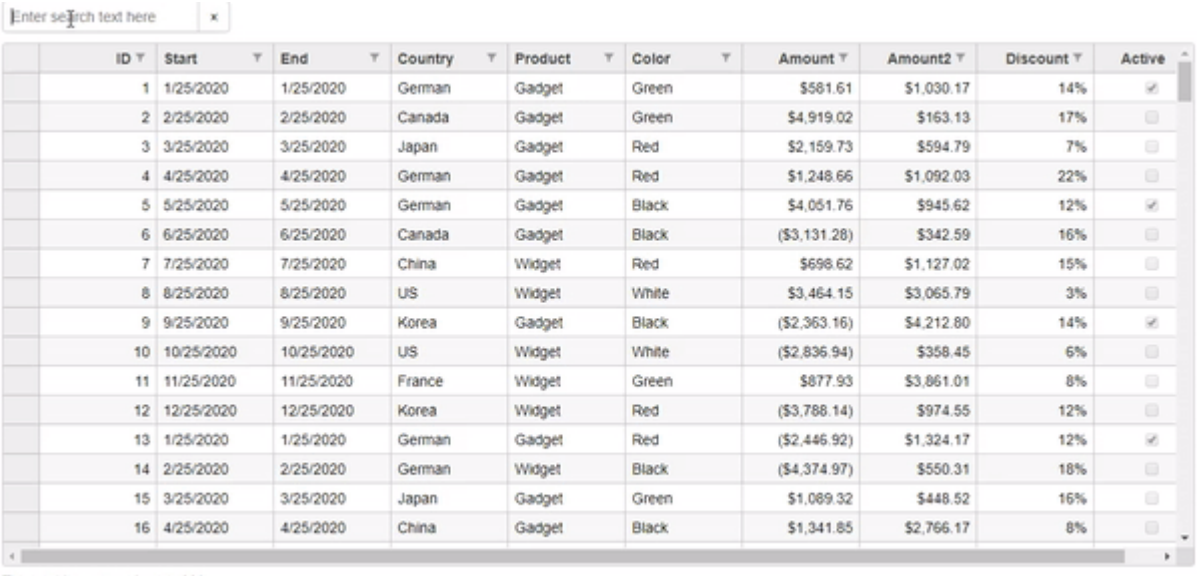
Index.cshtml copyCode

```
@model IEnumerable<Sale>

@*Instantiate FlexGrid and set its properties*@
@(Html.C1().FlexGrid<Sale>()
    .Id("ovFlexGrid").ShowGroupPanel(s => s
        .MaxGroups(Convert.ToInt32(4))
        .Placeholder("Please add columns for grouping here")
        .HideGroupedColumns(Convert.ToBoolean(false)))
    .AutoGenerateColumns(false)
    .Bind(Model)
    .PageSize(10)
    .CssClass("grid")
    .IsReadOnly(true)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("Start"));
        bl.Add(cb => cb.Binding("End"));
        bl.Add(cb => cb.Binding("Country"));
        bl.Add(cb => cb.Binding("Product"));
        bl.Add(cb => cb.Binding("Color"));
        bl.Add(cb => cb.Binding("Active"));
    })
    .Selector(sb => sb.ShowCheckAll(true))
)
```

フルテキスト検索

Searching a grid with huge data source can be a tedious task even in a grid that supports column filtering and sorting. To make this task easier, FlexGrid supports searching the entire data source connected to the grid through full-text search. To support full-text search in FlexGrid, you need to use the **FlexGridSearch** control which is represented by the **FlexGridSearch** class. This control appears similar to a search box that allows users to quickly search the items displayed in a FlexGrid.



ID	Start	End	Country	Product	Color	Amount	Amount2	Discount	Active
1	1/25/2020	1/25/2020	German	Gadget	Green	$581.61	$1,030.17	14%	☑
2	2/25/2020	2/25/2020	Canada	Gadget	Green	$4,919.02	$163.13	17%	☐
3	3/25/2020	3/25/2020	Japan	Gadget	Red	$2,159.73	$594.79	7%	☐
4	4/25/2020	4/25/2020	German	Gadget	Red	$1,248.66	$1,092.03	22%	☐
5	5/25/2020	5/25/2020	German	Gadget	Black	$4,051.76	$945.62	12%	☑
6	6/25/2020	6/25/2020	Canada	Gadget	Black	($3,131.28)	$342.59	16%	☐
7	7/25/2020	7/25/2020	China	Widget	Red	$698.62	$1,127.02	15%	☐
8	8/25/2020	8/25/2020	US	Widget	White	$3,464.15	$3,065.79	3%	☐
9	9/25/2020	9/25/2020	Korea	Gadget	Black	($2,363.16)	$4,212.80	14%	☑
10	10/25/2020	10/25/2020	US	Widget	White	($2,836.94)	$358.45	6%	☐
11	11/25/2020	11/25/2020	France	Widget	Green	$877.93	$3,861.01	8%	☐
12	12/25/2020	12/25/2020	Korea	Widget	Red	($3,788.14)	$974.55	12%	☐
13	1/25/2020	1/25/2020	German	Gadget	Red	($2,446.92)	$1,324.17	12%	☑
14	2/25/2020	2/25/2020	German	Widget	Black	($4,374.97)	$550.31	18%	☐
15	3/25/2020	3/25/2020	Japan	Gadget	Green	$1,089.32	$448.52	16%	☐
16	4/25/2020	4/25/2020	China	Gadget	Black	$1,341.85	$2,766.17	8%	☐

Copyright © GrapeCity, Inc. All rights reserved.

The following example shows the use of FlexGridSearch control to search the items in FlexGrid. The example uses **Sale.cs** model added in the QuickStart. The grid updates dynamically as you enter text and highlights any full or partial matches according to the text you enter.

Controller Code

C#

```
public ActionResult Index(FormCollection data)
{
        return View(Sale.GetData(200));
}
```

View Code

CSHTML

```
<p id="theSearch"></p>
@(Html.C1().FlexGrid<Sale>()
        .AutoGenerateColumns(false)
        .Id("theFlexGrid")
        .CssClass("grid")
        .Height(500)
        .ScrollIntoView(10, 2)
        .IsReadOnly(true).AutoSizeMode(C1.Web.Mvc.Grid.AutoSizeMode.Both)
        .Bind(Model)
        .Columns(bl =>
        {
                bl.Add(cb => cb.Binding("ID"));
                bl.Add(cb => cb.Binding("Start"));
                bl.Add(cb => cb.Binding("End"));
                bl.Add(cb => cb.Binding("Country"));
                bl.Add(cb => cb.Binding("Product"));
                bl.Add(cb => cb.Binding("Color"));
                bl.Add(cb => cb.Binding("Amount").Format("c"));
                bl.Add(cb => cb.Binding("Amount2").Format("c"));
                bl.Add(cb => cb.Binding("Discount").Format("p0"));
                bl.Add(cb => cb.Binding("Active"));
        })
        .Filterable(f => f.DefaultFilterType(FilterType.Both))
)

@(Html.C1().FlexGridSearch("#theSearch")
          .Grid("theFlexGrid")
          .Placeholder("Enter search text here")
)
<p>The total item count is now <b><span id="searchCount"></span></b>.</p>
@section Scripts{
        <script>
                function saveValue(key, value) {
                        if (sessionStorage) {
                        sessionStorage.setItem(key, value);
                        } else {
                        $.cookies.set(key, value);
```

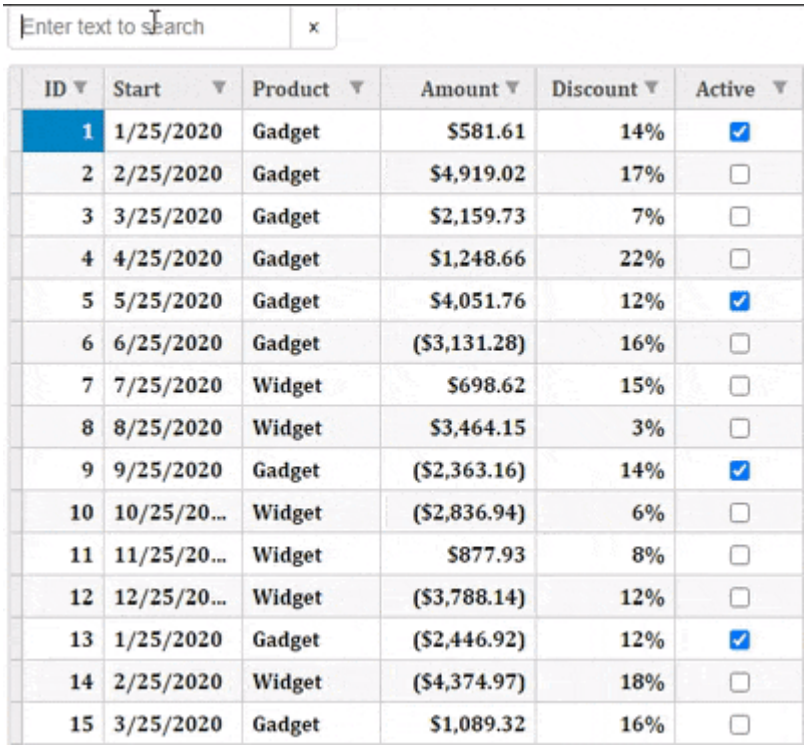

Copyright © GrapeCity, Inc. All rights reserved.

```
                }
            }
            function getValue(key) {
                if (sessionStorage) {
                return sessionStorage.getItem(key);
                } else {
                return $.cookies.get(key);
                }
            }
            function updateSearchCount(theGrid) {
                let cnt = theGrid.collectionView.items.length;
                document.getElementById('searchCount').textContent = cnt;
            }
            window.onbeforeunload = function () {
                let theSearch = wijmo.Control.getControl("#theSearch");
                saveValue("SearchValue", theSearch.text || "");
            }
            c1.documentReady(function () {
                let theSearch = wijmo.Control.getControl("#theSearch");
                theSearch.text = getValue("SearchValue") || "";
                let theGrid = wijmo.Control.getControl("#theFlexGrid");
                theGrid.collectionView.collectionChanged.addHandler(() => {
                    updateSearchCount(theGrid);
                });
            });
        </script>
}
```

大文字と小文字を区別する検索

Searching can be a tedious task in a grid with huge data. To make this task easier, you can use case-sensitive search supported by FlexGrid along with the FlexGridSearch control to apply full text searching inside the grid. With case-sensitive search, FlexGrid allows you to have more granular control over searched items in the grid. You can achieve the granularity in search by setting the **CaseSensitiveSearch** property to **true**. The **CaseSensitiveSearch** property determines whether the searches performed while the user types should be case-sensitive which makes our search more refined.

Copyright © GrapeCity, Inc. All rights reserved.



The following example shows how you can use the case-sensitive search along with the FlexGridSearch control to search the items in FlexGrid. The example uses Sale.cs model added in the Custom Editors topic.

Controller

CellMarkerController.cs copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(15));
}
```

View for the Controller

Index.cshtml copyCode

```
@model IEnumerable<Sale>

<style type="text/css">
    .grid {
        height: 500px;
        border: 2px solid #e0e0e0;
        font-family: Cambria;
        font-weight: bold;
    }
</style>
<div>
<p id="theSearch"></p>
 </div>
@*Instantiate FlexGrid and set its properties*@
@(Html.C1().FlexGrid<Sale>()
            .Id("fgrid")
            .AutoGenerateColumns(false)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            .Width(700)
            .AllowAddNew(true)
            .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
            .CssClass("grid")
            .Bind(Model)

            //Binding columns data to FlexGrid
            .Columns(bl =>
            {
                bl.Add(cb => cb.Binding("ID"));
                bl.Add(cb => cb.Binding("Start"));
                bl.Add(cb => cb.Binding("Product"));
                bl.Add(cb => cb.Binding("Amount").Format("c"));
                bl.Add(cb => cb.Binding("Discount").Format("p0"));
                bl.Add(cb => cb.Binding("Active"));
            })
            .Filterable(f => f.DefaultFilterType(FilterType.Both))
            .AutoSearch(true)
            .CaseSensitiveSearch(true)
)
@(Html.C1().FlexGridSearch("#theSearch")
    .Grid("fgrid")
    .Placeholder("Enter text to search")
 )
```

グループ化

Grouping refers to combining rows based on column values. When grouping is applied to a grid, it automatically sorts the data, splits it into groups, and adds collapsible group rows above or below each group.

FlexGrid supports column-wise grouping of grid data through the CollectionView class as a datasource. You can configure group description by using the **GroupBy** method in view. You can also add grouping using JavaScript by adding GroupDescription objects to the collectionView.groupDescriptions property. Here, the column for grouping the grid data is specified in the code.

You can customize the text displayed in group header rows using the **GroupHeaderFormat** property. By default, it displays the name of the group, followed by the current group and the number of items in the group. To format the aggregated data in the group header for a particular column, you can set the format property on each Column object.

The following image shows how the FlexGrid appears on using the **GroupBy** method. The example uses table, **Customer** from the **NWind.mdb** database.

Copyright © GrapeCity, Inc. All rights reserved.

	ID	連絡先名	住所	都市	地域	郵便番号	国名
	国名: Germany (11 項目)						
	ALFKI	Maria Anders	Obere Str. 57	Berlin	null	12209	Germany
	BLAUS	Hanna Moos	Forsterstr. 57	Mannheim		68306	Germany
	DRACD	Sven Ottlieb	Walserweg 21	Aachen		52066	Germany
	FRANK	Peter Franken	Berliner Platz 43	München		80805	Germany
	KOENE	Philip Cramer	Maubelstr. 90	Brandenburg		14776	Germany
	LEHMS	Renate Messner	Magazinweg 7	Frankfurt a.M.		60528	Germany
	MORGK	Alexander Feuer	Heerstr. 22	Leipzig		04179	Germany
	OTTIK	Henriette Pfal...	Mehrheimerst...	Köln		50739	Germany
	QUICK	Horst Kloss	Taucherstraße...	Cunewalde		01307	Germany

The following code examples demonstrate how to enable Grouping in the FlexGrid:

In Code

GroupingController.cs

C# copyCode

```
public ActionResult Grouping()
{
    var nwind = new C1NWindEntities();
    return View(nwind.Customers);
}
```

Grouping.cshtml

Razor copyCode

```
@using MVCFlexGrid.Models
@model IEnumerable<Customer>

@(Html.C1().FlexGrid<Customer>()
.Id("pagingGrid")
.AutoGenerateColumns(false)
.Height(300)
.Width(840)
.IsReadOnly(true)
.Bind(Model)
.GroupBy("Country")
    .Columns(columns =>
        {
            columns.Add(column => column.Binding("CustomerID"));
            columns.Add(column => column.Binding("ContactName"));
            columns.Add(column => column.Binding("Address"));
            columns.Add(column => column.Binding("City"));
            columns.Add(column => column.Binding("Region"));
            columns.Add(column => column.Binding("PostalCode").Format("p0"));
            columns.Add(column => column.Binding("Country"));
        })
```


Copyright © GrapeCity, Inc. All rights reserved.

```
)
@Html.Partial("_Pager", "pagingGrid")
```

実行時にグループ化

FlexGrid では、ユーザーがグリッドのデータをグループ化またはグループ化解除することができます。この機能は、Group Panel というコントロールでサポートされています。
Group Panel はドラッグアンドドロップによりグループ化する UI を提供し、FlexGrid から列をパネルにドラッグしてグループ化できます。

パネル内グループマーカーの順番を変更してグループ化されたデータの順序を簡単に変更でき、グループマーカーをクリックすると、該当する列に基づいてグループをソートできます。

MaxGroups プロパティを使用して、Group Panel でグループできる最大グループ数を設定することができます。Placeholder プロパティで、パネルにグループが存在していない場合にパネルで表示される文字列を設定します。また、HideGroupedColumns プロパティを Boolean 値に設定することで、グループ化した列を表示または非表示するように指定できます。

次の図は、Group Panel を使用した FlexGrid を示しています。この例では、「クイックスタート」トピックで作成した Sale.cs モデルを使用します。



次の図は、グリッドの上部にあるパネルに列を追加してグループ化した FlexGrid を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexGrid で Group Panel を有効にする方法を示します。

コードの場合

GroupAtRuntimeController.cs

C# copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(10));
}
```

GroupingAtRuntime.cshtml

HTML Helpers

Copyright © GrapeCity, Inc. All rights reserved.

Razor copyCode

```
@using C1.Web.Mvc.Grid
@using MVCFlexGrid_JP.Models
@model IEnumerable<Sale>

@(Html.C1().FlexGrid<Sale>()
    .Id("ovFlexGrid").ShowGroupPanel(s => s
        .MaxGroups(Convert.ToInt32(4))
        .Placeholder("グループする列をここにドラッグしてください。")
        .HideGroupedColumns(Convert.ToBoolean(false)))
    .AutoGenerateColumns(false)
    .Bind(Model)
    .PageSize(10)
    .CssClass("grid")
    .IsReadOnly(true)
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("開始日").Format("yyyy年/MM月/dd日"));
        bl.Add(cb => cb.Binding("終了日").Format("t"));
        bl.Add(cb => cb.Binding("国名"));
        bl.Add(cb => cb.Binding("製品名"));
        bl.Add(cb => cb.Binding("色"));
        bl.Add(cb => cb.Binding("アクティブ"));
    })
)
```

キーボードハンドリング

FlexGridは、キーボードイベントに応答するように簡単にカスタマイズできます。エンドユーザーが使いやすいように、Gridコントロールを使用するMVCアプリケーションでキーボードをサポートしたい場合があります。FlexGridには、キーボードを使用して機能を管理したりデータを編集または更新することができる柔軟性があります。

このセクションでは、キーボードを使用して操作できるようにFlexGridコントロールをカスタマイズするシナリオをいくつか紹介します。

FlexGridへのホットキーの割り当て

コードを使用して、MVCアプリケーションに適切なホットキーを割り当てることができます。これにより、多くの手動による作業が容易になり、時間を節約できます。たとえば、10列500行にもなる大量のデータを表示するグリッドでデータを編集する際は、ホットキーを使用できると便利なことがあります。次の例では、[Ctrl]+[Q]キーをホットキーとして使用して、アクティブな行内の列間でセルデータをコピーします。

以下のコードは、「クイックスタート」トピックで作成したアプリケーションを拡張しています。

Razor

```
  <script type=text/JavaScript>
        var grid;
        c1.mvc.Utils.documentReady(function () {
    grid = wijmo.Control.getControl('#gsFlexGrid');
    gridCV = grid.collectionView;
    if (grid) {
    var host = grid.hostElement;
    //keydownイベントを処理
```


Copyright © GrapeCity, Inc. All rights reserved.

```
host.addEventListener('keydown', function (e) {
if (e.key == "q" && e.ctrlKey)
{
grid.setCellData(grid.selection.row,4,
            grid.getCellData(grid.selection.row, 3));
console.log("現在の行のセルデータは、
            国名列から製品名列にコピーしました");
}
});
}
});
</script>
<div>
        @(Html.C1().FlexGrid<Sale>()
        .Id("gsFlexGrid")
        .AutoGenerateColumns(true)
        .AllowSorting(true)
        .Bind(Model)
        .Columns(bl =>
{
                bl.Add(cb => cb.Binding("ID"));
                bl.Add(cb => cb.Binding("Start"));
                bl.Add(cb => cb.Binding("Product"));
                bl.Add(cb => cb.Binding("Amount").Format("c"));
                bl.Add(cb => cb.Binding("Discount").Format("p0"));
                bl.Add(cb => cb.Binding("Active"));
})
</div>
```

ここでは、hostElementプロパティを使用して、FlexGridコントロールをホストするDOM要素を取得しました。コードは、FlexGridのkeydownイベントを処理し、[Ctrl]+[Q]キーをホットキーとしてキャプチャします。

先頭に戻る

クリップボードショートカットの使用によるFlexGridデータの編集

クリップボードショートカットの使用に慣れているなら、FlexGridコントロールでデータを編集する際に、簡単にクリップボードショートカットを利用できます。それには、次に示すように、FlexGridの**AutoClipboard**プロパティをtrueに設定します。

以下のコードは、「クイックスタート」トピックで作成したアプリケーションを拡張しています。

Razor

```
@(Html.C1().FlexGrid<Sale>().Id("cdInitMethod")
      .AutoGenerateColumns(false)
      .Bind(Model)
      .AutoClipboard(true)
      .Columns(bl =>
           {
             bl.Add(cb => cb.Binding("ID").Header("ID"));
             bl.Add(cb => cb.Binding("Start").Header("Start").Format("MMM d yy"));

             bl.Add(cb => cb.Binding("Country").Header("Country"));
             bl.Add(cb => cb.Binding("Product").Header("Product"));
             bl.Add(cb => cb.Binding("Discount").Format("p0"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            bl.Add(cb => cb.Binding("Active"));
            bl.Add(cb => cb.Binding("Amount").Format("n0").Visible(false));
        })
  )
```

ComponentOne FlexGridでは、次のクリップボードコマンドがサポートされています。

- Ctrl+C、Ctrl+Ins：グリッドの選択範囲をクリップボードにコピーします。
- Ctrl+V、Shift+Ins：クリップボードテキストをグリッドの選択範囲に貼り付けます。

読み取り専用のセルに貼り付け操作は効きません。さらに、クリップボード操作は、グリッドの表示されている行および列でのみサポートされます。

先頭に戻る

Enterキーによるフォーカス移動を列内を下へではなく行内を横への移動にする

C1FlexGridには、ユーザーが[Enter]キーを押したときに、グリッドの同じ列内の次のセルにフォーカスを移動する機能が組み込まれています。しかし、[Enter]キーが押されたときにフォーカスが同じ行内を横に移動するように動作を変更できます。たとえば、グリッドで売上データを扱っているときには、行単位で編集する方が作業しやすい場合があります。その場合は、1つのセルの編集が終了したら、同じ行内の次のセルに移動できると便利です。

それには、次に示すように、FlexGridのhostElementのkeydownイベントを処理し、選択範囲を同じ行内の次のセルに設定します。

以下のコードは、「クイックスタート」トピックで作成したアプリケーションを拡張しています。

Razor

```
<script type="text/javascript">
function onClientLoadedRows(e, args) {
        var g = e.hostElement;
        $(g).keydown(function (s, args1) {
        if (s.which == 13) {
        var flex = wijmo.Control.getControl("#gFlexGrid");
        var sel = flex.selection;
        var row = sel.row - 1;
        var col = sel.col;
        if (col < flex.columns.length - 1)
        {
          var rng = new
          wijmo.grid.CellRange(row, col + 1, row, col + 1);
          flex.select(rng, true);
        }
           else {
                 var rng = new
                 wijmo.grid.CellRange(row + 1, 0, row + 1, 0);
                 flex.select(rng, true);
                 }
        }
     });
  }</script>
<div>
@(Html.C1().FlexGrid<Sale>()
    .Id("gFlexGrid")
    .AutoGenerateColumns(true)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .AllowSorting(true)
    .Bind(Model)
    .IsReadOnly(false)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .OnClientLoadedRows("onClientLoadedRows")
    .Columns(bl =>{
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("Start"));
        bl.Add(cb => cb.Binding("Product"));
        bl.Add(cb => cb.Binding("Amount").Format("c"));
        bl.Add(cb => cb.Binding("Discount").Format("p0"));
    bl.Add(cb => cb.Binding("Active"));
        })
</div>
```

上記のコードでは、OnClientLoadedRowsイベントを処理しています。デフォルトでは、編集終了後に[Tab]キーを押すことでも、同じ動作を利用できます。

先頭に戻る

グリッドの行をロックして編集されないようにする

FlexGridでデータを編集中に、特定の行をロックして、行内のエントリが編集されないようにしたい場合があります。たとえば、人事担当者がGridコントロールを使用して会社の賃金データを管理しているとします。このとき、グリッド内の1行または2行をロックして、グリッドに表示されている従業員データに対して編集操作を実行しても、特定の従業員の詳細情報には影響しないようにすることができます。
それには、次に示すように、FlexGridのOnClientBeginningEditイベントを処理します。

以下のコードは、「クイックスタート」トピックで作成したアプリケーションを拡張しています。

Razor

```
<script type= "text\javascript">
function OnClientBeginningEdit(e, args) {
        if (args.row == 0) {
              args.cancel = true;
                  }
        }
 </script>
<div>
@(Html.C1().FlexGrid<Sale>().Id("gFlexGrid")
        .AutoGenerateColumns(true)
        .AllowSorting(true)
        .Bind(Model)
        .IsReadOnly(false)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .OnClientBeginningEdit("onClientBeginningEdit")
        .Columns(bl =>
        {
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("Start"));
        bl.Add(cb => cb.Binding("Product"));
        bl.Add(cb => cb.Binding("Amount").Format("c"));
        bl.Add(cb => cb.Binding("Discount").Format("p0"));
        bl.Add(cb => cb.Binding("Active"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        })
        </div>
```

ここでは、指定された行（この例では、最初の行）が編集モードに入るとたびに、そのイベントがイベントハンドラによってキャンセルされます。

先頭に戻る

マージ

FlexGrid は、同じコンテンツを含むセルのマージをサポートします。セルのマージを有効にするには、**AllowMerging** プロパティを設定して、グリッドのどの部分をマージするかを指定します。**AllowMerging** プロパティは、特定の行および列オブジェクトに対して設定できます。使用可能な値には、None、Cells、ColumnHeaders、RowHeaders、AllHeaders、および All があります。

AllowMerging プロパティをこれらのいずれかの値に設定すると、FlexGrid のセルが自動的にマージされ、データが視覚的にグループ化されます。

次の図は、**AllowMerging** プロパティを Cells にを設定した後の FlexGrid を示しています。この例では、「クイックスタート」でアプリケーションに追加した Sale.cs モデルを使用します。

	ID	国名	製品名	色	金額	割引
	1	ドイツ	Gadget	緑色	¥582	14%
	2	カナダ			¥4,919	17%
	3	日本		赤色	¥2,160	7%
	4	ドイツ			¥1,249	22%
	5	ドイツ		黒色	¥4,052	12%
	6	カナダ			-¥3,131	16%
	7	中国	Widget	赤色	¥699	15%
	8	米国		白色	¥3,464	3%
	9	韓国	Gadget	黒色	-¥2,363	14%
	10	米国	Widget	白色	-¥2,837	6%
	11	フランス		緑色	¥878	8%
	12	韓国		赤色	-¥3,788	12%
	13	ドイツ	Gadget		-¥2,447	12%
	14	ドイツ	Widget	黒色	-¥4,375	18%
	15	日本	Gadget	緑色	¥1,089	16%

次のコード例は、FlexGrid でマージを有効にする方法を示します。

コードの場合

MergingController.cs

C# copyCode

```
public ActionResult Index()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
{
    return View(Sale.GetData(15));
}
```

Merging.cshtml

Razor copyCode

```
@(Html.C1().FlexGrid<Sale>()
    .AllowMerging(C1.Web.Mvc.Grid.AllowMerging.Cells)
    .AutoGenerateColumns(false)
    .IsReadOnly(true)
    .Bind(Model)
    .CssClass("grid")
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("ID").Width("80"));
        columns.Add(column => column.Binding("国名"));
        columns.Add(column => column.Binding("製品名").AllowMerging(true));
        columns.Add(column => column.Binding("色").AllowMerging(true));
        columns.Add(column => column.Binding("金額").Format("c,ja-jp"));
        columns.Add(column => column.Binding("割引").Width("100").Format("p0"));
    })
)
```

ページング

FlexGrid はページングをサポートしています。ページャコントロールを使用して、グリッドにページ付きビューを実装します。また、**PageSize** プロパティの値には、1ページに入れる項目の数を設定できます。これで、ページ間を移動するための UI も提供されます。

FlexGrid でページャコントロールを使用してページングするには、次の設定を行う必要があります。

- FlexGridでIDを設定します、
- ページャコントロールの**Owner**プロパティを設定します。
- **PageSize** プロパティを設定します。

ページャコントロールを使用し、**Owner**プロパティを設定することで、先頭[<<]、前へ[<]、次へ[>]、最後[>>]の各ボタンを使用してページを切り替えることができます。

次の図は、**PageSize** プロパティを設定した後の FlexGrid を示しています。この例では、「クイックスタート」でアプリケーションに追加した Sale.cs モデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.



	ID	開始日	製品名	金額	割引	アクティブ
	1	2015年/01月/25日	Gadget	¥582	14%	☑
	2	2015年/02月/25日	Gadget	¥4,919	17%	☐
	3	2015年/03月/25日	Gadget	¥2,160	7%	☐
	4	2015年/04月/25日	Gadget	¥1,249	22%	☐
	5	2015年/05月/25日	Gadget	¥4,052	12%	☑
	6	2015年/06月/25日	Gadget	-¥3,131	16%	☐
	7	2015年/07月/25日	Widget	¥699	15%	☐
	8	2015年/08月/25日	Widget	¥3,464	3%	☐
	9	2015年/09月/25日	Gadget	-¥2,363	14%	☑
	10	2015年/10月/25日	Widget	-¥2,837	6%	☐

◀◀ ◀ 1 / 5 ▶ ▶▶

次のコード例は、FlexGrid でページングを有効にする方法を示します。

コードの場合

C# copyCode

```
@using C1MvcWebAppPager.Models
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>

@* FlexGridについてIDを設定します *@
@(Html.C1().FlexGrid<Sale>()
    .Id("pagerFlexGrid")
    .AutoGenerateColumns(false)
    .IsReadOnly(true)
    .Height(450)
    .Width(700)
    .AllowAddNew(true)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .CssClass("grid")
    .Bind(Model)
    .PageSize(10)
    .Columns(bl =>
       {    bl.Add(cb => cb.Binding("ID"));
            bl.Add(cb => cb.Binding("開始日").Format("yyyy年/MM月/dd日"));
            bl.Add(cb => cb.Binding("製品名"));
            bl.Add(cb => cb.Binding("金額").Format("c,ja-jp"));
            bl.Add(cb => cb.Binding("割引").Format("p0"));
            bl.Add(cb => cb.Binding("アクティブ"));
       })
)
@* PagerのOwnerプロパティを設定します *@
@(Html.C1().Pager().Owner("pagerFlexGrid"))
```


Copyright © GrapeCity, Inc. All rights reserved.

右から左へのレンダリング

FlexGrid には、アラビア語やヘブライ語などの言語で必要になる、ページの右から左にコンテンツをレンダリングする機能が組み込まれています。HTML ではこの機能は「dir」属性を使用してサポートされますが、FlexGrid ではコントロールには何もプロパティを設定せずに、任意の要素で「dir」を「rtl」に設定することで、この機能が自動的に実行されます。CSS の「direction」属性を使用しても同じ機能を実行できます。

「**dir**」属性値は継承されます。したがって、body タグでこの属性を設定すると、グリッドを含むページ全体が右から左にレンダリングされます。

アクティブ	製品名	国名	終了日	開始日	ID
☑	Gadget	ドイツ	0:00	…2年/01月/25日	1
☐	Gadget	カナダ	1:01	…2年/02月/25日	2
☐	Gadget	日本	2:02	…2年/03月/25日	3
☐	Gadget	ドイツ	3:03	…2年/04月/25日	4
☑	Gadget	ドイツ	4:04	…2年/05月/25日	5
☐	Gadget	カナダ	5:05	…2年/06月/25日	6
☐	Widget	中国	6:06	…2年/07月/25日	7
☐	Widget	米国	7:07	…2年/08月/25日	8
☑	Gadget	韓国	8:08	…2年/09月/25日	9
☐	Widget	米国	9:09	…2年/10月/25日	10
☐	Widget	フランス	10:10	…2年/11月/25日	11
☐	Widget	韓国	11:11	…2年/12月/25日	12
☑	Gadget	ドイツ	12:12	…2年/01月/25日	13
☐	Widget	ドイツ	13:13	…2年/02月/25日	14
☐	Gadget	日本	14:14	…2年/03月/25日	15
☐	Gadget	中国	15:15	…2年/04月/25日	16
☑	Widget	米国	16:16	…2年/05月/25日	17
☐	Widget	中国	17:17	…2年/06月/25日	18
☐	Gadget	フランス	18:18	…2年/07月/25日	19
☐	Gadget	フランス	19:19	…2年/08月/25日	20

次のコード例は、FlexGrid で右から左にレンダリングされるように、「dir」属性を設定する方法を示します。

コードの場合

RightToLeftController.cs

C# copyCode

```
public ActionResult Index()
{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    return View(Sale.GetData(20));
}
```

RightToLeft.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@model IEnumerable<Sale>

@(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .Bind(bl => bl.Bind(Model))
    .Width(700)
    .CssClass("grid")
    .Columns(bl =>
    {
        bl.Add(cb => cb.Binding("ID"));
        bl.Add(cb => cb.Binding("開始日").Format("yyyy年/MM月/dd日"));
        bl.Add(cb => cb.Binding("終了日").Format("t"));
        bl.Add(cb => cb.Binding("国名"));
        bl.Add(cb => cb.Binding("製品名"));
        bl.Add(cb => cb.Binding("アクティブ"));

    })
    .HtmlAttribute("dir", "rtl")
)
```

選択モード

Excelと同様に、FlexGridではデフォルトでマウスまたはキーボードを使用してセルの範囲を選択できます。**SelectionMode**プロパティでは、これを変更して、1行、行の範囲、隣接しない複数の行（リストボックスと同様）、1セル、セルの範囲を選択したり、または選択を完全に無効にできます。

FlexGridでの選択挙動を指定するには、**SelectionMode**を次のように設定します。

- **Cell**：SelectionModeを`Cell`に設定すると、ユーザーは一度に1つのセルだけを選択できます。
- **CellRange**：SelectionModeを`CellRange`に設定すると、ユーザーは連続するセルのブロックを選択できます。
- **ListBox**：SelectionModeを`ListBox`に設定すると、ユーザーは隣接しない複数の行を選択できます。
- **None**：SelectionModeを`None`に設定すると、ユーザーはマウスやキーボードでセルを選択することはできません。
- **Row**：SelectionModeを`Row`に設定すると、ユーザーは一度に1つの行を選択できます。
- **RowRange**：SelectionModeを`RowRange`に設定すると、ユーザーは隣接する行を選択できます。

次の図は、**SelectionMode**を**Row**に設定したときのFlexGridを示しています。この例では、「クイックスタート」セクションで追加したSale.csモデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.



ID	開始日	終了日	国名	製品名	色
1	2015/01/25	2015/01/25	ドイツ	Gadget	緑色
2	2015/02/25	2015/02/25	カナダ	Gadget	緑色
3	2015/03/25	2015/03/25	日本	Gadget	赤色
4	2015/04/25	2015/04/25	ドイツ	Gadget	赤色
5	2015/05/25	2015/05/25	ドイツ	Gadget	黒色
6	2015/06/25	2015/06/25	カナダ	Gadget	黒色
7	2015/07/25	2015/07/25	中国	Widget	赤色
8	2015/08/25	2015/08/25	米国	Widget	白色
9	2015/09/25	2015/09/25	韓国	Gadget	黒色
10	2015/10/25	2015/10/25	米国	Widget	白色
11	2015/11/25	2015/11/25	フランス	Widget	緑色
12	2015/12/25	2015/12/25	韓国	Widget	赤色
13	2015/01/25	2015/01/25	ドイツ	Gadget	赤色
14	2015/02/25	2015/02/25	ドイツ	Widget	黒色

次のコード例は、FlexGridの選択モードの設定方法を示します。

コードの場合

SelectionMode.cshtml

Razor

```
.SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)
```

スターサイズ

XAML スタイルのスターサイズを使用して、FlexGrid に柔軟なレイアウトを実装できます。次の例では、オブジェクトの Width プロパティで指定されたスターサイズを使用します。

このグリッドには3つの列があります。最初の列は **80 ピクセル**の幅で、**40 ピクセル**から **160 ピクセル**の間でサイズ変更できます。他の2つの列は、**2*** および * の幅を持ち、マウスを使用してサイズ変更することはできません。この2つの列の幅の比率は、最初の列やグリッド全体をサイズ変更しても変化しません。

次の図は、**Width** プロパティを使用して列にスターサイズを適用した FlexGrid を示しています。この例では、「クイックスタート」セクションで追加した Sale.cs モデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.

	開始日	製品名	国名
	2015年/01月/25日	Gadget	ドイツ
	2015年/02月/25日	Gadget	カナダ
	2015年/03月/25日	Gadget	日本
	2015年/04月/25日	Gadget	ドイツ
	2015年/05月/25日	Gadget	ドイツ
	2015年/06月/25日	Gadget	カナダ
	2015年/07月/25日	Widget	中国
	2015年/08月/25日	Widget	米国
	2015年/09月/25日	Gadget	韓国
	2015年/10月/25日	Widget	米国

次のコード例は、FlexGrid の列にスターサイズを適用する方法を示します。

コードの場合

StarSizingController.cs

C# copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(10));
}
```

StarSizing.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@using C1.Web.Mvc.Grid
@model IEnumerable<Sale>


@(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .CssClass("grid")
    .Columns(columns =>
    {
        columns.Add(column => column.Binding("開始日").Format("yyyy年/MM月/dd
日").Width("100").MaxWidth(160).MinWidth(40));
        columns.Add(column => column.Binding("製品
名").Width("2*").AllowResizing(false));
        columns.Add(column => column.Binding("国名").Format("c").Width("*"));
    })
    .Bind(Model)
)
```


Copyright © GrapeCity, Inc. All rights reserved.

スタイル設定とCSS

CSS Classes for FlexGrid control are as follows:

Class	Description
.wj-flexgrid	Styles the div host element of the FlexGrid control.
.wj-topleft .wj-cell	Styles the TopLeft corner cell/cells.
.wj-colheaders .wj-cell	Styles all the column header cells.
.wj-rowheaders .wj-cell	Styles all the row header cells.
.wj-header	Styles the header cells including column header, row header and top left cell.
.wj-row .wj-cell	Styles all the rows in grid.
.wj-alt	Styles the alternative rows in grid.
.wj-cells .wj-cell	Styles all the cells in the scrollable area (data cells).
.wj-cell	Styles all the cells in grid including column header, row header and top left cell.
.wj-bottomleft .wj-cell	Styles the div element containing cells collection for bottom left area.
.wj-colfooters .wj-cell	Styles the div element containing cells collection for column footers.
.wj-align-right	Styles the right aligned columns.
.wj-align-center	Styles the center aligned columns.
Selection	
.wj-state-selected	Styles the active cell and currently selected item in the drop down list.
.wj-state-multi-selected	Styles all the cells in the selected cell range except the active cell.
.wj-marquee	Styles the Excel-style element that highlights the selection when the showMarquee property is set to true.
.wj-state-active	Styles the active cell.
.wj-colheaders .wj-state-multi-selected	Styles the selected column headers when the showSelectedHeaders property is set to true.
.wj-rowheaders .wj-state-multi-selected	Styles the selected row headers when the showSelectedHeaders property is set to true.
Editing	
.wj-grid-editor	Styles the cell editor element.
.wj-glyph-pencil	Styles the pencil icon displayed in the row header of the currently editing row.
.wj-glyph-asterisk	Styles the asterisk icon displayed in the row header of the new row template.
Validation	

Copyright © GrapeCity, Inc. All rights reserved.

.wj-state-invalid	Styles the invalid cells.
Freezing	
.wj-frozen	Styles non-header frozen cells.
.wj-frozen-row	Styles cells in the last frozen row. This style can be used to create a horizontal divider line between the frozen and scrollable rows.
.wj-frozen-col	Styles cells in the last frozen column. This style can be used to create a vertical divider line between the frozen and scrollable columns.
Sorting	
.wj-glyph-up	Styles the up-arrow icon displayed in the column header of a column sorted in ascending order.
.wj-glyph-down	Styles the down-arrow icon displayed in the column header of a column sorted in descending order.
Grouping	
.wj-group	Styles the group row.
.wj-glyph-down-right	Styles the expand icon displayed in the group row for the expanded group.
.wj-glyph-right	Styles the collapse icon displayed in the group row for the collapsed group.
.wj-grouppanel	Styles the div host element for GroupPanel control.
.wj-groupmarker	Styles the div element for each grouped column displayed in the GroupPanel control.
.wj-remove	Styles the (x) mark displayed in the group panel for each column header element.
Filtering	
.wj-glyph-filter	Styles the filter icon displayed in column headers.
.wj-filter-on	Styles the filter icon when the column has been filtered.
.wj-filter-off	Styles the filter icon when the column has not been filtered.
.wj-columnfiltereditor	Styles the div host element for FilerDialog.
.wj-valuefilter-editor	Styles the div host element for Value Filter.
.wj-conditionfilter-editor	Styles the div host element for Condtion Filter.
Drag Drop	
.wj-marker	Styles the marker displayed in the row/column header of the row/column where the dragged row/column will be dropped.
.wj-state-dragsrc	Styles the target row/column which is being dragged.
RowDetail	
.wj-glyph-plus	Styles the Plus icon displayed in the row headers of the collapsed detail row.
.wj-glyph-minus	Styles the Minus icon displayed in the row headers for the

Copyright © GrapeCity, Inc. All rights reserved.

	expanded detail row.

スタイルの適用

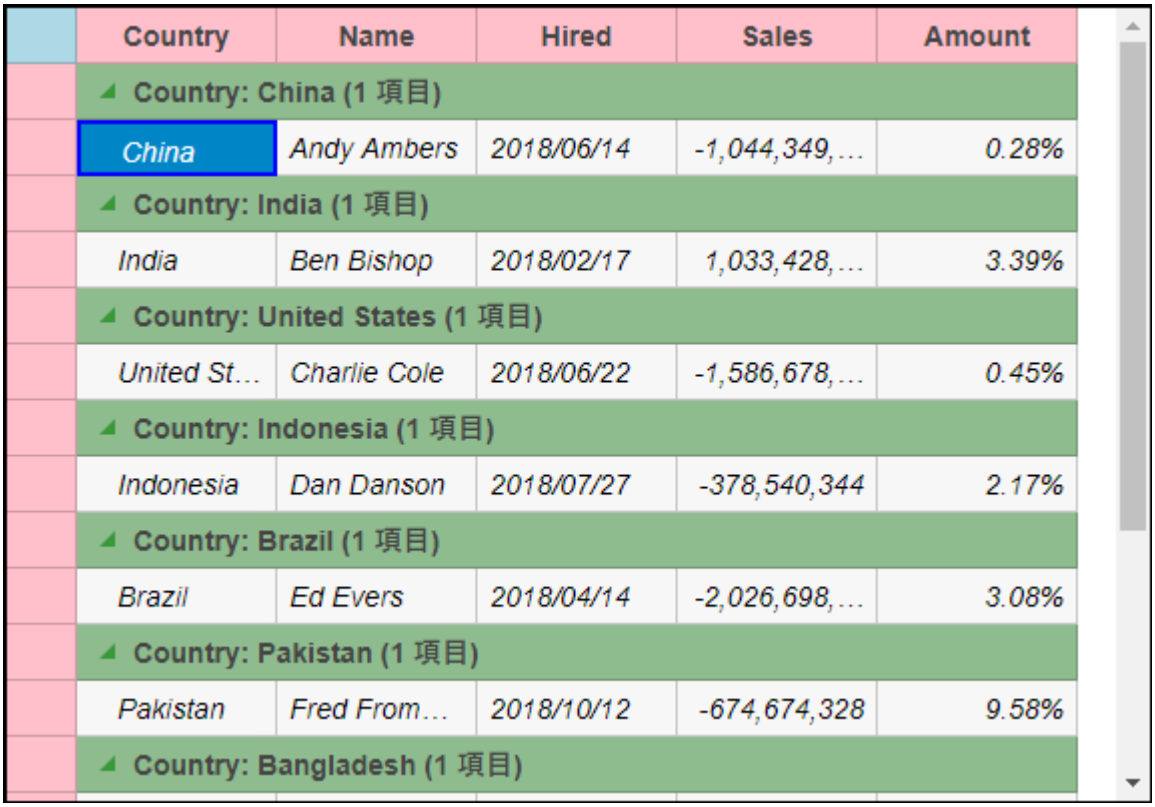
The appearance of the FlexGrid control is defined using CSS classes which represent different grid elements such as row, column, cell etc. You can customize these classes to change the appearance of the FlexGrid control.

In this example, we customize the default CSS rules to make the following customization to the grid's appearance:

1. FlexGrid border.
2. Background color for topLeft, columnHeader, rowHeader cells and GroupRow.
3. ColumnHeader text is centered aligned.
4. Font style italic for alternative rows.
5. Solid border for selected cells.
6. Sort icon color.
7. Collapsed/Expand Group Icon color.
8. Font color for FlexGrid editor.

These custom CSS rules can be applied to the FlexGrid by applying a CSS class to the FlexGrid control using the **CssClass** property. This property applies the CSS rules to the non-header cells in a row or column. Similarly, you can apply CSS rules to all the cells, including headers, in a column or row using **CssClassAll** property.

The following image shows how the FlexGrid control appears after applying styles using custom CSS.



Add Custom Stylesheet

Create a new ASP.NET MVC application. Once you have created the application, a **Content** folder is created in the **Solution Explorer** after adding the view to the application. To add a custom style sheet in your application, follow these steps:

1. In the Solution Explorer, right-click the **Content** folder.
2. From the context menu, select **Add | Style Sheet**. The **Specify Name for Item** dialog appears.

Copyright © GrapeCity, Inc. All rights reserved.

3. Set name of the style sheet (for example: `app.css`) and click **OK**.
4. Replace the default code of **app.css** file with the code given below.

app.css

```
.custom.wj-flexgrid{
        border: black 2px solid;
        border-radius: 0;
        margin-top:15px;
    }

    .custom .wj-topleft .wj-cell {
        background: lightblue;
    }
    .custom .wj-colheaders .wj-cell {
        background: pink;
        text-align:center !important;
    }
    .custom .wj-rowheaders .wj-cell {
        background: pink;
    }
    .custom .wj-alt{
        font-style:italic;
    }

    .custom .wj-group:not(.wj-state-selected):not(.wj-state-multi-selected) {
        font-weight: bold;
        background-color: darkseagreen;
    }
    .custom .wj-glyph-down-right{
        color:green;
    }
    .custom .wj-glyph-right{
        color:green;
    }
    .custom .wj-state-selected{
        border: blue 2px solid;

    }
    .custom .wj-state-multi-selected {
        border: blue 2px solid;
    }
    .custom .wj-grid-editor{
        color:blue;
    }
    .custom .wj-glyph-up {
        color: blueviolet;
    }
    .custom .wj-glyph-down{
        color:blueviolet;
    }
```

Styling.cshtml

Copyright © GrapeCity, Inc. All rights reserved.

Razor

```
@model IEnumerable<Person>
<div>
    <div style="width:90%;margin:0 auto;">
        @(Html.C1().FlexGrid()
            .Id("flexGrid")
            .CssClass("custom")
            .Width("575px")
            .Height("900px")
            .KeyActionTab(C1.Web.Mvc.Grid.KeyAction.Cycle)
            .Bind(b => b.Bind(Model).GroupBy("Country"))
            .Height(400)
            .AutoGenerateColumns(false)
            .Columns(c =>
            {
                c.Add().Binding("Country").Header("Country").Width("100");
                c.Add().Binding("Name").Header("Name").Width("100");

c.Add().Binding("Hired").Header("Hired").Width("100").IsRequired(false);
                c.Add().Binding("Sales").Format("n").Width("100");
                c.Add().Binding("Amount").Format("p2").Width("100");
            })
        )
    </div>
</div>
```

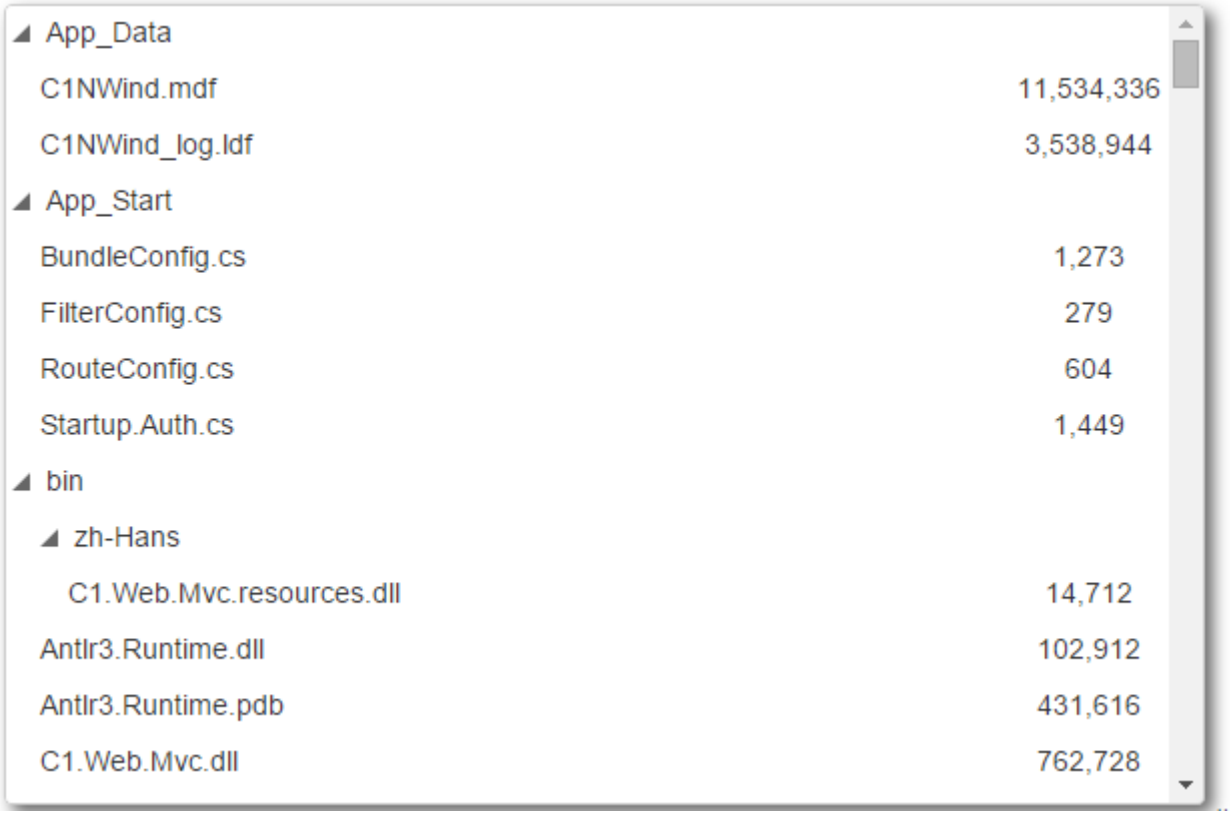
TreeGrid

FlexGrid は、それぞれの項目がサブ項目のリストから成る階層化データをサポートします。階層化データソースを保持するツリービューとして FlexGrid を使用するには、**ChildItemsPath** プロパティを子要素を含むデータ要素の名前に設定します。これでデータが自動的にスキャンされて、FlexGrid にツリーが構築されます。

次の例では、ASP.NET MVC プロジェクトのサンプルフォルダ構造を使用します。**ChildItemsPath** プロパティに任意のフォルダのパスを提供することで、そのフォルダのツリービュー構造を表示できます。

次の図は、**ChildItemsPath** プロパティを設定した後の FlexGrid を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexGrid でツリービューを有効にする方法を示します。

コードの場合

モデルの追加

1. **[モデル]**フォルダに新しいクラスを追加します（例：`TreeItem.cs`）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、サンプルフォルダのツリービュー構造を定義します。

TreeItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MVCFlexGrid.Models
{
    public interface ITreeItem
    {
        string Header { get; set; }
        IList<ITreeItem> Children { get; }
    }
    public class Folder : ITreeItem
    {
        public string Header { get; set; }
        public IList<ITreeItem> Children { get; private set; }

        public Folder(string name)
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            Header = name;
            Children = new List<ITreeItem>();
        }
        public static Folder Create(string path)
        {
    var folder = new Folder(System.IO.Path.GetFileName(path));
    System.IO.Directory.GetDirectories(path).ToList()
        .ForEach(d => folder.Children.Add(Folder.Create(d)));
    System.IO.Directory.GetFiles(path).ToList()
        .ForEach(f => folder.Children.Add(File.Create(f)));
    return folder;
        }
    }
    public class File : ITreeItem
    {
        public string Header { get; set; }
        public DateTime DateModified { get; set; }
        public long Size { get; set; }
        public IList<ITreeItem> Children { get { return null; } }

        public File(string name)
        {
            Header = name;
        }
        public static File Create(string path)
        {
        var file = new File(System.IO.Path.GetFileName(path));
        var info = new System.IO.FileInfo(path);
        file.DateModified = info.LastWriteTime;
        file.Size = info.Length;
        return file;
        }
    }
}
```

先頭に戻る

コントローラーおよびビューの追加

TreeViewController.cs

C# copyCode

```
public ActionResult Index()
{
    var list = MVCFlexGrid_JP.Models.Folder
    .Create(Server.MapPath("~")).Children;
    return View(list);
}
```

TreeView.cshtml

Razor copyCode

Copyright © GrapeCity, Inc. All rights reserved.

```
@using C1.Web.Mvc.Grid
@model IEnumerable<MVCFlexGrid_JP.Models.ITreeItem>

<style>
    .wj-flexgrid {
     height: 400px;
     background-color: white;
     box-shadow: 4px 4px 10px 0px rgba(50, 50, 50, 0.75);
     margin-bottom: 12px;
    }

    .custom-flex-grid .wj-header.wj-cell {
        color: #fff;
        background-color: #000;
        border-bottom: solid 1px #404040;
        border-right: solid 1px #404040;
        font-weight: bold;
    }

    .custom-flex-grid .wj-cell {
        background-color: #fff;
        border: none;
    }

    .custom-flex-grid
   .wj-alt:not(.wj-state-selected)
    :not(.wj-state-multi-selected)
      {
        background-color: #fff;
    }

    .custom-flex-grid .wj-state-selected {
        background: #000;
        color: #fff;
    }

    .custom-flex-grid .wj-state-multi-selected {
        background: #222;
        color: #fff;
    }
</style>

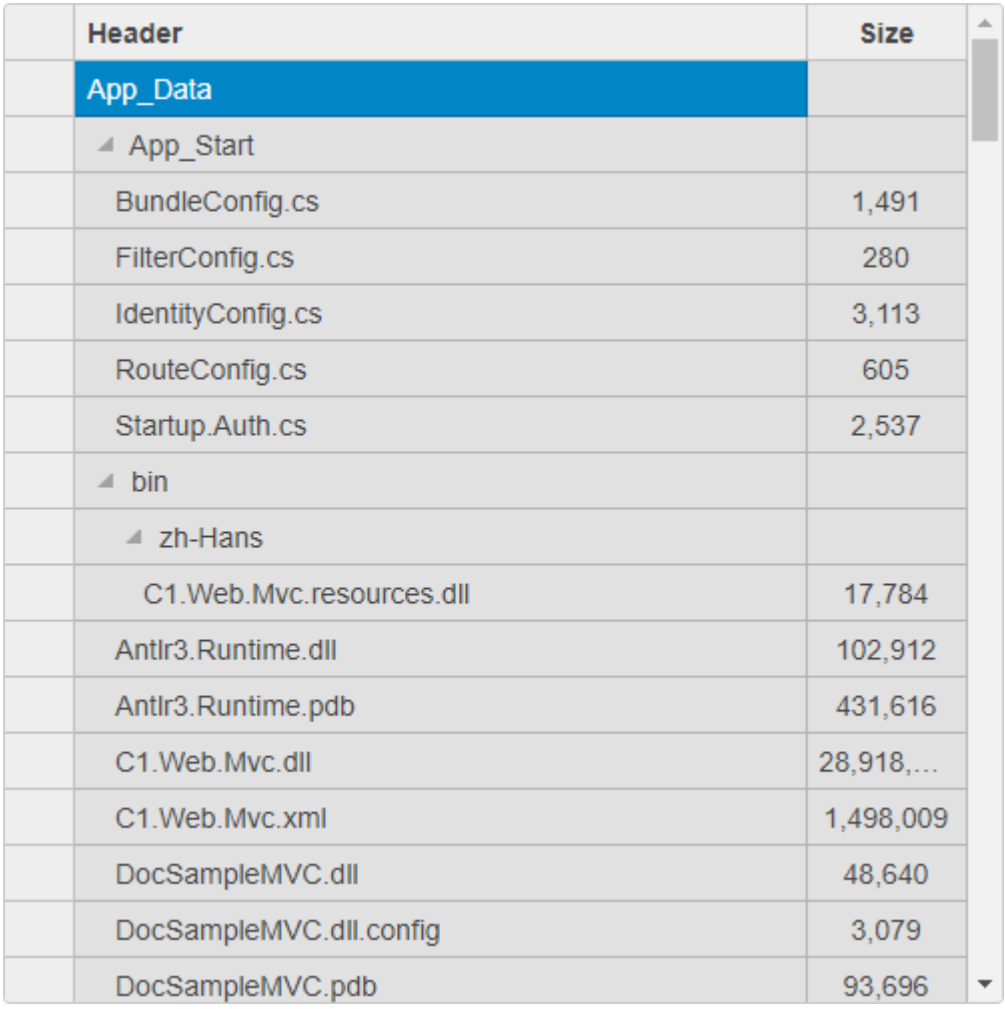
@(Html.C1().FlexGrid().CssClass("custom-flex-grid")
    .Bind(Model)
    .Width(600)
    .ChildItemsPath("Children")
    .AutoGenerateColumns(false)
    .Columns(columns =>
    {
        columns.Add().Binding("Header").Width("*");
        columns.Add().Binding("Size").Width("80")
    .Align("center");
    })
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .AllowResizing(AllowResizing.None)
    .HeadersVisibility(HeadersVisibility.None)
    .SelectionMode(SelectionMode.ListBox)
)
```

編集可能なTreeGrid

When you use FlexGrid's **ChildItemsPath** to show the data as a tree, the resulting grid will be read-only by default. This happens because every row in a tree is a GroupRow, and group rows are read-only by default. However, you can make the tree editable by handling the OnClientLoadedRows event to set the **isReadOnly** property of every row to **false**.



To make the TreeGrid editable, use the following code. The example uses **TreeItem.cs** model which was added to the application in the TreeGrid topic:

Controller code

EditableTreeGridController.cshtml copyCode

```
public ActionResult EditableTreeGrid()
{
    var list = Folder.Create(Server.MapPath("~")).Children;
    return View(list);
}
```


Copyright © GrapeCity, Inc. All rights reserved.

View code

EditableTreeGrid.razor copyCode

```
@model IEnumerable<ITreeItem>

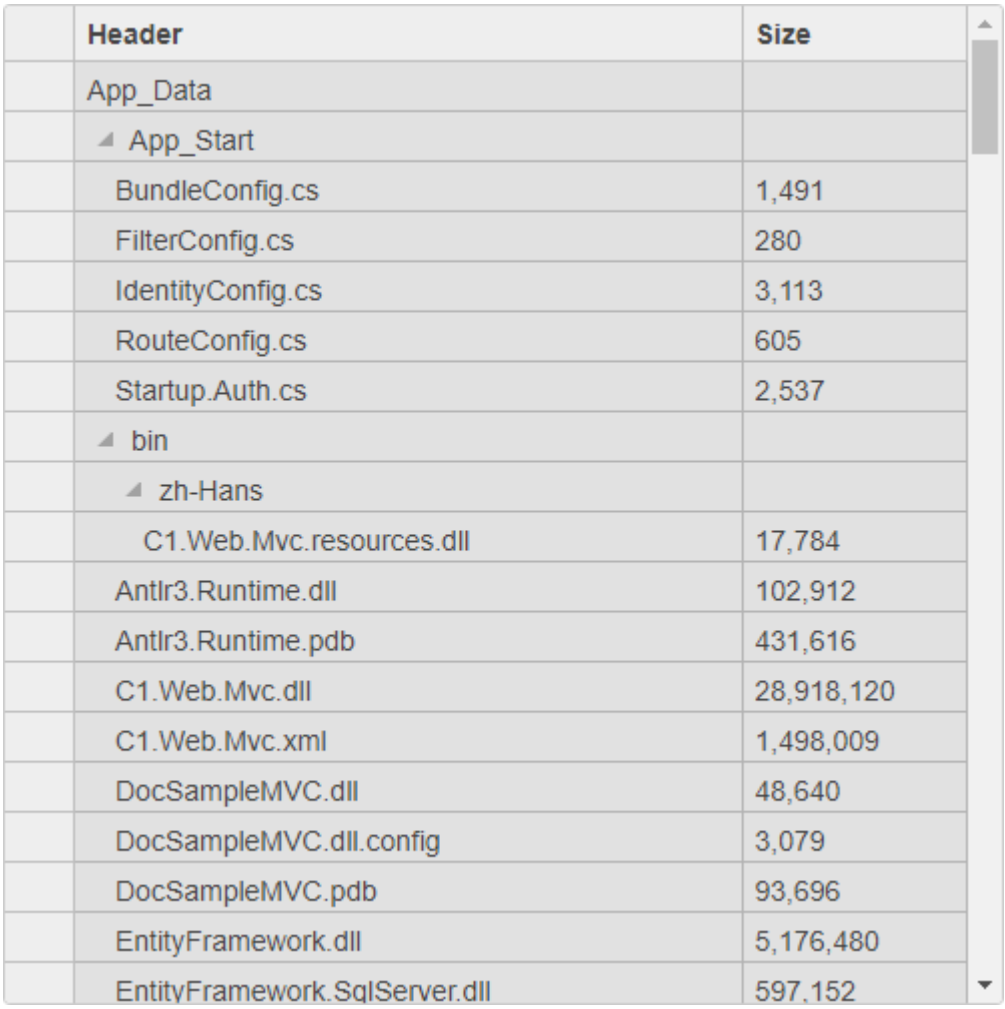
@(Html.C1().FlexGrid().Id("grid").Width(500).Height(500)
    .Bind(Model)
    .ChildItemsPath("Children")
    .AutoGenerateColumns(false)
    .Columns(columns =>
    {
        columns.Add().Binding("Header").Width("*");
        columns.Add().Binding("Size").Width("80").IsReadOnly(true).Align("center");
    })
    .OnClientLoadedRows("setEditableRows")
)

<script>
    function setEditableRows(s, e) {
        s.rows.forEach(function (row) {
            row.isReadOnly = false;
        });
    }
</script>
```

非連結のTreeGrid

As the name suggests, in unbound mode, grid is not bound to any data source and data is stored in the control itself. In this case, you can build trees by adding rows and columns in code.

Copyright © GrapeCity, Inc. All rights reserved.



To work in unbound mode, you need to start with a data source in an array, initialize the FlexGrid without binding it to the array, add columns and define properties, loop through the arrays and add group rows and cells as shown in the following code. The example uses **TreeItem.cs** model which was added to the application in the TreeGrid topic:

Controller code

UnboundTreeGridController.cshtml copyCode

```
public ActionResult UnboundTree()
{
    var list = Folder.Create(Server.MapPath("~")).Children;
    return View(list);
}
```

View code

UnboundTreeGrid.razor copyCode

```
@model IEnumerable<ITreeItem>

@(Html.C1().FlexGrid().Id("ubgrid").Width(500).Height(500)
    .AutoGenerateColumns(false)
)

<script>
```


Copyright © GrapeCity, Inc. All rights reserved.

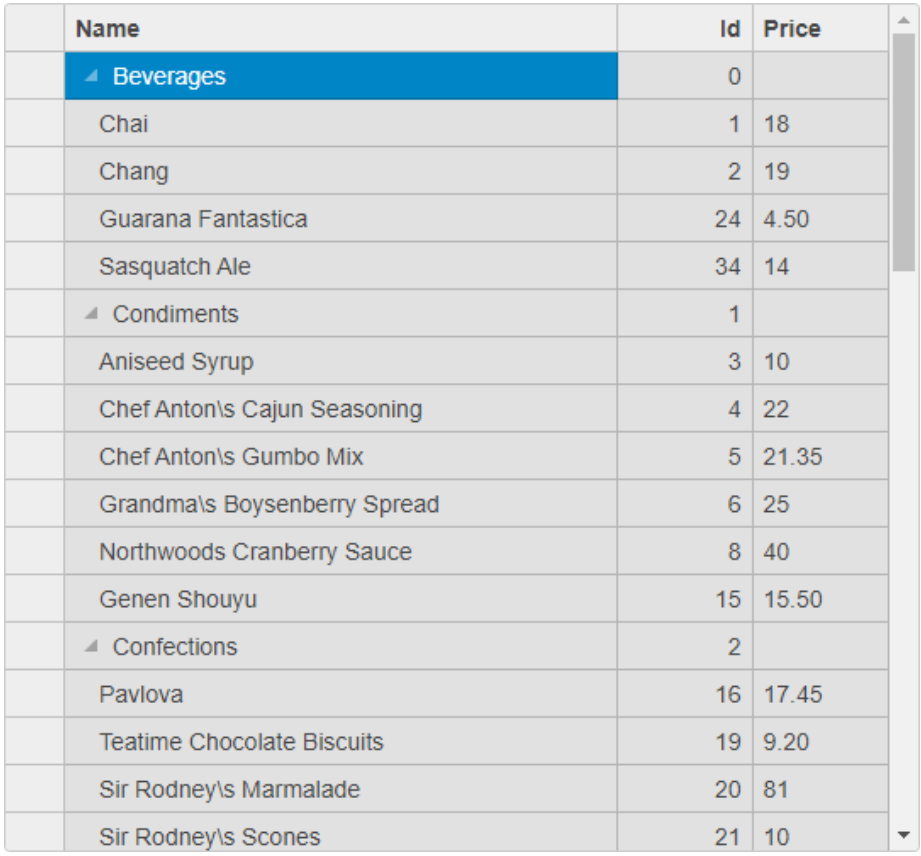
```
    c1.documentReady(function () {
        let grid = wijmo.Control.getControl("#ubgrid");
        grid.rows.defaultSize = 25;
        // 列を追加します。
        grid.columns.push(new wijmo.grid.Column({ header: 'Header', width: '2*' }));
        grid.columns.push(new wijmo.grid.Column({ header: 'Size' }));

        let data = @(Html.Raw(Json.Encode(Model)));
        // 行を追加します。
        for (let r = 0; r < data.length; r++) {
            // ヘッダを追加します。
            var header = data[r];
            var row = new wijmo.grid.GroupRow();
            row.dataItem = header;
            row.isReadOnly = false;
            row.level = 0;
            grid.rows.push(row);
            grid.setCellData(row.index, 0, header.Header);
            if (header.Children) {
                addChild(grid, header, 1);
            }
        }
    });
    function addChild(grid, parent, level) {
        for (var c = 0; c < parent.Children.length; c++) {
            // 子を追加します。
            var child = parent.Children[c];
            row = new wijmo.grid.GroupRow();
            row.dataItem = child;
            row.isReadOnly = false;
            row.level = level;
            grid.rows.push(row);
            grid.setCellData(row.index, 0, child.Header);
            grid.setCellData(row.index, 1, child.Size);
            if (child.Children) {
                addChild(grid, child, level + 1);
            }
        }
    }
</script>
```

XMLへの連結

You can use XML documents as a hierarchical data source for the FlexGrid control and display the data as tree as shown in the following image.

Copyright © GrapeCity, Inc. All rights reserved.



	Name	Id	Price
	Beverages	0	
	Chai	1	18
	Chang	2	19
	Guarana Fantastica	24	4.50
	Sasquatch Ale	34	14
	Condiments	1	
	Aniseed Syrup	3	10
	Chef Anton\s Cajun Seasoning	4	22
	Chef Anton\s Gumbo Mix	5	21.35
	Grandma\s Boysenberry Spread	6	25
	Northwoods Cranberry Sauce	8	40
	Genen Shouyu	15	15.50
	Confections	2	
	Pavlova	16	17.45
	Teatime Chocolate Biscuits	19	9.20
	Sir Rodney\s Marmalade	20	81
	Sir Rodney\s Scones	21	10

The following example shows how you can use XML documents as a hierarchical data source for the FlexGrid control. It uses **XElement.Load** method to load the XML element from file that contains the content of the XML document and loops through the XML element to build an array with "category" items, each with a "products" array. The array is used as an ItemsSource and the ChildItemsPath property is used to show the products for each category as a tree. The example uses **ProductCategories.xml** file as a data source.

Controller code

TreeGridXMLController.cshtml copyCode

```
public ActionResult TreeGridXML()
{
    return View();
}

public ActionResult GetProductsByCategory([C1JsonRequest] CollectionViewRequest<TCategory>
requestData)
{
    var items = new List<TCategory>();
    var xml =
XElement.Load(System.Web.HttpContext.Current.Server.MapPath("~/Content/data/ProductCategories.xml"));
    // カテゴリを取得します。
    var categories = xml.Elements("category");
    foreach (var cg in categories)
    {
        items.Add(new TCategory
        {
            Id = Convert.ToInt32(cg.Attribute("id").Value),
            Name = cg.Attribute("name").Value,
            Products = new List<TProduct>()
        });
        // このカテゴリの製品を取得します。
        var products = cg.Elements("product");
        foreach (var p in products)
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            items[items.Count - 1].Products.Add(new TProduct
            {
                Id = Convert.ToInt32(p.Attribute("id").Value),
                Name = p.Attribute("name").Value,
                Price = Convert.ToDouble(p.Attribute("price").Value)
            });
        }
    }
    return this.C1Json(CollectionViewHelper.Read(requestData, items));
}

public class TCategory
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<TProduct> Products { get; set; }
}

public class TProduct
{
    public int Id { get; set; }
    public string Name { get; set; }
    public double Price { get; set; }
}
```

View code

TreeGridXML.razor copyCode

```
@(Html.C1().FlexGrid().Id("grid").Width(700).Height(500)
    .Bind("GetProductsByCategory")
    .ChildItemsPath("Products")
    .AutoGenerateColumns(false)
    .Columns(columns =>
    {
        columns.Add().Binding("Name").Width("*");
        columns.Add().Binding("Id").Width("80");
        columns.Add().Binding("Price").Width("80");
    })
)
```

非連結のFlexGrid

FlexGrid allows you to populate data in unbound mode, at client-side using the following ways.

AJAX呼び出しを使用した非連結のFlexGrid
: Learn how to populate data in an unbound FlexGrid at client side using AJAX.

CollectionViewServiceを使用した非連結のFlexGrid
: Learn how to set the data from the client side using CollectionView Service.

AJAX呼び出しを使用した非連結のFlexGrid

FlexGridが連結モードの場合、グリッド内のデータは特定のデータソースから自動的に挿入されます。しかし、非連結FlexGridの場合は、APIで提供されている別の方法を使用してグリッドの列と行を追加/削除する必要があります。

このトピックでは、AJAXを使用して、クライアント側で非連結FlexGridにデータを挿入する方法を説明します。サーバーからのデータは辞書に保存され、クライアントに送信される前にJSONにシリアライズされます。

このトピックは次の手順で構成されます。

Copyright © GrapeCity, Inc. All rights reserved.

- **手順1：MVCアプリケーションの設定**
- **手順2：モデルの作成**
- **手順3：コントローラーアクションの追加**
- **手順4：ビューへのFlexGridの追加**
- **手順5：AJAXを使用したFlexGridでのデータ生成**
- **手順6：プロジェクトのビルドおよび実行**

次の図は、AJAXを使用してクライアント側で非連結FlexGridにデータを挿入する方法を示しています。この例では、「クイックスタート」セクションで追加した**Sale.cs**モデルを使用します。

AJAX Load

	Product	Country	Amount
	Gadget	German	581.61
	Gadget	Canada	4,919.02
	Gadget	Japan	2,159.73
	Gadget	German	1,248.66
	Gadget	German	4,051.76
	Gadget	Canada	-3,131.28
	Widget	China	698.62
	Widget	US	3,464.15
	Gadget	Korea	-2,363.16
	Widget	US	-2,836.94

手順1：MVCアプリケーションの設定

1. Visual Studioで、**ComponentOne ASP.NET MVC 5 Web Application**テンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションを作成する方法の詳細については、「ComponentOneテンプレートの使用」を参照してください。

2. **ComponentOne ASP.NET MVCアプリケーションウィザード**で、**[クライアントIntelliSenseの有効化]**オプションと**[TSを追加]**オプションをオンにします。

先頭に戻る

手順2：モデルの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Sale.cs`）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. `Sale.cs`モデル内の次のコードを置き換えます。ここでは、**Sale**クラスを使用して、データベース内の受注データを表しています。**Sale**オブジェクトの各インスタンスがFlexGridコントロール内の1つのレコードに対応します。

C#

```
public class Sale
  {
  public int ID { get; set; }
  public DateTime Start { get; set; }
  public DateTime End { get; set; }
  public string Country { get; set; }
  public string Product { get; set; }
  public string Color { get; set; }
  public double Amount { get; set; }
  public double Amount2 { get; set; }
  public double Discount { get; set; }
  public bool Active { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
  public MonthData[] Trends { get; set; }
  public int Rank { get; set; }

private static List <string> COUNTRIES = new List<string> { "米国", "イギリス", "カナダ", "日本", "中国", "フランス", "ドイツ", "イタリア", "韓国", "オーストラリア" };
private static List<string> PRODUCTS = new List<string> { "Widget", "Gadget", "Doohickey" };
        /// <summary>
        /// データを取得
        /// </summary>
        /// <param name="total"></param>
        /// <returns></returns>
        public static IEnumerable<Sale> GetData(int total)
        {
            var colors = new[] { "黒色", "白色", "赤色", "緑色", "青い色" };
            var rand = new Random(0);
            var dt = DateTime.Now;
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var color = colors[rand.Next(0, colors.Length - 1)];
                var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
                var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60, i % 60);

                return new Sale
                {
                    ID = i + 1,
                    開始日 = startDate,
                    終了日 = endDate,
                    国名 = country,
                    製品名 = product,
                    色 = color,
                    金額 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    金額2 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    割引 = Math.Round(rand.NextDouble() / 4, 2),
                    アクティブ = (i % 4 == 0),
                    傾向 = Enumerable.Range(0, 12).Select(x => new MonthData { Month = x + 1, Data = rand.Next(0, 100) }).ToArray(),
                    ランク = rand.Next(1, 6)
};
            });
            return list;
        }
        public static List<string> GetCountries()
        {
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }
        public static List<string> GetProducts()
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    }
    public class MonthData
    {
        public int Month { get; set; }
        public double Data { get; set; }
    }
    public class BasicSale
    {
        public int Sale { get; set; }
        public DateTime Date { get; set; }

        public BasicSale(int sale, DateTime date)
        {
            Sale = sale;
            Date = date;
        }
        public static List<BasicSale> GetBasicSales()
        {
            List<BasicSale> list = new List<BasicSale>();
            int[] sales = {
            96, 19, 54, 83, 15, 56, 36, 4, 29, 93,
            38, 71, 50, 77, 69, 13, 79, 57, 29, 62,
            4, 27, 66, 96, 65, 12, 52, 3, 61, 48, 50,
            70, 39, 33, 25, 49, 69, 46, 44, 40, 35,
            72, 64, 10, 66, 63, 78, 19, 96, 26};
            for (int i = 0; i < sales.Length; i++)
            {
                list.Add(new BasicSale(sales[i], new DateTime(2014, i / 31 + 1, i % 31 +
1)));
            }
            return list;
        }
    }
```

先頭に戻る

手順3：コントローラーアクションの追加

1. [コントローラー]フォルダからHomeController.csを開きます。
2. 次のコードを追加して、**GetDictionaryData**関数で、一意のキーと売上オブジェクトを持つ辞書を作成します。また、**GetData**関数で、JavaScriptSerializerを使用して辞書データをJSONにシリアライズする必要があります。

C#

```
public ActionResult Index()
  {
  return View();
  }
  [HttpPost]
  public JsonResult GetData()
  {
  // return Json(JsonConvert.SerializeObject(GetDictionaryData()));
  JavaScriptSerializer sz = new JavaScriptSerializer();
  string str = sz.Serialize(GetDictionaryData());
  return Json(str);
  }
  private Dictionary<string,Sale> GetDictionaryData()
  {
  var dict = new Dictionary<string, Sale>();
  var sales = Sale.GetData(10);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
  for(int i=0; i<=sales.Count()-1;i++)
  {
  dict.Add(i.ToString(), sales.ElementAt(i));
  }
  return dict;
  }
```

先頭に戻る

手順4：ビューへのFlexGridの追加

1. [View/Home]フォルダからIndex.htmlを開きます。
2. このファイルの内容を、FlexGridコントロールを設定する次のコードに置き換えます。以下のコードは、FlexGridのさまざまな列やプロパティを宣言していますが、データをコントロールには連結していません。

Razor

```
@using <ApplicationName>.Models
<script src="~/Scripts/app.js"></script>

@* 辞書からの読み込みます *@
<input type="button" onclick="Load()" value="Ajaxのロード" />

@(Html.C1().FlexGrid().Id("fg").Height(400).AutoGenerateColumns(false).AllowAddNew(false)
  .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
  .Columns( col => {
  col.Add(cb => cb.Name("Product").Header("製品名"));
  col.Add(cb => cb.Name("Country").Header("国名"));
  col.Add(cb => cb.Name("Amount").Header("金額"));
}))
```

先頭に戻る

手順5：AJAXを使用したFlexGridでのデータ生成

1. **ソリューションエクスプローラー**で、**[Scripts]**フォルダから**app.ts**ファイルを開きます。
2. **app.ts**ファイルに次のスクリプトを追加して、クライアント側でFlexGridコントロールにデータを挿入します。

app.ts

```
/// <reference path="typings/c1.mvc.basic.lib.d.ts" />
function Load() {
    $.ajax({
        type: "POST",
        url: "/Home/GetData",
        dataType: "json",
        success: function (result) {
            var fg = <wijmo.grid.FlexGrid>wijmo.Control.getControl("#fg"); //FlexGridのリファレンスを取得します
            var data = JSON.parse(result);  //サーバーの辞書データをJSON配列に解析します
            fg.rows.clear();  //FlexGridから既存の行をクリアします
         var j=0
            for (var i in data) {
                var obj = [data[i].Product, data[i].Country, data[i].Amount];
                var row = new wijmo.grid.Row();  // FlexGridに行を追加します
                fg.rows.push(row);
                for (var col = 0; col <= fg.columns.length - 1; col++) {
                    fg.setCellData(j, col, obj[col]);  //FlexGridセルにデータを追加します
                }
                j++;
            }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        },
        error: function (err) {
            alert("err");
        }
    });
```

先頭に戻る

手順6：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行した後、**[AJAX Load]**ボタンをクリックして、FlexGridにデータをロードして表示します。

先頭に戻る

CollectionViewServiceを使用した非連結のFlexGrid

FlexGrid for MVCは、非連結モードをサポートしませんが、JavaScriptまたはTypeScriptを使用してデータを設定するコードをクライアント側に追加することでこれを実現できます。

それには、モデルに連結する**CollectionViewService**を使用します。クライアント側でCollectionViewServiceの参照を取得し、**Sale.cs**モデル内の項目の数に合わせてFlexGridに行を追加します。最後に、3列のデータを取得し、**setCellData**メソッドを使用してFlexgridに追加します。

次の画像は、CollectionViewServiceを使用してクライアント側からデータを設定する方法を示しています。この例では、「クイックスタート」セクションで追加したSale.csモデルを使用します。

	製品名	国名	金額
	Gadget	ドイツ	¥582
	Gadget	カナダ	¥4,919
	Gadget	日本	¥2,160
	Gadget	ドイツ	¥1,249
	Gadget	ドイツ	¥4,052
	Gadget	カナダ	-¥3,131
	Widget	中国	¥699
	Widget	米国	¥3,464
	Gadget	韓国	-¥2,363
	Widget	米国	-¥2,837

次のコード例は、クライアント側にコードを追加してデータを設定する方法を示します。

App.ts（TypeScriptファイル）

App.ts

```
var fg;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
c1.documentReady(function () {
    populate();
})
function populate() {
    var fg = <wijmo.grid.FlexGrid>wijmo.Control.getControl("#fg");
    var cv = <wijmo.collections.CollectionView>c1.getService('cv');
    var total = cv.items.length;

    for (var i = 0; i <= total - 1; i++) {
        var obj = [cv.items[i].Product, cv.items[i].Country, cv.items[i].Amount];
        var row = new wijmo.grid.Row();
        fg.rows.push(row);
        for (var c = 0; c <= fg.columns.length - 1; c++) {
            fg.setCellData(i, c, obj[c]);
        }
    }
}
```

Index.cshtml

Razor

```
@using UnboundFlexGrid.Models
<script src="~/Scripts/app.js"></script>
@model IEnumerable<Sale>

@(Html.C1().CollectionViewService().Id("cv").Bind(Model))

@(Html.C1().FlexGrid().Id("fg").Height(400).AutoGenerateColumns(false)
.AllowAddNew(false)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .Columns( col => {
        col.Add(cb => cb.Name("Product").Header("製品名"));
        col.Add(cb => cb.Name("Country").Header("国名"));
        col.Add(cb => cb.Name("Amount").Header("金額"));
        }))
```

控えめな検証

FlexGridは、控えめな検証をサポートしており、クライアント側検証によってFlexGridの行に入力されたデータを検証することができます。控えめな検証は、大量の検証コードを書かずにシンプルなクライアント側検証を実装し、正しい属性を追加してスクリプトファイルをインクルードするだけで、ユーザーエクスペリエンスを向上させることができます。

任意のコントロールを検証する検証やクライアント側の検証を使用する一般的な検証シナリオでは、JavaScriptコードが生成されて、WebブラウザでHTMLとしてレンダリングされます。一方、控えめな検証では、クライアント側の検証を処理するためにレンダリングされるインラインJavaScriptは生成されません。その代わりに、HTML5 data-*属性をクライアント側検証に使用します。控えめな検証のサポートは、jQueryに依存します。

以下の手順を実装する前に、ComponentOneテンプレートまたはVisual Studioテンプレートを使用して、新しいMVCアプリケーションを作成する必要があります。

このトピックは4つの手順で構成されます。

- **手順1：MVCアプリケーションの構成**

Copyright © GrapeCity, Inc. All rights reserved.

- **手順2：FlexGridの検証とデータソースの作成**
- **手順3：FlexGridコントロールの追加**
- **手順4：プロジェクトのビルドおよび実行**



手順1：MVCアプリケーションの構成

HTML Helpers

次の手順を実行して、アプリケーションにjsファイルの参照を追加します。

1. **ソリューションエクスプローラー**で、**[ビュー]→[共有]**フォルダを開きます。
2. **_Layout.cshtml**をダブルクリックして開きます。
3. <head> </head>タグの間に以下のコードを追加します。

Razor

```
<script src="~/Scripts/jquery-1.10.2.js"></script>
<script src="~/Scripts/jquery.validate.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.js"></script>
```

手順2：FlexGridの検証とデータソースの作成

モデル - UserInfo.cs（検証を含む）

Razor

```
using System.ComponentModel.DataAnnotations;namespace UnobtrusiveValidation.Models{
public class UserInfo    {        public UserInfo() {            Birthdate =
DateTime.Now;        }        [Required]        [RegularExpression(pattern: "^[a-zA-
Z0-9]{4,10}$", ErrorMessage = "The username must be alphanumeric and contains 4 to 10
characters.")]        public string Name { get; set; }        [Required]
[RegularExpression(pattern: @"^\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$",
ErrorMessageResourceType = typeof(Resources.Validation), ErrorMessageResourceName =
"Register_Email_ErrorMessage")]        [EmailAddress]        public string Email {
get; set; }        [Required]        [MinLength(6)]        [MaxLength(16)]
public string Phone { get; set; }        [Required]        public string Country {
get; set; }        [Required]        public string Industry { get; set; }
[Required]        public DateTime Birthdate { get; set; }    }}
```

先頭に戻る

Copyright © GrapeCity, Inc. All rights reserved.

モデル - UserData.cs（データを含む）

```
Razor
public class UserData    {        public static List<UserInfo> Users        {
get            {                return new List<UserInfo>()                {
new UserInfo(){ Name="John", Email="John@gmail.com", Phone="1424685445",
Country="Albania", Industry="Computers", Birthdate= DateTime.Parse("2001/1/1")},
new UserInfo(){ Name="Mary", Email="Mary@gmail.com", Phone="1296479754",
Country="American", Industry="Electronics", Birthdate= DateTime.Parse("1985/3/2")},
new UserInfo(){ Name="David", Email="David@gmail.com", Phone="1217654653",
Country="Australia", Industry="Telecom", Birthdate= DateTime.Parse("1999/3/1")},
new UserInfo(){ Name="Sunny", Email="Sunny@gmail.com", Phone="1756456786",
Country="Bosnia", Industry="Internet", Birthdate= DateTime.Parse("1989/4/3")},
new UserInfo(){ Name="James", Email="James@gmail.com", Phone="1209687543",
Country="Botswana", Industry="Accounting", Birthdate= DateTime.Parse("1994/3/2")},
new UserInfo(){ Name="Maria", Email="Maria@gmail.com", Phone="1543578643",
Country="Bahrain", Industry="Accounting", Birthdate= DateTime.Parse("1998/4/2")},
new UserInfo(){ Name="Michael", Email="Michael@gmail.com", Phone="1215457467",
Country="Argentina", Industry="Finance", Birthdate= DateTime.Parse("2003/2/2")},
new UserInfo(){ Name="Michelle", Email="Michelle@gmail.com", Phone="1534357546",
Country="Bulgaria", Industry="Finance", Birthdate= DateTime.Parse("2001/1/1")}
};            }        }    }
```

先頭に戻る

手順3：FlexGridコントロールの追加

UnobtrusiveValidationController.cs

```
Razor
public class IndexController : Controller    {        private static List<UserInfo>
users = UserData.Users;        public IActionResult Index()        {
return View(users);        }    }
```

View - Index.cshtml

```
Razor
@using FlexGridValidationsHTMLHelpers.Models;@model List<UserInfo>
@section Scripts{    @Scripts.Render("~/jquery")    @Scripts.Render("~/jqueryval")}
@(Html.C1().FlexGrid<UserInfo>()    .Id("flexGrid")    .AutoGenerateColumns(false)
.Columns(columns => columns        .Add(c => c.Binding("Name"))        .Add(c =>
c.Binding("Email"))        .Add(c => c.Binding("Phone"))        .Add(c =>
c.Binding("Country"))        .Add(c => c.Binding("Industry"))        .Add(c =>
c.Binding("Birthdate").Format("M/d/yyyy"))    )    .Bind(ib => ib.Bind(Model))
.AllowAddNew(true)    .AllowDelete(true)    .CssStyle("height", "400px")
```

先頭に戻る

手順4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

Copyright © GrapeCity, Inc. All rights reserved.

先頭に戻る

制限

- FlexGridの控えめな検証は、チェックボックスのbooleanデータをサポートしません。クリックすると、編集が直ちに終了するからです。
- FlexGridの控えめな検証は、読み取りアクションからモデルメタデータを取得できないため、読み取りアクションURLへの連結はサポートしません。
- FlexGridの控えめな検証は、編集中は実際の値を取得できないため、データマップ列はサポートしません。
- FlexGridの控えめな検証は、「onPrepareCellForEdit」がトリガされないため、セルテンプレートをサポートしません。

仮想スクロール

FlexGrid は、大量のデータを使用する際に仮想スクロールをサポートします。モデルや他のデータソースを使用した大規模なデータセットと FlexGrid を簡単に結合し、ちらつきや遅延のないスムーズなスクロールを行うことができます。グリッドを仮想スクロールモードにするには、**DisableServerRead** プロパティを false（デフォルト）に設定する必要があります。**InitialItemsCount** プロパティの値は、0より大きい数値に設定する必要があります。

次の図は、大規模なデータセットに連結された FlexGrid を示しています。この例では、「クイックスタート」セクションで追加した Sale.cs モデルを使用します。

ID	開始日	終了日	国名	製品名	色	金額	金額2	割引	アクテ
1	2015/01/25	2015/01/25	ドイツ	Gadget	緑色	581.61	-2,939.67	0.14	☑
2	2015/02/25	2015/02/25	カナダ	Gadget	緑色	4,919.02	-4,673.75	0.17	☐
3	2015/03/25	2015/03/25	日本	Gadget	赤色	2,159.73	-3,810.42	0.07	☐
4	2015/04/25	2015/04/25	ドイツ	Gadget	赤色	1,248.66	-2,815.93	0.22	☐
5	2015/05/25	2015/05/25	ドイツ	Gadget	黒色	4,051.76	-3,108.76	0.12	☑
6	2015/06/25	2015/06/25	カナダ	Gadget	黒色	-3,131.28	-4,314.81	0.16	☐
7	2015/07/25	2015/07/25	中国	Widget	赤色	698.62	-2,745.97	0.15	☐
8	2015/08/25	2015/08/25	米国	Widget	白色	3,464.15	1,131.58	0.03	☐
9	2015/09/25	2015/09/25	韓国	Gadget	黒色	-2,363.16	3,425.60	0.14	☑
10	2015/10/25	2015/10/25	米国	Widget	白色	-2,836.94	-4,283.10	0.06	☐
11	2015/11/25	2015/11/25	フランス	Widget	緑色	877.93	2,722.02	0.08	☐
12	2015/12/25	2015/12/25	韓国	Widget	赤色	-3,788.14	-3,050.89	0.12	☐
13	2015/01/25	2015/01/25	ドイツ	Gadget	赤色	-2,446.92	-2,351.67	0.12	☑
14	2015/02/25	2015/02/25	ドイツ	Widget	黒色	-4,374.97	-3,899.38	0.18	☐
15	2015/03/25	2015/03/25	日本	Gadget	緑色	1,089.32	-4,102.95	0.16	☐
16	2015/04/25	2015/04/25	中国	Gadget	黒色	1,341.85	532.34	0.08	☐

次のコード例は、FlexGrid で仮想スクロールを有効にする方法を示します。

コードの場合

VirtualScrollingController.cs

C# copyCode

```
public ActionResult Index()
{
    return View(Sale.GetData(1000));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

VirtualScrolling.cshtml

Razor copyCode

```
@using MVCFlexGrid_JP.Models

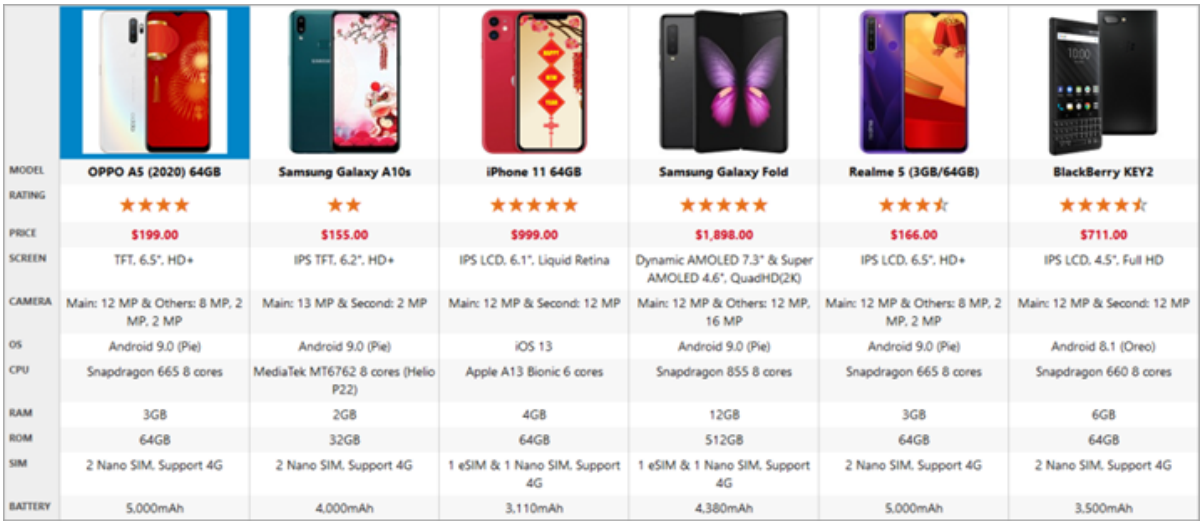
@(Html.C1().FlexGrid<Sale>()
    .Bind(bl => bl.InitialItemsCount(100).Bind(Model))
    .Width(1000)
    .Height(500)
    .CssClass("grid")
)
```

TransposedGrid

In regular grids, each item is represented by a row with columns that represent the item properties. Whereas, in transposed grids, each item is represented by a column with rows that represent the item properties. TransposedGrid is a FlexGrid extension that supports a grid where the rows and columns are transposed.

MVC allows you to work with the transposed grids by using the **TransposedGrid** class. This class allows the grid control to display data using a transposed layout, where columns represent data items and rows represent item properties. In such transposed grids, the column headers appear on the left side of the grid. each item is represented by a column with rows that represent the item properties.

To create an application using a TransposeGrid control for displaying products on the grid columns and their properties on the grid rows, follow these steps:



MODEL	OPPO A5 (2020) 64GB	Samsung Galaxy A10s	iPhone 11 64GB	Samsung Galaxy Fold	Realme 5 (3GB/64GB)	BlackBerry KEY2
RATING	★★★★	★★	★★★★★	★★★★★	★★★½	★★★★½
PRICE	$199.00	$155.00	$999.00	$1,898.00	$166.00	$711.00
SCREEN	TFT, 6.5", HD+	IPS TFT, 6.2", HD+	IPS LCD, 6.1", Liquid Retina	Dynamic AMOLED 7.3" & Super AMOLED 4.6", QuadHD(2K)	IPS LCD, 6.5", HD+	IPS LCD, 4.5", Full HD
CAMERA	Main: 12 MP & Others: 8 MP, 2 MP, 2 MP	Main: 13 MP & Second: 2 MP	Main: 12 MP & Second: 12 MP	Main: 12 MP & Others: 12 MP, 16 MP	Main: 12 MP & Others: 8 MP, 2 MP, 2 MP	Main: 12 MP & Second: 12 MP
OS	Android 9.0 (Pie)	Android 9.0 (Pie)	iOS 13	Android 9.0 (Pie)	Android 9.0 (Pie)	Android 8.1 (Oreo)
CPU	Snapdragon 665 8 cores	MediaTek MT6762 8 cores (Helio P22)	Apple A13 Bionic 6 cores	Snapdragon 855 8 cores	Snapdragon 665 8 cores	Snapdragon 660 8 cores
RAM	3GB	2GB	4GB	12GB	3GB	6GB
ROM	64GB	32GB	64GB	512GB	64GB	64GB
SIM	2 Nano SIM, Support 4G	2 Nano SIM, Support 4G	1 eSIM & 1 Nano SIM, Support 4G	1 eSIM & 1 Nano SIM, Support 4G	2 Nano SIM, Support 4G	2 Nano SIM, Support 4G
BATTERY	5,000mAh	4,000mAh	3,110mAh	4,380mAh	5,000mAh	3,500mAh

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Create a Data Source for TransposeGrid

1. Add a new class to the folder **Models** (for example: `ProductSheet.cs`). See Adding controls to know how to add a new model.
2. Add the following code in the new model to define the class that serve as a data source for the TransposeGrid control. There are few mobile images referred in this class which are added in this path in the sample: Content/images/ProductSheet, and are available at Documents\ComponentOne Samples\ASP.NET

Copyright © GrapeCity, Inc. All rights reserved.

MVC\v4.5.2\MVC\CS\TransposedGridExplorer\TransposedGridExplorer\Content\images\ProductSheet location on your system.

C#

```
public class ProductSheet
{
        public string Name { get; set; }
        public string Image { get; set; }
        public double Rating { get; set; }
        public double Price { get; set; }
        public string Screen { get; set; }
        public string Camera { get; set; }
        public string OS { get; set; }
        public string CPU { get; set; }
        public int RAM { get; set; }
        public int ROM { get; set; }
        public string SIM { get; set; }
        public double Battery { get; set; }

        public static List<ProductSheet> GetData()
        {
                return new List<ProductSheet>() {
                        new ProductSheet() {
                                Name = "OPPO A5 (2020) 64GB",
                                Image = "Content/images/ProductSheet/1.jpg",
                                Rating = 4,
                                Price = 199,
                                Screen = "TFT, 6.5\", HD+",
                                Camera = "Main: 12 MP & Others: 8 MP, 2 MP, 2 MP",
                                OS = "Android 9.0 (Pie)",
                                CPU = "Snapdragon 665 8 cores",
                                RAM = 3,
                                ROM = 64,
                                SIM = "2 Nano SIM, Support 4G",
                                Battery = 5000
                        },
                        new ProductSheet() {
                                Name = "Samsung Galaxy A10s",
                                Image = "Content/images/ProductSheet/2.jpg",
                                Rating = 2,
                                Price = 155,
                                Screen = "IPS TFT, 6.2\", HD+",
                                Camera = "Main: 13 MP & Second: 2 MP",
                                OS = "Android 9.0 (Pie)",
                                CPU = "MediaTek MT6762 8 cores (Helio P22)",
                                RAM = 2,
                                ROM = 32,
                                SIM = "2 Nano SIM, Support 4G",
                                Battery = 4000
                        },
                        new ProductSheet() {
                                Name = "iPhone 11 64GB",
                                Image = "Content/images/ProductSheet/3.jpg",
                                Rating = 5,
                                Price = 999,
                                Screen = "IPS LCD, 6.1\", Liquid Retina",
                                Camera = "Main: 12 MP & Second: 12 MP",
                                OS = "iOS 13",
                                CPU = "Apple A13 Bionic 6 cores",
                                RAM = 4,
                                ROM = 64,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                SIM = "1 eSIM & 1 Nano SIM, Support 4G",
                Battery = 3110
            },
            new ProductSheet() {
                Name = "Samsung Galaxy Fold",
                Image = "Content/images/ProductSheet/4.jpg",
                Rating = 5,
                Price = 1898,
                Screen = "Dynamic AMOLED 7.3\" & Super AMOLED 4.6\",
                QuadHD(2K)",
                Camera = "Main: 12 MP & Others: 12 MP, 16 MP",
                OS = "Android 9.0 (Pie)",
                CPU = "Snapdragon 855 8 cores",
                RAM = 12,
                ROM = 512,
                SIM = "1 eSIM & 1 Nano SIM, Support 4G",
                Battery = 4380
            },
            new ProductSheet() {
                Name = "Realme 5 (3GB/64GB)",
                Image = "Content/images/ProductSheet/5.jpg",
                Rating = 3.5,
                Price = 166,
                Screen = "IPS LCD, 6.5\", HD+",
                Camera = "Main: 12 MP & Others: 8 MP, 2 MP, 2 MP",
                OS = "Android 9.0 (Pie)",
                CPU = "Snapdragon 665 8 cores",
                RAM = 3,
                ROM = 64,
                SIM = "2 Nano SIM, Support 4G",
                Battery = 5000
            },
            new ProductSheet() {
                Name = "BlackBerry KEY2",
                Image = "Content/images/ProductSheet/6.jpg",
                Rating = 4.5,
                Price = 711,
                Screen = "IPS LCD, 4.5\", Full HD",
                Camera = "Main: 12 MP & Second: 12 MP",
                OS = "Android 8.1 (Oreo)",
                CPU = "Snapdragon 660 8 cores",
                RAM = 6,
                ROM = 64,
                SIM = "2 Nano SIM, Support 4G",
                Battery = 3500
            },
        };
    }
}
```

Back to Top

Add a TransposedGrid control

Complete the following steps to initialize a TransposeGrid control.

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.

Copyright © GrapeCity, Inc. All rights reserved.

3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `TransposeGridController`).
    3. Click **Add**.
4. Include the following MVC references:
    - using System.Web.Mvc;
    - using <*ApplicationName*>.Models;
5. Replace the method Index() with the following method.

TransposeGridController.cs

C#

```
public class TransposedGridController : Controller
{
        // GET: TransposedGrid
        public ActionResult Index()
        {
                return View(ProductSheet.GetData());
        }
}
```

Add a View for the controller:

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the controller (for example: `TransposeGridController`) to open it.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add**. A view has been added for the controller.
6. In the Solution Explorer, double click `Index.cshtml` to open it.
7. Replace the default code of the **Views\Index.cshtml** file with the one given below to initialize a **TransposeGrid** control.

CSHTML

```
@using C1.Web.Mvc.Grid
@using <ApplicationName>.Models

@model IEnumerable<ProductSheet>
@{
    var rows = new Action<ListItemFactory<TransposedGridRow, TransposedGridRowBuilder>>(br
=>
    {
        br.Add(cb => cb.Binding("Image").Header("
").Align("center").Width("80").Height(170));
        br.Add(cb => cb.Binding("Name").Header("Model").Align("center").WordWrap(true));
        br.Add(cb =>
cb.Binding("Rating").Header("Rating").Align("center").Format("n1").Width("130").Height(40));
        br.Add(cb =>
cb.Binding("Price").Header("Price").Align("center").Format("c2").Width("120"));
        br.Add(cb => cb.Binding("Screen").Header("Screen").Align("center").WordWrap(true));
        br.Add(cb => cb.Binding("Camera").Header("Camera").Align("center").WordWrap(true));
        br.Add(cb => cb.Binding("OS").Header("OS").Align("center").WordWrap(true));
        br.Add(cb =>
cb.Binding("CPU").Header("CPU").Align("center").WordWrap(true).Width("150"));
        br.Add(cb =>
cb.Binding("RAM").Header("RAM").Align("center").Format("n0").Width("80"));
        br.Add(cb =>
cb.Binding("ROM").Header("ROM").Align("center").Format("n0").Width("80"));
        br.Add(cb => cb.Binding("SIM").Header("SIM").Align("center").WordWrap(true));
        br.Add(cb =>
cb.Binding("Battery").Header("Battery").Align("center").Format("n0").Width("80"));
    });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}

@(Html.C1().TransposedGrid<ProductSheet>()
        .AutoGenerateRows(false)
        .Id("theTransposedGrid")
        .CssClass("product-sheet-grid")
        .IsReadOnly(true)
        .Bind(Model)
        .HeadersVisibility(HeadersVisibility.Row)
        .Rows(rows)
        .ItemFormatter("formatItem")
        .OnClientLoadedRows("loadedRows")
        .Height(615)
)


<style type="text/css">
    .product-sheet-grid {
        max-height: 2000px;
    }

        .product-sheet-grid .wj-rowheaders .wj-cell {
            text-transform: uppercase;
            font-size: 80%;
        }

        .product-sheet-grid .wj-cell {
            padding: 4px;
            border-bottom: none;
        }

            .product-sheet-grid .wj-cell span.rating {
                font-size: x-large;
                color: #e7711b;
                margin-right: 0;
            }

            .product-sheet-grid .wj-cell span.price {
                font-weight: bold;
                color: #d0021b;
            }

        .product-sheet-grid img {
            max-width: 100%;
            max-height: 100%;
        }

    .star {
        font-size: x-large;
        width: auto;
        display: inline-block;
        color: gray;
    }

        .star.half:after {
            content: '\2605';
            color: #e7711b;
            margin-left: -20px;
            width: 10px;
            position: absolute;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            overflow: hidden;
        }
</style>

@section Scripts{
    <script>
        function loadedRows(grid) {
                let g = grid;
                g.columns.defaultSize = 210;
                setTimeout(function () {
                    g.autoSizeColumn(0, true, 30);
                    g.autoSizeRows();
                });
        }

        function formatItem(panel, r, c, cell) {
            if (panel.cellType == 1) {
                let binding = panel.rows[r].binding || panel.columns[c].binding;
                switch (binding) {
                    case 'Image':
                        cell.innerHTML = '<img src="' + '@Url.Content("~/")' +
cell.textContent + '" draggable="false"/>';
                        break;
                    case 'Name':
                        cell.innerHTML = '<b>' + cell.innerHTML + '</b>';
                        break;
                    case 'Price':
                        cell.innerHTML = '<span class="price">' + cell.innerHTML +
'</span>';
                        break;
                    // 星としての評価。
                    case 'Rating':
                        let rating = panel.getCellData(r, c, false) * 1,
                            stars = Math.floor(rating),
                            html = new Array(stars + 1).join('★');
                        html = '<span class="rating">' + html + '</span>';
                        if (rating > stars) {
                            html += '<span class="star half">☆</span>';
                        }
                        cell.innerHTML = html;
                        break;
                    case 'RAM':
                    case 'ROM':
                        cell.textContent += "GB";
                        break;
                    case 'Battery':
                        cell.textContent += "mAh";
                        break;
                }
            }
        }
    </script>
}
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.

Copyright © GrapeCity, Inc. All rights reserved.

2. Press **F5** to run the project.

Back to Top

Limitations

Some features available in a simple grid control or FlexGrid do not apply to the Transposed grid as it works differently with data. These features or limitations are listed below.

- Sorting
- Grouping
- Filtering
- Paging
- Adding new row
- Deleting row
- Selection
- DataMap/DataMapEditor
- Aggregate value in the column footer
- DefaultTypeWidth
- Automatic generation of columns

FlexMap

Create stunning GeoJSON maps with MVC FlexMap, a map control which provides rich and interactive visualization of geographic data. It enables you to create attractive visualizations with your geo data to bring your data to life and make it much easier to analyze. FlexMap supports point and choropleth maps and uses GeoJSON for binding geographical feature layers and point layers. It allows you to display data bound points or markers on the map and visualize geographies with color-coded regions. In addition, it provides fully interactive UI with zoom, pan and tooltip support.



主な特長

The FlexMap control allows you to create powerful map applications. Make the most of FlexMap by taking advantage of the following key features:

- **Layers Support**

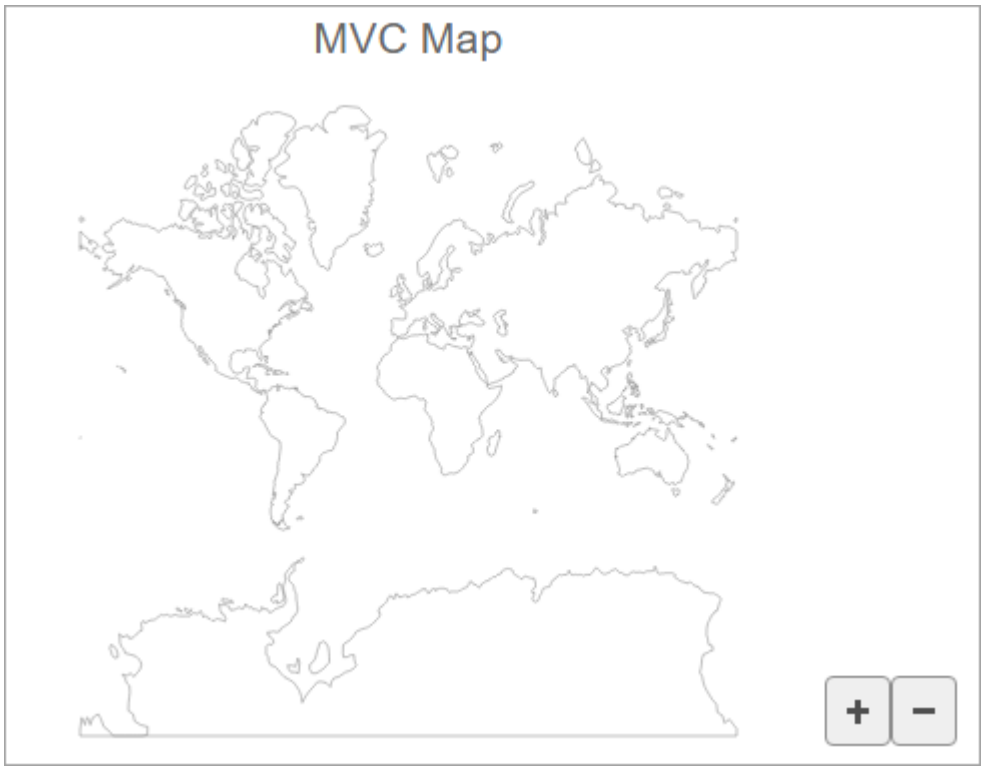
Copyright © GrapeCity, Inc. All rights reserved.

Use layers to draw data and add elements to the maps that are linked to geographical locations. For more information, see Layers topic.

- **Draw any Geometry**
  Draw geometries/shapes/paths with geo coordinates on top of the map. For example, draw Political borders, Geo details (showing automobile or airplane routes) and Choropleth maps (based on statistical data, such as population per country and other demographic data).
- **Display Rich Geographical Information**
  Use GeoJSON for binding geographical feature layers and point layers, and display rich geographical information from various sources.
- **Legend**
  FlexMap allows you to display legend at the desired position in the map.
- **Zoom and Pan**
  FlexMap provides support for zooming and panning maps.
- **Tooltips**
  FlexMap lets you display chart values using tooltips.

クイックスタート

The quick start guides you through the steps of adding the FlexMap control to your MVC web application, adding a layer to the map and setting the data source of map layer to display the world map. In the following example, we have used **Layers** property of the **FlexMap** class to set the map layer and added a GeoMap layer to it using the **AddGeoMapLayer** method. In addition, we specified the data source for the map layer using the **Url** property.



To accomplish this, follow these steps:

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Copyright © GrapeCity, Inc. All rights reserved.

Add FlexMap

To add the FlexMap control to your application, add **C1.Web.MVC** reference and follow these steps:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `FlexMapController`).
    3. Click **Add**.

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `FlexMapController.`
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller. Copy the following code and paste it inside **Index.cshtml**.

