GrapeCity | Developer Solutions

BEST PRACTICES FOR USING **DATAGRIDS** EFFECTIVELY IN **.NET** APPLICATIONS

BY DIRK **STRAUSS**

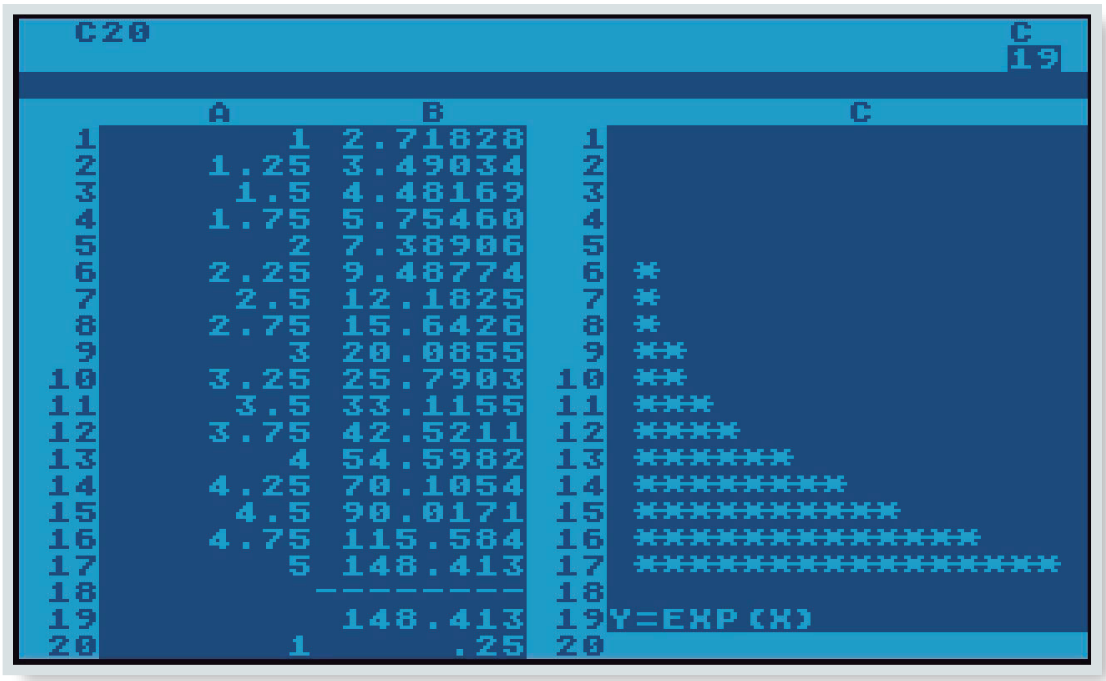
Table of Content

Introducción

Since I started my career, I can always remember having a datagrid control (also known as a grid view) available to use in my applications — whether WinForms or web-based applications.

This got me thinking: when were there not grid views for data? For how long have computer folks had the notion of using a datagrid in applications?

It would seem as though the first widely used spreadsheet was VisiCalc, which was released in 1979 by Software Arts. It was so popular that many believe that VisiCalc allowed the computer to graduate from a hobby for tech geeks to a serious business tool.



```
  C20                                          C
                                              19

               A       B                   C
  1            1 2.71828  1
  2         1.25 3.49034  2
  3          1.5 4.48169  3
  4         1.75 5.75460  4
  5            2 7.38906  5
  6         2.25 9.48774  6 *
  7          2.5 12.1825  7 *
  8         2.75 15.6426  8 *
  9            3 20.0855  9 **
 10         3.25 25.7903 10 **
 11          3.5 33.1155 11 ***
 12         3.75 42.5211 12 ****
 13            4 54.5982 13 ******
 14         4.25 70.1054 14 ********
 15          4.5 90.0171 15 **********
 16         4.75 115.584 16 *************
 17            5 148.413 17 ****************
 18              --------18
 19              148.413 19Y=EXP(X)
 20            1     .25 20
```

VisiCalc image from atariwiki.org licensed under Creative Commons Share Alike License.

The success of VisiCalc was relatively short-lived in today's terms when compared to juggernauts such as Microsoft Office. In 1983, Lotus Software released the famous Lotus 1-2-3. Lotus 1-2-3 was popular enough to be a significant contributor to the success of the IBM PC. So the idea of giving users a grid-based interface for working with tabular data goes back at least to with VisiCalc — and seems to have been a pretty popular idea.

What About Grid Views for Developers?

Before the .NET Framework was even a thing, there were default Visual Basic and ActiveX datagrid controls. GrapeCity was offering developers datagrids such as VSFlexGrid and TrueDBGrid, and software systems relying on these grids are still in use today.

Modern developers are fortunate to have wide-ranging industry support for datagrids in their applications. There are so many control sets for .NET developers, and all of them include a datagrid control.

Developers can now create a grid-based UI across multiple platforms and application types.

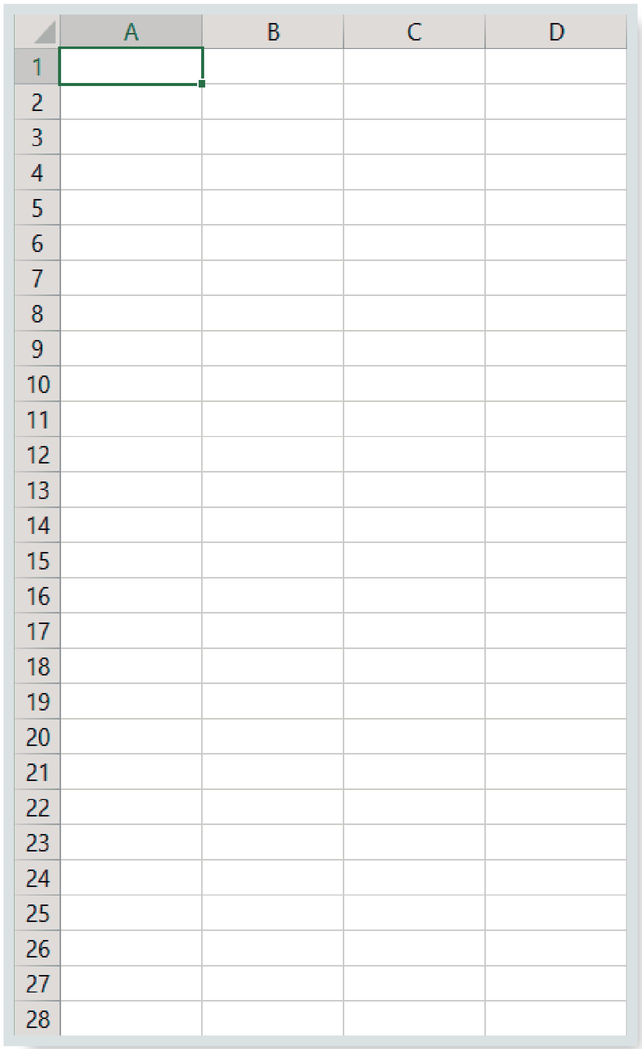
We'll take a look at the options for integrating datagrids into .NET apps. Specifically, we'll start WinForms-based desktop apps, but also take a look at options for web apps using ASP.NET MVC or Blazor. We'll also consider the pros and cons of building your own (extending DataGridView) versus using a commercial datagrid.

What Is a Datagrid?

Many applications focus on data — manipulating data, analyzing data, reporting data — and tabular data is usually displayed in an on-screen grid. It's therefore important to provide to the user as much functionality as possible. If you have ever used any kind of spreadsheet software, you will know why datagrids provide such flexibility and power in data-focused applications.

Regardless of what control set you decide to use, datagrids will be the cornerstone of your application. They provide a rich set of features that allow users to structure their data into something that makes sense — even functionality as simple as sorting and filtering data.

Apart from datagrids available out of the box in the Visual Studio default controls, there are also many open source and commercial controls available. These allow you to integrate datagrids into your applications using just a few lines of code.



