

RadGridView For Silverlight and WPF

This tutorial will introduce the RadGridView control, part of the Telerik suite of XAML controls.

Setting Up The Project

To begin, open Visual Studio and click on the Telerik menu option. Under *Rad Controls For Silverlight* click on *Create New Telerik Project*. Name your project, accept Silverlight 5 and in the Project Configuration Wizard dialog check *GridView* (notice that the dependent references are automatically checked as well), as shown in figure 1.

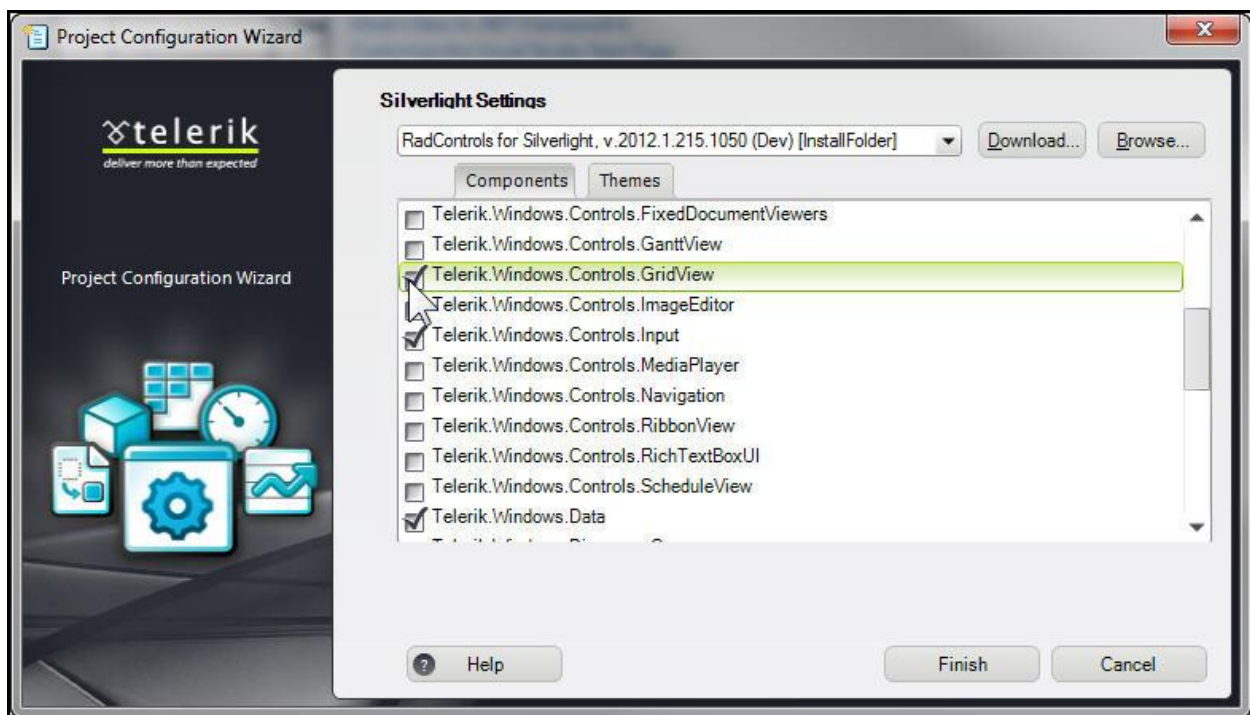


Figure 1

When you click ok, the necessary assemblies are added to the References as shown in figure 1

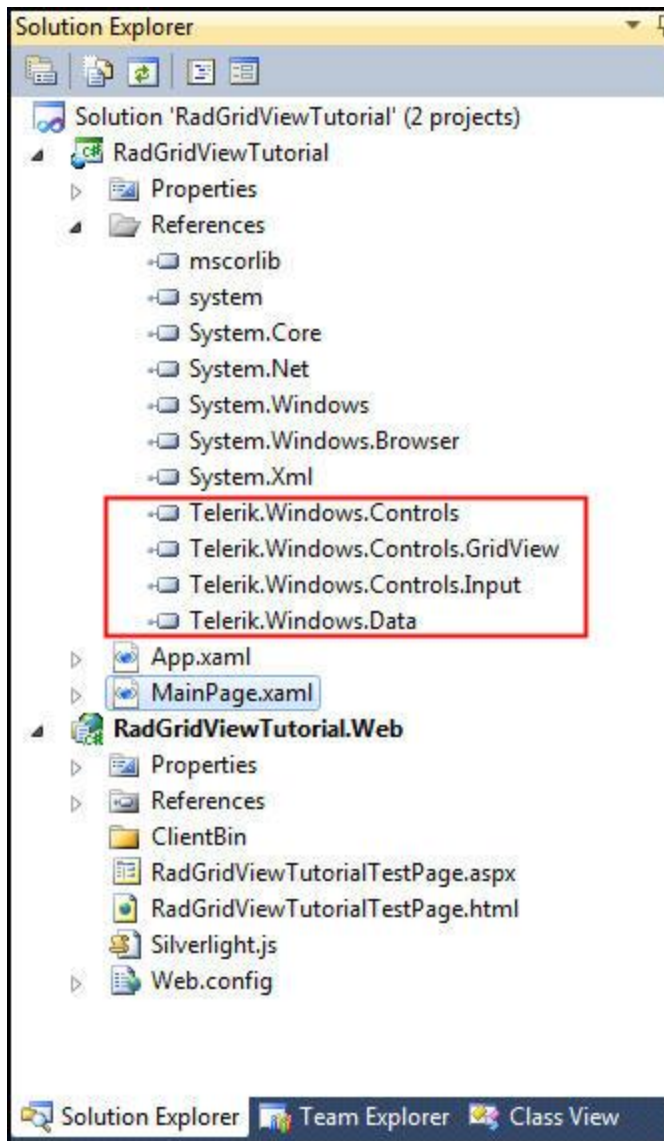


Figure 2

Your application will open to `MainPage.xaml` and, thanks to the Telerik Visual Studio extensions, the namespace `telerik` will already have been created in the XAML heading.

```
<UserControl x:Class="RadBarcode.GettingStarted.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
```

Drag RadGridView from the toolbox onto the design surface. If you take a look at the XAML, you'll see that Visual Studio set the alignments and margin to mark the placement of the Gridview where you happened to drop it,

```
<telerik:RadGridView HorizontalAlignment="Left" Margin="77,46,0,0" Name="radGridView1"
VerticalAlignment="Top" />
```

We can clean that up by right clicking on the GridView and choosing Reset Layout -> All, as shown in figure 3,

