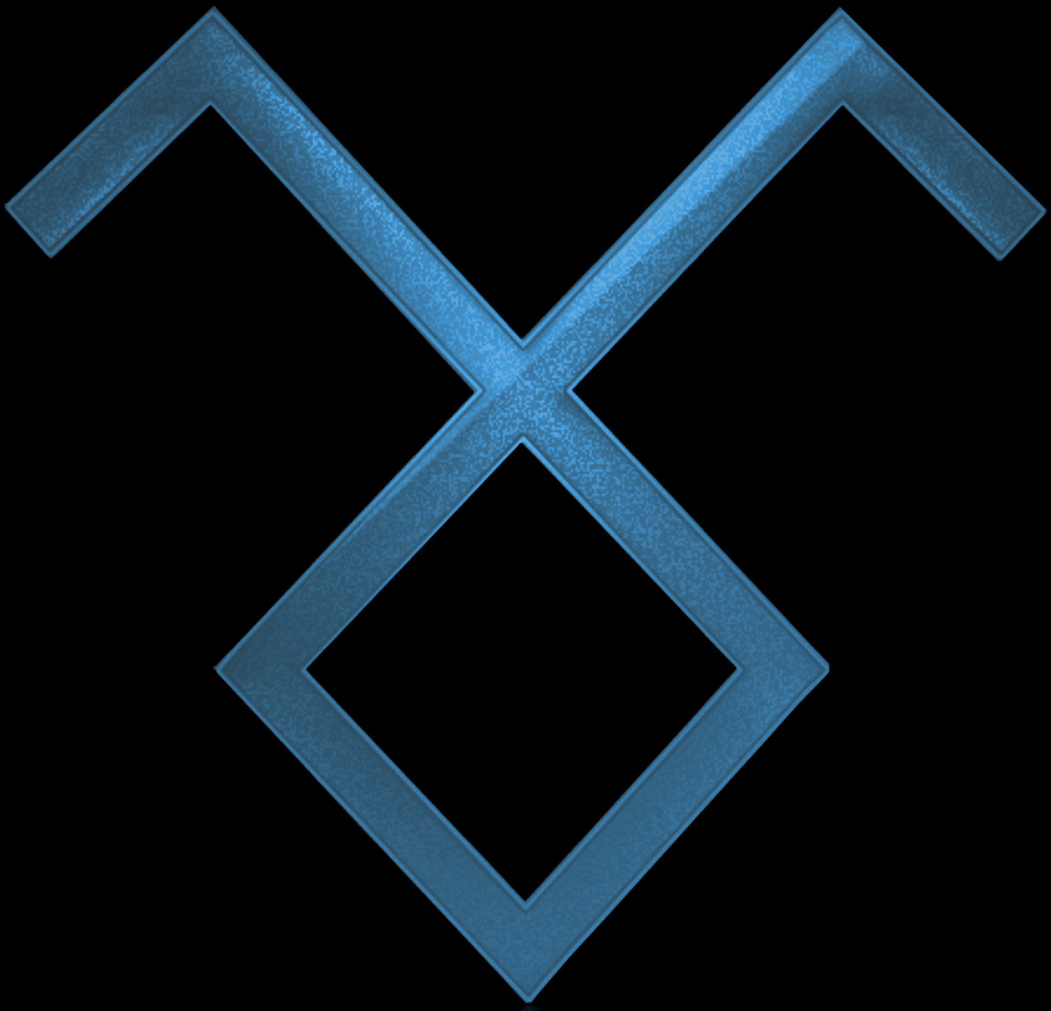
RadControls *for* SILVERLIGHT *made easy*



telerik

deliver more than expected

RadControls for Silverlight Made Easy

Welcome to Telerik RadControls for Silverlight Made Easy.

RadControls for Silverlight

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: January 2011

Special thanks to:

Valeri Hristov
Nikolay Atanasov
Vassil Petev
Pavel Pavlov
Miroslav Nedyalkov
Tihomir Petkov
Boryana Miloshevska
Konstantin Petkov
Konstantina Gocheva
Tina Stancheva
Georgi Dangalov
Maya Zhecheva
Evgenia Milcheva
Veselin Vasilev
Stephen Forte.

Last but not least, thank you to all our families for their support and patience while we wrote the book.

Table of Contents

Part I

Introduction

1 Introduction

1.1 Who Should Read This Courseware

This courseware assumes that you are familiar with VB.NET or C# code. The courseware uses Visual Studio 2010 and assumes you know your way around this environment. You should be able to navigate the basic functional areas of the IDE (e.g. Solution Explorer, Properties, code/designer web pages etc.) and be able to run and debug applications and class libraries. You should have a basic familiarity with XML syntax.

1.2 What Do You Need to Have Before Reading This Courseware

Computer Setup

The courseware assumes you are running Windows X86 or x64 500-megahertz (MHz) or higher processor with 128-megabytes (MB) of RAM.

Silverlight Setup

For information on system requirements for Silverlight, see "Microsoft Silverlight System Requirements" at http://www.microsoft.com/silverlight/get-started/install/default.aspx. This will show you the combinations of operating system and browser that will support Silverlight.

Development Tools

This courseware assumes that you have installed:

- Visual Studio 2010 or better
- Expression Blend 4 [optional]
- Silverlight 4 SDK
- Silverlight Toolkit
- WCF RIA Services

See the Silverlight Getting Started site at http://silverlight.net/getstarted/ for links to these resources.

RadControls for Silverlight

Get RadControls for Silverlight at http://www.telerik.com/purchase/individual/silverlight.aspx.

1.3 How This Courseware is Organized

The initial chapters of this courseware will help you quickly get up to speed with Silverlight and will discuss some of the issues in common with all RadControls such as binding or theming. In the "Controls" section of this Courseware, the chapter structure is as follows:

- **Objectives**: A summary of what you will learn in the chapter.
- **Overview**: A high-level overview of the control, what it does and its key features.
- **Getting Started**: A step-by-step walk through of building an application that uses the control in the minimal number of steps.
- **Control Details**: One or more topics that dig into specific properties, methods and events of the control.
- **Binding**: How to bind the control to data.
- **Customization**: How to give the control a custom look and feel or special functionality. Typically this will be through the use of templates and includes many examples of using Expression Blend.
- **Wrap Up**: A summary of what you learned in the chapter.

1.4 About Telerik

Telerik is a leading vendor of User Interface (UI) components for Microsoft .NET technologies (ASP.NET AJAX, WinForms, Windows Presentation Foundation and Silverlight), as well as tools for .NET Reporting, ORM and web content management. Building on our expertise in interface development and Microsoft technologies, Telerik helps customers build applications with unparalleled richness, responsiveness and interactivity. Created with passion, Telerik products help thousands of developers every day to be more productive and deliver reliable applications under budget and on time.

1.5 About Falafel

Founded in 2003, Falafel Software, Inc. provides the highest quality software development, consultation, and training services available. Starting initially with consulting and training, Falafel Software found itself expanding rapidly on the excellence of its engineers and the incredible sense of teamwork exhibited by everyone in the company. This common mutual respect for each other's talents has been a major asset for Falafel, causing extraordinary growth, and a level of quality that very few other IT companies can match. Employees include best-selling authors, industry speakers, technology decision makers, and former Microsoft and Borland engineers. All of Falafel engineers are Microsoft Certified Professionals, Certified Application Developers, or Most Valuable Professionals.

Falafel has written the following Telerik courseware:

- RadControls for ASP.NET
- RadControls for ASP.NET AJAX
- RadControls for Winforms
- Telerik Reporting
- Telerik OpenAccess ORM

1.6 Introducing RadControls for Silverlight

RadControls are built on Microsoft Silverlight 3 and include UI controls for building rich line-of-business Silverlight applications. Sharing the same code base with Telerik WPF controls, the Silverlight controls offer a clean and intuitive API, Blend support and powerful theming capabilities that will radically improve your RIA development.

Comprehensive Toolset from the Masters of Web UI: An established leader in web interface technologies, Telerik now offers you RadControls for Silverlight 3 – a comprehensive suite of controls that bring style and interactivity to your LOB applications.

Engineered for Great Performance: Telerik Silverlight 3 controls are engineered for outstanding performance through native UI virtualization, an innovative LINQ-based data engine, asynchronous data binding, RadCompression module and other techniques that help reduce page loading time and speed up data operations.

WCF RIA Services support: All data-bound Telerik Silverlight 3 controls support binding to WCF RIA Services. This enables all data operations to execute on the server, which results in speedier sorting, filtering and paging for RadGridView. With completely codeless binding, it takes almost no extra effort to bind RadControls for Silverlight 3 to WCF RIA Services.

Validation support: All Telerik Silverlight 3 input controls support metadata-driven validation via data annotations.

Integration with Leading-Edge Technologies: RadControls for Silverlight 3 are designed to work effortlessly with cutting-edge Silverlight technologies like MVVM and Microsoft Composite Application Guidance (Prism).

Code Re-Use with RadControls for WPF: RadControls for Silverlight 3 and RadControls for WPF suites are derived from the same codebase and share the same API. They represent two almost mirror toolsets for building rich line-of-business web and desktop applications, allowing for substantial code and skills reuse between Silverlight and WPF development.

Rich Data Visualization Capabilities: Featuring everything from a super powerful Silverlight grid, to animated, fully customizable Silverlight charts and gauges, Telerik Silverlight 3 controls enable developers to transform data into interactive, animated visuals that empower end-users to analyze complex business scenarios.

Full Interoperability with ASP.NET AJAX: RadControls for Silverlight 3 are a perfect addition for existing ASP.NET AJAX applications and work nicely with Telerik RadControls for ASP.NET AJAX. This enables you to add islands of rich functionality to standards-based websites when needed, without rewriting your working applications for scratch.

Ready-to-Use Themes: Styling is made easy thanks to the integrated theme support. Telerik Silverlight controls ship with 5 major themes: Vista, Summer, Office Blue, Black and Silver. These themes help you deliver a consistent look-and-feel throughout your application.

3D Charts for Silverlight: Pushing the envelope of rich data presentation, Telerik offers the first commercial 3D chart control for Silverlight.

Support for Expression Blend: RadControls for Silverlight are styleable through Microsoft Expression Blend. As a result, you can unleash your imagination and redefine how elements look and behave.

Enhanced Routed Events Framework: To help your code become even more elegant and concise, we implemented an Enhanced Routed Events Framework for RadControls for Silverlight 3. This allows you to handle "bubbling" and "tunneling" events from other elements in the visual tree.

Free Testing Framework for RadControls: The free WebAii Testing Framework helps developers build automated functional tests and end–to-end scenario tests for both Silverlight- and AJAX-powered applications. The framework ships with special wrappers for Telerik RadControls for Silverlight and ASP.NET AJAX, making it easier than ever before to create and maintain tests.

Part II

Silverlight Introduction

2 Silverlight Introduction

2.1 Objectives

This chapter will give you a feel for how Silverlight fits relative to other web technologies. You will get a rundown on the parts that make up a Silverlight application and how those parts fits in a Visual Studio project. You will get an overview of the Silverlight Life Cycle. Finally, you will learn about the development tools available for Silverlight development and where each tool is used.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\SilverlightIntroduction\SilverlightIntroduction.sln.

2.2 Overview

Silverlight is a web application framework that runs on the client and delivers Rich Internet Applications (RIA) capability. Here's a blurb from the MSDN Silverlight Architecture page that summarizes:

> "Silverlight is not only an appealing canvas for displaying rich and interactive Web and media content to end users. It is also a powerful yet lightweight platform for developing portable, cross-platform, networked applications that integrate data and services from many sources. Furthermore, Silverlight enables you to build user interfaces that will significantly enhance the typical end user experience compared with traditional Web applications."

Silverlight started out life as a video streaming plug-in. As the platform has matured, capabilities have expanded to more closely rival those of Adobe Flash: interactive graphics, multi-media and animations in a single runtime environment.

Silverlight is routinely compared to Flash or AJAX and there is frequent speculation on what technology might prevail. Silverlight actually fits within a spectrum of existing technologies with "simple" HTML and ASP.NET web pages on one end and desk top applications on the other. Silverlight uses a small plug-in that can be hosted within traditional web pages either as an "island" or constituting the entire browser experience. Even when used as an island, you can communicate between the Silverlight application and the HTML page through Javascript.

Like AJAX, the Silverlight display is updated without refreshing the entire page. But where AJAX development has a patchwork feel, Silverlight lets you write managed code using your favorite .NET language. Silverlight user interfaces are defined in XAML (Extensible Application Markup Language) and can be built using Visual Studio or Expression Blend. The isolation between UI and functionality allows web designers to build high-end graphic interfaces that really shine without colliding with development efforts.

Silverlight is supported by a lightweight class library. This slim subset of the .NET framework Base Class Library handles collections, reflection, regular expressions, threading, web services, rich media, vector graphics and user interface tasks. Parts of the framework that are not compatible with security or that would make the plugin too heavy have been omitted, e.g. System.IO is not present in the Silverlight class library. Don't expect to find all the classes, methods or properties of the original framework, even in familiar namespaces. For example, the Enum class exists in Silverlight and has GetName() and Parse() methods, but no GetValues() method. Likewise, WebClient running in Silverlight does not allow synchronous communication. Even with these changes, Silverlight has significant functionality for such a small footprint.

Silverlight runs in a "Sandbox" and can't invoke the platform API. Files are accessed through the native "Save" and "Open" file dialogs and files can also be persisted in local "Isolated Storage". To connect to databases or access server-side classes, you need to go through an intermediary, typically a web service. Silverlight can talk with WCF, RIA, ADO.NET Data Services and REST based services. Silverlight can converse using JSON and XML formats, including RSS and ATOM.

2.3 Silverlight Anatomy 101

2.3.1 Silverlight Files

At a high level we can think of a Silverlight application as assembly that includes two sets of files:

- **App.xaml** & **App.xaml.cs**: This class is the entry point for the application as a whole. The class handles application startup and has several events for handling application wide events such as startup, exit and unhandled exceptions.

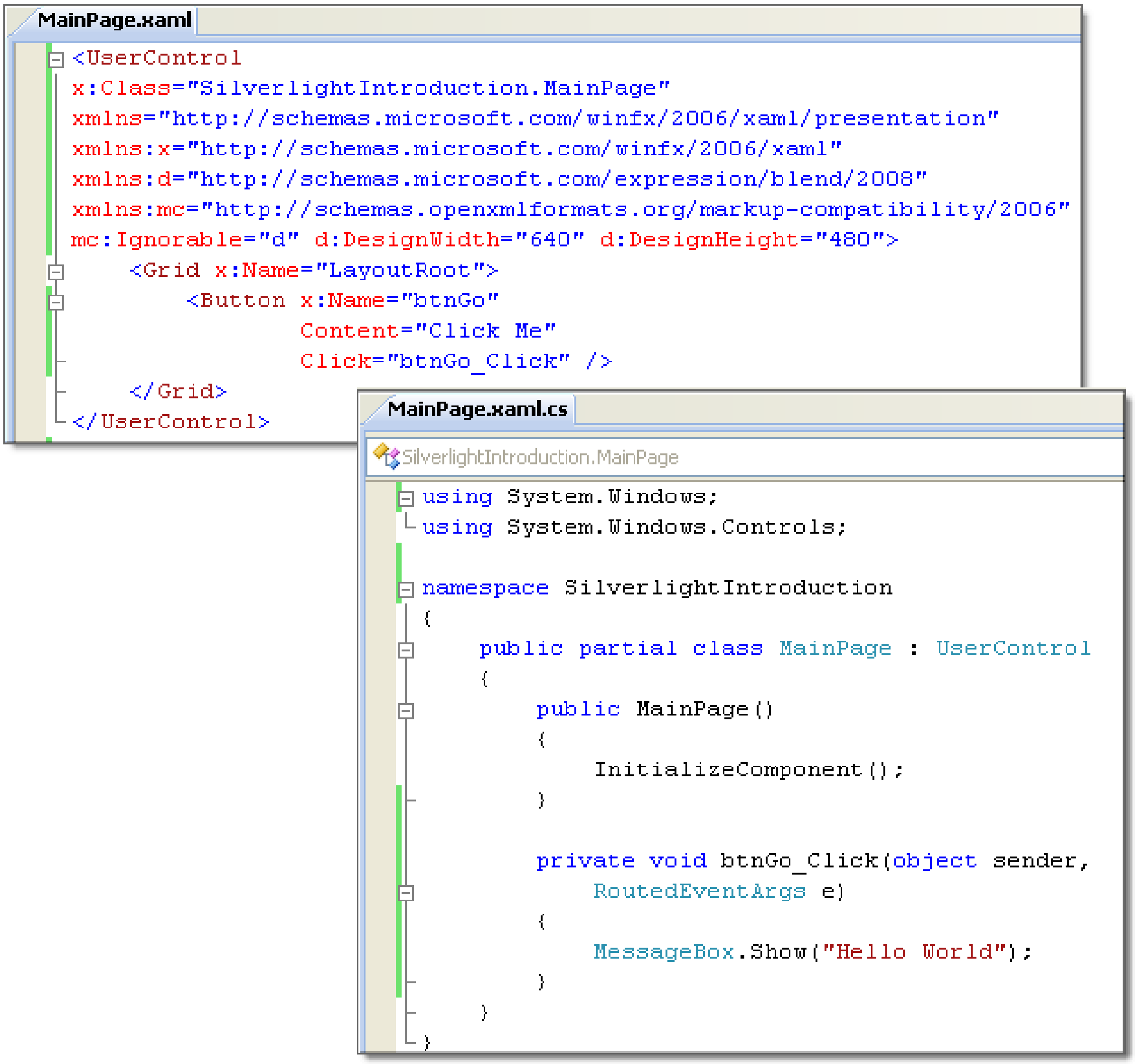
VB

```
Public Sub New()
  AddHandler Me.Startup, AddressOf Application_Startup
  Me.Exit += Me.Application_Exit
  Me.UnhandledException += Me.Application_UnhandledException

  InitializeComponent()
End Sub

Private Sub Application_Startup(ByVal sender As Object, ByVal e As StartupEventArgs)
  Me.RootVisual = New MainPage()
End Sub
'. . .
```

C#

```
public App()
{
   this.Startup += this.Application_Startup;
   this.Exit += this.Application_Exit;
   this.UnhandledException += this.Application_UnhandledException;

   InitializeComponent();
}

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

- **MainPage.xaml** & **MainPage.xaml.cs**: The initial main page is created automatically and contains a XAML (Extensible Application Markup Language) file to describe the user interface and managed code-behind to define the client logic. The screenshot below shows a minimal "Hello World" example. Most of your typical development effort will center around these files.

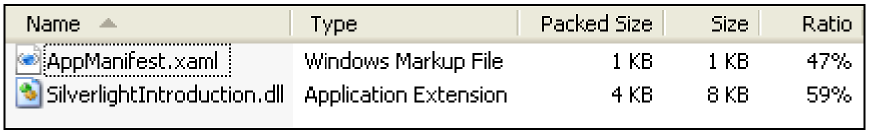


```
<UserControl
x:Class="SilverlightIntroduction.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
    <Grid x:Name="LayoutRoot">
        <Button x:Name="btnGo"
                Content="Click Me"
                Click="btnGo_Click" />
    </Grid>
</UserControl>
```

```
using System.Windows;
using System.Windows.Controls;

namespace SilverlightIntroduction
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void btnGo_Click(object sender,
            RoutedEventArgs e)
        {
            MessageBox.Show("Hello World");
        }
    }
}
```

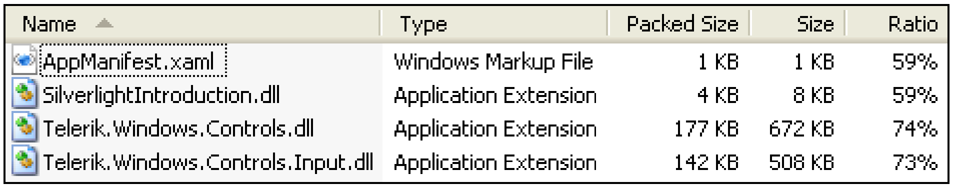
2.3.2 The XAP FIle

When the application is compiled, the assembly DLL is placed into a "*.xap" file along with a manifest. The "xap" file is simply a compressed file, like a "*.zip". When a browser requests a Silverlight application to run, the xap file is downloaded on demand and executed by the Silverlight plugin. To view the "xap" file, change the extension to "zip" and open it. The screenshot below shows the xap contents for a minimal "Hello World" application named "SilverlightIntroduction".



Name	Type	Packed Size	Size	Ratio
AppManifest.xaml	Windows Markup File	1 KB	1 KB	47%
SilverlightIntroduction.dll	Application Extension	4 KB	8 KB	59%

If we reference assemblies that are not part of the base Silverlight class library, these DLL's will also be included in the xap. Adding a RadCalendar to the "Hello World" application requires references to the Telerik.Windows.Controls and Telerik.Windows.Controls.Input assemblies. Once referenced, these assemblies are automatically included in the xap:



Name	Type	Packed Size	Size	Ratio
AppManifest.xaml	Windows Markup File	1 KB	1 KB	59%
SilverlightIntroduction.dll	Application Extension	4 KB	8 KB	59%
Telerik.Windows.Controls.dll	Application Extension	177 KB	672 KB	74%
Telerik.Windows.Controls.Input.dll	Application Extension	142 KB	508 KB	73%

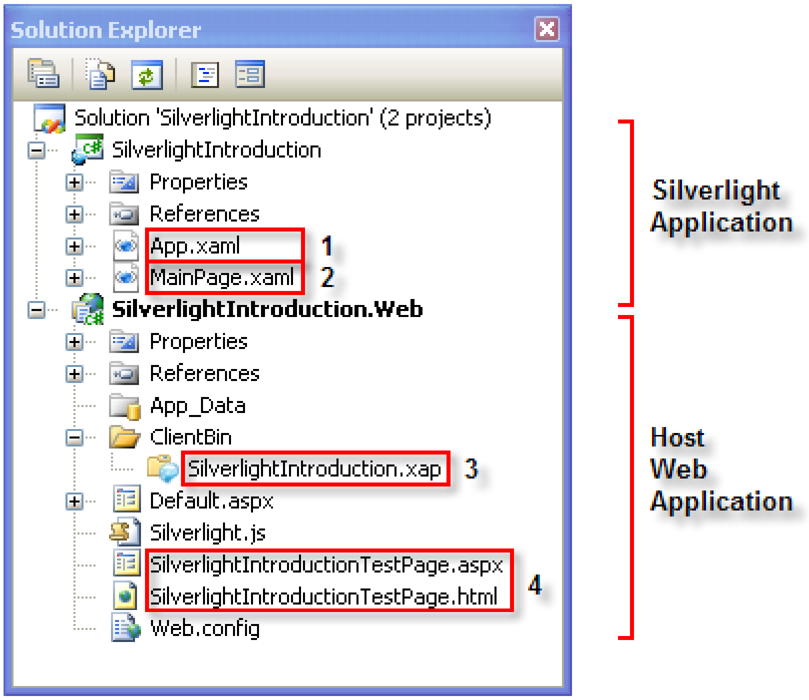
2.3.3 The Application Manifest

The application manifest describes the contents of the xap file and tells Silverlight how to run a particular application. Looking inside the AppManifest.xaml you can see that the **Deployment.Parts** element lists and names each assemblies in the xap. The Deployment **RuntimeVersion** is the version of Silverlight required to run the application. The Deployment **EntryPointAssembly** and **EntryPointType** are the names of the assembly and the class that will be instantiated to start the Silverlight application.

```
<Deployment
 xmlns="http://schemas.microsoft.com/client/2007/deployment"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 EntryPointAssembly="SilverlightIntroduction"
 EntryPointType="SilverlightIntroduction.App"
 RuntimeVersion="3.0.40624.0">
 <Deployment.Parts>
  <AssemblyPart x:Name="SilverlightIntroduction" Source="SilverlightIntroduction.dll" />
  <AssemblyPart x:Name="Telerik.Windows.Controls" Source="Telerik.Windows.Controls.dll" />
  <AssemblyPart x:Name="Telerik.Windows.Controls.Input"
    Source="Telerik.Windows.Controls.Input.dll" />
 </Deployment.Parts>
</Deployment>
```

2.3.4 Silverlight Applications in Visual Studio

When you create a new Silverlight application in Visual Studio, two projects are created, the Silverlight Application project and a Host Web Application project. The screenshot below shows the key elements:



Solution Explorer

```
Solution 'SilverlightIntroduction' (2 projects)
  SilverlightIntroduction
    Properties
    References
    App.xaml
    MainPage.xaml
  SilverlightIntroduction.Web
    Properties
    References
    App_Data
    ClientBin
      SilverlightIntroduction.xap
    Default.aspx
    Silverlight.js
    SilverlightIntroductionTestPage.aspx
    SilverlightIntroductionTestPage.html
    Web.config
```

1. "App" represents the application class. This class handles instantiation and other application level tasks.

2. "MainPage" has xaml to describe the user interface for your application and code-behind to define the logic.

3. After the Silverlight application is compiled, the "xap" is placed in the host web application \ClientBin folder.

4. Standard ASP.NET or HTML test pages are automatically added to the host web application. These pages contain "<object>" tags that reference the Silverlight plugin. The two pages are substantially the same. Looking at the "*.html" file we can see that it contains boilerplate Javascript code for handling Silverlight errors. In the HTML portion of the file below the Javascript, there is an "<object>" element. The object element represents the Silverlight plugin. Notice that it has a series of parameter tags that tell it where the "xap" file can be found and what Javascript should be run if there's an error. "minRuntimeVersion" and "autoUpgrade" are configured to automatically update the plugin if it is out of date.

```
<body>
  <form id="form1" runat="server" style="height:100%">
  <div id="silverlightControlHost">
    <object data="data:application/x-silverlight-2," type="application/x-silverlight-2"
      width="100%" height="100%">
    <param name="source" value="ClientBin/SilverlightIntroduction.xap"/>
    <param name="onError" value="onSilverlightError" />
    <param name="background" value="white" />
    <param name="minRuntimeVersion" value="3.0.40624.0" />
    <param name="autoUpgrade" value="true" />
    <a href="http://go.microsoft.com/fwlink/?LinkID=149156&v=3.0.40624.0"
         style="text-decoration:none">
      <img src="http://go.microsoft.com/fwlink/?LinkId=108181"
       alt="Get Microsoft Silverlight" style="border-style:none"/>
    </a>
    </object>
     <iframe id="_sl_historyFrame" style="visibility:hidden;height:0px;width:0px;border:0px">
     </iframe></div>
  </form>
</body>
```

2.3.5 The Silverlight Life Cycle

The sequence below is a simplified Silverlight Life Cycle.

1) The cycle begins when a user requests a web page that contains a Silverlight application.

2) The host page downloads to the client browser.

3) The browser renders the HTML.

4) The browser reaches the "<object>" element and loads the Silverlight Plugin.

5) The Silverlight Plugin downloads the "xap" file specified in the "source" parameter of the object element.

6) The plugin extracts the xap and reads the manifest.

7) Using information from the manifest, the plugin creates an instance of your application.

8) The default constructor for the App class:

a) Hooks up event handlers for the Startup, Exit and UnhandledException events.

b) Calls InitializeComponent() that loads the application XAML.

9) The plugin fires the application Startup event.

10)The application Startup event handler creates an instance of the MainPage.

11)The constructor for the MainPage calls InitializeComponent() for the page where the XAML for the page is loaded and creates any Silverlight elements defined there.

12)The MainPage instance is assigned to the applications RootVisual property. The RootVisual is the page that the application is displaying at any one time.

2.4 Silverlight Development Tools

Visual Studio and Expression Blend

Both Visual Studio and Expression Blend can be used to build Silverlight applications. In fact, you can work in both environments for the same project and pass the project back in forth. Both environments sense when changes have been made outside their environments and update their interfaces accordingly. Both environments have shortcuts to invoke each other.

Why would you use one over the other? Visual Studio can be thought of as primarily a developers tool set. Visual Studio is the appropriate tool when you need to build disparate parts of a larger solution. Expression Blend can be thought of as a designer's environment. While you can still code in Blend, the toolset leans towards rich media display and animation, all accomplished with visually oriented tools and methods.

Notes

Expect the dichotomy between Visual Studio and Expression Blend to be a temporary state of affairs. Future versions of both environments are likely to overlap to a greater degree as the toolset progresses.

Links to Silverlight development tools and other resources can be found at http://silverlight.net/getstarted/.

Silverlight SDK

The SDK includes the necessary Silverlight assemblies, documentation, Visual Studio templates and examples. This download is required for developing Silverlight applications. The install currently includes the following DLL's:

System.ComponentModel.DataAnnotations.dll

System.Data.Services.Client.dll

System.Json.dll

System.Runtime.Serialization.Json.dll

System.ServiceModel.PollingDuplex.dll

System.ServiceModel.Syndication.dll

System.Windows.Controls.Data.dll

System.Windows.Controls.Data.Input.dll

System.Windows.Controls.dll

System.Windows.Controls.Input.dll

System.Windows.Controls.Navigation.dll

System.Windows.Data.dll

System.Windows.VisualStudio.Design.dll

System.Xml.Linq.dll

System.Xml.Serialization.dll

System.Xml.Utils.dll

WCF RIA Services

"Microsoft WCF RIA Services simplifies the traditional n-tier application pattern by bringing together the ASP.NET and Silverlight platforms. RIA Services provides a pattern to write application logic that runs on the mid-tier and controls access to data for queries, changes and custom operations."

Using WCF RIA services is demonstrated in this Courseware as part of the "Data Binding" chapter.

Silverlight Toolkit

The toolkit has a collection of useful controls and classes. While its not required to build Silverlight applications, some of the resources in this download are used in this Courseware. The toolkit is a community project at located at http://silverlight.codeplex.com.

2.5 Wrap Up

In this chapter you got a feel for where Silverlight fits relative to other web technologies. You were given a rundown on the elements that make up a Silverlight application and how those parts fits in a Visual Studio project. You learned about the basic Silverlight Life Cycle. Finally, you learned about the development tools available for Silverlight development and where each tool is used.

Part III

Working with Silverlight

3 Working with Silverlight

3.1 Objectives

This chapter is intended to get you started working with Silverlight. This is not intended to be a comprehensive coverage of all things Silverlight, but rather a quick guide to basic "up-and-running" information. In this chapter you will build simple "Hello World" Silverlight applications using both Visual Studio and Expression Blend. You will also learn how to hand off projects back and forth between the two environments and in the process learn the basic working styles of both. In this chapter you will learn the basics about XAML, including XML namespaces, defining objects, properties and collections in XAML, attached properties, markup extensions, how to define and apply styles to Silverlight elements, how to define and use resources and how RadControls are defined and themed in XAML.

You will learn how templates are used to create free form arrangements of Silverlight elements without affecting underlying functionality. You will learn the basics on how to bind data to Silverlight elements.

You will learn about Silverlight best practices including MVVM and Prism. Finally, you will learn the settings for debugging a Silverlight application.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\WorkingWithSilverlight\WorkingWithSilverlight.sln.

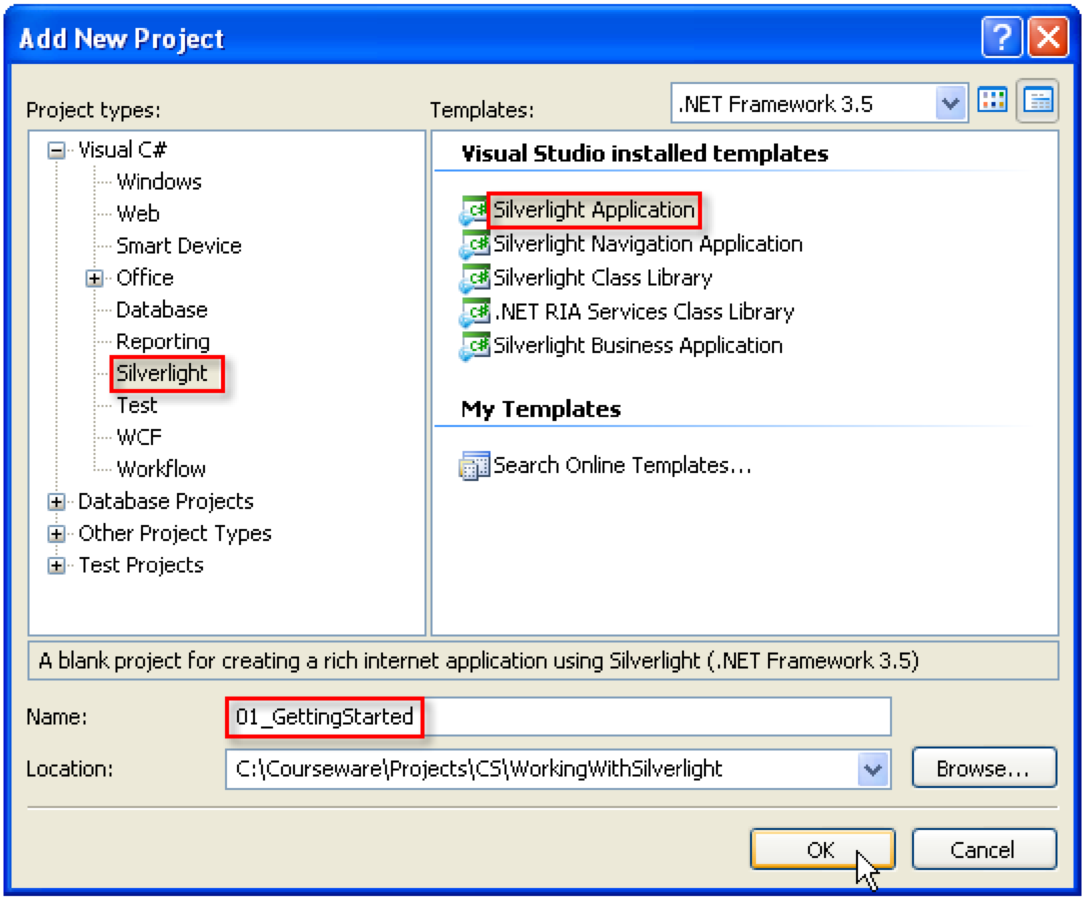
3.2 Getting Started

3.2.1 Starting with Visual Studio

This walk through will take you step-by-step building a classic "Hello World" Silverlight application.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.



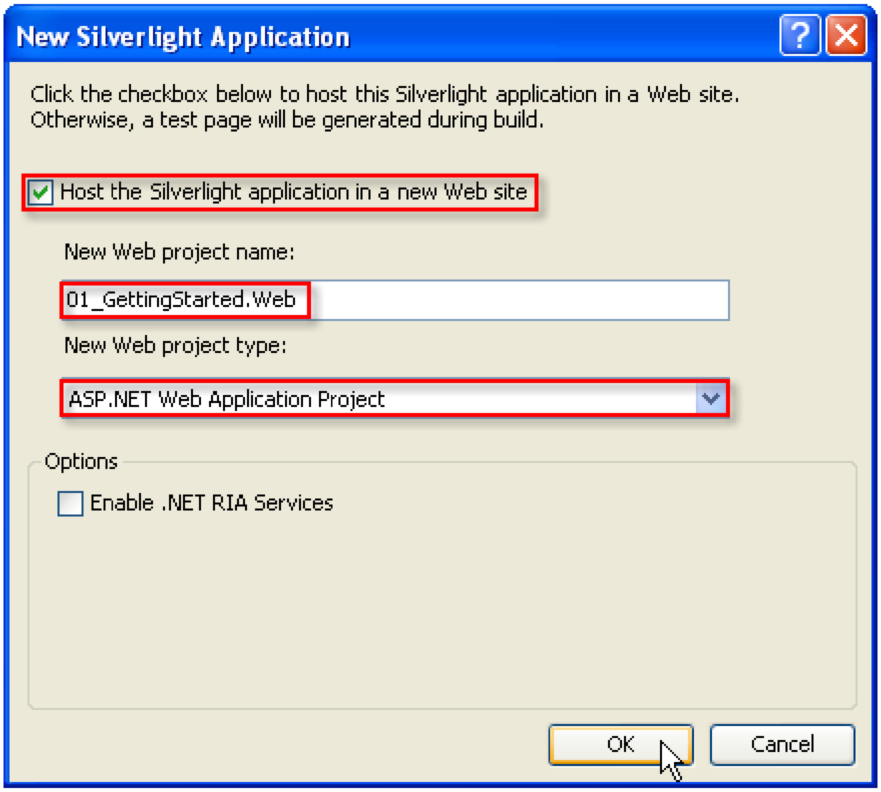
Templates: .NET Framework 3.5

A blank project for creating a rich internet application using Silverlight (.NET Framework 3.5)

Name: 01_GettingStarted

Location: C:\Courseware\Projects\CS\WorkingWithSilverlight

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.



Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new Web site

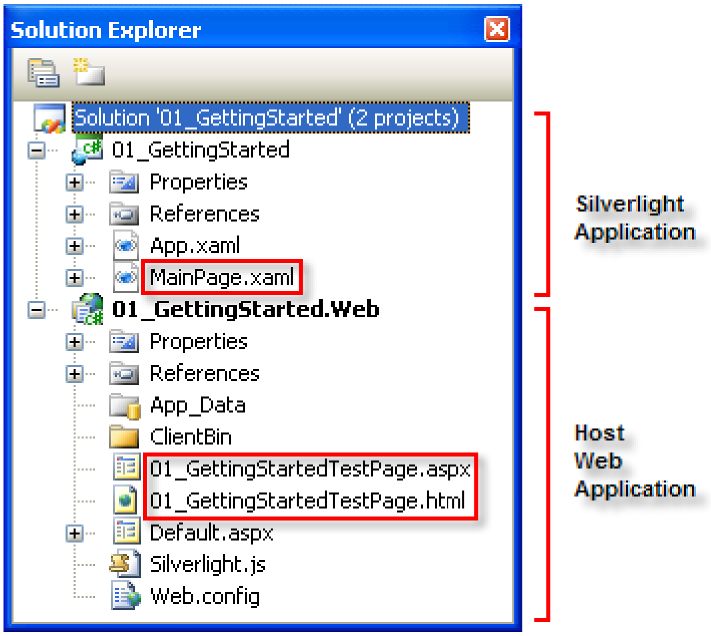
New Web project name: 01_GettingStarted.Web

New Web project type: ASP.NET Web Application Project

Options

□ Enable .NET RIA Services

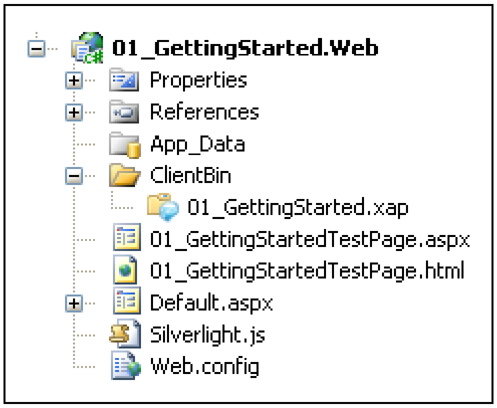
3) Review the contents of the solution in the Solution Explorer. There are two projects. The Silverlight Application project contains "MainPage" where MainPage.xaml contains the markup for the user interface and "MainPage.xaml.cs" is the code-behind that defines the client-side logic. The host project contains two test pages: an "aspx" test page and a "html" test page.



4) In the Solution Explorer, right-click the host project and select "Set as Startup Project" from the context menu.

5) In the Solution Explorer, right-click the host project "aspx" test page and select "Set as Start Page".

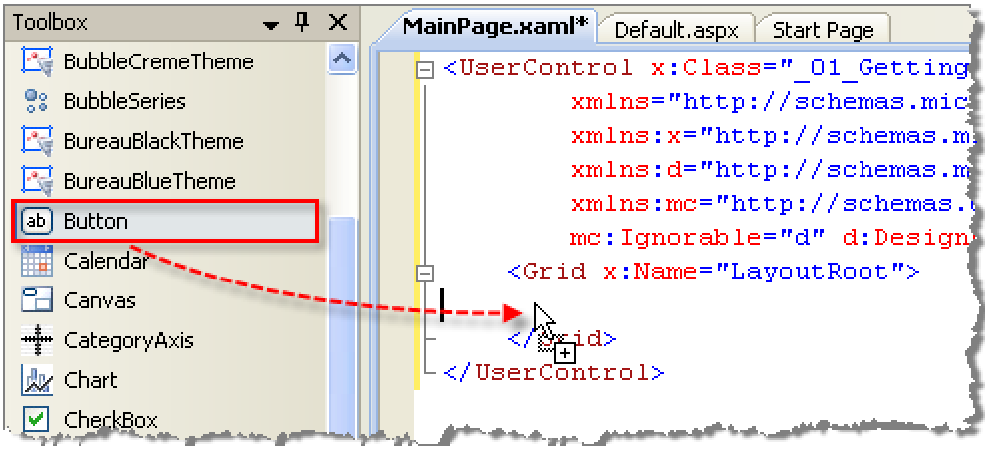
6) Build the solution. In the Solution Explorer, find the "ClientBin" folder and notice that a "xap" file has been created



7) Navigate to the Silverlight project and open MainPage.xaml for editing. It should contain markup similar to the example shown below.

```
<UserControl x:Class="_01_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"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
  <Grid x:Name="LayoutRoot">

  </Grid>
</UserControl>
```

8) Drag a **Button** control from the "Silverlight XAML Controls" tab of the Toolbox to a point between the main Grid element tags.



```
<UserControl x:Class="_01_Getting
        xmlns="http://schemas.mic
        xmlns:x="http://schemas.m
        xmlns:d="http://schemas.m
        xmlns:mc="http://schemas.
        mc:Ignorable="d" d:Design
    <Grid x:Name="LayoutRoot">

    </Grid>
</UserControl>
```

9) Replace the Button tag with the XAML shown below.

```
<Button Content="Click Me" Click="Button_Click"></Button>
```

10)The completed XAML should now look like the example below:

XAML

```
<UserControl x:Class="_01_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"
      mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
   <Grid x:Name="LayoutRoot">
      <Button Content="Click Me" Click="Button_Click"></Button>
   </Grid>
</UserControl>
```

11)Press **F7** to navigate to the code behind.

12)Locate the button Click event handler and insert a call to the static MessageBox.Show() method as shown in the example below.

VB

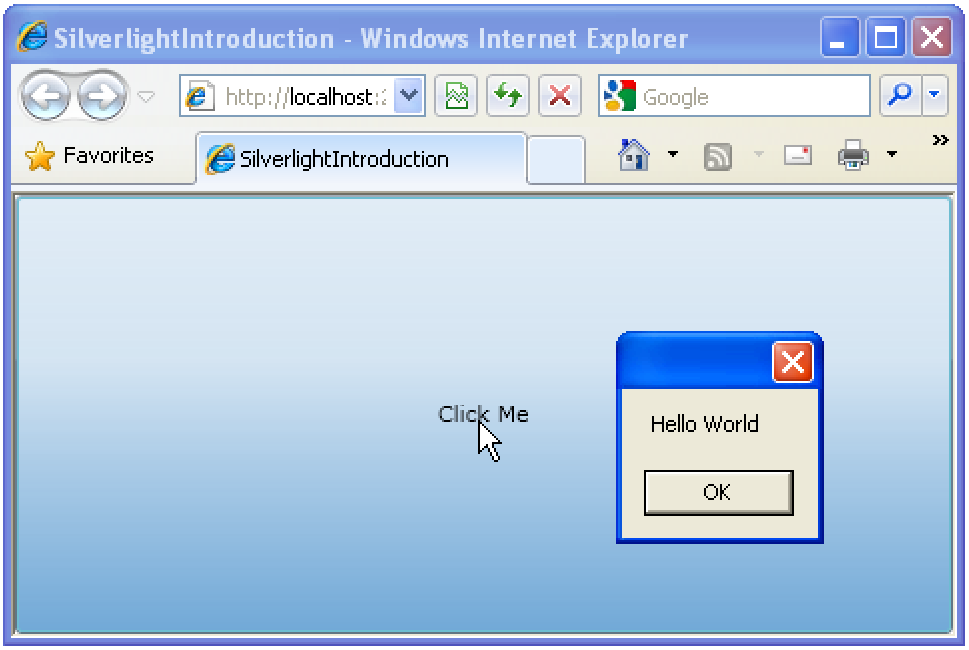
```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  MessageBox.Show("Hello World")
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   MessageBox.Show("Hello World");
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.

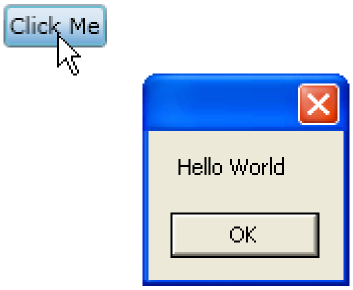


Modify the Application

By default, the Button stretches to take up the entire width and height of the screen. Hardly optimal but easily fixed. Return to the MainPage.xaml file to continue editing. Replace the Button element with the markup example below. These changes will align the button to the top left of the page and move it 20 pixels off all sides.

```
<Button Content="Click Me" Click="Button_Click"
    HorizontalAlignment="Left" VerticalAlignment="Top"
    Margin="20"></Button>
```

Press **F5** to re-run the application which should now look like this:



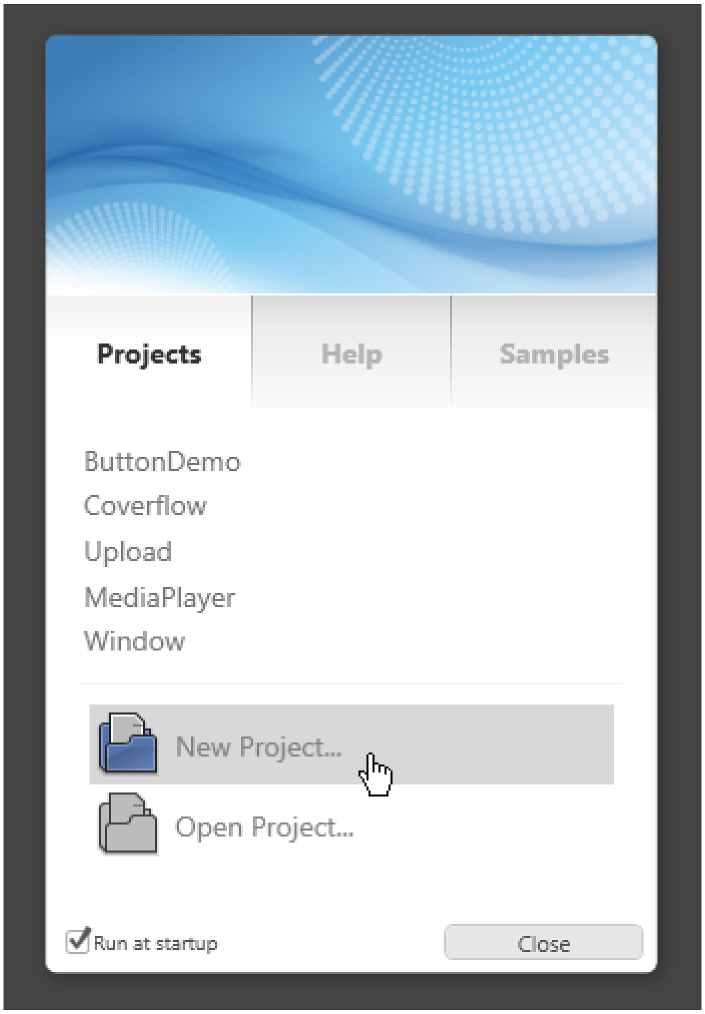
3.2.2 Starting with Expression Blend

This walk through will take you step-by-step building a classic "Hello World" Silverlight application in Expression Blend. You will also get a chance to contrast the working styles of Expression Blend and Visual Studio from the previous example.

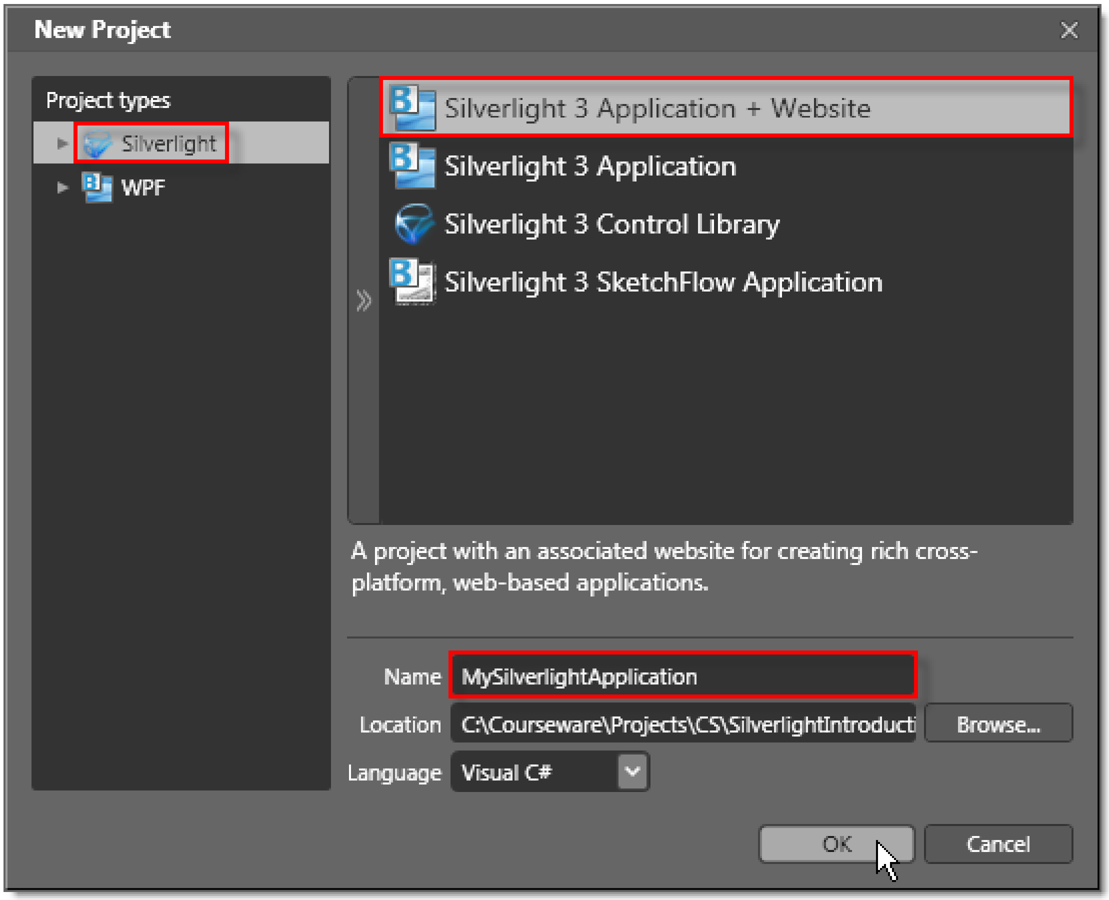
Project Setup

1) Run Expression Blend.

2) From the **File** menu select **New Project**. If Expression Blend is configured to show the startup dialog, click the **New Project...** option.



3) In the "New Project" dialog select the "Silverlight" project type. In the right hand list select the "Silverlight 3 Application + Website". Provide a unique name for the application and click **OK** to close the dialog.



A project with an associated website for creating rich cross-platform, web-based applications.

Name: MySilverlightApplication

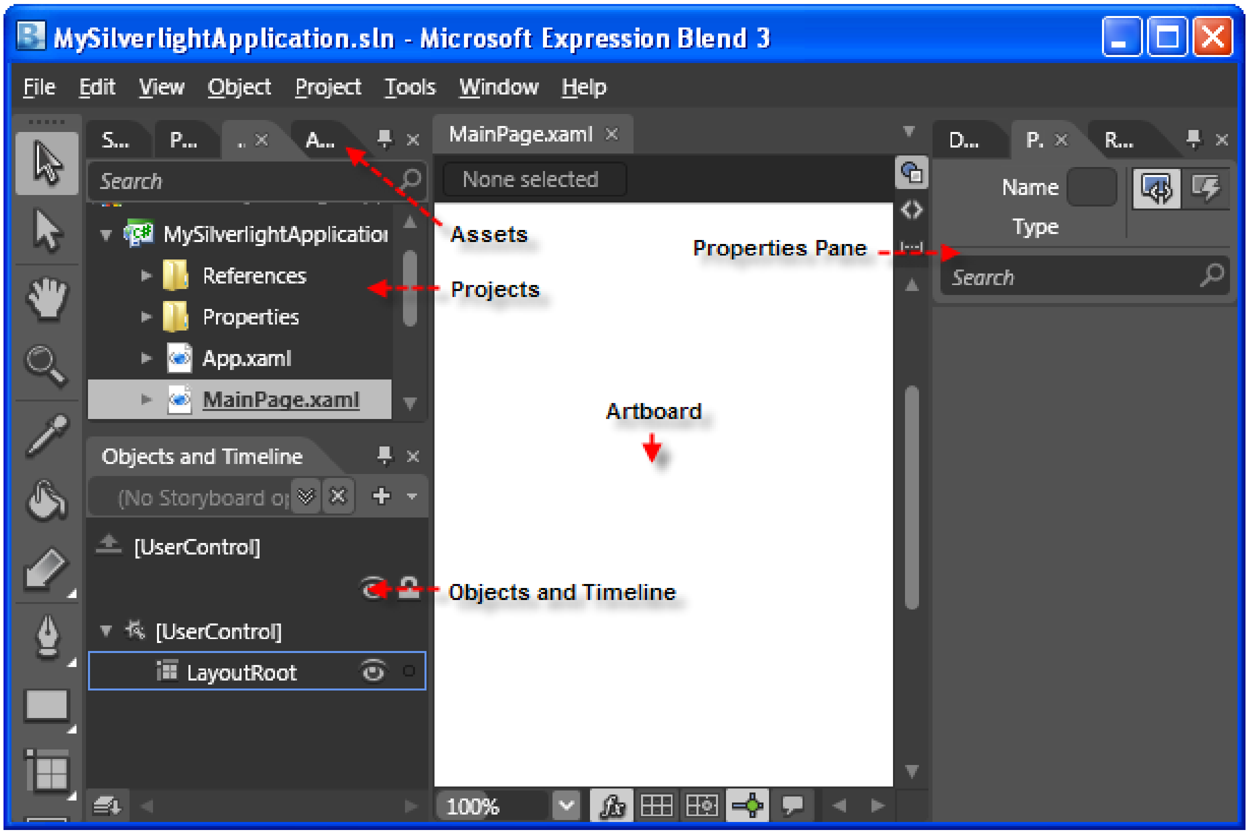
Location: C:\Courseware\Projects\CS\SilverlightIntroducti

Language: Visual C#

New Project
Project types
Silverlight
WPF
Silverlight 3 Application + Website
Silverlight 3 Application
Silverlight 3 Control Library
Silverlight 3 SketchFlow Application
Browse...
OK
Cancel

4) When Expression Blend displays there are quite a number of features available at once. The key features we will use right away are the Assets pane, the Projects pane, the Objects and Timeline pane, the Artboard and the Properties pane.

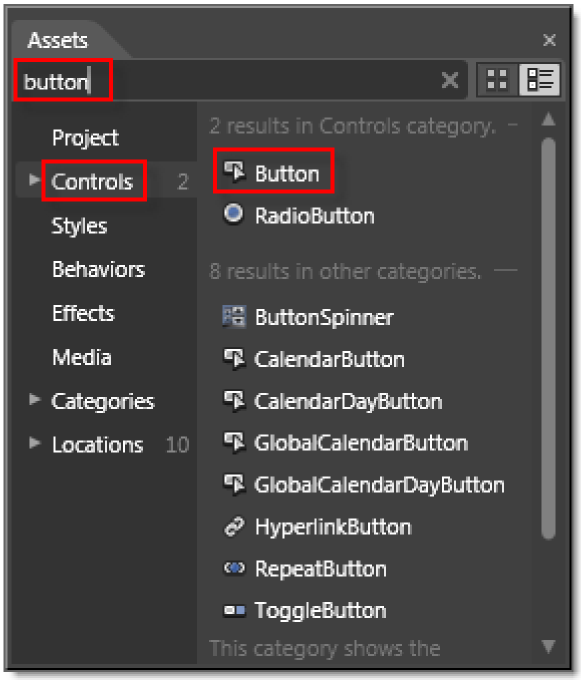
The **Assets pane** can be thought of as a smart Toolbox that makes it easy to search for controls but also contains effects, styles, behaviors and media. The **Projects pane** is analogous to the Solution Explorer and contains the projects, files and other resources for a solution. The **Objects and Timeline pane** displays the visual tree of Silverlight elements in outline form and helps orient you even when the page is very complex. The **Artboard** is the design surface for the page where controls are added and modified. The **Properties pane** is much like the Visual Studio Properties window but includes extensive tools for visual properties such as alignment or color gradients.



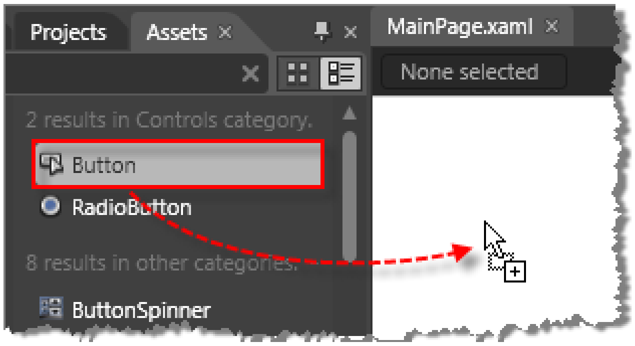
5) Open the Assets pane.

6) On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.

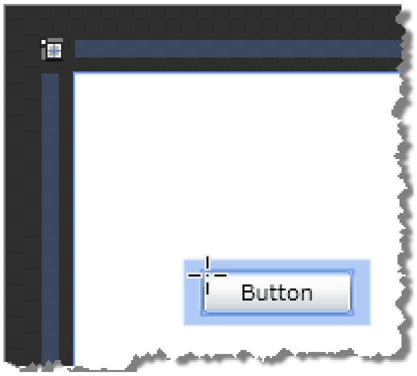
7) In the Assets pane, just above the tree view is the Assets Find entry text box. Enter "button" and notice that the list of controls narrows as you type.



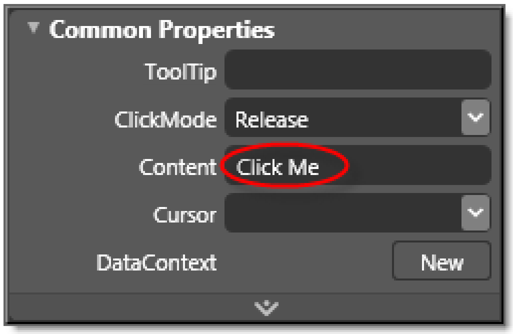
8) From the Assets pane, drag the Button control onto the Artboard.



9) Looking at the Button in the Artboard you can see that it does not stretch to take up the entire page and remains at the location on the page where you dropped it.



10)Look at the Properties pane, locate the Layout tab and notice that the Width, Height, Horizontal Alignment, Vertical Alignment and Margin have all been set. Locate the Common Properties tab and set the **Content** property to "Click Me". ***Note****: You can also set the text on the control directly in the Artboard by double-clicking or selecting the control and clicking F2.*



Common Properties

ToolTip

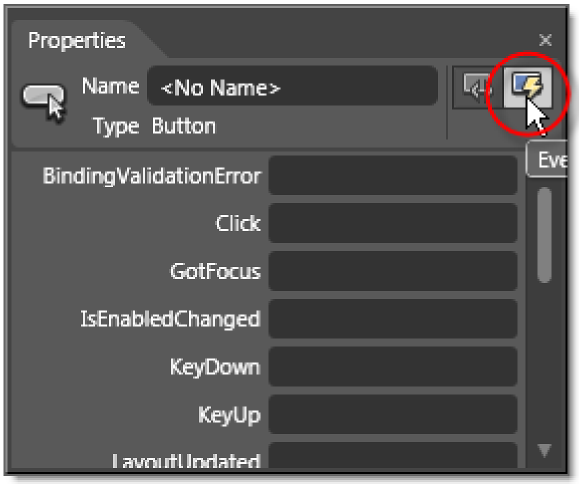
ClickMode Release

Content Click Me

Cursor

DataContext New

11)Locate the Events button in the Properties pane and click it.



Properties

Name <No Name>

Type Button

BindingValidationError

Click

GotFocus

IsEnabledChanged

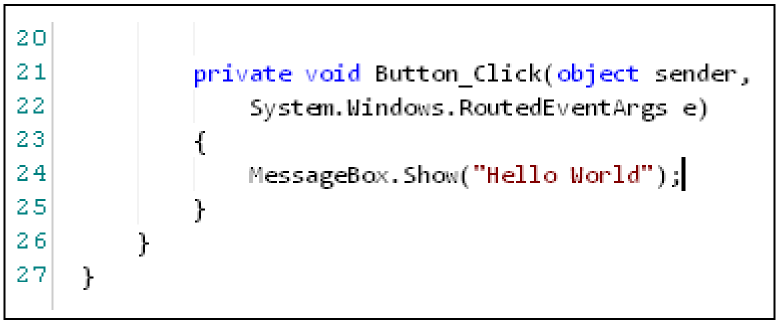
KeyDown

KeyUp

LayoutUpdated

12)Double-click the "Click" event to create an event handler. This will navigate you to the Expression Blend code editor.

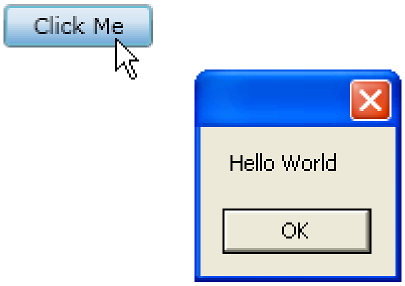
13)In the code editor, add a call to MessageBox.Show() inside the Click event handler.



```

        private void Button_Click(object sender,
            System.Windows.RoutedEventArgs e)
        {
            MessageBox.Show("Hello World");
        }
    }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Hello World

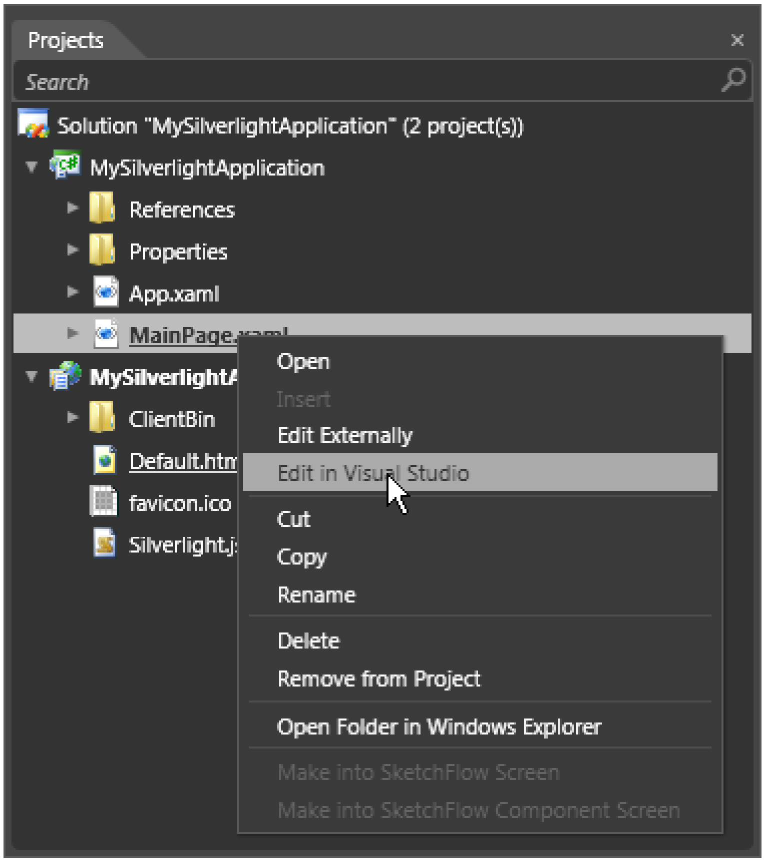
3.2.3 Visual Studio With Expression Blend

The process of building the same application in two environments provides a glimpse of the stylistic differences between the two environments. This is a glimpse only. As we use Expression Blend more extensively you will see how it excels when handling design tasks, animation and rich media. Visual Studio and Expression Blend are designed to work together, handing off the project to the best suited environment for a given task.

Let's walk through how the hand off occurs, starting with the previous Expression Blend project.

1) Start with the previous Expression Blend project.

2) In the Projects pane, locate MainPage.xaml, right-click and select **Edit in Visual Studio** from the context menu. Do **NOT** close Expression Blend. We will want to test how the two environments are notified of changes by the other.



3) Visual Studio will start and automatically open the entire Silverlight project, including the host web application.

4) In Visual Studio, drag an image file "Folder_open.png" to the Silverlight project. You can find this file in the "\courseware\images" directory.

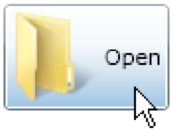
5) Open MainPage.xaml for editing. Replace the Button with the XAML below.

This version of the Button replaces the Content attribute with Content as a full sub-element. Inside the Content element, a StackPanel contains an Image and a TextBlock.

```
<Button HorizontalAlignment="Left" Margin="61,99,0,0"
      VerticalAlignment="Top" Click="Button_Click">
   <Button.Content>
      <StackPanel Orientation="Horizontal">
         <Image Source="Folder_open.png" Width="48"
               Height="48" />
         <TextBlock Text="Open" VerticalAlignment="Center"
               HorizontalAlignment="Center" />
      </StackPanel>
   </Button.Content>
</Button>
```

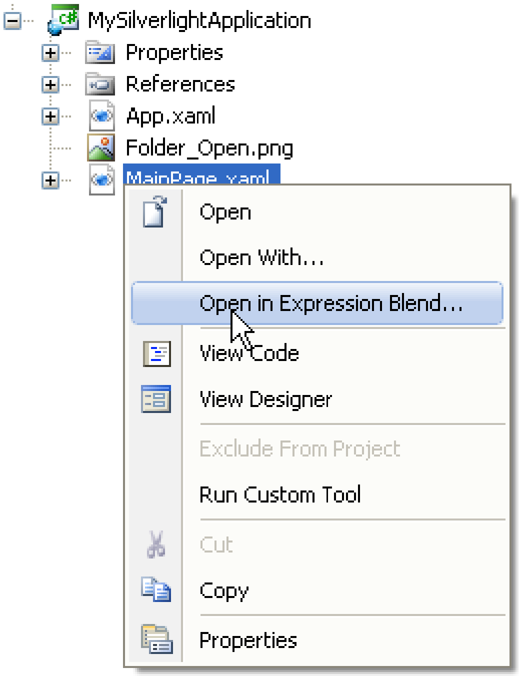
6) Select **File > Save All** from the Visual Studio menu.

7) Press **F5** to run the application. The Button in the Silverlight application running from Visual Studio now looks like the screenshot below.

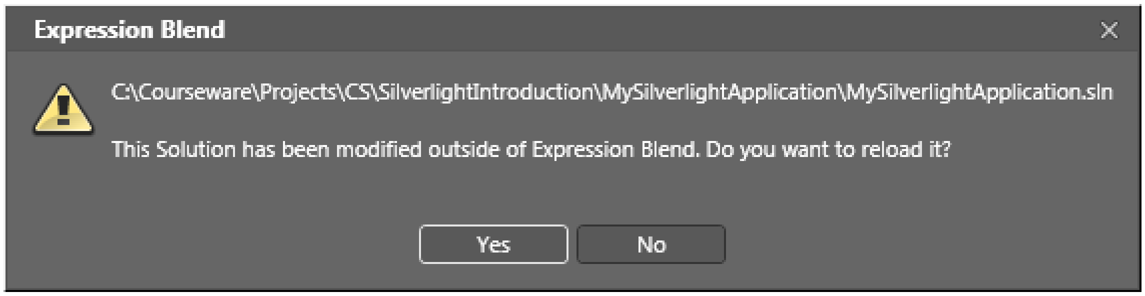


8) Close the running Silverlight Application.

9) In the Visual Studio Solution Explorer, right-click MainPage.xaml and select **Open in Expression Blend...** from the context menu.



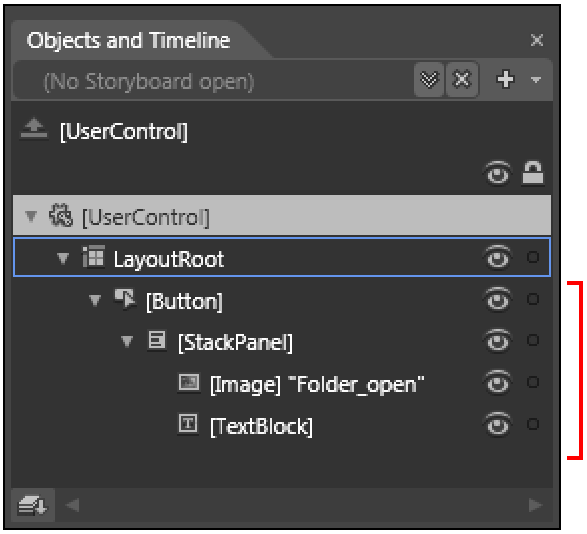
10)Expression Blend will come to the front and display a dialog "This Solution has been modified outside of Expression Blend. Do you want to reload it?". Click the **Yes** button to close the dialog and reload the project.



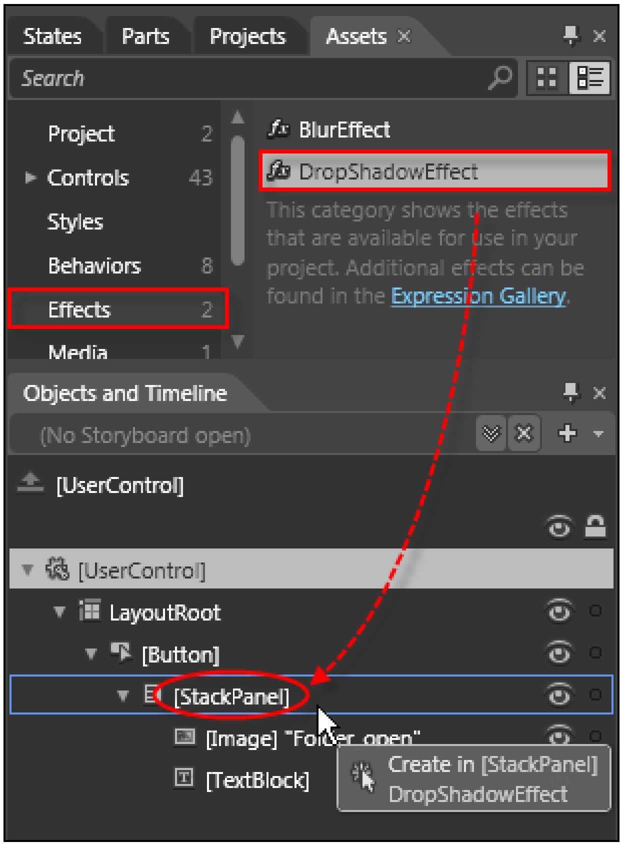
C:\Courseware\Projects\CS\SilverlightIntroduction\MySilverlightApplication\MySilverlightApplication.sln

This Solution has been modified outside of Expression Blend. Do you want to reload it?

11)Notice how the Objects and Timeline pane reflects the new Button content.



12)In the Assets pane, select the "Effects" node in the tree on the left. Locate the "Drop Shadow Effect" effect, drag it to Objects and Timeline pane and drop it on the StackPanel.



This category shows the effects that are available for use in your project. Additional effects can be found in the Expression Gallery.

Notes

Keep your eye on the tooltip as you drag assets to the Objects and Timeline pane. The tooltip will change to let you know what container it will be dropped into.

13)Press **F5** to run the application. The shadow effect added by Expression Blend shows on the Image and Text elements.

From this walk through you have seen how the project is passed back and forth between Visual Studio and Expression Blend.

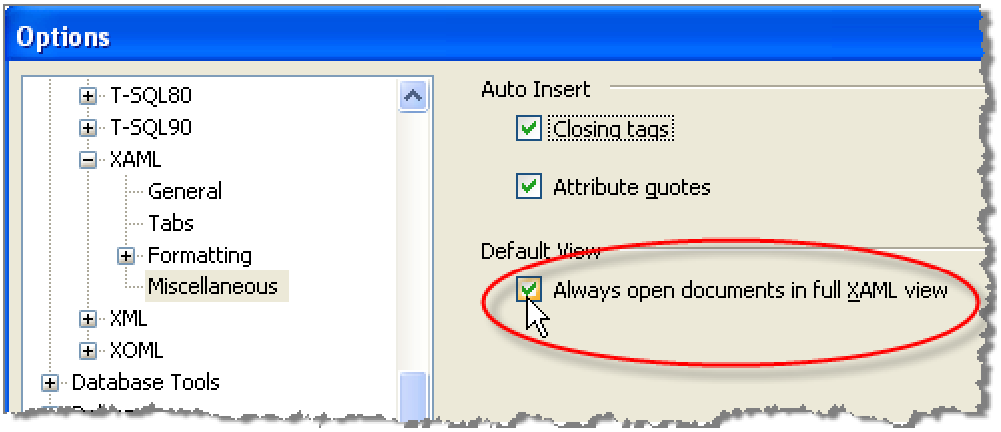
3.3 Working with XAML

3.3.1 XAML Basics

In this section we will look at how to think about XAML syntax, how to reference assemblies, use IntelliSense, create styles, build templates and refactor your XAML with resources.

Gotcha!

Before working with XAML in Visual Studio you can tweak the environment to make editing a little faster. Go to **Tools > Options... > Text Editor > XAML > Miscellaneous**. Set the "Always open documents in full XAML view" option on. Now when you open a XAML file the read-only Visual Designer view will not load. It is substantially quicker to edit XAML in this manner for Visual Studio 2008. Use Expression Blend when you need to visually edit your pages.



Auto Insert

☑ Closing tags

☑ Attribute quotes

Default View

☑ Always open documents in full XAML view

XAML is a declarative markup language that simplifies creating user interfaces. XAML elements directly represent instantiation of objects. The visible elements are defined in XAML, then matching code-behind is used to manipulate those objects. This separation of concerns allows a work flow where separate parties can work on UI and logic at the same time, using different tools.

MainPage.xaml

Your typical starting point for a XAML file is the either the App.xaml or MainPage.xaml. In App.xaml you can define resources for the entire application. We will discuss resources a little later but for now take a look at MainPage.xaml. The page starts out looking like the example below:

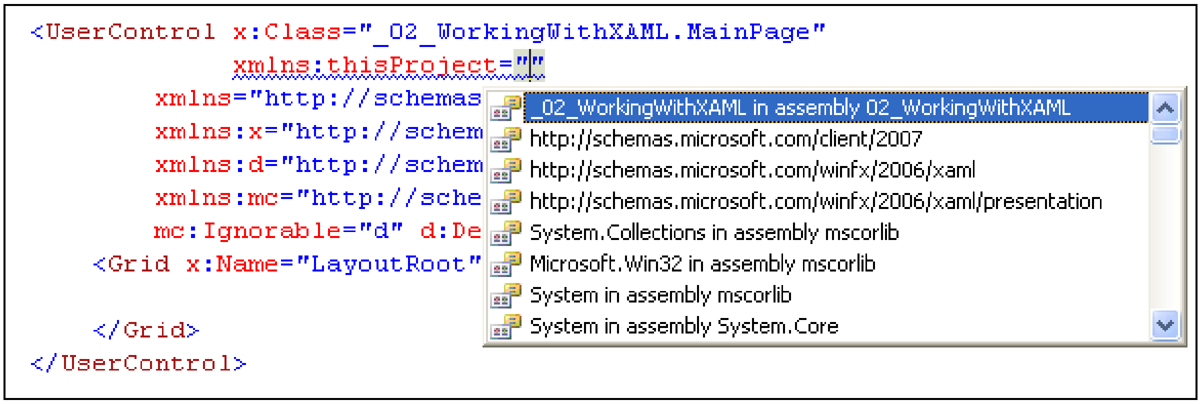
```
<UserControl x:Class="_02_WorkingWithXAML.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
  <Grid x:Name="LayoutRoot">

  </Grid>
</UserControl>
```

The root element is "UserControl". The "x:Class" attribute lets us know the name of the UserControl class in the code-behind, i.e. "MainPage".

XML Namespaces

Following the class declaration is a series of "xmlns" or XML namespaces. The syntax is "xmlns" followed by a name, an equals sign, then the namespace itself. The first "xmlns" has no name and is the default namespace.

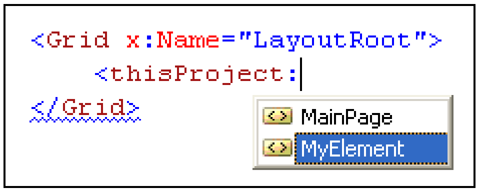
What happens if we want to include references to assemblies, similar to "Imports" (VB) or "using" (C#) statements? For that we can type in "xmlns:" plus a name and an equals sign. IntelliSense will display a list of assemblies in scope for the project. The screenshot below shows a namespace called "thisProject". The list drop down to reveal all the possible namespaces in scope.



```
<UserControl x:Class="_02_WorkingWithXAML.MainPage"
             xmlns:thisProject=""
        xmlns="http://schemas
        xmlns:x="http://schem
        xmlns:d="http://schem
        xmlns:mc="http://sche
        mc:Ignorable="d" d:De
    <Grid x:Name="LayoutRoot"

    </Grid>
</UserControl>
```

- _02_WorkingWithXAML in assembly 02_WorkingWithXAML
- http://schemas.microsoft.com/client/2007
- http://schemas.microsoft.com/winfx/2006/xaml
- http://schemas.microsoft.com/winfx/2006/xaml/presentation
- System.Collections in assembly mscorlib
- Microsoft.Win32 in assembly mscorlib
- System in assembly mscorlib
- System in assembly System.Core

If you have a class called "MyElement" defined in the code-behind, you can reference it in XAML by starting your tag and adding "thisProject:" namespace. Once you type a colon, IntelliSense will display a list of objects in the namespace as shown in the screenshot below.



```
<Grid x:Name="LayoutRoot">
    <thisProject:
</Grid>
```

Defining Objects in XAML

XAML elements are used to declare instances of a classes. Elements can be expressed as starting and ending XML tags or can use the shorter "self closing" syntax. Here is an example that shows both syntaxes:

```
<!--using start and end tags...-->
<Button></Button>
<!--or self closing syntax-->
<Button />
```

Defining Properties in XAML

Properties are defined as attributes whose values are always strings. In the example below, Button Width and Height attributes are both assigned the string value "100".

```
<Button Width="100" Height="100"></Button>
```

Properties can also be assign with "property element", "content element" and "collection" syntax. Property element syntax takes the form of "<typeName.PropertyName>". The Width and Height example below makes the XAML more verbose in this case, but if properties are complex or can't be expressed in a single string value, this syntax becomes useful.

```
<Button >
  <Button.Width>100</Button.Width>
  <Button.Height>100</Button.Height>
</Button>
```

Another example shows property element syntax where the Button's BorderBrush contains another object called a SolidColorBrush.

```
<Button>
  <Button.BorderBrush>
    <SolidColorBrush Color="Red"/>
  </Button.BorderBrush>
</Button>
```

For objects that have Content, the syntax can be simplified to place the content between the tags as a default.

```
<Button>Hello World</Button>
```

And lastly, collection syntax allows you to list a number of elements and have them automatically added to a collection. The XAML below shows a LinearGradientBrush object that has a GradientStop collection.

```
<Button>
  <Button.Background>
    <LinearGradientBrush>

      <LinearGradientBrush.GradientStops>
        <GradientStopCollection>
          <GradientStop Offset="0.0" Color="Red" />
          <GradientStop Offset="1.0" Color="Blue" />
        </GradientStopCollection>
      </LinearGradientBrush.GradientStops>

    </LinearGradientBrush>
  </Button.Background>
</Button>
```

The parser knows how to locate and create a GradientStopCollection, so we can leave the GradientStopCollection element out for a shorter syntax:

```
<Button>
  <Button.Background>
    <LinearGradientBrush>

      <LinearGradientBrush.GradientStops>
        <GradientStop Offset="0.0" Color="Red" />
        <GradientStop Offset="1.0" Color="Blue" />
      </LinearGradientBrush.GradientStops>

    </LinearGradientBrush>
  </Button.Background>
</Button>
```

It turns out that GradientStops is also the content property for the class and so we can simplify even further:

```
<Button>
  <Button.Background>
    <LinearGradientBrush>
      <GradientStop Offset="0.0" Color="Red" />
      <GradientStop Offset="1.0" Color="Blue" />
    </LinearGradientBrush>
  </Button.Background>
</Button>
```

3.3.2 Attached Properties and Events

A XAML language feature allows assigning a property or event from any owning element, even if the element that's being assigned doesn't have that property or event. You can think of an attached property as being essentially a global property. The syntax is "ownerType.propertyName". The example below demonstrates the utility of attached properties. The Button class has no ToolTip property. Button doesn't need one because we can attach a ToolTip property from the ToolTipService class.

```
<Button ToolTipService.ToolTip="Click for Summary">
</Button>
```

3.3.3 Markup Extensions

"Markup extensions" is an XAML concept where the curly braces "{}" are used as escape characters for syntax that differs from XML standard notation. Usually these refer to resources or binding (resources and binding are discussed in upcoming sections). Here's a brief example of what markup extensions look like in the wild:

```
<TextBlock
  Text="{Binding ProductNumber}"
  Style="{StaticResource MyTextStyle}">
</TextBlock>
```

3.3.4 Resources and Styles

Resources are an extremely powerful feature of XAML that let us define styles, templates, brushes, classes, properties and events in a separate location from their use. This allows resources to be reusable. As a side effect, you can refactor your XAML so that as a page becomes burdened with larger amounts of markup, chunks of the markup can be named and moved to separate files for reuse.

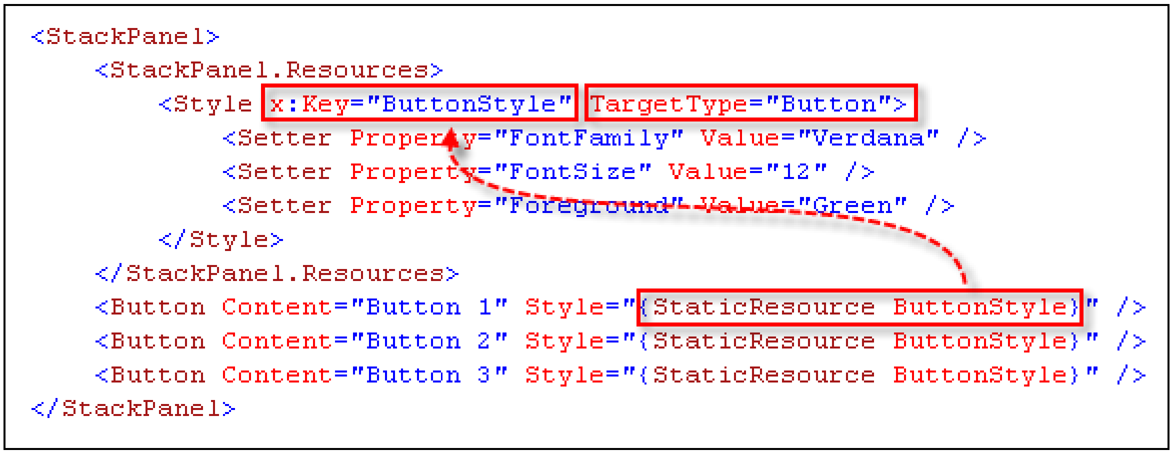
Resources are kept in **ResourceDictionary** objects and are properties of the Application and any ***FrameworkElement*** (any element participating in Silverlight layout). What do you store in a resource? Brushes for setting element colors, templates for customizing controls and data source objects can all be kept in resources. One common use for resource dictionaries is to store ***Styles,*** i.e. named groups of properties and values. Lets start with a Button that has several font related properties. Styles aren't used In the example below where the attributes are described directly on the button element.

```
<Button Content="Click Me!" FontFamily="Verdana" FontSize="12"
 Foreground="Green"></Button>
```

You can convert the font related properties into a single **Style**. For the moment, this is more verbose and isn't more useful than the previous example. Wait for it...

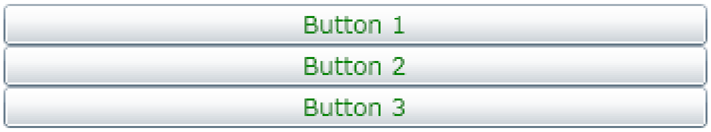
```
<Button Content="Click Me!">
  <Button.Style>
    <Style>
      <Setter Property="FontFamily" Value="Verdana" />
      <Setter Property="FontSize" Value="12" />
      <Setter Property="Foreground" Value="Green" />
    </Style>
  </Button.Style>
</Button>
```

If we make the Style into a Resource, the stye can be used over and over. The button markup gets cleaned up in the process. The Resources element in the example below is a sub property of the Grid, but could have been placed in UserControl.Resources or App.Resources. Notice that the Style has two new attributes that are very important, **x:Key** and **TargetType**. "x:Key" identifies the style so that we can refer to it later. TargetType specifies the type that the style should be applied to. Also notice how the Style is applied to each of the buttons. The curly braces enclose "StaticResource" followed by the style key.



```
<StackPanel>
    <StackPanel.Resources>
        <Style x:Key="ButtonStyle" TargetType="Button">
            <Setter Property="FontFamily" Value="Verdana" />
            <Setter Property="FontSize" Value="12" />
            <Setter Property="Foreground" Value="Green" />
        </Style>
    </StackPanel.Resources>
    <Button Content="Button 1" Style="{StaticResource ButtonStyle}" />
    <Button Content="Button 2" Style="{StaticResource ButtonStyle}" />
    <Button Content="Button 3" Style="{StaticResource ButtonStyle}" />
</StackPanel>
```

In the running Silverlight example, the styled buttons look like the screenshot below:



Gotcha!

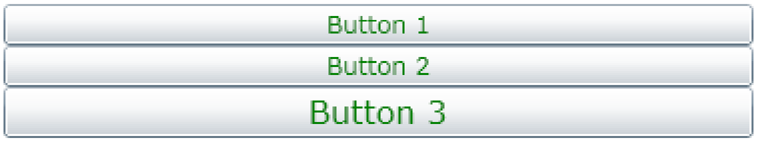
Its very easy to forget the TargetType attribute for a Style. The error can be hard to debug because the error message does not point straight back to the cause. If we forget TargetType in the above example, an XMLParseException is generated with message "Invalid attribute value FontFamily for property Property".

Inheriting Styles

If several styles have some common attributes, why repeat these attributes in every Style definition? Styles can become large, complex and difficult to manage. At some point you will want to use an existing style as a base for a new style. In the example below, the "BigButtonStyle" uses the **BasedOn** attribute to point back to "ButtonStyle". In the example we overwrite the FontSize to "16", and all the other "ButtonStyle" properties come along for the ride. The new "BigButtonStyle" is applied to the third button.

```
<StackPanel>
  <StackPanel.Resources>
    <Style x:Key="ButtonStyle" TargetType="Button">
      <Setter Property="FontFamily" Value="Verdana" />
      <Setter Property="FontSize" Value="12" />
      <Setter Property="Foreground" Value="Green" />
    </Style>
    <Style x:Key="BigButtonStyle" TargetType="Button"
       BasedOn="{StaticResource ButtonStyle}">
      <Setter Property="FontSize" Value="16" />
    </Style>
  </StackPanel.Resources>
  <Button Content="Button 1" Style="{StaticResource ButtonStyle}" />
  <Button Content="Button 2" Style="{StaticResource ButtonStyle}" />
  <Button Content="Button 3" Style="{StaticResource BigButtonStyle}" />
</StackPanel>
```

In the running Silverlight example, the styled buttons look like the screenshot below.



Resource Scope

Where resources are placed in the XAML determine scope. The example below fails with an XMLParseException, "Cannot find a Resource with the Name/Key ButtonStyle", because the last button can't find "ButtonStyle". "ButtonStyle" is a resource declared inside the StackPanel resources and is out of scope for the last button.

```
<StackPanel>
  <StackPanel.Resources>
    <Style x:Key="ButtonStyle" TargetType="Button">
      <Setter Property="FontFamily" Value="Verdana" />
      <Setter Property="FontSize" Value="12" />
      <Setter Property="Foreground" Value="Green" />
    </Style>
  </StackPanel.Resources>
  <Button Content="Button 1" Style="{StaticResource ButtonStyle}" />
  <Button Content="Button 2" Style="{StaticResource ButtonStyle}" />
</StackPanel>

<Button Content="Button 3" Style="{StaticResource ButtonStyle}" />
```

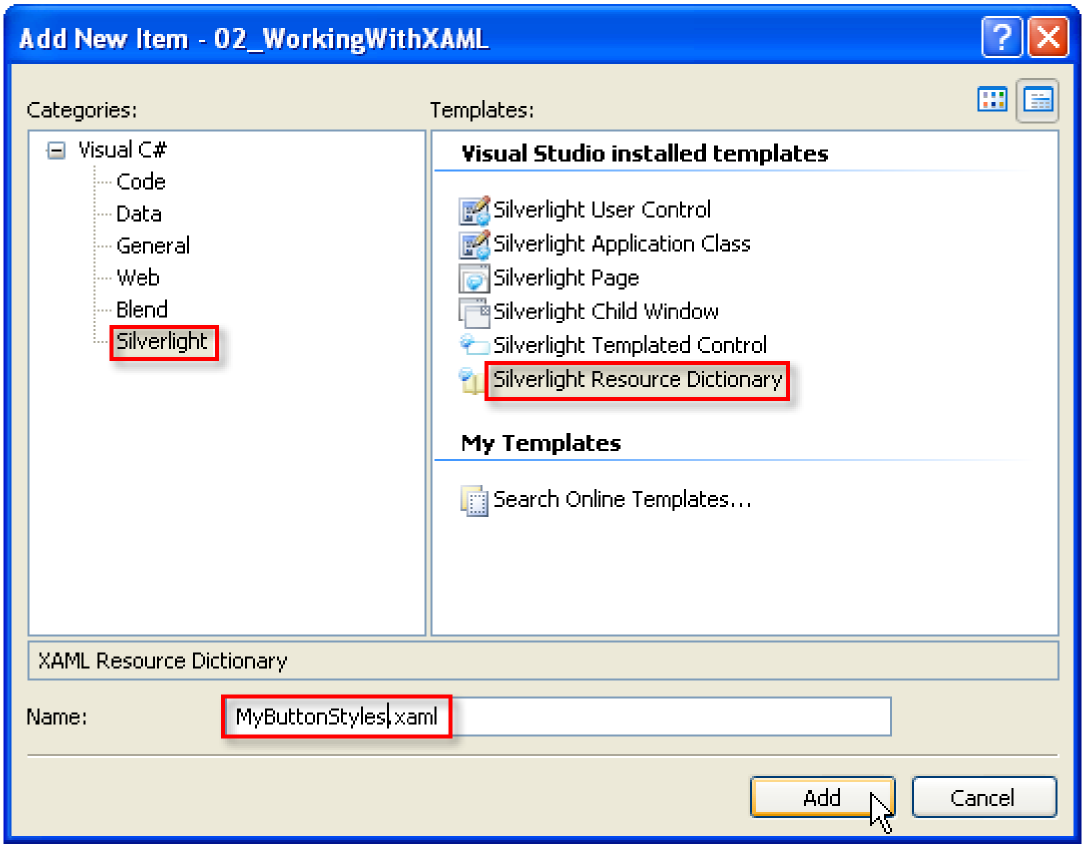
You can also move your resources out to the App.xaml file where they can be reused by the entire application. All you are doing is moving the resource to a wider scope. After moving "ButtonStyle" to App.xaml, App.xaml looks something like the example below:

```
<Application
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      x:Class="_02_WorkingWithXAML.App">
   <Application.Resources>
      <Style x:Key="ButtonStyle" TargetType="Button">
         <Setter Property="FontFamily" Value="Verdana" />
         <Setter Property="FontSize" Value="12" />
         <Setter Property="Foreground" Value="Green" />
      </Style>
   </Application.Resources>
</Application>
```

Merging Resources

Silverlight 3 introduced a feature called "Merged Resources" where resources can be broken out into several files. This feature allows you to better organize your project as it becomes larger. For the sake of discussion, lets place all our button related styles into their own resource dictionary.

To add a new resource dictionary using the Solution Explorer, right-click the project and select **Add > New Item...** from the context menu. You can find the "Silverlight Resource Dictionary" template in the "Silverlight" category. Provide a unique name for the resource dictionary and click **Add** to create the file.



Add New Item - 02_WorkingWithXAML

Categories: Visual C# > Silverlight

Templates: Silverlight Resource Dictionary

XAML Resource Dictionary

Name: MyButtonStyles.xaml

You can create as many resource files as you require to organize your application. The styles we have defined previously can be moved to this file where the contents now look like the example below.

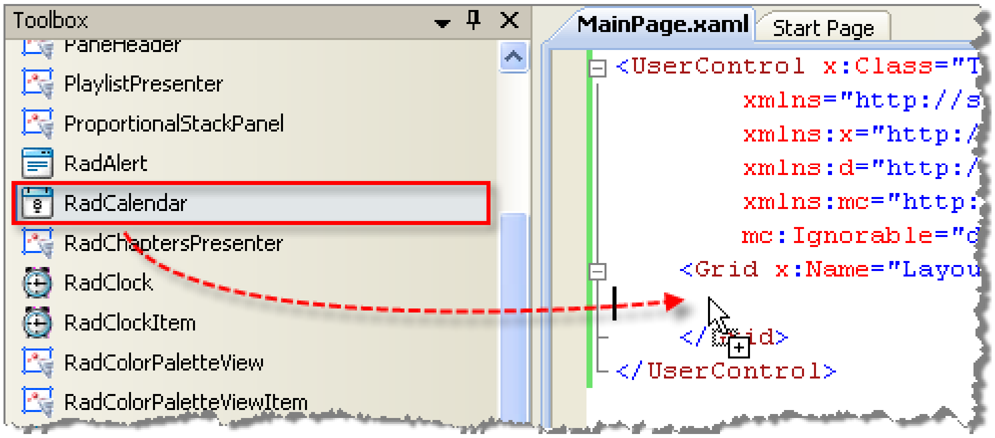
```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <Style x:Key="ButtonStyle" TargetType="Button">
    <Setter Property="FontFamily" Value="Verdana" />
    <Setter Property="FontSize" Value="12" />
    <Setter Property="Foreground" Value="Green" />
  </Style>
  <Style x:Key="BigButtonStyle" TargetType="Button"
      BasedOn="{StaticResource ButtonStyle}">
    <Setter Property="FontSize" Value="16" />
  </Style>

</ResourceDictionary>
```

We can use the MergedDictionaries collection to pull in the separate XAML files to any Resources element. The example below merges two dictionaries to the Application Resources element where they will be available application wide.

```
<Application.Resources>
   <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
         <ResourceDictionary Source="MyButtonStyles.xaml" />
         <ResourceDictionary Source="MyTextStyles.xaml" />
      </ResourceDictionary.MergedDictionaries>
   </ResourceDictionary>
</Application.Resources>
```

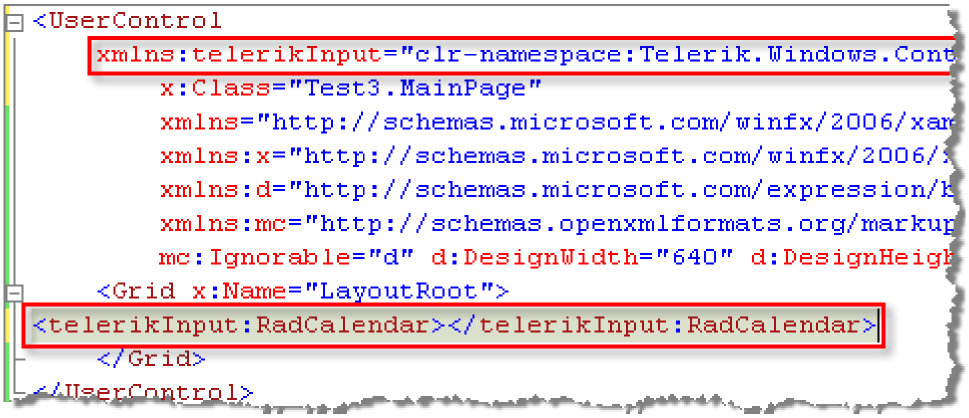
3.3.5 Adding and Theming RadControls

Once installed to the Toolbox, RadControls can be dragged into the XAML just as you would drag a standard Silverlight control. The screenshot below shows a RadCalendar being dragged inside the main "LayoutRoot" Grid element.



```
<UserControl x:Class="T
        xmlns="http://s
        xmlns:x="http:/
        xmlns:d="http:/
        xmlns:mc="http:
        mc:Ignorable="d
    <Grid x:Name="Layou

    </Grid>
</UserControl>
```

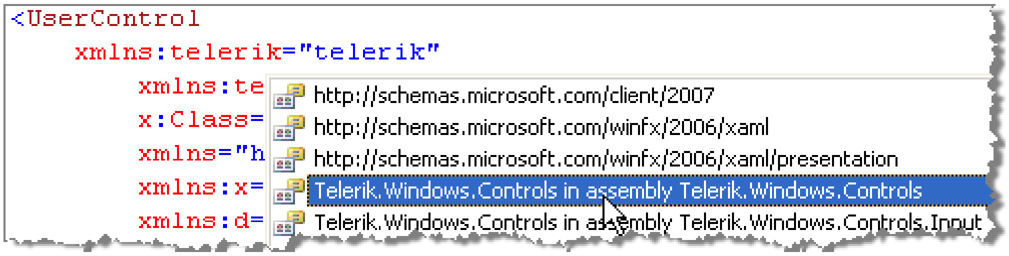
When the drag is complete, an XML namespace is automatically added to the UserControl, and the control is added, along with its corresponding namespace, to the grid.



```
<UserControl
    xmlns:telerikInput="clr-namespace:Telerik.Windows.Cont
        x:Class="Test3.MainPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xam
        xmlns:x="http://schemas.microsoft.com/winfx/2006/
        xmlns:d="http://schemas.microsoft.com/expression/b
        xmlns:mc="http://schemas.openxmlformats.org/markup
        mc:Ignorable="d" d:DesignWidth="640" d:DesignHeigh
    <Grid x:Name="LayoutRoot">
<telerikInput:RadCalendar></telerikInput:RadCalendar>
    </Grid>
</UserControl>
```

To theme a RadControl, first right-click the References node in the Solution Explorer and select **Add Reference...** Browse to the Telerik assemblies located in the Telerik RadControls for Silverlight installation and select **Telerik.Windows.Controls**. Also add a reference to one of the theme assemblies. These assemblies all start with "Telerik.Windows.Themes" and the name of the theme. The example below adds the **Telerik.Windows.Themes.Summer** assembly.

Once assembly DLL's are referenced in the project you can add XML namespace references in the markup. The screenshot below shows the reference being added with the name of "telerik".



```
<UserControl
    xmlns:telerik="telerik"
        xmlns:te
        x:Class=
        xmlns="h
        xmlns:x=
        xmlns:d=
```

http://schemas.microsoft.com/client/2007
http://schemas.microsoft.com/winfx/2006/xaml
http://schemas.microsoft.com/winfx/2006/xaml/presentation
Telerik.Windows.Controls in assembly Telerik.Windows.Controls
Telerik.Windows.Controls in assembly Telerik.Windows.Controls.Input

That will provide access to the StyleManager class. From there you can set the theme using the StyleManager Theme attached property. The example below sets the Summer theme to a RadCalendar control. The exact same steps are used to theme all RadControls.

```
<telerikInput:RadCalendar telerik:StyleManager.Theme="Summer">
</telerikInput:RadCalendar>
```

3.3.6 Templates

Templates are XAML patterns used visualize Silverlight elements. You can think of them as recipes for rendering element trees. Templates descend from the abstract FrameworkTemplate where there are a few commonly used descendants:

- **ControlTemplate** creates a visual representation of a UI control, such as a button. With ControlTemplate you can completely replace the visual tree of a control with any set of elements you care to use without altering the control's behavior. The ControlTemplate "Template" property is available to every descendant of "Control". This makes Template very general purpose and is useful when you need to "rip up the planks" to create a new visual representation.
- **DataTemplate** describes visual structure of a data object, such as a business object.
- ItemsControl uses **ItemTemplate** to visualize child elements. ListBox would be an example of an ItemsControl descendant that can be customized using ItemTemplate.
- **HierarchicalDataTemplate** is used to display child objects of a data object, for example master/detail grids or tree views.

Templates can also be surfaced for particular areas of a control that can be customized. This strategy is used by RadControls to allow customization of controls that can be displayed in different orientations or that have different states. For example, the RadPane element used in docking has a BottomTemplate, LeftTemplate, RightTemplate and TopTemplate where the position of the pane determines which template is being rendered. RadPane also has a TitleTemplate. All these templates allow you to customize the parts of the control that are of interest to you without altering the underlying functionality of the control.

ControlTemplate

Let's take our simple Button example and add a ControlTemplate definition. The example below shows that the Button has a Template property. The Template property is of type ControlTemplate. Inside the ControlTemplate we place a simple TextBlock.

```
<Button Content="Button 1" Click="Button_Click">
  <Button.Template>
    <ControlTemplate>
      <TextBlock Text="Print" />
    </ControlTemplate>
  </Button.Template>
</Button>
```

When you run this, the text "Print" shows, not the content "Button 1" and the Click event still fires. The usage you see in the example above is not usual for multiple reasons. First, templates are typically defined in resources:

```
<Grid x:Name="LayoutRoot">
  <Grid.Resources>
    <ControlTemplate x:Key="MyButtonTemplate">
      <TextBlock Text="Print" />
    </ControlTemplate>
  </Grid.Resources>

  <Button Content="Button 1" Click="Button_Click"
      Template="{StaticResource MyButtonTemplate}" />
</Grid>
```

The TextBlock in the template has hard coded text. What if we want the Content for the button to be used instead. Here the markup syntax "{TemplateBinding}" can be used to pass a property value through to the template. In this example, Content is used to populate the TextBlock Text property so that "Button 1" displays.

```
<Grid x:Name="LayoutRoot">
  <Grid.Resources>
    <ControlTemplate x:Key="MyButtonTemplate" TargetType="Button">
      <TextBlock Text="{TemplateBinding Content}" />
    </ControlTemplate>
  </Grid.Resources>

  <Button Content="Button 1" Click="Button_Click"
      Template="{StaticResource MyButtonTemplate}" />
</Grid>
```

3.4 Dependency Properties

Standard CLR (Common Language Runtime) properties are not responsive or extensible enough to support declarative interfaces with features like animation, templates, resources and data binding. Declarative interfaces with these requirements need to establish property value based on input from multiple sources and should be able to publish notification of property changes to multiple interested parties.

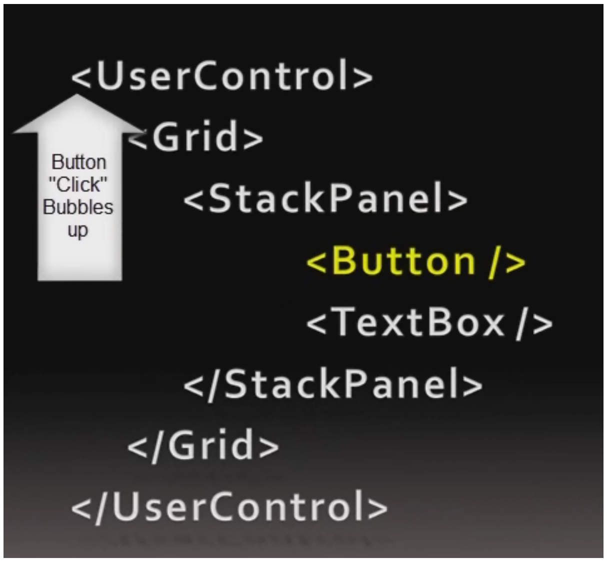
Dependency properties are an important cornerstone of Silverlight that make declarative user interfaces possible. Dependency properties allow Silverlight to determine value properties dynamically from a number of different inputs, including Resources, data binding and animations.

Dependency properties are set in the following order of precedence:

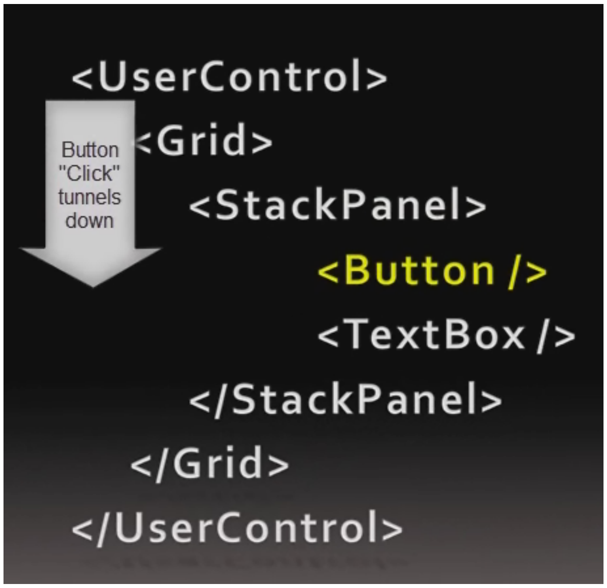
- Animation
- local values
- templated values
- style values
- default value

3.5 Routed Events

One particularly useful application of dependency properties are "routed events". Routed events are events that can travel a route along an element tree. Routed events can Bubble from from the source element that triggered the event to the root element or tunnel from the root element down to the source element. Each handler can set "Handled" argument to discontinue processing.



```
<UserControl>
  <Grid>
    <StackPanel>
      <Button />
      <TextBox />
    </StackPanel>
  </Grid>
</UserControl>
```



```
<UserControl>
  <Grid>
    <StackPanel>
      <Button />
      <TextBox />
    </StackPanel>
  </Grid>
</UserControl>
```

Routed events allow you to hook up disparate elements that perhaps aren't designed to be used together. Say you have a combo box in a stack panel. The stack panel can be notified that the combo box selection has changed. Events are no longer constrained to the element where they are defined.

Silverlight has been a little weak on some features that are standard with WPF.

- Routed events haven't been widely defined.
- Silverlight doesn't support custom routed events as of this writing.

Telerik RadControls for Silverlight provides WPF-like support for custom routed events and also allows expanded support for varying "routing strategies" (Bubbling, Tunneling and Direct). Telerik RadControls routinely implement their events as routed events so you can use them right "out of the box".

For more information on Silverlight routed events, download the Telerik Trainer application and the "RadControls for Silverlight - Overview of Routed Events" interactive video at:

http://www.telerik.com/support/trainer/silverlight.aspx

3.6 Basic Databinding

FrameworkElement is the abstract base class for elements participating in Silverlight layout. FrameworkElement introduces the key **DataContext** property. You can assign various data sources or your own custom business objects to the DataContext.

The example below shows a simple "Report" class that contains "Title" and "Description" properties. A single instance of the Report class is created and assigned to the DataContext of a Button.

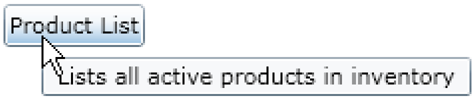
```
Partial Public Class MainPage
  Inherits UserControl
  Public Sub New()
    InitializeComponent()
    btnPrint.DataContext = New Report() With { _
.Title = "Product List", .Description = "Lists all active products in inventory"}
  End Sub

End Class

Public Class Report
  Private privateTitle As String
  Public Property Title() As String
    Get
      Return privateTitle
    End Get
    Set(ByVal value As String)
      privateTitle = value
    End Set
  End Property
  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
End Class
```

```
public partial class MainPage : UserControl
{
   public MainPage()
   {
      InitializeComponent();
      btnPrint.DataContext = new Report()
      {
         Title = "Product List",
         Description = "Lists all active products in inventory"
      };
   }
}

public class Report
{
   public string Title { get; set; }
   public string Description { get; set; }
}
```

To bind the particular properties of an element, use the markup extension syntax where curly braces surround "Binding" and specify a "Path" to a particular property of your bound object. In the example below, Button Content is bound to the "Title" property of the Report object and the ToolTip is bound to the "Description".