Putting Datagrids to Work

As mentioned earlier, if you are developing a data-driven application, it makes sense to use a datagrid. It allows your users to easily make sense of the data they are presented with.

Datagrids offer features such as:

- ✓ Being able to glance over large volumes of data with the scroll of a mouse wheel
- ✓ Sorting the data by a column or other subset of data
- ✓ Selecting and extracting patterns in the data presented
- ✓ Exporting data into easily distributable electronic or paper-based formats

This means that datagrids are not only suitable for displaying data; they are also capable of distributing that data.

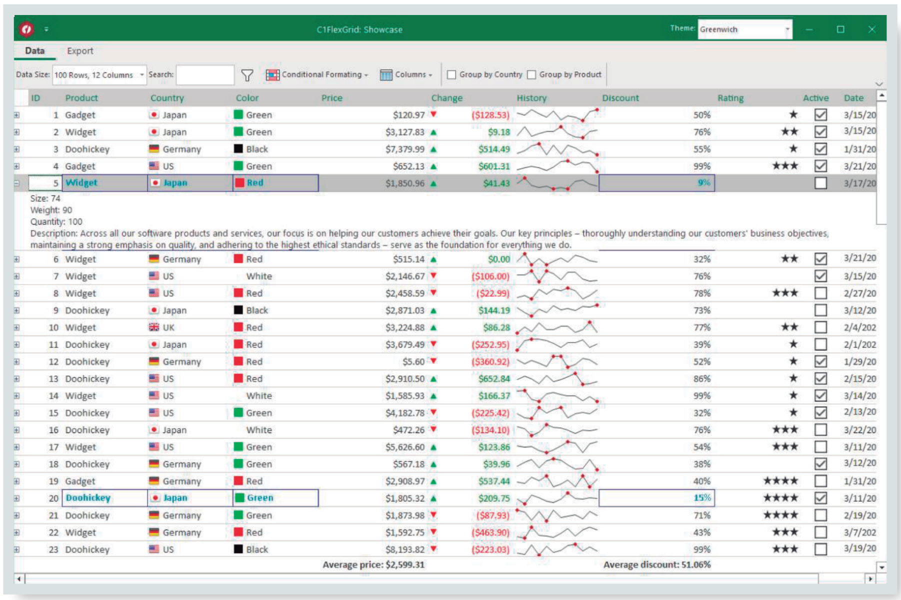
Datagrids in applications are invariably bound to some sort of data source. The source of the data is irrelevant to a datagrid.

Developers can pull data from a variety of different data sources such as:

- ✓ Industry-standard spreadsheets
- ✓ Third-party APIs
- ✓ Queries to locally attached or network databases

This also means users can modify data in place and update the data source, often in real time. Due to the update capabilities, datagrids provide users with important data validation features.

It is safe to say that datagrids and spreadsheet functionality have started to overlap more and more in recent years. Developers expect to be able to structure a datagrid with data, as they are accustomed to doing in a spreadsheet. This could include specific formatting of data cells based on the data represented in the cell. For example, think of a cell with a different font or color based on the data value passing a score or not. Custom cell formats might even include a visual representation of the data it contains through sparklines.



Data Size: 100 Rows, 12 Columns Search: Conditional Formating Columns

ID	Product	Country	Color	Price	Change	History	Discount	Rating	Active	Date
1	Gadget	Japan	🟩 Green	$120.97 ▼	($128.53)		50%	★	☑	3/15/20
2	Widget	Japan	🟩 Green	$3,127.83 ▲	$9.18		76%	★★	☑	3/15/20
3	Doohickey	Germany	■ Black	$7,379.99 ▲	$514.49		55%	★	☑	1/31/20
4	Gadget	US	🟩 Green	$652.13 ▲	$601.31		99%	★★★	☑	3/21/20
5	Widget	Japan	🟥 Red	$1,850.96 ▲	$41.43		9%		☐	3/17/20

Size: 74
Weight: 90
Quantity: 100
Description: Across all our software products and services, our focus is on helping our customers achieve their goals. Our key principles – thoroughly understanding our customers' business objectives, maintaining a strong emphasis on quality, and adhering to the highest ethical standards – serve as the foundation for everything we do.

ID	Product	Country	Color	Price	Change	History	Discount	Rating	Active	Date
6	Widget	Germany	🟥 Red	$515.14 ▲	$0.00		32%	★★	☑	3/21/20
7	Widget	US	White	$2,146.67 ▼	($106.00)		76%		☑	3/15/20
8	Widget	US	🟥 Red	$2,458.59 ▼	($22.99)		78%	★★★	☐	2/27/20
9	Doohickey	Japan	■ Black	$2,871.03 ▲	$144.19		73%		☐	3/12/20
10	Widget	UK	🟥 Red	$3,224.88 ▲	$86.28		77%	★★	☐	2/4/202
11	Doohickey	Japan	🟥 Red	$3,679.49 ▼	($252.95)		39%	★	☐	2/1/202
12	Doohickey	Germany	🟥 Red	$5.60 ▼	($360.92)		52%	★	☑	1/29/20
13	Doohickey	US	🟥 Red	$2,910.50 ▲	$652.84		86%	★	☑	2/15/20
14	Widget	US	White	$1,585.93 ▲	$166.37		99%	★	☑	3/14/20
15	Doohickey	US	🟩 Green	$4,182.78 ▼	($225.42)		32%	★	☑	2/13/20
16	Doohickey	Japan	White	$472.26 ▼	($134.10)		76%	★★★	☐	3/22/20
17	Widget	US	🟩 Green	$5,626.60 ▲	$123.86		54%	★★★	☐	3/11/20
18	Doohickey	Germany	🟩 Green	$567.18 ▲	$39.96		38%		☑	3/12/20
19	Gadget	Germany	🟥 Red	$2,908.97 ▲	$537.44		40%	★★★★	☐	1/31/20
20	Doohickey	Japan	🟩 Green	$1,805.32 ▲	$209.75		15%	★★★★	☑	3/11/20
21	Doohickey	Germany	🟩 Green	$1,873.98 ▼	($87.93)		71%	★★★★	☐	2/19/20
22	Widget	Germany	🟥 Red	$1,592.75 ▼	($463.90)		43%	★★★	☐	3/7/202
23	Doohickey	US	■ Black	$8,193.82 ▼	($223.03)		99%	★★★	☐	3/19/20
				Average price: $2,599.31			Average discount: 51.06%			

Another critical but often overlooked feature is accessibility and keyboard navigation.

At the beginning of my career, I developed an application for entering job cards. I was surprised to see that the users kept their eyes fixed on the paper job card while letting muscle memory do the work when they tabbed between the fields and entered the data. It is often extremely difficult for users to remap their muscle memory when using a new application.

Think back to the first mobile phones with physical keypads. Back then, we typed without looking at the keypad. Muscle memory did all the work for us.

Now imagine rearranging all those keys. It would be extremely difficult for users to type messages. Now imagine having a different keypad layout for each mobile phone. Productivity would plummet as users had to adapt to the change.

This is what makes datagrids so powerful. They function much in the same way that spreadsheets do, and users can easily adapt to the same type of data input across the different devices and platforms. Developers, however, must take care to keep the datagrids that they build into their applications as easily accessible and functional as possible.

Using Datagrids in .NET

Today, developers do not need to code their datagrid controls by hand. The quickest way to incorporate the functionality of a datagrid into your application is to use one that comes out of the box with Visual Studio.

The DataGridView control replaces the legacy DataGrid control in WinForms, and introduces additional functionality to the legacy control. You can still find the DataGrid control, but this is more for maintaining backward compatibility. As you have probably guessed, the DataGridView control displays data in a tabular form, and will display that data irrespective of the data source.

Binding data to the DataGridView control is easily accomplished by setting the DataGridView control's DataSource property. If you have a data source that contains multiple datasets, you use the DataMember property to specify which set of data the DataGridView should bind to.