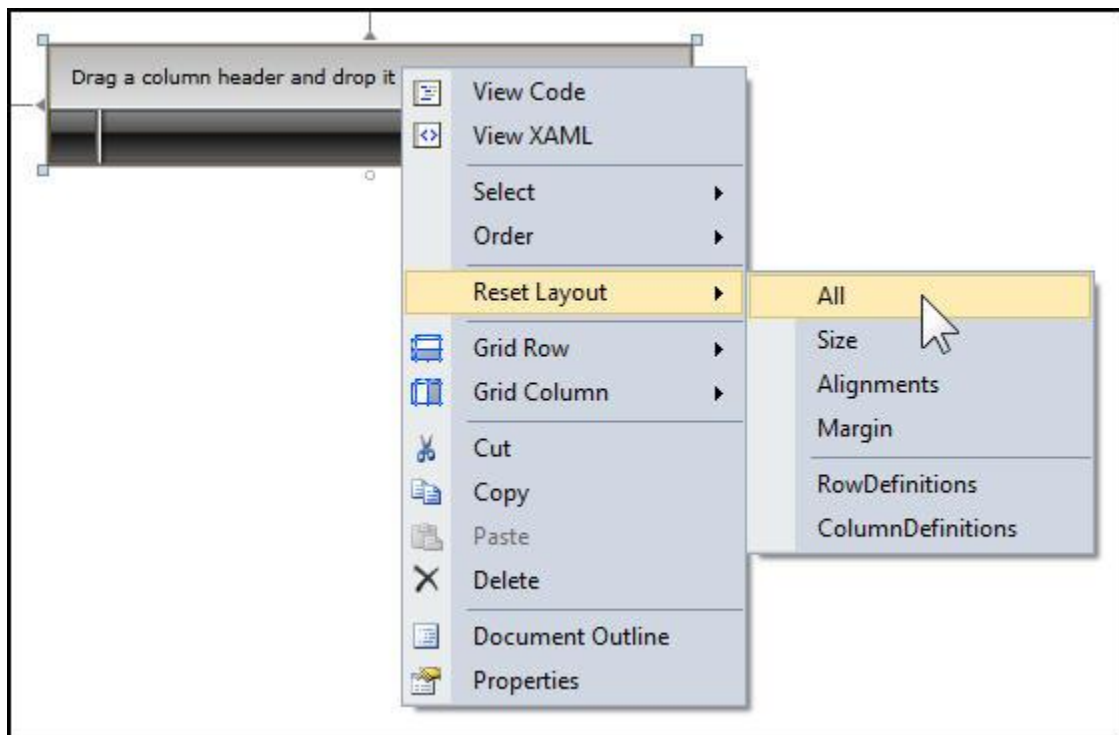


Figure 3

Jump over to the code behind, and in the MainPage.xaml.cs file, below the closing brace for the class, let's create a demonstration data class,

```
public class DemoClass
{
    public int ID { get; set; }
    public string Name { get; set; }
    public string Company { get; set; }
    public bool IsComplete { get; set; }
    public DateTime DueDate { get; set; }
}
```

In the constructor, initialize a collection of these class instances,

```
var myDemoClasses = new ObservableCollection<DemoClass>();
```

We can now create a for-loop to populate 2000 instances and add them to the collection,

```
for ( int x = 1; x <= 2000; x++)
{
    var dc = new DemoClass();
    dc.ID = x;
    dc.Name = "Person " + x.ToString();
    dc.Company = x % 2 == 0 ?
        "Super Company " + x.ToString() : "Sub-par company " + x.ToString();
    dc.IsComplete = x % 4 == 0 ? true : false;
    dc.DueDate = DateTime.Today.AddDays(x);
    myDemoClasses.Add(dc);
}
```

Finally, we set the collection as the ItemsSource property for our GridView,

```
radGridView1.ItemsSource = myDemoClasses;
```

Run the application and you'll see the 2000 records displayed, neat as a pin, in the GridView.

Note that the columns are not taking up the full width of the Browser. If you prefer, you can set the columns to take the available space by setting the column width of the GridView in the XAML file to *,

```
<telerik:RadGridView Name="radGridView1" ColumnWidth="*" />
```

The columns now expand to fill the available width, as shown in figure 4,

Drag a column header and drop it here to group by that column					
	ID	Name	Company	IsComplete	DueDate
>	1	Person 1	Sub-par company 1	<input type="checkbox"/>	3/20/2012 12:00:00 AM
	2	Person 2	Super Company 2	<input type="checkbox"/>	3/21/2012 12:00:00 AM
	3	Person 3	Sub-par company 3	<input type="checkbox"/>	3/22/2012 12:00:00 AM
	4	Person 4	Super Company 4	<input checked="" type="checkbox"/>	3/23/2012 12:00:00 AM
	5	Person 5	Sub-par company 5	<input type="checkbox"/>	3/24/2012 12:00:00 AM
	6	Person 6	Super Company 6	<input type="checkbox"/>	3/25/2012 12:00:00 AM
	7	Person 7	Sub-par company 7	<input type="checkbox"/>	3/26/2012 12:00:00 AM
	8	Person 8	Super Company 8	<input checked="" type="checkbox"/>	3/27/2012 12:00:00 AM
	9	Person 9	Sub-par company 9	<input type="checkbox"/>	3/28/2012 12:00:00 AM

Figure 4

Built In Functionality

A great deal of functionality is built into the control. For example, sorting. Just click on a column header to sort that column. Click again to reverse sort that column. Try this with IsComplete or DueDate.

You can sort on multiple columns by clicking on one column, then holding shift and clicking on a second column.

Advanced filtering is built in as well. To see this, click on the filter icon on any column, as shown in Figure 5,

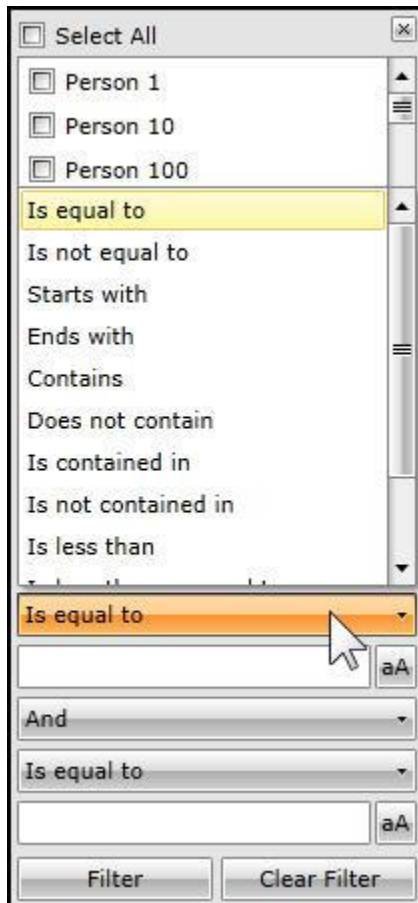


Figure 5

Grouping is built in and is as simple as dragging and dropping a column header into the top row. If we drag and drop IsComplete into the grouping area, the results are immediately sorted into the two groups, as shown in figure 6,



Figure 6

You can group by multiple columns by dragging additional columns into the grouping area. To removing grouping, hover over the grouped by column and click the X to remove it, as shown in figure 7,