Index.cshtml copyCode

```
@using System.Drawing

@(Html.C1().FlexMap().Id("flexMap")
        .Header("MVC Map")
        .Height(400)
        .Layers(ls =>
        {
            ls.AddGeoMapLayer()
                .Url("/Content/data/land.json");

        })
)
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/**FlexMap/Index**) in the address bar of the browser to see the view.

Back to Top

FlexMapの使用

This section comprises all the features and auxiliary functionalities offered by the FlexMap control.

Color Scale
: Learn how to calculate the color used for specific geographical features.

Layers

Copyright © GrapeCity, Inc. All rights reserved.

Learn about different layers included in FlexMap.

Legend
: Learn how to display legend with FlexMap.

Styling
: Learn how to apply styles to the FlexMap control.

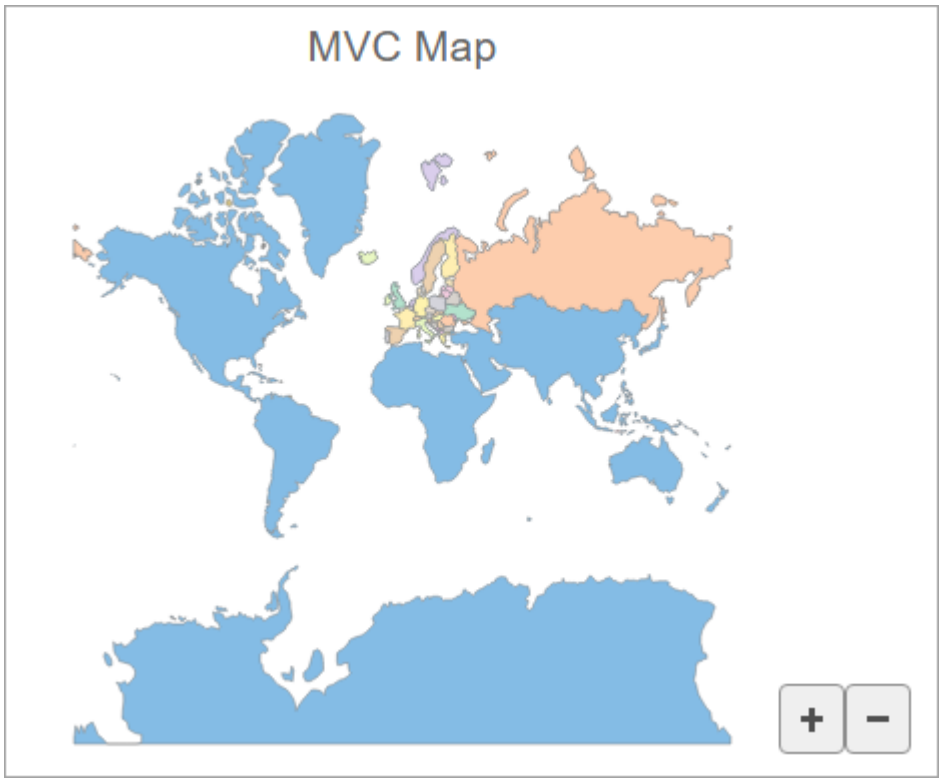
Supported Map Types
: Learn about the map types supported in FlexMap.

Tooltip
: Learn how to add a tooltip.

カラースケール

Color scale is one of the most useful ways to color a map. It can be used to color different areas in the map based on the data they represent. In FlexMap, color scale is supported in the GeoMapLayer and ScatterMapLayer through **ColorScale** property that, together with the **ColorScale** class, defines how to calculate the color used for specific geographical features. You can use the **ColorScale** property to set the color scale used for filling layers. In addition, you can use the **ColorScale.Colors** property to set the array of colors used for transformation and specify the **ColorScale.Binding** property as a function to provide flexible ways of specifying color-encoded values.

The following image displays the world map where countries are colored based on the data values from GeoJSON data.



The following code uses country color according to data values contained in the GeoJSON data:

Index.cshtml copyCode

```
@using System.Drawing
@{
    string[] palette = new string[]
    {
        "#fbb258", "#90cd97", "#f6aac9", "#bfa554", "#bc99c7",
        "#eddd46", "#f07e6e", "#88bde6", "#8c8c8c" };
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}

<script>
    function colorScale(v) {
        return 1 - v;
    }
    function colorScaleBindingLand(o) {
        return o.properties.color;
    }
    function colorScaleBindingEurope(o) {
        return o.properties.name.charCodeAt(0);
    }
</script>

@(Html.C1().FlexMap().Id("flexMap")
                .Header("MVC Map")
                .Height(400)
                .Layers(ls =>
                {
                    ls.AddGeoMapLayer()
                        .Url("/Content/data/land.json")
                        .ColorScale(cs => cs.Scale("colorScale")
                                            .Binding("colorScaleBindingLand")
                                            .Colors(palette));

                    ls.AddGeoMapLayer()
                        .Url("/Content/data/europe.json")
                        .ColorScale(cs => cs.Scale("colorScale")
                                            .Binding("colorScaleBindingEurope")

.Colors(C1.Web.Mvc.Chart.Palettes.Qualitative.Pastel2));
                })
)
```

レイヤー

Layers basically help draw data on a map. FlexMap includes three map layers to draw data on a map, which are described as follows:

- **GeoMapLayer**: Represents a map layer with geographical data.
- **ScatterMapLayer**: Represents a map layer with a collection of points in geographical coordinates.
- **GeoGridLayer**: Represents a map layer with grid.

FlexMap allows you to set the map layers collection using **Layers** property of the **FlexMap** class and then add layers to the collection.

The data drawn on the map is fetched from the data source of map layers which can be specified in two ways, either by setting the **ItemsSource** property to a data collection or by providing a path to data using the **Url** property. When the data source is specified as a **Url**, the map fetches the data and sets it to the **ItemsSource**.

Let us now discuss about the map layers in detail along with the programmatic approach of adding them to the FlexMap control.

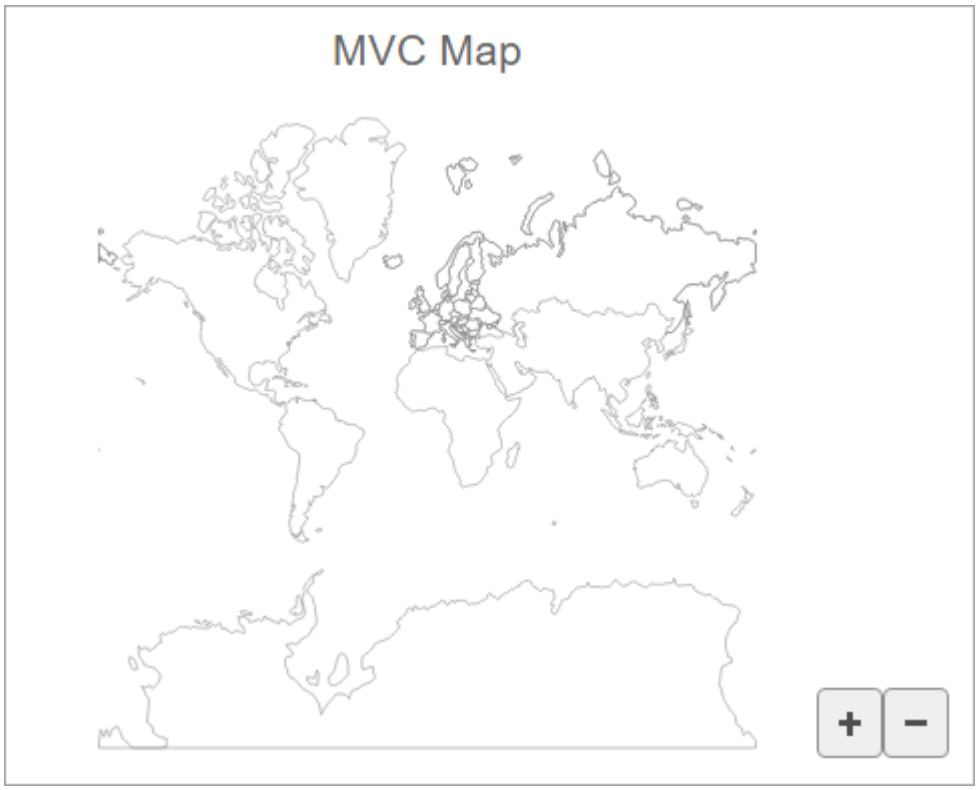
GeoMap Layer

Copyright © GrapeCity, Inc. All rights reserved.

The GeoMap layer displays data in GeoJSON format. GeoJSON is a JSON-based format that describes various geographical features (e.g. countries, provinces, cities, etc.) and can include other non-spatial data related to these features.

FlexMap allows you to add the GeoMapLayer to the map using the **AddGeoMapLayer** method. Then, you can set the data source for the layer using either the **Url** or **ItemsSource** property. Along with the data source, you can also set the color scale and style to the layer.

The following image shows FlexMap with two GeoMap layers.



The following code demonstrates how you can add multiple GeoMapLayers to the map using the **AddGeoMapLayer** method.

Index.cshtml copyCode

```
@using System.Drawing

@(Html.C1().FlexMap().Id("flexMap")
                .Header("MVC Map")
                .Height(500)
                .Layers(ls =>
                {
                ls.AddGeoMapLayer()
                    .Url("/Content/data/land.json");

                ls.AddGeoMapLayer()
                    .Url("/Content/data/europe.json");
    })
)
```

Back to Top

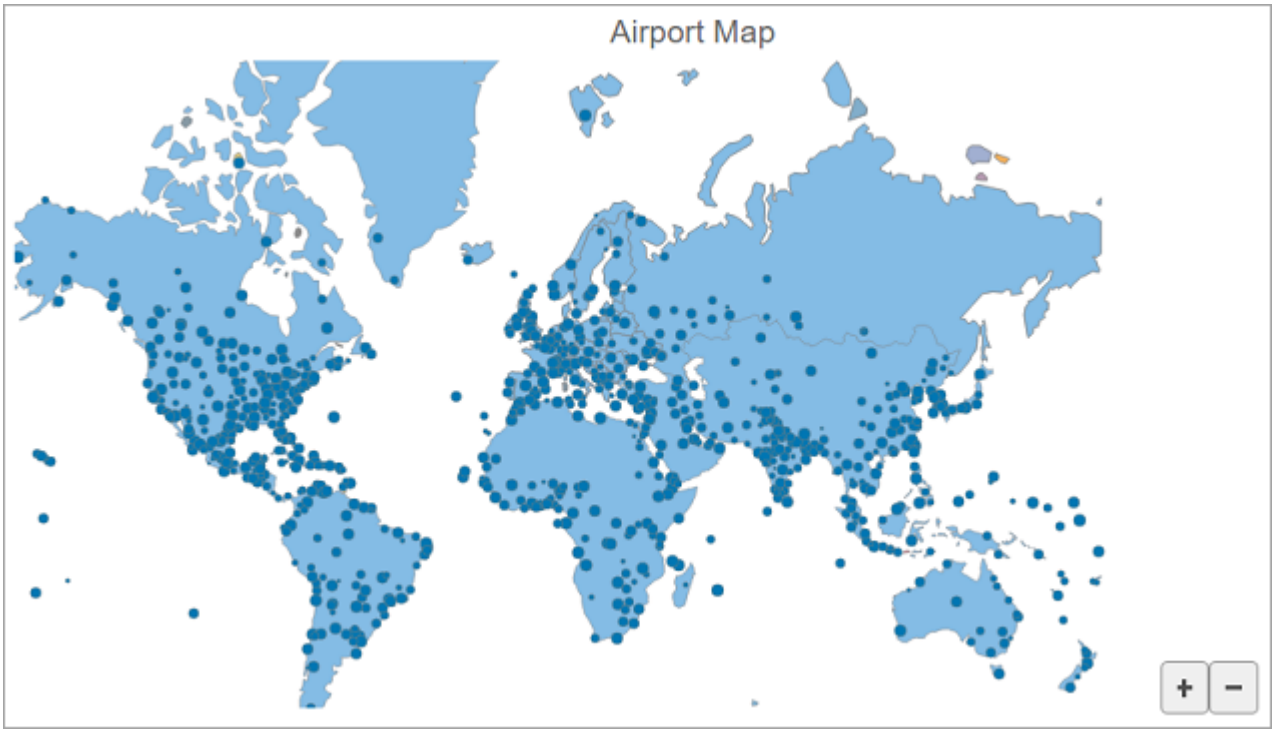
Copyright © GrapeCity, Inc. All rights reserved.

ScatterMap Layer

Scatter maps are used to show points of interest at specific coordinates on a map. The scatter map can be created in FlexMap using the **ScatterMap** layer. In FlexMap, the ScatterMap layer displays an array of points defined by geographical coordinates (latitude and longitude). You can add a ScatterMapLayer to the FlexMap using the **AddScatterMapLayer** method and set the data source for the layer using either the **Url** or **ItemsSource** property. In addition, you can **SymbolSize** property of the **ScatterMapLayer** class.

FlexMap can also be used to create bubble map, a variation of the scatter map. Bubble maps also show points of interest on maps, but they ascribe values to the point. The size of the bubbles depends on the value of the data (larger value = larger bubble). When you add the ScatterMapLayer to the map to create a bubble map, you can define the minimum and maximum size for the bubble plot using **SymbolMinSize** and **SymbolMaxSize** properties of the **ScatterMapLayer** class, respectively.

The following image shows airports on the world map using the combination of ScatterMapLayer with GeoMapLayer through bubble map.



The following code creates a combination of the GeoMapLayer that represents land on the map and the ScatterMapLayer displaying airports as dots.

Index.cshtml copyCode

```
@using System.Drawing
@{
    string[] palette = new string[]
    {
        "#fbb258", "#90cd97", "#f6aac9", "#bfa554", "#bc99c7",
        "#eddd46", "#f07e6e", "#88bde6", "#8c8c8c" };
}
<script>
    function scatterMapItemsSourceChanged(layer, a) {
        const bb = new wijmo.Rect(-200, -10, 500, 80);
        var map = wijmo.Control.getControl("#flexMap");
        map.zoomTo(bb);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        var features = layer.itemsSource;
        features.forEach(function (f) {
            var v = f.coordinates.split(",");
            f.x = v[0];
            f.y = v[1];
            f.flow = 100000 + Math.random() * 100000000;
        });
    }
    function colorScale(v) {
        return 1 - v;
    }
    function colorScaleBindingLand(o) {
        return o.properties.color;
    }
</script>

@(Html.C1().FlexMap().Id("flexMap")
        .Header("Airport Map")
        .Height(500)
        .Tooltip(tt => tt.Content("&#9992; <b>{iata_code}</b><br>{name}"))
        .Layers(ls =>
        {
            ls.AddGeoMapLayer()
                .Url("/Content/data/land.json")
                .ColorScale(cs => cs.Scale("colorScale")
                                    .Binding("colorScaleBindingLand")
                                    .Colors(palette));

            ls.AddGeoMapLayer()
                .Url("/Content/data/europe.json");

            ls.AddScatterMapLayer()
                .Url("/Content/data/airports.json")
                .Style(s => s.Fill("rgba(0,119,179,1)"))
                .Binding("x,y,flow")
                .SymbolMinSize(2)
                .SymbolMaxSize(8)
                .OnClientItemsSourceChanged("scatterMapItemsSourceChanged");
        })
)
```

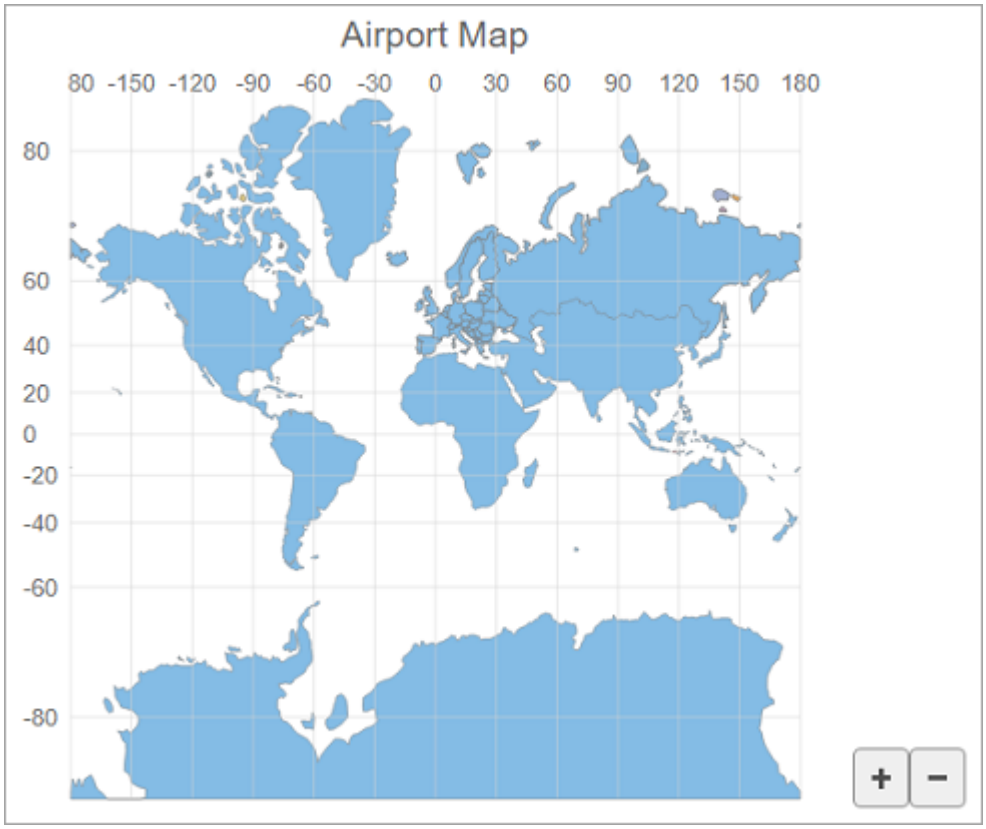
Back to Top

GeoGrid Layer

The GeoGrid layer displays grid for the geographical data. FlexMap adds the GeoGrid layer to the map using the **AddGeoGridLayer** method.

The following image shows FlexMap with grid created using the GeoGrid layer.

Copyright © GrapeCity, Inc. All rights reserved.



The following code demonstrates how you can add the GeoMapLayer to the map:

Index.cshtml copyCode

```
@using System.Drawing
@{
    string[] palette = new string[]
    {
        "#fbb258", "#90cd97", "#f6aac9", "#bfa554", "#bc99c7",
        "#eddd46", "#f07e6e", "#88bde6", "#8c8c8c" };
}
<script>
    function colorScale(v) {
        return 1 - v;
    }
    function colorScaleBindingLand(o) {
        return o.properties.color;
    }
</script>

@(Html.C1().FlexMap().Id("flexMap")
        .Header("Airport Map")
        .Height(500)
        .Layers(ls =>
        {
            ls.AddGeoMapLayer()
                .Url("/Content/data/land.json")
                .ColorScale(cs => cs.Scale("colorScale")
                                    .Binding("colorScaleBindingLand")
                                    .Colors(palette));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            ls.AddGeoMapLayer()
                .Url("/Content/data/europe.json");

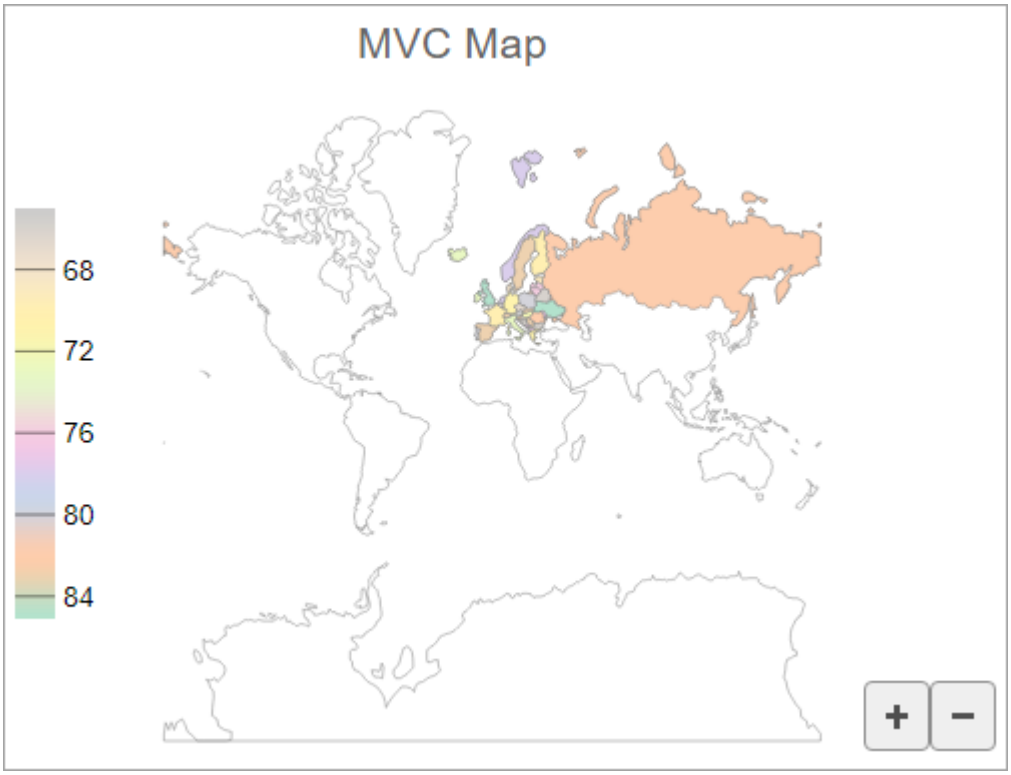
            ls.AddGeoGridLayer();
        })
)
```

Back to Top

凡例

Legend is an element which can be used to display a list of colors, symbols and text corresponding to each data or data series drawn on the map. You can add legend to the map using the **Legend** property and specify its position to display the legend on the map. The **Legend** property accepts the **Position** property as a parameter which sets the legend's position through **Position** enumeration.

The following image shows how the legend appears when you set the position of the legend to Left.



The following code example demonstrates how to set the position of a legend.

Index copyCode

```
@using System.Drawing

<script>
    function colorScale(v) {
        return 1 - v;
    }
    function colorScaleBindingEurope(o) {
        return o.properties.name.charCodeAt(0);
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
</script>

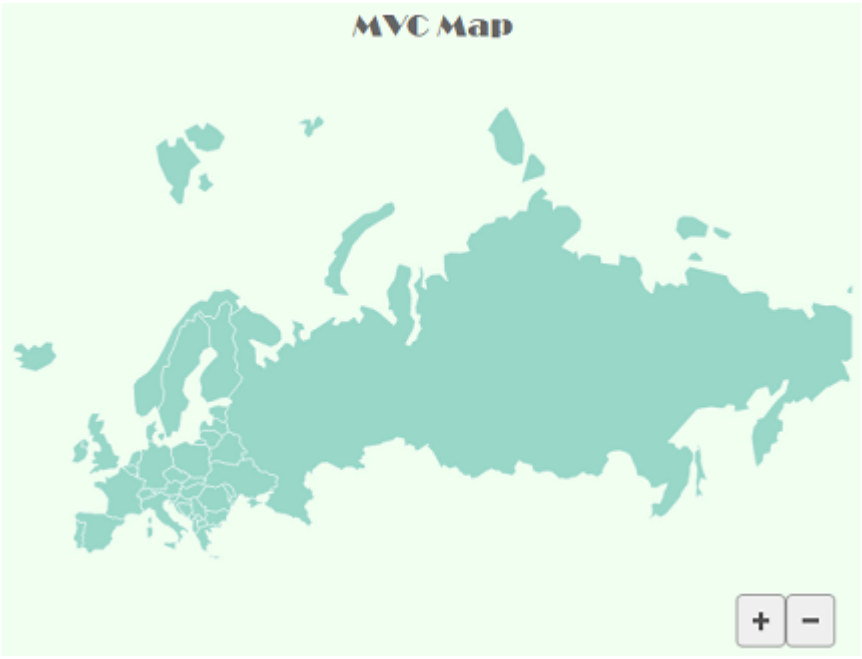
@(Html.C1().FlexMap().Id("flexMap")
                .Header("MVC Map")
                .Height(400)
                .Legend(C1.Web.Mvc.Chart.Position.Left)
                .Layers(ls =>
                {
                    ls.AddGeoMapLayer()
                        .Url("/Content/data/land.json");

                    ls.AddGeoMapLayer()
                        .Url("/Content/data/europe.json")
                        .ColorScale(cs => cs.Scale("colorScale")
                                            .Binding("colorScaleBindingEurope")

.Colors(C1.Web.Mvc.Chart.Palettes.Qualitative.Pastel2));
                })
)
```

スタイル設定

FlexMap allows you to change the appearance of the map by styling the map and its layers. It allows you to use CSS and create your own custom CSS to apply to the FlexMap control. It also provides you the **SVGStyle** class to define svg styles for the layers. For example, in the following image, the background color and font of FlexMap is added using CSS and the GeoMapLayer is filled with a color using **Fill** property of the **SVGStyle** class and the land borders are showcased with the white colored strokes using **Stroke** property of the **SVGStyle** class.



The following code illustrates how you can fill color in a map layer and display land borders using stroke color:

Index.cshtml copyCode

```
@using System.Drawing
<style>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .wj-flexmap {
        max-width: 600px;
        background-color: honeydew;
        font-family: Broadway;
    }
</style>
@(Html.C1().FlexMap().Id("flexMap")
                .Header("MVC Map")
                .Height(500)
                .Layers(ls =>
                {
                    ls.AddGeoMapLayer()
                        .Url("/Content/data/europe.json")
                        .Style(s => s.Fill("rgba(153,216,201,1)").Stroke("White"));
                })
)
```

サポートされているマップの種類

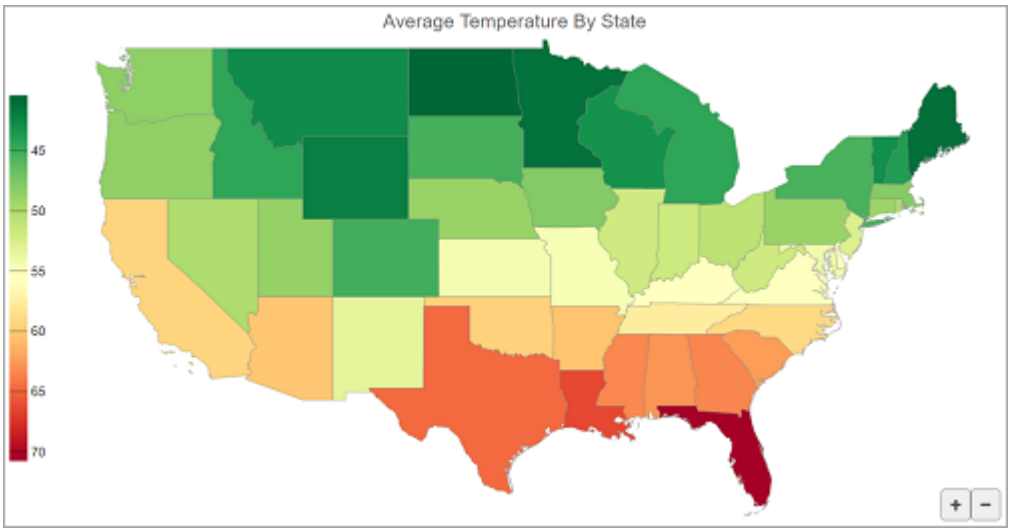
When it comes do presenting data using map, you need to make sure to choose the appropriate map type for your data. This determines how effectively the data is depicted when visualized using a map. FlexMap supports point and choropleth maps. Point maps display data points at bound coordinates on the map and can be used to create scatter maps and bubble maps. On the other hand, Choropleth maps are used to show statistical values for geographic areas. Let us discuss about these map types in detail in the following sections.

Point Map

FlexMap supports point maps, which creates scatter maps easily. The Scatter map shows points of interest at specific geographical coordinates on a map. It can be created in FlexMap using the **Scatter Map** layer. For detailed information on how FlexMap can be used to create scatter map, see ScatterMap Layer.

Choropleth Map

Choropleth map is one of the most common color-encoded map visualization which is used to show statistical values for geographic areas. With FlexMap, you can easily create a choropleth map as shown in the image below.



In the following example, we created a choropleth map and colored the map according to the data values contained in the GeoJSon data from the "temp.json" file (available in the FlexMap product sample at

Copyright © GrapeCity, Inc. All rights reserved.

Documents\ComponentOne Samples\ASP.NET MVC location on your system). In this choropleth map, each state is colored according to its average yearly temperature.

Index.html copyCode

```
    <script>
        c1.documentReady(function () {
            window.dataMap = new Map();
            wijmo.httpRequest('/Content/data/temp.json', {
                success: xhr => {
                    JSON.parse(xhr.responseText).forEach(el => dataMap.set(el.State,
parseFloat(el.AverageTemperature)));
                }
            });
        });

        function geoMapItemsSourceChanged(layer, a) {
            var map = wijmo.Control.getControl("#flexMap");
            map.zoomTo(layer.getGeoBBox());
        }

        function colorScale(v) {
            return 1 - v;
        }

        function colorScaleBinding(o) {
            return dataMap.get(o.properties.name)
        }

        function tooltipPopup(tt, e) {
            if (!wijmo.isFunction(tt.content)) {
                tt.content = function (f) {
                    return f.name + ' ' + dataMap.get(f.name) + '°F';
                }
                e.cancel = true;
            }
        }
    </script>


@(Html.C1().FlexMap().Id("flexMap")
        .Header("Average Temperature By State")
        .Height(600)
        .Tooltip(tt => tt.Content("<b>{name}</b>").OnClientPopup("tooltipPopup"))
        .Legend(lg => lg.Position(C1.Web.Mvc.Chart.Position.Left))
        .Layers(ls =>
        {
            ls.AddGeoMapLayer()
                .Url("/Content/data/US.json")
                .ColorScale(cs => cs.Scale("colorScale")
                                    .Binding("colorScaleBinding")

.Colors(C1.Web.Mvc.Chart.Palettes.Diverging.RdYlGn))
```


Copyright © GrapeCity, Inc. All rights reserved.

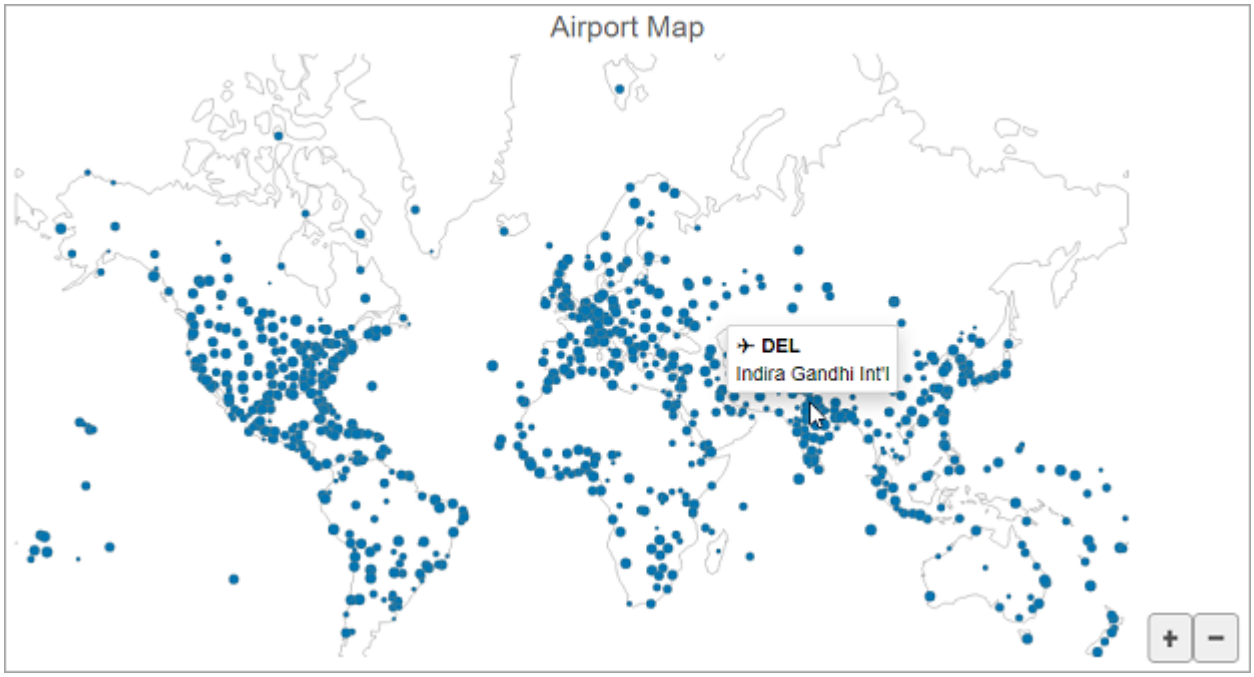
```
                .OnClientItemsSourceChanged("geoMapItemsSourceChanged");

        })
)
```

ツールチップ

Tooltip appears when you hover the mouse on a data point or an element. FlexMap allows you to display the tooltip using the **Tooltip** property and the information about the data points or elements in the tooltip using the **Content** property.

The following image displays an airport name in the tooltip appearing on mouse hover over a data point on map.



The following code demonstrates how to customize tooltip:

Index.cshtml copyCode

```
@using System.Drawing
@{
    string[] palette = new string[]
    {
        "#fbb258", "#90cd97", "#f6aac9", "#bfa554", "#bc99c7",
        "#eddd46", "#f07e6e", "#88bde6", "#8c8c8c" };
}
<script>
    function scatterMapItemsSourceChanged(layer, a) {
        const bb = new wijmo.Rect(-200, -10, 500, 80);
        var map = wijmo.Control.getControl("#flexMap");
        map.zoomTo(bb);

        var features = layer.itemsSource;
        features.forEach(function (f) {
            var v = f.coordinates.split(",");
            f.x = v[0];
            f.y = v[1];
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            f.flow = 100000 + Math.random() * 100000000;
        });
    }
</script>

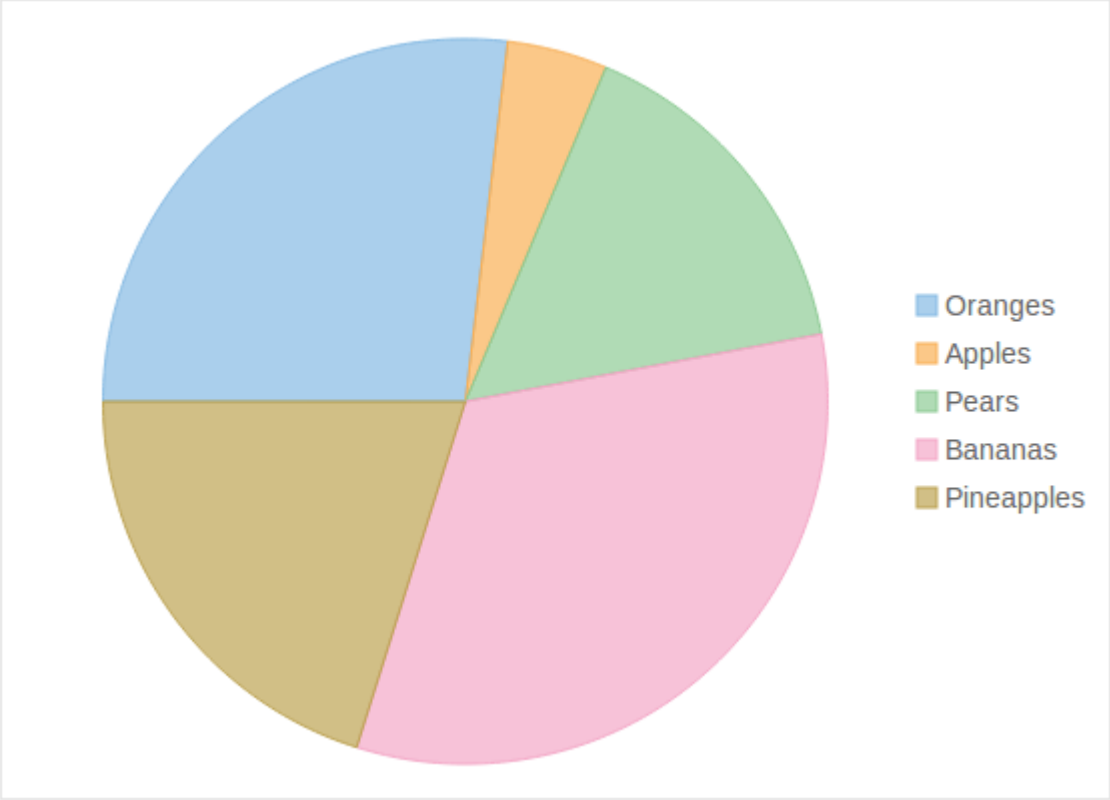
@(Html.C1().FlexMap().Id("flexMap")
        .Header("Airport Map")
        .Height(500)
        .Tooltip(tt => tt.Content("&#9992; <b>{iata_code}</b><br>{name}"))
        .Layers(ls =>
        {
            ls.AddGeoMapLayer()
                .Url("/Content/data/land.json");

            ls.AddScatterMapLayer()
                .Url("/Content/data/airports.json")
                .Style(s => s.Fill("rgba(0,119,179,1)"))
                .Binding("x,y,flow")
                .SymbolMinSize(2)
                .SymbolMaxSize(8)
                .OnClientItemsSourceChanged("scatterMapItemsSourceChanged");
        })
)
```

FlexPie

FlexPie コントロールを使用して、1つの系列を円グラフのいくつかのセグメントとして表すカスタマイズされた円グラフを作成できます。各セグメントの弧の長さが、そのセグメントによって表される値を示します。これらのグラフは、占有率などの割合データを表示する場合によく使用されます。セグメントに複数の色を使用すると、円グラフが見やすくなります。通常、各セグメントによって表される値は、ラベルで表示されます。

Copyright © GrapeCity, Inc. All rights reserved.



主な特長

FlexPieは、単純なデータ視覚化に加えて、多くの高度な機能を提供します。これらの機能は次のとおりです。

- **ヘッダーとフッター**
  単純なプロパティを使用して、タイトルやフッターテキストを設定できます。

- **凡例**
  必要に応じて凡例の位置を変更できます。

- **選択**
  選択モードを変更し、円グラフのセグメントの外観をカスタマイズできます。

- **割およびドーナツ円グラフ**
  単純なプロパティを使用して、円グラフを分割円グラフまたはドーナツ円グラフに変換できます。

クイックスタート:FlexPie へのデータの追加

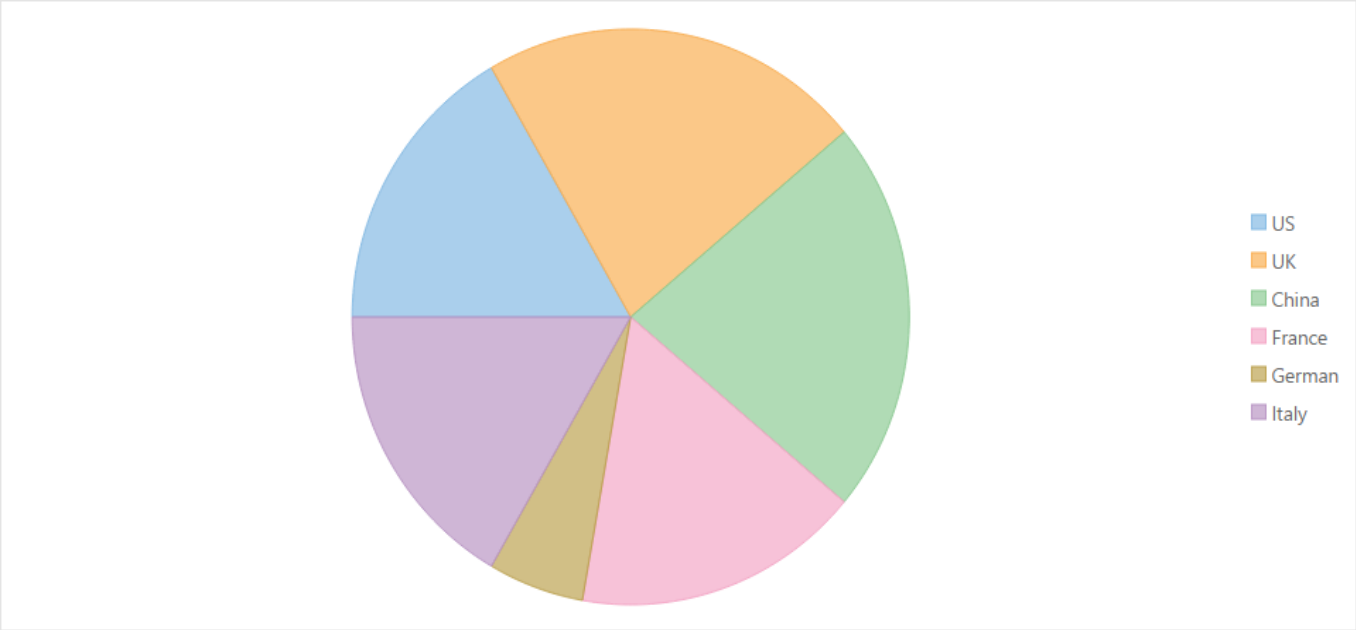
このセクションでは、MVC Web アプリケーションに FlexPie コントロールを追加し、そこにデータを追加する方法について説明します。

このトピックは3つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:FlexPie のデータソースの作成**
- **手順 3:FlexPie コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の FlexPie を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:FlexPie のデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`FlexPieDataSource.cs`)。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、FlexPie コントロールのデータソースになるクラスを定義します。

C#

C# copyCode

```
public class FlexPieDataSource
 {
   public string Country { get; set; }
   public int Sales { get; set; }

 public static IEnumerable<FlexPieDataSource> GetData()
  {
    var countries = new[] { "US", "UK",   "China", "France", "German", "Italy"
};
    var rand = new Random(0);
    List<FlexPieDataSource> list = new List<FlexPieDataSource>();
     for (int i = 0; i < 6; i++)
      {
        var sales = rand.Next(1, 5);
        list.Add(new FlexPieDataSource {  Sales = sales, Country = countries[i]
});
            }
            return list;
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    }
```

VB

VB

```
Public Class FlexPieDataSource
    Public Property Country() As String
        Get
            Return m_Country
        End Get
        Set
            m_Country = Value
        End Set
    End Property
    Private m_Country As String
    Public Property Sales() As Integer
        Get
            Return m_Sales
        End Get
        Set
            m_Sales = Value
        End Set
    End Property
    Private m_Sales As Integer

    Public Shared Function GetData() As IEnumerable(Of FlexPieDataSource)
        Dim countries = New String() {"US", "UK", "China", "France", "German",
"Italy"}
        Dim rand = New Random(0)
        Dim list As New List(Of FlexPieDataSource)()
        For i As Integer = 0 To 5
            Dim sales = rand.[Next](1, 5)
            list.Add(New FlexPieDataSource() With {
                Key.Sales = sales,
                Key.Country = countries(i)
            })
        Next
        Return list
    End Function
End Class
```

先頭に戻る

手順 3：FlexPie コントロールの追加

FlexPie コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。

Copyright © GrapeCity, Inc. All rights reserved.

1. コントローラーの名前を設定します(例:`QuickStartController`)。
2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
3. **[追加]**をクリックします。

4. 次に示すように MVC 参照を追加します。

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc.Chart;
```

5. メソッド `Index()` を次のメソッドに置き換えます。

C#

C# copyCode

```
public ActionResult QuickStart()
{
    return View(FlexPieDataSource.GetData());
}
```

VB

VB

```
Public Function QuickStart() As ActionResult
Return View(FlexPieDataSource.GetData())
End Function
```

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー `QuickStartController`をダブルクリックして開きます。
2. メソッド `QuickStart()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **QuickStart** で、ビューエンジンが **Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

6. FlexPie コントロールをビュー `QuickStart` で次のようにインスタンス化します。

Index.cshtml copyCode

```
@using MvcApplication1.Models
@model IEnumerable<FlexPieDataSource>
@(Html.C1().FlexPie<FlexPieDataSource>()
.Bind("Country", "Sales", Model)
)
```

Index.vbhtml

```
@ModelType IEnumerable(Of FlexPieDataSource)

@(Html.C1().FlexPie(Of FlexPieDataSource) _
```


Copyright © GrapeCity, Inc. All rights reserved.

```
.Bind("Country", "Sales", Model) _
.Height("300px"))
```

先頭に戻る

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例:http://localhost:1234/**QuickStart/QuickStart**)、ビューを表示します。

先頭に戻る

FlexPieの使用

This section comprises all the features and auxiliary functionalities offered by FlexPie.

FlexPie のカスタマイズ
: Learn how to add custom labels over the pie slices.

ドーナツ円グラフ
: Learn how to create a Donut pie chart.

分割円グラフ
: Learn how to create an exploded Donut pie chart.

ヘッダーとフッター
: Learn how to set different styles for header and footer in FlexPie.

凡例
: Learn how to display legend in FlexPie chart.

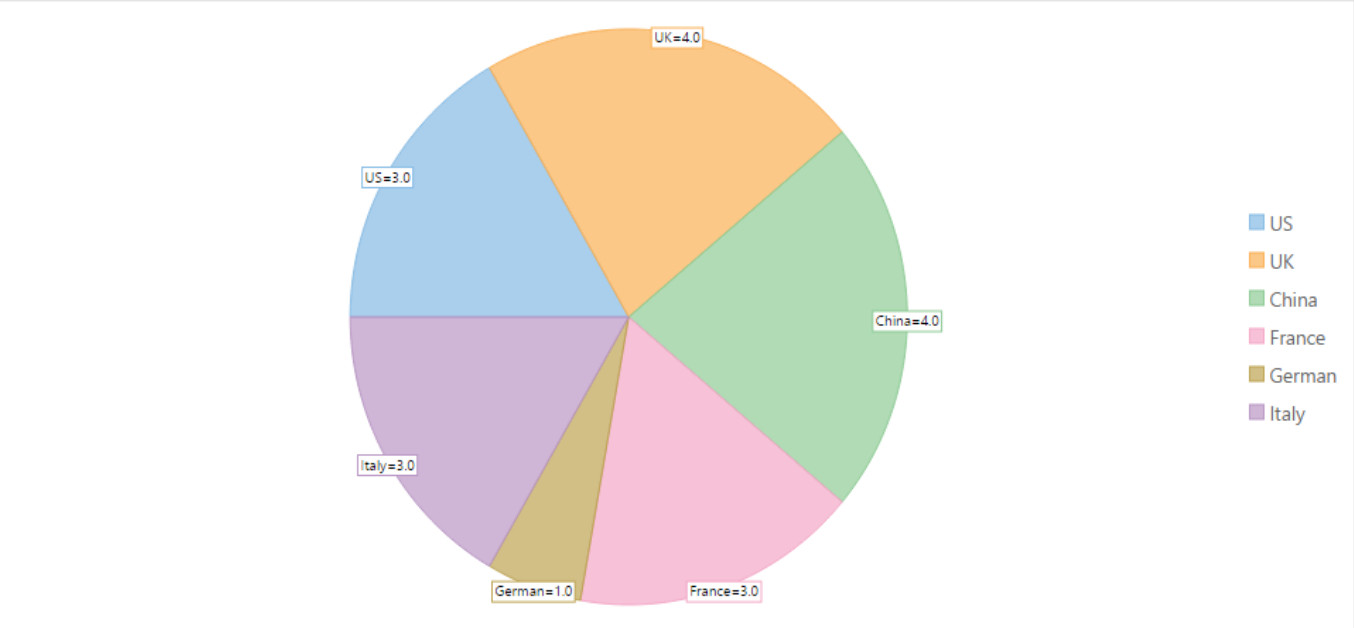
選択
: Learn how to select different elements and change the properties of those selected elements in FlexPie.

FlexPie のカスタマイズ

`ItemFormatter` プロパティを使用すると、JavaScript 関数を使用して FlexPie をカスタマイズできます。このトピックでは、このプロパティを使用して、円グラフのセグメントにラベルを追加する方法を示します。

次の図は、円グラフのセグメントにラベルを表示したところです。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、円グラフのセグメントにカスタムラベルを追加する方法を示します。

コードの場合

それには、最初に、FlexPie に対してラベルなどのカスタムコンテンツを追加するためのスクリプトを記述する必要があります。

JavaScript copyCode

```
<script>
    function formatItem(label, hitTestInfo, defaultFormatter)
    {
        var fsz = label.fontSize;
        label.fontSize = '10';
        defaultFormatter();
        var point = hitTestInfo.point.clone();
        var text = hitTestInfo.name + '=' + hitTestInfo.value.toFixed(1);
        var sz = label.measureString(text);
        var fill = label.fill;
        label.fill = 'white';
        label.drawRect(point.x - 2 - sz.width / 2, point.y - sz.height + 10, sz.width
+ 4, sz.height);
        label.fill = fill;
        point.x -= sz.width / 2;
        point.y += 9;
        label.drawString(text, point);
        label.fontSize = fsz;
    }
</script>
```

ItemFormatter プロパティを設定して、FlexPie にラベルを追加します。次の例では、「クイックスタート」セクションで作成したサンプルを使用しています。

Razor copyCode

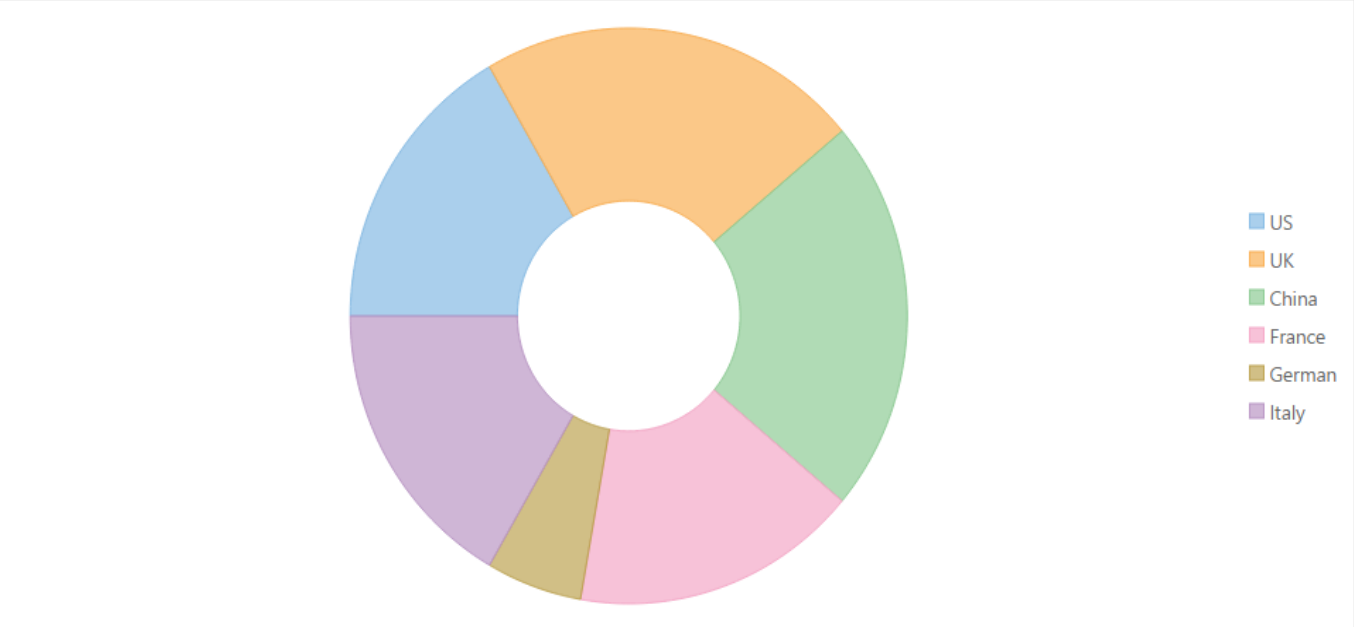
```
.ItemFormatter("formatItem")
```


Copyright © GrapeCity, Inc. All rights reserved.

ドーナツ円グラフ

InnerRadius プロパティを使用して、FlexPie の内側に空白部分を残し、ドーナツ円グラフを作成できます。この空白部分に追加データを表示することもできます。

次の図は、ドーナツ FlexPie を示しています。



次のコード例は、このプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor	copyCode
`.InnerRadius(0.4f)`	

分割円グラフ

Offset プロパティを使用して、FlexPie の中心から円グラフのセグメントを離し、分割円グラフを生成できます。このプロパティは、円グラフのセグメントを中心から離す距離を決定する 10 進値を受け入れます。

次の図は、分割された FlexPie を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、このプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

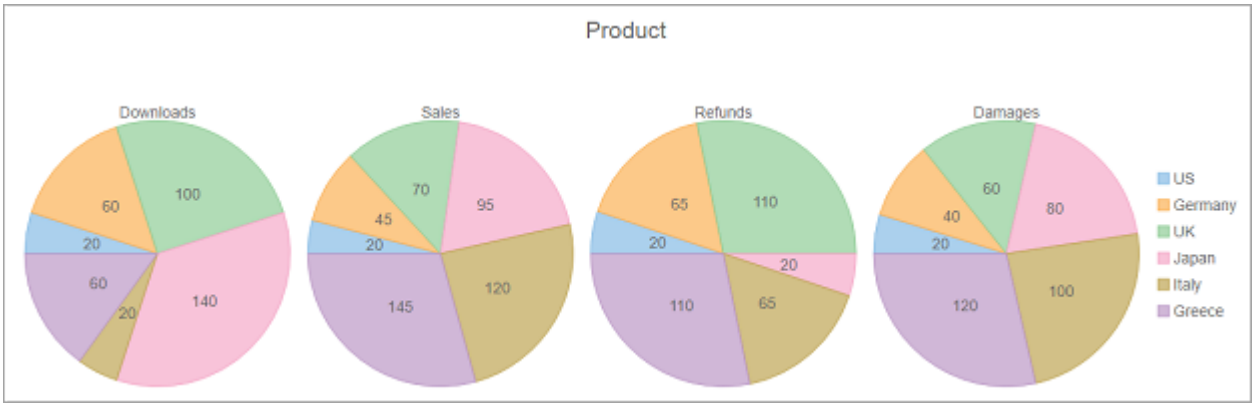
Razor	copyCode

```
.Offset(0.5f)
```

柔軟なレイアウト

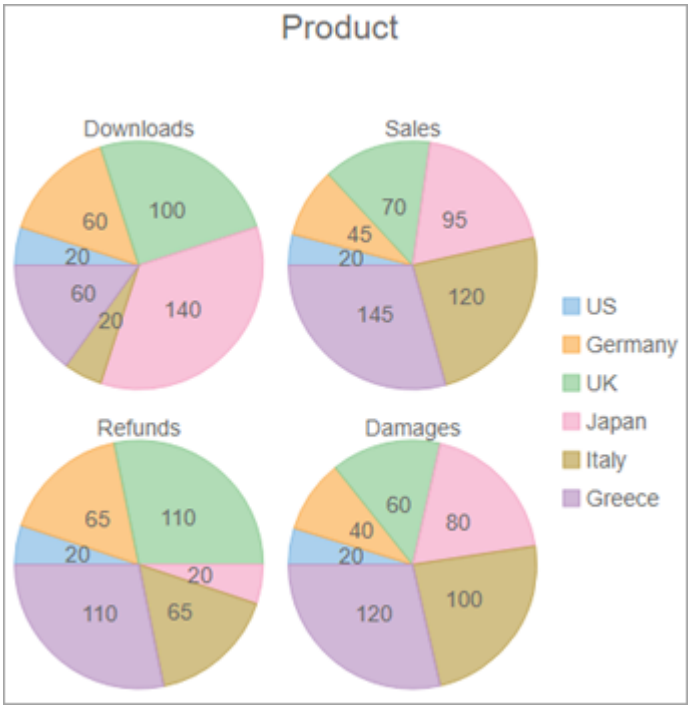
FlexPie supports multiple pie charts in one chart surface with the ability to control the layout of the charts. This option gives you much more control over how FlexPie fits into your application. FlexPie allows you to define how many charts you want to display in a line before wrapping to the next line using the **ChartsPerLine** property.

Here is how the default rendering of FlexPie with four charts looks like:



And if the **ChartsPerLine** is set to two, then we get the following result:

Copyright © GrapeCity, Inc. All rights reserved.



The following code demonstrates how to set the chart layout using the **ChartsPerLine** property. This code uses data from the ProductSales class.

ProductSales Data

ProductSales.cs copyCode

```
public class ProductSales
{
    public string Country { get; set; }
    public int Downloads { get; set; }
    public int Sales { get; set; }
    public int Refunds { get; set; }
    public int Damages { get; set; }
    public static List<ProductSales> GetData()
    {
        var countries = "US,Germany,UK,Japan,Italy,Greece".Split(new char[] { ',' });
        var data = new List<ProductSales>();
        for (var i = 0; i < countries.Length; i++)
        {
            data.Add(new ProductSales()
            {
                Country = countries[i],
                Downloads = ((i % 4) * 40) + 20,
                Sales = ((i % 7) * 25) + 20,
                Refunds = ((i % 3) * 45) + 20,
                Damages = ((i % 9) * 20) + 20
            });
        }
        return data;
    }
}
```

Controller Code

Copyright © GrapeCity, Inc. All rights reserved.

PieFlexibleLayout.cs copyCode

```
public ActionResult Index()
{
    return View(ProductSales.GetData());
}
```

View Code

Index.cshtml copyCode

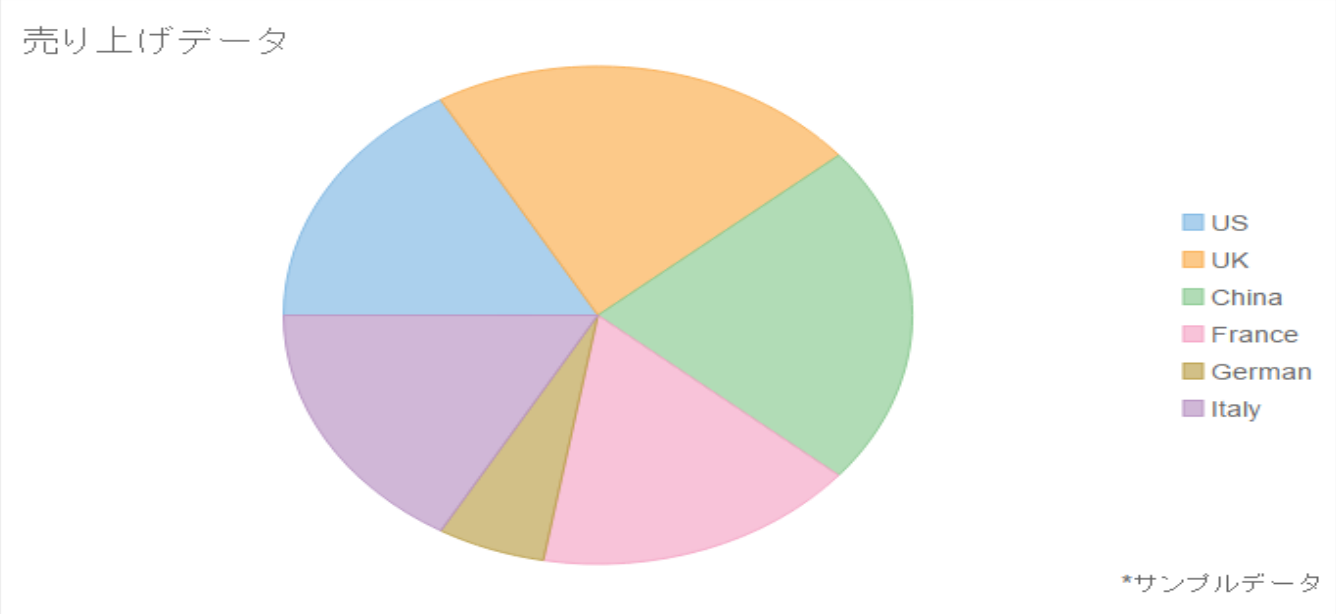
```
@model IEnumerable<ProductSales>

@(Html.C1().FlexPie<ProductSales>()
        .Id("multipleChart")
        .Header("Product")
        .Height(500)
        .Binding("Downloads,Sales,Refunds,Damages")
        .Bind(Model)
        .BindingName("Country")
        .DataLabel(dl => dl.Content("{value}"))
        .ChartsPerLine(2))
```

ヘッダーとフッター

FlexPie コントロールにタイトルを追加できます。それには、**Header** プロパティを設定します。タイトル以外に、チャートのフッターを設定することもできます。それには、**Footer** プロパティを設定します。ヘッダーおよびフッターテキストのスタイルを設定することもできます。それには、**HeaderStyle** および **FooterStyle** プロパティを使用します。

次の図は、これらのプロパティを設定した後の FlexPie を示しています。



次のコード例は、これらのプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

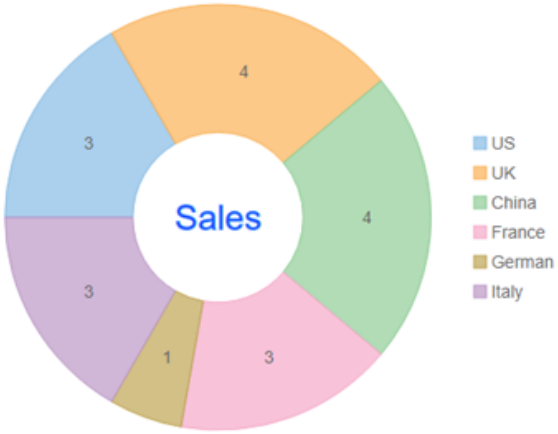
Copyright © GrapeCity, Inc. All rights reserved.

C# copyCode

```
.Header("Sales Data")
.HeaderStyle(style => style.FontSize("30").Halign("left"))
.Footer("*Random data by ComponentOne")
.FooterStyle(style => style.FontSize("15").Halign("right"))
```

内部テキスト

In addition to customizing appearance, FlexPie allows you to display text as the header or in the center of the chart. You can use **InnerText** property of the **FlexPie** class to display text inside the pie instead of showing it as a Header text at the top. Additionally, you can style or change the appearance of the text inside the pie using **InnerTextStyle** property of the FlexPie class.



The following examples shows how you can display text inside the pie using the **InnerText** and style it using the **InnerTextStyle** properties. This example uses the sample created in the Quick Start section.

View Code

Index.cshtml copyCode

```
@model IEnumerable<FlexPieDataSource>

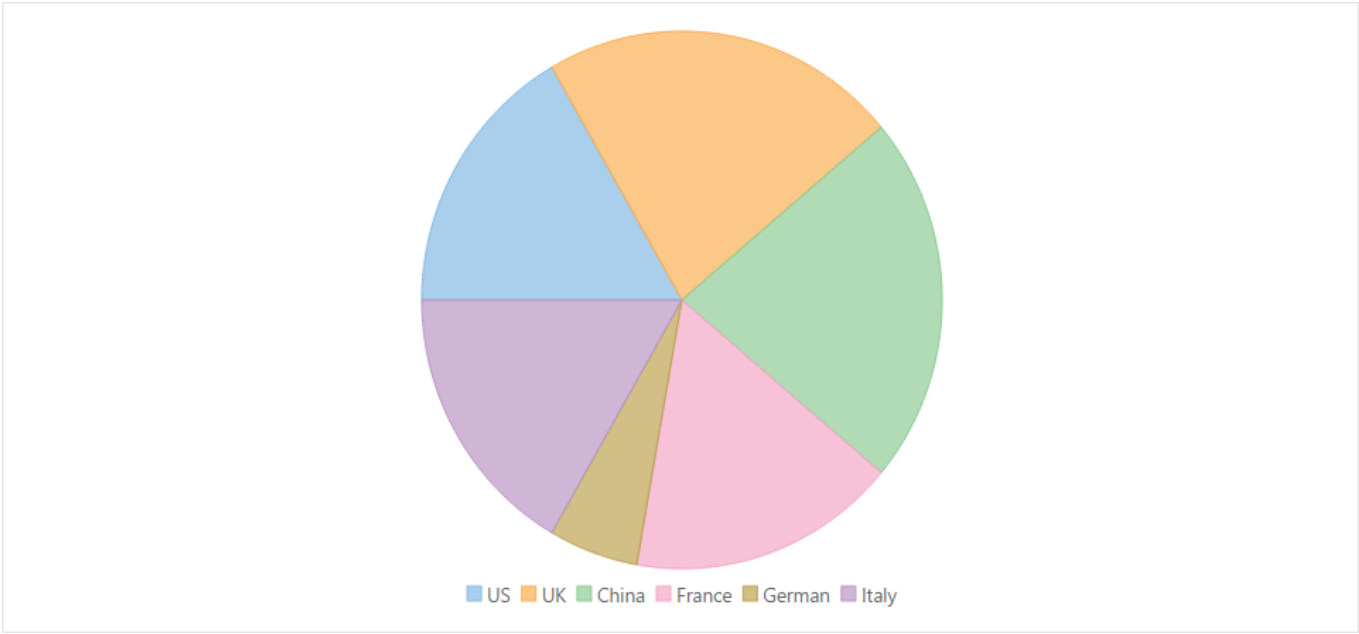
@(Html.C1().FlexPie<FlexPieDataSource>()
            .Bind("Country", "Sales", Model).DataLabel(dl=>dl.Content("{value}"))
            .InnerRadius(0.4f)
            .InnerText("Sales")
            .InnerTextStyle(svgb =>
svgb.FontSize(28).FontWeight("400").Fill("#2244ff").Stroke("#0099ff").StrokeWidth(1).StrokeOpacity(0.6))
)
```

凡例

凡例を表示する位置を指定できます。それには、FlexPie の **Legend** プロパティを使用します。

次の図は、これらのプロパティを設定した後の FlexPie を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、このプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor	copyCode
`.Legend(C1.Web.Mvc.Chart.Position.Bottom)`	

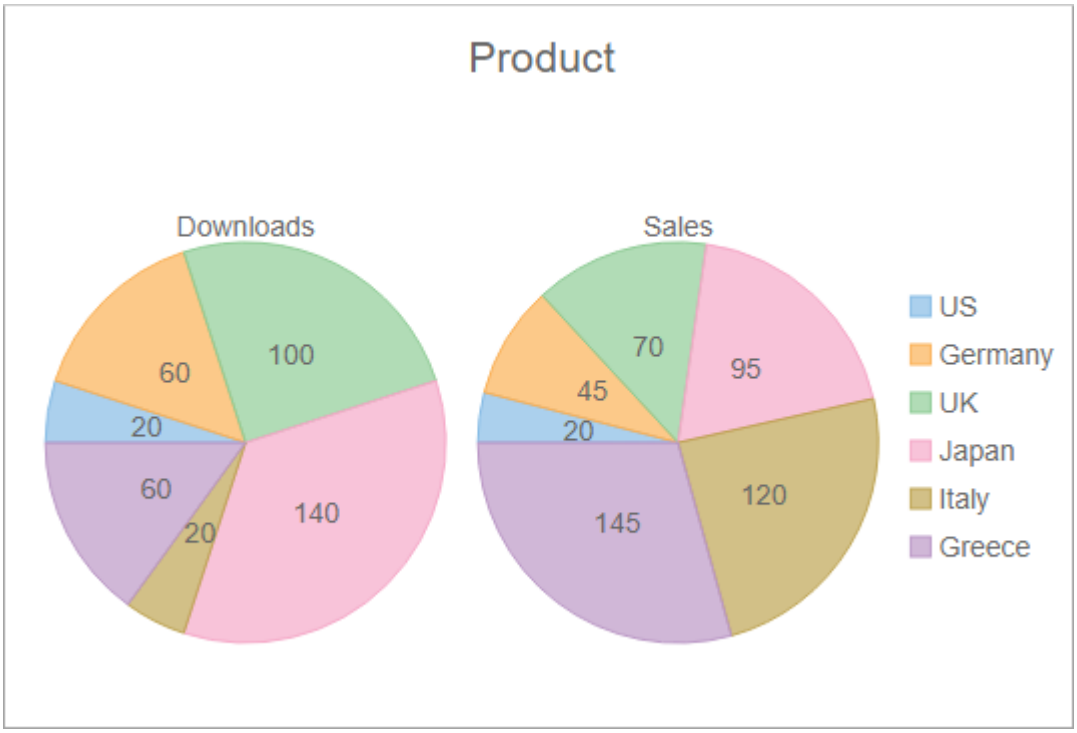
複数の円グラフ

There might be a scenario where you want to visualize and compare data of a category across different groups. This can be achieved plotting multiple pie charts together. In the following example, we are comparing the downloads and sales of a product in different countries using multiple pie charts.

FlexPie allows user to create multiple pie charts, based on the same data source, by simply specifying several comma-separated strings of field names in **Binding** property of the **FlexPie** class.

The following image shows two pie charts, namely Downloads and Sales, for six different countries.

Copyright © GrapeCity, Inc. All rights reserved.



The following code demonstrates how to add multiple pie charts using the same data source. This code uses data from the ProductSales class, available with the MvcExplorer sample.

Controller code

MultipleChartsController.cs copyCode

```
public ActionResult Index()
{
    return View(ProductSales.GetData());
}
```

View code

MultipleCharts.cshtml copyCode

```
@using  MVC_FlexPieMultiplePieChart.Models
@model IEnumerable<ProductSales>

@(Html.C1().FlexPie<ProductSales>()
        .Id("multipleChart")
        .Header("Product")
        .Binding("Downloads,Sales")
        .Bind(Model)
        .BindingName("Country")
        .DataLabel(dl => dl.Content("{value}")))
```

選択

ユーザーが FlexPie 内の任意の領域をクリックしたときに FlexPie のどの要素を選択するかを選択できます。それには、**SelectionMode** プロパティを設定します。このプロパティは3つのオプションを提供します。

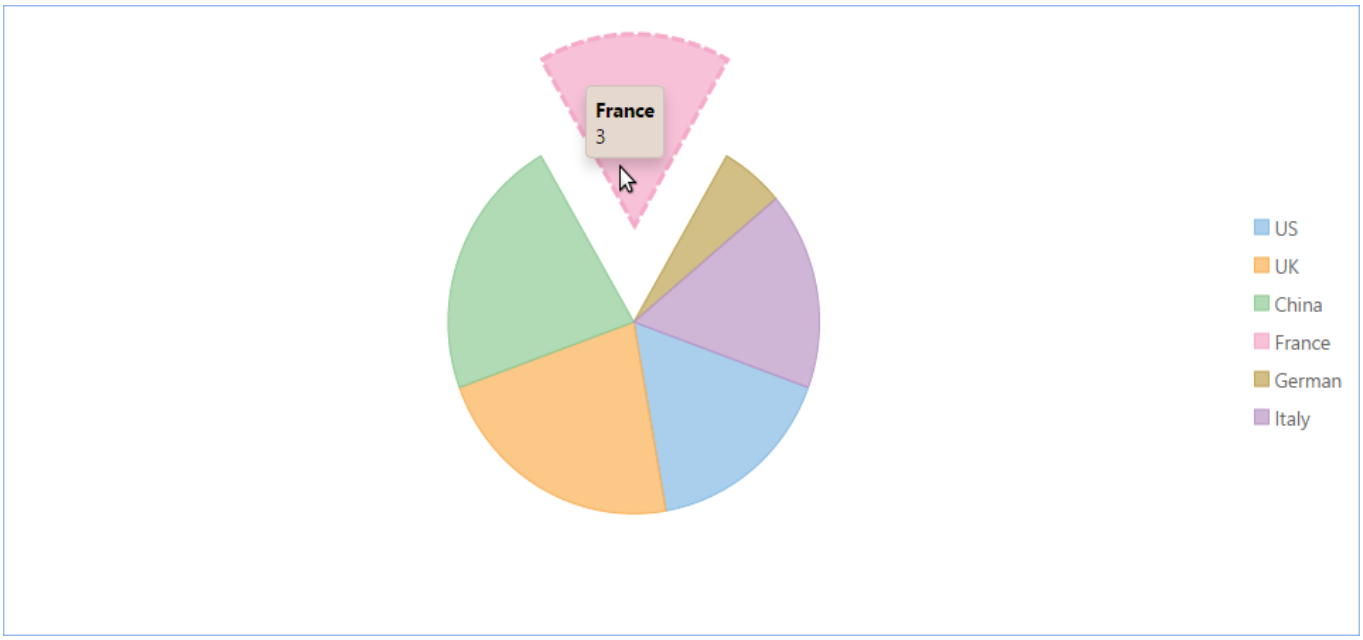
- **None**:どの要素も選択されません。

Copyright © GrapeCity, Inc. All rights reserved.

- **Point**:ユーザーがクリックした円グラフセグメントが強調表示されます。
- **Series**:円全体を強調表示します。

SelectionMode が **Point**に設定されている場合は、選択されている円グラフセグメントの位置を変更することもできます。それには、**SelectedItemPosition** プロパティを設定します。また、選択されている円グラフセグメントを FlexPie の中心から離すことができます。それには、**SelectedItemOffset** プロパティを設定します。

次の図は、これらのプロパティを設定した後の FlexPie を示しています。



次のコード例は、このプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

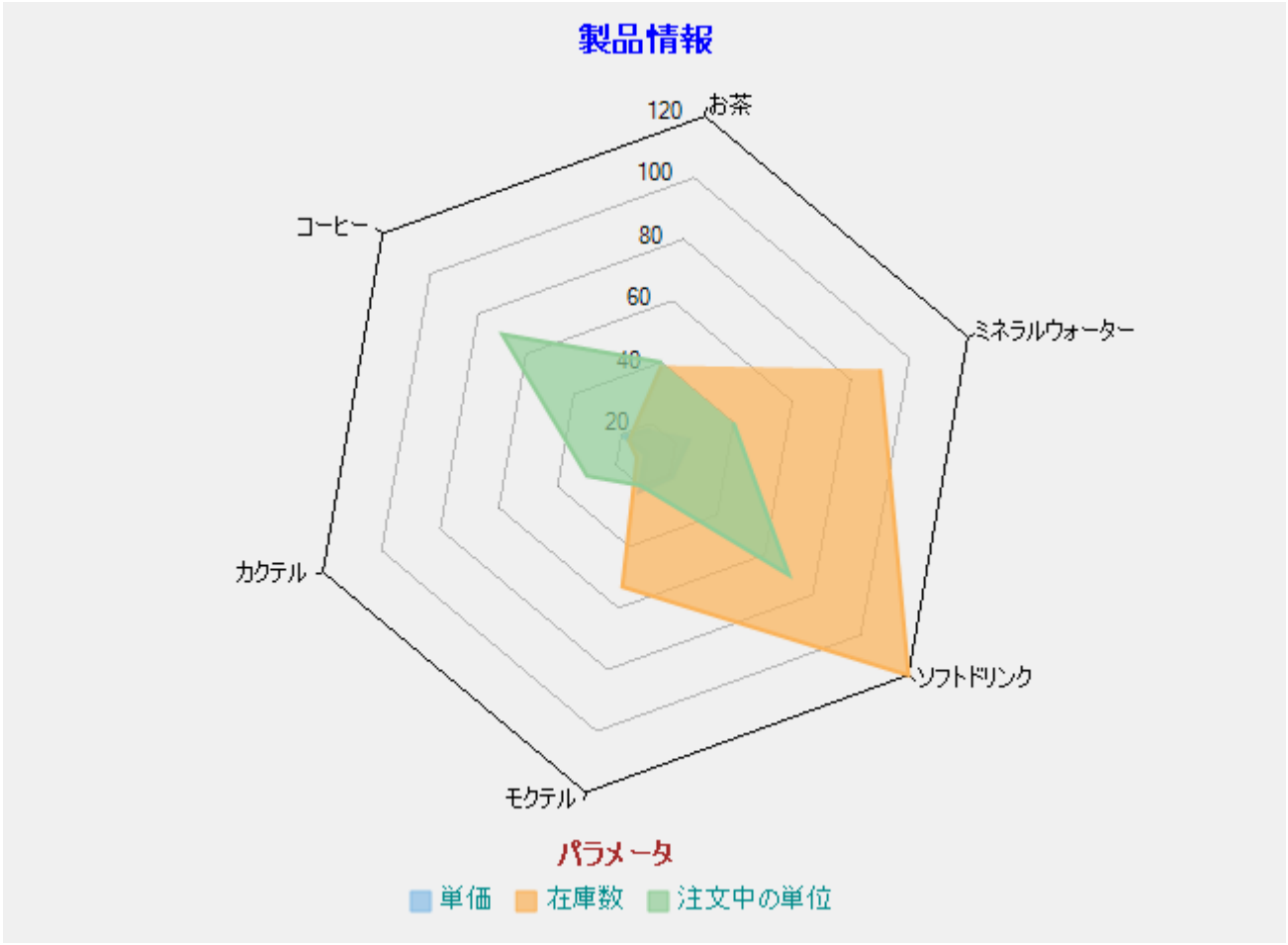
Razor copyCode

```
.SelectionMode(C1.Web.Mvc.Chart.SelectionMode.Point)
.SelectedItemPosition(C1.Web.Mvc.Chart.Position.Top)
.SelectedItemOffset(0.5f)
```

FlexRadar

FlexRadarチャートは、多変量データを3つ以上の量的変数から成る2次元チャートの形式で表示します。これらの変数は、同じ点から始まる複数の軸上に表されます。このチャートでは、中心から外側の円まで伸びる軸ごとに個別の変数の値がプロットされます。すべての軸は、等間隔の放射状に配置され、すべての軸で同じスケールが使用されます。各変数の値は個別の軸にプロットされ、すべての値は多角形の形に接続されます。チャートの中心は最小値を表し、チャートの端は最大値を表します。

Copyright © GrapeCity, Inc. All rights reserved.



FlexRadarのビジネスアプリケーションとしては、従業員のスキル分析、製品比較などによく使用されます。X値が角度値（度単位）を指定する数値なら、FlexRadarコントロールがポーラチャートになることは重要です。

主な特長

FlexRadarの主な特長は、次の通りです。

- **チャートタイプ**：FlexRadarでは、チャートに視覚化したいデータに応じて、さまざまな**チャートタイプ**を選択できます。サポートされているチャートタイプには、面グラフ、縦棒グラフ、折れ線グラフ、折れ線シンボルグラフ、および散布図グラフがあります。
- **開始角度**：FlexRadarでは、チャートの**開始角度**を設定できます（度単位）。デフォルトでは、放射軸は12時の位置から時計回り方向に描画されます。
- **ヘッダーとフッター**：FlexRadarでは、チャートのヘッダーとフッターを設定できます。これらを使用して、チャートタイトルを設定したり、チャートやデータに関する情報を追加してチャートを補足することができます。
- **凡例**：FlexRadarでは、チャートの各系列を表す凡例を表示することができます。凡例の方向、位置、スタイルなどをカスタマイズすることもできます。
- **反転したFlexRadar**：FlexRadarでは、プロットの方向を**反転**させて、データを反時計回り方向に表示することができます。

クイックスタート：FlexRadar へのデータの追加

Copyright © GrapeCity, Inc. All rights reserved.

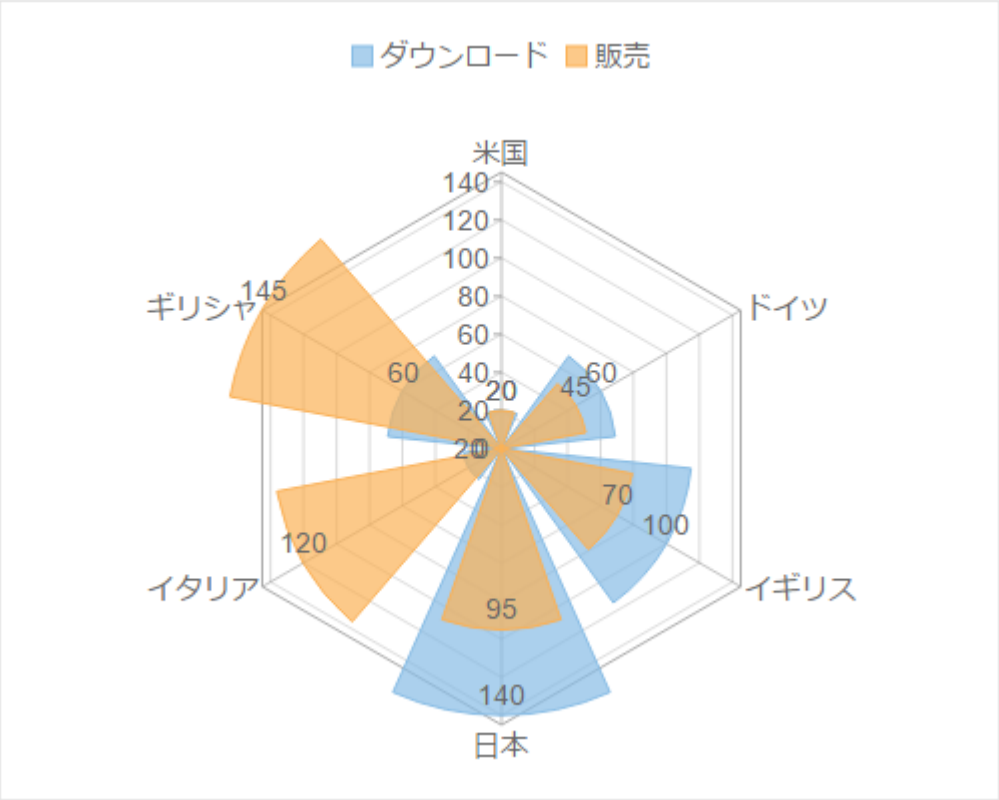
ASP.NET MVC コントロール

このトピックでは、MVC WebアプリケーションにFlexRadarチャートを追加し、そこにデータを追加する方法について説明します。

このトピックは3つの手順で構成されます。

- **手順1：MVC アプリケーションの作成**
- **手順2：FlexRadarのデータソースの作成**
- **手順3：FlexRadarチャートの追加**
- **手順4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のFlexRadarチャートを示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2：FlexRadarのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Sale.cs`）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`Sale.cs`モデルに追加します。ここでは、**Sale**クラスを使用して、データベース内の売上データを表しています。**Sale**オブジェクトの各インスタンスがFlexChartのデータに対応します。

Sale.cs copyCode

```
namespace FlexRadarChart.Models
{
    public class Sale
    {
        public int Id { get; set; }
        public string Country { get; set; }
        public int Downloads { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public int Sales { get; set; }
        public static List<Sale> GetData()
        {
            var countries = "米国,ドイツ,イギリス,日本,イタリア,ギリシャ".Split
            (new char[] { ',' });
            List<Sale> data = new List<Sale>();
            for (var i = 0; i < countries.Length; i++)
            {
                data.Add(new Sale()
                {
                    Country = countries[i],

                    Downloads = ((i % 4) * 40) + 20,

                    Sales = ((i % 7) * 25) + 20
                });
            }
            return data;
        }
    }
}
```

先頭に戻る

手順3：FlexRadarチャートの追加

アプリケーションにFlexRadarチャートを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`FlexRadarController`）。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

    C#

    ```
    using C1.Web.Mvc;
    using C1.Web.Mvc.Serializition;
    using C1.Web.Mvc.Chart;
    ```

5. メソッド**Index()**を次のメソッドに置き換えます。

    FlexRadarController.cs

    ```
    public ActionResult Index()
            {
                return View(Models.Sale.GetData());
            }
    ```

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`FlexRadarController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。

Copyright © GrapeCity, Inc. All rights reserved.

3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

Index.cshtml copyCode

```
@using FlexRadarChart.Models;
@model IEnumerable<Sale>
@using C1.Web.Mvc.Chart;

@(Html.C1().FlexRadar()
.Bind("Country", "Downloads", Model)
.ChartType(RadarChartType.Column)
.DataLabel(label =>
{
    label.Content("{y}");
})
.Series(ser =>
{
    ser.Add().Name("ダウンロード");

    ser.Add().Binding("Sales").Name("販売");
})
.Legend(Position.Top)
.Width("500px")
.Height("400px"))
```

手順4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http://localhost:1234/**FlexRadar/Index**)。

先頭に戻る

FlexRadarの使用

This section comprises all the features and auxiliary functionalities offered by FlexRadar.

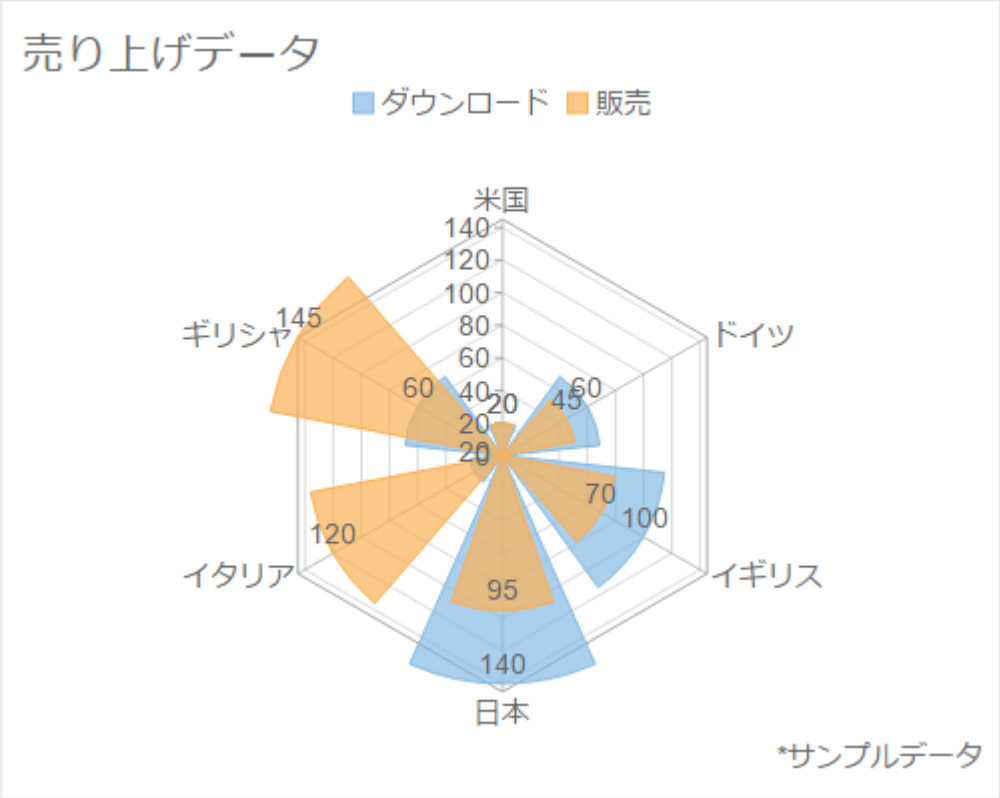
ヘッダーとフッター
Learn how to style the header and footer in FlexRadar.
凡例
Learn how and where to use legend in FlexRadar.

ヘッダーとフッター

FlexRadarコントロールにタイトルを追加できます。それには、**Header**プロパティを設定します。タイトル以外に、チャートのフッターを設定することもできます。それには、**Footer**プロパティを設定します。ヘッダーおよびフッターテキストのスタイルを設定することもできます。それには、**HeaderStyle**プロパティと**FooterStyle**プロパティを設定します。

次の図は、これらのプロパティを設定した後のFlexRadarを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



Razor

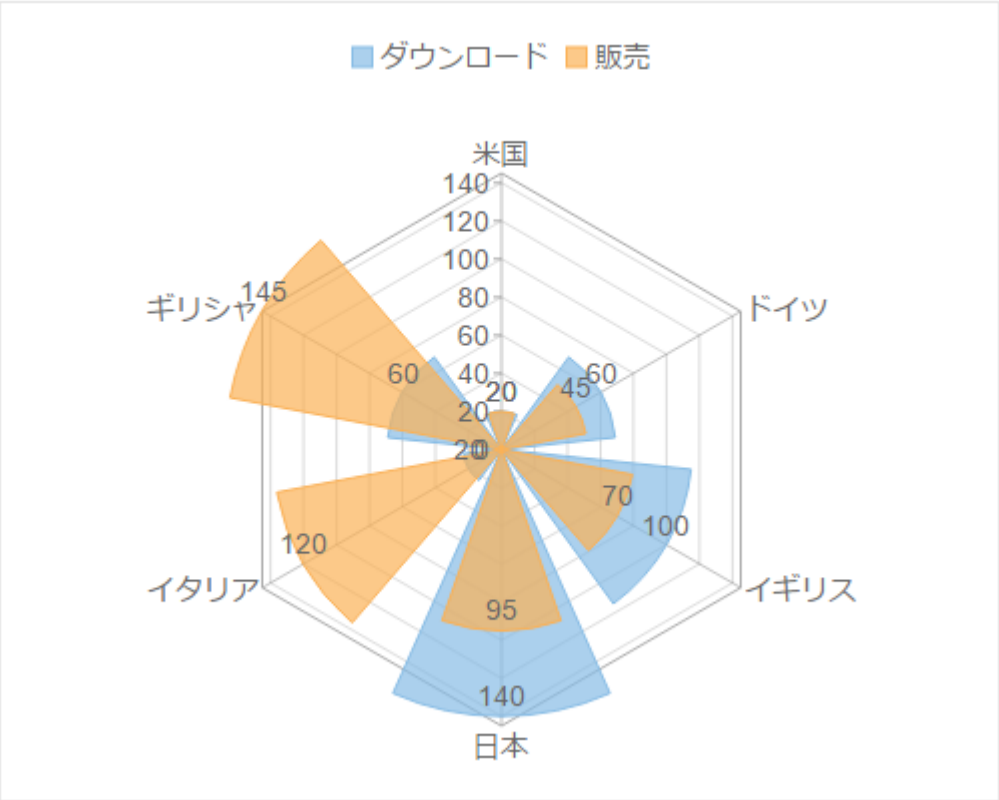
```
.Header("売り上げデータ")
.HeaderStyle(hs => hs.Halign("left"))
.FooterStyle(fs => fs.Halign("right"))
.Footer("*サンプルデータ")
```

凡例

FlexRadarチャートの**Legend**プロパティを使用して、凡例を表示する場所を指定できます。凡例を使用すると、チャートの系列を定義済みの記号および名前付きで表示できます。

以下の画像は、**Chart.Position**プロパティを**Top**に設定した後にFlexRadarチャートがどのように表示されるかを示します。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、Positionプロパティを設定する方法を示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

コードの場合

Razor

```
.Legend(C1.Web.Mvc.Chart.Position.Top)
```

FlexReport

レポートはビジネス意思決定とナレッジマネジメントに不可欠です。ASP.NET MVC では、Webでレポートを表示できるFlexReport for MVCが導入/リリースされています。

軽量で高速なコントロールが包括的なエンタープライズレポーティングツールとして登場しました。簡単にプレビュー、エクスポート、印刷を行える、複雑ながら高パフォーマンスなレポートを構築できます。FlexReportによって生成されたレポートは、汎用レポート表示ツール**FlexViewer**を使用して表示できます。

メモ:FlexReportコントロールを使用するための最低のサーバー構成要件はWindows Server 2008 R2です。

FlexReport for ASP.NET MVCは、次のコンポーネントで構成されます。

1. **FlexReport Designer**:レポートを作成するスタンドアロンデザイナアプリケーション。このデザイナアプリケーションは、デフォルトでは**C:\Program Files (x86)\ComponentOne\Apps\v4.0**にインストールされます。レポート定義を作成する方法とFlexReportデザイナの操作の詳細については、FlexReportDesignerドキュメントを参照してください。
2. **FlexViewerコントロール**:HTML5に基づくクライアントビューアコントロール。これは、レポートのプレビューに便利なJavaScriptコントロールです。
3. **FlexReport Web API**:ビューアがクライアントにレポートをロードして表示するための通信に使用するRESTサービス。

主な特長

Copyright © GrapeCity, Inc. All rights reserved.

FlexReportには主に次の便利な機能があります。

- **Light-weight and Fast**
  FlexReport is light-weight and fast in particular for smaller reports.

- **High Quality Rendering**
  FlexReport draws and generates high performance and quality report content that does not depend on measurement context like printer, screen, etc.

- **Comprehensive Tool**
  FlexReport is a comprehensive reporting tool which provides complete reporting solution- from building complex reports to previewing , printing and exporting.

FlexReport Web API の設定

FlexReport Designerを通して作成したレポートは、FlexReport Web APIとクライアントHTML 5レポートビューア（FlexViewer）コントロールを使用してWebで簡単に表示できます。

FlexReport Web APIは、FlexReportからエクスポートされたHTMLコンテンツをロードして表示するレポートビューアコントロール（クライアント側）との通信で重要な役割を果たします。

FlexReport WebAPIサービスは、次の2つの方法で作成できます。

1. ComponentOne Web API Editionを使用する
2. 標準のVisual Studio Web APIテンプレートを使用する

以下のセクションでは、ComponentOneテンプレートと標準のVisual Studioテンプレートを使用してFlexReport Web APIを作成および構成する方法を示します。

- ComponentOne WebAPI テンプレートの使用
- 標準のVisual for Web APIテンプレートの使用
- C1ReportViewer テンプレートの使用

ノート:

- レポートサービスの**Web API URL**を正しく作成できたら、FlexViewer for MVCやWijmo Viewerを使用して、サービスに保存されているレポートにアクセスして表示することができます。レポートの表示方法については、「FlexViewerでのレポートの表示」を参照してください。
- FlexViewerの詳細なデモサンプルについては「FlexViewerの使用方法」を、FlexReport Web APIの詳細なデモサンプルについては「Web API Editionのレポートサービス」を参照してください。

ComponentOne WebAPI テンプレートの使用

ComponentOne インストーラからWeb API Editionをインストールした場合は、FlexReport Web APIを使用して事前構成されたプロジェクトを簡単に作成できます。ComponentOne Web APIテンプレートを使用して、「FlexreportWebAPI」という名前の新しいプロジェクトを作成し、次の手順を実行します。

- **手順1：Web APIアプリケーションへのレポートファイルの追加**
- **手順2：レポートのルート位置の設定**
- **手順3：FlexReport Web APIサービスの展開**

手順1：Web APIアプリケーションへのレポートファイルの追加

アプリケーションにレポートファイルを追加するには、次の手順を実行します。

1. アプリケーションに**Files**という名前のフォルダを追加します。

Copyright © GrapeCity, Inc. All rights reserved.

2. それにFlexReportのレポート定義ファイルを追加します。
この例では、FlexCommonTasks.flxrレポート定義を使用しています。これは、**FlexViewerハウツー**サンプル（デフォルトでは**~\Documents\ComponentOne Samples\ASP.NET MVC\MVC\HowTo\FlexViewer**にインストールされる）内にあります。

レポートでローカルデータベース（MDBファイル、MDFファイルなど）を使用している場合は、アプリケーションの**[App_Data]**フォルダにそのデータベースを追加します。ただし、レポートの接続文字列が**[App_Data]**フォルダを指していること確認してください。

```
<DataSources>
  <DataSource>
    <Name>Main</Name>
    <DataProvider>OLEDB</DataProvider>
    <ConnectionString>Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|C1Demo.mdb;Persist Security Info=False</ConnectionString>
    <RecordSource>Employees</RecordSource>
    <RecordSourceType>TableDirect</RecordSourceType>
  </DataSource>
</DataSources>
```

手順1に戻る

手順2：レポートのルート位置の設定

1. **Startup.cs**ファイルで、**Startup**クラスの**Configuration**メソッド内に次のコードを追加します。

Startup.cs

```
app.UseCors(CorsOptions.AllowAll);
var folder = GetFullRoot("Files");
app.AddDiskStorage("root", folder);
```

このコードは、レポートファイルが格納されるフォルダ/場所を登録します。この場合は、"Files"フォルダです。

2. **Startup**クラスに**GetFullRoot**関数を追加します。

Startup.cs

```
private static string GetFullRoot(string root)
{
        var applicationBase =
AppDomain.CurrentDomain.SetupInformation.ApplicationBase;
        var fullRoot = Path.GetFullPath(Path.Combine(applicationBase, root));
        if (!fullRoot.EndsWith(Path.DirectorySeparatorChar.ToString(),
StringComparison.Ordinal))
        {
                fullRoot += Path.DirectorySeparatorChar;
        }
        return fullRoot;
}
```

手順1に戻る

手順3：FlexReport Web APIサービスの展開

1. プロジェクトをコンパイルします。
2. プロジェクトをIISに展開します。

ローカルIISでホストされるサービスの場合、Web API URLは**http://localhost/FlexReportwebAPI/api/report**です。

> **メモ**: レポートサービスの**Web API URL**を正しく作成できたら、FlexViewer for MVCやWijmo Viewerを使用して、サービスに保存されているレポートにアクセスして表示することができます。レポートの表示方法については、「FlexViewerでのレポートの表示」を参照してください。

手順1に戻る

Copyright © GrapeCity, Inc. All rights reserved.

標準のVisual for Web APIテンプレートの使用

標準のVisual Studio Template for Web APIを使用してFlexReport Web APIを構成するには、次の手順を実行します。

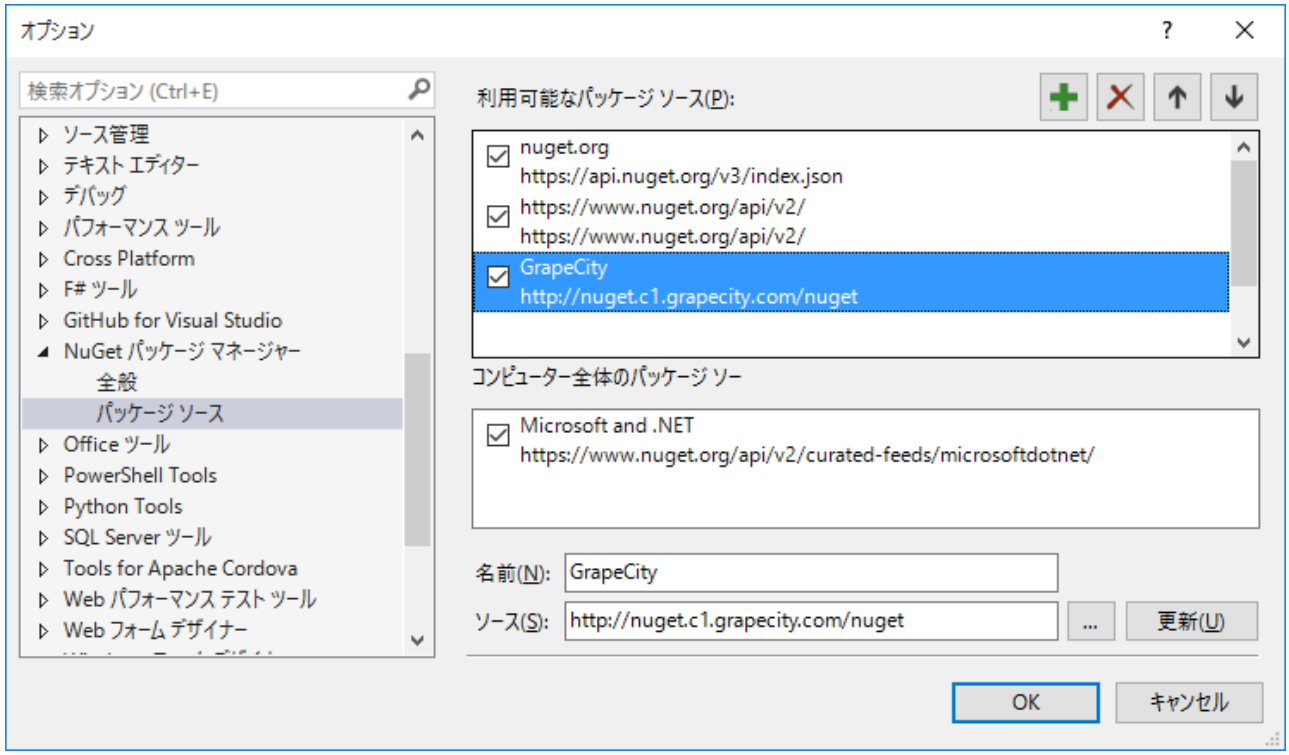
- **手順1：Web APIプロジェクトの構成**
- **手順2：プロジェクトへのレポートファイルの追加**
- **手順3：レポートのルート位置の設定とComponentOne Web APIの使用**
- **手順4：FlexReport Web APIサービスの展開**

手順1：Web APIプロジェクトの構成

Web APIプロジェクトを構成するには、次の手順を実行します。

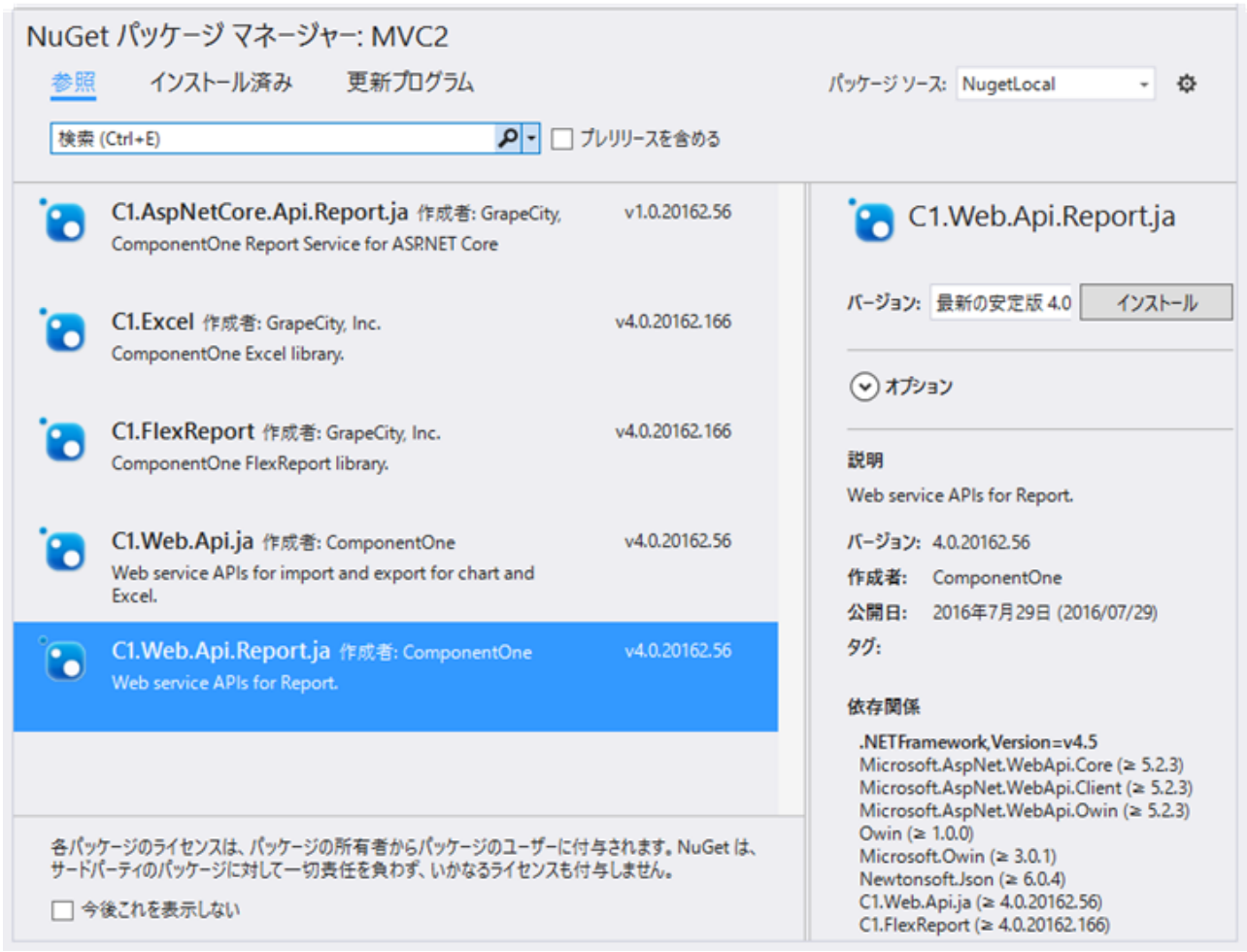
1. 新しいASP.NET Web APIプロジェクトを作成します。
2. nuget.orgからFlexReport Web APIパッケージを追加します。

   MVC をインストールした場合は、Visual Studio内にnuget.orgソースパスが既に設定されています。

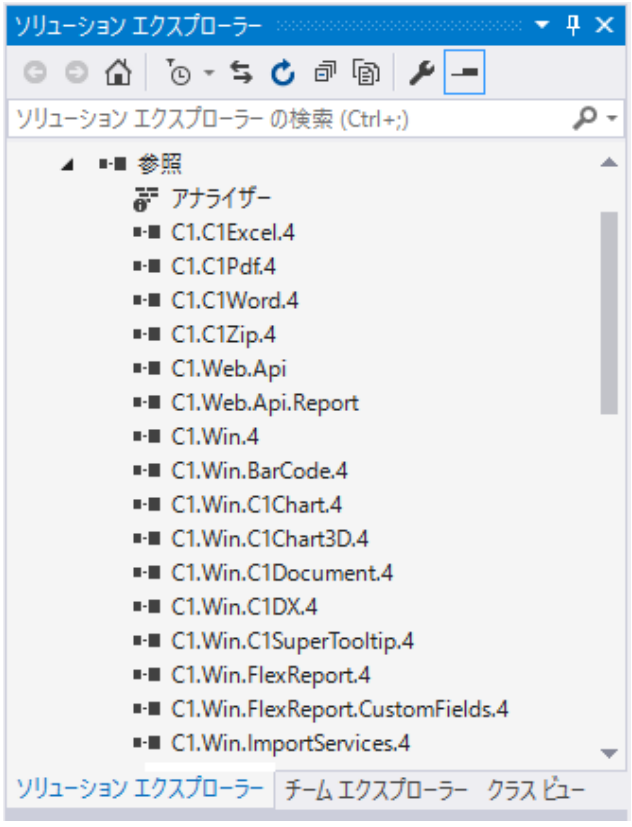


または、**[オプション]**ダイアログボックスにある**[NuGetパッケージマネージャー]**→**[パッケージソース]**オプションからソースパスを手動で追加できます。このダイアログボックスは、**[ツール]**→**[NuGetパッケージマネージャー]**→**[パッケージマネージャーの設定]**を選択すると表示されます。
NuGetパッケージマネージャーにReport Serviceが次の図のようにリストされます。

Copyright © GrapeCity, Inc. All rights reserved.



FlexReport Web APIは、プロジェクトに次の参照を追加します。



Copyright © GrapeCity, Inc. All rights reserved.

3. Web APIアプリケーションにライセンスを付与します。アプリケーションの[プロパティ]フォルダ内に**licenses.licx**ファイルを作成し、それに次のコードを追加します。

licenses.licx

```
C1.Web.Api.LicenseDetector, C1.Web.Api
```

手順1に戻る

手順2：プロジェクトへのレポートファイルの追加

1. アプリケーションに**Files**という名前のフォルダを作成します。
2. それにFlexReport定義ファイルを追加します。

   レポートでローカルデータベース（MDBファイル、MDFファイルなど）を使用している場合は、プロジェクトの**[App_Data]**フォルダにそのデータベースファイルを追加します。ただし、レポートの接続文字列が[App_Data]フォルダを指していること確認してください。

```
<DataSources>
  <DataSource>
    <Name>Main</Name>
    <DataProvider>OLEDB</DataProvider>
    <ConnectionString>Provider=Microsoft.Jet.OLEDB.4.0;Data Source=|DataDirectory|C1Demo.mdb;Persist Security Info=False</ConnectionString>
    <RecordSource>Employees</RecordSource>
    <RecordSourceType>TableDirect</RecordSourceType>
  </DataSource>
</DataSources>
```

手順1に戻る

手順3：レポートのルート位置の設定とComponentOne Web APIの使用

1. **Startup.cs**ファイルを開き、Configurationメソッドに次のコードを追加します。

Startup.cs

```
public void Configuration(IAppBuilder app)
{
    app.UseCors(CorsOptions.AllowAll);
    var folder = GetFullRoot("Files");
    app.AddDiskStorage("root", folder);
    ConfigureAuth(app);
}
```

> **メモ**: app.UseCors()のために、アプリケーションに**Microsoft.Owin.Cors** Nugetパッケージを追加してください。

2. Startupクラスに次のGetFullRoot関数を追加します。

Startup.cs

```
private static string GetFullRoot(string root)
        {
                var applicationBase =
AppDomain.CurrentDomain.SetupInformation.ApplicationBase;
                var fullRoot = Path.GetFullPath(Path.Combine(applicationBase, root));
                if (!fullRoot.EndsWith(Path.DirectorySeparatorChar.ToString(),
StringComparison.Ordinal))
        {
                        fullRoot += Path.DirectorySeparatorChar;

        }
                return fullRoot;
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

3. Web.Configを開き、system.webServerノード内のハンドラの下に次のエントリを追加します。

Web.config

```
<add name="ExtensionlessUrlHandler-Integrated-4.0" path="api/*" verb="*"
type="System.Web.Handlers.TransferRequestHandler"
preCondition="integratedMode,runtimeVersionv4.0" />
```

手順1に戻る

手順4：FlexReport Web APIサービスの展開

1. プロジェクトをコンパイルします。
2. プロジェクトをIISに展開します。ローカルIISでホストされるサービスの場合、Web API URLは **http://localhost/FlexReportwebAPI/api/report** です。

> **メモ**: レポートサービスの**Web API URL**を正しく作成できたら、FlexViewer for MVCやWijmo Viewerを使用して、サービスに保存されているレポートにアクセスして表示することができます。レポートの表示方法については、「FlexViewerでのレポートの表示」を参照してください。

FlexSheet

FlexSheet for MVCは、FlexGridの機能を拡張した高速かつ軽量なコントロールです。Excelと同様の環境をASP.NET Webアプリケーションのユーザーに提供できます。このコントロールは、基本的なスプレッドシート機能をMVCアプリケーションに導入して、数式を操作したり、Excelファイルをロードおよびエクスポートできるようにします。サーバー側の操作を拡張して、リモートデータ連結や、サーバーからのExcelファイルのロードを可能にします。さらに、エンドユーザーは、Excelファイルの表示、編集、保存などをクライアント側で柔軟に行うことができます。

（非連結または連結）FlexSheetアプリケーションを簡単に作成して、ビジネスデータの効果的な表示、編集、分析など、無限の可能性を探ってください。



	A	B	C	D	E	F	G	H
1	¥100.00	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00
2	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00
3	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00
4	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00
5	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00
6	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00
7	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00
8	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00
9	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00
10	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00	¥1,600.00
11	1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700
12	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800
13	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900
14	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000
15	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100
16	1,500	1,600	1,700	1,800	1,900	2,000	2,100	2,200
17	1,600	1,700	1,800	1,900	2,000	2,100	2,200	2,300
18	1,700	1,800	1,900	2,000	2,100	2,200	2,300	2,400

Sheet1

主な特長

FlexSheet の主な機能は以下のとおりです。

- **組み込みの数式のサポート**
  FlexSheetの計算エンジンがサポートする約100の組み込みの数式を自動補完機能付きで利用できます。数式演算は

Copyright © GrapeCity, Inc. All rights reserved.

Microsoft Excelと整合性があり、集計、数学計算、論理、およびテキストを処理できます。

- **マルチシート**
  アプリケーションに複数の独立したワークシートを追加できます。ワークシートは、タブページで切り替えることができます。

- **セルのスタイル**
  各セルにスタイルを適用することで、FlexSheetのセルをカスタマイズできます。FlexSheetのセルには、任意のデータ形式、フォント、水平方向の配置、前景色、塗りつぶし色などを簡単に設定できます。

- **セルの結合**
  FlexSheetコントロールの2つ以上の隣接するセルを簡単に結合できます。結合されたセルには、一番左上のセルのデータだけが含まれます。コンテンツ依存の結合しか許可されていないFlexGridとは異なり、FlexSheetでは隣接するセルを自由にグループにして結合できます。

- **元に戻す/やり直し**
  セルの編集、行/列の挿入/削除、セルスタイルの適用、セルの結合、行/列のサイズ変更、行/列のドラッグアンドドロップなどの複数の操作を元に戻す/やり直すことができます。

- **ドラッグアンドドロップ**
  行/列をドラッグアンドドロップして、現在の行/列を別の行/列に移動またはコピーできます。

- **列の選択**
  列ヘッダーをクリックするだけで、列全体を選択できます。

クイックスタート:FlexSheet

アプリケーションのFlexSheetコントロールは、データソースに連結することも非連結にすることもでき、Excelファイルからロードしたデータを表示することもできます。柔軟なスプレッドシート使用環境が提供されています。リモートで、またはクライアント側で、既存のExcelファイルまたはワークブックをFlexSheetにロードし、データを変更して、それをExcelファイルまたはワークブックファイルとして保存することもできます。

このセクションでは、アプリケーションでFlexSheetコントロールを使用する方法、およびサーバー側でExcelファイルまたはワークブックインスタンスをロードする方法を説明します。

これは、以下の手順で実現できます。

- **手順1:アプリケーションへの使用許諾の供与**
- **手順2:アプリケーションへの関連する参照の追加**
- **手順3:FlexSheetコントロールを使用するためのアプリケーションの設定**
- **手順4:リソースの登録**
- **手順5:アプリケーションへのjszip.jsライブラリの追加**
- **手順6:サーバーへのExcelファイルまたはワークブックの追加**
- **手順7:FlexSheetコントロールの追加**
- **手順8:プロジェクトのビルドおよび実行**

> **メモ**: ASP.NET MVC Editionの**ComponentOneテンプレート**は、自動的に必要なリソースを登録し、関連する参照とパッケージをアプリケーションに追加します。したがって、ComponentOneテンプレートを使用してアプリケーションを作成する場合、上記の手順1～5を実行する必要はありません。

次の図は、サーバーからFlexSheetコントロールにロードされたワークブックのデータを示します。

Copyright © GrapeCity, Inc. All rights reserved.



	A	B	C	D	E	F	G	H	I	J	K	L
1										事務所だけ使用		
2		経費明細書										
3												
4		目的:	商用			添付:	はい			給与期間:	開始日:	3/1/2015
5											終了日:	4/1/2015
6		従業員情報:										
7		名前:	松本 大貴			役職:	管理責任者			SSN	A12345	
8		所属:	営業			マネージャー:	原田 絢乃			従業員ID	E123456	
9												
10		日付	アカウント	記述	ホテル代	交通費	燃料代	食事代	電話代	演芸代	その他	合計
11		3/1/2015	12345678	要人お客様の訪問	¥1,913.00	¥2,145.00	¥1,114.00	¥4,158.00	¥3,413.00	¥2,459.00	¥1,100.00	¥16,302.00
12		3/3/2015	12345678	要人お客様の訪問	¥1,744.00	¥1,658.00	¥1,882.00	¥2,618.00	¥2,583.00	¥2,548.00	¥1,470.00	¥14,503.00
13		3/7/2015	12345678	要人お客様の訪問	¥8,665.00	¥2,109.00	¥1,537.00	¥1,837.00	¥1,253.00	¥1,453.00	¥1,365.00	¥18,219.00
14		3/11/2015	12345678	要人お客様の訪問	¥1,570.00	¥3,290.00	¥1,738.00	¥2,456.00	¥1,046.00	¥2,536.00	¥1,450.00	¥14,086.00
15		3/18/2015	12345678	要人お客様の訪問	¥1,254.00	¥2,498.00	¥2,010.00	¥1,850.00	¥2,201.00	¥4,265.00	¥1,879.00	¥15,957.00
16		3/21/2015	12345678	要人お客様の訪問	¥2,508.00	¥4,501.00	¥1,632.00	¥3,154.00	¥1,495.00	¥2,315.00	¥1,654.00	¥17,259.00
17		3/27/2015	12345678	要人お客様の訪問	¥3,730.00	¥3,569.00	¥1,835.00	¥1,987.00	¥1,296.00	¥1,456.00	¥1,024.00	¥14,897.00
18		合計			¥21,384.00	¥19,770.00	¥11,748.00	¥18,060.00	¥13,287.00	¥17,032.00	¥9,942.00	¥96,326.00
19											小計	¥95,326.00
20											現金進歩	¥1,000.00
21		承認:				注釈:					合計	¥96,326.00

経費明細書　所得報告書

手順1：アプリケーションへの使用許諾の供与

次の手順を実行して、アプリケーションにFlexSheetコントロールを使用するための許諾を与えます。

ASP.NET MVC

1. **ソリューションエクスプローラー**で、プロジェクト名（**MVCFlexSheet**など）をダブルクリックし、**[プロパティ]**ノードを展開します。
2. **licenses.licx**ファイルをダブルクリックして開きます。
3. **licenses.licx**ファイルに以下を追加します。

licenses.licx

```
C1.Web.Mvc.LicenseDetector, C1.Web.Mvc
C1.Web.Mvc.Sheet.LicenseDetector, C1.Web.Mvc.FlexSheet
```

アプリケーションにライセンスを追加する方法については、「ライセンス」を参照してください。

先頭に戻る

手順2：アプリケーションへの必要な参照の追加

ASP.NET

次の手順を実行して、ASP.NET MVC の参照とFlexSheetの参照をプロジェクトに追加します。

1. **ソリューションエクスプローラー**で、**[参照]**を右クリックし、**[参照の追加]**を選択します。
2. **C:\Program Files (x86)\ComponentOne\ASP.NET MVC \bin**を参照します。
3. **C1.Web.Mvc.dll**と**C1.Web.Mvc.FlexSheet.dll**を選択し、**[追加]**をクリックします。
4. **C1.Web.Mvc.dll**と**C1.Web.Mvc.FlexSheet.dll**の**[ローカルコピー]**プロパティを**True**に設定します。

先頭に戻る

手順3：FlexSheetコントロールを使用するためのアプリケーションの設定

Copyright © GrapeCity, Inc. All rights reserved.

次の手順を実行して、FlexSheetを使用するためのアプリケーションの設定を行います。

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開し、web.configファイルをダブルクリックして開きます。
2. <system.web.webPages.razor></system.web.webPages.razor>タグ内の<namespaces></namespaces>タグに、以下のマークアップを追加します。

HTML

```
<add namespace="C1.Web.Mvc" />
<add namespace="C1.Web.Mvc.Fluent" />
<add namespace="C1.Web.Mvc.Sheet" />
<add namespace="C1.Web.Mvc.Sheet.Fluent" />
```

先頭に戻る

手順4：リソースの登録

次の手順を実行して、FlexSheetコントロールを使用するために必要なリソースを登録します。

1. **ソリューションエクスプローラー**で、**[ビュー]**→**[共有]**フォルダを開きます。
2. _Layout.cshtmlをダブルクリックして開きます。
3. <head></head>タグの間に次のコードを追加します。

_Layout.cshtml

```
@Html.C1().Styles()
@Html.C1().Scripts().Basic().FlexSheet()
```

> FlexSheetは、**Office**テーマをサポートしています。しかし、_Layout.cshtmlファイルで**Office**を**default**に変更することで、アプリケーションでデフォルトのテーマを使用することもできます。

リソースを登録する方法については、「リソースの登録」を参照してください。

先頭に戻る

手順5：アプリケーションへのjszip.jsライブラリの追加（CDNリンクの使用）

ExcelファイルのFlexSheetへのロードおよびFlexSheetデータのExcelファイルへの保存は、**jszip.min.js**ファイルに依存しています。そのため、このファイルをアプリケーションに追加し、各ビューまたは**_Layout.cshtml**ファイルの<head>セクションにファイルへの参照を提供する必要があります。次の手順を実行して、**jszip.min.js**ファイルをアプリケーションに追加します。

1. 次のCDNリンクから**jszip.min.js**ファイルをダウンロードします。http://cdnjs.cloudflare.com/ajax/libs/jszip/2.2.1/jszip.min.js
2. jszip.min.jsファイルをアプリケーションの**[Scripts]**フォルダに置きます。
3. ソリューションエクスプローラーで**[Scripts]**フォルダを右クリックし、オプションから**[追加]**→**[既存の項目]**を選択します。
4. **[既存の項目の追加]**ダイアログで、プロジェクト内の**jszip.min.js**の場所を参照します。
5. **jszip.min.js**ファイルを選択し、**[追加]**をクリックしてプロジェクトにファイルを入れます。

先頭に戻る

手順6：サーバーへのExcelファイルまたはワークブックの追加

1. ロードするExcelファイルまたはワークブックファイルがサーバー上に置かれていることを確認します。この例では、ファイルがアプリケーションの**[コンテンツ]**フォルダに置かれています。

   > この例では、サーバーはVisual Studio IIS（インターネットインフォメーションサービス）、クライアントはブラウザとしています。

2. ソリューションエクスプローラーでプロジェクト名を右クリックし、オプションから**[追加]**→**[既存のファイルの追加]**を選択

Copyright © GrapeCity, Inc. All rights reserved.

して、Excelファイルをプロジェクトに追加します。

先頭に戻る

手順7：アプリケーションへのFlexSheetコントロールの追加

FlexSheetコントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します。(例: `ServerLoadController`)。
    2. **[空のMVCコントローラー]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

```
C#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

5. メソッド**Index()**を次のコードに置き換えます。

ServerLoadController.cs

C#

```
C#                                                                copyCode
// GET: ServerLoad
public ActionResult Index()
{
    return View();
}
```

VB

```
VB
' GET: ServerLoad
Public Function Index() As ActionResult
        Return View()
End Function
```

コントローラーのビューの追加

1. 上記の手順で追加したコントローラーで、メソッド`Index()`内にカーソルを置きます。
2. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
3. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
4. **[追加]**をクリックします。コントローラーにビューが追加されました。

Copyright © GrapeCity, Inc. All rights reserved.

5. ソリューションエクスプローラーで、Index.cshtmlをダブルクリックして開きます。
6. **Views\Index.cshtml**ファイルのデフォルトコードを次のコードに置き換えて、**FlexSheet**コントロールを初期化します。.

Index.cshtml copyCode

```
<script src=
"http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
<div>
    @(Html.C1().FlexSheet().Load("~/Content/ExcelFiles/
WorkBook.xlsx").Width(1500).Height(800)
    )
</div>
```

Index.vbhtml

```
@Imports C1.Web.Mvc
@Imports C1.Web.Mvc.FlexSheet.Fluent
@Imports C1.Web.Mvc.FlexSheet
@Imports C1.Web.Mvc.Grid
<script src="http://cdnjs.cloudflare.com/ajax/libs

/jszip/2.5.0/jszip.min.js"></script>
    @(Html.C1().FlexSheet() _
                .Load("~/Content/ExcelFiles/Product.xlsx") _
                .Width("1500px") _
                .Height("800px")
    )
```

先頭に戻る

手順8：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

先頭に戻る

FlexSheetの使用

This section comprises all the features and auxiliary functionalities offered by FlexSheet.

セルの結合
: Learn how to merge cells at run-time and through code.

クライアント側のExcelファイルのロードおよび保存
: Learn how to load and save data from an Excel file or workbook on client-side, in FlexSheet.

コンテキストメニューの使用
: Learn about the context menu available in FlexSheet.

データの連結
: Learn how to perform data binding in FlexSheet.

ドラッグアンドドロップ
: Learn how to handle drag and drop operation in FlexSheet.

フィルタリング
: Learn how to enable Filtering in the FlexSheet.

セルの書式

Copyright © GrapeCity, Inc. All rights reserved.

Learn how to apply styles on cells, in FlexSheet.

FlexSheetの数式
Learn about the in-built formulas and how to use them at runtime and in code.

固定のセル
Learn how to freeze cells in the FlexSheet.

クライアント側でのJSONのロードと保存
Learn how to use workbook object model to load Json string into FlexSheet control.

複数のヘッダー
Learn how to add or remove row and column headers in the FlexSheet.

リモートでロードおよび保存
Learn how to remotely load data from excel file or workbook in FlexSheet.

ソート
Learn how to enable Sorting in the FlexSheet.

表
Learn how to add a table in the FlexSheet control.

非連結 シート
Learn how to add unbound FlexSheet to an MVC application.

セルの結合

コンテンツ依存のセル結合を行うFlexGridとは異なり、FlexSheetはクライアント側でExcelと同様のセル結合をサポートします。FlexSheetでは、任意のデータ/値を含む隣接セルを1つに結合できます。セルを結合すると、最も左上にあるセルのコンテンツ/値/データが結合後のセルに保持されます。

シート内のいくつかのセルを選択して1つに結合するには、**mergeRange()**メソッドを呼び出します。ただし、選択範囲に既に結合されたセルが含まれる場合、mergeRangeメソッドは結合されたセルの結合を解除します。

次のセクションでは、実行時およびコードからセルを結合する方法を説明します。

> **メモ:** FlexSheetは、FlexGridと同様のサーバー側の結合もサポートします。

実行時にセルの結合

ワークシートのセルを結合すると、データをわかりやすく表示できます。コンテンツが同じセルを結合してまとめると、通常よりデータを読みやすくできます。

ワークシートデータの一部を結合しないときれいに表示できないという例はよくあります。たとえば、ある会社の四半期ごとの売上を地域別製品別に表示して分析するとします。このような場合は、四半期ごとに見出しのセルを結合すると、表示を改善できます。次はこのシナリオの具体例です。実行時にボタンをクリックすると、セル結合が実行されます。ここでは、結合するセルを選択し、[Merge]ボタンをクリックすると、**mergeRange()**が呼び出されます。

次の例のFlexSheetは、サーバーからロードされたワークブックファイルのデータを表示します。

Copyright © GrapeCity, Inc. All rights reserved.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	四半期ごとに販売の図 2015-2016												
2		一四半期			二四半期			三四半期			四半分		
3		ガジェット	ウィジェット	道具	ガジェット	ウィジェット	道具	ガジェット	ウィジェット	道具	ガジェット	ウィジェット	道具
4	ドイツ	¥226,800	¥576,428.00	¥494,070	¥196,700	¥666,400	¥515,980	¥213,268	¥466,400	¥511,600	¥500,100	¥430,305	¥551,900
5	ギリシャ	¥16,000	¥126,100	¥163,066	¥20,000	¥196,400	¥154,060	¥34,000	¥110,400	¥156,060	¥62,800	¥91,800	¥183,060
6	イタリア	¥236,100	¥89,800	¥620,020	¥426,800	¥81,800	¥534,300	¥324,060	¥83,800	¥812,100	¥145,800	¥216,100	¥612,300
7	日本	¥73,120	¥146,600.00	¥21,000	¥75,200	¥143,600	¥25,300	¥34,100	¥134,600	¥27,000	¥56,210	¥166,004	¥26,090
8	アメリカ	¥449,200	¥926,008	¥313,800	¥549,100	¥930,008	¥231,100	¥513,900	¥931,108	¥203,800	¥592,030	¥831,400	¥213,800
9	イギリス	¥200,000	¥316,600	¥41,400	¥206,900	¥362,600	¥31,500	¥370,000	¥326,600	¥34,070	¥280,010	¥426,100	¥37,500
10													

次のコード例は、実行時にFlexSheetコントロールでセル結合を行う方法を示します。

コードの場合

MergingController.cs

C# copyCode

```
public class MergeController : Controller
{
    // GET: Merge
    public ActionResult MergeIndex()
    {
        return View();
    }
}
```

Merging.cshtml

Razor copyCode

```
<script src=
"http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
<script>
    function mergeSelection() {
        var flex = wijmo.Control.getControl("#mSheet");
        flex.mergeRange();
    }
    function setBold() {
        var flex = wijmo.Control.getControl("#mSheet");
        flex.applyCellsStyle({ fontWeight: 'bold' });
    }
    function changeFont() {
        var flex = wijmo.Control.getControl("#mSheet");
        flex.applyCellsStyle({ fontFamily: 'Unicode' });
    }
    function setColor() {
        var flex = wijmo.Control.getControl("#mSheet");
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        flex.applyCellsStyle({ backgroundColor: '#FFC966' });
    }
    function align() {
        var flex = wijmo.Control.getControl("#mSheet");
        flex.applyCellsStyle({ textAlign: 'Center' });
    }
</script>

<div>
    <input id="font" type="button" onclick="changeFont()"
value="フォントを変更" />
    <input id="bold" type="button" onclick="setBold()"
value="大胆" />
    <input id="cellColor" type="button" onclick="setColor()"
value="色" />
    <input id="merge" type="button" onclick="mergeSelection()"
value="連結" />
    <input id="textAlign" type="button" onclick="align()"
value="中央に排す" />
    <br /><br />
    @(Html.C1().FlexSheet().Height("500px").Width("1500px")
.Load("~/Content/FlexSheet/QuarterlyData.xlsx").Id("mSheet")
    )
</div>
```

先頭に戻る

コードでセルの結合

FlexSheetコントロールでは、**mergeRange**()メソッドに結合する**CellRange**を指定することでセルをコードで結合するが可能です。このセクションでは、「実行時にセルの結合」と同じシナリオを使用しています。

ここでは、CellRangeにて、結合する隣接セルのグループをセル範囲の最初行列と最終行列のインデックスで指定します。

メモ: 次の例のFlexSheetではサーバーから読み込まれたWorkbookファイルのデータが使用されています。



Copyright © GrapeCity, Inc. All rights reserved.

次のコードは、FlexSheetコントロールでコードによってセル結合を有効にする方法を示します。

コードの場合

MergeController.cs

C# copyCode

```
public class CellMergeController : Controller
{
    // GET: CellMerge
    public ActionResult CodeMrgeIndex()
    {
        return View();
    }
}
```

Merging.cshtml

Razor copyCode

```
<script src=
"http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
<script>
function mergeSel() {
        //コードで所定の範囲をマージします。
        var flex = wijmo.Control.getControl("#mSheet");
        flex.mergeRange(new wijmo.grid.CellRange(0, 0, 0, 6));
        flex.mergeRange(new wijmo.grid.CellRange(1, 1, 1, 3));
        flex.mergeRange(new wijmo.grid.CellRange(1, 4, 1, 6));
        flex.mergeRange(new wijmo.grid.CellRange(1, 7, 1, 9));
        flex.mergeRange(new wijmo.grid.CellRange(1, 10, 1, 12));
        flex.refresh();
    }
</script>
<div>
    <input id="merge" type="button" onclick="mergeSel()
" value="Merge" />
    <br /><br />
    @(Html.C1().FlexSheet().Height("500").Width("1500")
.Load("~/Content/QuarterlyData.xlsx").Id("mSheet")
    )
</div>
```

先頭に戻る

クライアント側のExcelファイルのロードおよび保存

FlexSheetは、クライアントから提供されるExcelファイルまたはワークブックからのデータのロードをサポートします。さらに、FlexSheetのデータをクライアント側のExcelファイルまたはワークブックに保存できます。

Copyright © GrapeCity, Inc. All rights reserved.

次のコード例は、FlexSheetでクライアントからのExcelのロードを行う方法を示します。次の例では、ボタンクリックでクライアントからのExcelのロードが発生します。FlexSheetにExcelデータをロードするためのボタンをユーザーがクリックすると、**Load**メソッドが呼び出されます。

> クライアント側でExcelをロードおよび保存するには、FlexSheetに**jszip.min.js**ファイルが必要です。そのため、このファイルをアプリケーションに追加し、各ビューまたは**_Layout.cshtml**ファイルの<head>セクションにファイルへの参照を提供する必要があります。

コードの場合

ClientLoadController.cs

C# copyCode

```
public class ClientLoadController : Controller
{
    // GET: /<controller>/
    public ActionResult Index()
    {
        return View();
    }
}
```

ClientLoad.cshtml

Razor copyCode

```
@using C1MvcFSheetNew.Models;
@model IEnumerable<Sale>

<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
    <script>
        function load() {
            var flex = wijmo.Control.getControl("#clientLoadSheet");
            var fileInput = wijmo.getElement('#importFile');
            if (flex && fileInput.files[0]) {
                flex.load(fileInput.files[0]);
            }
        }
    </script>

    <div>
        <input type="file" class="form-control" id="importFile"
accept="application/vnd.openxmlformats-officedocument.spreadsheetml.sheet" />
        <button class="btn btn-default" onclick="load()">Load</button>
        <br /><br />
        @(Html.C1().FlexSheet().CssClass("flexSheet").Id("clientLoadSheet")
        .SelectedSheetIndex(0).Width("500px").Height("700px").AddBoundSheet(sheet =>
                sheet.Bind(cv =>
                    cv.Bind(Model).DisableServerRead(true)))
        .AddUnboundSheet("Unbound", 20, 8))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    </div>
```

先頭に戻る

クライアント側のExcelファイルへのFlexSheetデータの保存

クライアント側では、Excelファイルまたはワークブックへの FlexSheetデータの保存もサポートされます。これは、**save**メソッドによって行われます。次のコード例は、クライアント側のExcelファイルへのFlexSheetデータの保存を行う方法を示します。次の例では、ボタンクリックによってsaveメソッドが呼び出されます。

コードの場合

ClientSaveController.cs

C# copyCode

```
public class ClientSaveController : Controller
{
    // GET: /<controller>/
    public static List<Sale> SALES = Sale.GetData(50).ToList();
    public ActionResult Index()
    {
        return View(SALES);
    }
```

ClientSaving.cshtml

Razor copyCode

```
@using MVCFlexSheet.Models;
@model IEnumerable<Sale>

<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
    <script>
        function save() {
            var flex = wijmo.Control.getControl("#clientSaveSheet");
            var fileNameInput = wijmo.getElement("#fileName");
            var fileName = 'FlexSheet.xlsx';
            flex.save(fileName);
        }
    </script>

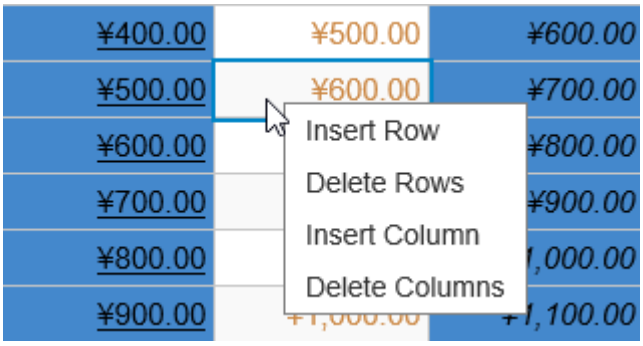
    <div>
        <button class="btn btn-default" onclick="save()">保存</button>
        <br /><br />
        @(Html.C1().FlexSheet().CssClass("flexSheet").Id("clientSaveSheet")
        .SelectedSheetIndex(0).Width("500px").Height("700px").AddBoundSheet(sheet =>
                sheet.Bind(cv =>
                    cv.Bind(Model).DisableServerRead(true)))
        .AddUnboundSheet("Unbound", 20, 8))
    </div>
```


Copyright © GrapeCity, Inc. All rights reserved.

先頭に戻る

コンテキストメニューの使用

FlexSheetコントロールは、Microsoft Excelと同様の使い慣れた機能に加えて、ユーザーエクスペリエンスを向上させる機能を提供しています。FlexSheetのコンテキストメニューは、このようなExcel風の機能の1つです。右クリックから、シートに行や列を挿入したり削除することができます。



FlexSheetのセルまたは行ヘッダー/列ヘッダーを右クリックすると、コンテキストメニューが表示されます。ここから、シートに次の操作を実行できます。

コンテキストメニューの操作	説明
行の挿入	アクティブな行の上に新しい行を追加します。
行の削除	アクティブな行を削除します。
列の挿入	アクティブな列の前に新しい列を追加します。
列の削除	アクティブな列を削除します。

先頭に戻る

データの連結

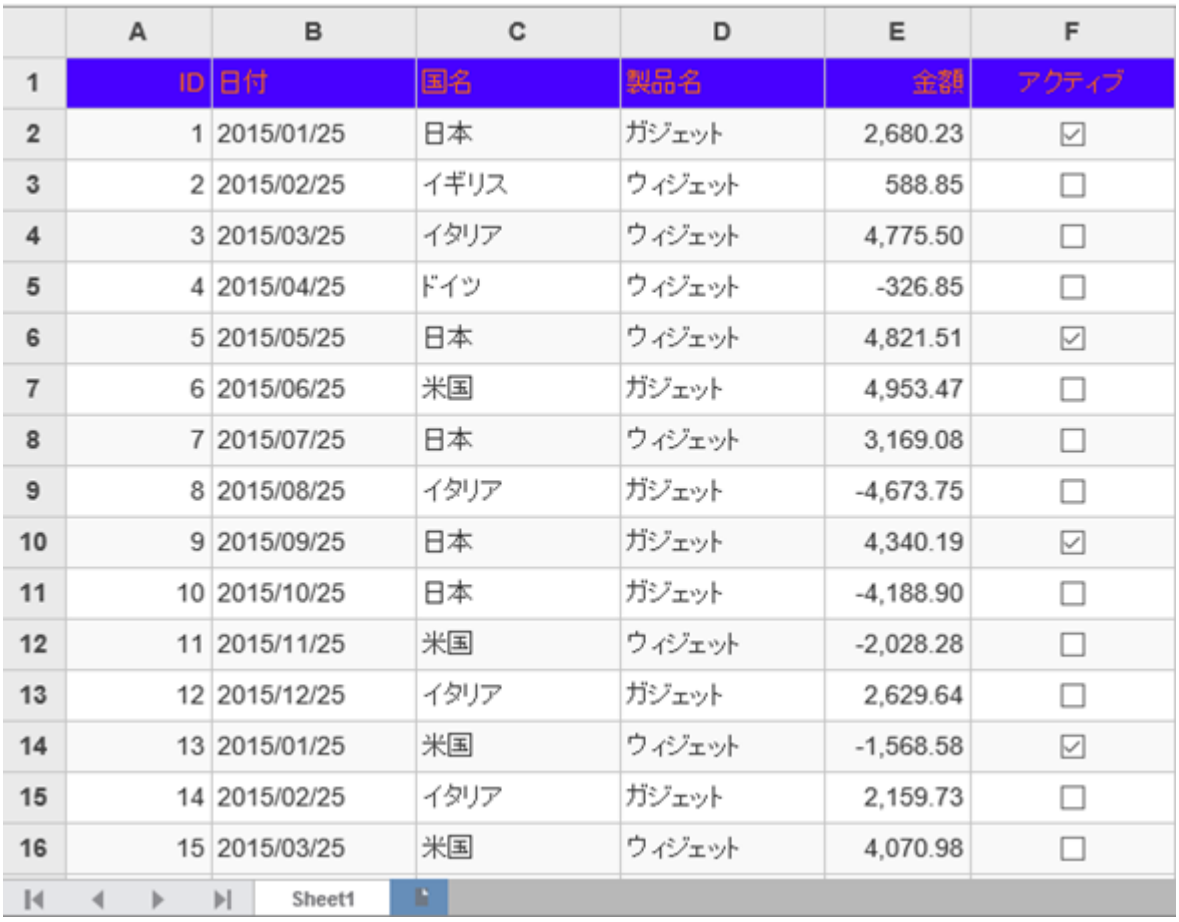
FlexSheetは、C1JsonRequestを使用してデータを直接取得できる、モデル連結とリモートデータ連結をサポートします。連結モードでは、列を定義でき、FlexGridと同様にFlexSheetをデータソースに連結します。

リモート連結では、**Bind**プロパティを使用して**アクションURL**メソッドを渡すことで、シートをコレクションに連結します。リモートデータ連結は、サーバー、テーブル、および列を含むリモートデータURLを指定します。返される配列は、CollectionViewオブジェクトのデータソースとして機能します。**CollectionViewHelper**クラスは、コレクションの編集、フィルタ処理、グループ化、ソートの各サービスを支援します。

このトピックでは、既存のデータソースからデータを取得する方法について説明します。これは、データをダッシュボードとして表すデータ集約型のアプリケーションやシナリオを開発する際に役立ちます。

次の図は、連結モードでモデルSale.csからデータをフェッチするFlexSheetです。

Copyright © GrapeCity, Inc. All rights reserved.