Developers can also consider implementing one of several open-source datagrid components into their projects. Open source is really big these days — it offers developers a chance to have a direct impact on the future development of open source offerings.

While these approaches to working with data in your application are valid options, I believe that commercial controls, such as GrapeCity's .NET FlexGrid Datagrid, offers some unique advantages. FlexGrid is fast, flexible, and packed with features. It is also available across:

DESKTOP
WinForms
WPF
UWP
ActiveX

WEB
ASP.NET MVC
Blazor
JavaScript

NATIVE MOBILE
Xamarin

LIGHTSWITCH

SILVERLIGHT

With assemblies ranging from 150 KB in ASP.NET MVC to 1.3 MB in WinForms, it has an extremely small footprint. It is also very fast, loading millions of records in seconds.

It includes flexible data binding, multi-format export and import, formatting of presented data, as well as exhaustive samples and customization.

Buy or Build your Datagrid?

As mentioned earlier, there are many open-source offerings to choose from. A commercial offering, however, does provide some definite advantages.

These can be things such as:

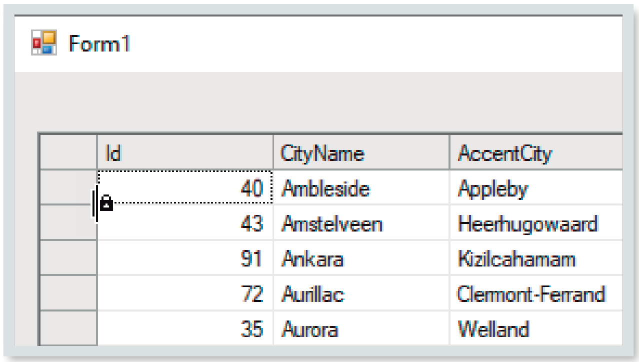
- Dedicated support for implementation as well as developer support
- Single point of contact when requiring support
- QEasier upgrade paths for newer versions

Drawbacks to commercial offerings, however, can include the following:

- Restrictive licensing
- Cost
- Unable to tailor the source code for your needs

It is therefore especially important that you choose a product with strong community support and adoption, and in the case of software controls, features.

Consider the ability to freeze rows in the grid. This comes built-in with GrapeCity's FlexGrid control. Set the `AllowFreezing` property to a column, row, or both — and when you hover your mouse over the grid, an icon displays allowing you to freeze that row or column.



	Id	CityName	AccentCity
	40	Ambleside	Appleby
	43	Amstelveen	Heerhugowaard
	91	Ankara	Kizilcahamam
	72	Aurillac	Clermont-Ferrand
	35	Aurora	Welland

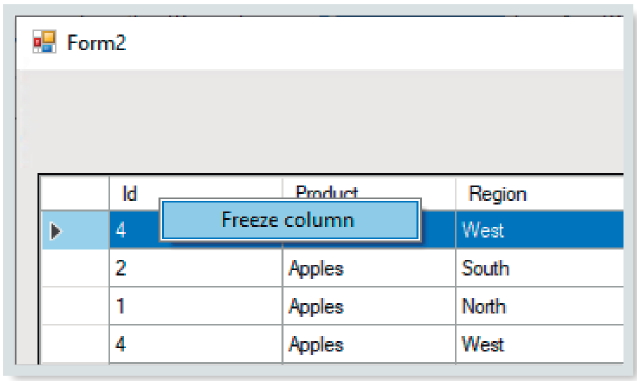
You can set the `AllowFreezing` property in code as follows:

```
dgFlexGrid.AllowFreezing = C1.Win.C1FlexGrid.AllowFreezingEnum.Both
```

Compare this to adding the same functionality to the standard WinForms DataGridView control. Essentially, you would need to add a context menu to your grid. Then you need to select the row by right-clicking the mouse on the grid via the `CellMouseDown` event. And finally you need to enable the freezing of the selected row in the context menu click event.

```
private void freezeColumnToolStripMenuItem_Click(object sender, EventArgs e)
{
   dgWinGrid.Rows[dgWinGrid.CurrentCell.RowIndex].Frozen = true;
}

private void dgWinGrid_CellMouseDown(object sender,
   DataGridViewCellMouseEventArgs e)
{
   var dataGrid = (DataGridView)sender;
   if (e.Button == MouseButtons.Right && e.RowIndex != -1)
   {
       var row = dataGrid.Rows[e.RowIndex];
       dataGrid.CurrentCell = row.Cells[e.ColumnIndex == -1 ? 1 : e.ColumnIndex];
       row.Selected = true;
       dataGrid.Focus();
   }
}
```

Looking at the way this implementation works in the DataGridView (shown below), it is clear that FlexGrid's implementation (the previous Form1 example above) is visually more pleasing and far simpler to implement.



	Id	Product	Region
▶	4		West
	2	Apples	South
	1	Apples	North
	4	Apples	West

You have to do a lot of heavy lifting with the standard DataGridView control when trying to add something as simple as row freezing. With GrapeCity's .NET FlexGrid Datagrid control, this functionality (and a lot more) comes baked in.

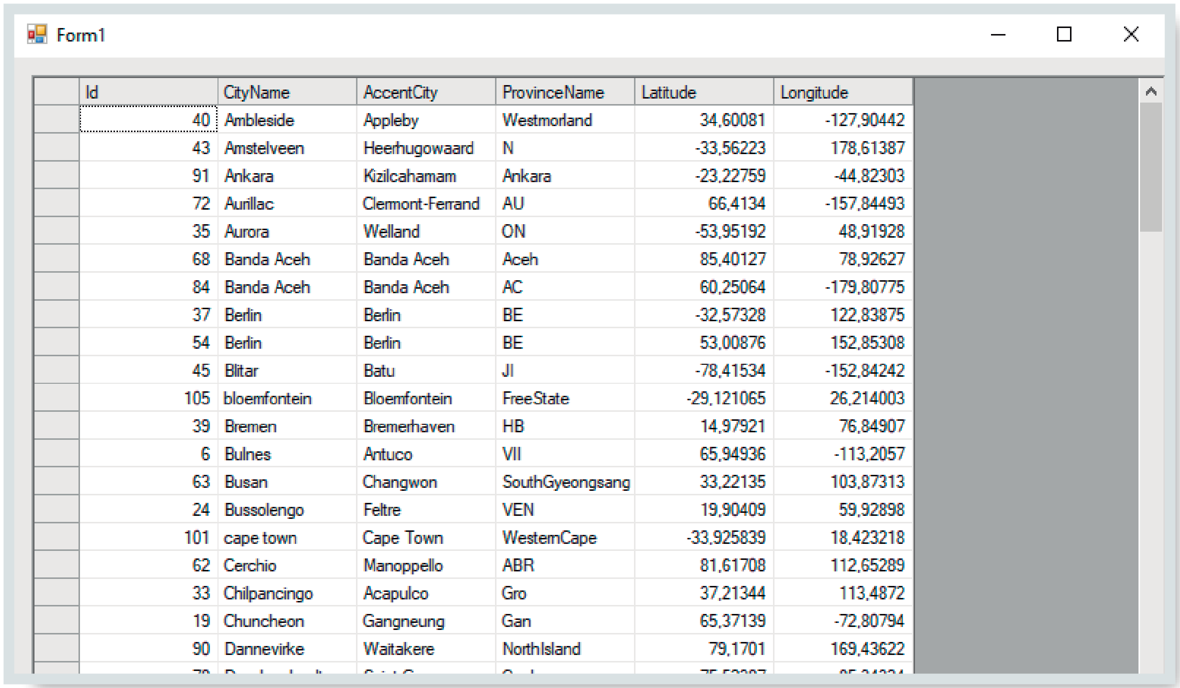
Trying to do this with a standard DataGridView when very polished versions of these features come built in with a commercial control is not a good use of developers' time. Googling for a solution took me longer than writing the code to implement grouping, aggregation, and collapsible rows with FlexGrid.