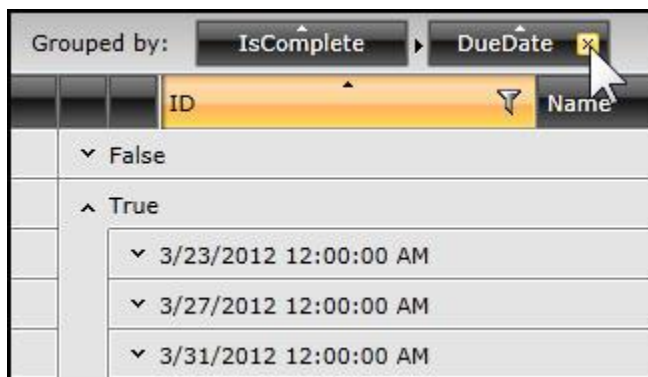


Figure 7

Finally, you can reorder your columns using drag and drop, as shown in figure 8, in which I'm dragging the ID column to a new position,



Figure 8

Exporting to Additional Formats

You can export your data from RadGridView into different formats:

- Text
- CSV
- Html
- Excel XML

Start a new project as we'll look at some additional features as we look at exporting.

Drag a RadGridView onto the design surface, but this time let's go to the XAML and add the attribute, AutoGenerateColumns and set it to false,

```
<telerik:RadGridView Name="radGridView1" ColumnWidth="*" AutoGenerateColumns="False">
</telerik:RadGridView>
```

This tells the grid not to generate its columns based on the properties in the data class; rather, you'll hand tool the columns,

```
<telerik:RadGridView Name="radGridView1" ColumnWidth="*" AutoGenerateColumns="False">
    <telerik:RadGridView.Columns>
        <telerik:GridViewDataColumn DataMemberBinding="{Binding LastName}"
            Header="Last Name" />
        <telerik:GridViewDataColumn DataMemberBinding="{Binding FirstName}"
            Header="First Name" />
        <telerik:GridViewDataColumn DataMemberBinding="{Binding Age}"
            Header="Age" />
        <telerik:GridViewDataColumn DataMemberBinding="{Binding Married}"
            Header="Is Married?" />
    </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

Add a new class, Employee to your project, in the file Employee.cs

```
public class Employee
{
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public int Age { get; set; }
    public bool Married { get; set; }
}
```

Download the EmployeeService.cs file from the source files, and import the file into your project with *Add->Existing Item*. This file creates approximately 15 employee records with the static method GetEmployees().

In MainPage.xaml.cs, in the Loaded event handler you'll set the ItemsSource property for the grid to the result of calling GetEmployees,

```
public MainPage()
{
    InitializeComponent();
    Loaded += new RoutedEventHandler(MainPage_Loaded);
}

void MainPage_Loaded(object sender, RoutedEventArgs e)
{
    this.radGridView1.ItemsSource = EmployeeService.GetEmployees();
}
```

Run the application to ensure that the GridView properly displays the 15 records. Shut down the application and return to MainPage.xaml. In the design surface, drag the bottom of the grid up a bit to make room to add a button.

Set the content of the button to Export. Add a click event for the button and return to the code behind to write the event handler.

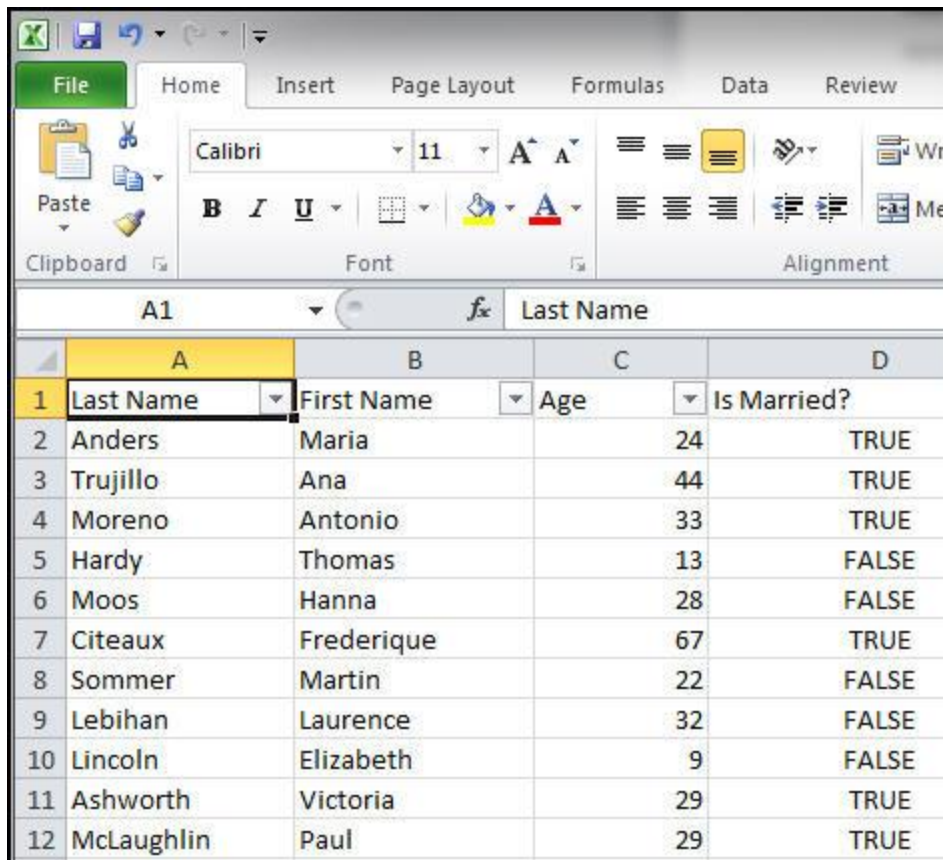
You will need to resolve a number of namespaces, or let JustCode do it for you.

```
private void button1_Click(object sender, RoutedEventArgs e)
{
    string extension = ".xls";
    SaveFileDialog dialog = new SaveFileDialog()
    {
        DefaultExt = extension,
        Filter = String.Format("{1} files (*.{{0}})|*.*|All files (*.*)|*.*", extension,
"Excel"),
        FilterIndex = 1
    };

    if ( dialog.ShowDialog() == true)
    {
        using (Stream stream = dialog.OpenFile())
        {
            radGridView1.Export(stream,
                new GridViewExportOptions()
                {
                    Format = ExportFormat.ExcelML,
                    ShowColumnHeaders = true,
                    ShowColumnFooters = true,
                    ShowGroupFooters = false,
                });
        }
    }
}
```

The essence of this event handler is to create a standard SaveFileDialog (with the default extension of Excel), and then if True is returned from the dialog (the user clicks OK) to open a stream and to export to the Format specified. The key is the Format, which is set to one of the four enumerated ExportFormat values.

Run the program and save the file to Employees.xls. Open the file in Excel and you'll see that the columns and data appear as expected, as seen in figure 9,



	A	B	C	D
1	Last Name	First Name	Age	Is Married?
2	Anders	Maria	24	TRUE
3	Trujillo	Ana	44	TRUE
4	Moreno	Antonio	33	TRUE
5	Hardy	Thomas	13	FALSE
6	Moos	Hanna	28	FALSE
7	Citeaux	Frederique	67	TRUE
8	Sommer	Martin	22	FALSE
9	Lebihan	Laurence	32	FALSE
10	Lincoln	Elizabeth	9	FALSE
11	Ashworth	Victoria	29	TRUE
12	McLaughlin	Paul	29	TRUE

Figure 9

Change the extension from “xls” to “htm” and change the name of the extension to HTML. Change the format to HTML as shown here,