	A	B	C	D	E	F
1	ID	日付	国名	製品名	金額	アクティブ
2	1	2015/01/25	日本	ガジェット	2,680.23	☑
3	2	2015/02/25	イギリス	ウィジェット	588.85	☐
4	3	2015/03/25	イタリア	ウィジェット	4,775.50	☐
5	4	2015/04/25	ドイツ	ウィジェット	-326.85	☐
6	5	2015/05/25	日本	ウィジェット	4,821.51	☑
7	6	2015/06/25	米国	ガジェット	4,953.47	☐
8	7	2015/07/25	日本	ウィジェット	3,169.08	☐
9	8	2015/08/25	イタリア	ガジェット	-4,673.75	☐
10	9	2015/09/25	日本	ガジェット	4,340.19	☑
11	10	2015/10/25	日本	ガジェット	-4,188.90	☐
12	11	2015/11/25	米国	ウィジェット	-2,028.28	☐
13	12	2015/12/25	イタリア	ガジェット	2,629.64	☐
14	13	2015/01/25	米国	ウィジェット	-1,568.58	☑
15	14	2015/02/25	イタリア	ガジェット	2,159.73	☐
16	15	2015/03/25	米国	ウィジェット	4,070.98	☐

Sheet1

次のコード例は、リモートデータソースからデータをフェッチするためにFlexSheetを連結する方法を示します。

コードの場合

Sale.csクラスを[モデル]フォルダに追加します。

Model

Sale.cs copyCode

```
public  class Sale
  {
      public int ID { get; set; }
      public DateTime 日付 { get; set; }
      public string 国名 { get; set; }
      public string 製品名 { get; set; }
      public double 金額 { get; set; }
      public bool アクティブ { get; set; }

      private static List<string> COUNTRIES = new List<string> { "米国", "ドイツ", "イギリス", "日本", "イタリア", "ギリシャ" };
      private static List<string> PRODUCTS = new List<string> { "ウィジェット", "ガジェット", "道具" };

      /// <summary>
      /// データを取得します。
      /// </summary>
      /// <param name="total"></param>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        /// <returns></returns>
        public static IEnumerable<Sale> GetData(int total)
        {
            var rand = new Random(0);
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var date = new DateTime(2015, i % 12 + 1, 25);
                //string.Format("yyyy年-MM月-dd日", date);

                return new Sale
                {
                    ID = i + 1,
                    日付 = date,
                    国名 = country,
                    製品名 = product,
                    金額 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    アクティブ = (i % 4 == 0)
                };
            });
            return list;
        }

        public static List<string> GetCountries()
        {
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }

        public static List<string> GetProducts()
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
    }
```

DataBindingController.cs

C# copyCode

```
public class DataBindingController : Controller
{
    public static List<Sale> SALES = Sale.GetData(50).ToList();

    public ActionResult RemoteBind([C1JsonRequest] CollectionViewRequest<Sale>
requestData)
    {
        return this.C1Json(CollectionViewHelper.Read(requestData, Sale.GetData(50)));
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public ActionResult Index()
    {
        return View(SALES);
    }
}
```

DataBinding.cshtml

Razor copyCode

```
@using MVCFlexSheet.Models;
@model IEnumerable<Sale>

<div>
    @(Html.C1().FlexSheet().CssClass("flexSheet").Height("700px").Width("700px")
          .AddBoundSheet(cv =>
                              cv.Bind(Url.Action("RemoteBind")))
    )
</div>
```

先頭に戻る

FlexSheetのローカルモデル連結

URLを渡すリモート連結とは別に、FlexSheetのデータ連結は、Bindプロパティでモデルをローカルに渡して実行することもできます。

次のコード例は、ローカルデータソースからデータをフェッチするためにFlexSheetコントロールを連結する方法を示します。

コードの場合

前述のように、モデルフォルダにSale.csモデルクラスを作成します。

ModelBindingController.cs

C# copyCode

```
public partial class ModelBindController : Controller
{
    // GET: ModelBind
    public static List<Sale> SALES = Sale.GetData(15).ToList();

    public ActionResult ModelDBIndex()
    {
        return View(SALES);
    }
}
```

ModelBinding.cshtml

Razor copyCode

```
@model IEnumerable<Sale>
<div>
    @(Html.C1().FlexSheet().CssClass("flexSheet").Id("boundSheet")
        .AddBoundSheet(sheet =>
```


Copyright © GrapeCity, Inc. All rights reserved.

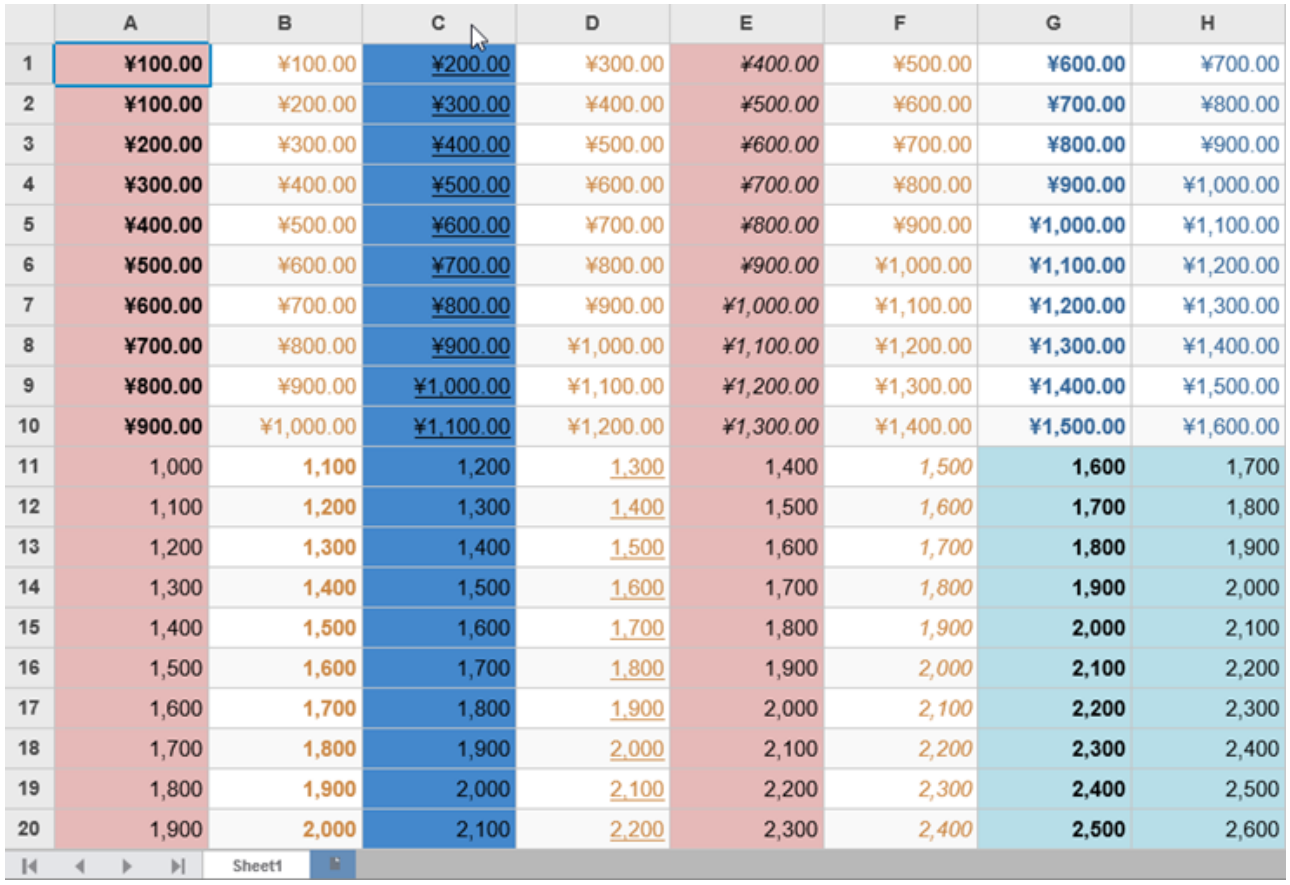
```
            sheet.Bind(cv =>
                        cv.Bind(Model).DisableServerRead(true)))
                    )
</div>
```

ドラッグアンドドロップ

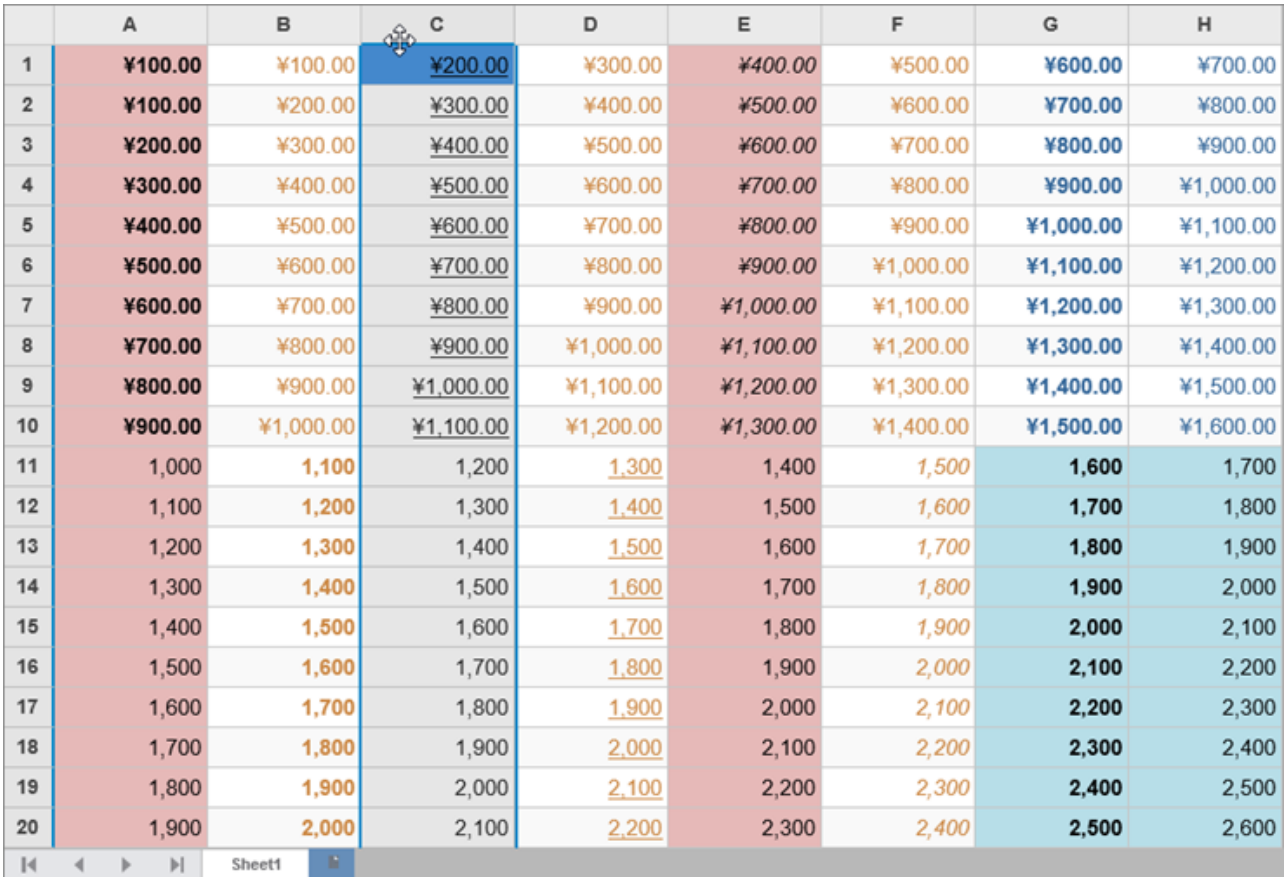
FlexSheetには、行と列のドラッグアンドドロップ編集のサポートが組み込まれています。マウスで行または列を選択し、そのコンテンツを別の行または列に簡単にドラッグして移動できます。これは、それらの行や列のコンテンツだけでなく、セルスタイルもコピーして移動します。

ドラッグアンドドロップ操作を処理するには、次の手順を実行するだけです。

1. マウスポインタが移動ポインタに変わるように、行/列範囲の境界にマウスポインタを置きます .



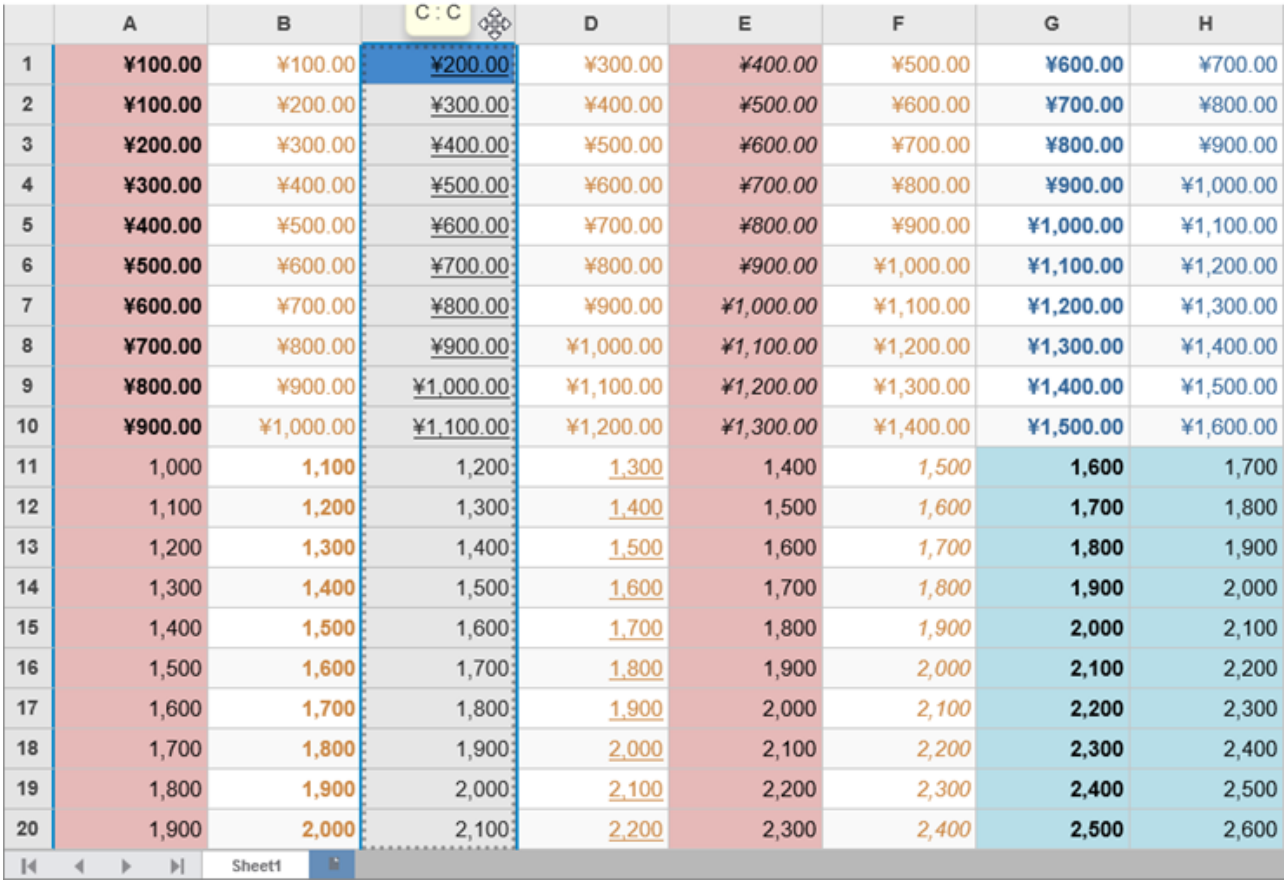
Copyright © GrapeCity, Inc. All rights reserved.



	A	B	C	D	E	F	G	H
1	¥100.00	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00
2	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00
3	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00
4	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00
5	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00
6	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00
7	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00
8	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00
9	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00
10	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00	¥1,600.00
11	1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700
12	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800
13	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900
14	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000
15	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100
16	1,500	1,600	1,700	1,800	1,900	2,000	2,100	2,200
17	1,600	1,700	1,800	1,900	2,000	2,100	2,200	2,300
18	1,700	1,800	1,900	2,000	2,100	2,200	2,300	2,400
19	1,800	1,900	2,000	2,100	2,200	2,300	2,400	2,500
20	1,900	2,000	2,100	2,200	2,300	2,400	2,500	2,600

Sheet1

2. [Ctrl]キーまたは[Shift]キー押しながら、またはキーを押さずに、行/列を選択してマウスを押したままにします。

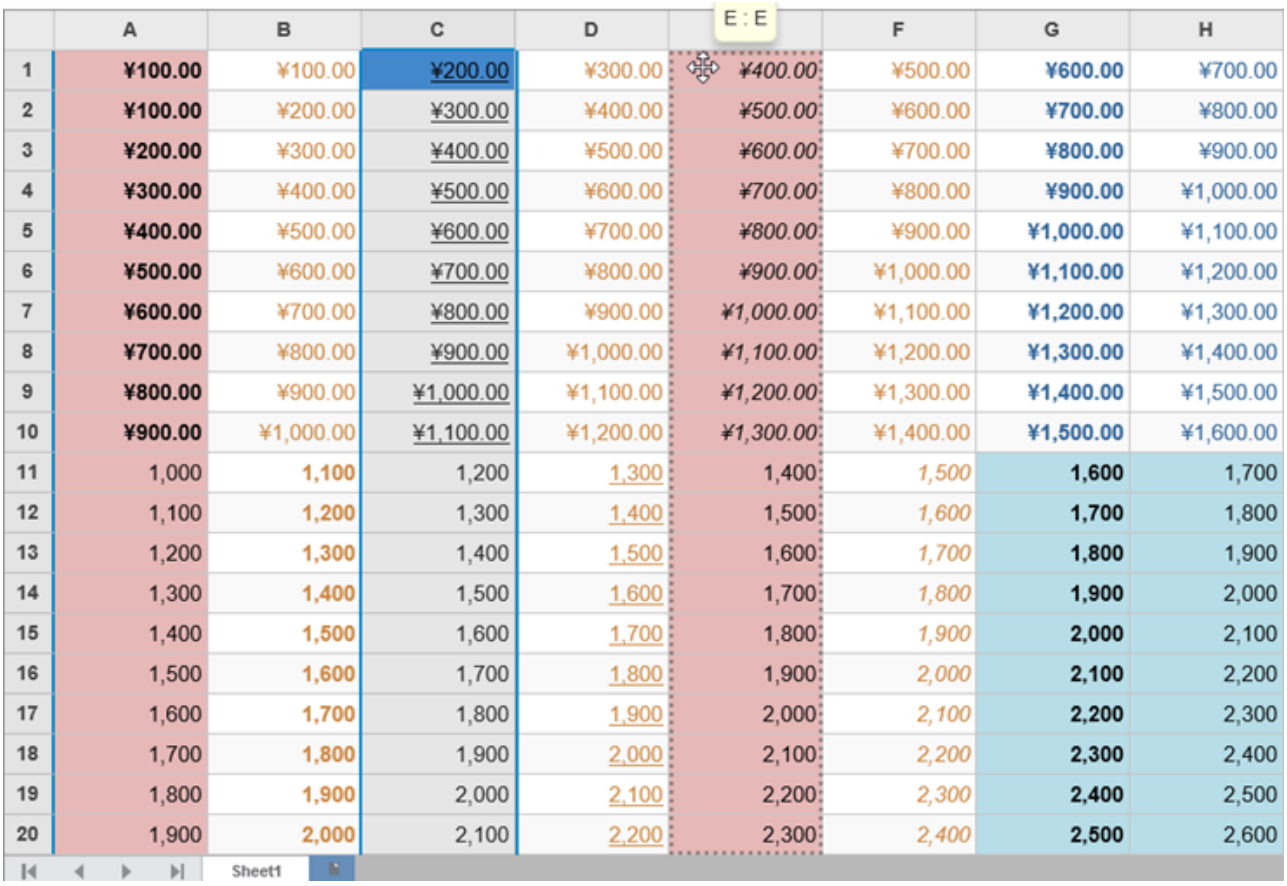


	A	B	C : C	D	E	F	G	H
1	¥100.00	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00
2	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00
3	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00
4	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00
5	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00
6	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00
7	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00
8	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00
9	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00
10	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00	¥1,600.00
11	1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700
12	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800
13	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900
14	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000
15	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100
16	1,500	1,600	1,700	1,800	1,900	2,000	2,100	2,200
17	1,600	1,700	1,800	1,900	2,000	2,100	2,200	2,300
18	1,700	1,800	1,900	2,000	2,100	2,200	2,300	2,400
19	1,800	1,900	2,000	2,100	2,200	2,300	2,400	2,500
20	1,900	2,000	2,100	2,200	2,300	2,400	2,500	2,600

Sheet1

3. 目的の行/列の位置まで選択内容をドラッグします。

Copyright © GrapeCity, Inc. All rights reserved.

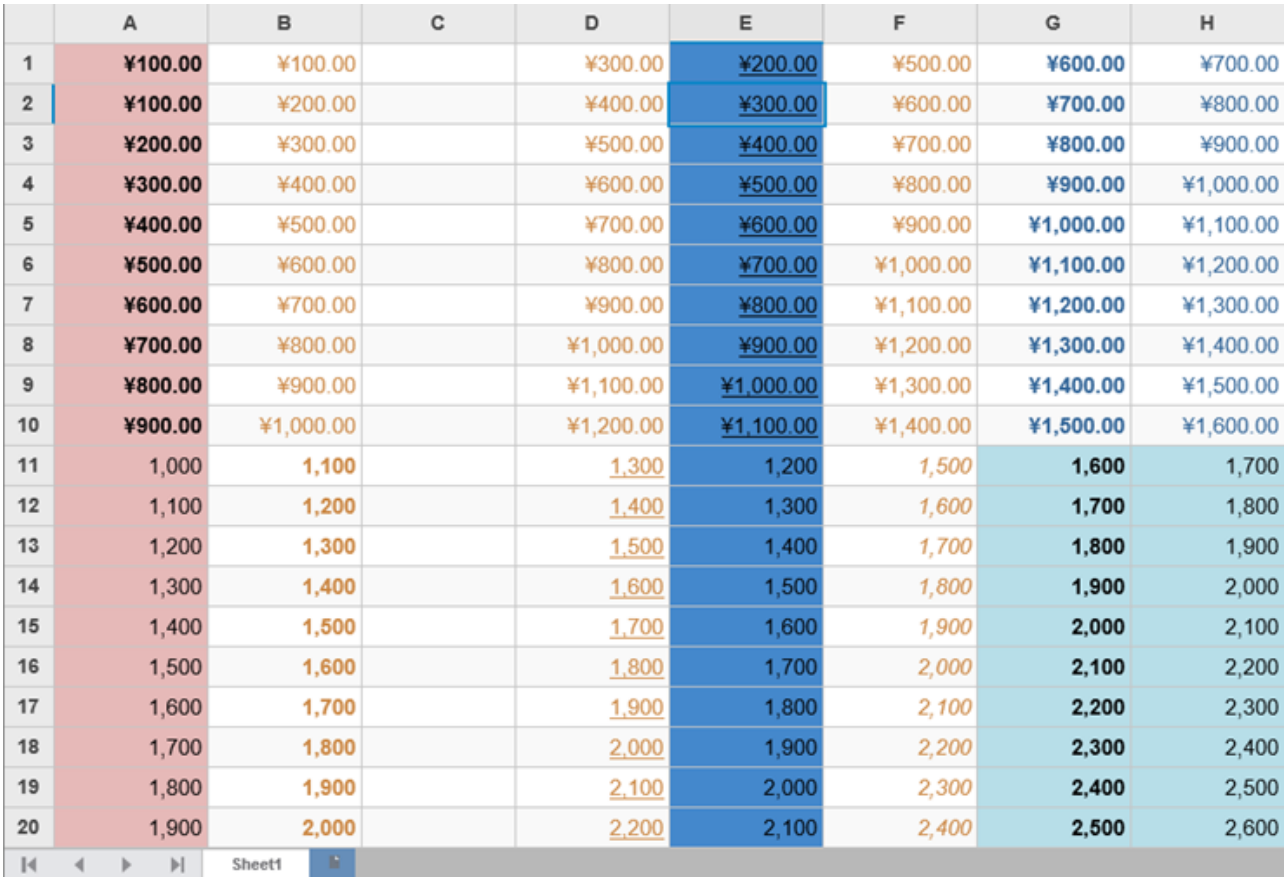


	A	B	C	D	E	F	G	H
1	¥100.00	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00
2	¥100.00	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00
3	¥200.00	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00
4	¥300.00	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00
5	¥400.00	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00
6	¥500.00	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00
7	¥600.00	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00
8	¥700.00	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00
9	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00
10	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,300.00	¥1,400.00	¥1,500.00	¥1,600.00
11	1,000	1,100	1,200	1,300	1,400	1,500	1,600	1,700
12	1,100	1,200	1,300	1,400	1,500	1,600	1,700	1,800
13	1,200	1,300	1,400	1,500	1,600	1,700	1,800	1,900
14	1,300	1,400	1,500	1,600	1,700	1,800	1,900	2,000
15	1,400	1,500	1,600	1,700	1,800	1,900	2,000	2,100
16	1,500	1,600	1,700	1,800	1,900	2,000	2,100	2,200
17	1,600	1,700	1,800	1,900	2,000	2,100	2,200	2,300
18	1,700	1,800	1,900	2,000	2,100	2,200	2,300	2,400
19	1,800	1,900	2,000	2,100	2,200	2,300	2,400	2,500
20	1,900	2,000	2,100	2,200	2,300	2,400	2,500	2,600

次のように機能します。

- キーを何も押さずにドラッグアンドドロップすると、選択した列または行のコンテンツがドロップ先の列または行に移動します。これにより、選択した行/列は空になります。

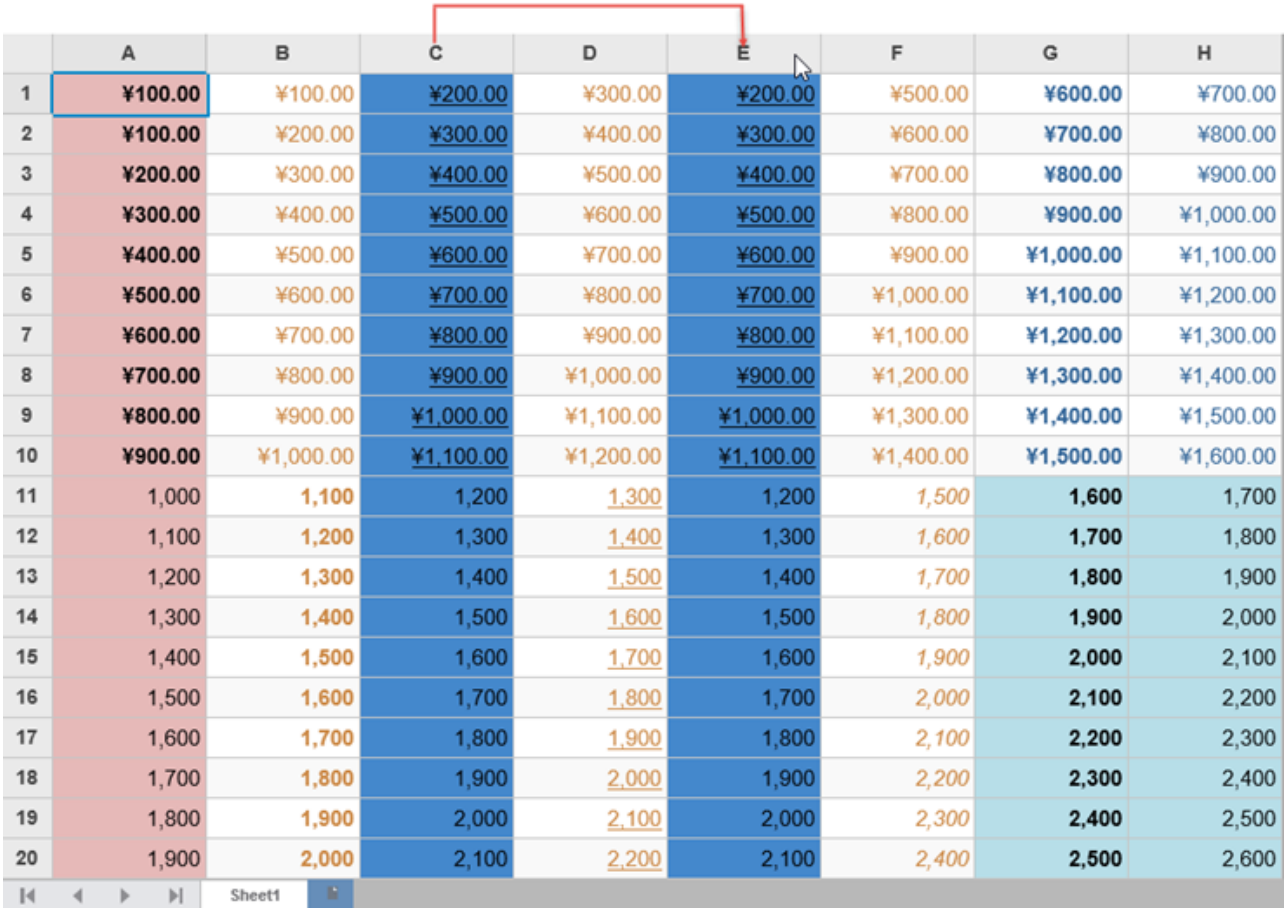
Copyright © GrapeCity, Inc. All rights reserved.



	A	B	C	D	E	F	G	H
1	¥100.00	¥100.00		¥300.00	¥200.00	¥500.00	¥600.00	¥700.00
2	¥100.00	¥200.00		¥400.00	¥300.00	¥600.00	¥700.00	¥800.00
3	¥200.00	¥300.00		¥500.00	¥400.00	¥700.00	¥800.00	¥900.00
4	¥300.00	¥400.00		¥600.00	¥500.00	¥800.00	¥900.00	¥1,000.00
5	¥400.00	¥500.00		¥700.00	¥600.00	¥900.00	¥1,000.00	¥1,100.00
6	¥500.00	¥600.00		¥800.00	¥700.00	¥1,000.00	¥1,100.00	¥1,200.00
7	¥600.00	¥700.00		¥900.00	¥800.00	¥1,100.00	¥1,200.00	¥1,300.00
8	¥700.00	¥800.00		¥1,000.00	¥900.00	¥1,200.00	¥1,300.00	¥1,400.00
9	¥800.00	¥900.00		¥1,100.00	¥1,000.00	¥1,300.00	¥1,400.00	¥1,500.00
10	¥900.00	¥1,000.00		¥1,200.00	¥1,100.00	¥1,400.00	¥1,500.00	¥1,600.00
11	1,000	1,100		1,300	1,200	1,500	1,600	1,700
12	1,100	1,200		1,400	1,300	1,600	1,700	1,800
13	1,200	1,300		1,500	1,400	1,700	1,800	1,900
14	1,300	1,400		1,600	1,500	1,800	1,900	2,000
15	1,400	1,500		1,700	1,600	1,900	2,000	2,100
16	1,500	1,600		1,800	1,700	2,000	2,100	2,200
17	1,600	1,700		1,900	1,800	2,100	2,200	2,300
18	1,700	1,800		2,000	1,900	2,200	2,300	2,400
19	1,800	1,900		2,100	2,000	2,300	2,400	2,500
20	1,900	2,000		2,200	2,100	2,400	2,500	2,600

Sheet1

- **[Ctrl]**キーを押しながらドラッグアンドドロップすると、選択した列または行がドロップ先の列または行にコピーされます。これにより、選択した列/行のコンテンツを維持したまま、ドロップ先の列/行のコンテンツを書き換えることができます。



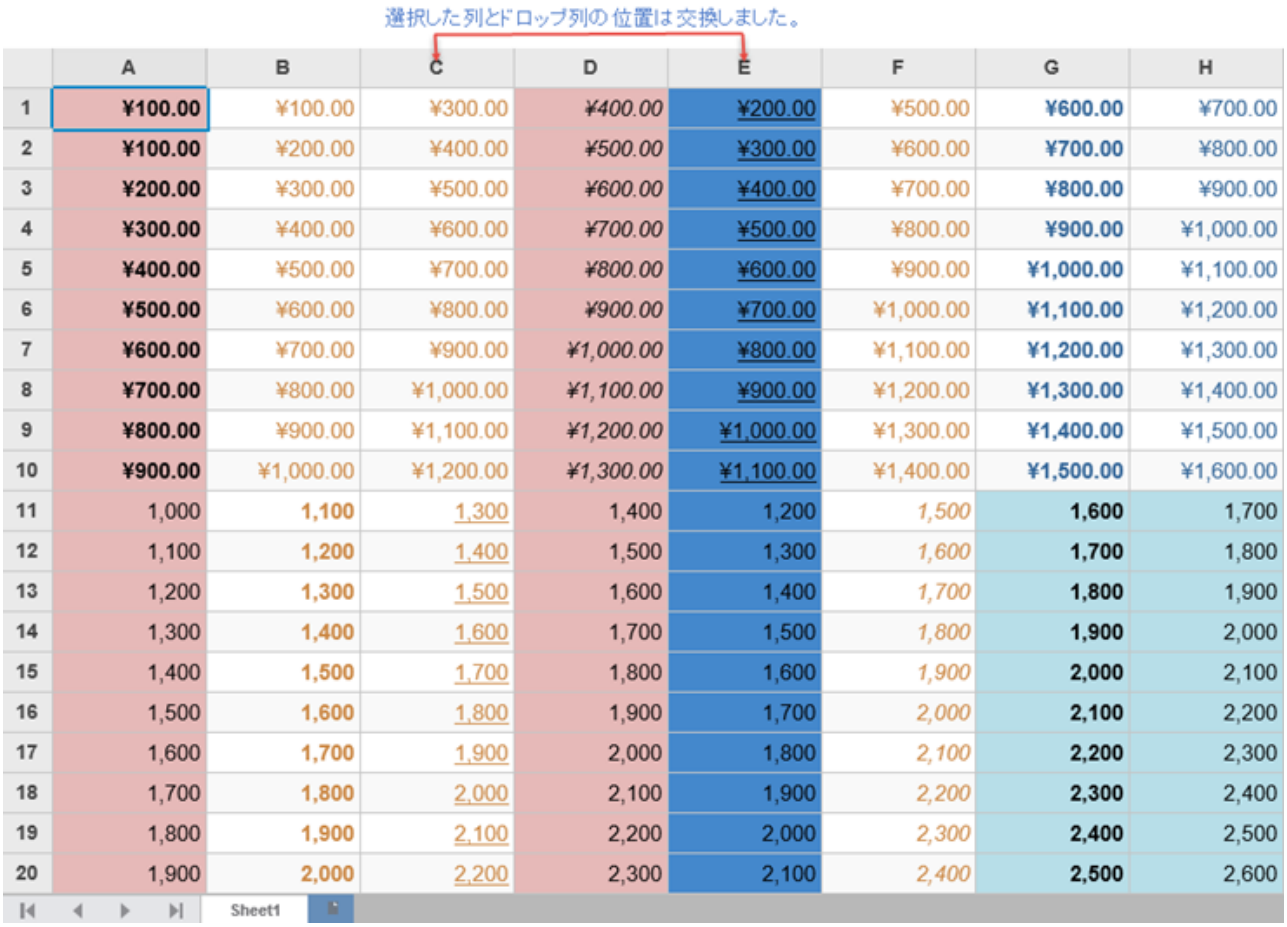
	A	B	C	D	E	F	G	H
1	¥100.00	¥100.00	¥200.00	¥300.00	¥200.00	¥500.00	¥600.00	¥700.00
2	¥100.00	¥200.00	¥300.00	¥400.00	¥300.00	¥600.00	¥700.00	¥800.00
3	¥200.00	¥300.00	¥400.00	¥500.00	¥400.00	¥700.00	¥800.00	¥900.00
4	¥300.00	¥400.00	¥500.00	¥600.00	¥500.00	¥800.00	¥900.00	¥1,000.00
5	¥400.00	¥500.00	¥600.00	¥700.00	¥600.00	¥900.00	¥1,000.00	¥1,100.00
6	¥500.00	¥600.00	¥700.00	¥800.00	¥700.00	¥1,000.00	¥1,100.00	¥1,200.00
7	¥600.00	¥700.00	¥800.00	¥900.00	¥800.00	¥1,100.00	¥1,200.00	¥1,300.00
8	¥700.00	¥800.00	¥900.00	¥1,000.00	¥900.00	¥1,200.00	¥1,300.00	¥1,400.00
9	¥800.00	¥900.00	¥1,000.00	¥1,100.00	¥1,000.00	¥1,300.00	¥1,400.00	¥1,500.00
10	¥900.00	¥1,000.00	¥1,100.00	¥1,200.00	¥1,100.00	¥1,400.00	¥1,500.00	¥1,600.00
11	1,000	1,100	1,200	1,300	1,200	1,500	1,600	1,700
12	1,100	1,200	1,300	1,400	1,300	1,600	1,700	1,800
13	1,200	1,300	1,400	1,500	1,400	1,700	1,800	1,900
14	1,300	1,400	1,500	1,600	1,500	1,800	1,900	2,000
15	1,400	1,500	1,600	1,700	1,600	1,900	2,000	2,100
16	1,500	1,600	1,700	1,800	1,700	2,000	2,100	2,200
17	1,600	1,700	1,800	1,900	1,800	2,100	2,200	2,300
18	1,700	1,800	1,900	2,000	1,900	2,200	2,300	2,400
19	1,800	1,900	2,000	2,100	2,000	2,300	2,400	2,500
20	1,900	2,000	2,100	2,200	2,100	2,400	2,500	2,600

Sheet1

- **[Shift]**キーを押しながらドラッグアンドドロップすると、選択した列または行の位置がドロップ先の列または行と入れ替わりま

Copyright © GrapeCity, Inc. All rights reserved.

す。したがって、選択した列/行のコンテンツとドロップ先の列/行のコンテンツが入れ替わります。



先頭に戻る

フィルタリング

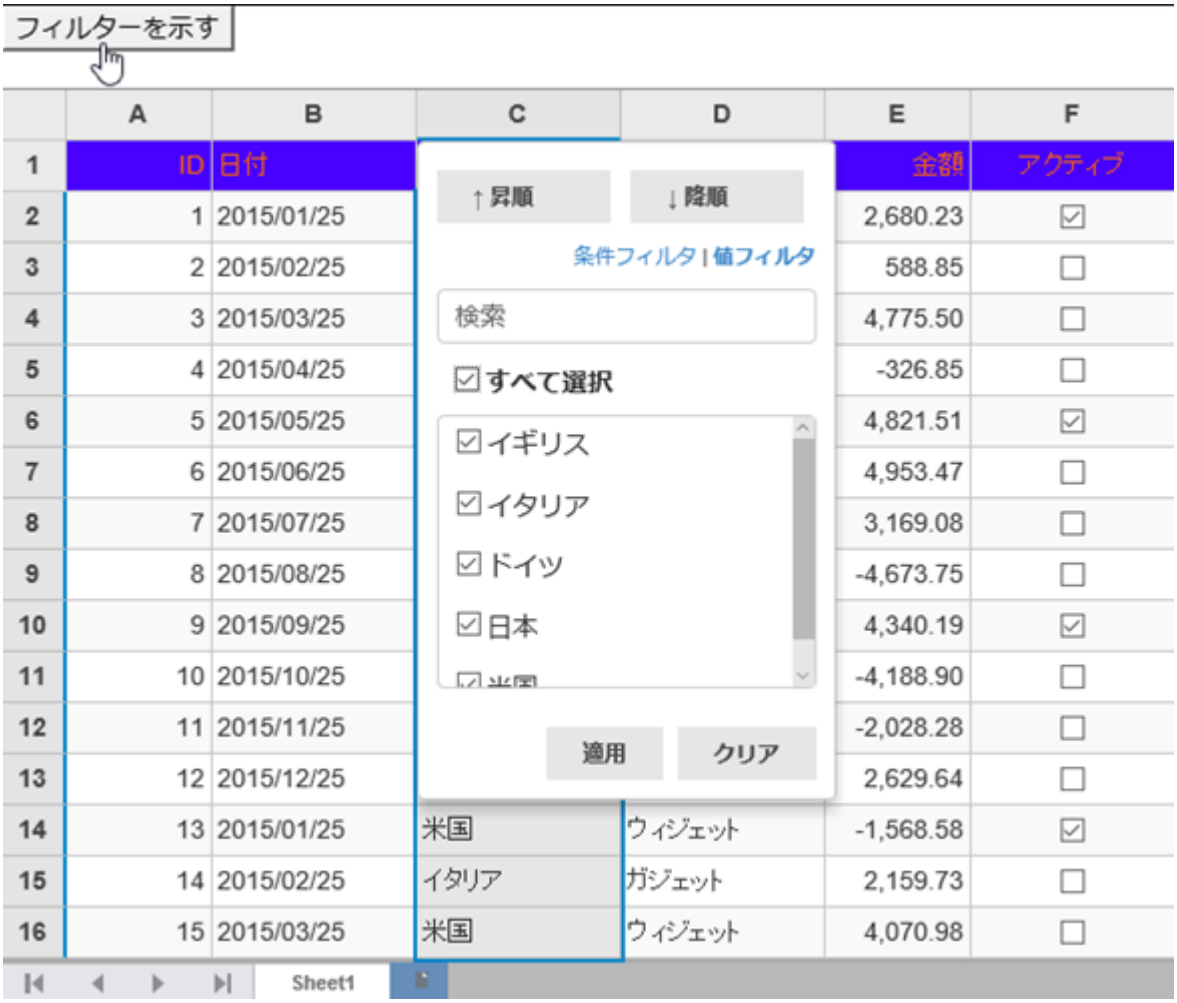
FlexSheetは、クライアント側での列エントリのフィルタ処理をサポートします。これはFlexGridフィルタを拡張し、シートの各列に**条件フィルタ**と**値フィルタ**を適用できるようにします。条件フィルタと値フィルタはフィルタエディタで指定できます。このエディタは、**showColumnFilter()**メソッドを呼び出すことで表示されます。

フィルタエディタでは、**条件によるフィルタ**によって条件を適用して検索を絞り込んだり、**値によるフィルタ**によって目的の列値に対応するデータを正確に見つけることができます。**[昇順]**ボタンと**[降順]**ボタンを使用して、特定の列をそれぞれ昇順および降順にソートできます。**[適用]**ボタンをクリックすると、フィルタされた値がフェッチされ、**[クリア]**ボタンをクリックすると、フィルタエディタが閉じます。

この機能は、ユーザーが特定の列エントリに対応するデータを即座にフェッチするときに役立ちます。これで、シート内の複雑なデータを分析したり表示するタスクが容易になります。たとえば、4か国で活動する会社の月間売上データから、2か国だけの売上詳細を表示するとします。それには、国名を含む列にフィルタを適用します。次は、このようなシナリオの具体例です。

次の図は、選択した列に**値によるフィルタ**を適用した後のデータ連結FlexSheetコントロールを示します。次の例のコントロールは、Sale.csモデルで生成されたデータを表示します。FlexSheetで列を選択してボタンをクリックすることでフィルタエディタを表示し、条件や正確な値を指定して検索を絞り込むことができます。

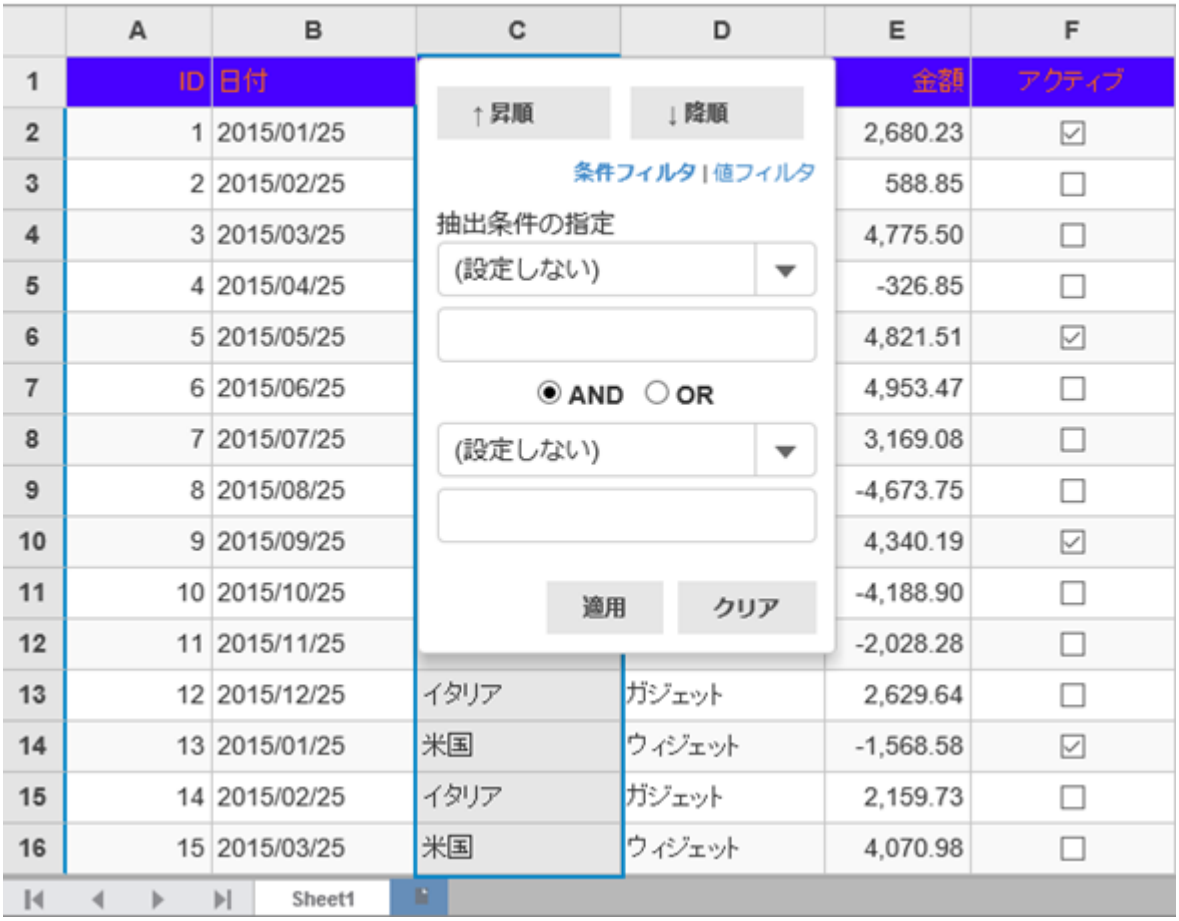
Copyright © GrapeCity, Inc. All rights reserved.



次の図は、条件フィルタ付きの連結FlexSheetです。

Copyright © GrapeCity, Inc. All rights reserved.

フィルターを示す



	A	B	C	D	E	F
1	ID	日付			金額	アクティブ
2	1	2015/01/25			2,680.23	☑
3	2	2015/02/25			588.85	☐
4	3	2015/03/25			4,775.50	☐
5	4	2015/04/25			-326.85	☐
6	5	2015/05/25			4,821.51	☑
7	6	2015/06/25			4,953.47	☐
8	7	2015/07/25			3,169.08	☐
9	8	2015/08/25			-4,673.75	☐
10	9	2015/09/25			4,340.19	☑
11	10	2015/10/25			-4,188.90	☐
12	11	2015/11/25			-2,028.28	☐
13	12	2015/12/25	イタリア	ガジェット	2,629.64	☐
14	13	2015/01/25	米国	ウィジェット	-1,568.58	☑
15	14	2015/02/25	イタリア	ガジェット	2,159.73	☐
16	15	2015/03/25	米国	ウィジェット	4,070.98	☐

次のコード例は、FlexSheetでフィルタ処理を行う方法を示します。

コードの場合

Sale.csクラスを[モデル]フォルダに追加します。

Model

Sale.cs copyCode

```
public  class Sale
  {
      public int ID { get; set; }
      public DateTime 日付 { get; set; }
      public string 国名 { get; set; }
      public string 製品名 { get; set; }
      public double 金額 { get; set; }
      public bool アクティブ { get; set; }

      private static List<string> COUNTRIES = new List<string>
      { "米国", "ドイツ", "イギリス", "日本", "イタリア", "ギリシャ" };
      private static List<string> PRODUCTS = new List<string>
      { "ウィジェット", "ガジェット", "道具" };

      /// <summary>
      /// データを取得します。
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        /// </summary>
        /// <param name="total"></param>
        /// <returns></returns>
        public static IEnumerable<Sale> GetData(int total)
        {
            var rand = new Random(0);
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var date = new DateTime(2015, i % 12 + 1, 25);
                //string.Format("yyyy年-MM月-dd日", date);

                return new Sale
                {
                    ID = i + 1,
                    日付 = date,
                    国名 = country,
                    製品名 = product,
                    金額 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    アクティブ = (i % 4 == 0)
                };
            });
            return list;
        }

        public static List<string> GetCountries()
        {
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }

        public static List<string> GetProducts()
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
    }
```

FilteringController.cs

C# copyCode

```
public class FilterController : Controller
{
    // GET: Filter
    public static List<Sale> SALES = Sale.GetData(15).ToList();
    public ActionResult FilterIndex()
    {
        return View(SALES);
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

Filtering.cshtml

Razor copyCode

```
@using MVCFlexSheet_JP.Models;
@model IEnumerable<Sale>

<script>
    function filterFlex() {
        var flex = wijmo.Control.getControl("#fSheet");
        flex.showColumnFilter();
    }
</script>

<div>
    <input id="filter" type="button" onclick="filterFlex()"
value="フィルターを示す" />
    <br /><br />
    @(Html.C1().FlexSheet().CssClass("flexSheet").Id("fSheet")
.IsReadOnly(false).Width("680px").Height("500px")
        .AddBoundSheet(sheet =>
            sheet.Bind(cv =>
                            cv.Bind(Model)))
    )
</div>
```

先頭に戻る

セルの書式

FlexSheetでは、クライアント側でセルやセル内のコンテンツを書式設定できます。簡単に各セルに塗りつぶし色を適用したり、フォントのサイズ、スタイル、色、ウェイト（太字/非太字）を変更できます。また、実行時にセルのテキストに下線を付けたり、斜体にしたり、整列することができます。さらに、各セルのデータに表示書式を設定できます（小数、整数、パーセンテージ、通貨などの書式）。

FlexSheetコントロールをカスタマイズして、表示データの外観を変更し、大きなワークシート内のデータを目立たせることができます。たとえば、会社の四半期ベースの地域別売上の数値を分類して比較するとします。それには、**applyCellsStyle()**メソッドを使用します。次の例に示すFlexSheetコントロールは、このようなシナリオとして、2015～16年度のさまざまな国における3つの製品の四半期売上データを表示します。

次の図は、セルとセル内のデータの書式を変更するボタンを含むFlexSheetです。



次の図は、セルに書式を適用した後のFlexSheetです。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexSheetでスタイルを適用する方法を示します。

コードの場合

FormatController.cs

C# copyCode

```
public class FormatController : Controller
{
    // GET: Format
    public ActionResult FormatIndex()
    {
        return View();
    }
}
```

FormatCells.cshtml

ASP.NET

Razor copyCode

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js"></script>
<script type="text/javascript">
    function setBold() {
        var flex = wijmo.Control.getControl("#fSheet");
        flex.applyCellsStyle({ fontWeight: 'bold' });
    }
    function changeFont() {
        var flex = wijmo.Control.getControl("#fSheet");
        flex.applyCellsStyle({ fontFamily: 'Unicode' });
    }
    function changeFontSize() {
        var flex = wijmo.Control.getControl("#fSheet");
        flex.applyCellsStyle({ fontSize: '18px' });
    }
    function setColor() {
        var flex = wijmo.Control.getControl("#fSheet");
        flex.applyCellsStyle({ backgroundColor: '#CCCCFF' });
    }

    function align() {
        var flex = wijmo.Control.getControl("#fSheet");
        flex.applyCellsStyle({ textAlign: 'Right' });
    }

</script>
<div>
    <input id="font" type="button" onclick="changeFont()" value="Change Font" />
    <input id="bold" type="button" onclick="setBold()" value="Bold" />
    <input id="cellColor" type="button" onclick="setColor()" value="Color" />
    <input id="fontSize" type="button" onclick="changeFontSize()" value="Font Size" />
    <input id="textAlign" type="button" onclick="align()" value="Align to Center" />
    <br /><br />
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().FlexSheet().Height("500px").Width("1500px").Load("~/Content/FlexSheet/QuarterlyData.xlsx").Id("fSheet")
    )
</div>
```

FlexSheetのセルの書式設定については、次のページにある詳細なデモサンプルを参照してください。

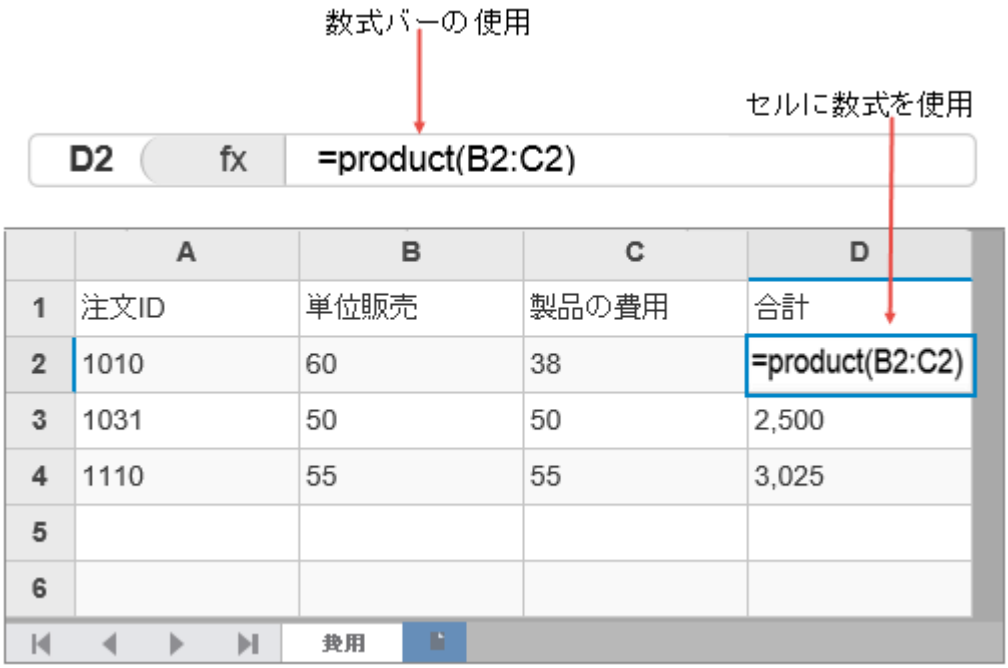
- FlexSheetセルの書式設定の詳細

先頭に戻る

FlexSheetの数式

FlexSheetには、約100の数式のサポートと自動補完が組み込まれています。これらの数式を使用して、データの操作やシートデータに対する計算を実行できます。FlexSheetは、合計を求める簡単な数学演算の実行から、複雑なエンジニアリング問題の解決まで、柔軟な機能を提供します。

FlexSheetの上部にある数式バーを使用すると、セルに式を入力したり、既存の式をコピーすることができます。数式バーには、関数のシンボル(fx)が付いています。数式バーをクリックするか、セルに等号(=)をキー入力すると、数式バーがアクティブになります。Excelと同様に、数式をセル内で使用できます。FlexSheetの数式演算は、基本演算から数学、論理、テキスト、集計、日付、ルックアップと参照、株価まで広範囲です。



以下のセクションでは、実行時およびプログラムからFlexSheetで組み込みの数式を使用する方法を示します。

FlexSheetでサポートされた関数

FlexSheetコントロールは、次の組み込みの数式をサポートします。

名前	説明
abs	数値の絶対値を返します。
acos	数値の逆余弦を返します。
and	引数のすべてがTRUEの場合に、TRUEを返します。
asin	数値の逆正弦を返します。

Copyright © GrapeCity, Inc. All rights reserved.

名前	説明
atan	数値の逆正接を返します。
atan2	XおよびY座標から逆正接を返します。
average	引数の平均値を返します。
ceiling	数値を最も近い整数または最も近い基準値の倍数に切り上げます。
char	コード番号で指定された文字を返します。
choose	値のリストから値を選択します。
code	テキスト文字列の最初の文字の数値コードを返します。
column	参照の列番号を返します。
columns	参照内の列の数を返します。
concatenate	複数のテキスト項目を1つのテキスト項目に結合します。
cos	数値の余弦を返します。
count	引数リストに含まれる数値の個数を返します。
counta	引数リストに含まれる値の個数を返します。
countblank	範囲内に含まれる空白セルの個数を返します。
countif	範囲内に含まれる特定の基準を満たすセルの個数を返します。
countifs	範囲内に含まれる複数の条件を満たすセルの個数を返します。
date	特定の日付のシリアル値を返します。
datediff	2つの日付の間の日数、月数、または年数を計算します。
day	シリアル値を月の日付に変換します。
dcount	データベース内の数値を含むセルをカウントします。
exp	eを底とする数値のべき乗を返します。
false	論理値FALSEを返します。
find	テキストから別のテキスト値を検索します(大文字と小文字は区別されます)。
floor	数値を0に近い方に切り捨てます。
hlookup	配列の先頭の行を検索し、示されたセルの値を返します。
hour	シリアル値を時間に変換します。
if	論理テストの実行を指定します。
index	インデックスを使用して、参照から値を選択します。
left	テキスト値の先頭から指定された数の文字を返します。
len	テキスト文字列の長さ(文字数)を返します。
ln	数値の自然対数を返します。
lower	テキストを小文字に変換します。
max	引数リスト内の最大値を返します。
mid	テキスト文字列の指定された位置から、指定された数の文字を返します。

Copyright © GrapeCity, Inc. All rights reserved.

名前	説明
min	引数リスト内の最小値を返します。
mod	剰余を返します。
month	シリアル値を月に変換します。
not	引数のロジックを反転させます。
now	現在の日時のシリアル値を返します。
or	いずれかの引数がTRUEの場合に、TRUEを返します。
pi	πの値を返します。
power	数値のべき乗を返します。
product	引数を乗算します。
proper	テキスト値の各単語の先頭の文字を大文字に変換します。
rand	0～1の間の乱数を返します。
rank	数値リスト内の数値のランクを返します。
rate	期間の年利を返します。
replace	テキスト内の文字を置換します。
rept	テキストを指定された回数だけ繰り返します。
right	テキスト値の末尾から指定された数の文字を返します。
round	数値を指定された桁数に丸めます。
rounddown	数値を0に近い方に切り捨てます。
roundup	数値を0から遠い方に切り上げます。
row	参照の行番号を返します。
rows	参照内の行の数を返します。
search	テキストから別のテキスト値を検索します（大文字と小文字は区別されません）。
sin	指定された角度の正弦を返します。
sqrt	正の平方根を返します。
stdev	標本標準偏差を返します。
stdevp	母集団標準偏差を返します。
substitute	テキスト文字列内の指定されたテキストを新しいテキストに置換します。
subtotal	リストまたはデータベースの小計を返します。
sum	引数を合計します。
sumif	指定された条件に合致するセルを加算します。
sumifs	範囲内に含まれる複数の条件を満たすセルを加算します。
tan	数値の正接を返します。
text	数値を書式設定し、テキストに変換します。
time	特定の時刻のシリアル値を返します。

Copyright © GrapeCity, Inc. All rights reserved.

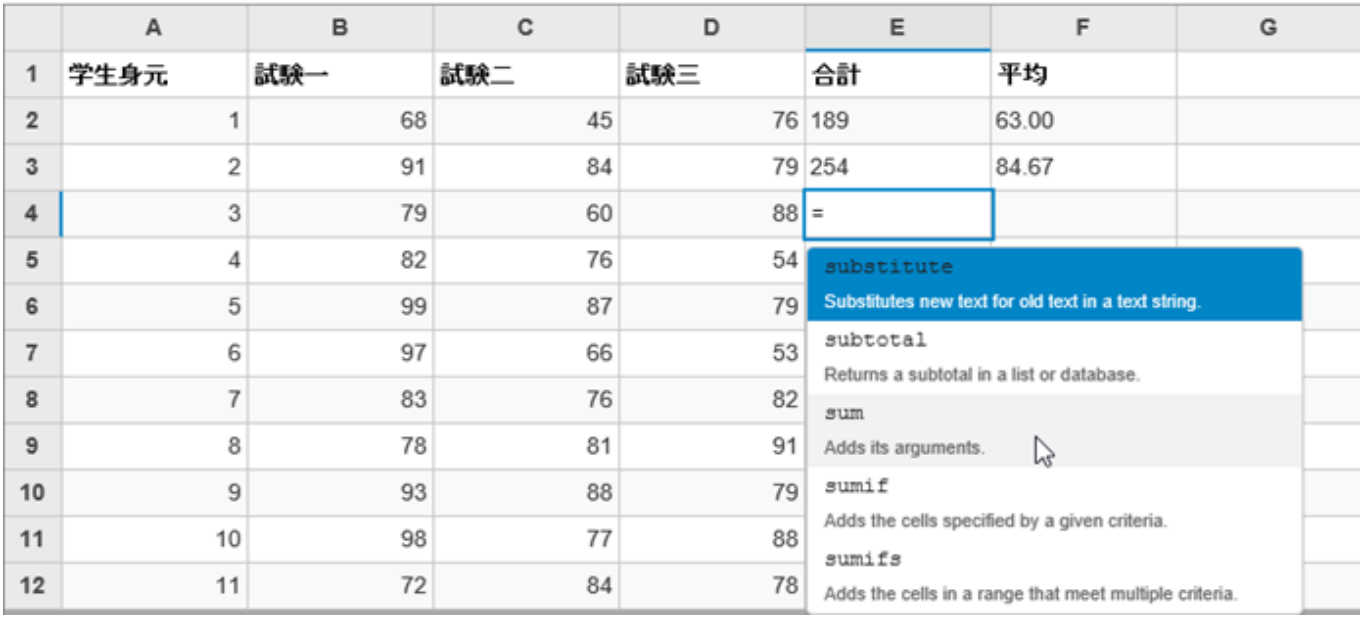
名前	説明
today	今日の日付のシリアル値を返します。
trim	テキストからスペースを削除します。
true	論理値TRUEを返します。
trunc	数値を整数に切り捨てます。
upper	テキストを大文字に変換します。
value	テキスト引数を数値に変換します。
var	標本分散を返します。
varp	母集団分散を返します。
year	シリアル値を年に変換します。

実行時に数式の使用

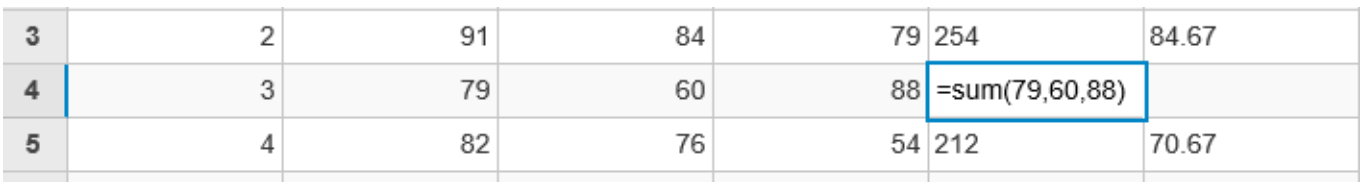
FlexSheetコントロールは、Excelと同様に、単に入力データや連結データを表示する以上の機能を持ちます。組み込み関数の大きな潜在能力を駆使し、データに対して複雑な計算を実行することで、FlexSheetを包括的なレポート分析ツールとして利用できます。

このトピックでは、組み込みの数式を使用してFlexSheetコントロールでデータを操作する方法を示す例を提供します。FlexSheetの定義済みの数式は、基本的なコンテキストメニュー操作を実行するように簡単に使用できます。セルを選択し、キーボードから等号(=)を入力し、表示されるポップアップで定義済み数式のリストから目的の数式を選択するだけです。

たとえば、次の図に示すFlexSheetコントロールは、10人の工学科の学生の3回の期末テストの点数(100点満点)を表示しています。次の例は、数式を使用して学生たちの点数の合計と平均を計算する方法を示します。

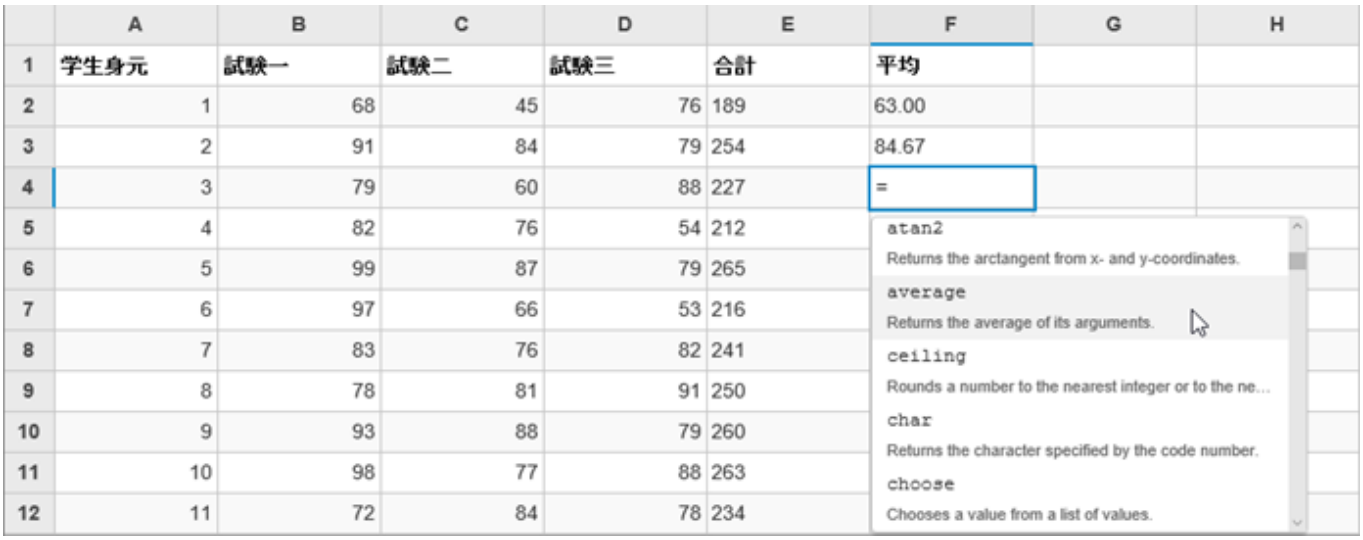


次の図は、数式を使用し、点数の合計を計算してセルE4に表示する方法を示します。



Copyright © GrapeCity, Inc. All rights reserved.

同様に、以下では、ID 003の学生の点数の平均をセルF4に表示します。



	A	B	C	D	E	F
1	学生身元	試験一	試験二	試験三	合計	平均
2	1	68	45	76	189	63.00
3	2	91	84	79	254	84.67
4	3	79	60	88	227	=average(79,60,88)

次の図では、セルC9の値が**Ceiling**式によって最も近い整数に丸められています。

9	8	78	=ceiling(80.5)	91

セル内で定義済みの数式を使用する詳細なデモサンプルについては、次を参照してください。

- FlexSheetでの数式の操作

コードで数式の使用

実行時に数式を使用するほかに、FlexSheetでは、プログラムからセルに数式を追加できます。このトピックでは、クライアント側のコードから、FlexSheetでサポートされる定義済み数式を使用する方法を示します。

次の図は、製品の「Unit Sold（販売個数）」と「Product Cost（製品原価）」を表示するFlexSheetコントロールを示します。この例では、注文ごとの合計額を計算する式がプログラムによって適用されています。これは、クライアント側で**setCellData()**メソッドを使用して行われます。

Copyright © GrapeCity, Inc. All rights reserved.

費用を表示 数式を適用

D2 fx =product(B2:C2)

	A	B	C	D
1	注文ID	販売個数	製品原価	合計
2	1010	60	38	2,280
3	1031	50	50	2,500
4	1110	55	55	3,025
5				
6				

費用

次のコード例は、FlexSheetでプログラムから式を使用する方法を示します。

コードの場合

FormulaController.cs

C# copyCode

```
public class FormulaController : Controller
{
    // GET: Formula
    public ActionResult FormlaCodeIndex()
    {
        return View();
    }
}
```

FormulaIndex.cshtml

Razor copyCode

```
<script>

    function generateCostSheet() {
        var flex = wijmo.Control.getControl("#formulaSheet");
        flex.setCellData(0, 0, "注文ID");
        flex.setCellData(0, 1, "販売個数");)
        flex.setCellData(1, 0, "1010");
        flex.setCellData(2, 0, "1031");
        flex.setCellData(3, 0, "1110");
        flex.setCellData(1, 1, "60");
        flex.setCellData(2, 1, "50");
        flex.setCellData(3, 1, "55");
        flex.setCellData(0, 2, "製品原価");
        flex.setCellData(1, 2, "38");
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        flex.setCellData(2, 2, "50");
        flex.setCellData(3, 2, "55");
        flex.setCellData(0, 3, "合計");
    }
    function setFormula() {
        var flex = wijmo.Control.getControl("#formulaSheet");
        flex.setCellData(1, 3, "=product(B2:C2)");
        flex.setCellData(2, 3, "=product(B3:C3)");
        flex.setCellData(3, 3, "=product(B4:C4)");
    }
<div>
    <input id="bold" type="button" onclick="generateCostSheet()"
value="費用を表示" />
    <input id="textAlign" type="button" onclick="setFormula()"
value="数式を適用" />
    <br /><br />
    @(Html.C1().FlexSheet().CssClass("flexSheet").Id("formulaSheet")
.Height("200px").Width("500px")
            .AddUnboundSheet("費用", 6, 4).IsReadOnly(false))
</div>
```

先頭に戻る

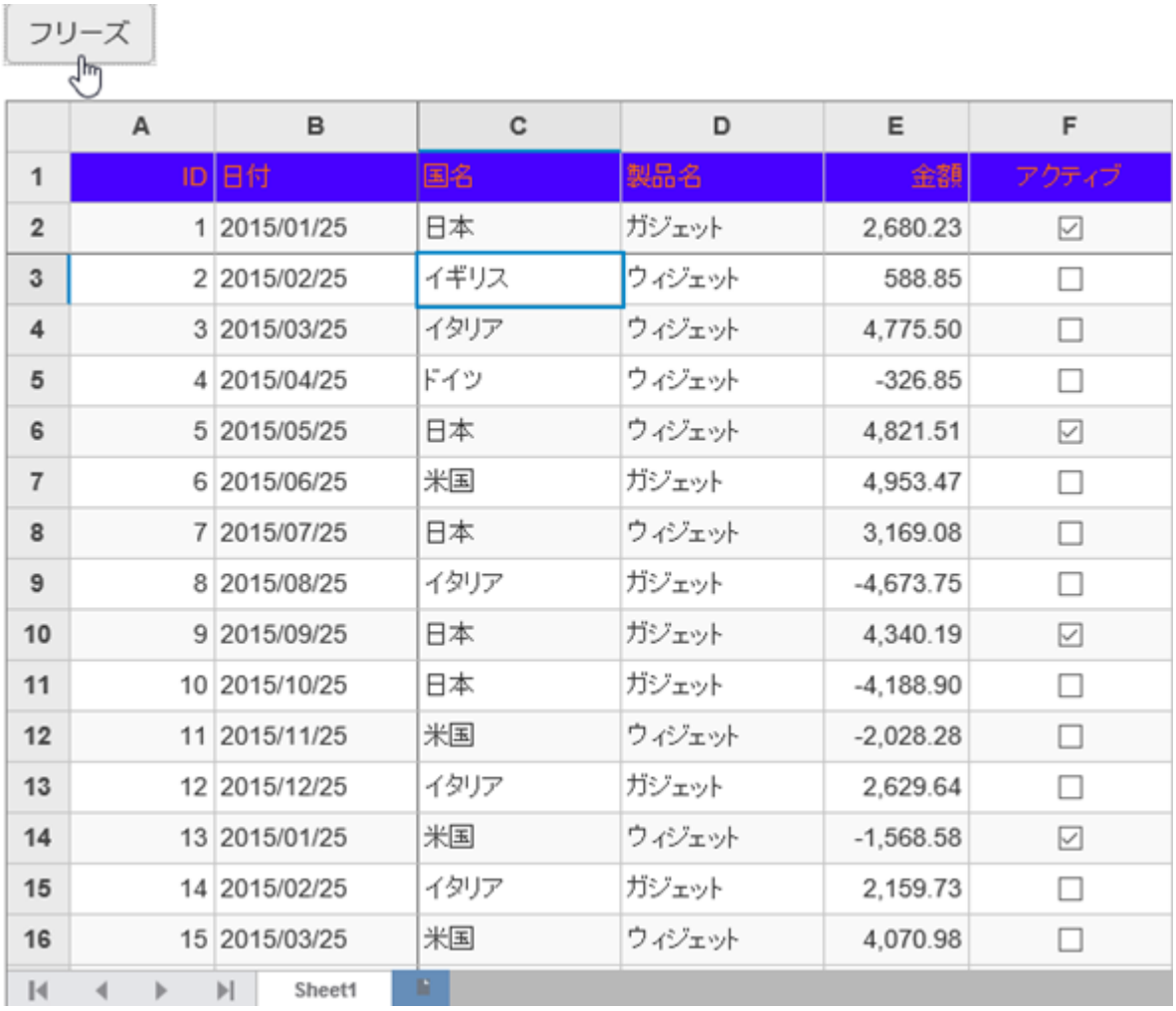
固定のセル

FlexSheetは、選択したセルの行と列をロックすることで、クライアント側でペインの固定をサポートします。このため、ワークシートの特定の領域を表示したまま、別の領域にスクロールできます。それには、**freezeAtCursor()**メソッドを使用します。

これは、FlexSheetに大量のデータを表示しているときにたいへん便利です。たとえば、先頭のヘッダー表示したまま、ワークシートを大きく下までスクロールしたい場合があります。あるセルを含む行と列をロック（固定）するには、そのセルを選択し、**freezeAtCursor()**メソッドを呼び出すだけです。ただし、FlexSheetコントロールに固定セルが既に含まれる場合は、**freezeAtCursor()**によってその固定が解除されます。

次の図は、売上データを表示するFlexSheetコントロールと[Freeze]ボタンです。次の例では、ボタンクリックによって**freezeAtCursor()**メソッドが呼び出されます。セルを選択した後に[Freeze]ボタンをクリックすると、freezeAtCursorメソッドが呼び出され、対応する行と列にあるセルがロックされます。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexSheetでセルを固定する方法を示します。

コードの場合

Sale.csクラスを[モデル]フォルダに追加します。

Model

Sale.cs copyCode

```
public  class Sale
  {
      public int ID { get; set; }
      public DateTime 日付 { get; set; }
      public string 国名 { get; set; }
      public string 製品名 { get; set; }
      public double 金額 { get; set; }
      public bool アクティブ { get; set; }

      private static List<string> COUNTRIES = new List<string>
      { "米国", "ドイツ", "イギリス", "日本", "イタリア", "ギリシャ" };
      private static List<string> PRODUCTS = new List<string>
      { "ウィジェット", "ガジェット", "道具" };

      /// <summary>
      /// データを取得します。
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        /// </summary>
        /// <param name="total"></param>
        /// <returns></returns>
        public static IEnumerable<Sale> GetData(int total)
        {
            var rand = new Random(0);
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var date = new DateTime(2015, i % 12 + 1, 25);
                //string.Format("yyyy年-MM月-dd日", date);

                return new Sale
                {
                    ID = i + 1,
                    日付 = date,
                    国名 = country,
                    製品名 = product,
                    金額 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    アクティブ = (i % 4 == 0)
                };
            });
            return list;
        }

        public static List<string> GetCountries()
        {
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }

        public static List<string> GetProducts()
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
    }
```

FrozenController.cs

C# copyCode

```
// GET: FrozenCells
public ActionResult Index()
{
    return View();
}
```


Copyright © GrapeCity, Inc. All rights reserved.

FrozenCells.cshtml

Razor copyCode

```
@using MVCFlexSheet_JP.Models
@model IEnumerable<Sale>

<script type="text/javascript">


    function freezeCells() {
        var flex = wijmo.Control.getControl("#freezeSheet");
        flex.freezeAtCursor();
    }
</script>

<div>
    <button type="button" class="btn btn-default"
onclick="freezeCells()" id="frozenBtn">フリーズ</button>
    <br /><br />
    @(Html.C1().FlexSheet().CssClass("flexSheet")
.Id("freezeSheet").Height("500px").Width("720px")
                .AddBoundSheet(cv => cv.Bind(Model)))

</div>
```

先頭に戻る

クライアント側でのJSONのロードと保存

FlexSheetは、クライアント側でのJSONのロードと保存をサポートします。FlexSheetコントロールへのJSON文字列のデータのロードは、簡単に行うことができます。また、FlexSheetデータをJSON文字列に保存することもできます。このコントロールは、そのためにWorkbookオブジェクトモデルを使用します。

次のコード例は、Workbookオブジェクトモデルを使用して、JSON文字列をFlexSheetコントロールにロードする方法を示します。

コードの場合

JsonLoadController.cs

C# copyCode

```
public class JsonLoadController : Controller
{
    public static List<Sale> SALES = Sale.GetData(15).ToList();
    // GET: Json
    public ActionResult JsonIndex()
    {
        return View(SALES);
    }
}
```

JsonLoading.cshtml

Copyright © GrapeCity, Inc. All rights reserved.

Razor copyCode

```
<script>
    var flex, jsonString;
    c1.mvc.Utils.documentReady(function () {
        flex = wijmo.Control.getControl('#jsonLoadSheet');
    });

    function SaveToJson() {
        var workBook = flex.saveToWorkbookOM();
        jsonString = JSON.stringify(workBook);

    }

    function loadJSON() {
        var workBook = JSON.parse(jsonString);
        flex.loadFromWorkbookOM(workBook);
    }
</script>

<div>
    <button class="btn btn-default" onclick="SaveToJson()">Save to Json</button>
    <button class="btn btn-default" onclick="loadJSON()">Load Json</button>
    <br /><br />

@(Html.C1().FlexSheet().Id("jsonLoadSheet").CssClass("flexSheet").Width(700).Height(700)
        .AddBoundSheet(sheet =>
            sheet.Bind(cv =>
                            cv.Bind(Model).DisableServerRead(true)))
    )
</div>
```

先頭に戻る

複数のヘッダー

FlexSheetコントロールをカスタマイズして、クライアント側で行/列ヘッダーの追加と削除を行うことができます。このため、ワークシートに複数の行/列ヘッダーを追加したり、既存の行/列ヘッダーを削除することもできます。

FlexSheetを含む基本的なMVCアプリケーションを作成して実行すると、デフォルトでは、FlexSheetコントロールに行ヘッダーと列ヘッダーが1つずつ置かれます。列ヘッダーには各列を表す英大文字（"A"、"B"、"C"など）が表示され、行ヘッダーには各行を表す数字（"1"、"2"、"3"など）が表示されます。ただし、ヘッダーを追加したり、既存のヘッダーを削除して、このデフォルトの動作を変更できます。それには、**rowHeaders.columns**コレクションと**columnHeaders.rows**コレクションに対して**push()**関数を呼び出して、それぞれ行ヘッダーと列ヘッダーを追加します。また、行ヘッダーや列ヘッダーを削除するには、**rowHeaders.columns**コレクションと**columnHeaders.rows**コレクションに対してそれぞれ**removeAt()**関数を呼び出します。

次の図は、行/列ヘッダーを追加および削除するボタンが付いた非連結FlexSheetコントロールです。次の例では、ボタンクリックによってremoveAt()とpush()が呼び出されます。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、FlexSheetで行/列ヘッダーを追加および削除する方法を示します。

コードの場合

MultipleHeaderController.cs

C# copyCode

```
public class MultipleHeaderController : Controller
{
    // GET: MultipleHeader
    public ActionResult MultipleHeaderIndex()
    {
        return View();
    }
}
```

MultipleHeader.cshtml

Razor copyCode

```
<script>
    function addRowHeader() {
     var flex = wijmo.Control.getControl('#mHeadersSheet');
     flex.rowHeaders.columns.push(new wijmo.grid.Column());
    }
    function removeRowHeader() {
     var colCnt, flex = wijmo.Control.getControl('#mHeadersSheet');
     flex.rowHeaders.columns.removeAt(colCnt - 1);
    }
    function addColumnHeader() {
     var rowCnt, flex = wijmo.Control.getControl('#mHeadersSheet');
     flex.columnHeaders.rows.push(new wijmo.grid.Row());
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    function removeColumnHeader() {
     var flex = wijmo.Control.getControl('#mHeadersSheet');
     flex.columnHeaders.rows.removeAt(rowCnt - 1);
    }
</script>
<div>
    <button type="button" class="btn btn-default"
onclick="addRowHeader()">行ヘッダーを追加</button>
    <button type="button" class="btn btn-default"
onclick="removeRowHeader()">行ヘッダーを削除</button>
    <button type="button" class="btn btn-default"
onclick="addColumnHeader()">列ヘッダーを追加</button>
    <button type="button" class="btn btn-default"
onclick="removeColumnHeader()">列ヘッダーを削除</button>
    <br /><br />
    @(Html.C1().FlexSheet().CssClass("flexSheet")
.Id("mHeadersSheet").IsReadOnly(false).Width("900px")
.Height("400px")
        .AddUnboundSheet("Sheet1", 20, 8))
</div>
```

先頭に戻る

リモートでロードおよび保存

FlexSheetでは、C1JSONRequestを使用してExcelファイルをリモートでロードおよび保存できます。それには、リモートURLを指定して、Excelファイルまたはワークブックから FlexSheetコントロールにデータをロードし、FlexSheetデータをExcelファイルまたはワークブックとしてサーバーに保存します。

次の例は、FlexSheetHelperクラスの**Load**メソッドを使用して、サーバーにあるExcelファイルをFlexSheetにロードします。

> クライアント側でExcelをロードおよび保存するには、FlexSheetに**jszip.min.js**ファイルが必要です。そのため、このファイルをアプリケーションに追加し、各ビューまたは**_Layout.cshtml**ファイルの<head>セクションにファイルへの参照を提供する必要があります。

次のコード例は、Excelファイルまたはワークブックから FlexSheetコントロールにデータをリモートにロードする方法を示します。

コードの場合

RemoteLoadController.cs

C# copyCode

```
public class RemoteController : Controller
{
    // GET: /<controller>/
    public ActionResult Index()
    {
        return View();
    }
    public ActionResult RemoteLoadXlsx()
    {
        return
this.C1Json(FlexSheetHelper.Load("~/Content/ExcelFiles/WorkBook.xlsx"), null, null,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
JsonRequestBehavior.AllowGet);
    }
}
```

ここでは、リモートでロードされるExcelワークブックは、アプリケーションの「Content」フォルダに配置されます。

RemoteLoading.cshtml

Razor copyCode

```
@using C1.Web.Mvc.Sheet;
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>

<div>
@(Html.C1().FlexSheet().CssClass("flexSheet").Height(700).Width(700)
        .RemoteLoad(Url.Action("RemoteLoadXlsx"))
        )
</div>
```

先頭に戻る

コードの場合

RemoteSaveController.cs

C# copyCode

```
public class RemoteSaveController : Controller
{
    // GET: /<controller>/
    public static List<Sale> SALES = Sale.GetData(50).ToList();
    public ActionResult Index()
    {
        return View(SALES);
    }

    private const string FILE_PATH = "\\wwwroot\\uploadFile\\save.xlsx";
    public JsonResult RemoteSaveFile([FlexSheetRequest]FlexSheetSaveRequest request)
    {
        var success = true;
        var error = "";
        var app = GetService<IApplicationEnvironment>();
        var savePath = app.ApplicationBasePath + FILE_PATH;
        try
        {
            Stream st = request.GetFileStream();
            using (FileStream fs = new FileStream(savePath, FileMode.Create))
            {
                if (st != null)
                {
                    st.CopyTo(fs);
                }
            }
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        catch (Exception e)
        {
            success = false;
            error = e.ToString();
        }

        return this.C1Json(FlexSheetHelper.Save(success, error));
    }

    public FileResult DownloadFile()
    {
        var app = GetService<IApplicationEnvironment>();
        var savePath = app.ApplicationBasePath + FILE_PATH;
        var name = Path.GetFileName(FILE_PATH);
        return File(new FileStream(savePath, FileMode.Open, FileAccess.Read),
            "application/msexcel", name);
    }

    public static T GetService<T>() where T : class
    {
        var serviceProvider = CallContextServiceLocator.Locator.ServiceProvider;
        return serviceProvider.GetService(typeof(T)) as T;
    }
}
```

RemoteSaving.cshtml

Razor copyCode

```
@using C1MvcFSheetNew.Models;
@model IEnumerable<sale>

<script>
        function remoteSave() {
            var flexSheet = wijmo.Control.getControl('#flexSheet');
            flexSheet.remoteSave(c1.mvc.grid.sheet.ContentType.Xlsx);
        }

        function onFileSaved(sender, args) {
            if (args.success) {
             window.location.href = '@Url.Action("DownloadFile")';
            } else {
                alert(args.error);
            }
        }
</script>

<div>
    @(Html.C1().FlexSheet().CssClass("flexSheet").Width("500px")
.Height("700px").RemoteSave(Url.Action("RemoteSaveFile"))
        .OnClientRemoteSaved("onFileSaved")
        .AddBoundSheet(sheet =>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            sheet.Bind(cv =>
                                cv.Bind(Model)))
    )
</div>
```

先頭に戻る

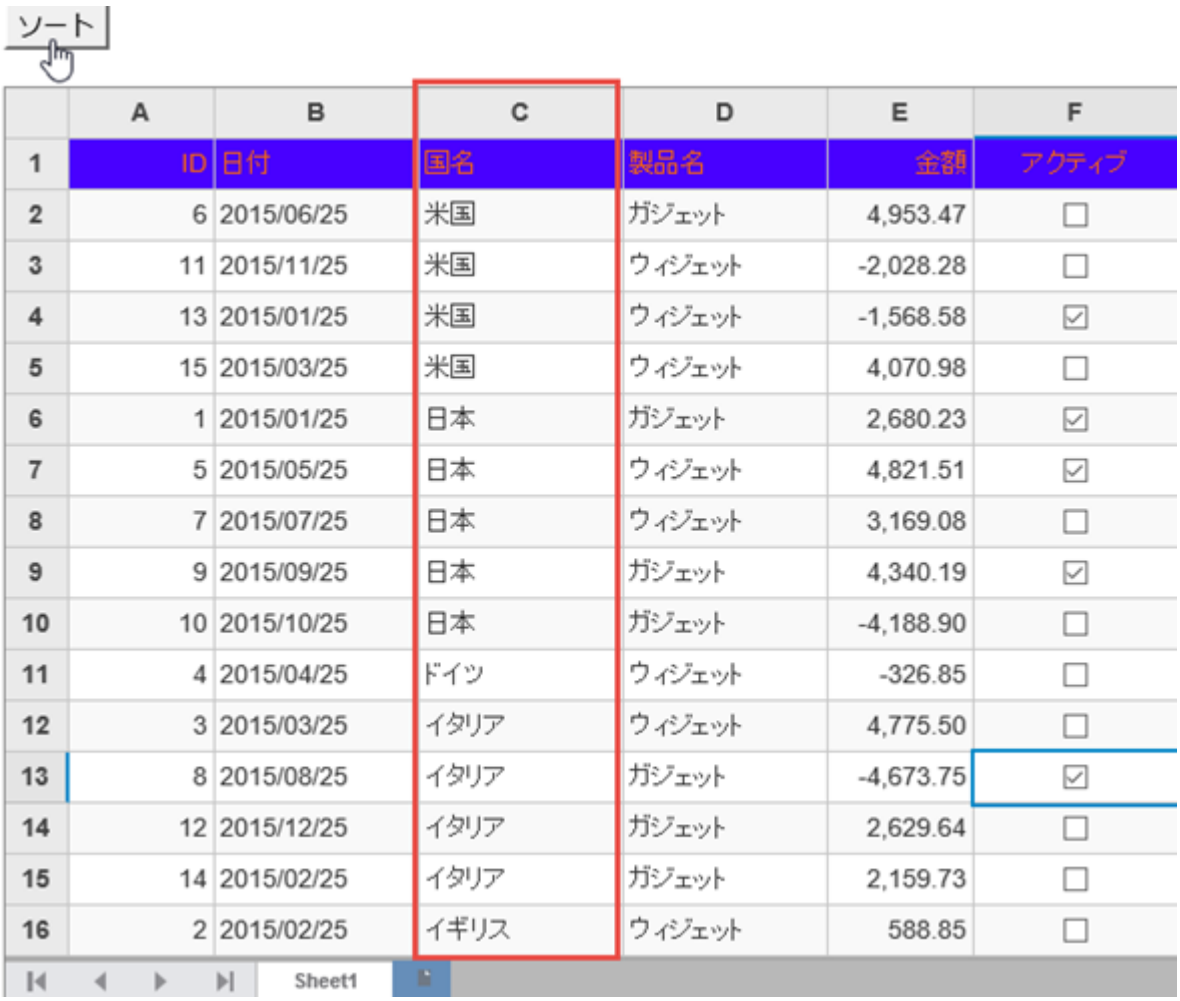
ソート

FlexSheetは、クライアント側のソートをサポートします。FlexSheetデータを任意の列に基づいてソートし、効果的なレポート、参照、データ分析を実現できます。

たとえば、FlexSheetで会社の売上レポートを分析しながら、製品名のアルファベット順や日付の昇順でレコードを表示できます。それには、FlexSheetで日付エントリや製品エントリを含む列をソートします。次は、このようなシナリオの具体例です。

SortManagerプロパティを使用すると、FlexSheetでソート処理を管理できます。クライアント側で、ソートの順序やソートする列を設定したり、ソート順を変更することもできます。さらに、レベルを追加して複数の列をソートすることもできます。

次の図は、2つの製品の日付ごとの売上データを表示する連結FlexSheetコントロールです。次の例では、ボタンをクリックすると、国名を含む3番目の列エントリ（列インデックス2）の降順でFlexSheetデータがソートされます。



次のコード例は、FlexSheetでソートを行う方法を示します。

コードの場合

Sale.csクラスを[モデル]フォルダに追加します。

Copyright © GrapeCity, Inc. All rights reserved.

Model

Sale.cs copyCode

```
public  class Sale
  {
      public int ID { get; set; }
      public DateTime 日付 { get; set; }
      public string 国名 { get; set; }
      public string 製品名 { get; set; }
      public double 金額 { get; set; }
      public bool アクティブ { get; set; }

      private static List<string> COUNTRIES = new List<string>
      { "米国", "ドイツ", "イギリス", "日本", "イタリア", "ギリシャ" };
      private static List<string> PRODUCTS = new List<string>
      { "ウィジェット", "ガジェット", "道具" };

      /// <summary>
      /// データを取得します。
      /// </summary>
      /// <param name="total"></param>
      /// <returns></returns>
      public static IEnumerable<Sale> GetData(int total)
      {
          var rand = new Random(0);
          var list = Enumerable.Range(0, total).Select(i =>
          {
              var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
              var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
              var date = new DateTime(2015, i % 12 + 1, 25);
              //string.Format("yyyy年-MM月-dd日", date);

              return new Sale
              {
                  ID = i + 1,
                  日付 = date,
                  国名 = country,
                  製品名 = product,
                  金額 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                  アクティブ = (i % 4 == 0)
              };
          });
          return list;
      }

      public static List<string> GetCountries()
      {
          var countries = new List<string>();
          countries.AddRange(COUNTRIES);
          return countries;
      }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public static List<string> GetProducts()
    {
        List<string> products = new List<string>();
        products.AddRange(PRODUCTS);
        return products;
    }
}
```

SortingController.cs

C# copyCode

```
public partial class SortController : Controller
{
    // GET: Sort
    public static List<Sale> SALES = Sale.GetData(15).ToList();

    public ActionResult SortIndex()
    {
        return View(SALES);
    }
}
```

Sorting.cshtml

Razor copyCode

```
@using MVCFlexSheet_JP.Models;
@model IEnumerable<Sale>

<script>
    var flex, sortManager;
    c1.documentReady(function () {
        flex = wijmo.Control.getControl("#sortsheet");
        sortManager = flex.sortManager;
    });
    function sortFlex() {
        sortManager.sortDescriptions.items[0].columnIndex = 2;
        sortManager.sortDescriptions.items[0].ascending = false;
        sortManager.commitSort();
    }
</script>

<div>
    <input id="sort" type="button" onclick="sortFlex()" value="ソート" />
    <br /><br />
    @(Html.C1().FlexSheet().CssClass("flexSheet").Id("sortsheet")
   .Height("500px").Width("680px")
        .AddBoundSheet(cv => cv.Bind(Model))
    )
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

先頭に戻る

スタイル設定とCSS

CSS classes for the FlexSheet control are as follows:

Class	Description
.wj-flexsheet	Styles the div host element of the FlexSheet control.
.wj-flexsheet .wj-content	Implementation not found.
.wj-flexsheet .wj-header-row	Implementation not found.
.wj-flexsheet .wj-tabholder	Styles the tab holder area that displays the FlexSheet tabs.
Selection	
.wj-state-multi-selected	Styles all the cells in the selected cell range except the active cell.
.wj-state-selected	Styles the active cell and currently selected item in the drop-down list.
Formula	
.wj-flexsheet-formula-list	Styles the list of formulas displayed in the FlexSheet control.
.wj-flexsheet-formula-list .wj-flexsheet-formula-name	Styles the formula names displayed in the formula list.
.wj-flexsheet-formula-list .wj-flexsheet-formula-description	Styles the formula description displayed under each formula in the list.
.wj-flexsheet-formula-list .wj-listbox-item.wj-state-selected > *	Styles the name of the formula when user hovers on each formula name in the list.
Tabholder	
.wj-flexsheet .wj-tabholder span.wj-sheet-icon	Styles the sheet icon displayed in the tabholder.
.wj-flexsheet .wj-tabholder div.wj-sheet-page.wj-btn-group	Styles the group of buttons displayed in the tabholder.
.wj-flexsheet .wj-tabholder div.wj-sheet-page.wj-btn-group > button	Styles the buttons displayed in the tabholder.
.wj-flexsheet .wj-tabholder .wj-sheet-tab ul	Styles the element of the sheet tab that displays the all sheets.
.wj-flexsheet .wj-tabholder .wj-sheet-tab ul li	Styles the element of the sheet tab that displays the individual sheet tab and name.
.wj-flexsheet .wj-tabholder .wj-sheet-tab ul li.active	Styles the active sheet tab.
.wj-flexsheet .wj-tabholder .wj-new-sheet	Styles the new sheet button that is displayed in the tabholder.
.wj-flexsheet .wj-tabholder .wj-new-sheet .wj-sheet-icon	Styles the new sheet icon that is displayed in the tabholder.
.wj-flexsheet .wj-tabholder .wj-sheet > div	Styles the div host element that displays the sheets inside the tablholder.

Copyright © GrapeCity, Inc. All rights reserved.

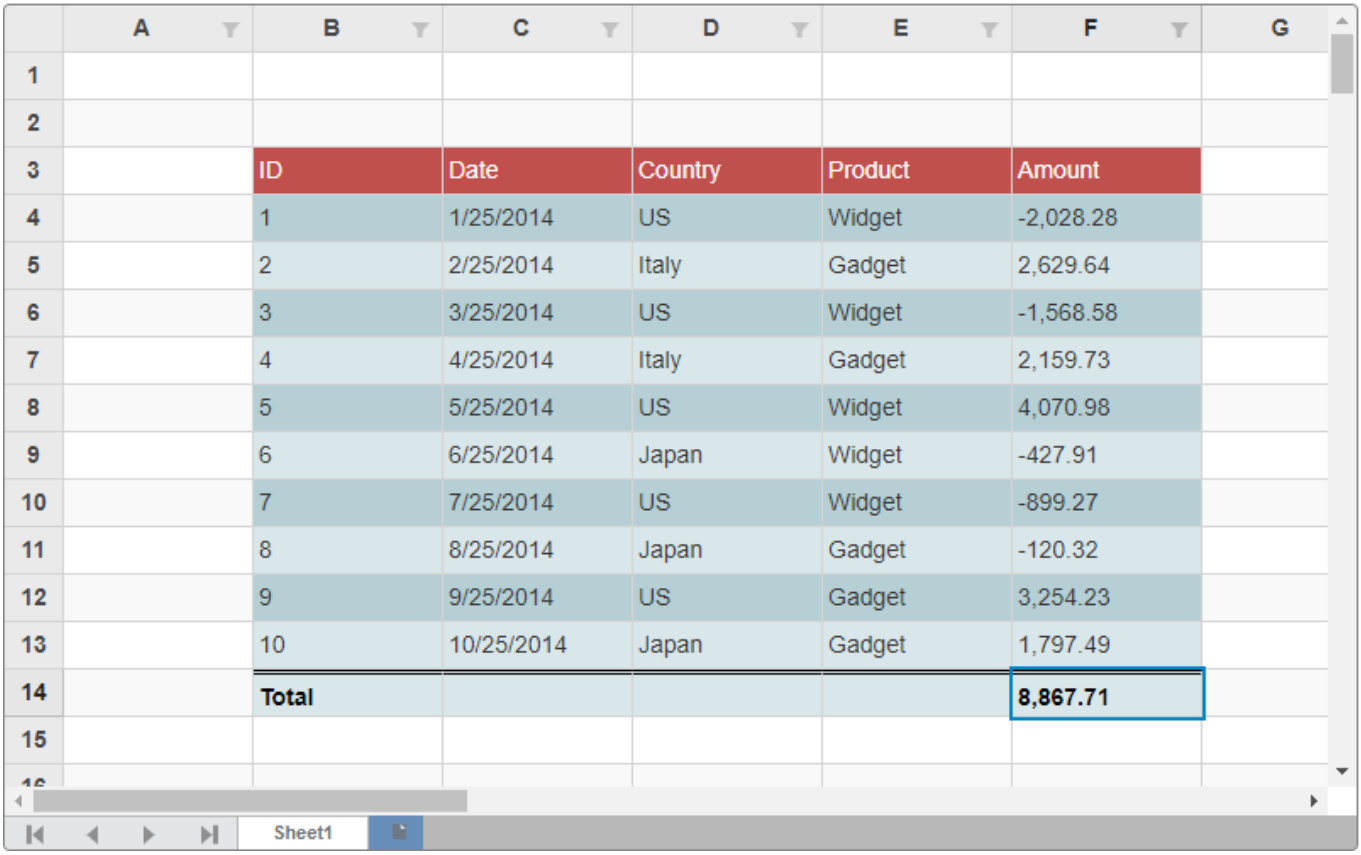
.wj-flexsheet.wj-rtl div.wj-sheet > div	Implementation not found.
Context Menu	
.wj-flexsheet-context-menu	Styles the context menu that is displayed when you make a right click on any of the cell in the FlexSheet control.
.wj-flexsheet-context-menu .wj-context-menu-item	Styles the menu items displayed in the context menu.
.wj-flexsheet-context-menu .wj-context-menu-item:not(.wj-context-menu-item-selected):hover	Styles the menu items displayed in the context menu when you hover your mouse on each menu item.
.wj-flexsheet-context-menu .wj-context-menu-item-selected	Styles the selected menu item in the context menu.
Fill Menu	
.wj-flexsheet-fill-menu	Styles the fill menu that is displayed when you make a left click on the smart tag.
.wj-flexsheet-fill-menu .wj-flexsheet-fill-menu-item	Styles the menu items displayed in the fill menu.
.wj-flexsheet-fill-menu .wj-flexsheet-fill-menu-item-active	Styles the currently active menu item in the fill menu.
.wj-flexsheet-fill-menu .wj-flexsheet-fill-menu-item:not(.wj-flexsheet-fill-menu-item-active):hover	Styles the menu items displayed in the fill menu when you hover your mouse on each menu item.
Smart Tag	
.wj-flexsheet-smart-tag	Styles the fill menu button.
.wj-flexsheet-smart-tag-active	Styles the fill menu button when it is in active state.

表

In the FlexSheet control, you can easily turn a range of cells into a table with data source. To add tables in the **FlexSheet** control, firstly, you need to add an unbound sheet, and then use its instance to access the **Tables** method, which allows you to add a table in your FlexSheet. To populate data and setting up the table, you need to set some basic methods, such as **Name**, **Range**, and **Bind** provided by the **TableBuilder** class. The columns are auto generated in the Table. However, in case you want to specify the columns in a different order or specify limited number of columns, you can use the **Add** method. Apart from this, you can also apply styles to the table with the help of **TableStyle** class.

The following image shows a table in the FlexSheet control.

Copyright © GrapeCity, Inc. All rights reserved.



	A	B	C	D	E	F	G
1							
2							
3		ID	Date	Country	Product	Amount	
4		1	1/25/2014	US	Widget	-2,028.28	
5		2	2/25/2014	Italy	Gadget	2,629.64	
6		3	3/25/2014	US	Widget	-1,568.58	
7		4	4/25/2014	Italy	Gadget	2,159.73	
8		5	5/25/2014	US	Widget	4,070.98	
9		6	6/25/2014	Japan	Widget	-427.91	
10		7	7/25/2014	US	Widget	-899.27	
11		8	8/25/2014	Japan	Gadget	-120.32	
12		9	9/25/2014	US	Gadget	3,254.23	
13		10	10/25/2014	Japan	Gadget	1,797.49	
14		**Total**				**8,867.71**	
15							

Sheet1

The following code snippets shows how to add a table in the FlexSheet control.

Model - Sale.cs

Sale.cs

```
public class Sale
    {
        public int ID { get; set; }
        public DateTime Date { get; set; }
        public string Country { get; set; }
        public string Product { get; set; }
        public double Amount { get; set; }
        public bool Active { get; set; }

        private static List<string> COUNTRIES = new List<string> { "US", "Germany",
"UK", "Japan", "Italy", "Greece" };
        private static List<string> PRODUCTS = new List<string> { "Widget", "Gadget",
"Doohickey" };

        public static IEnumerable<Sale> GetData(int total)
        {
            var rand = new Random(0);
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var date = new DateTime(2014, i % 12 + 1, 25);

                return new Sale
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                {
                    ID = i + 1,
                    Date = date,
                    Country = country,
                    Product = product,
                    Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Active = (i % 4 == 0)
                };
            });
            return list;
        }
    }
```

View - Index.csthml

Razor

```
@using C1.Web.Mvc
@using C1.C1Excel
@using ApplicationName.Models
@model IEnumerable<Sale>

@(Html.C1().FlexSheet().Id("tableSheet").CssClass("flexSheet").Height(500).Width(800)
      .AddUnboundSheet(s => s.Tables(ts =>
         {
             var style = TableStyle.CreateBuiltInTableStyleDark(9);
             ts.Add().Name("Table1").Range(2, 1).ShowTotalRow(true).
             Style(style).Bind(Sale.GetData(10)).Columns(cs =>
                {
                    cs.Add().Name("ID");
                    cs.Add().Name("Date");
                    cs.Add().Name("Country");
                    cs.Add().Name("Product");
                    cs.Add().Name("Amount").TotalRowLabel("Total Sum");
                });
})))
```

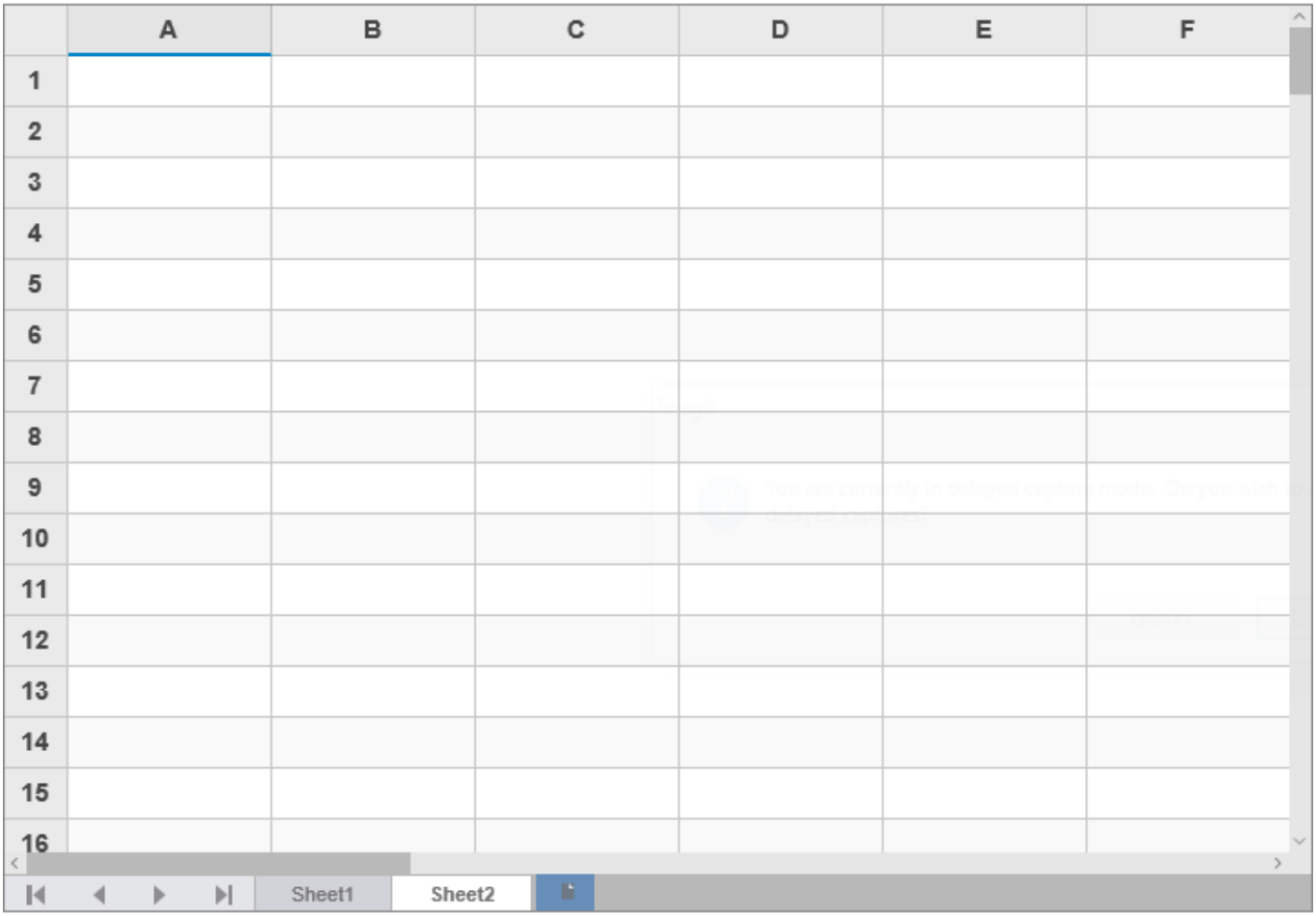
非連結 シート

アプリケーション内のFlexSheetコントロールは、データソースに連結することも、非連結にすることもできます。コードからビルダメソッド**AddUnboundSheet**を使用して、MVCアプリケーション内に非連結FlexSheetを作成できます。メソッドパラメータとして、シート名と共に、必要な行数と列数を指定します。

非連結FlexSheetの各セルにデータを手動で設定したり、プログラムによってセル値を設定することができます。このようなシートをアプリケーションフォームとして使用して、フォーム内のさまざまなフィールドにデータを提供することができます。

次の図は、非連結モードのMVC FlexSheetです。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、MVCアプリケーションに非連結FlexSheetを追加する方法を示します。

コードの場合

UnboundController.cs

C# copyCode

```
// GET: Unbound
public ActionResult Index()
{
    return View();
}
```

Unbound.cshtml

Razor copyCode

```
<div>
    @(Html.C1().FlexSheet().CssClass("flexSheet")
    .AddUnboundSheet("Sheet1", 25, 12)
    .AddUnboundSheet("Sheet2", 25, 12))
</div>
```

先頭に戻る

Copyright © GrapeCity, Inc. All rights reserved.

FlexViewer

FlexViewer for MVC comprises of two types of viewers, namely ReportViewer and PdfViewer. The ReportViewer can be used to view FlexReports in your web applications and the PdfViewer enables you to view PDF files. For detailed information on these viewers, see the following topics:

- ReportViewer
- PdfViewer

ReportViewer

ASP.NET MVC のReportViewerコントロールを使用すると、WebアプリケーションでFlexReportsをプレビューできます。ReportViewerは高速で柔軟なHTML5ベースのレポートビューアです。これを使用して、ブラウザにC1Report、FlexReportsやSSRSレポートを簡単に表示できます。本コントロールでは、対話式のユーザーフレンドリなユーザーインタフェースを使用してレポートを表示するさまざまなオプションが提供されています。レポートは、完全な印刷機能と最高品質のレポート表示エクスペリエンスを提供するScalable Vector Graphics（SVG）形式を使用してレンダリングされます。

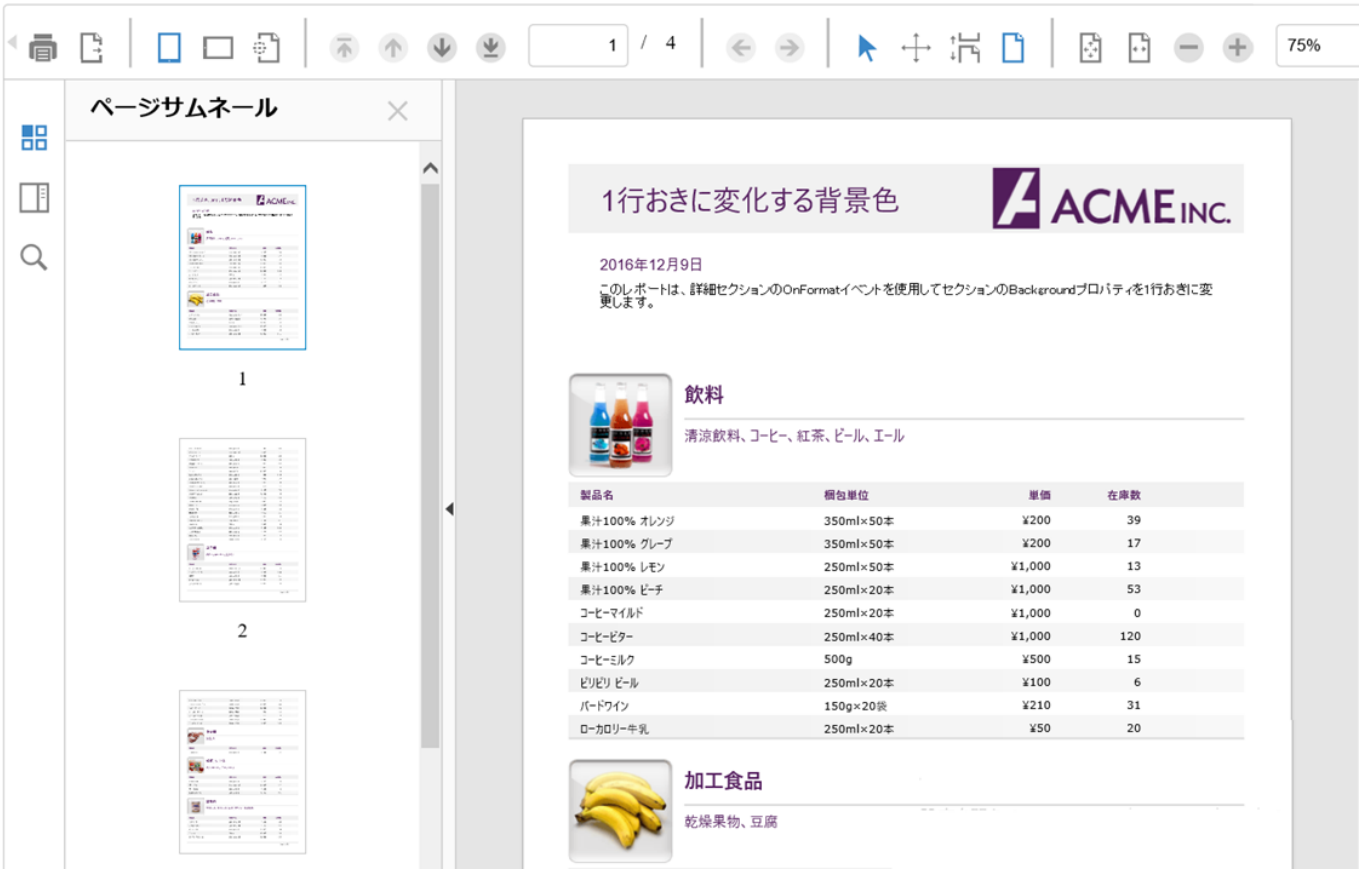


レポートは、エクスポートオプションを使用してFlexViewerコントロールから複数の形式に簡単にエクスポートできます。FlexViewerコントロールでサポートされているエクスポート形式は、Adobe PDF、HTML、RTF、Open XML Word、Open XML Excel、TIFF、BMP、PNG、JPEG、およびGIFです。

ReportViewerの要素

FlexViewer for ASP.NET MVC Editionは、すべてのレポートのための単一のビューアです。FlexViewerは多くのナビゲーションツールを備え、レポートを簡単にナビゲートおよび操作できます。FlexViewerでレポートをプレビューすると、FlexViewerの周囲に多数のツールが表示されます。このトピックは、FlexViewerにあるさまざまなナビゲーションツールの理解に役立ちます。

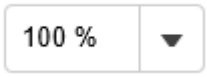
Copyright © GrapeCity, Inc. All rights reserved.



FlexViewerは、次の要素で構成されます。

要素	名前	説明
ツールバーオプション - レポートで基本的な操作を実行するためによく使用されるコマンドで構成されます。		
	印刷	印刷オプションを指定するための印刷ダイアログを表示します。
	エクスポート	レポートをエクスポートする形式をドロップダウンリストに表示します。有効な形式は、**Adobe PDF**、**HTML**、**RTF**、**Open XML Word**、**Open XML Excel**、**TIFF**、**BMP**、**PNG**、**JPEG**、および**GIF**です。
	縦	レポートを縦向きに表示します。
	横	レポートを横向きに表示します。
	ページ設定	現在選択されているページの**[用紙の種類]**、**[向き]**、および**[マージン(インチ)]**を設定するための[ページ設定]ダイアログを表示します。
	ツールを選択します	マウスポインタを使用してレポートコンテンツを選択します。
	移動ツール	レポートのレポートコンテンツを移動します。

Copyright © GrapeCity, Inc. All rights reserved.

	連続ページビュー	レポートを連続スクロールページビューに表示します。
	単一ページビュー	ページの幅をビューアのサイズに合わせます。
	ページ全体を表示	ページ全体を現在のビューアのサイズに合わせます。
	ページ幅を合わせる	ページの幅をビューアのサイズに合わせます。
	ズームイン/ズームアウト	ズームイン - レポートの拡大率を下げます。 ズームアウト - レポートの拡大率を上げます。
	選択によるズーム	現在の選択項目に合わせてズームし、選択されているエンティティすべてが表示されるようにします。
	拡大鏡	レポートのさまざまな部分を拡大します。
	ドキュメントを回転	レポートページをすべて回転します。
	ページを回転	現在選択されているページを回転します。
	拡大率(%)	パーセント値によってズーム値を設定できます。
	全画面表示	レポートをフルスクリーンモードで開きます。
サイドバーオプション - このセクションは、レポートに表示されるページサムネイル、見出しマップ、パラメータ、および検索ペインで構成されます。		
	ページサムネイル	ページサムネイルペインは、サイドバーの**[サイドバーの切り替え]**をクリックすると、デフォルトでサイドバーに表示されます。 このペインは、レポートのすべてのページのサムネイルビューで構成されます。サムネイルをクリックすると、選択したレポートページに直接移動します。[+]ボタンまたは[-]ボタンをクリックしてズームインまたはズームアウトし、サムネイルのサイズを変更することもできます。
	見出しマップ	サイドバーの**[見出しマップ]**ボタンをクリックすると、[見出しマップ]ペインが表示されます。このペインには、テキストボックス、グループ、またはサブレポートに付けたラベルが表示され、それらをクリックして、ビューア内のレポートの対応する領域に移動できます。

Copyright © GrapeCity, Inc. All rights reserved.

	パラメータ	サイドバーの**[パラメータ]**ボタンをクリックすると、[パラメータ]ペインが表示されます。FlexViewerでは、パラメータ付きのレポートを表示できます。レポートにパラメータが含まれていると、[パラメータ]ペインが自動的に表示されます。 [パラメータ]ペインでは、表示するデータをフィルタ処理する値を入力します。
	検索	サイドバーの**[検索]**ボタンをクリックすると、[検索]ペインが表示されます。[検索]ペインでは、レポート内の特定のテキストを検索できます。

C1MVC ReportViewerテンプレートでレポートの表示

このセクションでは、**ComponentOne MVC ReportViewer**テンプレートを使用して、MVCアプリケーション内にFlexReportを表示する方法について説明します。ComponentOne MVC ReportViewerテンプレートは、FlexViewerでレポートをプレビューする3つの方法（現在のプロジェクトのレポートと他のレポートサービスのレポート）を提供します。

- **現在のプロジェクトのレポート**

  MVC 5アプリケーションが対象の場合は、ComponentOne MVC ReportViewerテンプレートの[現在のプロジェクトのレポート]オプションを使用して、Visual Studioアプリケーションにレポートを追加してから、それをFlexViewerに表示できます。

  ComponentOne MVCレポートビューアを使用してアプリケーションを作成すると、必要なリソースが自動的に登録され、関連するWeb APIリソースとパッケージがアプリケーションに追加されます。

- **他のレポートサービスのレポート**

  MVC 3、4、およびASP.NET Core Frameworkアプリケーションが対象の場合、FlexViewerにレポートを表示するには、ComponentOne MVCレポートビューアテンプレートの[他のレポートサービスのレポート]オプションだけを使用できます。このオプションは、Web API Serviceプロジェクトでレポートがホストされている場合に使用できます。

- **SSRSサーバーのレポート**

  MVC 5、およびASP.NET Core Frameworkアプリケーションが対象の場合、FlexViewerにSSRSレポートを表示するには、ComponentOne MVCレポートビューアテンプレートの[SSRSサーバーのレポート]オプションだけを使用できます。このオプションは、サーバーでSSRSレポートがホストされている場合に使用できます。

ComponentOne MVC ReportViewerの操作

以下の手順では、ComponentOne MVC ReportViewerウィザードを使用し、Visual Studioテンプレートを使用してFlexViewerコントロールにレポートを表示する方法を示します。

- **手順1：アプリケーションへの使用許諾の供与**
- **手順2：リソースの登録**
- **手順3：コントローラーの追加**

メモ: ASP.NET MVC の**ComponentOne ASP.NET MVC 5 Web Application**テンプレートと**ComponentOne ASP.NET Core MVC Application**テンプレートは、自動的に必要なリソースを登録し、関連する参照とパッケージをアプリケーションに追加します。したがって、**ComponentOne**テンプレートを使用してアプリケーションを作成した場合は、**ComponentOne MVC ReportViewer** テンプレートを直接使用できます。

次の図は、上記の手順を実行した後のFlexViewer内のFlexReportを示しています。

Copyright © GrapeCity, Inc. All rights reserved.

手順1：アプリケーションへの使用許諾の供与

1. **ソリューションエクスプローラー**で、プロジェクトを右クリックし、**[追加]**→**[新しい項目]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、**[C#]**→**[一般]**を選択し、右ペインで**[テキストファイル]**を選択します。
3. テキストファイルに「**licenses.licx**」という名前を付けます。
4. **licenses.licx**ファイルに以下を追加します。

licenses.licx

```
C1.Web.Mvc.LicenseDetector, C1.Web.Mvc
C1.Web.Mvc.Viewer.LicenseDetector, C1.Web.Mvc.FlexViewer
```

> **メモ: ASP.NET Coreアプリケーション**が対象の場合は、グレープシティライセンスマネージャーを使用して、リソースとアプリケーションに使用許諾を供与できます。詳細については、「ライセンス」を参照してください。

先頭に戻る

手順2：リソースの登録

ASP.NET MVC FlexViewerコントロールを使用するために必要なリソースを登録するには、次の手順を実行します。

1. **ソリューションエクスプローラー**で、**[ビュー]**→**[共有]**フォルダを開きます。
2. `_Layout.cshtml`をダブルクリックして開きます。
3. `<head></head>`タグの間に次のコードを追加します。

_Layout.cshtml