Incorporating Datagrids Into a .NET App

Using a FlexGrid control in your WinForms application is easy. If you are familiar with the standard DataGridView control that ships with Visual Studio, you will have no trouble using GrapeCity's .NET FlexGrid Datagrid control.

The example above loaded some data into the standard DataGridView, then we added a bit of customization code just to get the row freezing. The following example shows off a sample project I have created for illustrating .NET FlexGrid Datagrid in apps. Here you can see the same data in the FlexGrid control, and we'll see how easy it is to add more sophisticated control over the display and visualizations with just a few lines of code.

The application loads a JSON file with a list of cities into a datagrid. When the application is run, the list is displayed in the grid.



Form1

Id	CityName	AccentCity	ProvinceName	Latitude	Longitude
40	Ambleside	Appleby	Westmorland	34,60081	-127,90442
43	Amstelveen	Heerhugowaard	N	-33,56223	178,61387
91	Ankara	Kizilcahamam	Ankara	-23,22759	-44,82303
72	Aurillac	Clermont-Ferrand	AU	66,4134	-157,84493
35	Aurora	Welland	ON	-53,95192	48,91928
68	Banda Aceh	Banda Aceh	Aceh	85,40127	78,92627
84	Banda Aceh	Banda Aceh	AC	60,25064	-179,80775
37	Berlin	Berlin	BE	-32,57328	122,83875
54	Berlin	Berlin	BE	53,00876	152,85308
45	Blitar	Batu	JI	-78,41534	-152,84242
105	bloemfontein	Bloemfontein	FreeState	-29,121065	26,214003
39	Bremen	Bremerhaven	HB	14,97921	76,84907
6	Bulnes	Antuco	VII	65,94936	-113,2057
63	Busan	Changwon	SouthGyeongsang	33,22135	103,87313
24	Bussolengo	Feltre	VEN	19,90409	59,92898
101	cape town	Cape Town	WesternCape	-33,925839	18,423218
62	Cerchio	Manoppello	ABR	81,61708	112,65289
33	Chilpancingo	Acapulco	Gro	37,21344	113,4872
19	Chuncheon	Gangneung	Gan	65,37139	-72,80794
90	Dannevirke	Waitakere	NorthIsland	79,1701	169,43622

The code used to bind the data to the grid is as follows:

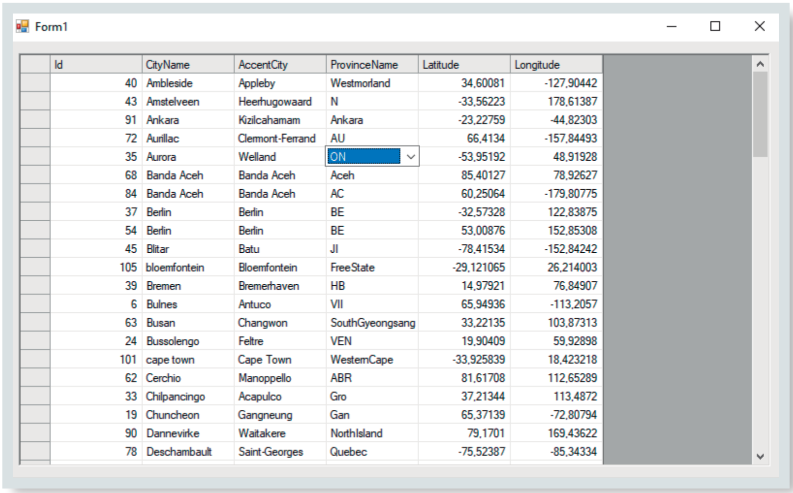
```
private void BindCityGrid()
{
    var tbl = Cities.CreateTable();
    dgFlexGrid.DataSource = tbl;
}
```

That is all there is to it. I have an extension method that creates a DataTable from my list of cities and sets the FlexGrid's `DataSource` property to DataTable.

Another interesting thing to note is the City model. The code looks like this:

```
public class City : IModelObject
{
    public int Id { get; set; }
    public string CityName { get; set; }
    public string AccentCity { get; set; }
    public Province ProvinceName { get; set; }
    public double Latitude { get; set; }
    public double Longitude { get; set; }
}
```

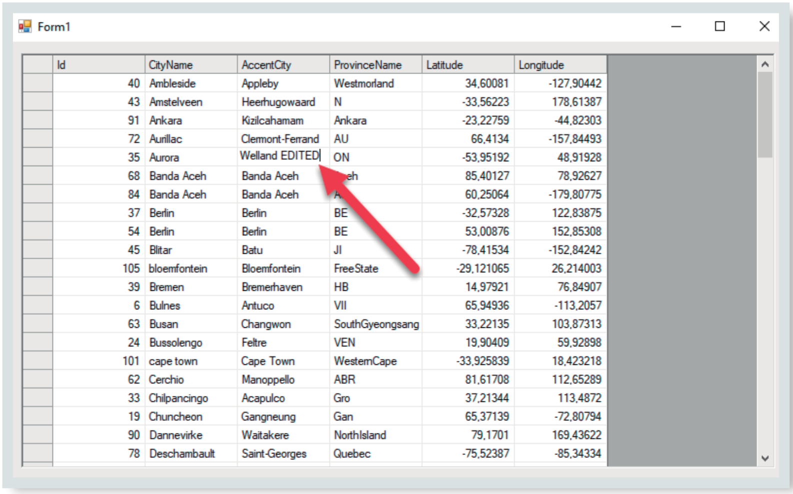
As you can see, an enum is defined as the type of the `ProvinceName` property. Because this property is an enum, FlexGrid knows that it needs to display the corresponding cell as a dropdown list.



Form1

	Id	CityName	AccentCity	ProvinceName	Latitude	Longitude
	40	Ambleside	Appleby	Westmorland	34,60081	-127,90442
	43	Amstelveen	Heerhugowaard	N	-33,56223	178,61387
	91	Ankara	Kizilcahamam	Ankara	-23,22759	-44,82303
	72	Aurillac	Clermont-Ferrand	AU	66,4134	-157,84493
	35	Aurora	Welland	ON	-53,95192	48,91928
	68	Banda Aceh	Banda Aceh	Aceh	85,40127	78,92627
	84	Banda Aceh	Banda Aceh	AC	60,25064	-179,80775
	37	Berlin	Berlin	BE	-32,57328	122,83875
	54	Berlin	Berlin	BE	53,00876	152,85308
	45	Blitar	Batu	JI	-78,41534	-152,84242
	105	bloemfontein	Bloemfontein	FreeState	-29,121065	26,214003
	39	Bremen	Bremerhaven	HB	14,97921	76,84907
	6	Bulnes	Antuco	VII	65,94936	-113,2057
	63	Busan	Changwon	SouthGyeongsang	33,22135	103,87313
	24	Bussolengo	Feltre	VEN	19,90409	59,92898
	101	cape town	Cape Town	WesternCape	-33,925839	18,423218
	62	Cerchio	Manoppello	ABR	81,61708	112,65289
	33	Chilpancingo	Acapulco	Gro	37,21344	113,4872
	19	Chuncheon	Gangneung	Gan	65,37139	-72,80794
	90	Dannevirke	Waitakere	NorthIsland	79,1701	169,43622
	78	Deschambault	Saint-Georges	Quebec	-75,52387	-85,34334

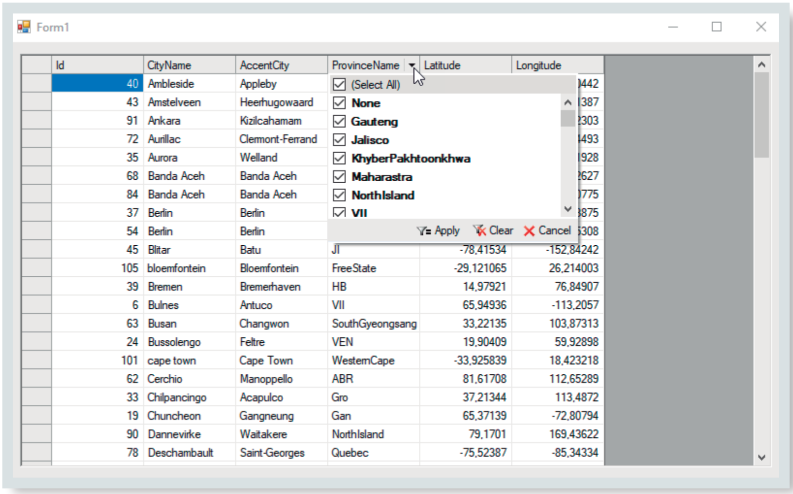
Editing the data in the grid happens directly inside FlexGrid.