```
<Button x:Name="btnPrint"
   Content="{Binding Path=Title}"
   ToolTipService.ToolTip="{Binding Path=Description}"
   HorizontalAlignment="Left" VerticalAlignment="Top"
   Margin="20"></Button>
```

The running Silverlight application shows the Button with bound properties:



Product List

Lists all active products in inventory

Notes

You can omit the "Path", as shown in the XAML below:

```
<Button Content="{Binding Title}" ToolTipService.ToolTip="{Binding Description}" . . ./>
```

To instantiate objects for binding in XAML, first create an XML namespace for the project that contains the object you want to bind to (see Working with XAML, XAML Basics for directions on how to do this). Then, in a Resources element add a reference to your object. The XML namespace for the example below is "thisProject" and the object is "Report". The two properties for the Report object, "Title" and "Description", are represented by attributes of the same name. You must set the "x:Key" to some unique name so you can refer to it during binding.

To bind the Report object instantiated in XAML, bind the DataContext to the resource key name.

```
<Grid.Resources>
 <thisProject:Report x:Key="Report"
   Title="Product Listing"
   Description="Lists all active products in stock" />
</Grid.Resources>

<Button x:Name="btnPrint"
   DataContext="{StaticResource Report}"
   Content="{Binding Title}"
   ToolTipService.ToolTip="{Binding Description}"
   HorizontalAlignment="Left" VerticalAlignment="Top"
   Margin="20" />
```

3.7 Best Practices

MVVM

When integration between the user interface and the data is too tight, testing, refactoring and maintenance become more difficult. Changes in the data schema or the querying logic can require changes throughout the application. Separating the application into clear separation of concerns makes the application easier to maintain and test as changes are introduced.

There are well defined architectural patterns that provide a map of where functional parts of the application should be placed. The MVC pattern, Model - View - Controller, can work well for applications that don't have declarative, i.e. XAML, interfaces. In this pattern, Model is the data, View is the user interface and Controller is the programmatic interface between model and view. But XAML can declare data binding, blurring the line between data and the view.

The MVVM pattern, Model - View - ViewModel, is a pattern more suited to declarative interfaces. In this pattern, the ViewModel sits between the Model and View, providing an abstraction that can be used both programmatically and declaratively by the view. The View can be programmatically *or* declaratively bound to the ViewModel without the View having direct knowledge of the Model.

Prism

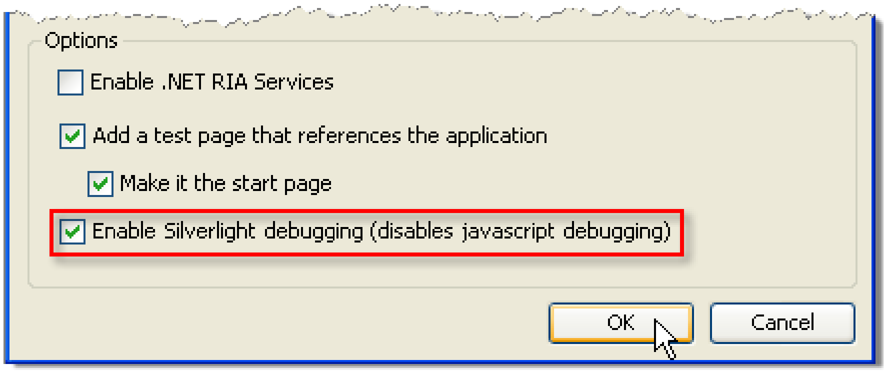
When you start building Silverlight applications to test various Silverlight and RadControls, your applications will be simple and small. As your applications become larger they need to allow separate testing and maintenance. But separation of concerns is only one of the requirements of a full scale production application including logging, load-on-demand and event publishing between services, to name a few.

Prism is a collection of resources for implementing best practices in Silverlight. You can pick and choose the pieces you need from the Prism libraries, source code, examples, quick starts and documentation.

For more information see "Composite Apps from Prism" at http://msdn.microsoft.com/en-us/magazine/dd943055.aspx. You can find the download for Prism at http://www.microsoft.com/downloads/details.aspx?FamilyID=fa07e1ce-ca3f-4b9b-a21b-e3fa10d013dd&DisplayLang=en.

3.8 Debugging

You can debug a Silverlight application in much the same way as you would for a WinForms application. Just set your breakpoints and press F5 to debug the application. Be aware that you cannot debug Javascript and Silverlight at the same time. To enable debugging when you first create the Silverlight application, leave the "Enable Silverlight debugging" option checked.



Options

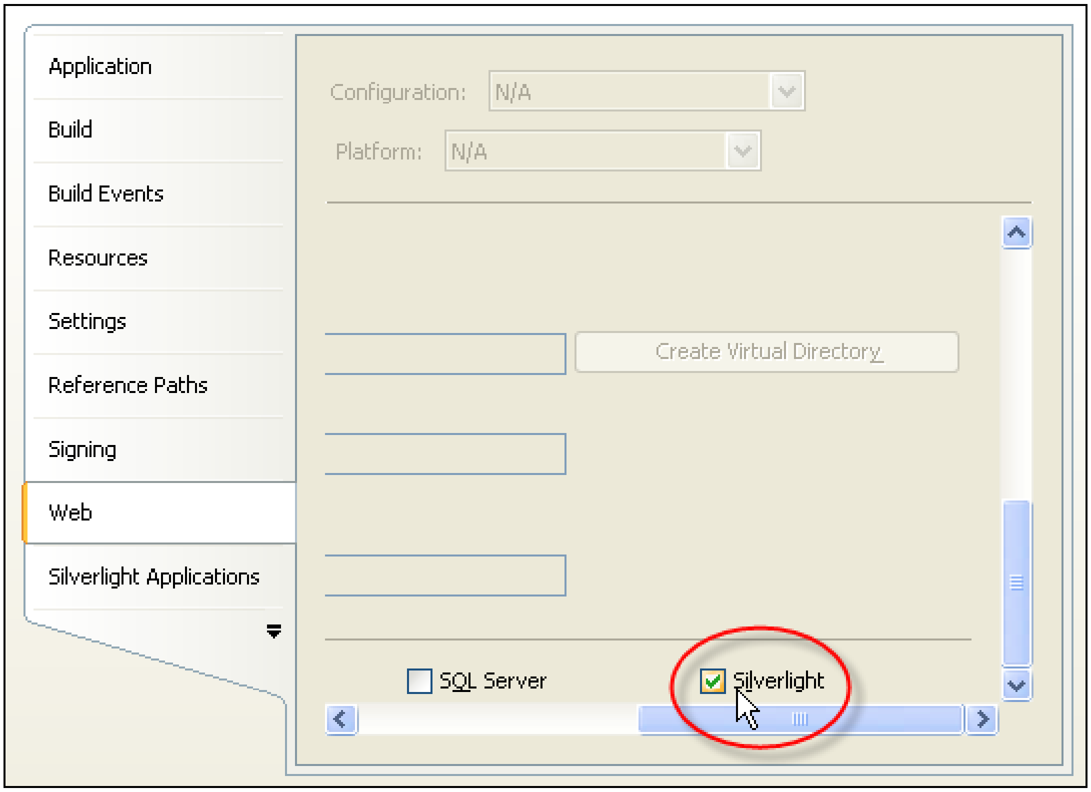
□ Enable .NET RIA Services

☑ Add a test page that references the application

☑ Make it the start page

☑ Enable Silverlight debugging (disables javascript debugging)

Later, you can go back to the Solution Explorer, right-click the host project and select Properties from the context menu. Select the Web tab, then locate the Silverlight checkbox.



Configuration: N/A

Platform: N/A

□ SQL Server

☑ Silverlight

Gotcha!

Question: When trying to debug I receive the error message "Unable to start debugging. The silverlight managed debugging package isn't installed.".

Answer: You may have installed the end-user runtime, not the developer runtime. See the "What Do You Need to Have Before Reading This Courseware" chapter and make sure you have everything installed correctly.

3.9 Wrap Up

This chapter provided a quick guide to basic "up-and-running" information. In this chapter you built simple "Hello World" Silverlight applications using both Visual Studio and Expression Blend. You also learned how to hand off projects back and forth between the two environments and in the process learned the basic working styles of both. In this chapter you learned the basics about XAML, including how XML namespaces are defined, defining objects, properties and collections in XAML, attached properties, markup extensions, how to define and apply styles to Silverlight elements, how to define and use resources and how RadControls are defined and themed in XAML.

You learned how templates are used to create free form arrangements of Silverlight elements without affecting underlying functionality. You learned the basics on how to bind data to Silverlight elements.

You learned about Silverlight best practices including MVVM and Prism. Finally, you learned the settings that control debugging of a Silverlight application.

Part IV

Data Binding

4 Data Binding

4.1 Objectives

In this chapter you will learn the basics of binding objects and collections of objects to Silverlight elements. You will learn syntax variations used to bind declaratively and how to bind elements using code alone. You will also learn how to detect changes in property data and changes to collections.

You will learn how to bind elements in two common templates: DataTemplate and HierarchicalDataTemplate.

You will learn how to bind getting data from a variety of popular data sources including straight XML and variations of XML, i.e. ATOM and RSS. In the process you will use a combination of XDocument and LINQ to read, parse and convert the data to a form suitable for binding. Building off the XML techniques we will discuss REST services, build WCF RIA services, WCF and ADO.NET Data Services.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Binding\Binding.sln.

\Courseware\Projects\<CS|VB>\Binding_Ria\Binding_Ria.sln.

4.2 Binding Basics

Data binding is the process of connecting Silverlight elements to standard CLR (Common Language Runtime) objects. **FrameworkElement** is a base class for Silverlight elements that participate in layout, but more importantly for our purposes, can be bound to data. FrameworkElement introduces the **DataContext** property. DataContext can be assigned objects in XAML or code-behind as shown in the example below. DataContext is available to the bound element and any children of the element. The example below binds the DataContext property of a TextBox to a custom object called "Category".

VB

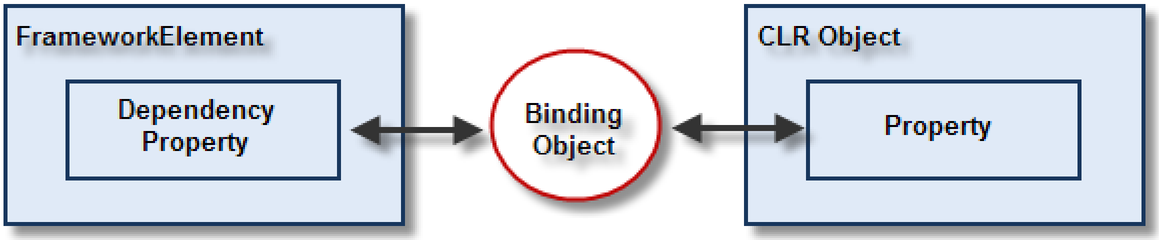
```
tbTitle.DataContext = category
```

C#

```
tbTitle.DataContext = category;
```

Binding in Silverlight requires three things:

- A dependency property in a FrameworkElement descendant that you are binding to.
- A standard property in a .NET object.
- A Binding object to handle communication between the two. The Binding object defines the DataContext, the identity of the objects being bound and the names of the properties involved.



Lets look at an example where we have a standard CLR object that we will call "Category". Category has two properties, "Title" and "Description".

```
Public Class Category
  Private privateTitle As String
  Public Property Title() As String
    Get
      Return privateTitle
    End Get
    Set(ByVal value As String)
      privateTitle = value
    End Set
  End Property
  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
```

```
public class Category
{
   public string Title { get; set; }
   public string Description { get; set; }
}
```

Because we're working with a dependency property, we can perform the binding right in the XAML. Take a look at the example below. The UserControl.Resources defines a single instance of Category. In this example, the FrameworkElement is a TextBox and the dependency property is the Text property. The Binding object is declared using "{Binding ...} syntax. The critical Binding properties are:

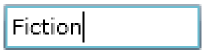
Path: The name of the source property to get the data from. The example below omits the "Path" attribute in favor of the shorter syntax where the property following "Binding" is the Path, i.e. "Title".

Source: The standard CLR object that contains the data. This object is assigned to the FrameworkElement DataContext.

Mode: The data's direction of travel. This can be **OneTime** where the data is set one time and is not updated after that, **OneWay** where the data flows one way from the CLR object to the Silverlight element and **TwoWay** where the source object updates the target element and the target element can also update the source object.

```
<UserControl.Resources>
  <local:Category x:Key="Category" Title="Fiction"
        Description="A wide range of fiction from the worlds best authors." />
</UserControl.Resources>

<Grid x:Name="LayoutRoot">
  <TextBox x:Name="tbTitle"
         Text="{Binding Title, Source={StaticResource Category}, Mode=OneTime}"
         HorizontalAlignment="Left" VerticalAlignment="Top" MinWidth="100"
         ></TextBox>
</Grid>
```

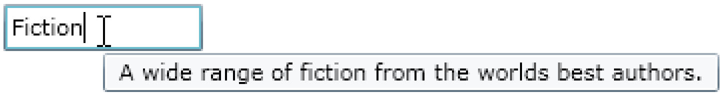
The bound TextBox in the running Silverlight application shows the title of the Category object.


Fiction

What if you want to bind more than one Category property? The example below shows how you can assign the DataContext of the element one time and then set the Binding for each property that needs data. This example binds the Text property to the Category "Title" and also adds an attached ToolTip property and binds it to the Category "Description".

```
<UserControl.Resources>
  <local:Category x:Key="Category" Title="Fiction"
        Description="A wide range of fiction from the worlds best authors." />
</UserControl.Resources>

<Grid x:Name="LayoutRoot">
  <TextBox x:Name="tbTitle"
        DataContext="{StaticResource Category}"
        Text="{Binding Title, Mode=OneTime}"
        ToolTipService.ToolTip="{Binding Description, Mode=OneTime}"
        HorizontalAlignment="Left" VerticalAlignment="Top"
        MinWidth="100" Margin="20"></TextBox>
</Grid>
```

The running application now shows that both the Text and ToolTip properties are bound.

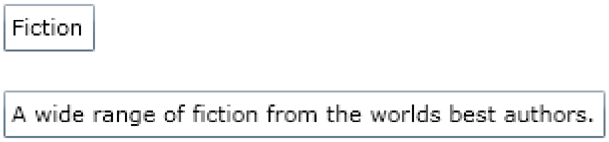

Fiction

A wide range of fiction from the worlds best authors.

DataContext is available to all children of an element. If we assign the Category object to a StackPanel containing two TextBox elements, we can set the Binding for each TextBox using the Category as the source.

```
<StackPanel DataContext="{StaticResource Category}">
  <TextBox x:Name="tbTitle"
      Text="{Binding Title, Mode=OneTime}"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      Margin="10" />
  <TextBox x:Name="tbDescription"
      Text="{Binding Description, Mode=OneTime}"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      Margin="10" />
</StackPanel>
```

The running application below shows that both TextBox elements contain data from the Category that was assigned to the StackPanel's DataContext.



Fiction

A wide range of fiction from the worlds best authors.

Binding can also be performed programmatically. The example below creates a new instance of a Category, creates a new Binding object and finally, calls the SetBinding() method. SetBinding() is a FrameworkElement method that takes a DependencyProperty and a Binding object. The Binding object is assigned the Source, Path and Mode, just as in the XAML examples.

```
Dim category As New Category() With { _
.Title = "Fiction", _
.Description = "A wide range of fiction from the worlds best authors."}
Dim binding As New Binding() With {_
.Source = category, _
.Path = New PropertyPath("Title"), _
.Mode = BindingMode.OneTime}

tbTitle.SetBinding(TextBox.TextProperty, binding)
```

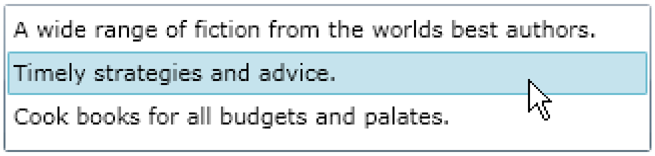
```
Category category = new Category()
{
   Title = "Fiction",
   Description = "A wide range of fiction from the worlds best authors."
};

Binding binding = new Binding()
{
   Source = category,
   Path = new PropertyPath("Title"),
   Mode = BindingMode.OneTime
};

tbTitle.SetBinding(TextBox.TextProperty, binding);
```

4.3 Binding Collections

Collection are represented by the **ItemsControl** class and its descendants. Binding a collection to an ItemsControl is as simple as assigning the **ItemsSource** property to some collection and assigning the **DisplayMemberPath** to a property of a collection item. The DataContext of each list box item maps to a member of the bound collection and the DisplayMemberPath to the Binding Path. The example below creates a list of Categories in XAML, assigns it to a ListBox ItemsSource and sets the "Description" property of the Category object to be the display member.

```
<Grid x:Name="LayoutRoot">
   <Grid.Resources>
      <local:Categories x:Key="Categories">
         <local:Category Title="Fiction"
               Description="A wide range of fiction from the worlds best authors." />
         <local:Category Title="Business and Investing" Description="Timely strategies and advice." />
         <local:Category Title="Cooking, Food and Wine"
               Description="Cook books for all budgets and palates." />
      </local:Categories>
   </Grid.Resources>
   <ListBox ItemsSource="{StaticResource Categories}" DisplayMemberPath="Description" />
</Grid>
```

The running Silverlight project displays the ListBox populated with category descriptions.



A wide range of fiction from the worlds best authors.
Timely strategies and advice.
Cook books for all budgets and palates.

4.4 Change Notification

How are changes in objects and collections automatically propagated to elements? The "Binding Basics" section used simple objects with straightforward property accessors and set the Binding Mode to "OneTime". Implementing the **INotifyPropertyChanged** interface will allow changes to object properties to show up instantly in Silverlight elements. A second interface, **INotifyCollectionChanged** notifies Silverlight elements about changes to collections. Fortunately, you don't have to implement INotifyCollectionChanged. Instead, descend from **ObservableCollection<T>** and the interface is implemented for you.

INotifyPropertyChanged

To allow changes in business object properties to be reflected automatically by Silverlight elements, add a reference to **System.ComponentModel** and implement the **INotifyPropertyChanged** interface. INotifyPropertyChanged has a single method **PropertyChanged().** When setting a property value, you need to call PropertyChanged if its been assigned and pass the name of the property that's being changed.

```
Public Class Category
  Implements INotifyPropertyChanged
  Private title_Renamed As String
  Public Property Title() As String
    Get
      Return title_Renamed
    End Get

    Set(ByVal value As String)
      If title_Renamed <> value Then
        title_Renamed = value
        OnPropertyChanged("Title")
      End If
    End Set
  End Property

  Private description_Renamed As String
  Public Property Description() As String
    Get
      Return description_Renamed
    End Get

    Set(ByVal value As String)
      If description_Renamed <> value Then
        description_Renamed = value
        OnPropertyChanged("Description")
      End If
    End Set
  End Property

  Public Event PropertyChanged As PropertyChangedEventHandler

  Public Sub OnPropertyChanged(ByVal propertyName As String)
    RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
  End Sub
End Class
```

C#

```
public class Category: INotifyPropertyChanged
{
   private string title;
   public string Title
   {
      get { return title;}

      set
      {
         if (title != value)
         {
            title = value;
            OnPropertyChanged("Title");
         }
      }
   }

   private string description;
   public string Description
   {
      get {return description;}

      set
      {
         if (description != value)
         {
            description = value;
            OnPropertyChanged("Description");
         }
      }
   }

   public event PropertyChangedEventHandler PropertyChanged;

   public void OnPropertyChanged(string propertyName)
   {
      if (PropertyChanged != null)
      {
         PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
      }
   }
}
```

ObservableCollection

To provide notifications of collection changes, inherit from **ObservableCollection<T>**. Add a reference to **System.Collections.ObjectModel** before using ObservableCollection. The example below adds several sample Category objects to the collection.

VB

```
Public Class Categories
  Inherits ObservableCollection(Of Category)
  Public Sub New()
    Me.Add(New Category() With { _
.Title = "Fiction", _
.Description = "A wide range of fiction from the worlds best authors."})
    Me.Add(New Category() With {_
.Title = "Business and Investing", _
.Description = "Timely strategies and advice."})
    Me.Add(New Category() With {_
.Title = "Cooking, Food and Wine", _
.Description = "Cook books for all budgets and palates."})
  End Sub
End Class
```

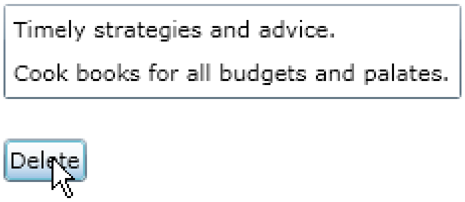
C#

```
public class Categories : ObservableCollection<Category>
{
  public Categories()
  {
    this.Add(new Category()
    {
      Title = "Fiction",
      Description = "A wide range of fiction from the worlds best authors."
    });
    this.Add(new Category()
    {
      Title = "Business and Investing",
      Description = "Timely strategies and advice."
    });
    this.Add(new Category()
    {
      Title = "Cooking, Food and Wine",
      Description = "Cook books for all budgets and palates."
    });
  }
}
```

The example below assigns a Categories collection to the ItemsSource of a ListBox and sets the DisplayMemberPath to the Category "Description" property. The example also has a Delete button that removes the top element in the collection so we can see ObservableCollection notification in action.

```
<UserControl.Resources>
   <local:Categories x:Key="Categories" />
</UserControl.Resources>

<Grid x:Name="LayoutRoot">
   <ListBox x:Name="lbCategories" ItemsSource="{StaticResource Categories}"
         DisplayMemberPath="Description" HorizontalAlignment="Left" VerticalAlignment="Top"
         Margin="10" />
  <Button x:Name="btnDelete" Content="Delete" Click="btnDelete_Click"></Button>
</Grid>
```

When elements are removed from the collection, the ListBox reflects the change.



Timely strategies and advice.

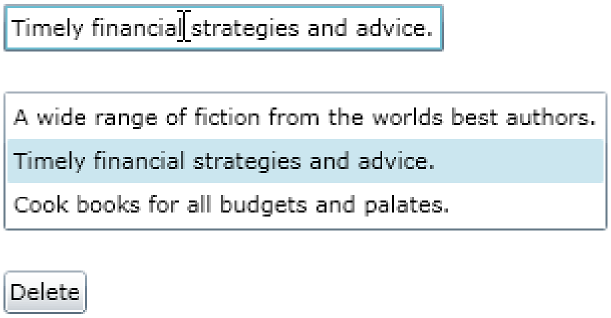
Cook books for all budgets and palates.

Updating Data

This next example shows several of the previous techniques together. When the user changes the description in the text box, then tabs off the text box, the Category business object is updated. Because Category implements INotifyPropertyChanged, the currently selected list box item displays the matching. text. Notice the new syntax in the Text Binding expression for the TextBox. It uses a new property **ElementName** that points to the list box. The Binding Path points to the SelectedItem property of the list box. The SelectedItem is actual a Category, so we can drill down to the "Description" property. Also notice that the Binding Mode is TwoWay, allowing updates made in the Silverlight element to propagate back to the business object.

```
<StackPanel DataContext="{StaticResource Categories}">

   <TextBox x:Name="tbDescription"
         Text="{Binding Path=SelectedItem.Description, ElementName=lbCategories, Mode=TwoWay}"
         HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10" />

   <ListBox x:Name="lbCategories" ItemsSource="{Binding}"
         DisplayMemberPath="Description" HorizontalAlignment="Left" VerticalAlignment="Top"
         Margin="10" />

   <Button x:Name="btnDelete" Content="Delete" Click="btnDelete_Click" HorizontalAlignment="Left"
         VerticalAlignment="Top" Margin="10"></Button>
</StackPanel>
```

When the user selects items in the list box, the TextBox displays the corresponding Category Description. When the TextBox Text is changed by the user and they tab off, the text is updated in the list box.



Timely financial strategies and advice.

A wide range of fiction from the worlds best authors.

Timely financial strategies and advice.

Cook books for all budgets and palates.

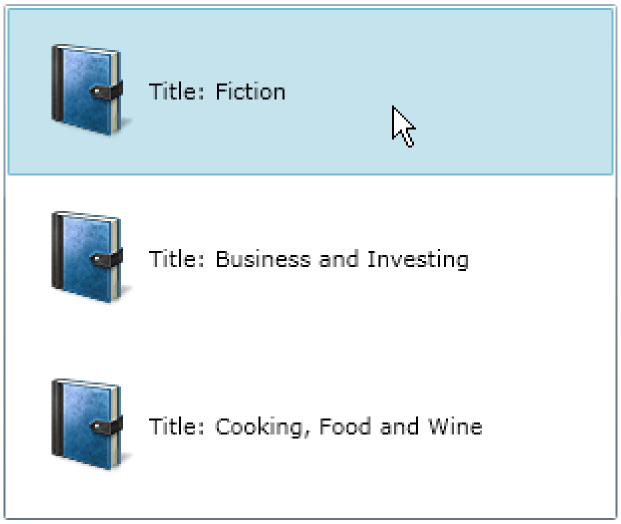
4.5 Binding in Templates

4.5.1 DataTemplate

Using templates you can bind your data but arrange the visual layout in any configuration that suits your purpose. You're not limited to just using the DisplayMemberPath because you can aggregate any number of elements in the template and bind each of them. The ListBox ItemTemplate is a DataTemplate type. Inside the ListBox element tag use the syntax "ListBox.ItemTemplate", i.e. "objectName.propertyName". Inside that element place the "<DataTemplate>". The DataTemplate can contain only a single item, but you can make that a StackPanel, Grid or other container item. From there you can populate the container with any elements that work for your requirements.

```
<UserControl.Resources>
  <local:Categories x:Key="Categories" />
</UserControl.Resources>

<Grid x:Name="LayoutRoot">
  <ListBox ItemsSource="{StaticResource Categories}">
    <ListBox.ItemTemplate>
      <DataTemplate>
        <StackPanel
            ToolTipService.ToolTip="{Binding Description}" Margin="10" Orientation="Horizontal">
          <Image Source="journal.png" Width="48" Height="48" Margin="5" />
          <TextBlock Text="Title: " VerticalAlignment="Center" />
          <TextBlock Text="{Binding Title}" VerticalAlignment="Center" />
        </StackPanel>
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
</Grid>
```

In the running Silverlight application we can see that using templates lets you achieve a free form layout with data bound elements.

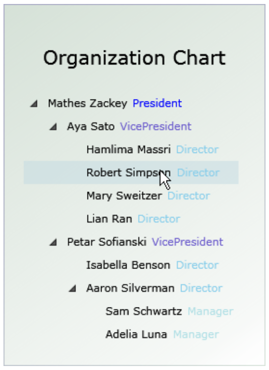


Title: Fiction

Title: Business and Investing

Title: Cooking, Food and Wine

4.5.2 HierarchicalDataTemplate



Organization Chart

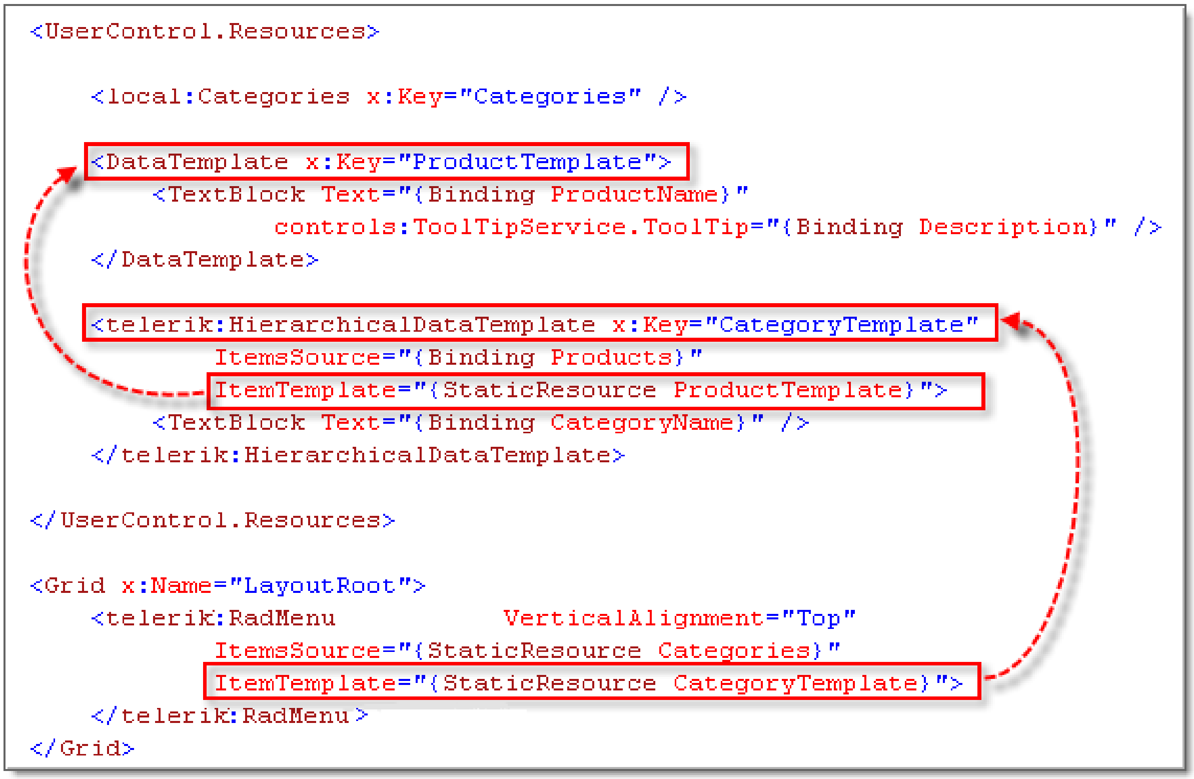
Mathes Zackey President
Aya Sato VicePresident
Hamlima Massri Director
Robert Simpson Director
Mary Sweitzer Director
Lian Ran Director
Petar Sofianski VicePresident
Isabella Benson Director
Aaron Silverman Director
Sam Schwartz Manager
Adelia Luna Manager

HierarchicalDataTemplates allow multiple layers of hierarchical data to be displayed. RadTreeView, RadPanelBar and RadMenu for example, all use HierarchicalDataTemplate. The control's ItemTemplate points to a HierarchicalDataTemplate. Each HierarchicalDataTemplate ItemTemplate attribute points up to another HierarchicalDataTemplate except the last template that has no children. This last template is a DataTemplate. Take a look at the TreeView example below that represents a corporate hierarchy with President/ Vice President, Director and Manager.

Each level of the hierarchy has its own template. The RadTreeView ItemTemplate points at the "PresidentTemplate", the "PresidentTemplate" ItemTemplate points to the "VPTemplate" and so on until the last "ManagerTemplate".

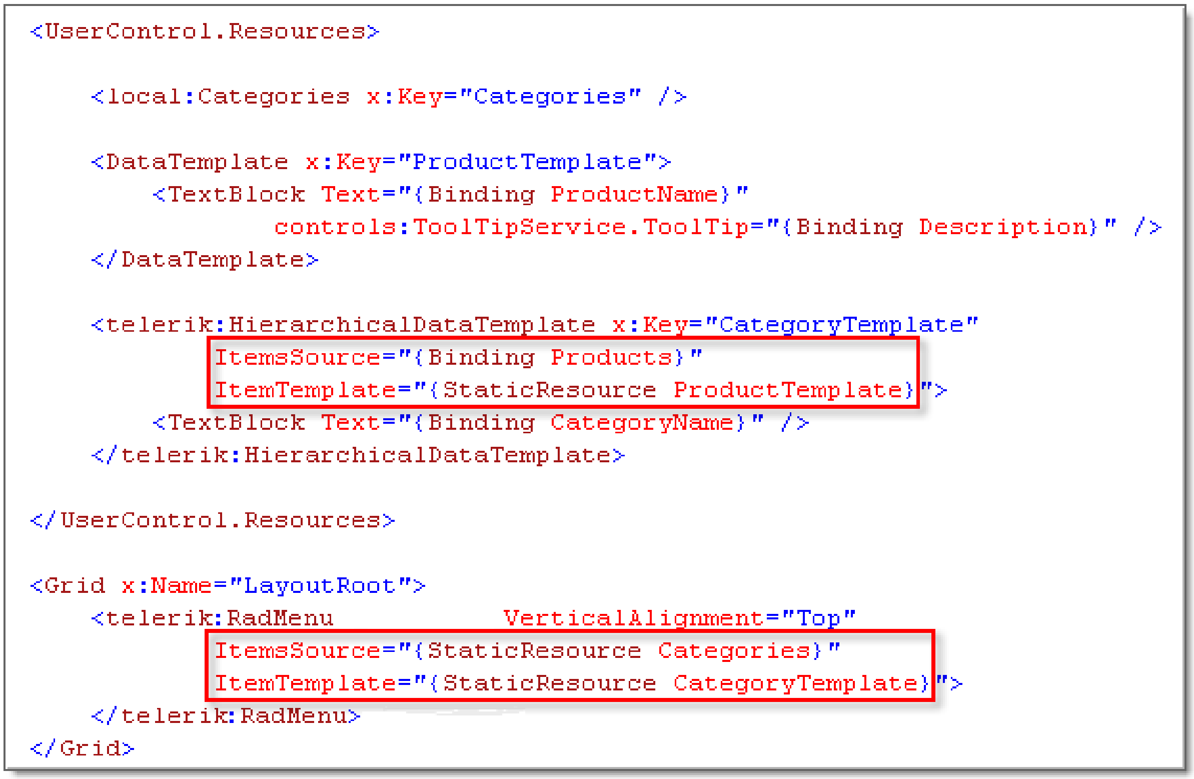
```
<UserControl.Resources>. . .
  <DataTemplate x:Key="ManagerTemplate">. . . content
  </DataTemplate>
  <telerik:HierarchicalDataTemplate x:Key="DirectorTemplate"
      ItemTemplate="{StaticResource ManagerTemplate}">. . . content
  </telerik:HierarchicalDataTemplate>
  <telerik:HierarchicalDataTemplate x:Key="VPTemplate"
      ItemTemplate="{StaticResource DirectorTemplate}">. . . content
  </telerik:HierarchicalDataTemplate>
  <telerik:HierarchicalDataTemplate x:Key="PresidentTemplate"
      ItemTemplate="{StaticResource VPTemplate}">. . . content
  </telerik:HierarchicalDataTemplate>
</UserControl.Resources>. . .
<telerik:RadTreeView x:Name="tvMain" ItemTemplate="{StaticResource PresidentTemplate}">
</telerik:RadTreeView>
```

Here's another example that shows a two level hierarchy with Categories/Products. Look at the relationship of the templates in the XAML below. The RadMenu ItemTemplate points to a HierarchicalDataTemplate "CategoryTemplate". "CategoryTemplate" also has an ItemTemplate that points to a DataTemplate called "ProductTemplate". There can be additional levels of templates. The rule is to use a HierarchicalDataTemplate where there are more child items, then use a standard DataTemplate when no more child levels remain.



```
<UserControl.Resources>

    <local:Categories x:Key="Categories" />

    <DataTemplate x:Key="ProductTemplate">
        <TextBlock Text="{Binding ProductName}"
                controls:ToolTipService.ToolTip="{Binding Description}" />
    </DataTemplate>

    <telerik:HierarchicalDataTemplate x:Key="CategoryTemplate"
            ItemsSource="{Binding Products}"
            ItemTemplate="{StaticResource ProductTemplate}">
        <TextBlock Text="{Binding CategoryName}" />
    </telerik:HierarchicalDataTemplate>

</UserControl.Resources>

<Grid x:Name="LayoutRoot">
    <telerik:RadMenu           VerticalAlignment="Top"
            ItemsSource="{StaticResource Categories}"
            ItemTemplate="{StaticResource CategoryTemplate}">
    </telerik:RadMenu>
</Grid>
```

Let's take another look at that same XAML to see how the data is being hooked up. The ItemsSource attribute works in concert with the ItemTemplate. In the RadMenu XAML you can see that the "CategoryTemplate" is supplied by data from the "Categories" object. In the "CategoryTemplate", the "ProductTemplate" is supplied by data from the "Products" object.



```
<UserControl.Resources>

    <local:Categories x:Key="Categories" />

    <DataTemplate x:Key="ProductTemplate">
        <TextBlock Text="{Binding ProductName}"
                controls:ToolTipService.ToolTip="{Binding Description}" />
    </DataTemplate>

    <telerik:HierarchicalDataTemplate x:Key="CategoryTemplate"
            ItemsSource="{Binding Products}"
            ItemTemplate="{StaticResource ProductTemplate}">
        <TextBlock Text="{Binding CategoryName}" />
    </telerik:HierarchicalDataTemplate>

</UserControl.Resources>

<Grid x:Name="LayoutRoot">
    <telerik:RadMenu           VerticalAlignment="Top"
            ItemsSource="{StaticResource Categories}"
            ItemTemplate="{StaticResource CategoryTemplate}">
    </telerik:RadMenu>
</Grid>
```

See the "Menu Controls" and "TreeView" chapters "Binding" section for full walk throughs of these examples.

4.6 Data Sources

4.6.1 XML

If you need to bind to an XML file, consider using a combination of the **XDocument** class to read and parse the XML, and LINQ to transform the parsed XML into a collection suitable for binding.

XDocument is found in the **System.XML.Linq** namespace and has a static **Load()** method that reads an entire XML file and return an XDocument. Or you can use the static **Parse()** method that takes a string and also returns an XDocument. Using the same Category data from previous examples, we can create XML with the same data and structure.

```
<?xml version="1.0" encoding="utf-8" ?>
<Categories>
  <Category Title="Fiction=" Description="A wide range of fiction from the worlds best authors." />
  <Category Title="Business and Investing" Description="Timely strategies and advice." />
  <Category Title="Cooking, Food and Wine=" Description="Cook books for all budgets and palates." />
</Categories>
```

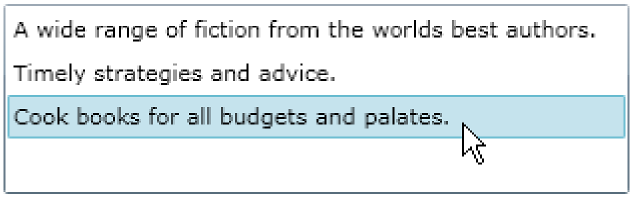
This XML can be copied to a "Categories.xml" file and added to the project. In the constructor of the UserControl, call the XDocument.Load() method and pass the path of the file, i.e. "Categories.xml". XDocument has **Element()** and **Elements()** methods that return XElement and collections of XElement, respectively. The example below first returns a collection of "Category" elements. A LINQ statement is used to drill down to each Category "Title" and "Description" attribute and create new Category objects. This collection of categories is assigned to the ItemsSource property and the DisplayMemberPath points to "Description".

```
Imports System.Linq
Imports System.Windows.Controls
Imports System.Xml.Linq

Namespace _06_XML
  Partial Public Class MainPage
    Inherits UserControl
    Public Sub New()
      InitializeComponent()
      Dim document As XDocument = XDocument.Load("Categories.xml")
      Dim categoryElements = document.Element("Categories").Elements("Category")
      Dim categories = _
        From c In categoryElements _
        Select New Category()
          c.Attribute("Title").Value, Description = c.Attribute("Description").Value
          Title = c.Attribute("Title").Value, Description
      lbCategories.ItemsSource = categories
      lbCategories.DisplayMemberPath = "Description"
    End Sub
  End Class
End Namespace
```

C#

```
using System.Linq;
using System.Windows.Controls;
using System.Xml.Linq;

namespace _06_XML
{
  public partial class MainPage : UserControl
  {
    public MainPage()
    {
      InitializeComponent();
      XDocument document = XDocument.Load("Categories.xml");
      var categoryElements = document.Element("Categories").Elements("Category");
      var categories = from c in categoryElements
        select new Category()
        {
          Title = c.Attribute("Title").Value,
          Description = c.Attribute("Description").Value
        };
      lbCategories.ItemsSource = categories;
      lbCategories.DisplayMemberPath = "Description";
    }
  }
}
```

The running Silverlight application results in the same display as when we bound the collection of Category through other means.



A wide range of fiction from the worlds best authors.
Timely strategies and advice.
Cook books for all budgets and palates.

4.6.2 REST

You can find a number of RESTful web services from popular web sites like YouTube, Flicker and Twitter. These services are especially attractive because you don't need to deal with SOAP protocol, but instead, you can simply use a WebClient object to download a resource from a standard URL. The returned string is likely to be in XML, RSS or ATOM format. RSS and ATOM are XML variants and so the XDocument and LINQ techniques explained in the previous XML section work well with REST service results.

For numerous example walk through using REST services, see the Binding sections of the "GridView", "Docking", "ComboBox", "CoverFlow", "MediaPlayer" and "Charting" chapters.

The following explains the basic techniques of using REST services and a discussion of issues involved with downloading images in Silverlight.

Using the WebClient

The backbone of REST binding projects is the **WebClient** object from the Silverlight **System.Net** namespace. It has two key methods, **DownloadStringAsync()** and **OpenReadAsync()**. DownloadStringAsync() is used to bring down the initial XML document from a service and is paired with a **DownloadStringCompleted** event handler. OpenReadAsync() can be used to download images or other binary data and is paired with a **OpenReadCompleted** event handler. Here's a short example of using the WebClient DownloadStringAsync() method that searches Twitter:

VB

```
Dim webClient As New WebClient()
AddHandler webClient.DownloadStringCompleted, _
 AddressOf client_DownloadStringCompleted

If (Not webClient.IsBusy) Then
  webClient.DownloadStringAsync( _
New Uri("http://search.twitter.com/search.atom?q=telerik"))
End If
```

C#

```
WebClient webClient = new WebClient();
webClient.DownloadStringCompleted +=
   new DownloadStringCompletedEventHandler(client_DownloadStringCompleted);

if (!webClient.IsBusy)
   webClient.DownloadStringAsync(
      new Uri("http://search.twitter.com/search.atom?q=telerik"));
```

Inside the DownloadStringCompleted event handler you have access to the argument's **Result** property. Result contains XML that looks something like this abbreviated version:

```
<feed
 <entry>
  <id>tag:search.twitter.com,2005:4175721855</id>
  <published>2009-09-22T16:37:21Z</published>
  <link type="text/html"
     href="http://twitter.com/timheuer/statuses/4175721855"
     rel="alternate" />
  <title>RT @nikiatanasov: Telerik announces new Book control for
    Silverlight/WPF for Q3 http://tinyurl.com/maguwd</title>
  <content type="html">RT &lt;a href="http://twitter.com/nikiatanasov"&gt;
    @nikiatanasov&lt;/a&gt;: &lt;b&gt;Telerik&lt;/b&gt; announces new Book
     control for Silverlight/WPF for Q3 &lt;a href="http://tinyurl.com/maguwd"
     &gt;http://tinyurl.com/maguwd&lt;/a&gt;</content>
  <updated>2009-09-22T16:37:21Z</updated>
  <link type="image/png"
     href="http://a1.twimg.com/profile_images/426936590/tim-twitter_normal.png"
     rel="image" />
  <twitter:source>&lt;a href="http://www.seesmic.com/"
      rel="nofollow"&gt;Seesmic&lt;/a&gt;</twitter:source>
  <twitter:lang>en</twitter:lang>
  <author>
    <name>timheuer (Tim Heuer)</name>
    <uri>http://twitter.com/timheuer</uri>
  </author>
 </entry>
</feed>
```

You can use the static XDocument Parse() method and pass the result XML to create an XDocument instance. LINQ works nicely against XDocument so that you can select data from the document and create a collection suitable for binding. The example below is primarily conceptual, so don't expect to copy and paste it.

```
Private Sub client_DownloadStringCompleted(ByVal sender As Object, _
 ByVal e As DownloadStringCompletedEventArgs)
  Dim xDocument As XDocument = XDocument.Parse(e.Result)

  gvMain.ItemsSource = _
    From item In xDocument.Descendants("entry") _
    Select
End Sub
```

C#

```
void client_DownloadStringCompleted(object sender,
  DownloadStringCompletedEventArgs e)
{
  XDocument xDocument = XDocument.Parse(e.Result);

  gvMain.ItemsSource = from item in xDocument.Descendants("entry")
    select;
}
```

The WebClient OpenReadAsync() method and OpenReadCompleted event have a similar pattern to their DownloadStringAsync siblings. In this case we take advantage of a second parameter called "userToken" that can be any object you care to send along. We can send a reference to an Image object that's already on the page.

VB

```
Dim webClient As New WebClient()
AddHandler webClient.OpenReadCompleted, _
AddressOf webClient_OpenReadCompleted
webClient.OpenReadAsync(New Uri(imageUrl), image)
```

C#

```
WebClient webClient = new WebClient();
webClient.OpenReadCompleted +=
  new OpenReadCompletedEventHandler(webClient_OpenReadCompleted);
webClient.OpenReadAsync(
  new Uri(imageUrl), image);
```

The Result parameter for the OpenReadCompleted event handler is a stream that contains our Image data. Create a BitmapImage and set its source to be the Result stream. Finally, assign the BitmapImage as the source of the Image control on the page.

```
Private Sub webClient_OpenReadCompleted( _
ByVal sender As Object, ByVal e As OpenReadCompletedEventArgs)
  Dim bitmap As New BitmapImage()
  Try
    bitmap.SetSource(e.Result)
    TryCast(e.UserState, Image).Source = bitmap
  Catch ex As System.Reflection.TargetInvocationException
    If Not(TypeOf ex.InnerException Is System.Security.SecurityException) Then
      MessageBox.Show(ex.Message)
      Throw
    End If
  End Try
End Sub
```

```
void webClient_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
   BitmapImage bitmap = new BitmapImage();
   try
   {
      bitmap.SetSource(e.Result);
      (e.UserState as Image).Source = bitmap;
   }
   catch (System.Reflection.TargetInvocationException ex)
   {
      if (!(ex.InnerException is System.Security.SecurityException))
      {
         MessageBox.Show(ex.Message);
         throw;
      }
   }
}
```

Image Issues

The image downloading code above is actually incomplete. There are two common problems that occur when grabbing images from uncontrolled sources:

- As of this writing, Silverlight supports only ".png" and ".jpg" images. For the Twitter service, you can check the file extension when you parse the initial document. If the image is not a supported extension you can either skip loading the image altogether or load a default image of some sort.

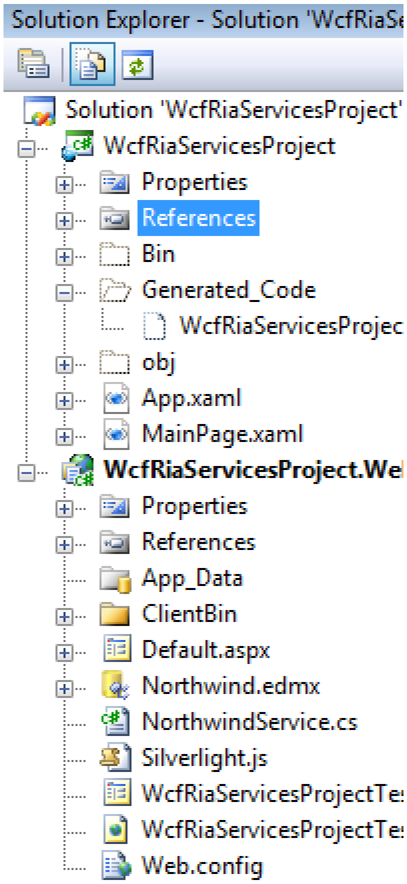
- Silverlight restricts "Cross Domain" access, i.e., where the image data is retrieved from an internet domain that is different from the requesting application. This may be allowed if the server where the image resides has a file called ClientAccessPolicy.xml and is configured to permit the download. In sites that permit access to both Silverlight and Flash, the server may have a "Crossdomain.xml" file instead (as is currently the case with Twitter). **System.Reflection.TargetInvocationException** is raised when one of these files doesn't exist or is not configured to allow the download. You should test that the inner exception is **System.Security.SecurityException**.

4.6.3 RIA

The marketing intro raises the key points:

> "Microsoft WCF RIA Services simplifies the traditional n-tier application pattern by bringing together the ASP.NET and Silverlight platforms. RIA Services provides a pattern to write application logic that runs on the mid-tier and controls access to data for queries, changes and custom operations. It also provides end-to-end support for common tasks such as data validation, authentication and roles by integrating with Silverlight components on the client and ASP.NET on the mid-tier."

The very short story is that WCF RIA Services allows you can write code describing your entities once and to share that data with the client. You first create a "domain service" in the ASP.NET application that is flagged to "EnableClientAccess". You describe your entities there (typically using LINQ to Entities, but this is not required). The Silverlight client application must be in the same solution with the service and linked to the service. Visual Studio takes care of generating client proxy code. The client code is termed the "Context" object.



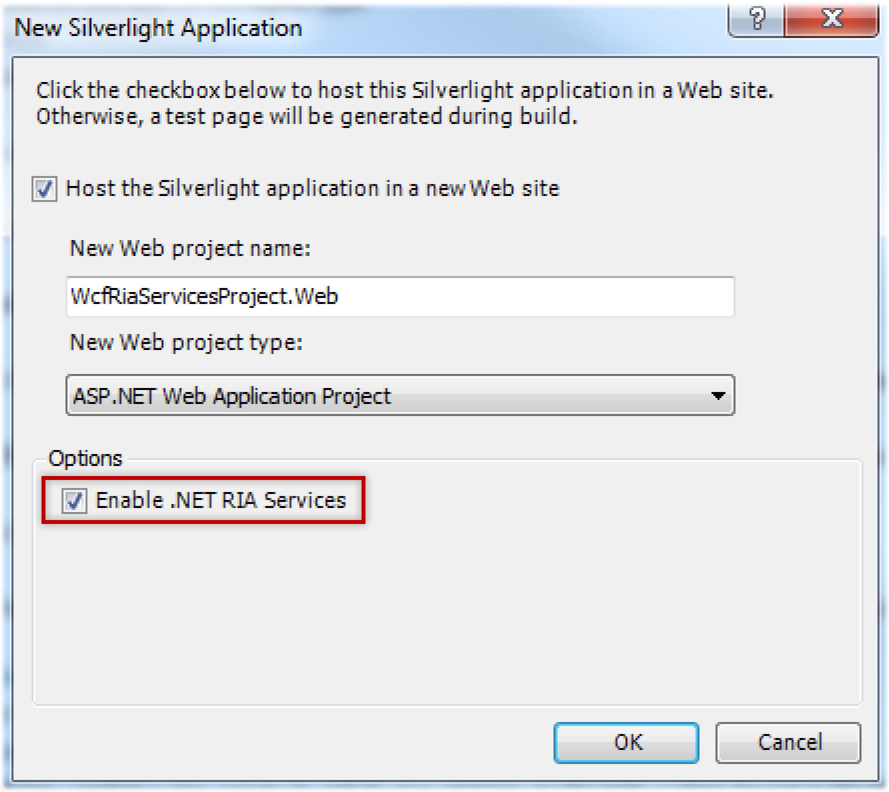
A typical WCF RIA Services solution in the Solution Explorer might look like this screenshot. The top-most project, "WcfRiaServicesProject" is a Silverlight application. The "Show All Files" button is pressed and we can see a folder called "Generated_Code" that contains the automatically created proxy class. The " WcfRiaServicesProject.Web" project is an ASP.NET web application that contains a domain service.

For more high-level information on RIA Services, read the blog "What is .NET RIA Services?" at http://blogs.msdn.com/brada/archive/2009/03/19/what-is-net-ria-services.aspx by Brad Abrams.

The following walk through barely scratches the surface with RIA Services, but will make you familiar with the basic process of defining and consuming a service. The example service consumes the Northwind "Employees" table using LINQ to Entities. The Silverlight client instantiates a proxy client, retrieves a list of entities and binds them to a RadGridView.

4.6.3.1 Project Setup

1) In Visual Studio, create a new Silverlight Application. This will display the **New Silverlight Application** dialog. Check the "Enable .NET RIA Services" option. Leave the other defaults and click **OK** to close the dialog and create the projects. *The RIA service layer will be created in the ASP.NET host application, in this case, the WcfRiaServicesProject.Web project.*



New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new Web site

New Web project name:

WcfRiaServicesProject.Web

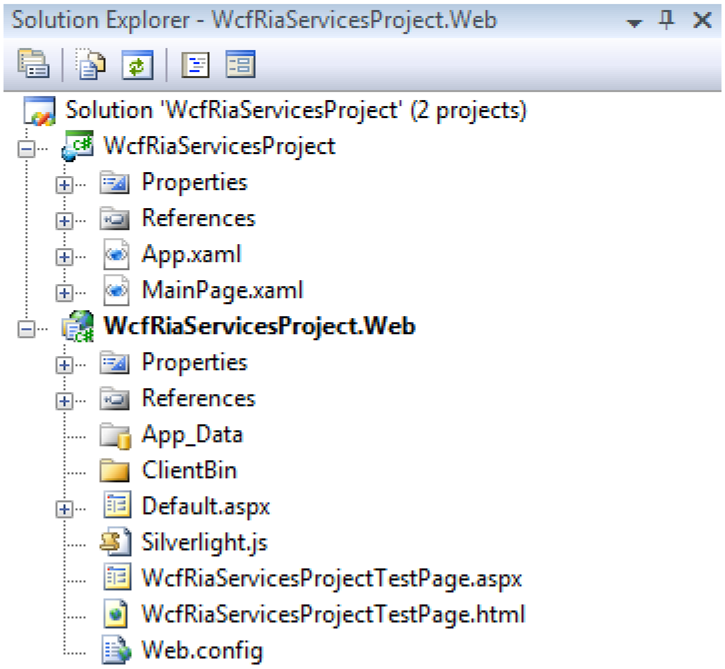
New Web project type:

ASP.NET Web Application Project

Options

☑ Enable .NET RIA Services

2) Take a look at the Solution Explorer and notice that you now have two projects, the ASP.NET host project ("WcfRiaServicesProject.Web in the screenshot below) that contains the RIA service and the Silverlight client application ("WcfRiaServicesProject").

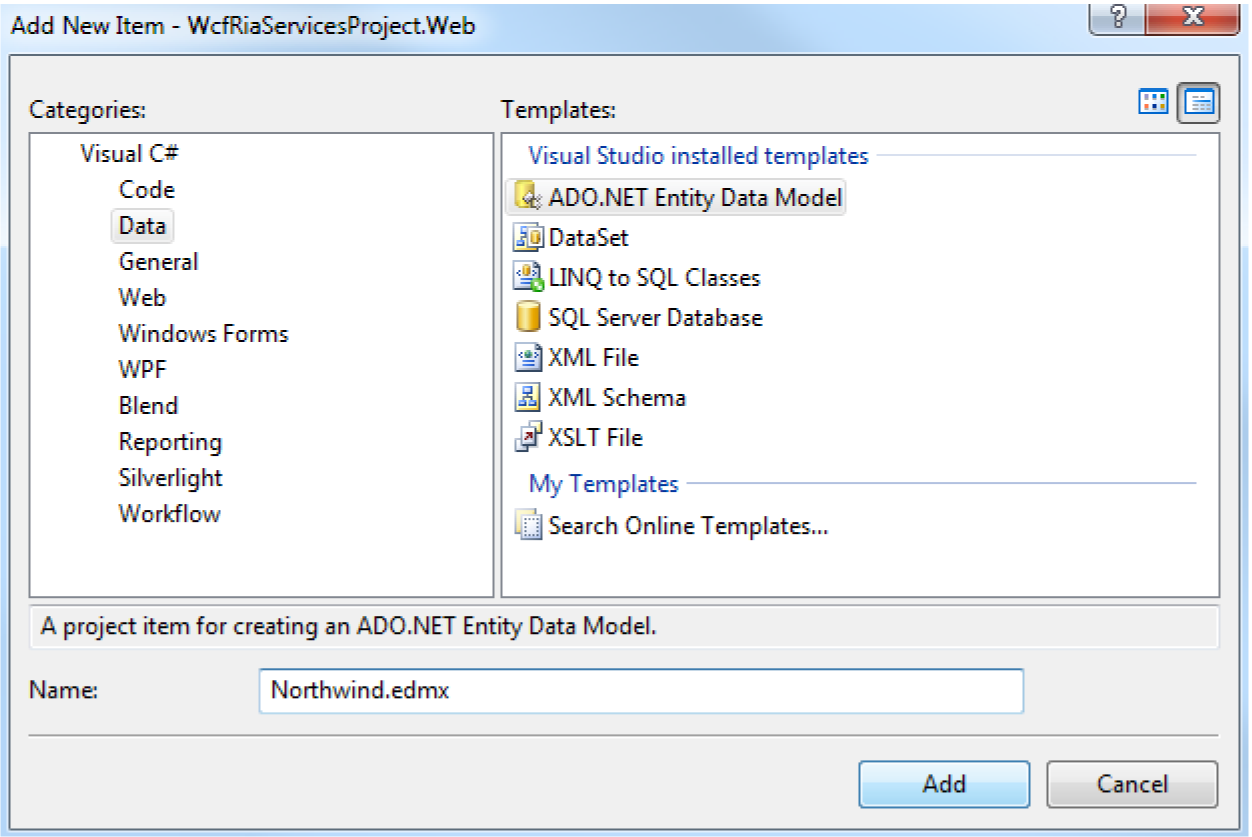


Solution Explorer - WcfRiaServicesProject.Web

```
Solution 'WcfRiaServicesProject' (2 projects)
⊟ WcfRiaServicesProject
    ⊞ Properties
    ⊞ References
    ⊞ App.xaml
    ⊞ MainPage.xaml
⊟ WcfRiaServicesProject.Web
    ⊞ Properties
    ⊞ References
      App_Data
      ClientBin
    ⊞ Default.aspx
      Silverlight.js
      WcfRiaServicesProjectTestPage.aspx
      WcfRiaServicesProjectTestPage.html
      Web.config
```

4.6.3.2 Building the RIA Service

1) Add a "ADO.NET Entity Data Model" item to the ASP.NET project. Name the data model "Northwind.edmx" and click the **Add** button. *This will display the Entity Data Model Wizard dialog.*



A project item for creating an ADO.NET Entity Data Model.

Name: Northwind.edmx

2) In the first page of "Entity Data Model Wizard", select the "Generate from Database" icon and click the **Next** button.

3) Create a connection to the Northwind database file that ships with RadControls for Silverlight.

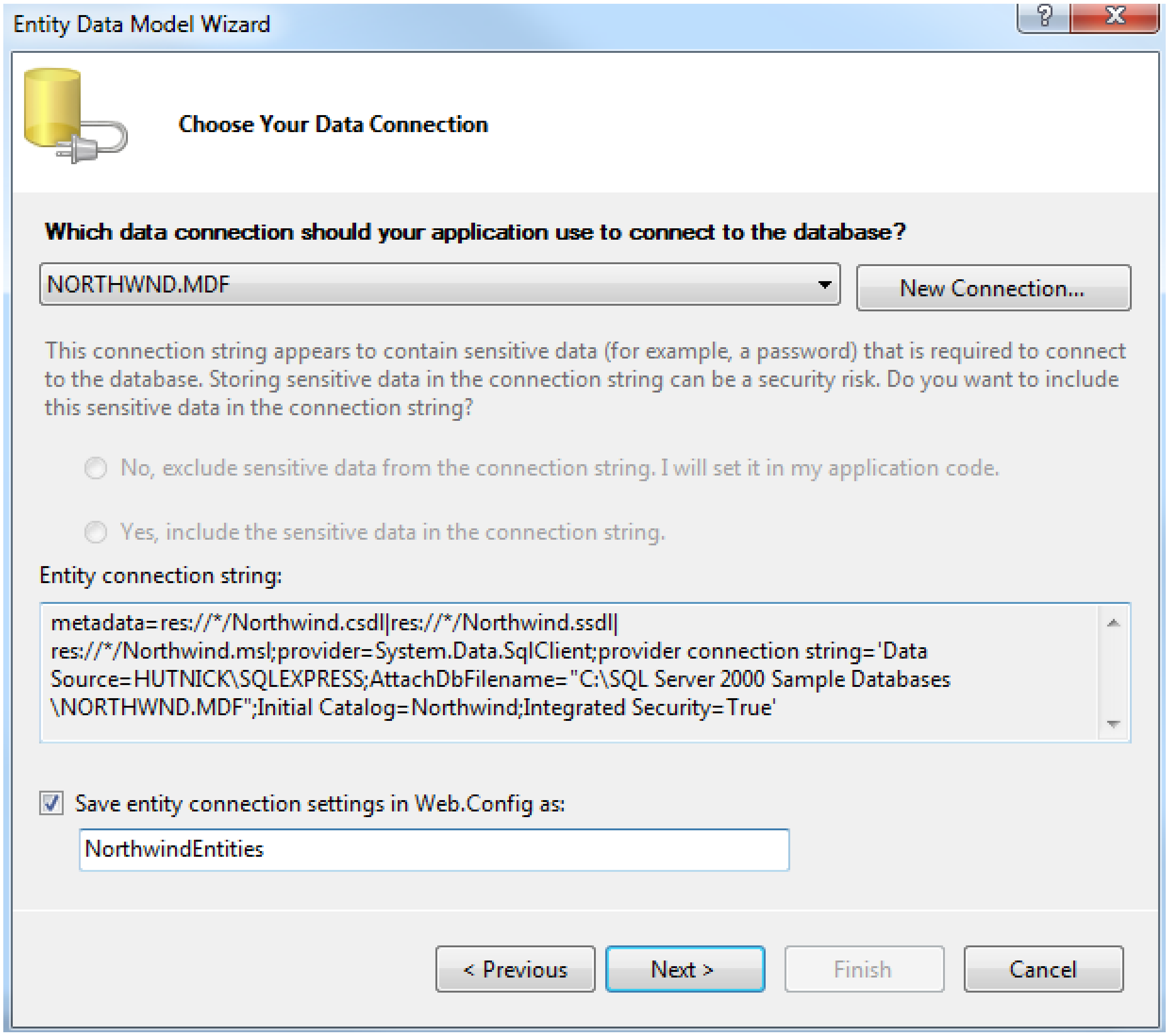
a) In the "Choose your Data Connection" page of the wizard, click the **New Connection** button. This will display the Connection Properties dialog.

b) Click the **Change** button to show the "Change Data Source" dialog, select the "Microsoft SQL Server Database File" option and click the **OK** button to return to the "Change Data Source" dialog.

c) Click the **Browse** button, locate the file "Northwind.mdf" file in the RadControls for Silverlight installation directory under \Demos\DataSources.

d) Click **OK** to create the connection and return to the "Entity Data Model Wizard" "Choose Your Data Connection" page.

4) In the "Choose Your Data Connection" page of the wizard, enter "NorthwindEntities" in the "Save entity connection settings in Web.Config" text box and click the **Next** button.



Which data connection should your application use to connect to the database?

NORTHWND.MDF

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

○ No, exclude sensitive data from the connection string. I will set it in my application code.

○ Yes, include the sensitive data in the connection string.

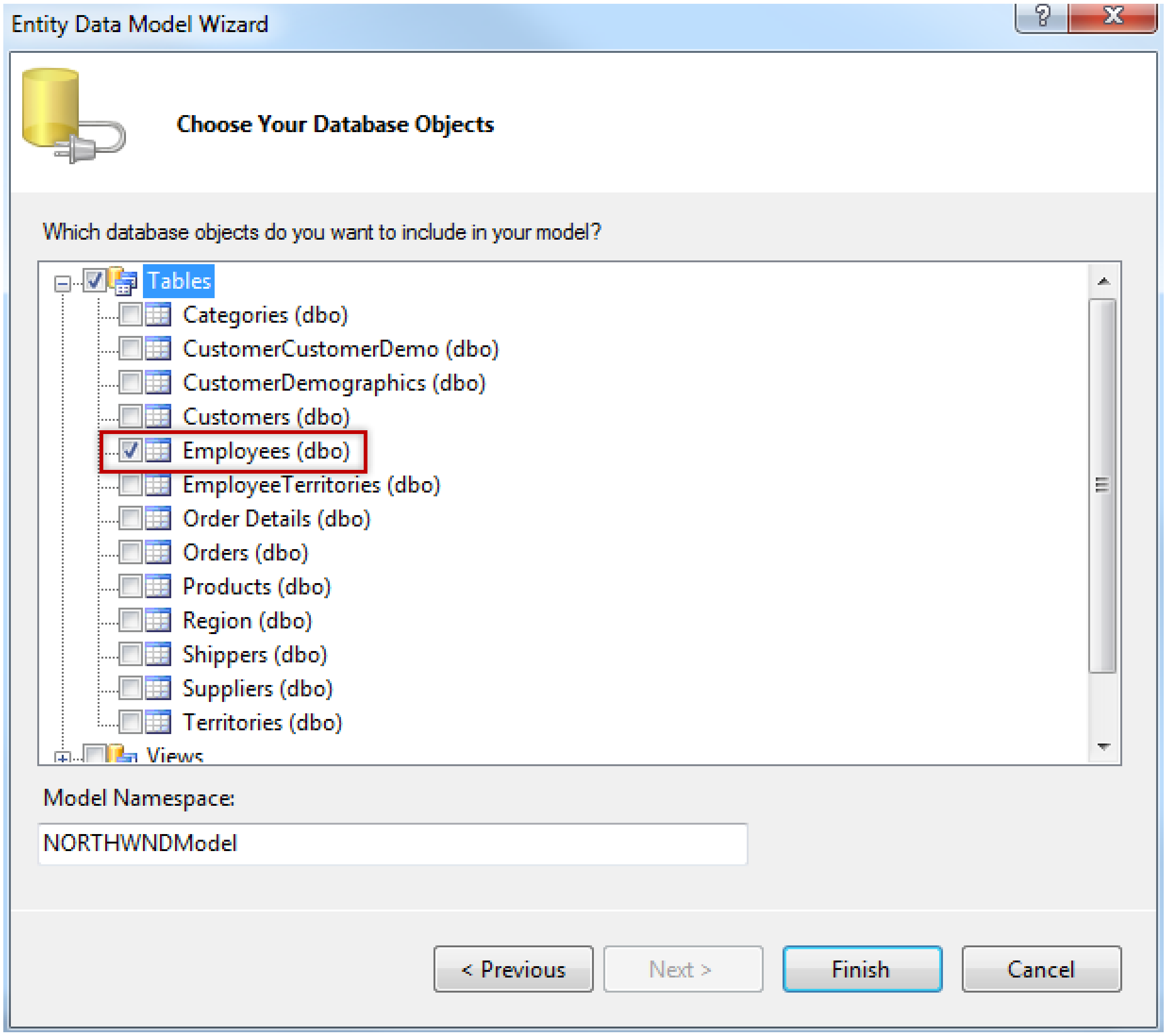
Entity connection string:

```
metadata=res://*/Northwind.csdl|res://*/Northwind.ssdl|
res://*/Northwind.msl;provider=System.Data.SqlClient;provider connection string='Data
Source=HUTNICK\SQLEXPRESS;AttachDbFilename="C:\SQL Server 2000 Sample Databases
\NORTHWND.MDF";Initial Catalog=Northwind;Integrated Security=True'
```

☑ Save entity connection settings in Web.Config as:

NorthwindEntities

5) In the "Choose Your Database Objects" page of the wizard, expand the "Tables" node in the tree view and select the "Employees" table check box.



Choose Your Database Objects

Which database objects do you want to include in your model?

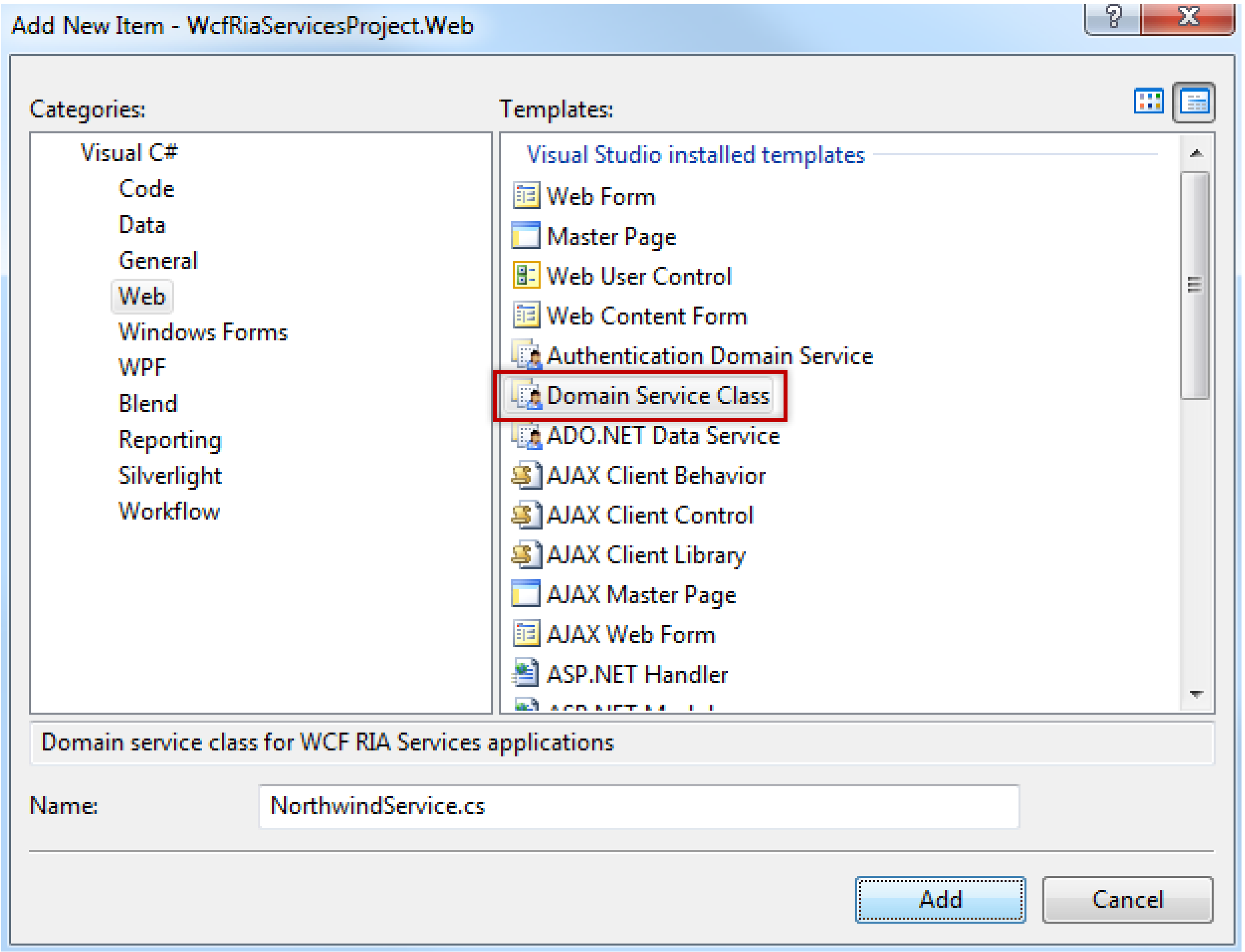
- ☑ Tables
  - □ Categories (dbo)
  - □ CustomerCustomerDemo (dbo)
  - □ CustomerDemographics (dbo)
  - □ Customers (dbo)
  - ☑ Employees (dbo)
  - □ EmployeeTerritories (dbo)
  - □ Order Details (dbo)
  - □ Orders (dbo)
  - □ Products (dbo)
  - □ Region (dbo)
  - □ Shippers (dbo)
  - □ Suppliers (dbo)
  - □ Territories (dbo)
- □ Views

Model Namespace: NORTHWNDModel

6) Click the **Finish** button to create the data model.

7) Build the ASP.NET project. **This step is important**: the following step where you create the Domain Service Class will not see the entity data model information without building the project.

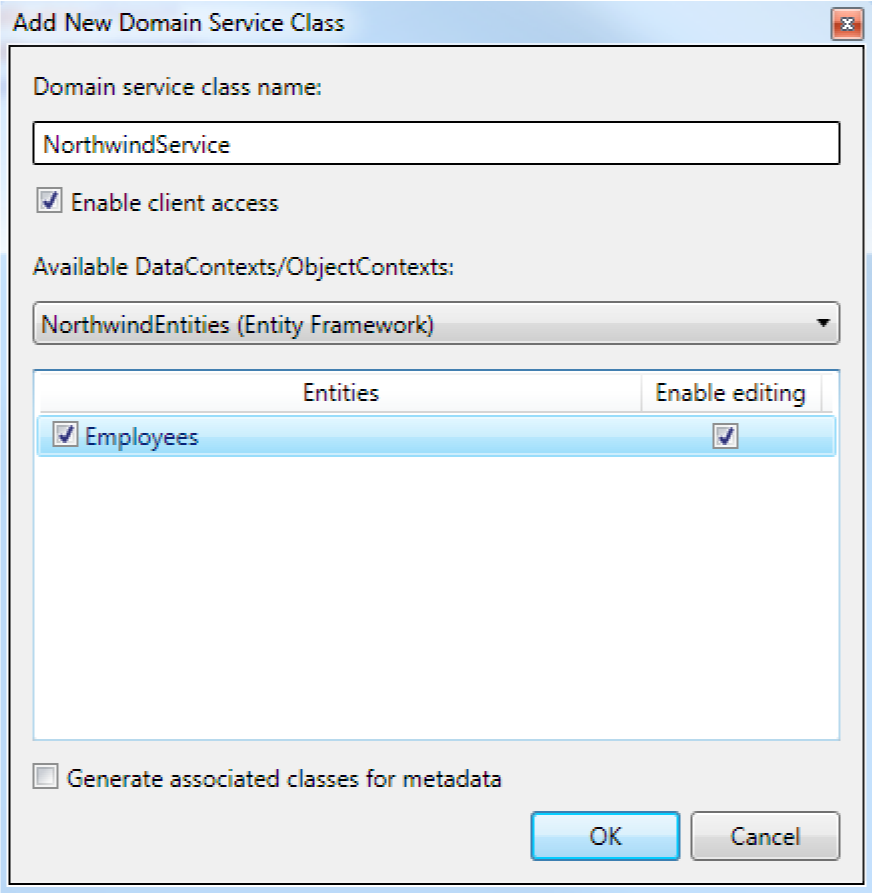
8) Add a "Domain Service Class" item to your project. Name it "NorthwindService" and click the **Add** button to create the service and close the dialog. This step will display the "Add New Domain Service Class" dialog.



Domain service class for WCF RIA Services applications

Name: NorthwindService.cs

9) In the "Add New Domain Service Class" dialog, name the Domain Service Class "NorthwindService", make sure that the "Enable client access" option is checked (this allows the client proxy code to be generated), select "NorthwindEntities" from the drop down list of available context objects and check the "Employees" entity check box. Click the **OK** button to create the domain service class and close the dialog.



Add New Domain Service Class

Domain service class name:

NorthwindService

☑ Enable client access

Available DataContexts/ObjectContexts:

NorthwindEntities (Entity Framework)

Entities	Enable editing
☑ Employees	☑

☐ Generate associated classes for metadata

10)Build the project.

11)Review the generated NorthwindService code. Notice that a GetEmployees() (along with Insert, Update, and Delete, since we enabled editing) method has been created. that returns an IQueryable<> of Employees.

4.6.3.3 Building the RIA Client

1) In the Solution Explorer, open the Silverlight client application node.
2) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add references to these assemblies:
   a) **Telerik.Windows.Controls**
   b) **Telerik.Windows.Data**
   c) **Telerik.Windows.Controls.GridView**
   d) **Telerik.Windows.Controls.Input**
3) Open "MainPage.xaml" for editing.
4) Drag a RadGridView from the Toolbox into the main "LayoutRoot" grid element of the page. Set the "x:Name" attribute to "gvMain" so that we can refer to it later in code.
5) Open "MainPage.xaml.cs" for editing.
6) Add a namespace reference to the ASP.NET service project "Imports" (VB) or "using" (C#) portion of code.
7) In the constructor for the UserControl, get the data from the domain service by way of the context object and bind it to the grid view using the code below:
   a) Create the "context" object (the generated client counterpart to the domain service).
   b) Assign the context "Employees" property to the grid view ItemsSource property.
   c) Call the context object Load() method and pass the context "GetEmployeesQuery()" method.

VB

```
Public Sub New()
  InitializeComponent()

  Dim context As New NorthwindContext()
  gvMain.ItemsSource = context.Employees
  context.Load(context.GetEmployeesQuery())
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();

   NorthwindContext context = new NorthwindContext();
   gvMain.ItemsSource = context.Employees;
   context.Load(context.GetEmployeesQuery());
}
```

Notes

If you place your cursor on the Load() method and press F12 to see its definition, the Load() method is of type LoadOperation<TEntity> and represents an asynchronous load operation. You can get the return value from the Load() call and use it to attach a Completed event handler. From there you can respond to errors or fine tune the data before assigning to the grid. Here's an example that checks for the Error exception object and uses LINQ to only show employees with first names starting with "A".

VB

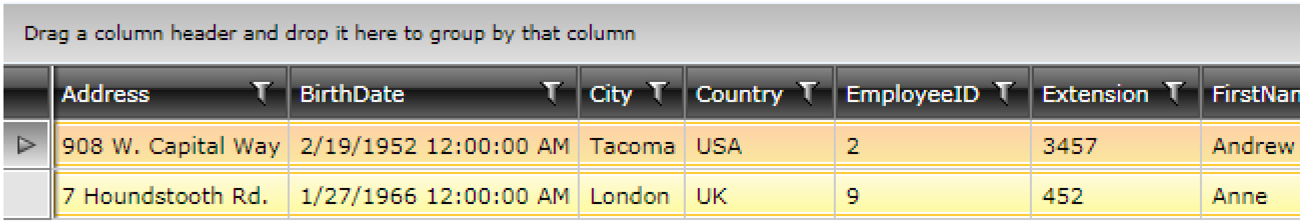
```
Public Sub New()
  InitializeComponent()
  Dim context As New NorthwindContext()
  Dim operation As LoadOperation = context.Load(context.GetEmployeesQuery())
  AddHandler operation.Completed, AddressOf operation_Completed
End Sub

Private Sub operation_Completed(ByVal sender As Object, ByVal e As EventArgs)
  Dim loadOperation As LoadOperation = TryCast(sender, LoadOperation)
  If loadOperation.Error Is Nothing Then
    gvMain.ItemsSource = _
      From employee As Employees In loadOperation.Entities _
      Where employee.FirstName.StartsWith("A") _
      Select employee
  End If
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();
   NorthwindContext context = new NorthwindContext();
   LoadOperation operation = context.Load(context.GetEmployeesQuery());
   operation.Completed += new EventHandler(operation_Completed);
}

void operation_Completed(object sender, EventArgs e)
{
   LoadOperation loadOperation = sender as LoadOperation;
   if (loadOperation.Error == null)
   {
      gvMain.ItemsSource = from Employees employee in loadOperation.Entities
         where employee.FirstName.StartsWith("A")
         select employee;
   }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Drag a column header and drop it here to group by that column

	Address	BirthDate	City	Country	EmployeeID	Extension	FirstNar
▷	908 W. Capital Way	2/19/1952 12:00:00 AM	Tacoma	USA	2	3457	Andrew
	7 Houndstooth Rd.	1/27/1966 12:00:00 AM	London	UK	9	452	Anne

4.6.4 WCF

Windows Communication Foundation (WCF) is an industrial strength infrastructure for moving data around on the network. WCF separates the communication plumbing from the code that actually works with the data. Silverlight applications can call WCF services provided we include a few tweaks that make WCF and Silverlight happy with our security arrangements.

If the WCF service resides in a different domain than the client, i.e. a different port number for example, the call is considered to be cross domain. To provide "cross domain access" to resources from a Silverlight application, you must include a properly configured "ClientAccessPolicy.xml" file in the root where the service lives. ClientAccessPolicy.xml defines the resources that can be retrieved and where they can be retrieved from. The walk through following will show you how to do this using a generic ClientAccessPolicy.xml that allows all traffic. You can "batten down the hatches" to restrict traffic later by making the policy file entries more specific.

See the articles "Network Security Access Restrictions in Silverlight" (http://msdn.microsoft.com/en-us/library/cc645032(VS.95).aspx and Http Communication with Silverlight (http://msdn.microsoft.com/en-us/library/cc838250(VS.95).aspx) for more information.

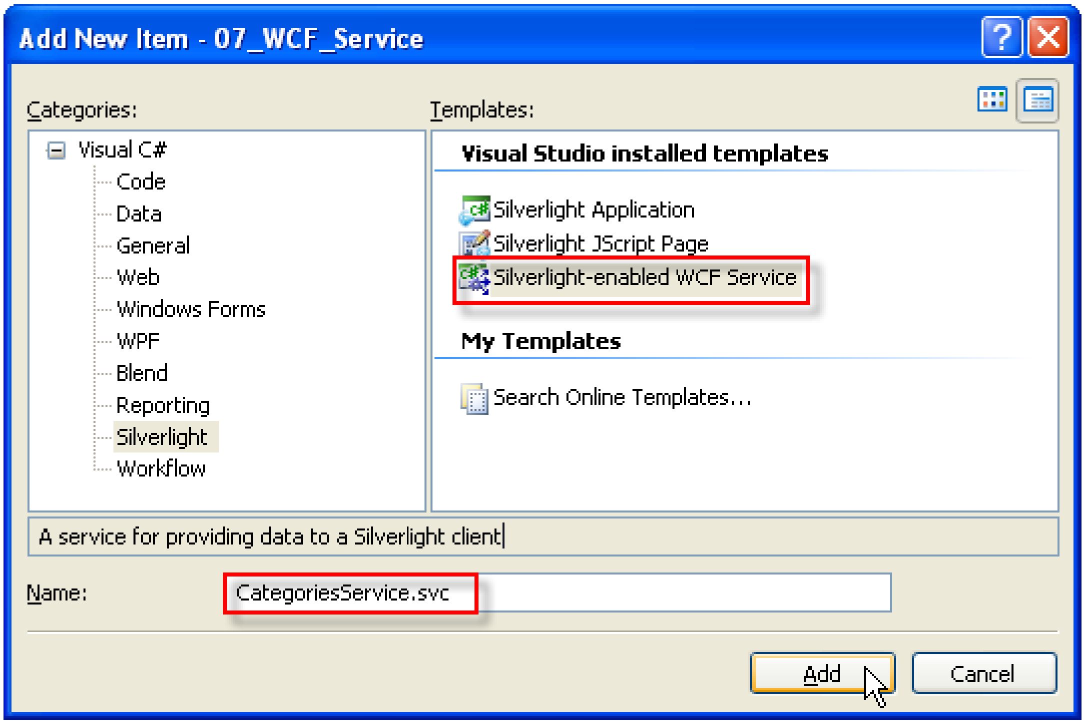
The service for this walk through will bring back a simple list of categories. The actual logic or data access in the service is not critical. In the background you could be using a SQL command, creating objects on the fly, using Entities, WebClient, reading XML files, reading from isolated storage or any combination. This all takes place in a server environment, so you have a lot of latitude.

4.6.4.1 Building the WCF Service

1) Create a new ASP.NET Web Application project. *This application will host the WCF service.*

2) In the Solution Explorer, right-click the web application project and select **Add > New Item...** from the context menu.

3) Select the "Silverlight-enabled WCF Service" template, name it "CategoriesService.svc" and click the **Add** button to create the service and close the dialog.



A service for providing data to a Silverlight client

Name: CategoriesService.svc

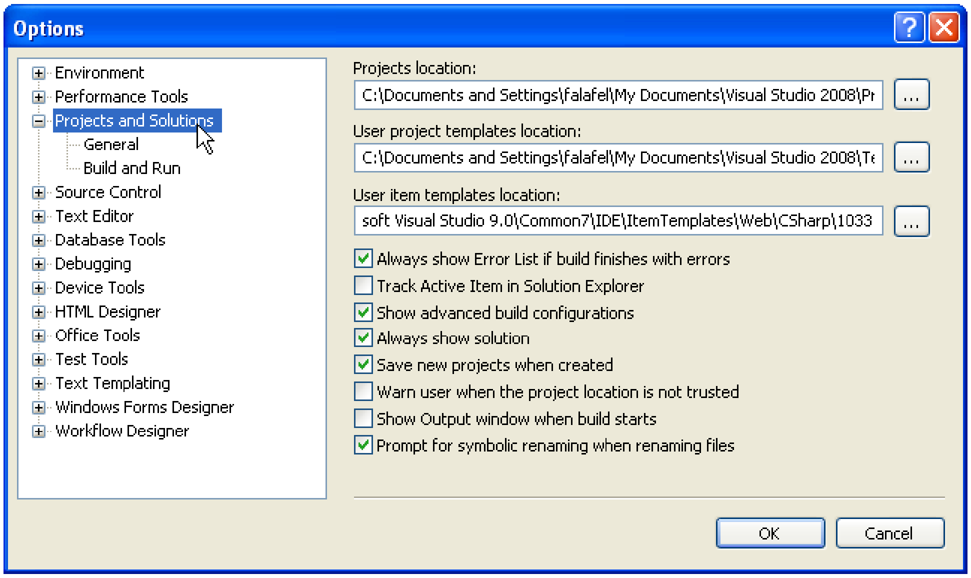
Gotcha!

The Silverlight-enabled WCF Service template type is available for Visual Studio, SP1. Be aware that it's not a project type, but rather a new item type that you add to an existing project, i.e. a web application. If you have Visual Studio SP1, are adding an item to an existing ASP.NET Web Application project and still don't see the "Silverlight-enabled WCF Service" template type, you may have an issue with the path that defines the template location.

Using the Visual Studio menu, select the **Tools > Options** item and navigate to "Projects and Solutions" in the tree view. Locate the "User item templates location:" entry on the right and verify that it points to this path:

\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\ItemTemplates\Web\CSharp\1033

The screenshot below shows the options dialog with the default User Item Templates location filled in.

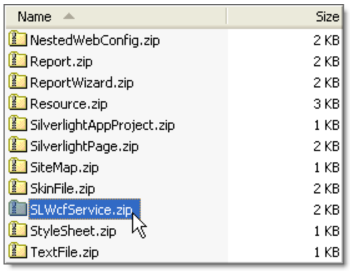


Projects location:
C:\Documents and Settings\falafel\My Documents\Visual Studio 2008\Pr

User project templates location:
C:\Documents and Settings\falafel\My Documents\Visual Studio 2008\Te

User item templates location:
soft Visual Studio 9.0\Common7\IDE\ItemTemplates\Web\CSharp\1033

☑ Always show Error List if build finishes with errors
□ Track Active Item in Solution Explorer
☑ Show advanced build configurations
☑ Always show solution
☑ Save new projects when created
□ Warn user when the project location is not trusted
□ Show Output window when build starts
☑ Prompt for symbolic renaming when renaming files



Name	Size
NestedWebConfig.zip	2 KB
Report.zip	2 KB
ReportWizard.zip	2 KB
Resource.zip	3 KB
SilverlightAppProject.zip	1 KB
SilverlightPage.zip	2 KB
SiteMap.zip	1 KB
SkinFile.zip	2 KB
SLWcfService.zip	2 KB
StyleSheet.zip	1 KB
TextFile.zip	1 KB

The path should contain the template file "SLWcfService.zip". If you have changed your "User Item Template Location" path to some custom location, then just copy "SLWcfService.zip" to the new location.

4) Add a new class file "Category.cs" and replace the class with the code below. The code defines a simple Category class and a generic list of Category with sample categories defined in the constructor.

```
Public Class Categories
  Inherits List(Of Category)
  Public Sub New()
    Me.Add(New Category() With { _
.Title = "Fiction",  _
.Description = "A wide range of fiction from the worlds best authors."})
    Me.Add(New Category() With { _
.Title = "Business and Investing", _
 .Description = "Timely strategies and advice."})
    Me.Add(New Category() With { _
.Title = "Cooking, Food and Wine",  _
.Description = "Cook books for all budgets and palates."})
  End Sub
End Class

Public Class Category
  Private privateTitle As String
  Public Property Title() As String
    Get
      Return privateTitle
    End Get
    Set(ByVal value As String)
      privateTitle = value
    End Set
  End Property
  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
End Class
```

C#

```
public class Categories : List<Category>
{
   public Categories()
   {
      this.Add(new Category()
      {
         Title = "Fiction",
         Description = "A wide range of fiction from the worlds best authors."
      });
      this.Add(new Category()
      {
         Title = "Business and Investing",
         Description = "Timely strategies and advice."
      });
      this.Add(new Category()
      {
         Title = "Cooking, Food and Wine",
         Description = "Cook books for all budgets and palates."
      });
   }
}

public class Category
{
   public string Title { get; set; }
   public string Description { get; set; }
}
```

5) Replace the service DoWork() method with a new GetCategories() method that returns a "Categories" collection.

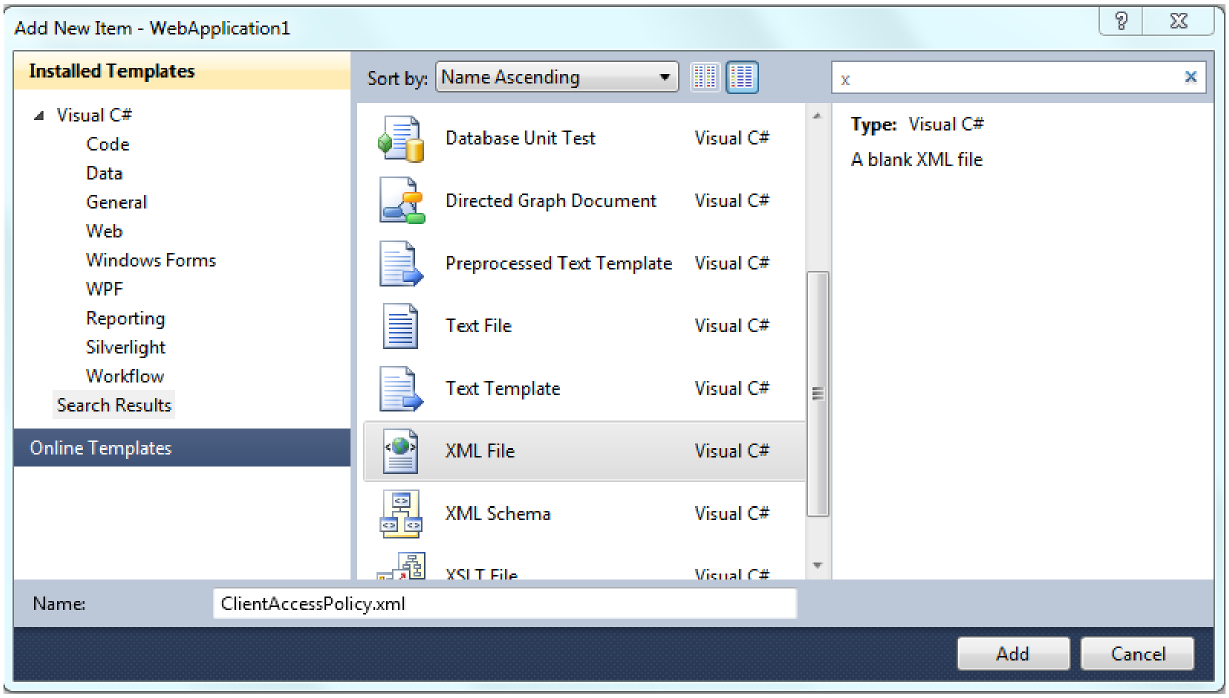
VB

```
<OperationContract> _
Public Function GetCategories() As Categories
  Return New Categories()
End Function
```

C#

```
[OperationContract]
public Categories GetCategories()
{
   return new Categories();
}
```

6) In the Solution Explorer, right-click the project and select **Add > New Item...** from the context menu. Name the file "ClientAccessPolicy.xml", then click the **Add** button to create the file and close the dialog.



Sort by: Name Ascending

Name: ClientAccessPolicy.xml

7) Replace the contents of "ClientAccessPolicy.xml" with the following XML. *This XML allows requests from all domains to get resources from all locations on the server.*

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
 <cross-domain-access>
  <policy>
   <allow-from http-request-headers="*">
    <domain uri="*"/>
   </allow-from>
   <grant-to>
    <resource path="/" include-subpaths="true"/>
   </grant-to>
  </policy>
 </cross-domain-access>
</access-policy>
```

8) In the Solution Explorer, right-click the "ClientAccessPolicy.xml" file and select "Properties" from the context menu. Set the "Copy to Output Directory" property to "Copy if Newer". *This step will make sure that policy file ends up in the \bin directory, i.e. the root directory for the service. When Silverlight tries to access the service, it will find the policy file there and can continue interacting with the service.*

4.6.4.2 Building the WCF Client

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.GridView**

c) **Telerik.Windows.Data**

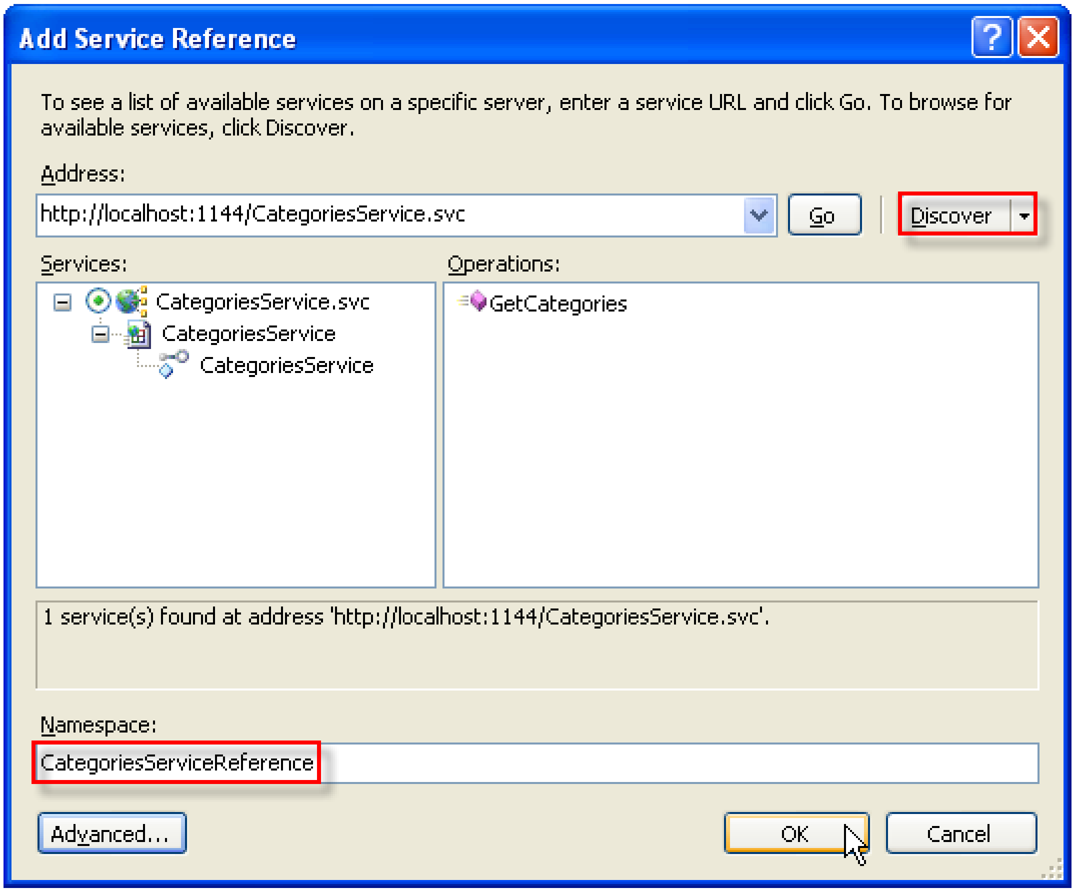
XAML Editing

1) Drag a RadGridView from the Toolbox to a point between the main "LayoutRoot" grid. Set the "x:Name" attribute to "gvMain" so we can reference it later in code.

Reference The WCF Service

1) In the Solution Explorer, right-click the References node and select **Add Service Reference...** This will display the "Add Service Reference" dialog.

2) In the "Add Service Reference" dialog, click the **Discover** button. The CategoriesService.svc server will display. Enter "CategoriesServiceReference" as the Namespace and click **OK** to create the client proxy.



Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:1144/CategoriesService.svc

Services:

- CategoriesService.svc
  - CategoriesService
    - CategoriesService

Operations:

- GetCategories

1 service(s) found at address 'http://localhost:1144/CategoriesService.svc'.

Namespace: CategoriesServiceReference

Code Behind

1) Add a namespace reference for the proxy in the "Imports" (VB) or "using" (C#) section of code. This will be the name of your project + "." + "CategoriesServiceReference", i.e. "MyProject. CategoriesServiceReference".

2) Add the code below to the page constructor to create the proxy and call its methods. . Be sure that you leave the call to InitializeComponent().

The client proxy methods are asynchronous. The proxy will have a "Completed" event, i.e. "GetCategoriesCompleted" and an "Async" method, i.e. "GetCategoriesAsync()". Create an instance of the client proxy object, hook up the "GetcategoriesCompleted" event, then call the "GetCategoriesAsync()" method.

VB

```
Public Sub New()
  InitializeComponent()

  Dim client As New CategoriesServiceClient()
  AddHandler client.GetCategoriesCompleted, AddressOf client_GetCategoriesCompleted

  client.GetCategoriesAsync()
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();

   CategoriesServiceClient client = new CategoriesServiceClient();
   client.GetCategoriesCompleted +=
      new EventHandler<GetCategoriesCompletedEventArgs>(
         client_GetCategoriesCompleted);

   client.GetCategoriesAsync();
}
```

3) Handle the GetCategoriesCompleted event. The result of the method is passed back in e.Result. Assign the result to the grid view **ItemsSource** property. *Note: The result, even though it left the service as a List<Customer>, ends up in Silverlight as ObservableCollection<Customer>.*

VB

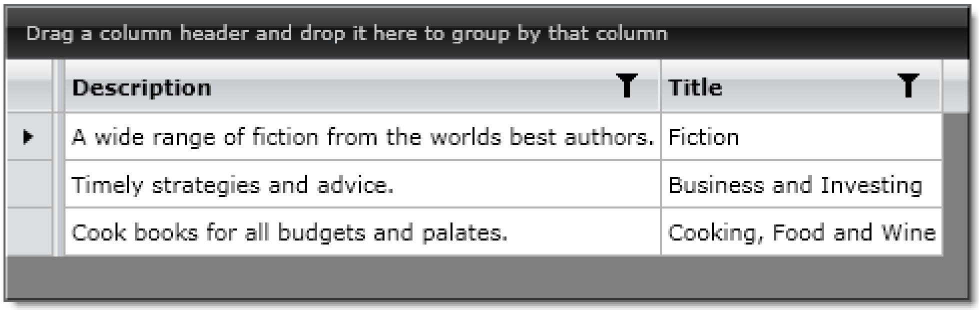
```
Private Sub client_GetCategoriesCompleted( _
 ByVal sender As Object, ByVal e As GetCategoriesCompletedEventArgs)
  gvMain.ItemsSource = e.Result
End Sub
```

C#

```
void client_GetCategoriesCompleted(object sender, GetCategoriesCompletedEventArgs e)
{
   gvMain.ItemsSource = e.Result;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



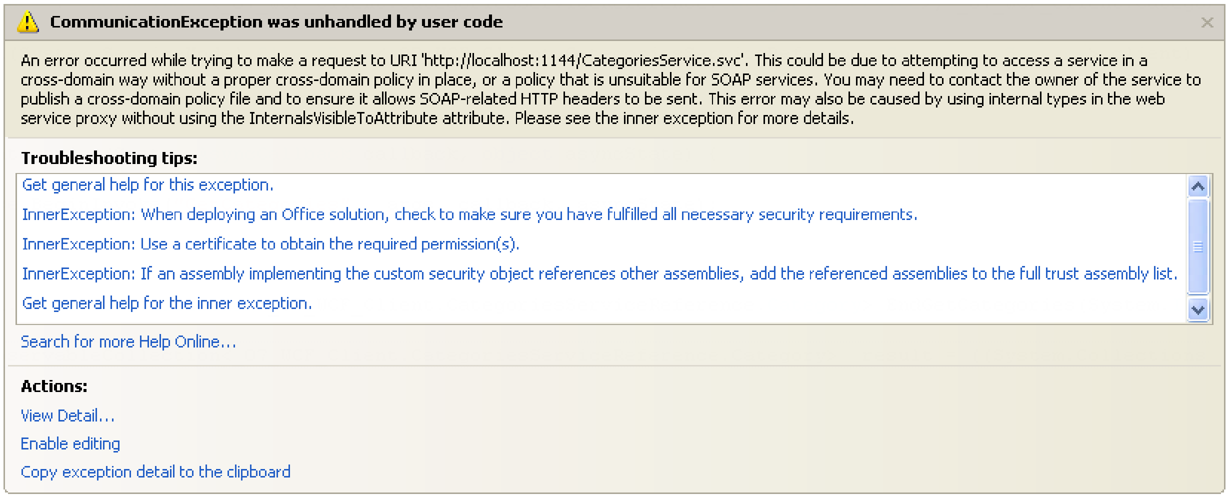
Drag a column header and drop it here to group by that column

Description	Title
A wide range of fiction from the worlds best authors.	Fiction
Timely strategies and advice.	Business and Investing
Cook books for all budgets and palates.	Cooking, Food and Wine

Test Application Features

1) All data with all columns from the Categories collection should show up in the grid.

2) Test the application without the ClientAccessPolicy.xml present. Running the Silverlight client should display an error "...due to attempting to access a service in a cross-domain way without a proper cross-comain policy in place...".



CommunicationException was unhandled by user code

An error occurred while trying to make a request to URI 'http://localhost:1144/CategoriesService.svc'. This could be due to attempting to access a service in a cross-domain way without a proper cross-domain policy in place, or a policy that is unsuitable for SOAP services. You may need to contact the owner of the service to publish a cross-domain policy file and to ensure it allows SOAP-related HTTP headers to be sent. This error may also be caused by using internal types in the web service proxy without using the InternalsVisibleToAttribute attribute. Please see the inner exception for more details.

Troubleshooting tips:

Get general help for this exception.
InnerException: When deploying an Office solution, check to make sure you have fulfilled all necessary security requirements.
InnerException: Use a certificate to obtain the required permission(s).
InnerException: If an assembly implementing the custom security object references other assemblies, add the referenced assemblies to the full trust assembly list.
Get general help for the inner exception.

Search for more Help Online...

Actions:

View Detail...
Enable editing
Copy exception detail to the clipboard

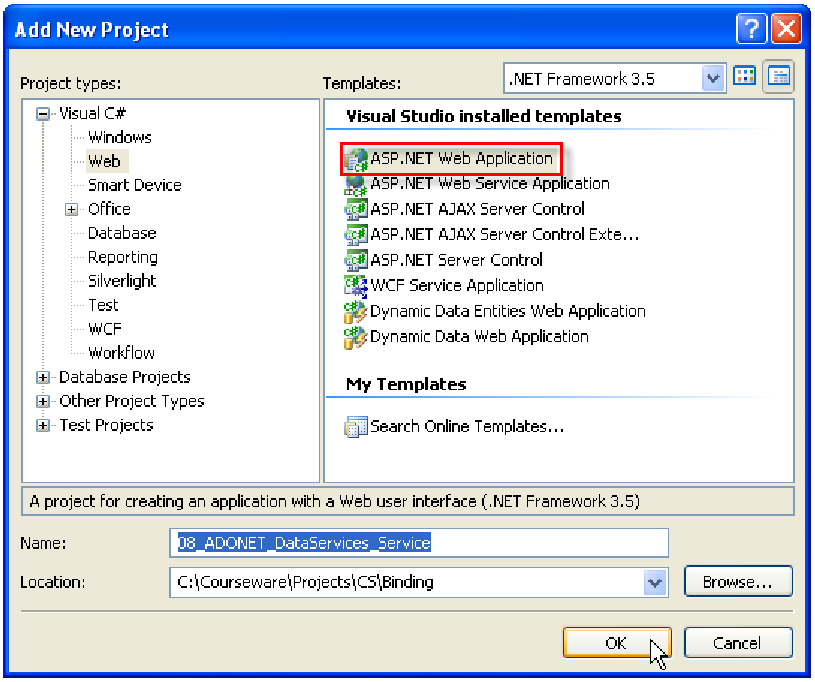
You may not get the error to fire right away. Try renaming or removing the ClientAccessPolicy.xml altogether. Make sure the service is stopped and rebuild it. Then, on the client, right-click the service reference node in the Solution Explorer and choose "Update Service Reference" from the context menu. Its a good idea to get familiar with how these errors occur while you have control in this lab setting.

4.6.5 ADO.NET Data Services

ADO.NET Data Services expose data models as a set of REST Uri's that map to HTTP verbs POST (Create), GET (Read), PUT (Update) and DELETE (Delete). The following walk through demonstrates creating an ADO.NET Entity Data Model of the Northwind database. The example returns a simple address table and displays it in a RadGridView control.

4.6.5.1 Building the Service

1) Create a new ASP.NET Web Application. Give it a unique name, click the **OK** button to create it and close the dialog.

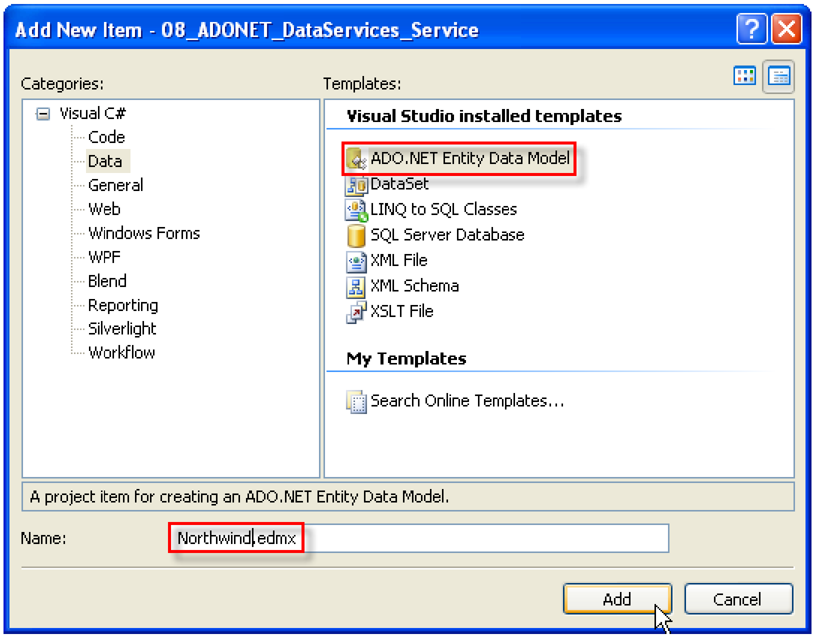


A project for creating an application with a Web user interface (.NET Framework 3.5)

Name: 08_ADONET_DataServices_Service

Location: C:\Courseware\Projects\CS\Binding

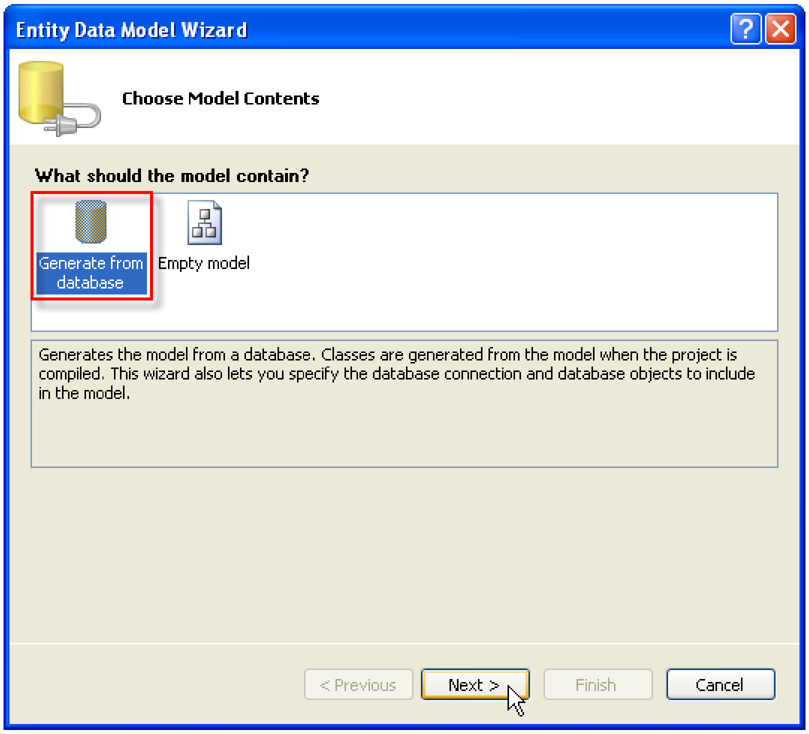
2) Add a "ADO.NET Entity Data Model" item to the ASP.NET project. Name the data model "Northwind.edmx" and click the **Add** button. *This will display the Entity Data Model Wizard dialog.*



A project item for creating an ADO.NET Entity Data Model.