```
@Html.C1().Styles()
@Html.C1().Scripts().Basic().FlexViewer()
```

FlexViewerのリソースを登録する方法については、「リソースの登録」を参照してください。

先頭に戻る

手順3：コントローラーの追加

アプリケーションにコントローラーを追加するには、次の手順を実行します。

1. **[コントローラー]**フォルダを右クリックし、**[追加]**→**[新規スキャフォールディングアイテム]**を選択します。
2. **[スキャフォールディングを追加]**ウィザードで、**[MVC5コントローラー - 空]**を選択し、**[追加]**をクリックします。
3. コントローラーに名前を付けます。たとえば、コントローラーに**ReportController**という名前を付けます。

アプリケーションの**[コントローラー]**フォルダ内に新しいコントローラーが追加されます。

先頭に戻る

現在のプロジェクトのレポート

現在のプロジェクトに保存されているレポートを表示するには、MVCアプリケーションを構成する必要があります。詳細については、「**MVCアプリケーションの構成**」を参照してください。

現在のプロジェクトに保存されているレポートを表示するには、以下の手順を完了します。

1. **[ビュー]**の下で**[レポート]**フォルダを右クリックしてから、**[追加]**→**[新しい項目]**を選択して[新しい項目の追加]ダイアログを開きます。
2. **[インストール済み]**→**[テンプレート]**の下で、**[Visual C#]**→**[Web]**→**[ComponentOne ReportViewer View Page]**を選択して[ComponentOne MVC ReportViewer]ダイアログを開きます。

Copyright © GrapeCity, Inc. All rights reserved.

3. [ComponentOne MVC ReportViewer]ダイアログで、**[現在のプロジェクトのレポート]**を選択します。



4. **[レポートファイル]**フィールドで、**[参照]**をクリックして、システムにあるFlexReport(.flxr)ファイルを見つけます。この例では、FlexCommonTasks.flxrレポートを使用しました。
5. **[レポート名]**ドロップダウンで、リストから**レポート名**を選択して、それをFlexViewerに表示します。この例では、**Simple List**レポートを指定しました。
6. **[OK]**をクリックして、Index.cshtmlビューページを作成します。

Index.cshtml

```
<head>
    <title>ComponentOne MVC ReportViewer</title>
    @Html.C1().Resources(typeof(C1Grid))
    @Html.C1().FlexViewerResources()
</head>
<body>
@(Html.C1().ReportViewer().FilePath
     (@"~\Content\ReportsRoot\FlexCommonTasks.flxr")
```


Copyright © GrapeCity, Inc. All rights reserved.

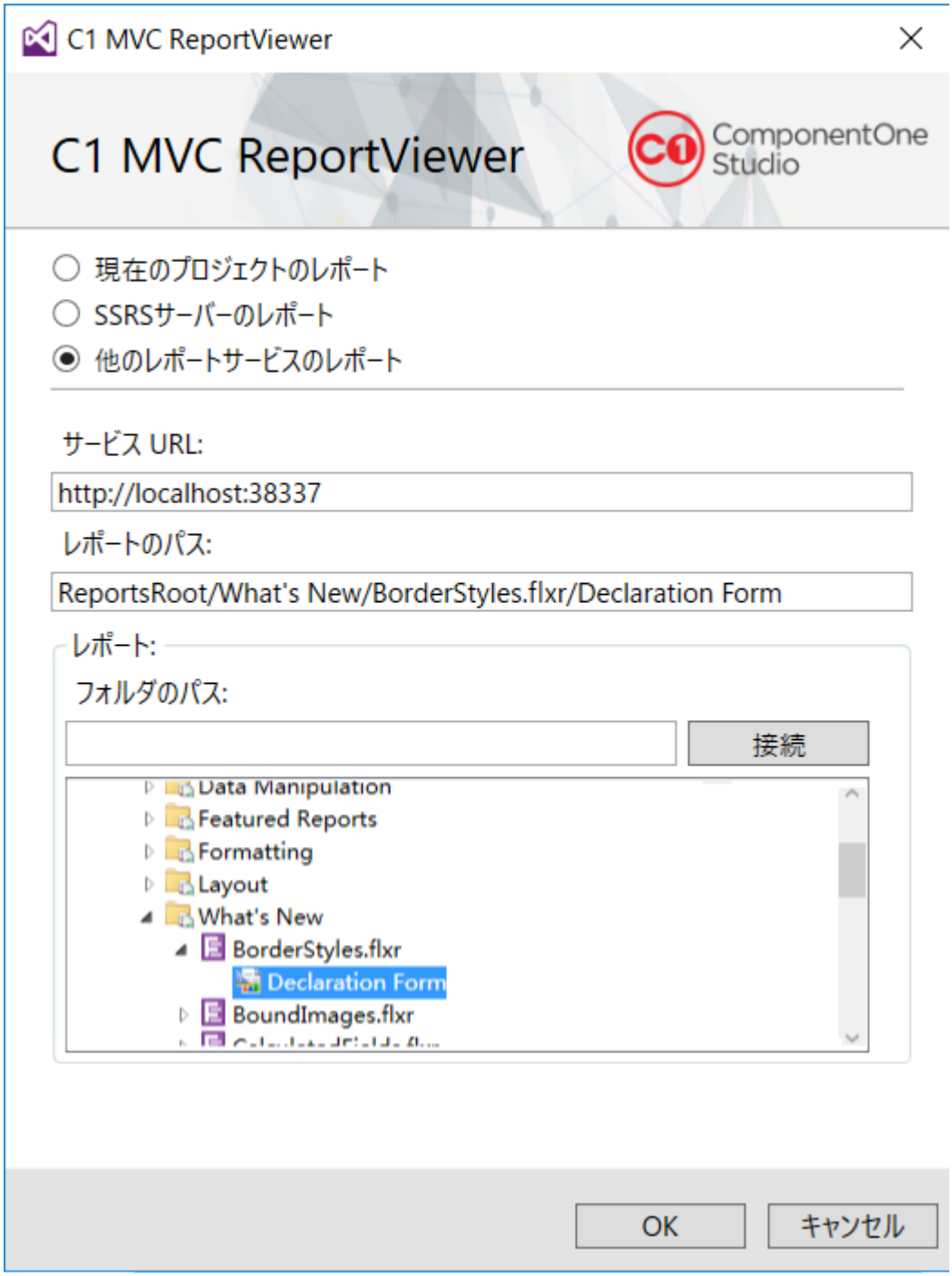
```
        .ReportName(@"Simple List"))
</body>
```

他のレポートサービスのレポート

レポートサービスに保存されているレポートを表示するには、MVCアプリケーションを構成する必要があります。アプリケーションにビューページを追加するには、次の手順を実行します。レポートビューアの詳細については、「ComponentOne ReportViewerテンプレートの使用」を参照してください。

1. **[ビュー]**の下で**[レポート]**フォルダを右クリックしてから、**[追加]**→**[新しい項目]**を選択して[新しい項目の追加]ダイアログを開きます。
2. **[インストール済み]**→**[テンプレート]**の下で、**[Visual C#]**→**[Web]**→**[ComponentOne ReportViewer View Page]**を選択して[ComponentOne MVC ReportViewer]ダイアログを開きます。
3. **[ComponentOne MVC ReportViewer]**ダイアログで、**[他のレポートサービスのレポート]**オプションを選択します。

Copyright © GrapeCity, Inc. All rights reserved.



4. **[サービスURL]**フィールドにWeb APIサービスURLを入力してレポートにアクセスします。たとえば、http://localhost:4615/FlexReportWebAPIです。
5. **[レポートのパス]**フィールドで、レポートが保存されているフォルダを指定します。
6. **[接続]**をクリックしてWebApiサーバーとの接続を確立します。WebApiサーバーに接続すると、**[レポート]**セクションにすべてのレポートの名前が表示されます。
7. リストからレポートを選択し、**[OK]**をクリックして、**Index.cshtml**ビューページを作成します。

Index.cshtml

```
<head>
    <title>ComponentOne MVC ReportViewer</title>
    @Html.C1().Styles()
        @Html.C1().Scripts().FlexViewer()
</head>
<body>
```


Copyright © GrapeCity, Inc. All rights reserved.

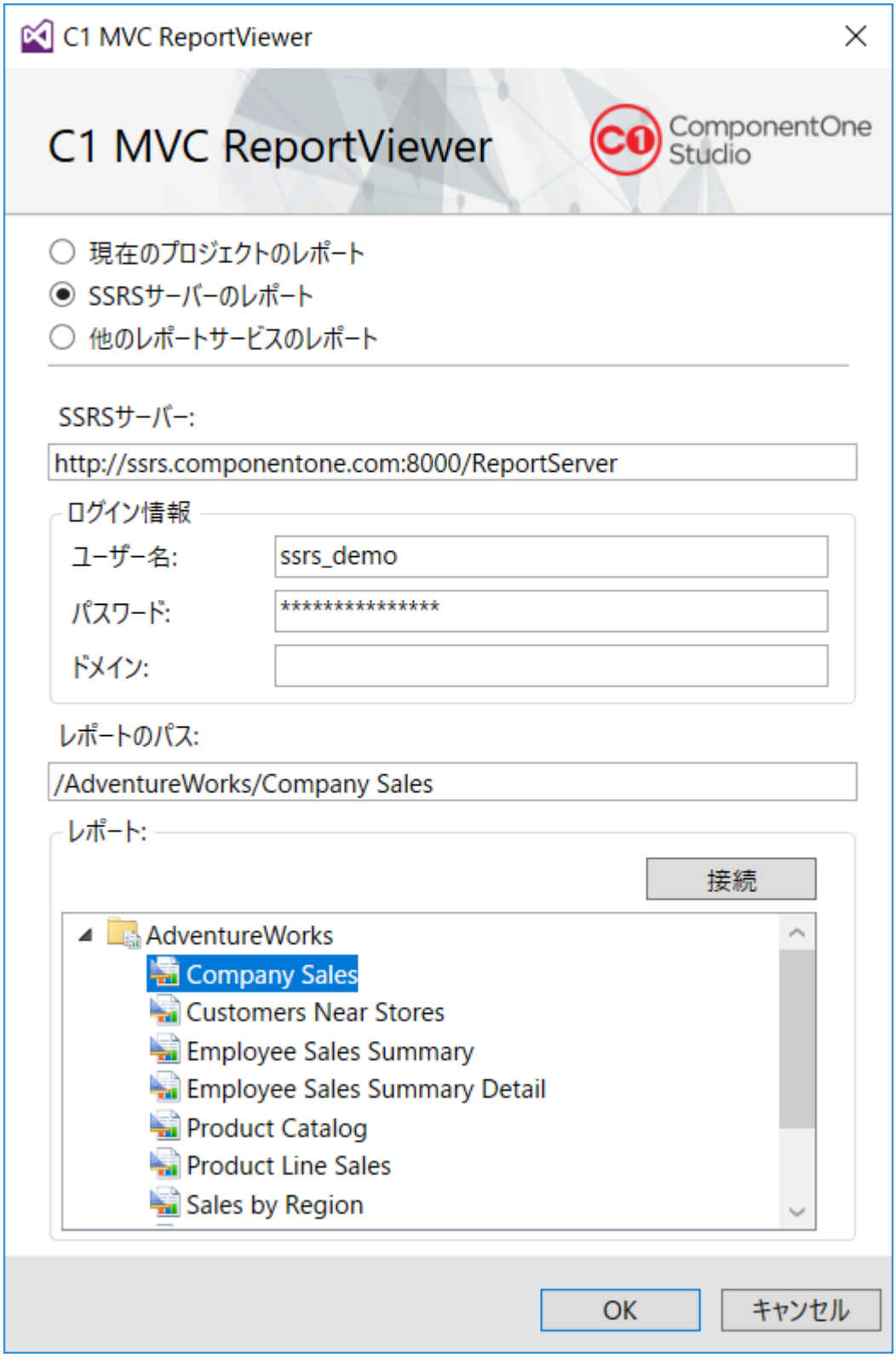
```
@(Html.C1().ReportViewer().ServiceUrl(@"http://localhost:4615/
FlexReportWebAPI").FilePath(@"FlexCommonTasks.flxr").ReportName(@"Simple List"))
</body>
```

SSRSサーバーのレポート

SSRSレポートサーバーに保存されているレポートを表示するには、MVCアプリケーションを構成する必要があります。アプリケーションにビューページを追加するには、次の手順を実行します。レポートビューアの詳細については、「ComponentOne ReportViewerテンプレートの使用」を参照してください。

1. **[ビュー]**の下で**[レポート]**フォルダを右クリックしてから、**[追加]**→**[新しい項目]**を選択して[新しい項目の追加]ダイアログを開きます。
2. **[インストール済み]**→**[テンプレート]**の下で、**[Visual C#]**→**[Web]**→**[ComponentOne ReportViewer View Page]**を選択して[ComponentOne MVC ReportViewer]ダイアログを開きます。
3. **[ComponentOne MVC ReportViewer]**ダイアログで、**[SSRSサーバーのレポート]**を選択します。

Copyright © GrapeCity, Inc. All rights reserved.



4. **[SSRSサーバー]**フィールドに、レポートにアクセスするためのSSRSサーバーURLを入力します。たとえば、http://sh-tools-dev03/reportserverです。
5. **[ログイン情報]**セクションで、SSRSサーバーURLにアクセスするためのユーザー名とパスワードを指定します。
6. **[レポートのパス]**フィールドで、SSRSレポートが保存されているフォルダを指定します。
7. **[接続]**をクリックしてSSRSサーバーとの接続を確立します。SSRSサーバーに接続すると、**[レポート]**セクションにすべてのレポートの名前が表示されます。
8. リストからレポートを選択し、**[OK]**をクリックして、**Index.cshtml**ビューページを作成します。

Index.cshtml

Copyright © GrapeCity, Inc. All rights reserved.

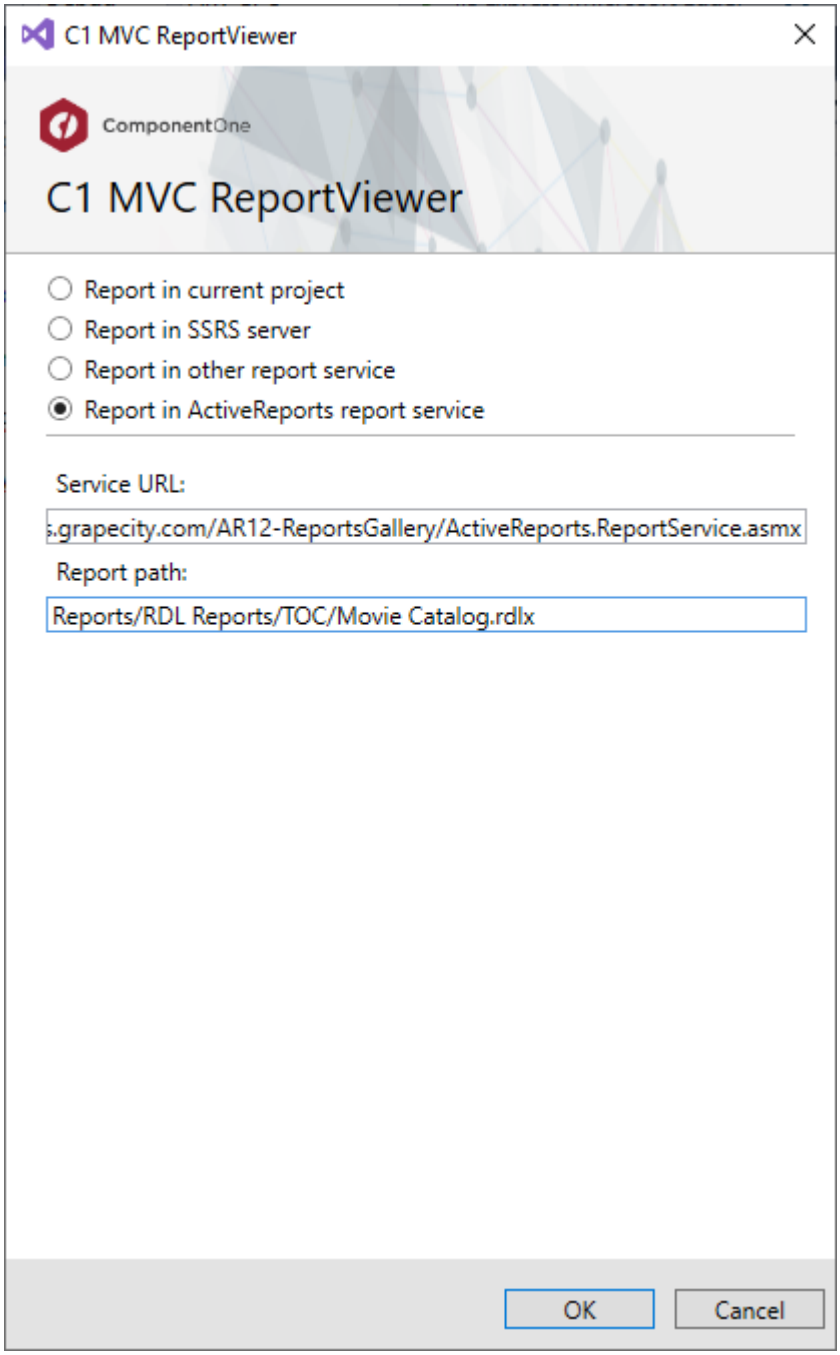
```
<head>
    <title>ComponentOne MVC ReportViewer</title>
    @Html.C1().Styles()
    @Html.C1().Scripts().FlexViewer()
</head>
<body>
@(Html.C1().ReportViewer().FilePath(@"SSRS/AdventureWorks/Company Sales"))
</body>
```

ActiveReportsの表示

To view a ActiveReports, you need to configure your MVC application. Complete the following steps to add the view page to your application. For more information about using Report Viewer, see Using C1 ReportViewer Template.

1. Under **Views**, right-click the **Report** folder, and then select **Add | New Item...** to open the Add New Item dialog.
2. Under **Installed | Templates**, select **Visual C# | Web | C1 ReportViewer View Page** to open the C1 MVC ReportViewer dialog.
3. In the **C1 MVC ReportViewer** dialog, select **Report in ActiveReports report service** option.

Copyright © GrapeCity, Inc. All rights reserved.

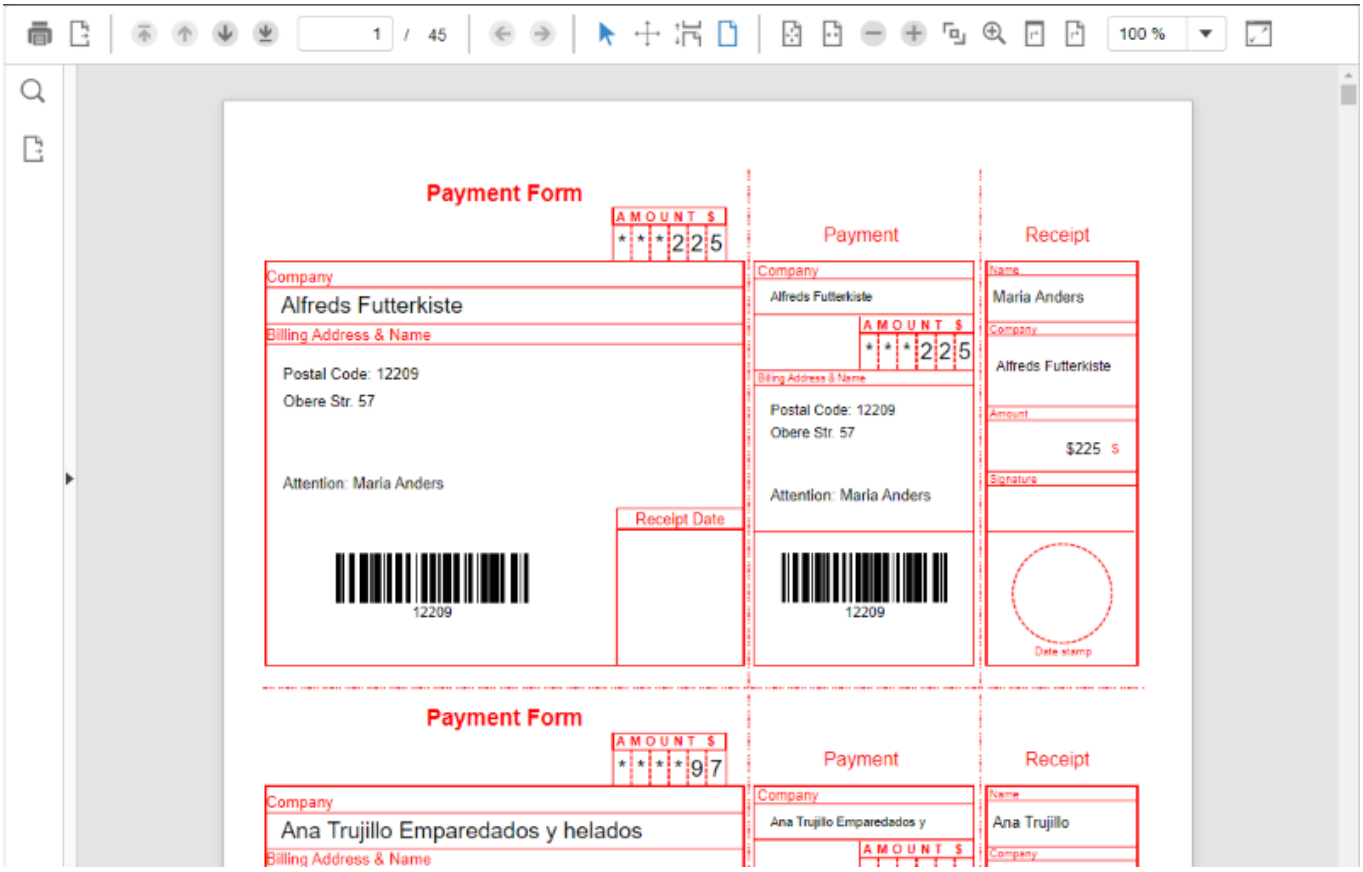


4. In the **Service URL** field, enter the service URL to access the reports. In our case, we used http://ardemos.grapecity.com/AR12-ReportsGallery/ActiveReports.ReportService.asmx.
5. In the Report path, enter the path of the report. In our case, we used **Reports/RDL Reports/TOC/Movie Catalog.rdlx**.
6. Click **OK**. The View page is created.

ActiveReportsサービスの構成

You can configure the ActiveReports service locally in an ASP.NET MVC application to view ActiveReports in FlexViewer. To view ActiveReports in the FlexViewer control using the ActiveReports Web Service, follow the steps given below.

Copyright © GrapeCity, Inc. All rights reserved.



Note: Make sure that the latest version of **ActiveReports** is installed on your system, before implementing the steps mentioned below.

1. Create a new ASP.NET MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.
2. Add the FlexViewer library to your application.
   - If using ComponentOne template, select "Include FlexViewer Library" option from the ComponentOne ASP.NET MVC Application Wizard.
   - If using Visual Studio template, add the `C1.Web.Mvc.FlexViewer.dll` reference and the following markup in <namespace></namespace> tags below `C1.Web.Mvc` markup.

     ```
     <add namespace="C1.Web.Mvc.Viewer" />
     <add namespace="C1.Web.Mvc.Viewer.Fluent" />
     ```

3. In the SolutionExplorer, right click the project name and then select **Add | New Item** to open the **New Item** wizard.
4. In the **New Item** wizard, on the left panel under the installed templates, select **Reporting | ActiveReports 13 Web Service** and then click **Add**. As soon as you add the service, you will notice that it adds all the dependent assembly references and updates the **Web.Config** file to include instances of assembly references and http handlers.
5. Add the following code in the **App_Start | RouteConfig.cs** file.

   C#

   ```
   routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
   routes.IgnoreRoute("{*allActiveReport}", new { allActiveReport =
   @".*\.ar12(/.*)?" });
   ```

6. Add a folder named **Reports** to your application. In the **Reports** folder, add the **BillingInvoice.rdlx** report file from the following location:

Copyright © GrapeCity, Inc. All rights reserved.

Documents\GrapeCity Samples\ActiveReports (latest version)\Reports Gallery\Reports\Page Report\Invoice

7. In **Web.config** file, update the **Reports** folder path to make it accessible for the application.

C#

```
<ActiveReports13>
    <WebService reportsFolder="~/Reports" assemblyFolder="~/" />
</ActiveReports13>
```

8. Add the following code in **Views | Home | Index.cshtml** file, to initialize the FlexViewer control and view an Active Report using ActiveReports 13 Web Service.

Index.cshtml

```
@(Html.C1().ReportViewer()
.ServiceUrl(@"~/ActiveReports.ReportService.asmx")
.FilePath("BillingInvoice.rdlx"))
```

FlexViewerを手動で設定

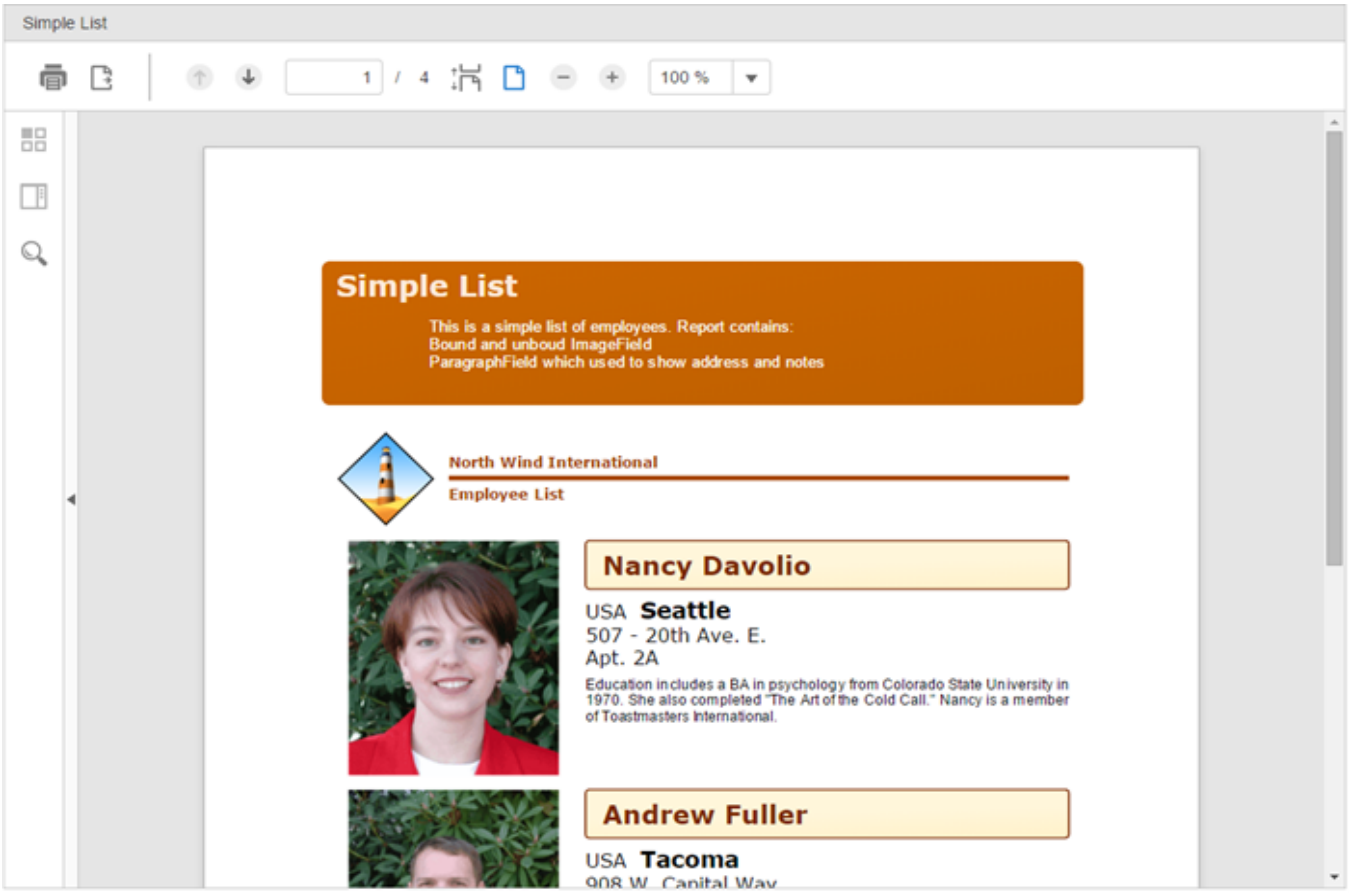
FlexViewerは、ComponentOne Web APIレポートサービスを使用してレポートを表示するためにMVCアプリケーション内で使用できるMVCコントロールとして利用できます。このコントロールを使用してレポートを表示するには、次の手順を実行します。

- **手順1：MVCアプリケーションの作成**
- **手順2：アプリケーションへの使用許諾の供与**
- **手順3：アプリケーションへの関連する参照の追加**
- **手順4：FlexViewerを使用するためのアプリケーションの設定**
- **手順5：リソースの登録**
- **手順6：コントローラーの追加**
- **手順7：コントローラーのビューの追加**
- **手順8：プロジェクトのビルドおよび実行**

メモ: ASP.NET MVC の**ComponentOne ASP.NET MVC 5 Web Application**テンプレートと**ComponentOne ASP.NET Core MVC Application**テンプレートは、自動的に必要なリソースを登録し、関連する参照とパッケージをアプリケーションに追加します。したがって、**ComponentOne**テンプレートを使用してアプリケーションを作成する場合は、上の手順の6～8を実行するだけです。

次の図は、上の手順を完了したときにFlexViewerに表示されるSimple Listという名前のレポートです。

Copyright © GrapeCity, Inc. All rights reserved.



手順1：MVCアプリケーションの作成

Visual Studioテンプレートを使用して、ASP.NET Webアプリケーション（.NET Framework）を作成します。詳細については、「Visual Studioテンプレートの使用」を参照してください。

先頭に戻る

手順2：アプリケーションへの使用許諾の供与

1. **ソリューションエクスプローラー**で、プロジェクトを右クリックし、**[追加]**→**[新しい項目]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、**[C#]**→**[一般]**を選択し、右ペインで**[テキストファイル]**を選択します。
3. テキストファイルに「**licenses.licx**」という名前を付けます。
4. **licenses.licx**ファイルに以下を追加します。

licenses.licx

```
C1.Web.Mvc.LicenseDetector, C1.Web.Mvc
C1.Web.Mvc.Viewer.LicenseDetector, C1.Web.Mvc.FlexViewers
```

アプリケーションにライセンスを追加する方法については、「ライセンス」を参照してください。

先頭に戻る

手順3：アプリケーションへの関連する参照の追加

次の手順を実行して、ASP.NET MVC の参照とFlexViewerの参照をプロジェクトに追加します。

1. **ソリューションエクスプローラー**で、**[参照]**を右クリックし、**[参照の追加]**を選択します。
2. **C:\Program Files (x86)\ComponentOne\ASP.NET MVC \bin**を参照します。
3. **C1.Web.Mvc.dll**と**C1.Web.Mvc.Finance.dll**を選択し、**[追加]**をクリックします。

Copyright © GrapeCity, Inc. All rights reserved.

4. **C1.Web.Mvc.dll**と**C1.Web.Mvc.FlexViewer.dll**の**[ローカルコピー]**プロパティを**True**に設定します。

先頭に戻る

手順4：FlexViewerを使用するためのアプリケーションの設定

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開し、`web.config`ファイルをダブルクリックして開きます。
2. `<system.web.webPages.razor></system.web.webPages.razor>`タグ内の<namespaces></namespaces>タグに次のマークアップを追加します。

HTML

```
<add namespace="C1.Web.Mvc" />
<add namespace="C1.Web.Mvc.Viewer" />
<add namespace="C1.Web.Mvc.Viewer.Fluent" />
```

先頭に戻る

手順5：リソースの登録

ASP.NET MVC FlexViewerコントロールを使用するために必要なリソースを登録するには、次の手順を実行します。

1. **ソリューションエクスプローラー**で、**[ビュー]**→**[共有]**フォルダを開きます。
2. `_Layout.cshtml`をダブルクリックして開きます。
3. 以下のコードを`<head></head>`タグ内追加します。

_Layout.cshtml

```
@Html.C1().Styles()
@Html.C1().Scripts().FlexViewer()
```

FlexViewerのリソースを登録する方法については、「リソースの登録」を参照してください。

先頭に戻る

手順6：コントローラーの追加

アプリケーションにコントローラーを追加するには、次の手順を実行します。

1. **[コントローラー]**フォルダを右クリックし、**[追加]**→**[新規スキャフォールディングアイテム]**を選択します。
2. **[スキャフォールディングを追加]**ウィザードで、**[MVC5コントローラー - 空]**を選択し、**[追加]**をクリックします。
3. コントローラーに名前を付けます。たとえば、コントローラーに**ReportController**という名前を付けます。

アプリケーションの**[コントローラー]**フォルダ内に新しいコントローラーが追加されます。

先頭に戻る

手順7：コントローラーのビューの追加

コントローラーに対応するビューを追加するには、次の手順を実行します。

1. コントローラー（この例ではReportController）のIndex()メソッド内にカーソルを置きます。
2. 右クリックし、オプションから**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
3. **[ビューの追加]**ダイアログで、**[ビュー名]**を設定します。たとえば、この例では「Index」です。
4. **[追加]**をクリックします。
5. プロジェクトに**index.cshtml**ページが追加されたら、次のコードを追加して、FlexViewerにレポートを表示します。

コントローラーにビューが追加されます。次のコードでは、**サービスURL**、**ファイルパス**、および**レポート名**を指定しました。

Index.cshtml

```
@(Html.C1().ReportViewer()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
.ServiceUrl(@"https://demos.componentone.com/ASPNET/c1webapi/4.0.20171.91/api/report")
        .FilePath(@"ReportsRoot/FlexCommonTasks/FlexCommonTasks.flxr")
        .ReportName(@"Simple List"))
```

コントローラーにビューが追加されます。

先頭に戻る

手順8：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行し、FlexViewerにレポートを表示します。FlexViewerの詳細については、「FlexViewerの要素」を参照してください。

   > **メモ**：ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：**http://localhost:1234/Report/Index**）。

> **メモ**：詳細については、FlexViewerを使用する詳細なデモを参照してください。

先頭に戻る

MobileView

ASP.NET MVC EditionのReportViewerコントロールは、MobileViewモードをサポートしています。MobileViewerを使用すると、あらゆる携帯機器でレポートをレンダリングおよび表示できます。これは、C1Report、FlexReports、SSRS Reportsを表示できる高速で柔軟なHTML5ベースのレポートビューアです。レスポンシブレイアウトがデバイスのディスプレイに自動的に適応し、最高の表示エクスペリエンスが提供されます。MobileViewerでは、対話式のユーザーフレンドリなユーザーインタフェースを使用して、レポートを表示するさまざまなオプションが提供されています。ユーザーインタフェースの詳細については、「MobileViewerの要素」を参照してください。

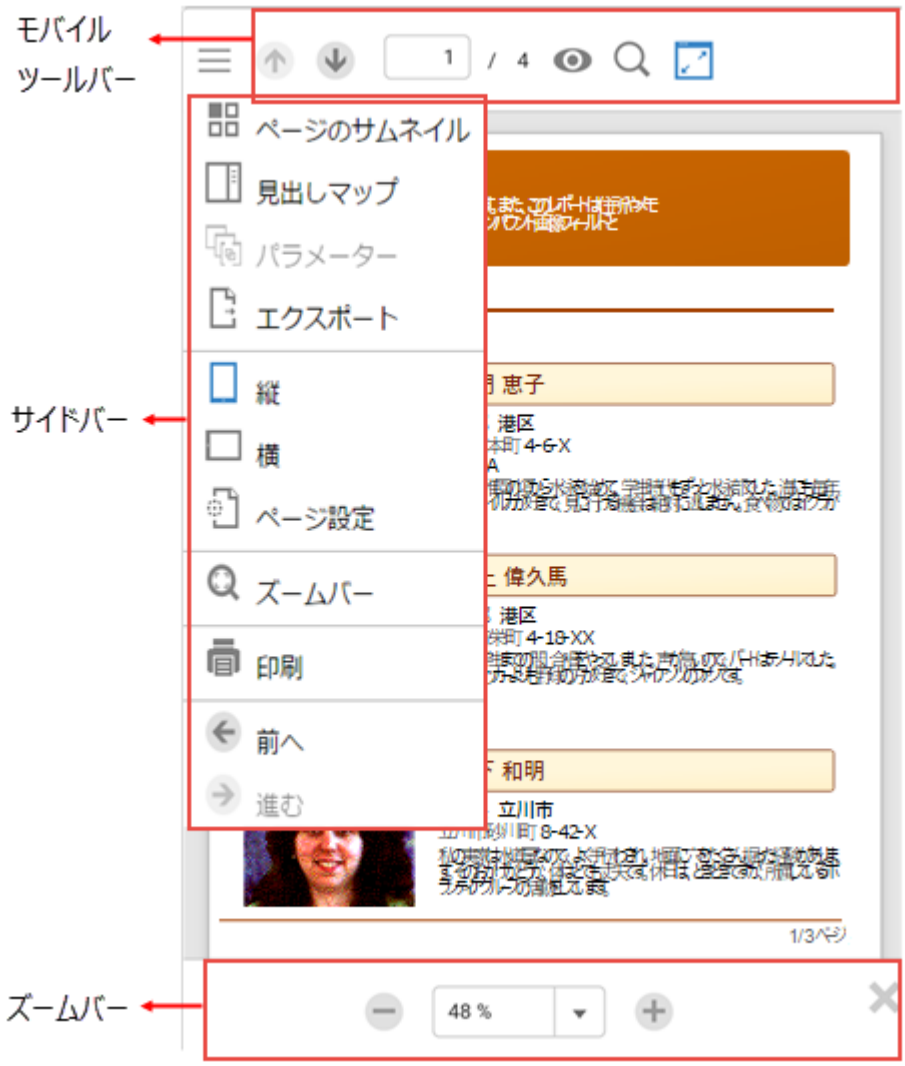
Copyright © GrapeCity, Inc. All rights reserved.



MobileViewerの要素

ASP.NET MVC EditionのMobileViewerを使用すると、あらゆる携帯機器でレポートを表示およびレンダリングすることができます。MobileViewerはさまざまなナビゲーションツールを備え、レポートを簡単にナビゲートおよび操作できます。MobileViewerでレポートをプレビューすると、サイドメニューにあるオプションを使用してレポートを操作することができます。このトピックは、MobileViewerにあるさまざまなナビゲーションツールの理解に役立ちます。

Copyright © GrapeCity, Inc. All rights reserved.



MobileViewerは、次の要素で構成されます。

要素	名前	説明
モバイルツールバーのオプション - このセクションは、レポートで基本的な操作を実行するためによく使用されるコマンドで構成されます。ツールバーで提供されるコマンドはすべて、ビューアに現在表示されているレポートに対して動作します。		
	前のページ/次のページ	前のページまたは次のページに移動することでレポート内をナビゲートできます。
	レイアウトオプション	レイアウトオプションは、単一ページビュー、連続ページビュー、ページ全体を表示、ページ幅を合わせるの各オプションで構成されます。
	単一ページビュー	ページの幅をビューアのサイズに合わせます。
	連続ページビュー	レポートを連続スクロールページビューに表示します。
	ページ全体を表示	ページ全体を現在のビューアのサイズに合わせます。

Copyright © GrapeCity, Inc. All rights reserved.

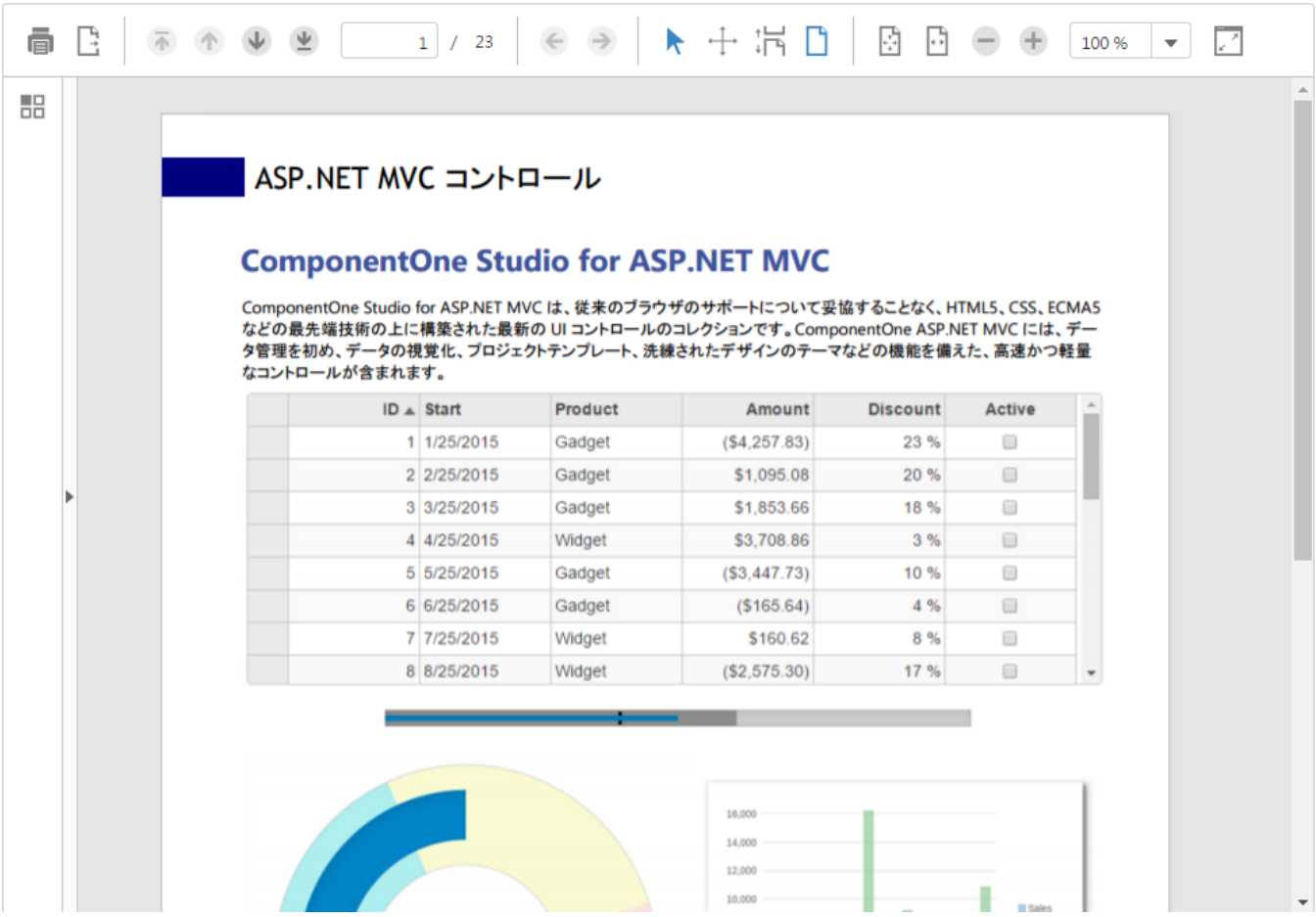
	ページ幅を合わせる	ページの幅をビューアのサイズに合わせます。
	検索バーの表示	ツールバーの**[検索バーの表示]**ボタンをクリックすると、ビューアの下部に検索バーが表示されます。検索バーでは、レポート内の特定のテキストを検索できます。
	フルスクリーン	レポートをフルスクリーンモードで表示します。
サイドバーのオプション - このセクションは、レポートに表示されるページのサムネイル、見出しマップ、パラメータ、および検索ペインで構成されます。		
	ページのサムネイル	[ページのサムネイル]ペインは、サイドバーの**[サイドバーの切り替え]**をクリックすると、デフォルトでサイドバーに表示されます。 このペインは、レポートのすべてのページのサムネイルビューで構成されます。サムネイルをクリックすると、選択したレポートページに直接移動します。
	見出しマップ	サイドバーの**[見出しマップ]**ボタンをクリックすると、[見出しマップ]ペインが表示されます。このペインには、テキストボックス、グループ、またはサブレポートに付けたラベルが表示され、それらをクリックして、ビューア内のレポートの対応する領域に移動できます。
	パラメータ	サイドバーの**[パラメータ]**ボタンをクリックすると、[パラメータ]ペインが表示されます。FlexViewerでは、パラメータ付きのレポートを表示できます。レポートにパラメータが含まれていると、[パラメータ]ペインが自動的に表示されます。 [パラメータ]ペインでは、表示するデータをフィルタ処理する値を入力します。
	エクスポート	サイドバーの**[エクスポート]**ボタンをクリックすると、[エクスポート]ペインが表示されます。[エクスポート]ペインでは、レポートをエクスポートする前にさまざまなオプションを設定できます。使用可能な形式は、Adobe PDF、HTML、RTF、Open XML Word、Open XML Excel、TIFF、BMP、PNG、JPEG、およびGIFです。
	縦	レポートを縦向きに表示します。
	横	レポートを横向きに表示します。
	ページ設定	現在選択されているページの**[用紙の種類]**、**[向き]**、および**[マージン(インチ)]**を設定するための[ページ設定]ペインを表示します。

Copyright © GrapeCity, Inc. All rights reserved.

	ズームバー	サイドバーの[ズームバー]オプションをクリックすると、ビューアの下部にズームバーが表示されます。
	印刷	印刷オプションを指定するための印刷ダイアログを表示します。
ズームバーのオプション - このセクションは、レポートのズームイン/ズームアウトおよび拡大率(%)の各オプションで構成されます。		
	ズームイン/ズームアウト	ズームイン - レポートの拡大率を下げます。 ズームアウト - レポートの拡大率を上げます。
100 %	拡大率(%)	パーセント値によってズーム値を設定できます。

PDFViewer

PDFViewer(ベータ版)for ASP.NET MVC では、**ComponentOne MVC PDFViewer**テンプレートを利用して、Webブラウザで PDFファイルを表示できます。PDFViewerには、対話式の機能とユーザーフレンドリーなユーザーインタフェースを使用してPDF ファイルを表示するさまざまなオプションがあります。PDFViewerは、WebブラウザでPDFファイルを表示できる高速で柔軟な HTML5ベースのPDFビューアです。基本的にPDFViewerは、アドビに依存しないネイティブのASP.NET MVCコントロールです。 軽量で、アプリケーションと一緒に簡単に配布できます。



また、PDFViewerでは、ビューアのエクスポートオプションを使用して、PDFファイルを複数のファイル形式に簡単にエクスポートできます。PDFViewerコントロールでサポートされているエクスポート形式は、HTML、メタファイル、TIFF、BMP、PNG、JPEG、

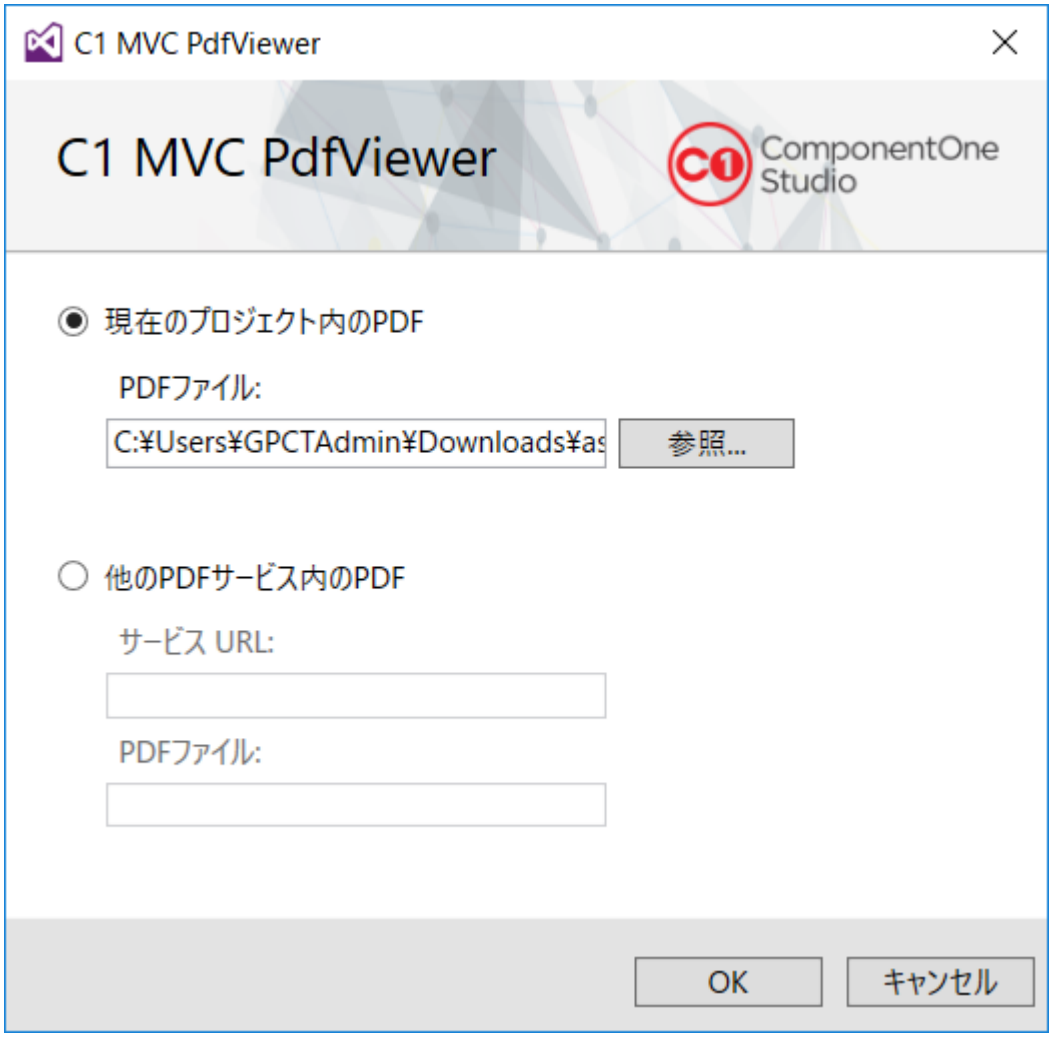
Copyright © GrapeCity, Inc. All rights reserved.

およびGIFです。

PdfViewer の使用

このセクションでは、**ComponentOne PdfViewer**テンプレートを使用して、MVCアプリケーション内にPDFファイルを表示する方法について説明します。ComponentOne PdfViewerテンプレートには、Webブラウザで PDFファイルをプレビューする2つのオプションがあります。**現在のプロジェクト内のPDF**を使用して、ローカルシステムに保存されているPDFファイルをプレビューするか、**他のPDFサービス内のPDF**オプションを使用して、WebApiサービスURLにホストされているPDFファイルを表示できます。

1. **[ビュー]**の下で**[PDFViewer]**フォルダを右クリックしてから、**[追加]→[新しい項目]**を選択して[新しい項目の追加]ダイアログを開きます。
2. **[インストール済み]→[テンプレート]**の下で、**[Visual C#]→[Web]→[ComponentOne PdfViewer ビュー ページ]**を選択して[ComponentOne MVC PdfViewer]ダイアログを開きます。
3. **[ComponentOne MVC PdfViewer]**ダイアログで、[現在のプロジェクト内のPDF]オプションを選択します。



4. PDFファイルフィールドで、**[参照]**をクリックして、システムにある**Portable Document File(.pdf)**ファイルを選択します。ここでは、サンプルのPDFドキュメントを使用しました。
5. **[OK]**をクリックして、**Index.cshtml**ビューページを作成します。

Index.cshtml

```
<head>
    <title>ComponentOne MVC PdfViewer</title>
    @Html.C1().Styles()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    @Html.C1().Scripts().FlexViewer()
</head>
<body>
@(Html.C1().PdfViewer().FilePath(@"~\Content\PdfFilesRoot\Selection.pdf"))
</body>
```

FlexViewerのスタイル設定

CSS classes for the FlexViewer control are as follows:

FlexViewer CSS Classes

Class	Description
Basic	
.wj-viewer	Styles the div host element of the FlexViewer control.
.wj-viewer .wj-viewer-outer	Styles the top and left toolbar containers.
.wj-viewer wj-viewer-container	Styles the viewer container of the FlexViewer control which includes the report display area and the left panel.
.wj-viewer .wj-viewer-loading	Styles the viewer when it is in loading state.
.wj-viewer .wj-viewer-loading.errormessage	Styles the div host element that represents the error message.
.wj-viewer .wj-magnifier	Styles the magnified area displayed while using the magnifier option from the toolbar.
Toolbar	
.wj-viewer .wj-toolbar	Styles the div host element of the toolbar.
.wj-viewer .wj-toolbarcontainer	Styles the container div that represents the toolbar elements.
.wj-viewer .wj-btn-group	Styles the div host element that represents the button group in the toolbar.
.wj-viewer .wj-btn-group > .wj-btn	Styles each button in the button group.
.wj-viewer .wj-toolbar-move.right	Styles the right move button on the right end of the toolbar.
.wj-viewer .wj-toolbar-move.left	Styles the left move button on the left end of the toolbar.
.wj-viewer .wj-toolbar-move .wj-glyph-left	Styles the left-arrow icon displayed on the left move button.
.wj-viewer .wj-toolbar-move .wj-glyph-right	Styles the right-arrow icon displayed on the right move button.
.wj-viewer .wj-toolbarwrapper	Styles the toolbar wrapper.
.wj-viewer .wj-toolbarwrapper.wj-btn-group>:not(.wj-control)	Styles the div element containing button(s) in the toolbar.
.wj-viewer .wj-toolbarwrapper.wj-btn-group > .wj-separator	Styles the separators placed between the buttons in the toolbar.
.wj-viewer .wj-toolbarwrapper .wj-pagenumber	Styles the area where the page number is displayed in the toolbar.
.wj-viewer .wj-toolbarwrapper .wj-pagecount	Styles the area where the page count is displayed in the toolbar.
.wj-viewer .wj-toolbarwrapper .wj-input-zoom	Styles the area where the user can input the zoom value.
.wj-viewer .wj-toolbarwrapper .slash	Styles the slash used to divide the current page and total pages in

Copyright © GrapeCity, Inc. All rights reserved.

	the toolbar.
Left Panel	
.wj-viewer wj-viewer-leftpanel	Styles the left panel of the FlexViewer control.
.wj-viewer .wj-nav	Styles the list of options displayed in the left panel of the viewer.
.wj-viewer .wj-tabcontent	Styles the tab pane displayed when an option is clicked/selected in the left panel.
.wj-viewer .wj-tabcontent .active	Styles the currently active tab pane of the left panel.
.wj-viewer .wj-tabcontent h3	Styles the tab pane header.
.wj-viewer .wj-tabcontent .wj-tabpane .wj-tabcontent-inner	Styles the inner section of tab pane.
.wj-viewer .wj-tabtitle-wrapper .wj-close	Styles the close button used for closing a tab pane.
Left Panel Options Tabs	
Page Thumbnails Tab Pane	
.wj-viewer .wj-thumbnaillist	Styles the Thumbnail List tab pane.
.wj-viewer .wj-thumbnail-item	Styles the thumbnail item.
.wj-viewer .wj-thumbnail-item .wj-pagethumbnail	Styles the thumbnail page.
.wj-viewer .wj-thumbnail-item.wj-state-selected	Styles the currently selected thumbnail item.
Document Map Tab Pane	
.wj-viewer .wj-outlines-tree	Styles the Document Map tab pane.
Parameters Tab Pane	
.wj-viewer .wj-parameterscontainer	Styles the container div that includes the report parameter section in the viewer.
.wj-viewer .wj-parametercontainer	Styles the container div that represents the report parameter in the viewer.
.wj-viewer .wj-parametercontainer .wj-parametercontrol	Styles the input control which is used to input the report parameter value.
.wj-parameterhead	Styles the header of parameter.
Search Tab Pane	
.wj-viewer .wj-tabcontent-wrapper.search-wrapper	Styles the wrapper layout for search tab pane.
.wj-viewer .wj-searchoption	Styles the search options available in the Search tab pane.
.wj-viewer .wj-searchoption input	Styles the input controls which is used to toggle the search options.
.wj-viewer .wj-searchoption label	Styles the label used to represent the text for the search options.
.wj-viewer .wj-searchresult	Styles the search results header in the Search tab pane.
.wj-viewer .wj-searchresultlist	Styles the search result list section.
.wj-search-page	Styles the page number of search result in search pane.

Copyright © GrapeCity, Inc. All rights reserved.

.wj-viewer .wj-searchresultlist .wj-search-item	Styles the search result item.
.wj-viewer .wj-searchresultlist .wj-search-item .wj-search-text	Styles the search text displayed in the search result section.
.wj-viewer .wj-searchbar	Styles the search bar which is displayed when you click on the search button from the mobile toolbar.
.wj-viewer .wj-control .wj-applybutton	Styles the apply button displayed in the Parameters tab pane.
.wj-viewer .wj-searchbox	Styles the search box that appears in the Search tab pane.
.wj-viewer .wj-searchcontainer	Styles the container div element that represents the search box.
.wj-viewer .mobile .wj-searchbox	Styles the search box that appears in the Search tab pane in mobile mode.
.wj-viewer .wj-searchcontainer .wj-btn-group	Styles the container div element that represents the search-pre and search-next buttons.
Export Tab Pane	
.wj-viewer .wj-tabcontent legend	Styles the legend entries displayed in the Export tab.
.wj-viewer .wj-tabtitle-wrapper .wj-exportcontainer label	Styles the label used for the Export tab.
.wj-viewer .wj-tabcontent-wrapper.wj-exportformats-wrapper	Styles the wrapper layout for export tab pane.
.wj-viewer .wj-tabcontent-wrapper.wj-exportformats-wrapper .wj-export-editor	Styles the export options section on the export tab pane.
.wj-viewer .wj-tabcontent-wrapper.wj-exportformats-wrapper .wj-export-editor .wj-exportformats-group	Styles the different export format groups on the export tab pane.
.wj-viewer .wj-tabcontent-wrapper.wj-exportformats-wrapper .wj-exportformats-footer	Styles the footer section in the export tab pane.
Splitter	
.wj-viewer .wj-viewer-splitter button	Styles the splitter panel of FlexViewer control.
.wj-viewer .wj-viewer-splitter span	Styles the glyph icon represented for the viewer splitter.
View Page	
.wj-viewer .wj-pageview	Styles the page view panel of the FlexViewer.
.wj-view-page	Styles the report page in the viewer.
.wj-viewer .wj-view-page .wj-loading	Styles the view page when it is in the loading state.
.wj-viewer .wj-view-page rect	Styles the SVG rect elements that are used to render different section of the report in the viewer.
.wj-viewer .wj-pageview-vscroller	Styles the vertical scroll bar.
Mini Toolbar	
.wj-viewer .wj-mini-toolbar	Styles the mini toolbar that appears at the bottom of the viewer. By default, it is hidden and appears when the viewer switches to full screen mode.

Copyright © GrapeCity, Inc. All rights reserved.

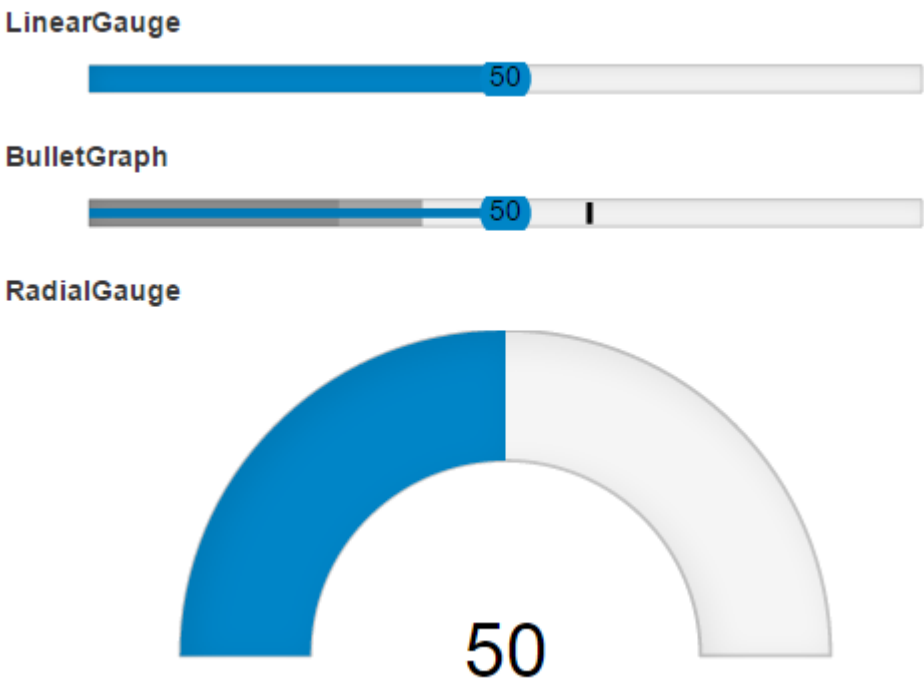
.wj-viewer .wj-mini-toolbar .slash	Styles the slash used to divide the current page and total pages in the mini toolbar.
.wj-viewer .wj-mini-toolbar .wj-pagenumber	Styles the area where the page count is displayed in the mini toolbar.
.wj-viewer .wj-mini-toolbar .wj-pagenumber input	Styles the area where user can input the page number in the mini toolbar.
.wj-viewer .wj-mini-toolbar .wj-toolbar-move	Styles the mini toolbar move buttons.
Mobile CSS Classes	
.wj-viewer .mobile.with-footer	Styles the footer section of the viewer in mobile view.
.wj-viewer .mobile.with-searchbar	Styles the search bar section of the viewer in mobile view.
.wj-viewer .mobile .wj-viewer-leftpanel	Styles the left panel of the FlexViewer control in mobile view.
.wj-viewer .mobile .wj-viewer-splitter	Styles the div host element of the viewer splitter in mobile view.
.wj-viewer .mobile .wj-toolbar	Styles the div host element of the toolbar in mobile view.
.wj-viewer .mobile .wj-toolbar .wj-pagenumber	Styles the area where the page number is displayed in the toolbar.
.wj-viewer .mobile .wj-toolbar .wj-pagecount	Styles the area where the page count is displayed in the toolbar.
.wj-viewer .mobile .wj-toolbar .wj-toolbar-move	Styles the toolbar move buttons.
.wj-viewer .mobile .wj-nav	Styles the navigation list displayed on the left panel of the viewer.
.wj-viewer .mobile .wj-viewer-tabsleft	Styles the tab on the left panel of the viewer.
.wj-viewer .mobile .wj-tabcontent	Styles the tab content section on the left panel.
.wj-viewer .mobile .wj-tabcontent .wj-tabpane .wj-tabtitle-wrapper	Styles the wrapper layout for the tab title.
.wj-viewer .mobile .wj-tabcontent .active	Styles the currently active tab in the left panel.
.wj-viewer .mobile .wj-btn-group > .wj-btn	Styles the buttons displayed in the viewer.
.wj-viewer .mobile .wj-viewer-footer	Styles the footer section of the mobile viewer.
.wj-viewer.full-screen	Styles the viewer in the full screen mode.
.wj-viewer.full-screen .wj-viewer-outer	Styles the outer boundaries of the FlexViewer control in full screen mode.
.wj-viewer.full-screen .wj-viewer-outer.mobile	Styles the outer boundaries of the FlexViewer control in full screen mode.
.wj-viewer.full-screen .wj-viewer-outer.mobile .wj-toolbar	Styles the div host element of the toolbar.
.wj-viewer.full-screen .wj-viewer-outer.mobile .wj-mini-toolbar	Styles the mini toolbar in mobile view mode.
.wj-viewer .wj-btn-group .wj-btn.wj-state-active svg.wj-svg-btn	Styles the buttons that are currently in active state.
.wj-viewer .wj-btn-group .wj-btn:hover svg.wj-svg-btn	Styles the buttons
.wj-viewer-menu.wj-btn-group-vertical .wj-listbox-item.wj-state-active .wj-btn .wj-svg-btn	Styles the currently active list box item in the viewer menu.

Copyright © GrapeCity, Inc. All rights reserved.

.wj-viewer-menu > .wj-listbox-item	Styles the list box item in the viewer menu.
.wj-viewer-menu > .wj-listbox-item a.wj-btn	Styles the button images in the viewer menu.
Viewer Footer	
.wj-viewer .wj-viewer-footer	Styles the footer section. This is only visible in the mobile mode.
.wj-viewer .wj-viewer-footer .wj-toolbar	Styles the toolbar available in the footer section.
.wj-viewer .wj-viewer-footer .wj-close	Styles the close button displayed in the toolbar.

Gauge

ゲージコントロールを使用すると、ダイナミックかつユニークな方法で情報を表示できます。これはまさに必要なグラフィカル表現です。ゲージには範囲も表示されるため、ユーザーは現在の値が大きいか、小さいか、それとも中間であるかを即座に判断することができます。したがって、単なるラベルよりゲージの方がわかりやすくて便利です。



主な特長

Gauges には主に次の便利な機能があります。

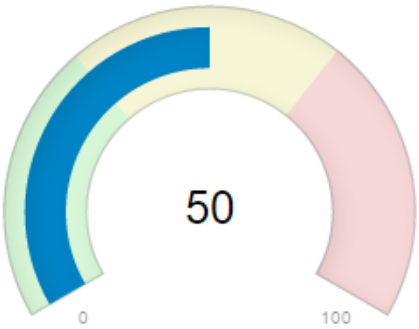
- **簡単なカスタマイズ**
  プロパティを変更してスタイルを再設定し、カスタム色や塗りつぶしなどのあるゲージを作成できます。
- **範囲**
  特定の範囲にある値が注目されるように、ゲージに色付き範囲を追加できます。単純なプロパティを使用して、開始点、終了点、および外観をカスタマイズします。
- **方向**
  LinearGauge および BulletGraph のポインタ方向をカスタマイズできます。
- **ポインタのカスタマイズ**

Copyright © GrapeCity, Inc. All rights reserved.

ポインタの色や太さなどをカスタマイズして、見栄えのよいゲージを作成できます。

ゲージのタイプ

ComponentOne Controls for ASP.NET MVC には、**直線型ゲージ**、**円形ゲージ**、**ブレットグラフ**の3種類のゲージがあります。

タイプ	画像	使用法
直線型ゲージ:直線型ゲージは、直線型のポインタを使用して、直線の目盛りに沿って値を表示します。		直線型ゲージは、長さや気温などの尺度としてデータを示す場合によく使用されます。
円形ゲージ: 円形ゲージは、曲線型のポインタを使用して、円形の目盛りに沿って値を表示します。目盛りは、`StartAngle` および **SweepAngle** プロパティで定義された角度で回転されます。		円形ゲージは、容量や速度などのデータを示す場合によく使用されます。
ブレットグラフ:ブレットグラフは、直線目盛り上に1つの値を表示します。**Good** および **Bad** プロパティを使用して、値が良好か不良かを一目で確認できるようにするための範囲とターゲット値が一緒に表示されます。		ブレットグラフは直線型ゲージの一種で、主に、年間売上高などのさまざまな単一値データを表示するダッシュボードで使用することを意図して設計されています。

Copyright © GrapeCity, Inc. All rights reserved.

クイックスタート

This section contains quick start topics for all the three types of gauges. For the detailed information, see the following topics:

- BulletGraph クイックスタート
- LinearGauge クイックスタート
- RadialGauge クイックスタート

BulletGraph クイックスタート

このセクションでは、MVC Web フォームに **BulletGraph** コントロールを追加し、コントロールに値を設定する方法について説明します。コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは3つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:新しいコントローラーおよびビューの追加**
- **手順 3:コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の BulletGraph を示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します(例:`Default1Controller`)。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー(例:`Default1Controller`)をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

モデルの追加

Copyright © GrapeCity, Inc. All rights reserved.

コントロールのデータをフェッチするにはモデルが必要です。詳細については、「**MVC の基本**」を参照してください。

1. **ソリューションエクスプローラー**で、**[モデル]**フォルダを右クリックし、**[追加]**→**[クラス]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、クラスの名前（例：`Class1.cs`）を設定します。
3. **[追加]**をクリックします。アプリケーションに新しいクラスが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**BulletGraph** コントロールを初期化し、次のプロパティを設定します。

    - ゲージの **Value** を **35**に設定します。**Value** プロパティは、ゲージの現在の値を示します。
    - **Min** および **Max** プロパティをそれぞれ0と 100 に設定します。**Min** および **Max** プロパティは、テージの範囲を指定します。1つのゲージに複数の範囲を追加できます。
    - **Target**、**Good**、および **Bad** プロパティをそれぞれ 70、60、40 に設定します。
    - ゲージの **isReadOnly** プロパティを **False**に設定します。**isReadOnly** は、ユーザーがゲージの **Value** をキーボードまたはマウスを使用して編集できるかどうかを指定します。

Razor copyCode

```
@(Html.C1().BulletGraph()
    .Min(0).Max(100)
    .Good(60)
    .Bad(40)
    .Value(50)
    .Target(70)
    .IsReadOnly(false).Step(1).Width(500)
    .IsAnimated(true)
)
```

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

    ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**Default1/index**）、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

LinearGauge クイックスタート

このセクションでは、MVC Web フォームに **LinearGauge** コントロールを追加し、コントロールに値を設定する方法について説明します。コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは3つの手順で構成されます。

- **手順 1：新しいコントローラーおよびビューの追加**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の LinearGauge を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します(例:`Default1Controller`)。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー(例:`Default1Controller`)をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

モデルの追加

コントロールのデータをフェッチするにはモデルが必要です。詳細については、「**MVC の基本**」を参照してください。

1. **ソリューションエクスプローラー**で、**[モデル]**フォルダを右クリックし、**[追加]**→**[クラス]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、クラスの名前(例:`Class1.cs`)を設定します。
3. **[追加]**をクリックします。アプリケーションに新しいクラスが追加されます。

手順 3:コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**LinearGauge** コントロールを初期化し、次のプロパティを設定します。

    - ゲージの **Value** を **35**に設定します。**Value** プロパティは、ゲージの現在の値を示します。
    - **Min** および **Max** プロパティをそれぞれ0と 100 に設定します。**Min** および **Max** プロパティは、テージの範囲を指定します。1つのゲージに複数の範囲を追加できます。
    - ゲージの **isReadOnly** プロパティを **False**に設定します。**isReadOnly** は、ユーザーがゲージの **Value** をキーボードまたはマウスを使用して編集できるかどうかを指定します。

Razor copyCode

```
@(Html.C1().LinearGauge()
    .Width(500)
    .Height(100)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .Value(35)
    .Thickness(0.1)
    .Min(0).Max(100)

.Direction(GaugeDirection.Right)
        // 範囲を作成してカスタマイズします
         .Ranges(items => items
            .Add(item => item.Min(0).Max(40).Color(System.Drawing.Color.Red))
            .Add(item =>
item.Min(40).Max(80).Color(System.Drawing.Color.Yellow))
            .Add(item =>
item.Min(80).Max(100).Color(System.Drawing.Color.Green))
        )
)
```

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]→[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例: http://localhost:1234/**Default1/index**)、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

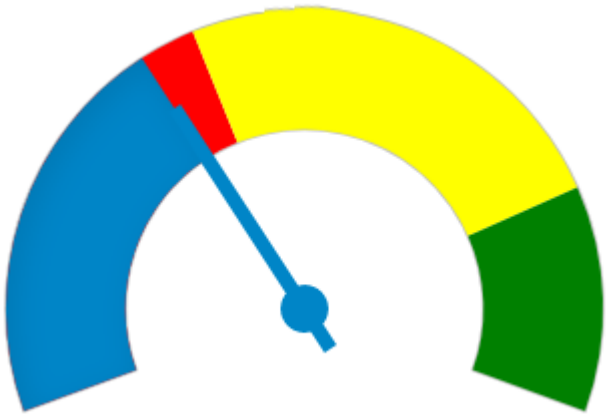
RadialGauge クイックスタート

このセクションでは、MVC Web フォームに **RadialGauge** コントロールを追加し、コントロールに値を設定する方法について説明します。コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは2つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:新しいコントローラーおよびビューの追加**
- **手順 3:コントロールの追加**
- **手順 3:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の RadialGauge を示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーション

Copyright © GrapeCity, Inc. All rights reserved.

の作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します(例:`Default1Controller`)。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー(例:`Default1Controller`)をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

モデルの追加

コントロールのデータをフェッチするにはモデルが必要です。詳細については、「**MVC の基本**」を参照してください。

1. **ソリューションエクスプローラー**で、**[モデル]**フォルダを右クリックし、**[追加]**→**[クラス]**を選択します。**[新しい項目の追加]**ダイアログボックスが表示されます。
2. **[新しい項目の追加]**ダイアログで、クラスの名前(例:`Class1.cs`)を設定します。
3. **[追加]**をクリックします。アプリケーションに新しいクラスが追加されます。

手順 3:コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**RadialGauge** コントロールを初期化し、次のプロパティを設定します。

    - ゲージの **Value** を **35**に設定します。**Value** プロパティは、ゲージの現在の値を示します。
    - **Min** および **Max** プロパティをそれぞれ0と 100 に設定します。**Min** および **Max** プロパティは、テージの範囲を指定します。1つのゲージに複数の範囲を追加できます。
    - ゲージの **isReadOnly** プロパティを **False**に設定します。**isReadOnly** は、ユーザーがゲージの **Value** をキーボードまたはマウスを使用して編集できるかどうかを指定します。
    - **StartAngle** プロパティを **-20**、**SweepAngle** プロパティを **220** に設定します。**StartAngle** プロパティは RadialGauge の開始角度、**SweepAngle** プロパティは RadialGauge の弧の長さを表す角度を指定します。
    - **AutoScale**プロパティを True に設定します。**AutoScale** プロパティは、中の要素が収まるように、円形ゲージを自動的に拡大縮小します。

Razor	copyCode

```
@(Html.C1().RadialGauge()
    .Value(35)
    .Width(500)
    .Height(200)
    .Min(0).Max(100)
    .StartAngle(-20)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .SweepAngle(220)
    .ShowText(ShowText.None)
    .AutoScale(true)
         .Ranges(items => items
            .Add(item => item.Min(0).Max(40).Color(System.Drawing.Color.Red))
            .Add(item =>
item.Min(40).Max(80).Color(System.Drawing.Color.Yellow))
            .Add(item =>
item.Min(80).Max(100).Color(System.Drawing.Color.Green))
               )
 )
```

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例:http://localhost:1234/**Default1/index**)、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

Gaugeの使用

This section comprises all the features and auxiliary functionalities offered by Gauges.

外観のカスタマイズ
: Learn how to customize the gauge in ASP.NET MVC Edition using Razor.

方向
: Learn how to change the direction of the gauge pointer.

値の表示
: Learn how to select the values that you want to display as text by the gauge.

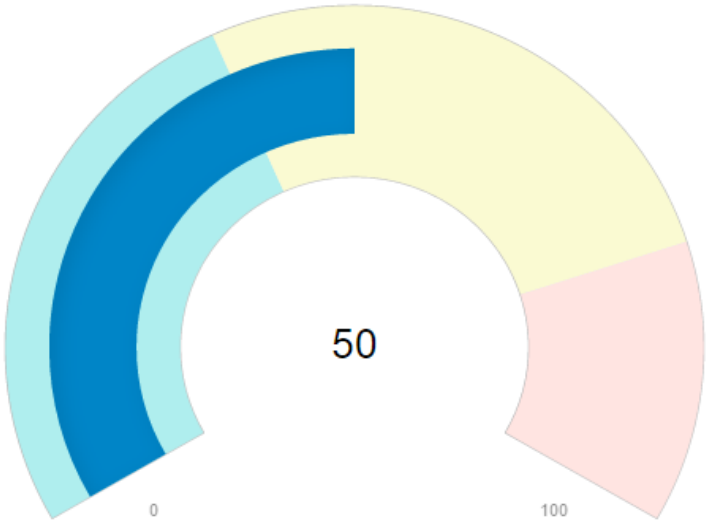
範囲
: Learn how to add ranges to a gauge and set their properties.

外観のカスタマイズ

ゲージとその要素に色や境界線を追加して、見栄えをよくすることができます。次のコード例は、さまざまなスタイル設定プロパティを使用して、RadialGauge をカスタマイズする方法を示します。

次の図は、外観をカスタマイズした **RadialGauge** を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、Razor を使用する ASP.NET MVC でゲージをカスタマイズする方法を示します。この例では、「RadialGauge クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor copyCode

```
    // 範囲を作成してカスタマイズします
.Ranges(items => items
      .Add(item => item.Min(0).Max(40).Color(System.Drawing.Color.Red))
      .Add(item => item.Min(40).Max(80).Color(System.Drawing.Color.Yellow))
      .Add(item => item.Min(80).Max(100).Color(System.Drawing.Color.Green))
 )
```

Range

方向

LinearGauge および BulletGraph の **GaugeDirection** プロパティを使用して、ポインタの移動方向を変更できます。

GaugeDirection プロパティには4つのオプションがあります。

- **Up**:ポインタはゲージの下から上に向かって移動します。
- **Right**:ポインタはゲージの左から右に向かって移動します。これが**デフォルト**値です。
- **Down**:ポインタはゲージの上から下に向かって移動します。
- **Left**:ポインタはゲージの右から左に向かって移動します。

GaugeDirection プロパティはポインタの方向を変更しますが、ゲージの向きは変更しないため、それに応じてゲージの高さと幅を設定する必要があります。

次の図は、**GaugeDirection** プロパティが Down に設定され、それに応じて高さと幅が調整された後の LinearGauge を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、ASP.NET MVC でこのプロパティを設定する方法を示します。この例では、「LinearGauge クイックスタート」セクションで作成したサンプルを使用しています。

Razor copyCode

```
.Direction(GaugeDirection.Down)
.Height(400)
.Width(120)
```

値の表示

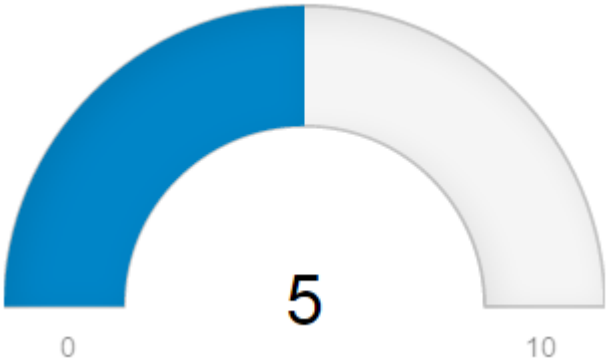
ゲージの最小値/最大値または現在値がわかるように、ゲージにいくつかの値を表示できます。**ShowText** プロパティを使用すると、ゲージにテキストとして表示する値を選択できます。

ShowText プロパティには、4つの有効な値があります。

- **Value**:ゲージの現在値だけを表示します。
- **MinMax**:ゲージの最小値/最大値だけを表示します。
- **All**:ゲージの最小値、最大値、および現在値を表示します。
- **None**:ゲージコントロールに何もテキストを表示しません。これが**デフォルト**値です。

次の図は、`ShowText` プロパティを All に設定した **RadialGauge** を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、Razor を使用する ASP.NET MVC でこのプロパティを設定する方法を示します。この例では、「RadialGauge クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor copyCode

```
@(Html.C1().RadialGauge()
    .Min(0).Max(10).Value(5)
    .Width(300).Height(180)
.ShowText(ShowText.All)
```

範囲

1つのゲージに複数の範囲を追加できます。各範囲は、ユーザーがゲージの値の状態を判断するために役立つ領域または状態を表します。どの範囲にも、ゲージ内での範囲の位置を指定する **Min** および **Max** プロパティと、範囲の外観を定義する **Color** および **Thickness** プロパティがあります。

次のコード例は、ゲージにいくつかの範囲を追加し、それらのプロパティを設定する方法を示します。

コードの場合

次のコード例は、タイプ `Ranges` の新しいインスタンスを作成し、それらのプロパティを設定し、新しく作成した範囲を LinearGauge（または RadialGauge/BulletGraph）に追加する方法を示します。

この例では、「LinearGauge クイックスタート」セクションで作成したサンプルを使用しています。

HTML copyCode

```
   // 範囲を作成してカスタマイズします
.Ranges(items => items
     .Add(item => item.Min(0).Max(40).Color(System.Drawing.Color.Red))
     .Add(item => item.Min(40).Max(80).Color(System.Drawing.Color.Yellow))
     .Add(item => item.Min(80).Max(100).Color(System.Drawing.Color.Green))
 )
```

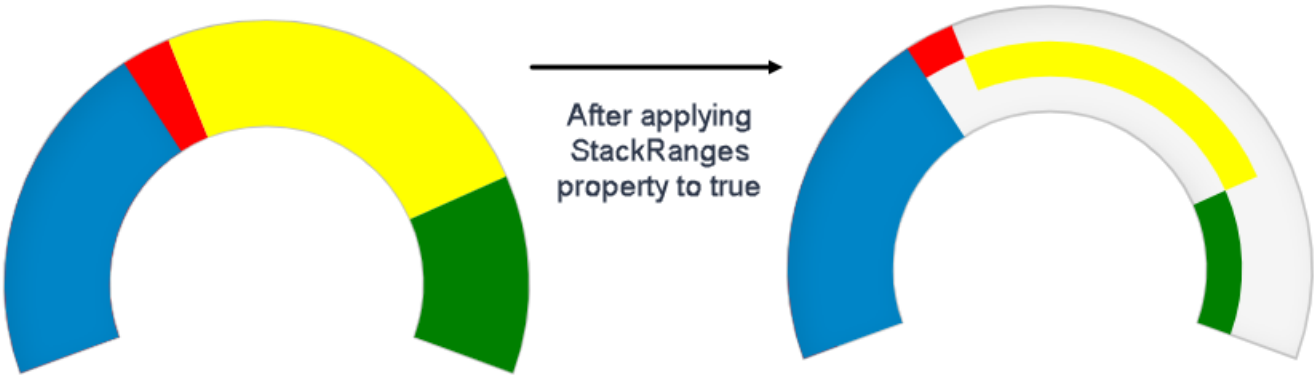
Stack Ranges

Apart from setting the min and max properties that specify the range's position on the gauge, you can also stack the ranges contained in the ranges collection within the Gauge control using the **StackRanges** property. By default, the StackRanges property is set to false, and the ranges in the collection are displayed with the same thickness in the control. When you set the StackRanges property to **true** it causes the ranges to become narrower, and to be displayed

Copyright © GrapeCity, Inc. All rights reserved.

side-by-side. This property is available for all the three types of gauges; Linear Gauge, Radial Gauge, and Bullet Gauge.

The following image shows stack ranges in the Radial Gauge control.



The following code examples demonstrate how to stack ranges together in the Radial Gauge control.

Razor

```
.StackRanges(true)
```

スタイル設定とCSS

CSS classes for the Gauge control are as follows:

Class	Description
.wj-gauge	Styles the div host element of the Gauge control.
.wj-gauge .wj-pointer	Styles the gauge pointer.
.wj-gauge .wj-ticks	Styles the tick marks, which are displayed when the **ShowTicks** property is set to **true**.
.wj-gauge .wj-min	Styles the min value text, which is displayed when you set the **ShowText** property to **MinMax**.
.wj-gauge .wj-max	Styles the max value text, which is displayed when you set the **ShowText** property to **MinMax**.
.wj-gauge text	Styles the text displayed on the Gauge control, which is displayed when you set the **ShowText** property to **Value**.
.wj-gauge svg	Styles the div host element of the SVG.
.wj-lineargauge	Styles the div host element of the linear gauge.
.wj-radialgauge .wj-value	Styles the text displayed on the Radial gauge control, which is displayed when you set the **ShowText** property to **Value**.

Gaugeのスタイル設定

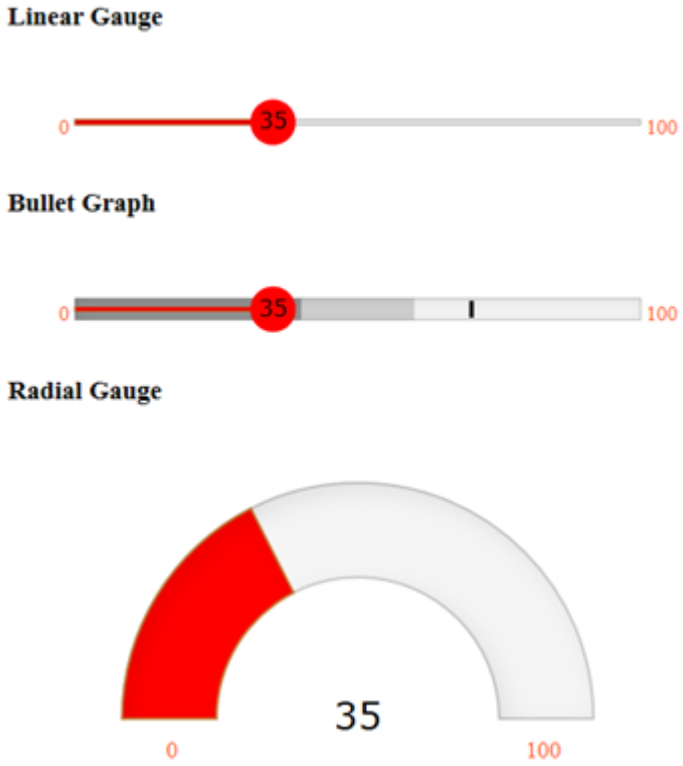
The Gauge control allows to customize its appearance using CSS. You can create your own custom CSS and then apply it to the Gauge control using CssClass property.

The example uses CssClass property to apply custom cascading styling sheets in the Gauge control. In this example, we customize the default CSS rules to make the following customizations to the Gauge control.

Copyright © GrapeCity, Inc. All rights reserved.

1. Change the font-size, opacity and color of the gauges.
2. Change the font-size, font-weight, font-family and color of the gauges.
3. Change the color and stroke of the gauge pointer.
4. Set the minimum and maximum values of the displayed text.

The following image shows how the Gauge control appears after using a custom CSS class.



Add Custom Stylesheet

Create a new ASP.NET MVC application. Once you have created the application, a Content folder is created in the Solution Explorer after adding the view to the application. To add a custom style sheet in your application, follow these steps:

1. In the **Solution Explorer**, right-click the **Content** folder.
2. From the context menu, select **Add | Style** Sheet. The Specify Name for Item dialog appears.
3. Set name of the style sheet (for example: app.css) and click **OK**.
4. Replace the default code of app.css file with the code given below.

app.css copyCode

```
<style>
    .gauge .wj-gauge .wj-min, .gauge .wj-gauge .wj-max {
        font-size: 15px;
        opacity: .9;
        fill: orangered;
    }

    .radialgauge .wj-gauge .wj-min, .radialgauge .wj-gauge .wj-max {
        font-size: 10px;
        opacity: .9;
        fill: orangered;
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .wj-pointer path {
        fill: #ff0000;
        stroke: #cd853f;
    }

    .wj-gauge .wj-value {
        font-size: 100%;
        font-weight: 500;
        font-family: Verdana, sans-serif;
        fill: black !important;
    }
</style>
<br /><br />
```

5. In the Views folder, open a view, say Styling.cshtml, to add the following code for applying custom CSS.

Styling.cshtml copyCode

```
<div class="gauge" style="min-width:500px; width:33.33%">
    <h3>Linear Gauge</h3>
    <br /><br />
    @(Html.C1().LinearGauge().Id("secondGauge")
          .HasShadow(false).Value(35).Min(0).Max(100).Step(5).IsReadOnly(false)
          .ShowText(ShowText.All).Face(face => face.Thickness(0.25))
          .Pointer(pointer => pointer.Thickness(0.25)
          .Color(System.Drawing.Color.Red)))
</div>
<br />
<div class="gauge">
    <h3>Bullet Graph</h3>
    <br /><br />
    @(Html.C1().BulletGraph()
          .Min(0).Max(100).Good(60).Bad(40)
          .Value(35).Target(70).ShowText(ShowText.All)
          .IsReadOnly(false).Step(1).Width(500)
          .IsAnimated(true)
          .Pointer(pointer =>
pointer.Thickness(0.25).Color(System.Drawing.Color.Red)))
</div>
<br />
<h3>Radial Gauge</h3>
<br /><br />
<div class="radialgauge">
    @(Html.C1().RadialGauge()
          .Value(35)
          .Width(500)
          .Height(200)
          .Min(0).Max(100)
          .ShowText(ShowText.All)
          .AutoScale(true))
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

Input

ComponentOne ASP.NET MVC は、Web UI でユーザーからデータを収集するためのさまざまな入力コントロールを提供します。これらのコントロールを使用すると、収集するデータのタイプに応じて、数値、マスク付き、パーセント、日付、通貨、およびテキストによる入力を作成できます。これらの入力コントロールは高度にカスタマイズ可能であり、データ依存型のアプリケーションの作成に役立ちます。

次の表に、ASP.NET MVC でサポートされている**入力**コントロールをリストします。

コントロール	説明
AutoComplete	サーバーから動的に取得されるリストから値を選択します。AutoComplete コントロールは、ユーザーがキー入力すると、サーバーからオプションを取得します。
Calendar	日、月、年の間を移動しながら、月間カレンダーから日付を選択します。Min および Max プロパティを使用して、選択可能な日付を制限します。
ColorPicker	色チャンネル（色相、彩度、明度、アルファ）を調整するためのパネルをクリックして、色を選択します。
ComboBox	オートコンプリート機能を備えたリストから値を選択します。displayMemberPath および displayValuePath プロパティを使用して、複雑なオブジェクト（キー/値のペア）を含むテーブルから選択を行います。
InputColor	HTML でサポートされる色文字列を入力して色を選択するか、ドロップダウンして表示される ColorPicker コントロールから色を選択します。このコントロールの Value プロパティは、現在選択されている色を取得または設定します。
InputDate	日付値を編集するか、ドロップダウンカレンダーから日付を選択します。min および max プロパティを使用して、有効な日付の範囲を制限します。
InputMask	無効な入力を防止したりリテラルをスキップするためのマスクを使用して、文字列を編集します。マスクプロパティを使用して、入力の書式を指定します。
InputNumber	数値を編集するか、スピナーボタンを使用して現在の値を増減します。Min および Max プロパティを使用して、有効な数値の範囲を制限します。
InputTime	時間値を編集するか、ドロップダウンリストから時間を選択します。min および max プロパティを使用して、有効な時間の範囲を制限します。
ListBox	プレーンテキストまたは HTML を含む値のリストを表示し、ユーザーが値を選択できるようにします。このコントロールは、ComboBox、AutoComplete、TimePicker、および Menu コントロールのドロップダウン部分に表示されます。
Menu	1つのテキスト要素とドロップダウンリストを表示します。ドロップダウンリストには、ユーザーがクリックすることで呼び出すことができるコマンドが含まれます。
MultiSelect	ドロップダウンリストにてカスタム項目のリストまたは文字列を表示して、ユーザーがリストから複数の項目を選択できます。

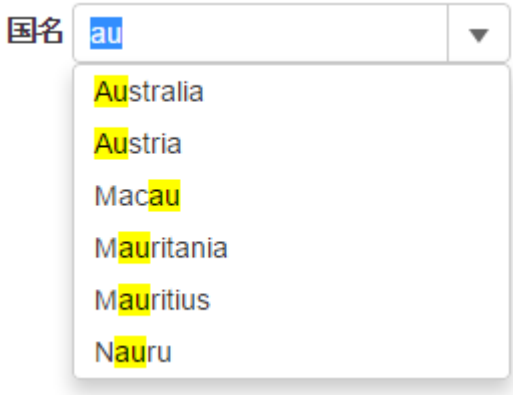
Copyright © GrapeCity, Inc. All rights reserved.

Popup	任意のコンテンツの小さなオーバーレイをポップオーバーまたはモーダルダイアログとしてアプリケーションに表示でき、このファイルを使用してエンドーユーザーからご要望の情報を取得します。
MultiAutoComplete	サーバーから動的に取得されるリストから値を選択します。カスタムオブジェクトまたは簡単な文字列を含むリストから項目を選択することができます。

AutoComplete

AutoComplete コントロールでは、キー入力しながら項目リストをフィルタ処理し、コントロールのドロップダウンリストから値を直接選択することができます。ユーザーは、事前に項目が挿入されたリストから特定のデータフィールドを検索したりフィルタ処理しやすくなります。

次の AutoComplete コントロールは、itemsSource として文字列配列を使用します。たとえば、「au」と入力すると、このコントロールは、国名に文字列「au」を含む国を検索結果に表示します。



主な特長

AutoComplete には主に次の便利な機能があります。

- **CSSMatch**
  入力した検索項目と一致する部分をコンテンツ内で強調表示するには、**CSSMatch** プロパティを使用します。

- **ItemsSourceFunction**
  ユーザーのキー入力に伴って動的に検索項目を提供します。
- **MaxItems**
  ドロップダウンに表示する項目の最大数を定義するには、この値を設定します。

- **MinLength**
  オートコンプリート候補の検索に必要な入力の最小長さを設定するには、この値を設定します。

- **Delay**
  キーストロークから検索を実行するまでの遅延時間（ミリ秒）を指定するには、この値を設定します。

クイックスタート

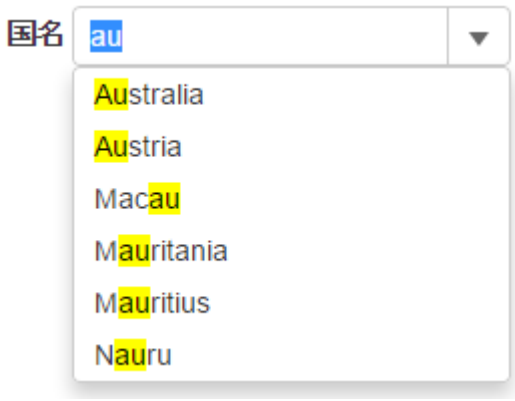
このセクションでは、MVC Web アプリケーションに **AutoComplete** コントロールを追加し、そこにデータを追加する方法について説明します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

Copyright © GrapeCity, Inc. All rights reserved.

このトピックは4つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:AutoComplete のデータソースの作成**
- **手順 3:AutoComplete コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の AutoComplete を示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:AutoComplete のデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`Countries.cs`)。
2. 次のコードを新しいモデルに追加して、AutoComplete コントロールのデータソースになるクラスを定義します。

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MVCFlexGrid.Models
{
    public class Countries
    {
        public static List<string> GetCountries()
        {
            return new List<string>
            {
                "Afghanistan", "Albania", "Algeria", "American Samoa",
"Andorra", "Angola", "Anguilla", "Antigua", "Argentina", "Armenia",
        "Aruba", "Australia", "Austria", "Azerbaijan", "Bahamas", "Bahrain",
"Bangladesh", "Barbados", "Belarus", "Belgium", "Belize",
        "Benin", "Bermuda", "Bhutan", "Bolivia", "Bonaire", "Bosnia",
"Botswana", "Brazil", "Brunei", "Bulgaria", "Burkina Faso", "Burundi",
        "Cambodia", "Cameroon", "Canada", "Canary Islands", "Cape Verde",
"Cayman Islands", "Central African Republic", "Chad", "Channel Islands",
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        "Chile", "China", "Christmas Island", "Cocos Island", "Colombia",
"Comoros", "Congo", "Cook Islands", "Costa Rica", "Cote D'Ivoire",
        "Croatia", "Cuba", "Curacao", "Cyprus", "Czech Republic", "Denmark",
"Djibouti", "Dominica", "Dominican Republic", "East Timor", "Ecuador",
        "Egypt", "El Salvador", "Equatorial Guinea", "Eritrea", "Estonia",
"Ethiopia", "Falkland Islands", "Faroe Islands", "Fiji", "Finland",
        "France", "French Guiana", "French Polynesia", "French Southern Ter",
"Gabon", "Gambia", "Georgia", "Germany", "Ghana", "Gibraltar",
        "Great Britain", "Greece", "Greenland", "Grenada", "Guadeloupe", "Guam",
"Guatemala", "Guinea", "Guyana", "Haiti", "Honduras",
        "Hong Kong", "Hungary", "Iceland", "India", "Indonesia", "Iran", "Iraq",
"Ireland", "Isle of Man", "Israel", "Italy", "Jamaica", "Japan",
        "Jordan", "Kazakhstan", "Kenya", "Kiribati", "Korea North", "Korea
South", "Kuwait", "Kyrgyzstan", "Laos", "Latvia", "Lebanon", "Lesotho",
        "Liberia", "Libya", "Liechtenstein", "Lithuania", "Luxembourg", "Macau",
"Macedonia", "Madagascar", "Malaysia", "Malawi", "Maldives",
        "Mali", "Malta", "Marshall Islands", "Martinique", "Mauritania",
"Mauritius", "Mayotte", "Mexico", "Midway Islands", "Moldova", "Monaco",
        "Mongolia", "Montserrat", "Morocco", "Mozambique", "Myanmar", "Nambia",
"Nauru", "Nepal", "Netherland Antilles", "Netherlands", "Nevis",
        "New Caledonia", "New Zealand", "Nicaragua", "Niger", "Nigeria", "Niue",
"Norfolk Island", "Norway", "Oman", "Pakistan", "Palau Island",
        "Palestine", "Panama", "Papua New Guinea", "Paraguay", "Peru",
"Philippines", "Pitcairn Island", "Poland", "Portugal", "Puerto Rico",
        "Qatar", "Republic of Montenegro", "Republic of Serbia", "Reunion",
"Romania", "Russia", "Rwanda", "St Barthelemy", "St Eustatius",
        "St Helena", "St Kitts-Nevis", "St Lucia", "St Maarten", "Saipan",
"Samoa", "Samoa American", "San Marino", "Saudi Arabia", "Scotland",
        "Senegal", "Serbia", "Seychelles", "Sierra Leone", "Singapore",
"Slovakia", "Slovenia", "Solomon Islands", "Somalia", "South Africa",
        "Spain", "Sri Lanka", "Sudan", "Suriname", "Swaziland", "Sweden",
"Switzerland", "Syria", "Tahiti", "Taiwan", "Tajikistan", "Tanzania",
        "Thailand", "Togo", "Tokelau", "Tonga", "Trinidad Tobago", "Tunisia",
"Turkey", "Turkmenistan", "Turks & Caicos Is", "Tuvalu", "Uganda",
        "Ukraine", "United Arab Emirates", "United Kingdom", "United States of
America", "Uruguay", "Uzbekistan", "Vanuatu", "Vatican City State",
        "Venezuela", "Vietnam", "Virgin Islands (British)", "Virgin Islands
(USA)", "Wake Island", "Yemen", "Zaire", "Zambia", "Zimbabwe"
            };
        }
    }
}
```

先頭に戻る

手順 3：AutoComplete コントロールの追加

AutoComplete コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。

Copyright © GrapeCity, Inc. All rights reserved.

3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：IndexController）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次に示すように MVC 参照を追加します。

サンプルのタイトル

```
using C1.Web.Mvc.Serializition;
```

5. メソッド Index() を次のメソッドに置き換えます。

C# copyCode

```
public ActionResult Index()
{
    ViewBag.Countries = MVCFlexGrid_JP.Models.Countries.GetCountries();
    return View();
}
```

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー QuickStartControllerをダブルクリックして開きます。
2. メソッド QuickStart() 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **QuickStart** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。
6. FlexChart コントロールをビュー QuickStart で次のようにインスタンス化します。

Razor copyCode

```
@{
    List<string> countries = ViewBag.Countries;
}

<style>
    .highlight {
        background-color: #ff0;
        color: #000;
    }
</style>
<div>
    <br />
    <label>国名</label>
    @(Html.C1().AutoComplete().Bind(countries).CssMatch("highlight")
    .Delay(2000)
    )


</div>
```

先頭に戻る

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

Copyright © GrapeCity, Inc. All rights reserved.

ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**QuickStart/QuickStart**）、ビューを表示します。

先頭に戻る

AutoCompleteの使用

This section comprises all the features and auxiliary functionalities offered by AutoComplete.

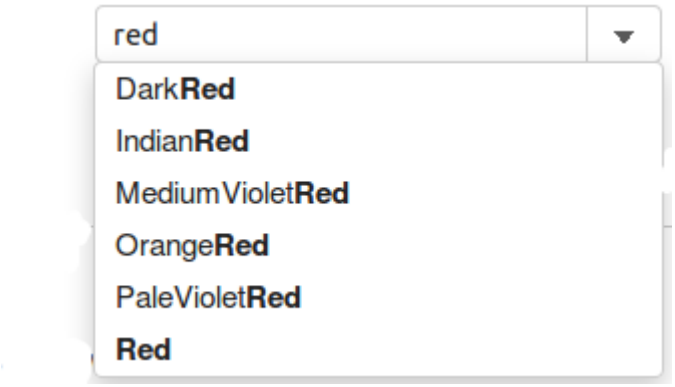
カスタムデータソースを使用する AutoComplete
: Learn how to enable complex type binding in AutoComplete.

カスタムアクション
: Learn how to use custom action in AutoComplete.

カスタムデータソースを使用する AutoComplete

SelectedValuePath プロパティと **DisplayMemberPath** プロパティを使用すると、複合型リストに AutoComplete を連結することができます。ここでは、名前空間 System.Drawing の SystemColors クラスに AutoComplete コントロールを連結します。このコントロールは、このクラスにあるすべてのメンバのリストを提供します。

次の図は、**SelectedValuePath** プロパティと **DisplayMemberPath** プロパティを設定した後の AutoComplete を示しています。この例では、「クイックスタート」でアプリケーションに追加した Countries.cs モデルを使用します。



次のコード例は、AutoComplete で複合型の連結を有効にする方法を示します。

コードの場合

AutoCompleteController.cs

Razor copyCode

```
public ActionResult ComplexType()
{
    var list = GetSystemColors();
    return View(list);
}

private static NamedColor[] GetSystemColors()
{
    return Enum.GetValues(typeof(KnownColor))
        .Cast<KnownColor>()
        .Select(c => new NamedColor
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                Name = c.ToString(),
                Value = "#" +
Color.FromKnownColor(c).ToArgb().ToString("X8").Substring(2)
            })
            .ToArray();
    }
}
```

AutoComplete.cshtml

Razor copyCode

```
@model IEnumerable<MVCFlexGrid_JP.Models.NamedColor>

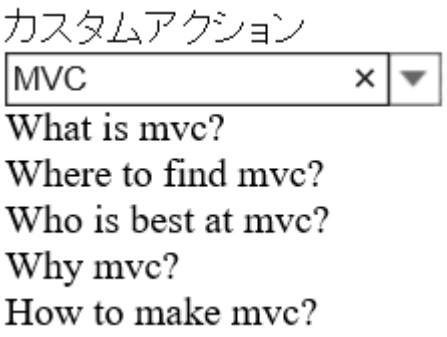
<div>
    <label>複合型リスト</label>

    @(Html.C1().AutoComplete()
        .Bind(Model)
        .DisplayMemberPath("Name")
        .SelectedValuePath("Value")
        .ItemsSourceAction(Url.Action("TypesInMscorlib"))
    )
</div>
```

カスタムアクション

カスタムアクションも、MVCアプリケーションで**AutoComplete**を使用する場合に重要な機能の1つです。AutoCompleteコントロールを使用して実行するカスタムアクションを指定できます。たとえば、以下のコードは、カスタムアクションを使用して、ユーザーが文字を入力した後にデータをフィルタ処理します。

次の画像は、カスタムアクションを使用して、AutoCompleteがどのように表示されるかを示します。



次のコード例は、AutoCompleteでカスタムアクションを使用する方法を示します。

コードの場合

AutoCompleteController.cs

Razor

```
public ActionResult Index()
        {
            return View();
        }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public ActionResult Heuristic(string query, int max)
        {
            var prefix = new[] { "What is ", "Where to find ", "Who is best at ",
"Why ", "How to make " };
            return this.C1Json(prefix.Select(f => f + query + "?").ToList(),
                        behavior: JsonRequestBehavior.AllowGet);
        }
```

AutoComplete.cshtml

Razor

```
<div>
    <label>カスタムアクション</label>
    @(Html.C1().AutoComplete()
        .ItemsSourceAction(Url.Action("Heuristic"))
    )
</div>
```

Calendar

Calendar コントロールには月間カレンダーを表示され、ユーザーはここから日付を選択することができます。日、月、年の間を移動することができます。**Min** および **Max** プロパティを使用して、選択可能な日付を制限します。Calendar コントロールは、DatePicker コントロールのドロップダウン部分に表示されます。



主な特長

Calendar には主に次の便利な機能があります。

- **FirstDayofWeek**
  この値には、カレンダーの最初の列に表示する曜日を設定します。
- **Value**
  ユーザーが現在選択している日付です。
- **Max**
  この値には、カレンダーでユーザーが選択できる範囲の最遅日を設定します。
- **Min**

Copyright © GrapeCity, Inc. All rights reserved.

この値には、カレンダーでユーザーが選択できる範囲の最早日を設定します。

- **MonthView**
  月間カレンダーを表示するか、年間カレンダーを表示するかを示します。

クイックスタート

このトピックでは、アプリケーションに Calendar コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:新しいコントローラーおよびビューの追加**
- **手順 3:Calendar コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **Calendar** を示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
   1. コントローラーの名前を設定します(例:`Default1Controller`)。
   2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
   3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー(例:`Default1Controller`)をダブルクリックして開きます。

Copyright © GrapeCity, Inc. All rights reserved.

2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**Calendar** コントロールを初期化します。

Razor copyCode

```
@using MVCFlexGrid_JP.Models
@{
    //変数を作成し、min, max および現在の値を設定します
    var today = DateTime.Now.Date;
    var minDate = new DateTime(today.Year, 1, 1);
    var maxDate = new DateTime(today.Year, 12, 31);
}
<div>
    //Calendarコントロールを初期化します
    @(Html.C1().Calendar()
            //コントロールをカスタマイズします
        .Value(today).Min(minDate).Max(maxDate).MonthView(true)
        .Width(300)

    )

</div>
```

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**Default1/index**）、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

Calendarの使用

This section comprises all the features and auxiliary functionalities offered by Calendar.

日付検証
: Learn how to add ItemValidator in the Calendar control.

日付検証

ComponentOne MVC EditionのCalendarコントロールに組み込まれたバリデータ**ItemValidator**を使用して、ユーザーがカレンダーコントロールから選択できるデータの型や値を制御できます。これを行うには、validation関数を作成して、ユーザーによる日付選択を許可または制限するItemValidatorに割り当てるだけです。

この例では、ユーザーがカレンダーコントロールで平日を選択でき、土曜と日曜は選択できないようにItemValidator関数が作

Copyright © GrapeCity, Inc. All rights reserved.

成されます。この種のvalidationは、重要な作業日、ジムに通う日、ダイエットプランなどを計画する際に便利です。

次の図は、ItemValidatorを適用した後の**Calendar**コントロールを示しています。



次のコード例は、ItemValidatorをCalendarコントロールに追加する方法を示します。

コードの場合

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します(例:`Default1Controller`)。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

Calendar.cshtml

Razor copyCode

```
@{
    var today = DateTime.Now.Date;
    var minDate = new DateTime(today.Year, 1, 1);
    var maxDate = new DateTime(today.Year, 12, 31);
}

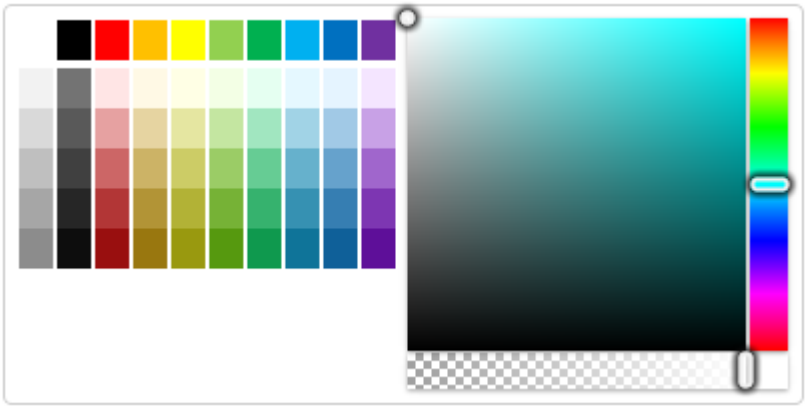
<script>
    function itemValidator(date) {
        var weekday = date.getDay();
        return weekday != 0 && weekday != 6;
    }
</script>

<div>
    @(Html.C1().Calendar().Value(today).Min(minDate).Max(maxDate).Width(300)
    .ItemValidator("itemValidator"))
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

ColorPicker

ColorPicker コントロールを使用すると、色チャンネル（色相、彩度、明度、アルファ）を調整するためのパネルをクリックして、色を選択することができます。このコントロールは、InputColor コントロールのドロップダウンとして使用されます。



主な特長

ColorPicker には主に次の便利な機能があります。

- **Value**
  ユーザーが現在選択している色を表します。
- **ShowAlphaChannel**
  ユーザーがコントロールの色のアルファチャンネル（透過性）を編集できるようにします。デフォルト値は true です。
- **ShowColorString**
  現在の色の文字列表現を示します。
- **Palette**
  パレット内の色を保持する配列を表します。Palette には、10 個の文字列の配列で表される 10 個の色が含まれます。

クイックスタート

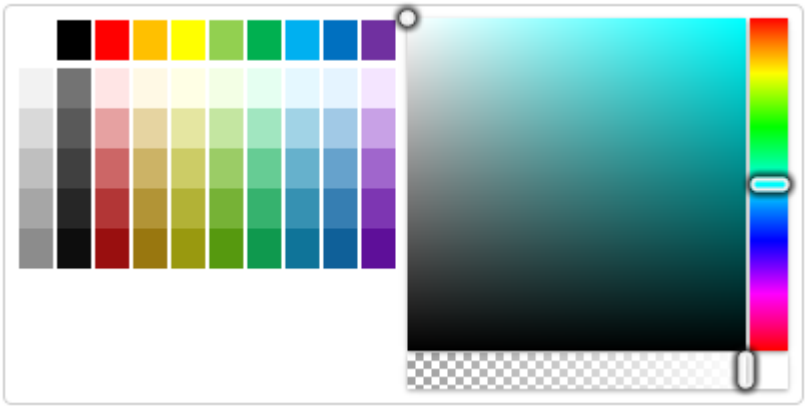
このトピックでは、アプリケーションに **ColorPicker** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：ColorPicker コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **ColorPicker** を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します(例:`Default1Controller`)。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー(例:`Default1Controller`)をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3:コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**ColorPicker** コントロールを初期化します。

Razor copyCode

```
@using System.Drawing
@using MVCFlexGrid_JP.Models
<div>
    @(Html.C1().ColorPicker()
    .Width(400)
    )
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

手順 4:プロジェクトのビルドおよび実行

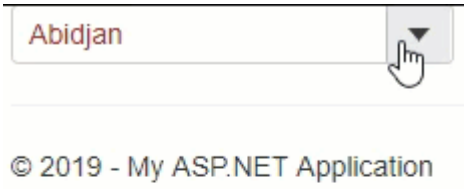
1. **[ビルド]→[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例: http://localhost:1234/**Default1/index**)、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

ComboBox

ComboBox コントロールを使用すると、リストから項目を選択することができます。このコントロールは、ユーザーのキー入力に伴って自動的にエントリを補完し、利用可能な項目をドロップダウンリストに表示することができます。ComboBox コントロールをモデルに連結することで、項目リストに値を設定することができます。



主な特長

ComboBox には主に次の便利な機能があります。

- **SelectedItem**
  ドロップダウンリストで現在選択されている項目を表します。

- **SelectedIndex**
  ドロップダウンリストの **SelectedItem** のインデックスを表します。

- **SelectedValue**
  SelectedValuePath を使用して取得される**SelectedItem**の値を表します。

- **SelectedValuePath**
  SelectedValue および **SelectedItem**の取得に使用されるプロパティの名前を表します。

- **MaxDropDownHeight**
  ドロップダウンリストの最大高さを設定するには、**MaxDropDownHeight** プロパティの値を設定します。

- **MaxDropDownWidth**
  ドロップダウンリストの最大幅を設定するには、**MaxDropDownWidth** プロパティの値を設定します。

Copyright © GrapeCity, Inc. All rights reserved.

- **IsEditable**
  ComboBox の項目リストにない値をユーザーが入力できるかどうかを決定します。

- **VirtualizationThreshold**
  Use the **VirtualizationThreshold** property to set the minimum number of rows and/or columns required to enable virtualization in the ComboBox control. For more information, see 仮想化.

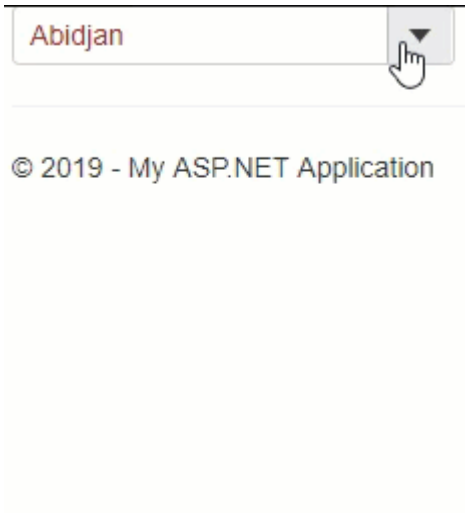
クイックスタート

このセクションでは、MVC Web アプリケーションに **ComboBox** コントロールを追加し、そこにデータを追加する方法について説明します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:ComboBox のデータソースの作成**
- **手順 3:ComboBox コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の ComboBox を示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2:ComboBox のデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`Cities.cs`)。
2. 次のコードを新しいモデルに追加して、ComboBox コントロールのデータソースになるクラスを定義します。

C#

```
public class Cities
    {
        public static List<string> GetCities()
        {
            return new List<string>
            {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                "Abidjan", "Accra", "Ahmedabad", "Alexandria", "Ankara",
"Atlanta", "Baghdad", "Bandung", "Bangkok", "Barcelona", "Beijing", "Belo
Horizonte",
        "Bengaluru", "Bogota", "Boston", "Buenos Aires", "Cairo", "Calcutta",
"Chengdu", "Chennai", "Chicago", "Chongqung", "Dalian", "Dallas", "Delhi",
        "Detroit", "Dhaka", "Dongguan", "Essen", "Fuzhou", "Guadalajara",
"Guangzhou", "Hangzhou", "Harbin", "Ho Chi Minh City", "Hong Kong", "Houston",
        "Hyderabad", "Istanbul", "Jakarta", "Johannesburg", "Karachi",
"Khartoum", "Kinshasa", "Kuala Lumpur", "Lagos", "Lahore", "Lima", "London",
        "Los Angeles", "Luanda", "Madrid", "Manila", "Medellin", "Mexico City",
"Miami", "Milan", "Monterrey", "Moscow", "Mumbai", "Nagoya", "Nanjing",
        "Naples", "New York", "Osaka", "Paris", "Pheonix", "Philadelphia",
"Porto Alegre", "Pune", "Qingdao", "Quanzhou", "Recife", "Rio de Janeiro",
        "Riyadh", "Rome", "Saint Petersburg", "Salvador", "San Francisco",
"Santiago", "Sao Paulo", "Seoul", "Shanghair", "Shenyang", "Shenzhen",
        "Singapore", "Surabaya", "Surat", "Suzhou", "Sydney", "Taipei",
"Tehran", "Tianjin", "Toronto", "Washington", "Wuhan", "Xi'an-Xianyang",
"Yangoon",
        "Zhengzhou", "Tokyo"
            };
        }
    }
}
```

先頭に戻る

手順 3：ComboBox コントロールの追加

ComboBox コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`IndexController`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー `QuickStartController`をダブルクリックして開きます。
2. メソッド `QuickStart()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **QuickStart** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。
6. ComboBox コントロールをビュー `QuickStart` で次のようにインスタンス化します。

Razor copyCode

```
@{List<string> cities = Cities.GetCities();}

@(Html.C1().ComboBox()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .Bind(cities)
        .SelectedIndex(0))
```

先頭に戻る

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]→[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例: http://localhost:1234/**QuickStart/QuickStart**)、ビューを表示します。

先頭に戻る

ComboBox の使用

This section comprises all the features and auxiliary functionalities offered by ComboBox.

大文字と小文字を区別する検索
: Learn how to perform case-sensitive search in ComboBox control.

大文字と小文字を区別する検索

By default, the search in ComboBox is case-insensitive. However, there might be a scenario where you need case-sensitive search in ComboBox. With case-sensitive search, ComboBox allows you to have more granular control over searched items. You can achieve the granularity in search by setting **CaseSensitiveSearch** property of the **ComboBox** class to **true**. The **CaseSensitiveSearch** property determines whether the searches performed while the user types should be case-sensitive which makes the search more refined.



The following example shows how you can use the case-sensitive search in the ComboBox control. The example uses Cities.cs model added in the Quick Start topic.

Controller

ComboBoxSearchController.cs copyCode

```
public ActionResult Index()
{
    return View();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

View for the Controller

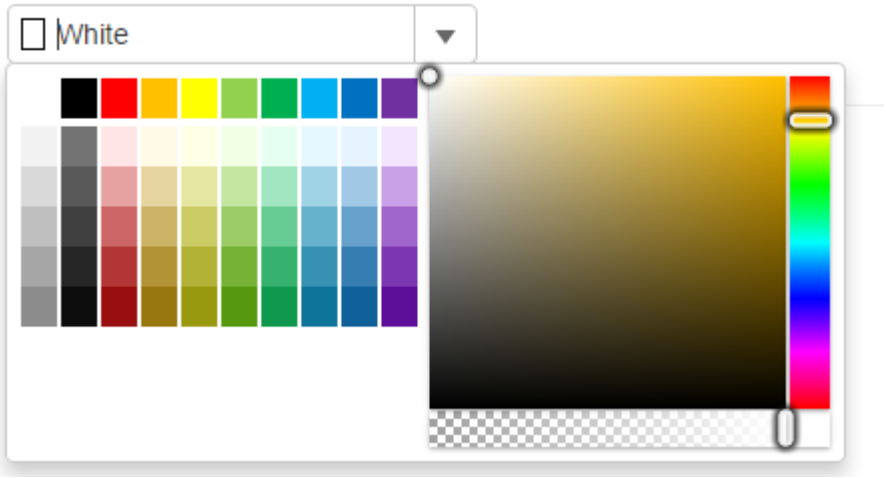
Index.cshtml copyCode

```
@{List<string> cities = Cities.GetCities();}

@(Html.C1().ComboBox()
        .Bind(cities)
        .SelectedIndex(0)
        .IsEditable(true)
        .CaseSensitiveSearch(true))
```

InputColor

InputColor コントロールを使用すると、HTML でサポートされる色文字列を入力して色を選択したり、ドロップダウンリストとして表示される ColorPicker コントロールから色を選択することができます。この ColorPicker コントロールの Value プロパティは、現在選択されている色を取得または設定します。



主な特長

InputColor には主に次の便利な機能があります。

- **Value**
  ユーザーが現在選択している色を表します。
- **ShowAlphaChannel**
  ユーザーがコントロールの色のアルファチャンネル（透過性）を編集できるようにします。デフォルト値は true です。
- **Required**
  コントロール値を色にする必要があるか、それとも null に設定できるかを示します。

クイックスタート

このトピックでは、アプリケーションに **InputColor** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC

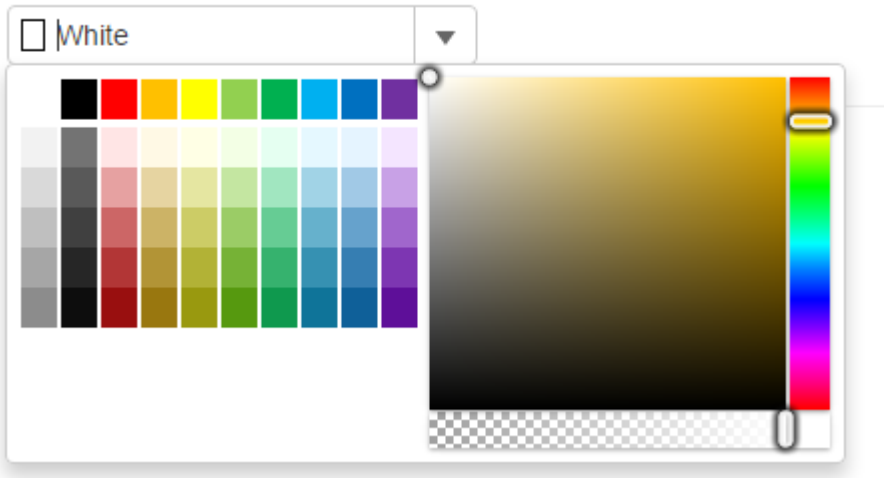
Copyright © GrapeCity, Inc. All rights reserved.

コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：inputColor コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **InputColor** を示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。

Copyright © GrapeCity, Inc. All rights reserved.

1. コントローラーの名前を設定します（例：`Default1Controller`）。
2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
3. **[追加]**をクリックします。

4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**InputColor** コントロールを初期化します。

Razor copyCode

```
@using System.Drawing
<br />
<br />
<script>
    function changeColor(sender, e) {
        document.getElementById("image1").style.backgroundColor = sender.value;
    }
</script>
// 画像を追加して高さと幅を設定します
<p>
    <img id="image1" height="300" width="400"
         src="@Href("~/Content/images/componentone_logo.png")" />
</p>
<div>
    //画像の背景色を設定します

@Html.C1().InputColor().Value(Color.White).OnClientValueChanged("changeColor")
</div>
```

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**Default1/index**）、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

InputColor の使用

This section comprises all the features and auxiliary functionalities offered by InputColor.

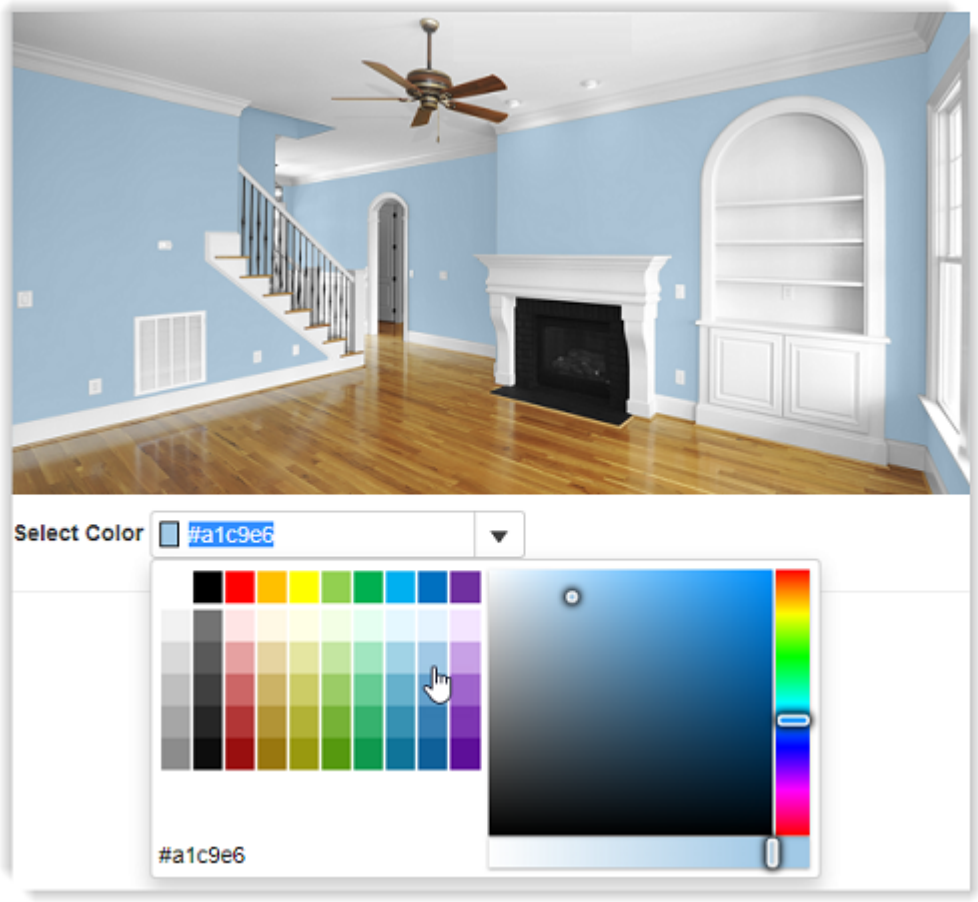
色の文字列表現の表示

Copyright © GrapeCity, Inc. All rights reserved.