Form1

	Id	CityName	AccentCity	ProvinceName	Latitude	Longitude
	40	Ambleside	Appleby	Westmorland	34,60081	-127,90442
	43	Amstelveen	Heerhugowaard	N	-33,56223	178,61387
	91	Ankara	Kizilcahamam	Ankara	-23,22759	-44,82303
	72	Aurillac	Clermont-Ferrand	AU	66,4134	-157,84493
	35	Aurora	Welland EDITED	ON	-53,95192	48,91928
	68	Banda Aceh	Banda Aceh	eh	85,40127	78,92627
	84	Banda Aceh	Banda Aceh	A	60,25064	-179,80775
	37	Berlin	Berlin	BE	-32,57328	122,83875
	54	Berlin	Berlin	BE	53,00876	152,85308
	45	Blitar	Batu	JI	-78,41534	-152,84242
	105	bloemfontein	Bloemfontein	FreeState	-29,121065	26,214003
	39	Bremen	Bremerhaven	HB	14,97921	76,84907
	6	Bulnes	Antuco	VII	65,94936	-113,2057
	63	Busan	Changwon	SouthGyeongsang	33,22135	103,87313
	24	Bussolengo	Feltre	VEN	19,90409	59,92898
	101	cape town	Cape Town	WesternCape	-33,925839	18,423218
	62	Cerchio	Manoppello	ABR	81,61708	112,65289
	33	Chilpancingo	Acapulco	Gro	37,21344	113,4872
	19	Chuncheon	Gangneung	Gan	65,37139	-72,80794
	90	Dannevirke	Waitakere	NorthIsland	79,1701	169,43622
	78	Deschambault	Saint-Georges	Quebec	-75,52387	-85,34334

Out of the box, without adding any code, you can apply filters to the FlexGrid data.



Form1

	Id	CityName	AccentCity	ProvinceName	Latitude	Longitude
	40	Ambleside	Appleby			442
	43	Amstelveen	Heerhugowaard			387
	91	Ankara	Kizilcahamam			303
	72	Aurillac	Clermont-Ferrand			493
	35	Aurora	Welland			928
	68	Banda Aceh	Banda Aceh			627
	84	Banda Aceh	Banda Aceh			775
	37	Berlin	Berlin			875
	54	Berlin	Berlin			308
	45	Blitar	Batu	JI	-78,41534	-152,84242
	105	bloemfontein	Bloemfontein	FreeState	-29,121065	26,214003
	39	Bremen	Bremerhaven	HB	14,97921	76,84907
	6	Bulnes	Antuco	VII	65,94936	-113,2057
	63	Busan	Changwon	SouthGyeongsang	33,22135	103,87313
	24	Bussolengo	Feltre	VEN	19,90409	59,92898
	101	cape town	Cape Town	WesternCape	-33,925839	18,423218
	62	Cerchio	Manoppello	ABR	81,61708	112,65289
	33	Chilpancingo	Acapulco	Gro	37,21344	113,4872
	19	Chuncheon	Gangneung	Gan	65,37139	-72,80794
	90	Dannevirke	Waitakere	NorthIsland	79,1701	169,43622
	78	Deschambault	Saint-Georges	Quebec	-75,52387	-85,34334

As you can see, with a minimal amount of code, we can create a datagrid with plenty of built-in functionality.

Additional Datagrid Features

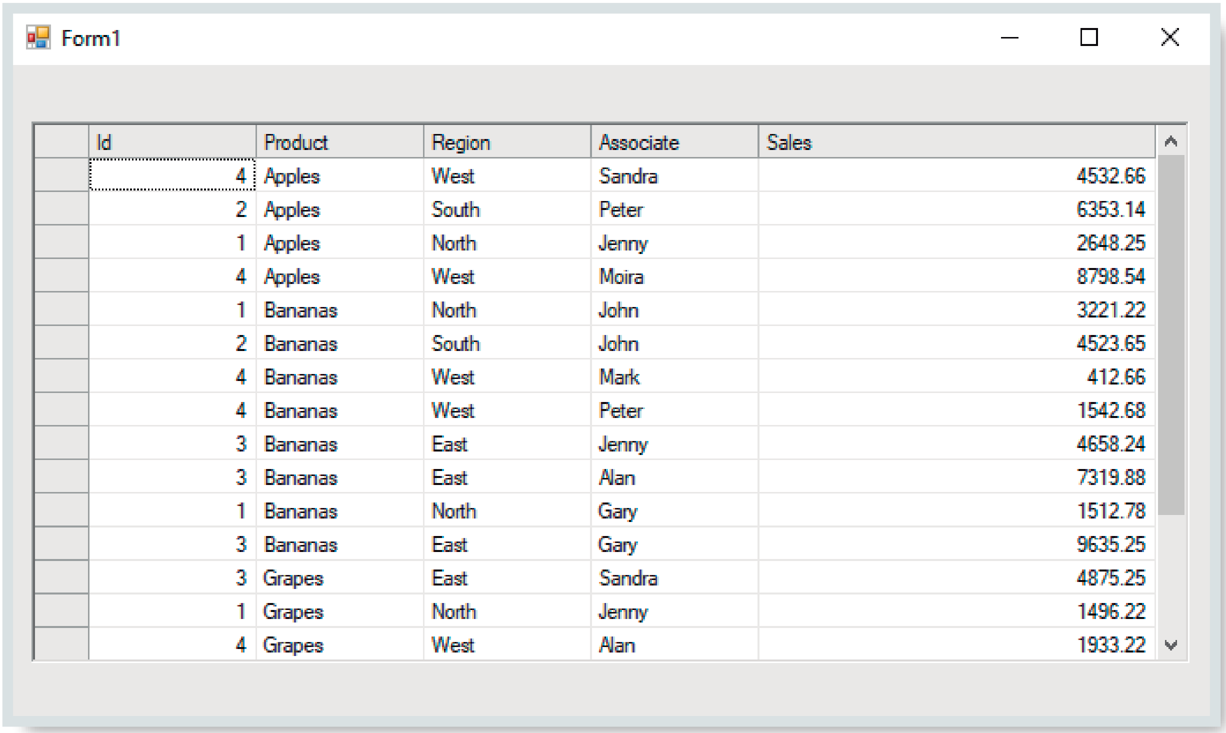
GrapeCity's .NET FlexGrid Datagrid control allows developers to filter, group, and sort their data, as well as offers the ability to handle right-to-left languages with the globalized FlexGrid. It also supports Excel-like editing, expandable detail rows, rapidly updating grid info such as stock prices, built-in formatting, and real-time user input validation.

You can enable users to pin columns, freeze cells (think Excel row and column freezing, as demonstrated earlier), merge cells, perform virtual scrolling, as well as use the TransposedGrid extension. The latter allows you to represent columns as data items and rows as item properties. Think of those sites that offer comparison grids between products. This is what the TransposedGrid extension allows you to do.

On mobile devices, developing applications using Xamarin, the grid is responsive and feature-rich.

Another "I'm sold" feature of FlexGrid is the ability to export selected data to Excel or PDF files. Even more impressive is the ability to import data from an Excel file. FlexGrid data can also be printed with full support for page selection and many more printing options.