```
string extension = "htm";
SaveFileDialog dialog = new SaveFileDialog()
{
    DefaultExt = extension,
    Filter = String.Format("{1} files (*.{0})|*.{0}|All files (*.*)|*.*",
        extension, "HTML"),
    FilterIndex = 1
};

if ( dialog.ShowDialog() == true)
{
    using (Stream stream = dialog.OpenFile())
    {
        radGridView1.Export(stream,
            new GridViewExportOptions()
            {
                Format = ExportFormat.Html,
                ShowColumnHeaders = true,
                ShowColumnFooters = true,
                ShowGroupFooters = false,
            });
    }
}
```

}

And the result is an HTML table that you can display in a Browser as shown in Figure 10,

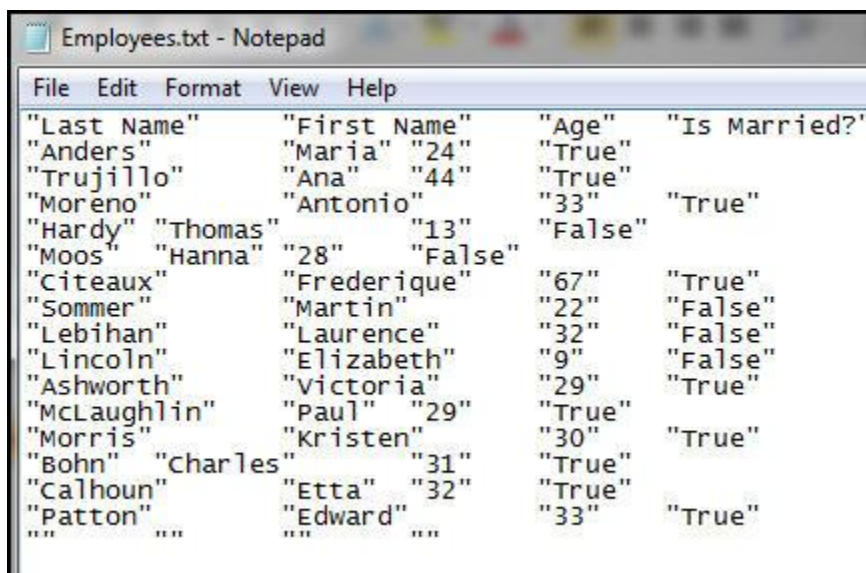


The screenshot shows a web browser window with the address bar displaying 'C:\Temp\Employee...'. The browser has several tabs open, including 'RoboForm', 'Safenotes', 'Logins', '(passcards)', 'Jesse MSoft', and 'Jesse Personal'. The main content area displays an HTML table with the following data:

Last Name	First Name	Age	Is Married?
Anders	Maria	24	True
Trujillo	Ana	44	True
Moreno	Antonio	33	True
Hardy	Thomas	13	False
Moos	Hanna	28	False
Citeaux	Frederique	67	True
Sommer	Martin	22	False
Lebihan	Laurence	32	False
Lincoln	Elizabeth	9	False
Ashworth	Victoria	29	True
McLaughlin	Paul	29	True
Morris	Kristen	30	True
Bohn	Charles	31	True
Calhoun	Etta	32	True
Patton	Edward	33	True

Figure 10

Similarly, change it to Text and you save a text file as seen in figure 11 and if you change it to CSV you get a comma separated values file as shown in figure 12.



The screenshot shows a Notepad window titled 'Employees.txt - Notepad'. The text content is as follows:

"Last Name"	"First Name"	"Age"	"Is Married?"
"Anders"	"Maria"	"24"	"True"
"Trujillo"	"Ana"	"44"	"True"
"Moreno"	"Antonio"	"33"	"True"
"Hardy"	"Thomas"	"13"	"False"
"Moos"	"Hanna"	"28"	"False"
"Citeaux"	"Frederique"	"67"	"True"
"Sommer"	"Martin"	"22"	"False"
"Lebihan"	"Laurence"	"32"	"False"
"Lincoln"	"Elizabeth"	"9"	"False"
"Ashworth"	"Victoria"	"29"	"True"
"McLaughlin"	"Paul"	"29"	"True"
"Morris"	"Kristen"	"30"	"True"
"Bohn"	"Charles"	"31"	"True"
"Calhoun"	"Etta"	"32"	"True"
"Patton"	"Edward"	"33"	"True"
"..."	"..."	"..."	"..."

Figure 11

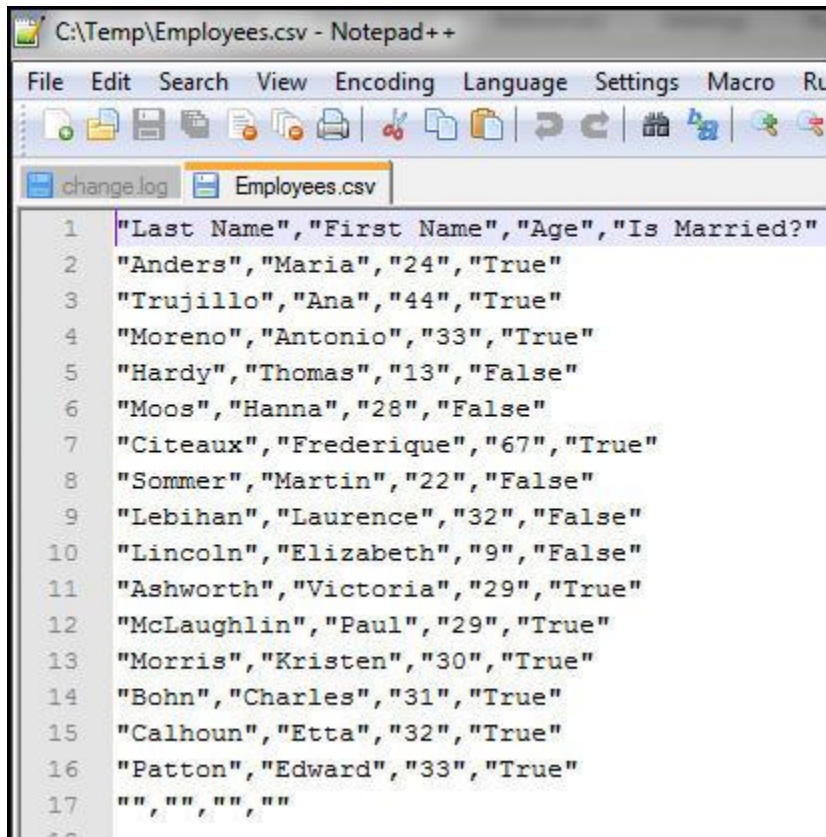


Figure 12

Data Paging

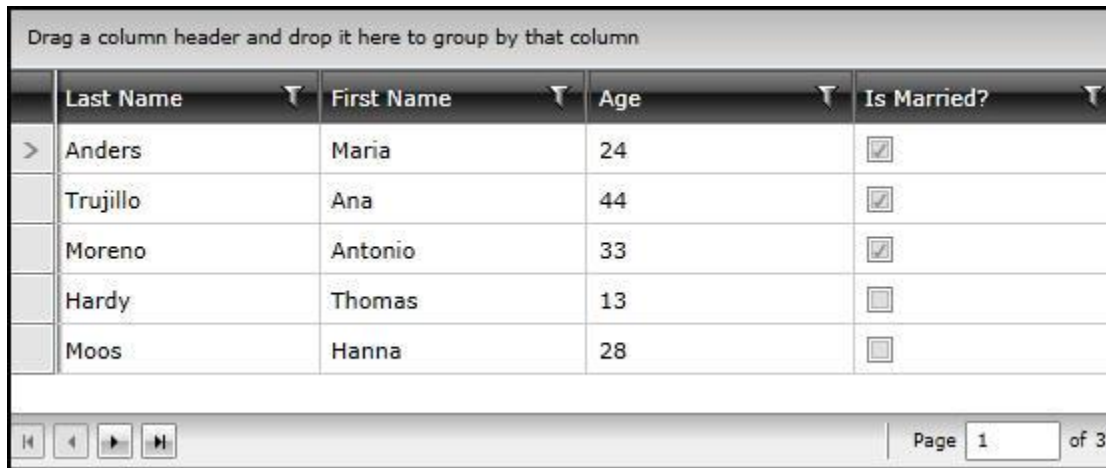
There are certainly times when you do not want to send all of the possible records matching your search criteria to the client machine. While RadGridView is extremely fast and responsive, it simply takes a bit of time to transmit, e.g., 100,000 records. In this case, you want to be able to “page” through the data, seeing (e.g.,) 10 records at a time.

To support this, you’ll use a second control, the RadDataPager. Open your XAML file to the last line of the GridView and add a RadDataPager,