Learn how to show string representation of color in the InputColor control.

色の文字列表現の表示

InputColor control allows you to set a value indicating whether the ColorPicker shows a string representation of the current color using the **ShowColorString** property. By default, the value for the **ShowColorString** property is set to **false.** However, you can set it to true in order to display string representation of a color.



The following code showcases how to display string representation of the currently selected color. In this example, an image of a room is added from the available InputColor product sample whose walls change color according to the color you select from the ColorPicker at runtime.

Index.cshtml copyCode

```
@using System.Drawing

<p>
    <img id="image1" src="@Href("~/Content/images/room.png")" />
</p>

<script>
    function changeColor(sender, e) {
        document.getElementById("image1").style.backgroundColor = sender.value;
    }
</script>

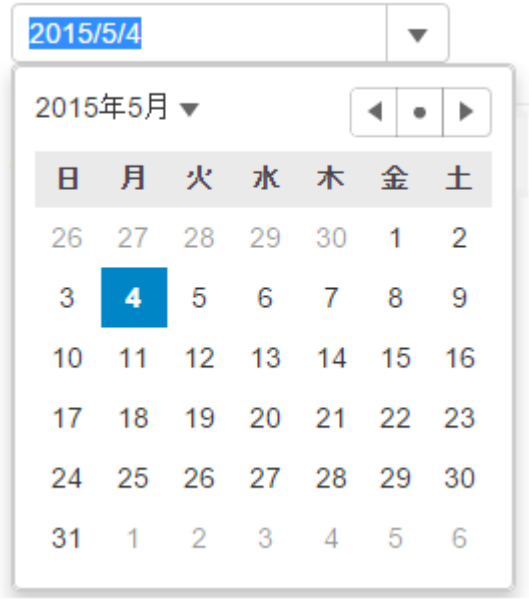
<div>
    <label>Select Color</label>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    @(Html.C1().InputColor()
        .Id("invalidInput")
        .Value(Color.White)
        .ShowColorString(true)
        .ClickAction(ClickAction.Toggle)
        .OnClientValueChanged("changeColor")
    )
</div>
```

InputDate

InputDate コントロールを使用すると、Globalize クラスでサポートされる任意の書式で日付を入力したり、ドロップダウンボックスに表示されるカレンダーコントロールから日付を選択することもできます。入力できる値の範囲を制限するために、**Min** プロパティと **Max** プロパティを使用できます。



主な特長

InputDate には主に次の便利な機能があります。

- **Value**
  ユーザーが現在選択している日付を表します。
- **Max**
  入力できる最遅日の値を設定するには、**Max** プロパティを使用します。
- **Min**
  入力できる最早日の値を設定するには、**Min** プロパティを使用します。
- **Format**
  さまざまな書式で日付を表すには、**Format** プロパティを使用します。
- **Required**
  コントロールを Date にする必要があるか、それとも null に設定できるかを示します。

Copyright © GrapeCity, Inc. All rights reserved.

- **DateSelectionMode**
  このプロパティを使用して、ユーザーが日または月を選択できるか、それとも値を選択できないかを示します。

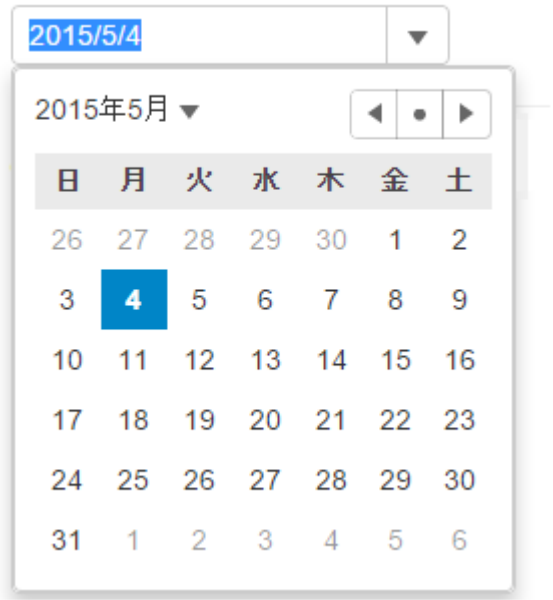
クイックスタート

このトピックでは、アプリケーションに **InputDate** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：InputDate コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **InputDate** を示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
   1. コントローラーの名前を設定します（例：`Default1Controller`）。
   2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
   3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

Copyright © GrapeCity, Inc. All rights reserved.

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**InputDate** コントロールを初期化します。

Razor copyCode

```
<br />
<br />
@{
    var today = DateTime.Now.Date;
    var minDate = new DateTime(today.Year, 1, 1);
    var maxDate = new DateTime(today.Year, 12, 31);
}

<div>
    @(Html.C1().InputDate().Id("idcInputDate")

.Value(today).Min(minDate).Max(maxDate).OnClientValueChanged("changeDate")
    )
</div>
```

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**Default1/index**）、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

InputDateの使用

This section comprises all the features and auxiliary functionalities offered by InputDate.

日付検証
: Learn how to add ItemValidator in the InputDate control.

日付検証

ComponentOne MVC Editionの**InputDate**コントロールに組み込まれたバリデータ**ItemValidator**を使用して、ユーザーがInputDateドロップダウンから選択するデータの型や値を制御できます。これを行うには、validation関数を作成して、ユーザーによる日付選択を許可または制限するItemValidatorに割り当てるだけです。

この例では、ユーザーがInputDateコントロールで平日を選択でき、土曜と日曜は選択できないようにItemValidatorが作成されます。この種のvalidationは、重要な作業日、ジムに通う日、ダイエットプランなどを計画する際に便利です。

Copyright © GrapeCity, Inc. All rights reserved.

次の図は、ItemValidator関数を適用した**InputDate**コントロールを示しています。



次のコード例は、ItemValidatorをInputDateコントロールに追加する方法を示します。

コードの場合

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

InputDate.cshtml

Razor copyCode

```
@{
    var today = DateTime.Now.Date;
    var minDate = new DateTime(today.Year, 1, 1);
    var maxDate = new DateTime(today.Year, 12, 31);
}

<script>
    function itemValidator(date) {
        var weekday = date.getDay();
        return weekday != 0 && weekday != 6;
    }
</script>

<div>

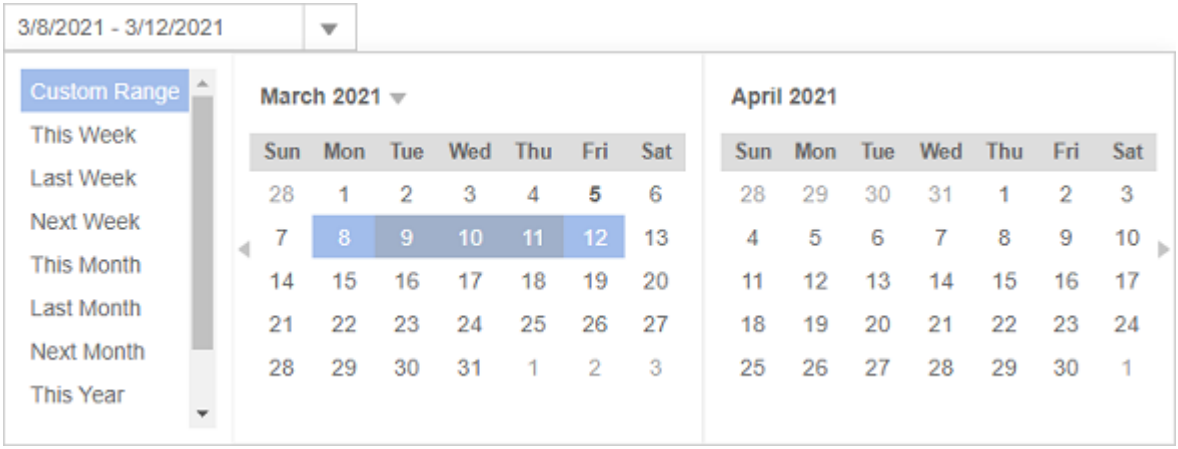
    @(Html.C1().InputDate().Value(today).Min(minDate).Max(maxDate)
    .ItemValidator("itemValidator"))
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

HTML HTML

InputDateRange

Incorporate **InputDateRange for ASP.NET MVC**, an advanced date picker, in your applications to provide users the ease of selecting a range of dates. The InputDateRange control can be used in scenarios where you need to analyze data or run reports for a specific date window. It extends the InputDate control and provides a simple and intuitive UI for selecting start and end date values. Moreover, it allows you to specify custom date ranges that users can pick from (like this week, last month, next year, etc.), or they can use the multi-month calendar in the drop-down to select date ranges with the mouse or keyboard.



主な特長

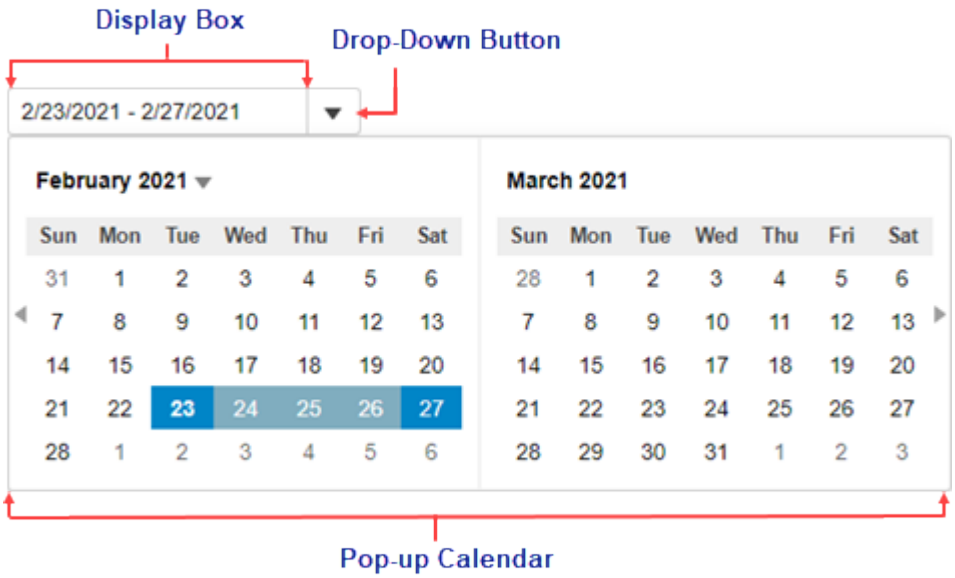
Following are some of the main features of InputDateRange that you may find useful:

- **Predefine Date Ranges**
  The InputDateRange control helps you define the date ranges (like last 30 days or last week) beforehand to set the date ranges frequently used by the users.
- **Limit the Date Selection Range**
  Restrict the users to select date range within specific span of days by setting maximum and minimum range option.
- **Date Range Validation**
  InputDateRange supports validation by allowing only valid values to be entered by the user and validating them.
- **Customize Appearance**
  InputDateRange allows you to change its appearance and customize the style based on your application's look and feel.

要素

InputDateRange control is a simple control with a clean and interactive interface which looks similar to the following image:

Copyright © GrapeCity, Inc. All rights reserved.

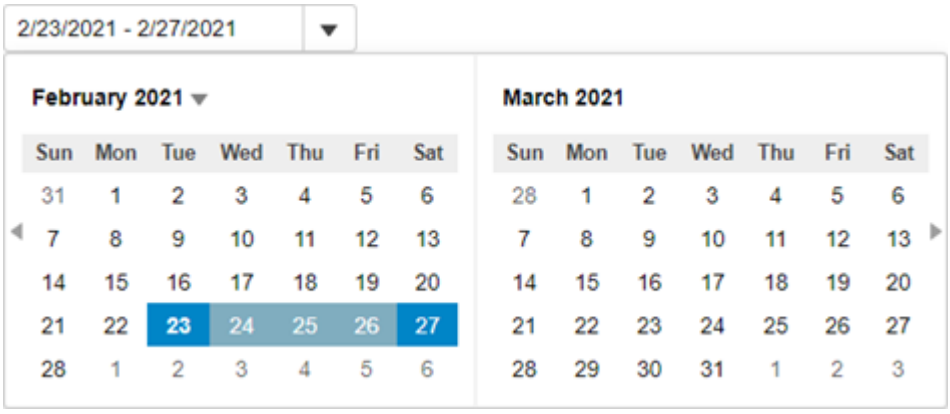


The **InputDateRange** control consists of the following elements:

- **Display Box**
  The display box shows the selected date range. You can set the date range by entering numeric values or by selecting a date from the pop-up calendar at runtime.
- **Drop-Down Button**
  The drop-down button opens a calendar from where you can select a date range to display in the display box.
- **Pop-up Calendar**
  The pop-up calendar displays a two month calendar, by default, and allows you to select a date range to be displayed in the display box. It appears when you click the drop-down button.

クイックスタート

The following quick start guide is intended to get you up and running with the InputDateRange control. In this quick start, you start with creating a new application, add the **InputDateRange** control to it, set its value and display a date range in it. When you click the drop-down button to open the calendar, the InputDateRange control shows two calendar months, by default. However, you can change the month count displayed in the control as per your requirements. For more information on changing the number of months displayed in the InputDateRange control, see Date Range topic.



To accomplish this, follow these steps:

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about

Copyright © GrapeCity, Inc. All rights reserved.

creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add InputDateRange

To add the InputDateRange control to your application, add **C1.Web.MVC** reference and follow these steps:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add New Scaffold Item** dialog appears.
3. In the **Add New Scaffold Item** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template.
    2. Click **Add**. The **Add Controller** dialog appears.
    3. In the **Add Controller** dialog, set name of the controller (for example: `InputDateRangeController`).
    4. Click **Add**.

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `InputDateRangeController.`
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller. Copy the following code and paste it inside **Index.cshtml** to create a simple application with the InputDate control.

Index.cshtml copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
}

@(Html.C1().InputDateRange().Id("demoControl")
        .Value(today)
        .RangeEnd(rangeEnd)
)
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/InputDateRange/Index) in the address bar of the browser to see the view.

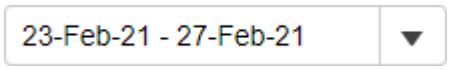
Back to Top

日時書式

InputDateRange allows you to set the date format for displaying the date range in a specified format. You can use **Format** property to set the standard date format or any other date format of your choice for the InputDateRange

Copyright © GrapeCity, Inc. All rights reserved.

control.



The following code showcases the use of **Format** property to set "dd-MMM-yy" date format for the displayed date range in the control.

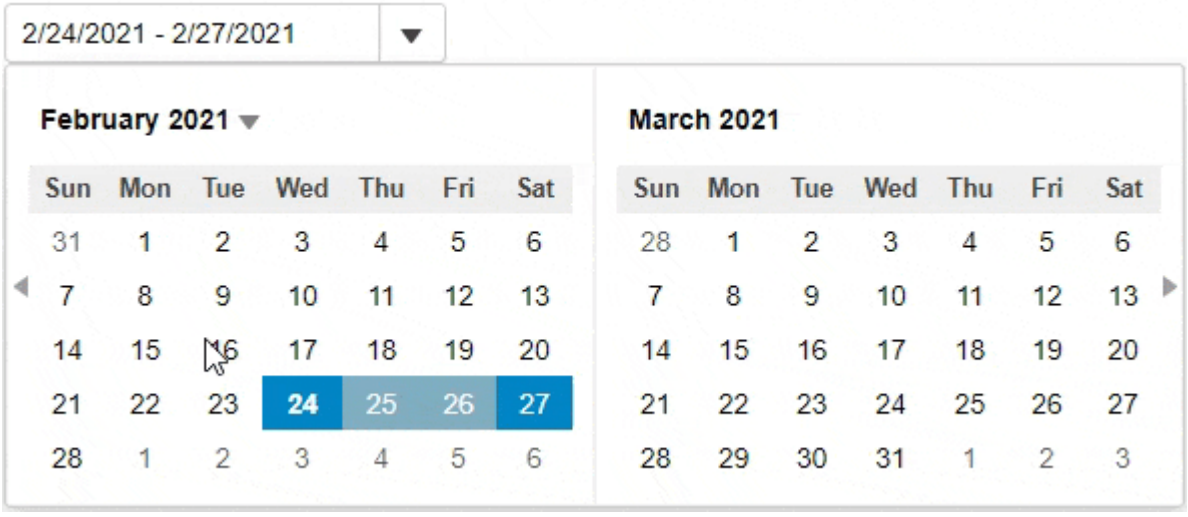
Index.cshtml copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
}

@(Html.C1().InputDateRange().Id("demoControl")
        .Value(today)
        .RangeEnd(rangeEnd)
        .Format("dd-MMM-yy")
)
```

日範囲

InputDateRange lets you to customize the dropdown calendar to show a limited set of days in the visible calendar month(s) and restrict the selection to the specified range of dates from the calendar. This can be done using the **RangeMin** and **RangeMax** properties. The **RangeMin** and **RangeMax** properties allow you to limit the minimum and maximum number of days to be displayed in the selected date range at runtime, respectively.



The following code uses the RangeMin and RangeMax properties to display a minimum three days and maximum five days in the selected date range in the control:

C# copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(3);
}

@(Html.C1().InputDateRange().Id("demoControl")
        .CloseOnSelection(false)
        .SelectionMode(DateSelectionMode.Range)
```


Copyright © GrapeCity, Inc. All rights reserved.

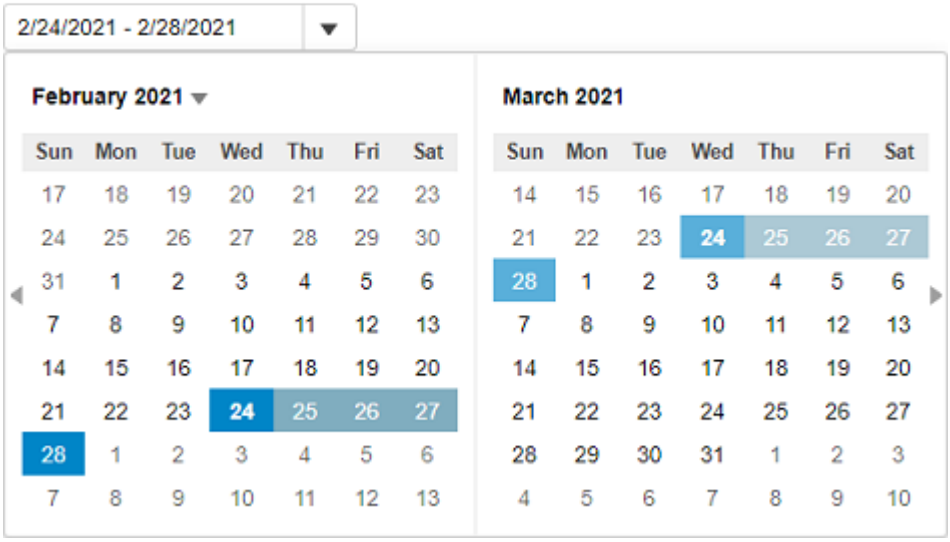
```
        .Value(today)
        .RangeEnd(rangeEnd)
        .RangeMin(3)
        .RangeMax(5)
)
```

カレンダーの最小日と最大日の設定

In addition to setting the day range, InputDateRange allows you to customize the dropdown calendar to show the specified number of weeks before and after the currently visible calendar month. Furthermore, it lets you display multiple calendar months at a time in the dropdown calendar. Let us dive into the details of how to achieve these in the following sections.

Specify Weeks to Display Before and After the Current Month

InputDateRange allows you to specify the number of weeks you want to display on the calendar before and after the current calendar month. However, these weeks are displayed in disabled state on the calendar. InputDateRange provides the **WeeksBefore** property to set the number of weeks to be displayed before the current month and the **WeeksAfter** property to set the number of weeks to be displayed after the current month.



The following code uses the **WeeksBefore** and **WeeksAfter** properties to display two weeks before and one week after the currently displayed months in the calendar:

C# copyCode

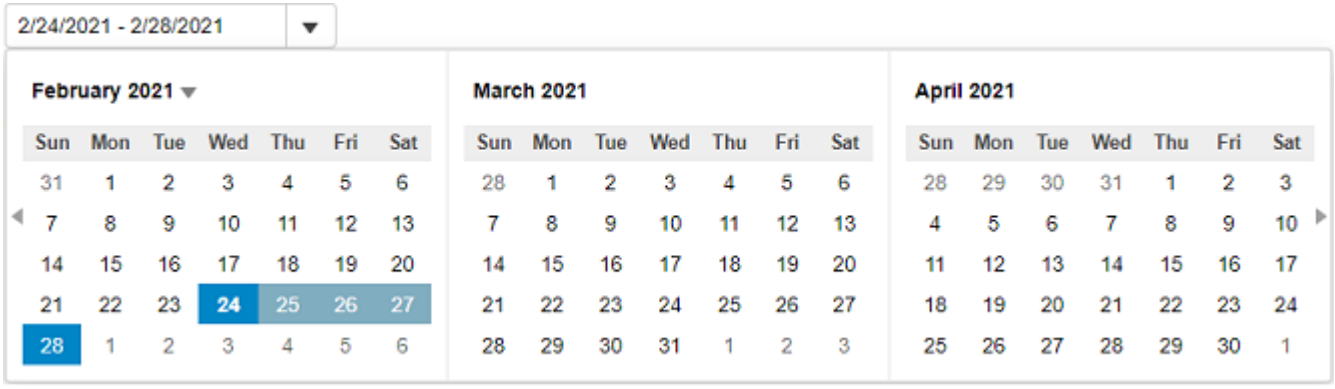
```
@{
        var today = DateTime.Now.Date;
        var rangeEnd = today.AddDays(4);
    }

    @(Html.C1().InputDateRange().Id("demoControl")
            .CloseOnSelection(false)
            .Value(today)
            .RangeEnd(rangeEnd)
            .WeeksBefore(2)
            .WeeksAfter(1)
    )
```


Copyright © GrapeCity, Inc. All rights reserved.

Display Multiple Calendar Months

The InputDateRange control supports multi-month view. It allows you to show multiple months in the calendar, at a time, using the **MonthCount** property. The value for the **MonthCount** property is set to 2, by default, for the InputDateRange control. However, you can always change it based on your requirements.



The following code showcases the use of **MonthCount** property to display three months at a time in the calendar of the InputDateRange control.

C# copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
}

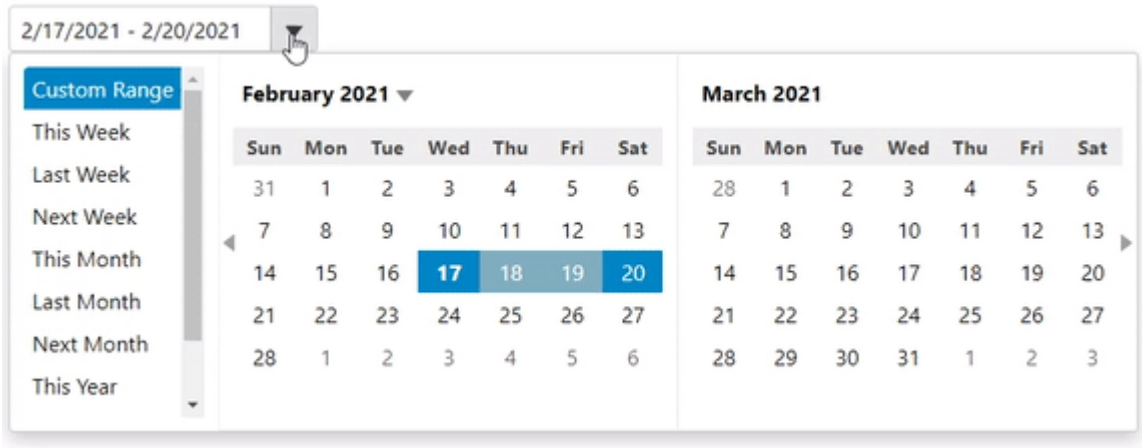
@(Html.C1().InputDateRange().Id("demoControl")
        .CloseOnSelection(false)
        .MonthCount(3)
        .Value(today)
        .RangeEnd(rangeEnd)
)
```

事前定義された範囲

InputDateRange lets you to predefine date ranges based on your requirements. Using the InputDateRange control, you can define various date ranges, such as current week, last week, next month, this year range, custom ranges, etc., in your applications for users to choose from. You can use the **PredefinedRanges** property to define the day, week, month, year, or custom ranges in the InputDateRange control.

Note that this property works when the **SelectionMode** property is set to **DateSelectionMode.Range**.

Copyright © GrapeCity, Inc. All rights reserved.



The following code snippets showcases how you can define different week, month, year or custom ranges using the **PredefinedRanges** property.

Index.cshtml copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(3);
}

<script>
    // 事前定義された日付範囲を取得します
    function getPredefinedRanges() {
        let dt = wijmo.DateTime, now = new Date();
        return {
            // カスタムの範囲
            'Custom Range': null,
            // 曜日
            'This Week': [dt.weekFirst(now), dt.weekLast(now)],
            'Last Week': [dt.weekFirst(dt.addDays(now, -7)),
dt.weekLast(dt.addDays(now, -7))],
            'Next Week': [dt.weekFirst(dt.addDays(now, +7)),
dt.weekLast(dt.addDays(now, +7))],
            // 月
            'This Month': [dt.monthFirst(now), dt.monthLast(now)],
            'Last Month': [dt.monthFirst(dt.addMonths(now, -1)),
dt.monthLast(dt.addMonths(now, -1))],
            'Next Month': [dt.monthFirst(dt.addMonths(now, +1)),
dt.monthLast(dt.addMonths(now, +1))],
            // 年
            'This Year': [dt.yearFirst(now), dt.yearLast(now)],
            'Last Year': [dt.addYears(dt.yearFirst(now), -1),
dt.addYears(dt.yearLast(now), -1)],
            'Next Year': [dt.addYears(dt.yearFirst(now), +1),
dt.addYears(dt.yearLast(now), +1)],
        };
    }
</script>

@(Html.C1().InputDateRange().Id("demoControl")
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        .CloseOnSelection(false)
        .MonthCount(2)
        .SelectionMode(DateSelectionMode.Range)
        .PredefinedRanges("getPredefinedRanges")
        .Value(today)
        .RangeEnd(rangeEnd)
)
```

表示モード

InputDateRange control allows you to control its editability and state. This can be useful in the scenarios where you want limit the user from modifying the control value until some other conditions have been met, like selecting a radio button or a checkbox. Let us explore more about these modes in the following sections.

Disabled Mode

You can keep the InputDateRange control in deactivated state using the **IsDisabled** property. This allows you to inactivate the InputDateRange control and limit any modifications.



To disable the InputDaterange control using the **IsDisabled** property, use the following code:

Index.cshtml copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
}

@(Html.C1().InputDateRange().Id("demoControl")
        .Value(today)
        .RangeEnd(rangeEnd)
        .IsDisabled(true)
)
```

ReadOnly Mode

InputDateRange provides the **IsReadOnly** property which lets you choose whether to allow the user to modify the control value with the mouse and keyboard at runtime. To make the InputDateRange control non-editable using the **IsReadOnly** property, use the following code:

Index.cshtml copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
}

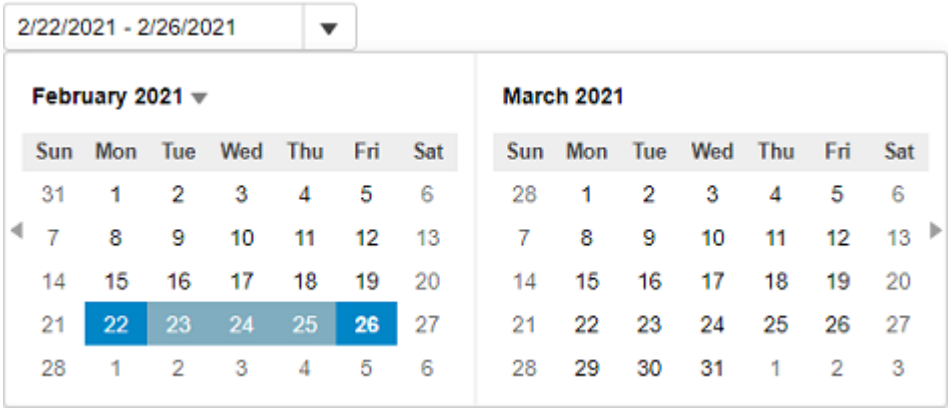
@(Html.C1().InputDateRange().Id("demoControl")
        .Value(today)
        .RangeEnd(rangeEnd)
        .IsReadOnly(true)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
)
```

日付の検証

The **InputDateRange** control has a built-in validator named **ItemValidator** that lets you control the type of data or the value that users select from the drop down calendar. All you need to do is create a validation function and assign it to the ItemValidator to allow or restrict the selection of dates by the user.



The following code examples demonstrate how to add ItemValidator in the InputDateRange control. In this example, ItemValidator is created so that a user can select a weekday, but not Saturdays and Sundays (weekend). This kind of validation is useful while planning your work sprints, workout days or diet plan.

View Code

Index.cshtml copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
    var minDate = new DateTime(today.Year, 1, 1);
    var maxDate = new DateTime(today.Year, 12, 31);
}

<script>
    function itemValidator(date) {
        var weekday = date.getDay();
        return weekday != 0 && weekday != 6;
    }
</script>

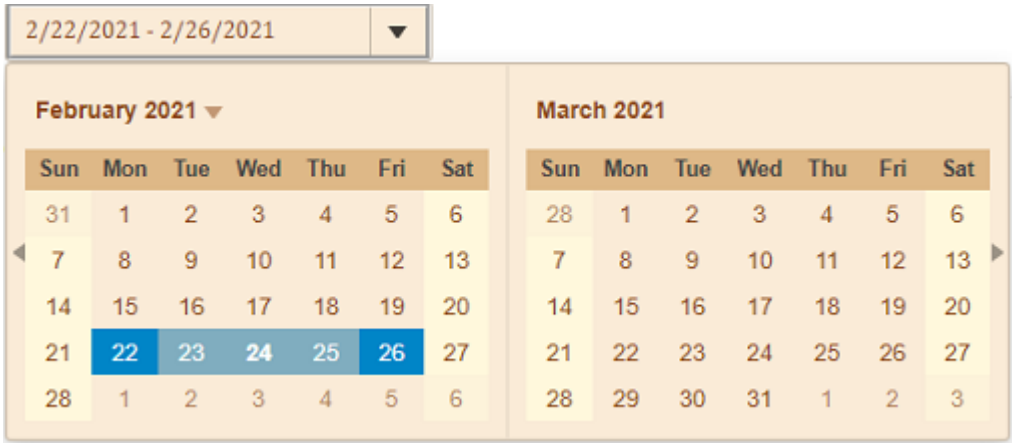
@(Html.C1().InputDateRange()
        .Value(today)
        .RangeEnd(rangeEnd)
        .Min(minDate)
        .Max(maxDate)
        .ItemValidator("itemValidator"))
```

レイアウトおよび外観

The InputDateRange control has a simple layout which makes it easy to style using CSS. InputDateRange provides

Copyright © GrapeCity, Inc. All rights reserved.

various options to customize the appearance of the InputDateRange control and all its elements using CSS, so that you can style and display the InputDateRange control as per your requirement.



The following code demonstrates the use of CSS to style the InputDateRange control:

C# copyCode

```
@{
    var today = DateTime.Now.Date;
    var rangeEnd = today.AddDays(4);
}
<style>
    .wj-control {
        color: saddlebrown;
        background-color: antiquewhite;
    }

    .wj-input-group {
        font-family: 'Cambria Math';
        border: groove;
    }

    .wj-header {
        background-color: burlywood;
    }

    .wj-day-weekend {
        background-color: cornsilk;
    }

    .wj-inputdate-dropdown {
        max-height: initial;
    }
</style>

@(Html.C1().InputDateRange().Id("demoControl")
        .CloseOnSelection(false)
        .Value(today)
        .RangeEnd(rangeEnd)
        .IsDisabled(true)
)
```


Copyright © GrapeCity, Inc. All rights reserved.

InputMask

InputMask コントロールは、ユーザーによって入力される値を制御する方法を提供します。このコントロールを使用すると、キー入力時に無効な入力を防止したりリテラル（日付の中のスラッシュなど）をスキップするためのマスクを使用して、文字列を編集できます。InputMask コントロールの値を指定するには、**Mask** プロパティを使用します。



入力の検証に使用するマスクは、マスクプロパティで定義されます。マスクには、次の特殊文字を使用できます。

0	数字。
9	数字またはスペース。
#	数字、符号、またはスペース。
L	英字。
L	英字またはスペース。
A	英数字。
a	英数字またはスペース。
.	ローカライズされた小数点。
,	ローカライズされた桁区切り。
:	ローカライズされた時間の区切り文字。
/	ローカライズされた日付の区切り文字。
$	ローカライズされた通貨記号。
<	後続の文字を小文字に変換します。
>	後続の文字を大文字に変換します。
|	大文字小文字の変換を無効にします。
****	任意の文字をエスケープして、リテラルに変えます。
9	(\uff19) DBCS数字。
J	(\uff2a) DBCSひらがな。
G	(\uff27) DBCS大文字ひらがな。
K	(\uff2b) DBCSカタカナ。
N	(\uff2e) DBCS大文字カタカナ。
K	SBCSカタカナ。
N	SBCS大文字カタカナ。

Copyright © GrapeCity, Inc. All rights reserved.

Z	(\uff3a) 任意のDBCS文字。
H	任意のSBCS文字。
その他	リテラル

主な特長

InputMask には主に次の便利な機能があります。

- **Mask**
  コントロールのマスクを指定するには、**Mask** プロパティを設定します。これは、ユーザーのキー入力に伴う入力の検証に使用されます。
- **Placeholder**
  コントロールが空のときにヒントとして表示される **Placeholder** 値を設定します。
- **Value**
  InputMask コントロールの値を指定します。
- **PromptChar**
  コントロール内の入力位置を示すために使用する記号を設定するには、**PromptChar** プロパティを使用します。

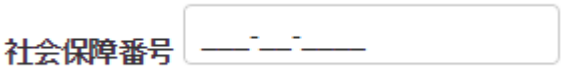
クイックスタート

このトピックでは、アプリケーションに **InputMask** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：InputMask コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **InputMask** を示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。

Copyright © GrapeCity, Inc. All rights reserved.

2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
3. **[追加]**をクリックします。

4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**InputMask** コントロールを初期化します。

Razor copyCode

```
<div>
    <label>社会保障番号</label>
    @Html.C1().InputMask().Mask("000-00-0000").HtmlAttributes(new { title =
"Mask: 000-00-0000" })
</div>
```

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**Default1/index**）、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

InputMaskの使用

This section comprises all the features and auxiliary functionalities offered by InputMask.

入力マスクのタイプ
Learn how to bind Menu to an InputNumber control.

入力マスクのタイプ

Mask プロパティを使用してさまざまなタイプのデータを検証するために、InputMask コントロールを使用できます。たとえば、電話番号の前に3桁の局番を付けるマスクを追加できます。このコントロールを InputDate および InputTime と組み合わせて使用して、ユーザーが誤って無効なデータを入力しないようにすることもできます。

次の図は、さまざまなマスクを適用した **InputMask** コントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、Menu を InputNumber コントロールに連結する方法を示します。

コードの場合

InputMaskController.cs

Razor copyCode

```
public ActionResult Feature()
{
    return View();
}
```

InputMask.cshtml

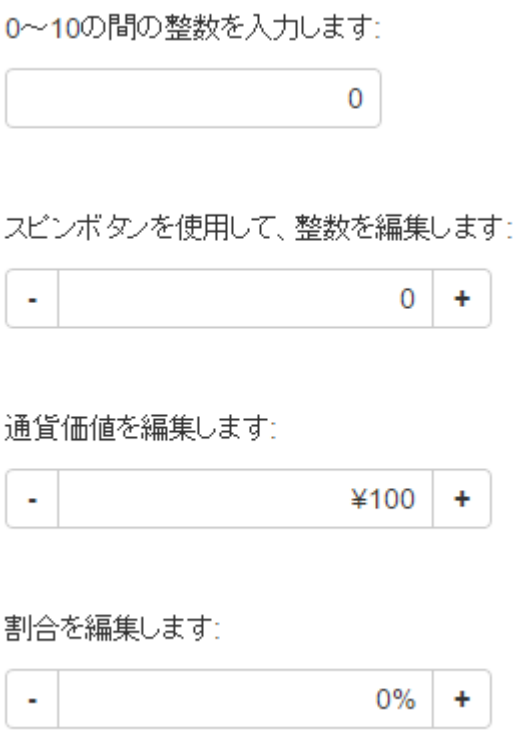
Razor copyCode

```
<p> 電話番号用のInputMask</p>
<div>
    @(Html.C1().InputMask().Mask("(999) 000-0000")
          .HtmlAttributes(new { title = "Mask: (999) 000-0000" })
    )
</div>
<p> InputDateを含むInputMask</p>
<div>
    @(Html.C1().InputDate().Value(DateTime.Now)
            .Format("d").Mask("99/99/9999")
            .HtmlAttributes(new { title = "Mask: 99/99/9999" })
    )
</div>
<p> InputTimeを含むInputMask</p>
<div>
    @(Html.C1().InputTime().Value(DateTime.Now)
            .Format("t").Mask("00:00 >LL").Step(new TimeSpan(0, 15, 0))
            .HtmlAttributes(new { title = "Mask: 00:00 >LL" })
    )
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

InputNumber

InputNumber コントロールは、ユーザーが数字を入力できるようにします。このコントロールは、ユーザーが編集時に無効なデータや書式を入力することを防止します。有効な値の範囲を制限するために、**Min** プロパティと **Max** プロパティを使用することができます。



主な特長

InputNumber には主に次の便利な機能があります。

- **Value**
  コントロールの現在の値を表します。
- **Max**
  ユーザーが入力できる最大数を設定するには、**Max** プロパティを使用します。
- **Min**
  ユーザーが入力できる最小値を設定するには、**Min** プロパティを使用します。
- **ShowSpinner**
  この値は、値を増減するスピナーボタンを表示するかどうかを示します。
- **InputType**
  コントロールによってホストされる HTML 入力要素の type 属性を設定するには、**InputType** プロパティを使用します。

クイックスタート

me="top">

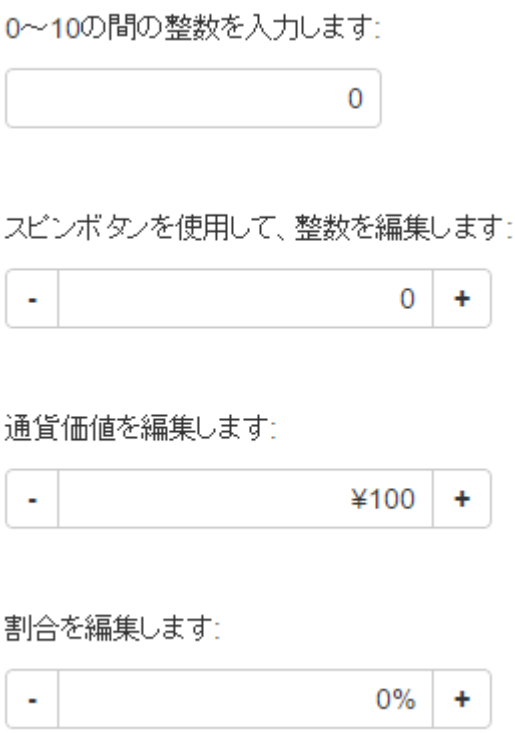
Copyright © GrapeCity, Inc. All rights reserved.

このトピックでは、アプリケーションに **InputNumber** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：InputNumber コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **InputNumber** を示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。

Copyright © GrapeCity, Inc. All rights reserved.

2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**InputNumber** コントロールを初期化します。

Razor copyCode

```
<div>
    <p>
        0～10の間の整数を入力します：
    </p>
    @(Html.C1().InputNumber().Value(0).Format("n0")
    .Min(0).Max(10)
    .Placeholder("integer between zero and ten")
    )
</div>
<div>
    <p> スピンボタンを使用して、整数を編集します：</p>
    @(Html.C1().InputNumber().Value(0).Format("n0")
    .Min(0).Max(10)
    .Step(1)
    .Placeholder("integer between zero and ten")
    )
</div>
<div>
    <p> 通貨価値を編集します：</p>
    @(Html.C1().InputNumber()
        .Value(100).Format("c")
        .Step(100).Min(100).Max(1000)
        .Placeholder("amount below $1,000")
    )
</div>
<div>
    <p> 割合を編集します：</p>
    @(Html.C1().InputNumber()
        .Placeholder("percentage")
        .Format("p0")
        .Step(.1)
    )
</div>
```

手順 4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**Default1/index**）、ビューを表示します。または、ビューをホームページにリンクします。

Copyright © GrapeCity, Inc. All rights reserved.

先頭に戻る

InputTime

InputTime コントロールを使用すると、Globalize クラスでサポートされる任意の書式で時間を入力したり、ドロップダウンリストから時間を選択することができます。**Min** プロパティ、**Max** プロパティ、および **Step** プロパティは、リストに表示される値を決定します。



主な特長

InputTime には主に次の便利な機能があります。

- Value
  現在の入力時間を表します。
- Max
  ユーザーが入力できる最遅時間を設定するには、Max プロパティを使用します。
- Min
  ユーザーが入力できる最早時間を設定するには、Min プロパティを使用します。
- Format
  さまざまな書式で時間を表示するには、Format プロパティを使用します。
- SelectedItem
  ドロップダウンリストで現在選択されている項目を表します。
- SelectedIndex
  SelectedItem のインデックスを示します。
- SelectedValue
  SelectedValuePath を使用して取得される SelectedItem の値を表します。
- Step
  ドロップダウンリストのエントリ間の分数を指定するには、Step プロパティを使用します。

Copyright © GrapeCity, Inc. All rights reserved.

クイックスタート

このトピックでは、アプリケーションに **InputTime** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：InputTime コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **InputTime** を示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

Copyright © GrapeCity, Inc. All rights reserved.

手順 3:コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**InputTime** コントロールを初期化します。

Razor copyCode

```
<br />
<br />
@{
    var minTime = DateTime.Parse("07時");
    var maxTime = DateTime.Parse("19時");
}

<div>
    <label>時刻を選択</label>
    @(Html.C1().InputTime().Id("iditInputTime")
        .Value(DateTime.Now).Min(minTime).Max(maxTime)
        .Step(new TimeSpan(0, 15, 0))
    )
</div>
```

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例: http://localhost:1234/**Default1/index**)、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

InputTimeの使用

InputTime also offers many advanced features. These features are listed below:

データ連結
: Learn how to bind InputTime to fetch data from a list.

データ連結

InputTimeを使用すると、ユーザーはInputTimeドロップダウンに表示されるカスタム時刻のリストを作成できます。InputTimeは、そのBindプロパティを使用して特定の操作を実行するための変数を渡すことにより、コレクションに連結します。このトピックでは、既存のデータソースからデータを取得する方法について説明します。これは、InputTimeコントロールにカスタムタイミングを連結する際に便利です。

次の図では、リストに連結された後のInputTimeがBindプロパティを使用してリストからデータをフェッチしています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、リストからデータをフェッチするためにInputTimeを連結する方法を示します。

コードの場合

DataBindController.cs

C# copyCode

```
    public ActionResult Index()
    {
        ViewBag.TimeList = new List<object>
        { "8:20", "10:00", "11:35", "12:08", "13:25", "13:30", "14:26" };
        return View();
    }
}
```

DataBind.cshtml

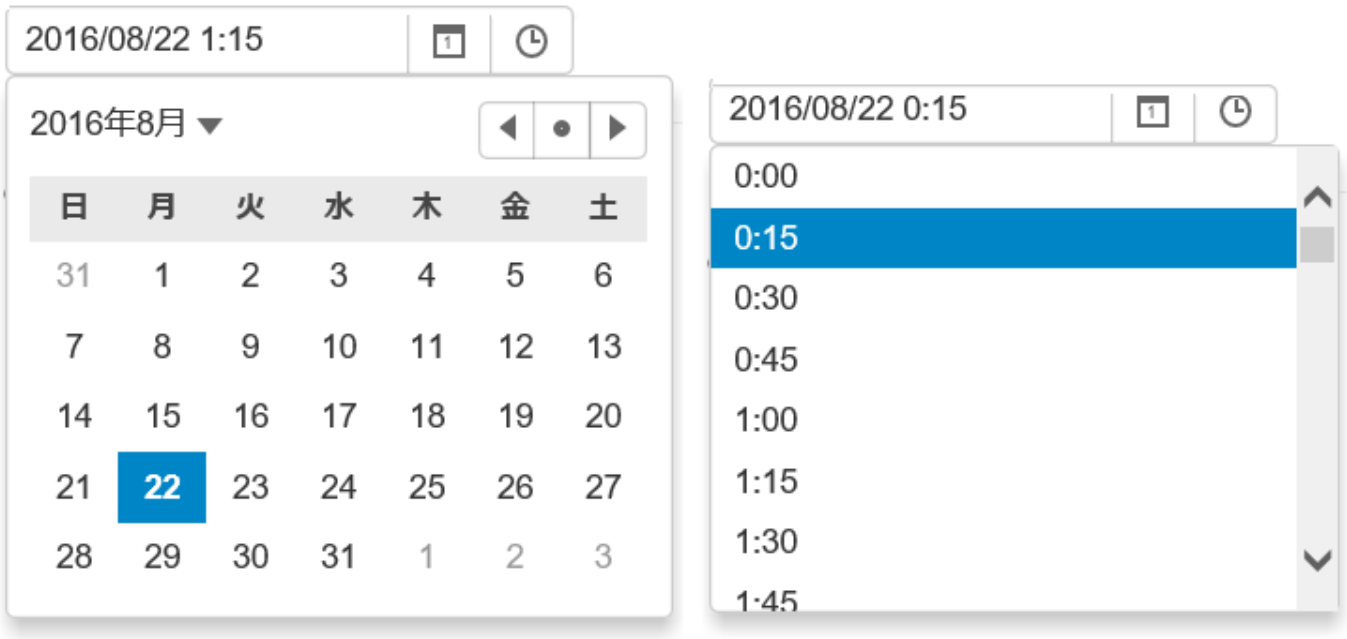
Razor copyCode

```
@{
    var timeList = ViewBag.TimeList as List<object>;
}
<div>
    <label>時刻を選択</label>
    @(Html.C1().InputTime().Bind(timeList).Value(DateTime.Parse("11:35")))
</div>
```

InputDateTime

InputDateTimeコントロールを使用すると、ユーザーは、Globalizeクラスでサポートされている任意の書式を使用して日付/時刻を入力できます。また、ドロップダウンボックスに表示されたカレンダーから日付を選択したり、ドロップダウンリストから時刻を選択することもできます。コントロールの**Min**プロパティと**Max**プロパティを設定することで、ユーザーが入力できる値の範囲を定義できます。

Copyright © GrapeCity, Inc. All rights reserved.



主な特長

InputDateTime には主に次の便利な機能があります。

- Value
  ユーザーが現在選択している日時を表します。
- Max
  ユーザーが入力できる最遅時刻を設定します。
- Min
  ユーザーが入力できる最早時刻を設定します。
- Format
  このプロパティを使用して、時刻をさまざまな書式で表示できます。
- TimeStep
  このプロパティを使用して、ドロップダウンリストのエントリ間の分数を指定できます。

クイックスタート

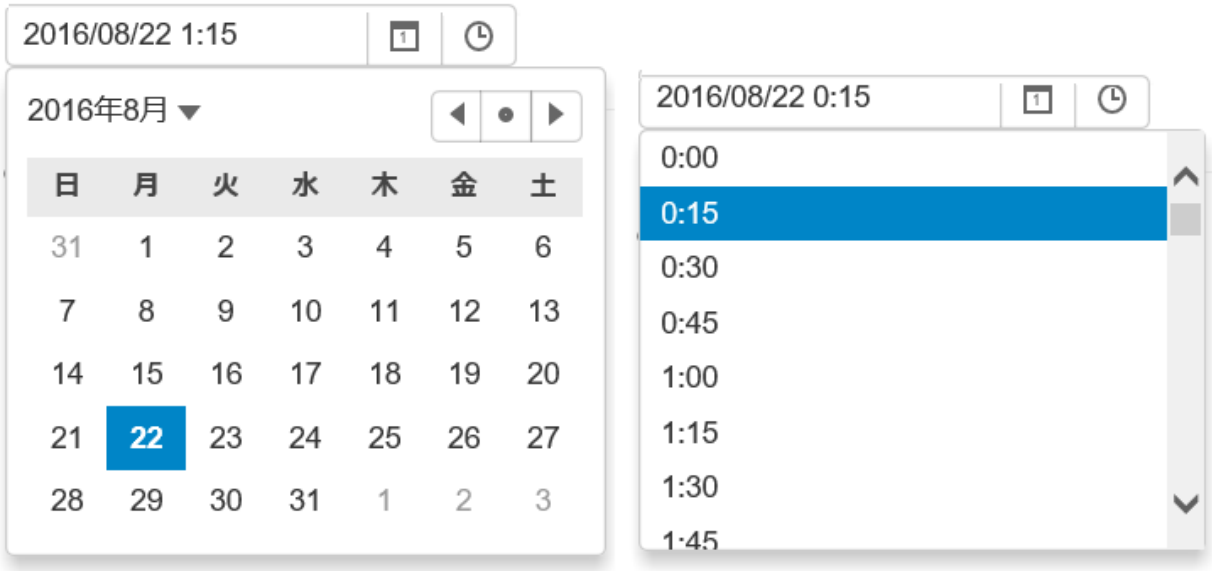
このトピックでは、アプリケーションにInputDateTimeコントロールを追加する方法を示します。ASP.NET MVC コントロールの追加方法の詳細については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順1：MVC アプリケーションの作成**
- **手順2：新しいコントローラーおよびビューの追加**
- **手順3：InputDateTimeコントロールの追加**
- **手順4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の**InputDateTime**を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



手順1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：Index`Controller`）。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

コントローラーのビューの追加

1. メソッド`Index()`内にカーソルを置きます。
2. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
3. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor（CSHTML）**であることを確認します。
4. **[追加]**をクリックしてコントローラーのビューを追加します。

手順3：コントロールの追加

1. `Index.cshtml`ファイルのデフォルトコードを次のコードに置き換えて、**InputDateTime**コントロールを初期化します。

Razor

```
@{
  var today = DateTime.Now.Date;
  var minDate = new DateTime(today.Year, 1, 1);
  var maxDate = new DateTime(today.Year, 12, 31);
}
<div>
@(Html.C1().InputDateTime().Id("idtInputDateTime").Value(today).Min(minDate).Max(maxDate))
</div>
```

手順4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

Copyright © GrapeCity, Inc. All rights reserved.

ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**Index/index**）。または、ビューをホームページにリンクします。

先頭に戻る

ListBox

ListBox コントロールは、プレーンテキストまたは HTML を含む値のリストを表示し、ユーザーがキーボードまたはマウスで項目を選択できるようにします。ListBox をモデルに連結して、項目のリストに値を設定することができます。



主な特長

ListBox には主に次の便利な機能があります。

- **DisplayPath**
  項目のビジュアル表現に使用される名前を指定するには、DisplayPath プロパティを使用します。
- **IsContentHtml**
  項目にプレーンテキストと HTML のどちらが含まれるかを指定します。
- **SelectedItem**
  ドロップダウンリストで現在選択されている項目を表します。
- **SelectedIndex**
  SelectedItem.のインデックスを示します。
- **SelectedValuePath**
  SelectedValue および **SelectedItem**.の取得に使用されるプロパティの名前を表します。
- **SelectedValue**
  SelectedValuePath を使用して取得される **SelectedItem** の値を示します。
- **MaxHeight**
  リストの最大高さを設定するには、**MaxHeight** プロパティを使用します。
- **VirtualizationThreshold**
  Use the **VirtualizationThreshold** property to set the minimum number of rows and/or columns required to enable virtualization in the ListBox control. For more information, see 仮想化.

クイックスタート

このトピックでは、アプリケーションに **ListBox** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コン

Copyright © GrapeCity, Inc. All rights reserved.

トロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：ListBox コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **ListBox** を示しています。この例では、ComboBox の「クイックスタート」セクションで追加した Cities.cs モデルを使用します。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`Default1Controller`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`Default1Controller`）をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3：コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**ListBox** コントロールを初期化します。

C#	copyCode
`@{`	

Copyright © GrapeCity, Inc. All rights reserved.

```
    List<string> cities = Cities.GetCities();
}
    <div>
        <label>Select your city:</label>
    </div>
    <div>
        @(Html.C1().ListBox()
        .Id("list")
        .Bind(cities)
        .Width(180).Height(200)
        .CaseSensitiveSearch(true)
        )
    </div>
```

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]→[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例: http://localhost:1234/**Default1/index**)、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

ListBoxの使用

This section comprises all the features and auxiliary functionalities offered by ListBox.

カスタムの ListBox
: Learn how to use ItemTemplates to customize the ListBox control.

大文字と小文字を区別する検索
: Learn how to perform case-sensitive search in ListBox control.

カスタムの ListBox

ItemTemplates を使用してコントロール内のカスタムコンテンツを指定することで、ListBox コントロールをカスタマイズすることができます。ListBox のテンプレート内で ComponentOne MVC コントロールを使用することもできます。

次の図は、ItemTemplates を適用した **ListBox** を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、ItemTemplates を使用して ListBox コントロールをカスタマイズする方法を示します。

コードの場合

ListBoxController.cs

Razor copyCode

```
public ActionResult Index()
{
    var list = MVCFlexGrid_JP.Models.CustomerOrder.GetOrderData(100).ToList();
    return View(list);
}
```

ListBox.cshtml

C# copyCode

```
@model List<MVCFlexGrid_JP.Models.CustomerOrder>

<style>
    .badge {
        color: white;
        background-color: darkred;
        border-radius: 10px;
        padding: 1px 10px;
    }

    .label {
        color: black;
        background-color: orange;
        border-radius: 10px;
        padding: 1px 10px;
    }
</style>

<script id="template1" type="text/template">
    <span>
        <span class="label">{{Product}}</span>
```


Copyright © GrapeCity, Inc. All rights reserved.

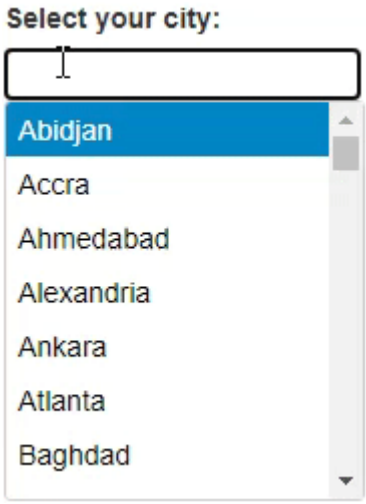
```
        <span class="badge">{{Count}}</span>
    </span>
</script>

<div>
    <label>Custom HTML</label>
    @(Html.C1().ListBox()
        .Bind(Model)
        .ItemTemplateId("template1")
        .Width(300).Height(250)
    )
</div>
```

先頭に戻る

大文字と小文字を区別する検索

By default, the search in ListBox is case-insensitive. However, there might be a scenario where you need case-sensitive search in ListBox. With case-sensitive search, ListBox allows you to have more granular control over searched items. You can achieve the granularity in search by setting **CaseSensitiveSearch** property of the **ListBox** class to **true**. The **CaseSensitiveSearch** property determines whether the searches performed while the user types should be case-sensitive which makes the search more refined.



The following example shows how you can use the case-sensitive search in the ListBox control. The example uses Cities.cs model added in the Quick Start topic.

Controller

ListBoxSearchController.cs copyCode

```
public ActionResult Index()
{
    return View();
}
```

View for the Controller

Index.cshtml copyCode

Copyright © GrapeCity, Inc. All rights reserved.

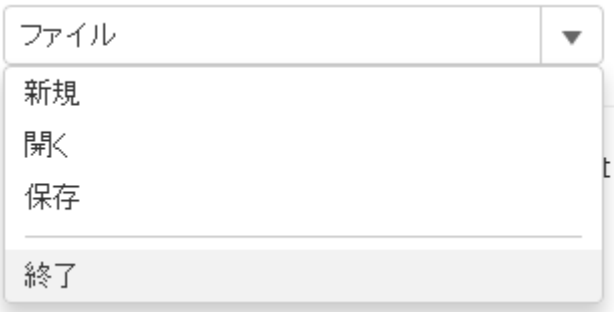
```
@{
    List<string> cities = Cities.GetCities();
}
<div>
    <label>Select your city:</label>
</div>
<div>
    <input type="text" id="searchValue" />
</div>
<div>
    @(Html.C1().ListBox()
        .Id("list")
        .Bind(cities)
        .Width(180).Height(200)
        .CaseSensitiveSearch(true)
    )
</div>

@section Scripts{
    <script>
        c1.documentReady(function () {
            document.getElementById("searchValue").addEventListener("keyup", function
(e) {
                var listbox = wijmo.Control.getControl('#list');
                listbox.selectedIndex = -1;
                listbox._search = this.value;
                listbox.selectedIndex = listbox._findNext();
            });
        });
    </script>
}
```

Menu

Menu コントロールは、1つのテキスト要素とドロップダウンリストを表示します。ドロップダウンリストには、ユーザーがクリックすることで呼び出すことができるコマンドが含まれます。Menu は **ComboBox** を継承しているため、ComboBox と同様に、**ItemsSource** プロパティを使用して値やスタイルを設定することができます。Menu コントロールには、ユーザーがメニューから項目の1つを選択したときに発生する ItemClicked イベントが追加されています。イベントハンドラで Menu コントロールを調べて、クリックされた項目を特定します。

Menu コントロールは、ユーザーがメニューから項目の1つを選択したときに発生する ItemClicked イベントもサポートしています。イベントハンドラでこのコントロールを調べて、クリックされた項目を特定します。



Copyright © GrapeCity, Inc. All rights reserved.

主な特長

Menu には主に次の便利な機能があります。

- **ItemsSource**
  選択される項目を含む配列または ICollectionView オブジェクトを設定するには、ItemSource プロパティを使用します。

- **Placeholder**
  コントロールが空のときにヒントとして表示される文字列を設定するには、Placeholder プロパティを使用します。

- **SelectedIndex**
  SelectedItem のインデックスを指定するには、SelectedIndex プロパティを使用します。

- **SelectedValue**
  SelectedValuePath を使用して取得される SelectedItem の値を設定するには、SelectedVaue プロパティを使用します。

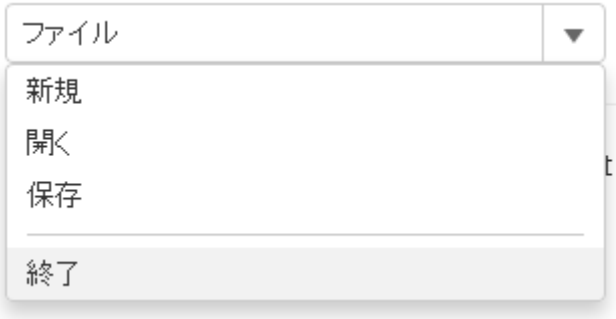
クイックスタート

このトピックでは、アプリケーションに **Menu** コントロールを追加する方法を示します。ComponentOne ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順 1：MVC アプリケーションの作成**
- **手順 2：新しいコントローラーおよびビューの追加**
- **手順 3：Menu コントロールの追加**
- **手順 4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の **Menu** を示しています。



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]** ダイアログで、次の手順を実行します。

Copyright © GrapeCity, Inc. All rights reserved.

1. コントローラーの名前を設定します(例:`Default1Controller`)。
2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
3. **[追加]**をクリックします。

4. アプリケーションに新しいコントローラーが追加されます。

ビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー(例:`Default1Controller`)をダブルクリックして開きます。
2. メソッド `Index()` 内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順 3:コントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml` をダブルクリックして開きます。
3. `Index.cshtml` ファイルのデフォルトのコードを次のコードに置き換えて、**Menu** コントロールを初期化します。

C# copyCode

```
<div>
    @(Html.C1().Menu().Header("File")
        .Width(300)
        .MenuItems(items => items
            .Add(item => item.Header("New"))
            .Add(item => item.Header("Open"))
            .Add(item => item.Header("Save"))
            .Add(item => item.IsSeparator(true))
            .Add(item => item.Header("Exit"))
        )
    )
</div>
```

手順 4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]** キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例:http://localhost:1234/**Default1/index**)、ビューを表示します。または、ビューをホームページにリンクします。

先頭に戻る

Menuの使用

This section comprises all the features and auxiliary functionalities offered by Menu.

コンテキストメニュー
: Learn how to use menu control as a Context Menu with FlexGrid.

InputNumber を使用した Menu
: Learn how to bind Menu to an InputNumber control.

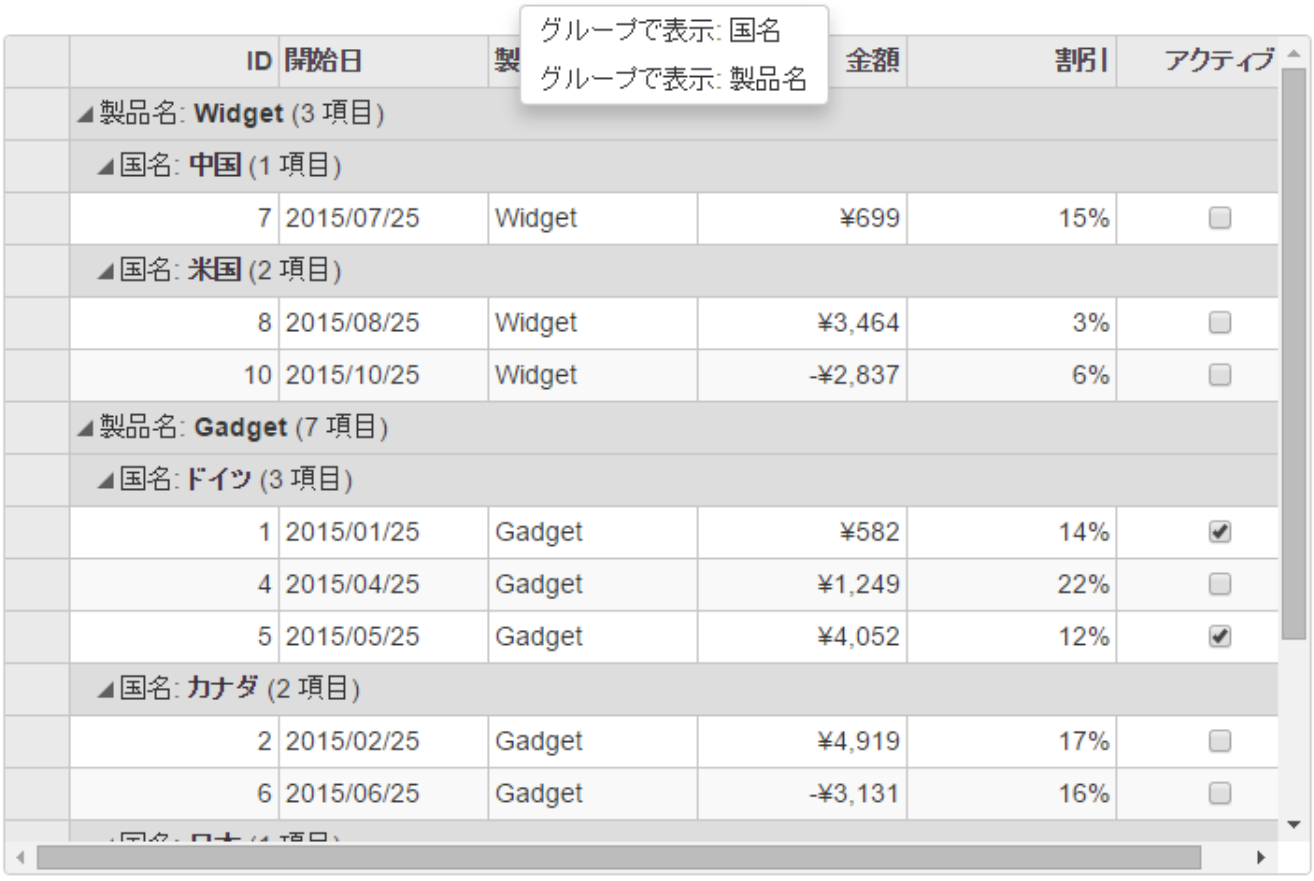
コンテキストメニュー

Copyright © GrapeCity, Inc. All rights reserved.

Menuコントロールは、**FlexGrid**、**ListBox**などの他のMVCコントロールと一緒にコンテキストメニューとして使用できます。ユーザーがコントロール内で右クリックすると、ユーザーによって追加されたカスタムフィールドを含むコンテキストメニューとしてMenuが表示されます。さまざまな項目をMenuに追加することで、FlexGridの**フィルタ処理**、**ソート**、**グループ化**などの各種操作ができます。Menuコントロールと他のMVCコントロールとのシームレスな統合により、ユーザーは項目を追加し、コントロールのIDをMenuの**Owner**プロパティに割り当てて、コントロールのコンテキストメニューをカスタマイズできます。

この例では、MenuはFlexGridコントロールの**グループ化**を有効にするコンテキストメニューとして使用されます。**GroupBy: Country**と**GroupBy: Product**という2つのフィールドがMenuに追加されることで、FlexGridのデータがCountry列とProduct列でグループ化されます。ユーザーは、グリッド内で右クリックしてコンテキストメニューから項目を選択できます。

次の図は、Menuコントロールをコンテキストメニューとして追加した後の**FlexGrid**を示しています。



次のコード例は、メニューコントロールをFlexGridでContext Menuとして使用する方法を示します。

コードの場合

ContextMenuController.cs

Razor copyCode

```
    public ActionResult Index()
    {
        return View(Sale.GetData(10));
    }
}
```

ContextMenu.cshtml

C# copyCode

```
@using MVCFlexGrid_JP.Models;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@using C1.Web.Mvc.Grid


@model IEnumerable<Sale>

<style>
    .flexGrid {
        margin: 10px;
        padding: 20px;
        color: white;
        display: inline-block;
    }
</style>
<script>
    //CollectionViewを使用するグリッドのソート関数。
    function setGrouping(arg) {
        var grid = wijmo.Control.getControl("#Sales");
        var cv = grid.collectionView;
        var gd = cv.groupDescriptions;
        gd.push(new wijmo.collections.PropertyGroupDescription(arg));

    }
</script>

<div class="multi-grid">
    @(Html.C1().FlexGrid<Sale>()
    .AutoGenerateColumns(false)
    .Id("Sales")
    .Height(450)
    .Width(700)
    .AllowAddNew(true)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Cell)
    .CssClass("grid")
    .Bind(Model)

 // FlexGridで列が表示されるように定義します。

     .Columns(bl =>
       {
           bl.Add(cb => cb.Binding("ID"));
           bl.Add(cb => cb.Binding("開始日"));
           bl.Add(cb => cb.Binding("製品名"));
           bl.Add(cb => cb.Binding("金額").Format("c"));
           bl.Add(cb => cb.Binding("割引").Format("p0"));
           bl.Add(cb => cb.Binding("アクティブ"));
       })
    )

</div>
//C1Menuコントロールを初期化します。
@(Html.C1().Menu().Header("グループで表示")
        .Id("ctxMenu")
```


Copyright © GrapeCity, Inc. All rights reserved.

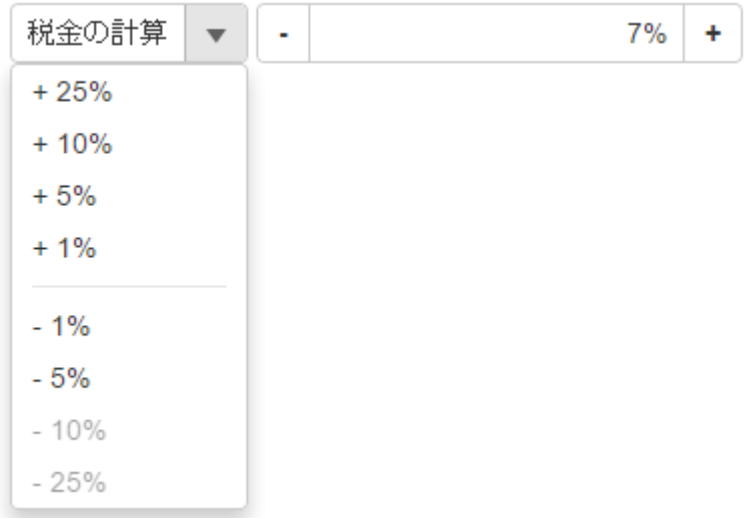
```
        .Command("setGrouping")
        //コンテキストメニューにて表示する項目を追加します。
        .MenuItems(items =>
        {
            items.Add("グループで表示： 国名", "国名");
            items.Add("グループで表示： 製品名", "製品名");


        })
    .CssStyle("display", "none")
    .Owner("#Sales")
)
```

InputNumber を使用した Menu

Menu コントロールは、クイックスタートで示した文字列と同様にさまざまなタイプの値を簡単に追加したり、別のコントロールと連結することができます。このセクションでは、Menu コントロールを InputNumber と組み合わせて使用し、InputNumber コントロールで選択された値に基づいて Menu の値を更新する方法について説明します。

次の図は、InputNumber に連結された **Menu** を示しています。



次のコード例は、Menu を InputNumber コントロールに連結する方法を示します。

コードの場合

MenuController.cs

Razor	copyCode

```
public ActionResult Feature()
{
    return View();
}
```

Menu.cshtml

C#	copyCode

```
<script>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    function execute(arg) {
        var inputNumber = wijmo.Control.getControl("#mInputNumber");
        // 引数を数字に変換します
        arg = wijmo.changeType(arg, wijmo.DataType.Number);
        // 変換が成功かどうかを確認します
        if (wijmo.isNumber(arg)) {
            // 値を更新します
            inputNumber.value += arg;
        }
    }
    function canExecute(arg) {
        var inputNumber = wijmo.Control.getControl("#mInputNumber");
        // 引数を数字に変換します
        arg = wijmo.changeType(arg, wijmo.DataType.Number);
        // 変換が成功かどうかを確認しますl
        if (wijmo.isNumber(arg)) {
            var val = inputNumber.value + arg;
            // 更新した値が有効であるかどうかを確認します
            return val >= 0 && val <= 1;
        }
        return false;
    }
</script>
<div>
    @(Html.C1().Menu().Header("税金の計算")
        .Command("execute", canExecute: "canExecute")
        .MenuItems(items =>
        {
            items.Add("+ 25%", 0.25);
            items.Add("+ 10%", 0.10);
            items.Add("+ 5%", 0.05);
            items.Add("+ 1%", 0.01);
            items.AddSeparator();
            items.Add("- 1%", -0.01);
            items.Add("- 5%", -0.05);
            items.Add("- 10%", -0.10);
            items.Add("- 25%", -0.25);
        })
    )
    @(Html.C1().InputNumber().Id("mInputNumber")
        .Value(0.07).Step(0.05).Format("p0").Min(0).Max(1)
    )
</div>
```

MultiAutoComplete

MultiAutoCompleteコントロールでは、カスタムオブジェクトまたは簡単な文字列を含むリストから複数の項目を選択することができます。コントロールのフィールドに何らかのテキストを入力すると、MultiAutoCompleteが入力された文字列と一致する項目の検索を開始し、可能性がある結果のリストを表示します。リストからいずれかの項目を選択すると、その項目がコントロールのフィールドに表示されます。

Copyright © GrapeCity, Inc. All rights reserved.



このコントロールでは、同様の方法で複数の項目を選択できます。また、選択できる項目の数を制限するオプションも提供されています。

主な特長

MultiAutoCompleteコントロールには、直感的で本格的な外観のアプリケーションを構築するためのさまざまな機能が提供されています。

- **最大選択項目数**
  MultiAutoCompleteでは、コントロールで選択できる項目の最大数を設定できます。このプロパティを**null**に設定すると、MultiAutoCompleteで項目をいくつでも選択できます。

- **最大表示項目数**
  MultiAutoCompleteでは、ドロップダウンに表示される項目の数を制御できます。

- **ヘッダーパスの設定**
  MultiAutoCompleteでは、ドロップダウンリストに表示された値のうち、入力要素に表示されている値を強調表示することができます。

- **動的リスト項目**
  MultiAutoCompleteでは、ユーザーが文字列を入力するときに動的にコントロールにリスト項目を提供するための関数を設定できます。

クイックスタート：MultiAutoComplete へのデータの追加

このトピックでは、MVC WebアプリケーションにMultiAutoCompleteコントロールを追加し、そこにモデル連結を使用してデータを追加する方法について説明します。このトピックは次の4つの手順で構成されます。

- **手順1：MVC アプリケーションの作成**
- **手順2：MultiAutoCompleteのデータソースの作成**
- **手順3：MultiAutoCompleteコントロールの追加**
- **手順4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のMultiAutoCompleteコントロールを示しています。

次のMultiAutoCompleteコントロールでは、データソースとして国名リストを使用しています。入力要素に「ind」と入力すると、国名に「ind」の文字列が含まれる国のリストが示されます。

Copyright © GrapeCity, Inc. All rights reserved.



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2:MultiAutoCompleteのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`Countries.cs`)。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`Countries.cs`モデルに追加します。ここでは、**Countries**クラスを使用して国名リストを表しています。

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class Countries
    {
        public static List<string> GetCountries()
        {
            return new List<string>
            {
                "Afghanistan", "Albania", "Algeria", "American Samoa",
"Andorra", "Angola", "Anguilla", "Antigua", "Argentina", "Armenia",
        "Aruba", "Australia", "Austria", "Azerbaijan", "Bahamas", "Bahrain",
"Bangladesh", "Barbados", "Belarus", "Belgium", "Belize",
        "Benin", "Bermuda", "Bhutan", "Bolivia", "Bonaire", "Bosnia",
"Botswana", "Brazil", "Brunei", "Bulgaria", "Burkina Faso", "Burundi",
        "Cambodia", "Cameroon", "Canada", "Canary Islands", "Cape Verde",
"Cayman Islands", "Central African Republic", "Chad", "Channel Islands",
        "Chile", "China", "Christmas Island", "Cocos Island", "Colombia",
"Comoros", "Congo", "Cook Islands", "Costa Rica", "Cote D'Ivoire",
        "Croatia", "Cuba", "Curacao", "Cyprus", "Czech Republic", "Denmark",
"Djibouti", "Dominica", "Dominican Republic", "East Timor", "Ecuador",
        "Egypt", "El Salvador", "Equatorial Guinea", "Eritrea", "Fiji",
"Finland",
        "France", "French Guiana", "French Polynesia", "French Southern Ter",
"Gabon", "Gambia", "Georgia", "Germany",
        "Great Britain", "Greece", "Greenland", "Grenada", "Guadeloupe", "Guam",
```


Copyright © GrapeCity, Inc. All rights reserved.

```
"Guatemala", "Guinea", "Guyana", "Haiti", "Honduras",
        "Hong Kong", "Hungary", "Iceland", "India", "Indonesia", "Iran", "Iraq",
"Ireland", "Isle of Man", "Israel", "Italy",  "Japan",
        "Qatar", "Republic of Montenegro", "Republic of Serbia", "Reunion",
"Romania", "Russia", "Rwanda",
        "St Helena", "St Kitts-Nevis", "St Lucia", "St Maarten", "Saipan",
"Samoa", "Samoa American", "San Marino", "Saudi Arabia", "Scotland",
        "Senegal", "Serbia", "Seychelles", "Sierra Leone", "Singapore",
"Slovakia", "Slovenia", "Solomon Islands", "Somalia", "South Africa",
        "Spain", "Sri Lanka", "Sudan", "Suriname", "Swaziland", "Sweden",
"Switzerland",
        "Thailand", "Turkey", "Turkmenistan", "Tuvalu", "Uganda",
        "Ukraine", "United Arab Emirates", "United Kingdom", "United States of
America", "Uruguay", "Uzbekistan", "Vanuatu", "Vatican City State",
        "Venezuela", "Vietnam", "Zimbabwe"
            };
        }
    }
```

先頭に戻る

手順3：MultiAutoCompleteコントロールの追加

MultiAutoCompleteコントロールをアプリケーションに追加する手順は、次のとおりです。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`MultiAutoCompleteController`）。
    3. **[追加]**をクリックします。
4. 次に示すように参照を追加します。

C#

```
using <ApplicationName>.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

5. **Index()**メソッドを次のメソッドに置き換えます。

MultiAutoCompleteController.cs

```
public ActionResult Index()
{
   return View(Models.Countries.GetCountries());
}
```

コントローラーのビューの追加

ビュー内で、MultiAutoCompleteコントロールのインスタンスを作成し、それを**.Bind**プロパティを使用してデータソースに連結します。また、**MaxSelectedItems**プロパティを使用して、選択できる項目の最大数を定義します。

Copyright © GrapeCity, Inc. All rights reserved.

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`MultiAutoCompleteController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加したら、次のコードをコピーして**Index.cshtml**内に貼り付けます。

C# copyCode

```
@model List<string>

@section Styles{
    <style>
        .highlight {
            background-color: #ff0;
            color: #000;
        }
    </style>
}
<br />
<p>Type in a country name</p>
@(Html.C1().MultiAutoComplete().Bind(Model).CssMatch("highlight")
    .SelectedIndexes(1)
    .Width(300)
    //最大選択値を4に設定します
    .MaxSelectedItems(4))
```

手順4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http://localhost:1234/**MultiAutoComplete/Index**)。

先頭に戻る

機能

MultiAutoCompleteコントロールの使用方法を理解するために役立つ概念について説明します。

このセクションには、次の情報が含まれます。

インクリメンタル検索
: MultiAutoCompleteでカスタムアクションを使用してインクリメンタル検索を実装する方法を説明します。

複合型
: MultiAutoCompleteで複合型データを使用する方法を説明します。

複合型

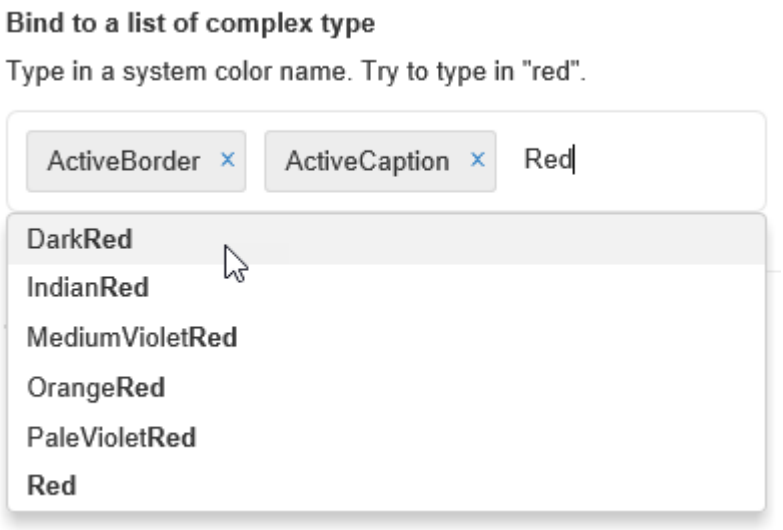
複合データ型(複合型)は、他の既存のデータ型の複合です。ユーザー定義のデータ型と比べて複合データ型が優れている主な点は、ユーザーが複合データ型のコンポーネントに個別にアクセスして操作できることです。MultiAutoCompleteでは、ComplexTypeデータのリストを連結できます。それには、**DisplayMemberPath**プロパティと**SelectedValuePath**プロパティを

Copyright © GrapeCity, Inc. All rights reserved.

利用します。

以下の例では、MultiAutoCompleteで、**System.Drawing**名前空間に含まれる**KnownColor列挙**を使用しています。KnownColor列挙には、システムによって認識されるすべてのシステムカラーのリストが含まれます。この列挙値にアクセスするには、**NamedColor**モデルを作成し、2つのプロパティ、**Name**と**Value**を定義します。これらのプロパティを**View**コードで使用して、列挙値にアクセスします。

次の図に、**SelectedValuePath**プロパティと**DisplayMemberPath**プロパティを設定した後のMultiAutoCompleteコントロールを示します。



次のコード例は、AutoCompleteで複合型データの連結を有効にする方法を示します。

MultiAutoCompleteのカスタムデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`NamedColor.cs`）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`NamedColor.cs`モデルに追加します。ここでは、2つの文字列変数**Name**および**Value**を定義してKnownColor列挙の値にアクセスしています。

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace MultiAutoComplete.Models
{
    public class NamedColor
    {
        public string Name { get; set; }
        public string Value { get; set; }
    }
}
```

先頭に戻る

新しいコントローラーの作成

AutoCompleteController.cs

Copyright © GrapeCity, Inc. All rights reserved.

Razor

```
using System;
using System.Drawing;
using System.Linq;
using System.Web.Mvc;
using <ApplicationName>.Models;

namespace MultiAutoComplete.Controllers
{
    partial class MultiAutoCompleteController
    {
        public ActionResult ComplexType()
        {
            var list = GetSystemColors();
            return View(list);
        }

        private static NamedColor[] GetSystemColors()
        {
            return Enum.GetValues(typeof(KnownColor))
                .Cast<KnownColor>()
                .Select(c => new NamedColor
                {
                    Name = c.ToString(),
                    Value = "#" +
Color.FromKnownColor(c).ToArgb().ToString("X8").Substring(2)
                })
                .ToArray();
        }
    }
}
```

ビュー - AutoComplete.cshtml

Razor

```
@model IEnumerable<MvcExplorer.Models.NamedColor>

<div>
    <label>複合型リストへの連結</label>
    <p>システムカラー名をキー入力します。「red」と入力してみます。</p>
    @(Html.C1().MultiAutoComplete()
        .Bind(Model)
        .DisplayMemberPath("Name")
        .SelectedIndexes(0,1)
    )
</div>
```

先頭に戻る

インクリメンタル検索

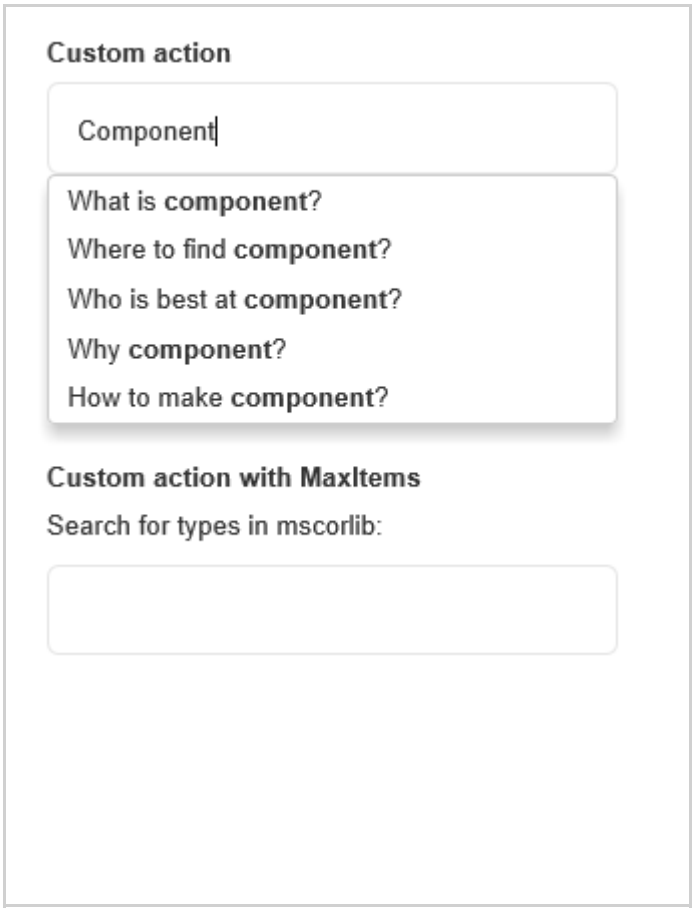
インクリメンタル検索（リアルタイム候補）を使用すると、ユーザーは逐次的にテキストを検索およびフィルタ処理することができ

Copyright © GrapeCity, Inc. All rights reserved.

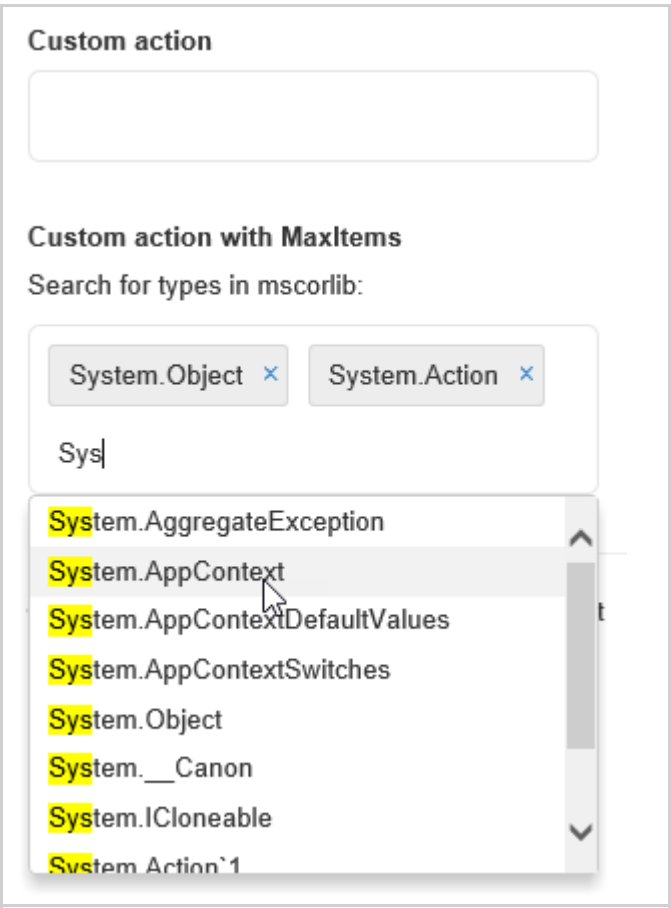
ます。テキストを入力すると、そのテキストと一致する可能性がある候補が1つ以上検索され、瞬時にユーザーに提示されます。ユーザーは、提示されたリストから関連性の高いオプションを選択できます。MultiAutoCompleteでは、コントローラーのカスタム**ActionResult**を作成することができ、これを使用してインクリメンタル検索を実装できます。

ActionResultで実行するカスタムアクションを指定できます。たとえば、以下のコードでは、ユーザーが何らかのテキストを入力した後にデータをフィルタ処理するカスタムアクションと、mscorlib.dllのオブジェクトモデルの名前空間を取得するカスタムアクションを作成しています。

次の図に、カスタムアクションメソッドを備えたMultiAutoCompleteを示します。



カスタムアクション



MaxItemsを使用したカスタムアクション

次のコード例は、MultiAutoCompleteでカスタムアクションを使用する方法を示します。

コードの場合

MultiAutoCompleteController.cs

Razor

```
using System.Linq;
using System.Web.Mvc;
using C1.Web.Mvc.Serialization;

namespace MultiAutoComplete.Controllers
{
    partial class MultiAutoCompleteController
    {
        public ActionResult CustomAction()
        {
            return View();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }

        public ActionResult Heuristic(string query, int max)
        {
            var prefix = new[] { "What is ", "Where to find ", "Who is best at ",
"Why ",
                        "How to make " };
            return this.C1Json(prefix.Select(f => f + query + "?").ToList(),
                behavior: JsonRequestBehavior.AllowGet);
        }

        public ActionResult TypesInMscorlib(string query, int max)
        {
            var types = typeof(object).Assembly.GetTypes();
            return this.C1Json(types
                .Where(t => t.FullName.ToUpper().Contains(query.ToUpper()))
                .Select(t => t.FullName)
                .Take(max).ToList(),
                behavior: JsonRequestBehavior.AllowGet);
        }
    }
}
```

MultiAutoComplete.cshtml

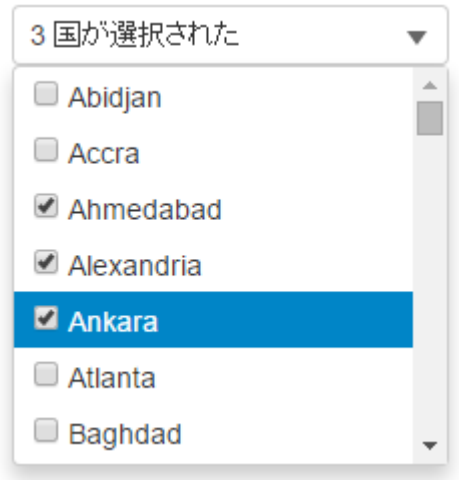
Razor

```
@section Styles{
    <style>
        .highlight {
            background-color: #ff0;
            color: #000;
        }
    </style>
}
<br />
<div>
    <label>カスタムアクション</label><br />
    @(Html.C1().MultiAutoComplete().Width(300)
        .ItemsSourceAction(Url.Action("Heuristic"))
    )
</div>
<br />
<div>
    <label>MaxItemsを使用したカスタムアクション</label><br />
    <p>mscorlibで型を検索します:</p>
    @(Html.C1().MultiAutoComplete().Width(300)
        .MaxItems(10).CssMatch("highlight")
        .ItemsSourceAction(Url.Action("TypesInMscorlib"))
    )
</div>
```


Copyright © GrapeCity, Inc. All rights reserved.

Multi-select

Multi-selectコントロールを使用して、複数選択が可能なドロップダウンリストをアプリケーションに追加できます。コントロールは、カスタムオブジェクトまたは簡単な文字列のコレクションに連結でき、これらはドロップダウンに表示されます。これは、各ListBox項目に対応するチェックボックスを制御する**CheckedMemberPath**プロパティを使用します。
コントロールは、自由にカスタマイズできるヘッダーを持ちます。ヘッダーには、リストから選択された項目がデフォルトで2つまで表示されます。リストから3つ以上の項目が選択された場合、ヘッダーには選択項目の数が表示されます。ただし、このデフォルトの挙動を制御して、ヘッダーに表示する選択項目の最大数を調整し(**MaxHeaderItems**)、項目が選択されていない場合に表示されるメッセージをカスタマイズしたり、項目数を表示する書式文字列を設定(**HeaderFormatter**)することもできます。



主な特長

MultiSelect には主に次の便利な機能があります。

- **CheckedMemberPath**
  各項目の横に配置されるチェックボックスを制御するプロパティ名を設定します。
- **ItemsSource**
  選択元になる項目を含む配列またはICollectionViewオブジェクトを指定します。
- **MaxHeaderItems**
  ヘッダーに表示する最大選択項目数の数を指定します。
- **HeaderFormat**
  オンに設定された項目数が in **MaxHeaderItems** で指定されたカウントよりも多いときに、ヘッダーコンテンツの書式文字列をカスタマイズします。

クイックスタート

このトピックでは、アプリケーションに**multi-select**コントロールを追加する方法を示します。ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順1:MVC アプリケーションの作成**
- **手順2:新しいコントローラーおよびビューの追加**
- **手順3:Multi-selectコントロールの追加**

Copyright © GrapeCity, Inc. All rights reserved.

- **手順4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の**Multi-select**コントロールを示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

メモ: この例では、「AutoCompleteクイックスタート」で追加された**Countries.cs**モデルを使用します。

手順2:新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します(例:`QuickStartController`)。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. 次の参照を追加します。

C# copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using MVCFlexGrid_JP.Models;
```

5. メソッドIndex()を次のメソッドに置き換えます。

C# copyCode

```
    public ActionResult Index()
    {
        ViewBag.Cities = MVCFlexGrid_JP.Models.Cities.GetCities();
        return View();
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

アプリケーションに新しいコントローラーが追加されます。

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`DefaultController`）をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順3：Multi-selectコントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml`をダブルクリックして開きます。
3. `Index.cshtml`ファイルのデフォルトコードを次のコードに置き換えて、**Multi-select**コントロールを初期化します。

Razor copyCode

```
@{
    List<string> cities = ViewBag.Cities;
}
<script>
    var checkedItemsChanged = function (sender, e) {
        var i, result = document.getElementById("result"),
            items = sender.checkedItems;

        result.innerHTML = "";

        for (i = 0; i < items.length; i++) {
            result.innerHTML += "<span>" + (i + 1) + ". " + items[i] + "<\/span>
<br>";
        }
    }
</script>
<div>
    @*<label>国を選択します。:</label>*@@(Html.C1().MultiSelect()
.Name("cities")
.Id("multiselect")
.Bind(cities)
.Placeholder("国を選択します。")
.HeaderFormat("{count} 国が選択された")
.OnClientCheckedItemsChanged("checkedItemsChanged"))
    @*<label>選択結果:</label>*@
    <div id="result"></div>
</div>
```

手順4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**Default/Index**）。または、ビューをホームページにリンクします。

Copyright © GrapeCity, Inc. All rights reserved.

先頭に戻る

MultiSelectの使用

This section comprises all the features and auxiliary functionalities offered by MultiSelect.

オブジェクトのコレクションに連結するMultiSelect
Learn how to bind a MultiSelect control to a collection of objects.

オブジェクトのコレクションに連結するMultiSelect

MultiSelectコントロールのドロップダウンにカスタムオブジェクトを表示することができます。これを実現するには、SelectedValuePath プロパティ、DisplayMemberPath プロパティ、およびCheckedMemberPathプロパティを使用して、コントロールをオブジェクトのコレクションと連結する必要があります。
次の図は、C1NWind.mdf データベース内の Products テーブルのコレクションと連結しているMultiSelectコントロールを示しています。



次のコード例は、MultiSelectコントロール を オブジェクトのコレクションに連結する方法を示します。

コードの場合

ComplexTypeController.cs

C# copyCode

```
    private C1NWindEntities db = new C1NWindEntities();
    public ActionResult Index()
    {
        return View(db);
    }
}
```

ComplexType.cshtml

Razor copyCode

```
@model C1NWindEntities
```


Copyright © GrapeCity, Inc. All rights reserved.

```
<br />
<div>
    <label>オブジェクトの配列と連結した MultiSelect:</label>

    @(Html.C1().MultiSelect()
    .Bind(Model.Products)
    .Name("products")
    .DisplayMemberPath("ProductName")
    .SelectedValuePath("ProductID")
    .CheckedMemberPath("Discontinued")
    )
</div>
```

先頭に戻る

MultiSelectフィルター

MultiSelect is a simple drop-down control with checkable options which allows you to select or deselect an item with a single click. However, it might become a tedious task to select items from a long list. So, MultiSelect provides a search element that can filter the items from the list as you type. It provides the **ShowFilterInput** property that can be used to display a filter input above the items to filter the items displayed in the control. In addition, MultiSelect also provides the **CheckOnFilter** property that determines whether the control should automatically select all the filtered items when the filter text changes.



The following example shows how you can use the **ShowFilterInput** and **CheckOnFilter** properties to filter items from a smaller set of options.

Add Model

Countries.cs copyCode

```
public class Countries
{
    public static List<string> GetCountries()
    {
        return new List<string>
        {
            "Afghanistan", "Albania", "Algeria", "American Samoa", "Andorra",
"Angola", "Anguilla", "Antigua", "Argentina", "Armenia",
    "Aruba", "Australia", "Austria", "Azerbaijan", "Bahamas", "Bahrain",
"Bangladesh", "Barbados", "Belarus", "Belgium", "Belize",
    "Benin", "Bermuda", "Bhutan", "Bolivia", "Bonaire", "Bosnia", "Botswana",
"Brazil", "Brunei", "Bulgaria", "Burkina Faso", "Burundi",
    "Cambodia", "Cameroon", "Canada", "Canary Islands", "Cape Verde", "Cayman
```


Copyright © GrapeCity, Inc. All rights reserved.

```
Islands", "Central African Republic", "Chad", "Channel Islands",
    "Chile", "China", "Christmas Island", "Cocos Island", "Colombia", "Comoros",
"Congo", "Cook Islands", "Costa Rica", "Cote D'Ivoire",
    "Croatia", "Cuba", "Curacao", "Cyprus", "Czech Republic", "Denmark", "Djibouti",
"Dominica", "Dominican Republic", "East Timor", "Ecuador",
    "Egypt", "El Salvador", "Equatorial Guinea", "Eritrea", "Estonia", "Ethiopia",
"Falkland Islands", "Faroe Islands", "Fiji", "Finland",
    "France", "French Guiana", "French Polynesia", "French Southern Ter", "Gabon",
"Gambia", "Georgia", "Germany", "Ghana", "Gibraltar",
    "Great Britain", "Greece", "Greenland", "Grenada", "Guadeloupe", "Guam",
"Guatemala", "Guinea", "Guyana", "Haiti", "Honduras",
    "Hong Kong", "Hungary", "Iceland", "India", "Indonesia", "Iran", "Iraq",
"Ireland", "Isle of Man", "Israel", "Italy", "Jamaica", "Japan",
    "Jordan", "Kazakhstan", "Kenya", "Kiribati", "Korea North", "Korea South",
"Kuwait", "Kyrgyzstan", "Laos", "Latvia", "Lebanon", "Lesotho",
    "Liberia", "Libya", "Liechtenstein", "Lithuania", "Luxembourg", "Macau",
"Macedonia", "Madagascar", "Malaysia", "Malawi", "Maldives",
    "Mali", "Malta", "Marshall Islands", "Martinique", "Mauritania", "Mauritius",
"Mayotte", "Mexico", "Midway Islands", "Moldova", "Monaco",
    "Mongolia", "Montserrat", "Morocco", "Mozambique", "Myanmar", "Nambia", "Nauru",
"Nepal", "Netherland Antilles", "Netherlands", "Nevis",
    "New Caledonia", "New Zealand", "Nicaragua", "Niger", "Nigeria", "Niue", "Norfolk
Island", "Norway", "Oman", "Pakistan", "Palau Island",
    "Palestine", "Panama", "Papua New Guinea", "Paraguay", "Peru", "Philippines",
"Pitcairn Island", "Poland", "Portugal", "Puerto Rico",
    "Qatar", "Republic of Montenegro", "Republic of Serbia", "Reunion", "Romania",
"Russia", "Rwanda", "St Barthelemy", "St Eustatius",
    "St Helena", "St Kitts-Nevis", "St Lucia", "St Maarten", "Saipan", "Samoa",
"Samoa American", "San Marino", "Saudi Arabia", "Scotland",
    "Senegal", "Serbia", "Seychelles", "Sierra Leone", "Singapore", "Slovakia",
"Slovenia", "Solomon Islands", "Somalia", "South Africa",
    "Spain", "Sri Lanka", "Sudan", "Suriname", "Swaziland", "Sweden", "Switzerland",
"Syria", "Tahiti", "Taiwan", "Tajikistan", "Tanzania",
    "Thailand", "Togo", "Tokelau", "Tonga", "Trinidad Tobago", "Tunisia", "Turkey",
"Turkmenistan", "Turks & Caicos Is", "Tuvalu", "Uganda",
    "Ukraine", "United Arab Emirates", "United Kingdom", "United States of America",
"Uruguay", "Uzbekistan", "Vanuatu", "Vatican City State",
    "Venezuela", "Vietnam", "Virgin Islands (British)", "Virgin Islands (USA)", "Wake
Island", "Yemen", "Zaire", "Zambia", "Zimbabwe"
        };
    }
}
```

Add Controller

MultiSelectFilter.cs copyCode

```
// GET: ShowFilterInput
public ActionResult Index()
{
    return View(Countries.GetCountries());
}
```


Copyright © GrapeCity, Inc. All rights reserved.

Add View for the Controller

Index.cshtml copyCode

```
@using MultiSelect_MVC.Models

@{
    List<string> countries = Countries.GetCountries();
}
@section Scripts{
    <script>
    var checkedItemsChanged = function (sender, e) {
        var i, result = document.getElementById("result"),
            items = sender.checkedItems;

        if (result) {
            result.innerHTML = "";
        }

        for (i = 0; i < items.length; i++) {
            result.innerHTML += "<span>" + (i + 1) + ". " + items[i] + "<\/span>
<br>";
        }
    }
    </script>
}

<div>
    <label>Select Countries:</label>

@(Html.C1().MultiSelect()
.Bind(countries)
.Name("countries")
.Id("multiselect")
.Placeholder("Please select countries")
.HeaderFormat("{count} countries selected")
.ShowFilterInput(true)
.CheckOnFilter(true)
.OnClientCheckedItemsChanged("checkedItemsChanged")
)

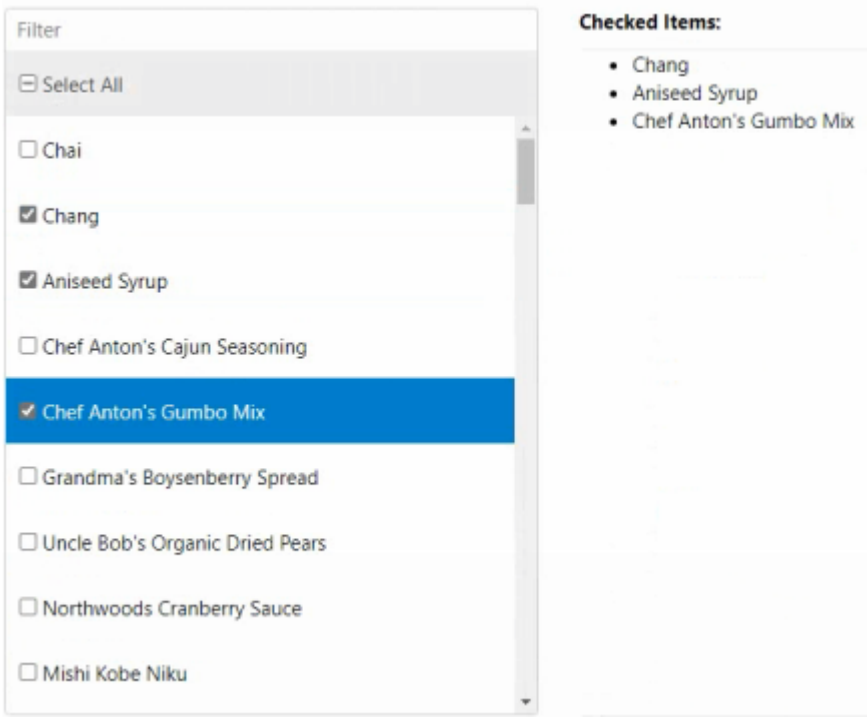
    <label>Select results:</label>
    <div id="result"></div>
</div>
```

MultiSelectListBox

The MultiSelectListBox control is an advanced ListBox control that supports check boxes for each item, ability to select all items and a feature to filter the list. The control is represented by the **MultiSelectListBox** class which allows it to displays a list of items which may contain plain text or HTML, and allows users to select multiple items. The **MultiSelectListBox** class allows you to select all items in the list using the **ShowSelectAllCheckbox** property and filter the items using the **ShowFilterInput** property. In addition, it allows you to programmatically retrieve the

Copyright © GrapeCity, Inc. All rights reserved.

checked items using the CheckedItems property.



The following example uses the Products table from NWind to show the list of products in the MultiSelectListBox control.

Controller

bgm

CellMarkerController.cs

```
private C1NWindEntities db = new C1NWindEntities();
public ActionResult Index(FormCollection collection)
{
    return View(db.Products);
}
```

View for the Controller

Index.cshtml

```
@model IEnumerable<Product>

<div style="overflow: auto">
    <div class="col-md-5">
        <div id="multiSelectList" style="width:100%;max-width:400px;height:530px">
</div>
    </div>
    <div class="col-md-7">
        <p>
            <b>Checked Items:</b>
        </p>
        <div style="height:500px; overflow-y:auto; border-top: 1px solid #eeeeee;
border-bottom: 1px solid #eeeeee;">
            <ul id="checkedItems" class="col-md-5"></ul>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <ul id="checkedItems1" class="col-md-5"></ul>
        </div>
    </div>
</div>

@section Scripts{
    <script>
    c1.documentReady(function () {
        let multiSelectList = wijmo.Control.getControl("#multiSelectList");
        checkedItemsChanged(multiSelectList);
    })
    // CheckedItemsChangedイベントハンドラー
    function checkedItemsChanged(sender) {
        let html = '',
            html1 = '',
            items = sender.checkedItems,
            n = items.length;
        if (n > 40) n = n / 2;
        else if (n > 20) n = 20;
        items.forEach(function(item) {
            n--;
            if (n >= 0) {
                html += "<li>" + item.ProductName + "</li>";
            } else {
                html1 += "<li>" + item.ProductName + "</li>";
            }
        });
        document.querySelector('#checkedItems').innerHTML = html;
        document.querySelector('#checkedItems1').innerHTML = html1;
    }
    </script>
}

@(Html.C1().MultiSelectListBox("#multiSelectList")
    .Bind(Model)
    .DisplayMemberPath("ProductName")
    .CheckedMemberPath("Discontinued")
    .ShowSelectAllCheckbox(true)
    .ShowFilterInput(true)
    .OnClientCheckedItemsChanged("checkedItemsChanged")
)
```

In addition, it lets you set the minimum number of rows and/or columns required to enable virtualization in the ListBox control using the **VirtualizationThreshold** property. For more information, see 仮想化.

Popup

ポップアップコントロールを使用すると、任意のコンテンツの小さなオーバーレイをポップオーバーまたはモーダルダイアログとしてアプリケーションに追加できます。ポップオーバーは、表示/非表示を制御するオーナー要素を持つポップアップで、画面に表示される際はオーナー要素を基準に配置されます。一方、ダイアログはオーナー要素を持たないポップアップで、画面の中央付近に表示され、画面上の他のコンテンツから区別されます。

Copyright © GrapeCity, Inc. All rights reserved.



主な特長

Popup offers many advanced features beyond simply adding small overlays of arbitrary content. These features are listed below:

- **Create popovers and dialogs**
  The Popup control enables you to add small overlays of arbitrary content to your applications, as popovers or modal dialogs. Popovers have owner elements to control their visibility whereas dialogs are the pop ups with no owner elements. Hence, a popup behaves like a popover when the element that owns the popup is specified by the **Owner** property, otherwise, it behaves like a dialog.

- **Manage popup display**
  You can control the actions that show the popup using the **ShowTrigger** property. By default, this property causes the popup to appear when user clicks the owner element. If it is set to None, the popup can only be shown when the show method is called.

- **Alerts and prompts**
  The Popup control can be used to implement static methods for showing alert and prompt dialogs.

- **Transition effects**
  You can animate the popup when it is shown or hidden using the FadeIn and FadeOut properties of the Popup class.

- **Resizable dialogs**
  Popup allows you to determine whether the popup can be resized by dragging its edges with the mouse using the IsResizable property. In addition, you can limit the size of the popup using the CSS properties.

クイックスタート

このトピックでは、アプリケーションに単純なポップオーバー（オーナー要素を持つ**ポップアップ**）コントロールを追加する方法について説明します。このようなポップアップコントロールは、**ShowTrigger**および**HideTrigger**によって指定されるアクションに基づいて表示または非表示にされる機能豊富なツールチップに似ています。ASP.NET MVC コントロールを追加する方法については、「コントロールの追加」を参照してください。

このトピックは4つの手順で構成されます。

- **手順1：MVC アプリケーションの作成**
- **手順2：新しいコントローラーおよびビューの追加**
- **手順2：Popupコントロールの追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の**Popup**コントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2：新しいコントローラーおよびビューの追加

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[コントローラーの追加]**ダイアログが表示されます。
3. **[コントローラーの追加]**ダイアログで、次の手順を実行します。
    1. コントローラーの名前を設定します（例：`QuickStartController`）。
    2. **[MVC 5 コントローラー -空]**テンプレートを選択します。
    3. **[追加]**をクリックします。
4. アプリケーションに新しいコントローラーが追加されます。

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、コントローラー（例：`DefaultController`）をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックします。コントローラーにビューが追加されます。

手順3：Popupコントロールの追加

1. **ソリューションエクスプローラー**で、**[ビュー]**フォルダを展開します。
2. `Index.cshtml`をダブルクリックして開きます。
3. `Index.cshtml`ファイルのデフォルトコードを次のコードに置き換えて、**Popup**コントロールを初期化します。

Razor copyCode

```
@using C1.Web.Mvc
@Styles.Render("~/Content/css/bootstrap/css/bootstrap.css")
<style>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .btn {
        height: auto;
    }

    .popover {
        display: block;
    }

    .tab-content > .tab-pane {
        display: block;
    }
</style>

<!-- Popup コントロールの内容。 -->
<div class="popover" id="popup">
    <h3 class="popover-title">
        タイトル
    </h3>
    <div class="popover-content ">
    <form name="popoverForm">
     <p>こんにちは<br>これは、複数行のメッセージです！</p>
     @*<pre>2 + 3 = <span class="">5</span></pre>
     <div class="form-group">
        <div class="input-group">
         <div class="input-group-addon">@@</div>
<input class="form-control" type="email" placeholder="e-メールを入力">
                </div>
            </div>
            <div class="form-actions">
<button type="button" class="btn btn-danger wj-hide">閉めます</button>
<button type="button" class="btn btn-primary wj-hide">変更を保存</button>
            </div>*@
        </form>
    </div>
</div>
<br />

<!-- Popup Owner -->
<button id="btn" type="button" class="btn btn-default">
    クリックしてPopup フォームを表示します
    <br>
    <small>(ShowTrigger=Click, HideTrigger=None)</small>
</button>

@* HtmlHelperでPopup コントロールを作成します。 *@
@(Html.C1().Popup("#popup").Owner("#btn")
.ShowTrigger(PopupTrigger.Click).HideTrigger(PopupTrigger.None))
```

手順4：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

Copyright © GrapeCity, Inc. All rights reserved.

ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**Default/Index**）。または、ビューをホームページにリンクします。

先頭に戻る

Popupの使用

This section comprises all the features and auxiliary functionalities offered by Popup.

ダイアログ：Owner要素のないポップアップ
Learn how to add a modal dialog to your application.