FlexGrid offers many easy-to-implement features. Consider the sales data for products in the following grid.

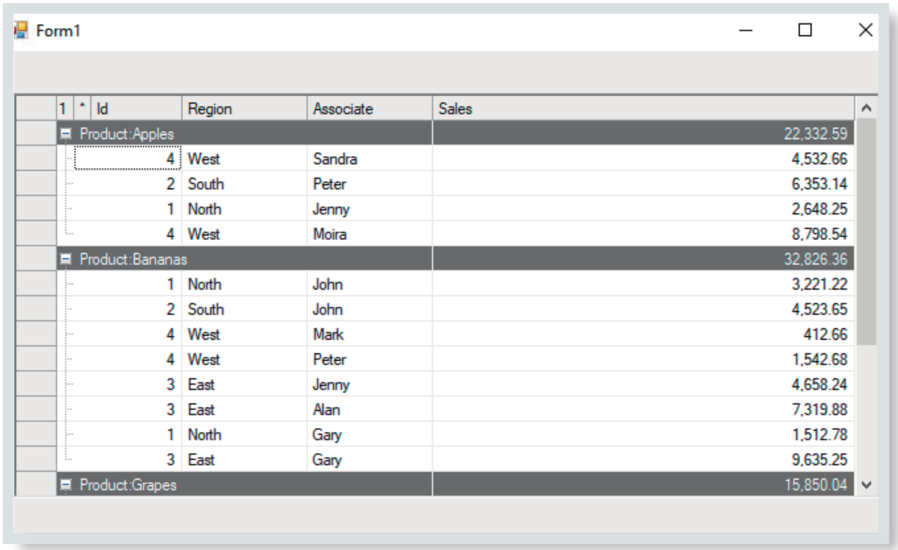


Id	Product	Region	Associate	Sales
4	Apples	West	Sandra	4532.66
2	Apples	South	Peter	6353.14
1	Apples	North	Jenny	2648.25
4	Apples	West	Moira	8798.54
1	Bananas	North	John	3221.22
2	Bananas	South	John	4523.65
4	Bananas	West	Mark	412.66
4	Bananas	West	Peter	1542.68
3	Bananas	East	Jenny	4658.24
3	Bananas	East	Alan	7319.88
1	Bananas	North	Gary	1512.78
3	Bananas	East	Gary	9635.25
3	Grapes	East	Sandra	4875.25
1	Grapes	North	Jenny	1496.22
4	Grapes	West	Alan	1933.22

This data doesn't make much sense to a user. With FlexGrid, we can easily introduce data grouping and aggregation. Imagine that we wanted to see the total sales per product. Adding the following code will accomplish this.

```
private void Group()
{
    //Add group
    dgFlexGrid.GroupDescriptions =
        new GroupDescription[] { new GroupDescription("Product") };
    //Showing aggregate(sum) on the group header rows
    var col = dgFlexGrid.Cols[5];
    col.Aggregate = AggregateEnum.Sum;
    col.Format = "N2";

    //Setting grid's AllowMerging property to Nodes so that the group header
    //content can spill into adjacent empty cells
    dgFlexGrid.AllowMerging = AllowMergingEnum.Nodes;

    //Setting HideGroupedColumns property to true
    //in order to hide the grouped columns
    dgFlexGrid.HideGroupedColumns = true;

    //Customizing the string which is displayed on the group headers
    dgFlexGrid.GroupHeaderFormat = "{name}:{value}";

    //Customizing the appearance of the outline tree
    dgFlexGrid.Tree.Style = TreeStyleFlags.CompleteLeaf;
}
```

Run the application again: here is your useful, grouped data.



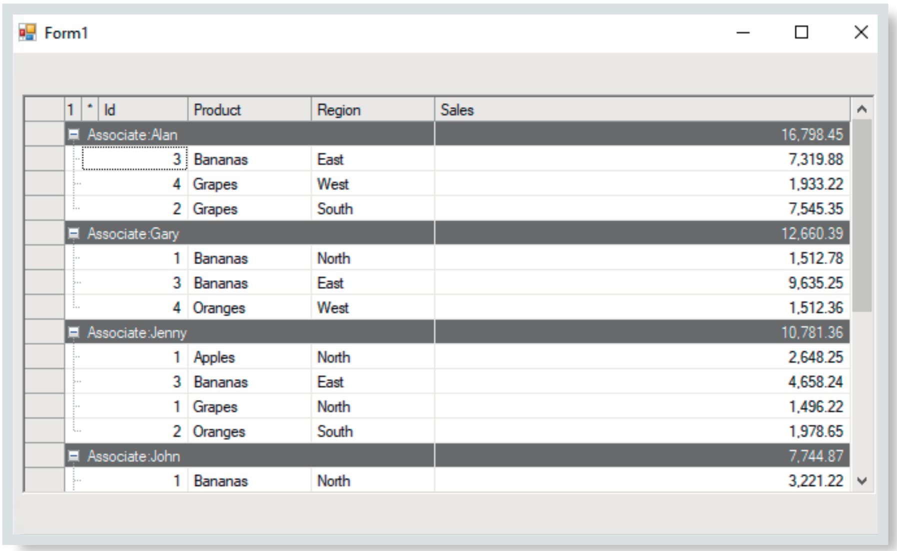
Form1

1	* Id	Region	Associate	Sales
Product:Apples				22,332.59
	4	West	Sandra	4,532.66
	2	South	Peter	6,353.14
	1	North	Jenny	2,648.25
	4	West	Moira	8,798.54
Product:Bananas				32,826.36
	1	North	John	3,221.22
	2	South	John	4,523.65
	4	West	Mark	412.66
	4	West	Peter	1,542.68
	3	East	Jenny	4,658.24
	3	East	Alan	7,319.88
	1	North	Gary	1,512.78
	3	East	Gary	9,635.25
Product:Grapes				15,850.04

Want to see the sales by associate? Just change the group description from “Product” to “Associate”.

```
//Add group
dgFlexGrid.GroupDescriptions =
    new GroupDescription[] { new GroupDescription("Associate") };
```

The data is now grouped differently.



1	*	Id	Product	Region	Sales
Associate:Alan					16,798.45
		3	Bananas	East	7,319.88
		4	Grapes	West	1,933.22
		2	Grapes	South	7,545.35
Associate:Gary					12,660.39
		1	Bananas	North	1,512.78
		3	Bananas	East	9,635.25
		4	Oranges	West	1,512.36
Associate:Jenny					10,781.36
		1	Apples	North	2,648.25
		3	Bananas	East	4,658.24
		1	Grapes	North	1,496.22
		2	Oranges	South	1,978.65
Associate:John					7,744.87
		1	Bananas	North	3,221.22

With only a few lines of code, we enabled grouping of data, calculation of totals per grouped item, and collapsible rows.

FlexGrid for the Web

Another advantage of using datagrid controls like those from GrapeCity is that they give you the flexibility to adapt your application to other frameworks or expand from desktop platforms to the web.

For example, WPF has its own DataGrid control with limited features. GrapeCity's FlexGrid for WPF, while similar in features to the WinForms version, has been designed to take advantage of the Xaml platform. For example, it uses CellTemplates instead of OwnerDraw.

Many teams, however, are embracing web technologies for application development, enabling them to use a common, familiar set of languages and frameworks across a variety of platforms.

But you quickly run into the same problem as .NET datagrids: default datagrid implementations are either simplistic or missing altogether. For example, the popular Blazor framework has no datagrid component in the toolbox, so developers only have a basic HTML table! GrapeCity FlexGrid for Blazor is essential if you want any sophisticated datagrid features.

All of the features we've discussed so far for .NET datagrids are also supported by GrapeCity ASP.NET MVC FlexGrid for .NET-based web apps, and the same features are also available in Wijmo FlexGrid if your development is more focused on JavaScript.