Name: Northwind.edmx

3) In the Entity Data Model Wizard dialog, select "Generate from database" and click the **Next** button to continue.



Generates the model from a database. Classes are generated from the model when the project is compiled. This wizard also lets you specify the database connection and database objects to include in the model.

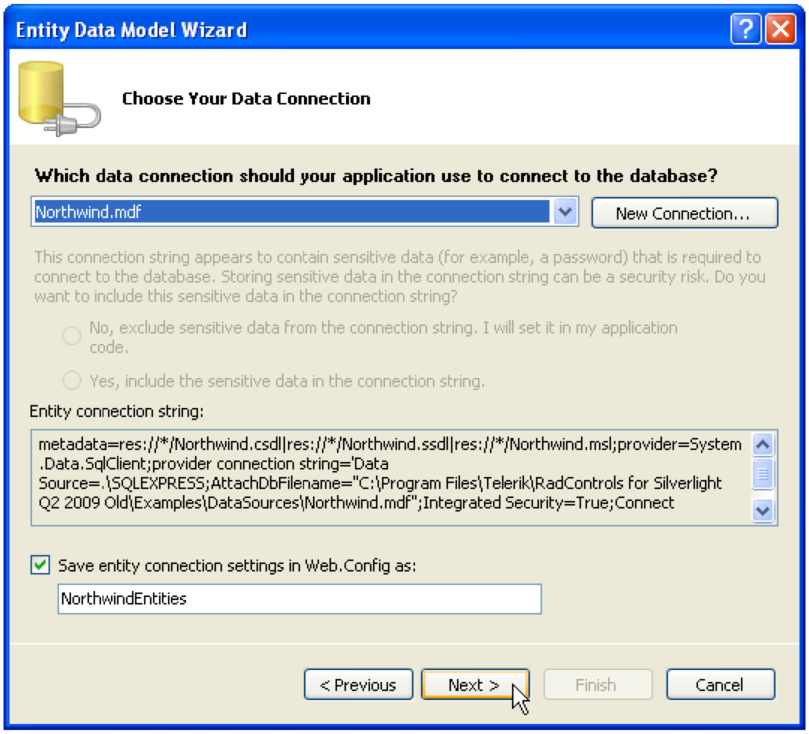
4) Create a connection to the Northwind database file that ships with RadControls for Silverlight.

a) In the "Choose your Data Connection" page of the wizard, click the **New Connection** button. This will display the "Connection Properties" dialog.

b) Click the **Change** button to show the "Change Data Source" dialog, select the "Microsoft SQL Server Database File" option and click the **OK** button to return to the "Change Data Source" dialog.

c) Click the **Browse** button, locate the file "Northwind.mdf" file in the RadControls for Silverlight installation directory under \Demos\DataSources.

d) Click **OK** to create the connection and return to the "Entity Data Model Wizard" "Choose Your Data Connection" page.



Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

Northwind.mdf

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

○ No, exclude sensitive data from the connection string. I will set it in my application code.

○ Yes, include the sensitive data in the connection string.

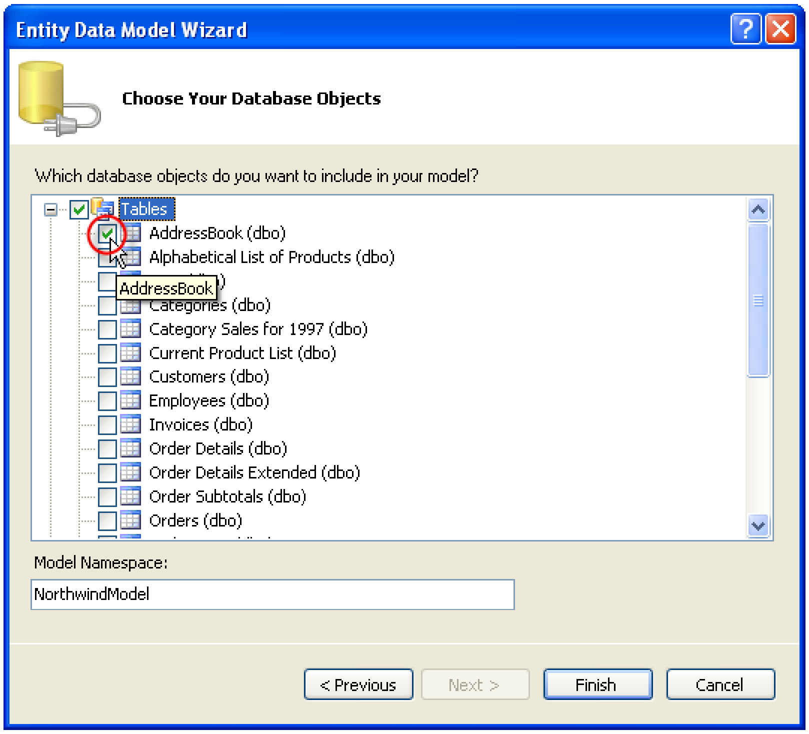
Entity connection string:

```
metadata=res://*/Northwind.csdl|res://*/Northwind.ssdl|res://*/Northwind.msl;provider=System
.Data.SqlClient;provider connection string='Data
Source=.\SQLEXPRESS;AttachDbFilename="C:\Program Files\Telerik\RadControls for Silverlight
Q2 2009 Old\Examples\DataSources\Northwind.mdf";Integrated Security=True;Connect
```

☑ Save entity connection settings in Web.Config as:

NorthwindEntities

5) In the "Choose Your Database Objects" page of the wizard, click the "AddressBook" table. Click the **Finish** button to close the dialog.



Entity Data Model Wizard

Choose Your Database Objects

Which database objects do you want to include in your model?

- ☑ Tables
  - ☑ AddressBook (dbo)
  - □ Alphabetical List of Products (dbo)
  - □ AddressBook)
  - □ Categories (dbo)
  - □ Category Sales for 1997 (dbo)
  - □ Current Product List (dbo)
  - □ Customers (dbo)
  - □ Employees (dbo)
  - □ Invoices (dbo)
  - □ Order Details (dbo)
  - □ Order Details Extended (dbo)
  - □ Order Subtotals (dbo)
  - □ Orders (dbo)

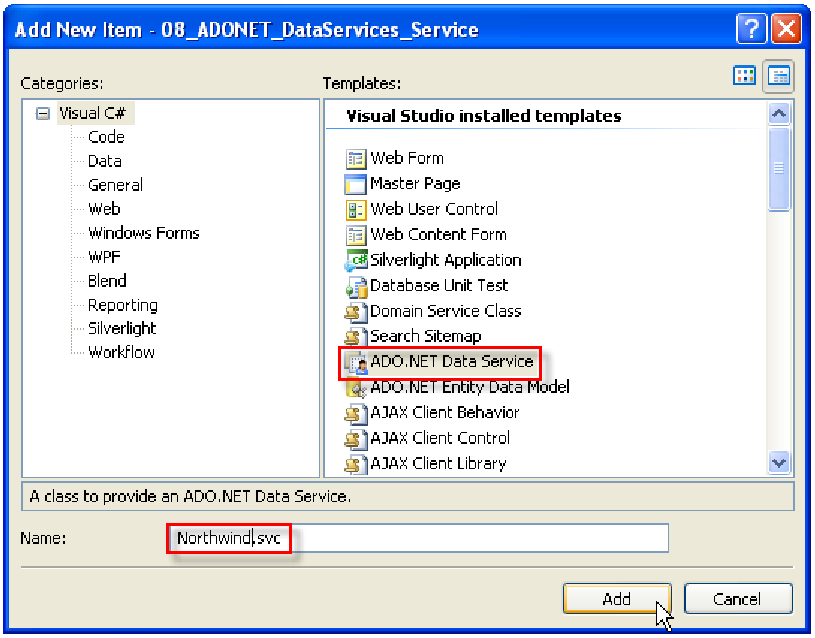
Model Namespace:

NorthwindModel

6) Build the project.

Create Service

1) From the Solution Explorer, right-click the project and select **Add > New Item...** from the context menu. Select the ADO.NET Data Service template, name it "NorthwindDataService.svc" and click the **Add** button to create the service. This step will open "NorthwindDataService.svc.cs" for editing.



A class to provide an ADO.NET Data Service.

Name: Northwind.svc

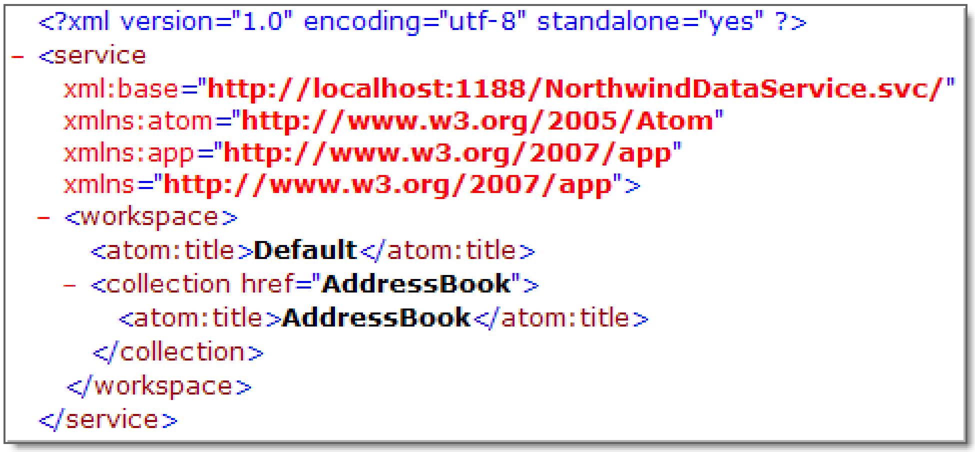
2) Change the NorthwindDataService class to the match the code below. "NorthwindDataService.svc.cs" has a "TODO" comment marked in the source to insert a type into the DataService<T> class declaration and another reminder to set access rules in the InitializeService() method. .

```
Public Class NorthwindDataService
  Inherits DataService(Of NorthwindEntities)
  Public Shared Sub InitializeService(ByVal config As IDataServiceConfiguration)
    config.SetEntitySetAccessRule("*", EntitySetRights.All)
  End Sub
End Class
```

```
public class NorthwindDataService : DataService<NorthwindEntities>
{
   public static void InitializeService(IDataServiceConfiguration config)
   {
      config.SetEntitySetAccessRule("*", EntitySetRights.All);
   }
}
```

3) Press **F5** to run the application.

4) The browser shows that the "AddressBook" collection is available.

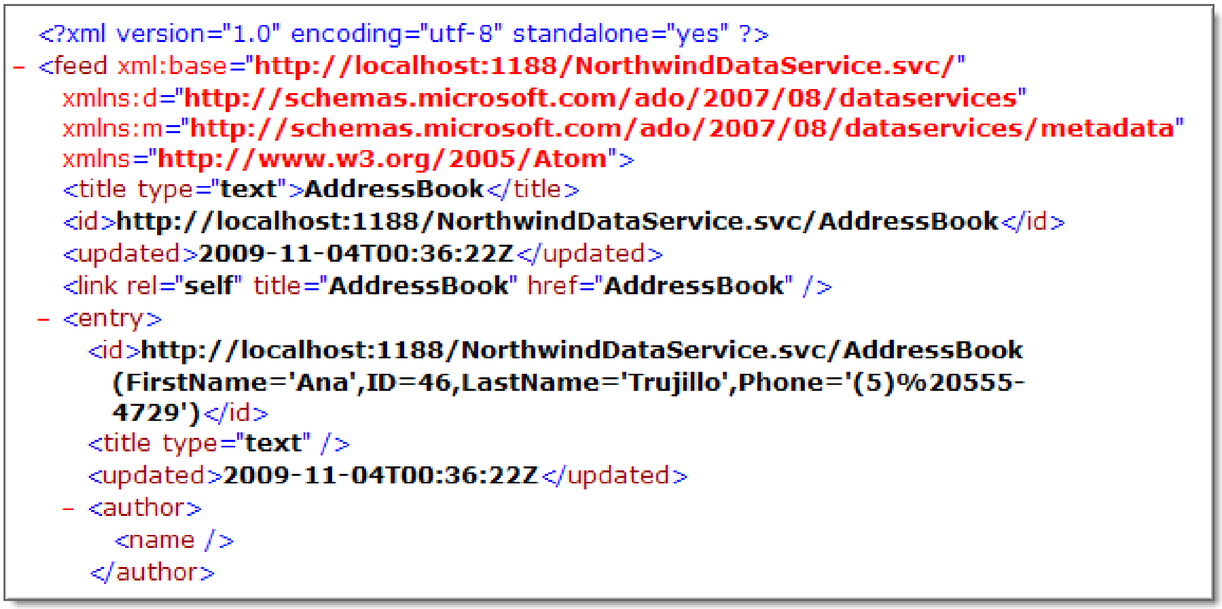


```
  <?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <service
    xml:base="http://localhost:1188/NorthwindDataService.svc/"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns:app="http://www.w3.org/2007/app"
    xmlns="http://www.w3.org/2007/app">
  - <workspace>
      <atom:title>Default</atom:title>
    - <collection href="AddressBook">
        <atom:title>AddressBook</atom:title>
      </collection>
    </workspace>
  </service>
```

Notes

The XML/ATOM above may display as a news feed. You can shut this off temporarily in Internet Explorer from **Tools > Internet Options > Content > Settings** and unselecting the "Turn on feed reading view" checkbox. Reopen the browser to display the XML.

5) Add "AddressBook" to the browser url to see the data:



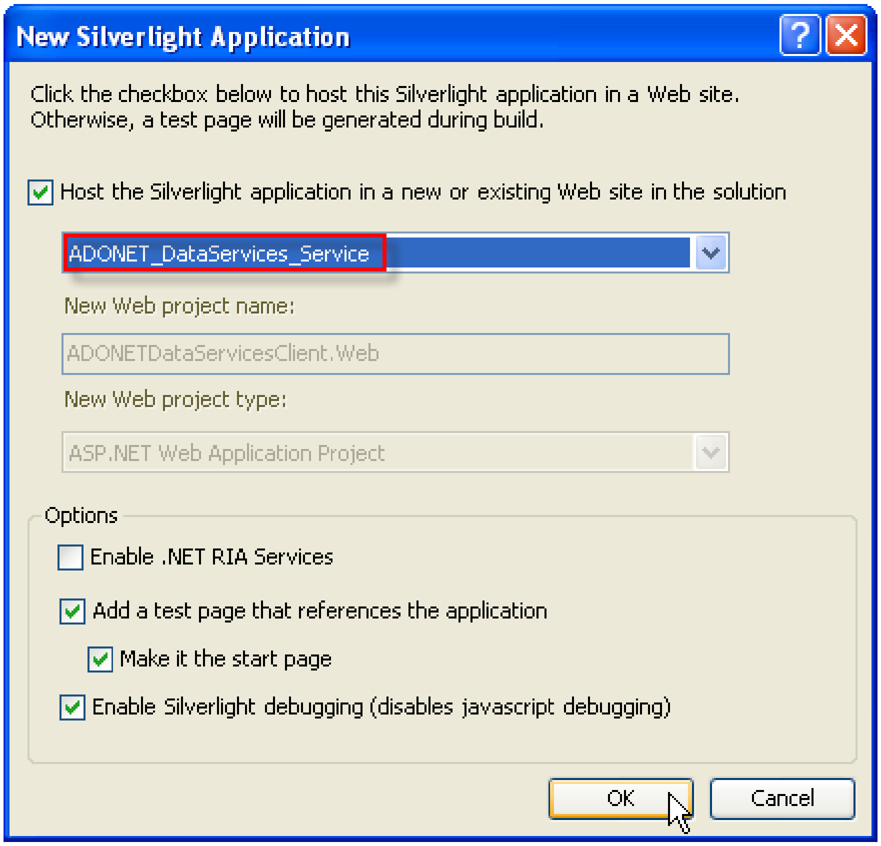
```
  <?xml version="1.0" encoding="utf-8" standalone="yes" ?>
- <feed xml:base="http://localhost:1188/NorthwindDataService.svc/"
    xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
    xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
    xmlns="http://www.w3.org/2005/Atom">
    <title type="text">AddressBook</title>
    <id>http://localhost:1188/NorthwindDataService.svc/AddressBook</id>
    <updated>2009-11-04T00:36:22Z</updated>
    <link rel="self" title="AddressBook" href="AddressBook" />
  - <entry>
      <id>http://localhost:1188/NorthwindDataService.svc/AddressBook
        (FirstName='Ana',ID=46,LastName='Trujillo',Phone='(5)%20555-
        4729')</id>
      <title type="text" />
      <updated>2009-11-04T00:36:22Z</updated>
    - <author>
        <name />
      </author>
```

4.6.5.2 Building the Client

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked. **Important note:** Select the data service ASP.NET application as the host. Currently the client and service must have the same domain and using the service application as the host accomplishes this. Click **OK** to close the dialog and create the project.



Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new or existing Web site in the solution

ADONET_DataServices_Service

New Web project name:

ADONETDataServicesClient.Web

New Web project type:

ASP.NET Web Application Project

Options

☐ Enable .NET RIA Services

☑ Add a test page that references the application

☑ Make it the start page

☑ Enable Silverlight debugging (disables javascript debugging)

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.GridView**

c) **Telerik.Windows.Data**

XAML Editing

1) Drag a RadGridView from the Toolbox to a point between the main "LayoutRoot" grid. Replace the RadGridView tag with the XAML below.

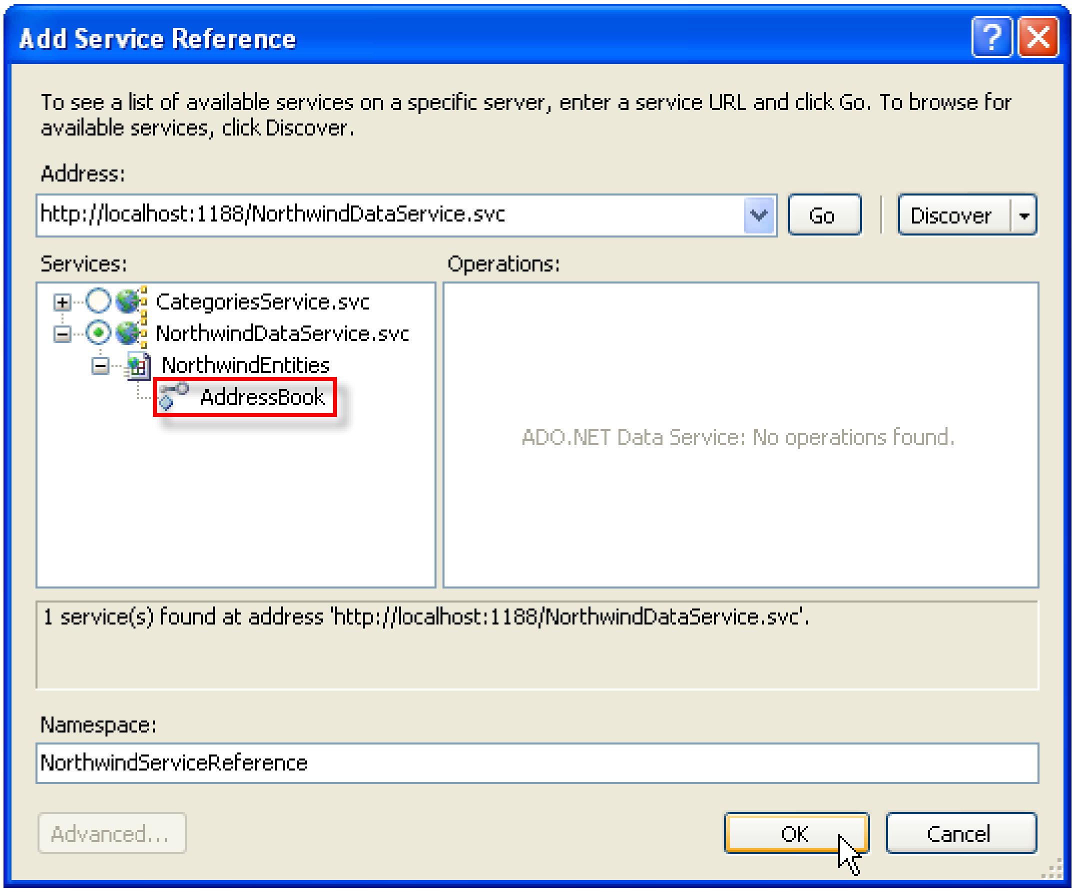
   The XAML will define the name of the grid as "gvAddressBook" so we can refer to it later in code behind. The columns are configured to fill the available width and column generation is turned off. Instead, "FirstName", "LastName" and "Phone" columns of the table are defined.

```
<telerik:RadGridView x:Name="gvAddressBook"
    ColumnWidth="SizeToHeader" AutoGenerateColumns="False">
  <telerik:RadGridView.Columns>
    <telerik:GridViewDataColumn Header="First"
        DataMemberPath="FirstName" />
    <telerik:GridViewDataColumn Header="Last"
        DataMemberPath="LastName" />
    <telerik:GridViewDataColumn Header="Phone"
        DataMemberPath="Phone" />
  </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

Reference The ADO.NET Data Service

1) In the Solution Explorer, right-click the References node and select **Add Service Reference...** This will display the "Add Service Reference" dialog.

2) In the "Add Service Reference" dialog, click the **Discover** button. The NorthwindDataService.svc server will display. Expand the node and select "AddressBook". Enter "NorthwindServiceReference" as the Namespace and click **OK** to create the client proxy.



Add Service Reference

To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address:

http://localhost:1188/NorthwindDataService.svc

Services:

CategoriesService.svc
NorthwindDataService.svc
NorthwindEntities
AddressBook

Operations:

ADO.NET Data Service: No operations found.

1 service(s) found at address 'http://localhost:1188/NorthwindDataService.svc'.

Namespace:

NorthwindServiceReference

Gotcha!

An exception that contains the phrase "Pattern constraint failed", can be caused by a leading underscore "_" character in the project name. Also know that projects beginning with numeric characters will also be automatically modified to include a leading underscore in the namespace. The short story is to avoid leading underscores because the parser will generate an exception.

Code Behind

1) Add a namespace reference for the proxy in the "Imports" (VB) or "using" (C#) section of code. This will be the name of your project + "." + "NorthwindServiceReference", i.e. "MyProject. NorthwindServiceReference".

2) Add the code below to the page constructor to query the data service. Be sure that you leave the call to InitializeComponent().

   First create a Uri that points at the service. You can use the HtmlPage.Document.DocumentUri to get the base address and then add the name of the service. Create an instance of NorthwindEntities, passing the Uri in the constructor. The NorthwindEntities instance will be our "context". The context contains "AddressBook" in the form of a DataServiceQuery that can be executed asynchronously to return data. Call the DataServiceQuery BeginExecute() method and pass the name of a handler (to be defined next) and the DataServiceQuery instance itself.

```
Public Sub New()
  InitializeComponent()

  Dim uri As New Uri( _
System.Windows.Browser.HtmlPage.Document.DocumentUri, "NorthwindDataService.svc")
  Dim context As New NorthwindEntities(uri)

  Dim query As DataServiceQuery(Of AddressBook) = context.AddressBook
  query.BeginExecute(Query_Completed, query)
End Sub
```

```
public MainPage()
{
   InitializeComponent();

   Uri uri = new Uri(System.Windows.Browser.HtmlPage.Document.DocumentUri,
      "NorthwindDataService.svc");
   NorthwindEntities context = new NorthwindEntities(uri);

   DataServiceQuery<AddressBook> query = context.AddressBook;
   query.BeginExecute(Query_Completed, query);
}
```

Gotcha!

Attempting to simply access the data, e.g.

```
gvAddressBook.ItemsSource = context.AddressBook.ToList();
```

...will generate a NotSupportedException with message "Specified method is not supported."

3) Handle the DataServiceQuery execution result event. *The signature of the event handler expects an* ***IAsyncResult*** *"result" parameter. The* ***AsyncState*** *property of the result is the DataServiceQuery instance you passed into BeginExecute(). Cast the AsyncState back to its original DataServiceQuery type so that you can call the* ***EndExecute()*** *method, passing the IAsyncResult as the parameter. EndExecute() returns an IEnumerable of AddressBook. Call the ToList() function on the collection and assign it to the ItemsSource of the grid.*

```
Private Sub Query_Completed(ByVal result As IAsyncResult)
  Dim query = TryCast(result.AsyncState, DataServiceQuery(Of AddressBook))
  Dim addressBook As IEnumerable(Of AddressBook) = query.EndExecute(result)
  gvAddressBook.ItemsSource = addressBook.ToList()
End Sub
```

```
private void Query_Completed(IAsyncResult result)
{
   var query = result.AsyncState as DataServiceQuery<AddressBook>;
   IEnumerable<AddressBook> addressBook =
      query.EndExecute(result);
   gvAddressBook.ItemsSource = addressBook.ToList();
}
```

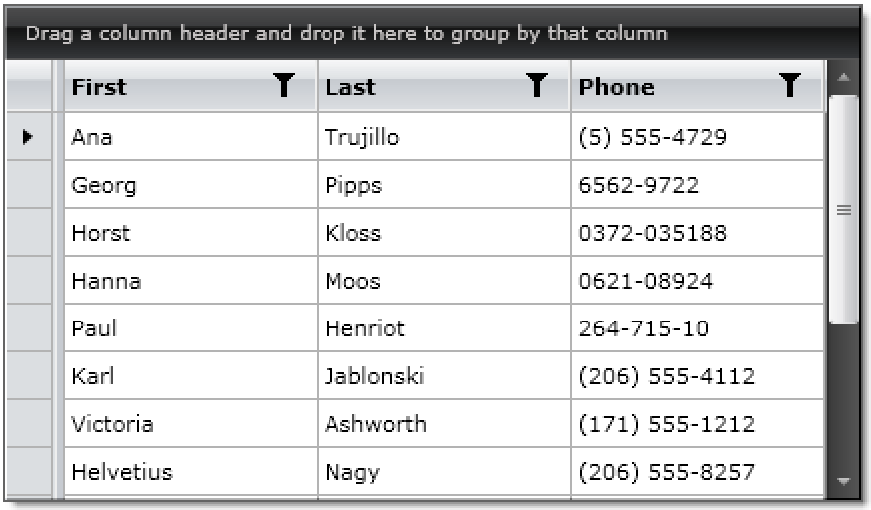
Gotcha!

If you don't include the call ToList(), you may get the error "Only a single enumeration is supported by this IEnumerable". A Microsoft engineer's take on this from their forums:

> "In the current implementation, the result from a query can only be enumerated once. This is because the underlying feed is not saved locally, and once we have gone through it once, we save all entities in the context . . ."

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Drag a column header and drop it here to group by that column

First	Last	Phone
Ana	Trujillo	(5) 555-4729
Georg	Pipps	6562-9722
Horst	Kloss	0372-035188
Hanna	Moos	0621-08924
Paul	Henriot	264-715-10
Karl	Jablonski	(206) 555-4112
Victoria	Ashworth	(171) 555-1212
Helvetius	Nagy	(206) 555-8257

4.6.6 OpenAccess

OpenAccess is a Object Relational Mapping technology from Telerik. OpenAccess is used to generate objects that map to database tables and have data access tasks taken care of for you automatically. We can use OpenAccess against databases such as SQL Server or together with other sources of data such as REST, WCF and ADO.NET Data Services. These latter services can be code intensive and prone to error if coded by hand. At the time of this writing, Telerik Labs has introduced the Data Services Wizard to handle creating OpenAccess infrastructure for REST, WCF and ADO.NET Data Services. At this time, the download is located at:

http://www.telerik.com/community/labs/telerik-data-services-wizard.aspx

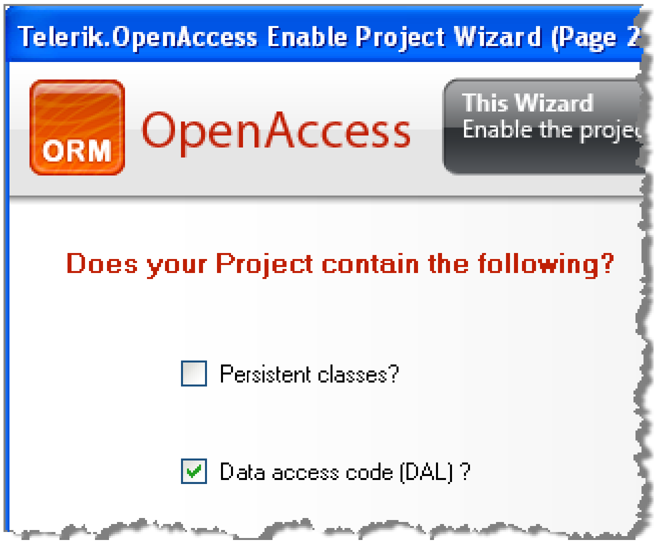
This walk through requires OpenAccess to be installed.

4.6.6.1 Building the Data Access Layer

1) In Visual Studio, create a new Class Library project to contain the Data Access Layer (DAL).

2) In Visual Studio, Right-click the class library project and select **OpenAccess > Enable Project** from the context menu.

3) Take the defaults and on the "Does your project contain the following" page of the OpenAccess Wizard, uncheck "Persistent Classes" and check the "Data Access code (DAL)" options.



Does your Project contain the following?

☐ Persistent classes?

☑ Data access code (DAL) ?

4) Define the connection in the wizard:

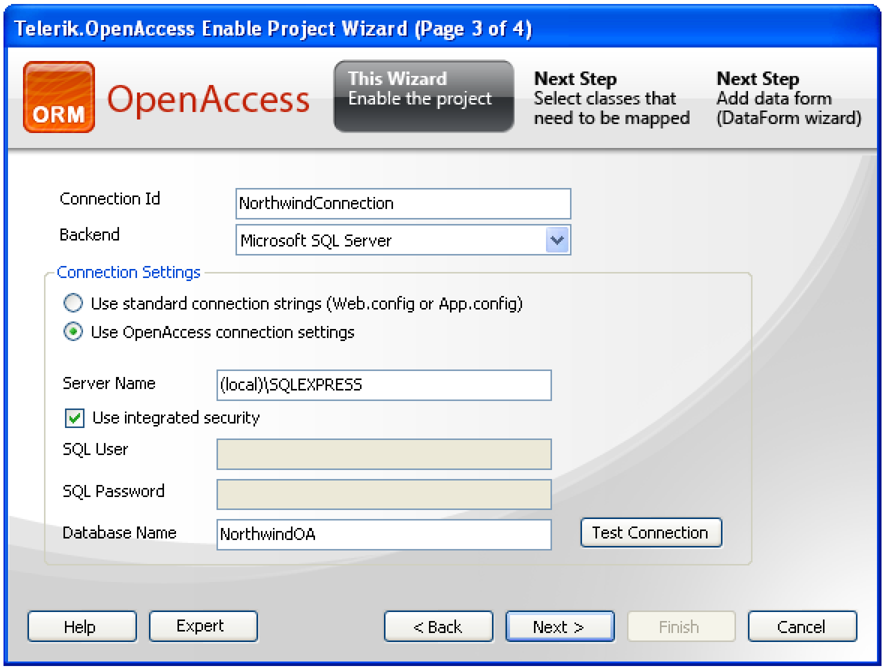
a) Connection Id = "NorthwindConnection"

b) Backend = "Microsoft SQL Server"

c) Select "Use OpenAccess Connection Settings" radio button option.

d) Server Name = "(local)\SQLEXPRESS" (assuming you installed the sample database to the default location)

e) Database Name = "NorthwindOA"



Telerik.OpenAccess Enable Project Wizard (Page 3 of 4)

Connection Id: NorthwindConnection

Backend: Microsoft SQL Server

Connection Settings

○ Use standard connection strings (Web.config or App.config)

● Use OpenAccess connection settings

Server Name: (local)\SQLEXPRESS

☑ Use integrated security

SQL User:

SQL Password:

Database Name: NorthwindOA

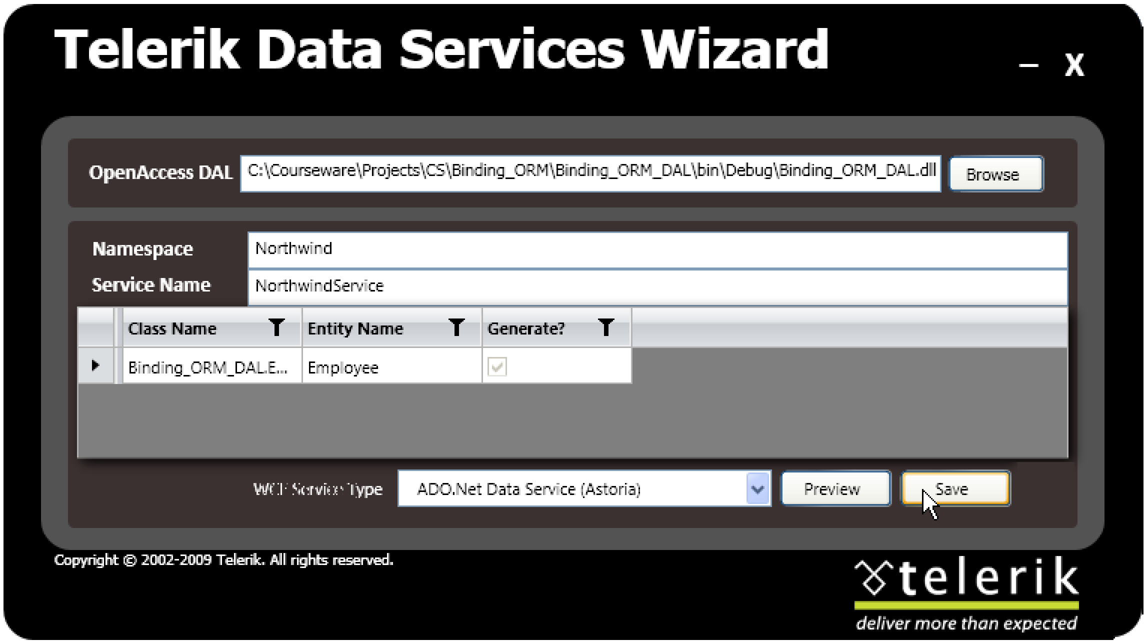
5) When you have finished ORM enabling the project using the wizard, open the Visual Studio menu and select **Telerik > Open Access > Reverse Mapping**. Select only the Employees table and ignore the other tables. Generate and save the configuration.

6) Build the DAL class library project.

4.6.6.2 Building the Service

1) Add a Silverlight Application to the solution and select the option to create a new web application to host the Silverlight client.
2) Using the **Telerik > Enable Project** option against the ASP.NET host application. Uncheck both "Persistent Classes" and "Data Access Code" options. In the connection portion of the wizard you can select "None" from the drop down list.
3) In the Solution Explorer, make sure there are references to the following:
   a) **The DAL class library project.**
   b) **Telerik.OpenAccess**
   c) **Telerik.OpenAccess.Query**
   d) **System.Data.Services**
   e) **System.Data.Services.Client**
   f) **System.ServiceModel**
4) Run the **Telerik Data Services Wizard** and generate your data context and data service classes:
   a) Click the **Browse** button for the "OpenAccess DAL". Locate and select the DAL assembly DLL.
   b) Enter a new Namespace "Northwind" and Service Name "NorthwindService"
   c) The server type should be "ADO.NET Data Service".

d) Click the **Save** button to generate the files. When the "Choose a Folder" dialog appears, locate the folder in the host web application.



Telerik Data Services Wizard

OpenAccess DAL: C:\Courseware\Projects\CS\Binding_ORM\Binding_ORM_DAL\bin\Debug\Binding_ORM_DAL.dll

Namespace: Northwind

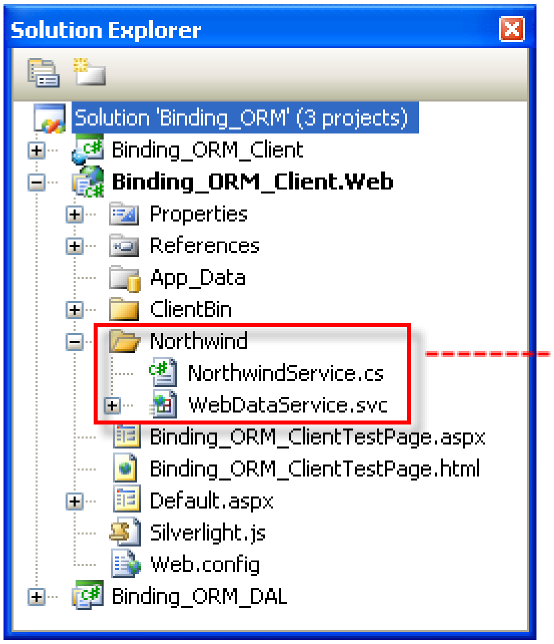
Service Name: NorthwindService

Class Name	Entity Name	Generate?
Binding_ORM_DAL.E...	Employee	☑

WCF Service Type: ADO.Net Data Service (Astoria)

Browse
Preview
Save
Copyright © 2002-2009 Telerik. All rights reserved.
telerik
deliver more than expected

The sample project in the Solution Explorer looks like the screenshot below. Notice the generated "Northwind" folder and that it contains "WebDataService.svc". We will refer to this file later in code.



Solution Explorer

```
Solution 'Binding_ORM' (3 projects)
  Binding_ORM_Client
  Binding_ORM_Client.Web
    Properties
    References
    App_Data
    ClientBin
    Northwind
      NorthwindService.cs
      WebDataService.svc
    Binding_ORM_ClientTestPage.aspx
    Binding_ORM_ClientTestPage.html
    Default.aspx
    Silverlight.js
    Web.config
  Binding_ORM_DAL
```

Generated by the Data Services Wizard

4.6.6.3 Building the Silverlight Client

1) In the Solution Explorer, create a reference to the service (See the ADO.NET Data Services walk through for more information on how to do this).

2) In the Solution Explorer, add references to the following assemblies:

   a) **Telerik.Windows.Controls**

   b) **Telerik.Windows.Controls.GridView**

   c) **Telerik.Windows.Data**

   d) **Telerik.Windows.Themes.Windows7** (optional)

3) Open MainPage.xaml for editing.

4) Drag a RadGridView from the Toolbox to the a point inside the main "LayoutRoot" grid element. Replace the RadGridView element with the XAML below. *The "x:Name" is defined so we can refer to the grid in code for binding. The other top level properties are optional and deal with the appearance of the grid. AutoGenerateColumns is set to false and two columns are defined for the first and last names in the Employee table.*

```
<telerik:RadGridView x:Name="gridEmployees"
      telerik:StyleManager.Theme="Windows7"
      ShowGroupPanel="False" AutoGenerateColumns="False"
      GridLinesVisibility="None">
   <telerik:RadGridView.Columns>
      <telerik:GridViewDataColumn Header="First"
            DataMemberBinding="{Binding FirstName}" />
      <telerik:GridViewDataColumn Header="Last"
            DataMemberBinding="{Binding LastName}" />
   </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

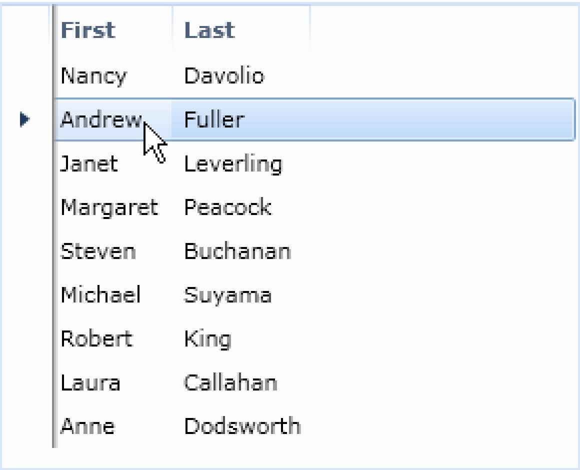
5) In the code behind, use the code below to replace the constructor and add a handler for the result of the data service query.

```
Public Sub New()
  ' leave this method call in place!
  InitializeComponent()

  ' create a Uri to the data service file
  Dim uri As New Uri(System.Windows.Browser.HtmlPage.Document.DocumentUri, _
"Northwind/WebDataService.svc")
  ' instantiate the data service
  Dim context As New NorthwindService(uri)
  ' create a DataServiceQuery against employee
  Dim query = TryCast(( _
        From employee In context.Employees _
        Select employee), DataServiceQuery(Of Employee))
  ' execute asynchronously
  query.BeginExecute(AddressOf Query_Completed, query)
End Sub

Private Sub Query_Completed(ByVal result As IAsyncResult)
  ' get a reference to the DataServiceQuery
  Dim query = TryCast(result.AsyncState, DataServiceQuery(Of Employee))
  ' retrieve the data
  gridEmployees.ItemsSource = query.EndExecute(result).ToList()
End Sub
```

```
public MainPage()
{
  // leave this method call in place!
  InitializeComponent();

  // create a Uri to the data service file
  Uri uri =
    new Uri(System.Windows.Browser.HtmlPage.Document.DocumentUri,
    "Northwind/WebDataService.svc");
  // instantiate the data service
  NorthwindService context = new NorthwindService(uri);
  // create a DataServiceQuery against employee
  var query =
    (from employee in context.Employees
     select employee) as DataServiceQuery<Employee>;
  // execute asynchronously
  query.BeginExecute(Query_Completed, query);
}

private void Query_Completed(IAsyncResult result)
{
  // get a reference to the DataServiceQuery
  var query = result.AsyncState as DataServiceQuery<Employee>;
  // retrieve the data
  gridEmployees.ItemsSource =
    query.EndExecute(result).ToList();
}
```

Run the Application

Press **F5** to run the application.



First	Last
Nancy	Davolio
Andrew	Fuller
Janet	Leverling
Margaret	Peacock
Steven	Buchanan
Michael	Suyama
Robert	King
Laura	Callahan
Anne	Dodsworth

4.7 Wrap Up

In this chapter you learned the basics of binding objects and collections of objects to Silverlight elements. You learned some of the syntax variations used to bind declaratively and how to bind elements using code alone. You also learned how to detect changes in property data and changes to collections.

We looked at how elements are bound in two common templates: DataTemplate and HierarchicalDataTemplate.

You learned how to bind data from a variety of popular data sources including straight XML and variations of XML, i.e. ATOM and RSS. In the process you used a combination of XDocument and LINQ to read, parse and convert the data to a form suitable for binding. Building off the XML techniques we discussed REST services. Then you worked through examples of WCF RIA services, WCF and ADO.NET Data Services.

Part V

Expression Blend

5 Expression Blend

5.1 Objectives

In this chapter you will apply your general Silverlight knowledge to the Expression Blend environment. First we will tour project types available in Expression Blend, then delve deeper into elements of the Expression Blend environment, becoming familiar with the major functional areas of the application. You will learn how Expression Blend organizes and manipulates resources. You will use Expression Blend to create and edit styles and also restyle a RadControl. You will use Expression Blend to edit a control template. You will see how Expression Blend creates animation and how animation can be triggered in code and through state changes. Finally, you will learn how to interactively bind data to elements.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Blend\Animation

\Courseware\Projects\<CS|VB>\Blend\Binding

\Courseware\Projects\<CS|VB>\Blend\Resources

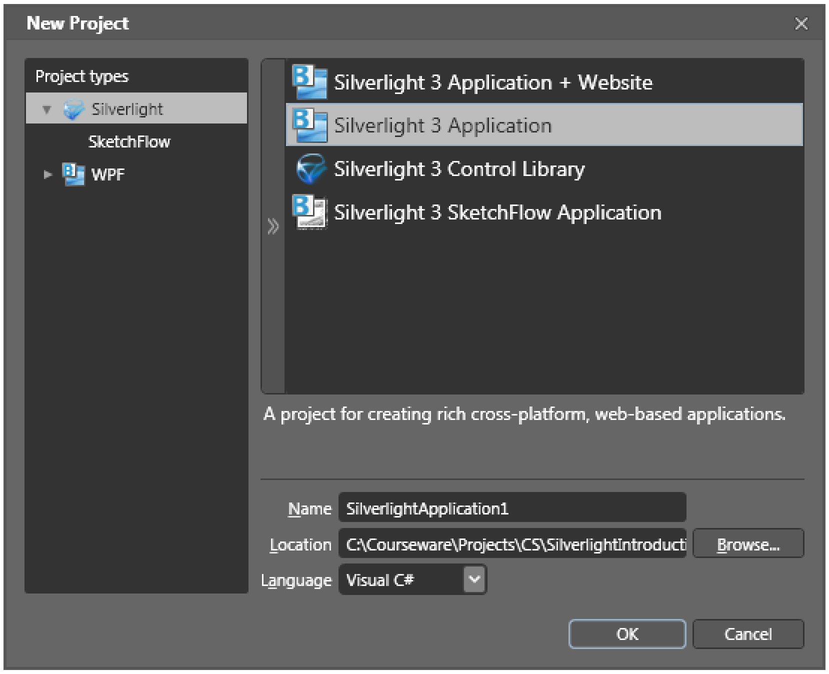
5.2 Overview

Expression Blend is an interactive, WYSIWYG design tool that helps you create compelling graphical web and desktop applications. Expression Blend is designed to work well with Visual Studio to allow smooth coordination between designers and developers. Both environments use the same design file format (XAML), solution and project format. Expression Blend is part of the Microsoft Expression suite of tools that also includes a website designer, a website compatibility preview, a vector graphics editor and applications to manage and encode digital media.

5.2.1 Expression Blend Project Types

Currently in Expression Blend you can create the following project types:

- Silverlight Application alone or with a hosting web site.
- Silverlight Control Library to contain custom controls.
- Silverlight SketchFlow applications allow you to rapidly evolve applications from prototype to production.
- WPF application, control library or SketchFlow application.


New Project
Project types
Silverlight
SketchFlow
WPF
Silverlight 3 Application + Website
Silverlight 3 Application
Silverlight 3 Control Library
Silverlight 3 SketchFlow Application
Browse...
OK
Cancel

A project for creating rich cross-platform, web-based applications.

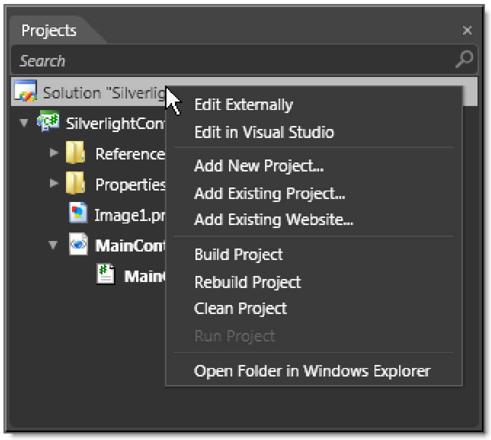
Name SilverlightApplication1

Location C:\Courseware\Projects\CS\SilverlightIntroducti

Language Visual C#

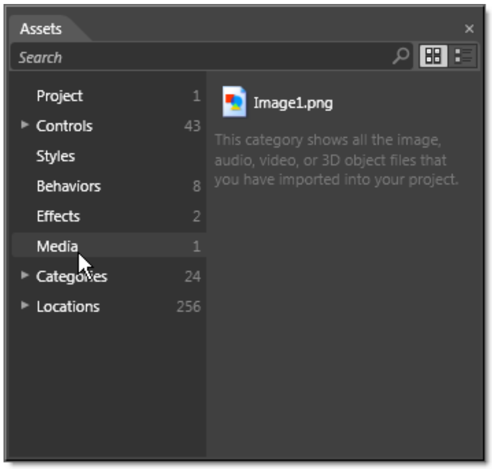
5.2.2 The Expression Blend Environment

In the "Working with Silverlight" chapter you briefly explored the Expression Blend environment and became acquainted with some of the key areas such as the Artboard, Properties pane and the Objects and Timeline pane. This section takes a deeper dive into the Expression Blend environment.



Projects pane

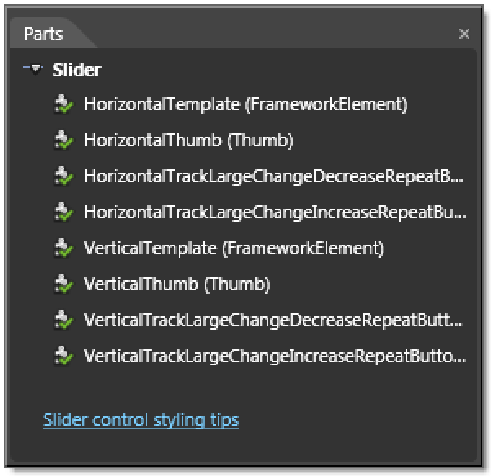
The Projects pane is very much like the Visual Studio Solution Explorer that organizes solutions, projects and items. Right-clicking a node in the project tree displays a context menu that has many of the same options as its Solution Explorer counterpart. The "Edit Externally" option opens the solution or project in the associated application for the file type.



This category shows all the image, audio, video, or 3D object files that you have imported into your project.

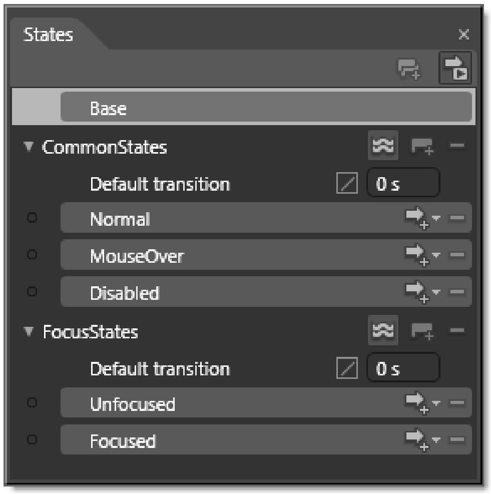
Assets pane

The Assets pane is like a standard toolbox, but contains more than just controls and has a search capability that makes it easy to find items when there are large numbers of possibilities. A category tree on the left hand side of the Assets pane pane also makes it easier to locate specific types of items. For example "Project" shows assets that are defined in the current project and "Locations" show all assets for a selected DLL


Parts
Slider
HorizontalTemplate (FrameworkElement)
HorizontalThumb (Thumb)
HorizontalTrackLargeChangeDecreaseRepeatB...
HorizontalTrackLargeChangeIncreaseRepeatBu...
VerticalTemplate (FrameworkElement)
VerticalThumb (Thumb)
VerticalTrackLargeChangeDecreaseRepeatButt...
VerticalTrackLargeChangeIncreaseRepeatButto...
Slider control styling tips

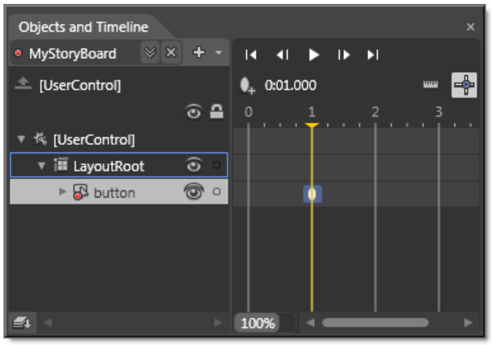
Parts pane

Expression Blend supports the "Parts and States" model, a pattern that provides a contract for describing a control without locking down the visual design of the control. "Parts" in this model are named elements. The Parts pane lists all the named elements in a control. For example, a Slider control has named parts for its thumb, increase and decrease buttons in both vertical and horizontal orientations.



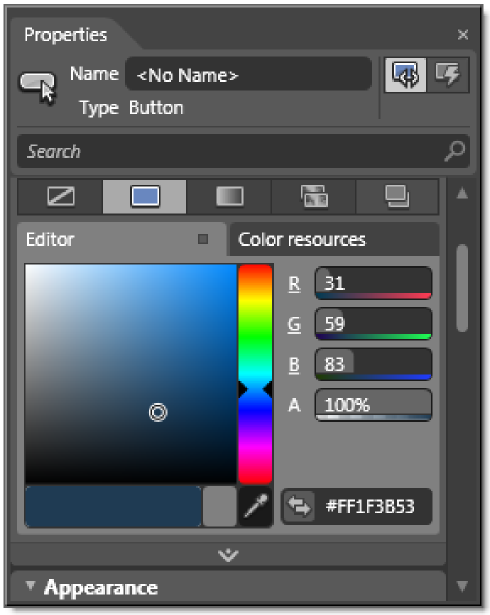
States pane

In the "Parts and States" model, "State" represents how a control looks in a logical state such as "MouseOver" or "Disabled". The States pane lists the possible states for a control, allows you to define transitions between states and also allows animation to be recorded that map to states.



Objects and Timeline pane

The Objects and Timeline pane displays the visual tree in the left side of the pane. You can drag controls, effects, behaviors and other assets to locations within the tree. This can be especially helpful if the Artboard has become two complicated to navigate easily. Element visibility can be toggled and elements can also be locked in place to prevent accidental rearranging. "StoryBoards" can be created to animate element properties.

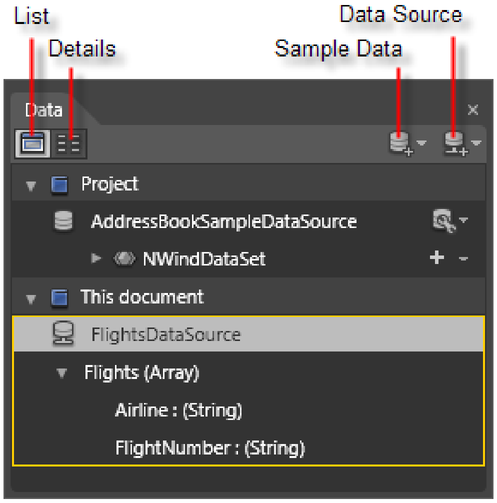


Name <No Name>
Type Button
R 31
G 59
B 83
A 100%
#FF1F3B53

Properties pane

The Properties pane is analogous to the Properties pane in Visual Studio, but is much more designer friendly. For example, brushes are represented with interactive editors that let you assign gradient brushes or use brushes from resources. Wherever possible, properties are represented with visual cues and tools.

The Properties pane is broken up into categories: Brushes, Appearance, Layout, Common Properties, Text, Transform, Miscellaneous.

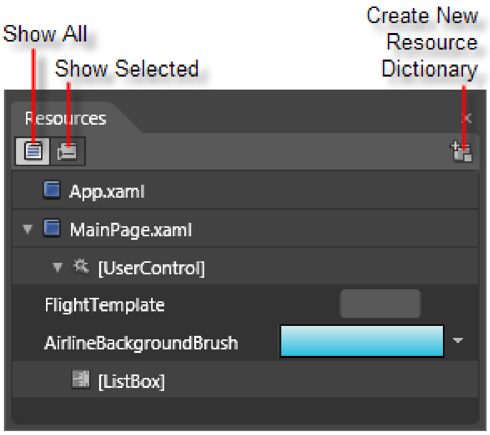


Data pane

The Data pane lets you define data sources and drag data to the design surface. Use the List button (upper left) to determine that dragged items will be interpreted as collections and will create a bound ListBox when dropped. Use the Details button when you want to drop individual items as a bound TextBlock.

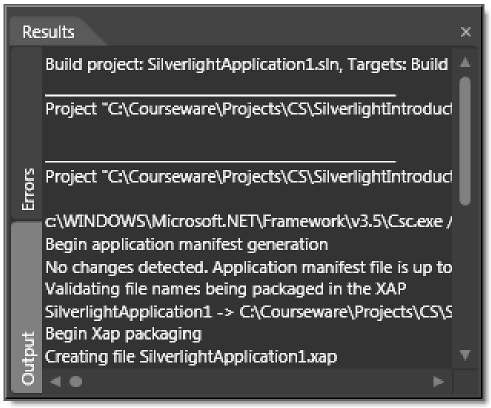
The Sample Data drop down lets you define or import "sample" data. The Data Source drop down lets you choose an existing object as the data source.

Data sources are automatically defined as resources for the project or the document (UserControl.Resources) only.



Resources pane

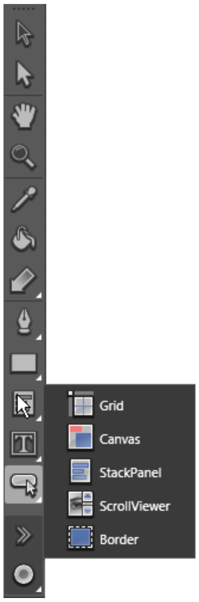
The Resources pane lists the templates, brushes and other resources in your application and in your user control. Brushes can be edited on the spot using the drop down list. Templates can be edited by clicking the button to the right of the template name.



```
Build project: SilverlightApplication1.sln, Targets: Build
___________________________________________
Project "C:\Courseware\Projects\CS\SilverlightIntroduc

___________________________________________
Project "C:\Courseware\Projects\CS\SilverlightIntroduc

c:\WINDOWS\Microsoft.NET\Framework\v3.5\Csc.exe /
Begin application manifest generation
No changes detected. Application manifest file is up to
Validating file names being packaged in the XAP
SilverlightApplication1 -> C:\Courseware\Projects\CS\S
Begin Xap packaging
Creating file SilverlightApplication1.xap
```

Results pane

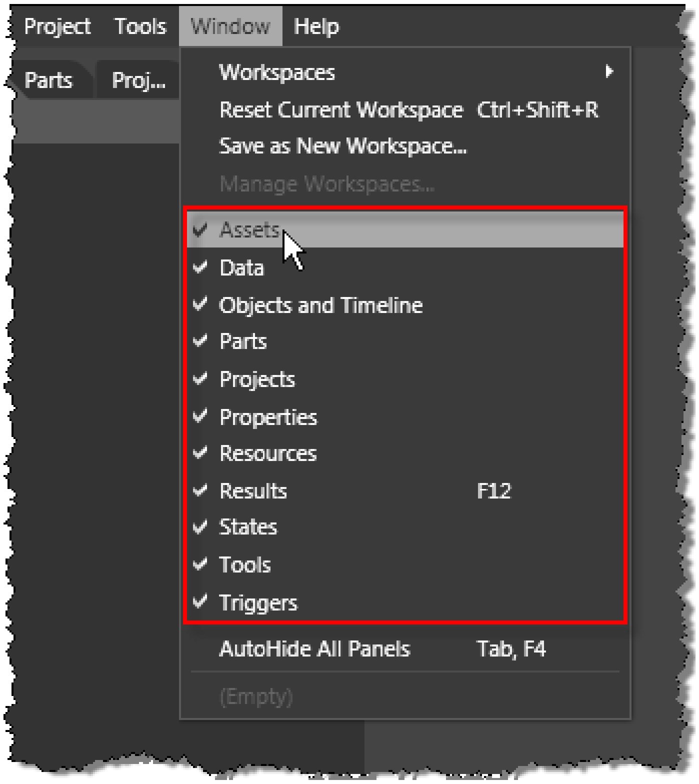
The Results pane combines the Output and the Errors windows.



Tools pane

The Tools pane provides quick access to the key tools you'll need for working in the Artboard without having to dig too deep into a menu. Look for the small arrow in the lower right of some icons that can be right-clicked to show a sub-menu. The screenshot shows all the available layout panels.

If one of the panes is not visible, use the main Expression Blend menu "Window" item to make the pane visible again.

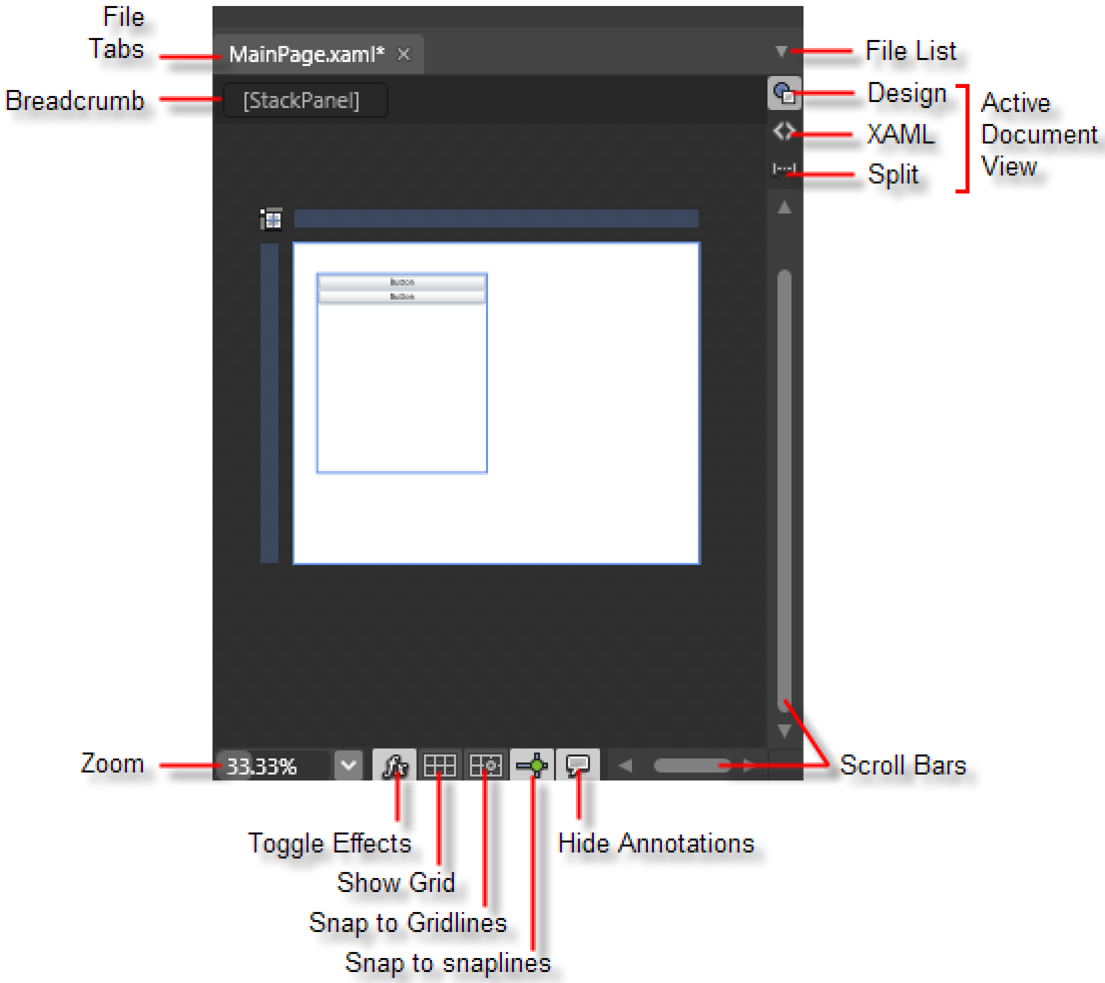

Project
Tools
Window
Help
Parts
Proj...
Workspaces
Reset Current Workspace Ctrl+Shift+R
Save as New Workspace...
Manage Workspaces...
✓ Assets
✓ Data
✓ Objects and Timeline
✓ Parts
✓ Projects
✓ Properties
✓ Resources
✓ Results F12
✓ States
✓ Tools
✓ Triggers
AutoHide All Panels Tab, F4
(Empty)

Artboard

The application's visual interface is described using the Artboard. One or more files can be edited and these files show as tabs across the top of the Artboard. Below these tabs is a breadcrumb bar that is synchronized with the selected element in the Objects and Timeline pane. The selected element may have a drop down list to enable template editing.

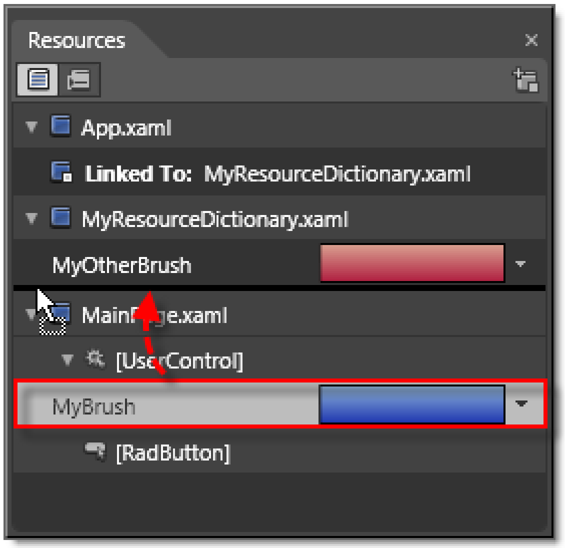
To the right of the file tabs, a drop down arrow displays a list of all files being edited currently. Below that are three buttons that determine the active document. By default you first see a pure design view. You can view the document as XAML only or split the view between design and XAML.

Below the design surface are controls that fine tune the interface. The Zoom control lets you enter a zoom percentage directly, select from a list of values or choose "Fit to Selection". The set of buttons next to zoom let you toggle rendering of effects, grid visibility, snap-to-grid, snap-to-snapline and annotations.



5.3 Resources

Resources are populated automatically by Expression Blend when you create new styles, templates and data sources. The Resources pane lets you drag a resource between locations: the App.xaml file, the UserControl.Resources element of MainPage.xaml or any other resource dictionaries that you define.



Resources

App.xaml

Linked To: MyResourceDictionary.xaml

MyResourceDictionary.xaml

MyOtherBrush

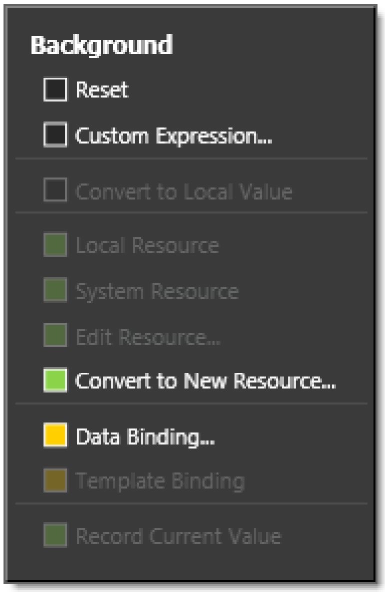
MainPage.xaml

[UserControl]

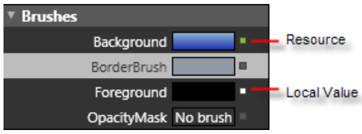
MyBrush

[RadButton]

You can also use the Properties pane Advanced Property Options button to move properties between locations, for example from "local", i.e. inside the element definition, to UserControl.Resources. Advanced Property Options button also lets you take a property and **Convert to New Resource.** This places the property into the resource element of the Application, the User Control or into a new resource dictionary. The screenshot below is the Advanced Property Options for a Background property.

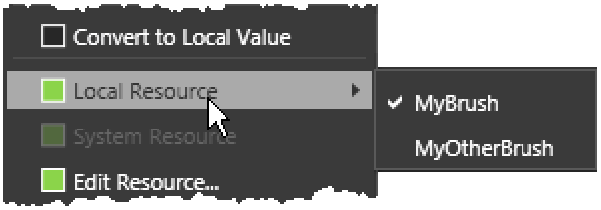


Notes

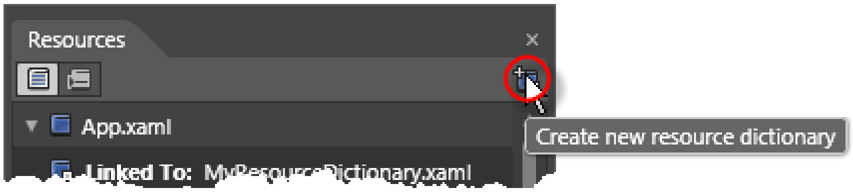


Notice the color key in the advanced properties menu. When you select an option such as a local resource, the Advanced Property Options button displays in the corresponding color. Here the Background has been assigned a resource and displays in green.

Once a resource has been created, it can be applied to other elements by using the Advanced Property Options button and selecting from the Local Resources list:



For larger production applications you will want to divide your resources into separate resource dictionaries. You can do this in the Resources pane using the "Create new resource dictionary" button. Resource dictionaries created this way are automatically linked into the App.xaml file and therefore available to your entire application. Right-click the App.xaml node to Link or Unlink a resource dictionary.


Resources
App.xaml
Linked To: MyResourceDictionary.xaml
Create new resource dictionary

5.4 Restyling RadControls for Silverlight

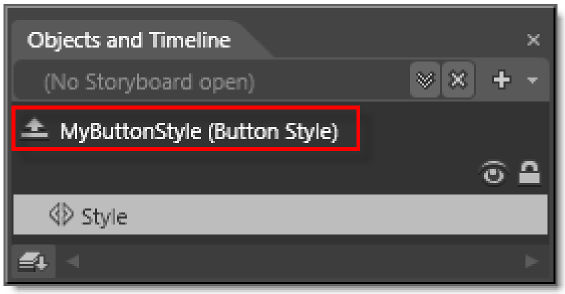
Creating and Editing Styles

Styles can be created interactively in Expression Blend and reused by applying to other elements. To copy a style from existing properties, first, select an element in the Artboard. In this example the element is a Button. In the menu, select **Object > Edit Style > Create Empty**. Note: you can also choose **Edit a Copy** if you want all the current settings copied to the new style). In the "Create Style Resource" dialog, give the style a descriptive name and decide what resource the style should be stored in (i.e. application, user control). The new style will show up in several places of the Expression Blend UI.

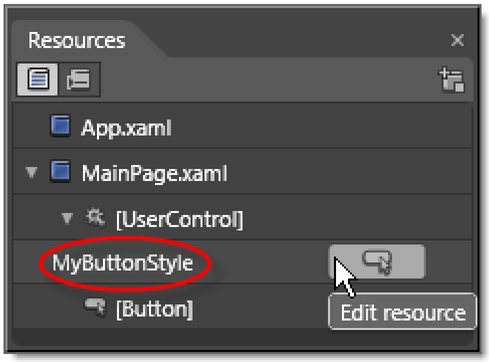
- The XAML for the page will now look like the example below. Notice that "MyButtonStyle" has been created automatically as a resource and hooked up to a Button control.

```
<UserControl.Resources>
  <Style x:Key="MyButtonStyle" TargetType="Button"/>
</UserControl.Resources>

<Grid x:Name="LayoutRoot" Background="White">
    <Button Margin="0" Content="Button" Style="{StaticResource MyButtonStyle}"/>
</Grid>
```

- The Objects and Timeline pane shows that we're currently editing "MyButtonStyle". Note: The Scope Up button lets you navigate back to editing the entire user control.



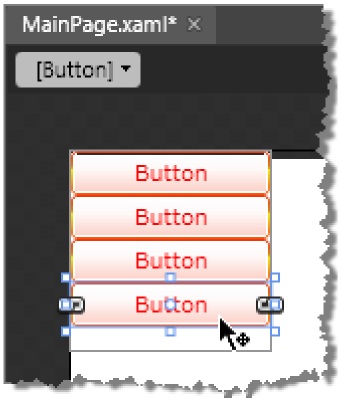
- The Resources pane lists the new "MyButtonStyle" and has a button to edit the resource.



Once we're editing the resource, you simply work directly in the Artboard and the changes become part of the style. For example, you could change the brush colors for the Button Background, BorderBrush and Foreground to some red, orange and yellow colors. Now the style reflects the changes made in the Artboard and can be seen in the XAML below.

```
<Style x:Key="MyButtonStyle" TargetType="Button">
  <Setter Property="Background" Value="#FFF63B14"/>
  <Setter Property="BorderBrush">
    <Setter.Value>
      <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
        <GradientStop Color="#FFEE330B" Offset="0"/>
        <GradientStop Color="#FFE28518" Offset="0.375"/>
        <GradientStop Color="#FFF6EC12" Offset="0.417"/>
        <GradientStop Color="#FFFE4B0A" Offset="1"/>
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
  <Setter Property="Foreground" Value="#FFFF0202"/>
</Style>
```

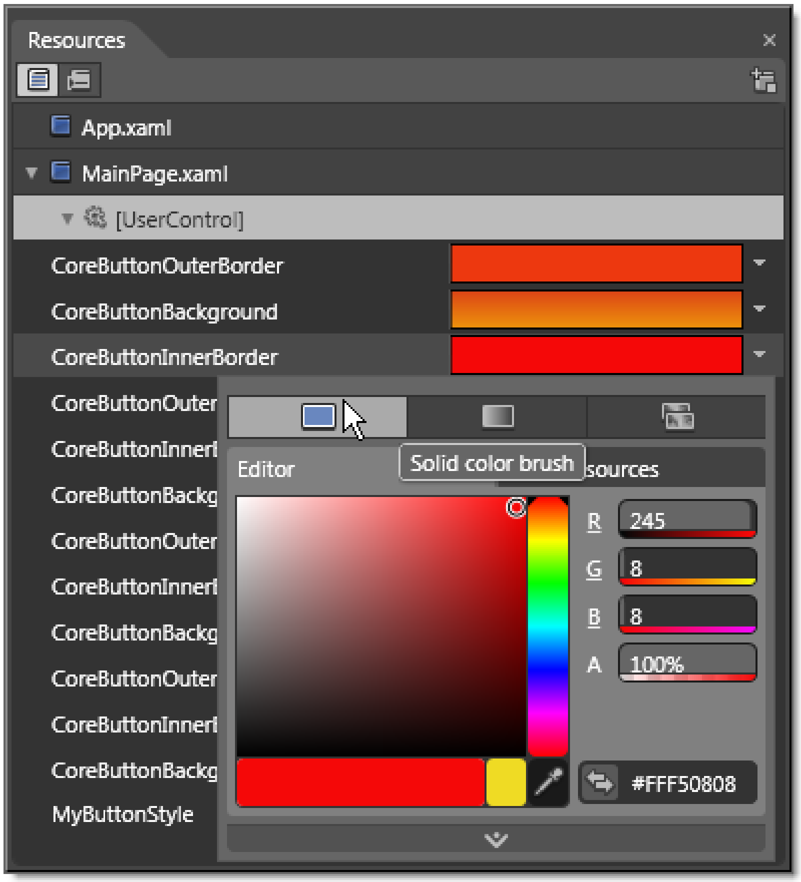
If there are other buttons on the page you can right-click them and choose **Edit Template > Apply Resource > "MyButtonStyle"** from the context menu. All the buttons will be styled exactly the same and take up very little room in the generated XAML.



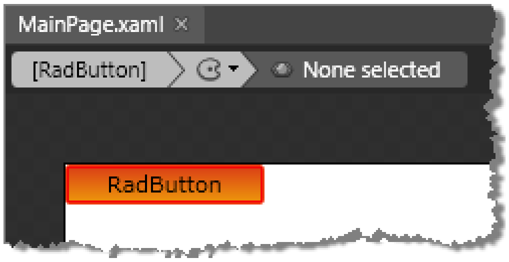
```
<StackPanel Height="100" HorizontalAlignment="Left" VerticalAlignment="Top" Width="100">
  <Button Content="Button" Style="{StaticResource MyButtonStyle}"/>
  <Button Content="Button" Style="{StaticResource MyButtonStyle}"/>
  <Button Content="Button" Style="{StaticResource MyButtonStyle}"/>
  <Button Content="Button" Style="{StaticResource MyButtonStyle}"/>
</StackPanel>
```

Styling a RadControl

Lets use a RadButton and select the **Edit a Copy** context menu option. Where before the XAML was easily viewed, the updated XAML will require scrolling through several screenfuls of styles, templates and brushes and storyboard animations. This doesn't present a problem because Expression Blend lets us work in a visual environment and forget about the XAML resource definitions. Looking at the Resources pane we can see that RadButton has a number brushes already defined. Each brush colors part of the control for a given state, e.g. "CoreButtonOuterBorder_Pressed". So we can edit colors right from the Resources pane without having to pinpoint specific parts or states of the control first.



Now the edited style for the button in the designer looks like the screenshot below.



5.5 Customizing RadControls Templates

You may need to add elements to a control or completely change the layout. Templates allow you "move the furniture around" without breaking the basic control functionality. This walk through demonstrates customizing a RadButton to include an image and a "shadow" image of the button content.

Project Setup

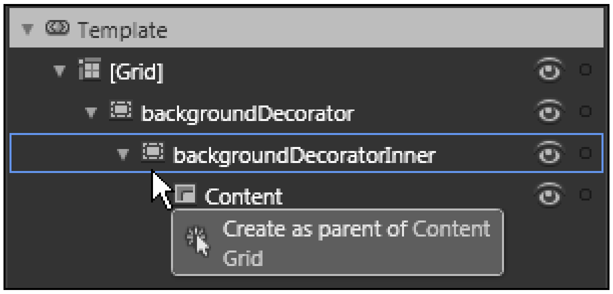
1) Run Expression Blend.
2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*
3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight Application" from the right-most list. Enter a unique name for the project and click **OK**.

Edit the Page in Expression Blend

1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.
2) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu. Add references to the following assemblies:
   a) **Telerik.Windows.Controls.dll**
   b) **Telerik.Windows.Controls.Input.dll**
3) In the Projects pane, right-click the project and select **Add New Folder** from the context menu. Name the new folder "Images".
4) Drag "search.png" to the "images" folder. This image file can be found in "\courseware\images" directory.
5) From the Project menu select **Build Project**.
6) Add a **RadButton** to the page.
   a) Open the Assets pane.
   b) On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.
   c) In the Assets pane, just above the tree view is the Assets Find entry text box.
   d) Type the first few characters of "RadButton" into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view.
   e) Locate the RadButton control and drag it onto the MainPage.xaml Artboard.
7) Set properties of the RadButton.
   a) **Layout > Width** = "200"
   b) **Layout > Height** = "100"
   c) **Common Properties > Content** = "Search Flights"
   d) **Text > Font** = "14pt"
8) Right-click the RadButton and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "MyButtonStyle". Click **OK** to create the style resource and close the dialog.

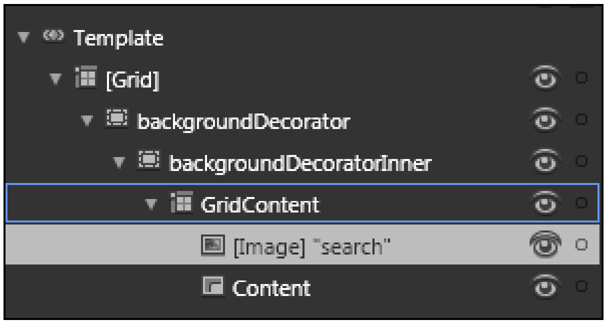
Edit the Control Template

1) Locate a Grid element from the Assets pane to a the "backgroundDecoratorInner" element. Watch for when the tool tip reads "Create as parent of Content". This will insert the Grid between "backgroundDecoratorInner" and the Content element.



2) Right-click the grid and select "Rename" from the context menu. Name the grid "GridContent".

3) From the **Assets pane > Media**, drag "search.png" inside the "GridContent". The tool tip will read "Create in GridContent". This should insert an Image control just before "Content". The screenshot below shows the state of the template at this point.



4) In the Properties pane, set properties of the Image.

a) **Appearance > Opacity** = "30%"

b) **Layout > MaxHeight** = "128"

c) **Layout > MaxWidth** = "128"

d) **Layout > HorizontalAlignment** = "Stretch"

e) **Layout > VerticalAlignment** = "Stretch"

f) If any **Layout > Margin** properties have been set, zero them.

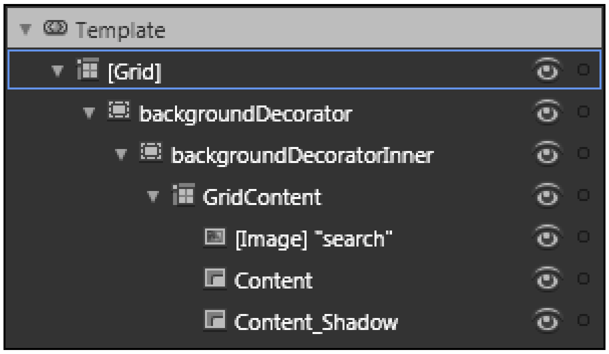
g) **Common Properties > Stretch** = "Uniform"

5) Make a copy of the "Content" element:

a) Right-click "Content" and select **Copy** from the context menu.

b) Select "GridContent", right-click and select **Paste** from the context menu.

c) Right-click, select **Rename** from the context menu and set the name to "Content_Shadow". The Objects and Timeline pane should look like the screenshot below.



6) Select the "Content_Shadow" element in the Objects and Timeline pane and set properties:

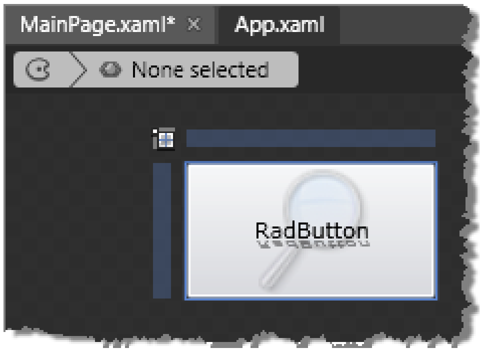
a) **Appearance > Opacity** = "30%"

b) **Transform > Scale > Y** = "-0.5"

c) **Transform > Skew > X** = "30"

d) **Transform > Center Point > Y** = "0.8"

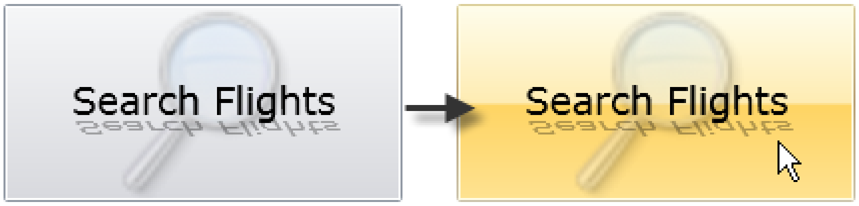
7) The template in the Artboard looks like this screenshot:



8) Press the "Up Scope" button until you reach the User Control.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



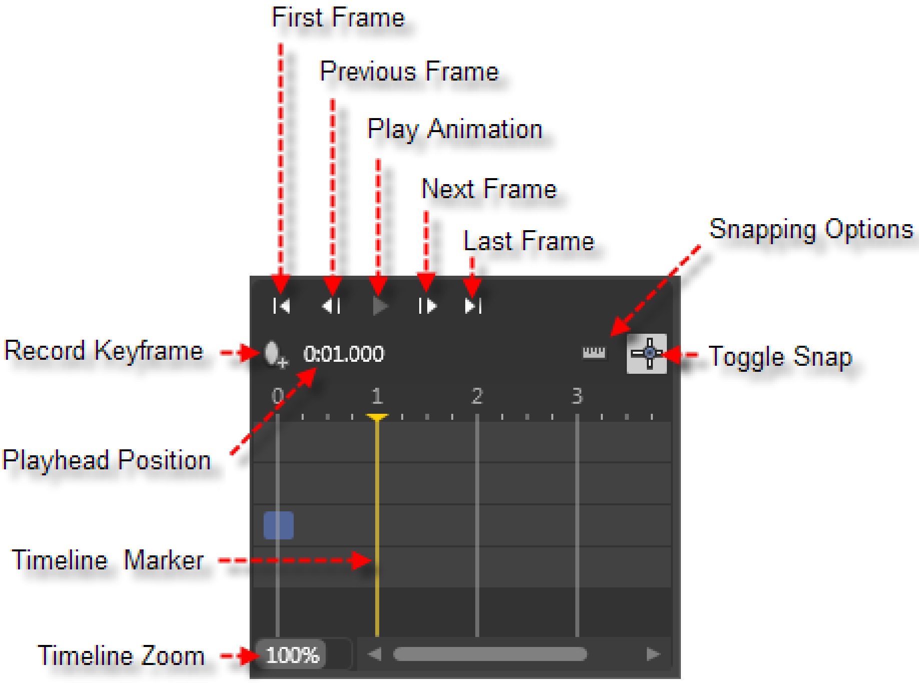
The screenshot shows the button in its normal state and with mouse over. The button has all of its previous capability and also includes the new "shadow" and image elements.

5.6 Bring RadControls to Life with Animations

Overview

Animation lets you control the transition between two property values. The property could be an element's location, size, brush color, nearly any property. Animations can be contained in a "storyboard" and triggered programmatically or can be triggered by a change in "state", e.g. the mouse passes over an element.

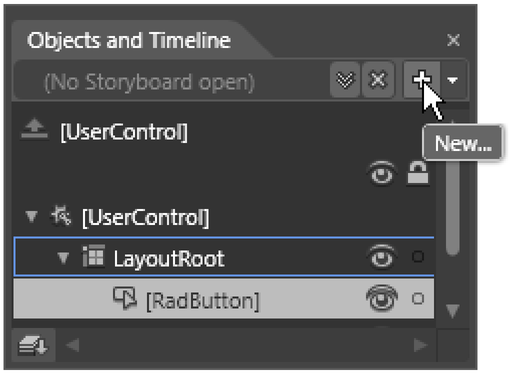
You work with animation in the Timeline portion of the Objects and Timeline pane. The main function of the Timeline is to record "Keyframe" objects that contain one or more properties for an element at a particular point in time. Each row in the Timeline corresponds to an element and each column is a "frame", i.e. a location in time. The Timeline marker indicates the frame that we're working with currently.



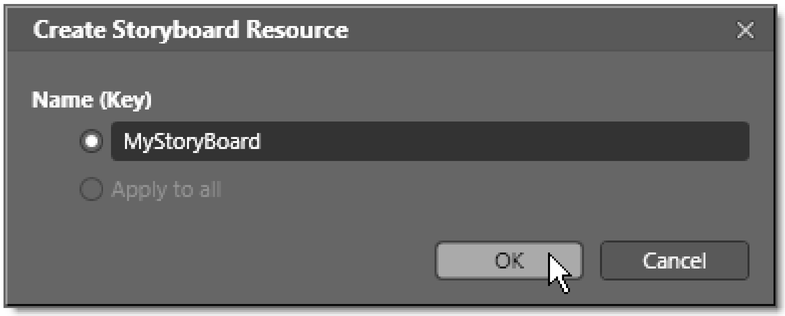
Clicking the Play Animation button snaps the Timeline marker to the start of the Timeline, where it begins to move steadily to the right, encountering Keyframes and changing properties along the way. When the animation is playing, you can watch the dynamically changing properties in the Artboard.

Creating a Simple Animation

Let's look at the steps to creating a simple animation using the Expression Blend designer that rotates a RadButton in place in response to the button's Click event. After the RadButton has been added to the Artboard we can create a new Storyboard in the Objects and Timeline pane:



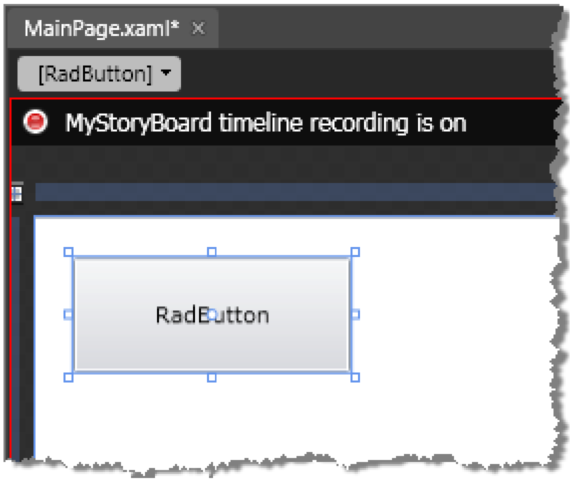
We provide a name in the Create Storyboard Resource dialog.



● MyStoryBoard

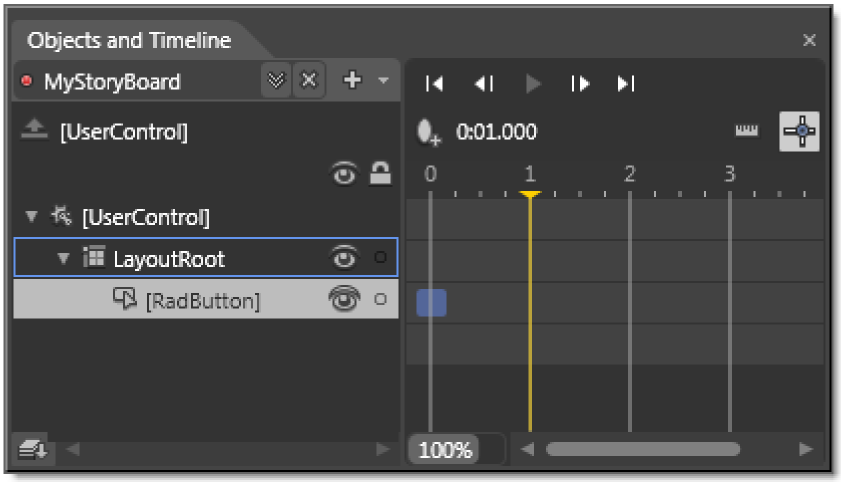
○ Apply to all

Once the storyboard is created, two things happen in the Expression Blend environment. The Artboard displays a message that indicates that property changes are being recorded:

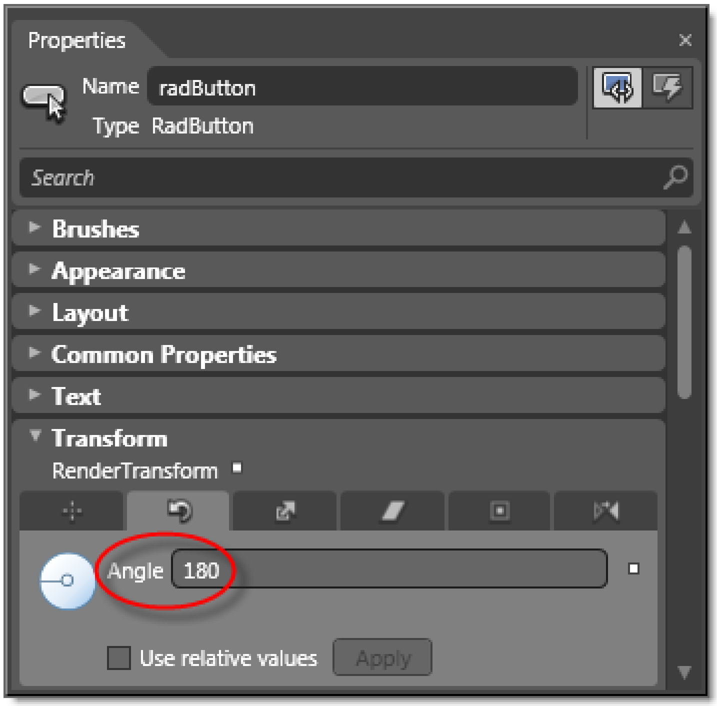


MyStoryBoard timeline recording is on

...and a Timeline has opened up in the Objects and Timeline pane. The Timeline will allow us to set property changes at specific times and to play those changes as if playing a video.

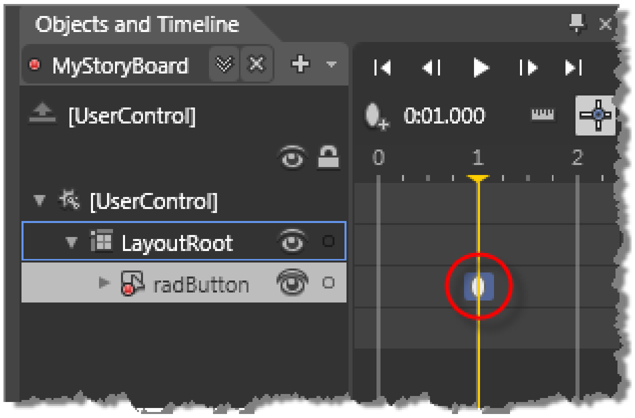


Now we can change property values of the RadButton and the changes will be recorded in the Keyframe for the current location in the Timeline. This example uses the Properties pane to change the **Transform > Rotate > Angle** property to "180".



Name radButton
Type RadButton
Angle 180

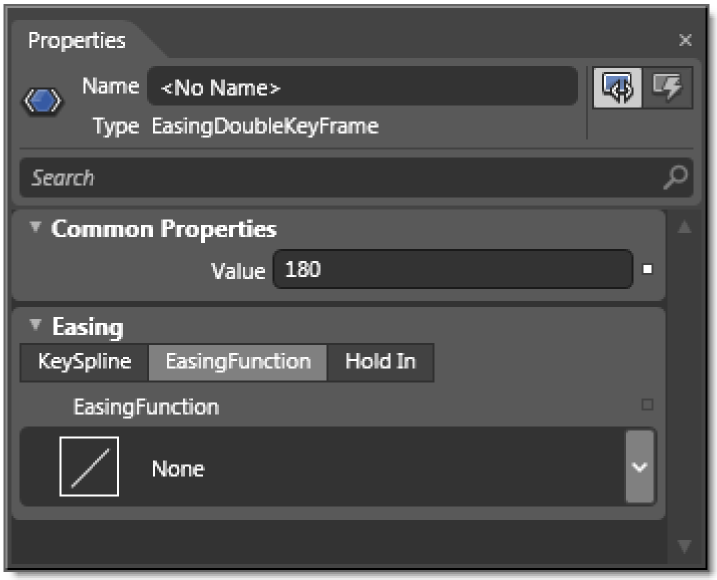
Looking back to the Objects and Timeline pane we can see a new Keyframe is created for the button at the Timeline marker location.



Tip!

If you don't want to change a property but instead want to use the current value of a property, use the Properties pane Advanced Property Options button and select the "Record Current Value" option if its enabled. If the button is not enabled, the property may not be supported for animation.

If you double-click the Keyframe, its properties will display in the Properties pane along with the Keyframes "Easing" settings. "Easing" is a path that the transition takes between two property values. For example, you may wish for the change to take place evenly as the timeline progresses or you might want the change to be nearly complete early on. You can use an EasingFunctions to establish a predefined easing profile such as "Sine" or "Elastic". The "Hold In" tab simply waits until the animation completes before changing the property value.

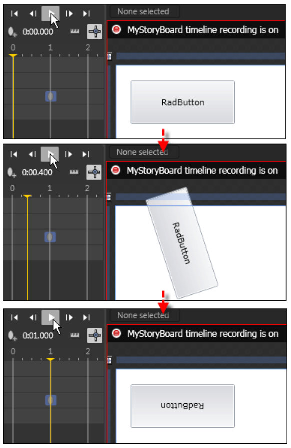

Properties

Name <No Name>

Type EasingDoubleKeyFrame

Value 180

EasingFunction None


None selected
0:00.000
MyStoryBoard timeline recording is on
RadButton
None selected
0:00.400
MyStoryBoard timeline recording is on
RadButton
None selected
0:01.000
MyStoryBoard timeline recording is on
RadButton

Clicking the Play button causes the RadButton to rotate at an even pace, ending up at 180 degrees.

You can initiate the animation in code using the Storyboard Begin() method.

```
Private Sub radButton_Click(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs)
  MyStoryBoard.Begin()
End Sub
```

```
private void radButton_Click(object sender, System.Windows.RoutedEventArgs e)
{
  MyStoryBoard.Begin();
}
```

Tip!

Click F6 in Expression Blend to switch between the "Design" and "Animation" workspaces. "Animation" arranges the workspace with maximum space available for the Timeline.

State Changes

Animations are often triggered by changes in control state such as a "mouse over" or "focused". RadControls have a great deal of animation built-in for state changes, right out-of-the-box. For example, RadMenu animates menu expansion and RadButton animates the MouseOver state. You can alter the existing animations or add your own. The following walk through demonstrates enlarging the content of a button when the mouse passes over.

1) Create a new Silverlight Application in Expression Blend.

2) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.

3) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu.

4) Add a reference to the **Telerik.Windows.Controls.dll** assembly.

5) From the Project menu select **Build Project**.

6) Add the RadButton to the page.

   a) Open the Assets pane.

   b) On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.

   c) In the Assets pane, just above the tree view is the Assets Find entry text box.

   d) Type the first few characters of RadButton into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view.

   e) Locate the **RadButton** control and drag it onto the MainPage.xaml Artboard.

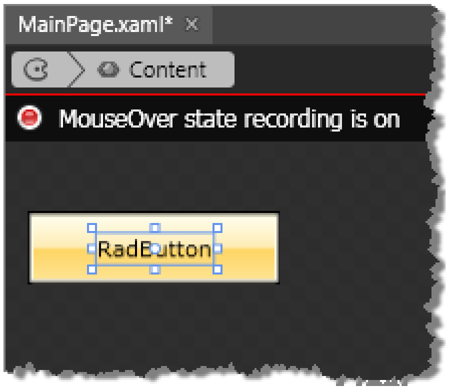
7) Set properties using the Properties pane:

   a) **Common Properties > Content** = "Click Me!"

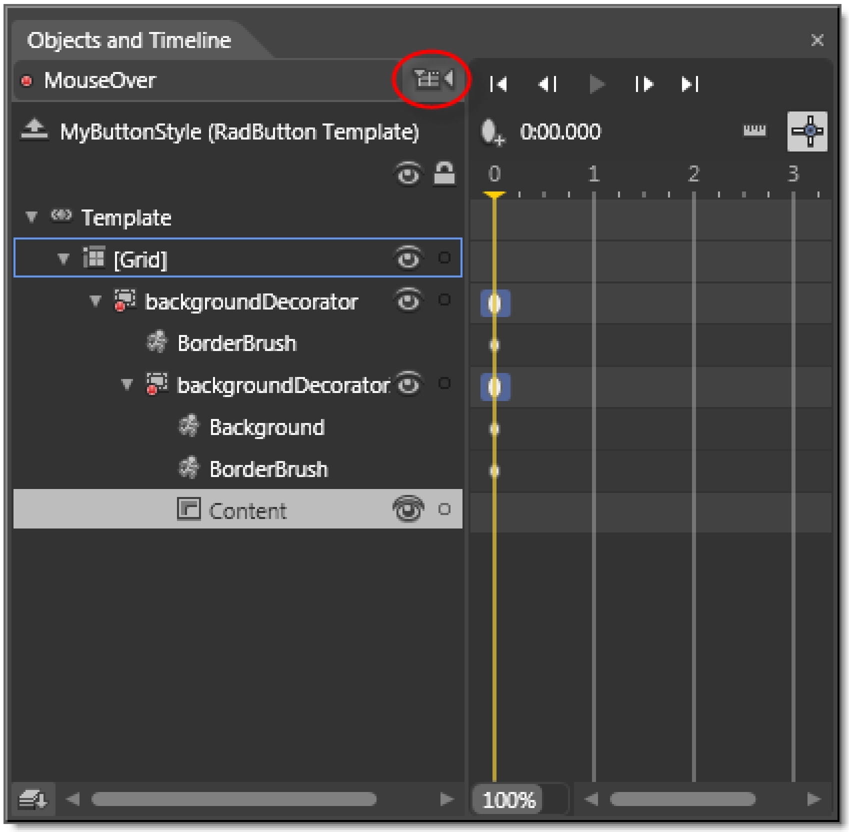
8) Right-click the RadButton and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "MyButtonStyle". Click **OK** to create the style resource and close the dialog.

9) In the Objects and Timeline pane, expand the tree view and locate and select the "Content" node.

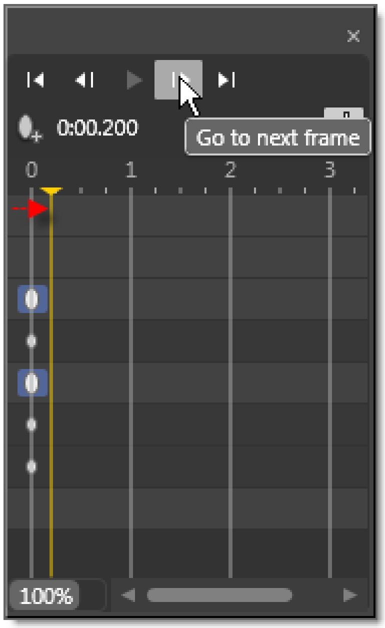
10)In the States pane, click the **MouseOver** state. This will turn on recording of properties. This should be reflected in the Artboard as shown in the screenshot below:



11)The Objects and Timeline pane should now display the animation Timeline. If the Timeline does not display, click the "camera" button (circled in red below) to open it. Notice the yellow Timeline marker at the "0" position in the Timeline. There are elliptical marks next to two border elements that indicate that animations have already been defined. These animations are responsible for changing brushes as the mouse passes over the button.



12)Move the Timeline marker to the next closest tick on the right of its current position. You can use the "Go to next frame" button to move the marker or simply drag it.

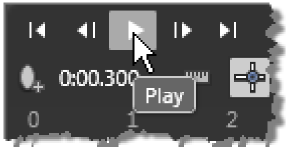


13)In the Properties pane, set **Transform > Scale > X** = "1.5" and **Transform > Scale > Y** = "1.5"

14)Click the "Go to next frame" button.

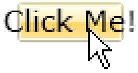
15)In the Properties pane, set **Transform > Scale > X** = "1" and **Transform > Scale > Y** = "1"

16)Click the **Play** button to preview the animation.



Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



As the mouse passes over the button, the content expands briefly. Also notice that the existing animations that change brush color also occur at the same time.

Ideas for Extending This Example

- Try changing other Transformation properties including location, skew or rotate.
- Try changing the duration of properties.
- Try adding animations for other state transitions.

5.7 Binding

5.7.1 Overview

Using Expression Blend you can interactively assign a DataContext and bind specific properties to any data in the project, including other elements in the page. The Data pane lets you drag-and-drop data onto elements in the Artboard to automatically assign the DataContext and begin configuring the ItemsSource. Data can come from business objects or you can use "sample" data where Expression Blend assigns mock data while the interface is still being developed.

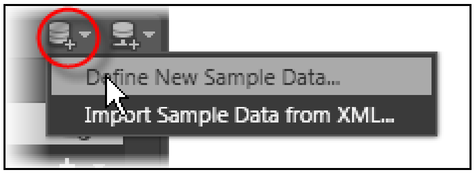
5.7.2 Create Sample Data Sources

Expression Blend introduces the notion of "Sample" data where you create data sources intended for use during prototyping and development. You can create objects or collections right in the designer and Expression Blend will populate with mock data. You can also import XML to populate your sample objects. Sample data can be switched off in the production compile to significantly reduce the applications footprint.

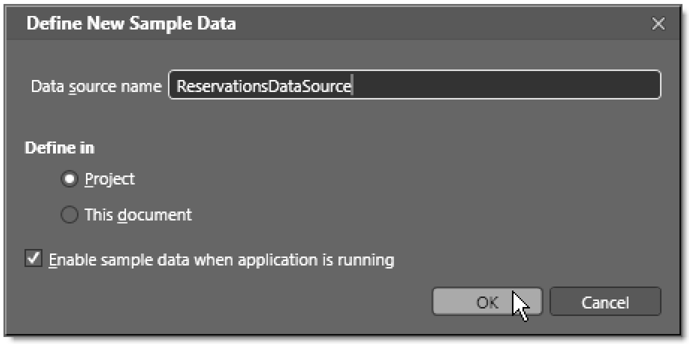
See the upcoming "Drag and Drop Binding" on how to bind sample and object data sources to elements in the page.

5.7.2.1 Define Sample Objects

You can create your own objects on the fly using the "Define New Sample Data" option.



This will display the "Define New Sample Data" dialog. Enter a new Data source name and click the **OK** button to create the data source and collection.


Define New Sample Data

Data source name: ReservationsDataSource

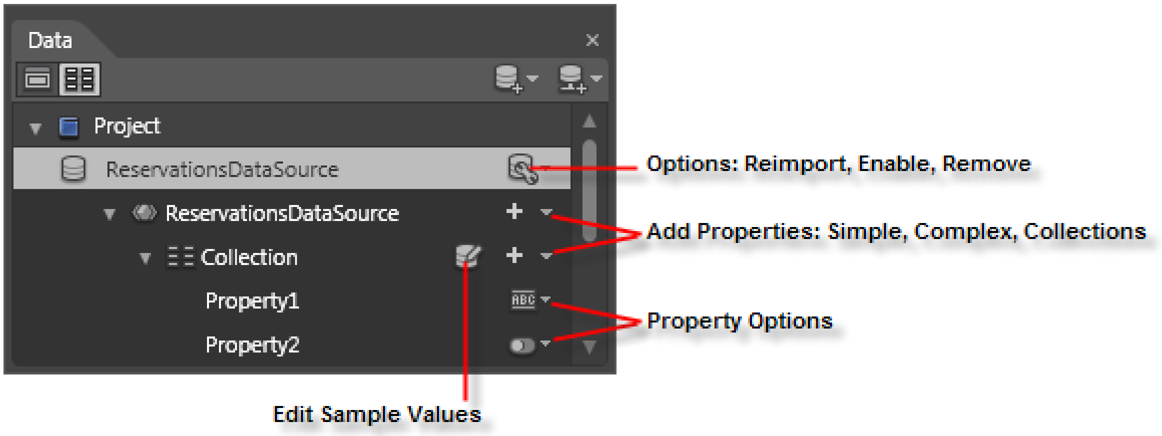
Define in

● Project

○ This document

☑ Enable sample data when application is running

The data source starts out as a collection with sample properties. Each property can be renamed by double-clicking. The drop down to the right of each property allows you to change the property types and configure options for each property type.



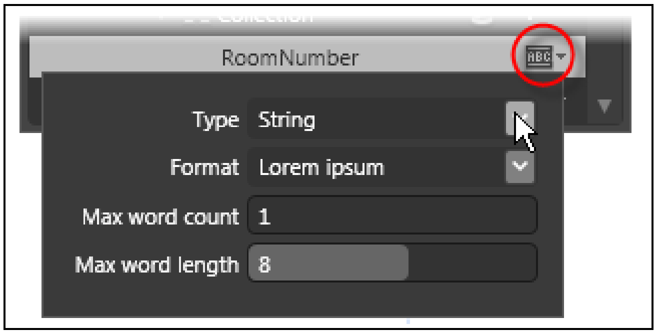
Options: Reimport, Enable, Remove

Add Properties: Simple, Complex, Collections

Property Options

Edit Sample Values

For example, we can rename the first property to "RoomNumber". By dropping down the Property Options, we can configure the property by choosing the type (String, Number, Boolean, Image), the Format (Name, URL, Date, Phone number, etc), maximum number of words and the maximum word length.