ダイアログ：Owner要素のないポップアップ

ダイアログは、表示/非表示を制御するOwner要素を持たないポップアップです。ダイアログは、クライアントの**show**メソッドを呼び出すことで表示されます。ダイアログはモーダルでも非モーダルでもかまいません。

モーダルダイアログは背景が暗くなり、プログラムを継続するにはユーザーの応答が必要です。非モーダルダイアログの場合、画面に表示されたままでもユーザーは他の活動を行うことができます。

本当のモーダルダイアログは、**hideTrigger**プロパティが**None**に設定され、背景や画面の他の部分をクリックしても閉じることはできません。ダイアログを閉じるためには、クライアントの**hide**メソッドを呼び出すか、または「wj-hide」クラスを使用してクリック可能な要素を介して閉じます。

次の図は、画面上で背景が暗くなり他のコンテンツから区別されているモーダルダイアログを示しています。



次のコード例は、モーダルダイアログをアプリケーションに追加する方法を示します。

コードの場合

DialogController.cs

C# copyCode

```
public ActionResult Index()
{
    return View();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

PopupDialog.cshtml

Razor copyCode

```
@using C1.Web.Mvc
@Styles.Render("~/Content/css/bootstrap/css/bootstrap.css")
<style>
    .modal-body label {
        padding: 0px;
    }
</style>

<script>
    var popups, modal = true;
    function showPopup(name) {
        if (!popups) {
            popups = {};
            popups["create"] = wijmo.Control.getControl("#popupCreate");

        }
        if (popups[name]) {
            popups[name].modal = modal;
            popups[name].show();
        }
    }

    function check(pwd1, pwd2) {
        if (pwd1.value !== pwd2.value) {
            pwd2.setCustomValidity("間違ったパスワード");
        } else {
            pwd2.setCustomValidity("");
        }
    }

    function checkCreate() {
        var pwd1 = document.getElementById("Pwd1を作成"),
            pwd2 = document.getElementById("Pwd2を作成");
        check(pwd1, pwd2);
    }

    function changeModal() {
        modal = wijmo.getElement("#modal").checked;
    }

    //最初入力要素はダイアログが表示された後にフォーカスされます。
    function autoFocus(control) {
        control.hostElement.querySelector("input").focus();
    }
</script>

<!-- アカウントを作成するポップアップダイアログの内容 -->
<div id="popupCreate" class="modal-content col-md-6">
    <form>
        <h4 class="modal-header">
            アカウントを作る
            <button type="button" tabindex="-1" class="close wj-hide">×</button>
        </h4>
        <div class="modal-body">
            <label>
                ユーザ名：
                <input class="form-control" required="" pattern=".{2,}" title="パスワードは少なくても 2 文字以上
に設定する必要があります。">
            </label>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <br>
            <label>
                メールアドレス:
                <input class="form-control" required="" type="email">
            </label>
            <br>
            <label>
                パスワード:
                <input class="form-control" type="password" id="createPwd1" required="" pattern=".{4,}"
 title="パスワードは少なくても 4 文字以上に設定する必要があります。">
            </label>
            <br>
            <label>
                パスワード(確認用に再入力):
                <input class="form-control " type="password" id="createPwd2" required="" pattern=".{4,}"
 title="パスワードは少なくても 4 文字以上に設定する必要があります。">
            </label>
        </div>
        <div class="modal-footer">
            <button class="btn btn-primary" type="submit" onclick="checkCreate()">
                アカウントを作る
            </button>
        </div>
    </form>
</div>
<br />
<button class="btn btn-default" onclick="showPopup('create')">アカウントを作る</button>
<label>Modal <input type="checkbox" id="modal" checked="checked" onchange="changeModal()" /></label>


@(Html.C1().Popup("#popupCreate").Modal(true).HideTrigger(PopupTrigger.None).OnClientShown("autoFocus"))
```

先頭に戻る

共通の機能

共通機能には、ASP.NET MVC で使用できるすべての**Input**コントロールの主要機能が含まれます。入力コントロールのさまざまな機能について、以下のように分けて説明します。

スキャフォールディング：Input
Inputコントロールでスキャフォールディングプロパティを使用する方法を説明します。

AutoExpandSelectionプロパティ
InputコントロールでAutoExpandSelectionプロパティを使用する方法を説明します。

DropDownCssClassプロパティ
コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定する方法について説明します。

控えめな検証
Inputコントロールでクライアント側のunobtrusive validationサポートを使用する方法を説明します。

AutoExpandSelection

AutoExpandSelectionプロパティを使用すると、ユーザーがコンボボックスにキー入力したときに、ドロップダウンコントロールが、入力した文字と一致する値をコンボボックスのリストからコンボボックスのテキストボックス部分に自動的に挿入するかどうかを指定できます。

> **メモ**: **InputColor**コントロールと**InputDate**コントロールの場合、AutoExpandSelectionプロパティのデフォルト値は**True**に設定されています。その他のInputコントロールのデフォルト値はFalseです。

AutoExpandSelectionプロパティは、次のInputドロップダウンコントロールで使用できます。

Copyright © GrapeCity, Inc. All rights reserved.

- InputColor
- InputDate
- AutoComplete、ComboBox
- InputTime

次のコード例は、InputコントロールでAutoExpandSelectionを使用する方法を示しています。

IndexController.cs

Razor

```
public ActionResult Index()
  {
ViewBag.Cities = Models.Cities.GetCities();
return View();
   }
```

Index.cshtml

Razor

```
@using C1.Web.Mvc;
@{

List<string> cities = ViewBag.Cities;

}
@(Html.C1().ComboBox().Bind(cities).SelectedIndex(0)
.AutoExpandSelection(true))
```

DropDownCssClass

コントロールのドロップダウン要素に追加するCSSクラス名を取得または設定できます。このプロパティは、ドロップダウン要素をスタイル設定する場合に便利です。ドロップダウン要素は、コントロール自体の子としてではなく、ドキュメントボディの子として表示され、親コントロールに基づいてCSSセレクタを使用することができないためです。このプロパティを使用すると、ComboBoxドロップダウンに複数の列を表示できます。

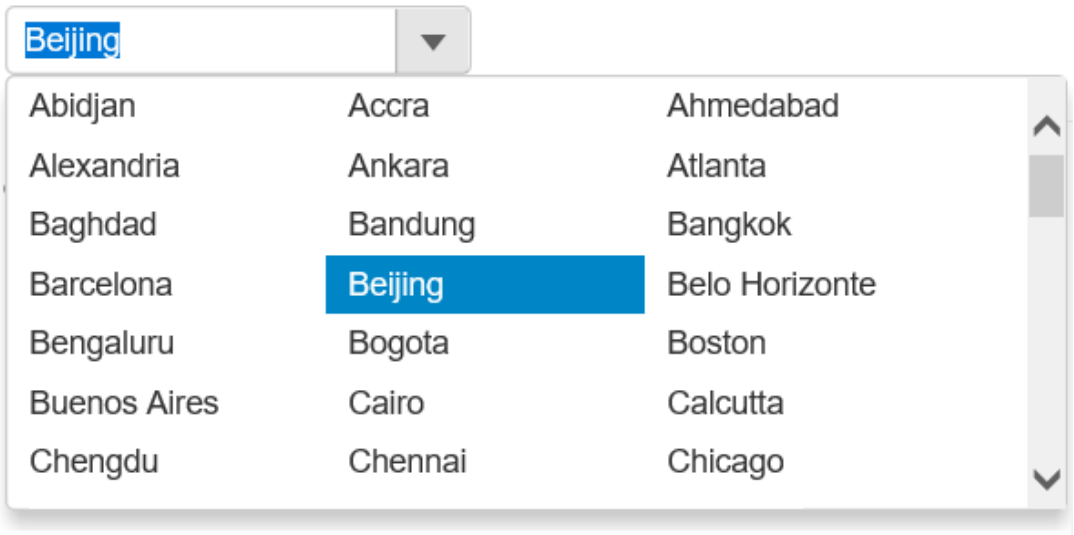
次のコード例は、**DropDownCssClass**プロパティを使用して複数列を有効にする方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

DropDownCssClassプロパティは、次のInputドロップダウンコントロールで使用できます。

- InputColor
- InputDate
- AutoComplete、ComboBox
- InputTime
- MultiSelect
- Menu

次の図は、**DropDownCssClass**プロパティを適用した後のMenuを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、DropDownCssClassプロパティを使用する方法を示します。

IndexController.cs

Razor

```
public ActionResult Index()
        {
        ViewBag.Cities = Models.Cities.GetCities();
        return View();
        }
```

ComboBox.cshtml

Razor

```
<style>
    .multi-column
    {   display: flex;
        flex-wrap: wrap;
        max-width: 450px;
    }
</style>
@{ List<string> cities = ViewBag.Cities;}
@Html.C1().ComboBox().Bind(cities).SelectedIndex(0).IsEditable(false).DropDownCssClass("multi-column")
```

スタイル設定とCSS

A set of common CSS classes is used by most of the Wijmo Input controls , this table lists the same:

Common Classes

Class	Description
wj-input-group	Styles the div element containing the input text and button(s) in the control.
wj-form-control	Styles the HTML input element in the control.
wj-input-	Styles the div element containing button(s) in the control.

Copyright © GrapeCity, Inc. All rights reserved.

group-btn	
.wj-btn	Styles the button for input controls.
wj-glyph-down	Styles the down arrow shown in the dropdown button of the input controls (AutoComplete, ComboBox, InputColor, InputDate, InputTime, Menu).
.wj-dropdown-panel	Styles the host element of the dropdown displayed for the input controls.

ListBox

Class	Description
wj-listbox	Styles the div host element of the ListBox control.
.wj-header	Styles the div element representing GroupHeader if the grouping is enabled.
.wj-listbox-item	Styles the div element representing the list items.

AutoComplete/ComboBox/InputTime/Menu/MultiSelect/MultiAutoComplete

The input controls listed in the below table use the **ListBox** control. Hence, the CSS classes defined for the ListBox control can be used to customize the ListBox element in these controls. This table further lists the CSS classes that are specific to these controls:

Class	Description
AutoComplete	
.wj-autocomplete	Styles the div host element of the AutoComplete control.
ComboBox	
.wj-combobox	Styles the div host element of the ComboBox control.
Input Time	
wj-inputtime	Styles the div host element of the InputTime control.
Menu	
.wj-menu	Styles the div host element of the Menu control.
.wj-separator	Styles the div element representing the separator in the dropdown of Menu control.
MultiSelect	
.wj-multiselect	Styles the div host element of the MultiSelect control.
MultiAutoComplete	
.wj-multi-autocomplete	Styles the div host element of the MultiAutoCompelete control.
.wj-token	Styles the div element displaying the selected item.
.wj-token-label	Styles the text of the selected item.
wj-token-close	Styles the (x) button used to remove the selected item.

Calendar

Class	Description

Copyright © GrapeCity, Inc. All rights reserved.

.wj-calendar	Styles the div host element of the Calendar control.
.wj-calendar-outer	Styles the container div which contains the calendar elements.
.wj-calendar-header	Styles the Calendar header displaying the currently selected month and navigation buttons.
.wj-month-select	Styles the div element which represents the currently selected month.
.wj-glyph-circle	Styles the circle glyph icon used for navigating to today's date.
.wj-header	Styles the Calendar header showing displaying days in month view and year in year view.
.wj-calendar-month	Styles the table showing the days for the currently selected month.
.wj-day-today	Styles the today's date.
.wj-day-weekend	Styles the weekend days dates.
wj-day-othermonth	Styles the other month dates shown in current month.
wj-calendar-year	Styles the table showing months for current year.
.wj-yearpicker	Styles the table showing list of years.

InputDate/InputDateTime

The InputDate/InputDateTime controls, implicitly use the Calendar control to display dates and the InputDateTime control even uses the ListBox control to display the list of times. Hence, the CSS classes defined for the ListBox and Calendar control can be used to customize these elements in the InputDate and InputDateTime controls. This table further lists the CSS classes that are specific to these controls.

Class	Description
InputDate	
.wj-inputdate	Styles the div host element of the InputDate control.
InputDateTime	
.wj-inputdatetime	Styles the div host element of the InputDateTime control.
.wj-glyph-calendar	Styles the calendar icon displayed in the control.
.wj-glyph-clock	Styles the clock icon displayed in the control.

Color Picker

Class	Description
.wj-colorpicker	Styles the div host element of the ColorPicker control.

InputColor

The InputColor uses the ColorPicker control, hence the CSS classes defined for the ColorPicker can be used to customize the ColorPicker element in the InputColor control. This table further lists the CSS classes that are specific to this control:

Class	Description
.wj-inputcolor	Styles the div host element of the InputColor control.
.wj-inputcolorbox	Styles the div element displaying the selected color.

Copyright © GrapeCity, Inc. All rights reserved.

InputMask/InputNumber

Class	Description
InputMask	
.wj-inputmask	Styles the div host element of the InputMask control.
InputNumber	
.wj-inputnumber	Styles the host element of the InputNumber control.

Popup

Class	Description
.wj-popup	Styles the div host element of the Popup control.
.wj-popup-backdrop	Styles the div element representing the dark backdrop if the Popup is shown as modal dialog.
.wj-dialog-header	Styles the Popup header content and also makes the Popup draggable if the isDraggable property is set to True.
.wj-dialog-body	Styles the Popup body content.
.wj-dialog-footer	Styles the Popup footer content.

スタイルの適用

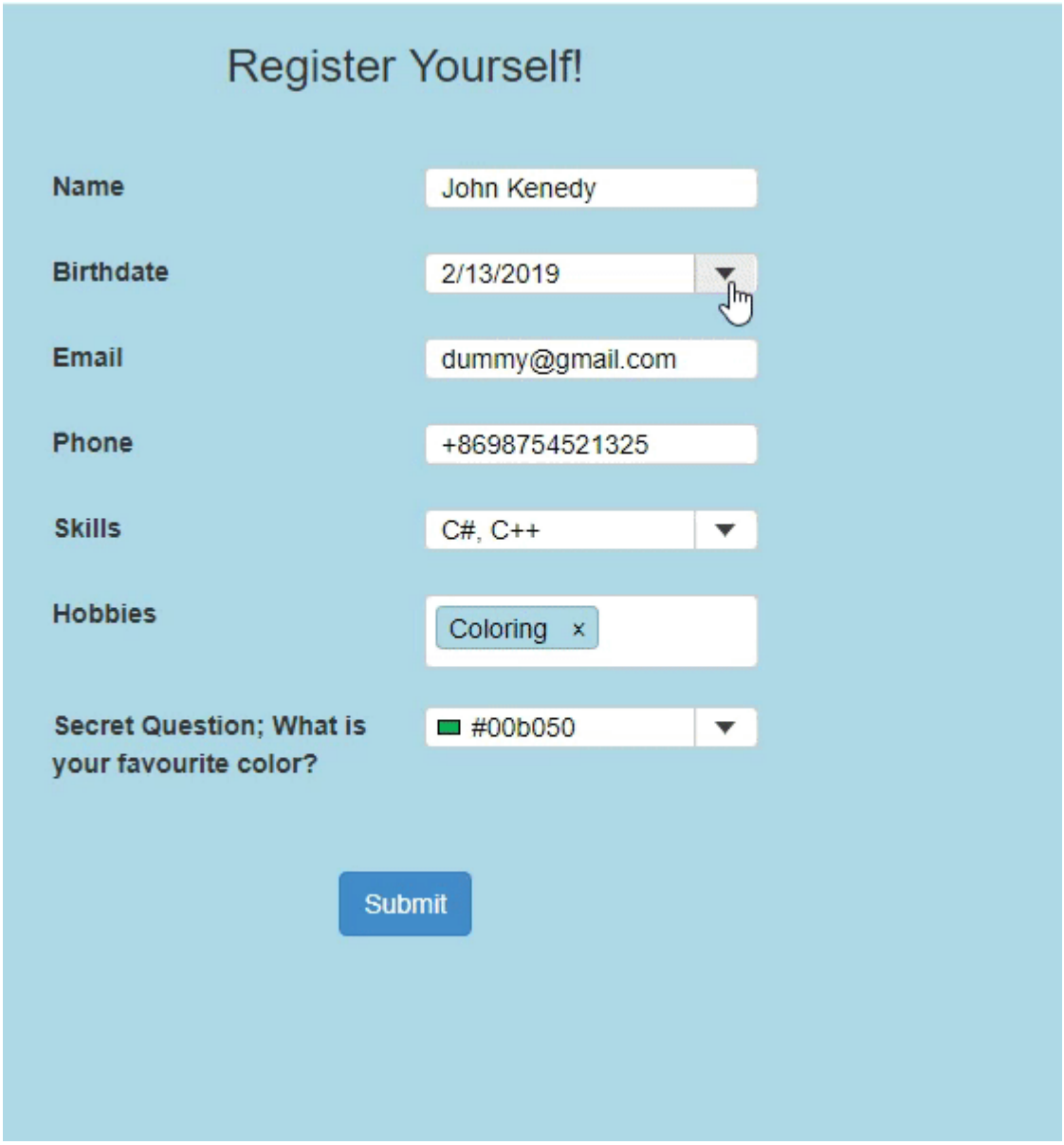
The appearance of the Input controls is defined using CSS classes which represent different input controls such as InputNumber, InputDate, InputColor etc. You can customize these classes to change the appearance of the Input controls. The custom CSS rules can be applied to the Input controls, by applying a CSS class to the Input controls using the **CssClass** property.

In this example, we customize the default CSS rules to make the following customizations to the input controls.

1. Reduce the height of the Input controls.
2. Change weekend days background.
3. Hide other month's days from current month view.
4. Reduce the dropdown items height.
5. Show the blank items to allow selection.
6. Change background color for listbox items on hover.
7. Set background-color for the MultiAutoComplete control's selected items.
8. Set the width and height properties for ColorPicker control.

The following GIF image shows how different input control appears after applying styles using custom CSS.

Copyright © GrapeCity, Inc. All rights reserved.



Add Custom Stylesheet

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

To add a custom style sheet in your application, follow these steps:

1. In the Solution Explorer, right-click the **Content** folder.
2. From the context menu, select **Add | Style Sheet**. The **Specify Name for Item** dialog appears.
3. Set name of the style sheet (for example: `app.css`) and click **OK**.
4. Replace the default code of **app.css** file with the code given below.

app.css

```
.custom{
    max-width:100%;
    width:100%;
    height:22px;
}
.custom .wj-input-group .wj-form-control {
    min-height: 1em;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    padding: 1px 8px;
}
.custom .wj-input-group .wj-btn {
    min-height: 1em;
}
.customDropDown .wj-day-weekend {
    background-color: lightgreen;
}
.customDropDown .wj-day-othermonth {
    opacity: 0;
    pointer-events: none;
}
.custom.wj-input-color .wj-form-control {
    padding-left: 24px;
}
.customDropDown .wj-listbox-item {
    padding: 0 6px;
    height: 22px;
    min-height: 22px;
}
.customDropDown .wj-listbox-item:not(.wj-state-selected):not(.wj-state-
disabled):not(.wj-separator):hover {
    background-color:cornflowerblue;
}
.customDropDown label {
    width: 100%;
}
.custom .wj-token{
    margin:0;
    border-width:1px;
    background:lightblue;
}
.wj-colorpicker {
    min-width: 335px !important;
    width: 335px !important;
    min-height: 134px !important;
    height: 134px !important;
}
.custom .wj-token .wj-token-label {
    padding: 0 6px;
}
.custom .wj-token .wj-token-close{
    padding:0 6px;
}
.form-control {
    height: 22px;
}
```

Styling.cshtml

Razor

```
@using <ApplicationName>.Models
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@model UserInfo
<div class="col-md-5" style="margin:20px 20px; width:700px; height:600px; background-
color:lightblue">
    @using (Html.BeginForm())
    {
        <h3 style="text-align:center;">Register Yourself!</h3>
        <div style="width:60%; margin:0 auto;padding:20px;">

            <div class="row">
                <div class="col-md-6" style="padding:10px">
                    @Html.LabelFor(m => m.Name)
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().InputMaskFor(m => m.Name).CssClass("custom")
                </div>
            </div>
            <div class="row">
                <div class="col-md-6" style="padding:10px">
                    @Html.LabelFor(m => m.Birthdate)
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().InputDateFor(m =>
m.Birthdate).CssClass("custom").DropDownCssClass("customDropDown")
                </div>
            </div>
            <div class="row">
                <div class="col-md-6" style="padding:10px">
                    @Html.LabelFor(m => m.Email)
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().InputMaskFor(m => m.Email).CssClass("custom")
                </div>
            </div>
            <div class="row">
                <div class="col-md-6" style="padding:10px">
                    @Html.LabelFor(m => m.Phone)
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().InputMaskFor(m =>
m.Phone).Mask("+8600000000000").CssClass("custom")
                </div>
            </div>
            <div class="row">
                <div class="col-md-6" style="padding:10px">
                    @Html.LabelFor(m => m.Skills)
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().MultiSelectFor(m => m.Skills).Bind(b =>
b.Bind(UsersData.SkillsList)).CssClass("custom").DropDownCssClass("customDropDown")
                </div>
            </div>
            <div class="row">
                <div class="col-md-6" style="padding:10px">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                    @Html.LabelFor(m => m.Hobbies)
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().MultiAutoCompleteFor(m => m.Hobbies).Bind(b =>
b.Bind(UsersData.HobbyList)).CssClass("custom").DropDownCssClass("customDropDown")
                </div>
            </div>
            <div class="row">
                @*<div class="col-md-12">
                    <label>Your Secret question:</label>
                </div>*@
                <div class="col-md-6" style="padding:10px">
                    <label>Secret Question; What is your favourite color?</label>
                </div>
                <div class="col-md-6" style="padding:10px">
                    @Html.C1().InputColorFor(m => m.FavoriteColor).CssClass("custom")
                </div>
            </div>
            <br />
            <div class="row" style="margin-top:12px;text-align:center;">
                <button type="submit" value="Submit" class="btn btn-
primary">Submit</button>
            </div>
        </div>
    }
</div>
```

控えめな検証

Inputコントロールは、控えめな検証をサポートしており、クライアント側検証によってC1入力コントロールを検証することができます。控えめな検証は、大量の検証コードを書かずにシンプルなクライアント側検証を実装し、正しい属性を追加してスクリプトファイルをインクルードするだけで、ユーザーエクスペリエンスを向上させることができます。

任意のコントロールを検証する検証やクライアント側の検証を使用する一般的な検証シナリオでは、JavaScriptコードが生成されて、WebブラウザでHTMLとしてレンダリングされます。一方、控えめな検証では、クライアント側の検証を処理するためにレンダリングされるインラインJavaScriptは生成されません。その代わりに、HTML5 data-*属性をクライアント側検証に使用します。

次のInputコントロールで、控えめな検証をサポートできます。

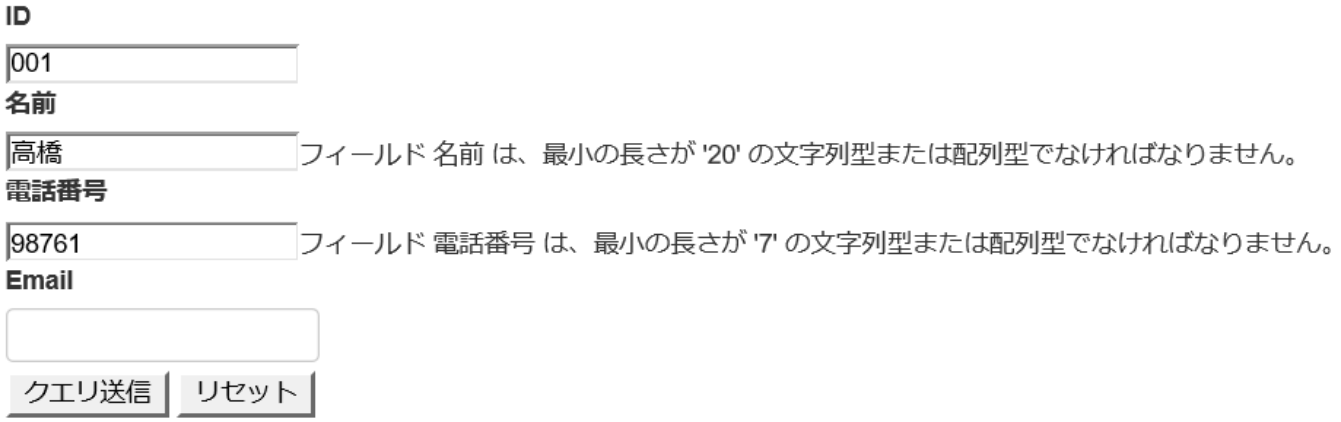
- InputColor
- InputDate
- InputDateTime
- InputMask
- InputNumber
- AutoComplete
- ComboBox
- InputTime
- MultiSelect

以下の手順を実装する前に、ComponentOneテンプレートまたはVisual Studioテンプレートを使用して、新しいMVCアプリケーションを作成する必要があります。

このトピックは4つの手順で構成されます。

Copyright © GrapeCity, Inc. All rights reserved.

- **手順1：MVCアプリケーションの構成**
- **手順2：入力コントロールの検証の作成**
- **手順3：入力コントロールの追加**
- **手順4：プロジェクトのビルドおよび実行**



手順1：MVCアプリケーションの構成

Razor

次の手順を実行して、アプリケーションにjsファイルの参照を追加します。
ソリューションエクスプローラーで、[ビュー]→[共有]フォルダを開きます。
_Layout.cshtmlをダブルクリックして開きます。
<head></head>タグの間に以下のコードを追加します。
RazorCOPYCODE

```
<script src="~/Scripts/jquery-1.10.2.js"></script>
<script src="~/Scripts/jquery.validate.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.js"></script>
```

手順2：入力コントロールの検証の作成

モデル - Form.cs

Razor

```
using System.ComponentModel.DataAnnotations;
namespace UnobtrusiveValidationSample.Models
{
    public class Form
    {
        [Required]

        public string ID { get; set; }

        [Required]
        [DataType(DataType.Text)]
        [MinLength(8)]
        [MaxLength(30)]

        public string 名前 { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        [Required]
        [DataType(DataType.PhoneNumber)]
        [Phone]
        [Range(0, 10000000000)]
        [StringLength(11)]
        [MinLength(7)]
        [MaxLength(10)]

        public string 電話番号 { get; set; }

        [Required]
        [DataType(DataType.EmailAddress)]
        [EmailAddress]
        [MinLength(10)]
        [MaxLength(50)]

        public string Email { get; set; }

    }
}
```

手順3：入力コントロールの追加

ビュー - Index.cshtml

Razor

```
@using UnObtrusiveValidationSample.Models;
@model Form
@using (Html.BeginForm())
{
    <br />
    @Html.LabelFor(m => m.ID)
    <br />
    @Html.EditorFor(m => m.ID)
    @Html.ValidationMessageFor(m => m.ID)
    <br />

    @Html.LabelFor(m => m.名前)
    <br />
    @Html.EditorFor(m => m.名前)
    @Html.ValidationMessageFor(m => m.名前)
    <br />

    @Html.LabelFor(m => m.電話番号)
    <br />
    @Html.EditorFor(m => m.電話番号)
    @Html.ValidationMessageFor(m => m.電話番号)
    <br />

    @Html.LabelFor(m => m.Email)
    <br />
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    @Html.C1().InputMaskFor(m => m.Email)
    @Html.ValidationMessageFor(m => m.Email)
    <br />
    <input type="クエリ送信" />
    <input type="リセット" />
}
```

先頭に戻る

手順4:プロジェクトのビルドおよび実行

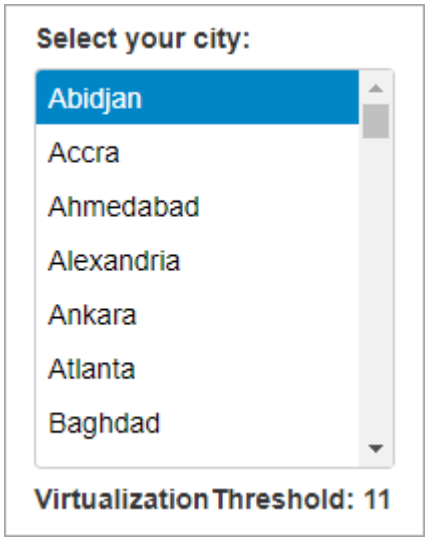
1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

先頭に戻る

仮想化

Virtualization is the process of keeping track of which portions of the data are visible and rendering only those parts. Some of the input controls, such as ListBox, ComboBox and MultiSelectListBox support virtualization. These controls allow you to use the **VirtualizationThreshold** property from their respective classes to set the minimum number of rows and/or columns required to enable virtualization in them. By default, the virtualization is disabled in these controls as the value for the **VirtualizationThreshold** property is set to a really large number. However, you can enable virtualization by setting the value of **VirtualizationThreshold** property to 0 or a positive number.

The following image showcases a Virtualized ListBox control.



In the following example, we enable virtualization in ListBox using **VirtualizationThreshold** property of the ListBox class. Similarly, you can enable virtualization in ComboBox and MultiSelectListBox controls using VirtualizationThreshold property of the ComboBoxBase and the MultiSelectListBox class respectively. The following example uses Cities.cs model added in the Quick Start topic.

View Code

Add the "@using <ApplicationName>.Models" reference and then use the following code to enable Virtualization.

Index.cshtml copyCode

```
@{
    List<string> cities = Cities.GetCities();
}
```


Copyright © GrapeCity, Inc. All rights reserved.

```
<div>
    <label>Select your city:</label>
</div>

<div>
    @(Html.C1().ListBox()
        .Id("list")
        .Bind(cities)
        .Width(180).Height(200)
        .VirtualizationThreshold(0)
    )
</div>

<p>
    <b>VirtualizationThreshold: <span id="lbSelVThreshold"></span></b>
</p>

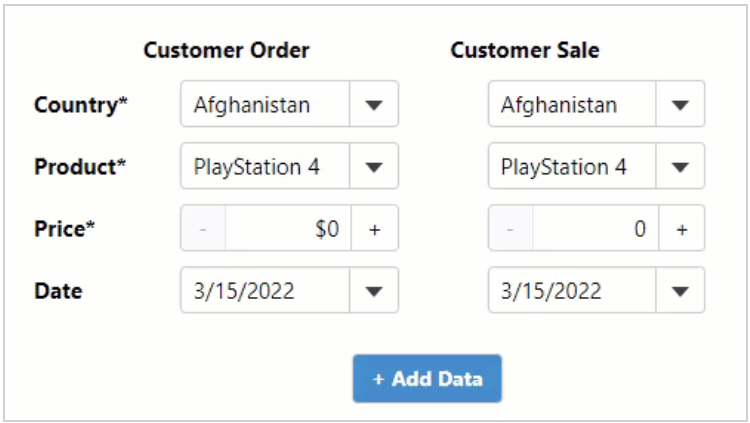
<script>
    c1.documentReady(function () {
        document.getElementById('lbSelVThreshold').innerHTML =
document.getElementById("list").childNodes.length;
    });
</script>
```

タブオーダー

Tab ordering is used in many modern applications which have dynamic pages/views that constantly change with user interaction. In such applications, the TabIndex can be set on all controls at runtime using the **TabOrder** property. This property allows the user to customize the tab navigation order of the input controls added in the application. You can specify the **TabOrder** property for each control, a tab order follows relationships in the content without following the order of the interactive elements in the code.

TabIndex attribute value can be defined statically for an ASP.NET MVC control by specifying it in the control's host HTML element. However, the value can't be modified later during application lifecycle. Therefore, to read or modify the input control's tab index dynamically you can use the **TabOrder** property.

The following image shows how you can customize the tab order navigation:



Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Add a Data source

1. Add a new class to the folder **Models** (for example: `Products.cs`).
2. Add the following code to the new model to define the classes that serve as a data source for the ComboBox control.

Products

```
public static List<string> GetProducts()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
            return new List<string>
            {
                "PlayStation 4", "XBOX ONE", "Wii U", "PlayStation Vita", "PlayStation 3", "XBOX 360",
"PlayStation Portable", "3 DS",
                "Dream Cast", "Nintendo 64", "Wii", "PlayStation 2", "PlayStation 1", "XBOX"
            };
        }
        public static List<string> GetCountries()
        {
            return new List<string>
            {
                "Algeria", "American Samoa", "Argentina", "Armenia", "Australia", "Austria", "Bahrain",
"Bangladesh","Belarus", "Belgium",
            "Bhutan", "Bolivia", "Brazil", "Bulgaria", "Cambodia", "Cameroon", "Canada", "Colombia", "Costa
Rica", "Cuba", "Croatia",
            "Curacao", "Cyprus", "Czech Republic", "Denmark", "Dominican Republic", "Egypt", "Ethiopia",
"Finland", "France", "French Guiana",
            "Georgia", "Germany", "Ghana", "Gibraltar", "Great Britain", "Greece", "Greenland", "Hong Kong",
"Hungary", "Iceland", "India",
            "Indonesia", "Iran", "Iraq", "Ireland", "Israel", "Italy", "Jamaica", "Japan", "Jordan", "Kenya",
"Korea North", "Korea South",
            "Kuwait", "Liberia", "Libya", "Lithuania", "Madagascar", "Malaysia", "Maldives", "Mauritius",
"Mexico", "Myanmar", "Nambia",
            "Nepal", "Netherlands", "New Zealand", "Nigeria", "Norway", "Oman", "Panama", "Paraguay", "Peru",
"Philippines","Poland",
            "Portugal", "Puerto Rico", "Qatar", "Romania", "Russia", "Samoa", "Saudi Arabia", "Scotland",
"Serbia", "Singapore", "Slovakia",
            "South Africa", "Spain", "Sri Lanka", "Sweden", "Switzerland", "Syria", "Taiwan", "Tanzania",
"Thailand", "Turkey", "Uganda",
            "Ukraine", "United Arab Emirates", "United Kingdom", "United States of America", "Uruguay",
"Uzbekistan", "Venezuela",
            "Vietnam", "Yemen", "Zambia", "Zimbabwe"
            };
        }
```

Back to Top

Add Input Controls

Complete the following steps to add Input controls in your application.

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **Empty MVC Controller** template.
    2. Set name of the controller (for example: `TabOrderController`).
    3. Click **Add**.
4. Replace the method `Index()` with the following method.

Products

```
public ActionResult Index()
{
  return View();
}
```

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the controller `TabOrderController` to open it.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the view name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add**. A view is added for the controller.
6. Add Input controls in the view as shown below.

Products

```
@using <ApplicationName>.Models
@using C1.Web.Mvc;

<html>
<body>
    <table class="input-form">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        <tr>
            <th></th>
            <th>Customer Order</th>
            <th>Customer Sale</th>
        </tr>
        <tr>
            <th>Country*</th>

<td>@Html.C1().ComboBox().Id("orderCountry").Bind(Products.GetCountries()).TabOrder(1).IsRequired(true)</td>

<td>@Html.C1().ComboBox().Id("saleCountry").Bind(Products.GetCountries()).TabOrder(5).IsRequired(true)</td>
        </tr>
        <tr>
            <th>Product*</th>

<td>@Html.C1().ComboBox().Id("orderProduct").Bind(Products.GetProducts()).TabOrder(2).IsRequired(true)</td>

<td>@Html.C1().ComboBox().Id("saleProduct").Bind(Products.GetProducts()).TabOrder(6).IsRequired(true)</td>
        </tr>
        <tr>
            <th>Price*</th>

<td>@Html.C1().InputNumber().Id("orderPrice").Min(0).Step(1).Value(0).Format("c0").IsRequired(true).TabOrder(3)
</td>

<td>@Html.C1().InputNumber().Id("salePrice").Min(0).Step(1).Value(0).Format("c0").IsRequired(true).TabOrder(7)
</td>
        </tr>
        <tr>
            <th>Date</th>
            <td>@Html.C1().InputDate().Id("orderDate").Value(DateTime.Today).IsRequired(false).TabOrder(4)</td>
            <td>@Html.C1().InputDate().Id("saleDate").Value(DateTime.Today).IsRequired(false).TabOrder(8)</td>
        </tr>
    </table>
    <button id="add-row" class="btn btn-primary col-sm-offset-2">
        <b><i class="glyphicon-plus"></i> Add Data</b>
    </button>
</body>
</html>
```

Back to Top

MultiRow

MultiRowコントロールは、複数の行を使用して各データ項目を表形式で表す強力で柔軟な方法を提供します。MultiRowコントロールでは、他の従来のグリッドと同様に、表形式でデータを表示して編集できます。従来のグリッドと異なるのは、**MultiRow**では、各データ項目を複数の行にまとめてフォーム状のインタフェースを作成し、最小限の水平スクロールで多くの列を表示できることです。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	注文	お客様	お客様のE一メル	
	金額	出荷	住所	都市	州
	0	2016/03/26	Makoto Kanagawa	Makoto.Kanagawa@gmail.com	
	¥4,462	2016/03/27	1129 Suzuki ,Japan.	ハンブルク	SC
	1	2014/05/10	Akemi Ikeda	Akemi.Ikeda@gmail.com	
	¥1,101	2014/05/14	2327 Kawasaki ,Japan.	カイロ	RS
	2	2013/07/08	Makoto Suzuki	Makoto.Suzuki@gmail.com	
	¥2,362	2013/07/12	4418 Kawasaki ,Japan.	カイロ	SC
	3	2014/04/23	Ayako Suzuki	Ayako.Suzuki@gmail.com	
	¥2,515	2014/04/27	2796 Yamaguchi ,Japan.	岩手	RN
	4	2010/05/08	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥2,527	2010/05/09	717 Kanagawa ,Japan.	岩手	RT
	5	2012/12/24	Daichi Sasaki	Daichi.Sasaki@gmail.com	
	¥248	2012/12/27	1372 Ikeda ,Japan.	カイロ	SC
	6	2011/12/16	Akemi Yamaguchi	Akemi.Yamaguchi@gmail.com	
	¥3,598	2011/12/17	3772 Kawasaki ,Japan.	大阪	RN
	7	2014/08/29	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥3,306	2014/08/31	717 Kanagawa ,Japan.	岩手	RT

MultiRowコントロールを使用するには、**C1.Web.Mvc.MultiRow**名前空間に含まれる**MultiRow**クラスのインスタンスを作成する必要があります。MultiRowクラスは、**FlexGrid**クラスを拡張しています。FlexGrid コントロールを使用したことのあるユーザーなら、MultiRowコントロールは簡単に理解して実装することができます。詳細については、「クイックスタート：MultiRowへのデータの追加」を参照してください。

主な特長

MultiRowの主な機能は以下のとおりです。

- **マルチレイアウトの定義**：MultiRowでは、**LayoutDefinition**プロパティを設定できます。これは、グリッド行とセルのレイアウトを記述するオブジェクトを受け取ります。LayoutDefinitionプロパティは、MultiRowコントロール内のセルのレイアウトを指定します。
- **完全に展開された列ヘッダー**：MultiRowでは、**CollapsedHeaders**プロパティをtrueに設定することで、グループ名を含む単一行の列ヘッダーをグリッドに表示することができます。このプロパティをfalseに設定すると、行グループがデータと同じレイアウトで表示され、セルに列名が表示されます。
- **折りたたみ可能な列ヘッダー**：MultiRowでは、**CollapsedHeaders**プロパティをtrueに設定して、列ヘッダーを1行に折りたたみ、個別のセルではなくグループ名だけを表示することができます。列ヘッダーを折りたたむと、個別のセルヘッダーを表示するよりもスペースの節約になります。
- **FlexGrid機能**：MultiRowは**FlexGrid**クラスを拡張しています。これは、FlexGridコントロールと同様の完全な機能を備えたグリッドで、複数の選択モード、ソート、フィルタ処理、行と列の固定、カスタムセル、データマッピングなどのさまざまな機能を提供します。

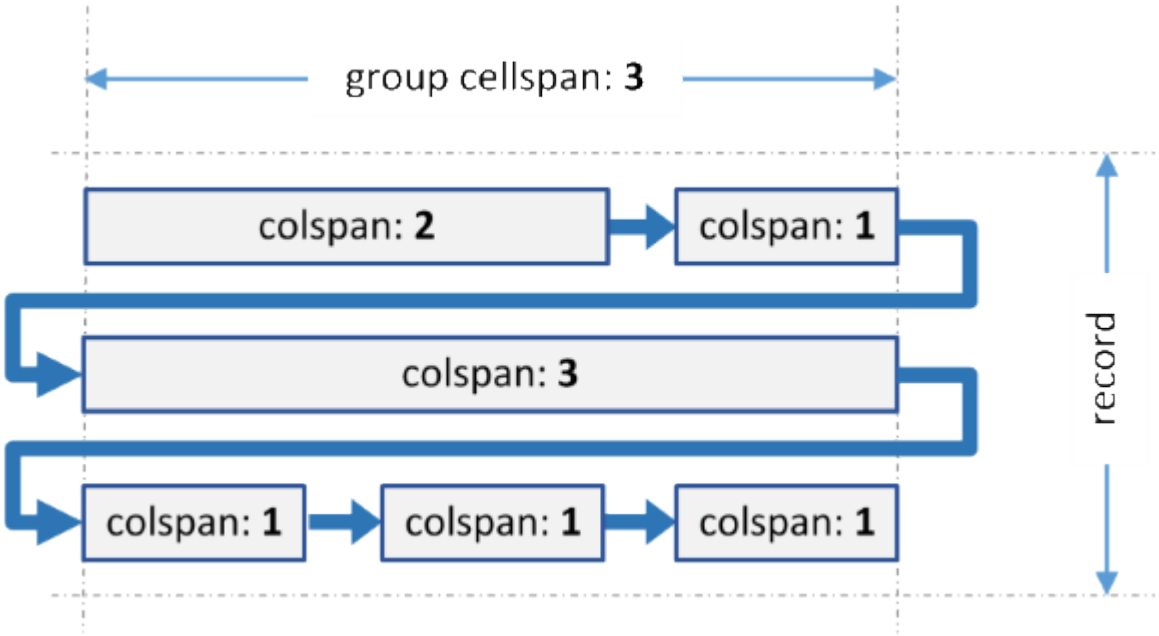
レイアウト定義

Copyright © GrapeCity, Inc. All rights reserved.

MultiRowコントロールのメインのプロパティは、**LayoutDefinition**です。これは、グリッド行とセルのレイアウトを記述するオブジェクトを受け取ります。LayoutDefinitionプロパティは、グリッド内のセルのレイアウトを指定します。このプロパティには、セルグループオブジェクトの配列が含まれます。各セルグループは、グループが占める列の数と、各グループを構成するセルを指定します。

LayoutDefinitionプロパティには、次のプロパティを持つセルグループオブジェクトの配列が入っています。

- **header** - グループ名
- **colspan** - グループが占める列の数。
- **cells** - グループ内のセルを表すオブジェクトの配列。それぞれのセルは、追加のcolspanプロパティを持つ列です。



このグループの幅はグリッド列3つ分です。そこに、さまざまな幅を持つ6つのセルが含まれています。レイアウトの生成時、グリッドは各行にできるだけ多くのセルを入れ、グループ幅に達すると次の行の先頭にセルを配置します。各行の最後のセルは、グループのcolspan全体に自動的に拡大されます。この処理は、テキストを折り返して段落を作成する方法に似ています。同じ処理がlayoutDefinitionオブジェクト内のすべてのグループに適用されます。

クイックスタート：MultiRow へのデータの追加

このトピックでは、MVC WebアプリケーションにMultiRowコントロールを追加し、そこにモデル連結を使用してデータを追加する方法について説明します。

このトピックは次の4つの手順で構成されます。

- **手順1：MVC アプリケーションの作成**
- **手順2：MultiRowのデータソースの作成**
- **手順3：MultiRowコントロールの追加**
- **手順4：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のデータを含むMultiRowコントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	注文	お客様	お客様のE一メル	
	金額	出荷	住所	都市	州
	0	2016/03/26	Makoto Kanagawa	Makoto.Kanagawa@gmail.com	
	¥4,462	2016/03/27	1129 Suzuki ,Japan.	ハンブルク	SC
	1	2014/05/10	Akemi Ikeda	Akemi.Ikeda@gmail.com	
	¥1,101	2014/05/14	2327 Kawasaki ,Japan.	カイロ	RS
	2	2013/07/08	Makoto Suzuki	Makoto.Suzuki@gmail.com	
	¥2,362	2013/07/12	4418 Kawasaki ,Japan.	カイロ	SC
	3	2014/04/23	Ayako Suzuki	Ayako.Suzuki@gmail.com	
	¥2,515	2014/04/27	2796 Yamaguchi ,Japan.	岩手	RN
	4	2010/05/08	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥2,527	2010/05/09	717 Kanagawa ,Japan.	岩手	RT
	5	2012/12/24	Daichi Sasaki	Daichi.Sasaki@gmail.com	
	¥248	2012/12/27	1372 Ikeda ,Japan.	カイロ	SC
	6	2011/12/16	Akemi Yamaguchi	Akemi.Yamaguchi@gmail.com	
	¥3,598	2011/12/17	3772 Kawasaki ,Japan.	大阪	RN
	7	2014/08/29	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥3,306	2014/08/31	717 Kanagawa ,Japan.	岩手	RT

手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2:MultiRowのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`Orders.cs`)。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`Orders.cs`モデルに追加します。ここでは、**Orders**クラスを使用して、データベース内の受注データを表しています。**Orders**オブジェクトの各インスタンスがMultiRowコントロール内の1つのレコードに対応します。

C#

Orders.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace MultiRow.Models
{
    public class Orders
    {
        private static object _lockObj = new object();
        private static Random Rand = new Random();
        private static IList<Order> _orders;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        private static IList<string> _cities;
        private static IList<Customer> _customers;
        public class Customer
        {
            public int Id { get; set; }
            public string Name { get; set; }
            public string Address { get; set; }
            public string City { get; set; }
            public string State { get; set; }
            public string Email { get; set; }
        }
        public class Order
        {
            public int Id { get; set; }
            public DateTime Date { get; set; }
            public DateTime ShippedDate { get; set; }
            public double Amount { get; set; }
            public Customer Customer { get; set; }
        }

        public static IList<Order> GetOrders()
        {
            if (_orders == null)
            {
                var today = DateTime.Now.Date;
                var customers = GetCustomers();
                _orders = new List<Order>();
                for (int i = 0; i < 8; i++)
                {
                    var shipped = today.AddDays(-Rand.Next(-1, 3000));
                    var order = new Order
                    {
                        Id = i,
                        Date = shipped.AddDays(-Rand.Next(1, 5)),
                        ShippedDate = shipped,
                        Amount = Rand.Next(10000, 500000) / 100,
                        Customer = customers[Rand.Next(0, customers.Count - 1)],
                    };
                    _orders.Add(order);
                }
            }
            return _orders;
        }
        public static IList<Customer> GetCustomers()
        {
            if (_customers == null)
            {
                var firstNames = new[] { "Akemi", " Ayako", "Atsuko", "Hanako",
"Daichi", "Katsuo",
                "Junko", "Ken", "Makoto", "Mio" };
                var lastNames = new[] { "Sasaki", "Suzuki", "Kawasaki", "Yamaguchi",
"Kimura",
                "Ikeda", "Kanagawa", "Minato" };
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                var cities = GetCities();
                var states = new[] { "SP", "RS", "RN", "SC", "CS", "RT", "BC" };

                _customers = new List<Customer>();
                for (int i = 0; i < 50; i++)
                {
                    var first = firstNames[Rand.Next(0, firstNames.Length - 1)];
                    var last = lastNames[Rand.Next(0, lastNames.Length - 1)];
                    var customer = new Customer
                    {
                        Id = i,
                        Name = first + " " + last,
                        Address = Rand.Next(100, 10000) + " " +
lastNames[Rand.Next(0, lastNames.Length - 1)]
                            + " ,Japan.",
                        City = cities[Rand.Next(0, cities.Count - 1)],
                        State = states[Rand.Next(0, states.Length - 1)],
                        Email = first + "." + last + "@gmail.com",
                    };
                    _customers.Add(customer);
                }
            }
            return _customers;
        }
        public static IList<string> GetCities()
        {
            {
                if (_cities == null)
                {
                    _cities = new[] { "京都", "岩手", "大阪", "カイロ", "石川", "東京", "ハ
ンブルク", "奈良" };
                }
            }
            return _cities;
        }
    }
}
```

先頭に戻る

手順3：MultiRowコントロールの追加

アプリケーションにMultiRowコントロールを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`MultiRowController`）。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

    C#

Copyright © GrapeCity, Inc. All rights reserved.

```
using MultiRow.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

5. メソッド**Index()**を次のメソッドに置き換えます。

C#

MultiRowController.cs

```
public ActionResult Index()
        {
            var model = Orders.GetOrders();
            return View(model);
        }
```

コントローラーのビューの追加

ビュー内で、MultiRowコントロールのインスタンスを作成し、それを**.Bind**プロパティを使用してデータソースに連結します。レイアウト定義のプロパティを使用すると、コントロールの列および行のレイアウトを定義できます。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、MultiRowControllerをダブルクリックして開きます。
2. メソッドIndex()内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

Index.cshtml copyCode

```
@using MultiRow.Models

@model IEnumerable<Orders.Order>

@section Styles{
    <link rel="stylesheet" href="~/Content/CustomMultiRow.css" />
}
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .GroupBy("Customer.State")
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

手順4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http://localhost:1234/**MultiRow/Index**)。

先頭に戻る

MultiRowの使用

This section comprises all the features and auxiliary functionalities offered by MultiRow.

一括編集
: Learn how to perform batch editing in MultiRow.

折りたたみ列ヘッダー
: Learn how to collapse the column headers.

データ連結
: Learn how to perform remote and model data binding.

エクスポート
: Learn how to export the MultiRow content in Excel and PDF format.

フィルタ
: Learn how to use Filtering in MultiRow control.

グループ化
: Learn how to group data in MultiRow.

グループパネル
: Learn how to use Group Panel for grouping in MultiRow.

ページング
: Learn how to enable Paging using Pager control in MultiRow control.

固定行や列
: Learn about freezing rows and columns in MultiRow.

選択モード
: Learn about different selection modes available in MultiRow.

レコード、グループ、セルのスタイル設定
: Learn about styling records, group delimiters and specific cells within groups.

仮想スクロール
: Learn how to apply virtual scrolling to the large data set.

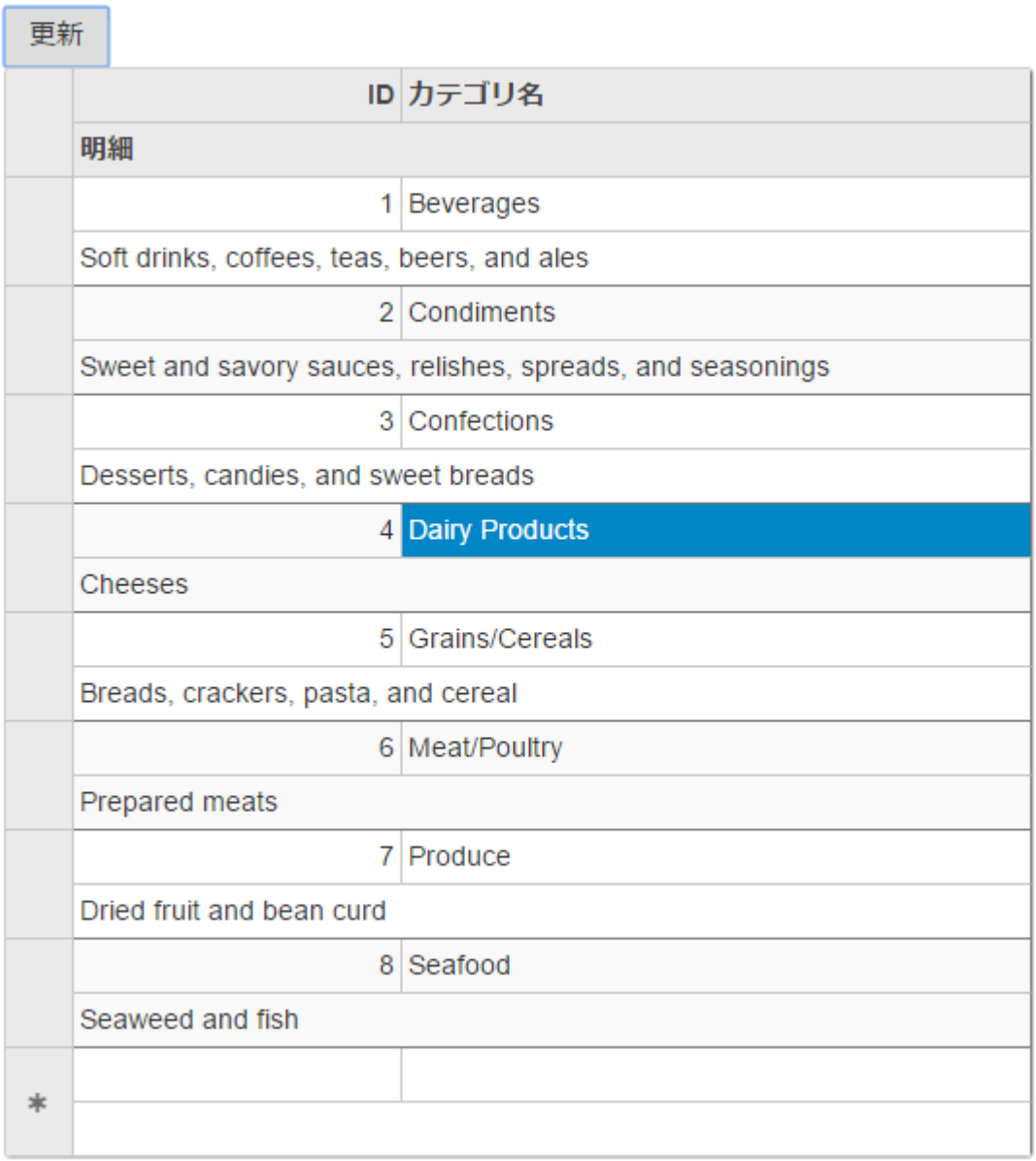
Copyright © GrapeCity, Inc. All rights reserved.

一括編集

MultiRowコントロールは一括編集をサポートしています。これにより、ユーザーは、複数の項目を更新、作成、または削除したり、実行時にデータ項目をデータソースにコミットすることができます。また、ソート、ページング、またはフィルタ処理を使用して、MultiRowコントロールで複数の変更を実行することもできます。一括編集モードでMultiRowコントロールにアクセスするには、**BatchEditActionUrl**プロパティを設定する必要があります。

> **メモ**:ソート、フィルタ処理、またはページ操作の実行中にデータ更新を無効にするには、MultiRow ItemsSourceの**DisableServerRead**プロパティをtrueに設定します。データを送信するには、collectionViewのcommitメソッドをクライアント側から明示的に呼び出します。

次の図は、BatchEditActionUrlプロパティを使用してデータをコミットするための[Update]ボタンを備えたMultiRowコントロールを示します。この例では、サーバーにデータをコミットする方法を2つ提供します。1つめは、セルのコンテンツを更新した後に[Enter]キーを押す方法です。2つめは、[Update]ボタンをクリックする方法です。



コードの場合

BatchEditingController.cs

```
BatchEditingController.cs
```


Copyright © GrapeCity, Inc. All rights reserved.

```
private C1NwindEntities db = new C1NwindEntities();
public ActionResult BatchEditing(CollectionViewRequest requestData)
        {
            return View(db.Categories.ToList());
        }

public ActionResult MultiRowBatchEdit([C1JsonRequest]CollectionViewBatchEditRequest
requestData)
        {
            return this.C1Json(CollectionViewHelper.BatchEdit(requestData, batchData
=>
            {
                var itemresults = new List>();
                string error = string.Empty;
                bool success = true;
                try
                {
                    if (batchData.ItemsCreated != null)
                    {
                        batchData.ItemsCreated.ToList().ForEach(st =>
                        {
                            db.Categories.Add(st);
                            itemresults.Add(new CollectionViewItemResult
                            {
                                Error = "",
                                Success = ModelState.IsValid,
                                Data = st
                            });
                        });
                    }
                    if (batchData.ItemsDeleted != null)
                    {
                        batchData.ItemsDeleted.ToList().ForEach(category =>
                        {
                            var fCategory = db.Categories.Find(category.CategoryID);
                            db.Categories.Remove(fCategory);
                            itemresults.Add(new CollectionViewItemResult
                            {
                                Error = "",
                                Success = ModelState.IsValid,
                                Data = category
                            });
                        });
                    }
                    if (batchData.ItemsUpdated != null)
                    {
                        batchData.ItemsUpdated.ToList().ForEach(category =>
                        {
                            db.Entry(category).State = EntityState.Modified;
                            itemresults.Add(new CollectionViewItemResult
                            {
                                Error = "",
                                Success = ModelState.IsValid,
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                                Data = category
                            });
                        });
                    }
                    db.SaveChanges();
                }
                catch (DbEntityValidationException e)
                {
                    error = string.Join(",", e.EntityValidationErrors.SelectMany
        (i => i.ValidationErrors).Select(i => i.ErrorMessage));
                    success = false;
                }
                catch (Exception e)
                {
                    error = e.Message;
                    success = false;
                }

                return new CollectionViewResponse
                {
                    Error = error,
                    Success = success,
                    OperatedItemResults = itemresults
                };
            }, () => db.Categories.ToList()));
        }
```

BatchEditing.cshtml

BatchEditing.cshtml

```
<script type="text/javascript">
        function batchUpdate() {
            var batchEditMultiRow = wijmo.Control.getControl('#batchEditMultiRow'),
                cv = batchEditMultiRow.collectionView;
            cv.commit();
        }
    </script>
}
<input type="button" value="更新" class="btn" onclick="batchUpdate()" />
@(Html.C1().MultiRow<Category>()
    .Id("batchEditMultiRow")
    .LayoutDefinition(ld =>
    {
        ld.Add().Colspan(2).Cells(columns =>
        {
            columns.Add(c => c.Binding("CategoryID").IsReadOnly(true).Format("d"))
            .Add(c => c.Binding("CategoryName").Width("*"))
            .Add(c => c.Binding("Description"));
        });
    })
    .Bind(ib =>
ib.DisableServerRead(true).Bind(Model).BatchEdit(Url.Action("MultiRowBatchEdit"))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .AllowAddNew(true)
    .AllowDelete(true)
)
```

折りたたみ列ヘッダー

MultiRowコントロールでは、列ヘッダーを1行に折りたたみ、個別のセルではなくグループ名だけを表示することができます。デフォルトでは、**MultiRow**コントロールは複数の行から成る列ヘッダーを表示し、そこに、LayoutDefinitionで定義された各セルのヘッダーが表示されます。列ヘッダーを折りたたむと、個別のセルヘッダーを表示するよりもスペースの節約になります。

次の図は、ヘッダーを折りたたんで表示したMultiRowコントロールを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

	注文		お客様		
	0	2015/10/21	Katsuo Suzuki	Katsuo.Suzuki@gmail.com	
	¥3,928	2015/10/24	9148 Kimura ,Japan.	大阪	CS
	1	2012/01/26	Akemi Suzuki	Akemi.Suzuki@gmail.com	
	¥1,825	2012/01/27	8831 Ikeda ,Japan.	ハンブルク	RN
	2	2013/07/19	Ayako Sasaki	Ayako.Sasaki@gmail.com	
	¥4,458	2013/07/22	6940 Yamaguchi ,Japan.	岩手	RS
	3	2009/01/24	Daichi Ikeda	Daichi.Ikeda@gmail.com	
	¥587	2009/01/25	4505 Ikeda ,Japan.	カイロ	CS
	4	2016/05/12	Hanako Ikeda	Hanako.Ikeda@gmail.com	
	¥2,506	2016/05/15	6396 Ikeda ,Japan.	ハンブルク	RS
	5	2012/06/09	Ken Suzuki	Ken.Suzuki@gmail.com	
	¥585	2012/06/13	4493 Sasaki ,Japan.	カイロ	RS
	6	2013/09/17	Hanako Suzuki	Hanako.Suzuki@gmail.com	
	¥4,611	2013/09/20	3129 Kanagawa ,Japan.	石川	SC
	7	2013/03/13	Daichi Yamaguchi	Daichi.Yamaguchi@gmail.com	
	¥3,835	2013/03/16	3313 Kimura ,Japan.	京都	RN

コードの場合

CollapsedHeaders.cshtml

列ヘッダーを折りたたむには、**CollapsedHeaders**プロパティをtrueに設定します。その場合は、列ヘッダーが空にならないように、グループのHeaderプロパティを忘れずに設定してください。

CollapsedHeaders.cshtml

```
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .CollapsedHeaders(true)
    .ShowHeaderCollapseButton(true)
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

データ連結

MultiRowコントロールとその機能を使用するには、データをMultiRowに連結する必要があります。MultiRowコントロールにデータを連結したら、フィルタ処理、ソート、選択モード、詳細行、グループ化などのさまざまな機能を使用できます。

このセクションには、次の情報が含まれます。

リモートデータ連結
: C1JSONRequestを使用して、リモートでMultiRowコントロールにデータを連結する方法を説明します。

モデル連結
: 基本のモデル連結を使用して、MultiRowコントロールにデータを追加する方法を説明します。

モデル連結

このトピックでは、MVCアプリケーションにMultiRowコントロールを追加し、コントロールにデータを挿入する方法について説明します。以下の例は、MultiRowコントロールでローカルモデル連結を行う方法を示しています。MultiRowコントロールでリモートデータ連結を行うこともできます。詳細については、「リモートデータ連結」を参照してください。

このトピックは次の3つの手順で構成されます。

- **手順1：MultiRowのデータソースの作成**
- **手順2：MultiRowコントロールの追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後に表示されるMultiRowコントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	注文	お客様	お客様のE一メル	
	金額	出荷	住所	都市	州
	0	2016/03/26	Makoto Kanagawa	Makoto.Kanagawa@gmail.com	
	¥4,462	2016/03/27	1129 Suzuki ,Japan.	ハンブルク	SC
	1	2014/05/10	Akemi Ikeda	Akemi.Ikeda@gmail.com	
	¥1,101	2014/05/14	2327 Kawasaki ,Japan.	カイロ	RS
	2	2013/07/08	Makoto Suzuki	Makoto.Suzuki@gmail.com	
	¥2,362	2013/07/12	4418 Kawasaki ,Japan.	カイロ	SC
	3	2014/04/23	Ayako Suzuki	Ayako.Suzuki@gmail.com	
	¥2,515	2014/04/27	2796 Yamaguchi ,Japan.	岩手	RN
	4	2010/05/08	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥2,527	2010/05/09	717 Kanagawa ,Japan.	岩手	RT
	5	2012/12/24	Daichi Sasaki	Daichi.Sasaki@gmail.com	
	¥248	2012/12/27	1372 Ikeda ,Japan.	カイロ	SC
	6	2011/12/16	Akemi Yamaguchi	Akemi.Yamaguchi@gmail.com	
	¥3,598	2011/12/17	3772 Kawasaki ,Japan.	大阪	RN
	7	2014/08/29	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥3,306	2014/08/31	717 Kanagawa ,Japan.	岩手	RT

手順1：MultiRowのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Orders.cs`）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`Orders.cs`モデルに追加します。ここでは、**Orders**クラスを使用して、データベース内の受注データを表しています。**Orders**オブジェクトの各インスタンスがMultiRowコントロール内の1つのレコードに対応します。

C#

Orders.cs copyCode

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
namespace MultiRow.Models
{
    public class Orders
    {
        private static object _lockObj = new object();
        private static Random Rand = new Random();
        private static IList<Order> _orders;
        private static IList<string> _cities;
        private static IList<Customer> _customers;
        public class Customer
        {
            public int Id { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            public string Name { get; set; }
            public string Address { get; set; }
            public string City { get; set; }
            public string State { get; set; }
            public string Email { get; set; }
        }
        public class Order
        {
            public int Id { get; set; }
            public DateTime Date { get; set; }
            public DateTime ShippedDate { get; set; }
            public double Amount { get; set; }
            public Customer Customer { get; set; }
        }

        public static IList<Order> GetOrders()
        {
            if (_orders == null)
            {
                var today = DateTime.Now.Date;
                var customers = GetCustomers();
                _orders = new List<Order>();
                for (int i = 0; i < 8; i++)
                {
                    var shipped = today.AddDays(-Rand.Next(-1, 3000));
                    var order = new Order
                    {
                        Id = i,
                        Date = shipped.AddDays(-Rand.Next(1, 5)),
                        ShippedDate = shipped,
                        Amount = Rand.Next(10000, 500000) / 100,
                        Customer = customers[Rand.Next(0, customers.Count - 1)],
                    };
                    _orders.Add(order);
                }
            }
            return _orders;
        }
        public static IList<Customer> GetCustomers()
        {
            if (_customers == null)
            {
                var firstNames = new[] { "Akemi", " Ayako", "Atsuko", "Hanako",
"Daichi", "Katsuo",
                "Junko", "Ken", "Makoto", "Mio" };
                var lastNames = new[] { "Sasaki", "Suzuki", "Kawasaki", "Yamaguchi",
"Kimura",
                "Ikeda", "Kanagawa", "Minato" };
                var cities = GetCities();
                var states = new[] { "SP", "RS", "RN", "SC", "CS", "RT", "BC" };

                _customers = new List<Customer>();
                for (int i = 0; i < 50; i++)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                {
                    var first = firstNames[Rand.Next(0, firstNames.Length - 1)];
                    var last = lastNames[Rand.Next(0, lastNames.Length - 1)];
                    var customer = new Customer
                    {
                        Id = i,
                        Name = first + " " + last,
                        Address = Rand.Next(100, 10000) + " " +
lastNames[Rand.Next(0, lastNames.Length - 1)]
                            + " ,Japan.",
                        City = cities[Rand.Next(0, cities.Count - 1)],
                        State = states[Rand.Next(0, states.Length - 1)],
                        Email = first + "." + last + "@gmail.com",
                    };
                    _customers.Add(customer);
                }
            }
            return _customers;
        }
        public static IList<string> GetCities()
        {
            {
                if (_cities == null)
                {
                    _cities = new[] { "京都", "岩手", "大阪", "カイロ", "石川", "東京", "ハ
ンブルク", "奈良" };
                }
            }
            return _cities;
        }
    }
}
```

先頭に戻る

手順2：MultiRowコントロールの追加

アプリケーションにMultiRowコントロールを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`MultiRowController`）。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C#

```
using MultiRow.Models;
using System;
using System.Collections.Generic;
using System.Linq;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
using System.Web;
using System.Web.Mvc;
```

5. メソッド**Index()**を次のメソッドに置き換えます。

C#

MultiRowController.cs

```
public ActionResult Index()
        {
            var model = Orders.GetOrders();
            return View(model);
        }
```

コントローラーのビューの追加

ビュー内で、MultiRowコントロールのインスタンスを作成し、それを**.Bind**プロパティを使用してデータソースに連結します。レイアウト定義のプロパティを使用すると、コントロールの列および行のレイアウトを定義できます。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`MultiRowController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor (CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

Index.cshtml copyCode

```
@using MultiRow.Models

@model IEnumerable<Orders.Order>

@section Styles{
    <link rel="stylesheet" href="~/Content/CustomMultiRow.css" />
}
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .GroupBy("Customer.State")
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

手順3:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http://localhost:1234/**MultiRow/Index**)。

先頭に戻る

リモートデータ連結

MultiRowコントロールでは、**C1JSONRequest**を使用してデータを直接取得できます。これは、サーバー、テーブル、および列を含むリモートデータのURLを指定します。返される配列は、**CollectionView**オブジェクトのデータソースとして使用されます。

`CollectionViewHelper` は、コレクションに編集、フィルタ処理、グループ化、ソートなどのサービスを提供する静的クラスです。このクラスには次のメソッドも含まれます。

- **`Read()`**:コレクションからデータを取得します。
- **`Edit()`**:MultiRowでExcel形式の編集を可能にします。
- **`BatchEdit()`**:一度に複数の項目を編集できます。

MultiRowの**`Bind`**プロパティは、特定の操作を実行するためのアクションURLメソッドを渡すことで、MultiRowをコレクションに連結するために使用されます。

このトピックでは、既存のデータソースからデータをリモートで取得する方法について説明します。これは、データをダッシュボードとして表すデータ集約型のアプリケーションやシナリオを開発する際に役立ちます。次の図は、C1JSON要求を行ってデータをモデルからフェッチした後のMultiRowコントロールを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	注文	お客様	お客様のE一メル	
	金額	出荷	住所	都市	州
	0	2016/03/26	Makoto Kanagawa	Makoto.Kanagawa@gmail.com	
	¥4,462	2016/03/27	1129 Suzuki ,Japan.	ハンブルク	SC
	1	2014/05/10	Akemi Ikeda	Akemi.Ikeda@gmail.com	
	¥1,101	2014/05/14	2327 Kawasaki ,Japan.	カイロ	RS
	2	2013/07/08	Makoto Suzuki	Makoto.Suzuki@gmail.com	
	¥2,362	2013/07/12	4418 Kawasaki ,Japan.	カイロ	SC
	3	2014/04/23	Ayako Suzuki	Ayako.Suzuki@gmail.com	
	¥2,515	2014/04/27	2796 Yamaguchi ,Japan.	岩手	RN
	4	2010/05/08	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥2,527	2010/05/09	717 Kanagawa ,Japan.	岩手	RT
	5	2012/12/24	Daichi Sasaki	Daichi.Sasaki@gmail.com	
	¥248	2012/12/27	1372 Ikeda ,Japan.	カイロ	SC
	6	2011/12/16	Akemi Yamaguchi	Akemi.Yamaguchi@gmail.com	
	¥3,598	2011/12/17	3772 Kawasaki ,Japan.	大阪	RN
	7	2014/08/29	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
	¥3,306	2014/08/31	717 Kanagawa ,Japan.	岩手	RT

コードの場合

次のMVC参照をRemoteBindController.csに追加します。

C#

```
using <ApplicationName>.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using C1.Web.Mvc;
using C1.Web.Mvc.Serialization;
```

RemoteBindController.cs

C#

```
public partial class MultiRowController : Controller
 {
     // GET: MultiRow
     public ActionResult RemoteBind_Read([C1JsonRequest]
CollectionViewRequest<Orders.Order> requestData)
     {
     return this.C1Json(CollectionViewHelper.Read(requestData, Orders.GetOrders()));
     }
        public ActionResult Index()
     {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        return View();
    }
}
```

RemoteBind.cshtml

Razor

```
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(Url.Action("RemoteBind_Read"))
    .AllowSorting(true)
    .IsReadOnly(true)
    .CssClass("multirow")
    .LayoutDefinition(ld =>
    {
            ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

エクスポート

MultiRow enables you to export it's content into different formats. These formats are listed below:

Excel エクスポート
: Learn how to export the MultiRow content in Excel format.

PDF エクスポート
: Learn how to export the MultiRow content in PDF format.

Excel エクスポート

MultiRowコントロールを使用して、データをExcelファイル（.xlsx）形式にエクスポートできます。MultiRowのコンテンツをExcel

Copyright © GrapeCity, Inc. All rights reserved.

形式にエクスポートするには、**FlexGridXlsxConverter.save**メソッドを使用する必要があります。このメソッドによってxlsxファイルコンテンツが生成され、それをファイルシステムに保存したり、サーバー上で共有することができます。

次の図は、MultiRowのコンテンツをExcel形式でエクスポートする方法を示します。



	ID	開始日	国		金額	金額2
	アクティブ	終了日	製品	色		割引
	◢ 製品: ガジェット (3 項目)					
	◢ 国: ドイツ (1 項目)					
	1	2017/01/25	ドイツ		581.61	-2,939.67
	☑	2017/01/25	ガジェット	緑		0.14
	◢ 国: 日本 (1 項目)					
	2	2017/02/25	日本		-2,080.94	-326.85
	☐	2017/02/25	ガジェット	白		0.16
	◢ 国: イタリア (1 項目)					
	4	2017/04/25	イタリア		1,999.42	262.84
	☐	2017/04/25	ガジェット	黒		0.23
	◢ 製品: ウィジェット (2 項目)					
	◢ 国: 韓国 (1 項目)					
	3	2017/03/25	韓国		4,953.47	1,771.81
	☐	2017/03/25	ウィジェット	緑		0.08
	◢ 国: 中国 (1 項目)					
	5	2017/05/25	中国		-466.73	-2,028.28
	☑	2017/05/25	ウィジェット	黒		0.25

エクスポート

Views > Shared> _Layout.cshtml

以下のコードを実装する前に、`jszip.min.js` JavaScriptライブラリをアプリケーションに追加して、MultiRowのコンテンツをExcelのxlsxファイルにエクスポートできるようにする必要があります。

HTML

```
<!-- SheetJS library -->
<script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.2.1/jszip.min.js">
</script>
```

モデル - Sale.cs

ここでは、**Sale**クラスを使用して、データベース内の受注データを表しています。**Sale**オブジェクトの各インスタンスがMultiRowコントロール内の1つのレコードに対応します。

C#

```
    public class Sale
    {
        public int ID { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public DateTime Start { get; set; }
        public DateTime End { get; set; }
        public string Country { get; set; }
        public string Product { get; set; }
        public string Color { get; set; }
        public double Amount { get; set; }
        public double Amount2 { get; set; }
        public double Discount { get; set; }
        public bool Active { get; set; }
        public int Rank { get; set; }
        private static List<string> COUNTRIES = new List<string>
        { "米国", "英国", "カナダ", "日本", "中国", "フランス", "ドイツ", "イタリア", "韓国",
"オーストラリア" };
        private static List<string> PRODUCTS = new List<string>
        { "ウィジェット", "ガジェット", "Doohickey" };
        public static IEnumerable<Sale> GetData(int total)
        {
            var colors = new[] { "黒", "白", "赤", "緑", "青" };
            var rand = new Random(0);
            var dt = DateTime.Now;
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var color = colors[rand.Next(0, colors.Length - 1)];
                var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
                var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60, i
% 60);
                return new Sale
                {
                    ID = i + 1,
                    Start = startDate,
                    End = endDate,
                    Country = country,
                    Product = product,
                    Color = color,
                    Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Amount2 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Discount = Math.Round(rand.NextDouble() / 4, 2),
                    Active = (i % 4 == 0),
                    Rank = rand.Next(1, 6)
                };
            });
            return list;
        }
        public static List<string> GetCountries()
        {
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }
        public static List<string> GetProducts()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
    }
}
```

コントローラー - ExcelExportController.cs

C#

```
public ActionResult Index()
{
         return View(Sale.GetData(5));
}
```

ビュー - Index.cshtml

ビューで、MultiRowコントロールのインスタンスを作成し、[Export]ボタンを追加します。レイアウト定義のプロパティを使用すると、コントロールの列および行のレイアウトを定義できます。JavaScriptにより、FlexGridXlsxConverterを使用してMultiRowコントロールをExcelファイルにエクスポートできます。

Razor

```
@model IEnumerable<Sale>
@using MultiRow.Models;
@section Scripts{
    <script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
    <script>
        var multiRow, colHeaderCheckBox;
        c1.documentReady(function () {
            multiRow = wijmo.Control.getControl("#excelExportMultiRow");
        });

        function exportXlsx() {
            if (multiRow) {
                wijmo.grid.xlsx.FlexGridXlsxConverter.save(multiRow,
        { includeCellStyles: false, includeColumnHeaders: true }, 'MultiRow.xlsx');
            }
        }
    </script>}
<br />
@(Html.C1().MultiRow<Sale>().Id("excelExportMultiRow")
    .Bind(Model)
    .ShowGroups(true)
    .GroupBy("Product", "Country")
    .IsReadOnly(true)
    .CssClass("multirow")
    .LayoutDefinition(ld =>
    {
        ld.Add().Cells(cells =>
        {
            cells.Add(cell => cell.Binding("ID").Header("ID"));
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            cells.Add(cell => cell.Binding("Active").Header("アクティブ"));
        });
        ld.Add().Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Start").Header("開始日"));
            cells.Add(cell => cell.Binding("End").Header("終了日"));
        });
        ld.Add().Colspan(2).Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Country").Header("国").Colspan(2));
            cells.Add(cell => cell.Binding("Product").Header("製品"));
            cells.Add(cell => cell.Binding("Color").Header("色"));
        });
        ld.Add().Colspan(2).Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Amount").Header("金額"));
            cells.Add(cell => cell.Binding("Amount2").Header("金額2"));
            cells.Add(cell => cell.Binding("Discount").Header("割引").Colspan(2));
        });
    }))
<a download="MultiRow.xlsx" class="btn btn-default" id="exportBtn"
        onclick="exportXlsx();">エクスポート</a>
```

PDF エクスポート

MultiRowコントロールを使用しているときに、同僚がデータにアクセスできるようにしたり、ファイルをサーバー上で共有するために、データをPDF形式にエクスポートする必要があることがあります。MultiRowコントロールは、クライアント側のPDFエクスポート機能を提供しています。PDFKitベースのJavaScriptライブラリである**FlexGridPdfConverter**を使用して、サーバー側のコードを使用することなく、MultiRowをPDF（Portable Document Format）にエクスポートできます。MultiRowコントロールは、現在の列順、表示/非表示、サイズを使用してPDFファイルの出力を生成します。

次の図は、MultiRowのコンテンツをPDFファイル形式でエクスポートする方法を示します。

Copyright © GrapeCity, Inc. All rights reserved.

	ID	開始日	国		金額	金額2
	アクティブ	終了日	製品	色		割引
	◢製品: ガジェット (3 項目)					
	◢国: ドイツ (1 項目)					
	1	2017/01/25	ドイツ		581.61	-2,939.67
	☑	2017/01/25	ガジェット	緑		0.14
	◢国: 日本 (1 項目)					
	2	2017/02/25	日本		-2,080.94	-326.85
	☐	2017/02/25	ガジェット	白		0.16
	◢国: イタリア (1 項目)					
	4	2017/04/25	イタリア		1,999.42	262.84
	☐	2017/04/25	ガジェット	黒		0.23
	◢製品: ウィジェット (2 項目)					
	◢国: 韓国 (1 項目)					
	3	2017/03/25	韓国		4,953.47	1,771.81
	☐	2017/03/25	ウィジェット	緑		0.08
	◢国: 中国 (1 項目)					
	5	2017/05/25	中国		-466.73	-2,028.28
	☑	2017/05/25	ウィジェット	黒		0.25

エクスポート

モデル - Sale.cs

ここでは、**Sale**クラスを使用して、データベース内の受注データを表しています。**Sale**オブジェクトの各インスタンスがMultiRowコントロール内の1つのレコードに対応します。

C#

```
    public class Sale
    {
        public int ID { get; set; }
        public DateTime Start { get; set; }
        public DateTime End { get; set; }
        public string Country { get; set; }
        public string Product { get; set; }
        public string Color { get; set; }
        public double Amount { get; set; }
        public double Amount2 { get; set; }
        public double Discount { get; set; }
        public bool Active { get; set; }
        public int Rank { get; set; }
        private static List<string> COUNTRIES = new List<string>
        { "米国", "英国", "カナダ", "日本", "中国", "フランス", "ドイツ", "イタリア", "韓国",
"オーストラリア" };
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        private static List<string> PRODUCTS = new List<string>
        { "ウィジェット", "ガジェット", "Doohickey" };
        public static IEnumerable<Sale> GetData(int total)
        {
            var colors = new[] { "黒", "白", "赤", "緑", "青" };
            var rand = new Random(0);
            var dt = DateTime.Now;
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var color = colors[rand.Next(0, colors.Length - 1)];
                var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
                var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, i % 60, i
% 60);
                return new Sale
                {
                    ID = i + 1,
                    Start = startDate,
                    End = endDate,
                    Country = country,
                    Product = product,
                    Color = color,
                    Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Amount2 = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Discount = Math.Round(rand.NextDouble() / 4, 2),
                    Active = (i % 4 == 0),
                    Rank = rand.Next(1, 6)
                };
            });
            return list;
        }
        public static List<string> GetCountries()
        {
            var countries = new List<string>();
            countries.AddRange(COUNTRIES);
            return countries;
        }
        public static List<string> GetProducts()
        {
            List<string> products = new List<string>();
            products.AddRange(PRODUCTS);
            return products;
        }
    }
}
```

コントローラー - PDFExportController.cs

C#

```
public ActionResult Index()
{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        return View(Sale.GetData(5));
}
```

ビュー - Index.cshtml

ビューで、MultiRowコントロールのインスタンスを作成し、[Export]ボタンを追加します。レイアウト定義のプロパティを使用すると、コントロールの列および行のレイアウトを定義できます。JavaScriptにより、FlexGridPdfConverterを使用してMultiRowコントロールをPDFファイルにエクスポートできます。

C#

```
@model IEnumerable<Sale>
@using C1.Web.Mvc
@using MultiRowPdfExport.Models

@section Scripts{
    <script>
        var multiRow;
            c1.documentReady(function () {
            multiRow = wijmo.Control.getControl("#exportPdfMultiRow");
        });

        function exportPdf() {
           wijmo.grid.pdf.FlexGridPdfConverter.export(multiRow, 'MultiRow.pdf',
{maxPages: 10});
        }
    </script>
    }
<br />
@(Html.C1().MultiRow<Sale>().Id("exportPdfMultiRow")
    .Bind(Model)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.ListBox)
    .HeadersVisibility(C1.Web.Mvc.Grid.HeadersVisibility.All)
    .ShowGroups(true)
    .GroupBy("Product", "Country")
    .CssClass("multirow custom-multi-row")
    .LayoutDefinition(ld =>
    {
       ld.Add().Cells(cells =>
        {
            cells.Add(cell => cell.Binding("ID").Header("ID"));
            cells.Add(cell => cell.Binding("Active").Header("アクティブ"));
        });
        ld.Add().Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Start").Header("開始日"));
            cells.Add(cell => cell.Binding("End").Header("終了日"));
        });
        ld.Add().Colspan(2).Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Country").Header("国").Colspan(2));
            cells.Add(cell => cell.Binding("Product").Header("製品"));
            cells.Add(cell => cell.Binding("Color").Header("色"));
        });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        ld.Add().Colspan(2).Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Amount").Header("金額"));
            cells.Add(cell => cell.Binding("Amount2").Header("金額2"));
            cells.Add(cell => cell.Binding("Discount").Header("割引").Colspan(2));
        });
    }))

<button class="btn btn-default" onclick="exportPdf()">エクスポート</button>
```

フィルタ

MultiRowは列エントリのフィルタ処理をサポートしており、ICollectionViewインタフェースを使用して有効にすることができます。これを使用して、コントロールに条件フィルタや値フィルタを適用できます。また、MultiRowは、データから目的のエントリを取得するためのフィルタ処理をサポートします。

条件フィルタを使用すると、条件を適用して検索を絞り込むことができ、**値フィルタ**を使用すると、目的の列値に対応するデータを正確に見つけることができます。[昇順]ボタンと[降順]ボタンを使用して、特定の列をそれぞれ昇順および降順にソートできます。MultiRowコントロールでフィルタ処理を有効にすると、各列の横にフィルタアイコンが表示され、これを使用して、必要に応じてデータをフィルタ処理できます。

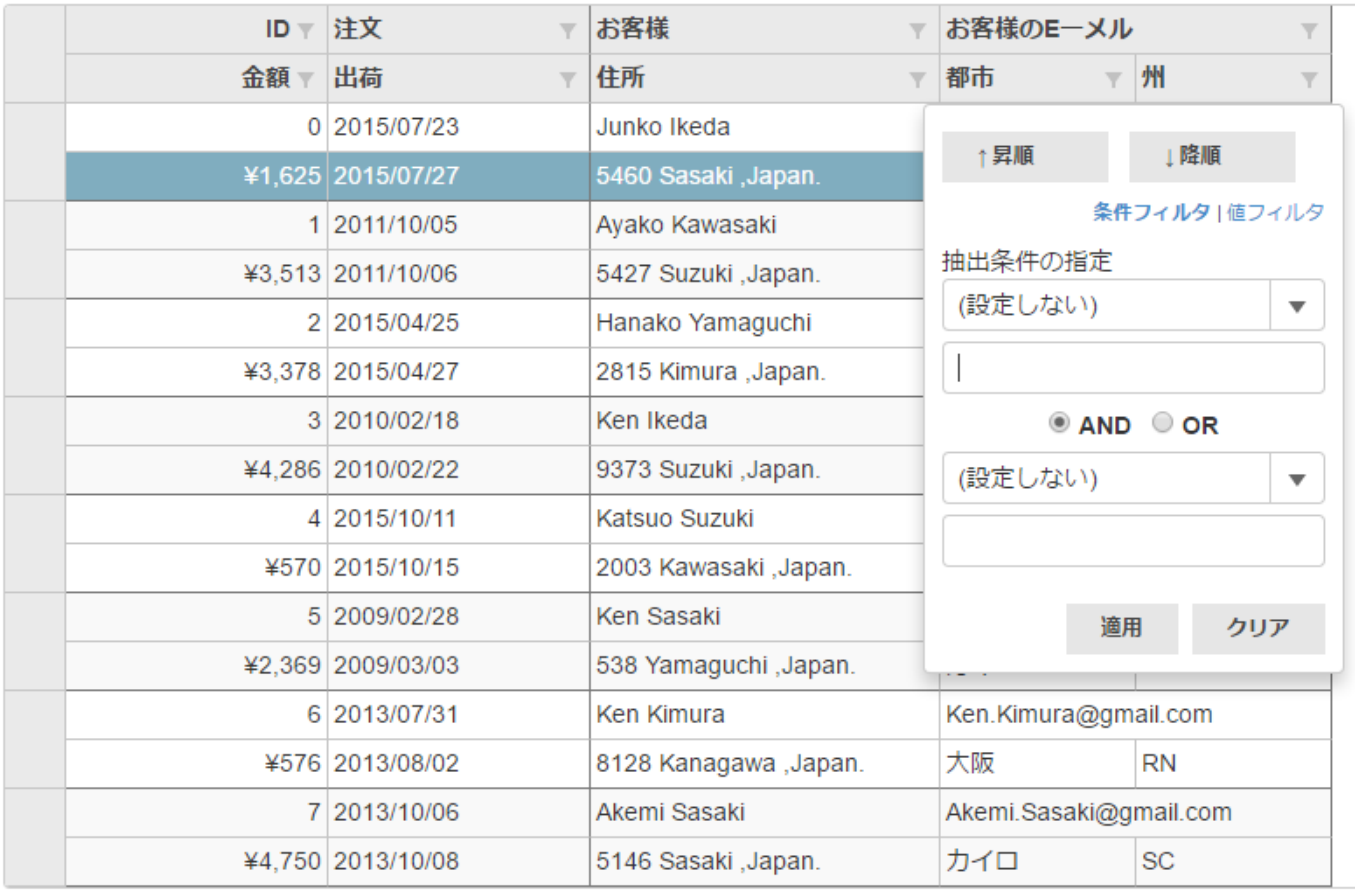
次に図は、値によるフィルタと条件によるフィルタを異なる列に適用した後のMultiRowコントロールを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

値フィルタ



条件フィルタ

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、MultiRowコントロールでフィルタ処理を使用する方法を示します。

コードの場合

Razor

```
@using MultiRow.Models
@model IEnumerable<Orders.Order>


@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .AllowSorting(true)
    .IsReadOnly(true)
    .SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)
    .Filterable(f => f.DefaultFilterType(FilterType.Both))
    .CssClass("multirow")
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

グループ化

MultiRowでは、**CollectionView**クラスをデータソースとして使用することで、グリッドデータを列方向にグループ化することができます。ビューで**GroupBy**プロパティを使用して、グループを設定できます。GroupDescriptionオブジェクトを**GroupDescriptions**プロパティに追加することで、JavaScriptを使用したデータのグループ化を行うこともできます。

MultiRowでは、**GroupHeaderFormat**プロパティを使用して、グループヘッダー行に表示されるテキストをカスタマイズすることもできます。デフォルトでは、グループの名前が表示され、その後に現在のグループとそのグループ内の項目数が表示されます。特定の列に対してグループヘッダーの集計データを書式設定するには、各Columnオブジェクトに Formatプロパティを設定します。

次の図は、状態に基づいてグループ化されたデータを含むMultiRowコントロールを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.

ID	注文	お客様	お客様のE一メル	
金額	出荷	住所	都市	州
◢ 州: **RS** (3 項目)				
0	2010/11/25	Daichi Yamaguchi	Daichi.Yamaguchi@gmail.com	
¥721	2010/11/26	4111 Kanagawa ,Japan.	岩手	RS
4	2016/02/25	Makoto Sasaki	Makoto.Sasaki@gmail.com	
¥877	2016/02/27	5455 Ikeda ,Japan.	東京	RS
5	2010/07/13	Makoto Sasaki	Makoto.Sasaki@gmail.com	
¥4,091	2010/07/16	5455 Ikeda ,Japan.	東京	RS
◢ 州: **CS** (1 項目)				
1	2011/05/16	Atsuko Kawasaki	Atsuko.Kawasaki@gmail.com	
¥3,756	2011/05/20	8333 Kimura ,Japan.	大阪	CS
◢ 州: **RT** (1 項目)				
2	2014/09/21	Daichi Sasaki	Daichi.Sasaki@gmail.com	
¥1,349	2014/09/25	2349 Kimura ,Japan.	カイロ	RT
◢ 州: **SP** (3 項目)				
3	2009/12/24	Makoto Kimura	Makoto.Kimura@gmail.com	
¥1,233	2009/12/28	7908 Kimura ,Japan.	カイロ	SP
6	2015/12/03	Katsuo Sasaki	Katsuo.Sasaki@gmail.com	
¥1,942	2015/12/04	2596 Yamaguchi ,Japan.	大阪	SP
7	2012/03/01	Ayako Kawasaki	Ayako.Kawasaki@gmail.com	
¥2,973	2012/03/05	7798 Kawasaki ,Japan.	大阪	SP

コードの場合

Grouping.cshtml

Grouping.cshtml

```
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .GroupBy("Customer.State")
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

グループパネル

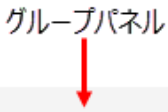
MultiRowコントロールでは、実行時に必要に応じてグリッドデータを柔軟にグループ化/グループ化解除することができます。グループパネルで、目的の列をMultiRowコントロールからドラッグアンドドロップしてグループ化することができます。

パネル内でグループを移動して、グリッドデータのグループ化の順序を変更することができます。さらに、パネル内のグループマーカーをクリックすると、特定の列エントリに基づいてグループをソートできます。

グループパネルで許容されるグループの最大数を設定するには、**MaxGroups**プロパティを設定します。**Placeholder**プロパティには、パネルにグループが含まれない場合にパネルに表示する文字列を設定します。

次の図は、MultiRowでグループパネルを使用してグループ化した列を示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.



ここでグループ化する列を追加します

	ID	注文	お客様	お客様のE一メル	
	金額	出荷	住所	都市	州
	0	2010/11/15	Makoto Kawasaki	Makoto.Kawasaki@gmail.com	
	¥136	2010/11/17	5882 Kimura ,Japan.	東京	RN
	1	2009/12/07	Makoto Suzuki	Makoto.Suzuki@gmail.com	
	¥3,323	2009/12/10	1284 Sasaki ,Japan.	岩手	RN
	2	2015/11/02	Atsuko Ikeda	Atsuko.Ikeda@gmail.com	
	¥882	2015/11/03	1451 Kawasaki ,Japan.	石川	RS
	3	2012/03/12	Atsuko Kimura	Atsuko.Kimura@gmail.com	
	¥2,000	2012/03/14	1091 Sasaki ,Japan.	大阪	RN
	4	2015/02/14	Hanako Sasaki	Hanako.Sasaki@gmail.com	
	¥3,288	2015/02/18	3998 Kawasaki ,Japan.	東京	SP
	5	2015/04/28	Hanako Kawasaki	Hanako.Kawasaki@gmail.com	
	¥1,995	2015/05/01	6327 Kanagawa ,Japan.	岩手	RS
	6	2009/02/06	Daichi Yamaguchi	Daichi.Yamaguchi@gmail.com	
	¥4,521	2009/02/07	112 Kimura ,Japan.	カイロ	RS
	7	2015/02/26	Atsuko Suzuki	Atsuko.Suzuki@gmail.com	
	¥2,920	2015/02/28	1609 Suzuki ,Japan.	カイロ	CS

Copyright © GrapeCity, Inc. All rights reserved.

州 × 都市 × ← グループマーカー

ID	注文	お客様	お客様のE一メル	
金額	出荷	住所	都市	州
◢州: **RT** (1 項目)				
◢都市: **ハンブルク** (1 項目)				
0	2009/12/02	Akemi Kanagawa	Akemi.Kanagawa@gmail.com	
¥3,628	2009/12/03	265 Sasaki ,Japan.	ハンブルク	RT
◢州: **RS** (2 項目)				
◢都市: **ハンブルク** (1 項目)				
1	2011/08/12	Katsuo Yamaguchi	Katsuo.Yamaguchi@gmail.com	
¥1,540	2011/08/15	1482 Yamaguchi ,Japan.	ハンブルク	RS
◢都市: **京都** (1 項目)				
5	2011/12/07	Akemi Kimura	Akemi.Kimura@gmail.com	
¥4,572	2011/12/08	3410 Ikeda ,Japan.	京都	RS
◢州: **SP** (2 項目)				
◢都市: **ハンブルク** (1 項目)				
2	2013/04/22	Makoto Kimura	Makoto.Kimura@gmail.com	
¥2,717	2013/04/23	1910 Kimura ,Japan.	ハンブルク	SP
◢都市: **東京** (1 項目)				
7	2016/05/18	Ken Sasaki	Ken.Sasaki@gmail.com	
¥2,451	2016/05/19	1176 Yamaguchi ,Japan.	東京	SP
◢州: **CS** (1 項目)				
◢都市: **ハンブルク** (1 項目)				
3	2013/04/11	Hanako Ikeda	Hanako.Ikeda@gmail.com	
¥4,501	2013/04/14	3567 Kanagawa ,Japan.	ハンブルク	CS

コードの場合

GroupPanel.cshtml

```
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .AllowDragging(C1.Web.Mvc.Grid.AllowDragging.Columns)
    .ShowGroupPanel(s => s
        .MaxGroups(4)
        .Placeholder("Add columns for grouping here"))
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
}))
```

ページング

MultiRowコントロールは、ページングの実装を可能にするPagerコントロールをサポートします。ページングにより、ページごとに表示する項目数をカスタマイズしたり、グリッド内でページ間を移動するためのUIを提供することができます。

Pagerコントロールを使用してMultiRowにページングを実装するには、次のプロパティを設定します。

- MultiRowコントロールの**ID**を設定します。
- **Pager.Owner**プロパティをMultiRowまたはCollectionViewServiceのIDに設定します。
- MultiRowまたCollectionViewServiceの**PageSize**プロパティを設定します。

> **メモ:** ページングUIは、グリッド外で実装されます。これにより、ページングメカニズムの外観と機能を完全に制御できます。JavaScriptを使用してPagerをカスタマイズする方法については、**CollectionView**クラスを参照してください。

PagerコントロールのOwnerプロパティを設定すると、PagerコントロールはMultiRowコントロールに連結し、ナビゲーションボタン([<<]、[<]、[>]、[>>])のクリックによってMultiRowの現在のページを変更する機能を提供します。PageSizeプロパティを使用すると、ページごとに表示する項目数を指定できます。この例では、ページングはサーバー側で行われます。これは、ここではCollectionViewがサービスのように動作して、サーバーデータと同期するためです。CollectionViewは、内部的にAjax呼び出しを行って次のデータセットを取得します。

次の図は、**PageSize**プロパティを設定した後のMultiRowコントロールを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、MultiRowコントロールでPagerコントロールを使用してページングを有効にする方法を示します。

Razor

```
@using MultiRow.Models
@model IEnumerable<Orders.Order>


@(Html.C1().CollectionViewService<Orders.Order>
().Bind(Model).Id("collectionViewService")
.PageSize(5))
@(Html.C1().Pager().Owner("collectionViewService"))
@(Html.C1().MultiRow<Orders.Order>()
    .Id("pagingMultiRow")
    .ItemsSourceId("collectionViewService").IsReadOnly(true)
    .CssClass("multirow")
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
@(Html.C1().Pager().Owner("pagingMultiRow"))
```

固定行や列

MultiRowコントロールでは、行や列を固定して、それらを常に表示したままグリッドをスクロールすることができます。行および列の固定は、ワークシートの別の領域にスクロールするときに、MultiRowの特定の領域を表示したままにするために使用されます。固定されたセルは、MultiRowコントロールの通常のセルと同様に編集したり選択することができます。

行および列の固定を使用すると、MultiRow内をスクロールする際に特定の行または列を表示したままにすることができます。たとえば、スクロール時に行や列のラベルを表示したままにしたい場合があります。

次の図は、**FrozenColumns**プロパティと**FrozenRows**プロパティを使用して3行2列を固定した後のMultiRowコントロールを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

	ID	注文	お客様	お客様のEーメル	
	金額	出荷	住所	都市	州
	0	2011/12/27	Ken Suzuki	Ken.Suzuki@gmail.com	
	¥3,140	2011/12/28	3217 Kanagawa ,Japan.	京都	SC
	1	2014/07/09	Ken Yamaguchi	Ken.Yamaguchi@gmail.com	
	¥4,772	2014/07/12	348 Kawasaki ,Japan.	東京	RN
	2	2016/02/24	Junko Yamaguchi	Junko.Yamaguchi@gmail.com	
	¥1,544	2016/02/27	3180 Ikeda ,Japan.	カイロ	RS
	3	2013/03/01	Akemi Kanagawa	Akemi.Kanagawa@gmail.com	
	¥656	2013/03/05	3267 Ikeda ,Japan.	京都	RS
	4	2014/04/13	Ken Kawasaki	Ken.Kawasaki@gmail.com	
	¥2,949	2014/04/15	5764 Suzuki ,Japan.	岩手	SC
	5	2009/12/24	Akemi Ikeda	Akemi.Ikeda@gmail.com	
	¥1,547	2009/12/26	7345 Kimura ,Japan.	岩手	CS
	6	2015/09/29	Katsuo Ikeda	Katsuo.Ikeda@gmail.com	
	¥3,774	2015/10/03	6464 Ikeda ,Japan.	京都	SC
	7	2013/02/01	Atsuko Kawasaki	Atsuko.Kawasaki@gmail.com	

コードの場合

FrozenCell.cshtml

Copyright © GrapeCity, Inc. All rights reserved.

Razor

```
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .Height(500)
    .Width(750)
    .FrozenColumns(2)
    .FrozenRows(3)
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

選択モード

Excelと同様に、MultiRowコントロールではマウスまたはキーボードを使用してセルの範囲を選択できます。**SelectionMode**プロパティを使用して、1行、行の範囲、隣接しない複数の行（リストボックスと同様）、1セル、セルの範囲を選択したり、選択を完全に無効にすることができます。

MultiRowコントロールの選択動作を指定するには、**SelectionMode**プロパティを次のいずれかに設定する必要があります。

- **Cell**：SelectionModeを`Cell`に設定すると、ユーザーは一度に1つのセルだけを選択できます。
- **CellRange**：SelectionModeを`CellRange`に設定すると、ユーザーは連続するセルのブロックを選択できます。
- **ListBox**：SelectionModeを`ListBox`に設定すると、ユーザーは隣接しない複数の行を選択できます。
- **None**：SelectionModeを`None`に設定すると、ユーザーはマウスやキーボードでセルを選択することはできません。
- **Row**：SelectionModeを`Row`に設定すると、ユーザーは一度に1つの行を選択できます。
- **RowRange**：SelectionModeを`RowRange`に設定すると、ユーザーは隣接する行を選択できます。

次の図は、**SelectionMode**を**Row**に設定したときのMultiRowコントロールを示しています。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.



ID	注文	お客様	お客様のE一メル	
金額	出荷	住所	都市	州
0	2015/05/19	Ayako Kawasaki	Ayako.Kawasaki@gmail.com	
¥1,819	2015/05/23	1823 Ikeda ,Japan.	東京	RS
1	2013/01/06	Makoto Ikeda	Makoto.Ikeda@gmail.com	
¥568	2013/01/07	3978 Kimura ,Japan.	京都	RS
2	2012/01/02	Atsuko Kimura	Atsuko.Kimura@gmail.com	
¥2,190	2012/01/06	3122 Yamaguchi ,Japan.	石川	RN
3	2016/07/25	Makoto Yamaguchi	Makoto.Yamaguchi@gmail.com	
¥338	2016/07/29	8717 Kanagawa ,Japan.	カイロ	SC
4	2015/11/06	Makoto Ikeda	Makoto.Ikeda@gmail.com	
¥2,827	2015/11/07	2911 Ikeda ,Japan.	東京	RN
5	2009/02/04	Akemi Kawasaki	Akemi.Kawasaki@gmail.com	
¥1,644	2009/02/08	4815 Ikeda ,Japan.	京都	SP
6	2013/07/08	Ayako Kawasaki	Ayako.Kawasaki@gmail.com	
¥2,809	2013/07/09	654 Suzuki ,Japan.	カイロ	SC
7	2011/02/27	Ken Kimura	Ken.Kimura@gmail.com	

コードの場合

SelectionMode.cshtml

Razor

```
.SelectionMode(C1.Web.Mvc.Grid.SelectionMode.Row)
```

スタイル設定とCSS

CSS classes for the MultiRow control are as follows:

Class	Description
.wj-cell.wj-header	Styles the header cells including column header, row header and top left cell.
.wj-multirow .wj-hdr-collapse	Styles the collapse column header icon displayed at the right edge of the last column header cell.
.wj-multirow .wj-hdr-collapse:hover	Styles the collapse column header icon when a user hovers the mouse over it.
.wj-cell	Styles all the cells in the MultiRow control.
.wj-cell.wj-alt	Styles the alternate rows in the MultiRow control.
.wj-multirow .wj-cell.wj-record-end:not(.wj-header)	Styles the line between each record.
.wj-multirow .wj-cell.wj-group-end	Styles the line between groups.

Copyright © GrapeCity, Inc. All rights reserved.

.wj-cell:not(.wj-hasdropdown)	Styles the overflow text in cell
Selection	
.wj-cells .wj-cell.wj-state-selected	Styles the currently selected cell in the MultiRow control.
.wj-cells .wj-cell.wj-state-multi-selected	Styles the currently multiple selected cell in the MultiRow control.
.wj-flexgrid .wj-marquee	Styles the Excel-style element that highlights the selection when the **showMarquee** property is set to true.
.wj-state-active	Styles the active cell.
.wj-cell.wj-header.wj-state-multi-selected	Styles the selected cell headers when the **showSelectedHeaders** or/and **showSelectedHeaders** property is set to true.
.wj-flexgrid .wj-colheaders .wj-header.wj-state-multi-selected	Styles the selected column headers when the **showSelectedHeaders** property is set to true.
.wj-flexgrid .wj-colheaders .wj-header.wj-state-multi-selected	Styles the selected row headers when the **showSelectedHeaders** property is set to true.
Editing	
.wj-flexgrid .wj-grid-editor	Styles the cell editor element.
.wj-glyph-pencil	Styles the pencil icon displayed in the row header of the currently editing row.
.wj-glyph-asterisk	Styles the asterisk icon displayed in the row header of the new row template.
Freezing	
.wj-frozen	Styles non-header frozen cells.
.wj-cell.wj-frozen-row	Styles cells in the last frozen row. This style can be used to create a horizontal divider line between the frozen and scrollable rows.
.wj-cell.wj-frozen-col	Styles cells in the last frozen column. This style can be used to create a vertical divider line between the frozen and scrollable columns.
Sorting	
.wj-glyph-up	Styles the up-arrow icon displayed in the column header of a column sorted in ascending order.
.wj-glyph-down	Styles the down-arrow icon displayed in the column header of a column sorted in descending order.
Grouping	
.wj-cell.wj-group	Styles the group row.
.wj-glyph-down-right	Styles the expand icon displayed in the group row for the expanded group.
.wj-glyph-right	Styles the collapse icon displayed in the group row for the collapsed group.
.wj-grouppanel	Styles the div host element for GroupPanel control.
.wj-grouppanel .wj-groupmarker	Styles the div element for each grouped column displayed in the GroupPanel control.
.wj-grouppanel .wj-groupmarker .wj-remove	Styles the (x) mark displayed in the group panel for each column header element.
.wj-glyph-drag	Styles the drag icon for each grouped column displayed in the GroupPanel

Copyright © GrapeCity, Inc. All rights reserved.

	control.
Filtering	
.wj-glyph-filter	Styles the filter icon displayed in column headers.
.wj-filter-on	Styles the filter icon when the column has been filtered.
.wj-filter-off	Styles the filter icon when the column has not been filtered.
.wj-columnfiltereditor	Styles the div host element for FilerDialog.
.wj-valuefilter-editor	Styles the div host element for Value Filter.
.wj-conditionfilter-editor	Styles the div host element for Condition Filter.
Filtering	
.wj-cell.input-validation-error	Styles the cell editor with invalid input value.
.wj-cell.wj-state-invalid	Styles the cells with invalid value.

レコード、グループ、セルのスタイル設定

MultiRowコントロールでは、独自のカスタムCSSファイルを作成し、それをアプリケーションに追加することができます。多くのアプリケーションでは、各レコードやグループがどこで始まりどこで終わるかを示しておくのが普通です。MultiRowコントロールは、グループごとに最初と最後の行/列にあるセル要素にCSSクラス名を追加することで、これを可能にしています。使用するクラス名は、**wj-record-start**、**wj-record-end**、**wj-group-start**、**wj-group-end**です。これらの組み込みクラス名をCSSルールで使用して、レコード区切りやグループ区切りの外観をカスタマイズできます。

次の例は、これらのクラス名をCSSルールで使用して、レコード区切りとグループ区切りの外観をカスタマイズする方法を示します。また、標準の**CssClass**プロパティを使用して、グループ内の特定のセルをカスタマイズする方法も示します。

次の図は、CSSクラス名とCssClassプロパティを使用した後のMultiRowを示します。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.

ID	注文	お客様	お客様のE一メル	
金額	出荷	住所	都市	州
0	2012/01/23	Katsuo Kawasaki	Katsuo.Kawasaki@gmail.com	
¥4,788	2012/01/25	7523 Kanagawa ,Japan.	石川	SP
1	2016/05/03	Atsuko Kanagawa	Atsuko.Kanagawa@gmail.com	
¥1,196	2016/05/07	9971 Ikeda ,Japan.	ハンブルク	SP
2	2015/04/17	Akemi Suzuki	Akemi.Suzuki@gmail.com	
¥4,013	2015/04/18	1820 Kawasaki ,Japan.	大阪	CS
3	2009/12/21	Ayako Kawasaki	Ayako.Kawasaki@gmail.com	
¥4,358	2009/12/25	1872 Suzuki ,Japan.	東京	RS
4	2012/05/03	Ken Kimura	Ken.Kimura@gmail.com	
¥3,710	2012/05/04	7567 Kanagawa ,Japan.	カイロ	RN
5	2009/03/08	Akemi Kanagawa	Akemi.Kanagawa@gmail.com	
¥4,577	2009/03/12	5605 Kawasaki ,Japan.	石川	CS
6	2015/07/22	Ken Kimura	Ken.Kimura@gmail.com	
¥3,021	2015/07/26	7567 Kanagawa ,Japan.	カイロ	RN
7	2012/12/06	Junko Suzuki	Junko.Suzuki@gmail.com	
¥3,793	2012/12/08	7300 Kanagawa ,Japan.	石川	RS

CustomMultiRow.cssの追加

新しいASP.NET MVCアプリケーションを作成します。アプリケーションを作成し、アプリケーションにビューを追加すると、ソリューションエクスプローラーに**[コンテンツ]**フォルダが作成されます。アプリケーションにカスタムスタイルシートを追加するには、次の手順に従います。

1. ソリューションエクスプローラーで、**[コンテンツ]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[スタイルシート]**を選択します。**[項目の名前を指定]**ダイアログが表示されます。
3. スタイルシートの名前を設定します（例：`CustomMultiRow.css`）。
4. **[OK]**をクリックして、CustomMultiRow.cssスタイルシートをアプリケーションに追加します。
5. **CustomMultiRow.css**ファイルのデフォルトのコードを次のコードに置き換えます。

CustomMultiRow.css

```
/* MultiRowのカスタムスタイル設定 */
.multirow-css .wj-cell.wj-record-end:not(.wj-header) {
    border-bottom-color: #8fabff; /* レコード間に青い線を設定する*/
}
.multirow-css .wj-cell.wj-group-end {
    border-right-color: #bc5505; /* グループ間に茶色の線を設定する*/
}
.multirow-css .wj-cell.id {
    color: #c0c0c0;
}
.multirow-css .wj-cell.amount {
    color: #014701;
    font-weight: bold;
}
.multirow-css .wj-cell.email {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    color: #0010c0;
    text-decoration: underline;
}
@@font-face {
    font-family: 'Fira';
    src: url("../fonts/fira/FiraSans-Regular.ttf");
    font-weight: normal;
    font-style: normal;
}
@@font-face {
    font-family: 'Fira';
    src: url("../fonts/fira/FiraSans-Bold.ttf");
    font-weight: bold;
    font-style: normal;
}
.custom-multi-row .wj-cell {
    font-family: Fira;
}
```

コードの場合

Styling.cshtml

Razor

```
@using MultiRow.Models
@model IEnumerable<Orders.Order>

@section Styles{
    <link rel="stylesheet" href="~/Content/CustomMultiRow.css" />
}
<br />
@(Html.C1().MultiRow<Orders.Order>()
    .Bind(bl => bl.Bind(Model))
    .Id("stylingMultiRow")
    .CssClass("multirow multirow-css")
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
```


Copyright © GrapeCity, Inc. All rights reserved.

```
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

仮想スクロール

MultiRowコントロールは、大量のデータを使用する際に仮想スクロールをサポートします。MultiRowコントロールをモデルや他のデータソースを使用した大規模なデータセットと連結して、ちらつきや遅延のないスムーズなスクロールを行うことができます。

次の図は、大規模なデータセットに仮想スクロールを適用した後のMultiRowコントロールを示しています。この例では、「クイックスタート」トピックで作成したサンプルを使用しています。

ID	注文	お客様	お客様のE一メル	
金額	出荷	住所	都市	州
0	2010/05/16	Junko Kimura	Junko.Kimura@gmail.com	
¥196	2010/05/19	3358 Yamaguchi ,Japan.	岩手	RS
1	2015/03/07	Daichi Yamaguchi	Daichi.Yamaguchi@gmail.com	
¥2,253	2015/03/11	7089 Ikeda ,Japan.	東京	CS
2	2014/02/10	Hanako Kimura	Hanako.Kimura@gmail.com	
¥4,993	2014/02/11	8546 Kawasaki ,Japan.	東京	CS
3	2012/01/05	Ken Kawasaki	Ken.Kawasaki@gmail.com	
¥3,487	2012/01/08	4521 Ikeda ,Japan.	カイロ	RT
4	2015/09/06	Daichi Kanagawa	Daichi.Kanagawa@gmail.com	
¥1,340	2015/09/07	3910 Kimura ,Japan.	ハンブルク	CS
5	2009/11/22	Makoto Ikeda	Makoto.Ikeda@gmail.com	
¥1,419	2009/11/24	8643 Kawasaki ,Japan.	東京	RT
6	2012/12/02	Ken Ikeda	Ken.Ikeda@gmail.com	
¥3,559	2012/12/04	8292 Kimura ,Japan.	ハンブルク	RT
7	2013/04/27	Atsuko Ikeda	Atsuko.Ikeda@gmail.com	

コードの場合

VirtualScroll.cshtml

MultiRowで仮想スクロールを使用するには、**DisableServerRead**プロパティを**false**（デフォルト）に設定し、**InitialItemsCount**プロパティを0より大きい値に設定します。

VirtualScroll.cshtml

```
@(Html.C1().MultiRow<Orders.Order>()
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .Bind(bl => bl.InitialItemsCount(10).Bind(Model).DisableServerRead(false))
    .IsReadOnly(true)
    .Height(500)
    .Width(780)
    .CssClass("multirow")
    .LayoutDefinition(ld =>
    {
  ld.Add().Header("注文").Colspan(2).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("注文").Width("150"))
            .Add(cell => cell.Binding("Amount").Header("金
額").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("出荷"));
        });
        ld.Add().Header("お客様").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("お客様").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("お客様のEーメ
ル").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("住所"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("都市"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("州"));
        });
    }))
```

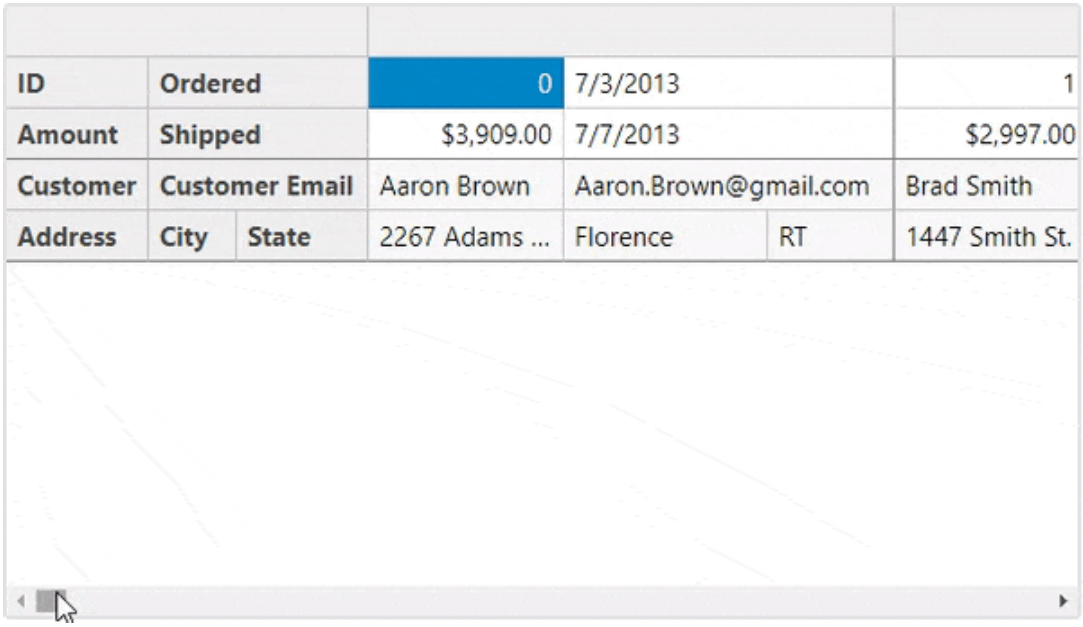
TransposedMultiRow

MultiRow for ASP.NET MVC supports a transposed view, which is another way to show tabular data and especially useful when there are few data items. The transposed multi-row aspect repeats the records horizontally instead of vertically like in a traditional grid.

The **TransposedMultiRow** control extends the FlexGrid control to display data using a transposed layout, where columns represent data items and rows represent item properties, similar to the TransposedGrid control. But in contrast to the TransposedGrid control, the TransposedMultiRow control uses multiple columns to represent each data item. This allows you to create form-like interfaces that can display a large number of rows with minimal vertical scrolling.

The TransposedMultiRow control is represented by the **TransposedMultiRow** class. The **TransposedMultiRow** class allows you to customize the grid layout using the **LayoutDefinition** property, just like the MultiRow control. This property specifies the layout of the cells in the grid, just like the MultiRow control but the resulting view looks different because of transposing. It contains an array of cell group objects. Each cell group specifies how many columns the group should span, and the cells that make up each group.

Copyright © GrapeCity, Inc. All rights reserved.



The following example demonstrates the use of a TransposeMultiRow control for displaying data using a transposed layout. This view uses two rows per record and the layout is divided into three groups: order, customer, and shipper.

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Create a Data Source for TransposedMultiRow

1. Add a new class to the **Models** folder (for example: `Orders.cs`). For more information on how to add a new model, see Adding Controls.
2. Add the following code to `Orders.cs` model. We are using **`Orders`** class to represent sales order data in the database. Each instance of **Orders** object will correspond to a record on MultiRow control.

Orders.cs copyCode

```
public class Orders
{
    private static object _lockObj = new object();
    private static Random Rand = new Random();

    private static IList<Order> _orders;
    private static IList<string> _cities;
    private static IList<Customer> _customers;
    private static IList<Shipper> _shippers;

    public class Shipper
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Email { get; set; }
        public string Phone { get; set; }
        public bool Express { get; set; }
    }

    public class Customer
    {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public int Id { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string Zip { get; set; }
    public string Email { get; set; }
    public string Phone { get; set; }
}

public class Order
{
    public int Id { get; set; }
    public DateTime Date { get; set; }
    public DateTime ShippedDate { get; set; }
    public double Amount { get; set; }
    public Customer Customer { get; set; }
    public Shipper Shipper { get; set; }
}

public static IList<Order> GetOrders()
{
    if (_orders == null)
    {
        lock (_lockObj)
        {
            if (_orders == null)
            {
                var today = DateTime.Now.Date;
                var customers = GetCustomers();
                var shippers = GetShippers();

                _orders = new List<Order>();
                for (int i = 0; i < 100; i++)
                {
                    var shipped = today.AddDays(-Rand.Next(-1, 3000));
                    var order = new Order
                    {
                        Id = i,
                        Date = shipped.AddDays(-Rand.Next(1, 5)),
                        ShippedDate = shipped,
                        Amount = Rand.Next(10000, 500000) / 100,
                        Customer = customers[Rand.Next(0, customers.Count - 1)],
                        Shipper = shippers[Rand.Next(0, shippers.Count - 1)]
                    };
                    _orders.Add(order);
                }
            }
        }
    }
    return _orders;
}

public static IList<Shipper> GetShippers()
{
    if(_shippers == null)
    {
        lock (_lockObj)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            {
                if (_shippers == null)
                {
                    _shippers = new List<Shipper>();
                    _shippers.Add(new Shipper { Id = 0, Name = "Speedy Express", Email =
"speedy@gmail.com", Phone = "431-3234", Express = true });
                    _shippers.Add(new Shipper { Id = 1, Name = "Flash Delivery", Email =
"flash@gmail.com", Phone = "431-6563", Express = true });
                    _shippers.Add(new Shipper { Id = 2, Name = "Logitrax", Email =
"logitrax@gmail.com", Phone = "431-3981", Express = false });
                    _shippers.Add(new Shipper { Id = 3, Name = "Acme Inc", Email =
"acme@gmail.com", Phone = "431-3113", Express = false });
                }
            }
        }
        return _shippers;
    }

    public static IList<Customer> GetCustomers()
    {
        if (_customers == null)
        {
            lock (_lockObj)
            {
                if (_customers == null)
                {
                    var firstNames = new[] { "Aaron", "Paul", "John", "Mark", "Sue", "Tom",
"Bill", "Joe", "Tony", "Brad", "Frank", "Chris", "Pat" };
                    var lastNames = new[] { "Smith", "Johnson", "Richards", "Bannon", "Wong",
"Peters", "White", "Brown", "Adams", "Jennings" };
                    var cities = GetCities();
                    var states = new[] { "SP", "RS", "RN", "SC", "CS", "RT", "BC" };

                    _customers = new List<Customer>();
                    for (int i = 0; i < 50; i++)
                    {
                        var first = firstNames[Rand.Next(0, firstNames.Length - 1)];
                        var last = lastNames[Rand.Next(0, lastNames.Length - 1)];
                        var customer = new Customer
                        {
                            Id = i,
                            Name = first + " " + last,
                            Address = Rand.Next(100, 10000) + " " + lastNames[Rand.Next(0,
lastNames.Length - 1)] + " St.",
                            City = cities[Rand.Next(0, cities.Count - 1)],
                            State = states[Rand.Next(0, states.Length - 1)],
                            Zip = string.Format("{0:d5}-{1:d3}", Rand.Next(10000, 99999),
Rand.Next(100, 999)),
                            Email = first + "." + last + "@gmail.com",
                            Phone = string.Format("{0:d3}-{1:d4}", Rand.Next(100, 999),
Rand.Next(1000, 9999))
                        };
                        _customers.Add(customer);
                    }
                }
            }
        }
        return _customers;
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public static IList<string> GetCities()
    {
        if (_cities == null)
        {
            lock (_lockObj)
            {
                if(_cities == null)
                {
                    _cities = new[] { "York", "Paris", "Rome", "Cairo", "Florence", "Sidney",
"Hamburg", "Vancouver" };
                }
            }
        }
        return _cities;
    }
}
```

Back to Top

Add the TransposedMultiRow Control

To add the TransposedMultiRow control to the application, add the following references and follow the steps below:

- C1.Web.Mvc
- C1.Web.Mvc.MultiRow
- C1.Web.Mvc.TransposedMultiRow

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `TransposedMultiRowController`).
    3. Click **Add**.
4. Replace the method **Index()** with the following method.

Index.cshtml copyCode

```
public ActionResult Index()
{
    var model = Orders.GetOrders();
    return View(model);
}
```

Add a View for the Controller

In the view, we create an instance of TransposedGridMultiRow control and bind it to a data source using **Bind** property. The layout definition property helps us to define the column and row layout of the control.

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `TransposedMultiRowController.`
2. Place the cursor inside the method `Index().`
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller. Copy the following code and paste it inside **Index.cshtml**.

Index.cshtml copyCode

```
@model IEnumerable<Orders.Order>

<h3>Transposed MultiRow</h3>
@(Html.C1().CollectionViewService().Id("itemsSource").Bind(Model).DisableServerRead(true))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().TransposedMultiRow<Orders.Order>()
    .Id("ovTransposedMultiRow")
    .Bind(bl => bl.Bind(Model))
    .LayoutDefinition(ld =>
    {
        ld.Add().Header("Order").Colspan(2).Cells(cells =>
        {
            cells.Add(cell => cell.Binding("Id").Header("ID").CssClass("id").Width("150"))
            .Add(cell => cell.Binding("Date").Header("Ordered").Width("150"))
            .Add(cell =>
cell.Binding("Amount").Header("Amount").Format("c").CssClass("amount"))
            .Add(cell => cell.Binding("ShippedDate").Header("Shipped"));
        });
        ld.Add().Header("Customer").Colspan(3).Cells(cells =>
        {
            cells.Add(cell =>
cell.Binding("Customer.Name").Name("CustomerName").Header("Customer").Width("200"))
                .Add(cell =>
cell.Binding("Customer.Email").Name("CustomerEmail").Header("Customer Email").Colspan(2))
                .Add(cell =>
cell.Binding("Customer.Address").Name("CustomerAddress").Header("Address"))
                .Add(cell =>
cell.Binding("Customer.City").Name("CustomerCity").Header("City"))
                .Add(cell =>
cell.Binding("Customer.State").Name("CustomerState").Header("State"));
        });
    })
    .CssClass("multirow")
)
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/TransposedMultiRow/Index) in the address bar of the browser to see the view.

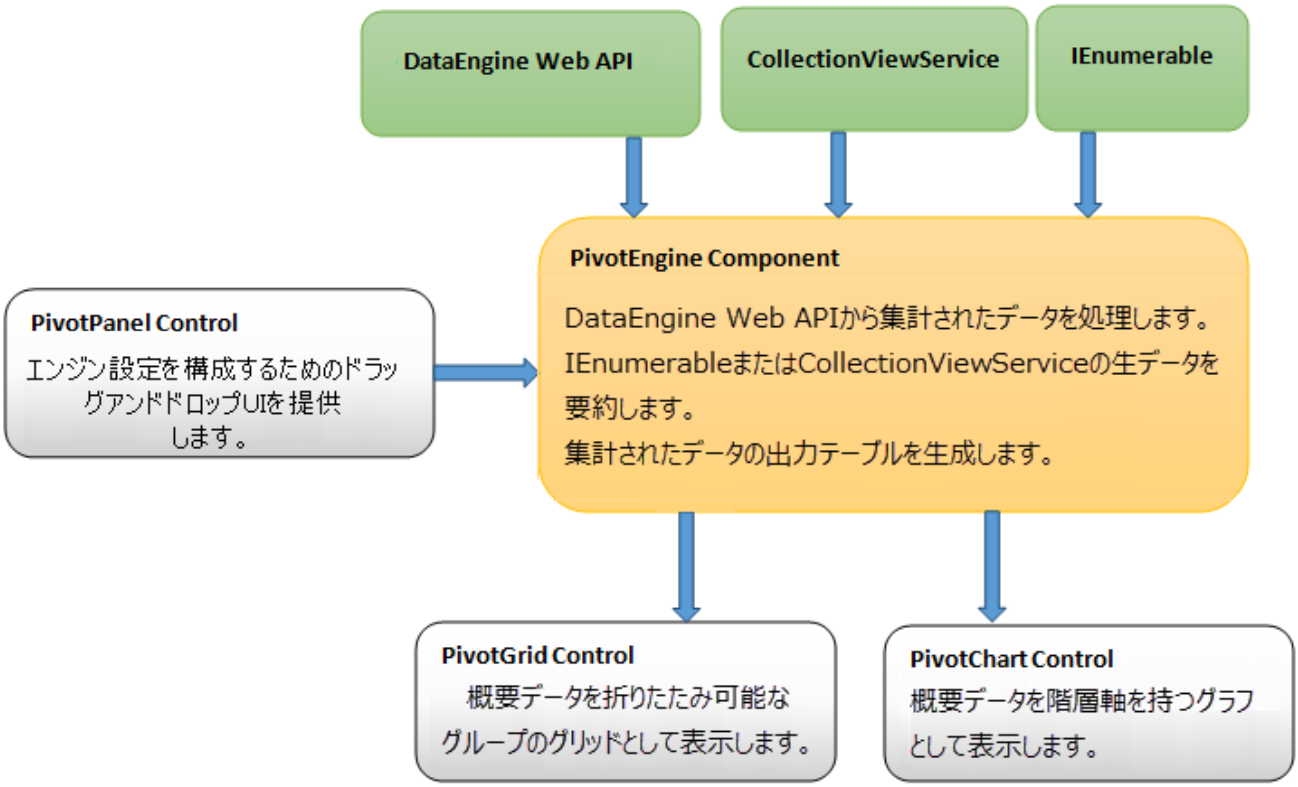
Back to Top

OLAP

ComponentOne ASP.NET MVCコントロールは、OLAPを導入しています。OLAPは、ハイエンドなビジネスインテリジェンスのニーズに対応するように設計および開発された強力なデータ分析コントロールです。Microsoft Excelのピボットテーブルやピボットチャートにある分析処理機能と同様の機能を提供します。OLAPは、複数のプロセスを別々のスレッドで同時に実行することで、コントロールのパフォーマンスを向上させる非同期処理を提供します。

OLAPは、任意の形式のデータを取得し、使いやすいインタフェースを備えているため、ユーザーはさまざまな方法でデータを表示するサマリーをすばやく直観的に作成できます。そのため、対話的に傾向を明らかにし、貴重な洞察を得ることができます。OLAPは、データベースからのデータの取得を高速化するように設計されたピボットエンジンを使用しています。これにより、大量のデータを簡単に処理できます。

Copyright © GrapeCity, Inc. All rights reserved.



OLAPは、オンライン分析処理に次のコンポーネントを使用します。

1. **PivotPanel** - PivotPanelは、OLAPの中核的なコンポーネントです。通常のデータテーブルをPivotTable、PivotGrid、PivotChartとして対話的に表すためのユーザーインタフェースを提供します。PivotPanelにデータベースを渡すだけで、PivotTableにデータを表示し、要件に応じてデータを要約するカスタムビューを提供することができます。

2. **PivotGrid** - PivotGridは、既存の**FlexGrid**クラスの拡張クラスです。PivotTableを作成し、OLAPコントロール内でFlexGridの機能を使用するために役立ちます。自動データ連結、行と列のグループ化、列のサイズ変更、データのフィルタ処理、グリッド内の各セルの詳細の表示など、MVC OLAPで活用できるさまざまなFlexGridの機能があります。

3. **PivotChart** - PivotChartコンポーネントは、**FlexChart**クラスの拡張クラスです。PivotPanelオブジェクトに自動的にデータを連結し、自動ツールチップ、チャートタイプの選択、パレットの選択などを利用できます。これらのチャートをPNGやJPEGなどのファイル形式にエクスポートしたり、チャートスタイルや対話性を要件に応じてカスタマイズすることができます。

4. **PivotEngine** - このデータエンジンは、OLAPアプリケーションに簡単に統合できるフットプリントが小さいC#コンポーネントです。このエンジンは、データをメモリマップファイルに保存するため、ファイルがインポートされると、遅延なく直ちにデータが取得されます。ピボットエンジンは、分析または表示するデータセットのサイズに制限がなく、それによってパフォーマンスが損なわれることもありません。

主な特長

OLAP for MVC には主に次の便利な機能があります。

- **複数の値フィールド**
  OLAPは、値リストで複数のフィールドをサポートします。ユーザーは、複数のフィールドを値リストにドラッグして、結果をグリッドやチャートで確認できます。

- **拡張された強力なデータエンジン**
  OLAPには、列指向技術を使用してデータをメモリマップファイルに保存する新しい強力なデータエンジンが組み込まれ

Copyright © GrapeCity, Inc. All rights reserved.

ています。C1DataEngineは、大規模なデータセットの高速処理を提供しており、数億件のレコードを1秒足らずで処理できる高いパフォーマンスを実現しています。また、このエンジンは、集計、結合、グループ化、フィルタ処理、単項/2項演算などのクエリー操作もサポートしています。詳細については、「データ連結」を参照してください。

- **強化されたユーザーインタフェース**
  OLAPは、Excel風の高度なユーザーインタフェースを備え、ツールバー、ドロップダウンメニュー、テーブル、チャートにモダンな配色を提供しています。

- **複数テーマのサポート**
  OLAPでは、開発者がアプリケーションの要件に適したテーマを選択できます。複数のテーマをサポートするために、ASP.NET MVCコントロールには、組み込みテーマがサポートされています。

- **PivotGridのピボットデータ**
  OLAPは、グリッド形式のビューにデータを表示するPivotGridコントロールをサポートします。FlexGridクラスライブラリを使用して、OLAPアプリケーションをカスタマイズできます。FlexGridコントロールの詳細については、FlexGridのトピックを参照してください。

- **実行時のデータの表示**
  PivotPanelを使用して、データソースのどのフィールドをどのように使用してデータを表示するかを決定します。PivotPanelの下部にある領域間でフィールドをドラッグして、フィルタ、列ヘッダー、行ヘッダーを作成したり、列や行の値の合計を取得します。詳細については、PivotPanelのセクションを参照してください。

- **クエリー操作のサポート**
  OLAP for MVCでは、集計、結合、フィルタ処理、範囲、ソート、グループ化、単項および2項演算、計算フィールド、投影などのクエリー操作がサポートされています。

OLAP の概要

従来の分析処理ツールには、**OLAPキューブ**やMicrosoft Excelなどに用意されているピボットテーブルがあります。これらのツールは、大規模なデータセットを取得し、条件セットに基づいてレコードをグループ化して、データを集計します。たとえば、OLAPキューブは、製品、地域、期間に基づいて売上データをグループ化することで売上データをまとめます。この場合、各グリッドセルには、特定の製品、特定の地域、および特定の期間の売上の合計が表示されます。通常、このセルは、元のデータソースのいくつかのレコードからのデータを表します。

データ分析処理ツールを使用すると、グループ化の基準を動的に(オンラインで)再定義できます。これにより、アドホックなデータ分析の実行や、隠れたパターンの発見が容易になります。

たとえば、次の表を見てください。

日付	製品	地域	売上
2015年10月	製品A	北部	12
2015年10月	製品B	北部	15
2015年10月	製品C	南部	4
2015年10月	製品A	南部	3
2015年11月	製品A	南部	6
2015年11月	製品C	北部	8
2015年11月	製品A	北部	10
2015年11月	製品B	北部	3

ここで、このデータを分析して次のような質問の回答を得るように求められたと仮定します。

Copyright © GrapeCity, Inc. All rights reserved.

- 売上は上がっているか、下がっているか。
- 会社にとってどの製品が最も重要であるか。
- 各地域で最も人気があるのはどの製品か。

これらの単純な質問に答えるには、これらのデータを集計し、次のような表を取得する必要があります。

日付別および製品別の売上

日付	製品A	製品B	製品C	合計
2007年10月	15	15	4	34
2007年11月	16	3	8	27
合計	**31**	**18**	**12**	**61**

製品別および地域別の売上

製品	北部	南部	合計
製品A	22	9	31
製品B	18		18
製品C	8	4	12
合計	**48**	**13**	**61**

サマリーテーブルの各セルは、元のデータソースのいくつかのレコードを表しています。1つまたは複数の値フィールドが集計され（ここでは、売上の合計）、他のフィールド（ここでは、日付、製品、または地域）に基づいて分類されています。

これは、スプレッドシートでは簡単に行うことができますが、作業は単調で、繰り返しが多く、ミスが多くなります。データを集計するためのカスタムアプリケーションを記述しても、新しいビューを追加するためのアプリケーションの管理に多くの時間がかかり、ユーザーの分析は実装されているビューに限定されます。

OLAPでは、ユーザーがアドホックに対話操作で必要なビューを定義できます。定義済みのビューを使用することも、新しいビューを作成して保存することもできます。基底のデータの変更はビューに自動的に反映されます。ユーザーはこれらのビューを表示するレポートを作成して共有できます。つまり、**OLAP**コントロールは、柔軟で効率的なデータ分析処理を提供します。

OLAP のアーキテクチャ

このセクションは、OLAPコントロールのアーキテクチャ機能を中心に説明します。OLAPは、強力なデータ分析コントロールです。主に次の3つのコンポーネントが含まれます。

PivotPanel
: OLAPコントロールのピボットパネルの各要素について学習します。

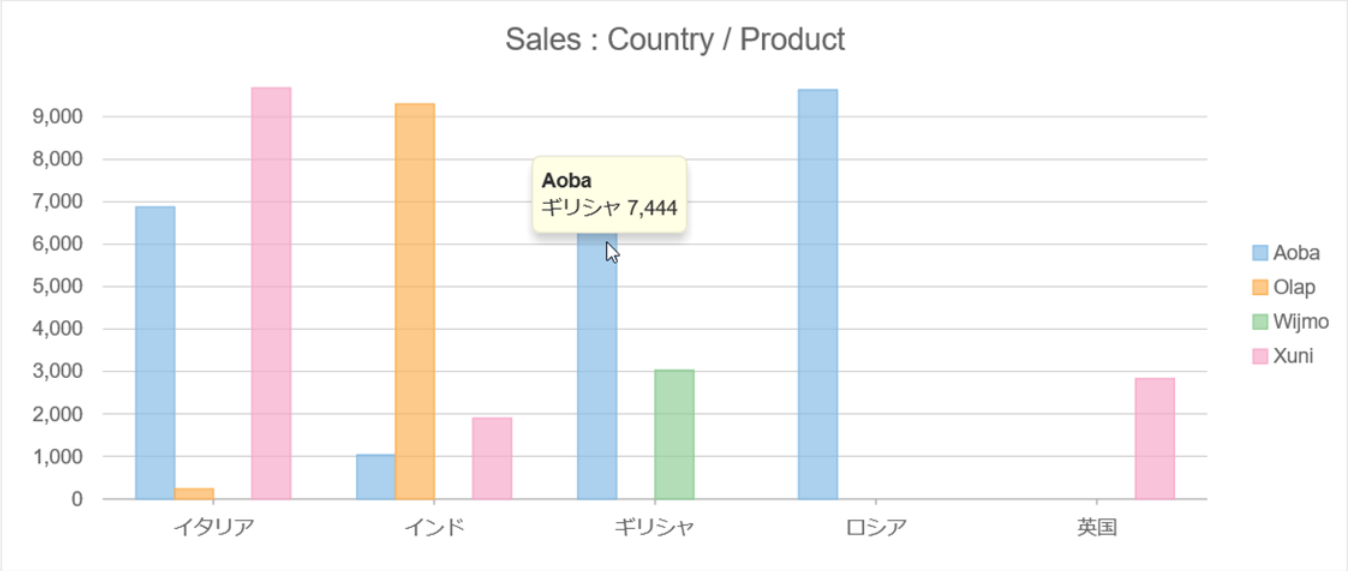
PivotChart
: OLAPコントロールのPivotChartの操作について学習します。

PivotGrid
: OLAPコントロールのPivotGridの操作について学習します。

Pivot Chart

PivotChartは、データソースからフェッチされたデータをピボットチャートの形式で表示します。PivotChartコントロールをデータソースに連結して、チャートにデータを挿入することができます。このコントロールは、PivotPanelオブジェクトとの自動データ連結、自動ツールチップ、チャートタイプ、統合されたパレットなどの機能を提供します。

Copyright © GrapeCity, Inc. All rights reserved.



これらのチャートをPNGおよびJPEGファイル形式にエクスポートしたり、チャートスタイルや対話性を要件に応じてカスタマイズすることができます。PivotChartの詳細については、「クイックスタート：OLAPへのデータの追加」を参照してください。

Pivot Grid

PivotGridは、データソースからフェッチされたデータをピボットテーブルの形式で表示します。PivotGridをデータソースに連結して、グリッドに順序付きデータセットを挿入することができます。このコントロールは、PivotPanelオブジェクトとの自動データ連結、行ヘッダーと列ヘッダーのグループ化、カスタム動作による列のサイズ変更、クリップボードへのデータのコピー、指定されたセルの詳細情報の表示などの機能を提供します。

Country	Aoba	Olap	Wijmo	Xuni	総計
イタリア	6,870	238		9,672	**16,780**
インド	1,041	9,293		1,904	**12,238**
ギリシャ	7,444		3,034		**10,478**
ロシア	9,628				**9,628**
英国				2,833	**2,833**
総計	**24,983**	**9,531**	**3,034**	**14,409**	**51,957**

PivotGridコントロールは、PivotGridコントロールの機能を拡張しています。これを使用して、グリッドのコンテンツをExcelシートにエクスポートしたり、スタイルとオーナー描画セルを使用してグリッドの全体的な外観をカスタマイズすることができます。PivotChartの詳細については、「クイックスタート：OLAPへのデータの追加」を参照してください。

Pivot Panel

PivotPanelコンポーネントは、データのカスタムビューを定義するためエクセル風のドラッグアンドドロップユーザーインタフェースを提供します。このコントロールには、データに存在するすべてのフィールドのリストが表示されます。これらのフィールドを、出力テーブルの行/列ディメンション、出力データセルの集計値、データのフィルタ処理に使用されるフィールドなどを表すリストにドラッグアンドドロップできます。When users modifies all these aspects of the view definition, the engine automatically updates the output data summary. Additionally, PivotPanel also manages the deferred update state of the engine.

Copyright © GrapeCity, Inc. All rights reserved.



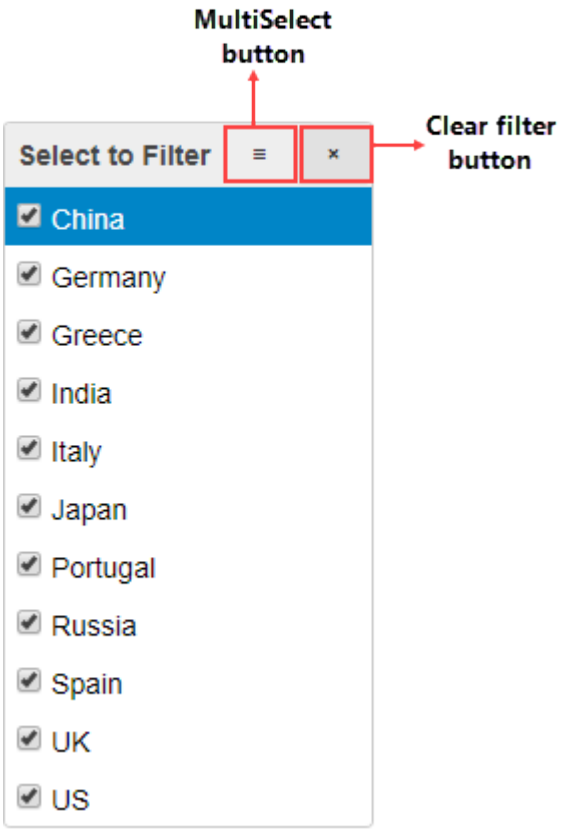
PivotPanelを使用する場合、このコンポーネントの中心は、ユーザーが選択した条件に従ってデータソースから生データをフェッチする強力なデータエンジンです。データ選択の条件は、パネルに表示されるさまざまなリストに含まれているフィールドによって決定されます。OLAPは、オープンアーキテクチャです。テーブル、汎用リストなどの標準的なコレクションをデータとして受け取ることができます。そして、LINQ列挙を追加してデータを集計し、出力として通常のデータテーブルやチャートを生成します。PivotPanelの詳細については、「クイックスタート：OLAPへのデータの追加」を参照してください。

Slicer

A slicer control can provide an easy way to filter a grid or a chart, making it an excellent visual tool for your dashboard. Slicers make it really convenient to filter data in a pivot table. Though the **OLAP** control is already incorporated with conventional filtering capacities, it does not provide any clue to decipher what exactly is being filtered in the current state. This is where the **Slicer** control becomes useful. The slicer offers a user-friendly interface to filter the PivotGrid or PivotChart data.

Apart from providing quick filtering, the slicer also indicates the current filtering state. This makes it easy to understand what exactly is shown in a filtered **PivotGrid** or **PivotChart**. The Slicer control for MVC displays all the unique values of the field which is being filtered in a list. You can then simply select the field values from that list to filter the **PivotGrid/PivotChart** data.

Copyright © GrapeCity, Inc. All rights reserved.



Slicer consists of elements as follows:

- **Header**: Header area of the FlexPivotSlicer contains a Caption that can be set using **Header** property. Along with Caption, header also displays Multi-select and Clear filter buttons on the right hand side. The Slicer control allows you to toggle the visibility of the header area using the **ShowHeader** property of the Slicer class. Purpose of the header buttons is elaborated below:
  - **Multi-Select button**: The Multi-Select button allows users to change the selection mode for the list at runtime. Once the Multi-Select button is enabled, multiple items can be selected from the list by normally clicking on the items or using **Shift+Click** or **Ctrl+Click**. The Multi-Select button shown in the header can be shown or hidden using the **MultiSelect** property of the Slicer class.
  - **Clear filter button**: The Clear filter button shown in the control's header allows you to clear the filtering applied to the PivotGrid/PivotChart using the Slicer control at runtime.
- **List**: Once the Slicer is bound to a field, which needs to be filtered, the list area of the control shows a list of all unique values from that field. PivotGrid or PivotChart data can be filtered based on the values selected in this list.

クイックスタート: OLAP へのデータの追加

このセクションでは、MVC Web アプリケーションに **OLAP** コントロールを追加し、そこにデータを追加する方法について説明します。

このトピックは3つの手順で構成されます。

- **手順 1:MVC アプリケーションの作成**
- **手順 2:OLAP のデータソースの作成**
- **手順 3:OLAP コントロールの追加**
- **手順 4:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後の OLAP を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



Country	Aoba	Olap	Wijmo	Xuni	総計
イタリア	6,870	238		9,672	16,780
インド	1,041	9,293		1,904	12,238
ギリシャ	7,444		3,034		10,478
ロシア	9,628				9,628
英国				2,833	2,833
総計	24,983	9,531	3,034	14,409	51,957

© 2016 - ComponentOne ASP.NET MVC Project

手順 1：MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーションの

Copyright © GrapeCity, Inc. All rights reserved.

作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順 2：OLAP のデータソースの作成

1. [**モデル**]フォルダに新しいクラスを追加します（例：ProductData.cs）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 次のコードを新しいモデルに追加して、OLAP コントロールのデータソースになるクラスを定義します。

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OlapQuickStart.Models
{
    public class ProductData
    {
        private static Random r = new Random();

        public int ID { get; set; }
        public string Product { get; set; }
        public string Country { get; set; }
        public DateTime Date { get; set; }
        public int Sales { get; set; }
        public int Downloads { get; set; }
        public bool Active { get; set; }
        public double Discount { get; set; }

        private static int randomInt(int max)
        {
            return (int)Math.Floor(r.NextDouble() * (max + 1));
        }
        public static DataTable GetDataTable(int cnt, string tableName)
        {
            var sufix = cnt / 10000;
            string[] countries =
            "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
            .Split(',');
            string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
            DataTable data = new DataTable(tableName);
            data.Columns.Add("ID", typeof(int));
            data.Columns.Add("Product", typeof(string));
            data.Columns.Add("Country", typeof(string));
            data.Columns.Add("Date", typeof(DateTime));
            data.Columns.Add("Sales", typeof(int));
            data.Columns.Add("Downloads", typeof(int));
            data.Columns.Add("Active", typeof(bool));
            data.Columns.Add("Discount", typeof(double));
            for (var i = 0; i < cnt; i++)
            {
                object[] values = new object[] {
                    i,
                    products[randomInt(products.Length - 1)],
                    countries[randomInt(countries.Length - 1)],
                    new DateTime(2015, randomInt(11) + 1, randomInt(27) + 1),
                    randomInt(10000),
                    randomInt(10000),
                    randomInt(1) == 1 ? true : false,
                    r.NextDouble()
                };
                data.Rows.Add(values);
            }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            return data;
        }

        public static IEnumerable<ProductData> GetData(int cnt)
        {
            string[] countries =
            "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
            .Split(',');
            string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
            List<ProductData> result = new List<ProductData>();
            for (var i = 0; i < cnt; i++)
            {
                result.Add(new ProductData
                {
                    ID = i,
                    Product = products[randomInt(products.Length - 1)],
                    Country = countries[randomInt(countries.Length - 1)],
                    Date = new DateTime(2015, randomInt(5) + 1, randomInt(27) + 1),
                    Sales = randomInt(10000),
                    Downloads = randomInt(10000),
                    Active = randomInt(1) == 1 ? true : false,
                    Discount = r.NextDouble()
                });
            }
            return result;
        }
    }
}
```

先頭に戻る

手順 3:OLAP コントロールの追加

OLAP コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを右クリックします。
2. コンテキストメニューから、[**追加** →]→[**コントローラー**]を選択します。
3. [**スキャフォールディングを追加**]ダイアログで、次の手順を実行します。
    1. [**MVC 5 コントローラー -空**]テンプレートを選択します。
    2. コントローラーの名前を設定します(例:`OLAPController`)。
    3. [**追加**]をクリックします。
4. メソッド Index() を次のメソッドに置き換えます。

```
 public class OlapController : Controller
 {
// GET: Olap
 private static System.Collections.IEnumerable Data =
 ProductData.GetData(10).ToList();
// GET: PivotGrid
 public ActionResult Index()
      {
          return View(Data);
      }
 }
```

ビューの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを展開し、コントローラー`OLAPController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、[**ビューの追加**]を選択します。[**ビューの追加**]ダイアログが表示されます。
4. [**ビューの追加**]ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor (CSHTML)**であることを確認します。

Copyright © GrapeCity, Inc. All rights reserved.