```
<telerik:RadDataPager Name="xRadDataPager" Source="{Binding Items,
ElementName=radGridView1}" PageSize="5" />
```

The source in this case is using element binding – that is we’re binding the RadDataPager to another element on the page, in this case the GridView. We are specifically binding to the Items in

radGridView1. The PageSize attribute determines how many items will be in a given page. We've set it to 5, and since we have a total of 15 records, we'll see 3 pages of data, as shown in figure 13,



	Last Name	First Name	Age	Is Married?
>	Anders	Maria	24	<input checked="" type="checkbox"/>
	Trujillo	Ana	44	<input checked="" type="checkbox"/>
	Moreno	Antonio	33	<input checked="" type="checkbox"/>
	Hardy	Thomas	13	<input type="checkbox"/>
	Moos	Hanna	28	<input type="checkbox"/>

Page 1 of 3

Figure 13

Notice that the DataPager attaches itself to the bottom of the GridView. In the default pager there are four buttons on the left and a “jump” box on the right (which displays the page you are on and lets you type in the page number you want to jump to).

The four buttons are, left to right

- Go to first page
- Go to previous page
- Go to next page
- Go to last page

Enabling and disabling of the various buttons (e.g. disabling next page on the last page) is automatic.

This, of course is the default display, but there are a host of alternatives. You can see this by entering DisplayMode in the XAML and Intellisense will bring up a long list of options, as shown in Figure 14,

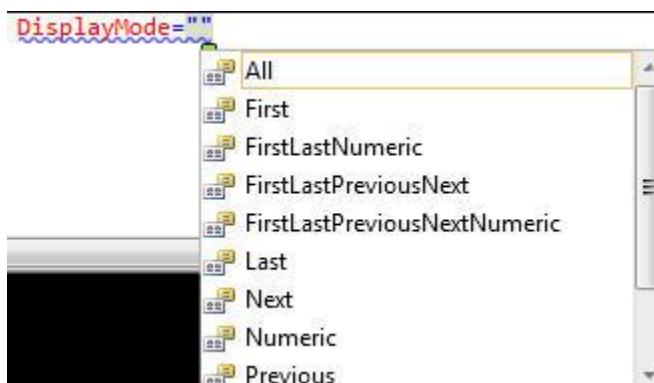


Figure 14

Let's scroll down and choose PreviousNextNumeric,

```
<telerik:RadDataPager DisplayMode="PreviousNextNumeric" Name="xRadDataPager"
Source="{Binding Items, ElementName=radGridView1}" PageSize="5" />
```

When you run the application, you'll see that the visible buttons are previous, next and a numeric button for each page, as shown in Figure 15



Figure 15

You can also manipulate the DataPager programmatically, as shown by Intellisense in figure 16,

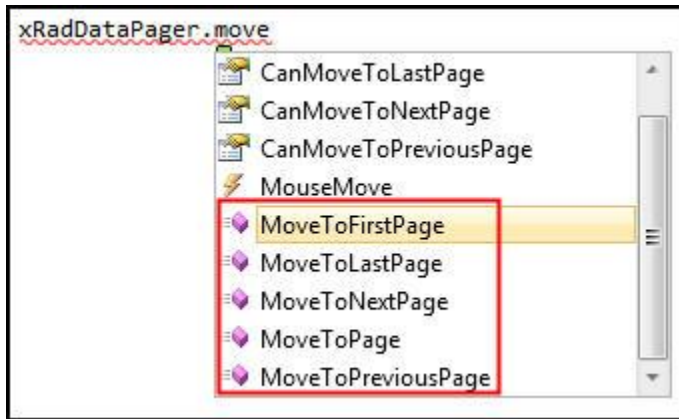


Figure 16

Aggregate Functions

The RadGridView provides 7 modes of aggregate functions to provide aggregated data at the bottom of columns in the grid. The 7 are,

- Average
- Count
- First / Last
- Minimum / Maximum
- Sum

Let's return to our application but comment out the RadDataPager control. In the XAML we'll go up to the LastName GridViewDataColumn and open it up, adding in AggregateFunctions,

```
<telerik:GridViewDataColumn DataMemberBinding="{Binding LastName}" Header="Last Name" >
    <telerik:GridViewDataColumn.AggregateFunctions>