RoomNumber

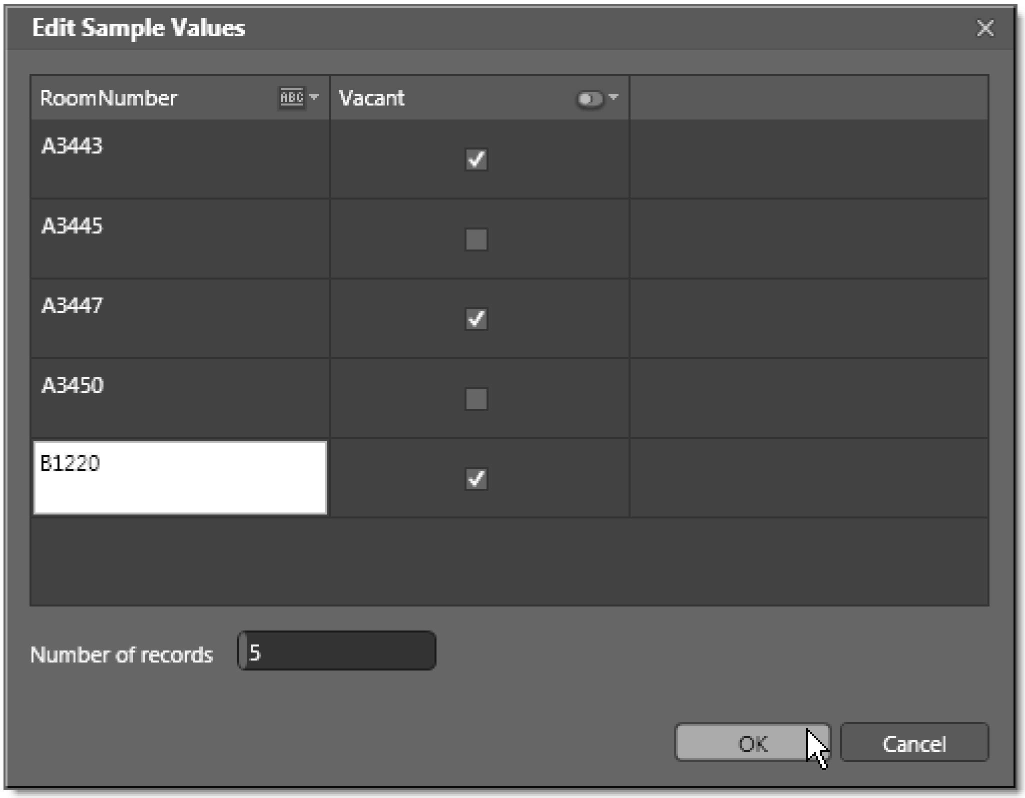
Type: String

Format: Lorem ipsum

Max word count: 1

Max word length: 8

Selecting the "Edit Sample Values" button displays the "Edit Sample Values" dialog. Rows are pre-populated by default with random data that fits the property options rules. In the dialog you can access the property options in the header of each column, edit the data directly in the grid or change the number of records.



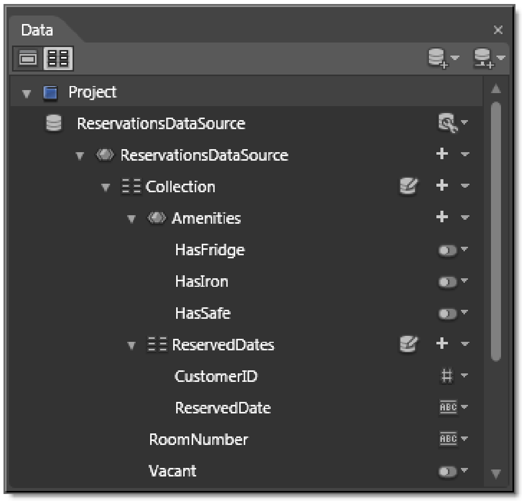
Edit Sample Values

RoomNumber	Vacant
A3443	☑
A3445	□
A3447	☑
A3450	□
B1220	☑

Number of records: 5

When the dialog is closed, the data refreshes the sample data XAML file maintained by Expression Blend. The generated XAML is shown below:

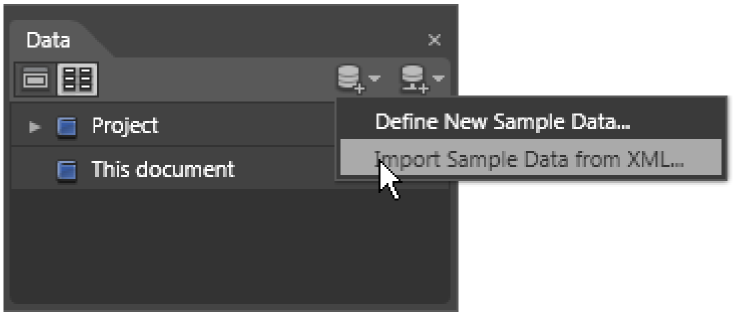
```
<!--
   *********   DO NOT MODIFY THIS FILE    *********
   This file is regenerated by a design tool. Making
   changes to this file can cause errors.
-->
<SampleData:ReservationsDataSource
xmlns:SampleData="clr-namespace:Expression.Blend.SampleData.ReservationsDataSource"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">
 <SampleData:ReservationsDataSource.Collection>
  <SampleData:Item RoomNumber="A3443" Vacant="True" />
  <SampleData:Item RoomNumber="A3445" Vacant="False" />
  <SampleData:Item RoomNumber="A3447" Vacant="True" />
  <SampleData:Item RoomNumber="A3450" Vacant="False" />
  <SampleData:Item RoomNumber="B1220" Vacant="True" />
 </SampleData:ReservationsDataSource.Collection>
</SampleData:ReservationsDataSource>
```



By using the Add Properties drop down you can build up records of arbitrary complexity. The screenshot shows a "Reservations" class with simple properties, a complex property called "Amenities" and a "ReservedDates" collection. The "complex" Amenities class contains several Boolean properties.

5.7.2.2 Import From XML

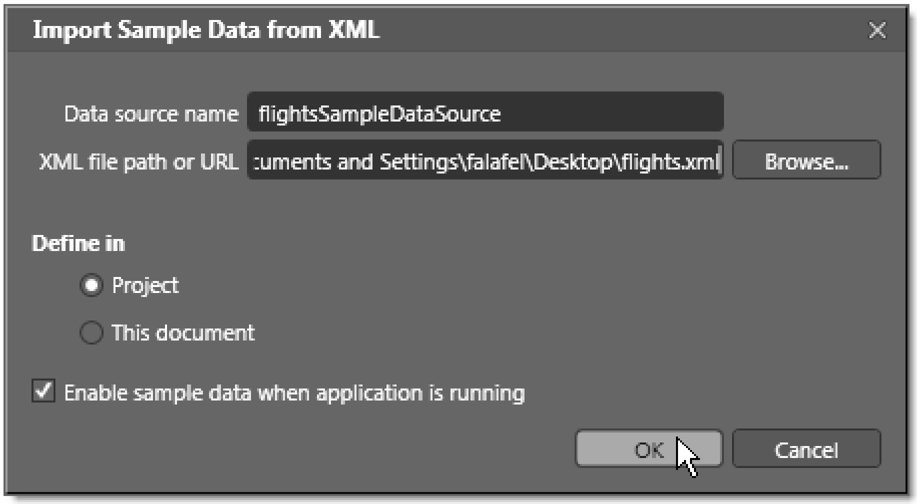
To create sample data from existing XML, use the Data pane "Add sample data source" drop down and select "Import Sample Data from XML...".



Shown below is a very small bit of XML that we can import as an example. The XML is stored in a file called "Flights.xml". You can also use the XML files that ship in the RadControls installation under the \Examples directory.

```
<Flights>
  <Flight Airline="United" FlightNumber="123" />
  <Flight Airline="Southwest" FlightNumber="3423" />
  <Flight Airline="American" FlightNumber="566" />
</Flights>
```

Clicking the "Import Sample Data from XML..." button displays the "Import Sample Data from XML" dialog. Use the **Browse...** button to locate an XML file for importing. You can store the resource that points to this data in either the user control or the application XAML. The "Enable sample data when application is running" checkbox can be shut off later when you don't need the overhead of the sample data. Click the **OK** to create the sample data objects in your project.



Data source name: flightsSampleDataSource

XML file path or URL: :uments and Settings\falafel\Desktop\flights.xml

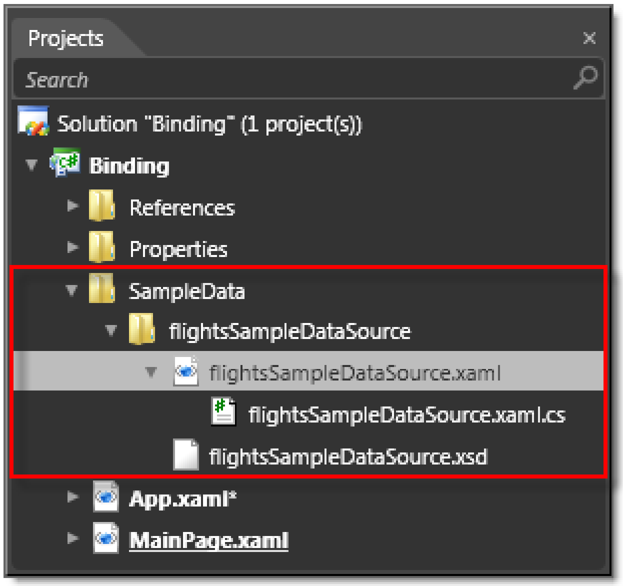
Define in

● Project

○ This document

☑ Enable sample data when application is running

So what happened when we imported the sample data? A "SampleData" directory was created in our project and populated with a second directory containing all the materials needed for our data source.



Projects

Search

Solution "Binding" (1 project(s))
Binding
References
Properties
SampleData
flightsSampleDataSource
flightsSampleDataSource.xaml
flightsSampleDataSource.xaml.cs
flightsSampleDataSource.xsd
App.xaml*
MainPage.xaml

The flightsSampleDataSource.xaml contains the imported data in XML form. The flightsSampleDataSource.xaml.cs wraps the data in class form complete with INotifyPropertyChanged and ObservableCollection<> support. These files are all generated and shouldn't be edited by hand. See an example of the generated code below:

```
//      *********    DO NOT MODIFY THIS FILE     *********
//      This file is regenerated by a design tool. Making
//      changes to this file can cause errors.
namespace Expression.Blend.SampleData.flightsSampleDataSource
{
  using System;

// To significantly reduce the sample data footprint in your production application, you can set
// the DISABLE_SAMPLE_DATA conditional compilation constant and disable sample data at runtime.
#if DISABLE_SAMPLE_DATA
  internal class Flights { }
#else

  public class Flights : System.ComponentModel.INotifyPropertyChanged
  {
    public event System.ComponentModel.PropertyChangedEventHandler PropertyChanged;
//. . .
```

In the App.xaml file, a resource has been added to allow easy declarative access when we bind the data source to an element in the page.

```
<Application . . .
  xmlns:SampleData1="clr-namespace:Expression.Blend.SampleData.flightsSampleDataSource" >
<Application.Resources>
  <SampleData1:Flights x:Key="flightsSampleDataSource" d:IsDataSource="True"/>
</Application.Resources>
</Application>
```

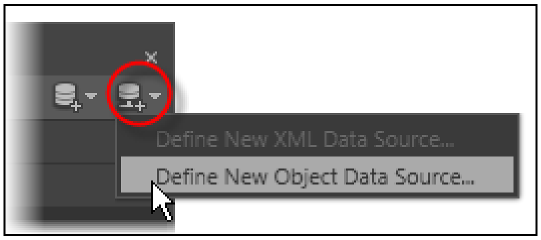
5.7.3 Create Object Data Sources

The sample data is handy for prototyping but limited in terms of the types of properties available and how they are implemented. Expression Blend lets you hook up to any object in an available namespace. Lets start with a "Flights" collection that defines instance of "Flight" in the constructor. This class could as easily be a web service client or some other business object.

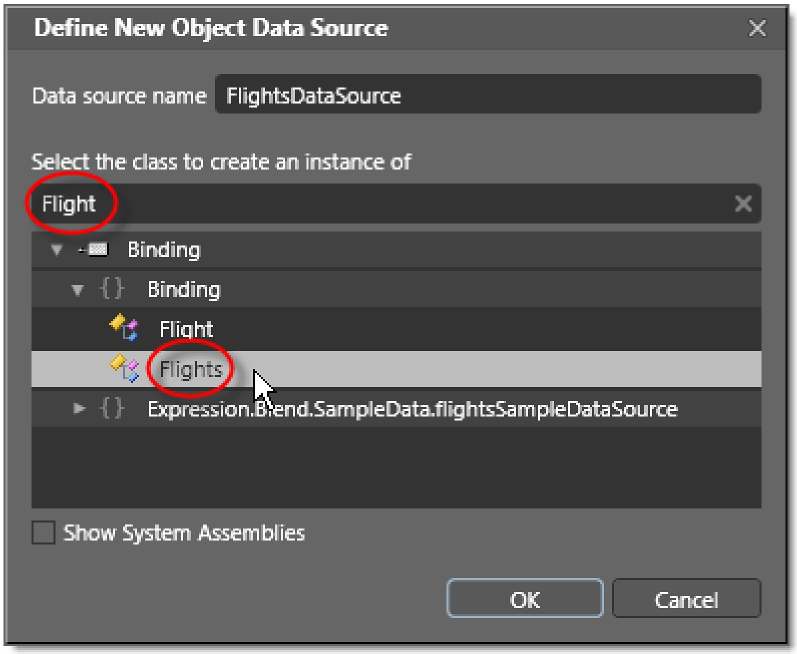
```
Public Class Flights
  Inherits ObservableCollection(Of Flight)
  Public Sub New()
   Me.Add(New Flight() With {.Airline = "American Airlines", .FlightNumber = "AA524"})
   Me.Add(New Flight() With {.Airline = "United", .FlightNumber = "UA55"})
   Me.Add(New Flight() With {.Airline = "Alaska Airline", .FlightNumber = "NA7785"})
  End Sub
End Class

Public Class Flight
  Private privateFlightNumber As String
  Public Property FlightNumber() As String
    Get
      Return privateFlightNumber
    End Get
    Set(ByVal value As String)
      privateFlightNumber = value
    End Set
  End Property
  Private privateAirline As String
  Public Property Airline() As String
    Get
      Return privateAirline
    End Get
    Set(ByVal value As String)
      privateAirline = value
    End Set
  End Property
End Class
```

```
public class Flights: ObservableCollection<Flight>
{
  public Flights()
  {
    this.Add(new Flight() { Airline = "American Airlines", FlightNumber = "AA524" });
    this.Add(new Flight() { Airline = "United", FlightNumber = "UA55" });
    this.Add(new Flight() { Airline = "Alaska Airline", FlightNumber = "NA7785" });
  }
}

public class Flight
{
  public string FlightNumber { get; set; }
  public string Airline { get; set; }
}
```

The first step to accessing a class for binding is to select "Define New Object Data Source..." from the Data pane.


Define New XML Data Source...
Define New Object Data Source...

This action displays the "Define New Object Data Source" dialog. The dialog has a tree view of classes available to the Silverlight application. Use the entry above the tree view to filter the list. In the screenshot below, the "Flights" class is selected.

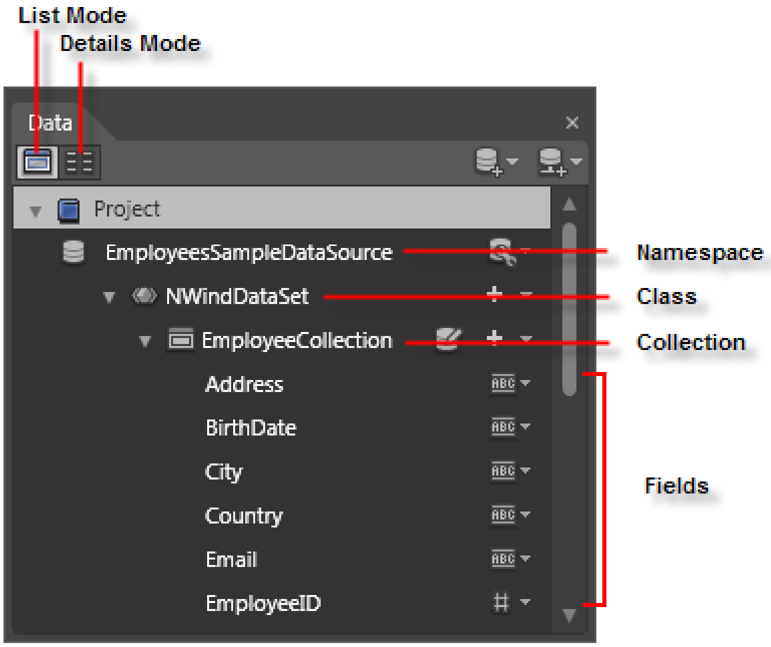

Define New Object Data Source
Data source name FlightsDataSource
Select the class to create an instance of
Flight
Binding
Binding
Flight
Flights
Expression.Blend.SampleData.flightsSampleDataSource
Show System Assemblies
OK
Cancel

The result of selecting **OK** in this dialog is to add a reference to the assembly that contains the data source class. In the example below, the assembly is called "Binding" and the XML namespace alias is "local". A resource is added automatically that references the "Flights" class called "FlightsDataSource". Now we can use this data source in binding expressions. See the next section "Drag and Drop Binding" on how to bind your data sources to elements in the page.

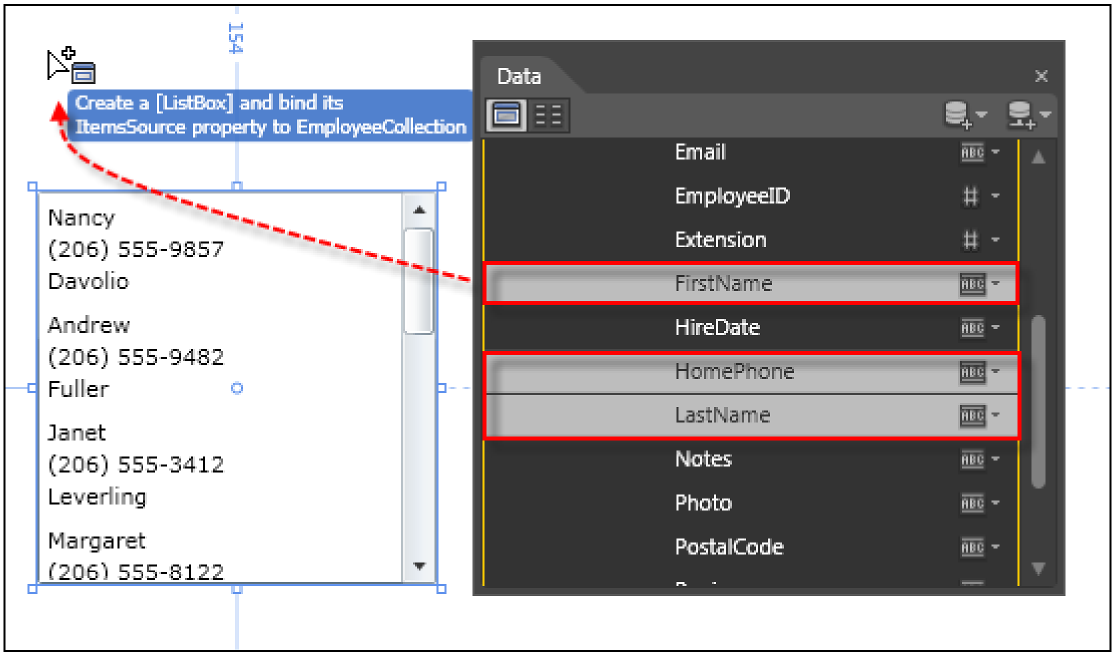
```
<UserControl xmlns:local="clr-namespace:Binding" . . .>

  <UserControl.Resources>
    <local:Flights x:Key="FlightsDataSource" d:IsDataSource="True"/>
  </UserControl.Resources>

  <Grid x:Name="LayoutRoot" Background="White" />
</UserControl>
```

5.7.4 Drag and Drop Binding

How a drag operation plays out between the Data pane and the Artboard depends on three things: the node being dragged, where the node is being dragged to and the "Mode" button that is currently pressed. Either the "List" or "Details" Mode button will be depressed at any one time. The node being dragged can be a namespace, a data source class, a collection within the class or a field within a collection.



In List Mode, if you drag fields to the Artboard, a ListBox is created and the ItemsSource is bound to the collection containing the fields.



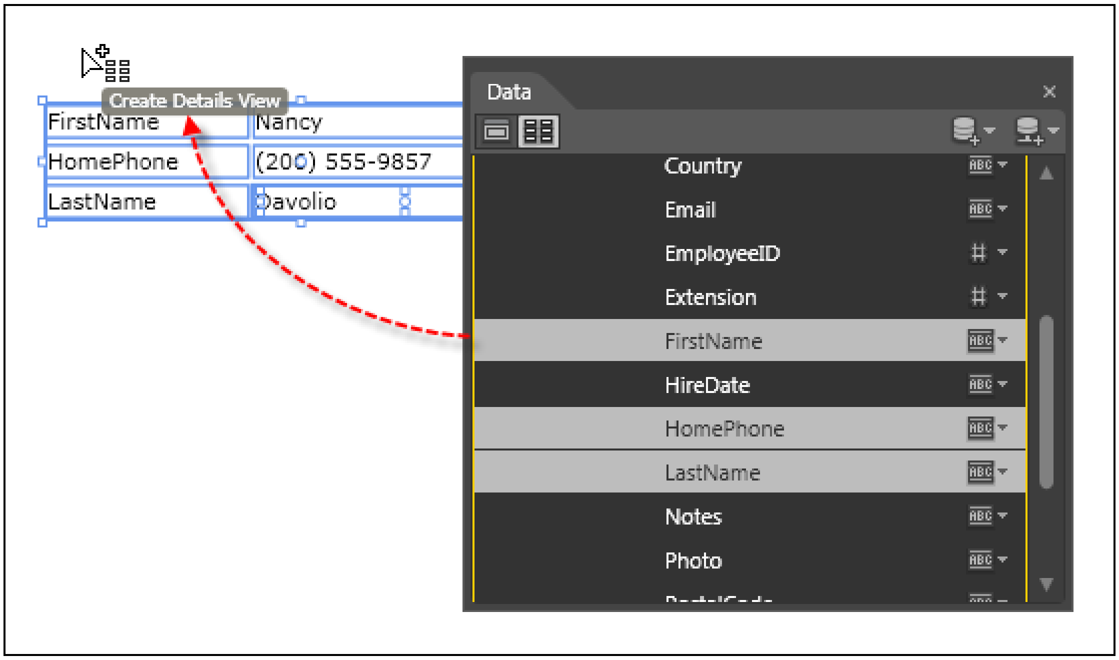
In XAML, the result of the drag operation in the screenshot above is:

- A DataTemplate is created to contain TextBlock elements bound to the Fields that were dragged
- The DataContext of the parent, i.e. the "LayoutRoot" Grid, is bound to the data source class
- The ListBox ItemsSource is bound to the collection.

The XAML below shows the placement of DataTemplate, DataContext and ItemsSource after the drag in List Mode.

```
<UserControl.Resources>
  <DataTemplate x:Key="EmployeeTemplate">
    <StackPanel>
      <TextBlock Text="{Binding FirstName}"/>
      <TextBlock Text="{Binding HomePhone}"/>
      <TextBlock Text="{Binding LastName}"/>
    </StackPanel>
  </DataTemplate>
</UserControl.Resources>

<Grid x:Name="LayoutRoot" Background="White"
   DataContext="{Binding Source={StaticResource EmployeesSampleDataSource}}" >
  <ListBox HorizontalAlignment="Left" Margin="25,154,0,129" Width="200"
   ItemTemplate="{StaticResource EmployeeTemplate}" ItemsSource="{Binding EmployeeCollection}"/>
</Grid>
```

In Details mode, fields dragged onto the Artboard are rendered as a set of TextBlocks wrapped in a bound grid. If we already have a bound ListBox on the page, Expression Blend automatically binds to the ListBox element SelectedItem property, so that the ListBox and the details are in sync.



FirstName	Nancy
HomePhone	(206) 555-9857
LastName	Davolio

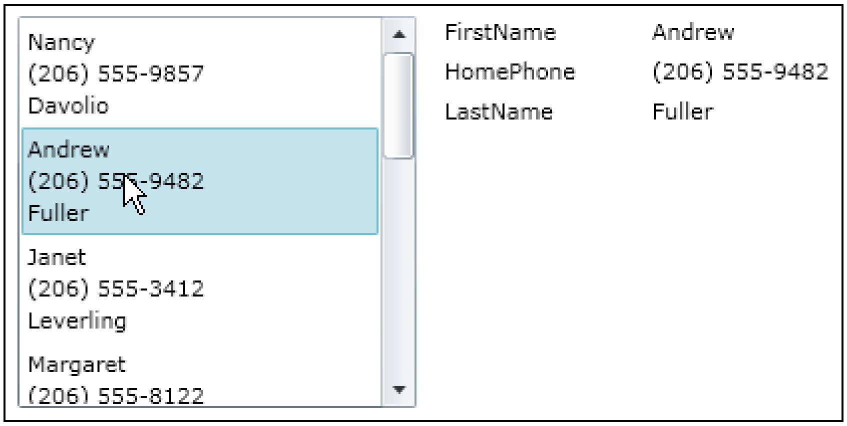
Here's an abbreviated look at the XAML produced by a drag of fields in Details mode. Attributes not directly related to this binding scenario have been removed. Notice that the innermost grid containing the detail items is bound to the "listBox" element, SelectedItem property. .

```
<UserControl.Resources>
  <DataTemplate x:Key="EmployeeTemplate">
    <StackPanel>
      <TextBlock Text="{Binding FirstName}"/>
      <TextBlock Text="{Binding HomePhone}"/>
      <TextBlock Text="{Binding LastName}"/>
    </StackPanel>
  </DataTemplate>
</UserControl.Resources>

<Grid x:Name="LayoutRoot"
   DataContext="{Binding Source={StaticResource EmployeesSampleDataSource}}" >
  <ListBox x:Name="listBox"  ItemTemplate="{StaticResource EmployeeTemplate}"
    ItemsSource="{Binding EmployeeCollection}" />
  <Grid DataContext="{Binding SelectedItem, ElementName=listBox}"
     d:DataContext="{Binding EmployeeCollection[0]}">
    <TextBlock Text="FirstName"/>
    <TextBlock Text="{Binding FirstName}" />
    <TextBlock Text="HomePhone"/>
    <TextBlock Text="{Binding HomePhone}" />
    <TextBlock Text="LastName"/>
    <TextBlock Text="{Binding LastName}" />
  </Grid>
</Grid>
```

Notes

Don't be confused by the "d:DataContext" tag in the XAML. This is for design time only and binds to the first element in the collection at design time just so you have some data to look at. You can remove this tag without impact to the running application

When you run the application, the fields you selected in the Data pane are displayed in the ListBox. When a list box item is selected, the corresponding data shows up in the detail TextBlocks.



Nancy
(206) 555-9857
Davolio

Andrew
(206) 555-9482
Fuller

Janet
(206) 555-3412
Leverling

Margaret
(206) 555-8122

FirstName	Andrew
HomePhone	(206) 555-9482
LastName	Fuller

5.7.5 Binding RadControls Walk Through

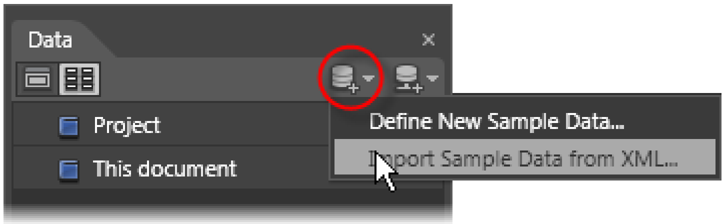
This walk through demonstrates binding a RadGridView to sample data.

Project Setup

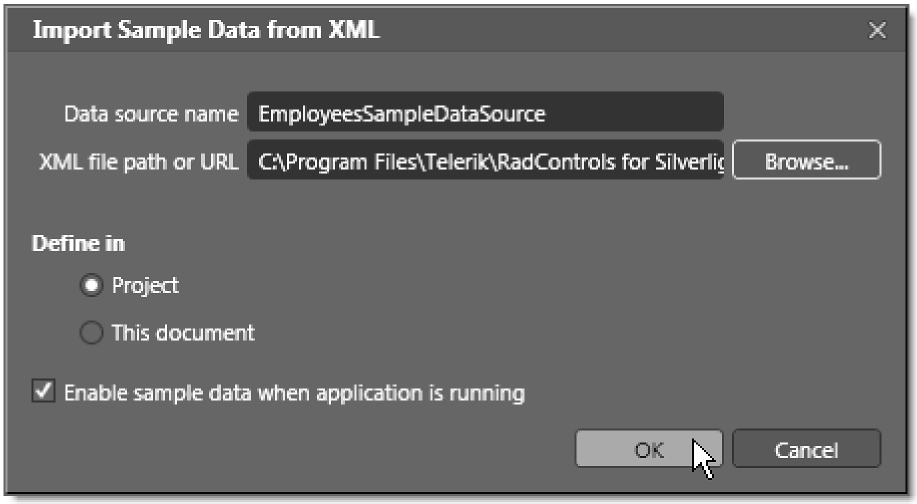
1) Run Expression Blend.
2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*
3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight Application" from the right-most list. Enter a unique name for the project and click **OK**.

Edit the Page in Expression Blend

1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.
2) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu. Add references to the following assemblies:
   a) **Telerik.Windows.Controls**
   b) **Telerik.Windows.Controls.GridView**
   c) **Telerik.Windows.Data**
3) From the Project menu select **Build Project**.
4) From the Assets pane, locate RadGridView and drag it onto the MainPage.xaml Artboard.
5) In the Data pane, click "Import Sample Data From XML...". This will display the "Import Sample Data From XML" dialog.



6) Click the **Browse...** button, locate and select the "Employees.xml" file. "Employees.xml" can be found in the RadControls installation directory under the \Examples\Datasources directory. Click **OK** to create the data source.



Import Sample Data from XML

Data source name: EmployeesSampleDataSource

XML file path or URL: C:\Program Files\Telerik\RadControls for Silverlig

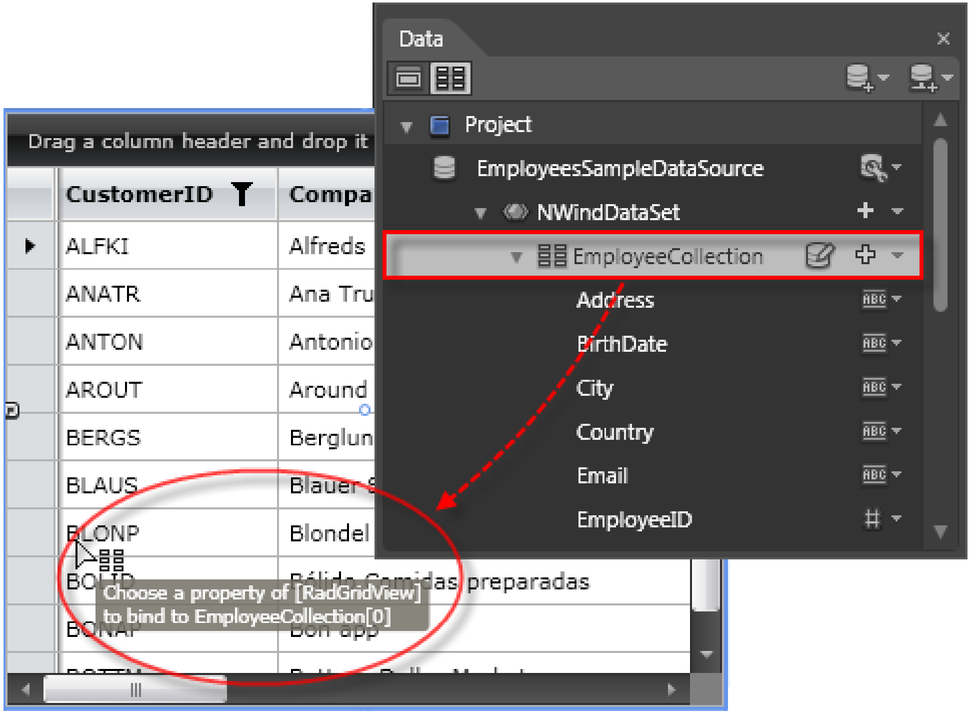
Define in

● Project

○ This document

☑ Enable sample data when application is running

7) Drag the "EmployeeCollection" from the Data pane onto the RadGridView in the Artboard. Notice the tooltip that appears "Choose a property of [RadGridView] to bind to EmployeeCollection[0]". The "Create Data Binding" dialog will appear in response to the drop.

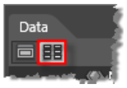


Drag a column header and drop it

CustomerID	Compa
ALFKI	Alfreds
ANATR	Ana Tru
ANTON	Antonio
AROUT	Around
BERGS	Berglun
BLAUS	Blauer S
BLONP	Blondel
BOLID	Bólido Comidas preparadas
BONAP	Bon app

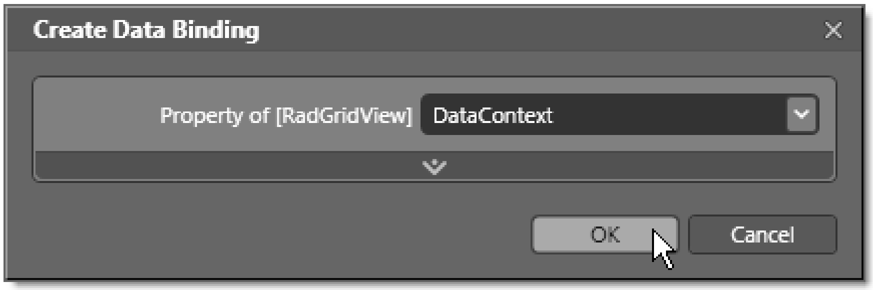
Choose a property of [RadGridView] to bind to EmployeeCollection[0]

Notes



Be sure that the "Details Mode" button is down before you drag the collection. The other button "List Mode" will create a bound ListBox.

8) In the "Create Data Binding" dialog, select the **DataContext** property and click **OK** to close the dialog.

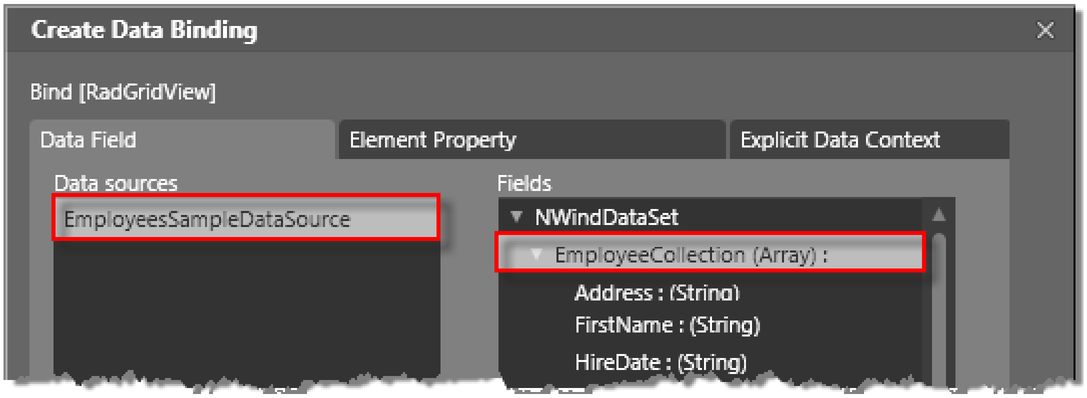


Create Data Binding

Property of [RadGridView] DataContext

9) In the **Properties pane > Miscellaneous > ItemsSource**, select **Data Binding...** from the Advanced Property Options button. This action will display the **Create Data Binding** dialog.

10)In the Create Data Binding dialog, select the Data Field tab, select "EmployeesSampleDataSource" from the Data sources list and "EmployeeCollection" in the Fields list. Click **OK** to create the binding and close the dialog.



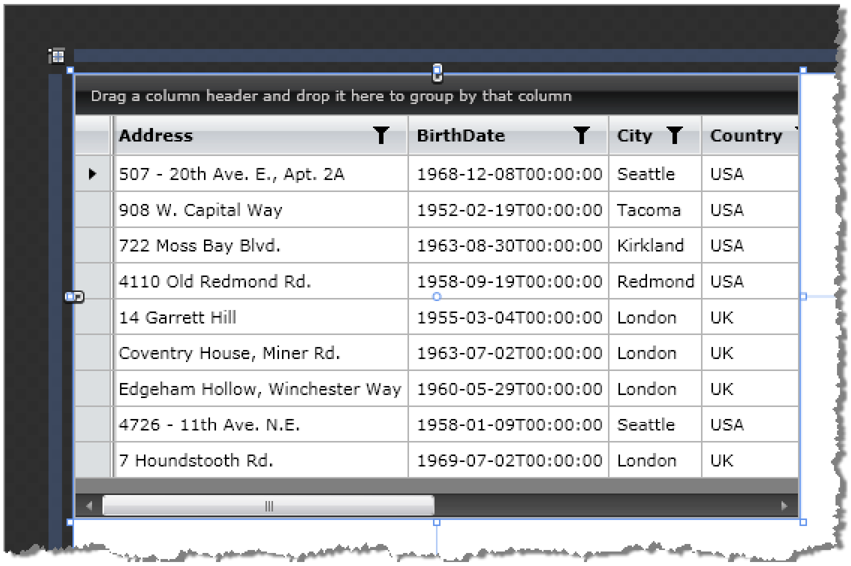
Create Data Binding

Bind [RadGridView]

Data sources: EmployeesSampleDataSource

Fields:
- NWindDataSet
  - EmployeeCollection (Array) :
    - Address : (String)
    - FirstName : (String)
    - HireDate : (String)

11)The RadGridView displays the data in the Artboard at design time.



Drag a column header and drop it here to group by that column

Address	BirthDate	City	Country
507 - 20th Ave. E., Apt. 2A	1968-12-08T00:00:00	Seattle	USA
908 W. Capital Way	1952-02-19T00:00:00	Tacoma	USA
722 Moss Bay Blvd.	1963-08-30T00:00:00	Kirkland	USA
4110 Old Redmond Rd.	1958-09-19T00:00:00	Redmond	USA
14 Garrett Hill	1955-03-04T00:00:00	London	UK
Coventry House, Miner Rd.	1963-07-02T00:00:00	London	UK
Edgeham Hollow, Winchester Way	1960-05-29T00:00:00	London	UK
4726 - 11th Ave. N.E.	1958-01-09T00:00:00	Seattle	USA
7 Houndstooth Rd.	1969-07-02T00:00:00	London	UK

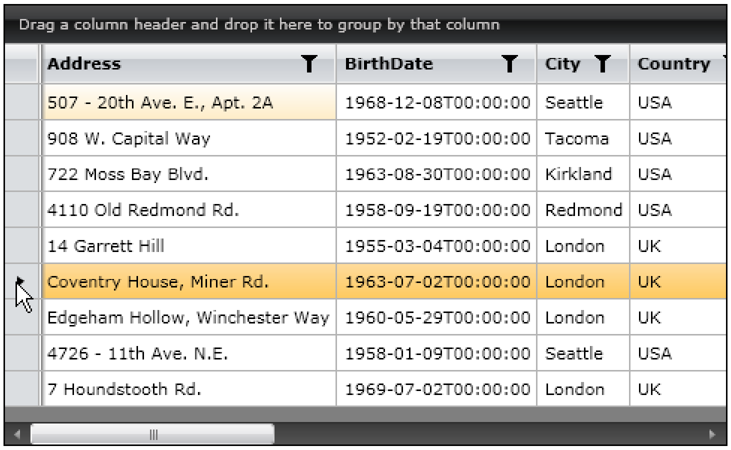
Notes

If you review the XAML at this point, you can see that the DataContext is bound to the data source through the "EmployeesSampleDataSource" resource. The ItemsSource is bound to the EmployeeCollection within the data source.

```
<telerik:RadGridView Margin="19,27,0,141" d:LayoutOverrides="Width, Height"
 DataContext="{Binding Source={StaticResource EmployeesSampleDataSource}}"
 ItemsSource="{Binding EmployeeCollection, Mode=OneWay}"/>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



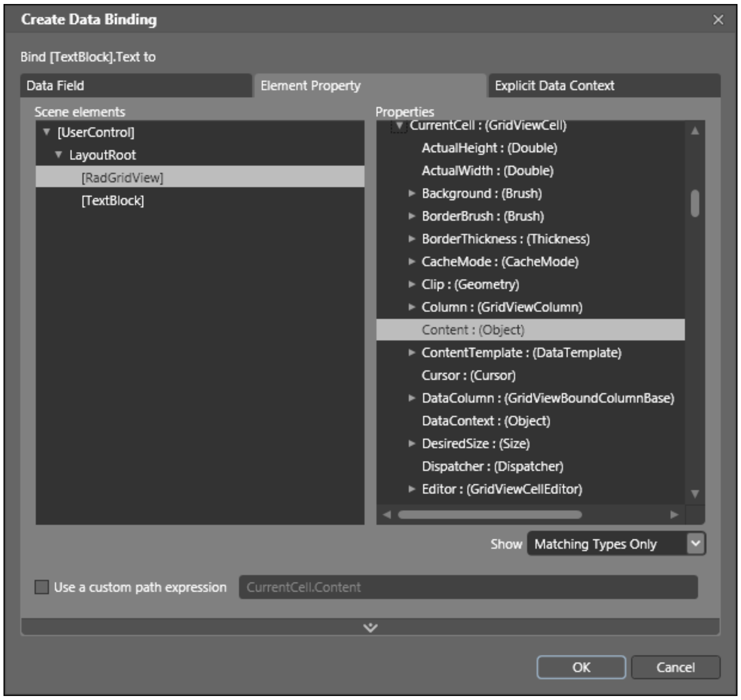
Drag a column header and drop it here to group by that column

Address	BirthDate	City	Country
507 - 20th Ave. E., Apt. 2A	1968-12-08T00:00:00	Seattle	USA
908 W. Capital Way	1952-02-19T00:00:00	Tacoma	USA
722 Moss Bay Blvd.	1963-08-30T00:00:00	Kirkland	USA
4110 Old Redmond Rd.	1958-09-19T00:00:00	Redmond	USA
14 Garrett Hill	1955-03-04T00:00:00	London	UK
Coventry House, Miner Rd.	1963-07-02T00:00:00	London	UK
Edgeham Hollow, Winchester Way	1960-05-29T00:00:00	London	UK
4726 - 11th Ave. N.E.	1958-01-09T00:00:00	Seattle	USA
7 Houndstooth Rd.	1969-07-02T00:00:00	London	UK

Ideas for Extending This Example

- Add a TextBlock that shows the contents of the current RadGridView cell item.

  This can be done through the **Properties pane > Common Properties > Text**, clicking the Advanced Property Options button and selecting **Data Binding...** from the menu. In the Create Data Binding dialog, select the Element property tab. Then select the RadGridView from the "Scene Elements" list on the left and the **CurrentCell > Content** from the "Properties" list on the right.



Bind [TextBlock].Text to

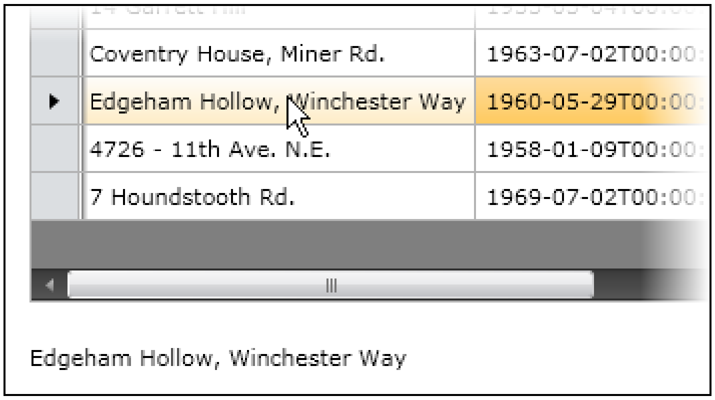
Show: Matching Types Only

☐ Use a custom path expression: CurrentCell.Content

Clicking **OK** on the dialog creates the following binding in the XAML where the Text property of the TextBlock is bound to the RadGridView CurrentCell.Content property.

```
<TextBlock HorizontalAlignment="Left" Margin="19,0,0,105" VerticalAlignment="Bottom"
Text="{Binding CurrentCell.Content, ElementName=radGridView, Mode=OneWay}" TextWrapping="Wrap"/>
```

When you run the modified application, the contents of the selected cell show up in the text block.



	Coventry House, Miner Rd.	1963-07-02T00:00
▶	Edgeham Hollow, Winchester Way	1960-05-29T00:00
	4726 - 11th Ave. N.E.	1958-01-09T00:00
	7 Houndstooth Rd.	1969-07-02T00:00

Edgeham Hollow, Winchester Way

5.8 Wrap Up

In this chapter you learned how to apply your general Silverlight knowledge to the Expression Blend environment. First we took a tour of project types available in Expression Blend. Then you delved deeper into elements of the Expression Blend environment, becoming familiar with the major functional areas of the application. You learned how Expression Blend organizes and manipulates resources. You used Expression Blend to create and edit styles and you also restyled a RadControl. You used Expression Blend to edit a control template. You saw how Expression Blend creates animation and how animation can be triggered in code and through state changes. Finally, you learned how to interactively bind data to elements.

Part VI

Theming and Skinning

6 Theming and Skinning

6.1 Objectives

In this chapter you will learn how Telerik themes are used with RadControls and other Silverlight elements to create a uniform look-and-feel for your application. First you will apply predefined themes to RadControls and observe how themes can be changed without otherwise altering target elements. You will learn how themes are distributed and applied to elements in XAML and in the code behind. You will apply themes to an entire application and to individual elements in the application. You will also create a custom theme and learn the syntax for applying the theme.

In the "Modifying Themes" section of the chapter you will learn about the "Themes" solution and its structure. You will modify theme brushes to skin the theme with a new color set. You will apply the modified theme to the UI in a Visual Studio project. Finally, you will expand this example to add effects and animation to a RadButton control.

Find the projects for this chapter at...

```
\Courseware\Projects\<CS|VB>\Theming\Theming.sln
```

6.2 Overview

A theme is a mechanism to organize multiple sets of styles and apply them all at one time to a control or an entire application. The theme engine of RadControls for Silverlight is quite powerful and provides the developers with the ability to apply an application-wide theme on all RadControls, as well as specific themes on a single control. If the theme does not contain the needed styles, the controls fallback to their default styles. You can create an application theme for just a couple of controls and the rest will display with their default styles. RadControls for Silverlight comes with a set of predefined themes, that can be used right out of the box, modified for your specific application requirements or new themes can be created from scratch.

6.3 Getting Started

In this walk through you will apply themes to RadControls and standard Silverlight elements.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:
   a) **Telerik.Windows.Controls**
   b) **Telerik.Windows.Controls.Input**
   c) **Telerik.Windows.Controls.Navigation**
   d) **Telerik.Windows.Themes.Summer**
   e) **Telerik.Windows.Themes.Windows7**

XAML Editing

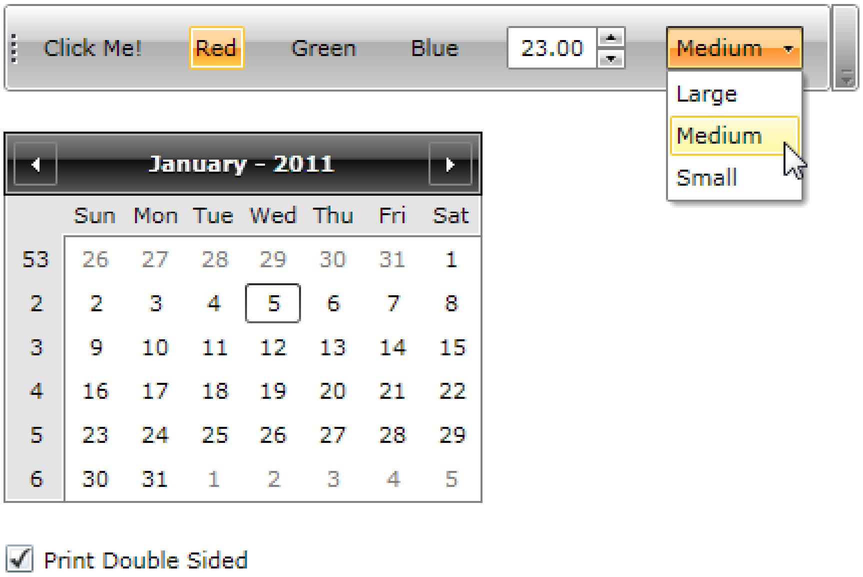
1) Open MainPage.xaml for editing.
2) Add XML namespaces to the UserControl element:

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
...>
```

3) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. This will define a number of RadControls and one standard Silverlight CheckBox control. No specific theme has been applied to any of the controls.

```
<Grid x:Name="LayoutRoot">
  <StackPanel>
    <telerik:RadToolBarTray Margin="10" HorizontalAlignment="Left" >
      <telerik:RadToolBar >
        <telerik:RadToolBar.Items>
          <telerik:RadButton Content="Click Me!" Margin="10" />
          <telerik:RadRadioButton Content="Red" IsChecked="True" Margin="10" />
          <telerik:RadRadioButton Content="Green" Margin="10" />
          <telerik:RadRadioButton Content="Blue" Margin="10" />
          <telerik:RadNumericUpDown Margin="10" Value="23" />
          <telerik:RadComboBox Margin="10">
            <telerik:RadComboBoxItem Content="Large" />
            <telerik:RadComboBoxItem Content="Medium" />
            <telerik:RadComboBoxItem Content="Small" />
          </telerik:RadComboBox>
        </telerik:RadToolBar.Items>
      </telerik:RadToolBar>
    </telerik:RadToolBarTray>
    <telerik:RadCalendar Margin="10" HorizontalAlignment="Left" />
    <StackPanel Orientation="Horizontal">
      <CheckBox Content="Print Double Sided" IsChecked="True" Margin="10" />
    </StackPanel>
  </StackPanel>
</Grid>
```

4) Press **F5** to run the application. The controls display their default appearance.



January - 2011

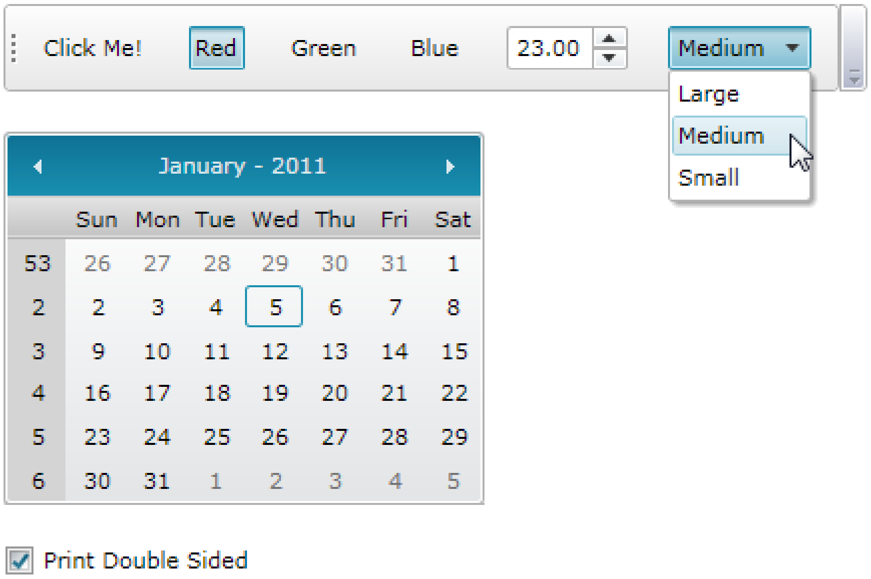
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
53	26	27	28	29	30	31	1
2	2	3	4	5	6	7	8
3	9	10	11	12	13	14	15
4	16	17	18	19	20	21	22
5	23	24	25	26	27	28	29
6	30	31	1	2	3	4	5

☑ Print Double Sided

5) Add the tag "telerik:StyleManager.Theme="Summer" to the RadToolBarTray, RadCalendar and CheckBox controls. An abbreviated version of the XAML is shown below.

```
<telerik:RadToolBarTray telerik:StyleManager.Theme="Summer" >
 . . .
</telerik:RadToolBarTray>
<telerik:RadCalendar  . . . telerik:StyleManager.Theme="Summer"  />
<StackPanel >
  <CheckBox  . . . telerik:StyleManager.Theme="Summer" />
</StackPanel>
```

6) Press **F5** to run the application. Now all the RadControls inside the tool bar tray, the RadCalendar and the standard Silverlight CheckBox control are all styled in the "Summer" theme.



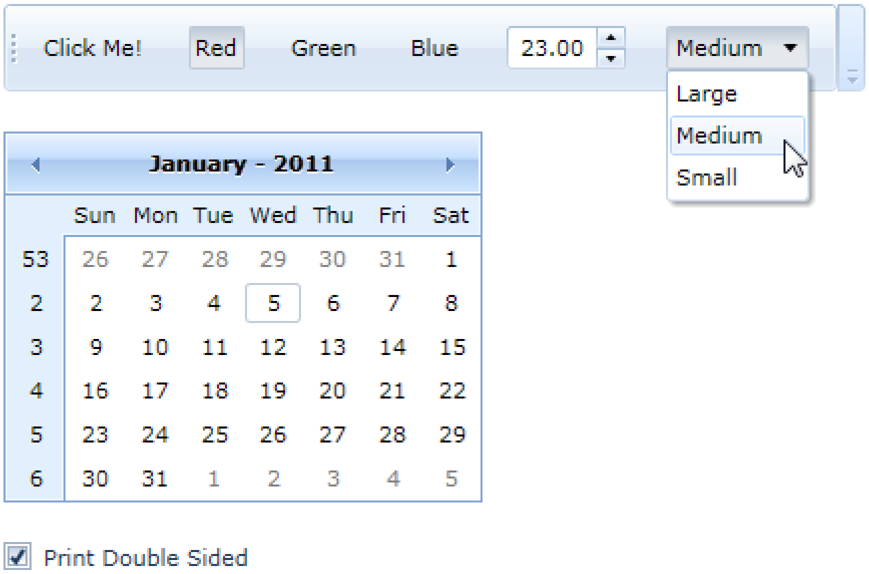
January - 2011

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
53	26	27	28	29	30	31	1
2	2	3	4	5	6	7	8
3	9	10	11	12	13	14	15
4	16	17	18	19	20	21	22
5	23	24	25	26	27	28	29
6	30	31	1	2	3	4	5

☑ Print Double Sided

7) Edit StyleManager.Theme tags in the XAML and assign the "Windows7" theme.

8) Press **F5** to run the application. Now all elements on the page display the "Windows7" theme.



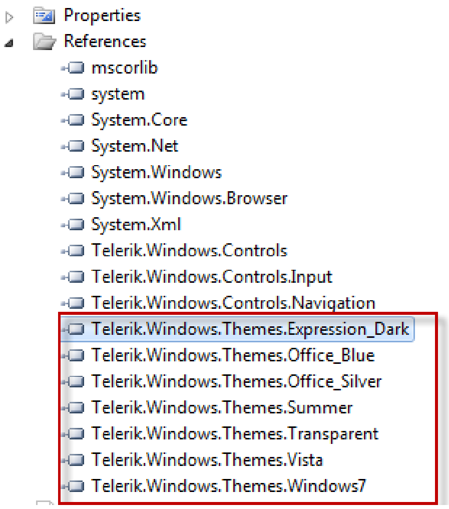
January - 2011

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
53	26	27	28	29	30	31	1
2	2	3	4	5	6	7	8
3	9	10	11	12	13	14	15
4	16	17	18	19	20	21	22
5	23	24	25	26	27	28	29
6	30	31	1	2	3	4	5

☑ Print Double Sided

6.4 Applying Themes to RadControls

Each theme is distributed as an assembly that contains a ResourceDictionary full of Styles. To use a theme in XAML or in code, first add a reference to the theme assembly. The screenshot below shows multiple assembly references in Solution Explorer. The "Office_Black" theme is the default theme and you do not have to add a reference to use it.



Properties
References
mscorlib
system
System.Core
System.Net
System.Windows
System.Windows.Browser
System.Xml
Telerik.Windows.Controls
Telerik.Windows.Controls.Input
Telerik.Windows.Controls.Navigation
Telerik.Windows.Themes.Expression_Dark
Telerik.Windows.Themes.Office_Blue
Telerik.Windows.Themes.Office_Silver
Telerik.Windows.Themes.Summer
Telerik.Windows.Themes.Transparent
Telerik.Windows.Themes.Vista
Telerik.Windows.Themes.Windows7

Applying Themes in XAML

To use a theme in XAML, first declare an XML namespace alias to **Telerik.Windows.Controls** so that you have access to the **StyleManager.Theme** attached property.

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .>
```

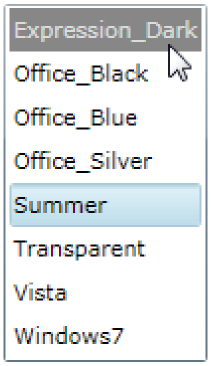
To apply a theme to a RadControl, add a **StyleManager.Theme** attached property. The XAML below applies the "Windows7" theme to a RadButton.

```
<telerik:RadButton telerik:StyleManager.Theme="Windows7" />
```

You can also apply Telerik themes to standard controls such as Button, Check, RadioButton, ScrollViewer, ListBox, etc. The example below sets themes to ListBoxItem elements, a different theme per item. **Note:** You typically use a single theme against all your elements to achieve a common look-and-feel, but this demonstrates that themes can be applied individually.

```
<ListBox >
   <ListBoxItem Content="Expression_Dark"
                 telerik:StyleManager.Theme="Expression_Dark" />
          <ListBoxItem Content="Office_Black"
                 telerik:StyleManager.Theme="Office_Black" />
          <ListBoxItem Content="Office_Blue"
                 telerik:StyleManager.Theme="Office_Blue" />
          <ListBoxItem Content="Office_Silver"
                 telerik:StyleManager.Theme="Office_Silver" />
          <ListBoxItem Content="Summer"
                 telerik:StyleManager.Theme="Summer" />
          <ListBoxItem Content="Transparent"
                 telerik:StyleManager.Theme="Transparent" />
          <ListBoxItem Content="Vista"
                 telerik:StyleManager.Theme="Vista" />
          <ListBoxItem Content="Windows7"
                 telerik:StyleManager.Theme="Windows7" />
</ListBox>
```

Here is a screenshot of the list box with themed items in the running Silverlight application.


Expression_Dark
Office_Black
Office_Blue
Office_Silver
Summer
Transparent
Vista
Windows7

You can also set a theme for the entire application by adding a resource to the App.xaml file. The resource points to one of the theme objects and sets the **IsApplicationTheme** property to "True".

XAML

```
<Application.Resources>
  <telerik:SummerTheme x:Key="theme" IsApplicationTheme="True" />
</Application.Resources>
```

Notes

At the time of this writing, application themes only work against RadControls. To apply themes to standard Silverlight controls they need to be applied individually.

Applying Themes in Code

Use the StyleManager static **SetTheme()** method to apply themes in code. Pass the target element to be themed and an instance of a theme object.

VB

```
StyleManager.SetTheme(calendar, New Office_BlackTheme())
```

C#

```
StyleManager.SetTheme(calendar, new Office_BlackTheme());
```

You can apply a theme to the entire application in code, but do this as early as possible. Use the Application.Startup event or the page constructor. The example below sets the application theme to "Vista" in the constructor before calling InitializeComponent().

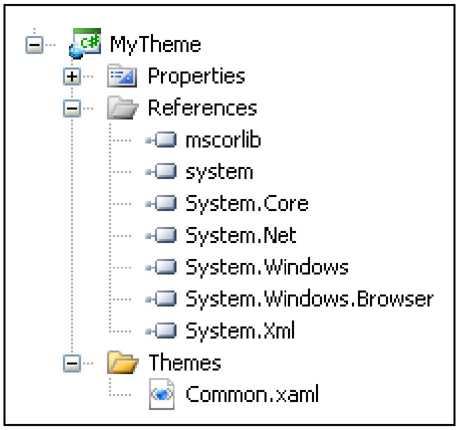
VB

```
Public Sub New()
  StyleManager.ApplicationTheme = New VistaTheme()
  ' or ...
  'new VistaTheme().IsApplicationTheme = true;
  InitializeComponent()
End Sub
```

```
public MainPage()
{
   StyleManager.ApplicationTheme = new VistaTheme();
   // or ...
   //new VistaTheme().IsApplicationTheme = true;
   InitializeComponent();
}
```

6.5 Creating a Custom Theme

You can create a custom theme that covers all of the controls you expect to have in your application or just a few controls. If your theme doesn't contain a particular style, the control will fallback to its default. The theme project is contained in a Silverlight Class Library. The example project below has a Themes folder that contains a Silverlight Resource Dictionary named "Common.xaml".



Gotcha!

Be sure to use a ***Silverlight*** class library, so that the proper Silverlight versions of the referenced DLL's are included.

The resource dictionary has one or more styles. The example below has a single Style element where the target type is Button and the Key is the full class name of the button. This style only sets the Background property to "Red".

```
<ResourceDictionary . . .>
   <Style TargetType="Button" x:Key="System.Windows.Controls.Button">
      <Setter Property="Background" Value="Red" />
   </Style>
</ResourceDictionary>
```

To use the custom theme, create a resource that points to the theme, then assign the resource to the StyleManager.Theme property using a "StaticResource" binding expression. Notice the syntax used in the resource below for the Source attribute follows the pattern required to access resource URIs located in other assemblies. The general syntax is:

"/assemblyShortName;component/resourceLocation".

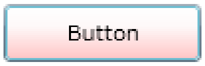
... where the leading slash and the component keyword followed by a slash are both required.

```
<UserControl . . .
 xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation">

  <UserControl.Resources>
    <telerik:Theme x:Key="myTheme" Source="/MyTheme;component/Themes/Common.xaml" />
  </UserControl.Resources>

  <Grid x:Name="LayoutRoot" Background="White">
    <Button Content="Button"  telerik:StyleManager.Theme="{StaticResource myTheme}" />
  </Grid>
</UserControl>
```

Notes

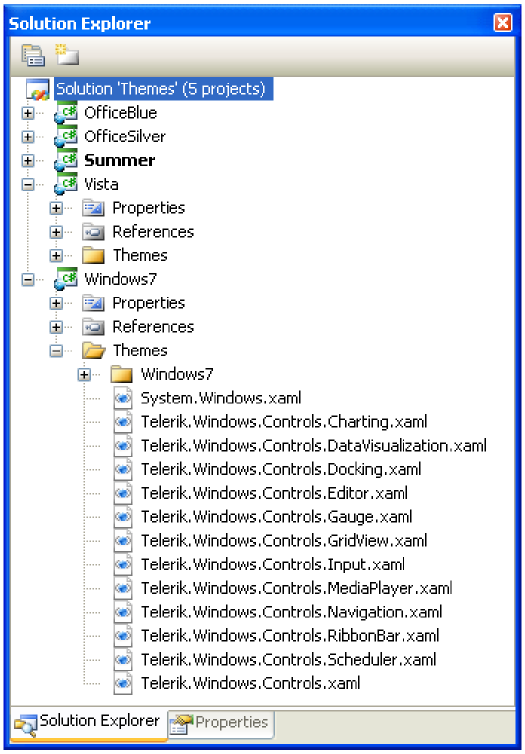
See the MSDN article "Resources Files" at http://msdn.microsoft.com/en-us/library/cc296240(VS.95).aspx for more information about accessing resources files and the syntax rules for building URIs to access resources.

Running in the browser, the button has a red background. The pattern used to create this simple theme can be used against any of the RadControls.



6.6 Modifying themes in Expression Blend

6.6.1 Overview



- Solution 'Themes' (5 projects)
  - OfficeBlue
  - OfficeSilver
  - **Summer**
  - Vista
    - Properties
    - References
    - Themes
  - Windows7
    - Properties
    - References
    - Themes
      - Windows7
      - System.Windows.xaml
      - Telerik.Windows.Controls.Charting.xaml
      - Telerik.Windows.Controls.DataVisualization.xaml
      - Telerik.Windows.Controls.Docking.xaml
      - Telerik.Windows.Controls.Editor.xaml
      - Telerik.Windows.Controls.Gauge.xaml
      - Telerik.Windows.Controls.GridView.xaml
      - Telerik.Windows.Controls.Input.xaml
      - Telerik.Windows.Controls.MediaPlayer.xaml
      - Telerik.Windows.Controls.Navigation.xaml
      - Telerik.Windows.Controls.RibbonBar.xaml
      - Telerik.Windows.Controls.Scheduler.xaml
      - Telerik.Windows.Controls.xaml

The RadControls for Silverlight installation comes with a "Themes" solution that contains a project for each theme assembly. Use these projects to edit existing themes within Visual Studio or Expression Blend.

Notice the project structure in the screenshot. Each theme has a Themes folder that contains a set of XAML files. There is a theme for every namespace being themed, e.g.Telerik.Windows.Controls.Navigation. These don't directly contain styles, but rather use MergedDictionary elements to include one or more XAML files with the actual styles for each group of controls. Inside the Themes folder is a directory named for the theme, e.g. "Windows7" that contain XAML files for each group of controls, e.g. "Button.xaml". "Button.xaml" contains a set of common brushes at the top of the file, and styles for each of the button related controls, i.e. RadButton, RadToggleButton, RadRadioButton, etc.

Using Expression Blend you can visually edit the brushes, resources and templates of the theme solution.

The steps to modifying a theme in Expression Blend are:

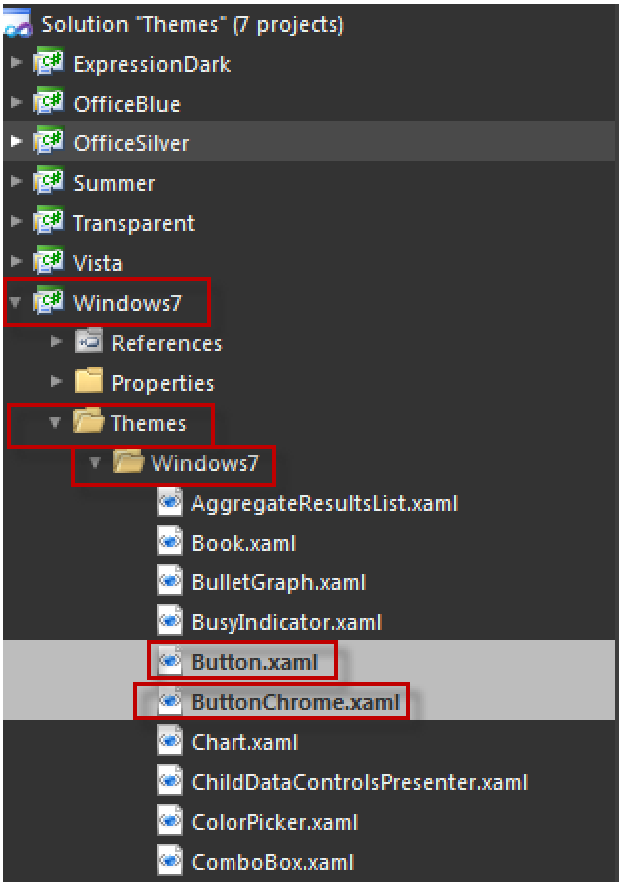
- Load the Themes solution into Expression Blend. Its a large file and may take some time.
- Also in Expression Blend, open one of the resource files and edit the styles, templates and brushes.
- In the Silverlight application to be themed, add a reference to the theme assembly.
- Add a StyleManager.Theme attribute and assign it the theme name.

6.6.2 Modifying the Theme Brushes

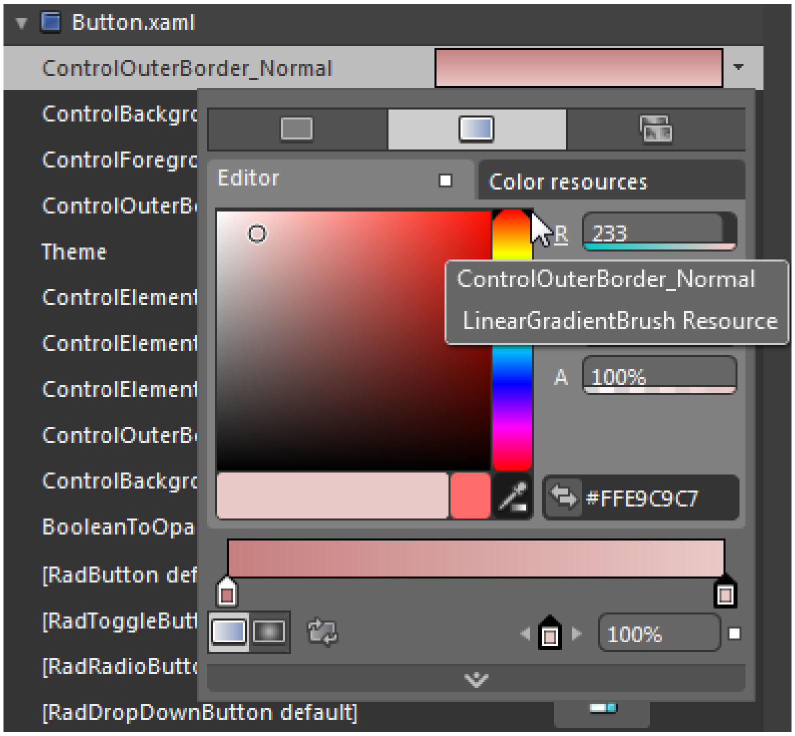
In this example we will modify the Windows7 theme in Expression Blend, changing the brushes for some of the button related controls including RadButton, RadRadioButton and RadToggleButton and RadDropDownButton.

Editing the Theme Project

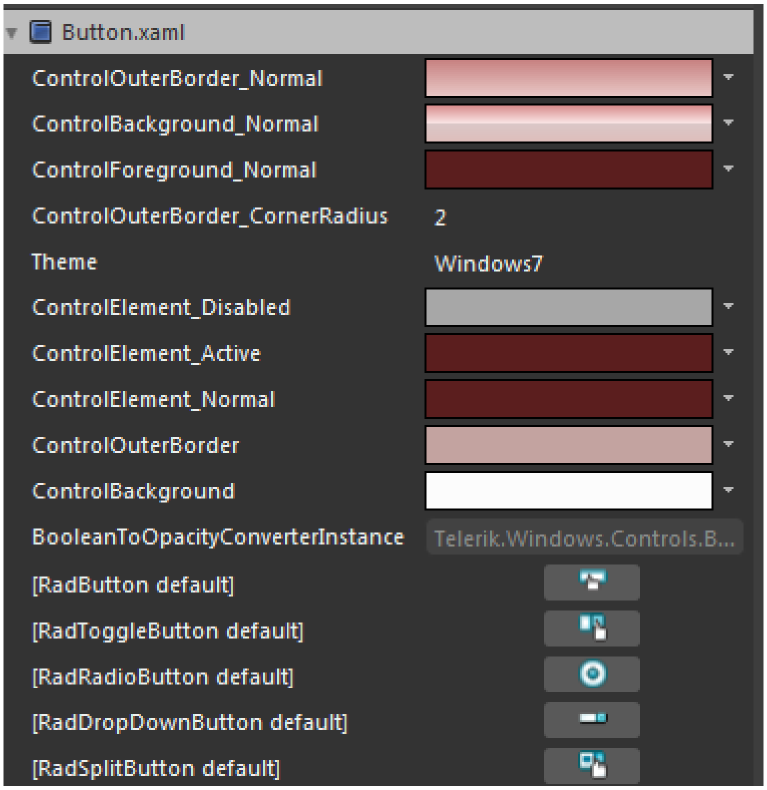
1) In Expression Blend, navigate to the RadControls for Silverlight install directory and locate the Themes directory. Load the Themes.sln solution.
2) In the Projects pane, locate the Windows7 project, \Themes\Windows7\Button.xaml file and double-click it.



3) In the Resources pane, the brushes for all of the buttons are listed. Use the drop down arrow for each brush and open the color editing popup. In the editor tab, make each of the colors some variant of a red shade by dragging the marker in the rainbow colored bar.



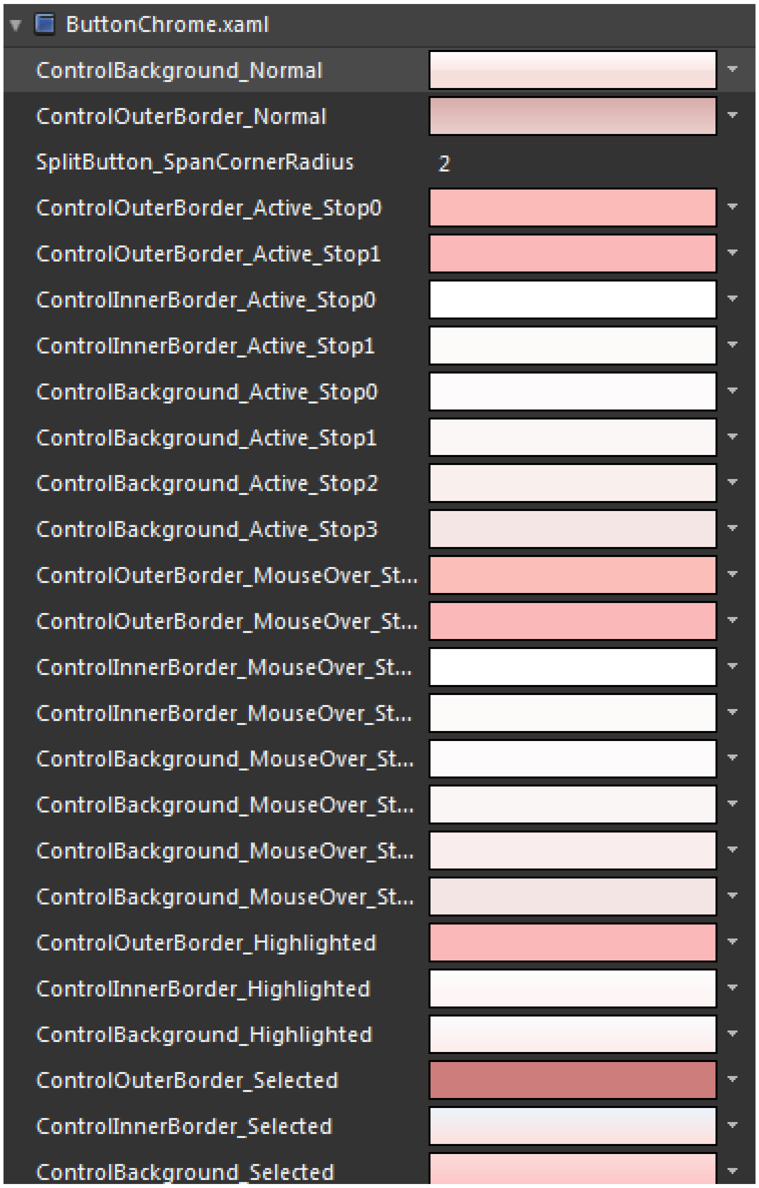
4) When you're done editing brushes, they should look something like the example below. ***Note**: the colors you choose here aren't significant, except that they should be very different from the shades of blue that we're starting with. We need to see that all of the colors have changed in the themed control.*



Button.xaml

ControlOuterBorder_Normal	🟥
ControlBackground_Normal	🟥
ControlForeground_Normal	🟥
ControlOuterBorder_CornerRadius	2
Theme	Windows7
ControlElement_Disabled	▨
ControlElement_Active	🟥
ControlElement_Normal	🟥
ControlOuterBorder	🟥
ControlBackground	⬜
BooleanToOpacityConverterInstance	Telerik.Windows.Controls.B...
[RadButton default]	
[RadToggleButton default]	
[RadRadioButton default]	
[RadDropDownButton default]	
[RadSplitButton default]	

5) Follow the same steps to modify the \Themes\Windows7\ButtonChrome.xaml file. When you're done editing the ButtonChrome brushes, they should look something like the example below:



ButtonChrome.xaml

ControlBackground_Normal

ControlOuterBorder_Normal

SplitButton_SpanCornerRadius 2

ControlOuterBorder_Active_Stop0

ControlOuterBorder_Active_Stop1

ControlInnerBorder_Active_Stop0

ControlInnerBorder_Active_Stop1

ControlBackground_Active_Stop0

ControlBackground_Active_Stop1

ControlBackground_Active_Stop2

ControlBackground_Active_Stop3

ControlOuterBorder_MouseOver_St...

ControlOuterBorder_MouseOver_St...

ControlInnerBorder_MouseOver_St...

ControlInnerBorder_MouseOver_St...

ControlBackground_MouseOver_St...

ControlBackground_MouseOver_St...

ControlBackground_MouseOver_St...

ControlBackground_MouseOver_St...

ControlOuterBorder_Highlighted

ControlInnerBorder_Highlighted

ControlBackground_Highlighted

ControlOuterBorder_Selected

ControlInnerBorder_Selected

ControlBackground_Selected

6) Save the project, then press **Ctrl-Shift-B** to build the project (or select Build from the Project menu).

6.6.3 Testing the Modified Theme

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

   a) **Telerik.Windows.Controls**

   b) **Telerik.Windows.Themes.Windows7**

   Important note! Be sure to choose your custom version of this assembly, not the version that ships with RadControls for Silverlight. You should be able to find this assembly in the \bin\debug directory of the Windows7 themes project.

XAML Editing

1) Open MainPage.xaml for editing.

2) Replace the main "LayoutRoot" Grid element with the XAML below.

```
<Grid x:Name="LayoutRoot">
  <StackPanel>
    <!--RadButton-->

    <!--RadToggleButton-->

    <!--RadRadioButton-->

    <!--RadDropDownButton-->
  </StackPanel>
</Grid>
```

3) Drag a **RadButton** from the Toolbox to a point just under the "<!--RadButton-->" comment.

4) Drag a **RadToggleButton**, **RadRadioButton** and **RadDropDownButton** to a point underneath their respective comments.

5) Set the attributes of all four buttons as follows:

   a) **Margin** = "10"

   b) **HorizontalAlignment** = "Left"

   c) **VerticalAlignment** = "Top"

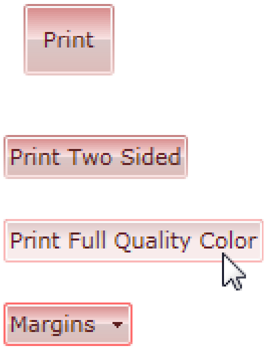
6) Set the **Content** attribute of each button to any text you choose.

7) Above the main "LayoutRoot" grid, add a UserControl.Resources element. Inside the resources, add a resource for the **Windows7Theme**, set the "x:Key" to "Windows7" and the **IsApplicationTheme** property to "True".

```
<UserControl.Resources>
  <telerik:Windows7Theme IsApplicationTheme="True"
      x:Key="Windows7" />
</UserControl.Resources>
```

Run The Application

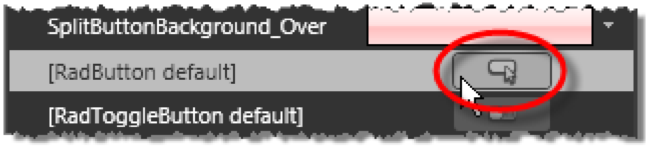
Press **F5** to run the application. The web page should look something like the screenshot below.



6.6.4 Modifying Theme Styles

Often, modifying theme brushes can make the color scheme of the controls compatible with application look-and-feel. But you don't need to stop there, you can also modify the entire style of a control, including changes to layout, adding effects and animation. For example, you can add a shadow effect to the RadButton Content and animate it so that it looks as if a light passed over the content text.

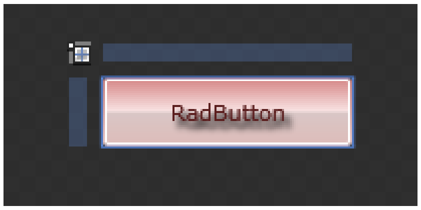
1) From the Expression Blend Resources pane, double-click the edit button next to the RadButton default style.



2) In the Objects and Timeline pane, right-click the "Style<>" item and select **Edit Template > Edit Current**. *From this point on you can simply edit the control by selecting elements of the control in the Objects and Timeline pane and Artboard.*

3) In the Objects and Timeline pane, expand the outline and locate the "Content" node.

4) Open the Assets pane and drag a **DropShadowEffect** to the "Content" node in the Objects and Timeline pane. The content in the button should now display a shadow as shown in the screenshot below.



5) Record animation for the MouseOver state

a) Open the States pane and click on the **MouseOver** state to begin state recording.

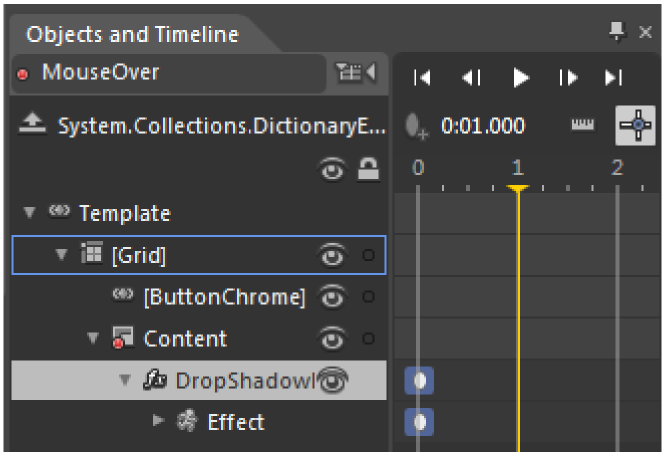
b) Click the **Show Timeline** button (▤)

c) Make sure the **DropShadowEffect** is selected

d) In the Properties pane, set the **ShadowDepth** property to "10".

e) Set the Timeline marker at the one second mark.

The Objects and Timeline pane will look something like the screenshot below.

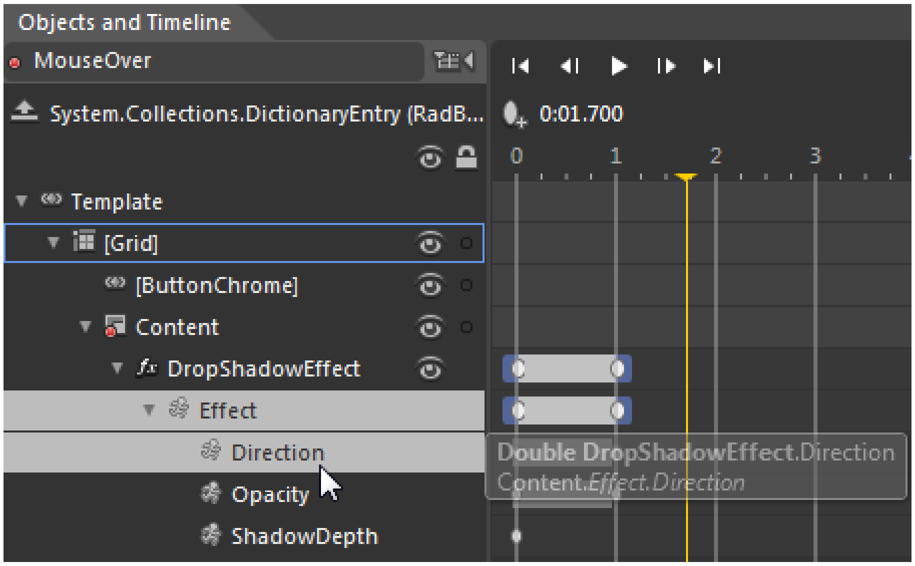


6) In the Properties pane, set both the **Direction** and **Opacity** properties to zero.

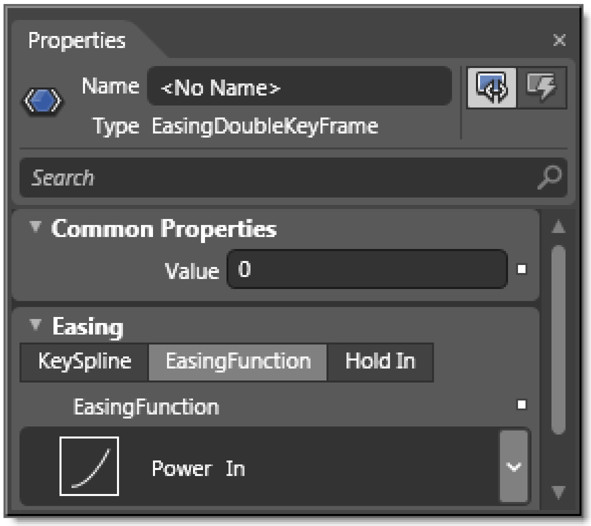
7) In the Objects and Timeline pane, click the play (▶) button to test the animation.

The animation shows the shadow moving evenly under the content. Next, to make the animation less mechanical, we can use an easing function to make the animation start gradually, then pick up speed.

8) In the Objects and Timeline pane, select the **Direction** property of the effect.



9) In the Properties pane, set the **EasingFunction** to "Power In".

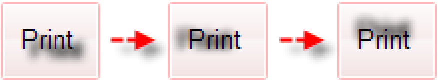


Name <No Name>
Type EasingDoubleKeyFrame
Value 0
EasingFunction
Power In

10)Select the **Opacity** property and set the **EasingFunction** to "Power In".

11)Build the theme assembly in Expression Blend using the menu **Project > Build Project** or press **Ctrl-Shift-B**.

12)Run the Visual Studio project that applies the theme and observe the new animated effects when the mouse passes over the button.



6.7 Wrap Up

In this chapter you learned how Telerik themes are used with RadControls and other Silverlight elements to create a uniform look-and-feel for your application. First you applied predefined themes to RadControls and observed how themes can be changed without otherwise altering target elements. You learned how themes are distributed and applied to elements in XAML and in the code behind. You applied themes to an entire application and to individual elements in the application. You will also created a custom theme and learned the syntax for applying the theme.

In the "Modifying Themes" section of the chapter you learned about the "Themes" solution and its structure. You modified theme brushes to skin the theme with a new color set. You applied the modified theme to the UI in a Visual Studio project. Finally, you expanded this example to add effects and animation to a RadButton control.

Part VII

Localization

7 Localization

7.1 Objectives

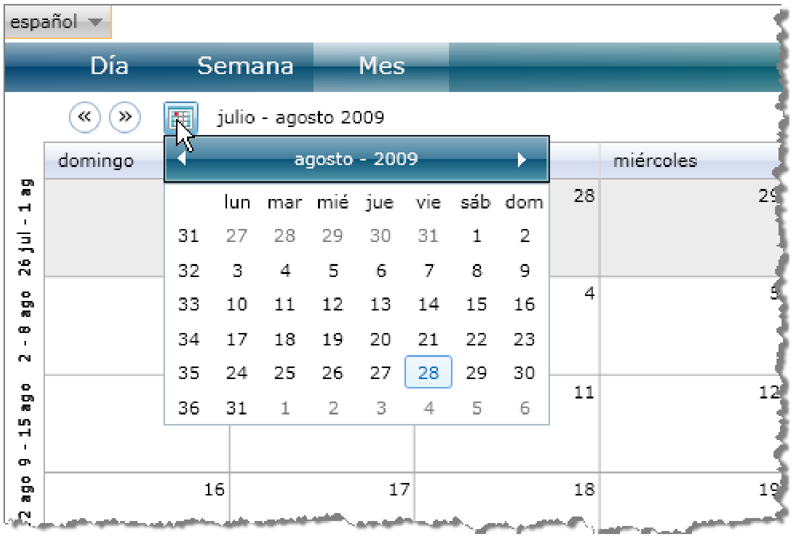
In this chapter you will learn how to localize your applications using the Telerik LocalizationManager class. You will learn how to retrieve translations from resource files and from custom storage. You will assign keys to translations in both XAML and in code.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Localization\Localization.sln

7.2 Overview

You may want your application to reflect some particular language, dialect or professional terminology. Localization is a way to present your software product in a given language or culture that doesn't require changes in the application itself. The translations are kept separate from application, allowing new translations to be added. The language of the application can be changed on-the-fly so that buttons, titles, column headings and other content will display in the appropriate language. The screenshot below shows RadScheduler localized for Spanish.



agosto - 2009

	lun	mar	mié	jue	vie	sáb	dom
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31	1	2	3	4	5	6

7.3 Getting Started

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

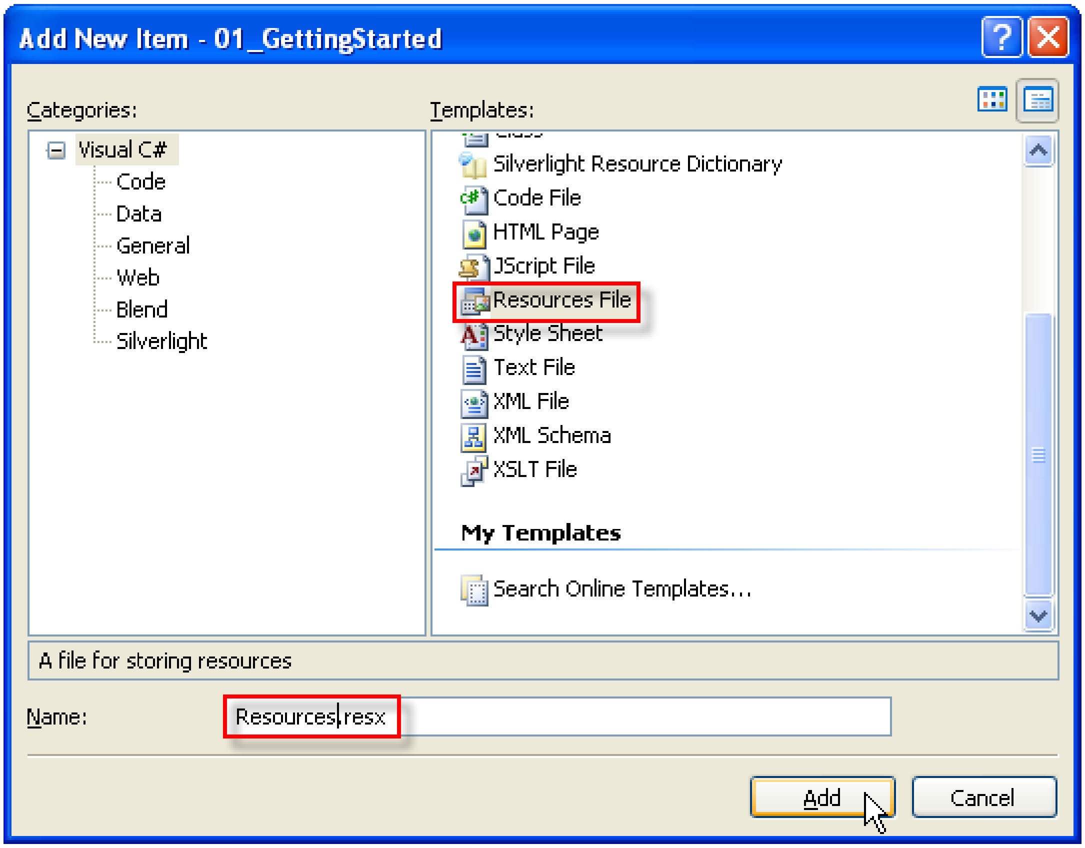
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) Telerik.Windows.Controls

b) Telerik.Windows.Controls.Input

c) Telerik.Windows.Themes.Windows7

4) In the Solution Explorer, right-click the Silverlight project and select **Add > New Item...** from the context menu. Select the "Resources File" template, name it "Resources.resx" and click the **OK** button.

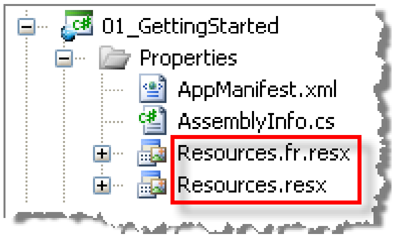


A file for storing resources

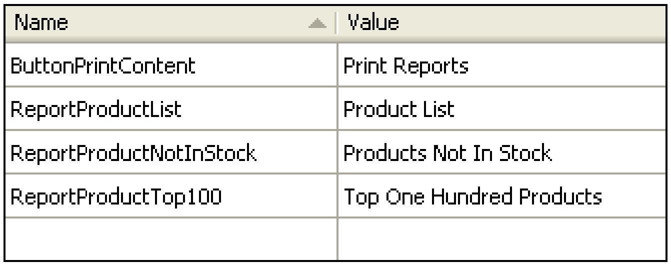
Name: Resources.resx

5) Add a second resource file to the project and name it Resources.fr.resx.

6) In the Solution Explorer, drag both files to the Properties folder.

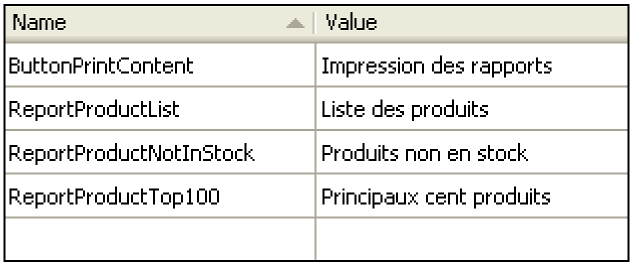


7) In the Solution Explorer, double-click Resources.resx and edit the file to include the following keys and values.



Name	Value
ButtonPrintContent	Print Reports
ReportProductList	Product List
ReportProductNotInStock	Products Not In Stock
ReportProductTop100	Top One Hundred Products

8) In the Solution Explorer, double-click Resources.fr.resx and edit the file to include the following keys and values.



Name	Value
ButtonPrintContent	Impression des rapports
ReportProductList	Liste des produits
ReportProductNotInStock	Produits non en stock
ReportProductTop100	Principaux cent produits

9) Open the project file for the Silverlight project (i.e. "myProject.csproj") in Notepad. Locate the SupportedCultures element and include a semi-colon delimited list of culture codes that you intend to localize for. In this walk through the languages are English and French i.e. "en;fr".

```
<SupportedCultures>en;fr</SupportedCultures>
```

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags.

```
<StackPanel Orientation="Horizontal">
  <!--Print Button-->
  <!--Reports List-->
</StackPanel>
```

3) Drag a **RadButton** from the Toolbox to a point just below the "<!--Print Button-->" comment. Add a **LocalizationManager.ResourceKey** attached property and assign it "ButtonPrintContent" (matching the resource key of the same name). Set the **VerticalAlignment** to "Top", **Margin** to "10" and **Padding** to "5".

XAML

```
<!--Print Button-->
<telerik:RadButton
      telerik:LocalizationManager.ResourceKey="ButtonPrintContent"
      VerticalAlignment="Top" Margin="10" Padding="5"></telerik:RadButton>
```

4) Drag a RadComboBox from the Toolbox to a point just below the "<!--Reports List-->" comment. Replace the element with the XAML below. Notice that each of the RadComboBoxItem elements has its own LocalizationManager.ResourceKey property defined. Each of the ResourceKey assignments exactly matches keys in the two resource files.

XAML

```
<!--Reports List-->
<telerik:RadComboBox VerticalAlignment="Top" Margin="10">
   <telerik:RadComboBoxItem
         telerik:LocalizationManager.ResourceKey="ReportProductList" />
   <telerik:RadComboBoxItem
         telerik:LocalizationManager.ResourceKey="ReportProductNotInStock" />
   <telerik:RadComboBoxItem
         telerik:LocalizationManager.ResourceKey="ReportProductTop100" />
</telerik:RadComboBox>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **Telerik.Windows.Controls**

2) Replace the constructor with the code below. *Notice that the DefaultResourceManager points to the Resources.ResourceManager. The file name "Resources" is arbitrary and can be whatever you like. The DefaultCulture is set to the French culture. All of this code needs to happen before the call to InitializeComponent().*

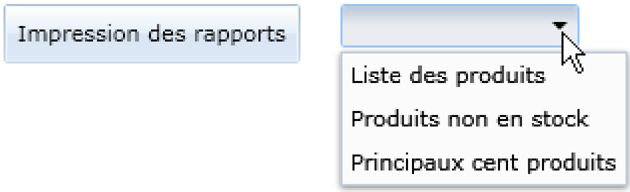
VB

```
Public Sub New()
  CType(New Windows7Theme(), Windows7Theme).IsApplicationTheme = True
  LocalizationManager.DefaultResourceManager = My.Resources.ResourceManager
  LocalizationManager.DefaultCulture = New CultureInfo("fr")

  InitializeComponent()
End Sub
```

```
public MainPage()
{
   new Windows7Theme().IsApplicationTheme = true;
   LocalizationManager.DefaultResourceManager =
      Properties.Resources.ResourceManager;
   LocalizationManager.DefaultCulture =
      new CultureInfo("fr");

   InitializeComponent();
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



7.4 Control Details

7.4.1 LocalizationManager

Telerik handles localization through the **LocalizationManager** class from the **Telerik.Windows.Controls** assembly. LocalizationManager can work with almost any storage and retrieval scenario and apply it to any Silverlight element. Currently, the LocalizationManager is designed to assign translations to Silverlight elements only, not to HTML that may be on the host page.

You can store your localized content in either resource files or in any custom storage (database, web service, isolated storage, etc). Translations are attached to Silverlight elements through keys that can be assigned in XAML or in code behind.

7.4.2 Resource File Storage

Configuring LocalizationManager to retrieve the correct resources involves assigning a **ResourceManager** to provide resources and a **CultureInfo** that points to a culture specific version of a resource file. You can get this done by creating a LocalizationManager instance and assigning the **Manager** and **Culture** properties or you can avoid creating an instance by assigning the static **DefaultResourceManager** and **DefaultCulture** properties. Both techniques are shown below.

VB

```
LocalizationManager.Manager = New LocalizationManager() With { _
.ResourceManager = My.Resources.ResourceManager,  _
.Culture = New CultureInfo("fr")}
'... or
LocalizationManager.DefaultResourceManager = My.Resources.ResourceManager
LocalizationManager.DefaultCulture = New CultureInfo("fr")
```

C#

```
LocalizationManager.Manager = new LocalizationManager()
{
   ResourceManager = Properties.Resources.ResourceManager,
   Culture = new CultureInfo("fr")
};
//... or
LocalizationManager.DefaultResourceManager = Properties.Resources.ResourceManager;
LocalizationManager.DefaultCulture = new CultureInfo("fr");
```

Notes

You can skip assigning the Culture or DefaultCulture properties and the current UI culture will be used.

If you use culture settings to determine the correct resource file, you must also modify a **SupportedCultures** element in your project file. To do this, open your project "*.csproj" file as a text file, locate the SupportedCultures element and include a semi-colon delimited list of culture codes that you intend to localize for. In the example below, languages are English, German and French.

```
<SupportedCultures>en;de;fr</SupportedCultures>
```

7.4.3 Custom Storage

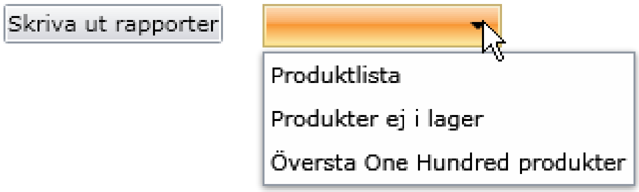
Resource files are effective for many uses, but you may need to store localization content in another medium such as a database or from a web service. For example you might want to leverage an existing database already containing a large number of translations. To introduce your own custom logic, descend from LocalizationManager and override the **GetStringOverride()** method. Use the "key" parameter passed in to determine the item being localized and return your translated value using any mechanism that suits your purpose. The example below simply returns Swedish strings for the given key, but you can use any custom storage and retrieval inside this method.

VB

```
Partial Public Class MainPage
  Inherits UserControl
  Public Sub New()
    LocalizationManager.Manager = New MyLocalizationManager()
    InitializeComponent()
  End Sub
End Class

Public Class MyLocalizationManager
  Inherits LocalizationManager
  Public Overrides Function GetStringOverride(ByVal key As String) As String
    ' your custom logic here
    Select Case key
      Case "ButtonPrintContent"
        Return "Skriva ut rapporter"
      Case "ReportProductList"
        Return "Produktlista"
      Case "ReportProductNotInStock"
        Return "Produkter ej i lager"
      Case "ReportProductTop100"
        Return "Översta One Hundred produkter"
    End Select
    Return MyBase.GetStringOverride(key)
  End Function
End Class
```

```
public partial class MainPage : UserControl
{
   public MainPage()
   {
      LocalizationManager.Manager = new MyLocalizationManager();
      InitializeComponent();
   }
}

public class MyLocalizationManager : LocalizationManager
{
   public override string GetStringOverride(string key)
   {
      // your custom logic here
      switch (key)
      {
         case "ButtonPrintContent": return "Skriva ut rapporter";
         case "ReportProductList": return "Produktlista";
         case "ReportProductNotInStock": return "Produkter ej i lager";
         case "ReportProductTop100": return "Översta One Hundred produkter";
      }
      return base.GetStringOverride(key);
   }
}
```

The running Silverlight application is identical to the example in "Getting Started" but with Swedish content.



Notes

You can call the GetStringOverride() method directly if you need a translated string for some other purpose than assigning to an element.

7.4.4 Assigning Resources to Elements

Use the attached **ResourceKey** property to identify the specific resource that should be applied to an element. The example below shows the XAML syntax where "telerik" is an XML namespace that points to **Telerik.Windows.Controls**.

```
<telerik:RadButton telerik:LocalizationManager.ResourceKey="ButtonPrintContent" />
```

In code, use the **SetResourceKey()** method and pass the element to be localized and the key name. The example below is identical to the previous XAML-only example.

VB

```
Public Sub New()
  InitializeComponent()
  LocalizationManager.SetResourceKey(btnPrint, "ButtonPrintContent")
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();
   LocalizationManager.SetResourceKey(btnPrint, "ButtonPrintContent");
}
```

The mechanism works against any Silverlight element, not just RadControls. For example, you could localize a window title, a standard button or a tool tip.

XAML

```
<Button>
   <ToolTipService.ToolTip>
      <ToolTip telerik:LocalizationManager.ResourceKey="MyKey" />
   </ToolTipService.ToolTip>
</Button>
```

7.4.5 Using Predefined Localization

Several of the RadControls are localized, right out of the box including RadUpload, RadTreeView, RadMediaPlayer, RadColorSelector, RadColorPicker, RadScheduler and RadGridView.

7.5 Wrap Up

In this chapter you learned how to localize your applications using the Telerik LocalizationManager class. You learned how to retrieve translations from resource files and using any custom storage mechanism. You also learned how to assign translation keys in both XAML and in code.

Part VIII

UI Automation Support

8 UI Automation Support

8.1 Objectives

In this chapter you will learn how RadControls for Silverlight provides UI Automation Support. First you will see how the Microsoft UI Automation Verify tool is used to locate RadControls elements and spelunk the element's properties and events. You will use AutomationProperties to support accessible applications in a Silverlight application. You will also learn how to use AutomationPeer descendents to automate RadControl elements.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\UI_Automation\UI_Automation.sln

8.2 Overview

Applications that speak to the widest possible audience need to allow their user interface (UI) elements to be automated. Making your applications accessible means thinking outside the visible rectangle to include other means of communication. Your application may be read and interacted with using alternatives such as Braille output, screen readers, head tracking or sip & puff devices to name a few.

Microsoft UI Automation (UIA) allows programmatic access to user interface elements that enable accessibility tools, such as screen readers, to provide information to end users and manipulate the UI without requiring the use of standard input devices. UIA masks differences between frameworks, exposing every piece of UI to client applications. UIA also allows automated test scripts to interact with the UI.

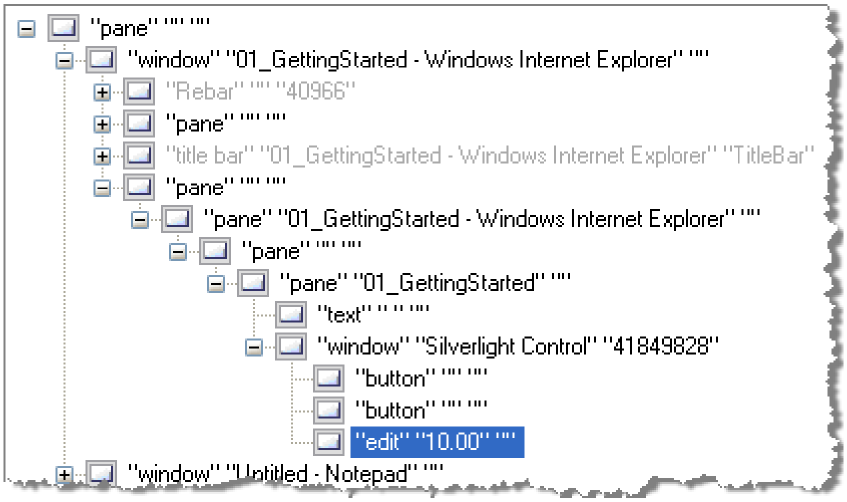
RadControls for Silverlight has excellent UI Automation Support that enables deep UI testing and control from accessibility tools. We can see the support in action using the UI Automation Verify tool from Microsoft. The tool and associated resources can be downloaded at:

http://www.codeplex.com/UIAutomationVerify

Lets use the tool to tour a simple application that contains a single RadNumericUpDown control. The XAML below shows that the current Value property is "10" that the Minimum and Maximum are "1" and "100".

```
<Grid x:Name="LayoutRoot">
  <telerik:RadNumericUpDown x:Name="myUpdownControl" Value="10"
      Minimum="1" Maximum="100" HorizontalAlignment="Left"
      VerticalAlignment="Top" Margin="20" ></telerik:RadNumericUpDown>
</Grid>
```

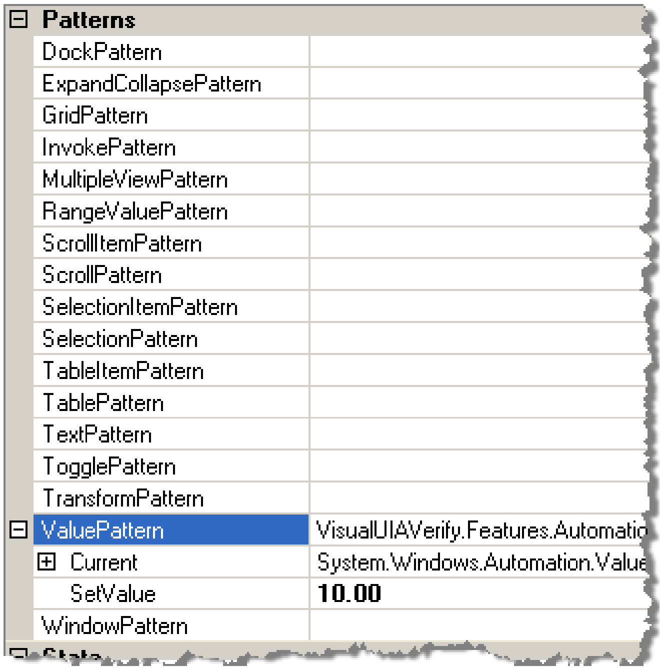
When you first run the tool, you can set the **Mode > Always On Top** menu option to make it easier to view the browser and the tool at the same time. Also turn on **Mode > Focus Tracking** so that the tool will automatically find elements as they get focus. After you run the Silverlight application, simply click on it. The focus rectangle generated by the tool will let you know what element is being examined. When you click on the edit box of the RadNumericUpDown, the "Automation Elements Tree" navigates to the edit control.