5. [**追加**]をクリックします。コントローラーにビューが追加されます。

```
@using OlapQuickStart.Models;

@model IEnumerable<ProductData>
<br />
@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))
@Html.C1().PivotPanel().ItemsSourceId("indexEngine")
@Html.C1().PivotChart().ItemsSourceId("indexEngine")
@Html.C1().PivotGrid().ItemsSourceId("indexEngine")
```

手順 4:プロジェクトのビルドおよび実行

1. [**ビルド**]→[**ソリューションのビルド**]をクリックして、プロジェクトをビルドします。
2. [**F5**]キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例:http://localhost:1234/**OLAP/Index**)、ビューを表示します。

先頭に戻る

OLAPの使用

This section comprises all the features and auxiliary functionalities offered by OLAP.

データ連結
: Learn about different data binding options that can be used to add data to the OLAP controls.

Pivotフィールドの定義
: Learn how to add Cube and Pivot fiels in the OAP control.

エクスポート
: Learn how to export OLAP data to excel file.

フィルタリング
: Learn about using value filter, range filter and silcer to filter data in OLAP control.

グループ化
: Learn grouping of data in the OLAP control.

ビューの保存とロード
: Learn how to save and load view of an OLAP control.

ソート
: Learn about sorting data in the OLAP control.

実行時にフィールドのプロパティを更新する
: Learn about accessing and updating field properties at runtime using the object model.

データ連結

OLAPは、ハイエンドなビジネスインテリジェンス機能を提供する強力なデータ分析ツールとして設計され、さまざまなデータ連結オプションからデータをOLAPコントロールに追加できます。データをOLAPコントロールに連結したら、フィルタ処理、グループ化、ソート、ドリルダウン、条件付き書式設定の適用、対話式レポートの作成など、多数の対話式機能を使用できます。

このセクションには、次の情報が含まれます。

モデル連結
: 基本のモデル連結を使用して、OLAPコントロールにデータを追加する方法を学習します。

リモートデータ連結
: C1JSONRequestを使用して、リモートでOLAPコントロールにデータを連結する方法を学習します。

Copyright © GrapeCity, Inc. All rights reserved.

キューブデータ連結
OLAPでキューブを使用する方法を学習します。
データエンジンサービス
データエンジンサービスの使用方法を学習します。

キューブデータ連結

The OLAP control allows you to directly bind cube data that is created using SSAS (SQL Server Analysis Services). The PivotEngine component of the OLAP control allows you to directly bind cube data with the help of **BindCubeService** method. This method accepts two parameters; URL (URL of the SSAS server) and Cube (OLAP cube in the SSAS server). The **PivotPanel** control and **PivotGrid** control binds to **PivotEngine**.

To accomplish this, follow these steps:

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Add the OLAP control

To add an OLAP control to the application, follow these steps:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template, and then click **Add**.
    2. Set name of the controller (for example: `CubeController`).
    3. Click **Add**.

CubeController.cs

```
public class CubeController : Controller
    {
        // GET: Cube
        public ActionResult Index()
        {
            return View();
        }
    }
```

Add a View for the Controller

In the view, we create an instance **PivotEngine** and bind it a cube data using the BindCubeService property.

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `CubeController`.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller. Copy the following code and paste it inside **Index.cshtml**.

Index.cshtml

```
@(Html.C1().PivotEngine().Id("cubeEngine")
  .BindCubeService("http://ssrs.componentone.com/OLAP/msmdpump.dll", "Adventure
Works")
  .RowFields(pfcb => pfcb.Items("[Customer].[Country]"))
```

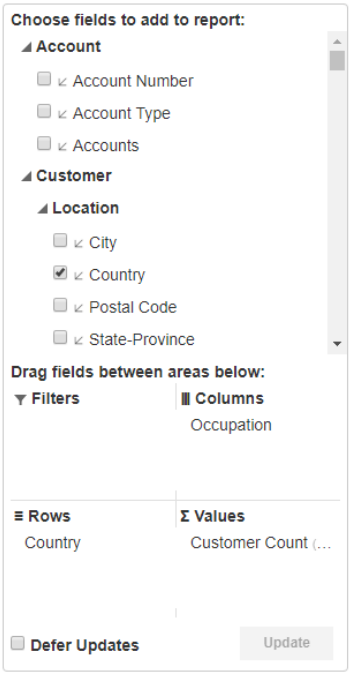

Copyright © GrapeCity, Inc. All rights reserved.

```
    .ColumnFields(cfcb => cfcb.Items("[Customer].[Occupation]"))
    .ValueFields(vfcb => vfcb.Items("[Measures].[Customer Count]")))

@(Html.C1().PivotPanel().ItemsSourceId("cubeEngine").Width(300))
@(Html.C1().PivotGrid().ItemsSourceId("cubeEngine").Width(800))
```

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.
   Append the folder name and view name to the generated URL (for example: http://localhost:1234/**Cube/Index**) in the address bar of the browser to see the view.



Country	Clerical	Management	Manual	Professional	Skilled Manual	Customer Count
Australia	436	771	248	1,436	700	3,591
Canada	166	245	25	548	587	1,571
France	558	95	788	235	134	1,810
Germany	566	129	673	259	153	1,780
United Kingdom	589	176	609	275	264	1,913
United States	613	1,659	41	2,767	2,739	7,819
Grand Total	2,928	3,075	2,384	5,520	4,577	18,484

Back to Top

DataEngineを使用したキューブデータ連結

このセクションでは、**SSAS**「**SQL Server Analysis Services**」を使用したOLAPコントロールのキューブデータの追加に必要な手順を説明します。以下の例では、**PivotEngine**コンポーネントがサービスにバインドされています。これでWeb APIの**DataEngine**がサポートされて、このデータエンジンがデータ集計に対応しています。**PivotPanel**コントロールと**PivotGrid**コントロールは、PivotEngineにバインドされています。PivotPanelコントロールでビューの定義を変更できます。集計されたデータはサービスから取得できます。PivotGridコントロールは、集計されたデータを表示します。

OLAPでリモートデータ連結とローカル連結を実行することもできます。データ連結の詳細については、「データ連結」を参照してください。

OLAPコントロールでキューブデータを使用するデータ連結を実装するには、次の手順を完了します。

- **手順 1：MVC アプリケーションの作成**
- **手順 2: DataEngine Web APIのインストール**
- **手順 3: Startup.csファイルの構成**
- **手順 4：OLAP コントロールの追加**
- **手順 5：プロジェクトのビルドおよび実行**

手順1：MVC アプリケーションの作成

Visual Studioテンプレートを使用してASP.NET MVCアプリケーションを作成し、WebAPI設定を有効にします。

Copyright © GrapeCity, Inc. All rights reserved.

1. **[ファイル]**→**[新規作成]**→**[プロジェクト]**を選択します。
2. インストール済みテンプレートから、**[Visual C#]**→**[Web]**→**[ASP.NETWeb アプリケーション (.NET Framework)]**を選択します。
3. **[新しい ASP.NET MVC プロジェクト(.NET Framework)]**ダイアログで、**[MVC]**を選択します。
4. **「フォルダーおよびコア参照を追加する」**セクションにWeb APIをチェックします。**[OK]**をクリックします。

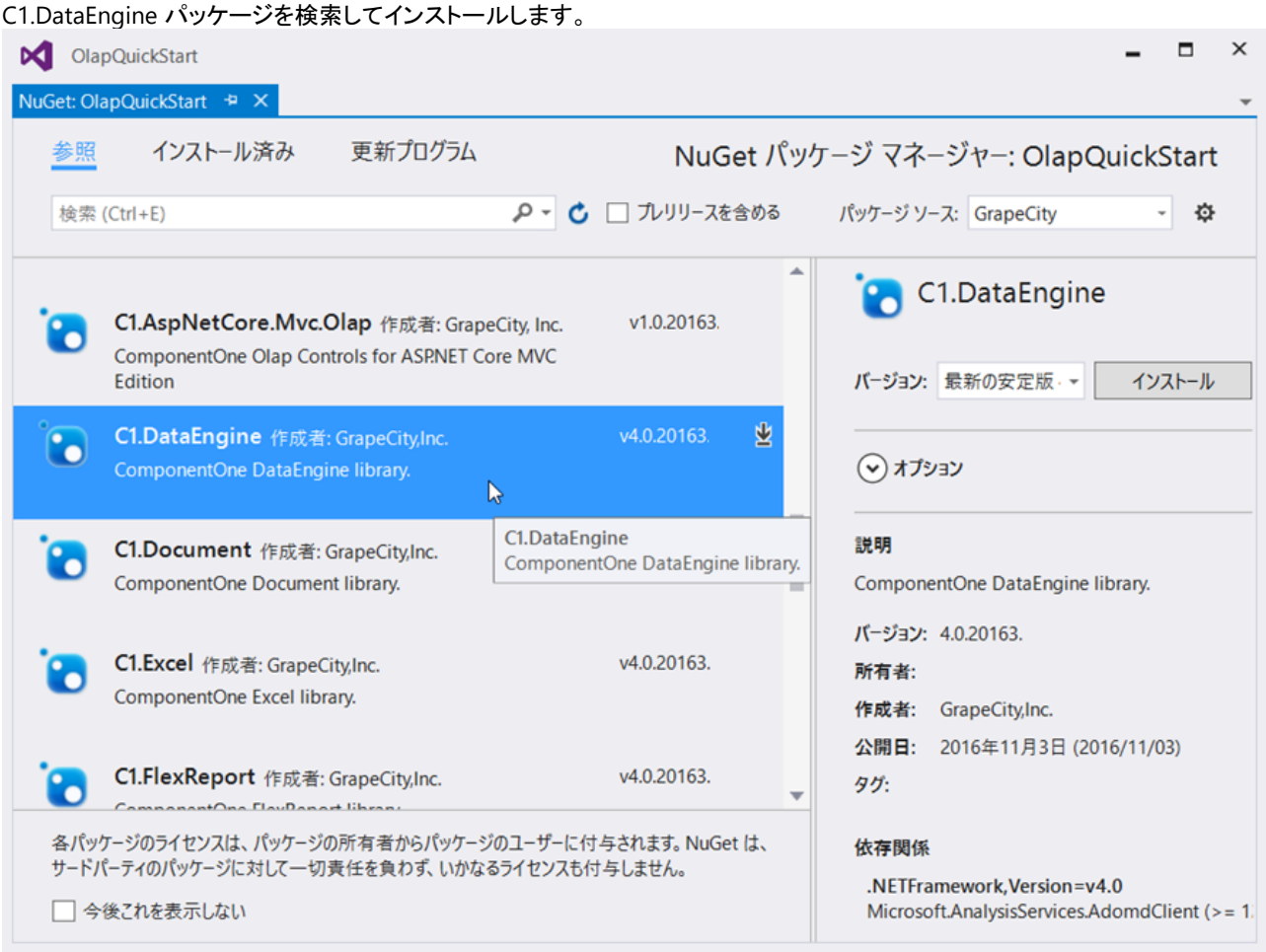
   ライセンス、リソースの登録、およびアセンブリの参照の詳細については、「Visual Studio テンプレートの使用」を参照してください。

先頭に戻る

手順 2: DataEngine Web APIのインストール

NuGetサーバーからDataEngine Web APIおよびC1.WebApiのパッケージをインストールします。

1. 「NuGetパッケージの管理」にて、パッケージソースとして**nuget.org** を選択します。
2. C1.DataEngine パッケージを検索してインストールします。



DataEngine Web API は、プロジェクトに次の参照を追加します。

- `C1.WebApi.dll`
- `C1.WebApi.DataEngine.dll`
- `C1.DataEngine.4.dll`
- `System.Net.Http.Formatting.dll`
- `System.Web.Http.dll`
- `System.Web.Http.Owin.dll`
- `System.Web.Http.WebHost.dll`

手順 3: Startup.csファイルの構成

Copyright © GrapeCity, Inc. All rights reserved.

必要な参照を追加した後、データエンジンサービスから集約されているデータを取得するためにStartup.csを構成する必要があります。

1. **ソリューションエクスプローラー**で、ターゲットプロジェクトを選択します。
2. **プロジェクト** メニューで、**新しい項目の追加** オプションをクリックします。
3. **新しい項目の追加** ダイアログに、Webを選択し、右のリストから**OWIN スタートアップ クラス** テンプレートを選択します。
4. Startup.csファイルで、**Startup**クラスに次のコードを追加します。

```
using C1.DataEngine;
using Microsoft.Owin;
using Owin;
using System.IO;
using System.Linq;
using System.Web.Http;
using OlapSSAS.Models;

[assembly: OwinStartupAttribute(typeof(OlapSSAS.Startup))]
namespace OlapSSAS
{
 public partial class Startup
   {
 private readonly HttpConfiguration config = GlobalConfiguration.Configuration;
 public void Configuration(IAppBuilder app)
   {
 app.UseDataEngineProviders()
 .AddCube("cube",
 @"Data Source=http://ssrs.componentone.com/OLAP/msmdpump.dll;
Provider=msolap;Initial Catalog=AdventureWorksDW2012Multidimensional",
 "Adventure Works");
        }
    }
}
```

先頭に戻る

手順 4:OLAP コントロールの追加

OLAP コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを右クリックします。
2. コンテキストメニューから、[**追加** →]→[**コントローラー**]を選択します。
3. [**スキャフォールディングを追加**]ダイアログで、次の手順を実行します。
    1. [**MVC 5 コントローラー -空**]テンプレートを選択します。
    2. コントローラーの名前を設定します(例:`OLAPController`)。
    3. [**追加**]をクリックします。
4. メソッド Index() を次のメソッドに置き換えます。

C#

```
// GET: SSAS
public ActionResult Index()
{
    return View();
}
```

ビューの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを展開し、コントローラー`OLAPController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、[**ビューの追加**]を選択します。[**ビューの追加**]ダイアログが表示されます。
4. [**ビューの追加**]ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor (CSHTML)**であることを確認します。

Copyright © GrapeCity, Inc. All rights reserved.

5. [**追加**]をクリックします。コントローラーにビューが追加されます。

```
@using C1.Web.Mvc.Olap;
@using C1.Web.Mvc.Grid
@Html.C1().Styles()
@Html.C1().Scripts().Basic().Olap()

@(Html.C1().PivotEngine().Id("ssasEngine")
            .BindService("~/api/dataengine/cube")
            .RowFields(pfcb => pfcb.Items("[Customer].[Country]"))
            .ColumnFields(cfcb => cfcb.Items("[Customer].[Occupation]"))
            .ValueFields(vfcb => vfcb.Items("[Measures].[Customer Count]")))
```

手順 5:プロジェクトのビルドおよび実行

1. [**ビルド**]→[**ソリューションのビルド**]をクリックして、プロジェクトをビルドします。
2. [**F5**]キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して(例:http://localhost:1234/**OLAP/Index**)、ビューを表示します。

次の図は、上記の手順を実行した後の OLAP を示しています。



Product	総計
AWC Logo Cap	2,190
All-Purpose Bike Stand	249
Bike Wash - Dissolver	908
Classic Vest, L	195
Classic Vest, M	199
Classic Vest, S	168
Fender Set - Mountain	2,121
HL Mountain Tire	1,396
HL Road Tire	858
Half-Finger Gloves, L	443
Half-Finger Gloves, M	499
Half-Finger Gloves, S	488
Hitch Rack - 4-Bike	328
Hydration Pack - 70 oz.	733
LL Mountain Tire	862
LL Road Tire	1,044
Long-Sleeve Logo Jersey, L	452
Long-Sleeve Logo Jersey, M	442

先頭に戻る

データエンジンサービス

このセクションでは、**OLAP**のデータ集約にデータエンジンサービスを使用するために必要な手順を説明します。以下の例では、PivotEngine コンポーネントがデータエンジンサービスを使用してデータエンジンのデータに接続されています。**PivotPanel**コントロールと**PivotGrid** コントロールは、**PivotEngine** にバインドされています。PivotPanel コントロールでビューの定義を変更できます。集計されたデータはサービスから取得できます。以下の例では、PivotGrid コントロールは集計されたデータを表示します。**PivotGrid** コントロールのセルをダブルクリックすると、グリッドに表示された詳細生データを見つけることができます。

OLAPでリモートデータ連結とモデル連結を実行することもできます。データ連結の詳細については、「データ連結」を参照してください。

OLAPコントロールでデータエンジンサービスを使用するデータ連結を実装するには、次の手順を完了します。

Copyright © GrapeCity, Inc. All rights reserved.

- **手順 1：MVC アプリケーションの作成**
- **手順 2: DataEngine Web APIのインストール**
- **手順 3：OLAPのデータソースの作成**
- **手順 4: Startup.csファイルの構成**
- **手順 5：OLAP コントロールの追加**
- **手順 6：プロジェクトのビルドおよび実行**

手順 1：MVC アプリケーションの作成

1. **[ファイル]**→**[新規作成]**→**[プロジェクト]**を選択します。
2. インストール済みテンプレートから、**[Visual C#]**→**[Web]**→**[ASP.NETWeb アプリケーション (.NET Framework)]**を選択します。
3. **[新しい ASP.NET MVC プロジェクト(.NET Framework)]**ダイアログで、**[MVC]**を選択します。
4. **「フォルダーおよびコア参照を追加する」**セクションにWeb APIをチェックします。**[OK]**をクリックします。

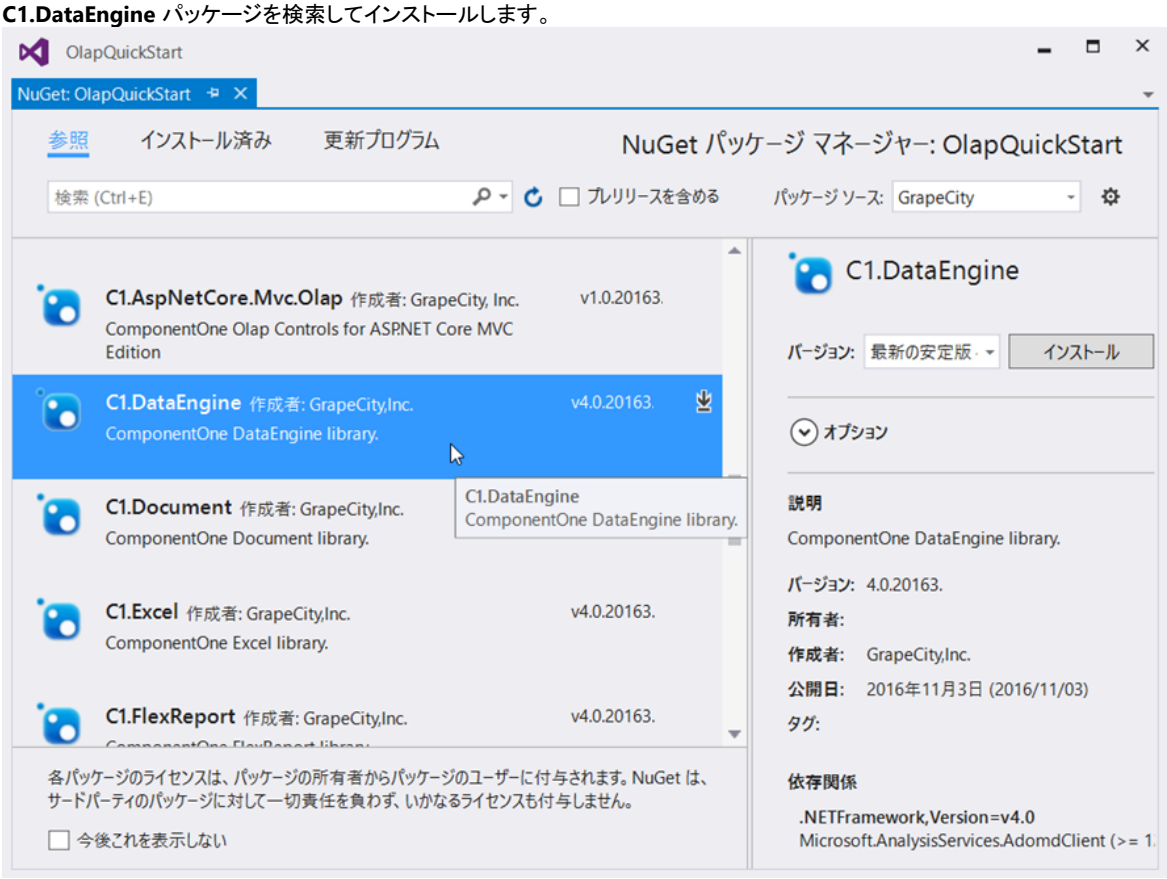
   ライセンス、リソースの登録、およびアセンブリの参照の詳細については、「Visual Studio テンプレートの使用」を参照してください。

先頭に戻る

手順 2: DataEngine Web APIのインストール

NuGetサーバーからDataEngine Web APIおよびC1.WebApiのパッケージをインストールします。

1. 「NuGetパッケージの管理」にて、パッケージソースとして**nuget.org** を選択します。
2. **C1.DataEngine** パッケージを検索してインストールします。



DataEngine Web API は、プロジェクトに次の参照を追加します。

- `C1.WebApi.dll`
- `C1.WebApi.DataEngine.dll`
- `C1.DataEngine.4.dll`
- `System.Net.Http.Formatting.dll`
- `System.Web.Http.dll`
- `System.Web.Http.Owin.dll`
- `System.Web.Http.WebHost.dll`

手順 3：OLAPのデータソースの作成

Copyright © GrapeCity, Inc. All rights reserved.

OLAPコントロールのデータソースを作成するために、[モデル]フォルダに新しいクラスを作成します。

1. **[モデル]**フォルダに新しいクラスを追加します(例:ProductData.cs)。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. モデルに次のコードを追加して、クラスを定義します。

C#

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OlapQuickStart.Models
{
    public class ProductData
    {
        private static Random r = new Random();

        public int ID { get; set; }
        public string Product { get; set; }
        public string Country { get; set; }
        public DateTime Date { get; set; }
        public int Sales { get; set; }
        public int Downloads { get; set; }
        public bool Active { get; set; }
        public double Discount { get; set; }

        private static int randomInt(int max)
        {
            return (int)Math.Floor(r.NextDouble() * (max + 1));
        }
        public static DataTable GetDataTable(int cnt, string tableName)
        {
            var sufix = cnt / 10000;
            string[] countries =
            "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
            .Split(',');
            string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
            DataTable data = new DataTable(tableName);
            data.Columns.Add("ID", typeof(int));
            data.Columns.Add("Product", typeof(string));
            data.Columns.Add("Country", typeof(string));
            data.Columns.Add("Date", typeof(DateTime));
            data.Columns.Add("Sales", typeof(int));
            data.Columns.Add("Downloads", typeof(int));
            data.Columns.Add("Active", typeof(bool));
            data.Columns.Add("Discount", typeof(double));
            for (var i = 0; i < cnt; i++)
            {
                object[] values = new object[] {
                    i,
                    products[randomInt(products.Length - 1)],
                    countries[randomInt(countries.Length - 1)],
                    new DateTime(2015, randomInt(11) + 1, randomInt(27) + 1),
                    randomInt(10000),
                    randomInt(10000),
                    randomInt(1) == 1 ? true : false,
                    r.NextDouble()
                };
                data.Rows.Add(values);
            }
            return data;
        }

        public static IEnumerable<ProductData> GetData(int cnt)
        {
            string[] countries =
            "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
            .Split(',');
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
            List<ProductData> result = new List<ProductData>();
            for (var i = 0; i < cnt; i++)
            {
                result.Add(new ProductData
                {
                    ID = i,
                    Product = products[randomInt(products.Length - 1)],
                    Country = countries[randomInt(countries.Length - 1)],
                    Date = new DateTime(2015, randomInt(5) + 1, randomInt(27) + 1),
                    Sales = randomInt(10000),
                    Downloads = randomInt(10000),
                    Active = randomInt(1) == 1 ? true : false,
                    Discount = r.NextDouble()
                });
            }
            return result;
        }
    }
}
```

先頭に戻る

手順 4: Startup.csファイルの構成

必要な参照を追加した後、データエンジンサービスから集約されているデータを取得するためにStartup.csを構成する必要があります。

1. **ソリューションエクスプローラー**で、ターゲットプロジェクトを選択します。
2. **プロジェクト** メニューで、**新しい項目の追加** オプションをクリックします。
3. **新しい項目の追加** ダイアログに、Webを選択し、右のリストから**OWIN スタートアップ クラス** テンプレートを選択します。
4. Startup.csファイルで、**Startup**クラスに次のコードを追加します。

```
using C1.DataEngine;
using Microsoft.Owin;
using Owin;
using System.IO;
using System.Linq;
using System.Web.Http;
using OlapSSAS.Models;

[assembly: OwinStartupAttribute(typeof(OlapSSAS.Startup))]
namespace OlapSSAS
{
public partial class Startup
    {
 private readonly HttpConfiguration config = GlobalConfiguration.Configuration;
 private static string DATAPATH = Path.Combine(System.Web.HttpRuntime.AppDomainAppPath, "Data");
 public void Configuration(IAppBuilder app)
     {
     app.UseDataEngineProviders()
     .AddDataEngine("complex10", () =>
     {
         return ProductData.GetData(100000);
     })
        }
    }
}
```

先頭に戻る

手順 5:OLAP コントロールの追加

OLAP コントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを右クリックします。
2. コンテキストメニューから、[**追加** →]→[**コントローラー**]を選択します。
3. [**スキャフォールディングを追加**]ダイアログで、次の手順を実行します。
    1. [**MVC 5 コントローラー -空**]テンプレートを選択します。
    2. コントローラーの名前を設定します(例:`OLAPController`)。
    3. [**追加**]をクリックします。
4. メソッド Index() を次のメソッドに置き換えます。

Copyright © GrapeCity, Inc. All rights reserved.

C#

```
// GET: SSAS
public ActionResult Index()
{
    return View();
}
```

ビューの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを展開し、コントローラー`OLAPController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、[**ビューの追加**]を選択します。[**ビューの追加**]ダイアログが表示されます。
4. [**ビューの追加**]ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor (CSHTML)**であることを確認します。
5. [**追加**]をクリックします。コントローラーにビューが追加されます。

```
@using C1.Web.Mvc
@Html.C1().Scripts().Basic().Olap()

@(Html.C1().PivotEngine().Id("dataEngine")
        .BindService("~/api/dataengine/complex10")
        .RowFields(pfcb => pfcb.Items("Country"))
        .ColumnFields(cfcb => cfcb.Items("Product"))
        .ValueFields(vfcb => vfcb.Items("Sales")))
@Html.C1().PivotPanel().ItemsSourceId("dataSourceEngine")
@Html.C1().PivotChart().ItemsSourceId("dataSourceEngine")
@Html.C1().PivotGrid().ItemsSourceId("dataSourceEngine")
```

手順 6:プロジェクトのビルドおよび実行

1. [**ビルド**]→[**ソリューションのビルド**]をクリックして、プロジェクトをビルドします。
2. [**F5**]キーを押してプロジェクトを実行します。

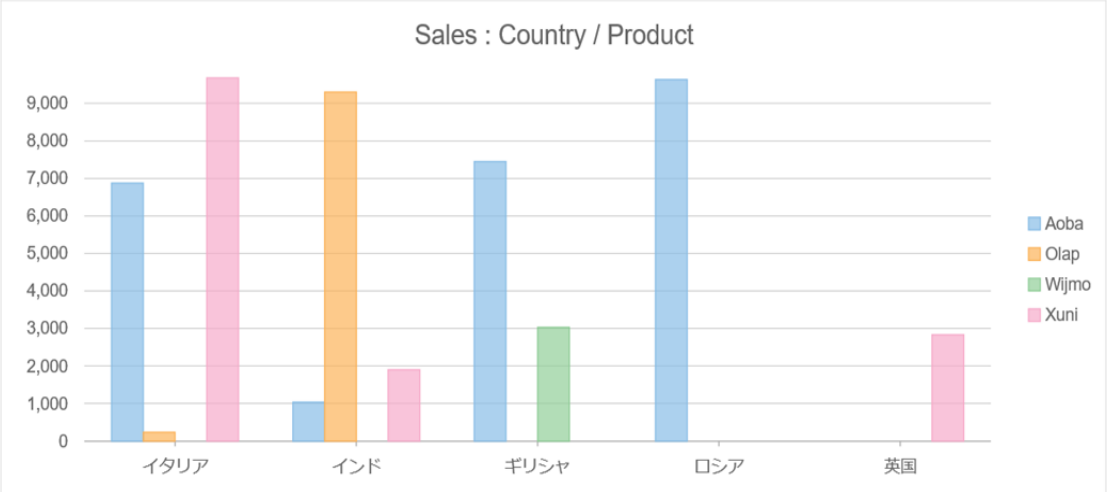
> ブラウザのアドレスバーで、生成された URL にフォルダ名とビュー名を追加して（例：http://localhost:1234/**OLAP/Index**）、ビューを表示します。

次の図は、上記の手順を実行した後の OLAP を示しています。

Copyright © GrapeCity, Inc. All rights reserved.



Country	Aoba	Olap	Wijmo	Xuni	総計
イタリア	6,870	238		9,672	16,780
インド	1,041	9,293		1,904	12,238
ギリシャ	7,444		3,034		10,478
ロシア	9,628				9,628
英国				2,833	2,833
総計	24,983	9,531	3,034	14,409	51,957



先頭に戻る

モデル連結

このセクションでは、モデル連結を使用したOLAPコントロールのデータの追加に必要な手順を説明します。OLAPでリモートデータ連結を実行することもできます。リモートデータ連結の詳細については、「リモートデータ連結」を参照してください。

OLAPコントロールでモデルデータ連結を実装するには、次の手順を完了します。

- **手順1：OLAPのデータソースの作成**
- **手順2：OLAPコントロールの追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後にブラウザに表示されるOLAPコントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



Country	Aoba	Olap	Wijmo	Xuni	総計
イタリア	6,870	238		9,672	**16,780**
インド	1,041	9,293		1,904	**12,238**
ギリシャ	7,444		3,034		**10,478**
ロシア	9,628				**9,628**
英国				2,833	**2,833**
総計	**24,983**	**9,531**	**3,034**	**14,409**	**51,957**

© 2016 - ComponentOne ASP.NET MVC Project

手順1：OLAPのデータソースの作成

OLAPコントロールのデータソースを作成するために、[モデル]フォルダに新しいクラスを作成します。

Copyright © GrapeCity, Inc. All rights reserved.

1. **[モデル]**フォルダに新しいクラスを追加します(例:ProductData.cs)。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. モデルに次のコードを追加して、OLAPのデータを定義します。

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OlapQuickStart.Models
{
    public class ProductData
    {
        private static Random r = new Random();

        public int ID { get; set; }
        public string Product { get; set; }
        public string Country { get; set; }
        public DateTime Date { get; set; }
        public int Sales { get; set; }
        public int Downloads { get; set; }
        public bool Active { get; set; }
        public double Discount { get; set; }

        private static int randomInt(int max)
        {
            return (int)Math.Floor(r.NextDouble() * (max + 1));
        }
        public static DataTable GetDataTable(int cnt, string tableName)
        {
            var sufix = cnt / 10000;
            string[] countries =
            "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
            .Split(',');
            string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
            DataTable data = new DataTable(tableName);
            data.Columns.Add("ID", typeof(int));
            data.Columns.Add("Product", typeof(string));
            data.Columns.Add("Country", typeof(string));
            data.Columns.Add("Date", typeof(DateTime));
            data.Columns.Add("Sales", typeof(int));
            data.Columns.Add("Downloads", typeof(int));
            data.Columns.Add("Active", typeof(bool));
            data.Columns.Add("Discount", typeof(double));
            for (var i = 0; i < cnt; i++)
            {
                object[] values = new object[] {
                    i,
                    products[randomInt(products.Length - 1)],
                    countries[randomInt(countries.Length - 1)],
                    new DateTime(2015, randomInt(11) + 1, randomInt(27) + 1),
                    randomInt(10000),
                    randomInt(10000),
                    randomInt(1) == 1 ? true : false,
                    r.NextDouble()
                };
                data.Rows.Add(values);
            }
            return data;
        }

        public static IEnumerable<ProductData> GetData(int cnt)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            {
                string[] countries =
                "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
                .Split(',');
                string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
                List<ProductData> result = new List<ProductData>();
                for (var i = 0; i < cnt; i++)
                {
                    result.Add(new ProductData
                    {
                        ID = i,
                        Product = products[randomInt(products.Length - 1)],
                        Country = countries[randomInt(countries.Length - 1)],
                        Date = new DateTime(2015, randomInt(5) + 1, randomInt(27) + 1),
                        Sales = randomInt(10000),
                        Downloads = randomInt(10000),
                        Active = randomInt(1) == 1 ? true : false,
                        Discount = r.NextDouble()
                    });
                }
                return result;
            }
        }
    }
```

先頭に戻る

手順2:OLAPコントロールの追加

OLAPコントロールのコントローラーとビューを作成し、次の手順を実行して、OLAPコントロールを初期化します。

OLAPコントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順を実行します。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します(例:`OLAPController`)。
    3. **[追加]**をクリックします。
4. メソッドIndex()を次のメソッドに置き換えます。

```
    public class OlapController : Controller
    {
// GET: Olap
 private static System.Collections.IEnumerable Data =
 ProductData.GetData(10).ToList();
// GET: PivotGrid
 public ActionResult Index()
      {
          return View(Data);
      }
  }
```

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`OLAPController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

Copyright © GrapeCity, Inc. All rights reserved.

```
@using OlapQuickStart.Models;

@model IEnumerable<ProductData>
<br />
@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))
@Html.C1().PivotPanel().ItemsSourceId("indexEngine")
@Html.C1().PivotChart().ItemsSourceId("indexEngine")
@Html.C1().PivotGrid().ItemsSourceId("indexEngine")
```

手順3：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**OLAP/Index**）。

先頭に戻る

リモートデータ連結

OLAPでは、**C1JSONRequest**を使用してデータを直接取得できます。これは、サーバー、テーブル、および列を含むリモートデータのURLを指定します。返される配列は、**CollectionView**オブジェクトのデータソースとして使用されます。

`CollectionViewHelper`は、コレクションに編集、フィルタ処理、グループ化、ソートなどのサービスを提供する静的クラスです。**PivotEngine**の`Bind`プロパティは、特定の操作を実行するためのアクションURLメソッドを渡すことで、コレクションに連結するために使用されます。このクラスには次のメソッドも含まれます。

- `Read()`：コレクションからデータを取得します。
- `Edit()`：OLAPでExcel形式の編集を可能にします。
- `BatchEdit()`：一度に複数の項目を編集できます。

このトピックは3つの手順で構成されます。

- **手順1：OLAPのデータソースの作成**
- **手順2：OLAPコントロールの追加**
- **手順3：プロジェクトのビルドおよび実行**

手順1：OLAPのデータソースの作成

OLAPコントロールのデータソースを作成するために、[モデル]フォルダに新しいクラスを作成します。

1. **[モデル]**フォルダに新しいクラスを追加します（例：`ProductData.cs`）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. モデルに次のコードを追加して、OLAPのデータを定義します。

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace OlapQuickStart.Models
{
    public class ProductData
    {
        private static Random r = new Random();

        public int ID { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
public string Product { get; set; }
public string Country { get; set; }
public DateTime Date { get; set; }
public int Sales { get; set; }
public int Downloads { get; set; }
public bool Active { get; set; }
public double Discount { get; set; }

private static int randomInt(int max)
{
    return (int)Math.Floor(r.NextDouble() * (max + 1));
}
public static DataTable GetDataTable(int cnt, string tableName)
{
    var sufix = cnt / 10000;
    string[] countries =
    "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
    .Split(',');
    string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
    DataTable data = new DataTable(tableName);
    data.Columns.Add("ID", typeof(int));
    data.Columns.Add("Product", typeof(string));
    data.Columns.Add("Country", typeof(string));
    data.Columns.Add("Date", typeof(DateTime));
    data.Columns.Add("Sales", typeof(int));
    data.Columns.Add("Downloads", typeof(int));
    data.Columns.Add("Active", typeof(bool));
    data.Columns.Add("Discount", typeof(double));
    for (var i = 0; i < cnt; i++)
    {
        object[] values = new object[] {
            i,
            products[randomInt(products.Length - 1)],
            countries[randomInt(countries.Length - 1)],
            new DateTime(2015, randomInt(11) + 1, randomInt(27) + 1),
            randomInt(10000),
            randomInt(10000),
            randomInt(1) == 1 ? true : false,
            r.NextDouble()
        };
        data.Rows.Add(values);
    }
    return data;
}

public static IEnumerable<ProductData> GetData(int cnt)
{
    string[] countries =
    "中国,インド,ロシア,米国,ドイツ,英国,日本,イタリア,ギリシャ,スペイン,ポルトガル"
    .Split(',');
    string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
    List<ProductData> result = new List<ProductData>();
    for (var i = 0; i < cnt; i++)
    {
        result.Add(new ProductData
        {
            ID = i,
            Product = products[randomInt(products.Length - 1)],
            Country = countries[randomInt(countries.Length - 1)],
            Date = new DateTime(2015, randomInt(5) + 1, randomInt(27) + 1),
            Sales = randomInt(10000),
            Downloads = randomInt(10000),
            Active = randomInt(1) == 1 ? true : false,
            Discount = r.NextDouble()
        });
    }
    return result;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
    }
}
```

先頭に戻る

手順2：OLAPコントロールの追加

OLAPコントロールのコントローラーとビューを作成し、次の手順を実行して、OLAPコントロールを初期化します。

OLAPコントロールを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順を実行します。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：RemoteBind`Controller`）。
    3. **[追加]**をクリックします。
4. メソッドIndex()を次のメソッドに置き換えます。

```
partial class RemoteBindController : Controller
{
    private static IEnumerable<ProductData> RemoteData =
     ProductData.GetData(100000).ToList();
    // GET: PivotGrid
    public ActionResult RemoteBind()
    {
        return View();
    }

    public ActionResult RemoteBind_Read([C1JsonRequest]
        CollectionViewRequest<ProductData> requestData)
    {
        return this.C1Json(CollectionViewHelper.Read(requestData, RemoteData));
    }
}
```

コントローラーのビューの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`OLAPController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

```
@using RemoteDataOlap.Models;
@model IEnumerable<ProductData>

@(Html.C1().PivotEngine().Id("remoteEngine").Bind(Url.Action("RemoteBind_Read"))
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))
@Html.C1().PivotPanel().ItemsSourceId("remoteEngine")
@Html.C1().PivotGrid().ItemsSourceId("remoteEngine")
```

手順3：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http://localhost:1234/**OLAP/Index**）。

次の図は、上記の手順を実行した後にブラウザに表示されるOLAPコントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



Country	Aoba	Olap	Wijmo	Xuni	総計
イタリア	6,870	238		9,672	16,780
インド	1,041	9,293		1,904	12,238
ギリシャ	7,444		3,034		10,478
ロシア	9,628				9,628
英国				2,833	2,833
総計	24,983	9,531	3,034	14,409	51,957

先頭に戻る

Pivotフィールドの定義

OLAPアプリケーションの主な長所の1つは、対話的操作です。ユーザーは、ビューを簡単に作成および変更して、その出力を直ちにWebブラウザで確認できる必要があります。OLAPでは、Excel形式のユーザーインタフェースとユーザーフレンドリなダイアログを使用して、これが可能になります。ただし、コードからビューを構成したりピボットフィールドを追加したい場合もあります。これは、**PivotField**クラスと**PivotEngine**クラスを使用して実現することができます。

> **メモ**:フィールド名に複数の大文字が含まれる(例:ProductName)場合、大文字が出現するたびにフィールド名の間に1つのスペースが追加され、例外が生成されます。これは、ピボットフィールドのheaderプロパティをフィールド名と同じ値に設定することで解決することができます。

このトピックは、OLAPコントロールでコードを使用してピボットフィールドを定義する方法の理解に役立ちます。次の図は、ピボットフィールドを定義した後にOLAPコントロールがどのように表示されるかを示します。

Country	Aoba	Olap	Wijmo	Grand Total
China		8,814.00		8,814.00
Germany			6,802.00	6,802.00
India			9,631.00	9,631.00
Japan		9,823.00	3,614.00	6,718.50
Portugal	4,067.00		8,046.00	6,056.50
Spain	6,503.00			6,503.00
UK			1,861.00	1,861.00
US	4,911.00			4,911.00
Grand Total	5,160.33	9,318.50	5,990.80	6,407.20

フィールドを自動生成しない場合は、次のコード例を使用して、OLAPコントロールにピボットフィールドを定義します。この例では、「クイックスタート:OLAPへのデータの追加」トピックで追加された**ProductData.cs**モデルを使用します。

Copyright © GrapeCity, Inc. All rights reserved.

Razor

```
@using <ApplicationName>.Models;
@model IEnumerable<ProductData>

@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
    .Fields(fld =>
    {
        fld.Items(it => it.AddPivotField(pf => pf.Binding("Country"))
        .AddPivotField(pf => pf.Binding("Product"))
        .AddPivotField(pf =>
pf.Binding("Sales").Aggregate(C1.Web.Mvc.Grid.Aggregate.Avg)));
    })
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))

@(Html.C1().PivotGrid().ItemsSourceId("indexEngine"))
```

エクスポート

PivotGridコントロールはFlexGridコントロールの拡張です。したがって、FlexGridに付属する拡張モジュールでサポートされている形式であれば、どの形式にもエクスポートできます。サポートされている形式には、XLSLX、CSV、PDFなどがあります。以下の例では、現在のビューとそれを転置したビューの2つのシートを含むExcelファイルをエクスポートします。

以下の画像は、以下のコードを実装した後にOLAPコントロールがどのように表示されるかを示しています。



Copyright © GrapeCity, Inc. All rights reserved.

次のコード例は、OLAPデータをExcelファイルにエクスポートする方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コード内（Index.cshtml）

Razor

```
@using OLAPSample.Models;
@model IEnumerable<ProductData>
<br />
@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))
@Html.C1().PivotPanel().ItemsSourceId("indexEngine")
@Html.C1().PivotChart().ItemsSourceId("indexEngine")
@Html.C1().PivotGrid().Id("indexGrid").ItemsSourceId("indexEngine")

<button type="button" class="btn btn-default" onclick="saveView()">ビューを保存</button>
<br />
<button type="button" class="btn btn-default" onclick="loadView()">ビューをロード
</button><br />
<button type="button" class="btn btn-default" onclick="excelExport()">XLSXにエクスポート
</button>

@section Scripts{
    <script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
    <script type="text/javascript">
        function saveView() {
            var ng = c1.getService('indexEngine');
            if (ng && ng.isViewDefined) {
                localStorage.viewDefinition = ng.viewDefinition;
            }
        }
        function loadView() {
            var ng = c1.getService('indexEngine');
            if (ng && localStorage.viewDefinition) {
                ng.viewDefinition = localStorage.viewDefinition;
                var cmbRowTotals = wijmo.Control.getControl('#RowTotals');
                if (cmbRowTotals) {
                    cmbRowTotals.selectedValue = ng.showRowTotals;
                }

                var cmbColTotals = wijmo.Control.getControl('#ColTotals');
                if (cmbColTotals) {
                    cmbColTotals.selectedValue = ng.showColumnTotals;
                }

                var chkShowZeros = document.getElementById('ColTotals');
                if (chkShowZeros) {
                    chkShowZeros.checked = ng.showZeros;
                }
            }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
function excelExport() {
    var pivotGrid = wijmo.Control.getControl('#indexGrid');
    // 現在のビューを含むブックを作成します
    var book = wijmo.grid.xlsx.FlexGridXlsxConverter.save(pivotGrid, {
        includeColumnHeaders: true,
        includeRowHeaders: true
    });
    book.sheets[0].name = 'Main View';
    addTitleCell(book.sheets[0], getViewTitle(pivotGrid.engine));

    // 転置したビューを含むシートを追加します
    transposeView(pivotGrid.engine);
    var transposed = wijmo.grid.xlsx.FlexGridXlsxConverter.save(pivotGrid, {
        includeColumnHeaders: true,
        includeRowHeaders: true
    });
    transposed.sheets[0].name = 'Transposed View';
    addTitleCell(transposed.sheets[0], getViewTitle(pivotGrid.engine));
    book.sheets.push(transposed.sheets[0]);
    transposeView(pivotGrid.engine);

    // ブックを保存します
    book.save('wijmo.OLAP.xlsx');
}
// 現在のビューのタイトルを構築します
function getViewTitle(ng) {
    var title = '';
    for (var i = 0; i < ng.valueFields.length; i++) {
        if (i > 0) title += ', ';
        title += ng.valueFields[i].header;
    }
    title += ' by ';
    if (ng.rowFields.length) {
        for (var i = 0; i < ng.rowFields.length; i++) {
            if (i > 0) title += ', ';
            title += ng.rowFields[i].header;
        }
    }
    if (ng.rowFields.length && ng.columnFields.length) {
        title += ' and by ';
    }
    if (ng.columnFields.length) {
        for (var i = 0; i < ng.columnFields.length; i++) {
            if (i > 0) title += ', ';
            title += ng.columnFields[i].header;
        }
    }
    return title;
}
function transposeView(ng) {
    ng.deferUpdate(function () {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                // 行/列フィールドを保存します
                var rows = [],
                    cols = [];
                for (var r = 0; r < ng.rowFields.length; r++) {
                    rows.push(ng.rowFields[r].header);
                }
                for (var c = 0; c < ng.columnFields.length; c++) {
                    cols.push(ng.columnFields[c].header);
                }
                // 行/列フィールドをクリアします
                ng.rowFields.clear();
                ng.columnFields.clear();

                // 転置した順序で行/列フィールドを復元します
                for (var r = 0; r < rows.length; r++) {
                    ng.columnFields.push(rows[r]);
                }
                for (var c = 0; c < cols.length; c++) {
                    ng.rowFields.push(cols[c]);
                }
            });
        }
        //xlxsシートにタイトルセルを追加します
        function addTitleCell(sheet, title) {

            // セルを作成します
            var cell = new wijmo.xlsx.WorkbookCell();
            cell.value = title;
            cell.style = new wijmo.xlsx.WorkbookStyle();
            cell.style.font = new wijmo.xlsx.WorkbookFont();
            cell.style.font.bold = true;

            // セルを保持する行を作成します
            var row = new wijmo.xlsx.WorkbookRow();
            row.cells[0] = cell;

            // 新しい行をシートに追加します
            sheet.rows.splice(0, 0, row);
        }
    </script>
}
```

フィルタリング

The OLAP control allows you to filter data using the following ways:

フィルタ処理
: Learn about filtering data using Value and Range filter in the OLAP control.

Slicerを使用してフィルタ処理
: Learn about filtering data using Slicer.

Copyright © GrapeCity, Inc. All rights reserved.

フィルタ処理

PivotPanelコントロールでは、実行時にパネルの**[次のボックス間でフィールドをドラッグ:]**セクションで、フィールドのデータをフィルタ処理できます。各フィールドには2つのフィルタがあります。リスト内の特定の値をチェックできる**値フィルタ**と、1つまたは2つの基準を指定できる**範囲フィルタ**です。この2つのフィルタは独立しており、この2つのフィルタを通過した値だけがOLAP PivotGridに表示されます。

値フィルタの使用

1. **[フィルタ]**、**[列フィールド]**、**[行フィールド]**、または**[値]**領域で、1つのフィールドを右クリックします。
2. コンテキストメニューの**[フィールドの設定]**をクリックします。**[フィールドの設定]**ダイアログボックスが開きます。
3. **[フィルタ]**オプションの隣にある**[編集]**ボタンをクリックし、**[値フィルタ]**タブを選択します。OLAPテーブルに表示しないフィールドについては、選択をクリアできます。



4. テーブルに表示するフィールドを選択したら、**[OK]**をクリックして、PivotGridとPivotChartのデータをフィルタ処理します。

範囲フィルタの使用

1. **[フィルタ]**、**[列フィールド]**、**[行フィールド]**、または**[値]**領域で、1つのフィールドを右クリックします。
2. コンテキストメニューの**[フィールドの設定]**をクリックします。**[フィールドの設定]**ダイアログボックスが開きます。
3. **[フィルタ]**オプションの隣にある**[編集]**ボタンをクリックし、**[条件フィルタ]**タブを選択します。OLAPテーブルに表示しないフィールドについては、選択をクリアできます。
4. ドロップダウンリストボックスから、次の項目のいずれかを選択します。

指定の値に等しい	このフィールドで指定された値と等しい項目を表示するフィルタを作成します。
指定の値に等しくない	このフィールドで指定された値と等しくない項目を表示するフィルタを作成します。
指定の値より大きい	指定された値で始まる項目が表示されるフィルタを作成します。

Copyright © GrapeCity, Inc. All rights reserved.

指定の値以上	指定された値で終わる項目が表示されるフィルタを作成します。
指定の値より小さい	指定された値を含む項目が表示されるフィルタを作成します。
指定の値以下	指定された値を含まない項目が表示されるフィルタを作成します。

5. 最初の空白のテキストボックスに、フィルタ処理する項目を追加します。



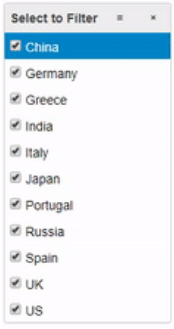
6. **[AND]**または**[OR]**を選択します。
7. 必要に応じて、2つめのフィルタ条件を追加します。**[(設定しない)]**以外のオプションを選択すると、2つめのテキストボックスがアクティブになり、項目を入力できます。
8. **[OK]**をクリックして**[カスタムフィルタ]**ダイアログボックスを閉じ、もう一度**[OK]**をクリックして**[フィールドの設定]**ダイアログボックスを閉じます。

Slicerを使用してフィルタ処理

OLAP Slicer allows users to filter data based on values and indicates the current filtering state, which makes it easy to understand what's shown in filtered PivotGrid and PivotChart controls.

The Slicer control provides a quick way to edit filters applied to PivotField objects. It provides buttons the user can click to filter data based on values and indicates the current filtering state, which makes it easy to understand what is shown in filtered PivotGrid and PivotChart controls.

Complete the following steps to filter data in the OLAP control using the slicer.



Country	Aoba	Olap	Wijmo	Xuni	Sales
China	112,728	78,885	127,450	141,238	460,301
Germany	76,493	127,642	162,670	126,118	492,923
Greece	88,387	108,760	135,147	132,564	464,858
India	103,790	113,638	120,065	156,425	493,918
Italy	121,702	55,279	101,816	62,184	340,981
Japan	114,751	72,910	84,181	83,042	354,884
Portugal	133,818	105,482	64,238	114,762	418,300
Russia	153,666	131,503	142,269	103,141	530,579
Spain	118,200	106,035	109,268	100,301	433,804
UK	180,337	115,513	112,345	144,838	553,033
US	120,629	115,237	119,310	97,658	452,834
Grand Total	1,324,501	1,130,884	1,278,759	1,262,271	4,996,415

Copyright © GrapeCity, Inc. All rights reserved.

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Create a Datasource for OLAP

Create a new class inside the Models folder to create a data source for the OLAP control.

1. Add a new class to the folder **Models** (for example: `ProductData.cs`). For more information about how to add a new model, see Adding controls.
2. Add the following code to the model to define the data for OLAP.

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;

namespace OlapSample.Models
{
    public class ProductData
    {
        private static Random r = new Random();
        public int ID { get; set; }
        public string Product { get; set; }
        public string Country { get; set; }
        public DateTime Date { get; set; }
        public int Sales { get; set; }
        public int Downloads { get; set; }
        public bool Active { get; set; }
        public double Discount { get; set; }

        private static int randomInt(int max)
        {
            return (int)Math.Floor(r.NextDouble() * (max + 1));
        }
        public static IEnumerable<ProductData> GetData(int cnt)
        {
            string[] countries = "China,India,Russia,US,Germany,UK,Japan,Italy,Greece,Spain,Portugal".Split(',');
            string[] products = "Wijmo,Aoba,Xuni,Olap".Split(',');
            List<ProductData> result = new List<ProductData>();
            for (var i = 0; i < cnt; i++)
            {
                result.Add(new ProductData
                {
                    ID = i,
                    Product = products[randomInt(products.Length - 1)],
                    Country = countries[randomInt(countries.Length - 1)],
                    Date = new DateTime(2015, randomInt(5) + 1, randomInt(27) + 1),
                    Sales = randomInt(10000),
                    Downloads = randomInt(10000),
                    Active = randomInt(1) == 1 ? true : false,
                    Discount = r.NextDouble()
                });
            }
            return result;
        }
    }
}
```

Back to Top

Add an OLAP control

Create a Controller and View for OLAP control and follow the below steps to initialize an OLAP control.

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **MVC 5 Controller - Empty** template.
    2. Set name of the controller (for example: `OlapController`).
    3. Click **Add**.
4. Replace the method Index() with the following method.

```
public class OlapController : Controller
{
    // GET: Olap
    private static System.Collections.IEnumerable data = ProductData.GetData(1000).ToList();
    public ActionResult Index()
    {
        return View(data);
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        }
```

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `OlapController`.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the view name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view is for the controller. Copy the following code and paste it inside **Index.cshtml**.

```
@using SlicerControl.Models;
@model IEnumerable<ProductData>

@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
        .RowFields(pfcb => pfcb.Items("Country"))
        .ColumnFields(cfcb => cfcb.Items("Product"))
        .ValueFields(vfcb => vfcb.Items("Sales")))
<div class="row">
    <div class="col-sm-3 col-md-3">
        @(Html.C1().Slicer().Id("slicer").Header("Select to Filter")
         .PivotEngineId("indexEngine").Field("Country").ShowCheckboxes(true))
    </div>
    <div class="col-sm-9 col-md-9">
        @Html.C1().PivotGrid().Id("indexGrid").ItemsSourceId("indexEngine")
    </div>
</div>
```

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/**Olap/Index**) in the address bar of the browser to see the view.

Back to Top

グループ化

フィールドの書式設定を使用して、データをグループ化することができます。製品の連結リストがあり、特定の四半期内に注文されたすべての項目をまとめてグループ化するとします。実行時に**[フィールドの設定]**ダイアログボックスを使用するか、コードを使用できます。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

実行時にデータを年（四半期）別にグループ化するには

1. PivotPanelで、*[Date]*、*[Product]*、[Downloads]の各フィールドを選択してグリッドビューに追加します。
2. **[Date]**フィールドを右クリックし、**[行フィールド]**で**[フィールドの設定]**を選択します。**[フィールドの設定]**ダイアログボックスが表示されます。
3. **[フィルタ]**オプションで、**[すべて選択]**オプションを選択します。
4. **[形式]**ドロップダウンリストで**[四半期(yyyy "Q"q)]**オプションを選択して、四半期ごとに**PivotGrid**内でデータをグループ化します。
5. **[OK]**を再度クリックして、**[フィールドの設定]**ダイアログボックスを閉じます。

以下の画像は、年（四半期）で製品がグループ化されたデータを示します。

Copyright © GrapeCity, Inc. All rights reserved.

Date	Product	総計
− 2015 Q1	Olap	3,672
	Wijmo	9,710
	集計	13,382
− 2015 Q2	Aoba	16,263
	Olap	9,354
	Wijmo	16,050
	集計	41,667
総計		55,049

ビューの保存とロード

次のコードは、エクスポートがどのようにOLAPコントロールのビューを保存およびロードするかを示しています。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合 (Index.cshtml)

Razor

```
@using OLAPSample.Models;
@model IEnumerable<ProductData>
<br />
@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))
@Html.C1().PivotPanel().ItemsSourceId("indexEngine")
@Html.C1().PivotChart().ItemsSourceId("indexEngine")
@Html.C1().PivotGrid().Id("indexGrid").ItemsSourceId("indexEngine")

<button type="button" class="btn btn-default" onclick="saveView()">ビューを保存</button>
<br />
<button type="button" class="btn btn-default" onclick="loadView()">ビューをロード
</button><br />@section Scripts{
    <script src="http://cdnjs.cloudflare.com/ajax/libs/jszip/2.5.0/jszip.min.js">
</script>
    <script type="text/javascript">
        function saveView() {
            var ng = c1.getService('indexEngine');
            if (ng && ng.isViewDefined) {
                localStorage.viewDefinition = ng.viewDefinition;
            }
        }
        function loadView() {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            var ng = c1.getService('indexEngine');
            if (ng && localStorage.viewDefinition) {
                ng.viewDefinition = localStorage.viewDefinition;
                var cmbRowTotals = wijmo.Control.getControl('#RowTotals');
                if (cmbRowTotals) {
                    cmbRowTotals.selectedValue = ng.showRowTotals;
                }

                var cmbColTotals = wijmo.Control.getControl('#ColTotals');
                if (cmbColTotals) {
                    cmbColTotals.selectedValue = ng.showColumnTotals;
                }

                var chkShowZeros = document.getElementById('ColTotals');
                if (chkShowZeros) {
                    chkShowZeros.checked = ng.showZeros;
                }
            }
        }
    </script>
}
```

ソート

デフォルトでは、OLAP出力テーブルの結果は、キー("Argentina"、"Brazil" など)でソートされます。これは、必ずしもデータの表示に最適な方法であるとは限りません。それために、**PivotGrid**の**AllowSorting**プロパティを**True**(デフォルト)に設定します。これにより、ユーザーは、標準グリッドと同様に、列ヘッダーをクリックするだけでデータをソートできるようになります。ヘッダーをクリックするたびに、ソート順は"昇順"、"降順"、"ソートなし"の順に変更されます。

Country	Aoba	Olap	Wijmo	総計
米国	2,784			2,784
英国		3,443		3,443
ポルトガル		3,672		3,672
イタリア		5,911		5,911
ギリシャ			10,439	10,439
ドイツ	13,479			13,479
インド			15,321	15,321
総計	16,263	13,026	25,760	55,049

スタイル設定とCSS

CSS classes for the OLAP control are as follows:

Copyright © GrapeCity, Inc. All rights reserved.

Class	Description
.wj-pivotpanel	Styles the div host element of the PivotPanel control.
.wj-pivotpanel div	Styles all the div elements representing the fields, filters, columns, rows and values sections contained in the pivot panel.
.wj-pivotpanel table	Styles the table containing the Filters, Columns, Rows and Values sections of the pivot panel.
.wj-pivotpanel tr	Styles the table rows containing the filters, columns, rows and values sections
.wj-pivotpanel td	Styles the table cells containing the filters, columns, rows and values sections.
.wj-pivotpanel .wj-flexgrid .wj-cell.wj-group	Styles the cells of all the grids contained in the pivot panel.
.wj-pivotpanel .wj-flexgrid .wj-aggregate	Styles the aggregate function text displayed in the Values section of the pivot panel.
.wj-pivotpanel .wj-flexgrid	Styles all the grids in the pivot panel which are used to represent the list of fields and the values in the filters, columns, rows and values sections.
.wj-pivotpanel .wj-flexgrid label	Styles the labels used to display a list of fields in the pivot panel.
.wj-pivotpanel table .wj-flexgrid	Styles the grids which are used to represent the values in the filters, columns, rows and values sections.
.wj-pivotpanel label	Styles all the labels in the pivot panel, which represents the values and headers in different sections.
.wj-pivotpanel .wj-marker	Styles the marker displayed while dragging the fields from one area to other.
.wj-pivotpanel .wj-flexgrid .wj-cell	Styles the cells of grids which displayed in pivot panel.
.wj-pivotpanel .wj-flexgrid .wj-glyph-filter	Styles the glyph filter of grids which displayed in pivot panel.
.wj-pivotpanel .wj-flexgrid .wj-cell.wj-group:not(.wj-header):hover	Styles the cells of all the grids contained in the pivot panel while mouse hover.
.wj-pivotpanel .wj-flexgrid .wj-glyph-dimension	Styles the dimension glyphs in the pivot panel.
.wj-pivotpanel .wj-flexgrid .wj-glyph-measure	Styles the measure glyphs in the pivot panel.
Pivot Field Settings Dialog	
.wj-pivotfieldeditor	Styles the field settings dialog.
.wj-pivotfieldeditor tr.wj-separator	Styles the area between each input control on the field settings dialog.
.wj-pivotfieldeditor td:first-child	Styles the label text displayed for each field setting.
Pivot Filter Editor Dialog	
.wj-pivotfiltereditor	Styles the filter editor.
.wj-pivotfiltereditor .wj-control	Styles the div host element of the control displaying the filter.
.wj-pivotfiltereditor .wj-listbox	Styles the div host element of the ListBox control displaying the filter values.
.wj-pivotfiltereditor .wj-listbox .wj-listbox-item	Styles the div element representing the list items.

Copyright © GrapeCity, Inc. All rights reserved.

Context Menu	
.wj-olap-context-menu	Styles the context menu displayed for the values in the Filters, Columns, Rows and Values sections.
.wj-olap-context-menu .menu-icon	Styles menu icon displayed for the context menu options.
.wj-olap-context-menu .menu-icon.menu-icon-remove	Styles menu icon displayed for the Remove Field option in the context menu.
PivotGrid	
.wj-pivotgrid	Styles the div host element of the PivotGrid control.
.wj-detaildialog .wj-flexgrid	Styles the div host element of the FlexGrid control which is displayed inside the pivot grid detail dialog.
.wj-pivotgrid .wj-colheaders .wj-cell.wj-header	Styles the column header cells in the content panel of the PivotGrid.
.wj-pivotgrid .wj-topleft .wj-cell.wj-col-field-hdr	Styles the value field header cells in the top-left panel of the PivotGrid, when the ShowColumFieldHeaders or showValueFieldHeaders property is set to True.
.wj-pivotgrid .wj-aggregate.wj-cell:not(.wj-header):not(.wj-group):not(.wj-state-selected):not(.wj-state-multi-selected)	Styles the cells that displays aggregate values in the PivotGrid.
.wj-pivotgrid .wj-cell.wj-aggregate	Styles the cells that displays aggregate values and the header cells for each of this aggregate value cell in the PivotGrid.
PivotChart	
.wj-pivotchart	Styles the div host element of the PivotChart control.
.wj-pivotchart .wj-flexchart	Styles the div host element of the FlexChart in the PivotChart.
.wj-pivotchart .wj-dropdown	Styles the dropdown in chart.
Slicer	
.wj-slicer > div	Styles the div host element of the Slicer control.
.wj-slicer .wj-header	Styles the header of the Slicer control.
.wj-slicer .wj-header div:first-child	Styles the div which display header text of the slicer.
.wj-slicer .wj-header .wj-btn.wj-state-active	Styles active state of buttons in slicer's header.
.wj-slicer .wj-header .wj-viewer .wj-control .wj-state-active.wj-applybutton	Styles the active state of the apply button in slicer's header.
.wj-slicer .wj-listbox	Styles the div host element of the ListBox control displaying the filter list items in the slicer.
.wj-slicer .wj-listbox .wj-listbox-itemlabel	Styles all the labels representing the filter list items in the slicer.
.wj-slicer.wj-nocheck .wj-listbox .wj-listbox-item .wj-state-selected	Styles the currently selected filter list item.
.wj-listbox-item .wj-state-checked	Styles the currently checked filter list item.
.wj-slicer.wj-nocheck .wj-listbox .wj-listbox-item	Styles the filter list items when the ShowCheckboxes property is set to false.

スタイルの適用

Copyright © GrapeCity, Inc. All rights reserved.

The appearance of the OLAP control is defined using CSS classes which represents different elements, such as pivot panel, pivot field settings dialog, pivot filter editor dialog, context menu, pivot grid, pivot chart, and slicer. You can

```
    font-weight: normal;
    color:mediumblue;
    margin: 0;
}
.wj-pivotpanel .wj-aggregate {
    font-size: 100% !important;
    opacity: .8 !important;
    font-weight:bold;
    color:forestgreen !important;
}
.wj-olap-context-menu
{
    font: normal 11px verdana !important;
    padding: 6px;
}
.wj-pivotgrid .wj-colheaders .wj-cell.wj-header {
    text-align: left !important;
    font-style:italic !important;
    font-family:'Trebuchet MS', 'Lucida Sans', 'Arial', 'sans-serif' !important;
}
.wj-pivotgrid .wj-cell.wj-aggregate {
    font-weight:bold !important;
}
.wj-pivotfiltereditor a
{
    color:black !important;
}
.wj-pivotpanel .wj-flexgrid .wj-cell {
    border: none;
    background: inherit;
    color: mediumblue !important;
}
.wj-pivotpanel .wj-flexgrid label
{
    font-weight: bold;
    color:mediumblue !important;
    margin: 0;
}
.wj-pivotfiltereditor {
    background-color: #7373e3 !important;
    font: normal 14px verdana !important;
    border: 2px solid black !important;
}
.wj-valuefilter-editor {
    background-color: #7373e3 !important;
}
.wj-btn {
    background-color: #7373e3 !important;
}
.wj-control button.wj-btn:not(.wj-btn-default)
{
    border: 2px solid black !important;
}
```


Copyright © GrapeCity, Inc. All rights reserved.

```
.wj-dropdown-panel {
    background-color: #7373e3 !important;
}
.wj-cell.wj-group {
    color: mediumblue;
}
```

Styling.cshtml

Razor

```
@using <ApplicationName>.Models;
@model IEnumerable<ProductData>

@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
            .RowFields(pfcb => pfcb.Items("Country"))
            .ColumnFields(cfcb => cfcb.Items("Product"))
            .ValueFields(vfcb => vfcb.Items("Sales")))

@Html.C1().PivotPanel().ItemsSourceId("indexEngine")
@Html.C1().PivotChart().ItemsSourceId("indexEngine")
@Html.C1().PivotGrid().ItemsSourceId("indexEngine")
```

実行時にフィールドのプロパティを更新する

OLAPコントロールでは、オブジェクトモデルを使用して、実行時にフィールドプロパティにアクセスして更新することができます。クライアント側でフィールドを自動生成し、値フィールドのプロパティを設定することができます。

> **メモ**:フィールド名に複数の大文字が含まれる(例:ProductName)場合、大文字が出現するたびにフィールド名の間に1つのスペースが追加され、例外が生成されます。これは、ピボットフィールドのheaderプロパティをフィールド名と同じ値に設定することで解決することができます。

このトピックは、クライアント側でOLAPコントロールのフィールドプロパティにアクセスして更新する方法の理解に役立ちます。次の図は、JavaScriptを使用して実行時にフィールドプロパティを更新した後に、OLAPコントロールがどのように表示されるかを示します。

Country	Aoba	Olap	Wijmo	Grand Total
China		8,814.00		8,814.00
Germany			6,802.00	6,802.00
India			9,631.00	9,631.00
Japan		9,823.00	3,614.00	6,718.50
Portugal	4,067.00		8,046.00	6,056.50
Spain	6,503.00			6,503.00
UK			1,861.00	1,861.00
US	4,911.00			4,911.00
Grand Total	5,160.33	9,318.50	5,990.80	6,407.20

実行時にフィールドを自動生成し、値フィールドのプロパティを設定/更新する場合は、次のコード例を使用して、OLAPコントロールにピボットフィールドを定義します。この例では、「クイックスタート:OLAPへのデータの追加」トピックで追加され

Copyright © GrapeCity, Inc. All rights reserved.

たProductData.csモデルを使用します。

Razor

```
@using <ApplicationName>.Models;
@model IEnumerable<ProductData>

<script type="text/javascript">
        c1.documentReady(function () {
            var engine = c1.getService('indexEngine');
            var field = engine.fields.getField("Sales");
            field.aggregate = wijmo.Aggregate.Avg;
            field.format = "n2";
        })
</script>
@(Html.C1().PivotEngine().Id("indexEngine").Bind(Model)
    .RowFields(pfcb => pfcb.Items("Country"))
    .ColumnFields(cfcb => cfcb.Items("Product"))
    .ValueFields(vfcb => vfcb.Items("Sales")))
@(Html.C1().PivotGrid().ItemsSourceId("indexEngine"))s
```

PDF

PDFコントロールでは、WebアプリケーションからAdobe PDFドキュメントを作成できます。ドキュメントにコンテンツを追加するためのコマンドは、.NET Graphicsクラスのコマンドに類似しているので、セキュリティ、圧縮、アウトライン、ハイパーリンク、およびファイル添付を使用できます。

PDF for .NETでは、アプリケーションからAdobe PDFドキュメントを作成できます。.NETアプリケーションからPDFを作成するメリットは、基本的にデータ連結PDFドキュメントを作成できることです。

詳細については、PDF for .NET を参照してください。

主な特長

PDF には主に次の便利な機能があります。

- **メタファイル画像の高速レンダリングと圧縮**:
  チャートや図面は、ビットマップ画像より適したメタファイルを使用して、PDFドキュメントに追加できます。メタファイルはビットマップに変換されず、解析されてベクターグラフィックコマンドに変換されるため、最適な解像度が維持されます。

- **AcroFormsのサポート**:
  AddFieldメソッドを使用して、Acrobatフォームフィールドを PDFドキュメントに追加します。このPDFコントロールは、テキストボックス、チェックボックス、ラジオボタン、プッシュボタン、コンボボックス、リストボックスなどのフィールドタイプをサポートします。

- **HTMLのサポート**:
  DrawStringHtmlメソッドを使用して、HTMLをPDFドキュメントにレンダリングします。HTMLコンテンツを複数のページまたは列に流し込み、既存のスタイルシートを使用して、HTMLを他の種類のコンテンツ（画像、RTF、プレーンテキスト、フォームフィールドなど）と混在させることができます。

- **DrawImageメソッド**:
  DrawImageメソッドを使用して、PDFの指定した位置に画像を追加できます。DrawImageには、画像の配置とスケーリングを制御するためのパラメータがあります。メタファイルを含む通常の.NET Imageオブジェクトも PDFにレンダリングできます。

Copyright © GrapeCity, Inc. All rights reserved.

- **ドキュメント権限の管理**:
  ユーザーに対してコンテンツのコピーと編集の許可、ドキュメントの印刷の制限、注釈編集権限の設定などの管理が可能です。

- **添付ファイル**:
  添付ファイル、添付ファイルが含まれるページ領域、および添付ファイルの外観(オプション)を指定するだけで、PDFに添付ファイルを追加できます。

- **パスワード保護**:
  機密情報が含まれるPDFドキュメントを暗号化して、許可されたユーザーだけがアクセスできるようにします。ドキュメントの所有者と他のすべてのユーザーに、それぞれ別のパスワードが用意されます。ユーザーのアクセス権は、特定の操作(ドキュメントの表示、印刷、または編集)のみを許可するように選択的に制限できます。

- **アウトライン(ブックマーク)**:
  リーダーの左ペインに表示されるアウトラインエントリ(ブックマーク)を追加して、大きなPDFドキュメントでアウトライン構造を構築できます。アウトラインがあると、ドキュメントの構造の参照や特定のトピックの検索が簡単になります。

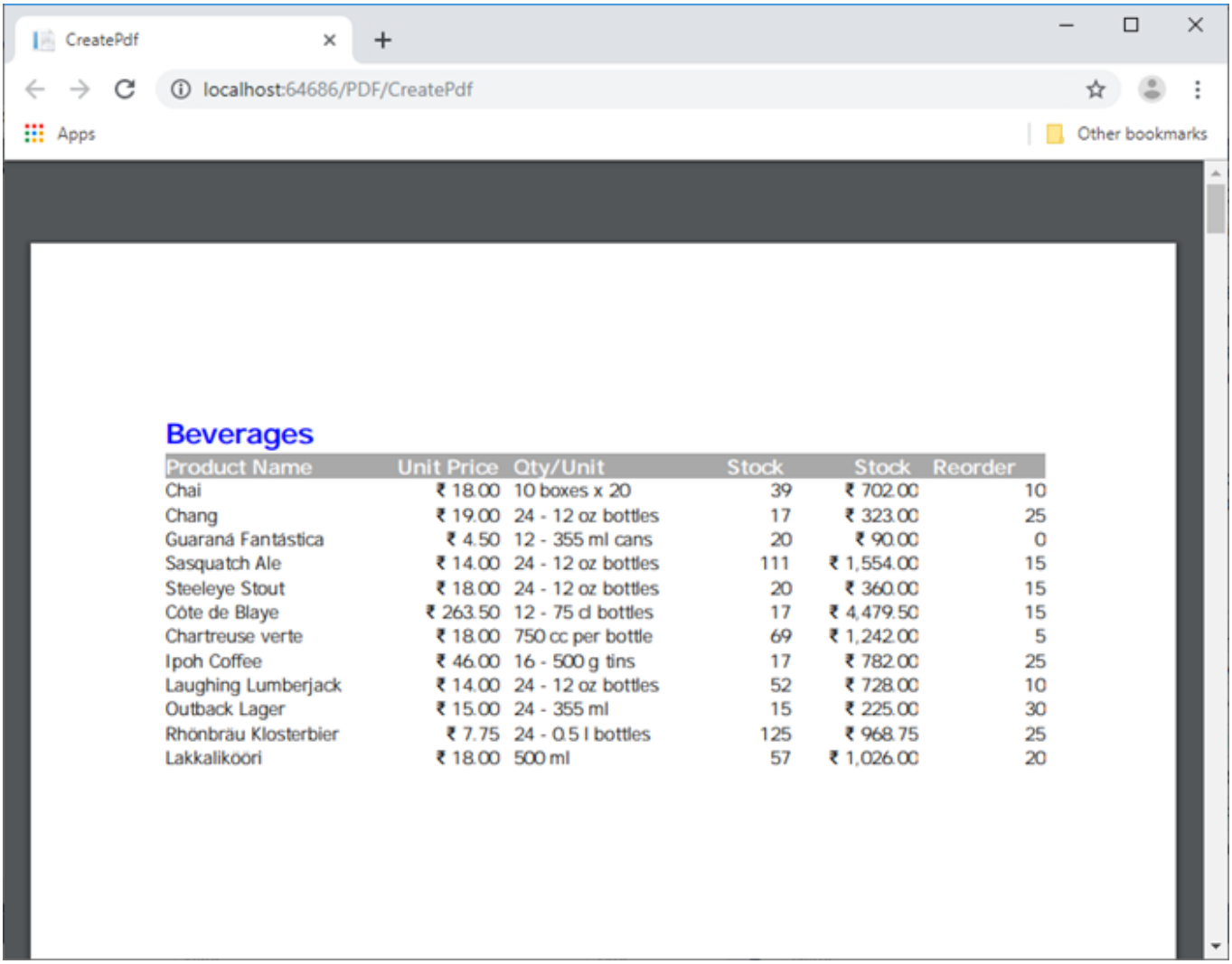
- **ハイパーリンクのサポート**:
  PDFドキュメントにハイパーリンクとハイパーリンクターゲットを追加できます。ローカルリンクも追加できます。ローカルリンクをクリックすると、ユーザーは同じPDFドキュメント内の別の場所に移動されます。

- **ドキュメントのメタデータと環境設定**:
  作成するPDFドキュメントにメタデータを追加できます。作成者、作成日付、キーワードなどを指定します。ドキュメントをAdobe Readerで開いたときに適用されるデフォルトのビューアオプションも提供できます。リーダーツールバーやメニュー表示のほかに、初期ページレイアウトやウィンドウ位置も指定します。

クイックスタート

The quick start guides you through the steps of adding C1PdfDocument in your MVC web application and adding data to it. Complete the following steps to see how the data appears in the generated PDF after data binding:

Copyright © GrapeCity, Inc. All rights reserved.



Back to Top

Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Configure the Data Source for Application

> The example uses **C1NWind** database. The **C1NWind.mdf** file is available on your system at the following location:
>
> *Documents\ComponentOne Samples\ASP.NET MVC\MVC\MvcExplorer\App_Data*

1. Add **C1NWind.mdf** file to the `AppData` folder in the Solution Explorer.
2. In the Solution Explorer, right-click `Models|Add New Item|Data`, and select **ADO.NET Entity Data Model**.
3. Name the model as **C1NWind**, and click **Add**.
4. In the Entity Data Model Wizard, select **EF Designer from database**, click **Next**. **C1NWind.mdf** database is added to the data connection dropdown.
5. Click **Next** to choose Entity Framework version, and click **Next**.
6. In the **Choose Your Database Objects and Settings**, select `Products` table and click **Finish**.

Copyright © GrapeCity, Inc. All rights reserved.

If you can see **C1NWind.edmx** added to your project under the **Models** folder, you have successfully configured the data source for your application.

Back to Top

Add C1PdfDocument

Complete the following steps to initialize a FlexGrid control.

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. Complete the following steps in the **Add Scaffold** dialog:
    1. Select **Empty MVC Controller** template.
    2. Set name of the controller (for example: `PDFController`).
    3. Click **Add**.
4. Include the MVC references as shown below.

C#

```
using <ApplicationName>.Models;
```

5. Generate a PDF using CreatePdf method as shown in the following code:

C# copyCode

```
private readonly C1NWindEntities _dataBase = new C1NWindEntities();
private C1PdfDocument _c1Pdf;
public ActionResult CreatePdf()
{
    try
    {
        InitialPdfDocument();
        var stream = new MemoryStream();
        _c1Pdf.Save(stream);
        stream.Position = 0;
        return File(stream, "application/pdf");
    }
    catch (Exception ex)
    {
        return Content(ex.Message);
    }
}
```

6. Add pages to the PDF for each product category using the following code.

C# copyCode

```
private readonly Font _fontBody = new Font("Tahoma", 10);
private readonly Font _fontTitle = new Font("Tahoma", 15, FontStyle.Bold);
private readonly Font _fontHeader = new Font("Tahoma", 11, FontStyle.Bold);
private StringFormat _sfRight;
private StringFormat _sfRightCenter;
public void InitialPdfDocument()
{
    _c1Pdf = new C1PdfDocument();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    //右揃えのフィールドに対してStringFormatを作成します。
    _sfRight = new StringFormat();
    _sfRight.Alignment = StringAlignment.Far;
    _sfRightCenter = new StringFormat();
    _sfRightCenter.Alignment = StringAlignment.Far;
    _sfRightCenter.LineAlignment = StringAlignment.Center;

    //PDFジェネレータを初期化します。
    _c1Pdf.Clear();

    //ページの四角形、割引率を取得します。
    RectangleF rcPage = _c1Pdf.PageRectangle;
    rcPage.Inflate(-72, -92);

    //選択したカテゴリをループします。
    int page = 0;
    List<Category> categories = _dataBase.Categories.ToList();

    foreach (Category category in categories)
    {
        //改ページを追加し、ページカウンタを更新します。
        if (page > 0) _c1Pdf.NewPage();
        page++;

        //現在のカテゴリ名を取得します。
        string catName = category.CategoryName;

        //ページにタイトルを追加します。
        _c1Pdf.DrawString(catName, _fontTitle, Brushes.Blue, rcPage);

        //アウトラインエントリを追加します。
        _c1Pdf.AddBookmark(catName, 0, 0);

        //行テンプレートを作成します。
        RectangleF[] rcRows = new RectangleF[6];
        for (int i = 0; i < rcRows.Length; i++)
        {
            rcRows[i] = RectangleF.Empty;
            rcRows[i].Location = new PointF(rcPage.X, rcPage.Y +
_fontHeader.SizeInPoints + 10);
            rcRows[i].Size = new SizeF(0, _fontBody.SizeInPoints + 3);
        }
        rcRows[0].Width = 110;          // 商品名
        rcRows[1].Width = 60;          // 単価
        rcRows[2].Width = 80;          // 数量/単位
        rcRows[3].Width = 60;          // 単位あたりの数量
        rcRows[4].Width = 60;          // 株価
        rcRows[5].Width = 60;          // 再注文
        for (int i = 1; i < rcRows.Length; i++)
            rcRows[i].X = rcRows[i - 1].X + rcRows[i - 1].Width + 8;

        //列ヘッダーを追加します。
        _c1Pdf.FillRectangle(Brushes.DarkGray, RectangleF.Union(rcRows[0],
```


Copyright © GrapeCity, Inc. All rights reserved.

```
rcRows[5]));
        _c1Pdf.DrawString("Product Name", _fontHeader, Brushes.White,
rcRows[0]);
        _c1Pdf.DrawString("Unit Price", _fontHeader, Brushes.White, rcRows[1],
_sfRight);
        _c1Pdf.DrawString("Qty/Unit", _fontHeader, Brushes.White, rcRows[2]);
        _c1Pdf.DrawString("Stock Units", _fontHeader, Brushes.White, rcRows[3],
_sfRight);
        _c1Pdf.DrawString("Stock Value", _fontHeader, Brushes.White, rcRows[4],
_sfRight);
        _c1Pdf.DrawString("Reorder", _fontHeader, Brushes.White, rcRows[5]);

        //このカテゴリの製品をループします。
        List<Product> products = _dataBase.Products.Where(pro => pro.CategoryID
== category.CategoryID).ToList();

        foreach (Product product in products)
        {
            //次の行に移動します。
            for (int i = 0; i < rcRows.Length; i++)
                rcRows[i].Y += rcRows[i].Height;
                _c1Pdf.DrawString(product.ProductName, _fontBody, Brushes.Black,
rcRows[0]);
                _c1Pdf.DrawString(string.Format("{0:c}", product.UnitPrice),
_fontBody, Brushes.Black, rcRows[1], _sfRight);
                _c1Pdf.DrawString(string.Format("{0}", product.QuantityPerUnit),
_fontBody, Brushes.Black, rcRows[2]);
                _c1Pdf.DrawString(string.Format("{0}", product.UnitsInStock),
_fontBody, Brushes.Black, rcRows[3], _sfRight);
                _c1Pdf.DrawString(string.Format("{0:c}", product.UnitPrice *
product.UnitsInStock), _fontBody, Brushes.Black, rcRows[4], _sfRight);
                _c1Pdf.DrawString(string.Format("{0}", product.ReorderLevel),
_fontBody, Brushes.Black, rcRows[5], _sfRight);
        }
    }
}
```

Add a View for the controller:

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the controller (for example: `ExcelController`) to open it.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add**. A view is added for the controller.
6. In the Solution Explorer, double click `Index.cshtml` to open it.
7. Replace the default code in the **Views\Index.cshtml** file with the following code to display a message about the PDF being generated.

C#

```
@{
    Layout = null;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
@Html.ActionLink("GenerateExcel", "GenerateExcel")
```

Back to Top

Build and Run the Project

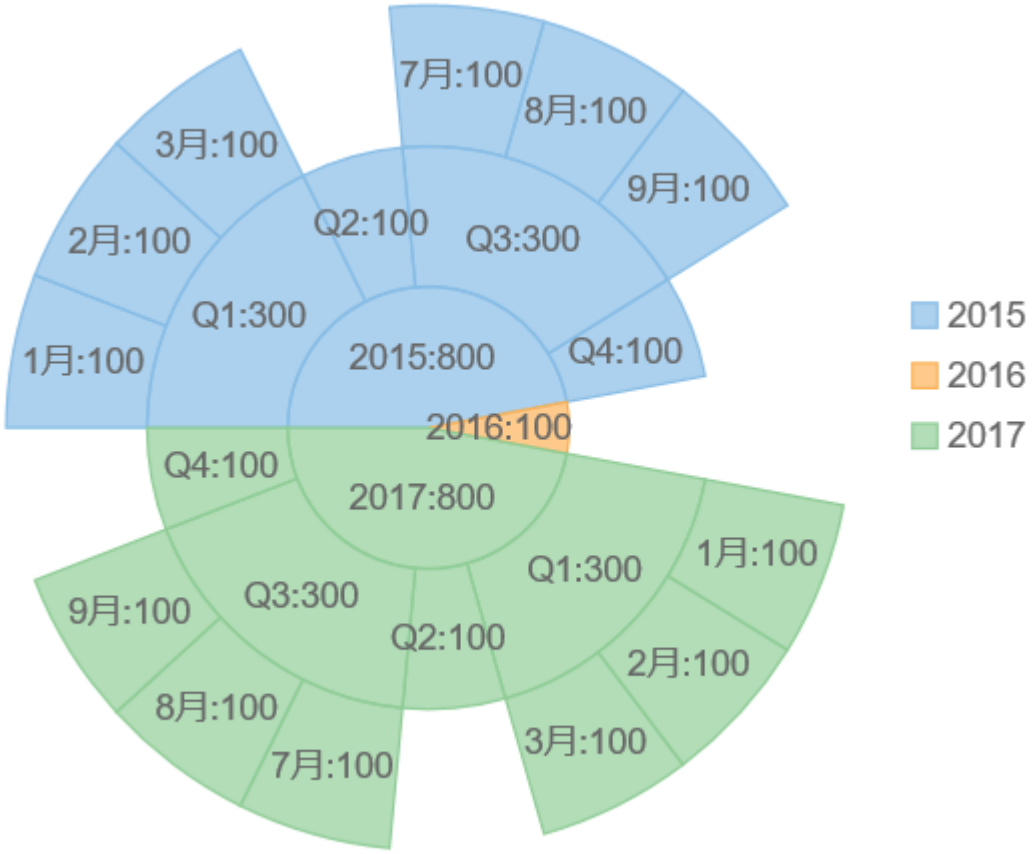
1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

This creates a PDF with NorthWind product information wherein each product category is placed on a separate page and added to an outline structure with bookmarks.

Back to Top

SunburstChart

Sunburstチャートは、同心円を使用して階層化データを表示するために使用されます。中心の円はルートノードを表し、中心から外に向けてデータが移動します。内側の円の各セクションは、外側の円のいくつかのセクションとの階層的な関係をサポートし、外側の円のセクションは親セクションの角度範囲に収まります。Sunburstチャートは、外側の輪と内側の輪の関係を使用して表される階層化データを表示するシナリオで効果的に使用できます。



主な特長

Sunburstチャートには主に次の便利な機能があります。

- **ドーナッツチャートのサポート**
  コントロールの内側半径のプロパティを指定してドーナッツチャートをサポートします。

Copyright © GrapeCity, Inc. All rights reserved.

- **凡例**
  チャートにレンダリングされるデータの簡単な説明を提供します。また、必要に応じて凡例の位置を変更できます。
- **選択**
  選択モードを変更したり、円グラフのセグメントの外観をカスタマイズすることができます。
- **カラーテーマ**
  デフォルトの色のオプションからさまざまな色を選択して、Sunburstグラフセグメントのレンダリング時に使用できます。
- **ヘッダーとフッター**
  タイトルヘッダー/フッターテキストを指定します。

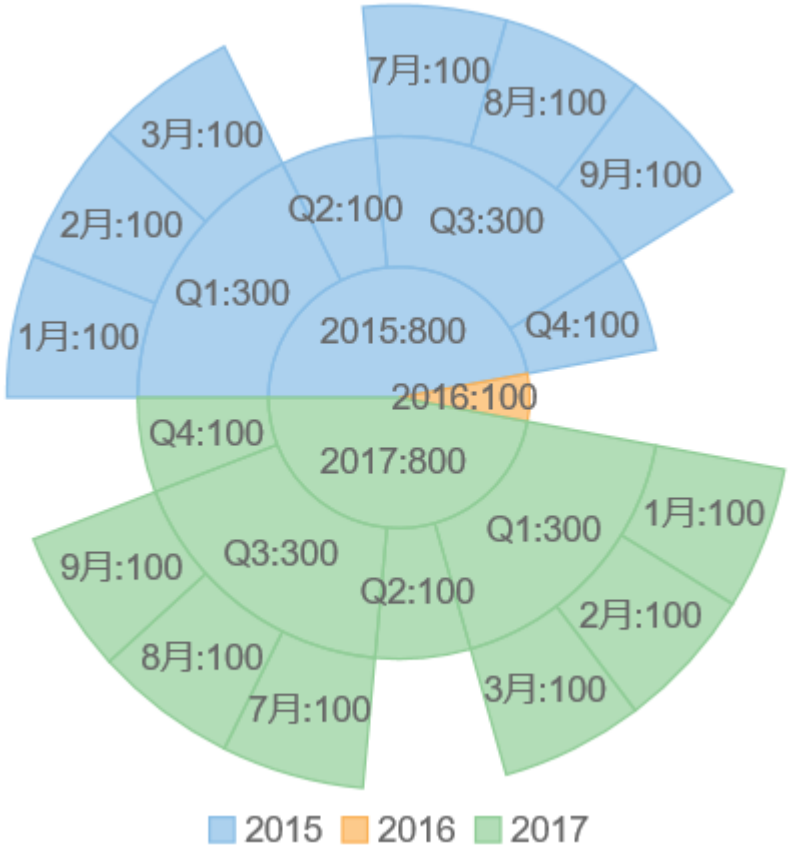
クイックスタート:Sunburst へのデータの追加

このセクションでは、MVC WebアプリケーションにSunburstチャートを追加し、そこにデータを追加する方法について説明します。

このトピックは3つの手順で構成されます。

- **手順1:MVC アプリケーションの作成**
- **手順2:Sunburstのデータソースの作成**
- **手順2:Sunburstチャートの追加**
- **手順3:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のSunburstチャートを示しています。



手順 1:MVC アプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して新しいMVCアプリケーションを作成します。MVCアプリケーション

Copyright © GrapeCity, Inc. All rights reserved.

の作成の詳細については、「MVCアプリケーションの設定」を参照してください。

手順2：Sunburstのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`HierarchicalData.cs`）。新しいモデルの追加方法については、「コントロールの追加」を参照してください。
2. 新しいモデルに次のコードを追加して、クラスを定義します。

C#

SunburstController.cs

```
public class HierarchicalData
{
 public int ID {get; set;}
 public int Year {get; set;}
 public string Quarter {get; set;}
 public string Month {get; set;}
 public int Value {get; set;}
public HierarchicalData(int year, string quarter, string month, int value)
 {
  Year = year;
  Quarter = quarter;
  Month = month;
  Value = value;
 }
 public HierarchicalData() {}
 public static List < HierarchicalData > GetData()
 {
  var data = new List < HierarchicalData > ();

  var times = new string[4, 3] {
   {"1月","2月","3月"},
   {"4月","5月","6月"},
   {"7月","8月","9月"},
   {"10月","11月","12月"}
  };
  var years = new int[] { 2015,2016,2017};

  for (int i = 0; i < years.Length; i++)
  {
   if (i % 2 == 0)
   {
    for (int j = 0; j < 4; j++)
    {
     string quar = "Q" + (j + 1);
     if (j % 2 == 0)
     {
      for (int k = 0; k < 3; k++)
      {
       data.Add(new HierarchicalData(years[i], quar, times[j, k], 100));
      }
     } else
```


Copyright © GrapeCity, Inc. All rights reserved.

```
      {
       data.Add(new HierarchicalData(years[i], quar, null, 100));
      }
     }
    } else
    {
     data.Add(new HierarchicalData(years[i], null, null, 100));
    }
   }
   return data;
  }
}
```

先頭に戻る

手順 3：Sunburst チャートの追加

Sunburst チャートを初期化するには、次の手順を実行します。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを右クリックします。
2. コンテキストメニューから、[**追加** →]→[**コントローラー**]を選択します。
3. [**スキャフォールディングを追加**]ダイアログで、次の手順を実行します。
    1. [**MVC 5 コントローラー -空**]テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`SunburstController`）。
    3. [**追加**]をクリックします。
4. 次に示すように MVC 参照を追加します。

C#

```
using C1.Web.Mvc;
using C1.Web.Mvc.Serializition;
using C1.Web.Mvc.Chart;
```

5. メソッド Index() を次のメソッドに置き換えます。

C#

SunburstController.cs

```
public ActionResult Index()
        {
            return View(Models.HierarchicalData.GetData());
        }
```

ビューの追加

1. **ソリューションエクスプローラー**で、[**コントローラー**]フォルダを展開し、コントローラー`SunburstController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、[**ビューの追加**]を選択します。[**ビューの追加**]ダイアログが表示されます。
4. [**ビューの追加**]ダイアログで、ビュー名が **Index** で、ビューエンジンが **Razor (CSHTML)**であることを確認します。
5. [**追加**]をクリックします。コントローラーにビューが追加されます。

Razor

Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().Sunburst<HierarchicalData>()
.Bind("Year", "Value", Model)
.DataLabel(dl => dl.Content("{name}:{value}").Position(PieLabelPosition.Center))
.Header("2015, 2016, 2017")
.BindingName("Year", "Quarter", "Month")
.Width(500).Height(500))
```

先頭に戻る

手順4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http://localhost:1234/**Sunburst/Index**)。

先頭に戻る

Sunburstの使用

This section comprises all the features and auxiliary functionalities offered by Sunburst Chart.

ヘッダーとフッター
: Learn how to set header and hooter properties.

凡例
: Learn how to set the position of legend in Sunburst chart.

選択
: Learn about the selection modes available in Sunburst chart.

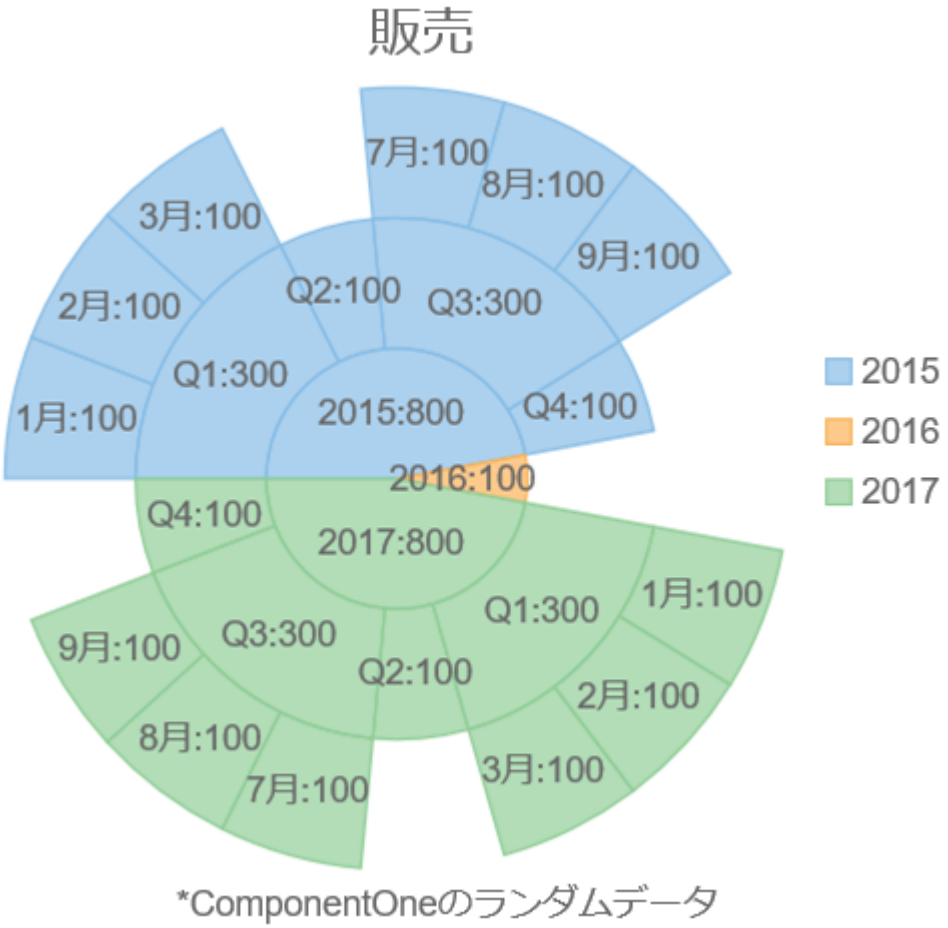
テーマ
: Learn how to use different themes on Sunburst chart.

ヘッダーとフッター

Headerプロパティを利用して、Sunburstチャートにヘッダーとタイトルを提供できます。さらに、**Footer**プロパティを利用して、Sunburstチャートにフッターを設定することもできます。これらのプロパティを使用すると、**HeaderStyle**および**FooterStyle**プロパティでヘッダー/フッターテキストのスタイルを設定できます。

以下の画像は、HeaderおよびFooterプロパティを設定した後にSunburstチャートがどのように表示されるかを示します。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、ヘッダーとフッターのプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

Razor

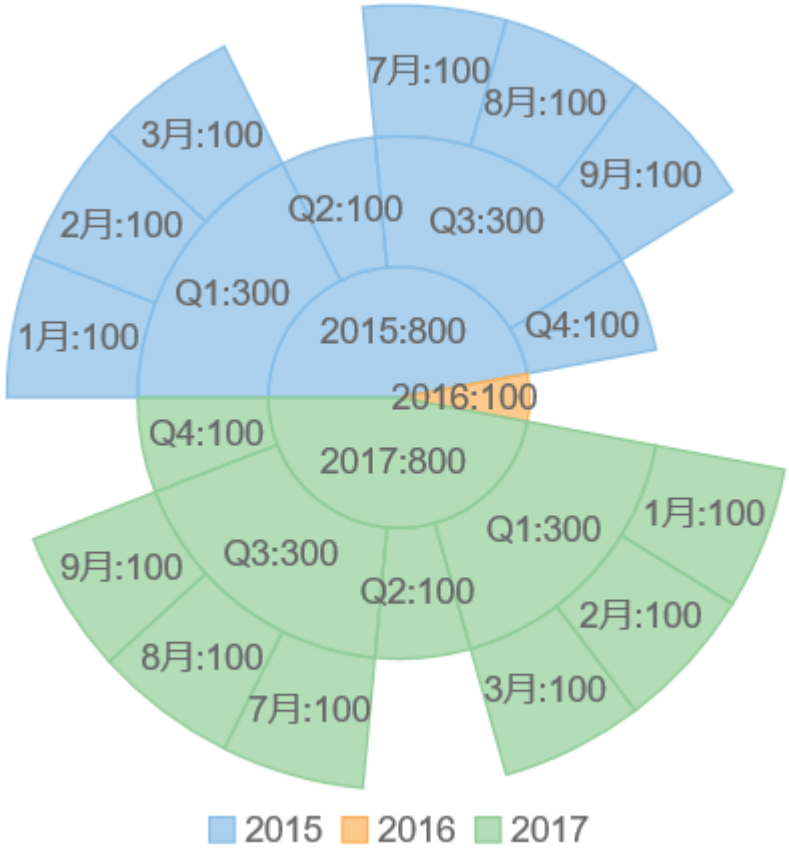
```
.Header("Sales Data")
.HeaderStyle(style => style.FontSize("30"))
.Footer("*ComponentOneのランダムデータ")
.FooterStyle(style => style.FontSize("15"))
```

凡例

SunburstチャートのLegendプロパティを使用して、凡例を表示する場所を指定できます。凡例を使用すると、チャートの系列を定義済みの記号および名前付きで表示できます。

以下の画像は、Chart.PositionプロパティをBottomに設定した後にSunburstチャートがどのように表示されるかを示します。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、Positionプロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor

```
.Legend(C1.Web.Mvc.Chart.Position.Bottom)
```

選択

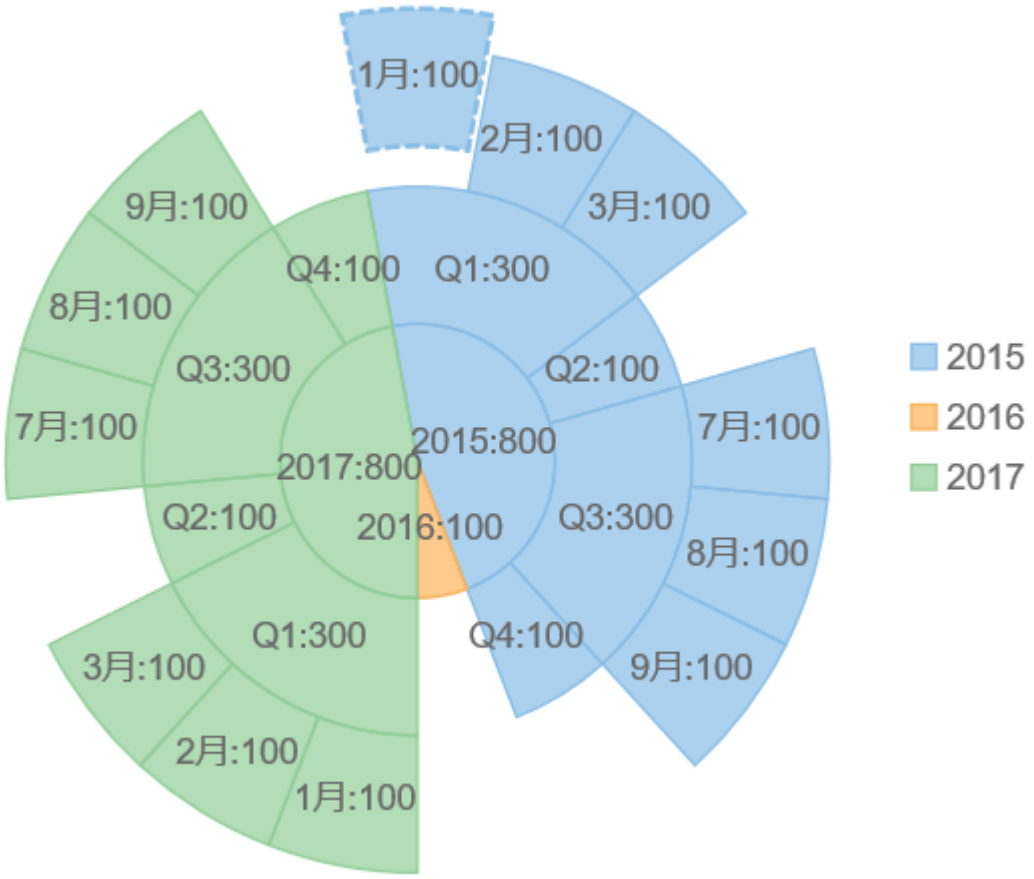
SelectionMode プロパティを設定して、ユーザーがSunburstチャート内の任意の領域をクリックしたときにSunburstチャートのどの要素を選択するかを選択できます。次の3つのオプションが表示されます。

- **None**:どの要素も選択されません。
- **Point**:ユーザーがクリックした円グラフセグメントが強調表示されます。
- **Series**:円全体を強調表示します。

SelectionMode が**Point**に設定されている場合は、**SelectedItemPosition** プロパティを設定して、選択されているSunburstグラフセグメントの位置を変更できます。また、**SelectedItemOffset** プロパティを設定して、選択したSunburstグラフセグメントを中心から離すことができます。

以下の画像は、**SelectionMode** プロパティを設定した後にSunburstチャートがどのように表示されるかを示します。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、SelectionMode プロパティを設定する方法を示します。この例では、「クイックスタート」セクションで作成したサンプルを使用します。

コードの場合

Razor

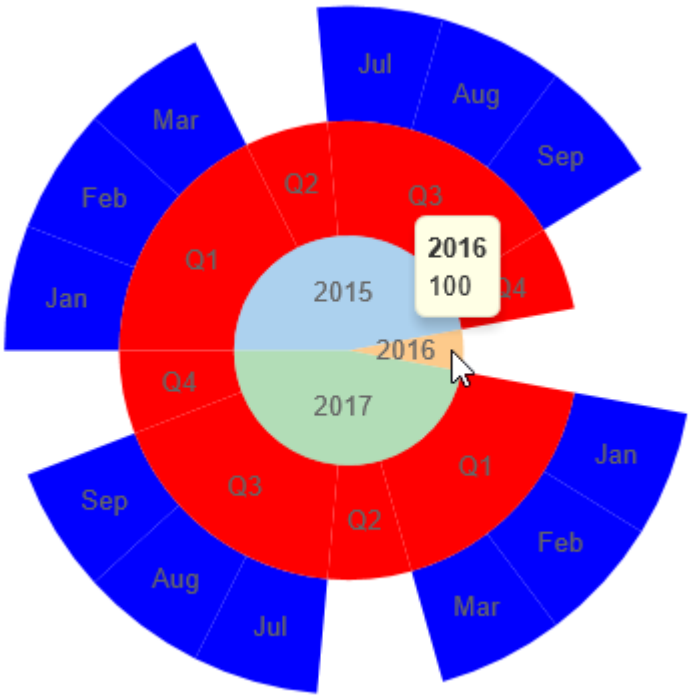
```
.SelectionMode(C1.Web.Mvc.Chart.SelectionMode.Point)
.SelectedItemPosition(C1.Web.Mvc.Chart.Position.Top)
.SelectedItemOffset(0.1f)
```

テーマ

Sunburstチャートコントロールの外観の大部分はCSSで定義されます。デフォルトのテーマのほかに、すべてのMVCコントロールの外観をカスタマイズして、魅力的で統一感のある見栄えを実現することができる洗練されたデザインのテーマが含まれています。CSSを使用してSunburstチャートコントロールの外観をカスタマイズできます。それには、CSSルールをデフォルトのテーマから新しいCSSファイルにコピーし、必要に応じてプロパティを変更します。

この例では、コントロールに「**custom-sunburst-chart**」CSSクラスを追加し、CSSルールを定義して、ヘッダーの塗りつぶし、フォントファミリ、およびフォントウェイト、そしてセグメントの塗りつぶし色を変更しています。以下のコード例では、「クイックスタート」セクションで追加した**HierarchicalData.cs**モデルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.



コードの場合

Razor

```
@using C1.Web.Mvc.Chart
<style type="text/css">
        .custom-sunburst-chart.wj-sunburst .wj-header .wj-title {
            fill: #666;
            font-family: 'Courier New', Courier, monospace;
            font-weight: bold;
        }

        .custom-sunburst-chart.wj-sunburst ellipse,
        .custom-sunburst-chart.wj-sunburst path {
            stroke-width: 0;
        }

        .custom-sunburst-chart.wj-sunburst ellipse {
            fill: yellow;
        }

        .custom-sunburst-chart.wj-sunburst .slice-level2 > path {
            fill: red;
        }

        .custom-sunburst-chart.wj-sunburst .slice-level3 > path {
            fill: blue;
        }

        .custom-sunburst-chart.wj-sunburst .slice-level4 > path {
            fill: black;
        }
```

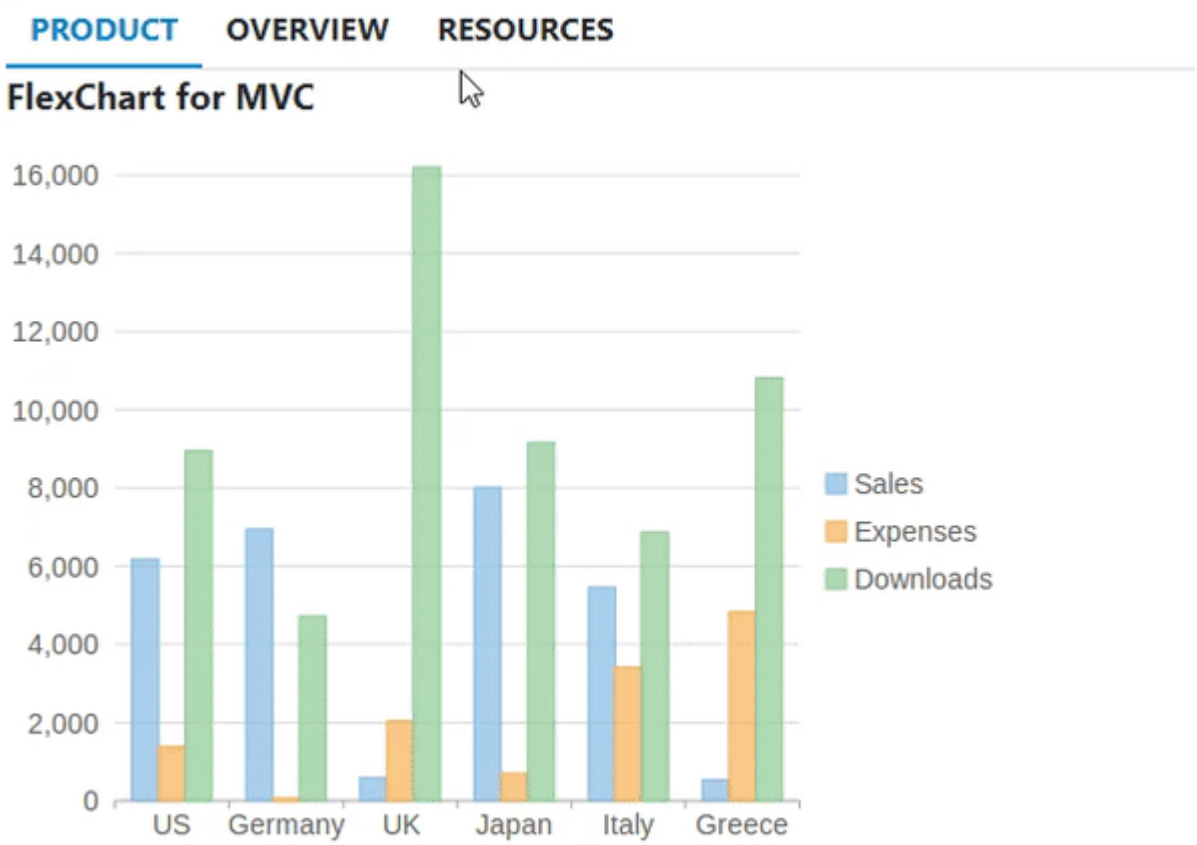

Copyright © GrapeCity, Inc. All rights reserved.

```
        .custom-sunburst-chart.wj-sunburst .slice-level5 > path {
            fill: white;
            stroke-width: 2;
            stroke: black;
        }
</style>
@using SunburstSample.Models;
@model IEnumerable<HierarchicalData>

@(Html.C1().Sunburst<HierarchicalData>()
    .Bind("Year", "Value", Model)
    .DataLabel(dl => dl.Content("{name}").Position(PieLabelPosition.Center))
    .BindingName("Year, Quarter, Month")
    .CssClass("custom-sunburst-chart"))
```

TabPanel

TabPanel control allows the user to enable high level content organization, such as switching between views, data sets, or functional aspects of an application. The control not only helps in organizing the content but is especially helpful when there is a lot of related content to be displayed on a page with a smaller screen real estate. TabPanel consists of a header section and a content pane. The header section consists of tabs and can be presented in a single row or column while content pane shows the associated content alongside the tabs on selecting a tab.



主な特長

Copyright © GrapeCity, Inc. All rights reserved.

TabPanel provides many different features that enable the developers to build intuitive and professional-looking applications. The main features for TabPanel are as follows:

- **Hosting Controls**
  The TabPanel control not only allows you to render html content, but also the MVC controls into the content pane.

- **Disabled and Invisible Tabs**
  The TabPanel control provides you an option to hide or disable individual tabs.

- **Styling and CSS**
  The TabPanel control has a simple layout, which makes it easy to style using CSS. For example, tabs appear above the content by default, but you can use CSS to change their position and show them below or vertically stacked, to the left or to the right of the content.

- **Accessibility**
  The TabPanel control implements WAI-ARIA accessibility guidelines that allows you to use all interactive UI components using a keyboard.. All tab elements have the proper role attributes as well as all applicable ARIA attributes.

基本クラス

The following table lists some of the key classes and their properties. You can click on the cross-references to get to the API documentation that provides detailed description for the key members.

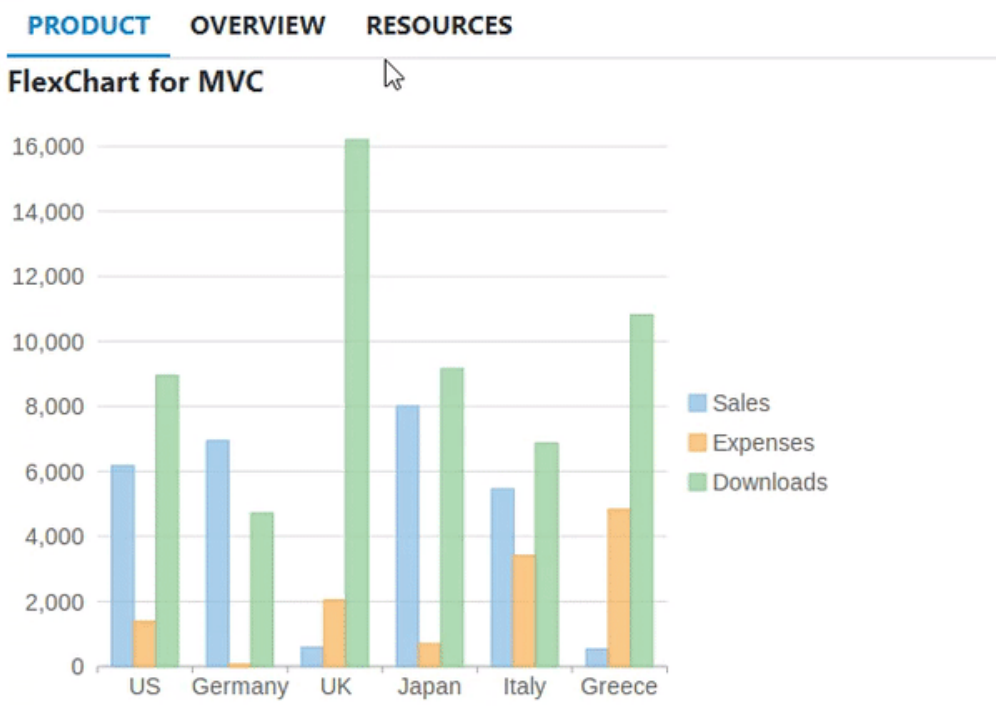
TabPanel
Properties: AutoSwitch, **IsAnimated**, **OnClientSelectedIndexChanged**, **SelectedIndex**, **Tabs**
Tab
Properties: Header, **IsDisabled**, **IsVisible**, **Pane**

クイックスタート:TabPanelへのデータの追加

The quick start guides you through the steps of adding the TabPanel control to your MVC web application and adding data to it.

To accomplish this, follow these steps:

Copyright © GrapeCity, Inc. All rights reserved.



Create an MVC Application

Create a new MVC application using the ComponentOne or VisualStudio templates. For more information about creating an MVC application, see Configuring your MVC Application topic.

Back to Top

Add a TabPanel control and add data

Steps to add a TabPanel control to the application, are as follows:

Add a new Controller

1. In the **Solution Explorer**, right click the folder **Controllers.**
2. From the context menu, select **Add | Controller**. The **Add Scaffold** dialog appears.
3. In the **Add Scaffold** dialog, follow these steps:
    1. Select the **MVC 5 Controller - Empty** template.
    2. Set name of the controller (For example: `TabPanelController`).
    3. Click **Add**.
4. Replace the **Index()** method with the following method.

TabPanelController.cs

```
public ActionResult Index()
{
    return View();
}
```

Add a View for the Controller

1. From the **Solution Explorer**, expand the folder **Controllers** and double click the `TabPanelController`.
2. Place the cursor inside the method `Index()`.
3. Right click and select **Add View**. The **Add View** dialog appears.
4. In the **Add View** dialog, verify that the View name is **Index** and View engine is **Razor (CSHTML).**
5. Click **Add** to add a view for the controller, and then copy the following code and paste it inside **Index.cshtml**.

The markup used to initialize the TabPanel control is a nested collection of **div** tags. The first div tag in the markup represents the TabPanel control. In this parent div tag, there exist few more div tags where each set of div tag represents a tab. Inside every **div** tag at this level, there is an **a** tag that renders the tab header and a div tag that renders the content pane.

Index.cshtml

Copyright © GrapeCity, Inc. All rights reserved.