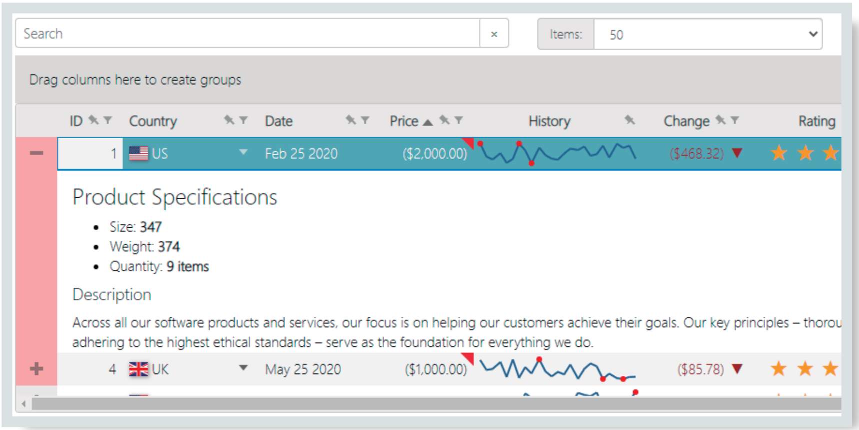
Of the JavaScript, ASP.NET MVC, and Blazor options, which is best for your application? In many cases your existing development codebase and application architecture will dictate suggest which GrapeCity datagrid will integrate most smoothly with your existing code, development workflows, and knowledge.

If you know you need datagrids in your application, but are unsure about what web platform or framework to use — and which GrapeCity datagrid component will be most useful — we'll break it down into some best practices and use cases that may help you choose a path forward.

ASP.NET MVC (or better still, Razor Pages) is for server-side HTML generation. Blazor is commonly used for Single Page Applications (SPA) — web applications that run in the browser. They perform different roles for different requirements. Blazor is not intended as a replacement for ASP.NET MVC or Razor Pages. It is intended to provide a solution for C# programmers who want to produce Angular/React style applications, but don't want to have to learn a lot of new tools.

A pure JavaScript solution is not usually running on the server — it runs in the browser as part of the web page — so there's less security, and more dependent on data services which may need to be developed using ASP.NET Core anyway. On the other hand, JavaScript controls can be easily integrated into almost any web application regardless of the server-side architecture or language.

Let's take a look at a brief example — and note that this same app could be built with FlexGrid for either ASP.NET MVC or any JavaScript framework.



Search × Items: 50

Drag columns here to create groups

	ID	Country	Date	Price ▲	History	Change	Rating
−	1	US	Feb 25 2020	($2,000.00)		($468.32) ▼	★★★

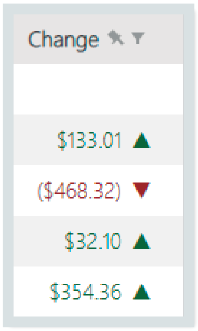
Product Specifications

- Size: 347
- Weight: 374
- Quantity: 9 items

Description

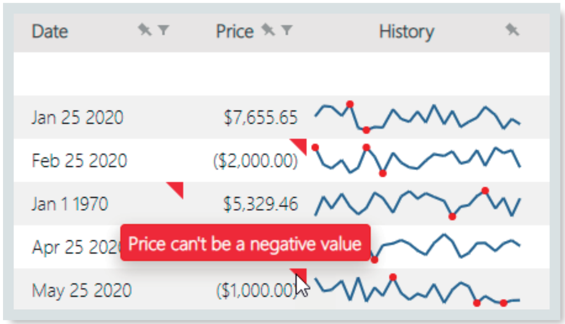
Across all our software products and services, our focus is on helping our customers achieve their goals. Our key principles – thorou
adhering to the highest ethical standards – serve as the foundation for everything we do.

	ID	Country	Date	Price ▲	History	Change	Rating
+	4	UK	May 25 2020	($1,000.00)		($85.78) ▼	★★★



Change
$133.01 ▲
($468.32) ▼
$32.10 ▲
$354.36 ▲

Data validation can also be incorporated into the DataGrid.



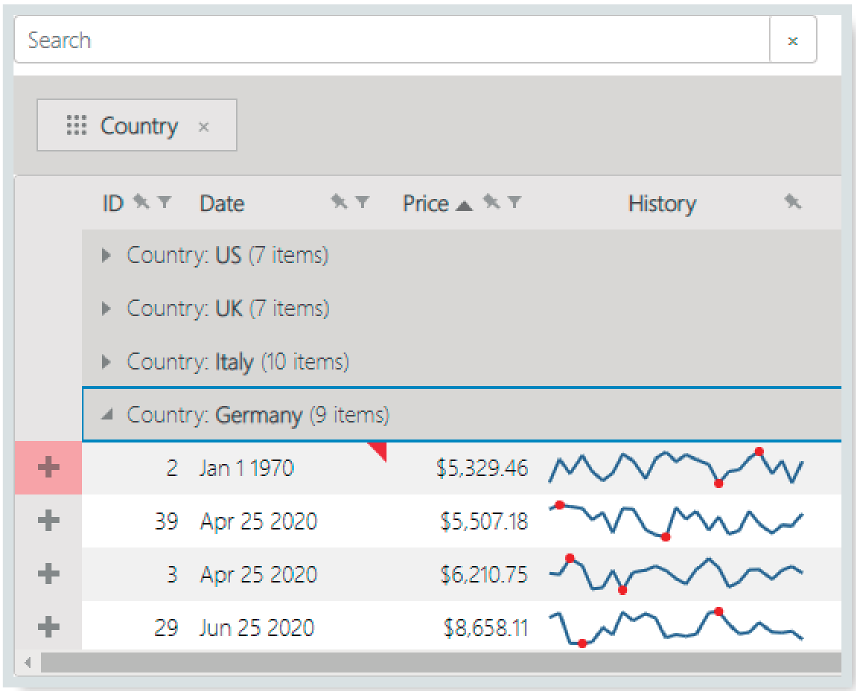
Date	Price	History
Jan 25 2020	$7,655.65	
Feb 25 2020	($2,000.00)	
Jan 1 1970	$5,329.46	
Apr 25 202		
May 25 2020	($1,000.00)	

Price can't be a negative value

From the onset, you will notice that rows can be collapsed or expanded as needed. Expanded rows display detailed information. Cell formatting can be based on the data these cells contain, as illustrated in the image below.

In the above image, the Price value can't be negative (negative values are indicated by the round brackets around the price). As you can see, the validation flagged two price values and a Date value that is too far in the past.

Grouping data is easily accomplished by dragging one or more columns to the header section of the grid. Data can also be exported to Excel and PDF directly from the grid.



Search ×

Country ×

	ID	Date	Price ▲	History
	▸ Country: **US** (7 items)			
	▸ Country: **UK** (7 items)			
	▸ Country: **Italy** (10 items)			
	◢ Country: **Germany** (9 items)			
+	2	Jan 1 1970	$5,329.46	
+	39	Apr 25 2020	$5,507.18	
+	3	Apr 25 2020	$6,210.75	
+	29	Jun 25 2020	$8,658.11	

Incorporating Data into Datagrids

How you bring data into your .NET application depends on your circumstances and requirements. In the WinForms application we've discussed earlier, I simply consumed a local JSON file with the cities' data.

Let's have a quick look at how the data logic is structured. You will see that the WinForms solution is split into three separate projects. These are:

- GrapeCity – contains the WinForms UI
- GrapeCity.Core
- GrapeCity.Data

SEPARATION OF CONCERNS

Keeping data in a separate project makes it really easy to separate your application's concerns from one another. It also allows developers to swap out the consumer (a DataGridView for example) from the built-in control to something like the FlexGrid control without having to change anything on the data end.

Best of all, this approach ensures that you're maintaining your data grid's state in your own code, instead of relying on FlexGrid as your application's sole source of truth. In most applications you'll need to validate and save the data in your grid — so it's helpful to build your application accordingly by keeping your data and UI code separate.