```
⊟ "pane" "" ""
    ⊟ "window" "01_GettingStarted - Windows Internet Explorer" ""
        ⊞ "Rebar" "" "40966"
        ⊞ "pane" "" ""
        ⊞ "title bar" "01_GettingStarted - Windows Internet Explorer" "TitleBar"
        ⊟ "pane" "" ""
            ⊟ "pane" "01_GettingStarted - Windows Internet Explorer" ""
                ⊟ "pane" "" ""
                    ⊟ "pane" "01_GettingStarted" ""
                            "text" " " ""
                        ⊟ "window" "Silverlight Control" "41849828"
                                "button" "" ""
                                "button" "" ""
                                "edit" "10.00" ""
    ⊞ "window" "Untitled - Notepad" ""
```

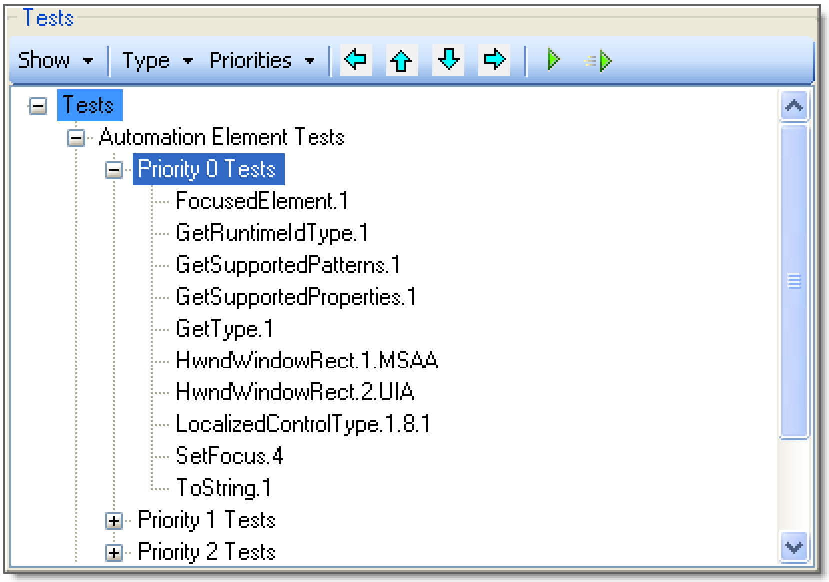
The "Properties" will display useful information about the selected element including **General Accessibility** (accelerator keys, help text, etc), **Identification** properties (hWnd, AutomationId, ClassName and ControlType), **Patterns** to expose control functionality independent of the control type or appearance (TextPattern, RangeValuePattern, ValuePattern, etc.), **State** (HasKeyBoardFocus, IsEnabled) and **Visibility** (BoundingRectangle, IsOffScreen).

You can directly interact with elements using patterns. If you edit the SetValue of the ValuePattern, the value in the Silverlight application is modified to match.



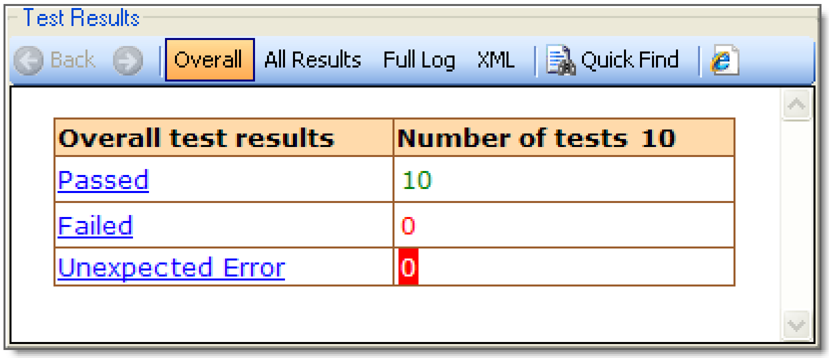
⊟ Patterns	
DockPattern	
ExpandCollapsePattern	
GridPattern	
InvokePattern	
MultipleViewPattern	
RangeValuePattern	
ScrollItemPattern	
ScrollPattern	
SelectionItemPattern	
SelectionPattern	
TableItemPattern	
TablePattern	
TextPattern	
TogglePattern	
TransformPattern	
⊟ ValuePattern	VisualUIAVerify.Features.Automatio
⊞ Current	System.Windows.Automation.Value
SetValue	**10.00**
WindowPattern	
⊟ State	

The UI Automation tool also includes a set of generic tests that can be run against the selected element or against an element and all of its children. These are fairly basic tests and aren't designed to test the functionality of your application, but instead check on low level element functionality, e.g. can we set focus to the control.



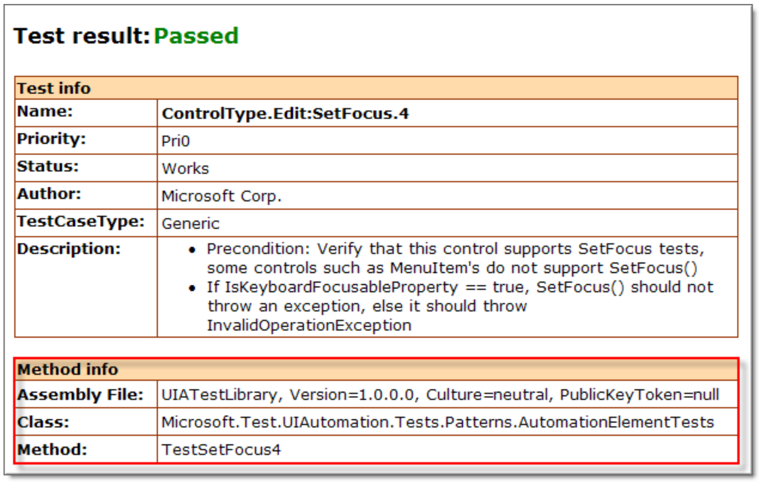
- Tests
  - Automation Element Tests
    - Priority 0 Tests
      - FocusedElement.1
      - GetRuntimeIdType.1
      - GetSupportedPatterns.1
      - GetSupportedProperties.1
      - GetType.1
      - HwndWindowRect.1.MSAA
      - HwndWindowRect.2.UIA
      - LocalizedControlType.1.8.1
      - SetFocus.4
      - ToString.1
    - Priority 1 Tests
    - Priority 2 Tests

The Test Results pane publishes a summary of the results and allows you to output the log of results as XML.



Overall test results	Number of tests 10
Passed	10
Failed	0
Unexpected Error	0

If you follow the links and drill into the test detail you can see what classes and methods were used to run the tests.



Test result: Passed

Test info	
Name:	ControlType.Edit:SetFocus.4
Priority:	Pri0
Status:	Works
Author:	Microsoft Corp.
TestCaseType:	Generic
Description:	• Precondition: Verify that this control supports SetFocus tests, some controls such as MenuItem's do not support SetFocus() • If IsKeyboardFocusableProperty == true, SetFocus() should not throw an exception, else it should throw InvalidOperationException

Method info	
Assembly File:	UIATestLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Class:	Microsoft.Test.UIAutomation.Tests.Patterns.AutomationElementTests
Method:	TestSetFocus4

.

8.3 Getting Started

Much of the automation support is built-in and doesn't require extra work on your part. But you can provide richer semantics that span a wider range of interaction using the static **AutomationProperties** class. With AutomationProperties you can set:

- **Name**: The name of the element. A screen reader will typically read off this name.
- **HelpText**: This text is semantically equivalent to a tool tip.
- **AcceleratorKey**/**AccessKey**: These properties are both strings that indicate keyboard shortcuts, such as "CTRL-C" for a copy operation.
- **IsRequiredForForm**: When true indicates that the element is required.
- **ItemStatus**: This property is used to get information about an item in a list, tree view, grid or other collection container. ItemStatus is string that indicates the elements status, for example an item being retrieved from a database might read "Loading...".
- **ItemType**: This property is used to get information about an item in a list, tree view, grid or other collection container. ItemType is a string that describes the nature of the item. For example, in a file list ItemType might be "document" or "folder".
- **LabeledBy**: This property points to an element that is the text label for the element being described.

The walk through below displays a RadComboBox where both the combo box and its items are decorated with AutomationProperties.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Input**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags.

```
<StackPanel>
  <!--TextBlock "label"-->
  <!--RadComboBox with list of print choices-->
</StackPanel>
```

3) Replace the comment "<!--TextBlock 'label'-->" with the XAML below. *The TextBlock has an added AutomationProperties.Name attached property.*

```
<!--TextBlock "label"-->
<TextBlock AutomationProperties.Name="tbPrintComboBox" Text="Print Choices:" />
```

4) Drag a RadComboBox from the Toolbox to a point underneath the "<!--*RadComboBox with list of print choices*-->" comment. Replace the RadComboBox element with the XAML below. *The RadComboBox has AutomationProperties defined for the Name and HelpText. Also notice that a SelectionChanged event handler is included that we will code later.*

```
<!--RadComboBox with list of print choices-->
<telerik:RadComboBox x:Name="cbPrintChoices"
      AutomationProperties.Name="PrintComboBox"
      AutomationProperties.HelpText="Print Choices"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      SelectionChanged="RadComboBox_SelectionChanged">

</telerik:RadComboBox>
```

5) Add three RadComboBoxItem elements inside the RadComboBox element begin and end tags. Each item will contain the AutomationProperties HelpText and AcceleratorKey.

```
<telerik:RadComboBoxItem Content="Fast"
      AutomationProperties.HelpText="Fast Economical Printing"
      AutomationProperties.AcceleratorKey="F" />
<telerik:RadComboBoxItem Content="Green"
      AutomationProperties.HelpText="Paper Saving Printing"
      AutomationProperties.AcceleratorKey="G" />
<telerik:RadComboBoxItem Content="Photo"
      AutomationProperties.HelpText="Photo Quality Printing"
      AutomationProperties.AcceleratorKey="P" />
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **Telerik.Windows.Controls**

b) **System.Windows.Automation**

2) In the constructor for the UserControl, add a line of code below the InitializeComponent() method call to set the "LabeledBy" element. *The first parameter is the element being labeled and the second, the label element.*

VB

```
Public Sub New()
  InitializeComponent()

  AutomationProperties.SetLabeledBy(Me.cbPrintChoices, Me.tbPrintComboBox)
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();

   AutomationProperties.SetLabeledBy(this.cbPrintChoices,
      this.tbPrintComboBox);
}
```

3) Handle the RadComboBox SelectionChanged event to react as items are selected.

VB

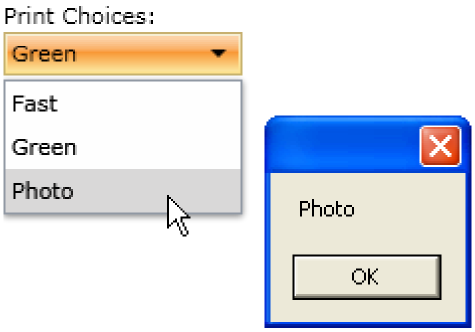
```
Private Sub RadComboBox_SelectionChanged( _
ByVal sender As Object, ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  If e.AddedItems.Count > 0 Then
    Dim text As String = (TryCast(e.AddedItems(0), RadComboBoxItem)).Content.ToString()
    MessageBox.Show(text)
  End If
End Sub
```

C#

```
private void RadComboBox_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   if (e.AddedItems.Count > 0)
   {
      string text =
         (e.AddedItems[0] as RadComboBoxItem).Content.ToString();
      MessageBox.Show(text);
   }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



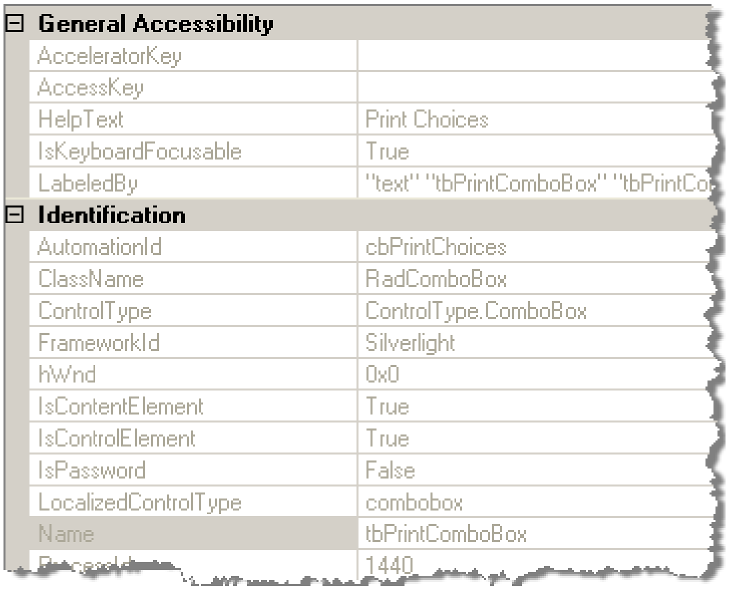
Test Application Features

1) Try pressing the accelerator keys "F", "G" and "P". The corresponding items should be selected.

Ideas for Extending This Example

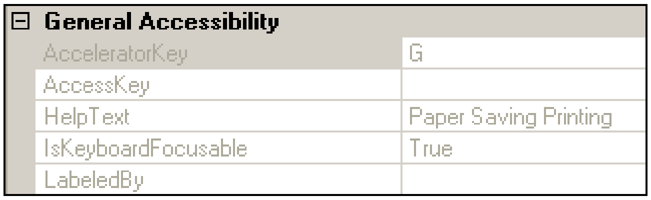
If you download the "UI Automation Verify Tool", run it and locate the RadComboBox, there's quite a bit of information available now. We can see in the "General Accessibility" section that the HelpText is populated and that LabeledBy points back to the text of the TextBlock above the combo box.

The Identification section has the automation Name, the ClassName "RadComboBox", ControlType and FrameworkId of "Silverlight". Notice that the IsContentElement (Typically FrameworkElement descendant) and IsControlElement (Control descendant) properties are both set True.



⊟ General Accessibility	
AcceleratorKey	
AccessKey	
HelpText	Print Choices
IsKeyboardFocusable	True
LabeledBy	"text" "tbPrintComboBox" "tbPrintCo
⊟ Identification	
AutomationId	cbPrintChoices
ClassName	RadComboBox
ControlType	ControlType.ComboBox
FrameworkId	Silverlight
hWnd	0x0
IsContentElement	True
IsControlElement	True
IsPassword	False
LocalizedControlType	combobox
Name	tbPrintComboBox
ProcessId	1440

If you navigate to one of the RadComboBoxItem elements, the UI Automation Verify Tool displays the AcceleratorKey and HelpText.



⊟ General Accessibility	
AcceleratorKey	G
AccessKey	
HelpText	Paper Saving Printing
IsKeyboardFocusable	True
LabeledBy	

8.4 Automating

In addition to using the AutomationProperties, you can use the descendents of the abstract **AutomationPeer** class to actually automate Silverlight elements. AutomationPeer has a number of properties and methods of interest, such as how to focus on an element, how to get AutomationProperties, how to raise automation events, execute patterns and get references to parent and child elements. **Note**: This would not be your first stop for things like setting values or opening a combo box. You would use the control's API for that. These methods are intended for applications that require generalized automation functionality such as accessibility and testing applications.

The example below builds off the Getting Started example and adds a single button that automates the RadComboBox. To do this, you can use the **RadComboBoxAutomationPeer CreatePeerForElement()** method and pass the RadComboBox element you wish to automate. As you might guess, there are AutomationPeer descendents implemented for other Silverlight elements and for RadControls elements. This particular peer has Expand() and Collapse() methods that can be called.

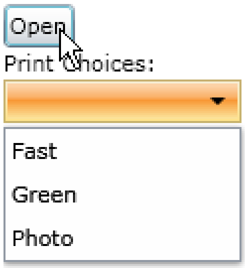
VB

```
Private Sub btnOpen_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim peer As RadComboBoxAutomationPeer = _
TryCast(RadComboBoxAutomationPeer.CreatePeerForElement(cbPrintChoices), _
RadComboBoxAutomationPeer)
  peer.Expand()
End Sub
```

C#

```
private void btnOpen_Click(object sender, RoutedEventArgs e)
{
   IExpandCollapseProvider peer =
      RadComboBoxAutomationPeer.CreatePeerForElement(cbPrintChoices)
        as IExpandCollapseProvider;
   peer.Expand();
}
```

The running Silverlight application looks like the screenshot below:



8.5 Wrap Up

In this chapter you learned how RadControls for Silverlight provides UI Automation Support. First you saw how the Microsoft UI Automation Verify tool is used to locate RadControls elements and spelunk the automation information. In a standard Windows application you used AutomationProperties to support accessible applications. You will also learned how to use AutomationPeer descendents to automate RadControl elements.

Part IX

Input Controls

9 Input Controls

9.1 Objectives

This chapter introduces controls for gathering user input, including a masked text box for restricting entry to predefined patterns, a set of "UpDown" controls for entering numeric or other input through repeater buttons, a set of color picking controls for selecting from a predefined palette of colors and a slider control for choosing values within a range.

You'll learn how the masked text box handles numeric, date/time and developer-defined formats using predefined masks or custom masks. We'll look at the important events for retrieving the user entered value and how to retrieve the value combined together with the mask. You will also see how to localize the masked text box.

In the section on "UpDown" controls you'll see how the numeric up-down control inherits from the RangeBase class. You'll learn the key common properties and events for both controls. We will discuss the key numeric up-down control ValueFormat property and some special case issues such as scrolling through integers. You will learn how to format numeric entry for data that uses custom units.

We will review three different types of color pickers along with their common properties and events. We will make use of the Silverlight Toolkit and the ObjectCollection class to define a custom palette of colors in XAML and will also define a custom palette in code. You'll learn how to customize text labels for elements of a color choosing control and how to hide specific elements of these controls.

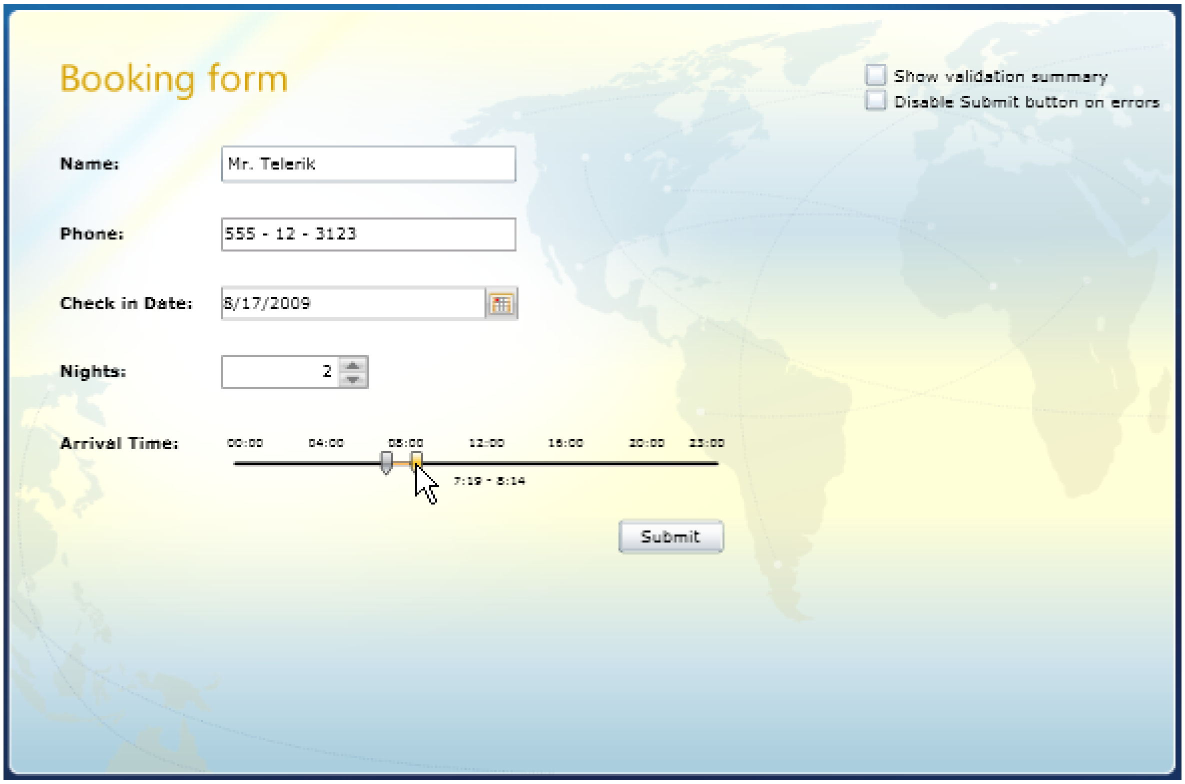
The last part of this chapter will deal with how the slider control is used to pick a value within a range or a range within a range. We will talk about the basic structure and parts of a slider, the slider's default behavior, how to retrieve new and previous values and how to control selection ranges. We will pay particular attention to how tick marks are placed along the slider and how to control tick mark frequency and visibility. We will delve a little deeper into how templates allow fine-tune control over each tick mark, how to use binding in templates to show data in a tick mark and finally how IValueConverter is used for highly customized binding scenarios.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Input\Input.sln.

9.2 Overview

Input controls make it easier for your end user to provide the right information to your application. RadMaskedTextBox restricts entry to date time, numeric and customized patterns. RadSlider allows selection from a defined range using an intuitive interface with one or two slider buttons. The up-down controls facilities paging through numbers or any custom series of values. Using a color picker the user can select from a configurable palette of colors.



Booking form

□ Show validation summary
□ Disable Submit button on errors

Name: Mr. Telerik

Phone: 555 - 12 - 3123

Check in Date: 8/17/2009

Nights: 2

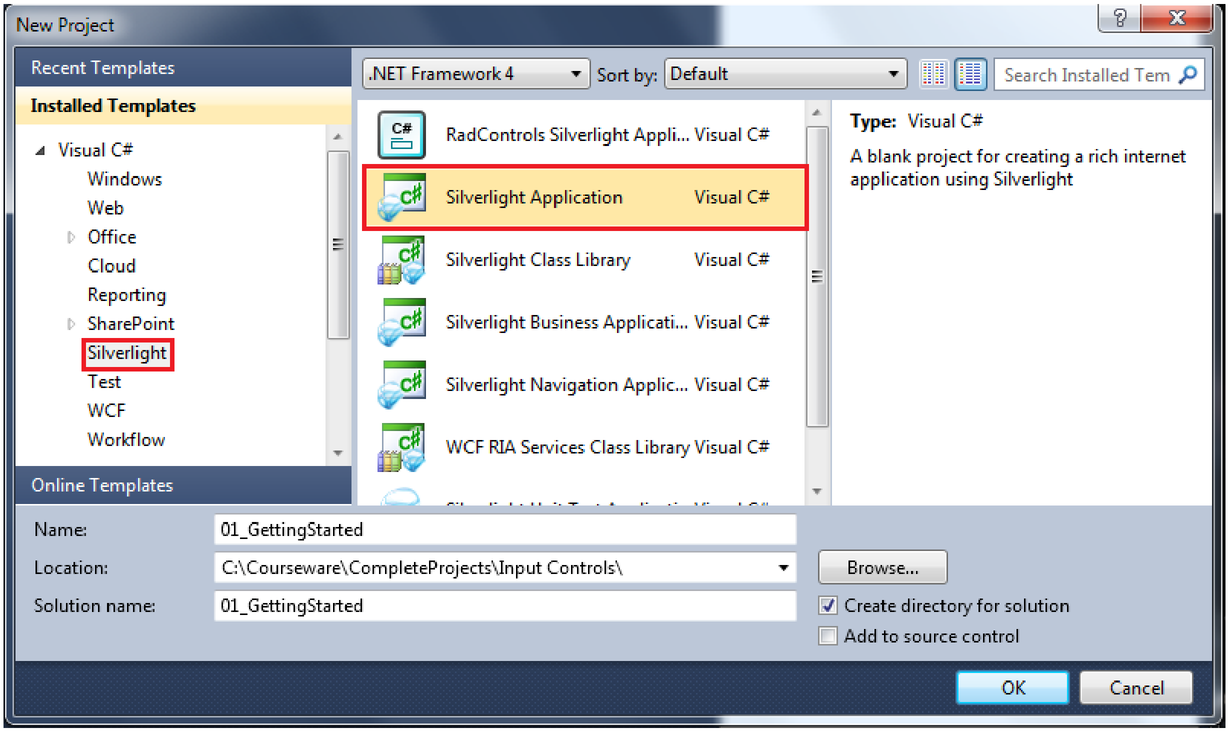
Arrival Time: 7:19 - 8:14

9.3 Getting Started

In this lab you will use a sampling of each input control type with some basic settings, then react to changing values using the primary events for each control. You will also apply a predefined theme to each of the RadControls: **RadMaskedTextBox**, **RadColorPicker**, **RadNumericUpDown** and **RadSlider**. Later in this chapter we will show a wider range of functionality available for each control.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the **Silverlight** project type, then select the **Silverlight Application** template. Provide a unique name for the project and click the **OK** button.


New Project
Recent Templates
Installed Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Online Templates
.NET Framework 4
Sort by: Default
Search Installed Tem
RadControls Silverlight Appli... Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
Silverlight Business Applicati... Visual C#
Silverlight Navigation Applic... Visual C#
WCF RIA Services Class Library Visual C#
Browse...
OK
Cancel

Type: Visual C#

A blank project for creating a rich internet application using Silverlight

Name: 01_GettingStarted

Location: C:\Courseware\CompleteProjects\Input Controls\

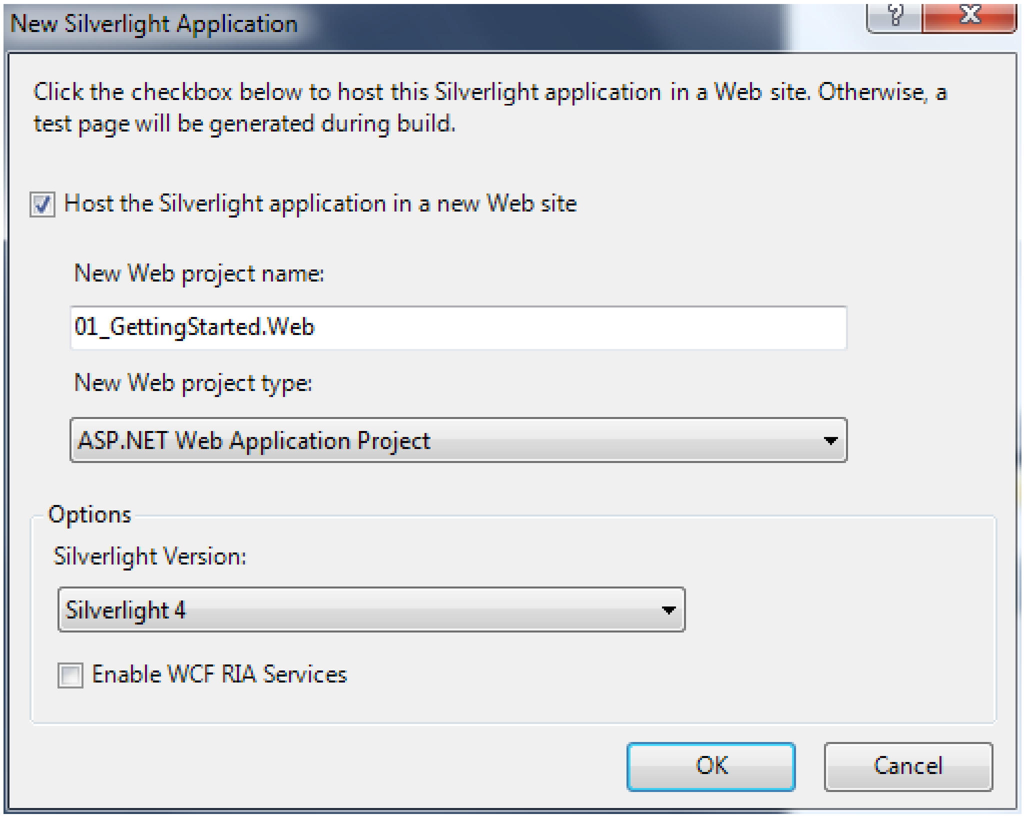
Solution name: 01_GettingStarted

☑ Create directory for solution

□ Add to source control

Figure 1

2) In the "New Silverlight Application" dialog select "Host the Silverlight application in a new Web site", give the project a unique name and select the "ASP.NET Web Application Project" New Web Project Type. Click **OK** to close the dialog and create the project.



Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new Web site

New Web project name:

01_GettingStarted.Web

New Web project type:

ASP.NET Web Application Project

Options

Silverlight Version:

Silverlight 4

□ Enable WCF RIA Services

Figure 2

XAML Editing

1) Use the Solution Explorer to open the MainPage.xaml file for editing.

2) Replace the <Grid> tag named "LayoutRoot" with the XAML markup below. *This will setup a basic grid with five rows and three columns. XML comments such as "<!--Title-->" will mark where you can paste additional XAML markup in later steps.*

```
<Grid
   x:Name="LayoutRoot"
   Background="White"
   Margin="20">
   <Grid.RowDefinitions>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
      <RowDefinition></RowDefinition>
   </Grid.RowDefinitions>
   <Grid.ColumnDefinitions>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
   </Grid.ColumnDefinitions>

   <!--Title-->


   <!--Masked Edit Box-->


   <!--Numeric Up Down-->


   <!--Color Picker-->


   <!--Slider-->

</Grid>
```

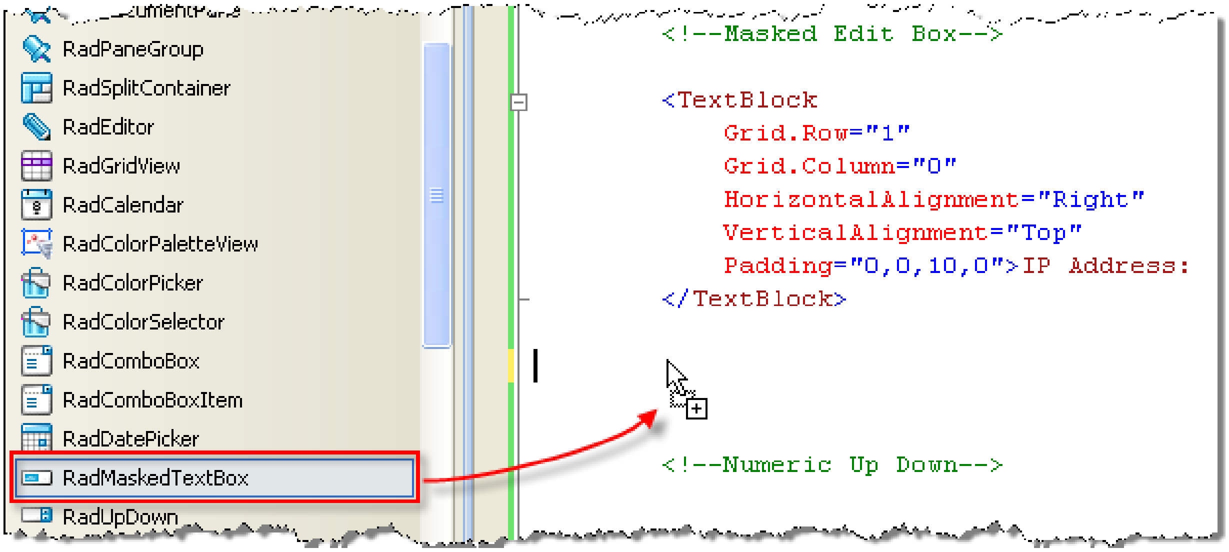
3) Replace the XAML comment "<!--Title-->" with the markup below. *This step will add a TextBlock control that displays the title "Preferences" at the top of the page. Notice the Grid.Row and Grid.Column that define where the TextBlock should be displayed within the grid.*

```
<!--Title-->

<TextBlock
   Grid.Row="0"
   Grid.Column="0"
   HorizontalAlignment="Left"
   VerticalAlignment="Top"
   FontWeight="Bold"
   Margin="5"
   MinHeight="25">
      Preferences
</TextBlock>
```

4) Replace the XAML comment "<!--Masked Edit Box-->" with the markup below. *This step will add a TextBlock with text "IP Address".*

```
<!--Masked Edit Box-->

<TextBlock
   Grid.Row="1"
   Grid.Column="0"
   HorizontalAlignment="Right"
   VerticalAlignment="Top"
   Padding="0,0,10,0">
IP Address:
</TextBlock>
```

5) Drag a **RadMaskedTextBox** control from the Toolbox to the XAML, just below the <TextBlock> tag defined in the previous step. *This step will include a xml name space reference to the assembly containing RadMaskedTextBox.*



```
<!--Masked Edit Box-->

<TextBlock
    Grid.Row="1"
    Grid.Column="0"
    HorizontalAlignment="Right"
    VerticalAlignment="Top"
    Padding="0,0,10,0">IP Address:
</TextBlock>



<!--Numeric Up Down-->
```

Figure 3

6) Replace the RadMaskedTextBox tag with the XAML markup below. *This step defines properties for the RadMaskedTextBox. The most important properties here are "Mask" that determines what kind of input will be accepted and "MaskType" that defines how the mask is interpreted. This particular mask will allow twelve numeric characters separated by literal "." characters, e.g. "1234.5678.9012.1234". Also notice that we've supplied a "Name" attribute so we can refer to the RadMaskedTextBox in code.*

```
<telerik:RadMaskedTextBox
   Grid.Row="1"
   Grid.Column="1"
   HorizontalAlignment="Left"
   VerticalAlignment="Top"
   Name="tbIP"
   Mask="###.###.###.###"
   MaskType="Standard">
</telerik:RadMaskedTextBox>
```

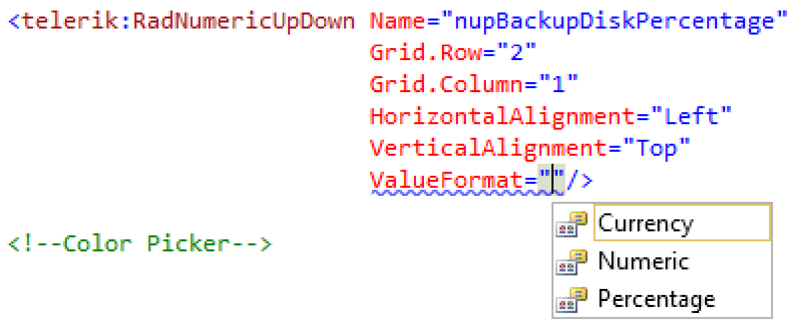
7) Below the RadMaskedTextBox add the TextBlock tag below. *This TextBlock will be used later to display new values as the user types into the RadMaskedTextBox.*

```
<TextBlock
   Name="tbNewIP"
   Margin="10,0,0,0"
   Grid.Row="1"
   Grid.Column="2"
   HorizontalAlignment="Left"
   VerticalAlignment="Top">
</TextBlock>
```

8) Replace the comment "<!--Numeric Up Down-->" with the XAML below. *This step will add a TextBlock that displays a title "Backup Disk Percentage" and will appear just above a RadNumericUpDown control.*

```
<TextBlock
   Grid.Row="2"
   Grid.Column="0"
   HorizontalAlignment="Right"
   VerticalAlignment="Top"
   Padding="0,0,10,0">
   Backup Disk Percentage:
</TextBlock>
```

9) Drag a **RadNumericUpDown** control from the Toolbox to the XAML, just below the TextBlock defined in the previous step. Set the **Name** attribute to "nupBackupDiskPercentage", **Grid.Row** ="2", **Grid.Column**="1", **HorizontalAlignment**="Left" and **VerticalAlignment**="Top". Add a **ValueFormat** attribute and notice the IntelliSense list of valid values when you first type the equal sign ("="). Select "Percentage" from the list.



```
<telerik:RadNumericUpDown Name="nupBackupDiskPercentage"
                          Grid.Row="2"
                          Grid.Column="1"
                          HorizontalAlignment="Left"
                          VerticalAlignment="Top"
                          ValueFormat=""/>

<!--Color Picker-->
```

Figure 4

10)Below the RadNumericUpDown, add the TextBlock tag shown below. *The TextBlock will be used to display value changes in the RadNumericUpDown control.*

```
<TextBlock
   Name="tbNewPercentage"
   Margin="10,0,0,0"
   Grid.Row="2"
   Grid.Column="2"
   HorizontalAlignment="Left"
   VerticalAlignment="Top">
</TextBlock>
```

11)Replace the "<!--Color Picker-->" comment with the XAML shown below. *This step creates a TextBlock with text "Font Color:" that will appear above a RadColorPicker control.*

```
<!--Color Picker-->

<TextBlock
   Grid.Row="3"
   Grid.Column="0"
   HorizontalAlignment="Right"
   VerticalAlignment="Top"
   Padding="0,0,10,0">
      Font Color:
</TextBlock>
```

12)Drag a **RadColorPicker** control from the Toolbox to the XAML, just below the TextBlock defined in the previous step. Set the **Name** attribute to "cpFont", **Grid.Row** = "3", **Grid.Column** = "1" and **HorizontalAlignment** = "Left".

13)Add an <Ellipse> tag below the RadColorPicker. *The ellipse will display the color selected from the RadColorPicker.*

```
<Ellipse
   Name="elNewColor"
   Margin="10,0,0,0"
   Grid.Row="3"
   Grid.Column="2"
   HorizontalAlignment="Left"
   VerticalAlignment="Top"
   Width="20"
   Height="20">
</Ellipse>
```

14)Replace the comment "<!--Slider-->" with the TextBlock defined below. *The TextBlock will add a label "Volume Range:" above a RadSlider control.*

```
<!--Slider-->

<TextBlock
   Grid.Row="4"
   Grid.Column="0"
   HorizontalAlignment="Right"
   VerticalAlignment="Top"
   Padding="0,0,10,0">
      Volume Range:
</TextBlock>
```

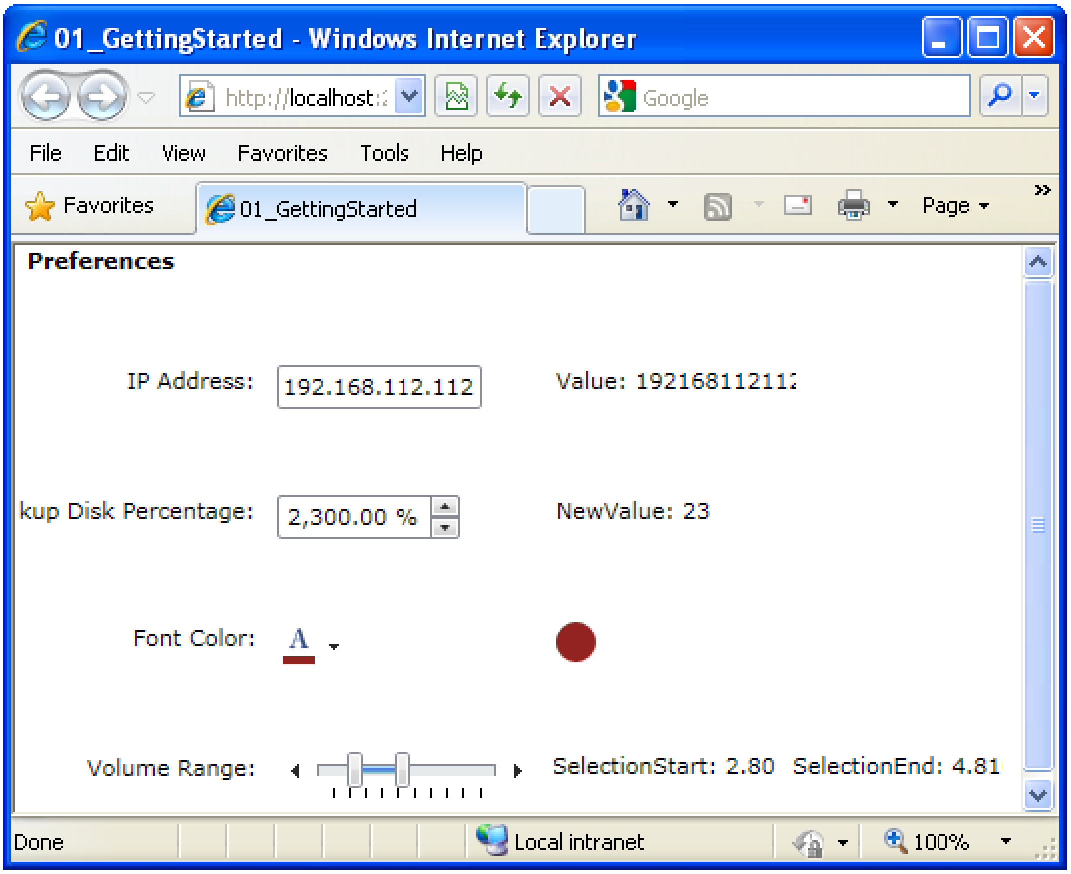
15)Drag a **RadSlider** control from the Toolbox to the XAML, just below the TextBlock defined in the previous step. Set the **Name** attribute to "slVolume", **Grid.Row**="4" and **Grid.Column**="1". Also set the following RadSlider-specific attributes: **Minimum**="1", **Maximum**="10", **SelectionStart**="2", **SelectionEnd**="8", **EnableSideTicks**="True", **TickFrequency**="1", **TickPlacement**="BottomRight", **IsSelectionRangeEnabled**="True" and **HandlesVisibility**="Visible".

16)Below the RadSlider add a TextBlock as defined below. *This TextBlock will be used to display the SelectionStart and SelectionEnd values as the slider is moved by the user.*

```
<TextBlock
   Name="tbNewVolume"
   Margin="10,0,0,0"
   Grid.Row="4"
   Grid.Column="3"
   HorizontalAlignment="Left"
   VerticalAlignment="Top">
</TextBlock>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Preferences

IP Address: 192.168.112.112 Value: 19216811211

kup Disk Percentage: 2,300.00 % NewValue: 23

Font Color:

Volume Range: SelectionStart: 2.80 SelectionEnd: 4.81

Figure 5

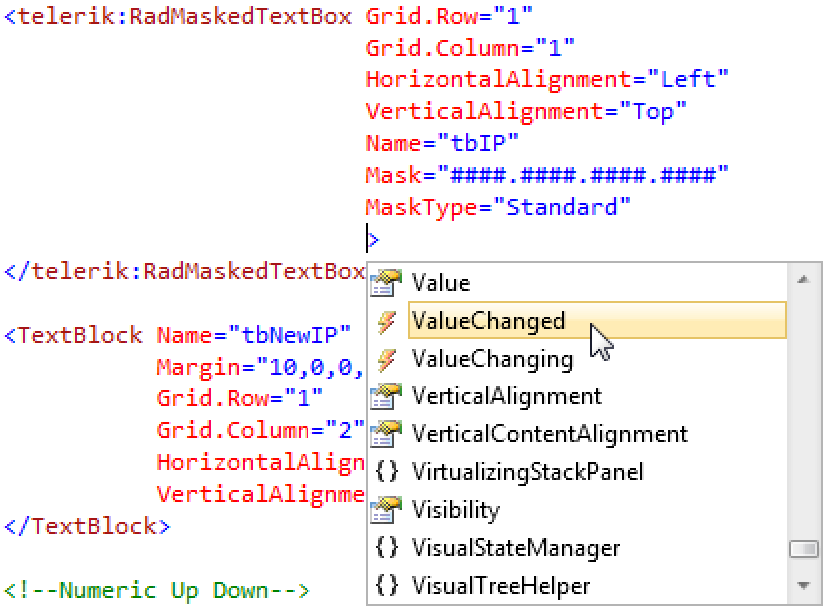
Test Application Features

- Experiment with the RadMaskedTextBox entry. Attempt to enter other than numeric characters.
- Use the RadNumericUpDown control using the buttons, mouse, mouse wheel, and direct keyboard entry.
- Drop down the RadColorPicker and select from the Standard or Theme Colors.
- Use the RadSlider control by way of clicking the +/- handles at the extreme ends of the slider using the mouse, holding down the mouse outside the thumbs, dragging the thumbs, dragging the area between the two thumbs and by using the mouse wheel.

Code Behind

The next step is to attach event handlers to the input controls and react to this feedback within the code behind.

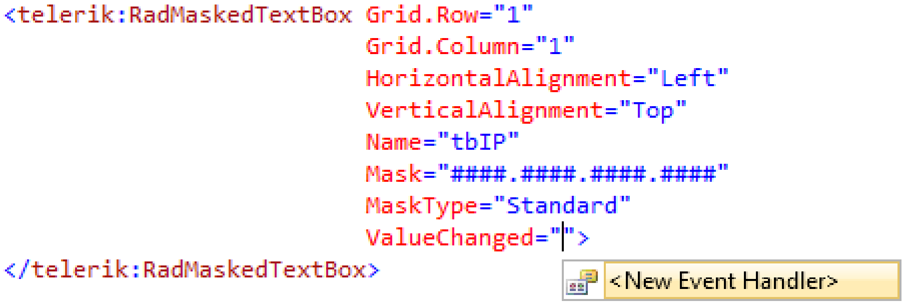
1) Navigate back to the RadMaskedTextBox in MainPage.xaml. Enter a new line inside the RadMaskedTextBox tag and click **Ctrl-Spacebar** to activate IntelliSense. Locate the **ValueChanged** event and press **Enter** to add it to the XAML.



```
<telerik:RadMaskedTextBox Grid.Row="1"
                          Grid.Column="1"
                          HorizontalAlignment="Left"
                          VerticalAlignment="Top"
                          Name="tbIP"
                          Mask="####.####.####.####"
                          MaskType="Standard"
                          >
</telerik:RadMaskedTextBox

<TextBlock Name="tbNewIP"
           Margin="10,0,0,
           Grid.Row="1"
           Grid.Column="2"
           HorizontalAlign
           VerticalAlignme
</TextBlock>

<!--Numeric Up Down-->
```

Figure 6

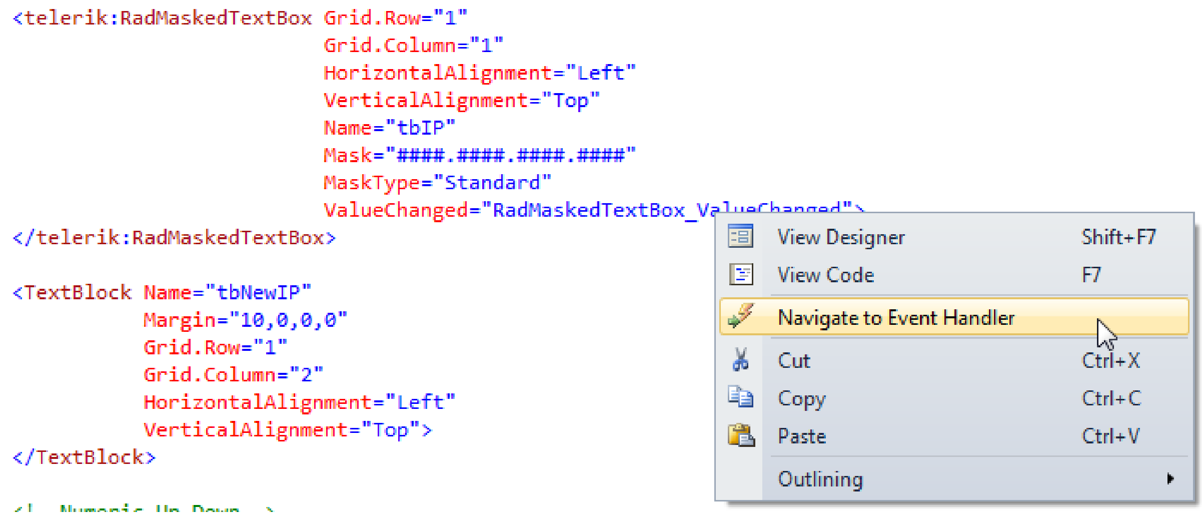
2) IntelliSense will display an option to add "**<New Event Handler>**". Press **Enter** to create the new event handler.



```
<telerik:RadMaskedTextBox Grid.Row="1"
                          Grid.Column="1"
                          HorizontalAlignment="Left"
                          VerticalAlignment="Top"
                          Name="tbIP"
                          Mask="####.####.####.####"
                          MaskType="Standard"
                          ValueChanged="">
</telerik:RadMaskedTextBox>
```

Figure 7

3) Right-click the new event handler and select "**Navigate to Event Handler**" from the context menu. *This step will take you to the code-behind for the event handler.*



```
<telerik:RadMaskedTextBox Grid.Row="1"
                          Grid.Column="1"
                          HorizontalAlignment="Left"
                          VerticalAlignment="Top"
                          Name="tbIP"
                          Mask="####.####.####.####"
                          MaskType="Standard"
                          ValueChanged="RadMaskedTextBox_ValueChanged">
</telerik:RadMaskedTextBox>

<TextBlock Name="tbNewIP"
           Margin="10,0,0,0"
           Grid.Row="1"
           Grid.Column="2"
           HorizontalAlignment="Left"
           VerticalAlignment="Top">
</TextBlock>

<!-- Numeric Up Down -->
```

Figure 8

4) Replace the code-behind for the event handler with the code below. *This event handling code corresponds to the ValueChanged event you defined in the XAML for the previous step. In this code we cast "sender" to be the RadMaskedTextBox, get its value property and display it in the TextBlock of the right-most column. Whenever the user types a new character, the ValueChanged event fires and this code re-displays the value.*

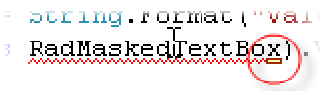
VB

```
Private Sub RadMaskedTextBox_ValueChanged(_
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  tbNewIP.Text = _
String.Format("Value: {0}", (TryCast(sender, RadMaskedTextBox)).Value)
End Sub
```

C#

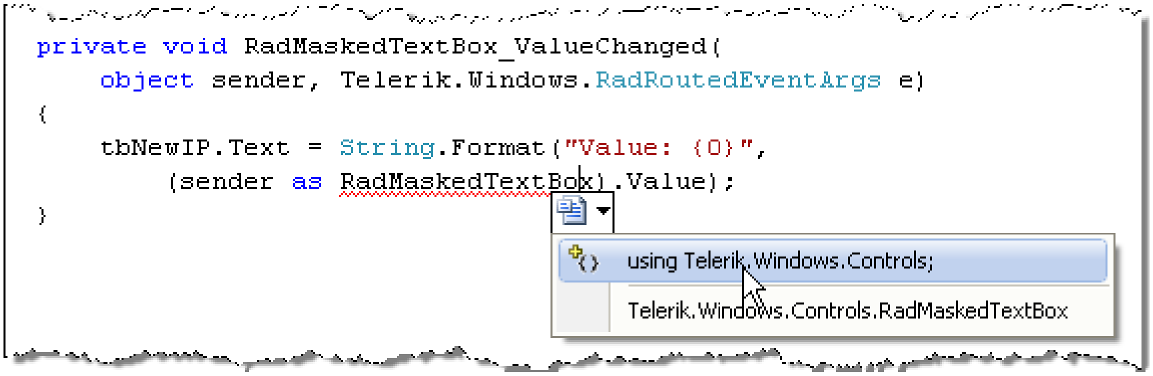
```
private void RadMaskedTextBox_ValueChanged(
   object sender, Telerik.Windows.RadRoutedEventArgs e)
{
   tbNewIP.Text = String.Format("Value: {0}",
      (sender as RadMaskedTextBox).Value);
}
```

5) The reference to RadMaskedTextBox in the code-behind will be underlined in red, signalling an error. This is because there is no namespace in scope that defines RadMaskedTextBox. To fix this, left-click the red underline and look for a dark underline indicator at the end of "RadMaskedTextBox".



```
String.Format("Val
RadMaskedTextBox).
```

6) Move your mouse over the indicator until an icon with a drop-down arrow appears, click the arrow and select "using Telerik.Windows.Controls;". *This step will automatically add a reference to the "Telerik. Windows.Controls" namespace.*



```
private void RadMaskedTextBox_ValueChanged(
    object sender, Telerik.Windows.RadRoutedEventArgs e)
{
    tbNewIP.Text = String.Format("Value: {0}",
        (sender as RadMaskedTextBox).Value);
}
```

using Telerik.Windows.Controls;
Telerik.Windows.Controls.RadMaskedTextBox

Figure 9

7) Navigate back to MainPage.xaml and locate the RadNumericUpDown tag. Enter a new **ValueChanged** attribute. Create a new event handler so that the XAML looks like the example below.

```
<telerik:RadNumericUpDown
  Name="nupBackupDiskPercentage"
  Grid.Row="2"
  Grid.Column="1"
  HorizontalAlignment="Left"
  VerticalAlignment="Top"
  ValueFormat="Percentage"
  ValueChanged="RadNumericUpDown_ValueChanged"
  >
</telerik:RadNumericUpDown>
```

8) Navigate to the code-behind for the new ValueChanged event handler and replace the code with the code below. *The event arguments for this event handler include OldValue and NewValue properties. Both properties are double types. In this event handler we are displaying the new value in a TextBlock.*

```
Private Sub RadNumericUpDown_ValueChanged(_
ByVal sender As Object, _
ByVal e As System.Windows.RoutedPropertyChangedEventArgs(Of Double))
  tbNewPercentage.Text = String.Format("NewValue: {0}", e.NewValue)
End Sub
```

C#

```
private void RadNumericUpDown_ValueChanged(
  object sender, System.Windows.RoutedPropertyChangedEventArgs<double> e)
{
  tbNewPercentage.Text =
    String.Format("NewValue: {0}", e.NewValue);
}
```

9) Navigate back to MainPage.xaml and locate the RadColorPicker tag. Enter a new **SelectedColorChanged** attribute. Create a new event handler so that the XAML looks like the example below.

XAML

```
<telerik:RadColorPicker
  Name="cpFont"
  Grid.Row="3"
  Grid.Column="1"
  HorizontalAlignment="Left"
  SelectedColorChanged="RadColorPicker_SelectedColorChanged"
  >
</telerik:RadColorPicker>
```

10)Navigate to the code-behind for the new SelectedColorChanged event handler and replace the code with the code below. *This step will create a new SolidColorBrush using the RadColorPicker's currently selected color. The new brush is assigned to an Ellipse shape displayed in the right-most column of the grid.*

VB

```
Private Sub RadColorPicker_SelectedColorChanged(_
ByVal sender As Object, ByVal e As EventArgs)
      elNewColor.Fill = New SolidColorBrush(cpFont.SelectedColor)
End Sub
```

C#

```
private void RadColorPicker_SelectedColorChanged(object sender, EventArgs e)
{
  elNewColor.Fill = new SolidColorBrush(cpFont.SelectedColor);
}
```

11)Verify that you have a reference to the **System.Windows.Media** namespace in the "Imports" (VB) or "using" (C#) section of code.

12)Navigate back to MainPage.xaml and locate the RadSlider tag. Enter a new **SelectionRangeChanged** attribute. Create a new event handler so that the XAML looks like the example below.

```
<telerik:RadSlider
   Name="slVolume"
   Grid.Row="4"
   Grid.Column="1"
   Minimum="1"
   Maximum="10"
   SelectionStart="2"
   SelectionEnd="8"
   EnableSideTicks="True"
   TickFrequency="1"
   TickPlacement="BottomRight"
   IsSelectionRangeEnabled="True"
   HandlesVisibility="Visible"
   SelectionRangeChanged="RadSlider_SelectionRangeChanged">
</telerik:RadSlider>
```

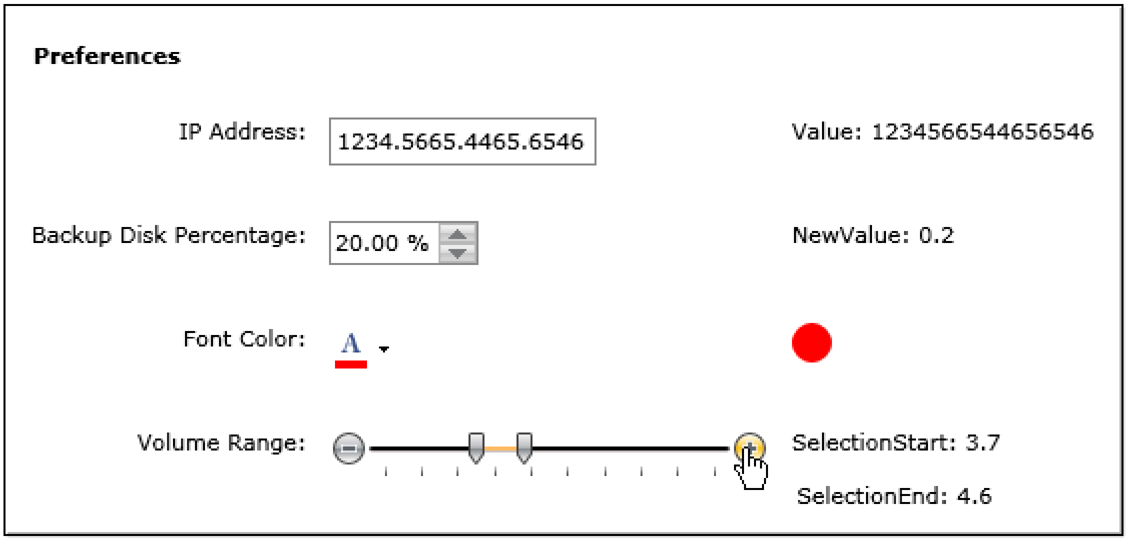
13)Navigate to the code-behind for the new SelectionRangeChanged event handler and replace the code with the code below. *This step displays the upper and lower value of the two slider thumbs using the event arguments NewValue.SelectionStart and NewValue.SelectionEnd properties.*

```
Private Sub RadSlider_SelectionRangeChanged(_
ByVal sender As Object, _
ByVal e As System.Windows.RoutedPropertyChangedEventArgs(_
Of SelectionRangeChangedEventArgs))
  If tbNewVolume IsNot Nothing Then
     tbNewVolume.Text = _
String.Format("SelectionStart: {0}  SelectionEnd: {1}", _
e.NewValue.SelectionStart, e.NewValue.SelectionEnd)
  End If
End Sub
```

```
private void RadSlider_SelectionRangeChanged(
   object sender,
   System.Windows.RoutedPropertyChangedEventArgs<SelectionRangeChangedEventArgs> e)
{
   if (tbNewVolume != null)
   {
      tbNewVolume.Text = String.Format("SelectionStart: {0}  SelectionEnd: {1}",
         e.NewValue.SelectionStart, e.NewValue.SelectionEnd);
   }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Preferences

IP Address: 1234.5665.4465.6546

Value: 1234566544656546

Backup Disk Percentage: 20.00 %

NewValue: 0.2

Font Color:

Volume Range:

SelectionStart: 3.7

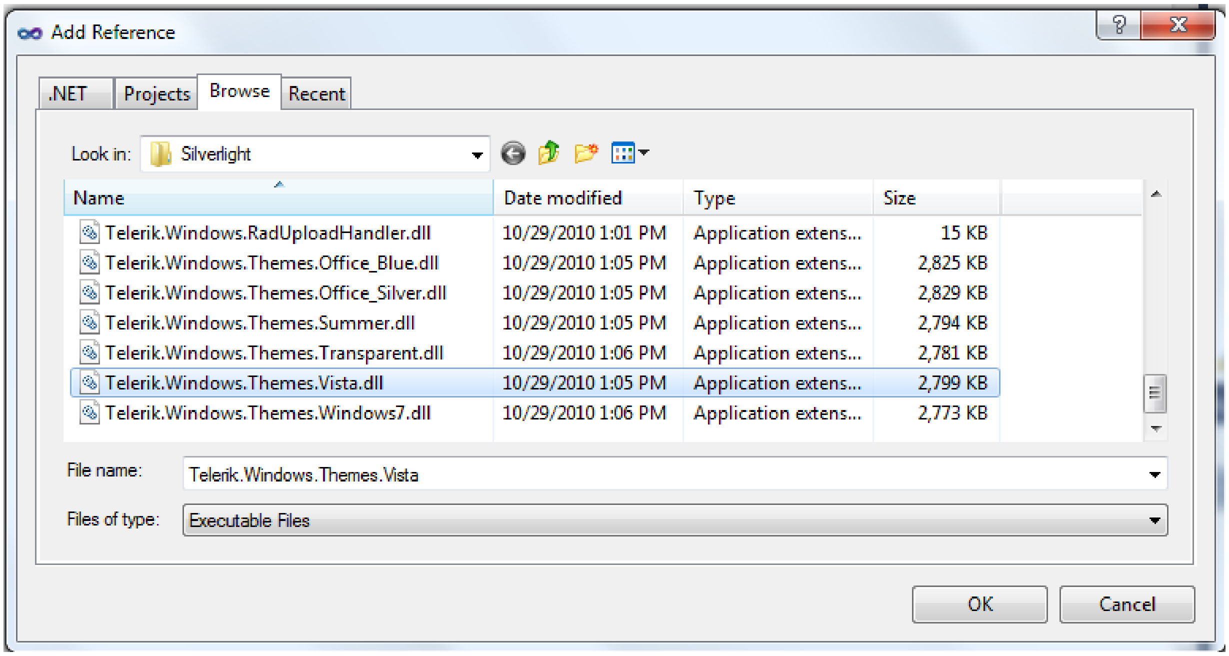
SelectionEnd: 4.6

Figure 10

Theming

The last step in this exercise is to apply a theme to each of the input controls.

1) Navigate to the Solution Explorer in Visual Studio. Right-click the References node in the project containing your MainPage.xaml file and select **Add Reference...**. In the "Add Reference" dialog, select the "Browse" tag, locate the RadControls for Silverlight installation directory and select the **Telerik.Windows.Themes.Vista.dll** assembly in the \Binaries\Silverlight directory. Press the **OK** button to add the assembly.



Look in: Silverlight

Name	Date modified	Type	Size
Telerik.Windows.RadUploadHandler.dll	10/29/2010 1:01 PM	Application extens...	15 KB
Telerik.Windows.Themes.Office_Blue.dll	10/29/2010 1:05 PM	Application extens...	2,825 KB
Telerik.Windows.Themes.Office_Silver.dll	10/29/2010 1:05 PM	Application extens...	2,829 KB
Telerik.Windows.Themes.Summer.dll	10/29/2010 1:05 PM	Application extens...	2,794 KB
Telerik.Windows.Themes.Transparent.dll	10/29/2010 1:06 PM	Application extens...	2,781 KB
Telerik.Windows.Themes.Vista.dll	10/29/2010 1:05 PM	Application extens...	2,799 KB
Telerik.Windows.Themes.Windows7.dll	10/29/2010 1:06 PM	Application extens...	2,773 KB

File name: Telerik.Windows.Themes.Vista

Files of type: Executable Files

Figure 11

2) In the MainPage.xaml, add the attribute '**telerik:StyleManager.Theme="Vista"**' to each of the input control tags (that includes RadMaskedTextBox, RadNumericUpDown, RadColorPicker and RadSlider). Here's an example of what this looks like for the RadMaskedTextBox.

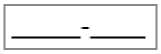
```
<telerik:RadMaskedTextBox
   Grid.Row="1"
   Grid.Column="1"
   HorizontalAlignment="Left"
   VerticalAlignment="Top"
   Name="tbIP"
   Mask="####.####.####.####"
   MaskType="Standard"
   ValueChanged="RadMaskedTextBox_ValueChanged"
   telerik:StyleManager.Theme="Vista"
   >
</telerik:RadMaskedTextBox>
```

3) Press **Ctrl-F5** to run the application. The changes to the control can be very subtle depending on the particular style and the version of the software. A text input may have slightly rounded corners with subtle shading instead of an abrupt square of a solid color. More pronounced styles may include complete changes in graphics for buttons and other visual elements of a given control.

9.4 Control Details

9.4.1 Masked Text Box

Use RadMaskedTextBox to validate user entry "up front". This control not only restricts entry to certain predefined patterns but makes it easier for the user to figure out what data the application is asking for. This help is provided courtesy of the **Mask** property that contains a set of characters to define the order and content of allowed input. For example, a mask of "#####-####" could define a United States postal code and looks something like this in the browser:



_____-____

The user can enter numbers only and automatically skips over the literal dash "-" symbol. The other key property that makes RadMaskedTextBox so flexible is the **MaskType** property. MaskType determines how the mask will be interpreted. MaskType can be one of four possible values:

MaskType	Description
None	If MaskType is set to "None", the RadMaskedTextBox does not validate and acts as a standard text box for free-form entry.
Standard	Use the Standard MaskType to define custom masks as in the previous "US Postal Code" example.
Numeric	Use predefined format codes to only allow currency, fixed point, decimal, percentages or integers.

MaskType	Description
DateTime	Use predefined format codes for a wide range of date and time layouts. You can also use format codes to create custom date formats.

Masks Cheat Sheet

You can use the tables below as a quick guide or check the help online for the latest information.

Standard Masks

<table>
<tr><th>Mask</th><th>Allowed Input</th><th>Example</th></tr>
<tr><td>#</td><td>Allows digit or space, optional. Plus (+) and minus (-) signs are also allowed.</td><td>&lt;TextBlock
  Text="Zip Code:">
</TextBlock>
<telerik:RadMaskedTextBox
  MaskType="Standard"
  Mask="#####-####" >
</telerik:RadMaskedTextBox&gt;</td></tr>
<tr><td>L</td><td>Letter, required. Restricts input to the ASCII letters a-z and A-Z. This mask element is equivalent to [a-zA-Z] in regular expressions.</td><td>&lt;TextBlock
  Text="Enter your four character promotional code:">
</TextBlock>
<telerik:RadMaskedTextBox
  MaskType="Standard"
  Mask="LLLL"
  >
</telerik:RadMaskedTextBox&gt;</td></tr>
<tr><td>a</td><td>Any symbols</td><td>&lt;TextBlock
  Text="Product Key:">
</TextBlock>
<telerik:RadMaskedTextBox
  MaskType="Standard"
  Mask="aaaaa-aaaaa-aaaaa-aaaaa-aaaaa"
  >
</telerik:RadMaskedTextBox&gt;</td></tr>
<tr><td>\</td><td>Escapes a mask character, turning it into a literal</td><td>&lt;TextBlock
  Text="Chapter Number:">
</TextBlock>
<telerik:RadMaskedTextBox
  MaskType="Standard"
  Mask="\###"
  >
</telerik:RadMaskedTextBox&gt;</td></tr>
<tr><td>|</td><td>Accepts letters or space, optional.</td><td>&lt;TextBlock
  Text="Name:">
</TextBlock>
<telerik:RadMaskedTextBox
  MaskType="Standard"
  Mask="|||||||||||||||||||"
  &gt;
</telerik:RadMaskedTextBox&gt;</td></tr>
<tr><td>9</td><td>Digit, required. This element will</td><td>&lt;TextBlock
  Text="Page Number"&gt;</td></tr>
</table>

Mask	Allowed Input	Example

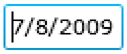
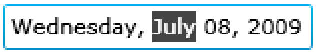
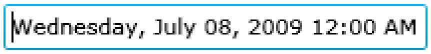
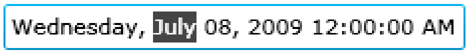
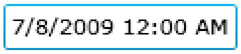
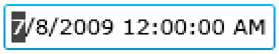
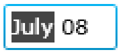
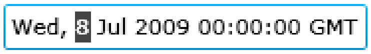
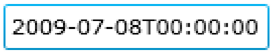
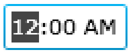
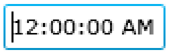
	accept any single digit between 0 and 9.	`</TextBlock>`
		`<telerik:RadMaskedTextBox`
		`    MaskType="Standard"`
		`    Mask="999"`
		`    >`
		`</telerik:RadMaskedTextBox>`

Numeric Masks

Mask	Allowed Input
c,C	Currency
g,G,f,F	Fixed point
n,N	Decimal
p,P	Percent
d,D	Standard (integers)

You can also add a length specifier to any of the above masks. For example. "d5" accepts a five digit integer.

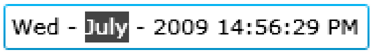
Date/Time Masks

Mask	Allowed Input	Example
d	Short date	 7/8/2009
D	Long date	 Wednesday, July 08, 2009
f	Full date and time (long date, short time)	 Wednesday, July 08, 2009 12:00 AM
F	Full date and time (long date, long time)	 Wednesday, July 08, 2009 12:00:00 AM
g	General (short date, short time)	 7/8/2009 12:00 AM
G	General (short date, long time)	 7/8/2009 12:00:00 AM
m,M	Month day	 July 08
r,R	RFC1123	 Wed, 8 Jul 2009 00:00:00 GMT
s	Sortable Date/Time pattern	 2009-07-08T00:00:00
t	Short time	 12:00 AM
T	Long time	 12:00:00 AM

To completely customize the date mask, use a combination of the codes in the table below.

Mask	Description
dd	The numeric day of the month.
ddd	The abbreviated day of the week name.
M	The month name followed by the numeric day.
MM	The numeric month
MMM	The abbreviated name of the month.
MMMM	The full name of the month.
y	The full month name and year numeric
yy	The year without the century.
yyyy	The year in four digits, including the century.
h, hh	The hour in a 12-hour clock.
H, HH	The hour in a 24-hour clock.
m, mm	The minute.
s, ss	The second.
t	The first character in the AM/PM designator.
tt	The AM/PM designator.

Here is an example of using date/time formatting codes in combination to create a custom mask: "ddd - MMMM - yyyy H:m:s tt". The result in the browser might look like this:



Wed - July - 2009 14:56:29 PM

The user is able to use the mouse to put focus on one of the codes, e.g. the "MMMM" portion that shows as "July", and then use the arrow keys to page through the months.

Properties and Events

Read the **Value** property to get the user entry without the mask. Use the **MaskedText** property to get both the mask and the formatting characters.

In the "Getting Started" example you saw how the **ValueChanged** event responds as the user types in the RadMaskedTextBox. The **ValueChanging** event lets you preview the **NewValue** and **NewMaskedText** properties of the event arguments. The change can be canceled by setting the **Handled** argument property to true. The example below prevents entry if the new value will be over 100.

VB

```
Private Sub RadMaskedTextBox_ValueChanging(_
ByVal sender As Object, ByVal e As RadMaskedTextBoxValueChangingEventArgs)
  If e.NewValue.ToString().Length > 0 Then
    e.Handled = Convert.ToInt32(e.NewValue) > 100
  End If
End Sub
```

C#

```
private void RadMaskedTextBox_ValueChanging(object sender,
   RadMaskedTextBoxValueChangingEventArgs e)
{
   if (e.NewValue.ToString().Length > 0)
   {
      e.Handled = Convert.ToInt32(e.NewValue) > 100;
   }
}
```

To localize the RadMaskedTextBox to a particular language and culture, set the **Culture** property to a culture code at design time:

XAML

```
<telerik:RadMaskedTextBox
   Culture="fr-FR"
   MaskType="DateTime"
   Mask="D"            >
</telerik:RadMaskedTextBox>
```

...or assign a new CultureInfo at run time:

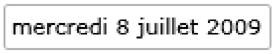
VB

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  tbCultureTest.Culture = New CultureInfo("fr-FR")
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   tbCultureTest.Culture = new CultureInfo("fr-FR");
}
```

Both the XAML markup and code-behind examples above show a long Date/Time in French:



mercredi 8 juillet 2009

Walk Through

In this lab you will use a sampling of each mask type with some basic settings, then react to changing values using the primary events for RadMaskedTextBox.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

XAML Editing

1) Open the MainPage.xaml file for editing.

2) Replace the <Grid> tag named "LayoutRoot" with the XAML markup below. *This will setup two nested StackPanel controls inside the grid. The outermost stack panel will contain three RadMaskedTextBox controls and the innermost stack panel will contain some TextBlock controls to display new values as the user edits.*

```
<Grid
  x:Name="LayoutRoot"
  Background="White">
  <StackPanel>

    <!--MaskType Standard-->

    <!--MaskType Numeric-->

    <!--MaskType DateTime-->

    <StackPanel
      Orientation="Horizontal">

      <!--Value-->

      <!--MaskedText-->

    </StackPanel>

  </StackPanel>
</Grid>
```

3) Locate the comment "<!–MaskType Standard-->" and drop a RadMaskedTextBox just below it. *This step will add the correct XAML name space assembly reference for RadMaskedTextBox.*

4) Replace the RadMaskedTextBox with the XAML shown below. *This step will add a TextBlock to act as a label and a RadMaskedTextBox with MaskType = "Standard". The Mask will display an escaped literal "#", followed by three alpha characters, a dash "-" and finally three more alpha characters. The ValueChanged event handler will be common among all three RadMaskedTextBoxes and will be coded in a later step.*

```
<!--MaskType Standard-->

<TextBlock
  Text="Standard:">
</TextBlock>
<telerik:RadMaskedTextBox
  Margin="20"
  HorizontalAlignment="Left"
  MaskType="Standard"
  Mask="\#LLL-LLL"
  ValueChanged="ValueChangedHandler">
</telerik:RadMaskedTextBox>
```

5) Locate the comment "<!--MaskType Numeric-->" and add the XAML shown below. *The RadMaskedTextBox here has the MaskType set to "Numeric" with a mask of "p2", a percentage with two places.*

```
<!--MaskType Numeric-->

<TextBlock
  Text="Numeric Percentage:">
</TextBlock>
<telerik:RadMaskedTextBox
  Margin="20"
  HorizontalAlignment="Left"
  MaskType="Numeric"
  Mask="p2"
  ValueChanged="ValueChangedHandler">
</telerik:RadMaskedTextBox>
```

6) Locate the comment "<!--MaskType DateTime -->" and add the XAML shown below. *The RadMaskedTextBox here has the MaskType set to "DateTime" with a mask of "F" to designate a "Full" date and time.*

```
<!--MaskType DateTime-->

<TextBlock
   Text="DateTime: Full date and time">
</TextBlock>
<telerik:RadMaskedTextBox
   Margin="20"
   HorizontalAlignment="Left"
   MaskType="DateTime"
   Mask="F"
   ValueChanged="ValueChangedHandler">
</telerik:RadMaskedTextBox>
```

7) Locate the comment "<!--Value-->" and add the XAML shown below. *The first TextBlock defined here acts as a simple label. The second is named so we can update it with the latest value from the code-behind.*

```
<!--Value-->

<TextBlock
   Margin="10,0,0,0"
   Text="Value:">
</TextBlock>
<TextBlock
   Name="tbValue"
   Margin="10,0,0,0">
</TextBlock>
```

8) Locate the comment "<!--MaskedText-->" and add the XAML shown below.

```
<!--MaskedText-->

<TextBlock
   Text="MaskedText:"
   Margin="10,0,0,0">
</TextBlock>
<TextBlock
   Name="tbMaskedText"
   Margin="10,0,0,0">
</TextBlock>
```

9) Locate one of the RadMaskedTextBox controls (any of them will do), find the ValueChanged property, right-click the "ValueChangedHandler" and select "Navigate to Event Handler" from the context menu. Replace the event handler with the code below. *The code casts "sender" as the RadMaskedTextBox that has triggered this particular event, then assigns the Value and MaskedText to TextBlock controls in the inner StackPanel. Notice that Value is an object type: depending on the MaskType, value may be a string, double, or DateTime.*

VB

```
Private Sub ValueChangedHandler(_
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  Dim tb As RadMaskedTextBox = TryCast(sender, RadMaskedTextBox)
  tbValue.Text = tb.Value.ToString()
  tbMaskedText.Text = tb.MaskedText
End Sub
```

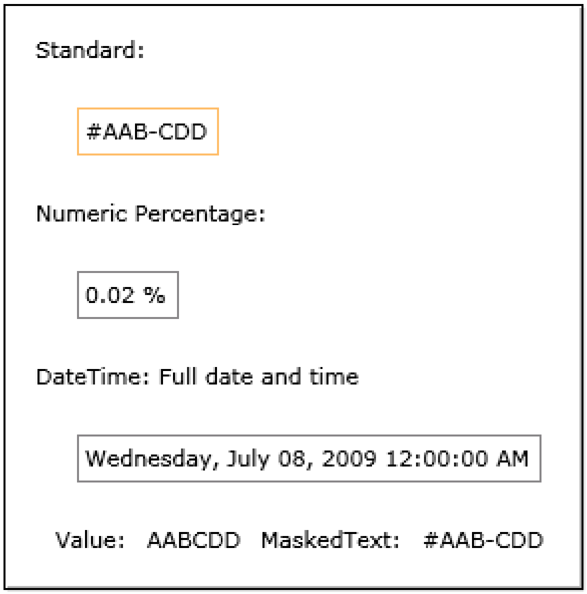
C#

```
private void ValueChangedHandler(object sender,
   Telerik.Windows.RadRoutedEventArgs e)
{
   RadMaskedTextBox tb = sender as RadMaskedTextBox;
   tbValue.Text = tb.Value.ToString();
   tbMaskedText.Text = tb.MaskedText;
}
```

10)Add a Telerik.Windows.Controls namespace reference to your "Imports" (VB) or "uses" (C#) section of code.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Standard:

#AAB-CDD

Numeric Percentage:

0.02 %

DateTime: Full date and time

Wednesday, July 08, 2009 12:00:00 AM

Value: AABCDD MaskedText: #AAB-CDD

Test Application Features

- Enter characters into the Standard masked text box.
- Enter characters into the Numeric masked text box.
- Enter characters into the DateTime masked text box.

Ideas for Extending This Example

- Set focus on the first masked text of the page so the user can start typing immediately.

Gotcha!

At the time of this writing, setting focus in Silverlight applications requires a work-around. First, you need to set focus to the Silverlight Plugin, then you need to set focus to control itself. Set focus using Dispatcher.BeginInvoke() to queue your code so that it occurs after the plugin has loaded.

VB

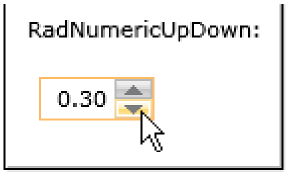
```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  SetFocus(tbAirportCode)
End Sub

Private Sub SetFocus(ByVal control As Control)
  ' get focus inside the Silverlight plugin
  System.Windows.Browser.HtmlPage.Plugin.Invoke("focus")
  ' queue this call to occur after the plugin focus
  Dispatcher.BeginInvoke(Function() control.Focus())
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   SetFocus(tbAirportCode);
}

private void SetFocus(Control control)
{
   // get focus inside the Silverlight plugin
   System.Windows.Browser.HtmlPage.Plugin.Invoke("focus");
   // queue this call to occur after the plugin focus
   Dispatcher.BeginInvoke(delegate() { control.Focus(); });
}
```

9.4.2 Up Down Controls

The **RadNumericUpDown** control has a paired set of "repeat" buttons and can be used to increment and decrement values and a text input element as well.



The story begins with the Telerik **RadRangeBase** abstract class. RadRangeBase represents an element that has a value within a specific range such as a progress bar or slider control. RadUpDown, RadNumericUpDown and RadSlider controls all ultimately descend from RadRangeBase. RadRangeBase introduces **LargeChange**, **SmallChange**, **Maximum**, **Minimum** and **Value** properties and the **ValueChanged** event. LargeChange determines the value change when the page up and down keys are used and SmallChange is the value change when the arrow up and down keys are used.

RadNumericUpDown Properties

RadNumericUpDown adds the ability to display and format a value. The key properties added by RadNumericUpDown are ValueFormat and NumberFormatInfo. **ValueFormat** is an enumeration that can be Numeric, Currency or Percentage. **NumberFormatInfo** is a type from the System.Globalization namespace that lets you fine-tune formatting and display depending on the culture. NumberFormatInfo properties are clumped into prefixes that correspond to the ValueFormat setting, e.g. the number of decimal digits is controlled by NumberDecimalDigits, CurrencyDecimalDigits and PercentageDecimalDigits.

If these formatting options aren't sufficient, you can assign a string to the **CustomUnit** property and the string will be appended to the value automatically. For example, if you assigned Value = "3" and CustomUnit = "Cases", the control will display "3 Cases". If you want to retrieve the formatted value, read the **ContentText** property. To make the numeric up down control read-only set the **IsEditable** property to false. You can set the **ShowButtons** property to false as well.

Walk Through

In this walk-through you will create a set of entries for price, quantity, discount and extended price. As the values change, the extended price is calculated and placed in a read-only RadNumericUpDown.

1) Start with the previous RadUpDown project or a copy.

2) Open the MainPage.xaml file for editing.

3) Add a new StackPanel inside the StackPanel that already exists in this project. *Notice that the Orientation property is set to "Horizontal" so all the controls we add next will be arranged from left to right in a line. The comments inside the StackPanel indicate where we will drop in groups of controls.*

```
<StackPanel
   Orientation="Horizontal">
   <!--Price-->
   <!--Quantity-->
   <!--Discount-->
   <!--Extended Price-->
</StackPanel>
```

4) Locate the "<!--Price-->" comment and replace it with the XAML below. *Notice that the ValueFormat is "Currency". The NumberFormatInfo will be set later in the Loaded event of the user control.*

```
<!--Price-->

<TextBlock
   Margin="5">Price:</TextBlock>
<telerik:RadNumericUpDown
   Name="tbPrice"
   Margin="5"
   HorizontalAlignment="Left"
   MinWidth="75"
   ValueFormat="Currency"
   Minimum="0"
   ValueChanged="RadNumericUpDown_ValueChanged">
</telerik:RadNumericUpDown>
```

Gotcha!

As of this writing, the NumberFormatInfo properties can't be set properly in XAML. In this exercise we will set the NumberFormatInfo properties in code.

5) Locate the "<!--Quantity-->" comment and replace it with the XAML below. *Notice that the ValueFormat is "Numeric", SmallChange is "1" and CustomUnit is "Items".*

```
<!--Quantity-->

<TextBlock
   Margin="5">Quantity:</TextBlock>
<telerik:RadNumericUpDown
   Name="tbQuantity"
   Margin="5"
   HorizontalAlignment="Left"
   MinWidth="75"
   ValueFormat="Numeric"
   CustomUnit="Items"
   SmallChange="1"
   Minimum="0"
   Maximum="1000"
   ValueChanged="RadNumericUpDown_ValueChanged">
</telerik:RadNumericUpDown>
```

Tip!

To use integers in your RadNumericUpDown:

· Set ValueFormat = "Numeric"

· Set SmallChange = "1"

· In the code-behind set the NumberFormatInfo.NumberDecimalDigits to "0"

6) Locate the "<!--Discount-->" comment and replace it with the XAML below. *Notice that the ValueFormat is "Percentage".*

```
<!--Discount-->

<TextBlock
   Margin="5">Discount:</TextBlock>
<telerik:RadNumericUpDown
   Name="tbDiscount"
   Margin="5"
   HorizontalAlignment="Left"
   MinWidth="75"
   ValueFormat="Percentage"
   ValueChanged="RadNumericUpDown_ValueChanged">
</telerik:RadNumericUpDown>
```

7) Locate the "<!--Extended Price-->" comment and replace it with the XAML below. *Notice that the ValueFormat is "Currency". Also notice that the IsEditable and ShowButtons properties are false to make this a read-only control.*

```
<!--Extended Price-->

<TextBlock
   Margin="5">Extended Price:</TextBlock>
<telerik:RadNumericUpDown
   Name="tbExtendedPrice"
   Margin="5"
   HorizontalAlignment="Left"
   MinWidth="75"
   IsEditable="False"
   ShowButtons="False"
   ValueFormat="Currency">
</telerik:RadNumericUpDown>
```

8) Locate the outermost <UserControl> tag and add a Loaded event and set the value to "UserControl_Loaded". The XAML markup will look something like the example below.

```
<UserControl
...
Loaded="UserControl_Loaded">
```

9) Right-click the "UserControl_Loaded" event and select "Navigate to Event Handler" from the context menu. Replace the event handler with the code below. *Here we set the number of decimal digits for the quantity to zero so that the control will be formatted as an integer. Also notice that we're setting the price and extended price Maximum properties to the maximum allowed by the Double type.*

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  tbQuantity.NumberFormatInfo.NumberDecimalDigits = 0
  tbPrice.Maximum = Double.MaxValue
  tbExtendedPrice.Maximum = Double.MaxValue
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   tbQuantity.NumberFormatInfo.NumberDecimalDigits = 0;
   tbPrice.Maximum = Double.MaxValue;
   tbExtendedPrice.Maximum = Double.MaxValue;
}
```

10) Go back to the XAML and locate one of the ValueChanged properties, right-click and select "Navigate to Event Handler" from the context menu. Replace the event handler with the code below. *Here we calculate an extended price, and subtract a discount if the user has entered a discount percentage.*

VB

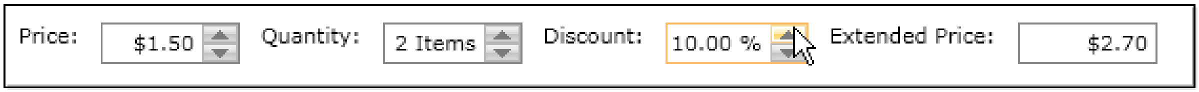
```
Private Sub RadNumericUpDown_ValueChanged(ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.RadRangeBaseValueChangedEventArgs)
  If (tbQuantity IsNot Nothing) AndAlso (tbPrice IsNot Nothing) Then
    Dim extended As Double = CDbl(tbQuantity.Value) * CDbl(tbPrice.Value)
    If tbDiscount.Value > 0 Then
      Dim discount As Double = extended * CDbl(tbDiscount.Value)
      extended = extended - discount
    End If
    tbExtendedPrice.Value = extended
  End If
End Sub
```

C#

```
private void RadNumericUpDown_ValueChanged(object sender,
   Telerik.Windows.Controls.RadRangeBaseValueChangedEventArgs e)
{
   if ((tbQuantity != null) && (tbPrice != null))
   {
      double extended = (double)tbQuantity.Value * (double)tbPrice.Value;
      if (tbDiscount.Value > 0)
      {
         double discount = extended * (double)tbDiscount.Value;
         extended = extended - discount;
      }
      tbExtendedPrice.Value = extended;
   }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



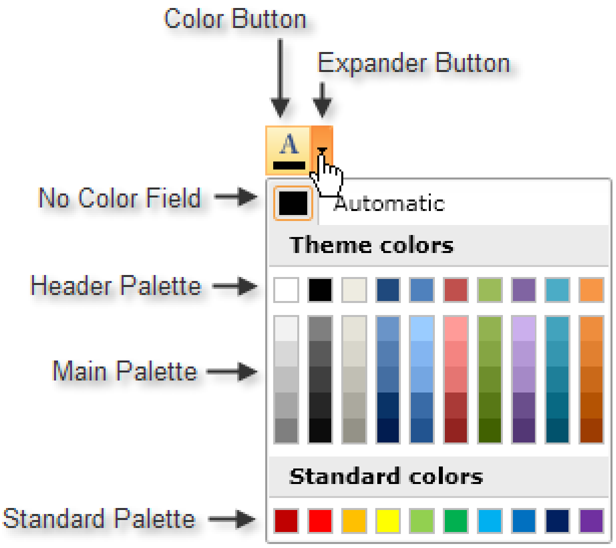
Price: $1.50 Quantity: 2 Items Discount: 10.00 % Extended Price: $2.70

Test Application Features

- Use the repeat buttons to change the price, quantity and discount.

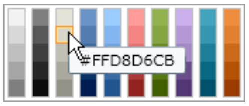
9.4.3 Color Pickers

There are three color choosing controls currently in the Toolbox: **RadColorPaletteView**, **RadColorSelector** and **RadColorPicker**. Since RadColorPicker has the superset of functionality in all the color choosing controls, lets take a quick look at its structure:



Each color choosing control builds on the functionality of the previous:

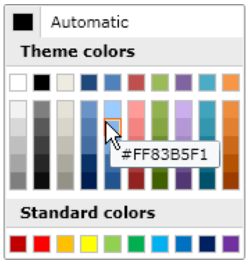
RadColorPaletteView



RadColorPaletteView is a simple grid of colors.

- Assign a ColorPreset enumeration value to the Palette property to display a ready-to-use predefined set of colors such as GrayScale, ReallyWebSafe, Office, etc.
- Assign your own set of colors by binding the ItemsSource property.
- Get the selected color by handling the SelectionChanged event and retrieving the SelectedValue property. **Note**: you must cast SelectedValue as a Color type to use it.

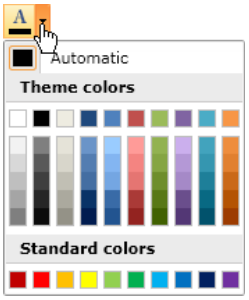
RadColorSelector



RadColorSelector is a more complex grid of colors that includes a "No Color" button, a "Header palette" and a main palette. Here you assign ColorPreset to the **HeaderPalette**, **MainPalette** or **StandardPalette** properties.

- To assign your own colors bind **HeaderPaletteItemsSource**, **MainPaletteItemsSource** or **StandardPaletteItemsSource**.
- To know when a new color is chosen, handle the **SelectedColorChanged** event and check the **SelectedColor** property.

RadColorPicker



RadColorPicker adds the ability to drop down the selector. When a color is selected, the selector rolls up and only the button with the new selected color is displayed.

Notes

Each of the controls above builds on the functionality of the previous control, but there is no direct inheritance between these controls as of this writing.

Using ObjectCollection

Silverlight, unlike WPF, does not support ArrayList or any type that will easily let you define a list of objects directly within XAML. To assign a list of any object in XAML you can use ObjectCollection which is included in the "Toolkit" at http://silverlight.net/getstarted/. The Toolkit contains transitional controls, samples, utilities and various goodies that haven't been incorporated to the base Silverlight product. Be aware that the "Toolkit" is not the same as the "Developer Tools for Silverlight", where the latter contains the SDK, Developer Runtime and the Visual Studio Project Templates.

Once you have the Tookit installed, you can find the binaries under:

\Program Files\Microsoft SDKs\Silverlight\v3.0\Toolkit\<version>\Bin

ObjectCollection is found in the System.Windows.Controls.Toolkit assembly. To use ObjectCollection, first include a reference to this assembly in the Solution Explorer References node of your project. Then, in the App.xaml, add a reference similar to the one below:

```
xmlns:controls=
"clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Toolkit"
```

Then in your App.xaml markup, within the Application.Resources tag, add any number of custom collections that can be referenced in your other XAML files. The collection below is named "FoodGroups" and contains three String objects. The "sys" below stands for the System namespace that contains basic CLR types and is found in the mscorlib.dll assembly.

```
<Application.Resources>

  <controls:ObjectCollection
    x:Key="FoodGroups">
    <sys:String>Burgers</sys:String>
    <sys:String>Fries</sys:String>
    <sys:String>Soft Drinks</sys:String>
  </controls:ObjectCollection>

</Application.Resources>
```

You can reference the ObjectCollection using the name specified in the "x:Key" attribute using the binding syntax below.

```
<telerik:RadComboBox
 ItemsSource="{Binding Source={StaticResource FoodGroups}}">
```

Gotcha!

Be aware, due to the transitional nature of the toolkit, that this information is volatile and that the naming, location and makeup of the toolkit may change.

Walk Through

This walk-through demonstrates how to respond to user selections for each color choosing control. The exercise shows how to define a list of colors as a resource and assign in the XAML to a palette. You will also see how a list of colors can be assigned programmatically to a palette.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

XAML Editing

1) From the Solution Explorer, open the App.xaml file for editing.
2) Add XML namespace references shown below to the App.xaml file. *The Toolkit namespace location may change in the future. See "Using ObjectCollection" above for more information.*

```
xmlns:controls="clr-namespace:System.Windows.Controls;assembly=System.Windows.Controls.Toolkit
xmlns:media="clr-namespace:System.Windows.Media;assembly=System.Windows"
```

3) Replace the Application.Resources tag with the markup below:

```
<Application.Resources>
  <controls:ObjectCollection
    x:Key="SmokyMountain">
    <media:Color>#FFDDDDDD</media:Color>
    <media:Color>#FFCCCCFF</media:Color>
    <media:Color>#FFAAAAFF</media:Color>
    <media:Color>WhiteSmoke</media:Color>
    <media:Color>LightGray</media:Color>
    <media:Color>Gray</media:Color>
    <media:Color>#FFFFBBBB</media:Color>
    <media:Color>#FFEEAAAA</media:Color>
    <media:Color>#FFDDAAAA</media:Color>
  </controls:ObjectCollection>
</Application.Resources>
```

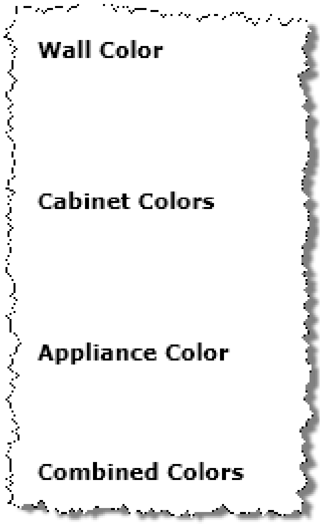
4) From the Solution Explorer, open the MainPage.xaml file for editing.

5) Insert the XAML below inside the Grid tag named "LayoutRoot". *This step will setup the basic user interface structure using StackPanels.*

```
<StackPanel>

    <TextBlock
        Margin="5"
        FontWeight="Bold">Wall Color</TextBlock>

    <StackPanel
        Orientation="Horizontal">

        <!--RadColorPaletteView-->

        <Rectangle
            Name="rectWallColor"
            Width="50"
            Height="50">
        </Rectangle>

    </StackPanel>

    <TextBlock
        Margin="5"
        FontWeight="Bold">Cabinet Colors</TextBlock>

    <StackPanel
        Orientation="Horizontal">

        <!--RadColorSelector-->

        <Rectangle
            Name="rectCabinets"
            Width="50"
            Height="50"
            HorizontalAlignment="Left">
        </Rectangle>

    </StackPanel>

    <StackPanel
        Orientation="Horizontal">

        <TextBlock
            Margin="5"
            FontWeight="Bold">Appliance Color</TextBlock>

        <!--RadColorPicker-->
```

```
            <Rectangle
                Name="rectAppliance"
                Margin="5"
                Width="50"
                Height="50"
                HorizontalAlignment="Left">
            </Rectangle>

        </StackPanel>

        <!--Combined Colors-->

        <TextBlock
            Margin="5"
            FontWeight="Bold">Combined Colors</TextBlock>

        <StackPanel
            Orientation="Horizontal">

            <Rectangle
                Name="rectCombined"
                Margin="20"
                Width="150"
                Height="150"
                HorizontalAlignment="Left">
            </Rectangle>

        </StackPanel>

    </StackPanel>
```

If you run the application now the web page looks something like this:



6) Drag a **RadColorPaletteView** control from the Toolbox to a place just under the comment "<!–RadColorPaletteView–>". Set the following properties:

a) Name="cpWallColor"

b) HorizontalAlignment="Left"

c) MaxWidth="200"

d) Margin="20, 5, 20, 5"

e) ItemsSource="{Binding Source={StaticResource SmokyMountain}}"

f) PaletteOrientation="Vertical"

g) PaletteColumnsCount="3"

h) SelectionChanged="RadColorPaletteView_SelectionChanged"

Notes

In particular you should notice the ItemsSource property binding to the resource named "SmokyMountain". This refers to the ObjectCollection you defined earlier in the App.xaml file. Also notice the SelectionChanged event handler that we will define in a later step.

7) Drag a **RadColorSelector** control from the Toolbox to a place just under the comment "<!–-RadColorSelector-->". Set the following properties:

a) Name="csCabinets"

b) HorizontalAlignment="Left"

c) MaxWidth="200"

d) Margin="20, 5, 20, 5"

e) MainPalette="Concourse"

f) MainPaletteHeaderText="Cabinets"

g) HeaderPaletteVisibility="Collapsed"

h) StandardPalette="Concourse"

i) StandardPaletteHeaderText="Base Colors"

j) NoColorVisibility="Collapsed"

k) SelectedColorChanged="RadColorSelector_SelectedColorChanged"

Notes

Notice that the "No Color" button and the HeaderPalette visibility are set to "Collapsed" and will not be visible in the browser. We have also customized the MainPaletteText and StandardPaletteHeaderText. The MainPalette is set to the ColorPreset enumeration value "Concourse". Finally, we have a SelectedColorChanged event handler that we will define in a later step.

8) Drag a **RadColorPicker** control from the Toolbox to a place just under the comment "<!–-RadColorPicker-->". Set the following properties:

a) Name="cpAppliance"

b) HorizontalAlignment="Left"

c) MaxWidth="200"

d) Margin="20, 5, 20, 5"

e) SelectedColorChanged="RadColorPicker_SelectedColorChanged"

f) Click="RadColorPicker_Click"

Notes

Notice that there are event handlers for when the color first changes, or for when you click the Color button after that.

9) Add a "Loaded" event handler to the UserControl tag. *Later we will use this event handler to add a custom set of colors to the RadColorPicker main palette.*

```
<UserControl
...
   Loaded="UserControl_Loaded">
```

10) Right-click the "Loaded" event and select "Navigate to Event Handler" from the context menu. Add the code below to the Loaded event handler. *This code demonstrates how to create a custom palette of colors in code. A generic list of Color is created and populated, then assigned to the MainPaletteItemsSource property. The color picker control is configured in code to display only the main palette.*

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim colors As List(Of Color) = _
New List(Of Color) (New Color() {Colors.Blue, Colors.Red, Colors.Black})
  cpAppliance.MainPaletteHeaderText = "Appliance Colors"
  cpAppliance.MainPaletteItemsSource = colors
  cpAppliance.StandardPaletteVisibility = Visibility.Collapsed
  cpAppliance.HeaderPaletteVisibility = Visibility.Collapsed
  cpAppliance.NoColorVisibility = Visibility.Collapsed
  cpAppliance.SelectedColor = Colors.Blue
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   List<Color> colors = new List<Color>()
   {
      Colors.Blue,
      Colors.Red,
      Colors.Black
   };
   cpAppliance.MainPaletteHeaderText = "Appliance Colors";
   cpAppliance.MainPaletteItemsSource = colors;
   cpAppliance.StandardPaletteVisibility = Visibility.Collapsed;
   cpAppliance.HeaderPaletteVisibility = Visibility.Collapsed;
   cpAppliance.NoColorVisibility = Visibility.Collapsed;
   cpAppliance.SelectedColor = Colors.Blue;
}
```

11) Create a private method to update the user interface using the code below. *This method extracts the currently selected color for each color choosing control, then assigns those colors to the fill of several Rectangle controls. The end of the method creates a RadialGradientBrush using a combination of all the colors and assigns it to a rectangle.*

```
Private Sub UpdateUI()
  ' show the selected wall color
  If cpvWallColor.SelectedValue IsNot Nothing Then
    rectWallColor.Fill = New SolidColorBrush(CType(cpvWallColor.SelectedValue, Color))
  End If

  ' show the selected cabinet color
  rectCabinets.Fill = New SolidColorBrush(csCabinets.SelectedColor)

  ' show the selected appliance color

  ' show the selected appliance color, blended with the wall and cabinet color
  Dim stopCollection As New GradientStopCollection()

  If cpvWallColor.SelectedValue IsNot Nothing Then
    Dim stop1 As New GradientStop()
    stop1.Color = CType(cpvWallColor.SelectedValue, Color)
    stop1.Offset = 1
    stopCollection.Add(stop1)
  End If

  Dim stop2 As New GradientStop()
  stop2.Color = csCabinets.SelectedColor
  stop2.Offset = 0.5
  stopCollection.Add(stop2)

  Dim stop3 As New GradientStop()
  stop3.Color = cpAppliance.SelectedColor
  stop3.Offset = 0.2
  stopCollection.Add(stop3)

  rectCombined.Fill = New RadialGradientBrush(stopCollection)
End Sub
```

C#

```
private void UpdateUI()
{
  // show the selected wall color
  if (cpvWallColor.SelectedValue != null)
  {
    rectWallColor.Fill = new SolidColorBrush((Color)cpvWallColor.SelectedValue);
  }

  // show the selected cabinet color
  rectCabinets.Fill = new SolidColorBrush(csCabinets.SelectedColor);

  // show the selected appliance color

  // show the selected appliance color, blended with the wall and cabinet color
  GradientStopCollection stopCollection = new GradientStopCollection();

  if (cpvWallColor.SelectedValue != null)
  {
    GradientStop stop1 = new GradientStop();
    stop1.Color = (Color)cpvWallColor.SelectedValue;
    stop1.Offset = 1;
    stopCollection.Add(stop1);
  }

  GradientStop stop2 = new GradientStop();
  stop2.Color = csCabinets.SelectedColor;
  stop2.Offset = 0.5;
  stopCollection.Add(stop2);

  GradientStop stop3 = new GradientStop();
  stop3.Color = cpAppliance.SelectedColor;
  stop3.Offset = 0.2;
  stopCollection.Add(stop3);

  rectCombined.Fill = new RadialGradientBrush(stopCollection);
}
```

12)Call the UpdateUI() method from each event handler:

VB

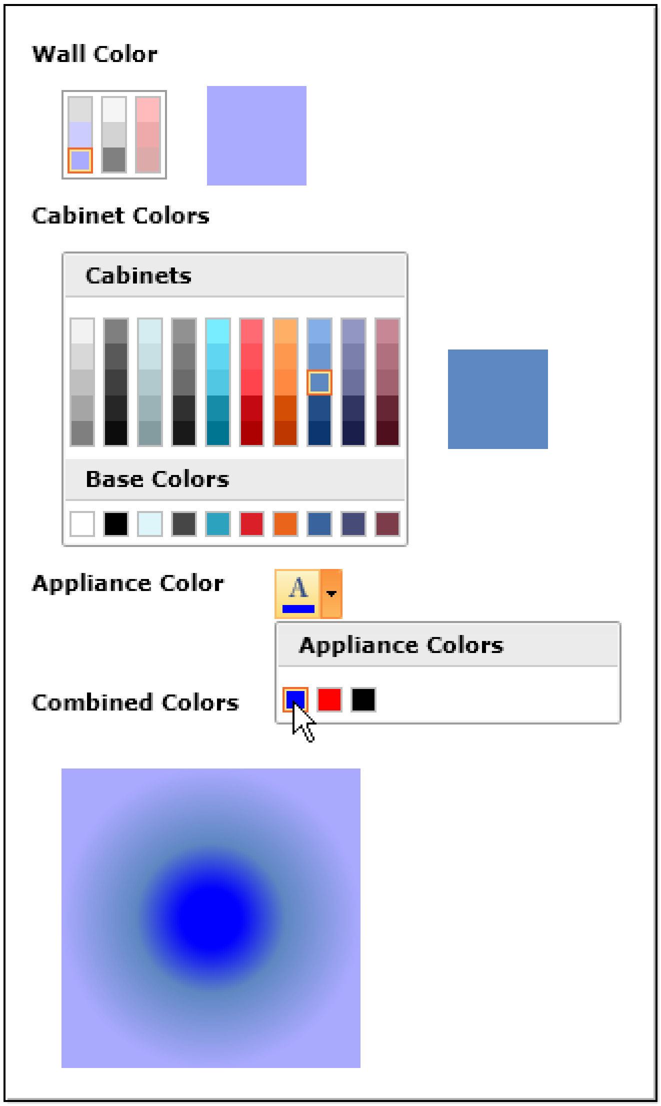
```
Private Sub RadColorPaletteView_SelectionChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  UpdateUI()
End Sub

Private Sub RadColorSelector_SelectedColorChanged( _
ByVal sender As Object, ByVal e As EventArgs)
  UpdateUI()
End Sub

Private Sub RadColorPicker_SelectedColorChanged( _
ByVal sender As Object, ByVal e As EventArgs)
  UpdateUI()
End Sub

Private Sub RadColorPicker_Click( _
ByVal sender As Object, ByVal e As EventArgs)
  UpdateUI()
End Sub
```

C#

```
private void RadColorPaletteView_SelectionChanged(
   object sender, Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   UpdateUI();
}

private void RadColorSelector_SelectedColorChanged(object sender, EventArgs e)
{
   UpdateUI();
}

private void RadColorPicker_SelectedColorChanged(object sender, EventArgs e)
{
   UpdateUI();
}

private void RadColorPicker_Click(object sender, EventArgs e)
{
   UpdateUI();
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



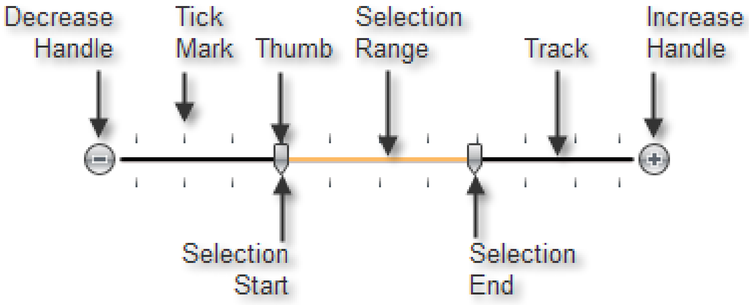
Test Application Features

- Verify that your custom list of colors defined in the App.xaml shows in the RadColorPaletteView under the "Wall Color" TextBlock.
- Verify that the RadColorSelector shows colors from the "Concourse" ColorPreset in both the main and standard palettes
- Verify that the RadColorPicker displays the limited set of colors you assigned in code.
- Check that the standard palette, header palette and "No Color" button have been hidden.
- Verify that the Rectangle at the bottom of the web page shows the combined set of colors in a radial gradient.

9.4.4 Slider Control

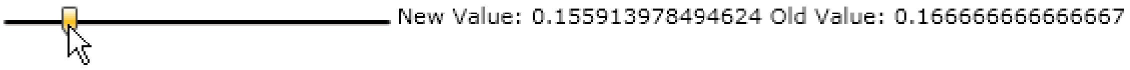
The RadSlider control lets users select a value from a defined range. The slider user interface makes the process easy, quick and intuitive. The control is completely customizable in terms of appearance and offers numerous configuration options like orientation, small change, mouse wheel support, snap to tick and tick placement. Tick Templates allow you to tailor the Tick appearance and to customize through binding.

This enhanced slider control can be configured to allow a selection range. The structure diagram below shows two "thumbs" that slide along a "track". The two thumbs delineate the start and end of the selection range.



Using the RadSlider Defaults

By default, RadSlider appears with a horizontal track and a single thumb. The range selection feature is not present, nor the increase/decrease handles at the far ends of the track, nor the tick marks. The thumb slides between a **Minimum** value "0" and a **Maximum** of "1".



New Value: 0.155913978494624 Old Value: 0.166666666666667

To use the slider defaults without the range selection feature enabled, handle the **ValueChanged** event. You can read the slider **Value** property from any location in code or use the event arguments **NewValue** and **OldValue** properties from within the ValueChanged event handler.

```
<StackPanel Orientation="Horizontal">

  <telerik:RadSlider
    MinWidth="200"
    ValueChanged="RadSlider_ValueChanged"></telerik:RadSlider>

  <TextBlock Name="tbSliderFeedback"></TextBlock>

</StackPanel>
```

VB

```
Private Sub RadSlider_ValueChanged( _
ByVal sender As Object, ByVal e As RoutedPropertyChangedEventArgs(Of Double))
  'tbSliderFeedback.Text = (sender as RadSlider).Value.ToString();
  tbSliderFeedback.Text = _
String.Format("New Value: {0} Old Value: {1}", e.NewValue, e.OldValue)
End Sub
```

C#

```
private void RadSlider_ValueChanged(
   object sender, RoutedPropertyChangedEventArgs<double> e)
{
   //tbSliderFeedback.Text = (sender as RadSlider).Value.ToString();
   tbSliderFeedback.Text = String.Format("New Value: {0} Old Value: {1}",
      e.NewValue, e.OldValue);
}
```

Using RadSlider With a Range

Once you change the **IsSelectionRangeEnabled** property to "true", then the game changes completely. Now you must handle the **SelectionRangeChanged** event and again look at the event argument's NewValue and OldValue properties. NewValue and OldValue are not "double" types in this situation, instead read the **SelectionStart** and **SelectionEnd** sub-properties.

XAML

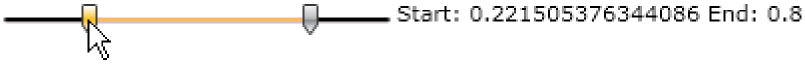
```
<telerik:RadSlider
   MinWidth="200"
   IsSelectionRangeEnabled="True"
   SelectionRangeChanged="RadSlider_SelectionRangeChanged">
</telerik:RadSlider>
```

VB

```
Private Sub RadSlider_SelectionRangeChanged( _
ByVal sender As Object,
ByVal e As RoutedPropertyChangedEventArgs(Of SelectionRangeChangedEventArgs))
  tbSliderFeedback.Text =
String.Format("Start: {0} End: {1}",
e.NewValue.SelectionStart, e.NewValue.SelectionEnd)

End Sub
```

```
private void RadSlider_SelectionRangeChanged(
   object sender, RoutedPropertyChangedEventArgs<SelectionRangeChangedEventArgs> e)
{
   tbSliderFeedback.Text = String.Format("Start: {0} End: {1}",
      e.NewValue.SelectionStart, e.NewValue.SelectionEnd);

}
```

Now the control displays thumbs for both selection start and end. By default the start value is "0.2" and the end is "0.8".