```
<div id="tabPanel" style="width:500px">
    <div>
        <a>Product</a>
        <div>
            <p style="font-size:large"><b>FlexChart for MVC</b></p>
            <!--Add a custom image-->
            <img src="~/flexchart.png" />
        </div>
    </div>
    <div>
        <a>Overview</a>
        <div>
            <p><b>FlexChart</b>—a powerful data visualization control for Web—lets you add
feature-rich and visually appealing charts to your MVC applications.
            The control empowers end-users to visualize data that resonates with their audiences.
The FlexChart control provides you with numerous 2D chart types,
            built-in tools for chart interactivity, and diversified formats for chart
rendering.Whether it is storytelling with data or interpreting complex data,
            FlexChart helps you accomplish everything seamlessly.</p>
        </div>
    </div>
    <div>
        <a>Resources</a>
        <div>
            <ul>
                <li><a
href="http://help.grapecity.com/componentone/NetHelp/FlexChartWin/webframe.html">Documentation</a>
</li>
                <li><a href="https://www.grapecity.com/en/demos">Demo</a></li>
                <li><a href="https://www.grapecity.com/en/forums">Forums</a></li>
            </ul>
        </div>
    </div>
</div>
@(Html.C1().TabPanel("#tabPanel"))
```

Back to Top

Build and Run the Project

1. Click **Build | Build Solution** to build the project.
2. Press **F5** to run the project.

> Append the folder name and view name to the generated URL (for example: http://localhost:1234/**TabPanel/Index**) in the address bar of the browser to see the view.

Back to Top

TabPanel の使用

This section contains information about

MVC コントロールのホスティング
: Learn how to render MVC controls in the TabPanel control.

遅延ロード無効と非表示タブ
: Learn how to disable or hide tabs in the TabPanel control.

ノード間の移動分離されたパネル
: Learn how to display tab content in a different element.

ノードの展開と折りたたみ右から左

Copyright © GrapeCity, Inc. All rights reserved.

Learn how TabPanel supports the right-to-left feature.

チェックボックスの使用スタイル設定

Learn how to customize the appearance of TabPanel using CSS.

ユーザー補助

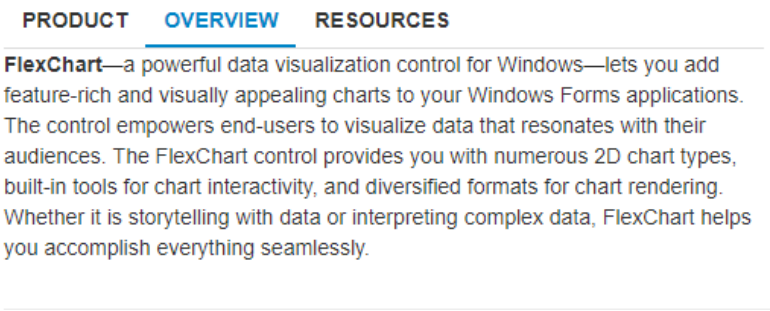
Learn WAI-ARIA accessibility guidelines.

ユーザー補助

The TabPanel implements WAI-ARIA accessibility guidelines that allow you to use all interactive UI components using a keyboard. All tab elements have the proper role attributes as well as all applicable ARIA attributes.

The example below assigns unique IDs to the tab headers and panes, which TabPanel automatically uses to generate "aria-controls" and "aria-labelledby" attributes. Moreover, it also adds a "wj-state-active" class to the "Product" pane in markup to define the tab that is initially selected. Otherwise, the first tab will be selected by default. For more information, see W3C ARIA standards and Simple Accessible article.

The following image shows how the TabPanel control appears with ARIA accessibility support.



The following code example demonstrates how to use ARIA accessibility support features.

Index.cshtml

```
@using C1.Web.Mvc
<div id="tabPanel" style="width:500px">
    <div>
        <a id="hdr-product">Product</a>
        <div id="pane-product">
            <p style="font-size:large"><b>FlexChart for MVC</b></p>
            <img src="~/flexchart.png" />
        </div>
    </div>
    <div>
        <a id="hdr-overview" class="wj-state-active">Overview</a>
        <div id="pane-overview">
    <p>
        <b>FlexChart</b>—a powerful data visualization control for Web—lets you add feature-rich and
        visually appealing charts to your MVC applications.     The control empowers end-users to visualize data
that
        resonates with their audiences. The FlexChart control provides you with numerous 2D chart types,
        built-in tools for chart interactivity, and diversified formats for chart rendering. Whether it is
        storytelling with data or interpreting complex data, FlexChart helps you accomplish everything
seamlessly.
        </p>
        </div>
    </div>
    <div>
        <a id="hdr-resources">Resources</a>
        <div id="pane-resources">
            <ul>
                <li><a
href="http://help.grapecity.com/componentone/NetHelp/FlexChartWin/webframe.html#FlexChart.html">Documentation</a>
</li>
                <li><a href="https://www.grapecity.com/en/demos">Demo</a></li>
                <li><a href="https://www.grapecity.com/en/forums">Forums</a></li>
            </ul>
        </div>
    </div>
</div>
```

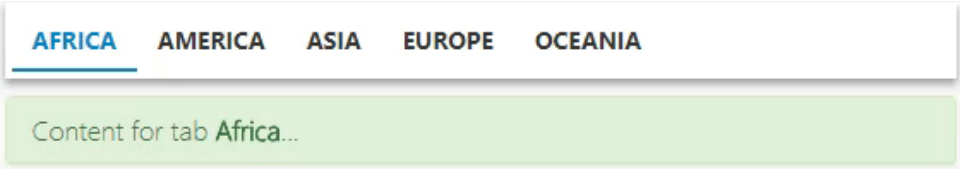

Copyright © GrapeCity, Inc. All rights reserved.

```
@(Html.C1().TabPanel("#tabPanel").AutoSwitch(true))
```

分離されたパネル

The TabPanel has a built-in panel that shows the content of the selected tab. However, in some cases, you may want to display the content in a different element. To implement this, simply hide the built-in content element and use the **SelectedIndex** property to update the content.

The following GIF image shows how TabPanel appears with detached panels.



The following code example demonstrates how to display tab content in a different html element.

Index.cshtml

```
<script>
    c1.documentReady(function () {
        var tabDetached = wijmo.Control.getControl('#tabDetached');
        tabDetached.hostElement.querySelector('.wj-tabpanes').style.display = 'none';
        tabDetached_selectedIndexChanged(tabDetached);
    });

    function tabDetached_selectedIndexChanged(s, e) {
        var div = document.getElementById('detachedContent');
        div.innerHTML = 'Content for tab <b>' +
            s.selectedTab.header.textContent +
            '</b>...';
    }
</script>

<div id="tabDetached">
    <!-- tab without any content -->
    <div>
        <a>Africa</a>
        <div></div>
    </div>
    <div>
        <a>America</a>
        <div></div>
    </div>
    <div>
        <a>Asia</a>
        <div></div>
    </div>
    <div>
        <a>Europe</a>
        <div></div>
    </div>
    <div>
        <a>Oceania</a>
        <div></div>
    </div>
</div>
@Html.C1().TabPanel("#tabDetached").OnClientSelectedIndexChanged("tabDetached_selectedIndexChanged")

<p></p>
<div class="panel panel-success">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    <div class="panel-heading">
        <h3 id="detachedContent" class="panel-title"></h3>
    </div>
</div>
```

無効と非表示タブ

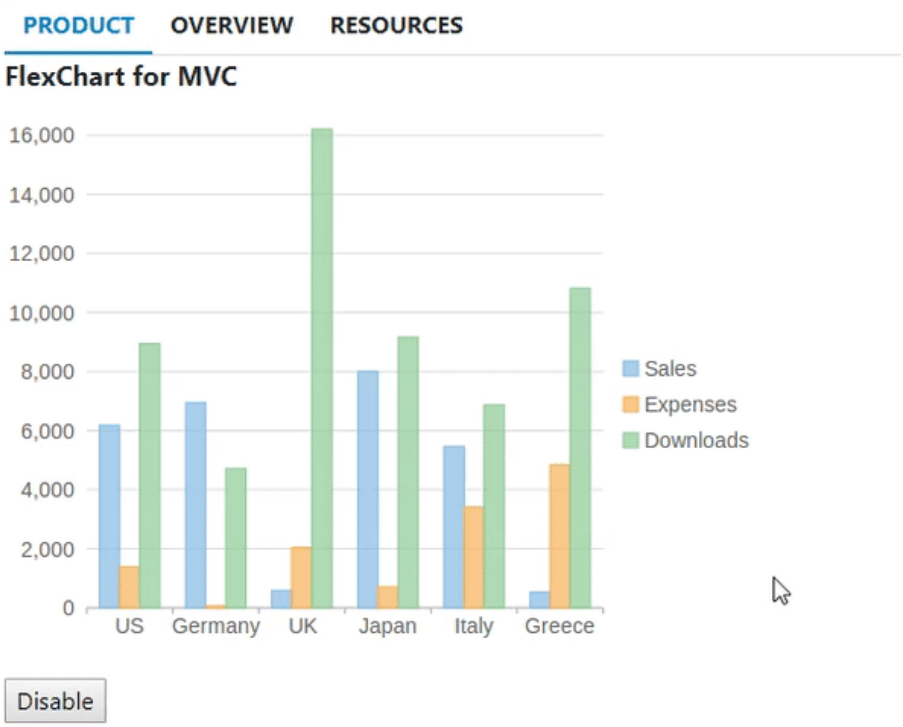
TabPanel control allows the user to disable or hide individual tabs using the **isDisabled** and **isVisible** properties respectively.

This topic includes the following sections.

- **Disabled Tabs**
- **Invisible Tabs**

Disabled Tabs

The following GIF image shows how TabPanel appears after setting the **isDisabled** property to true.



The following code example demonstrates how to disable any particular tab in the TabPanel control.

Index.cshtml

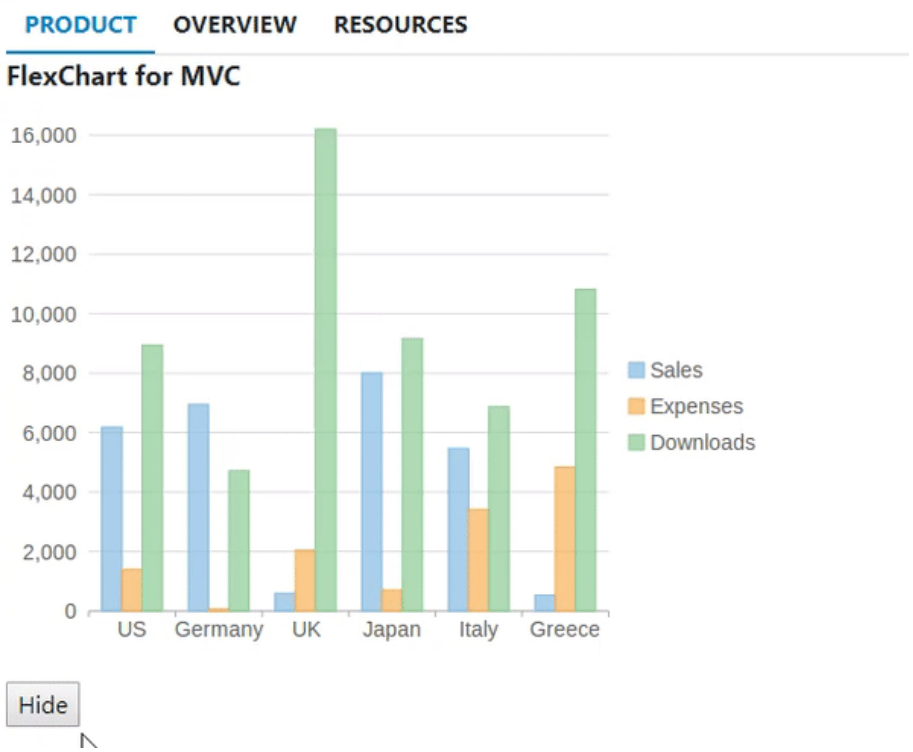
```
<script>
    function customChangeDisableResources(control) {
        var control = wijmo.Control.getControl("#tabPanel");
        control.getTab('tab-resources').isDisabled = true;
    }
</script>
<br />
<div id="tabPanel" style="width:500px">
    <div>
        <a>Product</a>
        <div>
            <p style="font-size:large"><b>FlexChart for MVC</b></p>
            <img src="~/flexchart.png" />
        </div>
    </div>
    <div>
        <a>Overview</a>
        <div>
        <p>
        <b>FlexChart</b>—a powerful data visualization control for Web—lets you add feature-rich and
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        visually appealing charts to your MVC applications.     The control empowers end-users to visualize data
that
        resonates with their audiences. The FlexChart control provides you with numerous 2D chart types,
        built-in tools for chart interactivity, and diversified formats for chart rendering. Whether it is
        storytelling with data or interpreting complex data, FlexChart helps you accomplish everything
seamlessly.
        </p>
        </div>
    </div>
    <div>
        <a id="tab-resources" >Resources</a>
        <div>
            <ul>
                <li><a
href="http://help.grapecity.com/componentone/NetHelp/FlexChartWin/webframe.html#FlexChart.html">Documentation</a>
</li>
                <li><a href="https://www.grapecity.com/en/demos">Demo</a></li>
                <li><a href="https://www.grapecity.com/en/forums">Forums</a></li>
            </ul>
        </div>
    </div>
</div><br />
<input type="button" onclick="customChangeDisableResources()" value="Disable"/>
@(Html.C1().TabPanel("#tabPanel"))
```

Invisible Tabs

The following GIF image shows how TabPanel appears after setting the **isVisible** property to false.



The following code example demonstrates how to hide any particular tab in the TabPanel control.

Index.cshtml

```
<script>
    function customChangeHideResources(control) {
        var control = wijmo.Control.getControl("#tabPanel");
        control.getTab('tab-resources').isVisible = false;
    }
</script>
<br />
<div id="tabPanel" style="width:500px">
    <div>
```

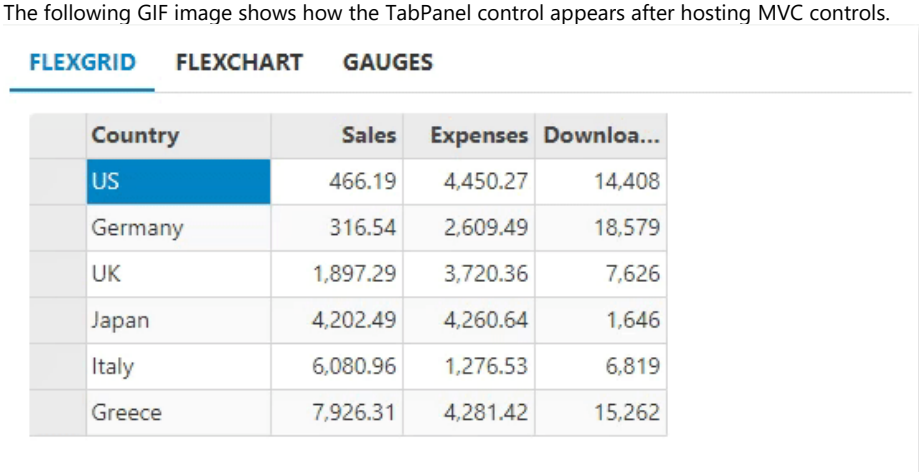

Copyright © GrapeCity, Inc. All rights reserved.

```
        <a>Product</a>
        <div>
            <p style="font-size:large"><b>FlexChart for MVC</b></p>
            <img src="~/flexchart.png" />
        </div>
    </div>
    <div>
        <a>Overview</a>
        <div>
            <p>                <b>FlexChart</b>—a powerful data visualization control for Web—lets you add
feature-rich and visually appealing charts to your MVC applications.                The control empowers end-
users to visualize data that resonates with their audiences.                The FlexChart control provides you
with numerous 2D chart types, built-in tools for chart interactivity, and diversified formats for chart
rendering.                Whether it is storytelling with data or interpreting complex data, FlexChart helps you
accomplish everything seamlessly.            </p>
        </div>
    </div>
    <div>
        <a id="tab-resources">Resources</a>
        <div>
            <ul>
                <li><a
href="http://help.grapecity.com/componentone/NetHelp/FlexChartWin/webframe.html#FlexChart.html">Documentation</a>
</li>
                <li><a href="https://www.grapecity.com/en/demos">Demo</a></li>
                <li><a href="https://www.grapecity.com/en/forums">Forums</a></li>
            </ul>
        </div>
    </div>
</div><br />
<input type="button" onclick="customChangeHideResources()" value="Hide" />
@(Html.C1().TabPanel("#tabPanel")))
```

MVC コントロールのホスティング

Hosting MVC controls in TabPanel, that is, rendering a control in its content pane is generally useful when you want to present the data using different visualization tools like grid, chart, gauge etc. The content pane of the TabPanel control is designed to render not only the html content but also the MVC controls. Also, it fetches and updates the data in control as soon as the user switches to a tab containing that control.

The following GIF image shows how the TabPanel control appears after hosting MVC controls.



The following code example demonstrates how to host multiple MVC controls in TabPanel.

Fruit.cs

```
public class Fruit
    {
        public string Name { get; set; }

        public int MarPrice { get; set; }
        public int AprPrice { get; set; }
        public int MayPrice { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    private IEnumerable<FruitSale> _sales = null;
    public IEnumerable<FruitSale> Sales
    {
        get
        {
            if (_sales == null)
            {
                _sales = GetSales();
            }
            return _sales;
        }
    }

    public static IEnumerable<Fruit> GetFruitsSales()
    {
        var rand = new Random(0);
        var fruits = new[] { "Oranges", "Apples", "Pears", "Bananas", "Pineapples" };
        var list = fruits.Select((f, i) =>
        {
            int mar = rand.Next(1, 6);
            int apr = rand.Next(1, 9);
            int may = rand.Next(1, 6);
            return new Fruit { Name = f, MarPrice = mar, AprPrice = apr, MayPrice = may };
        });

        return list;
    }

    private IEnumerable<FruitSale> GetSales()
    {
        var rand = new Random(0);
        var today = DateTime.Now.Date;
        var firstDay = new DateTime(today.Year - 1, 3, 1);
        var dataTimes = new List<DateTime>();
        for (int i = 0; i < 92; i++)
        {
            dataTimes.Add(firstDay.AddDays(i+1));
        }
        var list = dataTimes.Select((date, i) => {
            FruitSale sale = new FruitSale { Date = date };
            sale.SalesInChina = rand.Next(150, 250);
            if (i % 30 != 0)
            {
                sale.SalesInUSA = rand.Next(100, 200);
                sale.SalesInJapan = rand.Next(0, 100);
            }
            else
            {
                sale.SalesInUSA = null;
                sale.SalesInJapan = null;
            }

            return sale;
        });

        return list;
    }
}

public class FruitSale
{
    public DateTime Date { get; set; }
    public int? SalesInUSA { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public int? SalesInChina { get; set; }
        public int? SalesInJapan { get; set; }
    }
```

Sale.cs

```
public class Sale
    {
        public int ID { get; set; }
        public DateTime Start { get; set; }
        public DateTime End { get; set; }
        public string Country { get; set; }
        public string Product { get; set; }
        public string Color { get; set; }
        public double Amount { get; set; }
        public double Amount2 { get; set; }
        public double Discount { get; set; }
        public bool Active { get; set; }

        public MonthData[] Trends { get; set; }
        public int Rank { get; set; }

        private static List<string> COUNTRIES = new List<string> { "US", "UK", "Canada", "Japan",
"China", "France", "German", "Italy", "Korea", "Australia" };
        private static List<string> PRODUCTS = new List<string> { "Widget", "Gadget", "Doohickey" };
        public static IEnumerable<Sale> GetData(int total)
        {
            var colors = new[] { "Black", "White", "Red", "Green", "Blue" };
            var rand = new Random(0);
            var dt = DateTime.Now;
            var list = Enumerable.Range(0, total).Select(i =>
            {
                var country = COUNTRIES[rand.Next(0, COUNTRIES.Count - 1)];
                var product = PRODUCTS[rand.Next(0, PRODUCTS.Count - 1)];
                var color = colors[rand.Next(0, colors.Length - 1)];
                var startDate = new DateTime(dt.Year, i % 12 + 1, 25);
                var endDate = new DateTime(dt.Year, i % 12 + 1, 25, i % 24, (i % 2) * 30, 0);

                return new Sale
                {
                    ID = i + 1,
                    Start = startDate,
                    End = endDate,
                    Country = country,
                    Product = product,
                    Color = color,
                    Amount = Math.Round(rand.NextDouble() * 10000 - 5000, 2),
                    Amount2 = Math.Round(rand.NextDouble() * 5000, 2),
                    Discount = Math.Round(rand.NextDouble() / 4, 2),
                    Active = (i % 4 == 0),
                    Trends = Enumerable.Range(0, 12).Select(x => new MonthData { Month = x + 1, Data =
rand.Next(0, 100) }).ToArray(),
                    Rank = rand.Next(1, 6)
                };
            });
            return list;
        }
    }

    public class MonthData
    {
        public int Month { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public double Data { get; set; }
    }
```

HostControlsController

```
public class HostControlsController : Controller
    {
        // GET: HostControls
        public ActionResult Index()
        {
            ViewBag.GridData = Sale.GetData(20);
            ViewBag.ChartData = Fruit.GetFruitsSales();
            return View();
        }
    }
```

Index.cshtml

Index.cshtml

```
@{
    var gridData = ViewBag.GridData as IEnumerable<Sale>;
    var chartData = ViewBag.ChartData as IEnumerable<Fruit>;
}
<div id="tabGettingStarted">
    <div>
        <a>FlexGrid</a>
        <div>
            @(Html.C1().FlexGrid<Sale>().Id("theGrid").CssClass("grid")
                .AutoGenerateColumns(false).IsReadOnly(true)
                .Bind(gridData)
                .Columns(cs =>
                {
                    cs.Add().Binding("ID");
                    cs.Add().Binding("Country");
                    cs.Add().Binding("Product");
                    cs.Add().Binding("Color");
                    cs.Add().Binding("Amount");
                }))
        </div>
    </div>
    <div>
        <a>FlexChart</a>
        <div>
            @(Html.C1().FlexChart().Id("theChart")
                .Bind(chartData).BindingX("Name")
                .ChartType(C1.Web.Mvc.Chart.ChartType.Column)
                .Series(ss =>
                {
                    ss.Add().Binding("MarPrice");
                    ss.Add().Binding("AprPrice");
                    ss.Add().Binding("MayPrice");
                }))
        </div>
    </div>
    <div>
        <a>Gauges</a>
        <div style="overflow:hidden">

@Html.C1().RadialGauge().Id("theRadialGauge").Min(0).Max(10).Value(5).ShowText(ShowText.None).Width(300)
            @(Html.C1().LinearGauge().Id("theLinearGauge").Width(300)
                .Face(f => f.Min(0).Max(10))
                .Pointer(p=>p.Max(5))
            )
        </div>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    </div>
</div>
@Html.C1().TabPanel("#tabGettingStarted")
```

右から左

Right to left is an HTML feature that causes the content to flow from right to left of the page. This feature is generally used to render right to left languages (Arabic, Hebrew etc.) by setting the 'dir' attribute of HTML elements to 'rtl'. The TabPanel control automatically supports this HTML feature and renders the TabPanel in right-to-left direction when 'dir' attribute of any of the parent elements is set to 'rtl'.

For example, if the element hosting the control has the 'dir' attribute set to 'rtl', the TabPanel control renders with the tabs aligned from right to left.

The following GIF image shows how the TabPanel control appears with direction set from right to left.



The following code example demonstrates how to show content from right to left in the TabPanel control.

Index.cshtml

```
<div id="tabPanel" dir="rtl" style="width:500px">
    <div>
        <a>Product</a>
        <div>
            <p style="font-size:large"><b>FlexChart for MVC</b></p>
            <img src="~/flexchart.png" />
        </div>
    </div>
    <div>
        <a>Overview</a>
        <div>
            <p>
        <b>FlexChart</b>—a powerful data visualization control for Web—lets you add feature-rich and
        visually appealing charts to your MVC applications.     The control empowers end-users to visualize data
that
        resonates with their audiences. The FlexChart control provides you with numerous 2D chart types,
        built-in tools for chart interactivity, and diversified formats for chart rendering. Whether it is
        storytelling with data or interpreting complex data, FlexChart helps you accomplish everything
seamlessly.
        </p>
        </div>
    </div>
    <div>
        <a>Resources</a>
        <div>
            <ul>
                <li><a
href="http://help.grapecity.com/componentone/NetHelp/FlexChartWin/webframe.html#FlexChart.html">Documentation</a>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
</li>
                <li><a href="https://www.grapecity.com/en/demos">Demo</a></li>
                <li><a href="https://www.grapecity.com/en/forums">Forums</a></li>
            </ul>
        </div>
    </div>
</div>
@(Html.C1().TabPanel("#tabPanel"))
```

スタイル設定とCSS

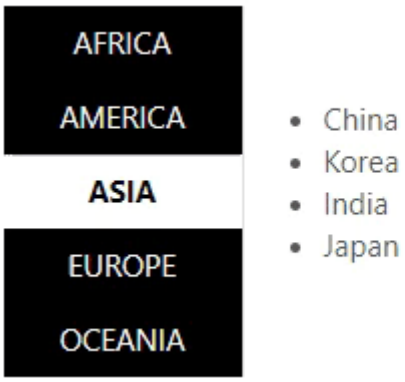
CSS classes for the TabPanel control are as follows:

Class	Description
.wj-tabpanel	Styles the div host element of the TabPanel control.
.wj-tabpanel > div > .wj-tabheaders	Styles all the tab headers.
.wj-tabpanel > div > .wj-tabheaders > .wj-header	Styles the individual tab header.
.wj-tabheader.wj-state-active	Styles the currently active tab header in the TabPanel control.
.wj-tabpane .wj-state-active	Styles the currently active tab pane in the TabPanel control.
.wj-tabpanel > div > .wj-tabpanes	Styles all the tab panes.
.wj-tabpanel > div > .wj-tabpanes > .wj-tabpane	Styles the individual tab pane.

スタイル設定

The TabPanel control allows the users to customize the appearance of the TabPanel control using CSS. The TabPanel control has a simple layout that makes it easy to style it using CSS. For example, tabs appear above the content by default, however, you can use CSS to change their position and show them below or vertically stacked, to the left or to the right of the content. Moreover, it also allows you to customize the tab headers.

The following GIF image shows how TabPanel control appears with custom styling.



The following code example demonstrates how to show custom styling in the TabPanel control.

Index.cshtml

```
@section Styles{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    <style>
        /* カスタムヘッダー */
        .wj-tabpanel.custom-headers .wj-tabpanes {
            border-top: none;
        }

        .wj-tabpanel.custom-headers .wj-tabheaders {
            background: black;
            color: white;
        }

            .wj-tabpanel.custom-headers .wj-tabheaders .wj-tabheader.wj-state-active
{
                background: white;
                color: black;
            }

            .wj-tabpanel.custom-headers .wj-tabheaders .wj-tabheader:not(.wj-state-
active) {
                font-weight: normal;
            }

            .wj-tabpanel.custom-headers .wj-tabheaders .wj-tabheader.wj-state-
active:after { /* hide underline */
                display: none;
            }

        /* 左側のタブ*/
        .wj-tabpanel.tabs-on-left > div {
            display: flex;
            align-items: center;
        }

        .wj-tabpanel.tabs-on-left .wj-tabheaders {
            display: flex;
            flex-direction: column;
            min-width: 8em;
            border-right: 1px solid #ddd;
        }

        .wj-tabpanel.tabs-on-left .wj-tabpanes {
            display: flex;
            flex-grow: 1;
            border-top: none;
            overflow-x: hidden;
        }
    </style>
}

<div id="tabPanel" class="custom-tab">
    <div>
        <a>Africa</a>
        <div>
```


Copyright © GrapeCity, Inc. All rights reserved.

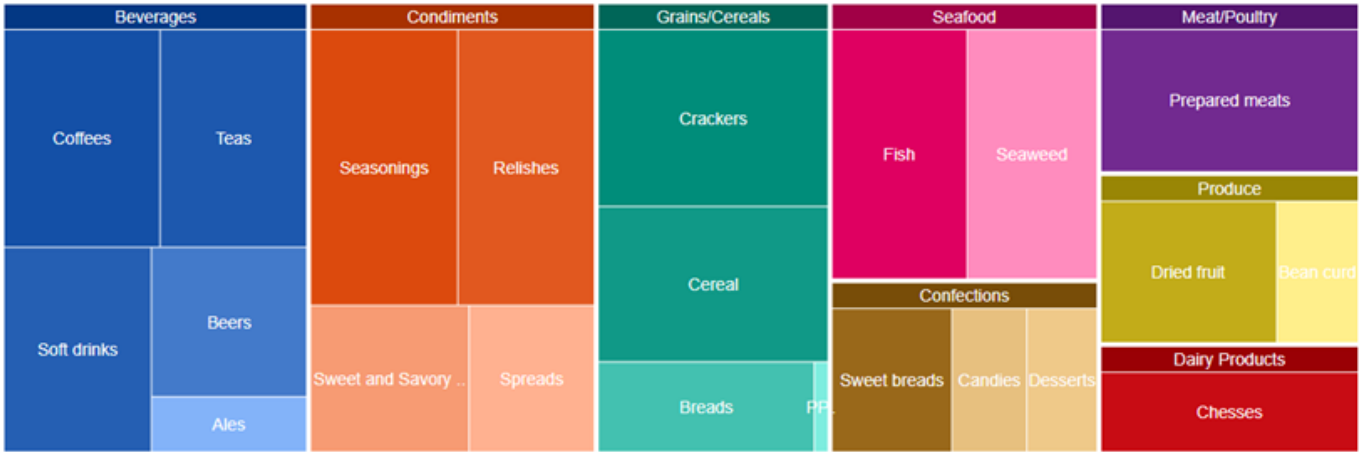
```
        <ul>
            <li>Algeria</li>
            <li>Angola</li>
            <li>Benin</li>
            <li>Botswana</li>
        </ul>
    </div>
</div>
<div>
    <a>America</a>
    <div>
        <ul>
            <li>Canada</li>
            <li>Chile</li>
            <li>Mexico</li>
            <li>United States</li>
        </ul>
    </div>
</div>
<div>
    <a>Asia</a>
    <div>
        <ul>
            <li>China</li>
            <li>Korea</li>
            <li>India</li>
            <li>Japan</li>
        </ul>
    </div>
</div>
<div>
    <a>Europe</a>
    <div>
        <ul>
            <li>Austria</li>
            <li>England</li>
            <li>France</li>
            <li>Germany</li>
            <li>Netherlands</li>
            <li>Switzerland</li>
        </ul>
    </div>
</div>
<div>
    <a>Oceania</a>
    <div>
        <ul>
            <li>Australia</li>
            <li>Fiji</li>
            <li>New Zealand</li>
            <li>Samoa</li>
        </ul>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        </div>
    </div>
</div>
@Html.C1().TabPanel("#tabPanel")
```

TreeMap

TreeMapコントロールは、階層化データを入れ子になった四角形で表示します。階層化データは、家系図、プログラミング、組織図、ディレクトリなどのさまざまな階層や組織に使用されます。このようなデータスポット情報を視覚化することは、特にデータが大きな場合は難しいタスクです。TreeMapコントロールを使用すると、階層化データを限られたスペースで入れ子の四角形として視覚化できます。大きなデータに含まれるパターンを簡単に把握したり、比率を比較するために便利です。



TreeMapコントロールは、階層を表示するためのデータ連結をサポートし、データをさまざまなレベルにドリルダウンして詳細に分析できるようにします。このコントロールは、構成している四角形を横長、縦長、正方形のレイアウトでデータを表示するようにカスタマイズできます。

主な特長

TreeMapには、直感的で本格的な外観のアプリケーションを構築するためのさまざまな機能が提供されています。TreeViewの主な機能は、次のとおりです。

- **マルチレイアウト**
  マルチレイアウトTreeMapは、複数の表示配置をサポートします。ツリーブランチを正方形、横長長方形、縦長長方形などで示すことができます。

- **外観のカスタマイズ**
  TreeMapでは、コントロールをスタイル設定し、好みに応じて外観を変更することができます。TreeMapでカテゴリをわかりやすく表示するために、さまざまなカラーパレットを使用できます。

- **カスタム階層レベル**
  TreeMapでは、データの深さを変えて視覚化したり、データをドリルダウン（または逆ドリルダウン）することで、分析や比較を行うことができます。

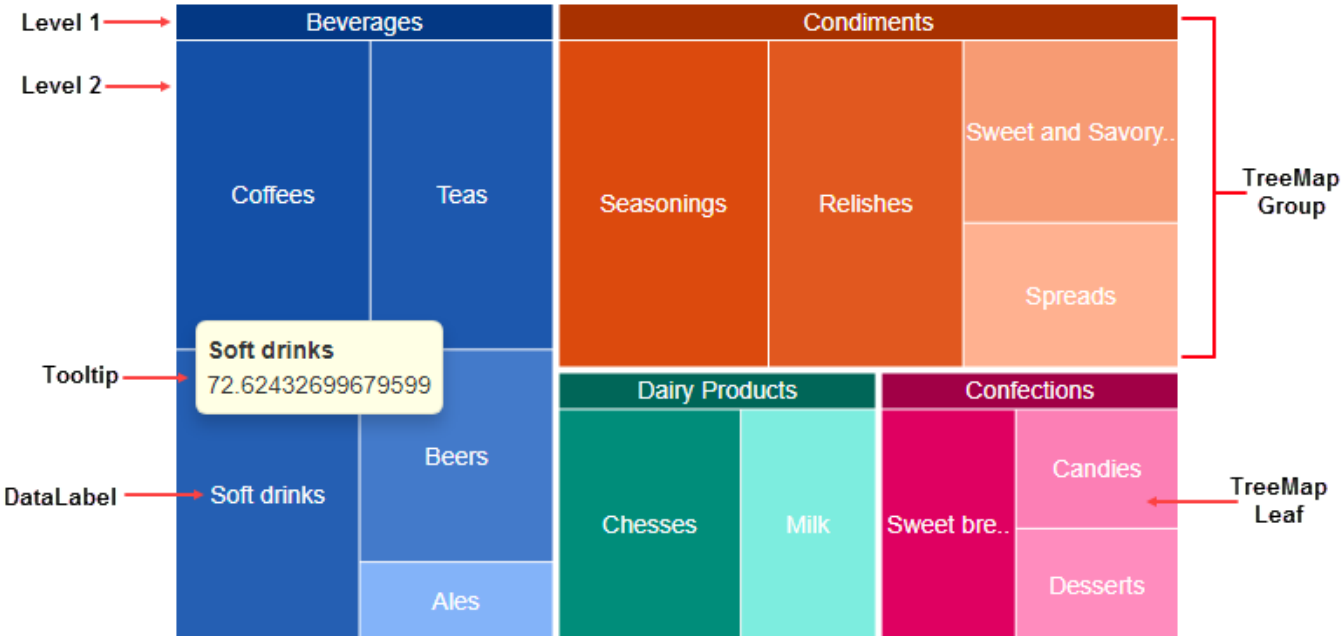
- **スペース活用**
  TreeMapは、大量のデータをコンパクトに表示して視覚化する場合に理想的です。TreeMapを構成する入れ子の四角形やグループは、表示領域に合わせてサイズが調整されます。

要素

Copyright © GrapeCity, Inc. All rights reserved.

TreeMapコントロールは、個別のデータ項目を表す長方形で構成され、データの階層的な性質を表すカテゴリにグループ化されます。グループを構成する個別のデータ項目は、リーフノードと呼ばれます。ノードのサイズは、表すデータに比例します。

次の図に、TreeMapコントロールのメイン要素を示します。



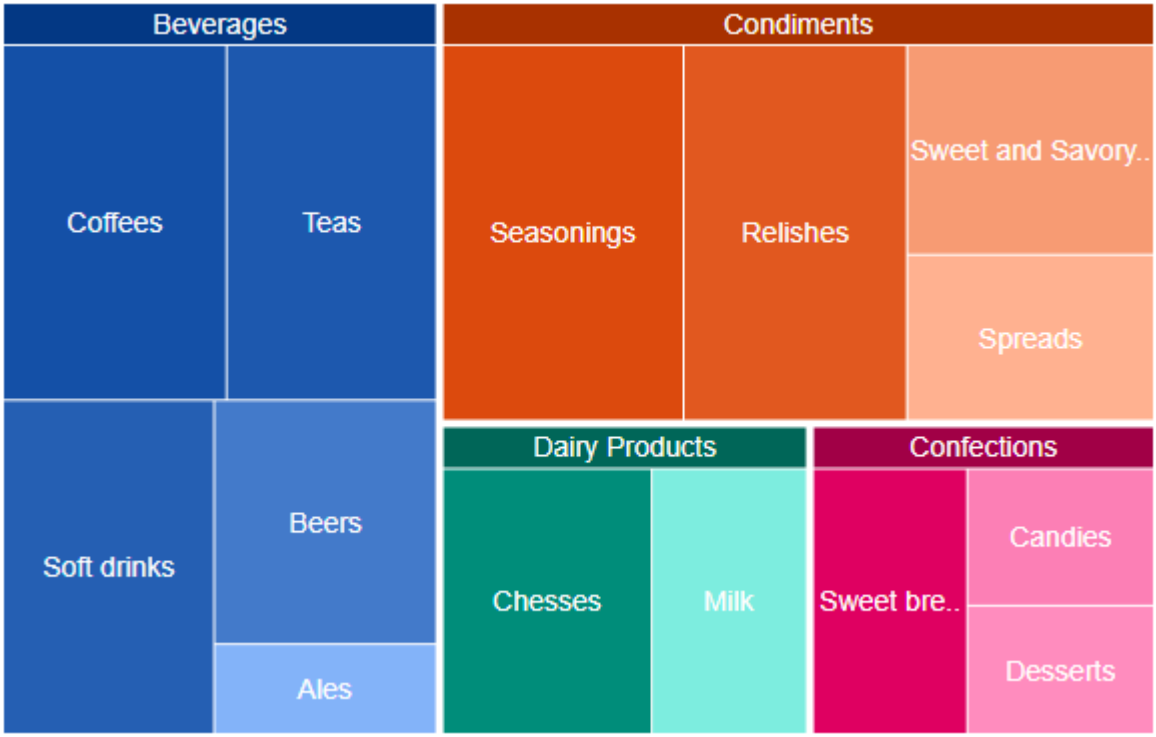
レイアウト

TreeMapでは、データ項目とグループを四角形で表し、さまざまな配置で表示できます。ツリーマップの四角形は、正方形、横長、縦長のレイアウトに配置できます。目的のツリーマップレイアウトを設定するには、**TreeMap**クラスの**Type**プロパティを使用する必要があります。このクラスは、**TreeMapType**列挙の値を受け取ります。TreeMapチャートコントロールのデフォルトのレイアウトは、正方形です。

正方形

正方形のレイアウトは、ツリーマップの四角形（データ項目とグループ）をほぼ正方形として配置します。このレイアウトでは、正方形の配置によって表現の正確さが向上するため、比較やパターンの把握が容易になります。このレイアウトは、大きなデータセットでたいへん便利です。

Copyright © GrapeCity, Inc. All rights reserved.



水平方向

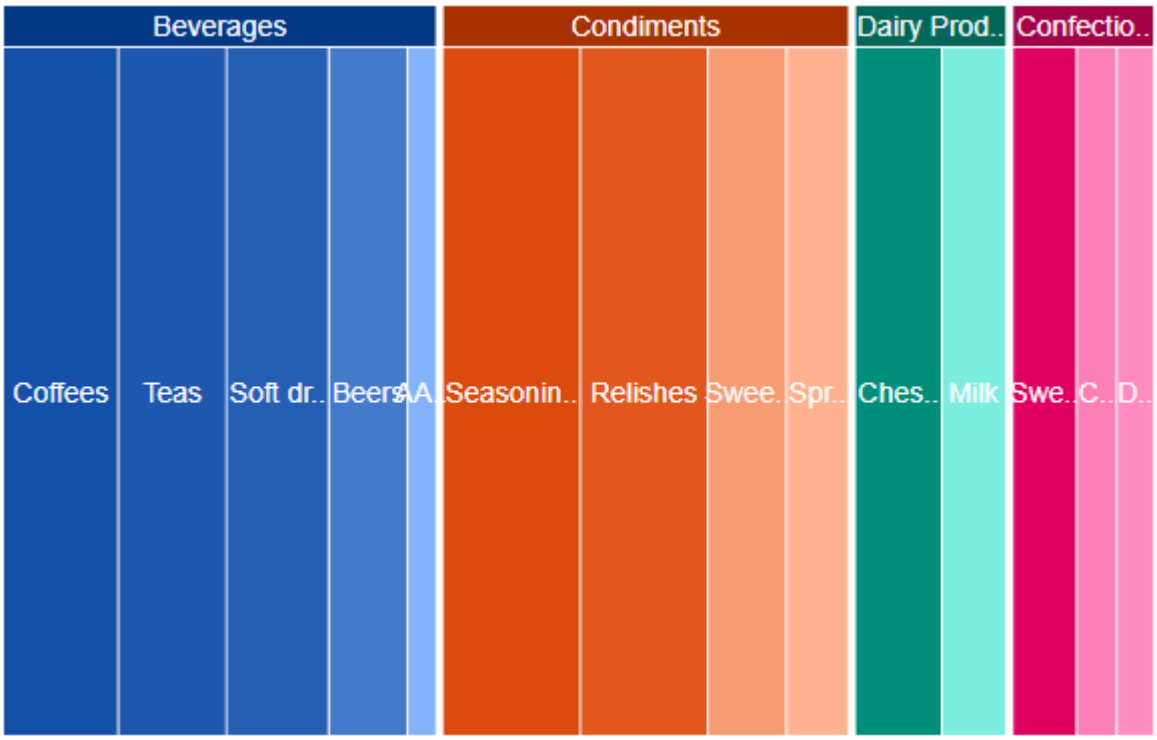
水平方向のレイアウトは、ツリーマップの四角形を積み重ねて行の形式にします。四角形の幅が高さより大きくなります。



垂直方向

垂直方向のレイアウトでは、ツリーマップの四角形を隣接させて列の形式にします。四角形の高さが幅より大きくなります。

Copyright © GrapeCity, Inc. All rights reserved.

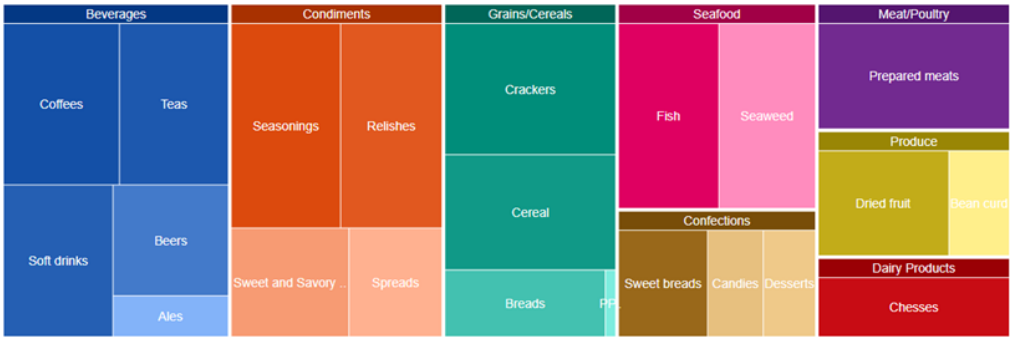


クイックスタート:TreeMapへのデータの追加

このトピックでは、MVC WebアプリケーションにTreeMapコントロールを追加し、そこにモデル連結を使用してデータを追加する方法について説明します。このトピックは次の4つの手順で構成されます。

- **手順1:MVCアプリケーションの作成**
- **手順2:TreeMapのデータソースの作成**
- **手順3:TreeMapコントロールの追加**
- **手順4:プロジェクトのビルドおよび実行**

次の図に、上の手順を実行した後のTreeMapコントロールを示します。



手順1:MVCアプリケーションの作成

ComponentOneまたはVisualStudioテンプレートを使用して、新しいMVCアプリケーションを作成します。MVCアプリケーションの作成方法については、「MVCアプリケーションの構成」を参照してください。

手順2:TreeMapのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(名前:`FoodSale.cs`)。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`FoodSale.cs`モデルに追加します。ここでは、**`FoodSale`**クラスを使用して階層化データリストを表しています。

C#

```
using System;
using System.Collections.Generic;

namespace TreeMapQuickStart.Models
{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    public class FoodSale
    {
        private static List<string> Categories =

                    _new List<string> { "飲料", "調味料", "菓子類", "乳製品"  };
        private static List<string[]> SubCategories = new List<string[]>
        {
            new string[] { "ソフトドリンク", "コーヒー", "お茶", "ビール", "アレス" },
            new string[] { "甘くて味の良いソース", "前菜", "スプレッド", "調味料" },
            new string[] { "デザート", "キャンディー", "パン" },
            new string[] { "チーズ", "牛乳" },
        };
        public string Category { get; set; }
        public string SubCategory { get; set; }
        public double Sales { get; set; }
        public static IEnumerable<FoodSale> GetData()
        {
            var result = new List<FoodSale>();
            var rand = new Random(0);
            var index = 0;
            foreach (var cat in Categories)
            {
                var subCategories = SubCategories[index++];
                foreach (var subCat in subCategories)
                {
                    result.Add(new FoodSale
                    {
                        Category = cat,
                        SubCategory = subCat,
                        Sales = rand.NextDouble() * 100
                    });
                }
            }
            return result;
        }
        public static IEnumerable<FoodSale> GetGroupData()
        {
            var result = new List<FoodSale>();
            var rand = new Random(0);
            var catLen = Categories.Count;
            for (var i = 0; i < 1000; i++)
            {
                var randomC = rand.Next(0, catLen - 1);
                var subCat = SubCategories[randomC];
                var randomSC = rand.Next(0, subCat.Length - 1);
                result.Add(new FoodSale
                {
                    Category = Categories[randomC],
                    SubCategory = subCat[randomSC],
                    Sales = rand.NextDouble() * 100
                });
            }
            return result;
        }
    }
}
```

先頭に戻る

手順3：TreeMapコントロールの追加

アプリケーションにTreeMapコントロールを追加する手順は、次のとおりです。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`TreeMapController`）。
    3. **[追加]**をクリックします。
4. 次に示すように参照を追加します。

Copyright © GrapeCity, Inc. All rights reserved.

C#

```
using <ApplicationName>.Models;
```

5. **Index()**メソッドを次のメソッドに置き換えます。

TreeMapController.cs

```
public ActionResult Index()
{
   return View(FoodSale.GetData());
}
```

コントローラーのビューの追加

ビュー内で、TreeMapのインスタンスを作成し、それを**Bind**メソッドを使用してデータソースに連結します。**Type**プロパティを使用して、アプリケーションで使用するTreeMapのタイプを選択できます。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`TreeMapController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加したら、次のコードをコピーして**Index.cshtml**内に貼り付けます。

Index.cshtml

```
@using <ApplicationName>.Models;
@using System.Drawing;
@model IEnumerable<FoodSale>

@(Html.C1().TreeMap().Id("TreeMap")
    .Type(TreeMapType.Squarified)
    .Binding("Sales")
    .BindingName("Category", "SubCategory")
    .Bind(Model)
    .DataLabel(dlb => dlb.Position

    _(C1.Web.Mvc.Chart.LabelPosition.Center).Content("{name}")))
```

手順4:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http://localhost:1234/**TreeMap/Index**)。

先頭に戻る

TreeMapの使用

TreeMapコントロールの使用方法を理解するために役立つ概念について説明します。

このセクションには、次の情報が含まれます。

最大深さ
: TreeViewコントロールのMaxDepthプロパティを設定する方法について説明します。

グループコレクション
: TreeMapコントロールでグループ化を実装する方法について説明します。

テーマ
: TreeMapコントロールでさまざまなテーマを適用する方法について説明します。

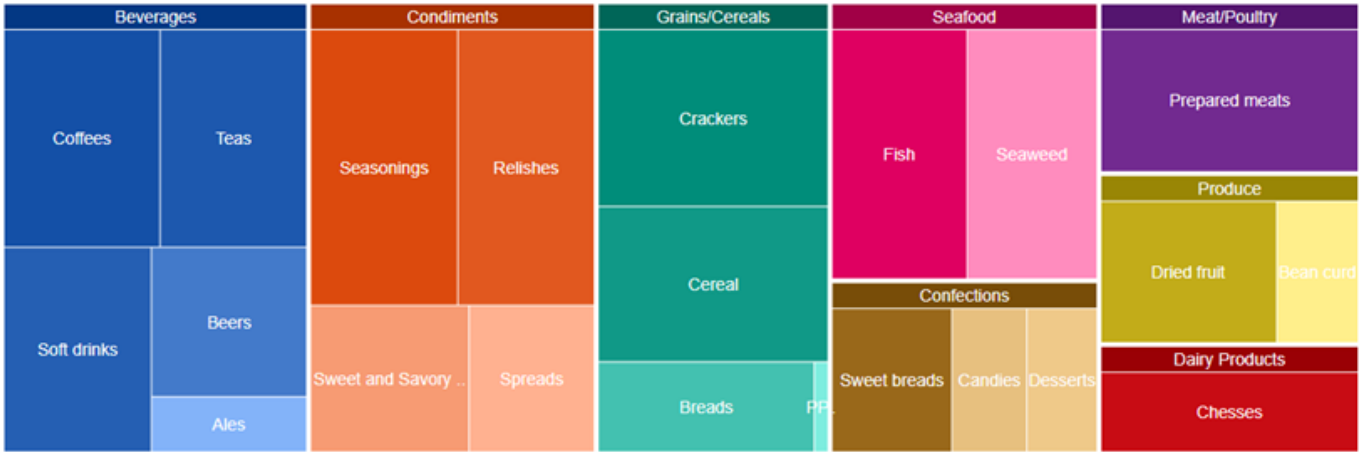
グループコレクション

TreeMapコントロールを使用すると、データをグループ化してコンテンツの表示を向上させることができます。TreeMapでグループを構成するには、ビューの**GroupBy**プロパティを使用します。グループ化されたTreeMap項目は、ある値でグループ化されたツリーマップのリーフ項目のコレクションを表します。GroupHeaderを使用すると、TreeMapコントロール内でリーフ項目

Copyright © GrapeCity, Inc. All rights reserved.

がどの値によってグループ化されているかを識別することができます。

次の図は、**GroupBy**プロパティを設定した後のTreeMapコントロールを示しています。



以下のコード例では、「クイックスタート」セクションで追加したFoodSale.csモデルを使用しています。

コードの場合

Grouping.cshtml

Index.cshtml

```
@using <ApplicationName>.Models;
@using System.Drawing;
@model IEnumerable<FoodSale>

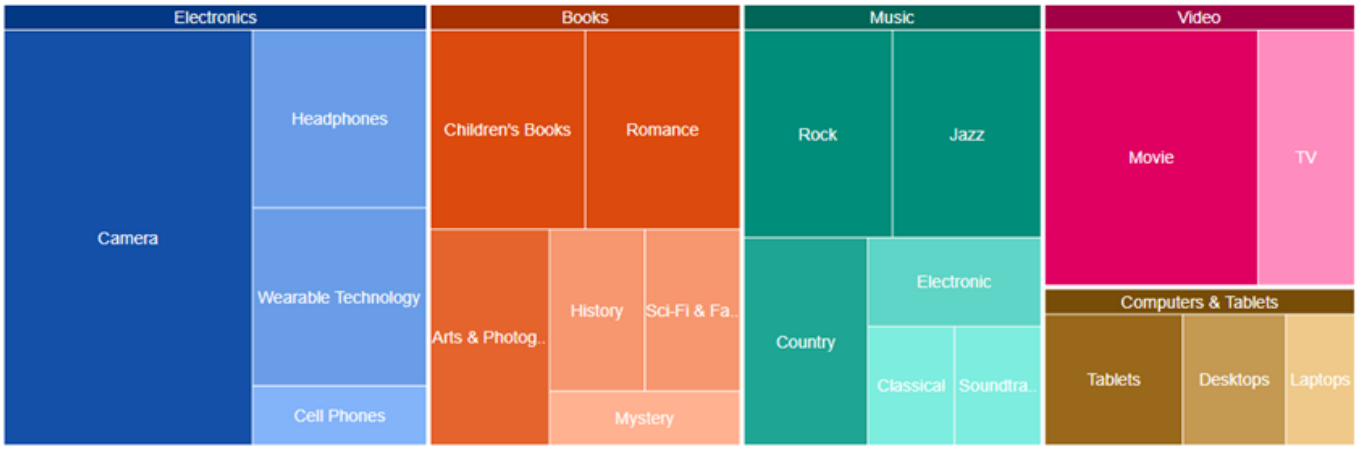
@(Html.C1().TreeMap()
    .Binding("Sales")
    .BindingName("Category", "SubCategory")
    .Bind(isb => isb.Bind(Model).GroupBy("Category", "SubCategory"))
    .DataLabel(dlb => dlb.Position(C1.Web.Mvc.Chart.LabelPosition.Center).Content("
{name}")))
```

MaxDepth

TreeMapコントロールでは、コントロールに同時に表示するレベルの数を定義することができます。必要な数の階層レベルを表示することができます。表示するレベルの数を引数として**MaxDepth**メソッドを使用します。表示するレベルの数が大きいと、TreeMapがわかりにくくなる可能性があります（レベルの数とそれらが表す値による）。この例では、MaxDepthプロパティを2 に設定します。

次の図は、MaxDepthプロパティを2に設定した後のTreeMapコントロールを示しています。

Copyright © GrapeCity, Inc. All rights reserved.



次のコード例は、TreeMapでMaxDepthプロパティを設定する方法を示します。この例では、**ThingSale.cs**モデルを使用します。

コードの場合

ThingSale.cs

C#

```
public class ThingSale
    {
        private static List<string> Categories = new List<string> { "Music", "Video",
"Books", "Electronics", "Computers & Tablets" };
        private static Dictionary<string, List<string>> AllCategories = new
Dictionary<string, List<string>>();
        public string Category { get; set; }

        public double? Sales { get; set; }
        public List<ThingSale> Items { get; set; }

        public static void EnsureInitAllCategories()
        {
            if (AllCategories.Count > 0)
            {
                return;
            }

            AllCategories.Add("音楽の種類", new List<string> { "カントリー", "ロック", "クラ
シック", "サウンドトラック", "ジャズ", "エレクトロニック" });
            AllCategories.Add("カントリー", new List<string> { "クラシックカントリー", "カウボー
イカントリー", "アウトローカントリー", "ウェスタンスイング", "ロードハウスカントリー" });
            AllCategories.Add("ロック", new List<string> { "ハードロック", "ブルースロック",
"ファンクロック", "ラップロック", "ギターロック", "プログレッシブロック" });
            AllCategories.Add("クラシック", new List<string> { "交響曲", "室内楽" });
            AllCategories.Add("サウンドトラック", new List<string> { "映画サウンドトラック",
"ミュージカルサウンドトラック" });
            AllCategories.Add("ジャズ", new List<string> { "スムーズジャズ", "ボーカルジャ
ズ", "ジャズフュージョン", "スイングジャズ", "クールジャズ", "トラッドジャズ" });
            AllCategories.Add("エレクトロニック", new List<string> { "エレクトロニカ", "ディス
コ", "ハウスミュージック" });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            AllCategories.Add("ビデオ", new List<string> { "映画", "テレビ" });
            AllCategories.Add("映画", new List<string> { "児童", "アクション", "アニメーション", "コメディ", "連続活劇", "恋愛" });
            AllCategories.Add("テレビ", new List<string> { "サイエンス・フィクション", "ドキュメンタリー", "ファンタジー", "戦争", "ホラー" });

            AllCategories.Add("本", new List<string> { "アート・アンド・フォタグラフィ", "児童書", "歴史", "ミステリー", "ロマンス", "SF＆ファンタジー" });
            AllCategories.Add("アート・アンド・フォタグラフィ", new List<string> { "建築", "グラフィックデザイン", "ドローイング", "フォタグラフィ", "舞台芸術" });
            AllCategories.Add("児童書", new List<string> { "学習図鑑", "童話", "章本", "塗り絵の本", "絵本", "学習まんが" });
            AllCategories.Add("歴史", new List<string> { "古代史", "中世史", "ルネサンス" });
            AllCategories.Add("ミステリー", new List<string> { "スリラー＆サスペンス", "ミステリー" });
            AllCategories.Add("ロマンス", new List<string> { "アクションアドベンチャー", "ロマンチックコメディ", "ロマンチックサスペンス", "西部劇", "歴史" });
            AllCategories.Add("SF＆ファンタジー", new List<string> { "ファンタジー", "ゲーム", "サイエンスフィクション" });

            AllCategories.Add("電子デバイス", new List<string> { "カメラ", "ヘッドフォン", "携帯電話", "ウェアブルデバイス" });
            AllCategories.Add("カメラ", new List<string> { "デジタルカメラ", "フィルムカメラ", "レンズ", "ビデオ", "アクセサリー" });
            AllCategories.Add("ヘッドフォン", new List<string> { "イヤホンヘッドホン", "オーバーイヤーヘッドフォン", "Bluetoothヘッドフォン", "ノイズキャンセリングヘッドフォン", "オーディオファイルヘッドフォン" });
            AllCategories.Add("携帯電話", new List<string> { "携帯電話", "アクセサリー" });
            AllCategories.Add("アクセサリー", new List<string> { "バッテリー", "Bluetoothヘッドセット", "Bluetoothスピーカー", "充電器", "スクリーンプロテクター" });
            AllCategories.Add("ウェアブルデバイス", new List<string> { "アクティビティトラッカー", "スマートウォッチ", "スポーツとGPS対応の時計", "3D VRゴーグルヘッドセット", "ウェアラブルカメラ", "スマートグラス" });

            AllCategories.Add("コンピュータ/タブレット", new List<string> { "デスクトップ", "パソコン", "タブレット" });
            AllCategories.Add("デスクトップ", new List<string> { "オールインワンPC", "ミニPC", "タワーPC" });
            AllCategories.Add("パソコン", new List<string> { "2in1パソコン", "通常のパソコン" });
            AllCategories.Add("タブレット", new List<string> { "IOS", "Android", "Fire OS", "Windows" });
        }
        public static IEnumerable<ThingSale> GetData()
        {
            EnsureInitAllCategories();
            var result = new List<ThingSale>();
            Categories.ForEach(cat =>
            {
                result.Add(Create(cat));
            });
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            return result;
        }

        private static ThingSale Create(string category)
        {
            var rand = new Random(0);
            var item = new ThingSale { Category = category };
            if (!AllCategories.ContainsKey(category))
            {
                item.Sales = rand.NextDouble() * 100;
            }
            else
            {
                item.Items = new List<ThingSale>();
                AllCategories[category].ForEach(subCat =>
                {
                    item.Items.Add(Create(subCat));
                });
            }
            return item;
        }
    }
```

TreeMapController.cs

C#

```
public class TreeMapController : Controller
    {
        // GET: TreeMap
        public ActionResult Index()
        {
            return View(ThingSale.GetData());
        }
    }
```

MaxDepth.cshtml

Razor

```
@using <ApplicationName.Models>
@using System.Drawing;
@model IEnumerable<ThingSale>

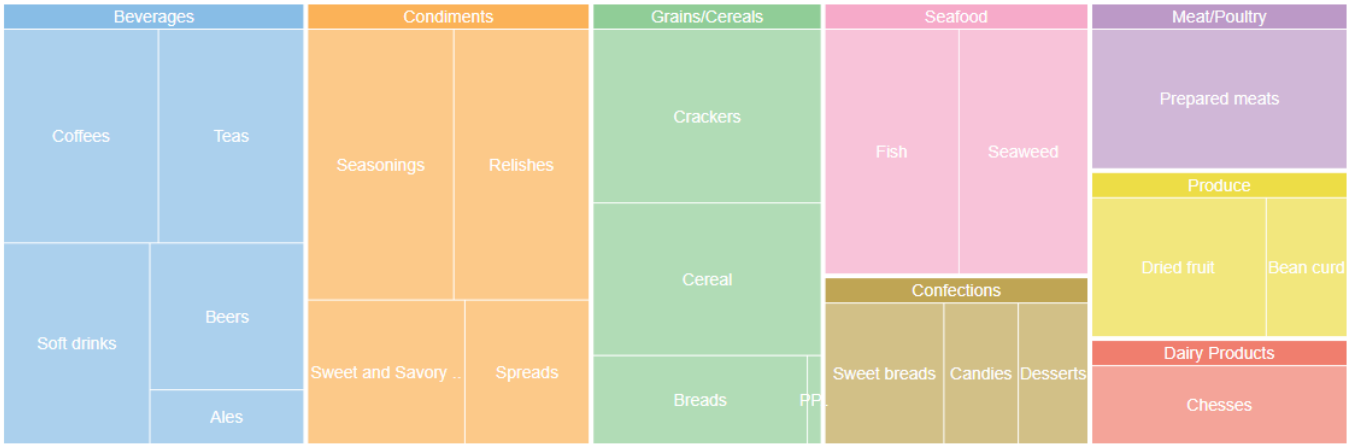
@(Html.C1().TreeMap().Id("TreeMap")
    .Binding("Sales")
    .BindingName("Category")
    .ChildItemsPath("Items")
    .Bind(Model)
    .MaxDepth(2)
    .DataLabel(dlb => dlb.Position(C1.Web.Mvc.Chart.LabelPosition.Center).Content("
{name}")))
```


Copyright © GrapeCity, Inc. All rights reserved.

テーマ

TreeMapチャートコントロールでは、**Palette**プロパティを使用して外観をカスタマイズすることができます。このプロパティにはコレクション値を設定できます。コレクション項目タイプは、**System.Drawing.Color**または**C1.Web.Mvc.TreeMapItemStyle**です。

次の図に、**Palette**プロパティを使用してテーマを適用した後のTreeMapコントロールを示します。



以下のコード例では、「クイックスタート」セクションで追加したFoodSale.csモデルを使用しています。

コードの場合

Themeing.cshtml

Index.cshtml

```
@using <ApplicationName>.Models;
@using System.Drawing;
@model IEnumerable<FoodSale>

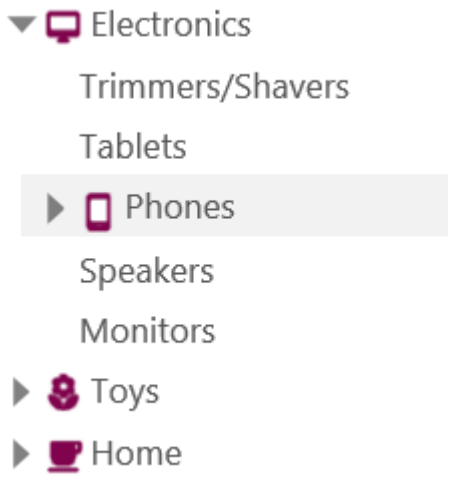
@{
    var colors = new List<string> { "#88bde6", "#fbb258", "#90cd97", "#f6aac9",
"#bfa554", "#bc99c7", "#eddd46", "#f07e6e", "#8c8c8c" };
}

@(Html.C1().TreeMap()
    .Binding("Sales")
    .BindingName("Category", "SubCategory")
    .Bind(Model)
    .Palette(colors.Select(ColorTranslator.FromHtml).ToList())
    .DataLabel(dlb => dlb.Position(C1.Web.Mvc.Chart.LabelPosition.Center).Content("
{name}")))
```

TreeView

TreeViewコントロールは、テキスト、チェックボックス、画像、または任意のHTMLコンテンツから成る階層リストを表示します。文書の見出し、索引内の項目、ディスク内のファイルやディレクトリなど、階層として表示すると便利な情報を表示するためによく使用されます。TreeViewコントロールは、選択や編集が可能なノードを使用してデータを管理します。ノードを使用して、単純なテキスト、画像のほか、チェックボックスなどの要素を表示できます。

Copyright © GrapeCity, Inc. All rights reserved.



ツリーの作成に必要なプロパティを次に示します。

- Bind/Source - 階層化データを含む配列。配列の各項目には、ノードおよび(オプションで)子ノードの配列に関する情報が含まれます。
- **DisplayMemberPath** - ツリーノードに表示されるテキストを含む、項目内のプロパティの名前を定義します。デフォルトでは、このプロパティは文字列「header」に設定されています。
- **ChildItemsPath** - 子ノードの配列を含む、項目内のプロパティの名前を定義します。デフォルトでは、このプロパティは文字列「items」に設定されています。

これらのプロパティのほかに、TreeViewはノード画像を連結するためのプロパティ、チェックボックスのプロパティ、およびitemSource配列の折りたたまれた状態のプロパティも提供します。詳細については、「機能」を参照してください。

主な特長

TreeViewには、直感的で本格的な外観のアプリケーションを構築するためのさまざまな機能が提供されています。TreeViewの主な機能は、次のとおりです。

- **ノードの展開**
  TreeViewでは、必要に応じて、ツリー内のノードを1つまたはすべて展開することができます。**TreeView**コントロールクラスには、任意のノードをクリックしたときに1つのノード(親または子)を展開するための**ExpandOnClick()**メソッドが用意されています。また、**AutoCollapse()**メソッドもあります。これは、現在選択されている親ノードの兄弟ノードを自動的に閉じます。ノードの展開の詳細については、「ノードの展開と折りたたみ」を参照してください。

- **ノードの選択とナビゲーション**
  TreeViewは、キーボードとマウスによるノード間の移動をサポートしています。また、1つまたは複数のノードの選択もサポートしています。複数のノードを連続して、または非連続に選択することができます。ナビゲーションの詳細については、「ナビゲーションノード」を参照してください。

- **遅延ロード**
  TreeViewでは、ノードの子項目が実際に必要になるまで、子項目のロードを延期することができます。これは、多数のレベルと子ノードから成る極めて深いツリーがある場合に特に役立ちます。Windowsコンピュータのフォルダ構成が典型的な例です。

- **カスタムノード**
  TreeViewでは、カスタムノードを設定することができます。このようなノードで、単純なテキスト、画像アイコン、複数のアイコン、チェックボックスなどを使用することができます。TreeViewノードのコンテンツは、**formatItem**イベントを使用してカスタマイズすることができます。イベントハンドラのパラメータには、ノードとレンダリングされるデータ項目を表す要素が含まれます。カスタムノードのコンテンツの詳細については、「カスタムノード」を参照してください。

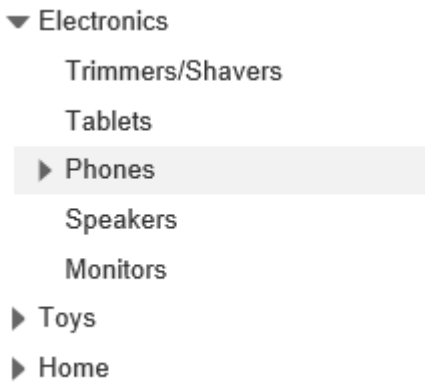
Copyright © GrapeCity, Inc. All rights reserved.

クイックスタート:TreeView へのデータの追加

このトピックでは、MVC WebアプリケーションにMultiRowコントロールを追加し、そこにモデル連結を使用してデータを追加する方法について説明します。このトピックは次の3つの手順で構成されます。

- **手順1:TreeViewのデータソースの作成**
- **手順2:TreeViewコントロールの追加**
- **手順3:プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後のデータを含むTreeViewコントロールを示しています。



手順1:TreeViewのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します(例:`Property.cs`)。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`Properties.cs`モデルに追加します。ここでは、**`Property`**クラスを使用して、階層的なデータを表しています。

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class Property
    {
        public string Header { get; set; }
        public string Image { get; set; }
        public bool NewItem { get; set; }
        public Property[] Items { get; set; }
        public static Property[] GetData()
        {
            return new Property[]
            {
                new Property
                {
                    Header = "電子機器",
                    Items = new Property[]
                    {
                        new Property { Header="トリマー/シェーバー" },
                        new Property { Header="タブレット" },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                        new Property { Header="携帯電話",
                            Items = new Property[] {
                                new Property { Header="Apple" },
                                new Property { Header="Motorola", NewItem=true
},
                                new Property { Header="Nokia" },
                                new Property { Header="Samsung" }}
                        },
                        new Property { Header="スピーカー", NewItem=true },
                        new Property { Header="モニター" }
                    }
                },
                new Property{
                    Header = "玩具",
                    Items = new Property[]{
                        new Property{ Header = "縫いぐるみ" },
                        new Property{ Header = "列車セット" },
                        new Property{ Header = "クレーンゲーム", NewItem = true },
                        new Property{ Header = "プロポ" },
                        new Property{ Header = "玩具銃" }
                    }
                },
                new Property{
                    Header = "ホーム",
                    Items = new Property[] {
                        new Property{ Header = "コーヒーメーカー" },
                        new Property{ Header = "パンメーカー", NewItem = true },
                        new Property{ Header = "ソーラーパネル", NewItem = true },
                        new Property{ Header = "仕事台" },
                        new Property{ Header = "プロパングリル" }
                    }
                }
            };
        }
    }
```

先頭に戻る

手順2：TreeViewコントロールの追加

アプリケーションにTreeViewコントロールを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`TreeViewController`）。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C#

```
using <ApplicationName>.Models;
```


Copyright © GrapeCity, Inc. All rights reserved.

5. メソッド**Index()**を次のメソッドに置き換えます。

TreeViewController.cs

```
public ActionResult Index()
{
    return View(Property.GetData(Url));
}
```

コントローラーのビューの追加

ビュー内で、TreeViewコントロールのインスタンスを作成し、それを**.Bind**メソッドを使用してデータソースに連結します。**DisplayMemberPath**はツリーノードに表示されるテキストを設定し、**ChildItemsPath**は子ノードの配列を設定します。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`TreeViewController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor(CSHTML)**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

Index.cshtml copyCode

```
@using TreeView.Models
@model Property[]

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items"))
```

手順3:プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します(例:http:: http://localhost:1234/**TreeView/Index**)。

先頭に戻る

TreeViewの使用

TreeViewコントロールの使用方法を理解するために役立つ概念について説明します。

このセクションには、次の情報が含まれます。

データ連結
: TreeViewコントロールでデータ連結を行う方法を説明します。

遅延ロード
: TreeViewコントロールで遅延ロードを実装する方法を説明します。

ノード間の移動
: TreeViewコントロール内のナビゲーションについて説明します。

ノードの展開と折りたたみ
: TreeViewノードの展開および折りたたみの概念について説明します。

チェックボックスの使用

Copyright © GrapeCity, Inc. All rights reserved.

TreeViewノードにチェックボックスを追加する方法を説明します。

カスタムノード

TreeViewノードにカスタムノードコンテンツを追加する方法を説明します。

画像

TreeViewノードに画像を追加する方法を説明します。

アコーディオン

TreeViewをアコーディオンとして表示する方法を説明します。

ドラッグアンドドロップ

TreeView内でのドラッグアンドドロップの概念について説明します。

ノードの編集

TreeViewノードを編集モードで使用する方法を説明します。

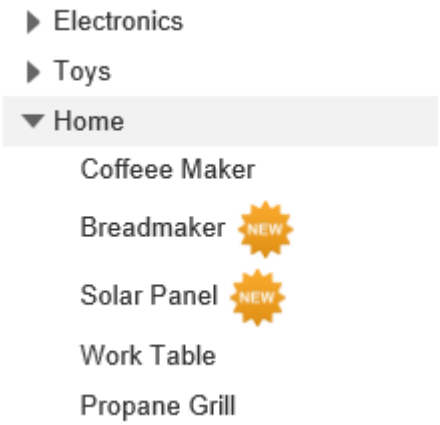
スタイル設定とCSS

TreeViewに対してスタイル設定を適用する方法を説明します。

カスタムノード

TreeViewコントロールでは、ノードにカスタムコンテンツを表示することができます。TreeViewノードのコンテンツは、**formatItem**イベントを使用してカスタマイズすることができます。イベントハンドラのパラメータには、ノードとレンダリングされるデータ項目を表す要素が含まれます。

この例では**OnClientFormatItem**イベントを使用して、TreeViewの新しい項目の右に「新しい」バッジを追加します。以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。



次のコード例では、Razorコード内でJavaScript関数名を割り当てることで、**OnClientFormatItem**メソッドをサブスクライブしています。**<Script>**セクションで、dataItemがリスト内の新しい項目かどうかをチェックし、新しいノードの横にカスタム画像を追加するformatItem関数を宣言します。画像は、MVCアプリケーションの**Content**フォルダ内に保存されます。

次のコード例では、**OnClientFormatItem()**メソッドで**formatItem**イベントを生成しています。**formatItem(treeview, args)**関数の**treeview**は送信元で、**args**は**FormatNodeEventArgs**クラス型です。

コードの場合

CustomNode.cshtml

Razor

```
@using <ApplicationName.Models>
@model Property[]

<script type="text/javascript">
     function formatItem(treeview, args) {
       if (args.dataItem.NewItem) {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        args.element.innerHTML +=
            '<img style="margin-left:6px" src="@Href("~/Content/new.png")"/>';
            }
        }
</script>

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .OnClientFormatItem("formatItem"))
```

データ連結

TreeViewコントロールとその機能を使用するには、データをTreeViewに連結する必要があります。TreeViewコントロールにデータを連結したら、遅延ロード、カスタムノード、ノードの編集、スタイルとCSS などのさまざまな機能を使用できます。

このセクションには、次の情報が含まれます。

モデル連結
: 基本のモデル連結を使用して、TreeViewコントロールにデータを追加する方法を説明します。

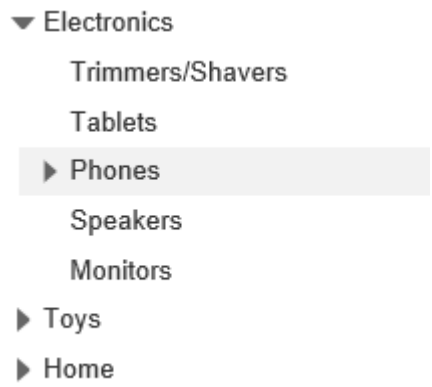
リモート連結
: C1JSONRequestを使用して、リモートでTreeViewコントロールにデータを連結する方法を説明します。

モデル連結

このトピックでは、MVCアプリケーションにモデル連結を使用して、TreeViewコントロールを追加し、コントロールにデータを挿入する方法について説明します。このトピックは次の3つの手順で構成されます。

- **手順1：TreeViewのデータソースの作成**
- **手順2：TreeViewコントロールの追加**
- **手順3：プロジェクトのビルドおよび実行**

次の図は、上記の手順を実行した後に表示されるMultiRowコントロールを示しています。



手順1：TreeViewのデータソースの作成

1. **[モデル]**フォルダに新しいクラスを追加します（例：`Property.cs`）。新しいモデルを追加する方法の詳細については、「コントロールの追加」を参照してください。
2. 次のコードを`Properties.cs`モデルに追加します。ここでは、**`Property`**クラスを使用して、階層的なデータを表しています。

Copyright © GrapeCity, Inc. All rights reserved.

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class Property
    {
        public string Header { get; set; }
        public string Image { get; set; }
        public bool NewItem { get; set; }
        public Property[] Items { get; set; }
        public static Property[] GetData()
        {
            return new Property[]
            {
               new Property
                {
                    Header = "電子機器",
                    Items = new Property[]
                    {
                        new Property { Header="トリマー/シェーバー" },
                        new Property { Header="タブレット" },
                        new Property { Header="携帯電話",
                            Items = new Property[] {
                                new Property { Header="Apple" },
                                new Property { Header="Motorola", NewItem=true
},
                                new Property { Header="Nokia" },
                                new Property { Header="Samsung" }}
                        },
                        new Property { Header="スピーカー", NewItem=true },
                        new Property { Header="モニター" }
                    }
                },
                new Property{
                    Header = "玩具",
                    Items = new Property[]{
                        new Property{ Header = "縫いぐるみ" },
                        new Property{ Header = "列車セット" },
                        new Property{ Header = "クレーンゲーム", NewItem = true },
                        new Property{ Header = "プロポ" },
                        new Property{ Header = "玩具銃" }
                    }
                },
                new Property{
                    Header = "ホーム",
                    Items = new Property[] {
                        new Property{ Header = "コーヒーメーカー" },
                        new Property{ Header = "パンメーカー", NewItem = true },
                        new Property{ Header = "ソーラーパネル", NewItem = true },
                        new Property{ Header = "仕事台" },
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                        new Property{ Header = "プロパングリル" }
                    }
                }
            };
        }
    }
```

先頭に戻る

手順2：TreeViewコントロールの追加

アプリケーションにTreeViewコントロールを追加するには、次の手順に従います。

新しいコントローラーの追加

1. **ソリューションエクスプローラー**で、**[Controllers]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[コントローラー]**を選択します。**[スキャフォールディングを追加]**ダイアログが表示されます。
3. **[スキャフォールディングを追加]**ダイアログで、次の手順に従います。
    1. **[空のMVCコントローラー]**テンプレートを選択します。
    2. コントローラーの名前を設定します（例：`TreeViewController`）。
    3. **[追加]**をクリックします。
4. 次に示すようにMVC参照を追加します。

C#

```
using <ApplicationName>.Models;
```

5. メソッド**Index()**を次のメソッドに置き換えます。

TreeViewController.cs

```
public ActionResult Index()
{
   return View(Property.GetData(Url));
}
```

コントローラーのビューの追加

ビュー内で、TreeViewコントロールのインスタンスを作成し、それを**.Bind**メソッドを使用してデータソースに連結します。**DisplayMemberPath**はツリーノードに表示されるテキストを設定し、**ChildItemsPath**は子ノードの配列を設定します。

1. **ソリューションエクスプローラー**で、**[コントローラー]**フォルダを展開し、`TreeViewController`をダブルクリックして開きます。
2. メソッド`Index()`内にカーソルを置きます。
3. 右クリックし、**[ビューの追加]**を選択します。**[ビューの追加]**ダイアログが表示されます。
4. **[ビューの追加]**ダイアログで、ビュー名が**Index**、ビューエンジンが**Razor（CSHTML）**であることを確認します。
5. **[追加]**をクリックしてコントローラーのビューを追加します。次のコードをコピーし、**Index.cshtml**に貼り付けます。

Index.cshtml　copyCode

```
@using TreeView.Models
@model Property[]

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    .ChildItemsPath("Items"))
```

手順3：プロジェクトのビルドおよび実行

1. **[ビルド]**→**[ソリューションのビルド]**をクリックして、プロジェクトをビルドします。
2. **[F5]**キーを押してプロジェクトを実行します。

> ブラウザのアドレスバーで、生成されたURLにフォルダ名とビュー名を付加してビューを確認します（例：http:: http://localhost:1234/**TreeView/Index**）。

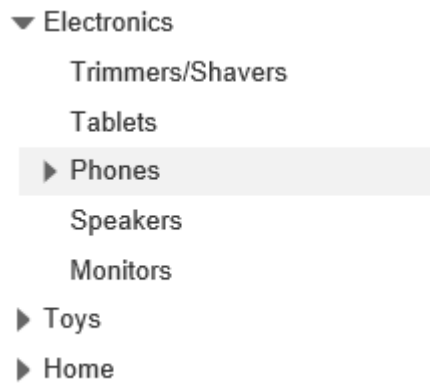
先頭に戻る

リモート連結

MultiRowコントロールでは、**C1JSONRequest**を使用してデータを直接取得できます。これは、サーバー、テーブル、および列を含むリモートデータのURLを指定します。返される配列は、CollectionViewオブジェクトのデータソースとして使用されます。Tag Helpersでは、TreeViewの**LoadActionURL** プロパティは、特定の操作を実行するためのアクションURLメソッドを渡すことで、TreeViewをコレクションに連結するために使用されます。同様に、HTML Helpersでは、Bind プロパティをコレクションに連結するために使用されます。

このトピックでは、既存のデータソースからデータをリモートで取得する方法について説明します。これは、データをダッシュボードとして表すデータ集約型のアプリケーションやシナリオを開発する際に役立ちます。

次の図は、C1JSON要求を行ってデータをモデルからフェッチした後のTreeViewコントロールを示します。この例では、「クイックスタート」トピックで作成した **Property** モデルのサンプルを使用しています。



コードの場合

RemoteBindController.cs

C#

```
public class TreeViewController : Controller
    {
        // GET: TreeView
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult RemoteLoading_LoadAction()
        {
            return Json(Property.GetData(url));
        }
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

RemoteBind.cshtml

Razor

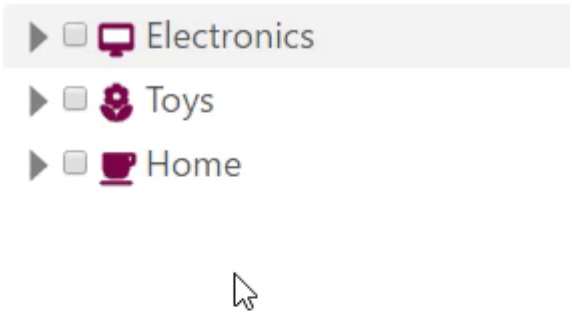
```
@(Html.C1().TreeView()
    .Bind(Url.Action("RemoteLoading_LoadAction"))
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items"))
```

ドラッグアンドドロップ

TreeViewコントロールでは、同じツリービュー内および複数のツリービュー間でのドラッグアンドドロップ操作がサポートされています。TreeViewノードでドラッグアンドドロップ操作を行うには、**TreeView**の**AllowDragging**プロパティをtrueに設定する必要があります。

TreeViewでドラッグアンドドロップ操作が有効になっている場合、ユーザーは任意のノードをツリー内の任意の位置にドラッグできます。この動作は、TreeViewのドラッグアンドドロップイベントを処理することでカスタマイズできます。

- **OnClientDragStart**:ドラッグまたはドロップ操作を開始しようとしたときに発生します。イベントのcancelパラメータをtrueに設定することで、ドラッグしようとしているノードを調べて操作をキャンセルすることができます。
- **OnClientDragOver**:ユーザーがノードをツリーの他のノード上にドラッグしているときに発生します。イベントのcancelパラメータとpositionパラメータを設定することで、現在のターゲットノードとドロップ位置を調べて、ドロップを防いだり、場所を変更することができます。
- **OnClientDrop**:ユーザーがノードを別の場所にドロップしたときに発生します。イベントのcancelパラメータとpositionパラメータを設定することで、現在のターゲットノードとドロップ位置を調べて、ドロップを防いだり、場所を変更することができます。
- **OnClientDragEnd**:ドラッグ/ドロップ操作が終了した後に、操作がキャンセルされたりソースノードが移動されなかった場合でも、発生します。



この例では、OnClientDragStartとOnClientDragOverを使用して、TreeViewコントロールでカスタマイズされたドラッグアンドドロップ操作を提供しています。**OnClientDragStart**イベントを使用して親ノードのドラッグを許可し、**OnClientDragOver**イベントを使用して空ノードへのドロップを許可しています。

Copyright © GrapeCity, Inc. All rights reserved.

以下のコード例では、**dragStart**イベントを発生させるOnClientDragStart()メソッドを使用しています。**dragStart(treeview, e)**関数の**treeview**は送信元で、**e**は**TreeNodeEventArgs**クラス型です。同様に、OnClientDragOver()メソッドは**dragOver**イベントを発生させます。**dragOver(treeview, e)**関数の**treeview**は送信元で、**e**は**TreeNodeEventArgs**クラス型です。

以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。

In Code

Razor

```
@using <ApplicationName.Models>
@model Property[]

<script type="text/javascript">
        var allowDraggingParentNodes = true;
        var allowDroppingIntoEmpty = true;

        // OnClientDragStartイベントを使用して、allowDraggingParentNodes設定を適用します。
        // それには、「cancel」イベントパラメータをtrueに設定します
        function dragStart(treeview, e) {
            if (e.node.hasChildren) {
                if (!allowDraggingParentNodes) {
                    e.cancel = true; // 親ノードのドラッグを防ぎます
                } else {
                    e.node.isCollapsed = true; // ドラッグ時に親ノードを折りたたみます
                }
            }
        }

        // OnClientDragOverイベントを使用して、allowDroppingIntoEmpty設定を適用します。
        // それには、「position」イベントパラメータを「Before」に変更します
        function dragOver(treeview, e) {
            if (!allowDroppingIntoEmpty &&
                !e.dropTarget.hasChildren &&
                 e.position == wijmo.nav.DropPosition.Into) {
                e.position = wijmo.nav.DropPosition.Before;
            }
        }
</script>

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .ImageMemberPath("Image")
    .ShowCheckboxes(true)
    .AllowDragging(true)
    .OnClientDragStart("dragStart")
    .OnClientDragOver("dragOver"))
```

ノードの編集

TreeViewコントロールでは、**IsReadOnly**プロパティを使用して、実行時のノードの編集を有効または無効にすることができま

Copyright © GrapeCity, Inc. All rights reserved.

す。IsReadOnlyプロパティがfalseに設定されている場合は、ノードをダブルクリックするか、ノードを選択した後に[F2]キーを押して、ノードの内容を編集できます。IsReadOnlyプロパティをtrueに設定すると、ツリーのノードを編集できないように制限できます。

ユーザーがノードの編集を終了すると、**DisplayMemberPath**プロパティを使用して、ノードの内容がItemsSource配列内の項目に自動的に適用されます。TreeViewコントロールでは、次のイベントを使用して編集動作をカスタマイズできます。

- **OnClientNodeEditStarting** - ノードが編集モードに入る前に発生します。
- **OnClientNodeEditStarted** - ノードが編集モードに入った後に発生します。
- **OnClientNodeEditEnding** - ノードが編集モードを出た後に発生します。
- **OnClientNodeEditEnded** - ノードが編集モードを出る前に発生します。

以下のコード例では、OnClientNodeEditStartingメソッドを使用して、子を持たないノードに対してのみ編集を有効にしています。次のコード例では、**OnClientNodeEditStarting()**メソッドで**nodeEditStarting**イベントを生成しています。**nodeEditStarting(treeview, e)**関数の**treeview**は送信元で、**e**は**TreeNodeEventArgs**クラス型です。

以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。

コードの場合

EditingNodes.cshtml

Razor

```
@using <ApplicationName.Models>
@model Property[]
<script type="text/javascript">
        function nodeEditStarting(treeview, e) {
            if (e.node.hasChildren) {
                e.cancel = true;
            }
        }
</script>

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .ImageMemberPath("Image")
    .ShowCheckboxes(true)
    .IsReadOnly(false)
    .OnClientNodeEditStarting("nodeEditStarting"))
```

ノードの展開と折りたたみ

TreeViewコントロールは階層的な項目リストを表示します。ツリービューの各ノードには複数の子ノードを含めることができ、これらの子ノードは親ノードの下にインデントされて表示されます。子ノードは、その親のノードが展開されると表示され、親ノードが折りたたまれると非表示になります。

TreeViewクラスでは、任意のノードをクリックしたときに1つのノード（親または子）を展開するための**ExpandOnClick**プロパティが提供されています。ExpandOnClickメソッドは、特定のノード内の子ノードを展開するかどうかを決定するブール値を受け取ります。このプロパティをtrueに設定すると、1回のマウスクリックですべての子ノードを展開できます。

In addition, the TreeView class also provides **ExpandOnLoad** property that determines whether to toggle checkboxes when the user clicks the node header. This property accepts Boolean value and by default, it is set to false.

TreeViewコントロールでは、現在選択されている親ノードの兄弟ノードを自動的に閉じるための**AutoCollapse**プロパティが提

Copyright © GrapeCity, Inc. All rights reserved.

供されています。これは、同じTreeView内の別の親ノードをクリックしたときに起こります。

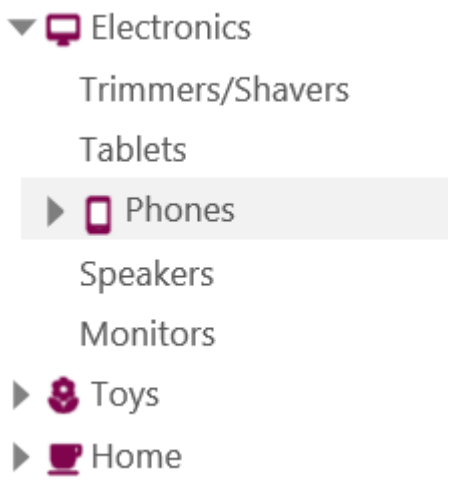
この例では、**ExpandOnClick**および**AutoCollapse**メソッドを使用して、ノードに対して展開および折りたたみ操作を実行します。以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。

Razor

```
@using <ApplicationName.Models>
@model Property[]@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .AutoCollapse(true)
    .ExpandOnClick(true))
```

画像

TreeViewコントロールにはさまざまな外観を与えることができます。そのために、柔軟な画像カスタマイズ機能とカスタムユーザーインタフェース(UI)オプションを指定するプロパティが提供されています。TreeViewでは、ノード、接続線、および展開/折りたたみアイコンを表す画像を使用できます。**ImageMemberPath**プロパティを使用して、画像URLを含むデータ項目のプロパティの名前を指定することで、ノードに画像を追加できます。



このサンプルでは、electronics.png、phones.png、toys.png、およびhome.pngを使用しています。これらの画像は、アプリケーションの**/Content/images/**フォルダに保存されています。データソースには**Image**フィールドが含まれており、その値として画像URLが含まれています。

次のコード例は、TreeViewノードに画像を追加する方法を示します。

コードの場合

モデル - Property.cs

C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class Property
    {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public string Header { get; set; }
        public string Image { get; set; }
        public bool NewItem { get; set; }
        public Property[] Items { get; set; }
        public static Property[] GetData(UrlHelper urlHelper)
        {
            return new Property[]
            {
                new Property
                {
                    Header = "電子機器",
                    Image = urlHelper.Content("~/Content/images/electronics.png"),
                    Items = new Property[]
                    {
                        new Property { Header="トリマー/シェーバー" },
                        new Property { Header="タブレット" },
                        new Property { Header="携帯電話",
                            Image =urlHelper.Content("~/Content/images/phones.png"),
                            Items = new Property[] {
                                new Property { Header="Apple" },
                                new Property { Header="Motorola", NewItem=true },
                                new Property { Header="Nokia" },
                                new Property { Header="Samsung" }}
                        },
                        new Property { Header="スピーカー", NewItem=true },
                        new Property { Header="モニター" }
                    }
                },
                new Property{
                    Header = "玩具",
                    Image = urlHelper.Content("~/Content/images/toys.png"),
                    Items = new Property[]{
                        new Property{ Header = "縫いぐるみ" },
                        new Property{ Header = "列車セット" },
                        new Property{ Header = "クレーンゲーム", NewItem = true },
                        new Property{ Header = "プロポ" },
                        new Property{ Header = "玩具銃" }
                    }
                },
                new Property{
                    Header = "ホーム",
                    Image = urlHelper.Content("~/Content/images/home.png"),
                    Items = new Property[] {
                        new Property{ Header = "コーヒーメーカー" },
                        new Property{ Header = "パンメーカー", NewItem = true },
                        new Property{ Header = "ソーラーパネル", NewItem = true },
                        new Property{ Header = "仕事台" },
                        new Property{ Header = "プロパングリル" }
                    }
                }
            };
        }
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

TreeViewController

TreeViewController.cs

```
using <ApplicationName>.Models;

public ActionResult Index()
{
   return View(Property.GetData(url));
}
```

ビュー - Index.cshtml

Razor

```
@using <ApplicationName.Models>
@model Property[]

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .ImageMemberPath("Image"))
```

遅延ロード

TreeViewを使用する場合は、項目のコレクションに連結するのが一般的なプロセスです。しかし、ノードの子項目が実際に必要になるまで、子項目のロードを遅らせたいこともあります。この手法は、多数のレベルと子ノードから成る極めて深いツリーがある場合に特に役立ちます。Windowsコンピュータのフォルダ構成が典型的な例です。

TreeViewコントロールでは、遅延ロードを簡単に実装できます。遅延ロードモードでTreeViewを表示するには、タグヘルパーの場合は、必要になったときに項目をロードするアクションを定義し、**lazy-load-action-url**タグにアクションURLを設定します。HTMLヘルパーの場合は、遅延ロードデータを返すアクションとして、**Bind**プロパティの2番目のパラメータを使用する必要があります。

これとは別に、JavaScriptコードを使用して、クライアント側で遅延ロードを設定することもできます。**LazyLoadFunction**プロパティを、ユーザーがノードを展開したときに呼び出す関数名に設定します。この関数は、2つのパラメータとして、親ノードと、データが利用可能になったときに呼び出されるコールバックを受け取ります。

以下のコード例を参考に、TreeViewコントロールで遅延ロード機能を実装する方法を理解してください。

メモ:このサンプルでは、TreeViewコントロールのデータソースとして、**C1Nwind.mdf**データベースファイルと**EmployeeEx.cs**を使用しています。デフォルトでは、**C1Nwind.mdf** ファイルは、システムの*C:\Users\<username>\Documents\ComponentOne Samples\ASP.NET MVC\MVC\CS\MvcExplorer\App_Data*フォルダに用意されています。C1Nwindデータベースの詳細については、「データの設定」を参照してください。

Copyright © GrapeCity, Inc. All rights reserved.



コードの場合

モデル - EmployeeEx.cs

C#

```
using System.Collections.Generic;
using System.Linq;
using System.Data.Entity;

namespace LazyLoading.Models
{
    public class EmployeeEx
    {
        private static DbSet employees = new C1NWindEntities().Employees;
        public int EmployeeID { get; set; }
        public string Name { get; set; }
        public EmployeeEx[] SubExployees { get; set; }

        public static IEnumerable GetEmployees(int? leaderID)
        {
            return employees.Where(e => e.ReportsTo == leaderID).ToList().Select(e =>
new EmployeeEx
            {
                EmployeeID = e.EmployeeID,
                Name = e.FirstName + " " + e.LastName,
                SubExployees = employees.Any(ep => ep.ReportsTo == e.EmployeeID) ?
new EmployeeEx[0] : null
            });
        }
    }
}
```

コントローラー - LazyLoadingController.cs

Copyright © GrapeCity, Inc. All rights reserved.

Razor

```
using C1.Web.Mvc;
using System.Web.Mvc;
using <ApplicationName.Models>;

namespace LazyLoading.Controllers
{
    partial class TreeViewController : Controller
    {

        // GET: LazyLoad
        public ActionResult LazyLoading()
        {
            return View();
        }

        public ActionResult Load()
        {
            return Json(EmployeeEx.GetEmployees(null));
        }

        public ActionResult LazyLoading_LoadAction([C1JsonRequest]TreeNode node)
        {
            var leaderID = (int?)node.DataItem["EmployeeID"];
            return Json(EmployeeEx.GetEmployees(leaderID));
        }
    }
}
```

ビュー - LazyLoading.cshtml

Razor

```
@(Html.C1().TreeView()
    .Bind(Url.Action("Load"), Url.Action("LazyLoading_LoadAction"))
    .DisplayMemberPath("Name")
    .ChildItemsPath("SubExployees"))
```

先頭に戻る

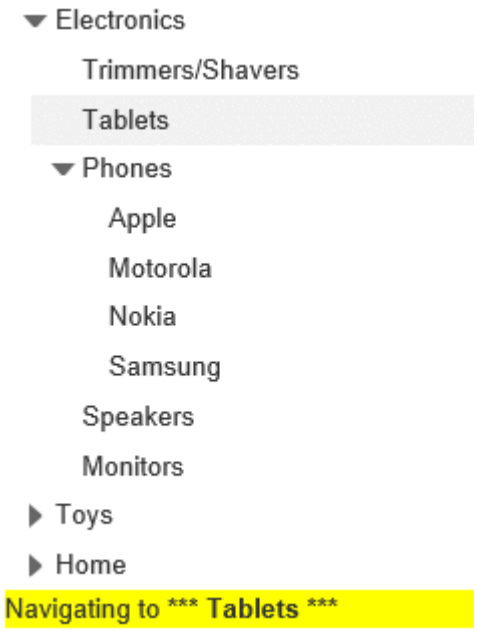
ノード間の移動

TreeViewコントロールの最も単純で一般的な用途はナビゲーションです。TreeViewの階層構造と自動検索機能により、目的の項目へのドリルダウンと検索を容易に行うことができます。TreeViewでは、マウスとキーボードによるノード間の移動がサポートされています。

TreeViewコントロール内のナビゲーションには、**OnClientSelectedItemChanged**イベントまたは**OnClientItemClicked**イベントを使用することができます。両者の違いは、キーボードを使用して選択項目を移動したときに**OnClientSelectedItemChanged**が発生するのに対して、項目をクリックするか[Enter]キーを押したときには**OnClientItemClicked**が発生する点です。

この例では、OnClientItemClickedイベントを使用してナビゲーションを行います。以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.



コードの場合

NavigationController.cs

C#

```
public class TreeViewController : Controller
    {
        // GET: TreeView
        public ActionResult Index()
        {
            return View(Property.GetData(Url));
        }
    }
```

Navigation.cshtml

Razor

```
@using <ApplicationName.Models>
@model Property[]

<script type="text/javascript">
function itemClicked(treeView)
        {
document.getElementById('tvNavItem').innerHTML = '<b> *** ' +
        treeView.selectedItem.Header + ' ***</b>' + 'へ移動';}
</script>

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .OnClientItemClicked("itemClicked"))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
<div id="tvNavItem" style="background-color:yellow"></div>
```

マウスによる移動

次の表に、TreeViewでノード間を移動するためのアクションと、対応するマウスコマンドを示します。

アクション	マウスコマンド
ノードの展開	ノード名の左側にあるプラス記号をクリックします。
ノードの折りたたみ	ノード名の左側にあるマイナス記号をクリックします。
ノードの選択	ノード名をクリックします。

キーボードナビゲーション

次の表に、TreeViewでノード間を移動する際に使用するアクションと、関連するキーを示します。

アクション	キーボードコマンド
ノードを上に移動	[↑]キー
ノードを下に移動	[↓]キー
非連続選択	マウス +[Ctrl]
連続選択	マウス +[Shift]

スタイル設定とCSS

CSS classes for the TreeView control are as follows:

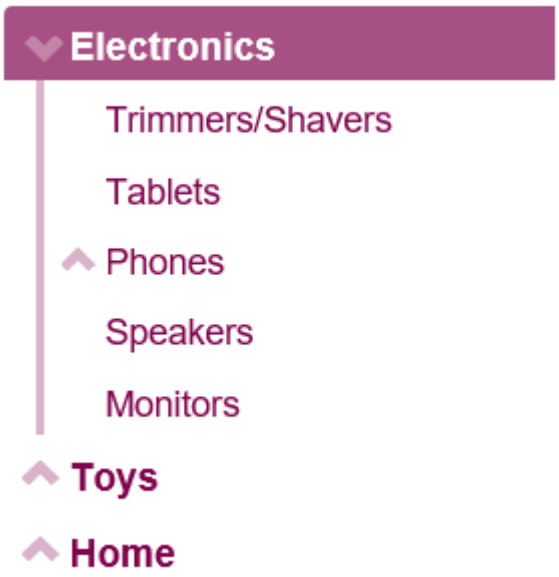
Class	Description
.wj-treeview	Styles the div host element of the TreeView control.
wj-nodelist	Styles all the nodes including the selected node in the TreeView control.
.wj-node .wj-node:before	Styles all the nodes/specific nodes, except the selected node in the TreeView control.
.wj-node .wj-node-text	Styles the node text.
.wj-node .wj-state-selected	Styles the currently selected node in the TreeView control.
.wj-node .wj-state-collapsed:before	Styles the node that is in collapsed state.
.wj-node .wj-state-collapsing:before	Styles the node that is in collapsing state.
.wj-node .wj-state-loading .wj-node .wj-state-loading:before	Styles the node that is in loading state.
.wj-node .wj-state-empty:before	Styles the node that has no child nodes.
.wj-node .wj-node-check	Styles the check-box element that is displayed for every node after setting the ShowCheckboxes property to true.
.wj-node img	Styles the image of node.

Copyright © GrapeCity, Inc. All rights reserved.

.wj-marker	Styles the marker line which appears when performing the drag drop operations.
.wj-rtl .wj-rtl .wj-nodelist .wj-node:before .wj-rtl .wj-nodelist .wj-nodelist .wj-rtl .wj-nodelist .wj-node.wj-state-collapsed:before .wj-rtl .wj-node img .wj-rtl .wj-node .wj-node-check .wj-rtl .wj-nodelist .wj-node.wj-state-collapsing:before	Styles the Treeview control rendered in right to left direction.

スタイルの適用

TreeViewコントロールは、**CSS**を使用して外観をカスタマイズできます。独自のカスタムCSSを作成し、**CssClass**プロパティを使用してTreeViewコントロールに適用できます。次の図は、カスタムCSSクラスを使用した後のTreeViewコントロールの外観を示します。



この例では、**CssClass**プロパティを使用して、TreeViewコントロールにカスタムカスケードスタイルシートを適用しています。以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。

カスタムスタイルシートの追加

新しいASP.NET MVCアプリケーションを作成します。アプリケーションを作成し、アプリケーションにビューを追加すると、**ソリューションエクスプローラー**に**[コンテンツ]**フォルダが作成されます。アプリケーションにカスタムスタイルシートを追加するには、次の手順に従います。

1. ソリューションエクスプローラーで、**[コンテンツ]**フォルダを右クリックします。
2. コンテキストメニューから、**[追加]**→**[スタイルシート]**を選択します。**[項目の名前を指定]**ダイアログが表示されます。

Copyright © GrapeCity, Inc. All rights reserved.

3. スタイルシートの名前を設定し(例:app.css)、**[OK]**をクリックします。
4. **app.css**ファイルのデフォルトのコードを次のコードに置き換えます。

app.css

```
.wj-treeview {
    font-size: 120%;
    margin-bottom: 8px;
}

/* カスタムツリースタイル*/
.custom-tree.wj-treeview {
    color: #80044d;
}

    /* デフォールトノード */
    .custom-tree.wj-treeview .wj-node {
    }

    /* レベル0以上のノード */
    .custom-tree.wj-treeview .wj-nodelist > .wj-node {
        font-size: 120%;
        font-weight: bold;
    }

    /* レベル1以上のノード(小さいフォント、左に沿った縦線) */
    .custom-tree.wj-treeview .wj-nodelist > .wj-nodelist > .wj-node,
    .custom-tree.wj-treeview .wj-nodelist > .wj-nodelist > .wj-nodelist {
        font-size: 110%;
        font-weight: normal;
        border-left: 4px solid rgba(128, 4, 77, 0.3);
    }

    /* レベル2以上のノード(小さいフォント、薄いボーダー) */
    .custom-tree.wj-treeview .wj-nodelist > .wj-nodelist  > .wj-nodelist > .wj-
node,
    .custom-tree.wj-treeview .wj-nodelist > .wj-nodelist  > .wj-nodelist > .wj-
nodelist {
        font-size: 100%;
        font-style: italic;
        opacity: 0.8;
        border-left: 2px solid rgba(128, 4, 77, 0.3);
    }

    /* 拡張されたノードグリフ */
    .custom-tree.wj-treeview .wj-nodelist .wj-node:before {
        content: "\e114";
        font-family: 'Glyphicons Halflings';
        top: 4px;
        border: none;
        opacity: .3;
        transition: all .3s cubic-bezier(.4,0,.2,1);
    }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
/* 折りたたまれたノードグリフ */
.custom-tree.wj-treeview .wj-nodelist .wj-node.wj-state-collapsed:before,
.custom-tree.wj-treeview .wj-nodelist .wj-node.wj-state-collapsing:before {
    transform: rotate(-180deg);
    transition: all .3s cubic-bezier(.4,0,.2,1);
}

/* 選択されたノード */
.custom-tree.wj-treeview .wj-node.wj-state-selected {
    color: white;
    background: rgba(128, 4, 77, 0.70);
}
```

コードの場合

このコード例では、折りたたみ/展開アイコンを変更し、ノードレベルに応じて異なるフォントサイズを使用し、レベル1のノードの左側に縦棒を追加しています。

Styling.cshtml

Razor

```
@using <ApplicationName.Models>
@model Property[]

@(Html.C1().TreeView().CssClass("custom-tree")
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items"))
```

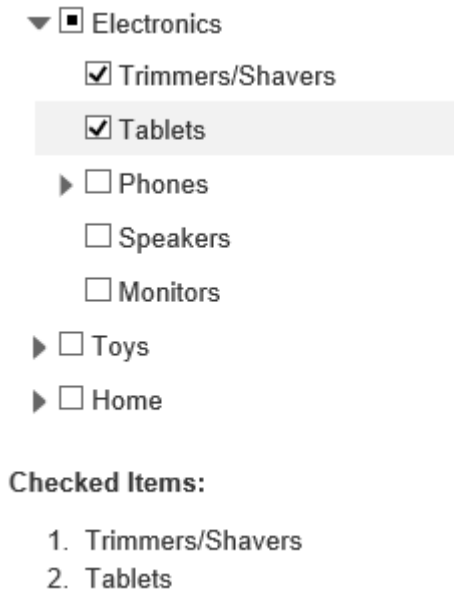
チェックボックスの使用

Treeviewには、ノードをチェックボックスとして表示するオプションがあります。TreeViewコントロールにチェックボックスを表示するには、**ShowCheckboxes**プロパティをtrueに設定する必要があります。

チェックボックスが表示されると、TreeViewコントロールがその階層を管理し、チェックボックスがオンまたはオフにされると、新しい値がすべての子ノードに自動的に適用され、親ノードの状態に反映されます。項目をオンまたはオフにすると、**OnClientCheckedItemsChanged**イベントが生成され、クライアントTreeViewオブジェクトの**checkedItems**プロパティに含まれる現在オンの項目のリストが更新されます。In addition, the TreeView class provides CheckOnClick property which determines whether to toggle checkboxes when the user clicks the node header.

この例では、OnClientCheckedItemsChangedイベントを使用して、TreeViewコントロールに現在オンのチェックボックスを表示しています。以下のコード例では、「クイックスタート」セクションで追加した**Property**モデルを使用しています。

Copyright © GrapeCity, Inc. All rights reserved.



コードの場合

Checkboxes.cshtml

Razor

```
@using <ApplicationName.Models>
@model Property[]

<script type="text/javascript">
        function checkedItemsChanged(treeView) {
            var items = treeView.checkedItems,
                msg = '';
            if (items.length) {
                msg = '<p><b>チェック項目:</b></p><ol>\r\n';
                for (var i = 0; i < items.length; i++) {
                    msg += '<li>' + items[i].Header + '</li>\r\n';
                }
                msg += '</ol>';
            }
            document.getElementById('tvChkStatus').innerHTML = msg;
        }
    </script>

@(Html.C1().TreeView()
    .Bind(Model)
    .DisplayMemberPath("Header")
    .ChildItemsPath("Items")
    .ShowCheckboxes(true)
    .OnClientCheckedItemsChanged("checkedItemsChanged"))
<br />
<div id="tvChkStatus"></div>
```

Furthermore, TreeView has a checkedMemberPath property, similar to the ListBox control, that defines the name of the property that determines whether an item is checked or not. The items currently checked (selected) can be

Copyright © GrapeCity, Inc. All rights reserved.

obtained using the checkedItems property.

The following code demonstrates how to determine whether an item is checked or not. Add the following code to the sample code above.

Razor

```
.CheckedMemberPath("IsSelected")
```


Copyright © GrapeCity, Inc. All rights reserved.

Globalization

Globalization is the process of designing and developing applications that function for multiple cultures. Localization is the process of customizing your application for a specific culture and locale.

By default, ComponentOne MVC controls format and parse the data using American English culture. The decimal symbol is a period, the thousand separator is a comma, and the days of the week are "Sunday" through "Saturday". However, these symbols or separator are different for other cultures. In case the application targets any other culture, register the appropriate culture using MVC scripts in your HTML pages.

For example, to localize an application for the Japanese culture, register the "**ja**" culture in **_Layout.cshtml** file.

_Layout.cshtml

```
@Html.C1().Scripts().Basic().Culture("ja")
```


Copyright © GrapeCity, Inc. All rights reserved.

スタイル設定

ASP.NET MVC Edition allows you to style and apply css to all the available controls. For detailed understanding, see the following topics:

CSS スタイル
: Learn about the styling that can be applied to the controls using CSS.

テーマ
: Learn how to apply themes to the MVC controls.

CSS スタイル

This topic lists the controls in ASP.NET MVC Edition whose appearance can be customized using the respective CSS classes.

Control	CSS Classes and Appearance Customization
FlexChart	• CSS classes • Styling FlexChart
FlexGrid	• CSS classes • Styling FlexGrid
Input Controls	• CSS classes • Styling Input controls
OLAP	• CSS classes • Styling OLAP
TabPanel	• CSS classes • Styling TabPanel
TreeView	• CSS classes • Styling TreeView • **Styling Accordion**

テーマ

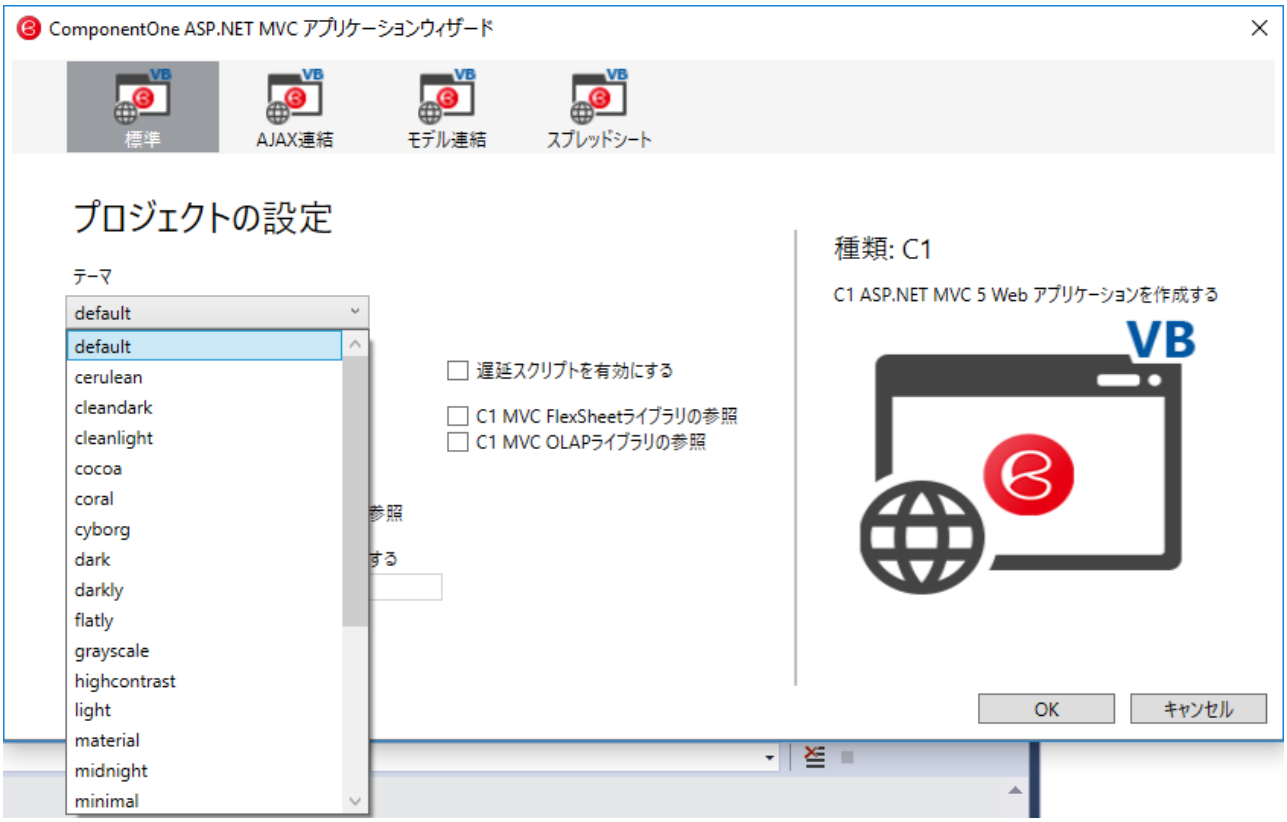
テーマは、コントロールの外観をカスタマイズするために適用できる、カラー、フォント、画像、スキンなどのGUI要素のコレクションです。MVC は数々の組み込みテーマをサポートしており、これらをMVCアプリケーションの作成時に直接選択したり、リソースの登録時に設定することができます。MVCアプリケーションのリソースを登録するときにマテリアルテーマを設定することもできます。

デフォルトでは、MVCアプリケーションを作成するときに共通のデフォルトテーマが使用されます。外観は、「**default**」という固有のC1テーマを使用して実装されます。ASP.NET MVCコントロールには、多数の定義済みのビジュアルテーマがあり、Webサイトページのすべてのコントロールに一貫した外観を提供できます。これらの定義済みのテーマは、**C1.Web.Mvc.Themes** の静的クラスで定義されています。

MVCコントロールにテーマを適用する方法は2つあります。

1. ComponentOneテンプレートを使用する - **ComponentOne ASP.NET MVCアプリケーションウィザード**を使用し、**テーマ**オプションを使用してアプリケーションにテーマを設定します。以下の画像には、ASP.NET MVCアプリケーションで使用できるテーマのリストが表示されています。

Copyright © GrapeCity, Inc. All rights reserved.



2. Visual Studioテンプレートを使用する - MVCアプリケーションのリソースを登録するときにテーマを設定することもできます。MVCアプリケーションでテーマを適用するには、_Layout.cshtmlファイルに次のコードを追加する必要があります。

_Layout.cshtml

```
//ComponentOne Themeを設定します
@Html.C1().Styles().Theme("cocoa")

//Theme Enumを使用して設定します
@Html.C1().Styles().Theme(C1.Web.Mvc.Themes.Cocoa)

//Material Themeを設定します
@Html.C1().Styles().Theme("material.deep_orange-red")
```

The complete list of material themes supported with ComponentOne MVC Controls are provided below.

Material Themes

material.amber-blue	material.cyan-pink	material.indigo-light_blue	material.pink-deep_orange
material.amber-cyan	material.cyan-purple	material.indigo-light_green	material.pink-deep_purple
material.amber-deep_orange	material.cyan-red	material.indigo-lime	material.pink-green
material.amber-deep_purple	material.cyan-teal	material.indigo-orange	material.pink-indigo
material.amber-green	material.cyan-yellow	material.indigo-pink	material.pink-light_blue
material.amber-indigo	material.deep_orange-amber	material.indigo-purple	material.pink-light_green
material.amber-light_blue	material.deep_orange-blue	material.indigo-red	material.pink-lime
material.amber-light_green	material.deep_orange-cyan	material.indigo-teal	material.pink-orange
material.amber-lime	material.deep_orange-	material.indigo-yellow	material.pink-purple

Copyright © GrapeCity, Inc. All rights reserved.

	deep_purple		
material.amber-orange	material.deep_orange-green	material.light_blue-amber	material.pink-red
material.amber-pink	material.deep_orange-indigo	material.light_blue-blue	material.pink-teal
material.amber-purple	material.deep_orange-light_blue	material.light_blue-cyan	material.pink-yellow
material.amber-red	material.deep_orange-light_green	material.light_blue-deep_orange	material.purple-amber
material.amber-teal	material.deep_orange-lime	material.light_blue-deep_purple	material.purple-blue
material.amber-yellow	material.deep_orange-orange	material.light_blue-green	material.purple-cyan
material.blue_grey-amber	material.deep_orange-pink	material.light_blue-indigo	material.purple-deep_orange
material.blue_grey-blue	material.deep_orange-purple	material.light_blue-light_green	material.purple-deep_purple
material.blue_grey-cyan	material.deep_orange-red	material.light_blue-lime	material.purple-green
material.blue_grey-deep_orange	material.deep_orange-teal	material.light_blue-orange	material.purple-indigo
material.blue_grey-deep_purple	material.deep_orange-yellow	material.light_blue-pink	material.purple-light_blue
material.blue_grey-green	material.deep_purple-amber	material.light_blue-purple	material.purple-light_green
material.blue_grey-indigo	material.deep_purple-blue	material.light_blue-red	material.purple-lime
material.blue_grey-light_blue	material.deep_purple-cyan	material.light_blue-teal	material.purple-orange
material.blue_grey-light_green	material.deep_purple-deep_orange	material.light_blue-yellow	material.purple-pink
material.blue_grey-lime	material.deep_purple-green	material.light_green-amber	material.purple-red
material.blue_grey-orange	material.deep_purple-indigo	material.light_green-blue	material.purple-teal
material.blue_grey-pink	material.deep_purple-light_blue	material.light_green-cyan	material.purple-yellow
material.blue_grey-purple	material.deep_purple-light_green	material.light_green-deep_orange	material.red-amber
material.blue_grey-red	material.deep_purple-lime	material.light_green-deep_purple	material.red-blue
material.blue_grey-teal	material.deep_purple-orange	material.light_green-green	material.red-cyan
material.blue_grey-yellow	material.deep_purple-pink	material.light_green-indigo	material.red-deep_orange
material.blue-amber	material.deep_purple-purple	material.light_green-light_blue	material.red-deep_purple
material.blue-cyan	material.deep_purple-red	material.light_green-lime	material.red-green
material.blue-deep_orange	material.deep_purple-teal	material.light_green-orange	material.red-indigo
material.blue-deep_purple	material.deep_purple-yellow	material.light_green-pink	material.red-light_blue
material.blue-green	material.green-amber	material.light_green-purple	material.red-light_green
material.blue-indigo	material.green-blue	material.light_green-red	material.red-lime
material.blue-light_blue	material.green-cyan	material.light_green-teal	material.red-orange

Copyright © GrapeCity, Inc. All rights reserved.

material.blue-light_green	material.green-deep_orange	material.light_green-yellow	material.red-pink
material.blue-lime	material.green-deep_purple	material.lime-amber	material.red-purple
material.blue-orange	material.green-indigo	material.lime-blue	material.red-teal
material.blue-pink	material.green-light_blue	material.lime-cyan	material.red-yellow
material.blue-purple	material.green-light_green	material.lime-deep_orange	material.teal-amber
material.blue-red	material.green-lime	material.lime-deep_purple	material.teal-blue
material.blue-teal	material.green-orange	material.lime-green	material.teal-cyan
material.blue-yellow	material.green-pink	material.lime-indigo	material.teal-deep_orange
material.brown-amber	material.green-purple	material.lime-light_blue	material.teal-deep_purple
material.brown-blue	material.green-red	material.lime-light_green	material.teal-green
material.brown-cyan	material.green-teal	material.lime-orange	material.teal-indigo
material.brown-deep_orange	material.green-yellow	material.lime-pink	material.teal-light_blue
material.brown-deep_purple	material.grey-amber	material.lime-purple	material.teal-light_green
material.brown-green	material.grey-blue	material.lime-red	material.teal-lime
material.brown-indigo	material.grey-cyan	material.lime-teal	material.teal-orange
material.brown-light_blue	material.grey-deep_orange	material.lime-yellow	material.teal-pink
material.brown-light_green	material.grey-deep_purple	material.orange-amber	material.teal-purple
material.brown-lime	material.grey-green	material.orange-blue	material.teal-red
material.brown-orange	material.grey-indigo	material.orange-cyan	material.teal-yellow
material.brown-pink	material.grey-light_blue	material.orange-deep_orange	material.yellow-amber
material.brown-purple	material.grey-light_green	material.orange-deep_purple	material.yellow-blue
material.brown-red	material.grey-lime	material.orange-green	material.yellow-cyan
material.brown-teal	material.grey-orange	material.orange-indigo	material.yellow-deep_orange
material.brown-yellow	material.grey-pink	material.orange-light_blue	material.yellow-deep_purple
material.cyan-amber	material.grey-purple	material.orange-light_green	material.yellow-green
material.cyan-blue	material.grey-red	material.orange-lime	material.yellow-indigo
material.cyan-deep_orange	material.grey-teal	material.orange-pink	material.yellow-light_blue
material.cyan-deep_purple	material.grey-yellow	material.orange-purple	material.yellow-light_green
material.cyan-green	material.indigo-amber	material.orange-red	material.yellow-lime
material.cyan-indigo	material.indigo-blue	material.orange-teal	material.yellow-orange
material.cyan-light_blue	material.indigo-cyan	material.orange-yellow	material.yellow-pink
material.cyan-light_green	material.indigo-deep_orange	material.pink-amber	material.yellow-purple
material.cyan-lime	material.indigo-deep_purple	material.pink-blue	material.yellow-red
material.cyan-orange	material.indigo-green	material.pink-cyan	material.yellow-teal

Copyright © GrapeCity, Inc. All rights reserved.