</telerik:GridViewDataColumn.AggregateFunctions>
```

```
</telerik:GridViewDataColumn>
```

Inside the AggregateFunctions Intellisense will show all the possible aggregations, as shown in figure 17,

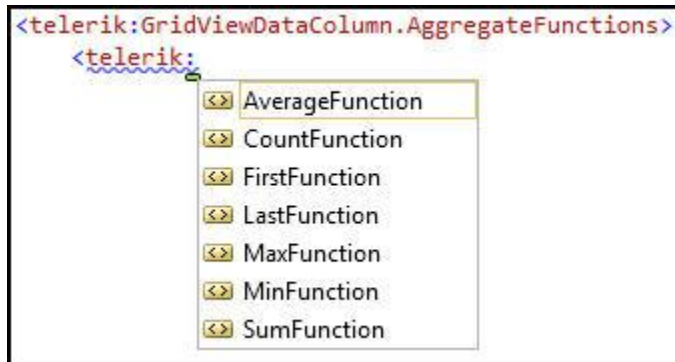


Figure 17

We'll choose Count. We can add an attribute for the caption to be associated with the count,

```
<telerik:GridViewDataColumn DataMemberBinding="{Binding LastName}" Header="Last Name" >
  <telerik:GridViewDataColumn.AggregateFunctions>
    <telerik:CountFunction Caption="Total # of records: " />
  </telerik:GridViewDataColumn.AggregateFunctions>
</telerik:GridViewDataColumn>
```

Before we go any further, we need to go up to the GridView itself and turn on ShowColumnFooters,

```
<telerik:RadGridView Name="radGridView1" ColumnWidth="*" AutoGenerateColumns="False"
  Margin="0,0,0,72" ShowColumnFooters="True">
```

When you run the application, the total number of records is shown in the footer for the LastName column, as shown in figure 18,

Drag a column header and drop it here to group by that column				
	Last Name	First Name	Age	Is Married?
>	Anders	Maria	24	<input checked="" type="checkbox"/>
	Trujillo	Ana	44	<input checked="" type="checkbox"/>
	Moreno	Antonio	33	<input checked="" type="checkbox"/>
	Hardy	Thomas	13	<input type="checkbox"/>
	Moos	Hanna	28	<input type="checkbox"/>
	Citeaux	Frederique	67	<input checked="" type="checkbox"/>
	Sommer	Martin	22	<input type="checkbox"/>
	Lebihan	Laurence	32	<input type="checkbox"/>
	Lincoln	Elizabeth	9	<input type="checkbox"/>
	Ashworth	Victoria	29	<input checked="" type="checkbox"/>
	McLaughlin	Paul	29	<input checked="" type="checkbox"/>
	Morris	Kristen	30	<input checked="" type="checkbox"/>
	Bohn	Charles	31	<input checked="" type="checkbox"/>
	Calhoun	Etta	32	<input checked="" type="checkbox"/>
	Patton	Edward	33	<input checked="" type="checkbox"/>
Total # of records: 15				

Figure 18

The most common column footer manipulates numeric data. Let's find the average age of our 15 records.

We do with Age just what we did with LastName,

```
<telerik:GridViewDataColumn DataMemberBinding="{Binding Age}" Header="Age">
  <telerik:GridViewDataColumn.AggregateFunctions>
    <telerik:AverageFunction Caption="Average age: " />
  </telerik:GridViewDataColumn.AggregateFunctions>
</telerik:GridViewDataColumn>
```

When you run this you should see, at the bottom of the age column Average age: 30.4

Localization

Localization is the process of adapting a product to a particular language or culture. We understand how important this is to our customers, and RadGridView has support for localization baked in.

Running the application shows that there are various areas that are in English, as shown in figure 19; we'll want to localize the application so that these are in Spanish.

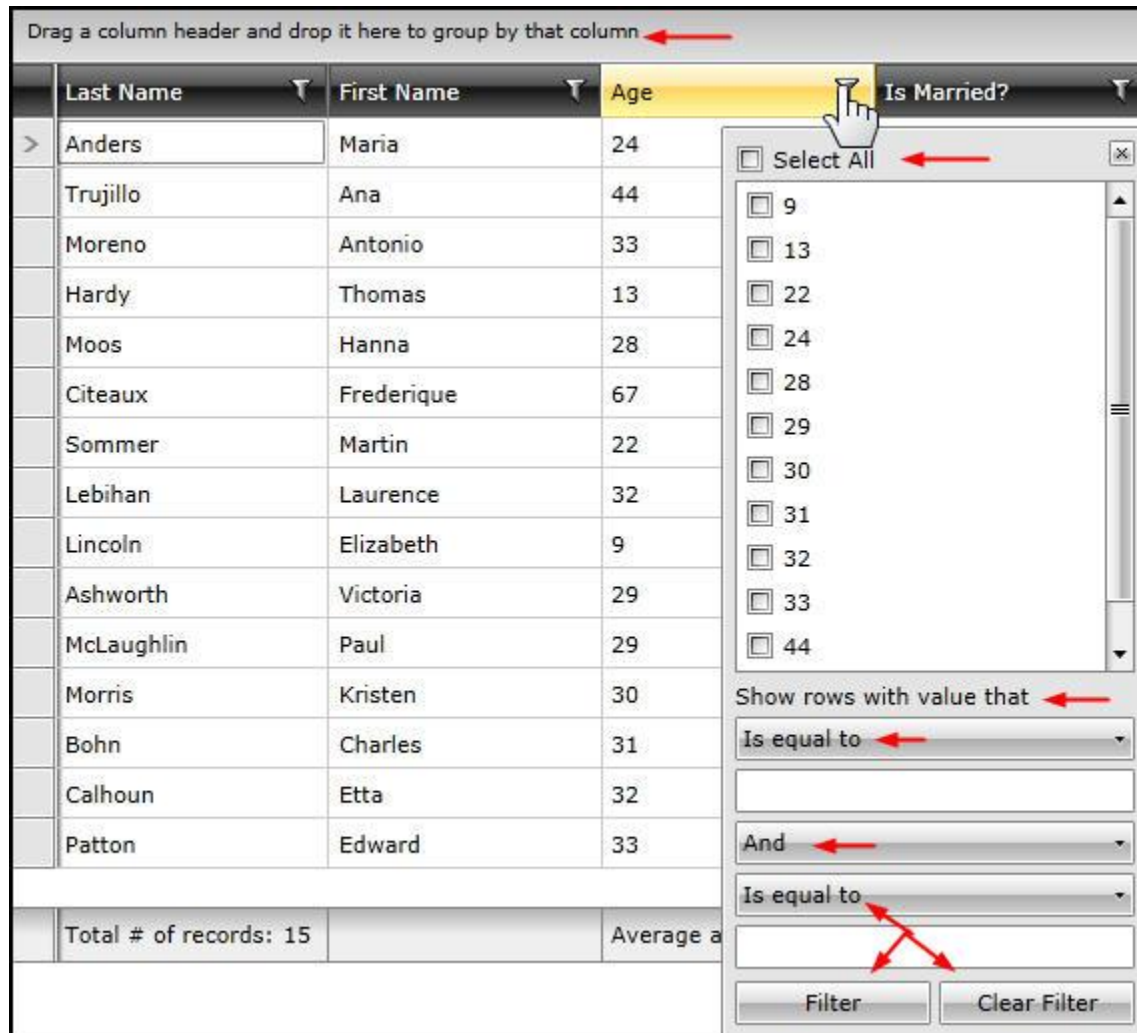


Figure 19

The first step (after stopping the application) is to unload the main project by right clicking on it and choosing Unload Project.

Once unloaded, right click on the project and choose Edit, as shown in figure 20,

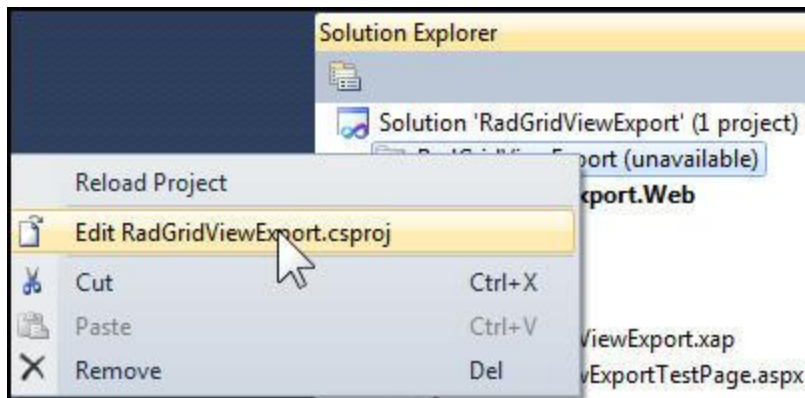


Figure 20

Find the element for SupportedCultures, and add es for *Español* (Spanish).