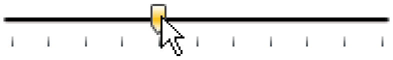
Start: 0.221505376344086 End: 0.8

Ticks

Default Behavior

To have tick marks placed along the track automatically, set the **EnableSideTicks** property to "true". By default the **TickPlacement** property is "None" so you won't see the ticks even with the EnableSideTicks property turned on. To show the ticks set TickPlacement to "TopLeft", "BottomRight" or "Both". To control the interval between ticks set the **TickFrequency** property to some value between the **Minimum** and **Maximum** property values that will display well. By default the thumb slides without interruption between tick marks but you can set the **IsSnapToTickEnabled** property so that the thumb jumps from one tick to the next.

```
<telerik:RadSlider
   MinWidth="200"
   EnableSideTicks="True"
   TickFrequency="0.1"
   TickPlacement="BottomRight"
   IsSnapToTickEnabled="True"
   >
</telerik:RadSlider>
```

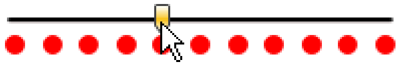


Templates

If the default tick display doesn't work for you, a **TickTemplate** defines what a single tick looks like. Inside the TickTemplate tag add a DataTemplate tag. Within the DataTemplate you can add whatever markup you want to represent the tick mark. The example markup below adds a 10 x 10 pixel red ellipse to represent a tick mark. Pay particular attention to the TickTemplate tag and syntax.

```
<telerik:RadSlider
  MinWidth="200"
  EnableSideTicks="True"
  TickFrequency="0.1"
  TickPlacement="BottomRight"
  IsSnapToTickEnabled="True">
  <telerik:RadSlider.TickTemplate>
    <DataTemplate>
      <Ellipse
        Width="10"
        Height="10"
        Fill="Red"></Ellipse>
    </DataTemplate>
  </telerik:RadSlider.TickTemplate>

</telerik:RadSlider>
```

The markup ends up looking like this at runtime:

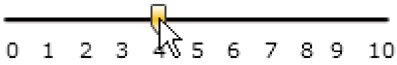


Templates and Binding

You can go another step by binding within the template. For example, to simply show the values directly next to the ticks, assign the expression "{Binding}" to the Text property of a TextBlock. Notice in this XAML example that the Minimum and Maximum are zero to ten and that TickFrequency is "1".

```
<telerik:RadSlider
    MinWidth="200"
    EnableSideTicks="True"
    Minimum="0"
    Maximum="10"
    TickFrequency="1"
    TickPlacement="BottomRight"
    IsSnapToTickEnabled="True">
    <telerik:RadSlider.TickTemplate>
        <DataTemplate>
            <Grid>
                <TextBlock
                    Text="{Binding}"
                    FontSize="11" />
            </Grid>
        </DataTemplate>
    </telerik:RadSlider.TickTemplate>
</telerik:RadSlider>
```

If you leave the default "0.1" to "1", the intervening double values are very long and don't display well. Setting TickFrequency to "1" outputs an array of integers:



Templates and Custom Binding

You can incorporate custom coding per tick mark using what one blogger called the "The Swiss Army Knife of Bindings"*, the versatile **IValueConverter** interface. IValueConverter lets you implement a Convert method that outputs an object of your choosing. For example, we could output the longhand name of the number instead of the digit. The steps are:

- Create a class that implements IValueConverter.
- Reference the project containing the class in the MainPage.xaml where the slider is being used.
- Create a resource for the UserControl that points to the class and gives it a name.
- Bind to the converter.

IValueConverter comes from the System.Windows.Data namespace.

* "IValueConverter: The Swiss Army Knife of Bindings", Delays Blog

Walk Through

This walk-through demonstrates how IValueConverter can accept a value and output a string.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

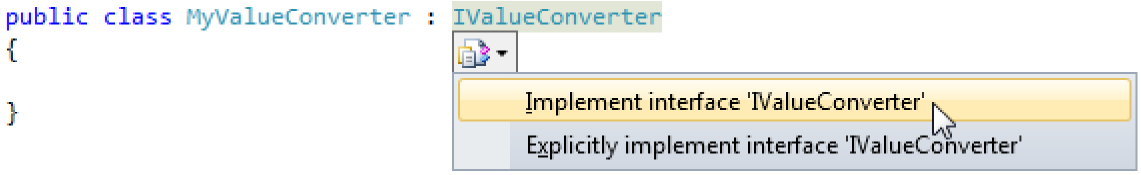
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

Implementing and Using IValueConverter

1) In Visual Studio, right-click the project and select **Add > Class...** from the context menu. Name the class file "MyValueConverter.cs".

1) Verify that references to the **System.Windows.Data** and **System.Collections.Generic** namespaces are included in the "Imports" (VB) or "using" (C#) section of code.

2) Name the class MyValueConverter and inherit from the IValueConverter interface. Right-click the IValueConverter reference and select "Implement Interface" from the context menu.



```
public class MyValueConverter : IValueConverter
{

}
```

Implement interface 'IValueConverter'
Explicitly implement interface 'IValueConverter'

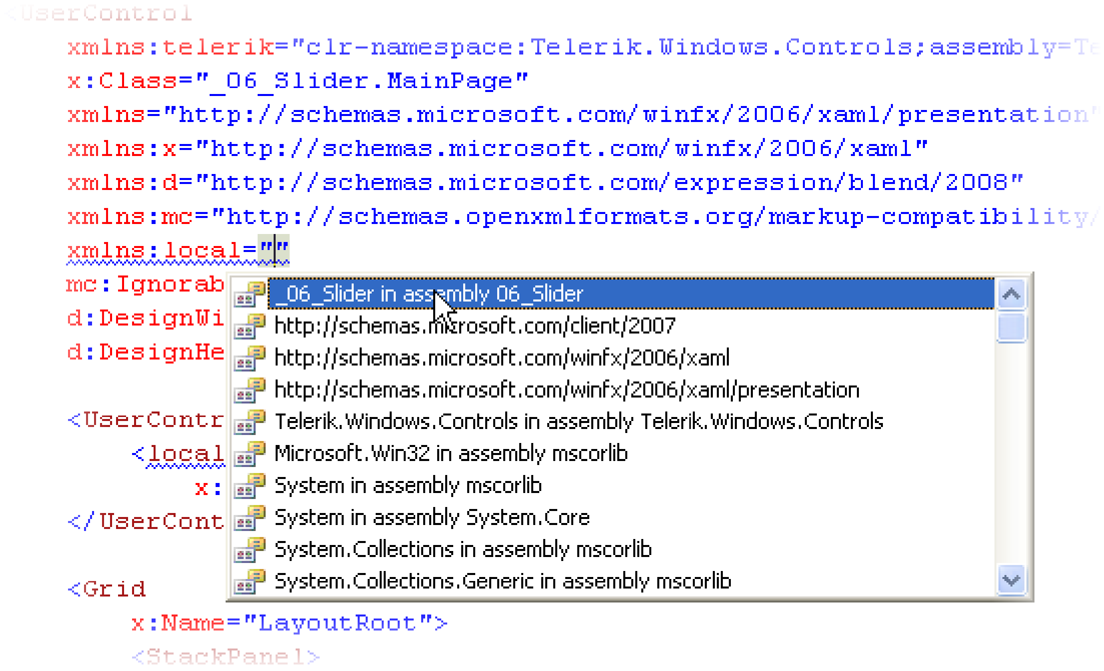
3) Modify the Convert() method to use the code below. This code creates a generic list of strings with the proper number names and indexes into this list using the "value" parameter passed in. "value" is the value for the tick mark being represented.

```
Public Class MyValueConverter
  Implements IValueConverter

  #Region "IValueConverter Members"

  Public Function Convert(ByVal value As Object, _
ByVal targetType As Type, _
ByVal parameter As Object, _
ByVal culture As System.Globalization.CultureInfo) As Object
    Dim numbers As List(Of String) = New List(Of String) (New String() _
{"Zero", "One", "Two", "Three", "Four", _
"Five", "Six", "Seven", "Eight", "Nine", "Ten"})

    Return numbers(CInt(Fix(CDbl(value))))
  End Function

  Public Function ConvertBack(ByVal value As Object, _
ByVal targetType As Type, _
ByVal parameter As Object, _
ByVal culture As System.Globalization.CultureInfo) As Object
    Throw New NotImplementedException()
  End Function

  #End Region
End Class
```

```
public class MyValueConverter : IValueConverter
{

    #region IValueConverter Members

    public object Convert(object value,
      Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        List<string> numbers = new List<string>()
    {
        "Zero", "One", "Two", "Three", "Four", "Five",
        "Six", "Seven", "Eight", "Nine", "Ten"
    };

        return numbers[(int)(double)value];
    }

    public object ConvertBack(object value,
      Type targetType, object parameter, System.Globalization.CultureInfo culture)
    {
        throw new NotImplementedException();
    }

    #endregion
}
```

4) Open the MainPage.xaml file for editing.

5) At the top of the file in the UserControl tag, add a reference to your project. To do this, type in "xmlns" (XML name space) and a colon. After the colon put in an alias for the namespace; in this case the alias will be called "local". Once you type in the equal sign ("=") a list of assemblies in scope will drop down. Select the project that contains the IValueConverter implementation class. In the screenshot below the project is "06_Slider".



```
<UserControl
    xmlns:telerik="clr-namespace:Telerik.Windows.Controls;assembly=Te
    x:Class="_06_Slider.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/
    xmlns:local=""
    mc:Ignorab
    d:DesignWi
    d:DesignHe

    <UserContr
        <local
            x:
    </UserCont

    <Grid
        x:Name="LayoutRoot">
        <StackPanel>
```

_06_Slider in assembly 06_Slider
http://schemas.microsoft.com/client/2007
http://schemas.microsoft.com/winfx/2006/xaml
http://schemas.microsoft.com/winfx/2006/xaml/presentation
Telerik.Windows.Controls in assembly Telerik.Windows.Controls
Microsoft.Win32 in assembly mscorlib
System in assembly mscorlib
System in assembly System.Core
System.Collections in assembly mscorlib
System.Collections.Generic in assembly mscorlib

6) Inside the UserControl element, add a resource that references your new MyValueConverter class. Use the "x:Key" attribute to identify the class for later use in the XAML.

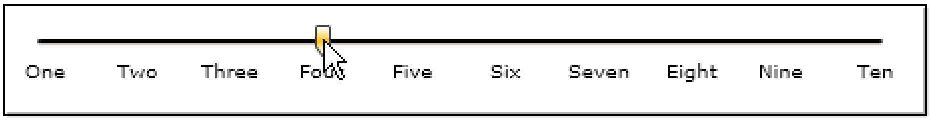
```
<UserControl.Resources>
  <local:MyValueConverter
    x:Key="MyValueConverter" />
</UserControl.Resources>
```

7) In the DataTemplate add the XAML below to define the RadSlider and the binding using the converter. *Notice the binding expression "{Binding Converter={StaticResource MyValueConverter}}" that references the MyValueConverter through the resource.*

```
<telerik:RadSlider
  Margin="30"
  Minimum="1"
  Maximum="10"
  TickFrequency="1"
  TickPlacement="BottomRight"
  IsSnapToTickEnabled="True">
  <telerik:RadSlider.TickTemplate>
    <DataTemplate>
      <TextBlock
        Text="{Binding Converter={StaticResource MyValueConverter}}"
        FontSize="10">
      </TextBlock>
    </DataTemplate>
  </telerik:RadSlider.TickTemplate>
</telerik:RadSlider>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

- Verify that the converter has supplied the names for the values.

Ideas for Extending This Example

- Use layout panels to make use of multiple elements for each tick
- Add an Ellipse above the text
- Rotate text items using a RenderTransform tag.

9.5 Customization

Walk Through

Customizing controls by editing styles and control templates involves a fair amount of XAML, but you still make these changes easily using Expression Blend. In this example we will customize the RadSlider control background and thumb.

"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

3) Save and close the solution.

Edit the Project in Expression Blend

1) Run Expression Blend and from the **File** menu select **Open Project/Solution...**

2) Locate the project you created in Visual Studio. *Note: you can also create the project directly in Expression Blend.*

3) In the **Projects** tab, locate MainPage.xaml and double-click to open it in the Expression Blend designer.

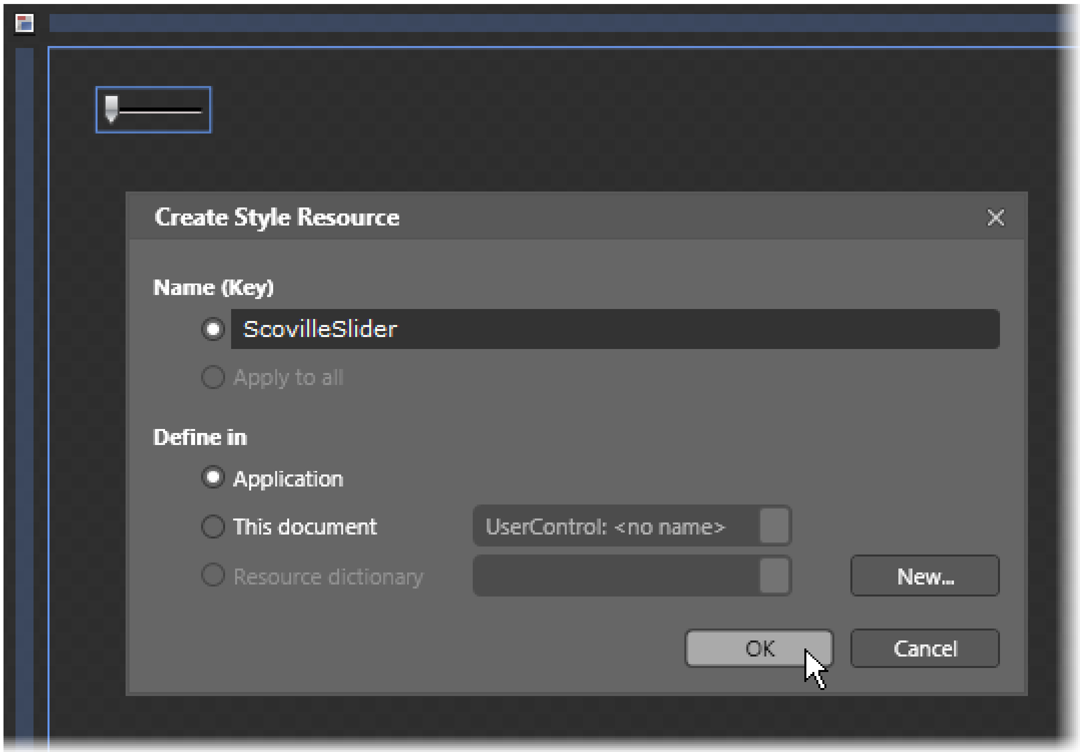
4) In the **Projects** tab, right-click the **References** node and select **Add Reference...** from the context menu.

5) Add a reference to the Telerik.Windows.Controls.dll assembly.

6) From the **Project** menu select **Build Project**.

7) Click on the **Assets** tab. On the left side of the of the area below the Assets tab is a tree view. Locate and open the "Controls" node, then click the "All" node. A list of all available controls shows to the right of the tree view. Locate the **RadSlider** control and drag it onto the MainPage.xaml design view.

8) Right-click the slider and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleSlider" and click the "Define in Application" radio button. Click **OK** to create the stye resource and close the dialog.



Create Style Resource

Name (Key)

● ScovilleSlider

○ Apply to all

Define in

● Application

○ This document UserControl: <no name>

○ Resource dictionary

Notes

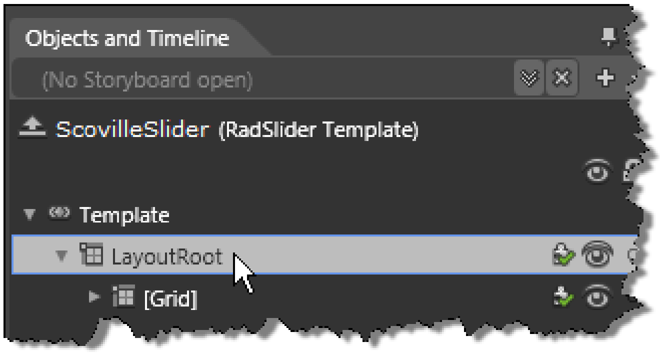
1) This last step opens just the RadSlider template for editing, not the RadSlider instance on the MainPage.xaml. The MainPage.xaml RadSlider has a reference to the "ScovilleSlider" style that looks something like the XAML below. Notice the "Style" attribute points at "StaticResource ScovilleSlider".

```
<telerik:RadSlider HorizontalAlignment="Left" Margin="24,20,0,0" VerticalAlignment="Top" Style="{Stat
```

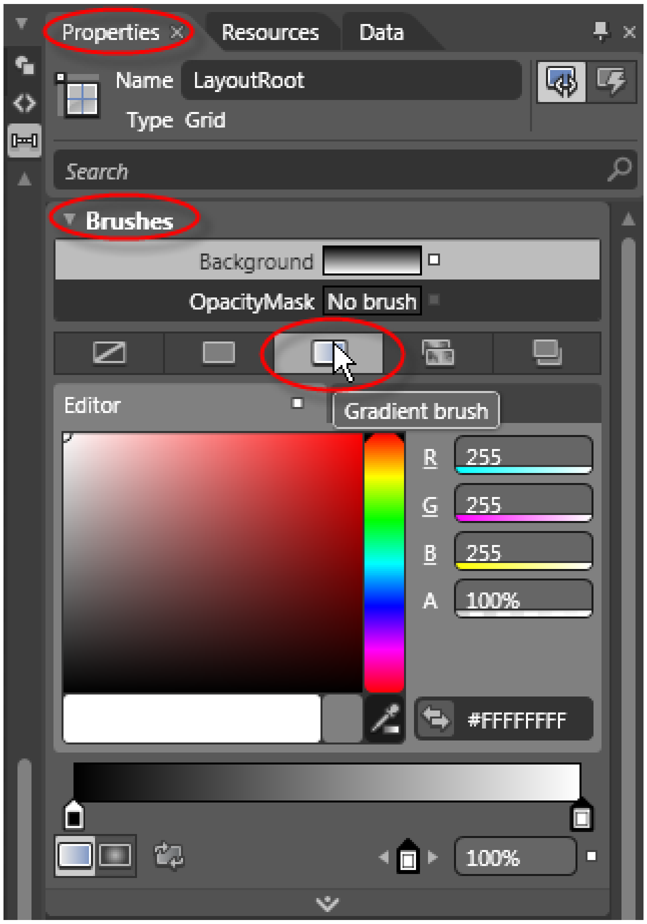
In the App.xaml file, the complete definition of the "ScovilleSlider" style has been copied from the RadSlider defaults and looks something like the small excerpt below:

```
<Style x:Key="ScovilleSlider" TargetType="telerik:RadSlider">
  ...
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="telerik:RadSlider">
        <Grid x:Name="LayoutRoot"
         HorizontalAlignment="{TemplateBinding HorizontalAlignment}"
         VerticalAlignment="{TemplateBinding VerticalAlignment}">
         ...
```

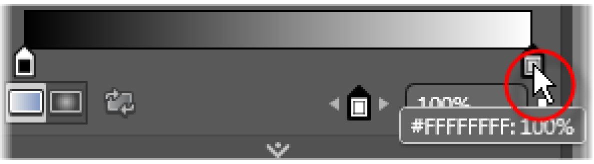
9) In the **Objects and Timeline** tab, select the LayoutRoot from the tree view.



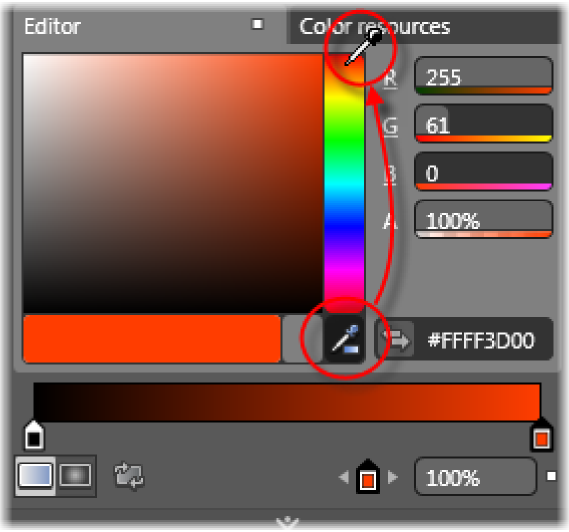
10)In the **Properties** tab, locate the **Brushes** group and select the "Gradient" brush.



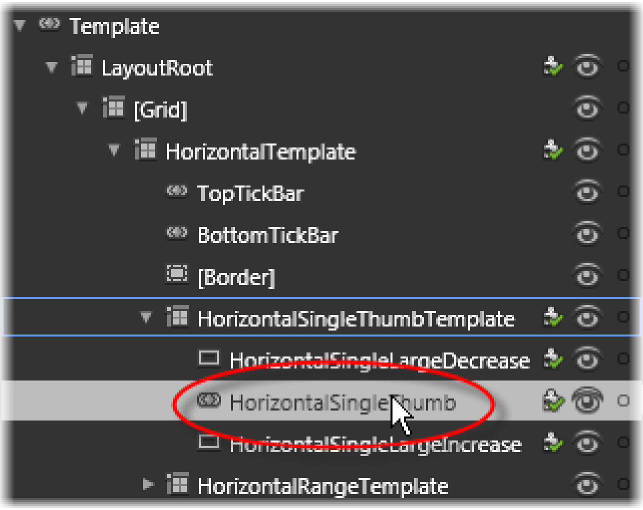
11)Find the right-most gradient stop indicator and click it with the mouse.



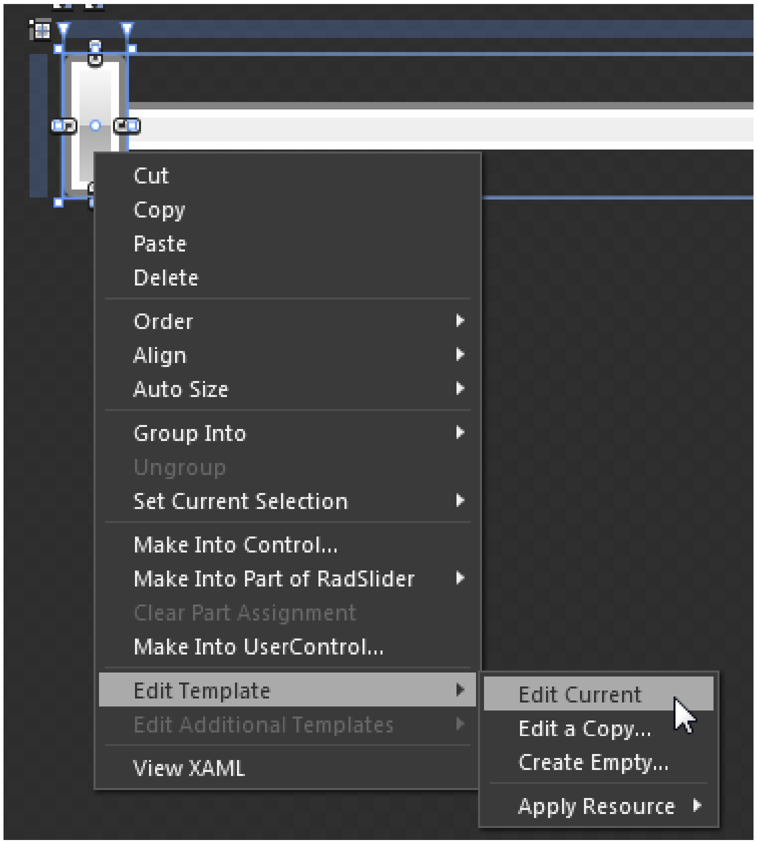
12) Grab the eye dropper tool and select a red color for the gradient stop.



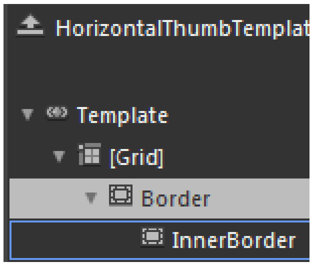
13) In the **Objects and Timeline** tab, select the "HorizontalSingleThumb" item.



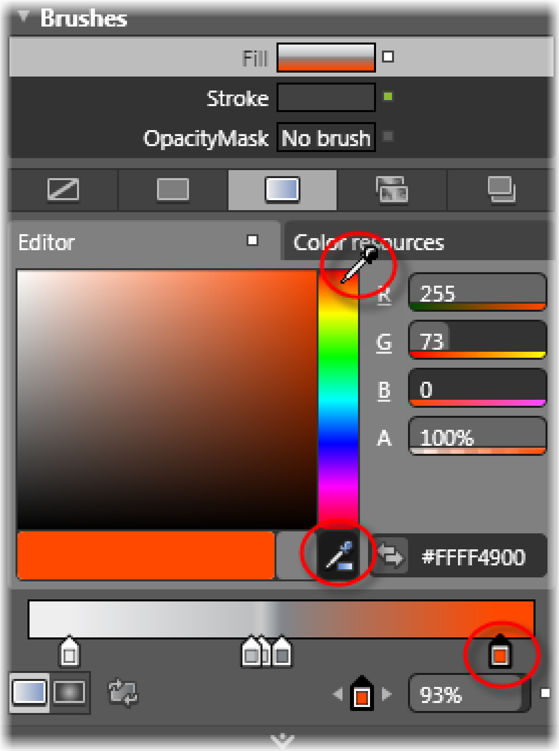
14) In the Expression Blend design surface, right-click the slider thumb and select **Edit Template > Edit Current** from the context menu. *As a result, only the slider thumb will be shown in the designer.*



15) In the **Objects and Timeline** tab, select "Border" from the tree view.



16) In the **Properties** tab, locate the **Brushes** section and again select the "Gradient" brush. Click the right-most gradient stop indicator and use the eye dropper tool to select a red color.

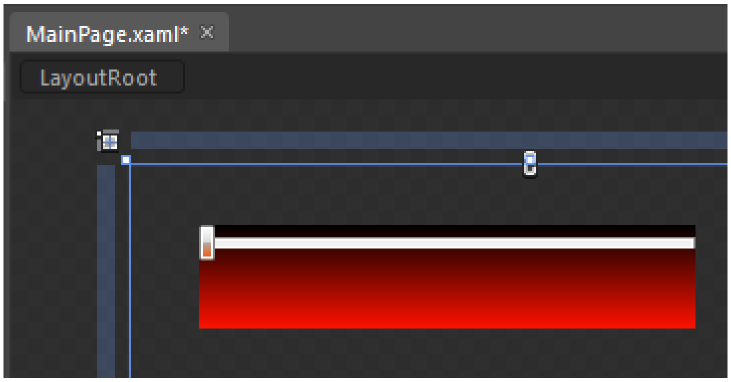


R 255, G 73, B 0, A 100%

#FFFF4900

93%

17) In the Projects tab, locate MainPage.xaml and double-click to display the RadSlider with the styling in play.

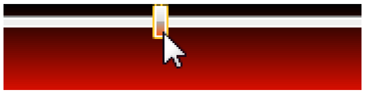


18) Save and close the project from Expression Blend.

19) Reopen the solution in Visual Studio.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

1) Notice that the slider functionality is completely untouched by all this styling work. The designer didn't need to know anything about how the slider works or what would be done with the slider within the application.
2) Open the MainPage.xaml and notice the Style reference in the RadSlider tag. No other styling information is kept directly with the slider. Open the App.xaml to see all the application resources that support the RadSlider look and feel.

Ideas for Extending This Example

- Add some logic to the application to verify that the slider events and behavior work as before the styling was applied.
- Add a RadMaskedTextBox to project and create a matching style for it in Expression Blend.

9.6 Wrap Up

This chapter introduced several controls used for gathering user input, including a masked text box for restricting entry to predefined patterns, a set of "UpDown" controls for entering numeric or other input through repeater buttons, a set of color picking controls for selecting from a predefined palette of colors and a slider control for choosing values within a range.

In this chapter you learned how the masked text box handles numeric, date/time and developer-defined formats using predefined masks or custom masks. We looked at important events for retrieving values and also learned how to retrieve the value combined together with the mask. You also saw how masked text boxes can be localized.

In the section on "UpDown" controls you saw how both the basic up-down control and numeric up-down control both inherit from the RangeBase class. You learned the key common properties and events for both controls. We discussed the key numeric up-down control ValueFormat property and some special case issues such as scrolling through integers. You learned how to format numeric entry for data that uses custom unit types.

We reviewed three different types of color pickers along with their common properties and events. We made use of the Silverlight Toolkit and the ObjectCollection class to define a custom palette of colors in XAML and also defined a custom palette in code. You learned how to customize text labels for elements of a color choosing control and how to hide specific elements of these controls.

The last part of this chapter dealt with how the slider control is used to pick a value within a range or a range within a range. We talked about the basic structure and parts of a slider, the slider's default behavior, how to retrieve new and previous values and how to control selection ranges. We paid particular attention to how tick marks are placed along the slider, how to control tick mark frequency and how to control visibility. We delved a little more deeply into how templates allow fine-tune control over each tick mark, how to use binding in templates to show data in a tick mark and finally how IValueConverter is used for highly customized binding scenarios.

Part X

Menu Controls

10 Menu Controls

10.1 Objectives

In this chapter you will use RadMenu, RadContextMenu and RadMenuItem to supply drop-down and pop-up lists of user selections. You will see how MenuBase supplies properties common to both RadMenu and RadContextMenu. You will build standard menus using RadMenu and use properties to control item opening behavior and check/uncheck support. You will learn how to respond to user selections, create menus dynamically in code and add images to menu items. Then you will apply these fundamentals to RadContextMenu and explore context menu specific issues such as initiating the popup, placement, sizing and right-click support. Finally, you will use Expression Blend to customize the appearance of a RadMenu and its items.

Find the projects for this chapter at...

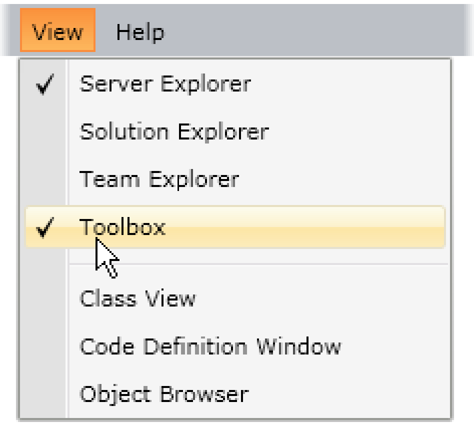
\Courseware\Projects\<CS|VB>\Menu\Menu.sln.

10.2 Overview

Telerik Menu for Silverlight lets you build powerful, complex menu systems that are easy for your customer to use. The control set includes a "main menu" style drop-down control and a context menu that can be associated with a control or popped up from any location.

The menu controls are fully customizable and have advanced functionality including:

- **Hierarchical Data Binding**: The menu controls bind to objects created in code-behind or from a web service.
- **Boundary Detection**: The menu controls detect the Silverlight plugin boundaries and open child items in the opposite direction to avoid crossing screen boundaries. When there is not enough space in all directions, the control adjusts its items' positions to make them visible whenever possible.
- **Checkable Items**: Menu items can have check marks that can be toggled. Companion properties like ClickToOpen and StaysOpenOnClick enhance usability.



- The RadMenu control **Orientation** can be Horizontal or Vertical..

10.3 Getting Started

In this walk through you will create a simple main menu at the top of the web page.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

   a) **Telerik.Windows.Controls**

   b) **Telerik.Windows.Controls.Navigation**

XAML Editing

1) Open MainPage.xaml for editing.
2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags.

```xml
<StackPanel>

  <!--Menu-->

</StackPanel>
```

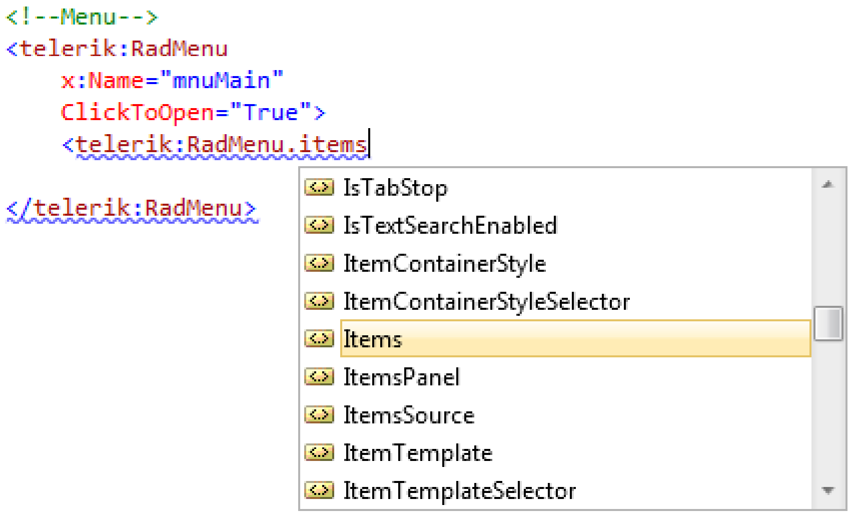
3) Drag a RadMenu control from the Toolbox to a point just under the "<!--Menu-->" comment. Set the following RadMenu properties:

   a) **x:Name** = "mnuMain"

   b) **ClickToOpen** = "True"

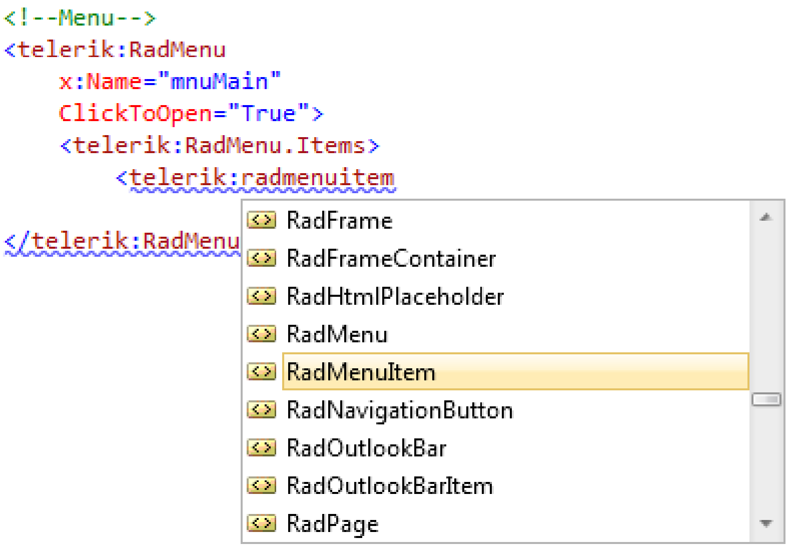
With the ClickToOpen property set true, the menu does not open on mouse over, but rather when the mouse button is clicked.

4) Inside the RadMenu tag, add a RadMenu.Items tag. *As of this writing, the namespace for this tag is "telerik".*



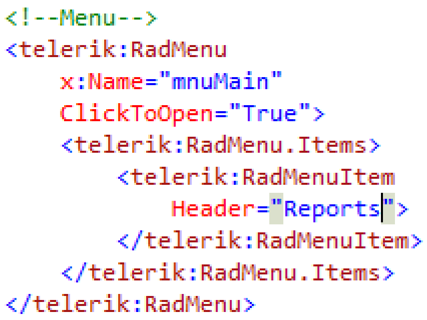
```
<!--Menu-->
<telerik:RadMenu
    x:Name="mnuMain"
    ClickToOpen="True">
    <telerik:RadMenu.items

</telerik:RadMenu>
```

5) Inside the RadMenu.Items tag, add a RadMenuItem tag.



```
<!--Menu-->
<telerik:RadMenu
    x:Name="mnuMain"
    ClickToOpen="True">
    <telerik:RadMenu.Items>
        <telerik:radmenuitem

</telerik:RadMenu
```

6) Set the RadMenuItem **Header** property to "Reports". The markup should now look something like the figure below:



```
<!--Menu-->
<telerik:RadMenu
    x:Name="mnuMain"
    ClickToOpen="True">
    <telerik:RadMenu.Items>
        <telerik:RadMenuItem
            Header="Reports">
        </telerik:RadMenuItem>
    </telerik:RadMenu.Items>
</telerik:RadMenu>
```

7) Add a second RadMenuItem with the **Header** property set to "Options".

8) Inside the "Reports" RadMenuItem, add two more RadMenuItem tags. Set the properties as follows:

a) **Header** = "Product Listing", **Click** = "Reports_Click".

b) **Header** = "Products by Category", **Click** = "Reports_Click".

9) Inside the "Options" RadMenuItem, add a RadMenuItem tag with properties set as follows:

a) **Header** = "Print Two Sided"

b) **StaysOpenOnClick** = "True"

c) **IsCheckable** = "True"

d) **Click**="OptionsMenuItem_Click"

The StaysOpenOnClick property set to true makes the "Print Two Side" menu item stay visible while IsCheckable allows the user to toggle the check mark back and forth. These two properties work well together and also combine nicely with the RadMenu ClickToOpen property.

The completed markup for the RadMenu is shown below.

```xml
<!--Menu-->
<telerik:RadMenu
  x:Name="mnuMain"
  ClickToOpen="True">
  <telerik:RadMenu.Items>
    <telerik:RadMenuItem
      Header="Reports">
      <telerik:RadMenuItem
        Header="Product Listing"
        Click="Reports_Click">
      </telerik:RadMenuItem>
      <telerik:RadMenuItem
        Header="Products by Category"
        Click="Reports_Click">
      </telerik:RadMenuItem>
    </telerik:RadMenuItem>
    <telerik:RadMenuItem
      Header="Options">
      <telerik:RadMenuItem
        Header="Print Two Sided"
        Click="OptionsMenuItem_Click"
        StaysOpenOnClick="True"
        IsCheckable="True">
      </telerik:RadMenuItem>
    </telerik:RadMenuItem>
  </telerik:RadMenu.Items>
</telerik:RadMenu>
```

Code Behind

1) In the code-behind, verify that references to the **Telerik.Windows.Controls** and **Telerik.Windows** namespaces are included in the "Imports" (VB) or "using" (C#) section of code. Add these references if they do not exist.

2) In the code-behind, add the "Reports_Click" and "OptionsMenuItem_Click" event handling code. *The code here simply shows the object that represents the clicked item. We could have used the System MessageBox.Show() method to display the class name, but instead we're using the RadWindow.Alert() function.*

```vb
Private Sub Reports_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadWindow.Alert((TryCast(e, RadRoutedEventArgs)).Source.ToString())
End Sub

Private Sub OptionsMenuItem_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadWindow.Alert((TryCast(e, RadRoutedEventArgs)).Source.ToString())
End Sub
```

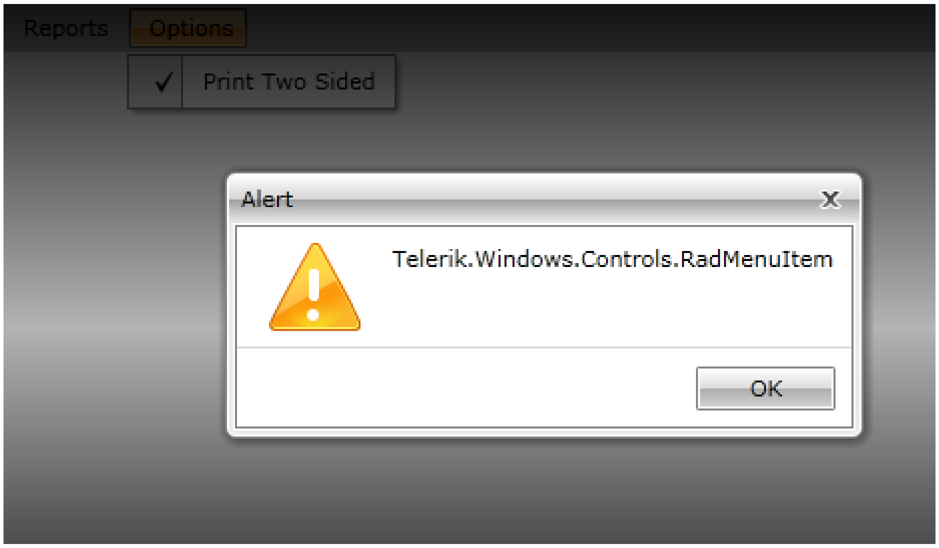
```
private void Reports_Click(object sender, RoutedEventArgs e)
{
   RadWindow.Alert((e as RadRoutedEventArgs).Source.ToString());
}

private void OptionsMenuItem_Click(object sender, RoutedEventArgs e)
{
   RadWindow.Alert((e as RadRoutedEventArgs).Source.ToString());
}
```

Notes

The **RoutedEventArgs** are seen everywhere in Silverlight code. In the example above these arguments are cast as **RadRoutedEventArgs** classes that retain more information than their RoutedEventArgs ancestors.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Telerik.Windows.Controls.RadMenuItem

Test Application Features

1) Notice that the menu items must be mouse clicked to open.
2) Try clicking the "Print Two Sided" menu item. Notice that it stays open when clicked and that the check mark toggles.
3) Notice that the RadWindow.Alert() method triggered by the Click event handlers displays the "RadMenuItem" class name.

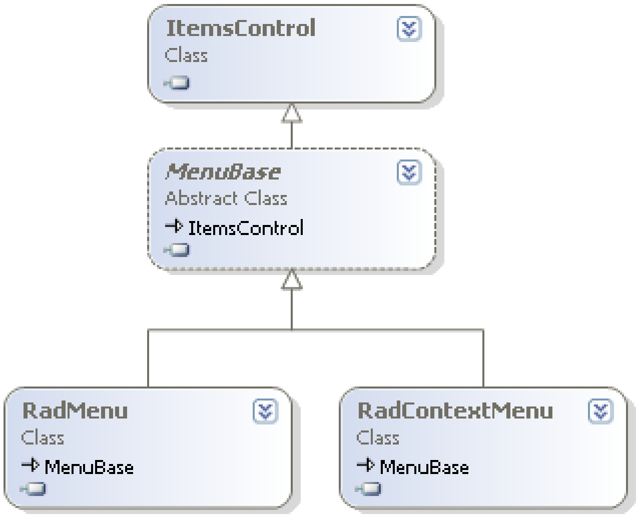
Ideas for Extending This Example

- Try extracting information from the RadMenuItem available in the Click event and displaying that information in the alert.

10.4 Control Details

10.4.1 RadMenu

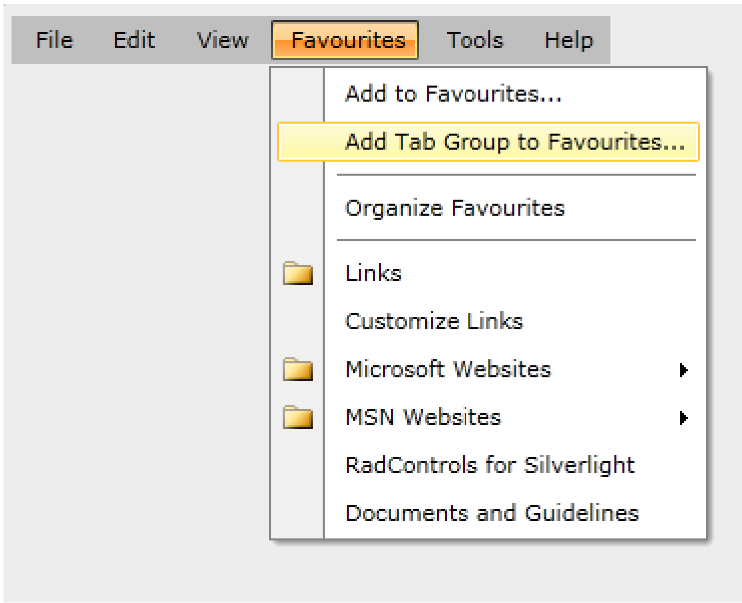
Both RadMenu and RadContextMenu descend from **MenuBase** and ultimately, **ItemsControl**. ItemsControl is the base class for objects that have multiple items.



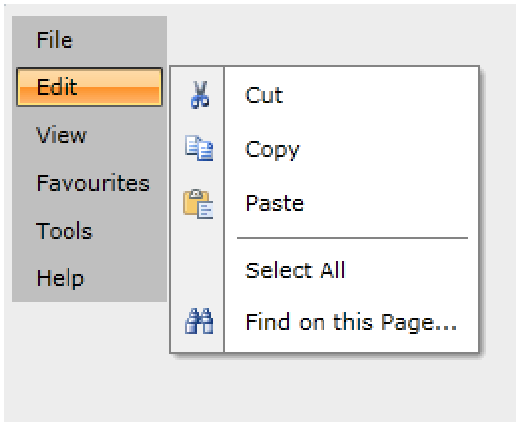
MenuBase introduces basic menu functionality where users can choose from a drop down list of items. MenuBase includes common events and properties:

- **ItemClick**: This event fires when any item in the menu is clicked.
- **ClickToOpen**: We saw in the "Getting Started" project how the ClickToOpen property, when true, prevents the menu from opening merely from mouse-over, but instead requires a mouse click to open the menu.
- **ShowDelay**, **HideDelay**: These are both Duration properties that define a number of milliseconds before the menu is shown or hidden.
- **NotifyOnHeaderClick** when true causes the ItemClick event to fire even when the menu header is clicked.

RadMenu adds an **Orientation** property that can be **Horizontal...**



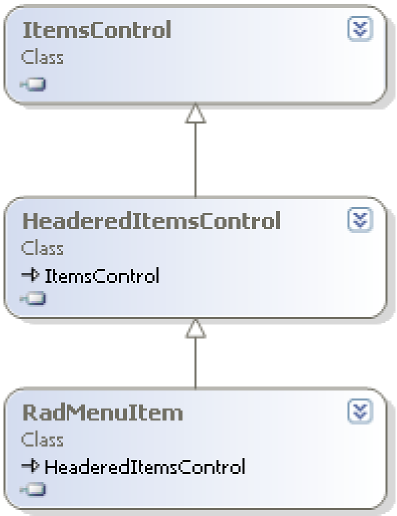
or **Vertical**



10.4.2 Items

Overview

Both RadMenu and RadContextMenu have an **Items** collection that contain a series of **RadMenuItem** objects. RadMenuItem descends from **HeaderedItemsControl** (representing a control that has multiple items and a header).



Here are a few significant properties of the RadMenuItem class:

- The **Role** property for RadMenuItem determines the menu item behavior, particularly if the item can invoke commands. Role can be **TopLevelItem**, **TopLevelHeader**, **SubMenuItem**, **SubMenuHeader** and **Separator**. TopLevelItem and SubLevelItem can invoke commands.
- **IsCheckable** when true determines that the item check mark can be toggled. **IsChecked** reflects the current checked state of the item and can also be set in code or XAML.
- **Icon** is an object property that can be assigned a Silverlight BitmapImage object that will display in the menu item.
- **StaysOpenOnClick** causes the menu item to stay open even after the click and is especially useful for checkable menu items.
- Set **IsSeparator** to true when the item should visually divide other menu items but not be able to invoke commands.

RadMenuItem has routed events for **Checked**, **Click**, **SubmenuClosed**, **SubmenuOpened** and **Unchecked**.

Using the Items Collection

To add an item to the menu, create a new RadMenuItem, set the item Header property to some value and add it to the Items collection.

VB

```
Dim item As New RadMenuItem()
item.Header = "Calendar"
mnuMain.Items.Add(item)
```

C#

```
RadMenuItem item = new RadMenuItem();
item.Header = "Calendar";
mnuMain.Items.Add(item);
```

To display an image in the menu item, assign a BitmapImage to the Icon property:

VB

```
Dim dayItem As New RadMenuItem()
dayItem.Icon = New Image() _
With {.Source = _
New BitmapImage(New Uri("Calendar.png", UriKind.Relative))}
dayItem.Header = day

myMenu.Items.Add(dayItem)
```

C#

```
RadMenuItem dayItem = new RadMenuItem();
dayItem.Icon = new Image()
{
   Source = new BitmapImage(new Uri("Calendar.png", UriKind.Relative))
};
dayItem.Header = day;

myMenu.Items.Add(dayItem);
```

Notes

BitmapImage is a member of the System.Windows.Media.Imaging namespace.

10.4.3 Walk Through: Creating Menu Items in Code

This walk through will give you some practice at creating menu items with text and icons in code.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

4) Drag the image file "Calendar.png" from the Windows Explorer to the project in Solution Explorer. You can find "Calendar.png" in the "\courseware\images" directory.

XAML Editing

1) Open MainPage.xaml for editing.

2) Add a Loaded event handler to the UserControl element.

```
<UserControl
...
  Loaded="UserControl_Loaded">
```

3) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. The "<!–menu–>" comment will be replaced with the actual RadMenu tag in the next step.

```
<StackPanel>
  <!--menu-->
</StackPanel>
```

4) Drag a RadMenu from the Toolbox to a point just below the "<!–menu–>". Set the following properties:

a) **x:Name** = "mnuMain"

b) **ItemClick** = "mnuMain_ItemClick"

Code Behind

1) In the code-behind, add the Loaded event handler as shown below. *The handler first defines a single top level RadMenuItem with Header "Days" and adds it to the RadMenu Items collection. Then the handler defines an array of short day names, creates RadMenuItem instances, assigns Icon and Header for each before adding the instances to the "Days" menu item.*

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim daysHeaderItem As New RadMenuItem()
  daysHeaderItem.Header = "Days"
  mnuMain.Items.Add(daysHeaderItem)

  Dim days() As String = { "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" }

  For Each day As String In days
    Dim dayItem As New RadMenuItem()
    dayItem.Icon = New Image() With _
{.Source = New BitmapImage(New Uri("Calendar.png", UriKind.Relative))}
    dayItem.Header = day

    daysHeaderItem.Items.Add(dayItem)
  Next day
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    RadMenuItem daysHeaderItem = new RadMenuItem();
    daysHeaderItem.Header = "Days";
    mnuMain.Items.Add(daysHeaderItem);

    string[] days = new string[]
     { "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" };

    foreach (string day in days)
    {
        RadMenuItem dayItem = new RadMenuItem();
        dayItem.Icon = new Image()
        {
            Source = new BitmapImage(new Uri("Calendar.png", UriKind.Relative))
        };
        dayItem.Header = day;

        daysHeaderItem.Items.Add(dayItem);
    }
}
```

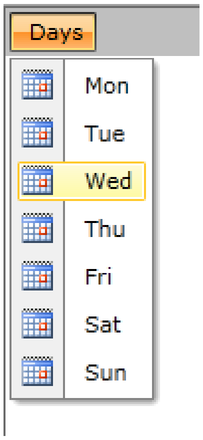
2) In the code-behind, add the handler for the menu's ItemClick event.

```
Private Sub mnuMain_ItemClick(ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  Dim item As RadMenuItem = TryCast(e.Source, RadMenuItem)
  RadWindow.Alert(item.Header.ToString())
End Sub
```

```
private void mnuMain_ItemClick(object sender, Telerik.Windows.RadRoutedEventArgs e)
{
   RadMenuItem item = e.Source as RadMenuItem;
   RadWindow.Alert(item.Header.ToString());
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



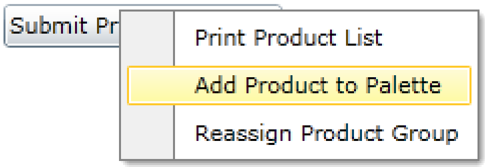
10.4.4 RadContextMenu

Using RadContextMenu

To use RadContextMenu in XAML, attach it to any UIElement using the **RadContextMenu.ContextMenu** property. The sample below shows a standard Silverlight Button. Inside the button tag is the RadContextMenu.ContextMenu tag, then the RadContextMenu tag with its ItemClick event defined and finally, a series of RadMenuItem tags.

```
<Button
  Content="Submit Product Request"
  MaxWidth="200"
  HorizontalAlignment="Left"
  Margin="20"
  >

  <telerik:RadContextMenu.ContextMenu>
    <telerik:RadContextMenu ItemClick="RadContextMenu_ItemClick">
      <telerik:RadMenuItem
        Header="Print Product List"></telerik:RadMenuItem>
      <telerik:RadMenuItem
        Header="Add Product to Palette"></telerik:RadMenuItem>
      <telerik:RadMenuItem
        Header="Reassign Product Group"></telerik:RadMenuItem>
    </telerik:RadContextMenu>
  </telerik:RadContextMenu.ContextMenu>

</Button>
```

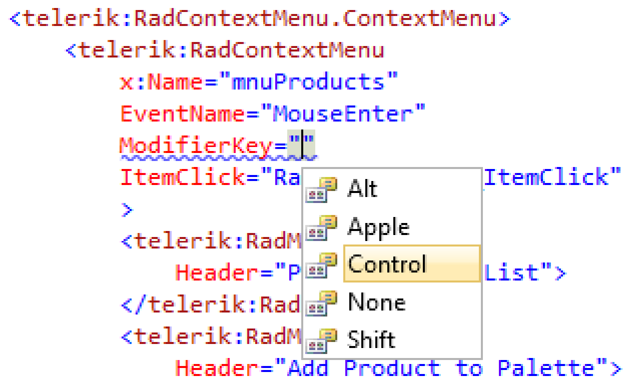
The context menu shows when the button is right-clicked by the mouse and looks something like the screenshot below:



The context menu can display in response to any event by specifying the **EventName** property. In this Button example, we can use the MouseEnter event so that when the mouse passes over the button, the menu displays.

```
<Button>
  <telerik:RadContextMenu.ContextMenu>
    <telerik:RadContextMenu
      EventName="MouseEnter"
...
```

For additional specificity, you can add the **ModifierKey** property in combination with the EventName. If we set the ModifierKey property to "Control"...



```
<telerik:RadContextMenu.ContextMenu>
    <telerik:RadContextMenu
        x:Name="mnuProducts"
        EventName="MouseEnter"
        ModifierKey=""
        ItemClick="Ra              ItemClick"
        >
        <telerik:RadM
            Header="P              List">
        </telerik:Rad
        <telerik:RadM
            Header="Add Product to Palette">
```

...now the menu will only show when the control key is held down and the mouse passes over the button.

Assigning the Context Menu in Code

Use the static RadContextMenu SetContextMenu() method to assign context menus in code. The example below builds menu items and adds them to the RadContextMenu Items collection just as you would for a RadMenu. The only other difference to this code is that the EventName property is assigned. Also notice in the example how the AddHandler() method is used to add a routed event handler to the menu's ItemClick event.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim contextMenu As New RadContextMenu()
  contextMenu.EventName = "Click"

  Dim topLevelItem As New RadMenuItem() With {.Header = "Reports"}

  topLevelItem.Items.Add(New RadMenuItem() With {.Header = "Product Listing"})

  topLevelItem.Items.Add(New RadMenuItem() With {.Header = "Products by Category"})

  contextMenu.Items.Add(topLevelItem)

  contextMenu.AddHandler(RadMenuItem.ClickEvent, _
New RoutedEventHandler(AddressOf OnMenuItemClick), False)

  RadContextMenu.SetContextMenu(MyButton, contextMenu)
End Sub

Private Sub OnMenuItemClick(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim args As RadRoutedEventArgs = TryCast(e, RadRoutedEventArgs)
  Dim item As RadMenuItem = TryCast(args.OriginalSource, RadMenuItem)
  RadWindow.Alert(item.Header.ToString())
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadContextMenu contextMenu = new RadContextMenu();
   contextMenu.EventName = "Click";

   RadMenuItem topLevelItem = new RadMenuItem()
   {
      Header = "Reports"
   };

   topLevelItem.Items.Add(new RadMenuItem()
   {
      Header = "Product Listing"
   });

   topLevelItem.Items.Add(new RadMenuItem()
   {
      Header = "Products by Category"
   });

   contextMenu.Items.Add(topLevelItem);

   contextMenu.AddHandler(
      RadMenuItem.ClickEvent, new RoutedEventHandler(OnMenuItemClick), false);

   RadContextMenu.SetContextMenu(MyButton, contextMenu);
}

private void OnMenuItemClick(object sender, RoutedEventArgs e)
{
   RadRoutedEventArgs args = e as RadRoutedEventArgs;
   RadMenuItem item = args.OriginalSource as RadMenuItem;
   RadWindow.Alert(item.Header.ToString());
}
```

Placement and Sizing

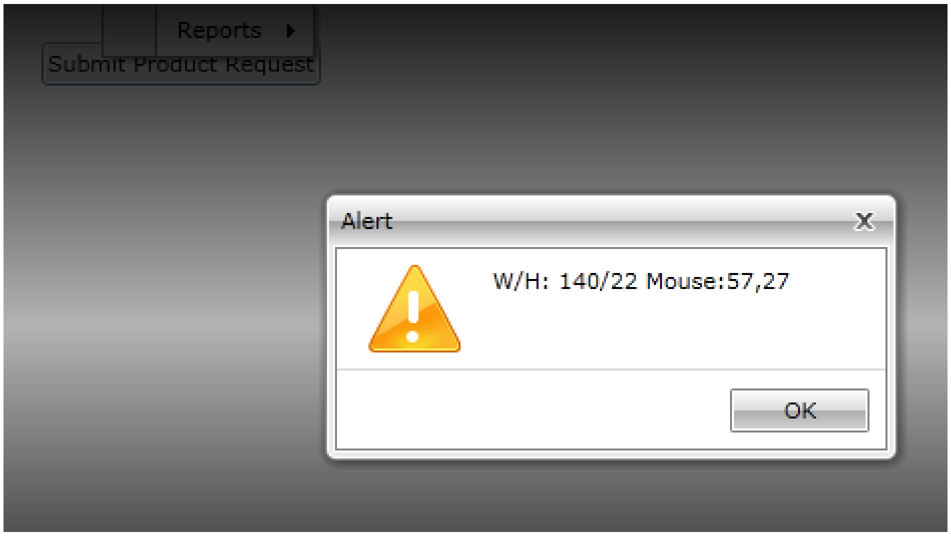
RadContextMenu has several properties that govern the location of the context menu and its relationship to the target control. The **Placement** property can be **Absolute** (default), **Bottom**, **Center**, **Right**, **Left** or **Top** in relation to the target control. RadContextMenu also handles "Boundary Detection", so the menu will only honor the Placement property if there is room. Instead of placing the menu relative to the target control, **PlacementRectangle** can be defined as the area relative to which the context menu is positioned. **HorizontalOffset** and **VerticalOffset** are the horizontal and vertical distances between the target control and the popup. If you need to very specifically control where and how big the menu is, handle the **Opened** event and look at the menu **MousePostion** and **UIElement.RenderSize** properties.

VB

```
Private Sub contextMenu_Opened(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim menu As RadContextMenu = TryCast(sender, RadContextMenu)
  Dim menuSize As Size = menu.UIElement.RenderSize
  RadWindow.Alert(String.Format("W/H: {0}/{1} Mouse:{2},{3}", _
menuSize.Width, menuSize.Height, menu.MousePosition.X, menu.MousePosition.Y))
End Sub
```

C#

```
void contextMenu_Opened(object sender, RoutedEventArgs e)
{
   RadContextMenu menu = sender as RadContextMenu;
   Size menuSize = menu.UIElement.RenderSize;
   RadWindow.Alert(String.Format("W/H: {0}/{1} Mouse:{2},{3}",
      menuSize.Width, menuSize.Height,
      menu.MousePosition.X, menu.MousePosition.Y));
}
```



W/H: 140/22 Mouse:57,27

10.5 Binding

In this walk through, you will populate a menu with hierarchical data.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Navigation**

Create the View Model Object

1) In the Solution Explorer, right-click the project and select **Add > Class...** Rename the class file "Categories.cs". Copy and paste the code below.

The Categories class will initialize a list of Categories and Products to assign to the ItemsSource for the menu. This will be a read-only list where the data is not expected to change. You would use ObservableCollection<> and implement if the data was dynamic. See the Data Binding chapter for more information.

```
Imports System.Collections.Generic
Imports System.Windows.Controls

Namespace _05_Databinding
  Public Class Product
    Private privateProductName As String
    Public Property ProductName() As String
      Get
        Return privateProductName
      End Get
      Set(ByVal value As String)
        privateProductName = value
      End Set
    End Property
    Private privateDescription As String
    Public Property Description() As String
      Get
        Return privateDescription
      End Get
      Set(ByVal value As String)
        privateDescription = value
      End Set
    End Property
  End Class

  Public Class Category
    Private privateCategoryName As String
    Public Property CategoryName() As String
      Get
        Return privateCategoryName
      End Get
      Set(ByVal value As String)
        privateCategoryName = value
      End Set
    End Property
    Private privateProducts As List(Of Product)
    Public Property Products() As List(Of Product)
      Get
        Return privateProducts
      End Get
      Set(ByVal value As List(Of Product))
        privateProducts = value
      End Set
    End Property
  End Class

  Public Class Categories
    Inherits List(Of Category)
    Public Sub New()
      Me.Add(New Category() With {.CategoryName = "Coffee"})
      Me.Add(New Category() With {.CategoryName = "Chocolate"})
      Me(0).Products = New List(Of Product) (New Product() { _
New Product() With {.ProductName = "Guatemala", _
.Description = "Grown above 4,000 feet"}, _
New Product() With {.ProductName = "Kenya", _
.Description = "Excellent body"}})
      Me(1).Products = New List(Of Product) (New Product() { _
New Product() With {.ProductName = "Bittersweet Chocolate", _
.Description = "Sweetened dark chocolate without milk"}, _
New Product() With {.ProductName = "White Chocolate", _
.Description = "Sugar, cocoa butter, milk solids"}})
    End Sub
  End Class
End Namespace
```

```
using System.Collections.Generic;
using System.Windows.Controls;

namespace _05_Databinding
{
   public class Product
   {
      public string ProductName { get; set; }
      public string Description { get; set; }
   }

   public class Category
   {
      public string CategoryName { get; set; }
      public List<Product> Products { get; set; }
   }

   public class Categories : List<Category>
   {
      public Categories()
      {
         this.Add(new Category() { CategoryName = "Coffee" });
         this.Add(new Category() { CategoryName = "Chocolate" });
         this[0].Products = new List<Product>() {
             new Product()
             {
               ProductName = "Guatemala",
               Description = "Grown above 4,000 feet"
            },
            new Product()
            {
               ProductName = "Kenya",
               Description = "Excellent body"
            }
            };
         this[1].Products = new List<Product>() {
            new Product()
            {
               ProductName = "Bittersweet Chocolate",
               Description = "Sweetened dark chocolate without milk"
            },
            new Product()
            {
               ProductName = "White Chocolate",
               Description = "Sugar, cocoa butter, milk solids"
            }
            };
      }
   }
}
```

XAML Editing

1) Open MainPage.xaml for editing.

2) Verify that the XML namespaces for **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Navigation** exist in the UserControl element. Add them if they do not exist. Also, add a reference to your project and name it "local".

```
<UserControl
        xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
        xmlns:local="clr-namespace:_05_Databinding"
        . . . >
```

3) Add a Resources sub-element to the UserControl element. Copy the XAML below into the Resources element.

This step creates a reference to the "Categories" object that contains the data we want to display in the menu. The two data templates defined here, "CategoryTemplate" and "ProductTemplate" supply the root node items in the menu and the children under those root nodes, respectively. Also notice that we're binding a product description to the ToolTip of the child menu items.

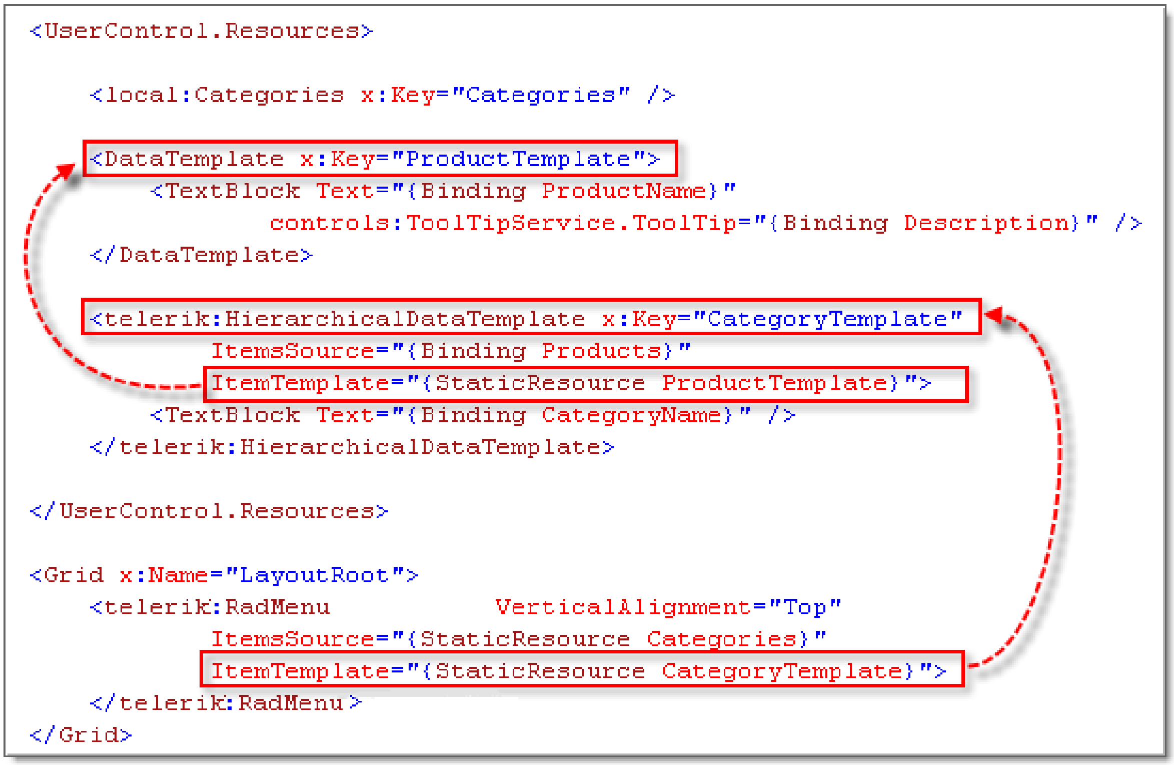
```
<UserControl.Resources>

  <local:Categories x:Key="Categories" />

  <DataTemplate x:Key="ProductTemplate">
    <TextBlock Text="{Binding ProductName}"
      controls:ToolTipService.ToolTip="{Binding Description}" />
  </DataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="CategoryTemplate"
      ItemsSource="{Binding Products}"
      ItemTemplate="{StaticResource ProductTemplate}">
    <TextBlock Text="{Binding CategoryName}" />
  </telerik:HierarchicalDataTemplate>

</UserControl.Resources>
```

4) Drag a **RadMenu** from the Toolbox to the main "LayoutRoot" Grid element. Set the **VerticalAlignment** attribute to "Top", the **ItemsSource** attribute to "{StaticResource Categories}" and the **ItemTemplate** to "{StaticResource CategoryTemplate}".

```
<Grid x:Name="LayoutRoot">
   <telerik:RadMenu VerticalAlignment="Top"
         ItemsSource="{StaticResource Categories}"
         ItemTemplate="{StaticResource CategoryTemplate}">
   </telerik:RadMenu>
</Grid>
```

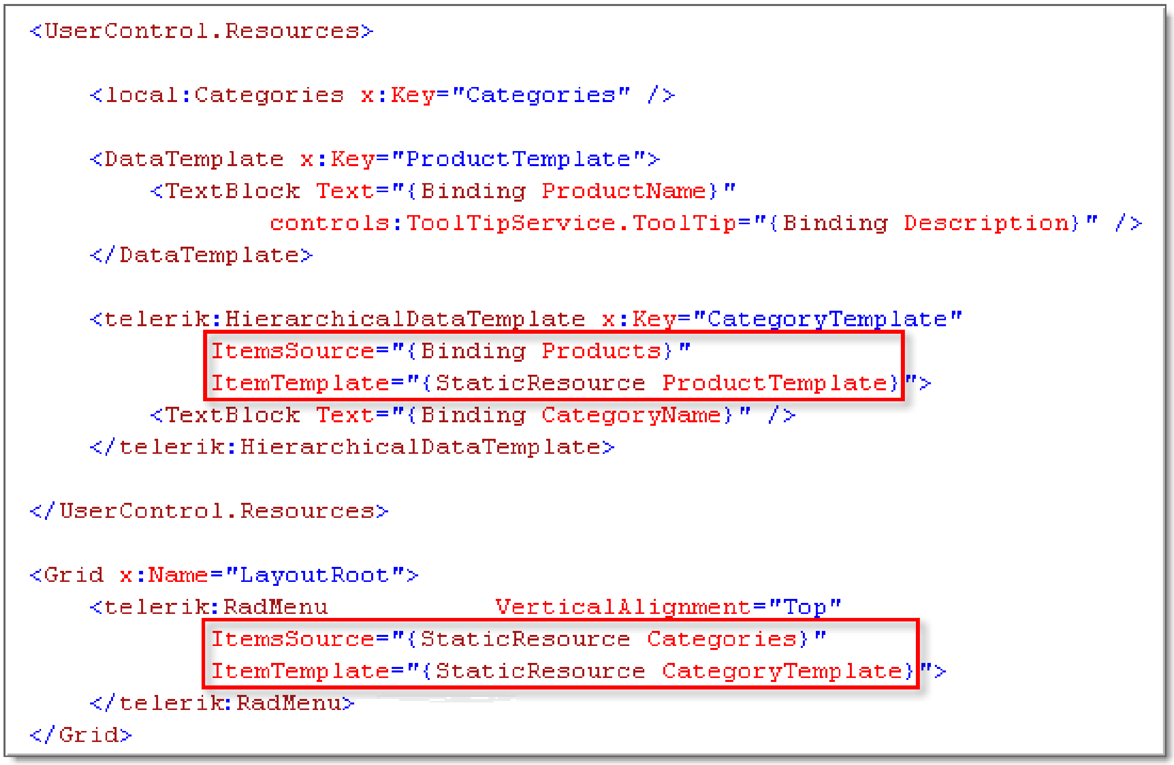
Notes

Look at the relationship of the templates in the XAML below. The RadMenu ItemTemplate points to a HierarchicalDataTemplate "CategoryTemplate". "CategoryTemplate" also has an ItemTemplate that points to a DataTemplate called "ProductTemplate". There can be additional levels of templates. The rule is to use a HierarchicalDataTemplate when there are child items, otherwise use a standard DataTemplate.



```
<UserControl.Resources>

    <local:Categories x:Key="Categories" />

    <DataTemplate x:Key="ProductTemplate">
        <TextBlock Text="{Binding ProductName}"
                controls:ToolTipService.ToolTip="{Binding Description}" />
    </DataTemplate>

    <telerik:HierarchicalDataTemplate x:Key="CategoryTemplate"
            ItemsSource="{Binding Products}"
            ItemTemplate="{StaticResource ProductTemplate}">
        <TextBlock Text="{Binding CategoryName}" />
    </telerik:HierarchicalDataTemplate>

</UserControl.Resources>

<Grid x:Name="LayoutRoot">
    <telerik:RadMenu           VerticalAlignment="Top"
            ItemsSource="{StaticResource Categories}"
            ItemTemplate="{StaticResource CategoryTemplate}">
    </telerik:RadMenu>
</Grid>
```

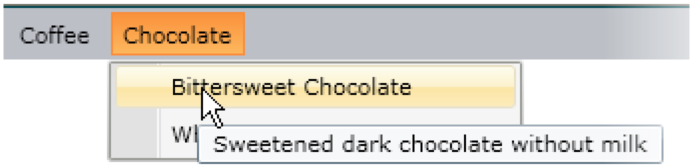
Let's take another look at that same XAML to see how the data is being hooked up. The ItemsSource attribute works in concert with the ItemTemplate. In the RadMenu XAML you can see that the "CategoryTemplate" is supplied by data from the "Categories" object. In the "CategoryTemplate", the "ProductTemplate" is supplied by data from the "Products" object.



```
<UserControl.Resources>

    <local:Categories x:Key="Categories" />

    <DataTemplate x:Key="ProductTemplate">
        <TextBlock Text="{Binding ProductName}"
                controls:ToolTipService.ToolTip="{Binding Description}" />
    </DataTemplate>

    <telerik:HierarchicalDataTemplate x:Key="CategoryTemplate"
            ItemsSource="{Binding Products}"
            ItemTemplate="{StaticResource ProductTemplate}">
        <TextBlock Text="{Binding CategoryName}" />
    </telerik:HierarchicalDataTemplate>

</UserControl.Resources>

<Grid x:Name="LayoutRoot">
    <telerik:RadMenu           VerticalAlignment="Top"
            ItemsSource="{StaticResource Categories}"
            ItemTemplate="{StaticResource CategoryTemplate}">
    </telerik:RadMenu>
</Grid>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Ideas for Extending This Example

- Define additional levels of HierarchicalDataTemplate.
- Make the data object self-referential and only define a single level of HierarchicalDataTemplate. For example, you can define a MenuItem object that has its own Items collection.

10.6 Customization

Walk Through

In this example we will customize the RadMenu control background and border. We will also add an animated effect that takes place when the menu becomes disabled.

"Scoville" Styles

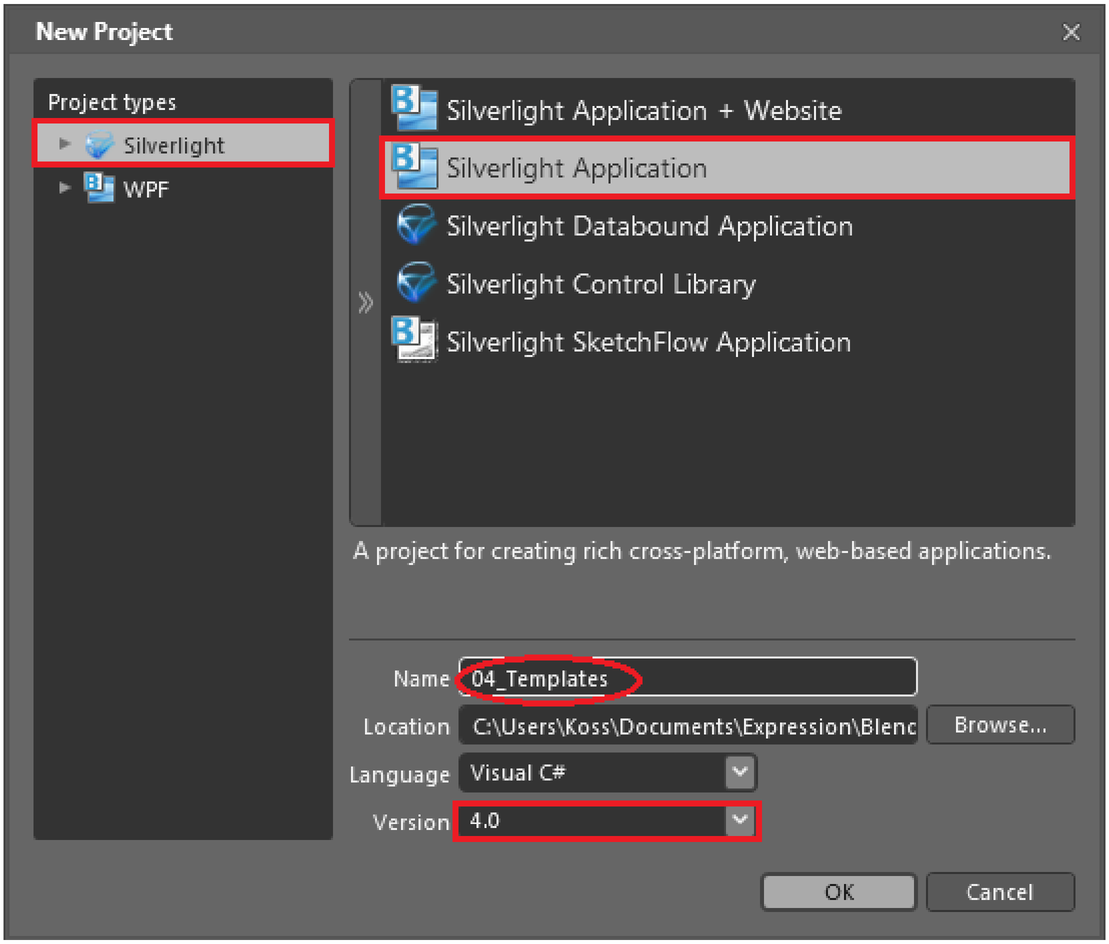
We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

1) Run Expression Blend.

2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*

3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight Application" from the right-most list. Enter a unique name for the project and click **OK**.



New Project
Project types
Silverlight
WPF
Silverlight Application + Website
Silverlight Application
Silverlight Databound Application
Silverlight Control Library
Silverlight SketchFlow Application
Browse...
OK
Cancel

A project for creating rich cross-platform, web-based applications.

Name 04_Templates

Location C:\Users\Koss\Documents\Expression\Blenc

Language Visual C#

Version 4.0

1) In the Projects pane, locate MainPage.xaml and double-click to open the project in Expression Blend.

2) In the Projects pane, right-click the **References** node and select **Add Reference...** from the context menu.

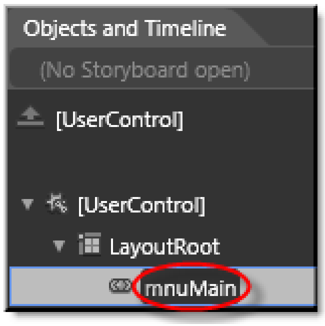
3) Add a reference to the **Telerik.Windows.Controls.dll** assembly.

4) Add a reference to the **Telerik.Windows.Controls.Navigation.dll** assembly.

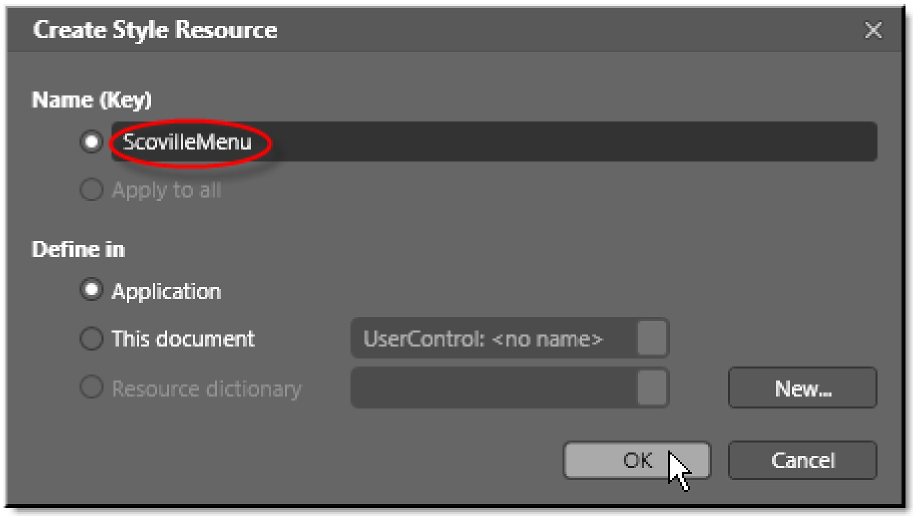
5) From the **Project** menu select **Build Project**.

6) Open the Assets pane. On the left side of the Assets pane is a tree view. Locate and open the "Controls" node, then click the "All" node. A list of all available controls will show to the right of the tree view. Locate the RadMenu control and drag it onto the MainPage.xaml Artboard.

7) In the Objects and Timeline pane, double-click "[RadMenu]" in the tree view. Enter a new name "mnuMain".



8) Right-click the menu and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleMenu" and click the "Define in Application" radio button. Click **OK** to create the stye resource and close the dialog.


Create Style Resource

Name (Key)

● ScovilleMenu

○ Apply to all

Define in

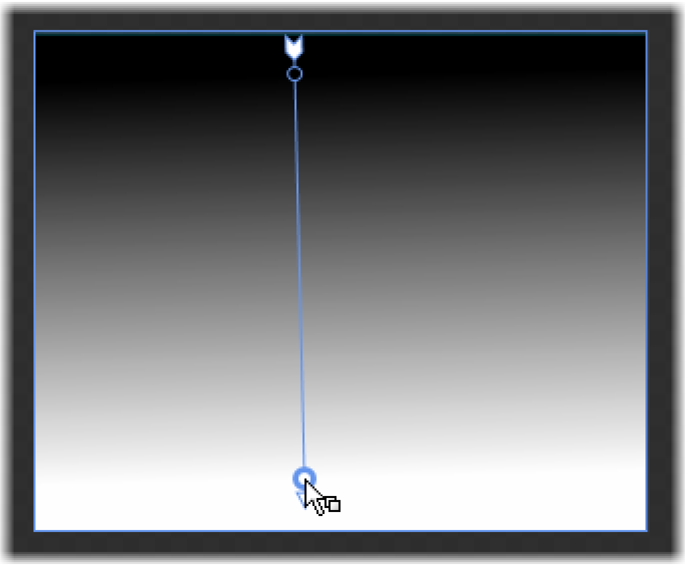
● Application

○ This document UserControl: <no name>

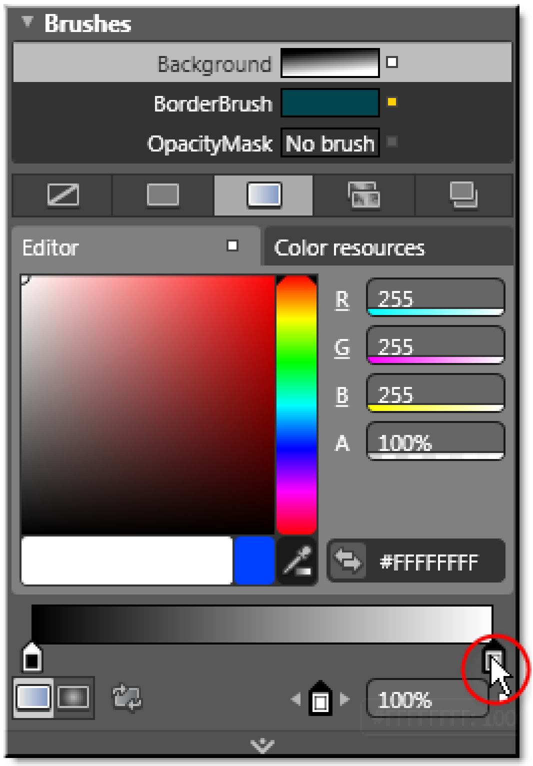
○ Resource dictionary

9) In the **Objects and Timeline pane** select "[border]" from the tree view.

10)Select the Gradient tool from the Tool Bar. Drag the tool from the top of the menu to the bottom. This step will reset the fill style for the RadMenu border object and provide it with a gradient fill.



11)Find the right-most gradient stop indicator and click it with the mouse.



Brushes

Background

BorderBrush

OpacityMask No brush

Editor

Color resources

R 255

G 255

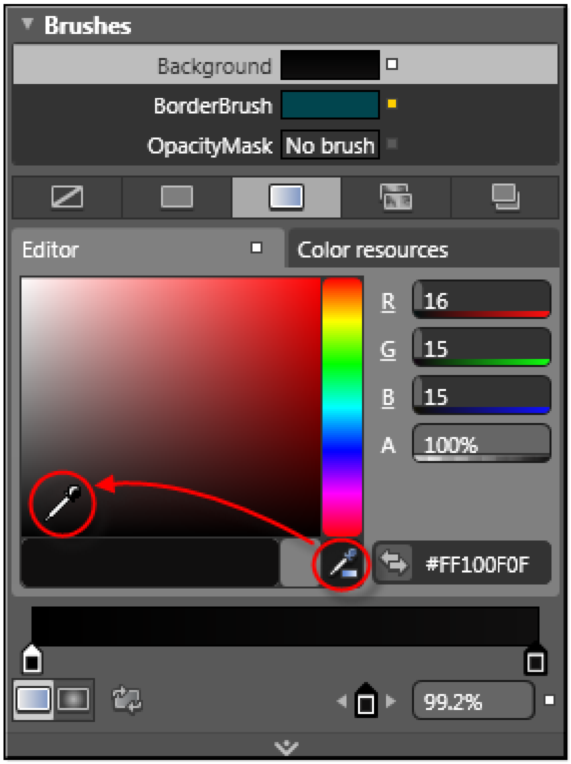
B 255

A 100%

#FFFFFFFF

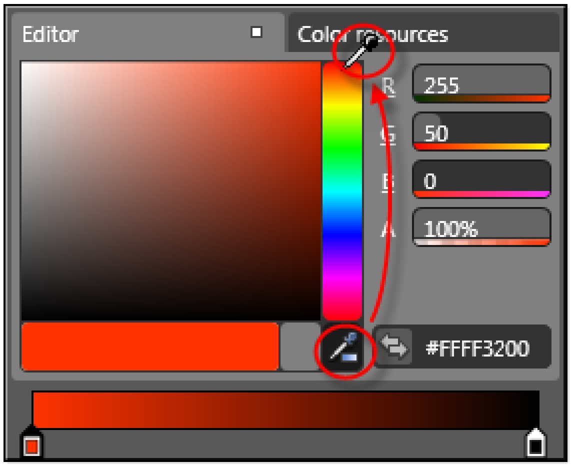
100%

12)Grab the and select a shade of black for the gradient stop.

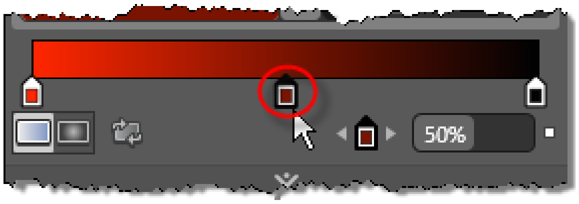


13)Find the left-most gradient stop indicator and click it with the mouse.

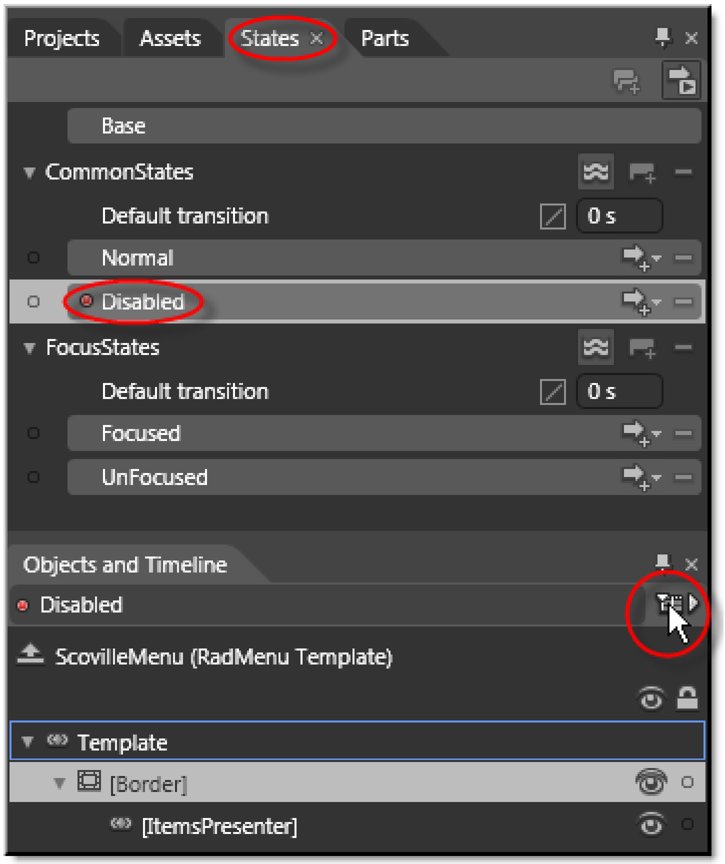
14)Grab the eye dropper tool and select a shade of red for the gradient stop.



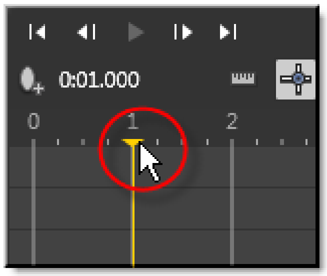
15)Click the Gradient Bar somewhere in the middle to create a new gradient stop.



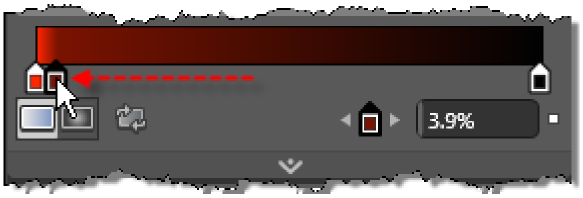
16)In the States pane, click the "Disabled" state. This will cause a "Disabled" item to appear at the top of the Objects and Timeline pane. In the Objects and Timeline pane, click the "Show Timeline" button.



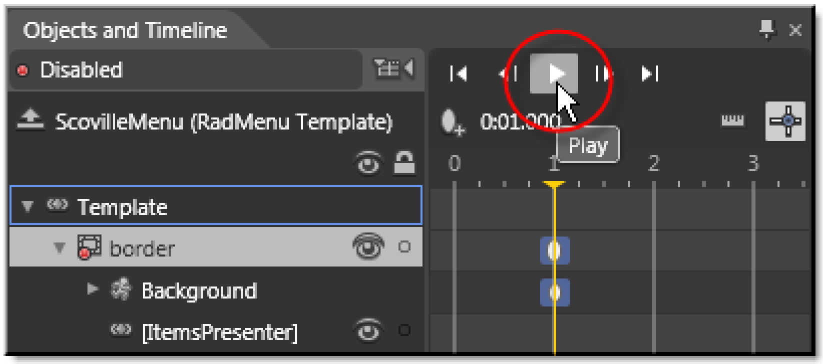
17)In the Timeline, drag the Timeline marker to the 1 second mark.



18)Back on the Properties pane, slide the middle gradient stop to the far left. *This step creates an animation of the middle gradient stop that plays automatically when the menu becomes disabled.*



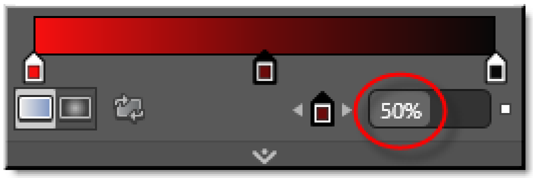
19)In the Objects and Timeline pane, click the **Play** button to see the animation in action. *During the one second animation, the darker shades should roll up to cover most of the menu.*



20)In the States pane, click the "Normal" state. This will cause a "Normal" item to appear at the top of the Objects and Timeline pane.

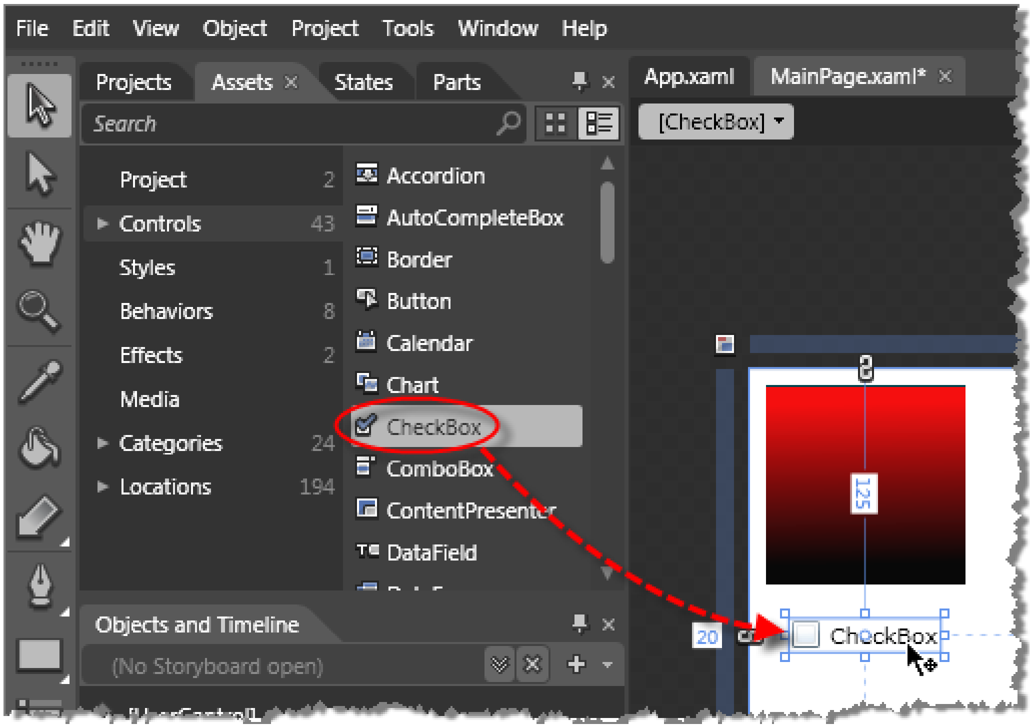
21)In the Timeline, drag the Timeline marker to the 1 second mark.

22)Back on the Properties pane, click the middle gradient stop. Type in "50" to the gradient step offset text box. *This step will create an animation that ends at one second leaving the "normal" property settings, i.e. with the middle gradient stop midway between the red and black.*

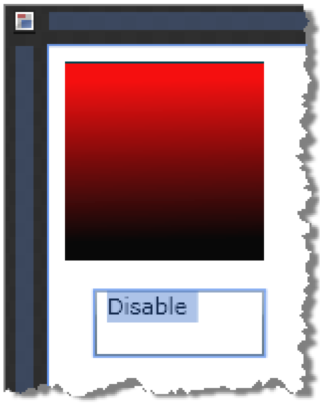


23)In the Projects pane, double-click MainPage.xaml to open it for editing in the Artboard.

24)Drag a CheckBox control from the Assets pane to a point just below the RadMenu.

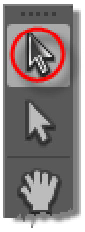


25)Double-click the CheckBox and type in new text "Disable".

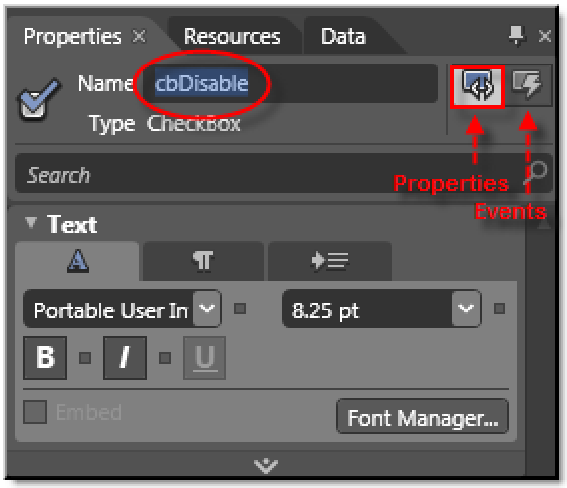


Notes

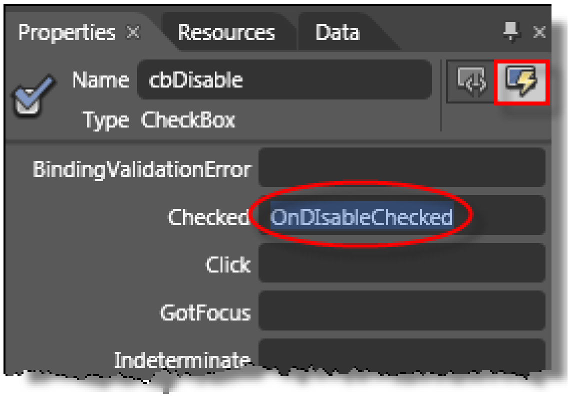
The "Select" tool must be active to allow you to double-click and rename the CheckBox control.



26)In the Properties pane make sure the Properties button is toggled down, then enter a new name "cbDisable" for the CheckBox. **Note:** To see the properties, be sure you have the Properties button toggled down, not the Events button.



27)In the Properties pane, toggle the "Events" button down. In the "Checked" event, enter an event handler name "OnDisableChecked". *The event handler will be created automatically by Expression Blend and the code-behind for MainPage.xaml will display Also note that the event handler name is arbitrary and can be whatever you choose to assign.*



28) In the code-behind for MainPage.xaml, code the Checked event handler to set the menu's IsEnabled property to false as shown below. *Note that the code editing happens right in Expression Blend and includes IntelliSense support. You can also switch to Visual Studio and edit the same project if you prefer.*

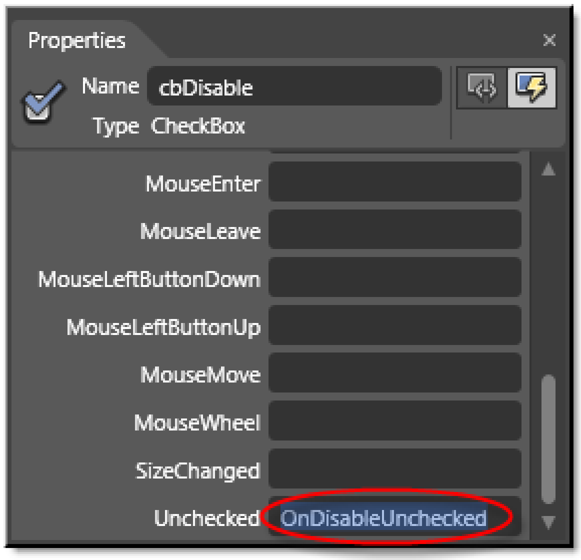
VB

```
Private Sub OnDisableChecked(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs)
  mnuMain.IsEnabled = False
End Sub
```

C#

```
private void OnDisableChecked(object sender, System.Windows.RoutedEventArgs e)
{
  mnuMain.IsEnabled = false;
}
```

29) Navigate back to editing MainPage.xaml in the Artboard. In the Properties pane select the Events button and add an event handler for the Unchecked event.



Name cbDisable
Type CheckBox
Unchecked OnDisableUnchecked

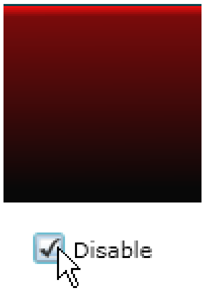
30)In the code-behind for MainPage.xaml, code the Unchecked event handler to set the menu's IsEnabled property to true as shown below.

```
Private Sub OnDisableChecked(ByVal sender As Object, ByVal e As System.Windows.RoutedEventArgs)
  mnuMain.IsEnabled = True
End Sub
```

```
private void OnDisableChecked(object sender, System.Windows.RoutedEventArgs e)
{
  mnuMain.IsEnabled = true;
}
```

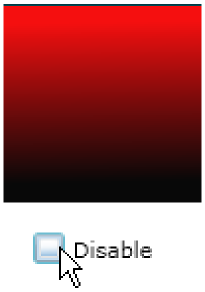
Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



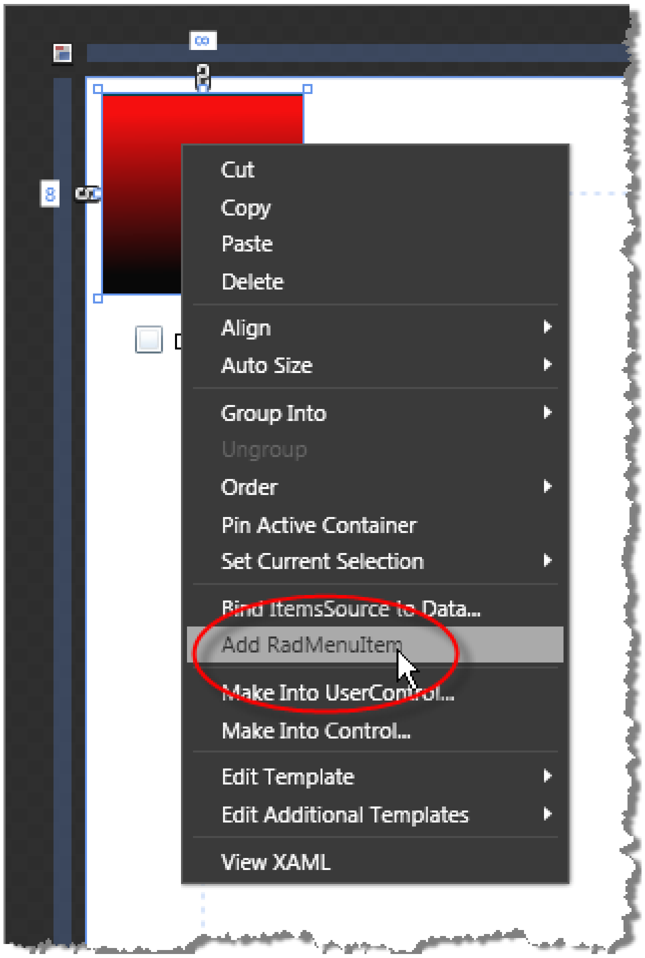
Test Application Features

- Toggle the disable check box and observe the effect on the menu. When you check "Disable", the animation of the center gradient stop should darken the menu. Unchecking "Disable" should animate the center gradient stop back to its lighter "Normal" state.

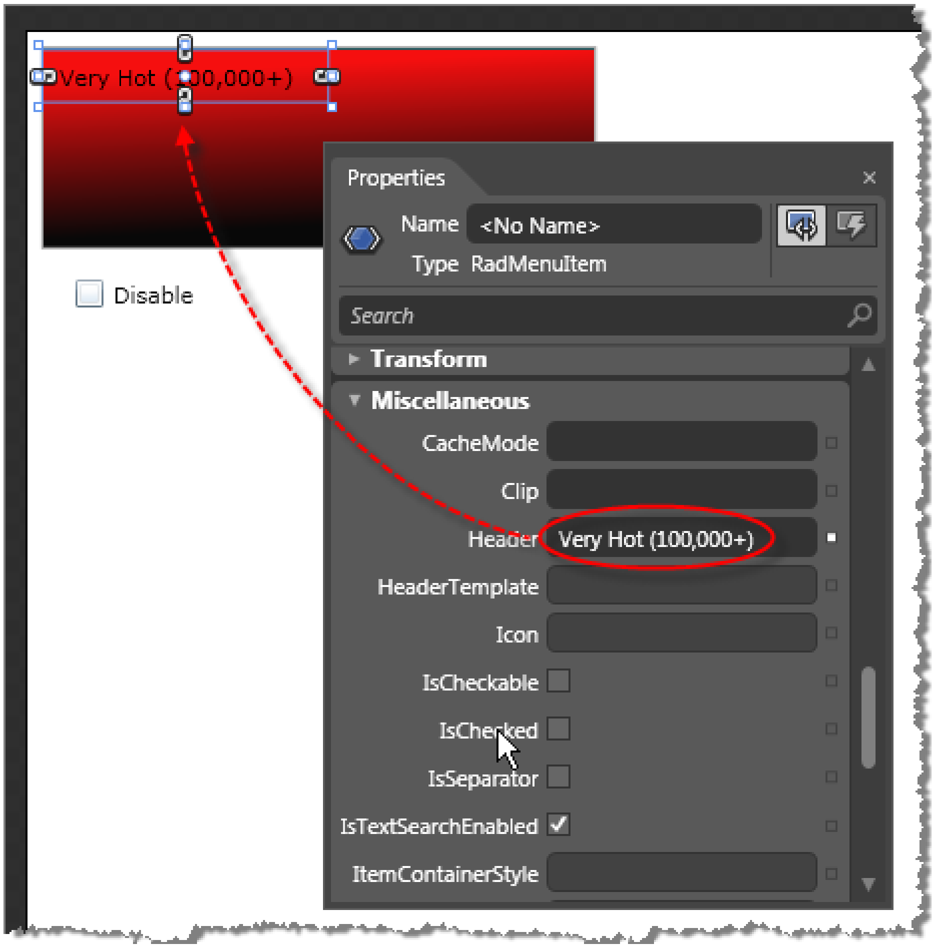


Ideas for Extending This Example

- Add menu items to the menu using Expression Blend. This is done by navigating to MainPage.xaml and editing the menu in the Artboard. Right-click the RadMenu control and select Add RadMenuItem from the context menu.



You can click on the RadMenuItems in the menu and set properties. Find the **Header** property in the "Miscellaneous" properties group.



Very Hot (100,000+)

□ Disable

Properties

Name <No Name>

Type RadMenuItem

Search

▸ Transform

▾ Miscellaneous

CacheMode

Clip

Header Very Hot (100,000+)

HeaderTemplate

Icon

IsCheckable □

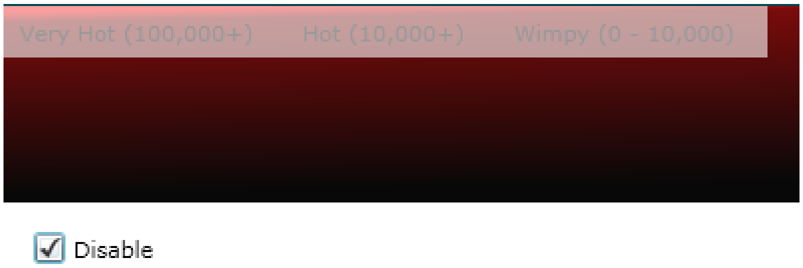
IsChecked □

IsSeparator □

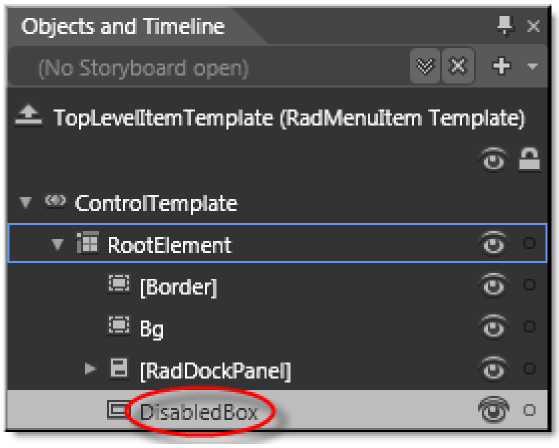
IsTextSearchEnabled ☑

ItemContainerStyle

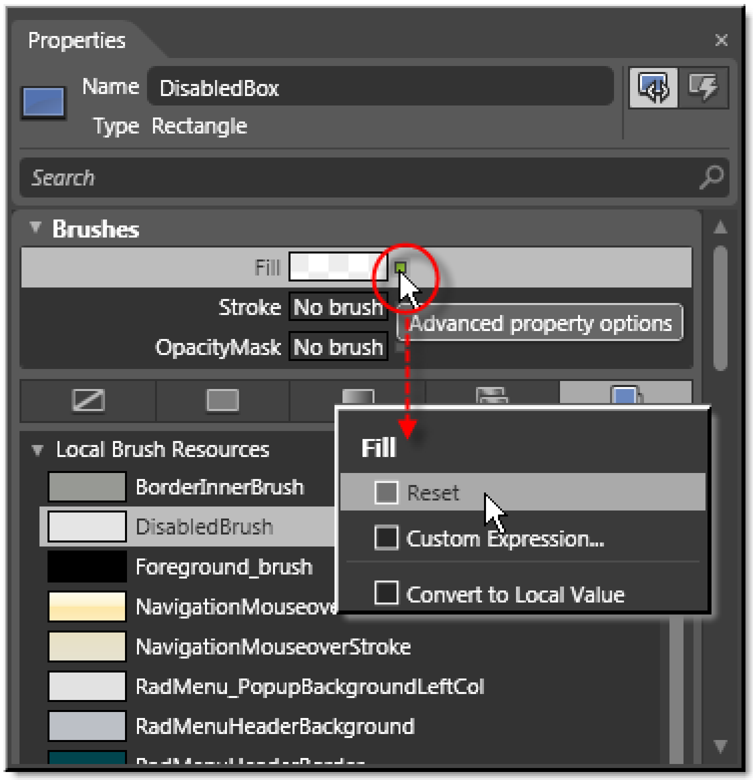
- Customize the template for the **RadMenuItem**, not just the RadMenu. You might do this so that menu items react as the mouse passes over or to make the visual style of the menu items agree with the overall menu. For example, the disabled state of the menu looks fine until you put items into it, but the disabled state of the menu items includes a partially transparent rectangle over the item. This works fine for the default menu, but is horrible for our new "Scoville" menu shown in the screenshot below.



This situation is fixed in the same manner as when creating the new RadMenu "ScovilleMenu" style, i.e. right-clicking a RadMenuItem and selecting **Edit Template > Edit a Copy** from the context menu. We might call this new template "ScovilleMenuItem" and edit it in the Artboard. The menu item includes a "DisabledBox" that you can find in the Objects and Timeline pane.



If you select "DisabledBox" and navigate to the Properties pane, you can locate the Fill brush. At this point the brush is set to "DisabledBrush" which happens to represent that unpleasant transparent white background. If you click the Fill "Advanced property options" button, you can click the "Reset" option from the drop down menu. This will set the brush fill to "Transparent", allowing the menu background to show through without interference.



Name DisabledBox

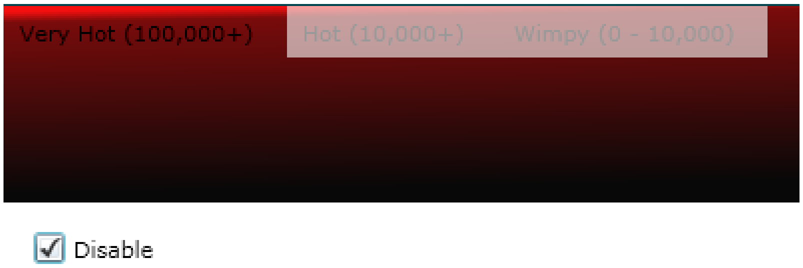
Type Rectangle

Fill

Stroke No brush

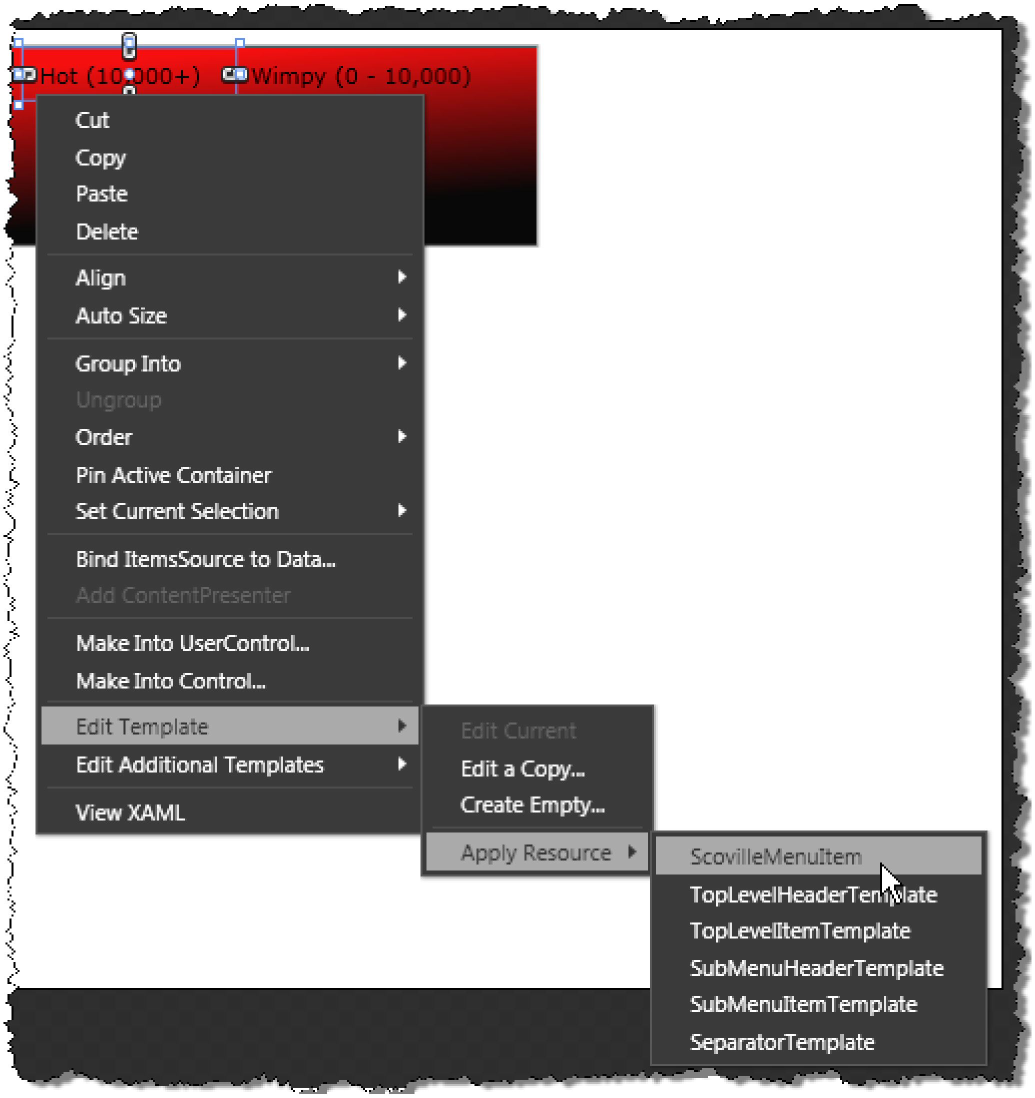
OpacityMask No brush

Now the menu item with the new "ScovilleMenuItem" template applied displays correctly when shown in a menu with the "ScovilleMenu" template applied.

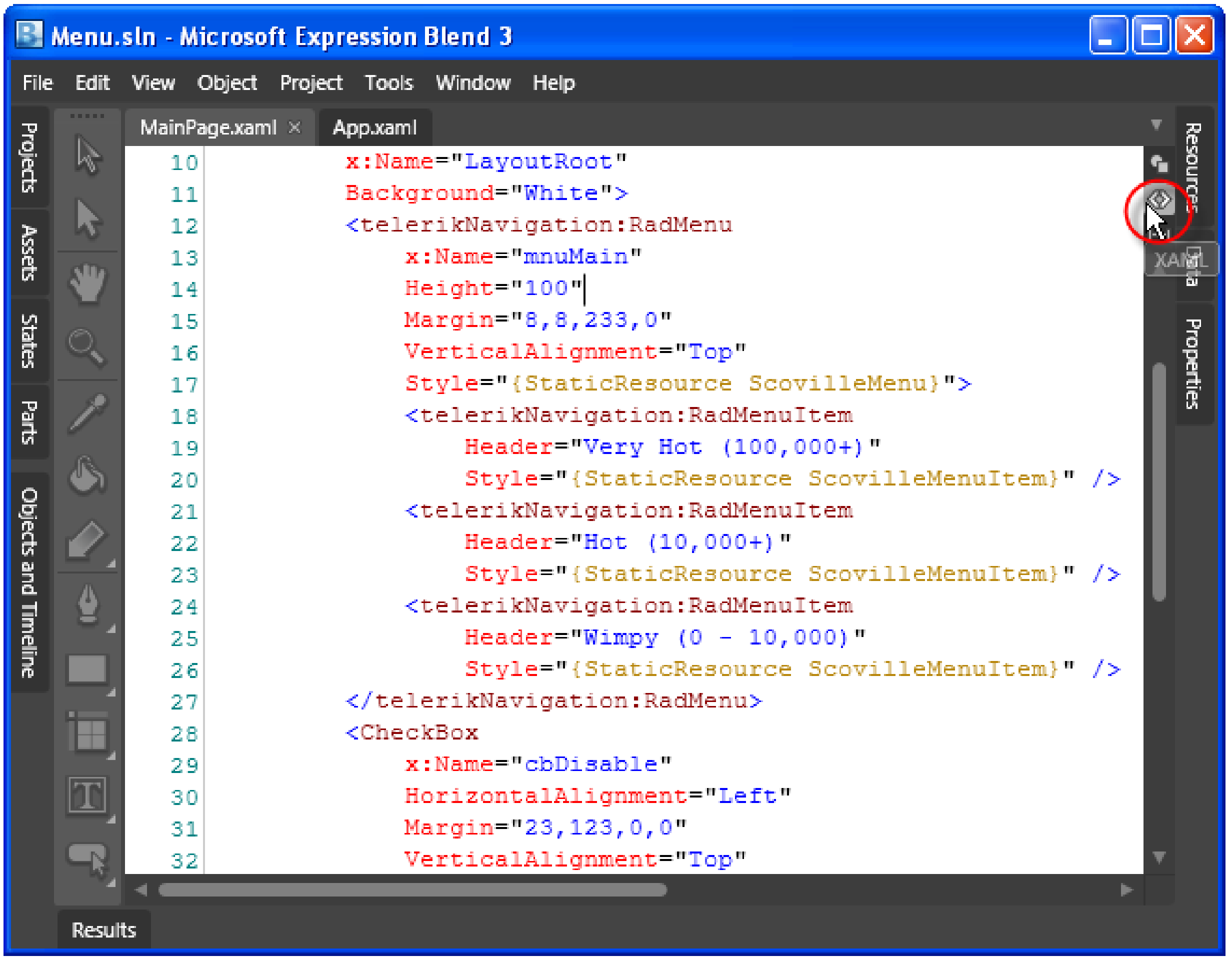


☑ Disable

To fix the other menu items, right-click each item and select **Edit Template > Apply Resource >** "<your menu item style>", i.e. "ScovilleMenuItem".

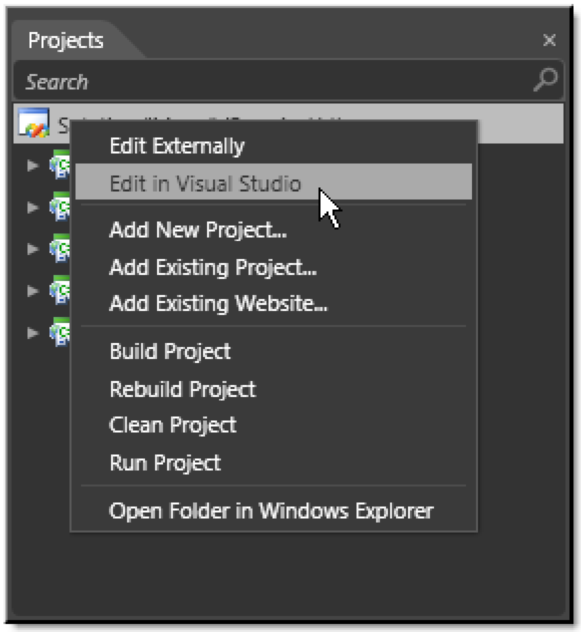


By clicking the "XAML" button we can see that the markup for the menu is still clean, with the complexity of these changes being stored in the resources for "ScovilleMenu" and "ScovilleMenuItem". In this case we have the resources tucked away in the App.xaml file where they can be used by other pages in the application, but they could also be stored in the same "MainPage.xaml" page or in another assembly altogether.