Several architectural patterns can help you enforce this separation:

MODEL-VIEW-PRESENTER (MVP)

MVP uses a Presenter class as an intermediary between your app's data model and UI code. UI code lives in the View, and when changes occur, the View calls the Presenter. The Presenter decides if and how to update the data.

Similarly, the application's data model code and call the Presenter when data changes. The Presenter then decides how to update the UI to match the data changes.

MODEL-VIEW-VIEWMODEL (MVVM)

MVVM is similar to MVP, but typically uses a View Model with automatic data binding instead of a Presenter, which must manually make UI updates. MVVM is very popular in WPF apps, because many built-in and third-party WPF components support automatic data binding.

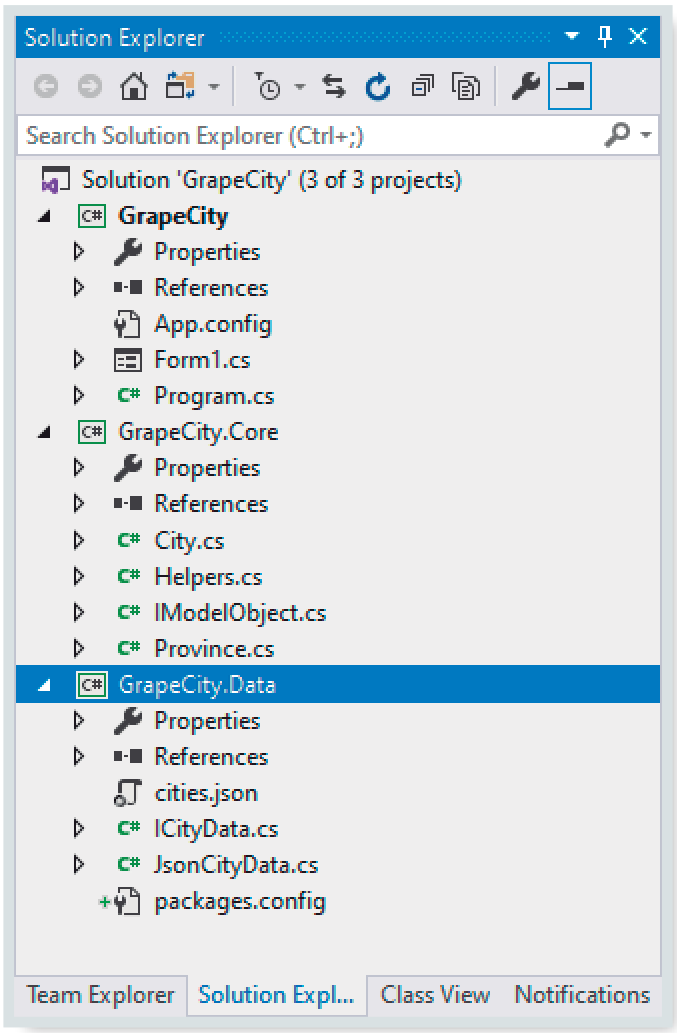
In addition, open source frameworks like MVVMCross make it easy to share you application's data model and application code between platforms such WPF, UWP, Xamarin.

DIRECT MODEL BINDING

FlexGrid lets you bind your grid's data to your own data class. While this doesn't separate your app's UI and data concerns as much as MVP or MVVM, it still gives you an automatically-updated data model that will always be ready for validation or serialization.

Since it's the simplest approach, let's take a look at how model binding works in practice using the Winforms project we described earlier.

The solution would look something like the following in the Solution Explorer.



The GrapeCity.Data project is where the data logic resides. In it, I have a class called `JsonCityData`. This class, as the name suggests, is responsible for providing my application with the city data read from a JSON store (whether from a file or an API request).

```
public class JsonCityData : ICityData
{
}
```

This class implements the `ICityData` interface. The Interface simply tells my data what it should implement.

```
public interface ICityData
{
    IEnumerable<City> GetCityByName(string name);
    City GetById(int Id);
}
```

The `JsonCityData` class then provides the implementation of this interface. In the constructor of my WinForms application, I tell it that it should expect an object of `ICityData` as a parameter.

```
private readonly ICityData _dataSource;

public Form1(ICityData dataSource)
{
    InitializeComponent();
    _dataSource = dataSource;
    LoadCityData();
    BindCityGrid();
}
```

I am now free to pass any object that inherits from `ICityData` to the Form's constructor in the Program.cs file.

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);

    var jsonDataSource = new JsonCityData();

    Application.Run(new Form1(jsonDataSource));
}
```

In this example, the application uses a JSON data source.

```
var jsonDataSource = new JsonCityData();
```

I can easily swap it out for an SQL data source. All I need to do is provide the implementation in the SQL class.

```
var sqlDataSource = new SqlCityData();
```

If I need to read data from Excel, I can pass it as an instance of the Excel class.

```
var excelDataSource = new ExcelCityData();
```

Should the data be pulled from an API, I can use an instance of a class that takes care of that.

```
var restDataSource = new RESTCityData();
```

In each of these class instantiations — `JsonCityData`, `SqlCityData`, `ExcelCityData`, and `RESTCityData` — I will create a specific class to contain an implementation of the `ICityData` Interface.

- The `SqlCityData` class will contain logic specific to reading the data from an SQL Server database.

- The `ExcelCityData` class will contain logic specific to reading data from an Excel spreadsheet stored somewhere on a network drive.

- The `RESTCityData` will contain the logic for performing an API call to pull the data into the application.

As for the application, it just expects a data source that implements `ICItyData`. It already knows what to do with the data. How that data is passed to it is up to the specific class implementation of the `ICityData` Interface.

Loading Excel Data into a .NET Datagrid

One of the FlexGrid features that stands out is its ability to import data from Excel. Many times I have heard developers wanting to roll their own, as it were. This becomes quite complex when dealing with Excel data. Go ahead and create an Excel file, and rename the file extension from .xlsx to .zip — you can unzip it as you would any other archive.

Have a look at the unzipped files and folders, you will find the metadata of the data contained in the Excel file. You will also find the data itself, stored in one of the XML files. You could definitely write your own parser that could process Excel files in this way and pull out the data you need. Depending what you need to do with the data, this parser can get rather complex.

Developers generally opt to use a third-party library (open source or commercial) for parsing Excel data, rather than spend the time and money writing their own parser.

There are a lot of resources you might find useful when implementing this functionality in your applications:

- For ASP.NET MVC, have a look at the ComponentOne ASP.NET MVC FlexGridExcel Import And Export demo.
- To see some sample code for a C# or VB.NET WinForms app, download the FlexGrid and Excel sample projects.
- The ASP.NET MVC Excel Import and Export tutorial walks you through the code for implementing Excel import/export functionality.
- GrapeCity provides a fantastic resource of sample code.

Next Steps

In this article, we briefly discussed what FlexGrid can do for your .NET applications. I encourage you to get the source code from GitHub and experiment with the various options. Try adding different data sources. Use the cities.json file and expand it to include more cities. See how fast the grid loads with a much longer list of cities.

To learn more about the FlexGrid control, as well as see some Wijmo examples, have a look at https://www.grapecity.com/wijmo/demos/Grid/Overview/purejs

There is a fantastic GrapeCity Support Portal available to developers that includes Forums, Blogs, and Documentation.

Personally, I learn easier when watching a video. Here, GrapeCity has you covered too with a rich collection of video content to get you going at the GrapeCity Videos page.

To get in touch with the support team to open a ticket or to access the community forums, head on over to https://www.grapecity.com/support/contact.

There is such a wealth of documentation, video tutorials, community support and dedicated technical support available, that developers using GrapeCity products will not be left in the lurch when it comes to integrating a control such as the FlexGrid into their own projects.

This makes the choice to use these controls a much easier one due to the increased developer confidence. Give the FlexGrid control a try today. I know that you will love what you see.