```
<SupportedCultures>es</SupportedCultures>
```

Save and close the file and then right click on the project and click on *Reload Project*.

Open App.xaml.cs and find Application_Startup,

```
private void Application_Startup(object sender, StartupEventArgs e)
{
    this.RootVisual = new MainPage();
}
```

Before the first line of this method we need to set the culture we'll be using.

```
Thread.CurrentThread.CurrentCulture = new CultureInfo("es");
Thread.CurrentThread.CurrentUICulture = new CultureInfo("es");
```

Run the application and Hey! Presto! All the strings that are not hard-coded into your application are now in Spanish, as shown in figure 21,

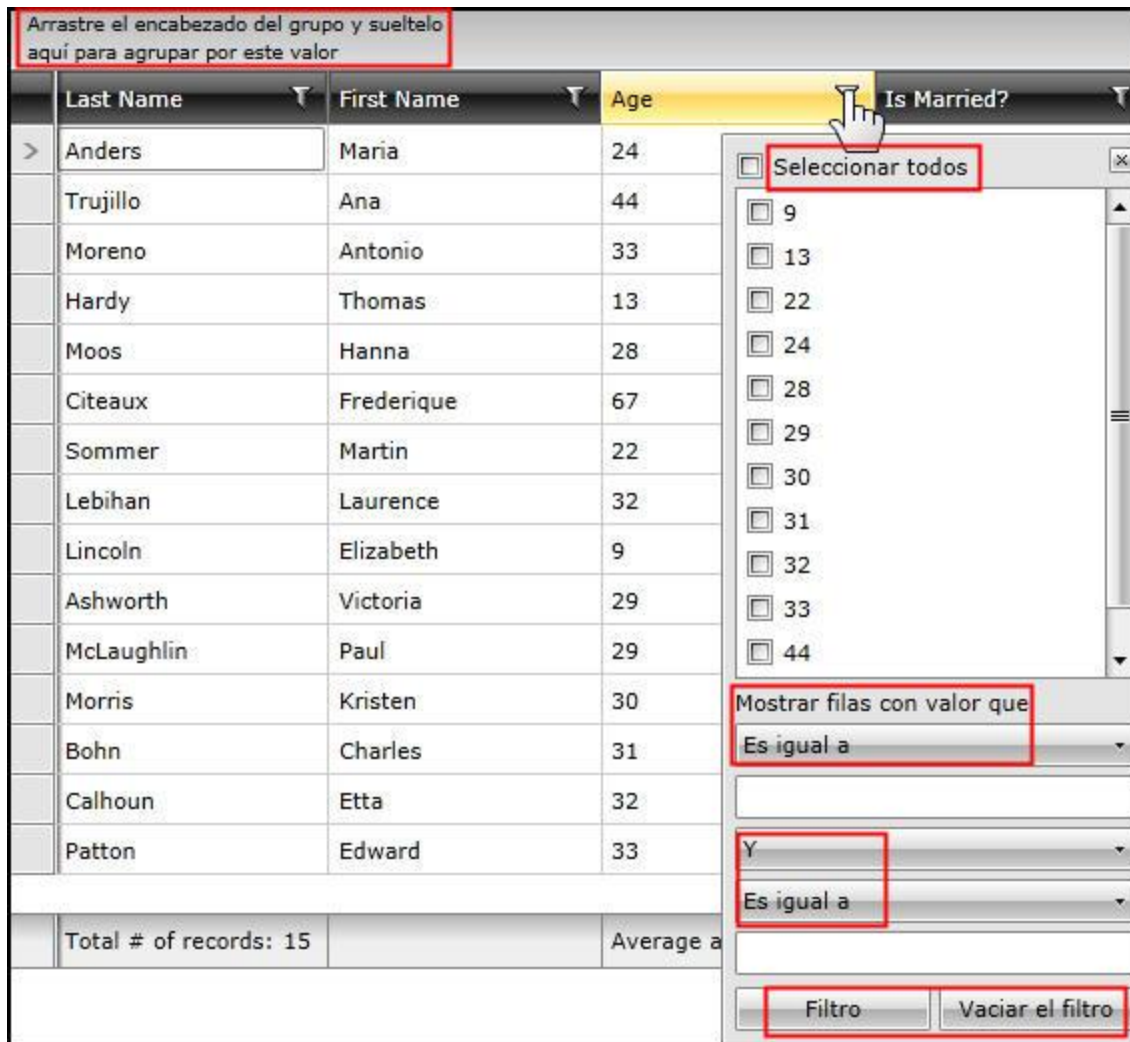


Figure 21

If you want the headings and footers to be in Spanish as well, you'll need to create resources for those strings.

Validation

Validation is intended to support user input – validating the user's data locally on the client. To see this, we'll return to the RadGridView and validate the age. To do this, we'll use the CellValidating event, which is always raised before the CellValidated event.

CellValidating lets you stop the commit process on a cell if the data is invalid.

To begin, return to MainPage.xaml and find the declaration of the GridView. Add an event for CellValidating,

```
<telerik:RadGridView Name="radGridView1" CellValidating="radGridView1_CellValidating"
ColumnWidth="*" AutoGenerateColumns="False" Margin="0,0,0,72" ShowColumnFooters="True">
```

Switch to the code behind and find the stub for the event handler,