```
x:Name="LayoutRoot"
Background="White">
<telerikNavigation:RadMenu
    x:Name="mnuMain"
    Height="100"
    Margin="8,8,233,0"
    VerticalAlignment="Top"
    Style="{StaticResource ScovilleMenu}">
    <telerikNavigation:RadMenuItem
        Header="Very Hot (100,000+)"
        Style="{StaticResource ScovilleMenuItem}" />
    <telerikNavigation:RadMenuItem
        Header="Hot (10,000+)"
        Style="{StaticResource ScovilleMenuItem}" />
    <telerikNavigation:RadMenuItem
        Header="Wimpy (0 - 10,000)"
        Style="{StaticResource ScovilleMenuItem}" />
</telerikNavigation:RadMenu>
<CheckBox
    x:Name="cbDisable"
    HorizontalAlignment="Left"
    Margin="23,123,0,0"
    VerticalAlignment="Top"
```

- Try running the application from Visual Studio. Start from the Expression Blend Projects pane, right-click the solution or project and select Edit in Visual Studio from the context menu.



10.7 Wrap Up

In this chapter you used RadMenu, RadContextMenu and RadMenuItem to supply drop-down and pop-up lists of user selections. You saw how MenuBase supplies properties common to both RadMenu and RadContextMenu. You built standard menus using RadMenu and used properties to control item opening behavior and check/uncheck support. You learned how to respond to user selections, created menus dynamically in code and added images to menu items. You applied these fundamentals to RadContextMenu and explored context menu specific issues such as initiating the popup, placement, sizing and right-click support. Finally, you used Expression Blend to customize the appearance of a RadMenu and its items.

Part XI

Tabbed Interfaces

11 Tabbed Interfaces

11.1 Objectives

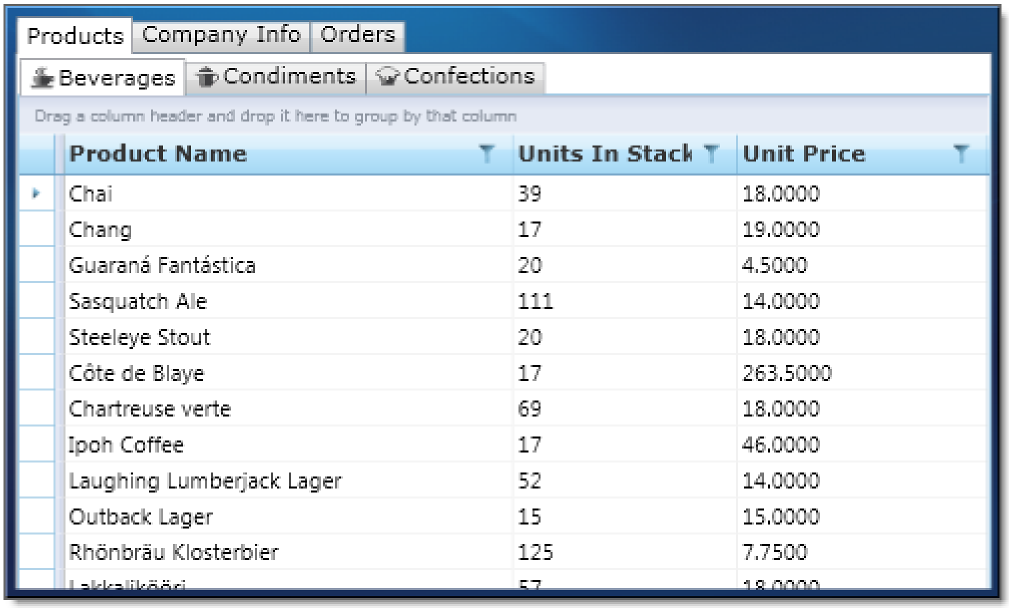
In this chapter you will learn how to create tabbed navigation systems and interfaces using RadTabControl and RadPanelBar controls. You will see the commonality and differences between the two controls. You will build tab controls and panel bars directly in the XAML and programmatically. You will assign simple text to the header and content areas of each control and also learn how to stuff the header and content with filling of any amount and complexity. You will embed controls into the RadTabControl header. You will also handle the significant events of each control. Finally, you will learn how to completely customize a RadTabItem by overriding the ControlTemplate.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Tabbed\Tabbed.sln.

11.2 Overview

Use **RadTabControl** to create tabbed navigation systems and interfaces.



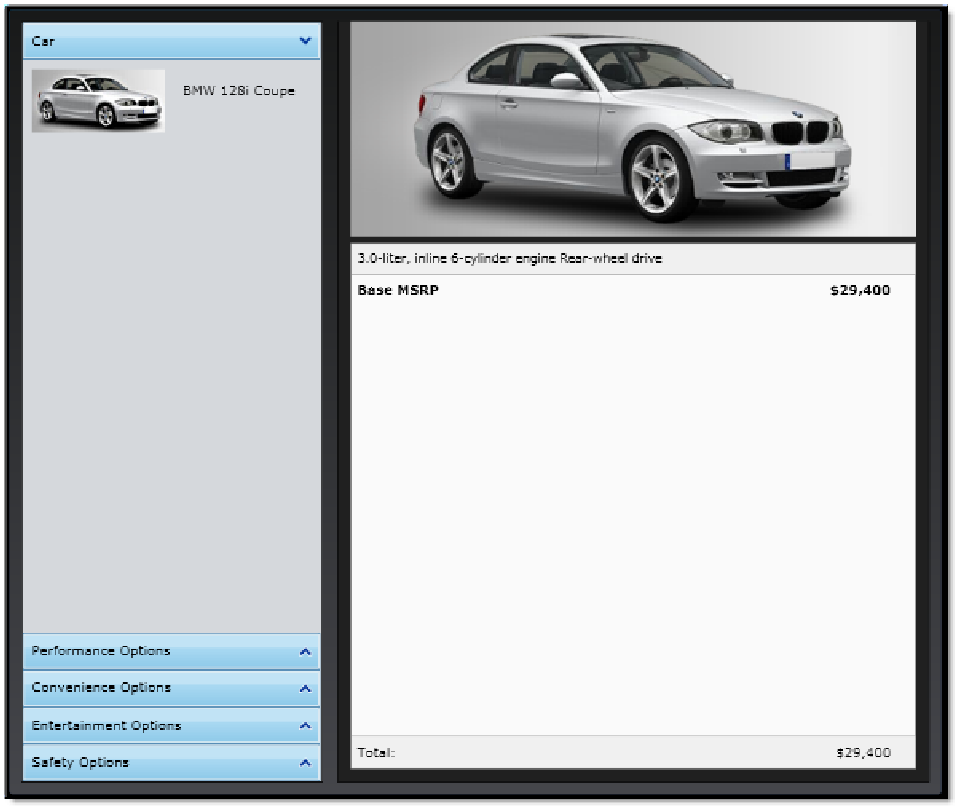
Drag a column header and drop it here to group by that column

Product Name	Units In Stack	Unit Price
Chai	39	18.0000
Chang	17	19.0000
Guaraná Fantástica	20	4.5000
Sasquatch Ale	111	14.0000
Steeleye Stout	20	18.0000
Côte de Blaye	17	263.5000
Chartreuse verte	69	18.0000
Ipoh Coffee	17	46.0000
Laughing Lumberjack Lager	52	14.0000
Outback Lager	15	15.0000
Rhönbräu Klosterbier	125	7.7500
Lakkalikööri	57	18.0000

RadTabControl is fully customizable and has advanced functionality including:

- **Nesting Controls** in the tab content: You can put any control inside the tabs, which allows you to build complex and flexible tabbed user interfaces.
- **Header Content**: You can put any content and templates in the headers.
- **Powerful Data Binding**: TabControl can be bound to various data source types, such as Object, XML and WCF services.
- **Tab Orientation**: Tabs can be positioned horizontally or vertically by setting a single property.
- **Multi-line Tabs**: You can set the end of a tabs row by marking the last tab in the row with IsBreak=”True”. The next tab will start on a new row.
- **Styling and Appearance**: RadTabControl and RadTabItem can be fully customized using Expression Blend. There are also several predefined themes that can be used to style the tab control.
- **Keyboard Support**: Your end-user can navigate through the tabs using the left and right arrow keys. The Home and End keys also navigate to the very first and last tabs.

RadPanelBar is a versatile component allowing you to build intuitive navigation systems such as left/right side menus and Outlook style panels.



BMW 128i Coupe

3.0-liter, inline 6-cylinder engine Rear-wheel drive

Base MSRP	$29,400
Total:	$29,400

RadPanelBar is fully customizable and has advanced functionality including:

- **Hierarchical Data Binding**: You can bind RadPanelBar to hierarchical data structures contained in various data source types, such as Object, XML and WCF services.
- **Styling and Appearance**: RadPanelBar can be fully customized using Expression Blend. Predefined themes let you instantly style the panel bar with a single setting.
- **Keyboard Support**: Your end-user can navigate through the panels using the arrow keys. The Up and Down arrow keys navigate through panel bar header and content items. Right and Left arrow keys expand and collapse panels.
- **Expand Modes**: The expand behavior can mimic popular tabbed interfaces using the single or multiple ExpandMode property settings.

Notes

You may have noticed a control similar to RadPanelBar in the Toolbox called **RadExpander**. RadExpander is a very simple HeaderedContent control. It has only header and content and a single action for toggling content visibility. When you insert several expanders into a single container, the expanders don't communicate with one another. By contrast, RadPanelBar is an ItemsControl descendant and supports hierarchy, data binding and a more complete set of functionality.

11.3 Getting Started

In this walk through you will define a RadTabControl and its items in XAML. One of the tabs will contain a RadPanelBar and its items.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Navigation**

c) **Telerik.Windows.Themes.Summer**

4) Add an "Images" directory to the project. Copy the following images from the "\courseware\images" directory to the new images directory

a) **Blue hills.jpg**

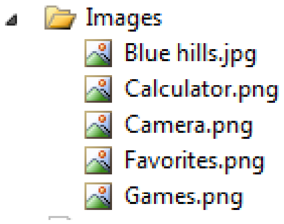
b) **Calculator.png**

c) **Camera.png**

d) **Favorites.png**

e) **Games.png**

The Images directory should look something like the screenshot below.



```
Images
    Blue hills.jpg
    Calculator.png
    Camera.png
    Favorites.png
    Games.png
```

XAML Editing

1) Open MainPage.xaml for editing.

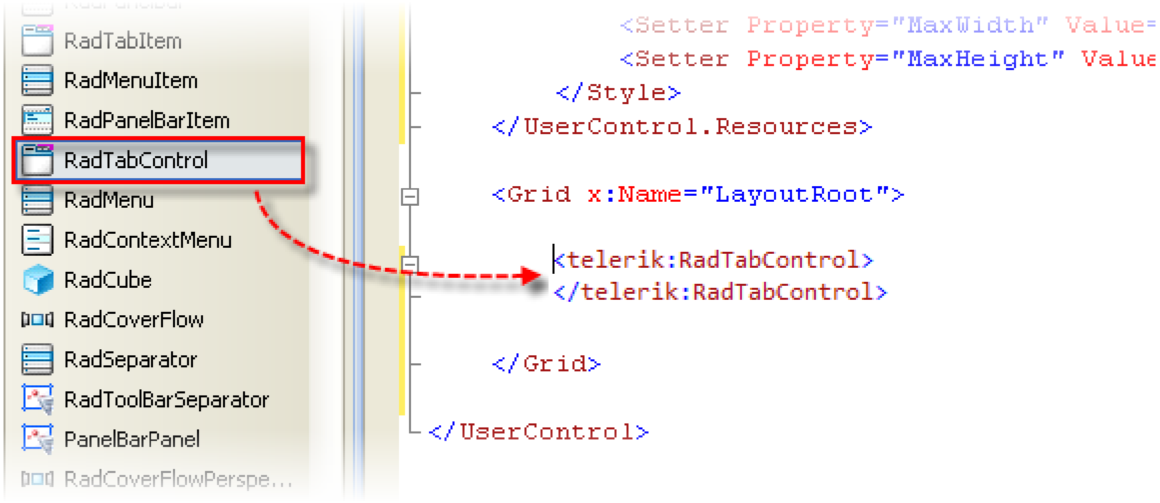
2) Add a XML namespace,"telerik", that references the **Telerik.Windows.Controls** assembly.

```
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
```

3) Inside the UserControl element, add the XAML below. *This XAML defines resources we will use later.*

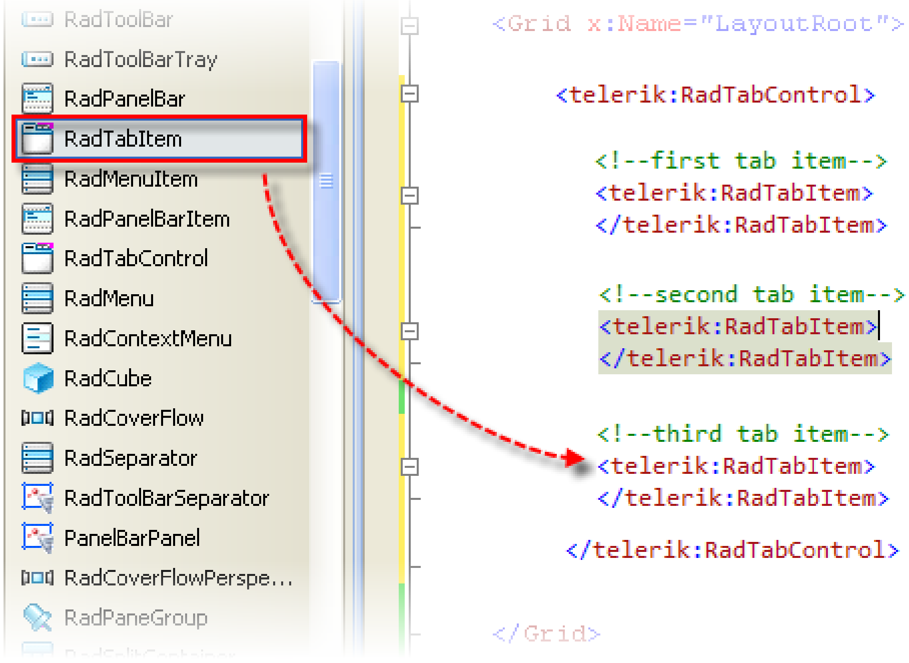
```
<UserControl.Resources>
    <Style x:Key="MyContentStyle" TargetType="FrameworkElement">
        <Setter Property="HorizontalAlignment" Value="Left"></Setter>
        <Setter Property="Margin" Value="10"></Setter>
        <Setter Property="MaxWidth" Value="200"></Setter>
        <Setter Property="MaxHeight" Value="200"></Setter>
    </Style>

    <Style
        x:Key="ImageStyle"
        TargetType="Image"
        BasedOn="{StaticResource MyContentStyle}">
        <Setter Property="Stretch" Value="Uniform"></Setter>
    </Style>

    <Style x:Key="IconStyle" TargetType="FrameworkElement">
        <Setter Property="HorizontalAlignment" Value="Left"></Setter>
        <Setter Property="Margin" Value="10"></Setter>
        <Setter Property="MaxWidth" Value="50"></Setter>
        <Setter Property="MaxHeight" Value="50"></Setter>
    </Style>
</UserControl.Resources>
```

4) Drag a **RadTabControl** from the Toolbox to a point within the main Grid tag.



```
                <Setter Property="MaxWidth" Value=
                <Setter Property="MaxHeight" Value
            </Style>
        </UserControl.Resources>

        <Grid x:Name="LayoutRoot">

            <telerik:RadTabControl>
            </telerik:RadTabControl>

        </Grid>
    </UserControl>
```

5) Add three **RadTabItem** controls between the RadTabControl tags. *Note: the XAML in the screenshot below was massaged slightly to place the tab items on two lines each and to label each tab item with comments for easier reference later in this tutorial.*



```
<Grid x:Name="LayoutRoot">

    <telerik:RadTabControl>

       <!--first tab item-->
       <telerik:RadTabItem>
       </telerik:RadTabItem>

       <!--second tab item-->
       <telerik:RadTabItem>
       </telerik:RadTabItem>

       <!--third tab item-->
       <telerik:RadTabItem>
       </telerik:RadTabItem>

     </telerik:RadTabControl>

</Grid>
```

6) Add a **StyleManager.Theme** property to the RadTabControl and set it equal to the "Summer" theme.

```
<telerik:RadTabControl telerik:StyleManager.Theme="Summer">

    <!--first tab item-->
    <telerik:RadTabItem>
    </telerik:RadTabItem>

    <!--second tab item-->
    <telerik:RadTabItem>
    </telerik:RadTabItem>

    <!--third tab item-->
    <telerik:RadTabItem>
    </telerik:RadTabItem>

</telerik:RadTabControl>
```

7) Add **Header** and **Content** attributes to the first RadTabItem tag and set these attributes equal to "Simple text in header" and "Simple text can go in the content", respectively.

```
<telerik:RadTabControl telerik:StyleManager.Theme="Summer">

    <!--first tab item-->
     <telerik:RadTabItem Header="Simple text in header"
                         Content="Simple text can go in the content">
     </telerik:RadTabItem>

  ...

</telerik:RadTabControl>
```

8) Inside the second tag add a **RadTabItem.Header** sub-element. Below the RadTabItem.Header element, add a **RadTabItem.Content** sub-element.

```
<telerik:RadTabControl telerik:StyleManager.Theme="Summer">

      ...

      <!--second tab item-->
      <telerik:RadTabItem>
      <telerik:RadTabItem.Header>

      </telerik:RadTabItem.Header>
      <telerik:RadTabItem.Content>

      </telerik:RadTabItem.Content>
      </telerik:RadTabItem>

       ...

</telerik:RadTabControl>
```

Tip!

You can populate the Header and Content attribute when you only need a single assignment. For more complex arrangements with multiple items and arbitrary layout within the tab item, use Header and Content sub-elements as shown in this last step. In following steps we will add controls to the Header and Content sub-elements.

9) Inside the second tab item Header element, add a TextBlock. You can use the text as shown below or add your own text

Notes

.The <Run> tags within the TextBlock element allow you to assign style to only parts of the TextBlock contents.

```
<telerik:RadTabControl
   telerik:StyleManager.Theme="Summer">

   . . .

   <!--second tab item-->
   <telerik:RadTabItem>
      <telerik:RadTabItem.Header>
         <TextBlock>
           Header accepts <Run
               FontWeight="Bold">Silverlight</Run>
           or <Run
               FontWeight="Bold">Telerik</Run> controls
         </TextBlock>
      </telerik:RadTabItem.Header>
      <telerik:RadTabItem.Content>

      </telerik:RadTabItem.Content>
   </telerik:RadTabItem>

   . . .

</telerik:RadTabControl>
```

10)Inside the second tab item's Content tags add a **StackPanel** tag. Within the StackPanel add **TextBlock,** a Telerik **RadCalendar** and an **Image** control from the Toolbox. Add text to the TextBlock either using the XAML below or any arbitrary text you care to add. To set the RadCalendar properties all at one time, assign "{StaticResource MyContentStyle}" to the RadCalendar Style attribute. Set the Image Style to "{StaticResource ImageStyle}". Point the Image Source property to the "Blue Hills.jpg" image.

```
<telerik:RadTabControl
  telerik:StyleManager.Theme="Summer">

  . . .

  <!--second tab item-->
  <telerik:RadTabItem>
    . . .
    <telerik:RadTabItem.Content>
      <StackPanel>
        <TextBlock
          Margin="10">
          Content can hold arbitrary <Run
            FontWeight="Bold">Silverlight</Run>
          and <Run
            FontWeight="Bold">Telerik</Run> controls
        </TextBlock>
        <telerik:RadCalendar
          Style="{StaticResource MyContentStyle}">
        </telerik:RadCalendar>
        <Image
          Style="{StaticResource ImageStyle}"
          Source="../Images/Blue Hills.jpg">
        </Image>
      </StackPanel>
    </telerik:RadTabItem.Content>
  </telerik:RadTabItem>

 . . .

</telerik:RadTabControl>
```

Gotcha!

Does the editor flag an error "The type 'telerik:RadCalendar' was not found"? Make sure that you actually drag the RadCalendar from the Toolbox rather than just copying the XAML here. Dragging from the toolbox also adds an XML namespace that references the correct assembly.

Gotcha!

Image not displaying? Here are a few things to check:

- Did you add a folder "Images" to your project?
- Did you spell the name of the image file correctly, e.g. "Blue Hills.jpg" and not "BlueHills.jpg" with no space or "Blue Hills.bmp" with an incorrect extension.
- Does the path leading up to the file name match the path in your project?

11)In the third RadTabItem, set the Header value to "RadPanelBar". Add a RadTabItem.Content element. Drag a **RadPanelBar** from the Toolbox to a point within the RadTabItem.Content beginning and ending tags. Set the RadPanelBar attributes HorizontalAlignment = "Left" and add a StyleManager.Theme attribute set to "Summer".

12)Add three RadPanelBarItem tags to the RadPanelBar from the Toolbox and assign Header attributes "Favorites", "Reports" and "Options". The Result should now look like the XAML below.

```
<telerik:RadTabControl
  telerik:StyleManager.Theme="Summer">

 . . .

 <!--third tab item-->
 <telerik:RadTabItem Header="RadPanelBar">

   <telerik:RadTabItem.Content>

     <!--panel bar-->
     <telerik:RadPanelBar
       HorizontalAlignment="Left"
       telerik:StyleManager.Theme="Summer">

       <!--first panel bar item-->
       <telerik:RadPanelBarItem Header="Favorites">
       </telerik:RadPanelBarItem>

       <!--second panel bar item-->
       <telerik:RadPanelBarItem Header="Reports">
       </telerik:RadPanelBarItem>

       <!--third panel bar item-->
       <telerik:RadPanelBarItem Header="Options">
       </telerik:RadPanelBarItem>

     </telerik:RadPanelBar>

   </telerik:RadTabItem.Content>

 </telerik:RadTabItem>

</telerik:RadTabControl>
```

13)Locate the first RadPanelBarItem and remove the 'Header="Favorites"' attribute. Add a **RadPanelBarItem.Header** element inside the RadPanelBarItem. Inside the RadPanelBarItem.Header, add an **Image** control. Set the Image **Style** equal to "{StaticResource IconStyle}", and point the **Source** to "../Images/Favorites.png".

```
<telerik:RadTabControl
   telerik:StyleManager.Theme="Summer">
  . . .

  <!--third tab item-->
  <telerik:RadTabItem Header="RadPanelBar">

      <telerik:RadTabItem.Content>

        <!--panel bar-->
        <telerik:RadPanelBar
           HorizontalAlignment="Left"
           telerik:StyleManager.Theme="Summer">

           <!--first panel bar item-->
           <telerik:RadPanelBarItem>
              <telerik:RadPanelBarItem.Header>
                 <Image Style="{StaticResource IconStyle}"
                    Source="../Images/Favorites.png"></Image>
              </telerik:RadPanelBarItem.Header>

           . . .

           </telerik:RadPanelBarItem>
        </telerik:RadPanelBar>
      </telerik:RadTabItem.Content>
  </telerik:RadTabItem>
</telerik:RadTabControl>
```

14)Below the RadPanelBarItem.Header, add a **RadPanelBarItem.Items** tag and three Image controls within the Items tag. Again, set the Style for each Image to "{StaticResource IconStyle}" and the image Source properties to "Calculator.png", "Camera.png" and "gamecontroller.png" (all image paths within the \Images folder as shown below). Now the relevant portion of XAML for that first RadPanelBarItem looks like the example below.

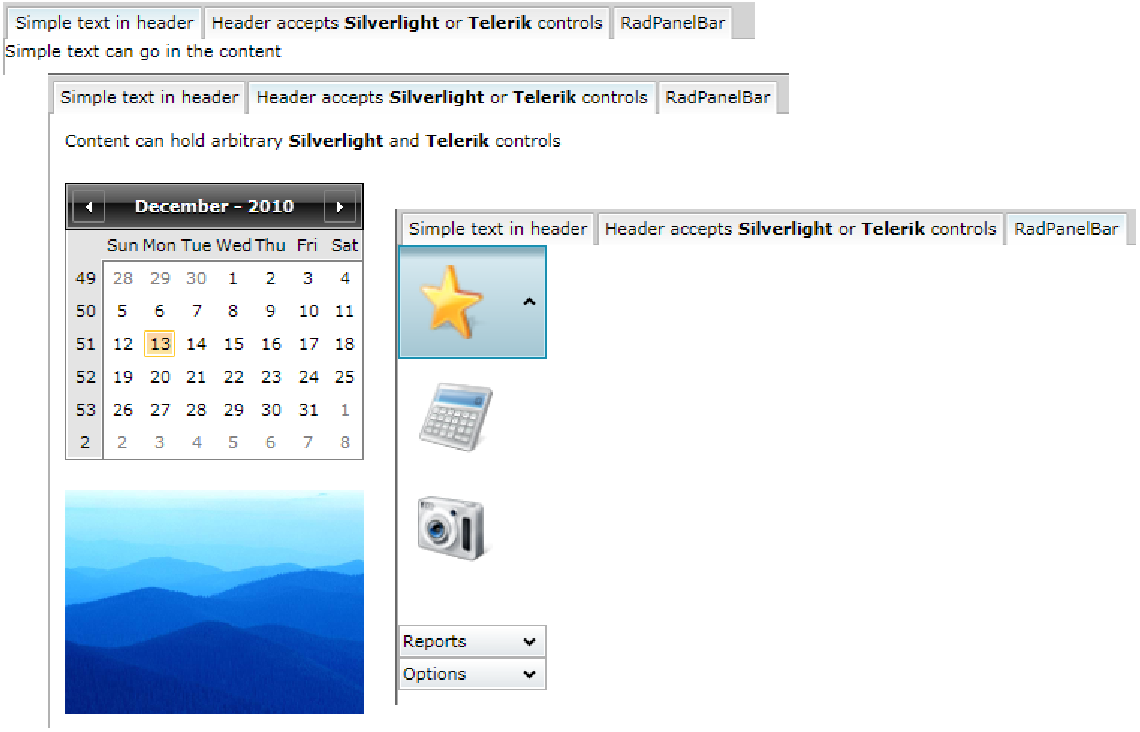
```
<telerik:RadTabControl
  telerik:StyleManager.Theme="Summer">

 . . .

  <!--third tab item-->
  <telerik:RadTabItem Header="RadPanelBar">

    <telerik:RadTabItem.Content>

      <!--panel bar-->
      <telerik:RadPanelBar
        HorizontalAlignment="Left"
        telerik:StyleManager.Theme="Summer">

        <!--first panel bar item-->
        <telerik:RadPanelBarItem>
          <telerik:RadPanelBarItem.Header>
            <Image Style="{StaticResource IconStyle}"
              Source="../Images/Favorites.png"></Image>
          </telerik:RadPanelBarItem.Header>
          <telerik:RadPanelBarItem.Items>
            <Image Style="{StaticResource IconStyle}"
              Source="../Images/Calculator.png"></Image>
            <Image Style="{StaticResource IconStyle}"
              Source="../Images/Camera.png"></Image>
            <Image Style="{StaticResource IconStyle}"
              Source="../Images/gamecontroller.png"></Image>
          </telerik:RadPanelBarItem.Items>
        </telerik:RadPanelBarItem>

      . . .

      </telerik:RadPanelBar>
    </telerik:RadTabItem.Content>
  </telerik:RadTabItem>
</telerik:RadTabControl>
```

15)Add two more RadPanelBarItem tags, each with a set of TextBlocks within the RadPanelBarItem.Items tag. You can use any arbitrary text to fill the TextBlock controls or borrow from the XAML below.

```
<telerik:RadTabControl
  telerik:StyleManager.Theme="Summer">

  . . .

  <!--third tab item-->
  <telerik:RadTabItem Header="RadPanelBar">

    <telerik:RadTabItem.Content>

      <!--panel bar-->
      <telerik:RadPanelBar
        HorizontalAlignment="Left"
        telerik:StyleManager.Theme="Summer">

        <!--first panel bar item-->
        . . .

        <!--second panel bar item-->
        <telerik:RadPanelBarItem Header="Reports">
          <telerik:RadPanelBarItem.Items>
            <TextBlock>Product List</TextBlock>
            <TextBlock>Products By Category</TextBlock>
            <TextBlock>Out of Stock</TextBlock>
          </telerik:RadPanelBarItem.Items>
        </telerik:RadPanelBarItem>

        <!--third panel bar item-->
        <telerik:RadPanelBarItem Header="Options">
          <telerik:RadPanelBarItem.Items>
            <TextBlock>Printer Settings</TextBlock>
            <TextBlock>Roles</TextBlock>
            <TextBlock>Password</TextBlock>
          </telerik:RadPanelBarItem.Items>
        </telerik:RadPanelBarItem>

      </telerik:RadPanelBar>
    </telerik:RadTabItem.Content>
  </telerik:RadTabItem>
</telerik:RadTabControl>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.


Simple text in header
Header accepts **Silverlight** or **Telerik** controls
RadPanelBar
Simple text in header
Header accepts **Silverlight** or **Telerik** controls
RadPanelBar
December - 2010
Simple text in header
Header accepts **Silverlight** or **Telerik** controls
RadPanelBar
Reports
Options

Simple text can go in the content

Content can hold arbitrary **Silverlight** and **Telerik** controls

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
49	28	29	30	1	2	3	4
50	5	6	7	8	9	10	11
51	12	13	14	15	16	17	18
52	19	20	21	22	23	24	25
53	26	27	28	29	30	31	1
2	2	3	4	5	6	7	8

Test Application Features

1) Navigate between tabs of the RadTabControl.
2) Expand and collapse RadPanelBar panels.
3) Notice that each Items collection member (the Favorites icon, calculator, camera, game controller, reports and options items) all react when the mouse rolls over and can be clicked. In this chapter section titled "RadPanelBar" you will see how to handle selected events.

11.4 Control Details

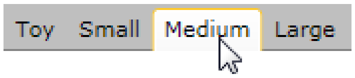
11.4.1 RadTabControl

Defining a RadTabControl in XAML

To minimally define a tab control and its items in XAML you only need a **RadTabControl** outer tag containing a series of **RadTabItem** tags, each with a **Header** property defined. Be aware that RadTabItem is available in the Toolbox and can be dragged to your XAML from there.

```
<telerik:RadTabControl x:Name="tcDogBreeds">
   <telerik:RadTabItem Header="Toy" />
   <telerik:RadTabItem Header="Small" />
   <telerik:RadTabItem Header="Medium" />
   <telerik:RadTabItem Header="Large" />
</telerik:RadTabControl>
```

The example running in the browser looks like the screenshot below:



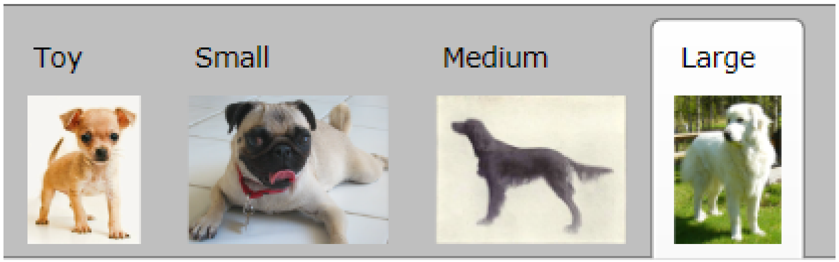
The Header can simply be text as in the example above, or can have a more complex arrangement of any elements. This is done by enclosing content markup within a RadTabItem Header tag and filling it with any Silverlight markup that suits your purpose. For example, the snippet below removes the "Header='Some text'" property and adds the RadTabItem Header tag with a StackPanel inside to handle layout.The StackPanel contains a TextBlock and an Image control.

Notes

Resources not shown in this example are defined in the <UserControl.Resources> tag simply to move non-relevant visual property settings out of the way. If you're interested you can review these resources in the example solution for this chapter.

```
<telerik:RadTabControl
  x:Name="tcDogBreeds">

  <telerik:RadTabItem>
    <telerik:RadTabItem.Header>
      <StackPanel>
        <TextBlock Text="Toy"
             Style="{StaticResource CaptionStyle}" />
        <Image Style="{StaticResource ImageStyle}"
         Source="../Images/ToyDog.png" />
      </StackPanel>
    </telerik:RadTabItem.Header>
  </telerik:RadTabItem>

 . . .

</telerik:RadTabControl>
```

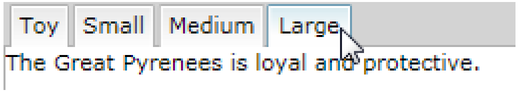
The result in the browser looks like the screenshot below:



The RadTabItem Content works in a similar manner to the Header. You could define "Content" just using strings...

```
<telerik:RadTabControl
  x:Name="tcDogBreeds" telerik:StyleManager.Theme="Summer">
  <telerik:RadTabItem
    Header="Toy"
    Content="The Chihuahua is a brave and good natured dog." />
  <telerik:RadTabItem
    Header="Small"
    Content="The Pug is gentle and affectionate." />
  <telerik:RadTabItem
    Header="Medium"
    Content="The Irish Setter is active and affectionate." />
  <telerik:RadTabItem
    Header="Large"
    Content="The Great Pyrenees is loyal and protective." />
</telerik:RadTabControl>
```

This results in the simple display shown in the screenshot below:

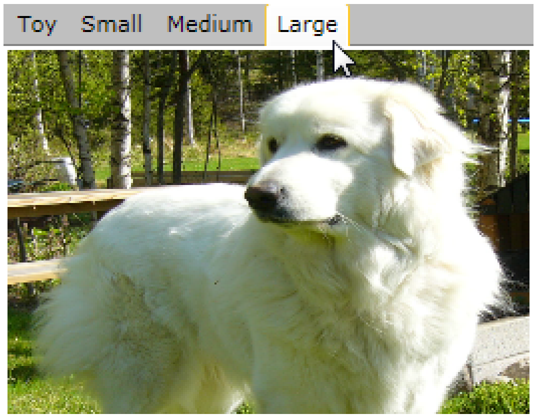


The Great Pyrenees is loyal and protective.

More commonly you want to define a **RadTabItem.Content** tag and add other markup for a richer display as demonstrated in this snippet:

```
<telerik:RadTabControl x:Name="tcDogBreeds">

  <telerik:RadTabItem Header="Toy">
    <telerik:RadTabItem.Content>
      <Border Style="{StaticResource ItemBorderStyle}">
        <Image Style="{StaticResource ImageStyle}"
          Source="../Images/ToyDog.png" />
      </Border>
     </telerik:RadTabItem.Content>
  </telerik:RadTabItem>
  . . .
```

The end result has simple text in the Header with Border and Image controls inside the RadTabItem.Content tags.


Toy
Small
Medium
Large

Creating Tabs in Code

To add a tab directly in code, create a RadTabItem, set the Header and Content properties, then add the RadTabItem to the RadTabControl Items collection.

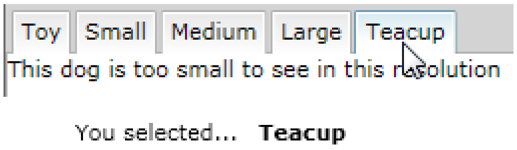
VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim item As RadTabItem = _
New RadTabItem With { _
.Header = "Teacup", _
.Content = "This dog is too small to see in this resolution"}
  tcDogBreeds.Items.Add(item)
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadTabItem item = new RadTabItem
   {
      Header = "Teacup",
      Content = "This dog is too small to see in this resolution"
   };
   tcDogBreeds.Items.Add(item);
}
```

The tab displays after tabs already defined in the XAML, as shown in the screenshot below.



This dog is too small to see in this resolution

You selected... Teacup

Responding to Tab Selections

Handle the **SelectionChanged** event to respond when the user clicks a tab. You can cast the "sender" argument of the event handler to **RadTabControl**. From there you can access the **SelectedItem** property and cast it to **RadTabItem**. In this example the XAML defines a TextBlock and Image to reflect the selected tab information. Some of the tabs have images, but the "Teacup" item only has text in the content. The code tests to see if the content is an image and uses the image if its available. Likewise, the code tests to see if any TextBlock controls exist in the item header and uses that text if available.

VB

```
Private Sub tcDogBreeds_SelectionChanged( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim tabControl As RadTabControl = TryCast(sender, RadTabControl)
  If tabControl.SelectedContent IsNot Nothing Then
    Dim item As RadTabItem = TryCast(tabControl.SelectedItem, RadTabItem)

    Dim image As Image = TryCast(item.Content, Image)
    imgSelected.Source = If(image IsNot Nothing, image.Source, Nothing)

    Dim textBlocks As IList(Of TextBlock) = _
(TryCast(item.Header, UIElement)).ChildrenOfType(Of TextBlock)()
    tbSelected.Text = _
If(textBlocks.Count > 0, textBlocks(0).Text, String.Empty)
  End If
End Sub
```

C#

```
private void tcDogBreeds_SelectionChanged(object sender, RoutedEventArgs e)
{
   RadTabControl tabControl = sender as RadTabControl;
   if (tabControl.SelectedContent != null)
   {
      RadTabItem item = tabControl.SelectedItem as RadTabItem;

      Image image = item.Content as Image;
      imgSelected.Source = image != null ? image.Source : null;

      IList<TextBlock> textBlocks = (item.Header as UIElement).ChildrenOfType<TextBlock>();
      tbSelected.Text = textBlocks.Count > 0 ? textBlocks[0].Text : String.Empty;
   }
}
```

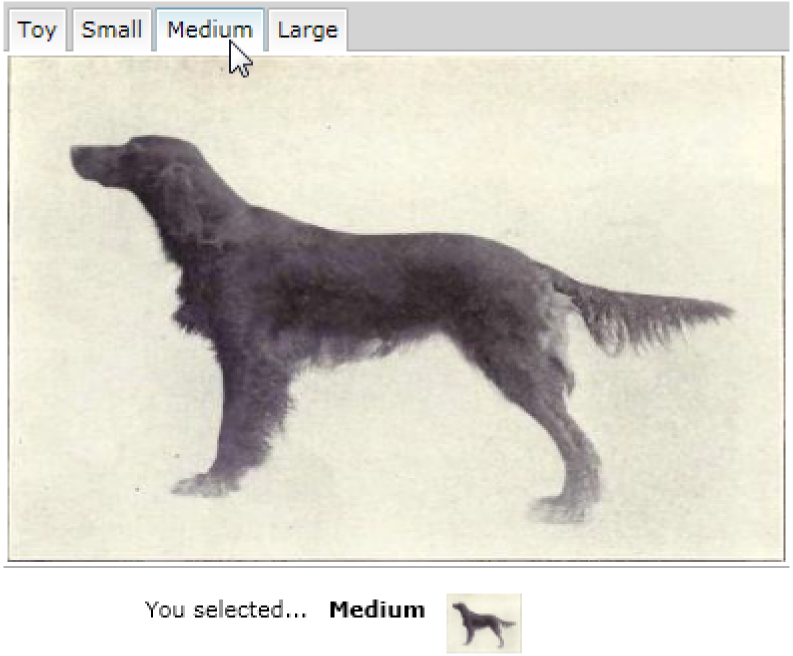
Notes

If you need only the content for the current selection then read the RadTabControl.**SelectedContent** property. You can also get the position of the current selection using the **SelectedIndex** property that indexes into the RadTabControl Items[] collection.

Tip!

The Telerik.Controls.Windows namespace defines a UIElementExtensions class that has three handy extension methods. ChildrenOfType<T>() takes a UIElement and returns a IList<T> of all children in the visual tree for a given type. FindChildByType<T>() takes a UIElement and returns the first child of the given type. ParentOfType<T>() takes a UIElement and returns a parent element.

The example running in the browser looks like the screenshot below. For the full source, see the solution for this chapter.



You selected... **Medium**

Embedding Controls in Tabs

One frequently asked question that comes up in the Telerik Forums is "How do I embed a control in a tab header or content area"? Particularly, people want to know how to put a FireFox style "close" button with an "X" graphic in a tab. Adding a Button with Image content is really a variation on adding any kind of content to the RadTabItem.Header. There are a number of ways to get this done including:

- Define the close button directly in XAML. This route is relatively straightforward with few surprises.
- Duplicate the same XAML elements in the code-behind. This way of getting the job done has a couple minor twists but is conceptually the same as the XAML-only route.
- Soft-code the layout using templates. This is the easiest to maintain but has the steepest learning curve in terms of Silverlight/WPF related concepts and mechanisms. See the Customization section of this chapter for a discussion on how to override the RadTabItem ControlTemplate.

Let's first look at building the close button directly into the XAML. Inside the RadTabItem.Header tag we place a StackPanel to contain a TextBlock and a Button. Inside the Button tag we place a Button.Content with an Image inside that. The short version of the tag organization is...

```
RadTabItem.Header
 StackPanel
  Button
   Button.Content
    Image
```

Take a look at the XAML below to see the markup for a single RadTabItem:

```xml
<telerik:RadTabControl x:Name="tcDogBreeds"
   SelectionChanged="tcDogBreeds_SelectionChanged">
   <telerik:RadTabItem  Style="{StaticResource TabItemStyle}">
      <telerik:RadTabItem.Header>
         <StackPanel Style="{StaticResource HorizontalPanelStyle}">
            <TextBlock Text="Toy" Style="{StaticResource CaptionStyle}" />
            <Button Click="Button_Click">
               <Button.Content>
                  <Image Style="{StaticResource ButtonImageStyle}"
                  Source="../Images/Close.png" />
               </Button.Content>
            </Button>
         </StackPanel>
      </telerik:RadTabItem.Header>

      <Image Style="{StaticResource ImageStyle}"
         Source="../Images/ToyDog.png" />
   </telerik:RadTabItem>
   . . .
</telerik:RadTabControl>
```

The remaining RadTabItem objects in the example XAML have a simple text header and an Image in the content. How could you populate the remaining tabs to replace the simple text header with the same items used in the XAML above? The code below runs in the UserControl Loaded event handler. First it gets all the styles it will need from UserControl.Resources. The remaining code iterates all the RadTabItem instances in the RadTabControl Items collection. If the RadTabItem Header does not have a StackPanel we assume it needs the entire StackPanel-TextBlock-Button-Image combination. The StackPanel is then built, replicating the settings of the previous XAML example.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  '. . .

  ' get all the styles we need from User.Resources
  Dim TabItemStyle As Style = _
CType(Me.Resources("TabItemStyle"), Style)
  Dim HorizontalPanelStyle As Style = _
CType(Me.Resources("HorizontalPanelStyle"), Style)
  Dim CaptionStyle As Style = _
CType(Me.Resources("CaptionStyle"), Style)
  Dim ButtonImageStyle As Style = _
CType(Me.Resources("ButtonImageStyle"), Style)

  ' add close buttons to any tabs that don't have them
  For Each tab As RadTabItem In tcDogBreeds.Items
    ' this item doesn't have a stack panel with a close button
    If Not(TypeOf tab.Header Is StackPanel) Then
      ' get the header text
      Dim caption As String = tab.Header.ToString()

      ' create the stack panel, text block, button and image
      Dim stackPanel As New StackPanel()
      stackPanel.Style = HorizontalPanelStyle

      Dim textBlock As New TextBlock()
      textBlock.Style = CaptionStyle
      textBlock.Text = caption

      Dim button As New Button()
      ' Note: BitmapImage needs a reference to System.Windows.Media.Imaging

      ' create an Image (a FrameworkElement) that can be styled
      Dim image As New Image()
      image.Source = _
New BitmapImage(New Uri("../images/Close.png", UriKind.Relative))
      image.Style = ButtonImageStyle
      ' put the "X" image inside the button
      button.Content = image
      AddHandler button.Click, AddressOf Button_Click

      ' assemble the stack panel pieces
      stackPanel.Children.Add(textBlock)
      stackPanel.Children.Add(button)
      tab.Header = stackPanel
      tab.Style = TabItemStyle

    End If
  Next tab
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
  // . . .

  // get all the styles we need from User.Resources
  Style TabItemStyle =
   (Style)this.Resources["TabItemStyle"];
  Style HorizontalPanelStyle =
   (Style)this.Resources["HorizontalPanelStyle"];
  Style CaptionStyle =
   (Style)this.Resources["CaptionStyle"];
  Style ButtonImageStyle =
   (Style)this.Resources["ButtonImageStyle"];

  // add close buttons to any tabs that don't have them
  foreach (RadTabItem tab in tcDogBreeds.Items)
  {
    // this item doesn't have a stack panel with a close button
    if (!(tab.Header is StackPanel))
    {
      // get the header text
      string caption = tab.Header.ToString();

      // create the stack panel, text block, button and image
      StackPanel stackPanel = new StackPanel();
      stackPanel.Style = HorizontalPanelStyle;

      TextBlock textBlock = new TextBlock();
      textBlock.Style = CaptionStyle;
      textBlock.Text = caption;

      Button button = new Button();
      // Note: BitmapImage needs a reference to
      // System.Windows.Media.Imaging

      // create an Image (a FrameworkElement) that can be styled
      Image image = new Image();
      image.Source =
        new BitmapImage(
         new Uri("../images/Close.png", UriKind.Relative));
      image.Style = ButtonImageStyle;
      // put the "X" image inside the button
      button.Content = image;
```

```
            button.Click += new RoutedEventHandler(Button_Click);

            // assemble the stack panel pieces
            stackPanel.Children.Add(textBlock);
            stackPanel.Children.Add(button);
            tab.Header = stackPanel;
            tab.Style = TabItemStyle;
        }
    }
}
```

The Button Click event handler gets the RadTabItem that the button is sitting in, using the ParentOfType<>() extension method. The RadTabItem is then removed from RadTabControl Items collection.

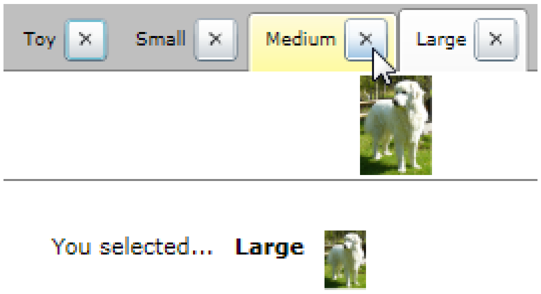
VB

```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' get the button that was clicked and get the parent tab
  Dim button As Button = TryCast(sender, Button)
  Dim tab As RadTabItem = button.ParentOfType(Of RadTabItem)()
  ' delete the tab
  tcDogBreeds.Items.Remove(tab)

  ' clear the text and image in case
  ' no tabs are left to delete
  tbSelected.Text = String.Empty
  imgSelected.Source = Nothing
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   // get the button that was clicked and get the parent tab
   Button button = sender as Button;
   RadTabItem tab = button.ParentOfType<RadTabItem>();
   // delete the tab
   tcDogBreeds.Items.Remove(tab);

   // clear the text and image in case
   // no tabs are left to delete
   tbSelected.Text = String.Empty;
   imgSelected.Source = null;
}
```

Running in the browser, the tabs look like the screenshot below where some of the tabs have already been closed.



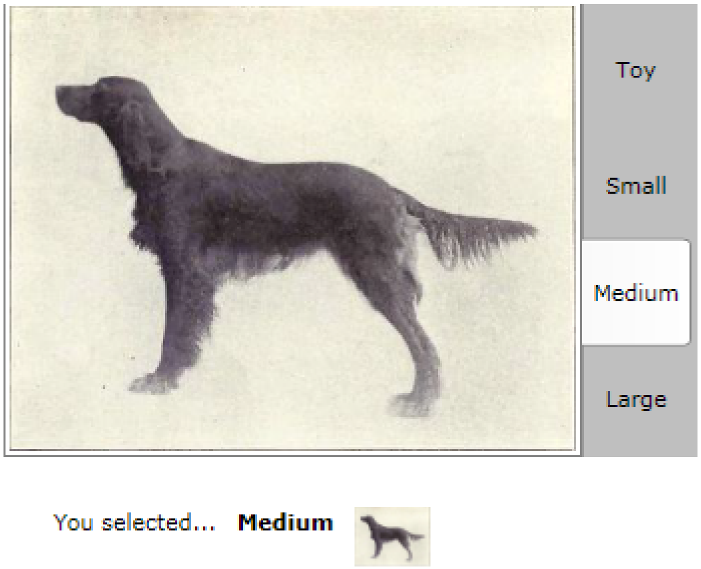
You selected... **Large**

Layout and Placement

Here are the significant properties that control how the RadTabControl and its individual tabs are arranged:

- The **Align** property arranges tab items **Left** (the default), **Right**, **Center** or **Justify**. If you use Justify, the items are scaled proportionally to fill the available space. If there isn't enough space to contain the tabs, each tab width is reduced and clipped if necessary. The tab items do not automatically move to the next row. Use the RadTabItem **IsBreak** property to move tabs to the next row.
- **TabStripPlacement** controls where the tabs are docked relative to the content. Tabs can be placed **Left**, **Right**, **Top** or **Bottom**.
- The **ReorderTabRows** property comes into play only when there are multiple rows of tabs, i.e. when a RadTabItem has its IsBreak property set to true. When ReorderTabRows is true (the default), rows are re-ordered so that the selected item is always in the row nearest to the content. For example, if TabStripPlacement is Top the selected item is in the bottommost row.
- **TabOrientation** determines if the individual tabs orientation is **Vertical** or **Horizontal**.
- **AllTabsEqualHeight** when true (the default) determines that the height of all tab items will be equal to the tallest item. This adjustment takes place on a per-row basis and may be different for each row.

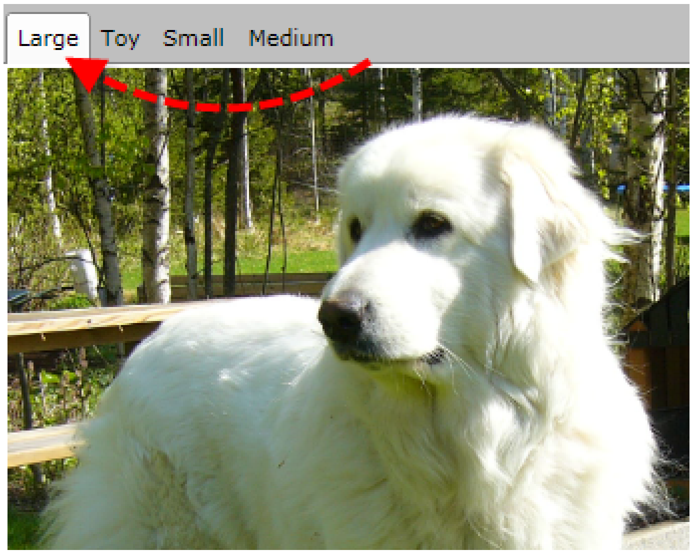
  The tab control in the example screenshot below has its TabOrientation property set to Vertical so that the header text runs from top-to-bottom, not left-to-right. The Align property is "Justify", causing the tabs to take the entire length next to the content instead of being bunched on the left, right or center. TabStripPlacement is set to "Right".



You selected... **Medium**

Drag and Drop

Tabs can be dragged within the tab control if you set the **AllowDragReorder** property to true.



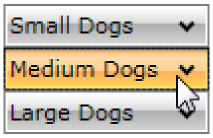
11.4.2 RadPanelBar

Defining RadPanelBar in XAML

Minimally defining a RadPanelBar and its items in XAML follows much the same pattern as RadTabControl. You only need a **RadPanelBar** outer tag containing a series of **RadPanelBarItem** tags, each with a **Header** property defined. Be aware that RadPanelBarItem is available in the Toolbox and can be dragged to your XAML from there.

```
<telerik:RadPanelBar
   x:Name="pbDogBreeds"
   HorizontalAlignment="Left"
   MaxWidth="300"
   Margin="20">
    <telerik:RadPanelBarItem Header="Small Dogs" />
    <telerik:RadPanelBarItem Header="Medium Dogs" />
    <telerik:RadPanelBarItem Header="Large Dogs" />
</telerik:RadPanelBar>
```

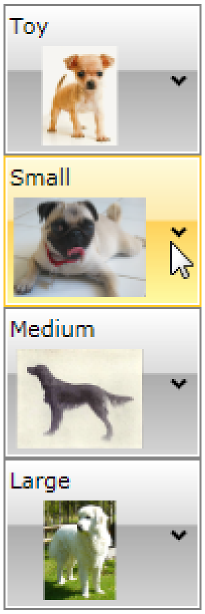
The example running in the browser looks like the screenshot below:



Creating panel bar item headers with more complex arrangements of contents also follows the same pattern as RadTabControl. Inside each RadPanelBarItem tag add a RadPanelBarItem.Header, and within the header add any contents that suit your purpose. Typically you will find a layout panel of some sort inside the RadPanelBarItem.Header tag, such as a StackPanel or Grid.

```
<telerik:RadPanelBar  x:Name="pbDogBreeds"
  HorizontalAlignment="Left"
  MaxWidth="300" Margin="20">
  <telerik:RadPanelBarItem>
    <telerik:RadPanelBarItem.Header>
      <StackPanel>
        <TextBlock Text="Toy"></TextBlock>
        <Border Style="{StaticResource ItemBorderStyle}">
          <Image Style="{StaticResource ImageStyle}"
            Source="../Images/ToyDog.png" />
        </Border>
      </StackPanel>
    </telerik:RadPanelBarItem.Header>
  </telerik:RadPanelBarItem>
 . . .
</telerik:RadPanelBar>
```

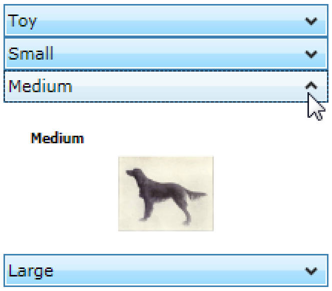
The panel bar running in the browser looks like the screenshot below.



Adding content to the items area below the header in the panel bar again follows the same pattern as RadTabControl, but instead of a Content tag we use RadPanelBarItem.Items. The example below adds a single item inside each header.

```
<telerik:RadPanelBar x:Name="pbDogBreeds"
  Margin="20" telerik:StyleManager.Theme="Vista">
  <telerik:RadPanelBarItem Header="Toy">
    <telerik:RadPanelBarItem.Items>
      <StackPanel Style="{StaticResource StackPanelStyle}">
        <TextBlock
          Text="Toy"
          Style="{StaticResource CaptionStyle}">
        </TextBlock>
        <Image
          Style="{StaticResource ImageStyle}"
          Source="../Images/ToyDog.png" />
      </StackPanel>
    </telerik:RadPanelBarItem.Items>
  </telerik:RadPanelBarItem>
  . . .
</telerik:RadPanelBar>
```

The results in the browser look something like the screenshot below.



You can define multiple items as shown in the XAML below. Notice that RadPanelBarItem.Items tag is the default and can be left out.

```
<telerik:RadPanelBar
  x:Name="pbDogBreeds"
  Margin="20"
  telerik:StyleManager.Theme="Vista">

  <telerik:RadPanelBarItem
    Header="Toy">
    <TextBlock
      Text="Chihuahua"
      Style="{StaticResource CaptionStyle}"></TextBlock>
    <TextBlock
      Text="English Toy Spaniel"
      Style="{StaticResource CaptionStyle}"></TextBlock>
    <TextBlock
      Text="Pekingese"
      Style="{StaticResource CaptionStyle}"></TextBlock>
  </telerik:RadPanelBarItem>

  . . .

</telerik:RadPanelBar>
```

Creating RadPanelBar Items in Code

There are two steps to creating panel bar items:

1. Create the RadPanelBarItem and define its Header property.
2. Add to the RadPanelBarItem.Items collection. This second step can take any object type.

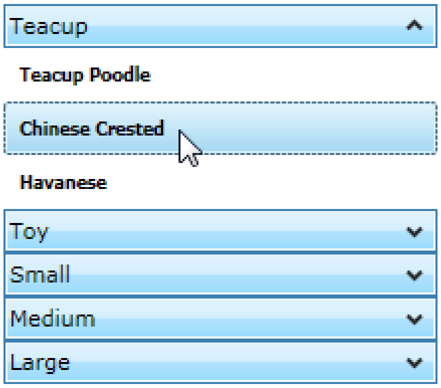
This next example adds a panel bar item headed "Teacup" and adds a series of TextBlock elements to the Items collection. Notice in this example...

- The RadPanelBarItem with the Header defined is inserted to the head of the list using the RadPanelBar.Items.Insert() method, rather than using the Add() method.
- The panel bar items that already exist in the XAML use a style defined in UserControl.Resources called "CaptionStyle". Now that we're adding more items on-the-fly, we need to apply that style to make the appearance match. Notice the code that extracts "CaptionStyle" to a Style object. When the TextBlock object is created, the object initializer for the TextBlock assigns the Style property.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim teacupBreeds As List(Of String) = _
New List(Of String) ( _
New String() {"Teacup Poodle", "Chinese Crested", "Havanese"})

  ' get "CaptionStyle" from UserControl.Resources so
  ' added items look identical to items added in xaml
  Dim style As Style = CType(Me.Resources("CaptionStyle"), Style)

  ' Create new "Teacup" item header
  Dim teacupItem As RadPanelBarItem = _
New RadPanelBarItem With {.Header = "Teacup"}

  ' put new "Teacup" item header at the top of list
  pbDogBreeds.Items.Insert(0, teacupItem)

  ' add a list of TextBlock items under the "Teacup" header,
  ' set style to match existing, xaml-defined TextBlocks
  For Each breed As String In teacupBreeds
    teacupItem.Items.Add(New TextBlock With {.Text = breed, .Style = style})
  Next breed
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   List<string> teacupBreeds =
      new List<string>
      {
         "Teacup Poodle", "Chinese Crested", "Havanese"
      };

   // get "CaptionStyle" from UserControl.Resources so
   // added items look identical to items added in xaml
   Style style = (Style)this.Resources["CaptionStyle"];

   // Create new "Teacup" item header
   RadPanelBarItem teacupItem =
      new RadPanelBarItem { Header = "Teacup" };

   // put new "Teacup" item header at the top of list
   pbDogBreeds.Items.Insert(0, teacupItem);

   // add a list of TextBlock items under the "Teacup" header,
   // set style to match existing, xaml-defined TextBlocks
   foreach (string breed in teacupBreeds)
   {
      teacupItem.Items.Add(new TextBlock { Text = breed, Style = style });
   }
}
```

Running in the browser, the new "Teacup" RadPanelBarItem shows at the beginning of the list with three new TextBlock controls added to the Items collection.



Events

You can react to panels opening and closing using the event pairs **PreviewExpanded**/**Expanded** and **PreviewCollapsed**/**Collapsed**. All four of these events provide access to the expanding/collapsing panel through the **OriginalSource** property. The "Preview" event versions allow you to cancel the event by setting the **Handled** property to true. The example below prevents the panel from expanding if there are fewer than four items in it.

VB

```
Private Sub pbDogBreeds_PreviewExpanded( _
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  Dim item As RadPanelBarItem = TryCast(e.OriginalSource, RadPanelBarItem)
  e.Handled = item.Items.Count < 4
End Sub
```

C#

```
private void pbDogBreeds_PreviewExpanded(
   object sender, Telerik.Windows.RadRoutedEventArgs e)
{
   RadPanelBarItem item = e.OriginalSource as RadPanelBarItem;
   e.Handled = item.Items.Count < 4;
}
```

The **Selected** event fires for all items including the header item and all the items underneath the header. The RadPanelBarItem **Level** property can give you a hand figuring out which item you have a reference to, i. e. "1" for the header item and "2" for the items underneath the header. The RadPanelBarItem **Item** property is a reference to the object actually being displayed.

The example below gets a reference to the RadPanelBarItem. If the Level is "2", the Item property is assumed to be a TextBlock and cast as such.

VB

```
Private Sub pbDogBreeds_Selected( _
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  Dim item As RadPanelBarItem = TryCast(e.OriginalSource, RadPanelBarItem)

  If item.Level = 2 Then
    Dim tb As TextBlock = TryCast(item.Item, TextBlock)
    RadWindow.Alert("You clicked item " & tb.Text)
  End If
End Sub
```

C#

```
private void pbDogBreeds_Selected(object sender, Telerik.Windows.RadRoutedEventArgs e)
{
   RadPanelBarItem item = e.OriginalSource as RadPanelBarItem;

   if (item.Level == 2)
   {
      TextBlock tb = item.Item as TextBlock;
      RadWindow.Alert("You clicked item " + tb.Text);
   }
}
```

11.5 Customization

While styles can let you change a number of properties all at one time, templates provide the powerful ability to completely change the makeup of a control without breaking its functionality. In this walk through you will define new contents for the RadTabItem "TopTemplate" that defines the layout of the control when the TabStripPlacement is "Top".

Notes

We can define templates in Expression Blend, but it helps sometimes to see how the XAML is structured so this time we will make these changes manually. Typically you will want to use Blend not only for the ease-of-use, but because Blend does a much more complete job of making sure all the styles, templates and brushes are included in your XAML.

Notes

RadTabItem and RadTabControl have four templates each, not just one. These templates change depending on the TabStripPlacement of the tab control and have corresponding names, i.e. "BottomTemplate", "TopTemplate", "LeftTemplate" and "RightTemplate". This approach was chosen by Telerik for performance reasons. The alternative would have been to have visual elements in a single template with different parts shown or hidden as needed.

"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) Telerik.Windows.Controls

b) Telerik.Windows.Controls.Navigation

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the XAML below to the XML name space attributes section of the UserControl tag.

```
<UserControl
    xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
    xmlns:Telerik_Windows_Controls_Primitives="clr-namespace:Telerik.Windows.Controls.Primitives;
assembly=Telerik.Windows.Controls.Navigation"
    . . .
>
```

3) Inside the UserControl tag, add the UserControl.Resources below. We'll use the comments to include later the style for the RadTabItem and the top template:

```
<UserControl.Resources>

  <!--ScovilleTopTemplate-->

  <!--ScovilleTab Style-->

</UserControl.Resources>
```

4) Replace the "<!--ScovilleTopTemplate-->" comment with the XAML below.

The XAML is a control template that will replace the top tabs. Notice that the template is encapsulated with a standard Silverlight Grid named "wrapper". Be sure to leave the "wrapper" name in place or the logic that changes content in response to clicking tabs will not work. Inside the Grid is an Ellipse and a TabItemContentPresenter. The TabItemContentPresenter renders the header content.

```
<!--ScovilleTopTemplate-->
<ControlTemplate x:Key="ScovilleTopTemplate" TargetType="telerik:RadTabItem">
  <Grid x:Name="wrapper">
    <Ellipse>
      <Ellipse.Fill>
        <RadialGradientBrush>
          <GradientStop Color="Red" Offset="0.1" />
          <GradientStop Color="DarkRed" Offset="0.5" />
          <GradientStop Color="Maroon" Offset=".7" />
        </RadialGradientBrush>
      </Ellipse.Fill>
    </Ellipse>

    <Telerik_Windows_Controls_Primitives:TabItemContentPresenter
        x:Name="HeaderElement"
        Content="{TemplateBinding Header}"
        Foreground="{TemplateBinding Foreground}"
        Margin="{TemplateBinding Margin}" />
  </Grid>
</ControlTemplate>
```

5) Replace the "<!--ScovilleTab Style-->" comment with the "ScovilleTab" style XAML below. The style incorporates the TopTemplate property of the RadTabItem.

```
<!--ScovilleTab Style-->
<Style x:Key="ScovilleTab" TargetType="telerik:RadTabItem">
  <Setter Property="TopTemplate" Value="{StaticResource ScovilleTopTemplate}" />
  <Setter Property="Background" Value="White" />
  <Setter Property="Foreground" Value="White" />
  <Setter Property="Margin" Value="10" />
</Style>
```

6) Drag a RadTabControl from the Toolbox to a point inside the main "LayoutRoot" grid. Set the **HorizontalAlignment** to "Left", **VerticalAlignment** to "Top" and **BackgroundVisibility** to "Collapsed". Add three RadTabItem and set the Style for each to "ScovilleTab". Set both the **Content** and **Header** to "Very Spicy", "Hot" and "Mild".

```
<telerik:RadTabControl HorizontalAlignment="Left"
    VerticalAlignment="Top"
    BackgroundVisibility="Collapsed">
  <telerik:RadTabItem Header="Very Spicy"
      Content="Very Spicy"
      Style="{StaticResource ScovilleTab}" />
  <telerik:RadTabItem Header="Hot" Content="Hot"
      Style="{StaticResource ScovilleTab}" />
  <telerik:RadTabItem Header="Mild" Content="Mild"
      Style="{StaticResource ScovilleTab}" />
</telerik:RadTabControl>
```

7) Here's all the XAML together so far.

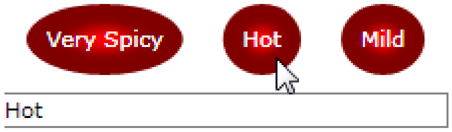
```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
    xmlns:Telerik_Windows_Controls_Primitives="clr-namespace:Telerik.Windows.Controls.Primitives;
assembly=Telerik.Windows.Controls.Navigation"
    x:Class="TabTemplateTest.MainPage" Width="640" Height="480">
  <UserControl.Resources>

    <!--ScovilleTopTemplate-->
    <ControlTemplate x:Key="ScovilleTopTemplate"
        TargetType="telerik:RadTabItem">
      <Grid x:Name="wrapper">
        <Ellipse>
          <Ellipse.Fill>
            <RadialGradientBrush>
              <GradientStop Color="Red" Offset="0.1" />
              <GradientStop Color="DarkRed" Offset="0.5" />
              <GradientStop Color="Maroon" Offset=".7" />
            </RadialGradientBrush>
          </Ellipse.Fill>
        </Ellipse>

        <Telerik_Windows_Controls_Primitives:TabItemContentPresenter
            x:Name="HeaderElement"
            Content="{TemplateBinding Header}"
            Foreground="{TemplateBinding Foreground}"
            Margin="{TemplateBinding Margin}" />
      </Grid>
    </ControlTemplate>

    <!--ScovilleTab Style-->
    <Style x:Key="ScovilleTab"
        TargetType="telerik:RadTabItem">
      <Setter Property="TopTemplate"
          Value="{StaticResource ScovilleTopTemplate}" />
      <Setter Property="Background" Value="White" />
      <Setter Property="Foreground" Value="White" />
      <Setter Property="Margin" Value="10" />

    </Style>
  </UserControl.Resources>
  . . .
</UserControl>
```

Gotcha!

TabStripPlacement must correspond to the template you're defining or you will only see the default appearance of the control. If you're using TabStripPlacement = "Top", then the "TopTemplate" must be defined and visa-versa.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

- Click each of the RadTabItem buttons and notice that the corresponding content changes. The significance of this is that we completely replace parts of the RadTabItem internals, but the tabs still work as before.

Ideas for Extending This Example

- Try adding a Click event with an RadWindow.Alert() called from the handler. Once again, we can see that the functionality for the tab control is completely intact.
- Try changing the elements that make up the RadTabItem ControlTemplate.
- Right-click the MainPage.xaml file and select "Open in Expression Blend" from the context menu. You can continue to edit the page and the template within Blend.

11.6 Wrap Up

In this chapter you learned how to create tabbed navigation systems and interfaces using RadTabControl and RadPanelBar controls. You built tab controls and panel bars directly in the XAML and programmatically. You also assigned simple text to the header and content areas of each control and learned how to add to the header and content with XAML markup of arbitrary complexity. You learned how to embed controls into the RadTabControl header. You also learned how to handle the significant events of each control. Finally, you learned how to completely customize a RadTabItem by overriding the ControlTemplate.

Part XII

ToolBar

12 ToolBar

12.1 Objectives

In this chapter you'll see how to organize multiple Silverlight controls into horizontal or vertical strips using RadToolBar. You will use RadToolBarSeparator to visually divide groups of controls. You will see how the theming mechanism for RadToolBar automatically styles both the tool bar and its items. You will use the RadToolBar OverflowMode property to handle overflow behavior when there are more controls than can be displayed at one time. You will handle events that react to the overflow area expanding and collapsing. You will add items to the tool bar in XAML, using the programmatic API and through data binding.

You will use the RadToolBarTray to manage RadToolBar position, sizing and order.

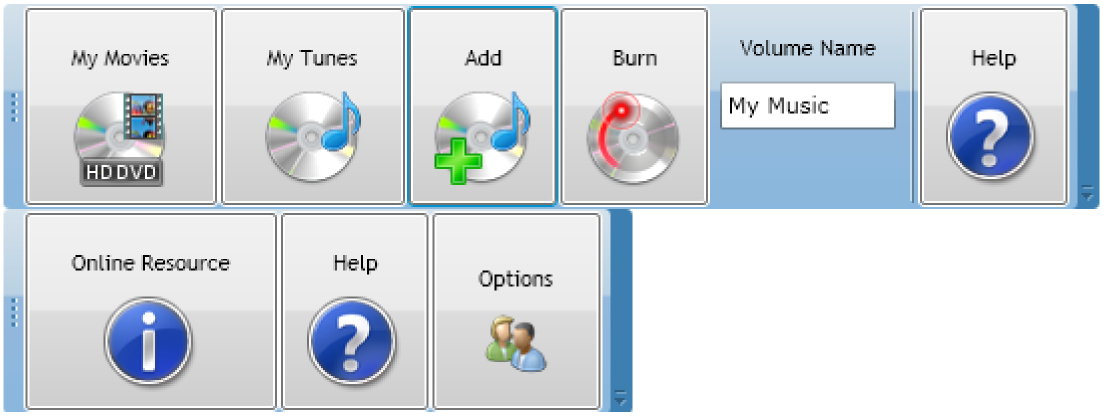
Finally, you will create custom templates for the RadToolBar background and RadToolBarSeparator.

Find the projects for this chapter at...

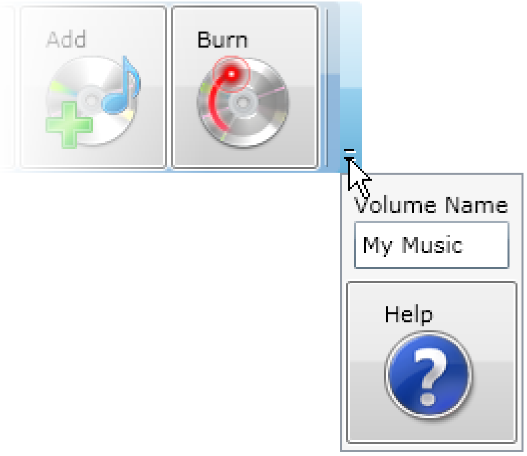
\Courseware\Projects\<CS|VB>\Toolbar\Toolbar.sln.

12.2 Overview

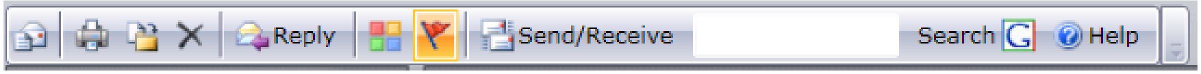
RadToolBar organizes multiple Silverlight controls into a strip where they can be presented in Horizontal or Vertical orientation. Multiple RadToolBar controls can be managed using the **RadToolBarTray**.



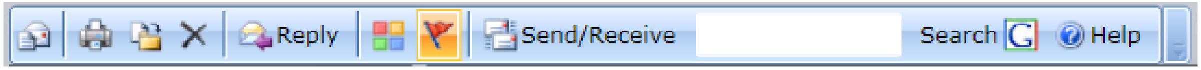
When the browser is resized, buttons that don't fit the visible area are automatically relocated to an "overflow" panel. The "down" overflow button at the right end of the tool bar displays the overflow panel area.



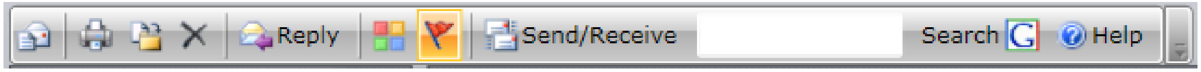
Predefined themes can be applied to the RadToolBarTray and to RadToolBars individually. Common control primitives, e.g. Button, RadioButton, CheckBox, etc, are styled automatically to agree with the RadToolBar theme. Take a look at some of the example screenshots below to see how the theme styles the tool bar and its items:



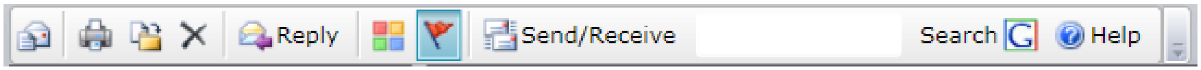
Office_Silver



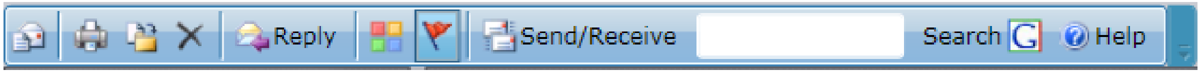
Office_Blue



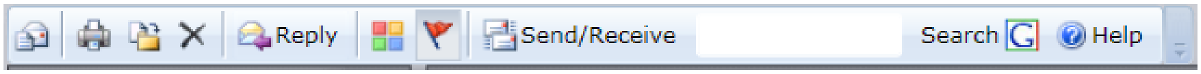
Office_Black



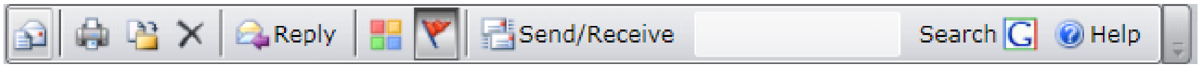
Summer



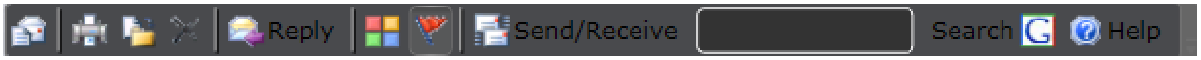
Vista



Windows7



Transparent



Expression_Dark

12.3 Getting Started

In this walk through you will build a simple, single tool bar containing buttons, text boxes and images.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

   a) **Telerik.Windows.Controls**

   b) **Telerik.Windows.Controls.Navigation**

   c) **Telerik.Windows.Themes.Vista**

4) In the Solution Explorer, right-click the project and select **Add > New Folder** from the context menu. Rename the folder "Images".

5) Add the images listed below to the new "Images" folder. The images can be found in the "\courseware\images" directory:

   a) CD_Add.png

   b) CD_Burn.png

   c) CD_Movies.png

   d) CD_Mustic.png

   e) Help.png

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the following reference to the Telerik assemblies in the UserControl element:

```
<UserControl
  ...
  xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation">
  ...
```

3) Add a UserControl.Resources element to the UserControl. Inside the Resources element, add a new Style with Key "ButtonStackPanelStyle" and TargetType "StackPanel". Add a single Setter element that assigns the Property to be "Margin" and the Value to be "5".

```
<UserControl . . . >

  <UserControl.Resources>

    <Style
      x:Key="ButtonStackPanelStyle"
      TargetType="StackPanel">
      <Setter Property="Margin" Value="5" />
    </Style>

  </UserControl.Resources>
  . . .
```

4) Add a **RadToolBarTray** inside the main "LayoutRoot" Grid element, from the Toolbox.

Adding a RadToolBarTray is not strictly necessary in this example because we will add only a single RadToolBar. But the RadToolBarTray does a nice job of constraining the RadToolBar dimensions so that you don't have to define a MaxHeight or HorizontalAlignment. You can try the example both ways, with and without the RadToolBarTray to see how they differ. In later sections of this chapter we will talk about how RadToolBarTray manages multiple RadToolBar controls.

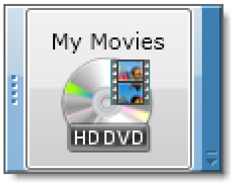
5) Inside the RadToolBarTray, add a **RadToolBar** from the Toolbox. Set the StyleManager.Theme attribute to "Vista". The markup should look something like the example below:

```
<UserControl . . .>

  . . .

  <Grid x:Name="LayoutRoot">

    <telerik:RadToolBarTray>

      <telerik:RadToolBar telerik:StyleManager.Theme="Vista">
      </telerik:RadToolBar>

    </telerik:RadToolBarTray>

  </Grid>

</UserControl>
```

6) Add a Button inside the RadToolBar element. Inside the Button, add a StackPanel and set its Style to the "ButtonStackPanelStyle" resource. Inside the StackPanel, add a TextBlock and an Image. Set the TextBlock Text to "My Movies" and the Image Source to the "images/CD_Movies.png" path.

```
. . .
<Button>
  <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
    <TextBlock Text="My Movies"></TextBlock>
    <Image Source="Images/CD_Movies.png"></Image>
  </StackPanel>
</Button>
. . .
```

7) Press **F5** to run the application. The application displays the tool bar with a single button in it.



8) Add a series of buttons and a TextBlock to the RadToolBar using the XAML below.

```
<Button>
   <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
      <TextBlock Text="My Tunes" ></TextBlock>
      <Image Source="Images/CD_Music.png"></Image>
   </StackPanel>
</Button>
<Button>
   <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
      <TextBlock Text="Add" ></TextBlock>
      <Image Source="Images/CD_Add.png"></Image>
   </StackPanel>
</Button>
<Button>
   <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
      <TextBlock Text="Burn" ></TextBlock>
      <Image Source="Images/CD_Burn.png"></Image>
   </StackPanel>
</Button>
   <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
   <TextBlock Text="Volume Name" ></TextBlock>
   <TextBox Text="My Music"></TextBox>
</StackPanel>
```

9) Drag a RadToolBarSeparator from the Toolbox to a point just below the buttons.

10)Below the RadToolBarSeparator, add one more button:

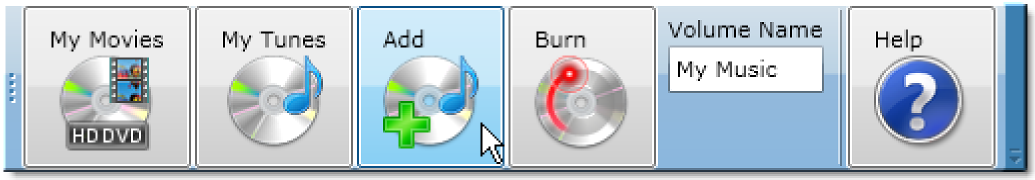
```
<Button>
   <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
      <TextBlock Text="Help" ></TextBlock>
      <Image Source="Images/Help.png"></Image>
   </StackPanel>
</Button>
```

11)The XAML so far should look like the example below:

```
. . .
<telerik:RadToolBarTray>
  <telerik:RadToolBar telerik:StyleManager.Theme="Vista" Margin="10">
    <Button>
      <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
        <TextBlock Text="My Movies"></TextBlock>
        <Image Source="Images/CD_Movies.png"></Image>
      </StackPanel>
    </Button>
    <Button>
      <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
        <TextBlock Text="My Tunes" ></TextBlock>
        <Image Source="Images/CD_Music.png"></Image>
      </StackPanel>
    </Button>
    <Button>
      <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
        <TextBlock Text="Add" ></TextBlock>
        <Image Source="Images/CD_Add.png"></Image>
      </StackPanel>
    </Button>
    <Button>
      <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
        <TextBlock Text="Burn" ></TextBlock>
        <Image Source="Images/CD_Burn.png"></Image>
      </StackPanel>
    </Button>
      <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
      <TextBlock Text="Volume Name" ></TextBlock>
      <TextBox Text="My Music"></TextBox>
    </StackPanel>
    <telerik:RadToolBarSeparator />
    <Button>
      <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
        <TextBlock Text="Help" ></TextBlock>
        <Image Source="Images/Help.png"></Image>
      </StackPanel>
    </Button>
  </telerik:RadToolBar>
</telerik:RadToolBarTray>
. . .
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Ideas for Extending This Example

- Add event handlers for the Button Click events.
- Add other control types to the Items collection.
- Change the Theme of the RadToolBar or RadToolBarTray.

12.4 Control Details

RadToolBarTray

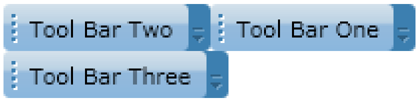
RadToolBarTray contains multiple RadToolBars and manages their position, sizing and order. The tool bar tray is divided into bands where each band can contain multiple tool bars. Use the tool bar's **Band** property to determine which row (or column if the tool bar Orientation is Vertical) that the tool bar appears on. Use the **BandIndex** property to control where the tool bar appears on the Band in relation to other tool bars.

For example, the XAML below defines three tool bars within a RadToolBarTray and sets the Bands and BandIndex properties for each tool bar.

```
<telerik:RadToolBarTray telerik:StyleManager.Theme="Vista" >

  <telerik:RadToolBar Band="0" BandIndex="1">
    <TextBlock Text="Tool Bar One" ></TextBlock>
  </telerik:RadToolBar>

  <telerik:RadToolBar Band="0" BandIndex="0">
    <TextBlock Text="Tool Bar Two" ></TextBlock>
  </telerik:RadToolBar>

  <telerik:RadToolBar Band="1">
    <TextBlock Text="Tool Bar Three" ></TextBlock>
  </telerik:RadToolBar>

</telerik:RadToolBarTray>
```

The screenshot below shows the result of these Band and BandIndex property settings. Tool bars "Tool Bar One" and "Tool Bar Two" have a Band property of "0" so you can expect to see them both on the first row of the RadToolBarTray. The Band property of tool bar "Tool Bar Three" is "1" and appears on the second row of the RadToolBarTray.

The BandIndex property of "Tool Bar Two" is "0" and is placed before "Tool Bar One".


Tool Bar Two
Tool Bar One
Tool Bar Three

Adding Items

You can add items simply by including Silverlight visual elements within the RadToolBar Items tags:

```
<telerik:RadToolBar>
  <telerik:RadToolBar.Items>
    <Button />
    <telerik:RadCalendar />
    <TextBlock />
  </telerik:RadToolBar.Items>
</telerik:RadToolBar>
```

The **Items** sub-element is the default and therefore implicit. You can leave out the Items sub-element and just start adding Silverlight controls inside the RadToolBar element:

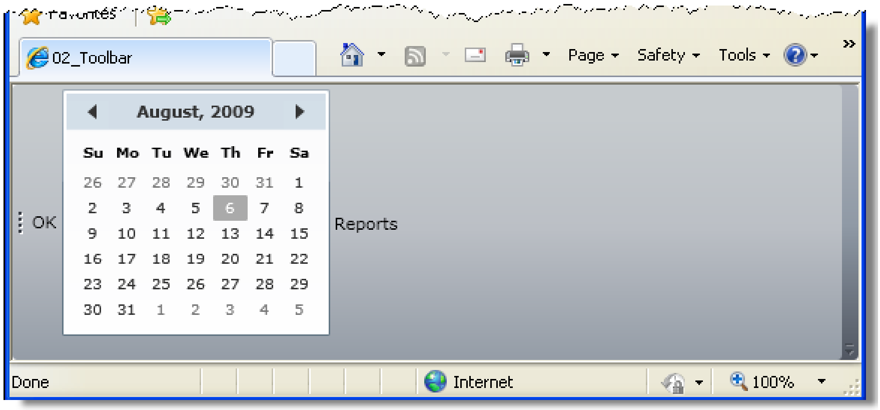
```
<telerik:RadToolBar>
    <Button />
    <telerik:RadCalendar />
    <TextBlock />
</telerik:RadToolBar>
```

To add tool bar items in code, add to the **Items** collection. The example code below adds Silverlight Button, RadCalendar and TextBlock controls as items to the tool bar.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim button As New Button()
  button.Content = "OK"
  tbMain.Items.Add(button)
  tbMain.Items.Add(New Calendar())
  Dim textBlock As New TextBlock()
  textBlock.Text = "Reports"
  tbMain.Items.Add(textBlock)
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   Button button = new Button();
   button.Content = "OK";
   tbMain.Items.Add(button);
   tbMain.Items.Add(new Telerik.Windows.Controls.RadCalendar());
   TextBlock textBlock = new TextBlock();
   textBlock.Text = "Reports";
   tbMain.Items.Add(textBlock);
}
```

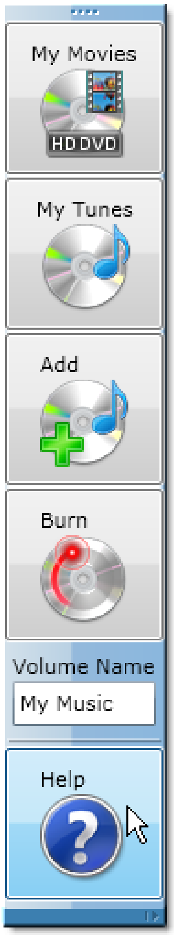
Without a RadToolBarTray and without any special attributes set to control sizing, the tool bar takes up the entire available space in the browser.



August, 2009

Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Orientation

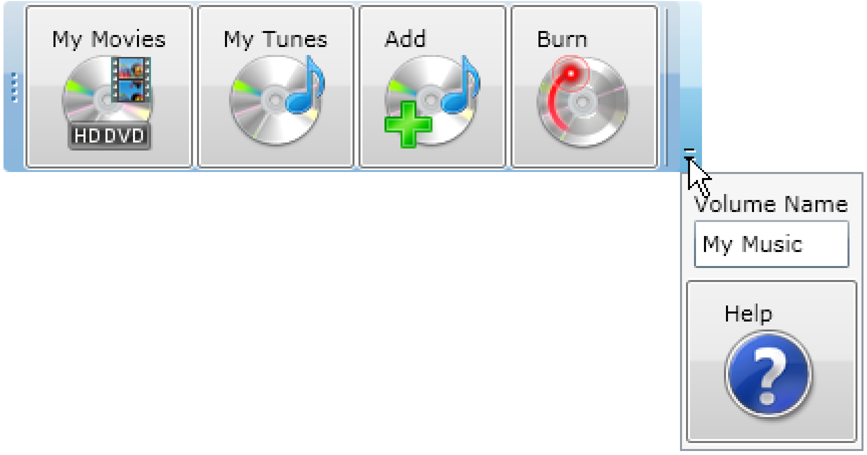


The RadToolBar **Orientation** property displays the tool bar in a Horizontal (default) or Vertical layout. The tool bar here is shown with Orientation = Vertical. The RadToolBarTray also has an Orientation property that takes precedence over the Orientation property for individual tool bars.

Note: You may need to adjust the horizontal and vertical alignments depending on tool bar orientation or the container that holds the tool bar.

Overflow

Tool bar items that can't fit in the visible area are automatically relocated to the overflow panel. The screenshot below shows the overflow panel with two items.



Overflow is managed by setting the **OverflowMode** attached property to individual items within the tool bar. The possible values that control how OverflowMode works on a particular item are:

- **Never**: The item will only be placed in the strip panel.
- **Always**: The item will only be placed in the overflow panel.
- **AsNeeded**: The item will be visible in the strip panel if there's available space, otherwise the item will be located in the overflow panel.

For example, the XAML for the button below determines that it will always be placed in the overflow panel:

```
<telerik:RadToolBar telerik:StyleManager.Theme="Vista" Margin="10" >
  . . .
  <Button telerik:RadToolBar.OverflowMode="Always">
    <StackPanel Style="{StaticResource ButtonStackPanelStyle}">
      <TextBlock Text="Help" ></TextBlock>
      <Image Source="Images/Help.png"></Image>
    </StackPanel>
  </Button>

</telerik:RadToolBar>
```

To set the overflow mode in code, use the static **SetOverflowMode()** method. Pass a reference to the item within the tool bar and an OverflowMode enumeration member.

VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadToolBar.SetOverflowMode(btnHelp, OverflowMode.Always)
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadToolBar.SetOverflowMode(btnHelp, OverflowMode.Always);
}
```

RadToolBar has two additional overflow related properties:

- **HasOverflowItems**: A read-only property that lets you know if there are any items in the overflow panel.
- **IsOverflowOpen**: This property determines if the overflow panel is visible or not.

Themes

You can apply predefined themes to the RadToolBar or RadToolBarTray. Themes automatically apply to certain Silverlight items contained in the tool bar. As of this writing, the controls that are automatically styled include:

- TextBlock
- TextBox
- Button
- CheckBox
- RadioButton
- ToggleButton
- RadToolBarSeparator

Gotcha!

You may encounter a control called "RadSeparator". RadSeparator is technically included in the above list, but is obsolete and included only for backward compatibility.

Events

As the overflow panel is opened or closed, the **OverflowAreaOpened** and **OverflowAreaClosed** routed events fire.

VB

```
Private Sub tbMain_OverflowAreaOpened( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadWindow.Alert("You can also get help by pressing F1")
End Sub
```

C#

```
private void tbMain_OverflowAreaOpened(object sender, RoutedEventArgs e)
{
   RadWindow.Alert("You can also get help by pressing F1");
}
```

12.5 Binding

In this walk through you will populate a tool bar from a custom list of objects. In code you will create a "Tool" object that represents each tool bar item, populate a generic List of Tool objects and assign the list to the tool bar **ItemsSource** property. In the XAML markup you will layout all the Silverlight items to be bound in a RadToolBar **ItemTemplate**. Along the way you will learn how to make the ItemTemplate markup into a resource, create a font "shadow" effect and bind a ToolTip for each button.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Navigation**

c) **Telerik.Windows.Themes.Vista**

4) In the Solution Explorer, add an Image directory. Add the image files below to the Image directory. You can find these images in the "\courseware\images" directory.

a) 3.5_Disk_Drive.png

b) Folder_Open.png

c) Printer.png

d) RecycleBin.png

Create the View Model Object

1) In the Solution Explorer, right-click the project and select **Add > Class...** Copy and paste the code below.

   This simple "Tool" object will supply our data to the "View", i.e. the tool bar.

```
Public Class Tool
  Public Sub New(ByVal title As String, ByVal path As String, _
ByVal description As String)
    Me.Title = title
    Me.Path = path
    Me.Description = description
  End Sub

  Private privateTitle As String
  Public Property Title() As String
    Get
      Return privateTitle
    End Get
    Set(ByVal value As String)
      privateTitle = value
    End Set
  End Property

  Private privatePath As String
  Public Property Path() As String
    Get
      Return privatePath
    End Get
    Set(ByVal value As String)
      privatePath = value
    End Set
  End Property

  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
End Class
```

C#

```
public class Tool
{
   public Tool(string title, string path, string description)
   {
      this.Title = title;
      this.Path = path;
      this.Description = description;
   }

   public string Title
   { get; set; }

   public string Path
   { get; set; }

   public string Description
   { get; set; }
}
```

XAML Editing

1) Open MainPage.xaml for editing.

2) Drag a RadToolBarTray from the Toolbox to the main "LayoutRoot" Grid element. Add a "telerik:StyleManager.Theme" attribute to the RadToolBarTray and set its value to "Vista".

3) Drag a RadToolBar from the Toolbox to the RadToolBarTray element. Set the "x:Name" attribute to "tbMain".

   Setting the x:Name attribute will allow us to reference the tool bar later in code.

4) Inside the RadToolBar element, add a RadToolBar ItemTemplate. Inside the RadToolBar ItemTemplate, add a DataTemplate. The XAML should now look something like the example below.

Each record within the ItemsSource will correspond to a tool bar item. When the ItemsSource property is assigned data, the ItemTemplate will contain the Silverlight elements that display for each record in the ItemsSource. Binding expressions in the ItemTemplate will decide where the data for each column will be placed.

```
<Grid x:Name="LayoutRoot">
  <telerik:RadToolBarTray
      telerik:StyleManager.Theme="Vista">
    <telerik:RadToolBar x:Name="tbMain">
      <telerik:RadToolBar.ItemTemplate>
        <DataTemplate>

        </DataTemplate>
      </telerik:RadToolBar.ItemTemplate>
    </telerik:RadToolBar>
  </telerik:RadToolBarTray>
</Grid>
```

5) Add a Button control inside the DataTemplate, set the "x:Name" attribute to "btnItem" and set the Content attribute to use the binding expression "{Binding Title, Mode=OneTime }".

```
<Grid x:Name="LayoutRoot">
  <telerik:RadToolBarTray
      telerik:StyleManager.Theme="Vista">
    <telerik:RadToolBar x:Name="tbMain">
      <telerik:RadToolBar.ItemTemplate>
        <DataTemplate >
          <Button x:Name="btnItem"
              Content="{Binding Title, Mode=OneTime }" />
        </DataTemplate>
      </telerik:RadToolBar.ItemTemplate>
    </telerik:RadToolBar>
  </telerik:RadToolBarTray>
</Grid>
```

6) Verify that the followin XML namespace for the **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Navigation** exists in the UserControl element. Add it if it does not exist. Also, add a "Loaded" event handler to the UserControl element.

```
<UserControl
. . .
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .
Loaded="UserControl_Loaded">. . .
```

Code Behind

1) Navigate to the code for the UserControl Loaded event handler. Add the code below to create and populate a generic List of Tool objects.

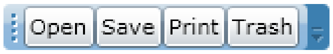
```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
 ' Assign a list of "Tool" objects to the tool bar ItemsSource
 Dim tools As List(Of Tool) = New List(Of Tool) (New Tool() { _
New Tool("Open", "images/Folder_Open.png", "Open a file for editing"), _
New Tool("Save", "images/3.5_Disk_Drive.png", "Save the current file"), _
New Tool("Print", "images/Printer.png", "Print the current file"), _
New Tool("Trash", "images/RecycleBin.png", "Discard the current file")})
  tbMain.ItemsSource = tools
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   // Assign a list of "Tool" objects to the tool bar ItemsSource
   List<Tool> tools = new List<Tool>() {
      new Tool("Open", "Images/Folder_Open.png", "Open a file for editing"),
      new Tool("Save", "Images/3.5_Disk_Drive.png", "Save the current file"),
      new Tool("Print", "Images/Printer.png", "Print the current file"),
      new Tool("Trash", "Images/RecycleBin.png", "Discard the current file")
   };
   tbMain.ItemsSource = tools;
}
```

Test and Refine

1) Press **F5** to run the application in its current state.

A tool bar item is created for every Tool object in the list. Inside each item is a Button object with the Content property bound to the Tool.Title property.



2) Now that you've implemented binding in its simplest form, replace the Button XAML with the new markup below.

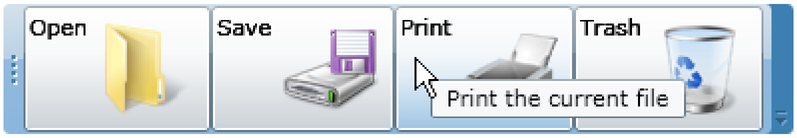
Notes

Notice that the new XAML for the Button adds a number of features:

- A ToolTip is added to the Button. The ToolTip is bound to the "Description" property of the Tool object. The Mode attribute is set to "OneTime": we need to initially load the item from the data, but we don't need to interact with the data after that.
- A Grid with two columns is defined that will layout the elements for the item.
- Two TextBlock controls are added to the Grid that are bound to the "Title" property of the Tool object. The second TextBlock will display the text as a shadow. Styling will be assigned to the TextBlock later to get the shadow effect.
- In Image is added to the second column of the Grid and bound to the "Path" property of the Tool object.

```
<Button
    ToolTipService.ToolTip="{Binding Description, Mode=OneTime }">
  <Grid ShowGridLines="False">
    <Grid.ColumnDefinitions>
      <ColumnDefinition></ColumnDefinition>
      <ColumnDefinition></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <TextBlock
        Text="{Binding Title, Mode=OneTime }"
        Grid.Column="0">
    </TextBlock>
    <TextBlock
        Text="{Binding Title, Mode=OneTime }"
        Grid.Column="0">
    </TextBlock>
    <Image
        Source="{Binding Path, Mode=OneTime }"
        Grid.Column="1" Margin="5"></Image>
  </Grid>
</Button>
```

3) Press **F5** to run the application again. The tool bar should look something like the screenshot below. Verify that the ToolTip shows when you hover the mouse over any of the buttons.

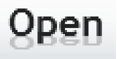


4) Now we can style the text to give a "shadow" effect using the image "reflection" technique explained in the "Expander" chapter. Add a UserControl.Resources element inside the UserControl element. Add the two styles shown below.

```
<UserControl.Resources>

  <Style x:Key="ButtonTextStyle" TargetType="TextBlock">
    <Setter Property="FontFamily" Value="Trebuchet MS" />
    <Setter Property="Margin" Value="10" />
    <Setter Property="FontSize" Value="20" />
    <Setter Property="Foreground" Value="#FF222222" />
    <Setter Property="FontWeight" Value="Bold" />
  </Style>

  <Style x:Key="ButtonTextShadowStyle" TargetType="TextBlock"
      BasedOn="{StaticResource ButtonTextStyle}">
    <Setter Property="Foreground" Value="#AAAAAAAA" />
    <Setter Property="RenderTransformOrigin" Value="0,0.5" />
    <Setter Property="RenderTransform">
      <Setter.Value>
        <TransformGroup>
          <ScaleTransform ScaleY="-.5"></ScaleTransform>
        </TransformGroup>
      </Setter.Value>
    </Setter>
  </Style>

</UserControl.Resources>
```

Notes

The styles defined above produce a shadow or reflection effect like this small sample below where the dark text facing straight forward is styled using "ButtonTextStyle" and the "reflection" is styled using "ButtonTextShadowStyle".



The "ButtonTextStyle" style simply assigns the basic properties for the title TextBlock. "ButtonTextShadowStyle" uses the "BasedOn" attribute and points back to the "ButtonTextStyle". The "BasedOn" attribute will let us use all the properties of the first style and allows us to add or modify additional properties. "ButtonTextShadowStyle" uses the technique described in the "Expander" chapter to display a "reflection" or "shadow" effect. This style makes the text color, i.e. "Foreground", a partially transparent gray. The RenderTransformOrigin is pushed down slightly by setting the Y value to "0.5". A RenderTransform sets ScaleY as "-.5" to flip and compress the text.

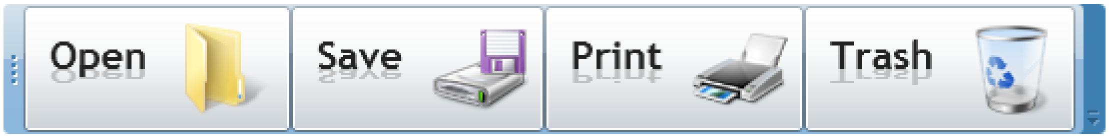
Notice that the Setter for the RenderTransform property uses a TransformGroup. This means that you can add other transforms inside the group. For example you could add a SkewTransform to slightly slant the shadow, as if the light source for the shadow was coming from an angle.

5) Bind the Style of the two TextBlock controls to "ButtonTextShadowStyle" and "ButtonTextStyle" respectively.

Notice that the first text block is styled to be the text "shadow".

```
<TextBlock . . .
      Style="{StaticResource ButtonTextShadowStyle}">
</TextBlock>
<TextBlock . . .
      Style="{StaticResource ButtonTextStyle}" >
</TextBlock>
```

6) Press **F5** to run the application again. Now the text has a slight shadow out in front of it.