```
private void radGridView1_CellValidating(  
    object sender, GridViewCellValidatingEventArgs e)  
{  
  
}
```

We only want (for now) to validate the Age column, so we begin with an if statement,

```
if ( e.Cell.Column.UniqueName == "Age" )
```

Inside the if statement we'll place the logic for our validation,

Here's the complete event handler,

```
private void radGridView1_CellValidating(  
    object sender, GridViewCellValidatingEventArgs e)  
{  
    if ( e.Cell.Column.UniqueName == "Age" )  
    {  
        var newValue = Int32.Parse(e.NewValue.ToString());  
        if (newValue < 0 || newValue > 130)  
        {  
            e.IsValid = false;  
            e.ErrorMessage = "The entered value for age must be between 0 and 130.";  
        }  
    }  
}
```

Run the application and change one of the ages to 500. Notice that the cell turns red and if you place the mouse on the red triangle, the error message appears next to the cell, as shown in figure 22,



Drag a column header and drop it here to group by that column				
	Last Name	First Name	Age	Is Married?
>	Anders	Maria	24	<input checked="" type="checkbox"/>
	Trujillo	Ana	44	<input checked="" type="checkbox"/>
	Moreno	Antonio	33	<input checked="" type="checkbox"/>
			500	<input checked="" type="checkbox"/>
			28	<input checked="" type="checkbox"/>
	Citeaux	Frederique	67	<input checked="" type="checkbox"/>

Figure 22

In addition to Cell Validating there is also a Row Validating event that occurs before the row is committed. You can see this by adding the RowValidating event and a handler to the grid,

```
<telerik:RadGridView
    Name="radGridView1"
    CellValidating="radGridView1_CellValidating"
    RowValidating="radGridView1_RowValidating"
    ColumnWidth="*"
    AutoGenerateColumns="False"
    Margin="0,0,0,72"
    ShowColumnFooters="True">
```

In the event handler, we'll just stub out a message box to indicate that the RowValidating event fired,

```
private void radGridView1_RowValidating(
    object sender, GridViewRowValidatingEventArgs e)
{
    MessageBox.Show("Row Validating");
}
```

To see this at work, change a cell in the row, and then notice the Row-Commit indicator in the left margin. Click on that, as shown in figure 23,

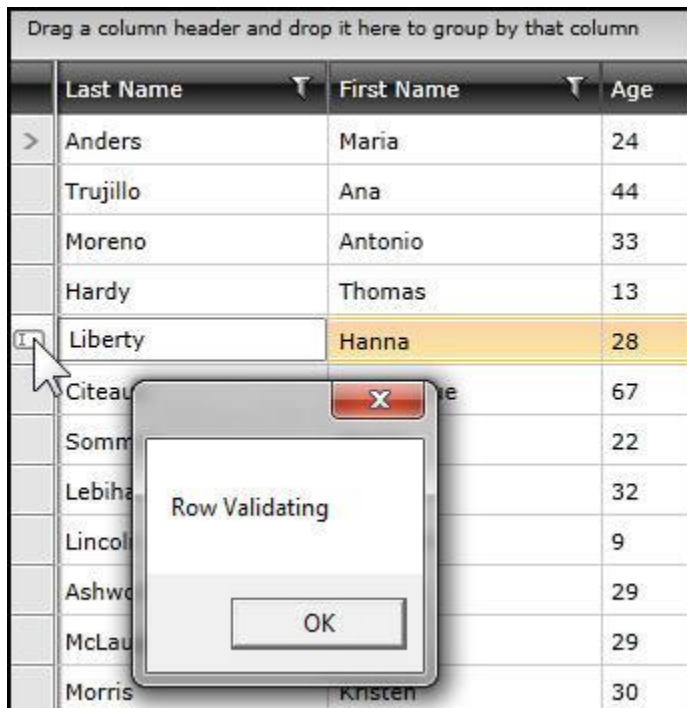


Figure 23

Copy And Paste To Excel

The ability to copy and paste values and rows to and from RadGridView greatly enhances the interoperability of your application with Microsoft Excel.

To begin, we return to MainPage.xaml and to our definition of the RadGridView, where we add four attributes,

```
<telerik:RadGridView
    Name="radGridView1"
    CellValidating="radGridView1_CellValidating"
    RowValidating="radGridView1_RowValidating"
    ColumnWidth="*"
    AutoGenerateColumns="False"
    Margin="0,0,0,72"
    ShowColumnFooters="True"
    ClipboardCopyMode="All"
    ClipboardPasteMode="Default"
    SelectionMode="Extended"
    SelectionUnit="Cell">
```

These determine how much will be copied to the clipboard and how entries on the clipboard will be pasted back into the grid. You can get a good idea of what the copy options are through Intellisense as shown in figure 24, and of the paste options as well, as shown in figure 25.



Figure 24

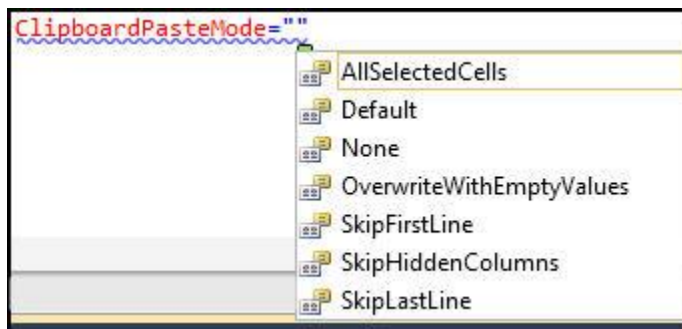


Figure 25

You can now make selections in the GridView and copy them to the clipboard and paste them to Excel. You can do this by row or by cell, or by using Control-click you can copy noncontiguous cells.

In Figure 26, you can see that I've control-clicked on a number of cells, then hit control-c to copy and then control-p to paste them into Excel. They are placed in Excel in the appropriate rows and columns, and because I chose All from the ClipboardCopyMode property; the headers and footers are copied as well,

Last Name	First Name	Age	Is Married?
Anders	Maria	24	<input checked="" type="checkbox"/>
Trujillo	Ana	44	<input checked="" type="checkbox"/>
Moreno	Antonio	33	<input checked="" type="checkbox"/>
Hardy	Thomas	13	<input type="checkbox"/>
Moos	Hanna	28	<input type="checkbox"/>
Citeaux	Frederique	67	<input checked="" type="checkbox"/>
Sommer	Martin	22	<input type="checkbox"/>
Lebihan	Laurence	32	<input type="checkbox"/>
Lincoln	Elizabeth	9	<input type="checkbox"/>
Total # of records: 15		Average age: 30.4	

A	B	C	D	E	F	G	H	I	J	K
Last Name	First Name	Age	Is Married?							
Trujillo			33							
	Antonio		13							
			TRUE							
			22	FALSE						
Lebihan	Laurence		32	FALSE						
Total # of records: 15		Average age: 30.4								

Figure 26

You can also edit cells in Excel and when you copy them back the modified values will be shown in the RadGridView.