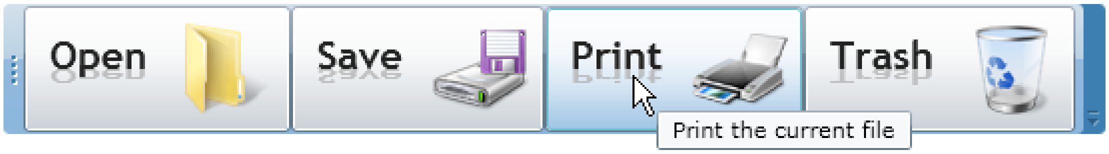
7) Now we will "refactor" the XAML and make the entire ItemTemplate into a resource. Locate the "DataTemplate" element inside the RadToolBar. Cut the entire element and paste it inside the UserControl.Resources element. Add a "x:Key" attribute to the DataTemplate element and set it to be "ToolBarTemplate".

```
<UserControl.Resources>
. . .
  <DataTemplate x:Key="ToolBarTemplate">
      <Button
          ToolTipService.ToolTip="{Binding Description, Mode=OneTime }">
        <Grid ShowGridLines="False">
          <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
          </Grid.ColumnDefinitions>
          <TextBlock
              Text="{Binding Title, Mode=OneTime }"
              Grid.Column="0"
              Style="{StaticResource ButtonTextShadowStyle}">
          </TextBlock>
          <TextBlock
              Text="{Binding Title, Mode=OneTime }"
              Grid.Column="0"
              Style="{StaticResource ButtonTextStyle}">
          </TextBlock>
          <Image
              Source="{Binding Path, Mode=OneTime }"
              Grid.Column="1" Margin="5"></Image>
        </Grid>
      </Button>
    </DataTemplate>

</UserControl.Resources>
```

8) Remove the "ItemTemplate" tags from inside the RadToolBar element. Add a ItemTemplate attribute to the RadToolBar and assign it the binding expression "{StaticResource ToolBarTemplate". The RadToolBar should now look like the example below.

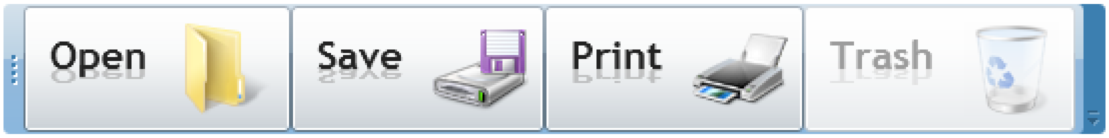
```
<Grid x:Name="LayoutRoot">
  <telerik:RadToolBarTray
      telerik:StyleManager.Theme="Vista">
    <telerik:RadToolBar
      x:Name="tbMain"
      ItemTemplate="{StaticResource ToolBarTemplate}" />
  </telerik:RadToolBarTray>
</Grid>
```

9) Press **F5** to run the application and verify that resourcing the ItemTemplate hasn't changed the functionality.



Ideas for Extending This Example

- Include additional transformations in the TransformGroup element of the ButtonTextShadowStyle style. Try adding a SkewTransform element and set the AngleX and AngleY attributes.
- Add more items to the template.
- Try adding other properties to the Tool object and bind to properties in the item template. For example, how about an "Enabled" property bound to the IsEnabled property of the button? The screenshot below shows the "Trash" icon disabled.



12.6 Customization

Walk Through

In this example we will customize the RadToolBar control to have a "Scoville" style. We will also customize the RadToolBarSeparator to show a gradient block of color.

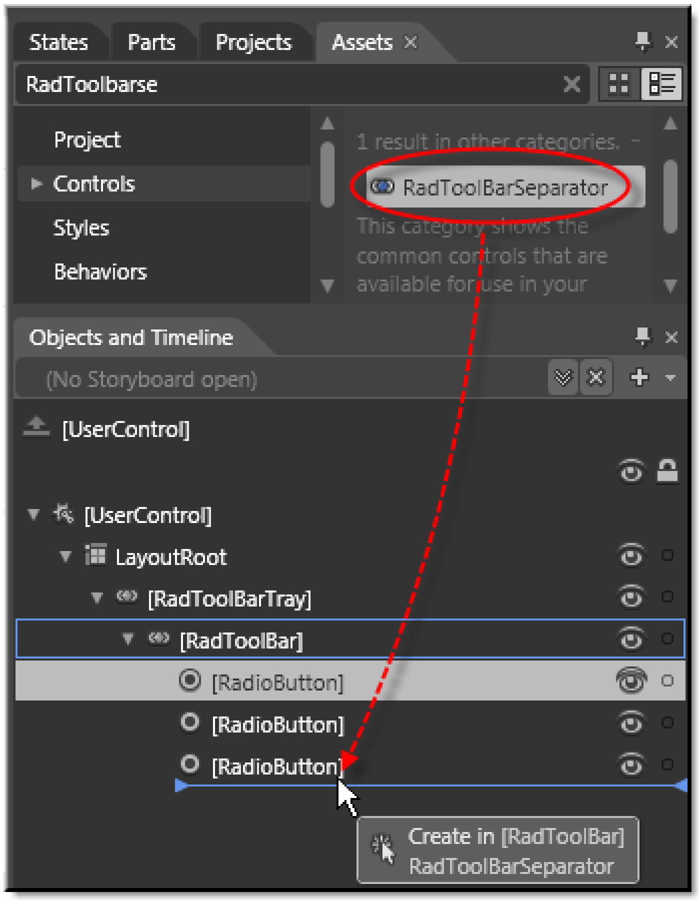
"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

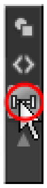
1) Run Expression Blend.
2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*
3) In the New Project dialog, select "Silverlight" from "Project types" and Silverlight 3 Application from the right-most list. Enter a unique name for the project and click **OK**.
4) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the project.
5) In the Projects pane, right-click the **References** node and select **Add Reference...** from the context menu. Add a references to the **Telerik.Windows.Controls and Telerik.Windows.Controls. Navigation** assemblies.
6) From the **Project** menu select **Build Project**.
7) Open the Assets pane. On the left side of the Assets pane is a tree view. Locate and select the "Controls" node. In the Assets pane, just above the tree view is the Assets Find entry text box. Type the first few characters of **RadToolBarTray** into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view. Locate the **RadToolBarTray** control and drag it onto the MainPage.xaml Artboard.
8) Drag a **RadToolBar** control from the Assets pane to the Objects and Timeline pane, just under the RadToolBarTray. The tool tip should read "Create in [RadToolBarTray]".
9) Drag three **RadioButton** controls from the Assets pane to the Objects and Timeline pane, just under the RadToolBar. The RadioButton controls should be children of the RadToolBar, i.e. the tool tip should read "Create in [RadToolBar]".

10)Drag a **RadToolBarSeparator** control from the Assets pane to the Objects and Timeline pane, just under the three RadioButton controls, as shown in the screenshot below. The RadToolBarSeparator should also be a child of the RadToolBar and a sibling of the three RadioButton controls.

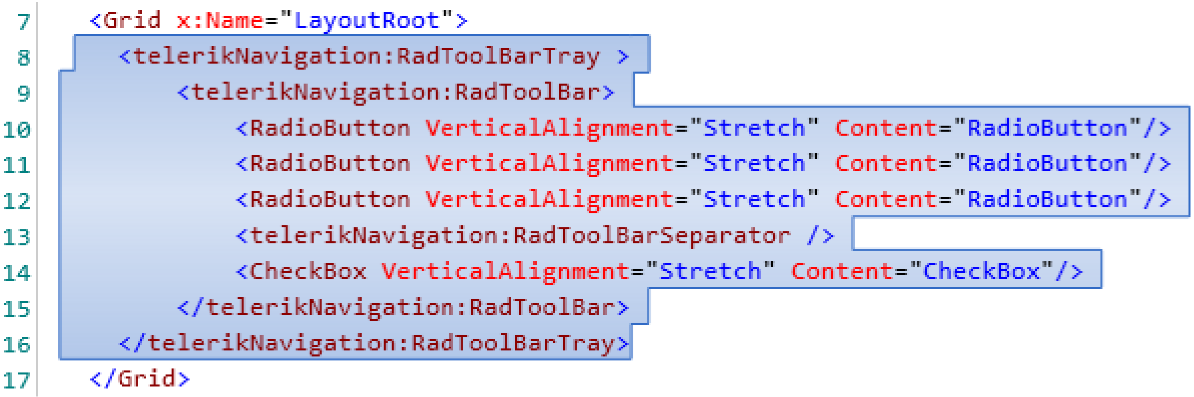


11)Drag a **CheckBox** control from the Assets pane to the Objects and Timeline pane, just under the RadToolBarSeparator. The CheckBox should be a sibling of the RadioButton and RadToolBarSeparator controls.

12)Find the Split button in the upper right-hand corner of the Artboard and click it to view both the Design and XAML at the same time.



13)If Expression Blend has automatically included any dimensions for any of the controls, remove those attributes manually. The XAML markup should look like the example below:

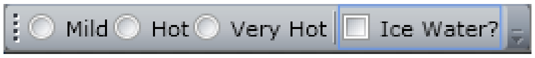


```
<Grid x:Name="LayoutRoot">
    <telerikNavigation:RadToolBarTray >
        <telerikNavigation:RadToolBar>
            <RadioButton VerticalAlignment="Stretch" Content="RadioButton"/>
            <RadioButton VerticalAlignment="Stretch" Content="RadioButton"/>
            <RadioButton VerticalAlignment="Stretch" Content="RadioButton"/>
            <telerikNavigation:RadToolBarSeparator />
            <CheckBox VerticalAlignment="Stretch" Content="CheckBox"/>
        </telerikNavigation:RadToolBar>
    </telerikNavigation:RadToolBarTray>
</Grid>
```

14)Click each of the RadioButton controls in the Objects and Timeline pane. In the **Properties pane** set **Common Properties > Content** to "Mild", "Hot", "Very Hot", respectively.

15)Click the CheckBox control in the Objects and Timeline pane. In the **Properties pane** set **Common Properties > Content** to "Ice Water?".

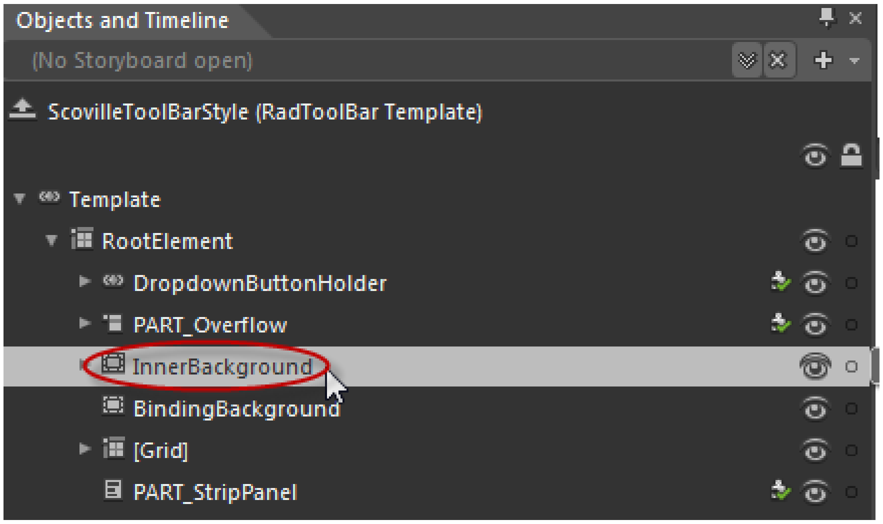
16)In the design view of the Artboard, the tool bar should look like the screenshot below.



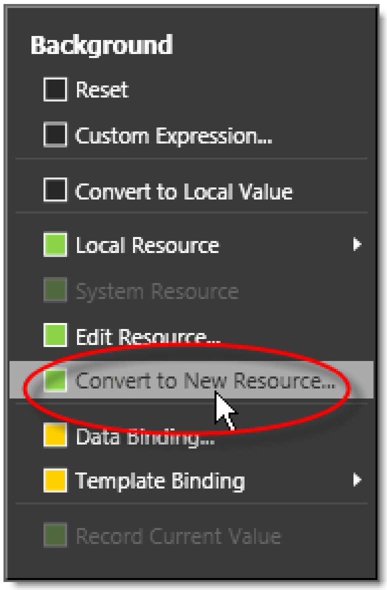
Customize Templates and Styles

1) Right-click the RadToolBar and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleToolBarStyle". Click **OK** to create the style resource and close the dialog.

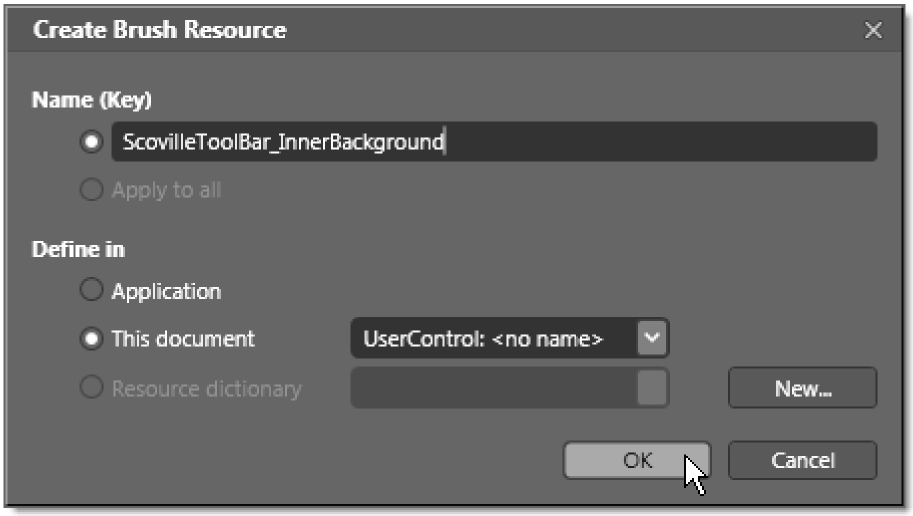
2) In the Objects and Timeline pane, select the element named "InnerBackground".



3) In the Properties pane, locate the **Brushes > Background** property. Click the Advanced Property Options button and select **Convert to New Resource...** from the drop down menu.



4) Enter the new name "ScovilleToolBar_InnerBackground". Click **OK** to close the dialog and create the brush resource.



Create Brush Resource

Name (Key)

● ScovilleToolBar_InnerBackground

○ Apply to all

Define in

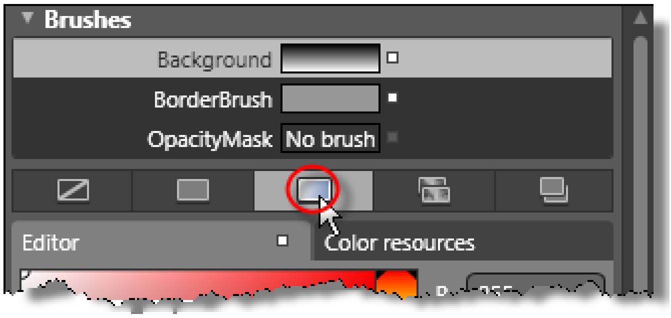
○ Application

● This document UserControl: <no name>

○ Resource dictionary

5) Locate the Properties pane, **Brushes > Background** property. Click the Advanced Property Options button and select "ScovilleToolBar_InnerBackground" from the **Local Resource** item of the drop down menu.

6) In the Properties pane, select the Gradient Brush to use for the background.



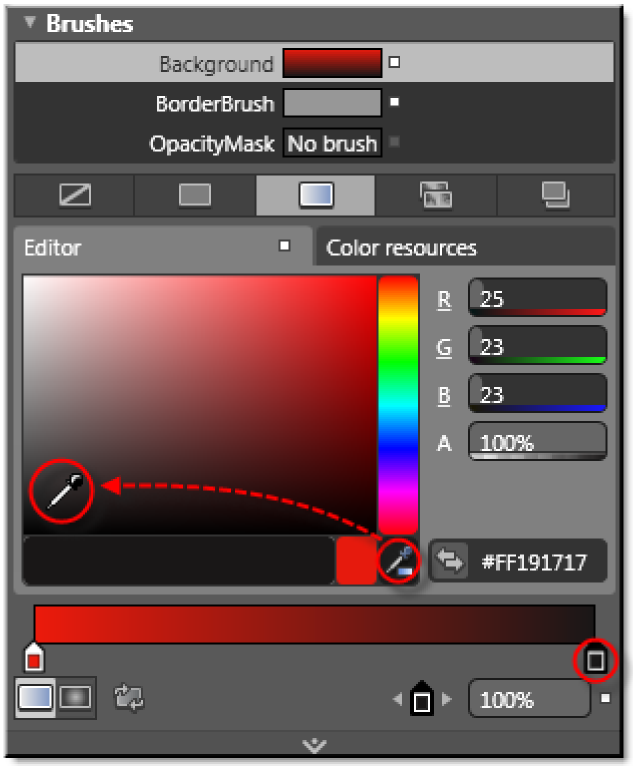
Brushes

Background

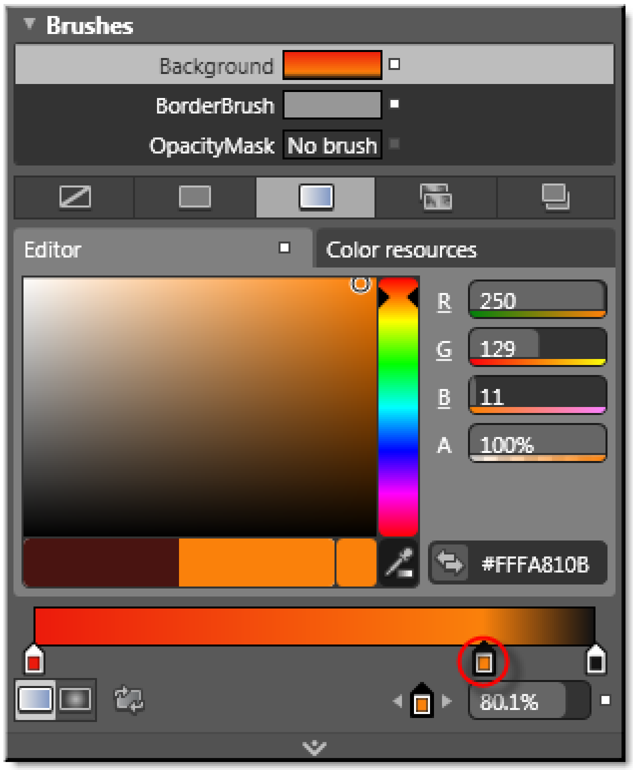
BorderBrush

OpacityMask No brush

7) Click the left-most gradient stop indicator, then drag the eye dropper tool to a red color. Click the right-most gradient stop indicator, then drag the eye dropper tool to a black color. The gradient should look like the red to black gradient shown in the screenshot below.



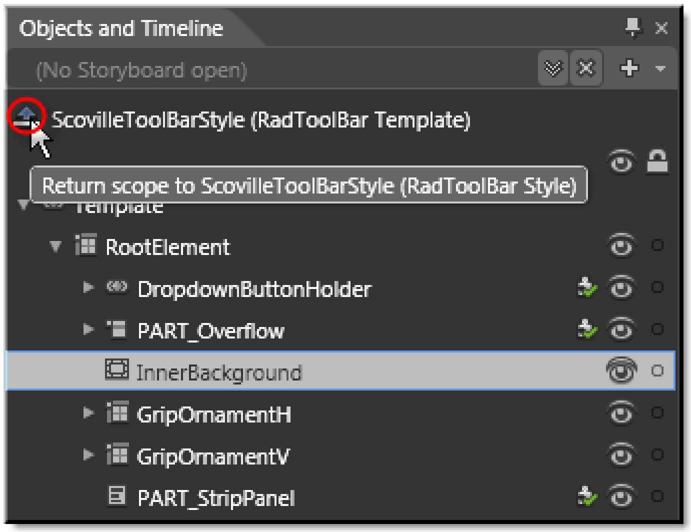
8) Click the Gradient Bar somewhere in the middle to create a new gradient stop. Set the new gradient stop indicator to an orange color.



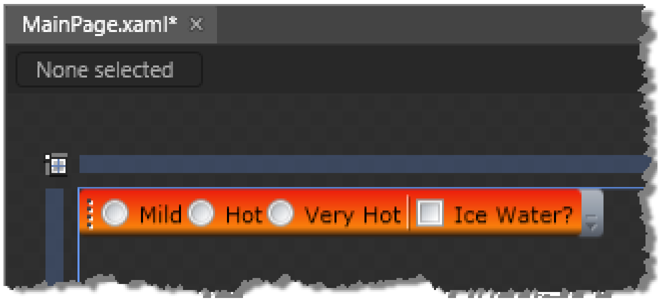
Background
BorderBrush
OpacityMask No brush

R 250
G 129
B 11
A 100%
#FFFA810B
80.1%

9) Click the Return Scope button until you return to editing the tool bar.

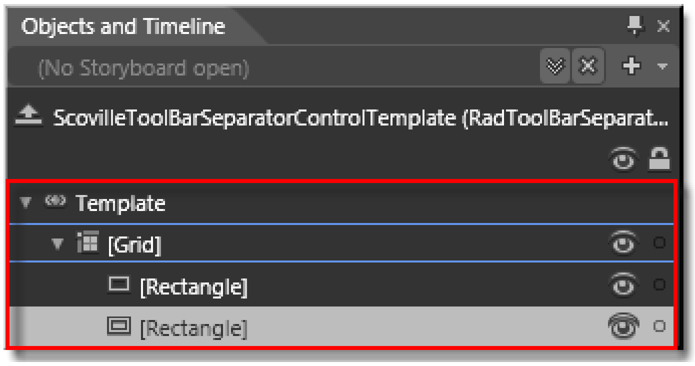


10)The tool bar should look something like the screenshot below when viewed in the Artboard:



11)In the Objects and Timeline pane, right-click the RadToolBarSeparator and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleToolBarSeparatorControlTemplate". Click **OK** to create the style resource and close the dialog.

The new template for the separator appears in the screenshot below. The template contains a grid with two rectangles, one colored black, the other white, to create a contrasting edge.



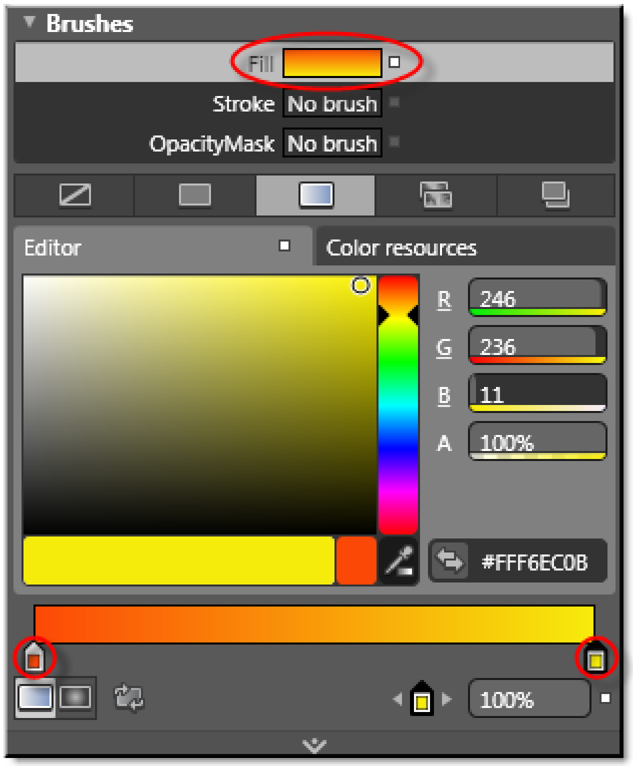
12)Select the "[Grid]" item from the Objects and Timeline pane.

13)In the Properties pane, set the **Layout > MinWidth** property to "5".

14)Select the second "[Rectangle]" item under the Grid from the Objects and Timeline pane.

15)In the Properties pane, locate the **Brushes > Fill** property. Click the Advanced Property Options button and click the **Reset** option from the drop down menu.

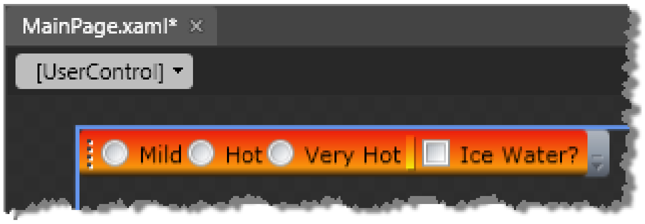
16)Select the Gradient Brush to use for the new fill. Use the two gradient stop indicators and the eye dropper tool to create a orange-to-yellow gradient as shown in the screenshot below.



Fill
Stroke No brush
OpacityMask No brush
R 246
G 236
B 11
A 100%
#FFF6EC0B
100%

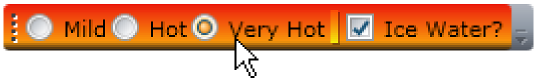
17)Click the Return Scope button until you return to editing the tool bar.

18)The tool bar should look something like the screenshot below when viewed in the Artboard:



Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Ideas for Extending This Example

- Try styling the other parts of the tool bar including the "Grip Ornament", the overflow area and the drop down button.
- Add animation to react to the tool bar's state. See the "Menu Controls" chapter, Customization section for an example of how this can be done.
- Style the tool bar for a consistent look in both horizontal and vertical orientations.

12.7 Wrap Up

In this chapter you learned how to organize multiple Silverlight controls into a horizontal or vertical strip using RadToolBar. You used RadToolBarSeparator to visually divide groups of controls. You saw how the theming mechanism for RadToolBar automatically styles both the tool bar and its items. You also saw how the RadToolBar OverflowMode property handles overflow behavior when there are more controls than can be displayed at one time. You handled events that reacted to the overflow area expanding and collapsing. You added items to the tool bar in XAML, using the programmatic API and through data binding.

You used the RadToolBarTray to manage RadToolBar position, sizing and order. You used the Band property to place each tool bar in a specific row and the BandIndex property to control each tool bar's position within a band.

Finally, you created custom templates for the RadToolBar background and RadToolBarSeparator. You created matching color schemes for both custom templates.

Part XIII

Expander

13 Expander

13.1 Objectives

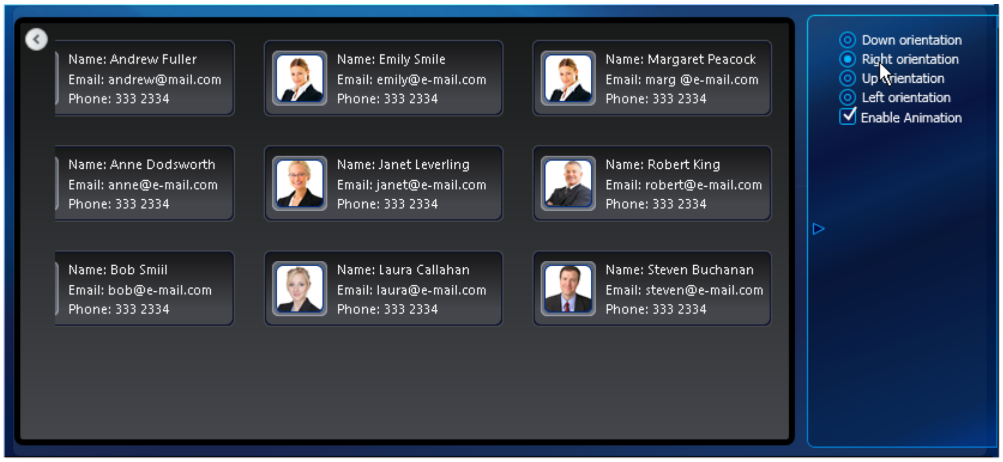
This chapter demonstrates using RadExpander to save space and to present Silverlight content as needed. You will learn how to place simple text material in the Header and Content. You will see which properties control expansion direction and animation. Then you will use the Header and Content sub-elements to present multiple Silverlight items of any complexity and arrangement. Along the way you will use control templates to display images, image reflections and custom buttons. Finally, you will use Expression Blend to replace the standard "arrow" graphic with an image that rotates when clicked.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Expander\Expander.sln.

13.2 Overview

RadExpander is a flexible, lightweight control that saves page real estate and aids site navigation. You can place the expander anywhere on the page and fill it with any content. The expander lets you control the expand animation and direction of expansion. Your end user can press the Space key to toggle the expansion without having to use the mouse.



Name: Andrew Fuller
Email: andrew@mail.com
Phone: 333 2334

Name: Emily Smile
Email: emily@e-mail.com
Phone: 333 2334

Name: Margaret Peacock
Email: marg @e-mail.com
Phone: 333 2334

Name: Anne Dodsworth
Email: anne@e-mail.com
Phone: 333 2334

Name: Janet Leverling
Email: janet@e-mail.com
Phone: 333 2334

Name: Robert King
Email: robert@e-mail.com
Phone: 333 2334

Name: Bob Smiil
Email: bob@e-mail.com
Phone: 333 2334

Name: Laura Callahan
Email: laura@e-mail.com
Phone: 333 2334

Name: Steven Buchanan
Email: steven@e-mail.com
Phone: 333 2334

○ Down orientation
● Right orientation
○ Up orientation
○ Left orientation
☑ Enable Animation

13.3 Getting Started

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Input**

c) **Telerik.Windows.Themes.Vista**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add an XML namespace reference to **Telerik** assemblies as follows:

```
<UserControl
xmlns:telrik="http://schemas.telerik.com/2008/xaml/presentation" ...>
```

3) Add the UserControl.Resources XAML below to the UserControl element.

```
<UserControl.Resources>
  <Style x:Key="CalendarStyle"
    TargetType="telerik:RadCalendar">
    <Setter Property="telerik:StyleManager.Theme" Value="Vista" />
    <Setter Property="MaxWidth" Value="250" />
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="10" />
  </Style>
</UserControl.Resources>
```

4) Drag a **RadExpander** control from the Toolbox to a point between the <Grid> and </Grid> tags.

5) Set the RadExpander **Header** property to "Set Backup Dates" and the **ExpandDirection** to "Down".

6) Inside the RadExpander element, add a RadExpander.Content tag. The XAML should look the the example below at this point.

```
<Grid x:Name="LayoutRoot">
  <telerik:RadExpander ExpandDirection="Down" Header="Set Backup Dates">
    <telerik:RadExpander.Content>

    </telerik:RadExpander.Content>
  </telerik:RadExpander>
</Grid>
```

7) Inside the RadExpander.Content, add a StackPanel element.

8) Inside the StackPanel element, add a TextBlock and set the Text property to "Start Date:" and Margin property to "10".

9) From the Toolbox, drag a RadCalendar control to a point just under the TextBlock. Set the Style property to "{StaticResource CalendarStyle}"

10)Inside the StackPanel element, add a TextBlock and set the Text property to "End Date:" and Margin property to "10".

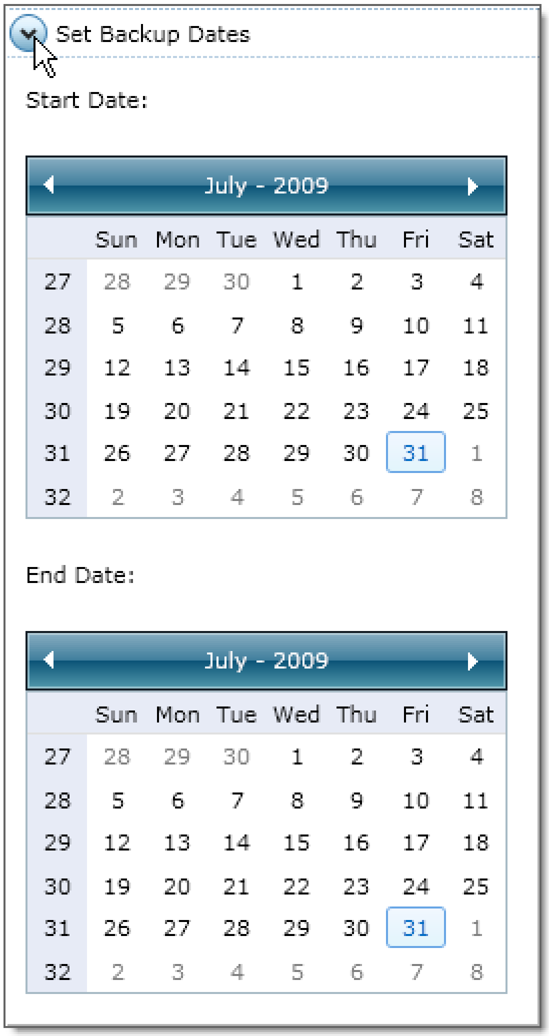
11)From the Toolbox, drag a RadCalendar control to a point just under the TextBlock. Set the Style property to "{StaticResource CalendarStyle}"

The XAML inside the UserControl element should look something like the example below:

```
<UserControl.Resources>
  <Style x:Key="CalendarStyle"
    TargetType="telerikInput:RadCalendar">
    <Setter Property="telerik:StyleManager.Theme" Value="Vista" />
    <Setter Property="MaxWidth" Value="250" />
    <Setter Property="HorizontalAlignment" Value="Left" />
    <Setter Property="Margin" Value="10" />
  </Style>
</UserControl.Resources>

<Grid
  x:Name="LayoutRoot">
  <telerik:RadExpander
    ExpandDirection="Down"
    Header="Set Backup Dates"
    telerik:StyleManager.Theme="Vista">
    <telerik:RadExpander.Content>
      <StackPanel>
        <TextBlock Text="Start Date:" Margin="10" />
        <telerik:RadCalendar
          Style="{StaticResource CalendarStyle}" />
        <TextBlock Text="End Date:" Margin="10" />
        <telerik:RadCalendar
          Style="{StaticResource CalendarStyle}" />
      </StackPanel>
    </telerik:RadExpander.Content>
  </telerik:RadExpander>
</Grid>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Set Backup Dates

Start Date:

July - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	1	2	3	4
28	5	6	7	8	9	10	11
29	12	13	14	15	16	17	18
30	19	20	21	22	23	24	25
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8

End Date:

July - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	1	2	3	4
28	5	6	7	8	9	10	11
29	12	13	14	15	16	17	18
30	19	20	21	22	23	24	25
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8

Test Application Features

- Test the expanding/collapsing functionality of the RadExpander button.

Ideas for Extending This Example

- Set the ExpandDirection property to Up, Down or Left and observe the effect.

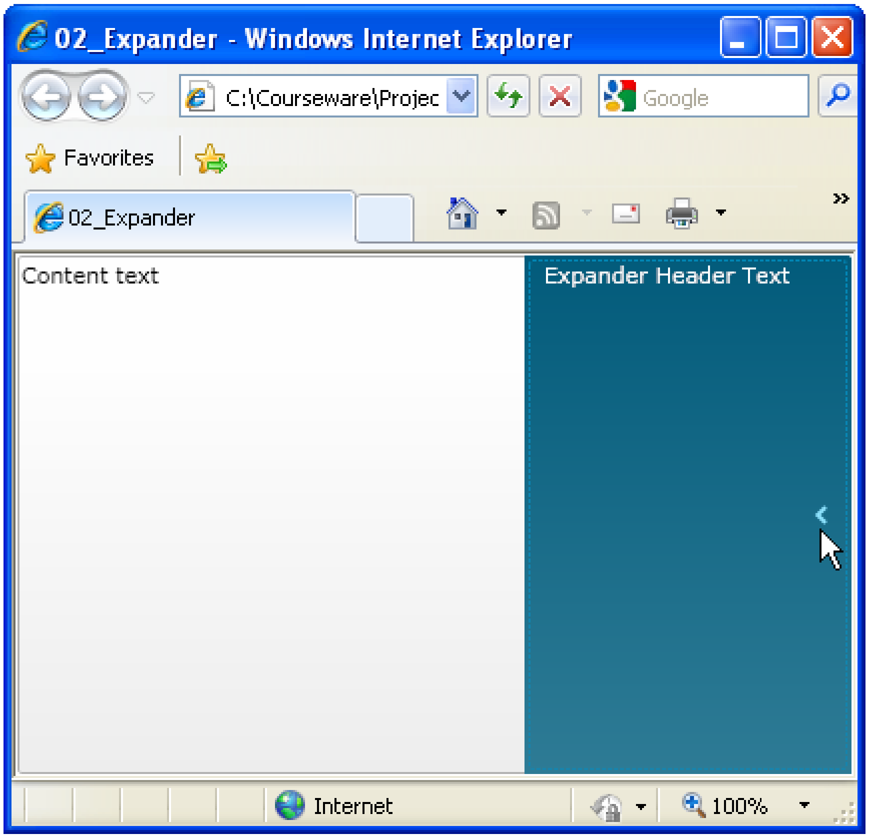
13.4 Control Details

13.4.1 Populating RadExpander

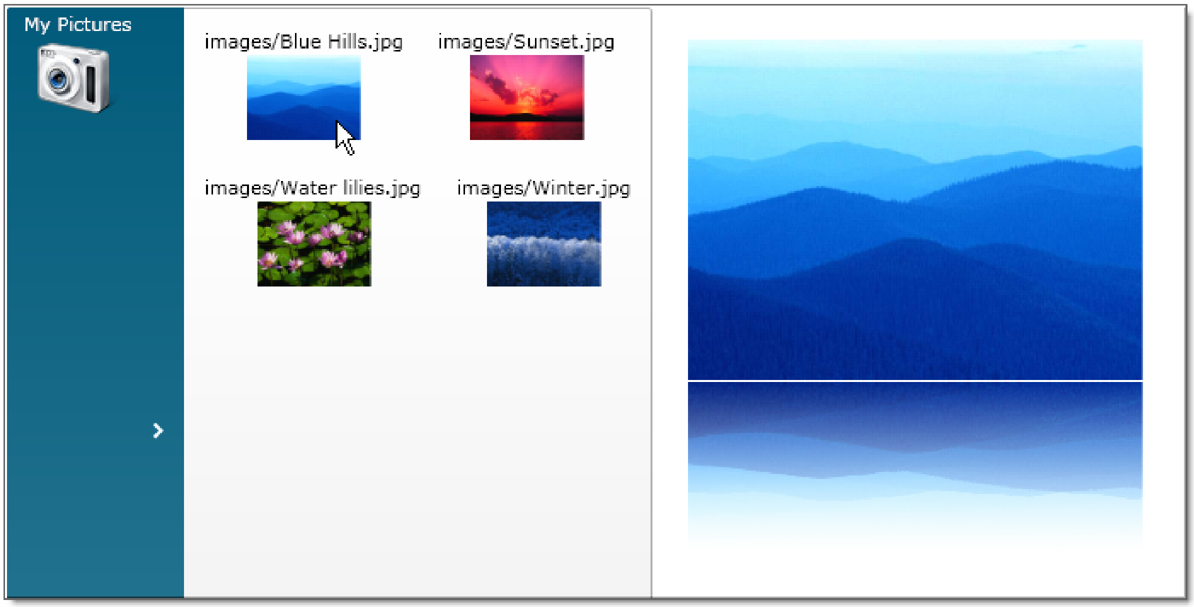
In its simplest form, RadExpander can be assigned Header and Content text. The **ExpandDirection** property by default is "Down", but can also be "Left", "Right" or "Up". You can get a lot of styling mileage simply by defining one of the predefined themes. The example below adds text to the Header and Content. The ExpandDirection is set to "Left", IsExpanded is true and Theme is "Summer".

```
<telerik:RadExpander
   Header="Expander Header Text"
   Content="Content text"
   ExpandDirection="Left"
   IsExpanded="True"
   telerik:StyleManager.Theme="Summer" >
</telerik:RadExpander>
```

Running in the browser, the RadExpander looks like the screenshot below.



To create more complex arrangements of Silverlight controls in the header or content, Header and Content sub-elements will let you pack as much XAML markup as you care to have. For example, the screenshot below shows an expander where the header contains a TextBlock and Image. The content area to the right of the header contains a RadWrapPanel with a series of TextBlock and Image controls. When one of the images is clicked, a larger version of the clicked image and the image reflection appears to the right of the RadExpander.



First lets look at the general layout of the page before breaking each part down for a closer look. The abbreviated XAML below shows styles defined that will be applied to Button, Image and ContentControl. The main "LayoutRoot" contains a StackPanel with Horizontal orientation. Inside the StackPanel is the RadExpander that will contain the thumbnail size image controls and the ContentControl will contain the full size image.

```
<UserControl . . .">
  <UserControl.Resources>
    <Style x:Key="ImageButtonStyle"  TargetType="Button">. . .</Style>
    <Style x:Key="ImageReflectionStyle" TargetType="Image">. . .</Style>
    <Style x:Key="ReflectedImagePanel" TargetType="ContentControl">. . .</Style>
  </UserControl.Resources>

  <Grid x:Name="LayoutRoot">
    <StackPanel Orientation="Horizontal">
      <telerik:RadExpander>. . .</telerik:RadExpander>
      <ContentControl>. . .</ContentControl>
    </StackPanel>
  </Grid>
</UserControl>
```

Notes

Notice the order that the styles are declared in. "ImageReflectionStyle" must be declared first as it is referenced in "ReflectedImagePanel".

First, let's take a look at the layout of the page, then we will talk about the styles used in this page. In the XAML below, the RadExpander has two main parts, Header and Content. The RadExpander.Header contains a StackPanel that in turn holds a TextBlock and an Image. The RadExpander.Content uses a RadWrapPanel as a container. In the RadWrapPanel are four buttons that hide much of their complexity in a ControlTemplate stored as a resource called "ImageButtonStyle". At the bottom of this XAML is the ContentControl that uses a custom style "ReflectedImagePanel" to get an image "reflection" effect.

```
<Grid x:Name="LayoutRoot">
   <StackPanel Orientation="Horizontal">
      <telerik:RadExpander x:Name="exMain" ExpandDirection="Right"
         telerik:StyleManager.Theme="Summer">
         <telerik:RadExpander.Header>
            <StackPanel>
               <TextBlock>My Pictures</TextBlock>
               <Image Source="Images/Camera.png" />
            </StackPanel>
         </telerik:RadExpander.Header>
         <telerik:RadExpander.Content>
            <telerik:RadWrapPanel MaxWidth="400">
               <Button
                  Style="{StaticResource ImageButtonStyle}"
                  Content="Images/Blue Hills.jpg" Click="Button_Click" />
               <Button
                  Style="{StaticResource ImageButtonStyle}"
                  Content="Images/Sunset.jpg" Click="Button_Click" />
               <Button
                  Style="{StaticResource ImageButtonStyle}"
                  Content="Images/Water lilies.jpg" Click="Button_Click" />
               <Button
                  Style="{StaticResource ImageButtonStyle}"
                  Content="Images/Winter.jpg" Click="Button_Click" />
            </telerik:RadWrapPanel>
         </telerik:RadExpander.Content>
      </telerik:RadExpander>

      <ContentControl x:Name="imgPanel" Margin="20"
         Style="{StaticResource ReflectedImagePanel}">
      </ContentControl>

   </StackPanel>
</Grid>
```

The Style used on the Button controls is called "ImageButtonStyle". It sets up a ControlTemplate to completely replace the contents of a Button. Inside the template is a StackPanel. The StackPanel contains a ContentPresenter that displays an image path and an Image control that displays the image pointed to by the path. Each button has the same "Click" event handler assigned.

```
<Style x:Key="ImageButtonStyle" TargetType="Button">
   <Setter Property="Template">
      <Setter.Value>
         <ControlTemplate TargetType="Button">
            <StackPanel Margin="10">
               <ContentPresenter Content="{TemplateBinding Content}" />
               <Image MaxHeight="50" Source="{TemplateBinding Content}" />
            </StackPanel>
         </ControlTemplate>
      </Setter.Value>
   </Setter>
</Style>
```

The XAML used for the reflected image is a little more complicated and requires two styles. One style creates an image "reflection" and the second is a ControlTemplate used to assemble the original image and the reflected image together.

First lets see how you can create the reflection effect using a style. The first two properties in "ImageReflectionStyle" are the **RenderTransformOrigin** property that determines the center point of any transformation operation and a **RenderTransform**. Transformations can scale objects to be larger or smaller, move objects to other locations, rotate objects or skew them. In this case we are changing the scale along the "Y" axis to a negative number. This has the effect of flipping the image. If you supply a negative one "-1" you simply flip the image. Negative fractions flip and compress the image at the same time. A ScaleY value of "-.5" is flipped and compressed to half its original size. Finally the **OpacityMask** property is configured to paint the image so that it fades.

```
<Style x:Key="ImageReflectionStyle" TargetType="Image">
  <Setter Property="RenderTransformOrigin" Value="0,1" />
  <Setter Property="RenderTransform">
    <Setter.Value>
      <ScaleTransform ScaleY="-.5" />
    </Setter.Value>
  </Setter>
  <Setter Property="OpacityMask">
    <Setter.Value>
      <LinearGradientBrush
        StartPoint="0.5,0"
        EndPoint="0.5,1">
        <GradientStop Offset="0" Color="Transparent" />
        <GradientStop Offset="1" Color="#FFFFFFFF" />
      </LinearGradientBrush>
    </Setter.Value>
  </Setter>
</Style>
```

The image and the reflected version of the image are assembled into a single ContentControl by way of the "ReflectedImagePanel" style below. The style places both Images into a grid and applies the "ImageReflectionStyle" to the second image.

```
<Style x:Key="ReflectedImagePanel"
  TargetType="ContentControl">
  <Setter
    Property="Template">
    <Setter.Value>
      <ControlTemplate
        TargetType="ContentControl">
        <Grid VerticalAlignment="Top">
          <Image
            Margin="0, 0, 0, 1"
            Source="{TemplateBinding Content}"
            MaxHeight="200" />
          <Image
            Source="{TemplateBinding Content}"
            MaxHeight="200"
            Grid.Row="1"
            Style="{StaticResource ImageReflectionStyle}" />
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Finally, the **Click** event handler for each button simply assigns the image path to the content of the ContentControl.

VB

```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim path As String = (TryCast(sender, Button)).Content.ToString()
  imgPanel.Content = path
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   string path = (sender as Button).Content.ToString();
   imgPanel.Content = path;
}
```

13.4.2 Events

RadExpander has two events you can handle: **Expanded** and **Collapsed**. As with most of the Telerik Silverlight controls, these are routed events and so pass the **RoutedEventArgs** parameter. You can cast this parameter to a **RadRoutedEventArgs** type (from the **Telerik.Windows** namespace) to get additional information. The example below changes the header as the control is expanded and collapsed.

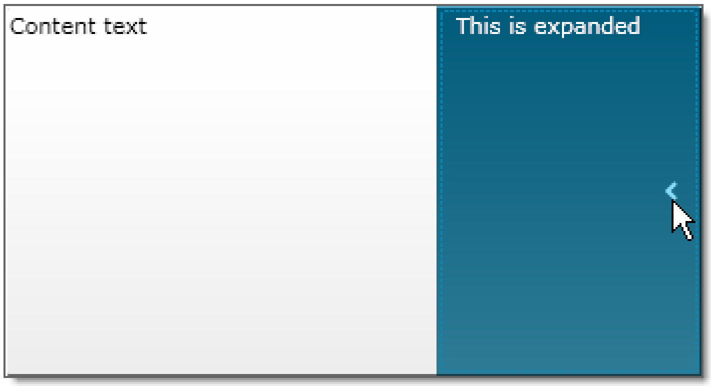
VB

```
Private Sub RadExpander_Expanded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim args As RadRoutedEventArgs = TryCast(e, RadRoutedEventArgs)
  Dim expander As RadExpander = TryCast(args.OriginalSource, RadExpander)
  expander.Header = "This is expanded"
End Sub

Private Sub RadExpander_Collapsed(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim args As RadRoutedEventArgs = TryCast(e, RadRoutedEventArgs)
  Dim expander As RadExpander = TryCast(args.OriginalSource, RadExpander)
  expander.Header = "This is collapsed"
End Sub
```

C#

```
private void RadExpander_Expanded(object sender, RoutedEventArgs e)
{
   RadRoutedEventArgs args = e as RadRoutedEventArgs;
   RadExpander expander = args.OriginalSource as RadExpander;
   expander.Header = "This is expanded";
}

private void RadExpander_Collapsed(object sender, RoutedEventArgs e)
{
   RadRoutedEventArgs args = e as RadRoutedEventArgs;
   RadExpander expander = args.OriginalSource as RadExpander;
   expander.Header = "This is collapsed";
}
```

Running in the browser, the RadExpander looks like the screenshot below.



Content text

This is expanded

13.4.3 Animation

To enable or disable animation in XAML markup, set the **AnimationManager IsAnimationEnabled** attribute:

XAML

```
<telerik:RadExpander
   . . .
   telerik:AnimationManager.IsAnimationEnabled="True"
   >
</telerik:RadExpander>
```

. . .or in code you can call the **AnimationManager.SetIsAnimationEnabled()**, passing the RadExpander instance and a Boolean to turn the animation on or off. You must first add a reference to the **Telerik.Windows.Controls.Animation** name space in your "Imports" (VB) or "using" (C#) section of code.

VB

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  AnimationManager.SetIsAnimationEnabled(Me.expander, True)
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   AnimationManager.SetIsAnimationEnabled(this.expander, true);
}
```

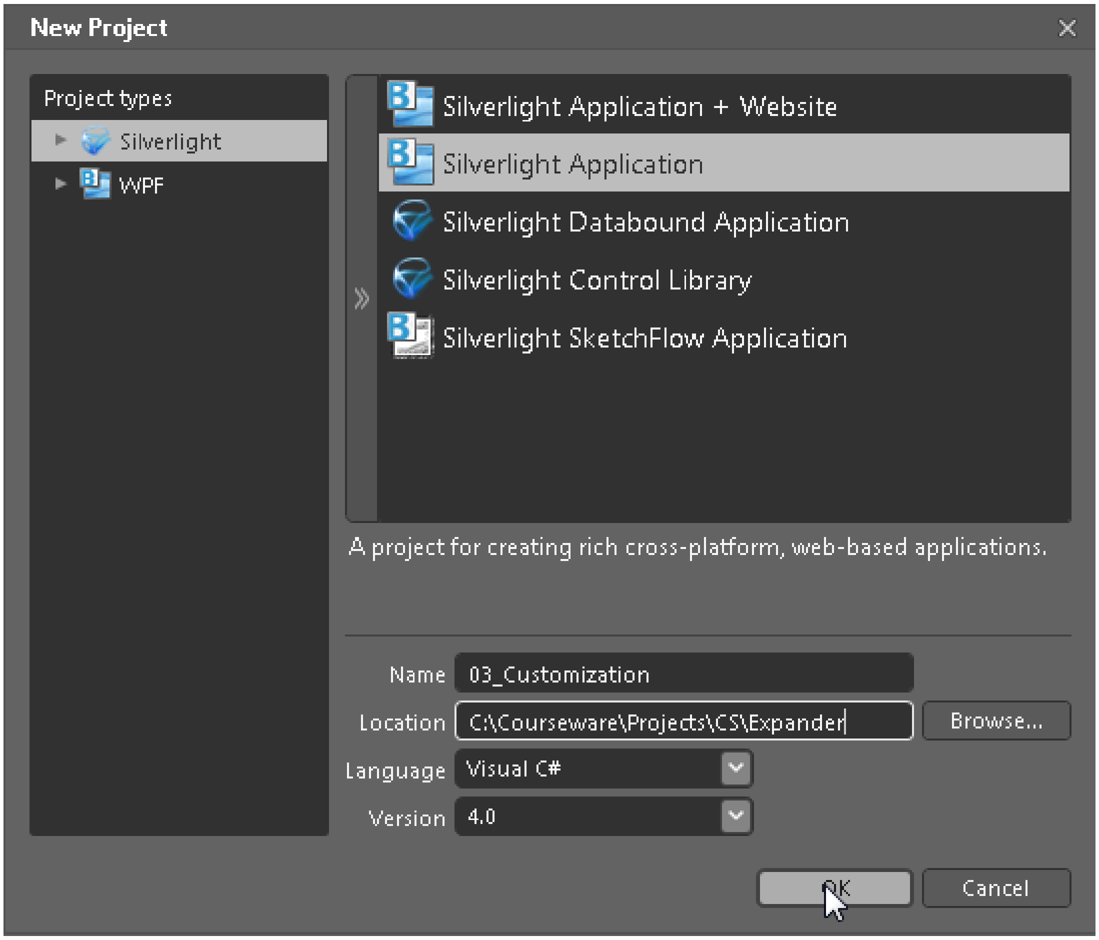
13.5 Customization

Walk Through

In this example we will customize the RadExpander control by swapping out the standard "arrow" button with an image of a pepper. The pepper image will be animated to swivel the pepper stem up and down as the expander is expanded and collapsed.

Project Setup

1) Run Expression Blend.
2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*
3) In the New Project dialog, select "Silverlight" from "Project types" and Silverlight 3 Application from the right-most list. Enter a unique name for the project and click **OK**.


New Project
Project types
Silverlight
WPF
Silverlight Application + Website
Silverlight Application
Silverlight Databound Application
Silverlight Control Library
Silverlight SketchFlow Application
Browse...
OK
Cancel

A project for creating rich cross-platform, web-based applications.

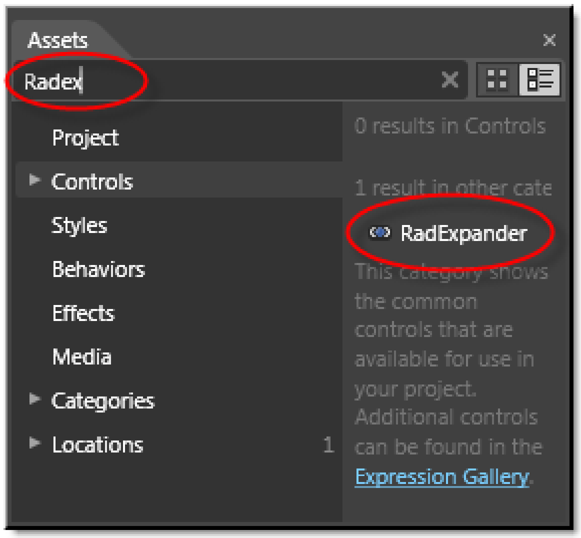
Name 03_Customization
Location C:\Courseware\Projects\CS\Expander
Language Visual C#
Version 4.0

4) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.
5) In the Projects pane, right-click the **References** node and select **Add Reference...** from the context menu.

6) Add a reference to the **Telerik.Windows.Controls.dll** assembly.

7) From the **Project** menu select **Build Project**.

8) Open the Assets pane. On the left side of the Assets pane is a tree view. Locate and select the "Controls" node. In the Assets pane, just above the tree view is the Assets Find entry text box. Type the first few characters of "RadExpander" into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view. Locate the **RadExpander** control and drag it onto the MainPage.xaml Artboard.

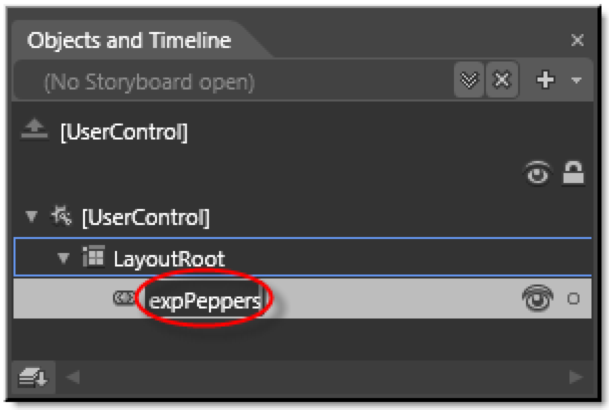


This category shows the common controls that are available for use in your project. Additional controls can be found in the Expression Gallery.

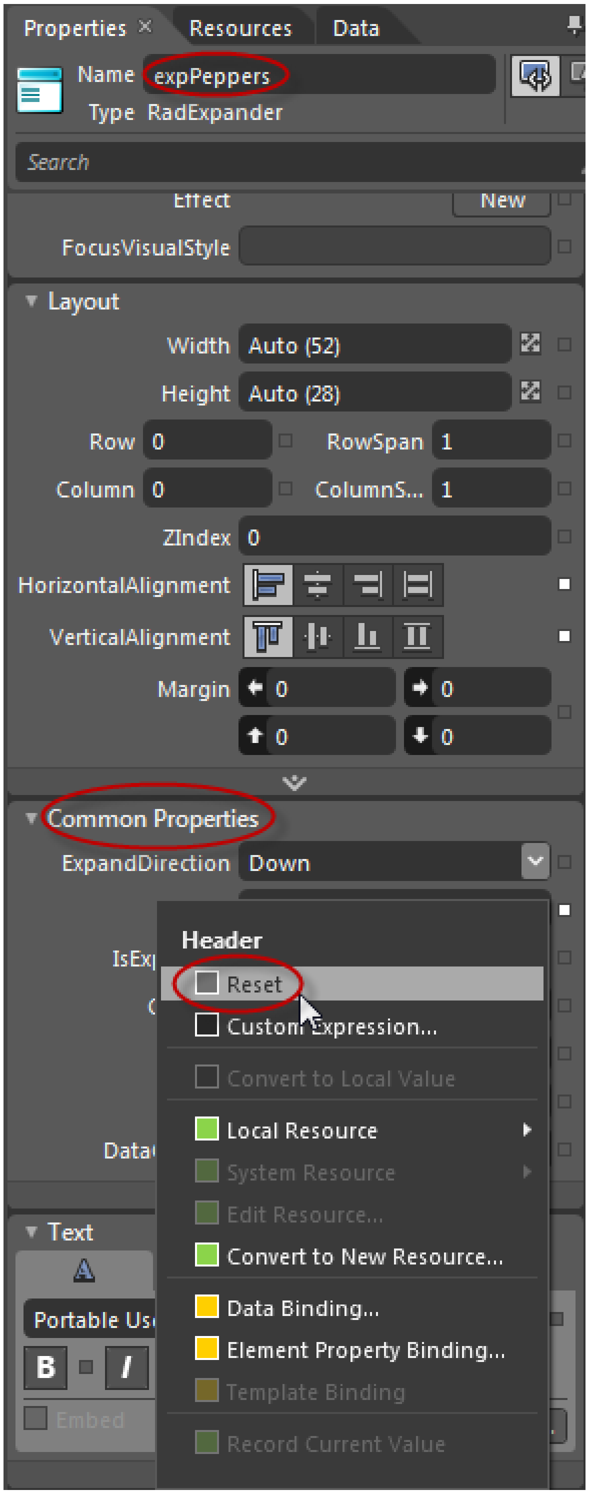
Tip!

If the Assets pane is not visible, select the main menu Windows item, locate "Assets" and click to re-enable it.

9) In the Objects and Timeline pane, double-click "[RadExpander]" in the tree view. Enter a new name "expPeppers".



10) In the Properties pane, locate the **Common Properties > Header** property and set Reset:

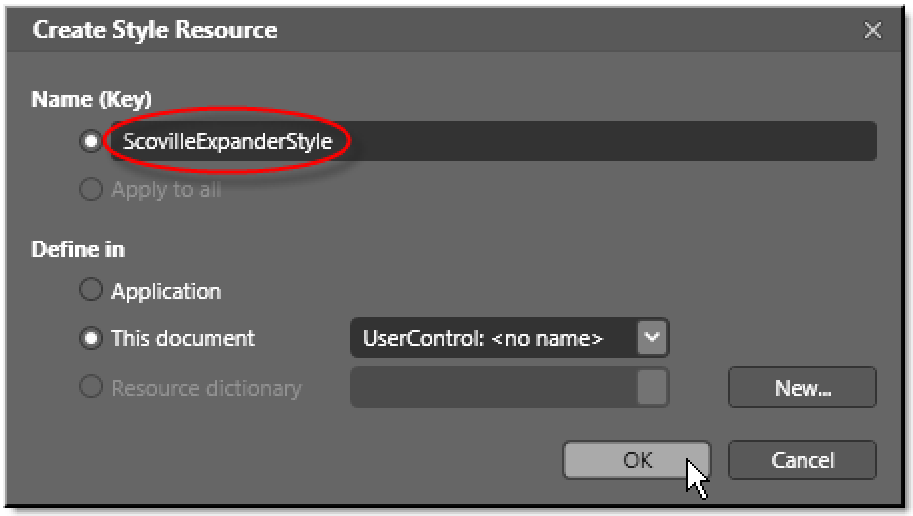


Name expPeppers
Type RadExpander

Width Auto (52)
Height Auto (28)
Row 0 RowSpan 1
Column 0 ColumnS... 1
ZIndex 0
Margin 0 0 0 0

ExpandDirection Down

11)Right-click "expPeppers" and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleExpanderStyle". Click **OK** to create the style resource and close the dialog.



Create Style Resource

Name (Key)

● ScovilleExpanderStyle

○ Apply to all

Define in

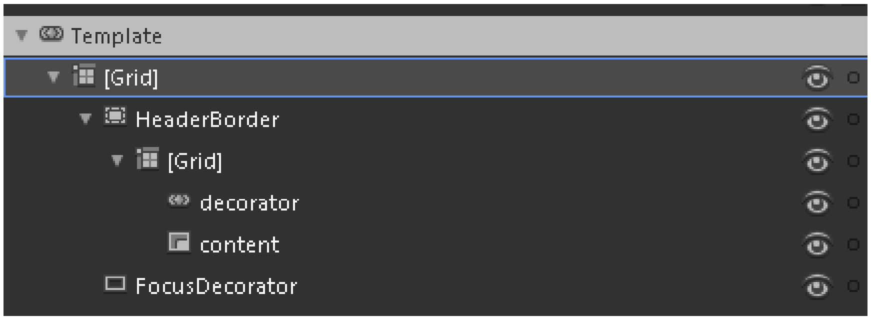
○ Application

● This document UserControl: <no name>

○ Resource dictionary

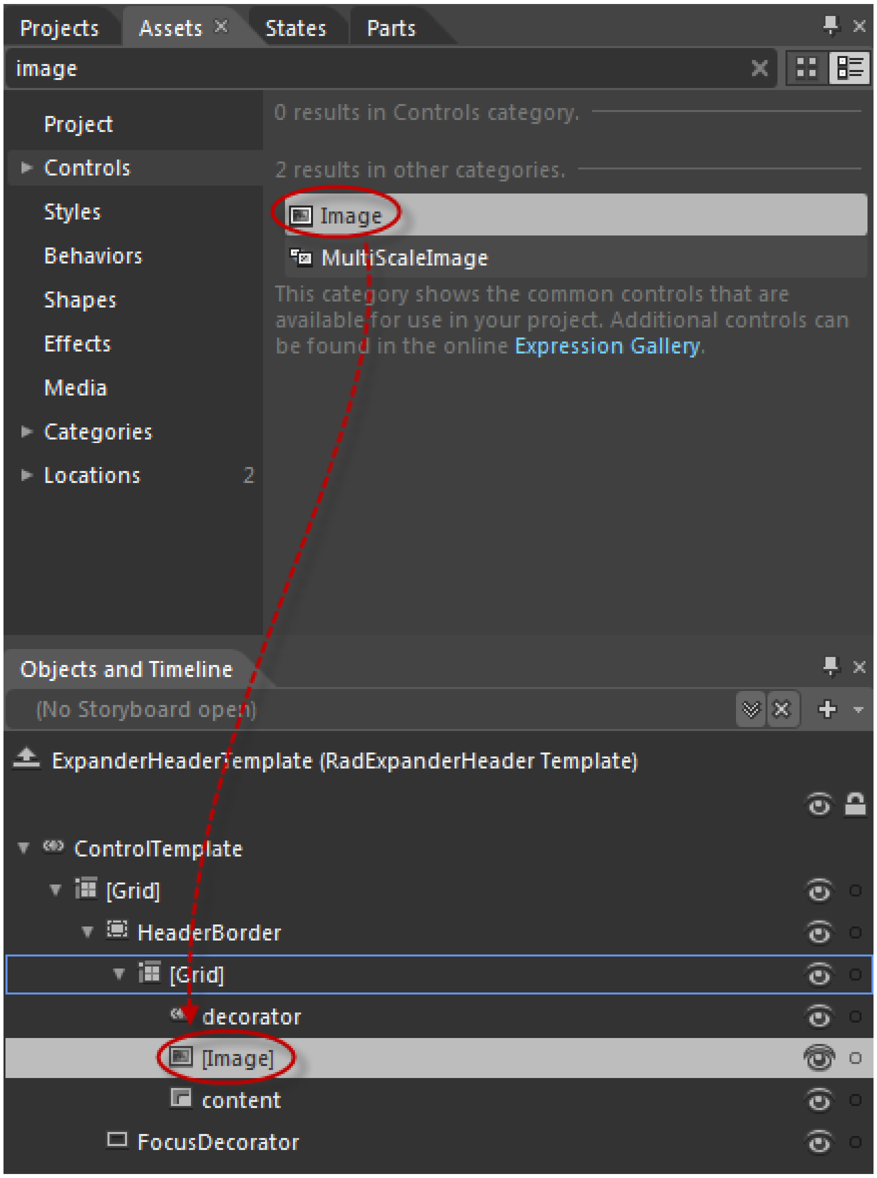
12)Right-click "HeaderButton" in the Objects and Timeline pane and select **Edit Template > Edit Current...**. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleHeaderButton". Click **OK** to create the style resource and close the dialog.

13)If you open all the nodes of the object tree, the Objects and Timeline pane should look something like the screenshot below:



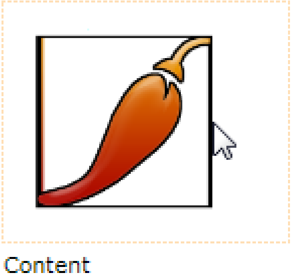
14)In the Objects and Timeline pane, select the "decorator" item in the object tree. In the Properties pane, set the **Appearance > Visibility** property to "Collapsed".

15)In the Assets pane, type the first few characters of "Image" into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view. Locate the Image control and drag it to a point just below the "decorator" node in the Objects and Timeline pane.


Projects
Assets
States
Parts
image
Project
Controls
Styles
Behaviors
Shapes
Effects
Media
Categories
Locations 2
0 results in Controls category.
2 results in other categories.
Image
MultiScaleImage
This category shows the common controls that are available for use in your project. Additional controls can be found in the online Expression Gallery.
Objects and Timeline
(No Storyboard open)
ExpanderHeaderTemplate (RadExpanderHeader Template)
ControlTemplate
[Grid]
HeaderBorder
[Grid]
decorator
[Image]
content
FocusDecorator

16)Double-click "[Image]" and rename it to "peppers". In the Properties pane, locate the **Common Properties > Source** property. Click the ellipses and select "peppers.png" located in the "\courseware\images" directory. Change the **Layout > Width** property to "120" and the **Layout > Height** property to "120".

17)Locate the project in the Projects pane, right-click the project and select "Startup Project" from the context menu. Press **F5** to run the application in its present state.

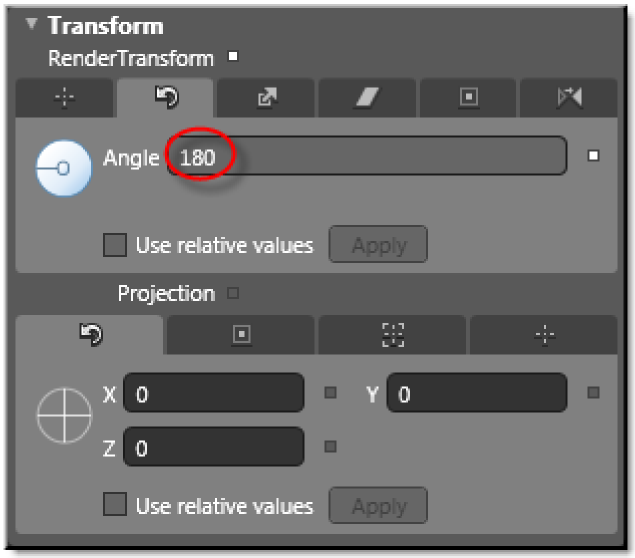


18)Open the States pane. Select the "peppers" image in the Objects and Timeline pane. *The following steps will animate the pepper image as the state toggles between "Expanded" and "Collapsed".*

19)In the States pane, select the "Expanded" state. *Notice the indicator in the upper left corner of the Artboard, that "Expanded state recording is on". This means that property changes you make now will be recorded.*

20)In the Timeline, drag the Timeline marker to the half second mark.

21)Make sure that the "peppers" item is still selected in the Objects and Timeline pane. Navigate to the Properties pane and set **Transform > Rotate > Angle** property to "180":



Angle 180

X 0

Y 0

Z 0

22)In the States pane, select the "Collapsed" state.

23)In the Timeline, drag the Timeline marker to the half second mark.

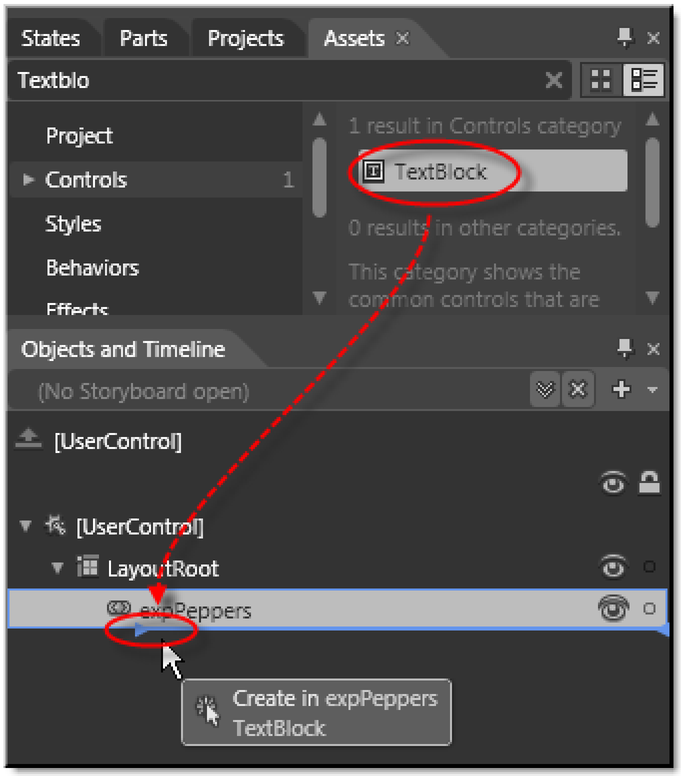
24)Make sure that the "peppers" item is still selected in the Objects and Timeline pane. Navigate to the Properties pane and set **Transform > Rotate > Angle** property to "0".

Note: The property will already be "0". Go ahead and re-enter the value anyway. This will add a new "key frame" item to the Timeline representing the Angle value at the half second mark:

25) In the Objects and Timeline pane, select the "Return scope" button. *Now that you've finished spelunking into the guts of the control, you can return to editing the RadExpander as a whole.*

26)In the Assets pane, type the first few characters of "TextBlock" into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view. Locate the TextBlock control and drag it to a point just below the "expPeppers" node in the Objects and Timeline pane.

Notice the light blue insert marker just before you drop the TextBlock control into place below the "expPeppers". Be aware of where the marker sits in relation to the other controls in the object tree. The marker can be a peer of "expPeppers", i.e. the hint reads "Create in LayoutRoot", or can be a child of "expPeppers", i.e. "Create in expPeppers". The TextBlock should be the content for the RadExpander and so should be a child of "expPeppers".

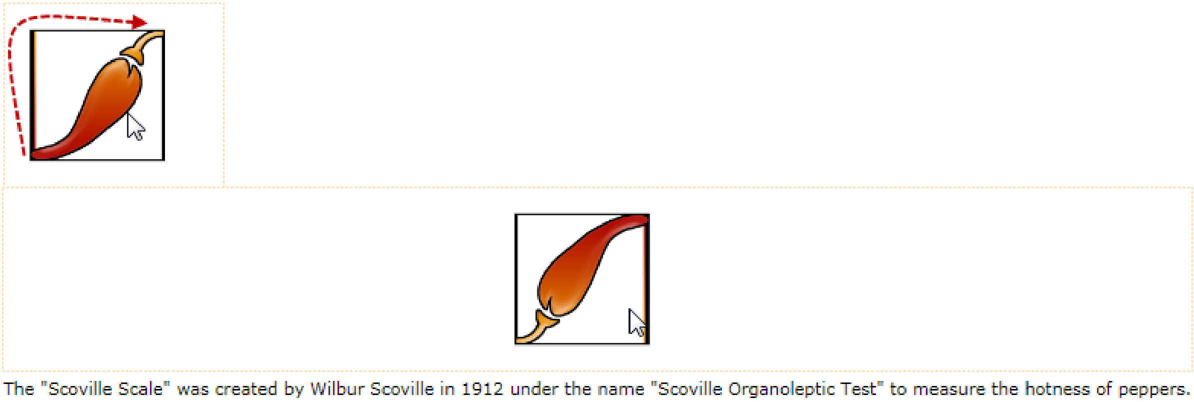


27)In the Properties pane, locate the **Common Properties > Text** property and paste the text below:

The "Scoville Scale" was created by Wilbur Scoville in 1912 under the name "Scoville Organoleptic Test" to measure the hotness of peppers.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



The "Scoville Scale" was created by Wilbur Scoville in 1912 under the name "Scoville Organoleptic Test" to measure the hotness of peppers.

Test Application Features

- Toggle the expand button to verify that the pepper graphic animates and rotates down and then back.

Ideas for Extending This Example

- Add styling to the rest of the header content area.
- Add styling to the content area.
- Turn on animation for the control so that expansion occurs smoothly. See the previous RadExpander > Animation section for more information.

13.6 Wrap Up

In this chapter you saw how RadExpander can save space and present Silverlight content as needed. You learned how to place simple text material in the Header and Content. You saw which properties control expansion direction and animation. You then used Header and Content sub-elements to present multiple Silverlight items of arbitrary complexity and arrangement. Along the way you used control templates to display images, image reflections and custom buttons. Finally, you used Expression Blend to replace the standard "arrow" graphic with an image that rotates when clicked.

Part XIV

Drag and Drop.

14 Drag and Drop.

14.1 Objectives

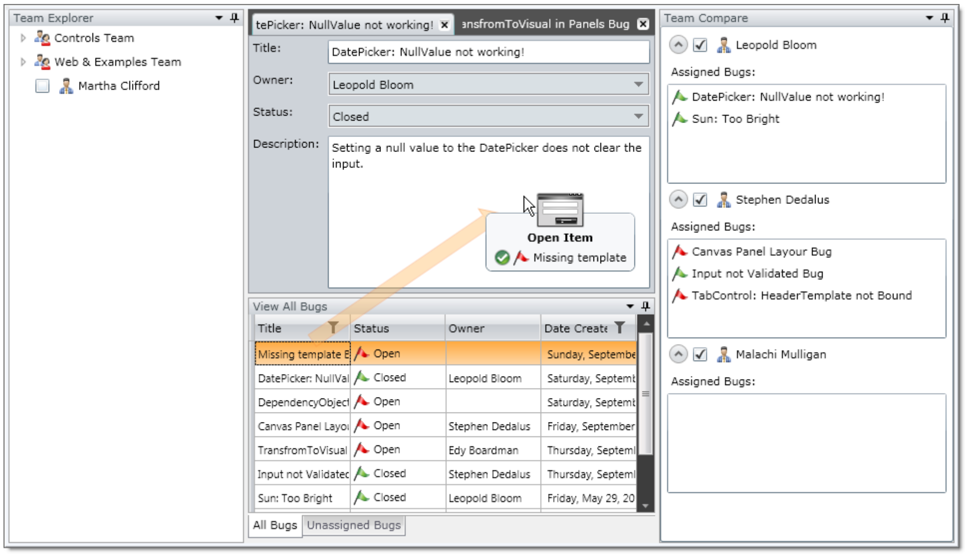
In this chapter you will learn how RadDragAndDropManager is used to allow intuitive drag-and-drop operations between any two Silverlight controls or elements. You will learn the basic property settings that allow drag from a source element and drop to a destination element as well as the event handling required to complete the operation. While discussing the drag-and-drop events you will see how RadDragAndDropManager properties, particularly the DragStatus property, is used to pinpoint the exact state of the operation at the time the event is called. You will learn how to allow or refuse to continue during event processing. You will also learn how to assign visual cues to notify the user of the progress of the operation. Finally, you will see how visual cue templates allow binding and customization.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\DragAndDrop\DragAndDrop.sln

14.2 Overview

Although RadDragAndDropManager is already built into the tree view and tab control, RadDragAndDropManager makes drag and drop possible between any Silverlight control or element.



Title: DatePicker: NullValue not working!

Owner: Leopold Bloom

Status: Closed

Description: Setting a null value to the DatePicker does not clear the input.

Title	Status	Owner	Date Create
Missing template E	Open		Sunday, Septembe
DatePicker: NullVal	Closed	Leopold Bloom	Saturday, Septemt
DependencyObject	Open		Saturday, Septemt
Canvas Panel Layo	Open	Stephen Dedalus	Friday, September
TransfromToVisual	Open	Edy Boardman	Thursday, Septeml
Input not Validatec	Closed	Stephen Dedalus	Thursday, Septeml
Sun: Too Bright	Closed	Leopold Bloom	Friday, May 29, 20

☑ Leopold Bloom

Assigned Bugs:

DatePicker: NullValue not working!
Sun: Too Bright

☑ Stephen Dedalus

Assigned Bugs:

Canvas Panel Layour Bug
Input not Validated Bug
TabControl: HeaderTemplate not Bound

☑ Malachi Mulligan

Assigned Bugs:

The DragAndDrop framework events provide complete control over every stage of the drag-and-drop operation. The drag-and-drop events are all routed and can be handled anywhere in the visual tree. Not only can you allow or prevent drag-and-drop at several points during the operation, you can display visual cues in the source, destination, the dragged object and an optional "Arrow" cue that stretches between the source and destination.

ScrollViewers automatically reveal hidden content when the destination of a drag-and-drop is not visible.

14.3 Getting Started

The minimal steps to bring drag-and-drop functionality to your application are:

- Set the RadDragAndDropManager.AllowDrag attached property to True for any "Source" elements that you want to drag.
- Set the RadDragAndDropManager.AllowDrop attached property to True for any "Destination" elements that should receive the dragged elements.
- Handle DragQueryEvent, DropQueryEvent and DropInfoEvent events.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

XAML Editing

1) Open MainPage.xaml for editing.

2) In the UserControl element, add a reference to the **Telerik.Windows.Controls.DragDrop** namespace in the **Telerik.Windows.Controls** assembly. Also add a handler for the **Loaded** event.

```
<UserControl . . .
xmlns:dragdrop=
"clr-namespace:Telerik.Windows.Controls.DragDrop;assembly=Telerik.Windows.Controls"
Loaded="UserControl_Loaded">
```

3) Inside the main "LayoutRoot" Grid element, add a StackPanel. *Notice the comments that indicate where we will place elements to drag and a destination element that will be the target of the drag operation.*

```
<Grid x:Name="LayoutRoot">

  <StackPanel>

    <!--Source elements to drag-->

    <!--Destination element-->

  </StackPanel>

</Grid>
```

4) Under the comment "<!--Source elements to drag-->", add an **Ellipse** element with the following property settings:

a) Width="50"

b) Height="50"

c) HorizontalAlignment="Center"

d) Margin="5"

e) Fill="Blue"

f) dragdrop:RadDragAndDropManager.AllowDrag="True"

5) Add two more **Ellipse** elements with the same settings. Change the **Fill** properties for the two ellipses to be "Red" and "Green", respectively.

6) Under the comment "<!--Destination element-->", add a **StackPanel** element with the following property settings:

a) MinHeight="200"

b) Margin="5"

c) Background="Silver"

d) dragdrop:RadDragAndDropManager.AllowDrop="True"

The completed XAML should look like the following:

XAML

```
<Grid x:Name="LayoutRoot">

  <StackPanel>

    <!--Source elements to drag-->
    <Ellipse Width="50" Height="50" HorizontalAlignment="Center"
        Margin="5" Fill="Blue"
        dragdrop:RadDragAndDropManager.AllowDrag="True"></Ellipse>

    <Ellipse Width="50" Height="50" HorizontalAlignment="Center"
        Margin="5" Fill="Red"
        dragdrop:RadDragAndDropManager.AllowDrag="True"></Ellipse>

    <Ellipse Width="50" Height="50" HorizontalAlignment="Center"
        Margin="5" Fill="Green"
        dragdrop:RadDragAndDropManager.AllowDrag="True"></Ellipse>

    <!--Destination element-->
    <StackPanel MinHeight="200" Margin="5" Background="Silver"
        dragdrop:RadDragAndDropManager.AllowDrop="True"></StackPanel>

  </StackPanel>

</Grid>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these name spaces:

a) **Telerik.Windows.Controls.DragDrop**

2) In the Loaded event handler, call RadDragAndDropManager methods to add handlers for routed events. The actual handlers will be added in later steps.

VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadDragAndDropManager.AddDragQueryHandler(Me, OnDragQuery)
  RadDragAndDropManager.AddDropQueryHandler(Me, OnDropQuery)
  RadDragAndDropManager.AddDropInfoHandler(Me, OnDropInfo)
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadDragAndDropManager.AddDragQueryHandler(this, OnDragQuery);
   RadDragAndDropManager.AddDropQueryHandler(this, OnDropQuery);
   RadDragAndDropManager.AddDropInfoHandler(this, OnDropInfo);
}
```

3) In the OnDragQuery event handler, set the event arguments **QueryResult** to "True". Also set the **Options.DragCue** to the string "Dragging...".

*Setting **QueryResult** ="True" allows the drag operation to continue. **Options.DragCue** is the object that displays visually under the mouse as it moves during the drag operation.*

VB

```
Private Sub OnDragQuery(ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  e.QueryResult = True
  e.Options.DragCue = "Dragging..."
  e.Handled = True
End Sub
```

C#

```
private void OnDragQuery(object sender, DragDropQueryEventArgs e)
{
   e.QueryResult = true;
   e.Options.DragCue = "Dragging...";
   e.Handled = true;
}
```

4) In the OnDropQuery event handler, set the event arguments QueryResult = "True" and Handled = "True". *Setting QueryResult allows the drop operation to continue. The Handled property is specific to routed events and indicates that the event should not propagate any further.*

VB

```
Private Sub OnDropQuery( _
ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  e.QueryResult = True
  e.Handled = True
End Sub
```

C#

```
private void OnDropQuery(object sender, DragDropQueryEventArgs e)
{
   e.QueryResult = true;
   e.Handled = true
}
```

5) In the OnDropInfo event handler, complete the drop operation by removing the dragged ellipse from its current parent and adding it to the drop Destination StackPanel.

a) Check that the event arguments Options.Status is in a DragStatus.DropComplete state. *Note: DropInfoEvent fires multiple times in varying statuses such as DragStatus.DropPossible.*

b) Get a reference to the Ellipse being dragged, get its Parent and remove the Ellipse from the Parent's Children collection.

c) Get a reference to the destination StackPanel. Add the Ellipse to the StackPanel Children collection.

d) Mark the routed event as Handled.

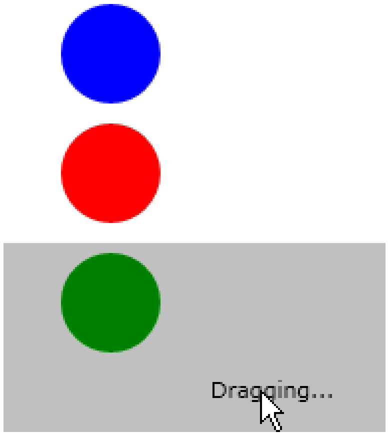
VB

```
Private Sub OnDropInfo(ByVal sender As Object, ByVal e As DragDropEventArgs)
  ' verify that the drop is completed
  If e.Options.Status = DragStatus.DropComplete Then
    ' get a reference to the dragged ellipse
    Dim ellipse As Ellipse = TryCast(e.Options.Source, Ellipse)
    ' remove the ellipse from its parent so it can be added elsewhere
    TryCast(ellipse.Parent, Panel).Children.Remove(ellipse)
    ' get a reference to the drag destination StackPanel so we
    ' can use its methods
    Dim panel As StackPanel = TryCast(e.Options.Destination, StackPanel)
    ' add the ellipse to the destination StackPanel children
    panel.Children.Add(ellipse)
    ' decline further routed event handling
    e.Handled = True
  End If
End Sub
```

```
private void OnDropInfo(object sender, DragDropEventArgs e)
{
  // verify that the drop is completed
  if (e.Options.Status == DragStatus.DropComplete)
  {
    // get a reference to the dragged ellipse
    Ellipse ellipse = e.Options.Source as Ellipse;
    // remove the ellipse from its parent so it can be added elsewhere
    (ellipse.Parent as Panel).Children.Remove(ellipse);
    // get a reference to the drag destination StackPanel so we
    // can use its methods
    StackPanel panel = e.Options.Destination as StackPanel;
    // add the ellipse to the destination StackPanel children
    panel.Children.Add(ellipse);
    // decline further routed event handling
    e.Handled = true;
  }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



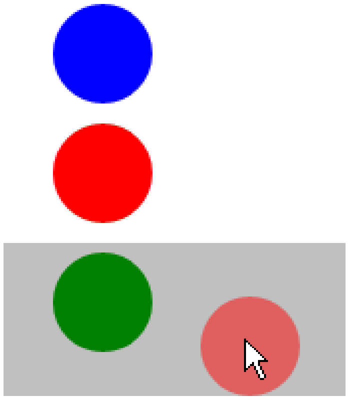
Test Application Features

1) Drag each of the elements to the StackPanel. There are no other drop destinations, so this will only allow a one-way drag to the StackPanel.

Ideas for Extending This Example

- Drag other element types.
- Include additional destinations.

- Change the DragCue to some other representation, such as a faded copy of the source ellipse:



The code would be changed to the example code below:

```
Private Sub OnDragQuery(ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  e.QueryResult = True
  Dim sourceEllipse As Ellipse = TryCast(e.Options.Source, Ellipse)
  Dim sourceColor As Color = (TryCast(sourceEllipse.Fill, SolidColorBrush)).Color
  Dim dragBrush As New SolidColorBrush() With {.Color = sourceColor, .Opacity = 0.5}

  e.Options.DragCue = New Ellipse() With { _
.Fill = dragBrush,  _
.Width = sourceEllipse.Width,  _
.Height = sourceEllipse.Height}
  e.Handled = True
End Sub
```

C#

```
private void OnDragQuery(object sender, DragDropQueryEventArgs e)
{
   e.QueryResult = true;
   Ellipse sourceEllipse = e.Options.Source as Ellipse;
   Color sourceColor = (sourceEllipse.Fill as SolidColorBrush).Color;
   SolidColorBrush dragBrush = new SolidColorBrush()
      {
         Color = sourceColor,
         Opacity = 0.5
      };

   e.Options.DragCue = new Ellipse()
   {
      Fill = dragBrush,
      Width = sourceEllipse.Width,
      Height = sourceEllipse.Height
   };
   e.Handled = true;
}
```

14.4 Control Details

14.4.1 Overview.

RadDragAndDropManager makes drag and drop possible between any Silverlight control or element (not just RadControls). RadDragAndDropManager is a static class and not a control and so is not available in the Toolbox. Instead, you add an XML namespace to Telerik.Windows.Controls.DragDrop in your XAML to reference RadDragAndDropManager as shown in the example below:

XAML

```
<UserControl . . .
xmlns:dragdrop=
"clr-namespace:Telerik.Windows.Controls.DragDrop;assembly=Telerik.Windows.Controls"
>
```

Once you have access to RadDragAndDropManager, the remaining steps are:

- Make one or more controls "draggable"
- Configure one or more controls to accept dropped objects.
- Handle drag and drop events. Event handling is necessary because any number of any type of controls that can be dragged. Using event handlers provides the flexibility to deal with any drag-and-drop situation.

14.4.2 Make a Control Draggable

To make an control "draggable", set the **AllowDrag** property to "True". AllowDrag is an attached property and can be set on an existing Silverlight control:

XAML

```
<Ellipse dragdrop:RadDragAndDropManager.AllowDrag="True"/>
```

In code, use the **SetAllowDrag()** method, passing the control to be dragged.

VB

```
RadDragAndDropManager.SetAllowDrag(BlueEllipse, True)
```

C#

```
RadDragAndDropManager.SetAllowDrag(BlueEllipse, true);
```

Notes

Setting the attached property in code is recommended only when the visual objects are created in code and a style is not / cannot be applied.

Another strategy that works when you have access to a style for the items, lets you avoid having to code for each item by setting a property that propagates to all items:

XAML

```
<ListBox x:Name="listBox">
   <ListBox.ItemContainerStyle>
      <Style TargetType="ListBoxItem">
         <Setter
               Property="dragDrop:RadDragAndDropManager.AllowDrag"
               Value="True" />
      </Style>
   </ListBox.ItemContainerStyle>
  . . .
</ListBox>
```

14.4.3 Accept Dropped Controls

The mirror of the AllowDrag property is the **AllowDrop** attached property. You can add AllowDrop to any Silverlight control:

XAML

```
<TextBlock
     dragdrop:RadDragAndDropManager.AllowDrop="True"></TextBlock>
```

To set the same property in code, use the **SetAllowDrop()** method. Like SetAllowDrag(), SetAllowDrop() shines when a number of elements that are unknown at design time need to be configured to allow drop. The example below allows "Time slots" in a RadScheduler to allow drop:

VB

```
Private Sub InitializeTimeSlotItems()
  Me.timeSlotItems = Me.Scheduler.ChildrenOfType(Of TimeSlotItem)()
  For Each item As TimeSlotItem In Me.timeSlotItems
    item.SetValue(RadDragAndDropManager.AllowDropProperty, True)
  Next item
End Sub
```

C#

```
private void InitializeTimeSlotItems()
{
   this.timeSlotItems = this.Scheduler.ChildrenOfType<TimeSlotItem>();
   foreach (TimeSlotItem item in this.timeSlotItems)
   {
      item.SetValue(RadDragAndDropManager.AllowDropProperty, true);
   }
}
```

14.4.4 RadDragAndDropManager

With RadDragAndDropManager you can:

- Flag the controls that can be dragged and that can accept a drag.
- Hook up drag-and-drop events.
- Maintain information on all aspects of the Drag-and-Drop operation using the **DragDropOptions** property.
- Automatically scroll content into view using the **AutoBringIntoView** property.
- Prevent unintended dragging on mouse-down by setting the **DragStartThreshold** property. This is the distance in pixels that the user needs to drag an object before a real drag operation starts.
- Generate default visual cues.

DragDropOptions

Most aspects of the drag-and-drop operation are tracked through the RadDragAndDropManager **Options** property. Some of the key Options sub-properties are:

- The **Source** and **Destination** FrameworkElement objects represent the drag source and target.
- While the drag operation is underway, you can supply feedback to the user with several visual cues: **ArrowCue** and **DragCue**. See the upcoming section "Visual Cues" for detail on how to use these properties.
- The **Payload** property lets you tuck away arbitrary object data.
- The **Status** property provides fine-grain understanding of the drag progress during drag-and-drop events. See the upcoming section "Events" for a detailed listing of these events.
- When you have popups involved with a drag-and-drop operation, be sure to add them to the **ParticipatingVisualRoots** collection. This collection contains visual roots what will participate in the drag/drop operation but are not descendants of the application root visual.

From the Forums...

Question: "Drag & Drop works except inside a Popup. I tried to use a Popup, unfortunately, all the drop events are not launched, we can see the visual cue, but cannot deliver 'the drop'." How can I get the drag and drop properties to work?

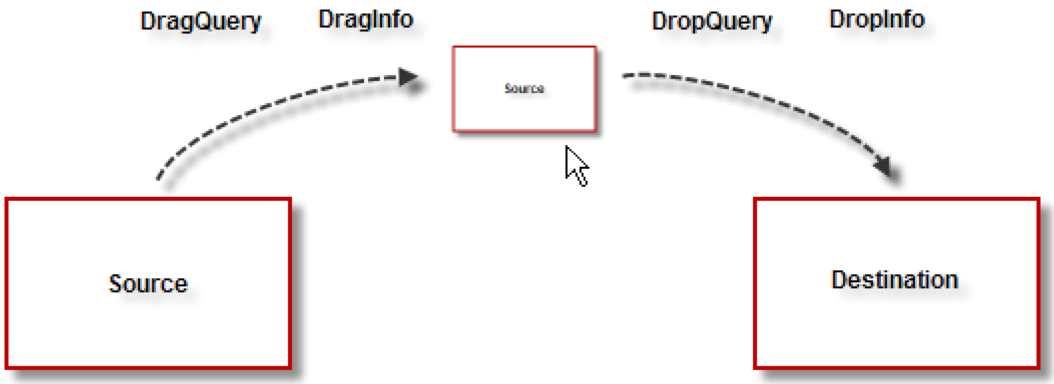
Answer: "'Windows' in Silverlight come up in popups and by default popups are not part of the main visual tree. Therefore, RadDragAndDrop does not know about them. There is a property on the DragDrop option, called ParticipatingVisualRoots. You can add the currently opened child windows to ParticipatingVisualRoots and they will participate in the DragDrop. You can keep a static collection with all the opened windows and add it during DragInfo. You can do this transparently by adding a DragInfo handler to the root visual of the application that will do this for every successful drag operation."

There is no need to do this for the RadWindow or popups that Telerik controls create. More typically you may need to do this for the ChildWindow control.

14.4.5 Events.

RadDragAndDropManager generates four events: DragQuery, DragInfo, DropQuery and DropInfo. The "Query" events fire just before some action is taken, allowing you to change settings or cancel the event altogether. The "Info" events notify you that some action has occurred. The diagram below shows the general order that these events occur in. These are all "routed" events, i.e. they can be handled by any of the elements in the visual or logical tree.

- **DragQuery** lets you allow or cancel the drag and also lets you specify visual cues that let the user know what's happening.
- **DragInfo** fires when the drag is already under way.
- **DropQuery** lets you allow or cancel dropping onto some other element. The event handler supplies an "options" parameter with complete information on what is being dragged, where its being dropped, the status of the operation and all the other data about the drag-and-drop.
- **DropInfo** occurs when the drop is complete. The RadDragAndDropManager infrastructure displays all the trappings of a drag-and-drop, but doesn't actually change the underlying state of elements in the page. The DropInfo handler is your opportunity to actually move or copy objects to their new locations in response to the drop.



To subscribe to the routed events, use the static methods of RadDragAndDropManager: **AddDragQueryHandler**, **AddDragInfoHandler**, **AddDropQueryHandler** and **AddDropInfoHandler**. The snippet below demonstrates hooking up DropQuery using the static AddDropQueryHandler() method. The first parameter is to the object handling the event, in this case the UserControl ("Me" or "this"). The second parameter is a reference to the handler itself.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  '...

  RadDragAndDropManager.AddDropQueryHandler(Me, AddressOf OnDropQuery)
End Sub


Private Sub OnDropQuery(ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  e.QueryResult = True
  e.Handled = True
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
  //. . .

  RadDragAndDropManager.AddDropQueryHandler(this, OnDropQuery);
}


private void OnDropQuery(object sender, DragDropQueryEventArgs e)
{
  e.QueryResult = true;
  e.Handled = true;
}
```

Event Status

The four events actually fire multiple times as they track progress the operation from the perspective of both source and destination. Each event handler contains a key property, **DragDropOptions**, that holds a **Status** sub-property. Status lets you know the nature of each event call:

None	No drag/drop is taking place.
DragQuery	A drag is about to take place for an element where AllowDrag = "True". To allow the drag to proceed, set event handler QueryResult argument to "True".
DragInProgress	Dragging has started, no drop locations have been found.
DragComplete	The drag is complete. Source objects may be removed at this point.
DragCancel	The drag was canceled.
DropDestinationQuery	The destination is asked if the element can be dropped from this particular source.
DropSourceQuery	The source is asked if the element can be dropped on this particular destination
DropPossible	The drop is acknowledged from both parties. You can use this status to signal visually to the user that a drop is possible.
DropImpossible	The drop is refused by one or both parties.
DropComplete	The drop is complete. Copying or moving elements takes place here.
DropCancel	The drop was canceled.

Generally a successful DragDrop involves the following events and statuses:

- DragQuery event with status DragQuery
- DragInfo event with status DragInProgress

Then when a possible target is reached:

- DragQuery event with status DropSourceQuery
- DropQuery event with status DropDestinationQuery
- DragInfo event with status DropPossible
- DropInfo event with status DropPossible

On a successful drop:

- DragInfo event with status DragComplete
- DropInfo event with status DropComplete

Here's a log of the typical event sequence taken from the "Getting Started" project. From the log you can see that this is a two-way, handshake process where both the source and destination must agree for the operation to continue. Notice that when the DragQuery and DragInProgress first fire, the destination object is unknown.

```
Event: OnDragQuery Status: DragQuery Source: Ellipse Destination: null
Event: OnDragInfo Status: DragInProgress Source: Ellipse Destination: null
Event: OnDropQuery Status: DropDestinationQuery Source: Ellipse Destination: StackPanel
Event: OnDragQuery Status: DropSourceQuery Source: Ellipse Destination: StackPanel
Event: OnDropInfo Status: DropPossible Source: Ellipse Destination: StackPanel
Event: OnDragInfo Status: DropPossible Source: Ellipse Destination: StackPanel
Event: OnDragInfo Status: DragComplete Source: Ellipse Destination: StackPanel
Event: OnDropInfo Status: DropComplete Source: Ellipse Destination: StackPanel
```

Gotcha!

Question: I'm handling the DragQuery event and have set QueryResult = "True" when the status is "DragQuery". But the drag does not take place. What is happening?

Answer: There are two checks that must both pass. QueryResult must be set true when the status is DragQuery ***and*** DropSourceQuery. DragQuery occurs at the very beginning of the event sequence and "Destination" is null at this point. When the status is DropSourceQuery, the operation can be canceled based on the destination object.

Responding to Events

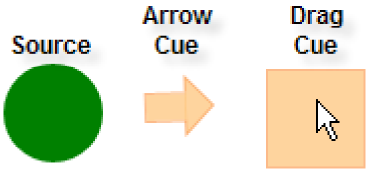
Now that you know the events and the status, you will want to make your visual layout respond to the drag operation. What happens here is up to your business logic. For example, you may have a "Tool Box" scenario where an item is copied onto an element on the page, or you may be developing a "Visio"-like diagram where node elements are moved around. In this second instance you have to remove an element from one parent element on the page and add it to another. Use the OnDragInfo event to remove an element from one location and the OnDropInfo is always used (for either copy or move type operations) to create the element in its new location. The example below removes an ellipse from its parent, then recreates the ellipse inside a StackPanel where it was dropped.

```
Private Sub OnDragInfo(ByVal sender As Object, ByVal e As DragDropEventArgs)
  Log("OnDragInfo", e.Options)

  If e.Options.Status = DragStatus.DragComplete Then
    ' get a reference to the dragged ellipse
    Dim ellipse As Ellipse = TryCast(e.Options.Source, Ellipse)
    ' remove the ellipse from its parent so it can be added elsewhere
    TryCast(ellipse.Parent, Panel).Children.Remove(ellipse)
  End If

End Sub

Private Sub OnDropInfo(ByVal sender As Object, ByVal e As DragDropEventArgs)
  Log("OnDropInfo", e.Options)


  ' verify that the drop is completed
  If e.Options.Status = DragStatus.DropComplete Then
    ' get a reference to the dragged ellipse
    Dim ellipse As Ellipse = TryCast(e.Options.Source, Ellipse)

    ' get a reference to the drag destination StackPanel so we
    ' can use its methods
    Dim panel As StackPanel = TryCast(e.Options.Destination, StackPanel)
    ' add the ellipse to the destination StackPanel children
    panel.Children.Add(ellipse)
    ' decline further routed event handling
    e.Handled = True
  End If
End Sub
```

```
private void OnDragInfo(object sender, DragDropEventArgs e)
{
   Log("OnDragInfo", e.Options);

   if (e.Options.Status == DragStatus.DragComplete)
   {
      // get a reference to the dragged ellipse
      Ellipse ellipse = e.Options.Source as Ellipse;
      // remove the ellipse from its parent so it can be added elsewhere
      (ellipse.Parent as Panel).Children.Remove(ellipse);
   }
}

private void OnDropInfo(object sender, DragDropEventArgs e)
{
   Log("OnDropInfo", e.Options);


   // verify that the drop is completed
   if (e.Options.Status == DragStatus.DropComplete)
   {
      // get a reference to the dragged ellipse
      Ellipse ellipse = e.Options.Source as Ellipse;

      // get a reference to the drag destination StackPanel so we
      // can use its methods
      StackPanel panel = e.Options.Destination as StackPanel;
      // add the ellipse to the destination StackPanel children
      panel.Children.Add(ellipse);
      // decline further routed event handling
      e.Handled = true;
   }
}
```

14.4.6 Visual Cues

Display a visual cue under the mouse when dragging and an "Arrow" cue between the source and the mouse by assigning the options **DragCue** and **ArrowCue** properties. The screenshot below shows the default appearance when a drag is underway.



You can use these defaults by calling the **GenerateVisualCue()** and **GenerateArrowCue()** methods. You may pass an optional Framework element to GenerateVisualCue() and the properties of that element will be used as a basis for the visual cue.

VB

```
Private Sub OnDragQuery(ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  Log("OnDragQuery", e.Options)

  e.QueryResult = True
  e.Options.DragCue = _
RadDragAndDropManager.GenerateVisualCue(TryCast(e.Options.Source, FrameworkElement))
  e.Options.ArrowCue = RadDragAndDropManager.GenerateArrowCue()
  e.Handled = True
End Sub
```

C#

```
private void OnDragQuery(object sender, DragDropQueryEventArgs e)
{
   Log("OnDragQuery", e.Options);

   e.QueryResult = true;
   e.Options.DragCue =
      RadDragAndDropManager.GenerateVisualCue(
         e.Options.Source as FrameworkElement);
   e.Options.ArrowCue = RadDragAndDropManager.GenerateArrowCue();
   e.Handled = true;
}
```

You can use visual elements of the source or destination to help the user decide what to do. The sample below uses the OnDragInfo when the status is DropPossible to highlight the background when it's ok to drop the dragged item.

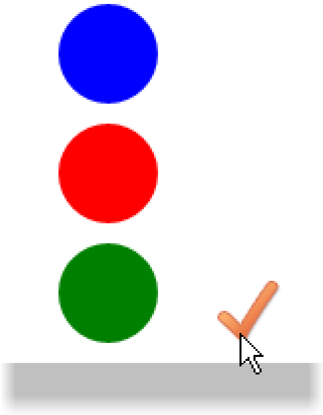
VB

```
Private Sub OnDragInfo(ByVal sender As Object, ByVal e As DragDropEventArgs)
  DropPanel.Background = _
If(e.Options.Status = DragStatus.DropPossible, _
New SolidColorBrush(Colors.Gray), _
New SolidColorBrush(Colors.LightGray))
End Sub
```

C#

```
private void OnDragInfo(object sender, DragDropEventArgs e)
{
   DropPanel.Background = e.Options.Status == DragStatus.DropPossible ?
      new SolidColorBrush(Colors.Gray) : new SolidColorBrush(Colors.LightGray);
}
```

You're not stuck with the built-in visual cues. All cue objects are simply objects, so you can assign a string or a visual element. The screenshot below shows the DragCue as a "check" image.



From the Forums...

Question: "I would like to show an array of objects with a specific foreground and background."

Answer: "You can put anything as the DragCue during DragDrop. What you can try is add a ContentControl. Then as Content you assign the collection of items being dragged. Then you give it a ContentTemplate which has an items control with a template for your items. The ItemTemplate of this inner items control can contain borders, TextBlocks, and other visual items which you can modify. If you assign a collection of the dragged items to the content of the DragCue then the inner items control can have its ItemsSource equal to {Binding}, i.e. bound directly to the items.

If do not like all the binding, you can create an ItemsControl in code and set its items source and item template properties. Alternatively you can build the whole visual object in code, starting from a panel and adding TextBlocks (or other visuals) as needed. Then you assign the panel as DragCue. In all these cases you can modify the brushes or fonts of the displayed items."

You can directly assign some object to Content, say a scaled down ellipse to indicate that we're dragging an ellipse, or you could assign a TextBlock that describes the action. There's no limit as to what you could assign here. The effective limit is complexity. Let's say you want the cue to show a description, an icon and an image. In this situation you should use the ContentControl **ContentTemplate** property and assign a DataTemplate to it. Now you can build templates of great complexity by hand or using Expression Blend and they can be used as your cue. The example from the screenshot above actually uses a simple template that contains an image:

XAML

```
<UserControl.Resources>

  <DataTemplate x:Key="DragTemplate">
    <Image Source="images/check.png" Stretch="None"
        VerticalAlignment="Top" />
  </DataTemplate>

</UserControl.Resources>
```

In code, create a ContentControl instance and assign the template from the Resources collection and cast it to be a DataTemplate type. Finally, assign the ContentControl to the cue.

VB

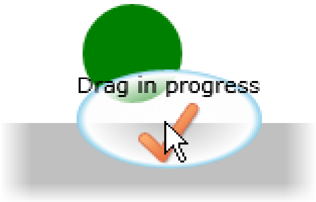
```
Private Sub OnDragQuery(ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  e.QueryResult = True
  Dim cue As New ContentControl()
  cue.ContentTemplate = TryCast(Me.Resources("DragTemplate"), DataTemplate)
  e.Options.DragCue = cue

  e.Handled = True
End Sub
```

C#

```
private void OnDragQuery(object sender, DragDropQueryEventArgs e)
{
   e.QueryResult = true;
   ContentControl cue = new ContentControl();
   cue.ContentTemplate = this.Resources["DragTemplate"] as DataTemplate;
   e.Options.DragCue = cue;

   e.Handled = true;
}
```

14.5 Binding

You can bind data to visual cues data templates for greater flexibility. The screenshot below shows a slightly more complex template that demonstrates binding.



The code assigns a simple "MyDragInfo" object to the Content property. MyDragInfo has "ImagePath" and "Comment" properties that are then bound in the ContentTemplate.

VB

```
Private Sub OnDragQuery(ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  Log("OnDragQuery", e.Options)

  e.QueryResult = True

  Dim cue As New ContentControl()
  cue.ContentTemplate = TryCast(Me.Resources("DragTemplate"), DataTemplate)
  cue.Content = New MyDragInfo() With {.ImagePath = "images/check.png", .Comment = "Drag in progress"}
  e.Options.DragCue = cue

  e.Handled = True
End Sub
```

C#

```
private void OnDragQuery(object sender, DragDropQueryEventArgs e)
{
   Log("OnDragQuery", e.Options);

   e.QueryResult = true;

   ContentControl cue = new ContentControl();
   cue.ContentTemplate =
      this.Resources["DragTemplate"] as DataTemplate;
   cue.Content = new MyDragInfo()
   {
      ImagePath = "images/check.png",
      Comment = "Drag in progress"
   };
   e.Options.DragCue = cue;

   e.Handled = true;
}
```

The XAML for the template is slightly more involved and contains the binding expressions for the ImagePath and Comment.

```
<UserControl.Resources>

  <RadialGradientBrush x:Key="DragBackgroundBrush">
    <GradientStop Offset="0" Color="Transparent" />
    <GradientStop Offset="0.9" Color="White" />
    <GradientStop Offset="1" Color="SkyBlue" />
  </RadialGradientBrush>

  <DataTemplate x:Key="DragTemplate">
    <Grid>
      <Ellipse MinWidth="50" MinHeight="50"
          Fill="{StaticResource DragBackgroundBrush}">
      </Ellipse>
      <StackPanel>
        <TextBlock Text="{Binding Comment}"
            HorizontalAlignment="Center"></TextBlock>
        <Image Source="{Binding ImagePath}"
            Stretch="None" />
      </StackPanel>
    </Grid>
  </DataTemplate>

</UserControl.Resources>
```

14.6 Wrap Up

In this chapter you learned how RadDragAndDropManager can be used to allow intuitive drag-and-drop operations between any two Silverlight controls or elements. You learned the basic property settings to allow drag from a source element and drop to a destination element as well as the event handling required to complete the operation. While discussing the drag-and-drop events you saw how RadDragAndDropManager properties, particularly the DragStatus property, is used to pinpoint the exact state of the operation at the time the event is called. You learned how to allow or refuse to continue during event processing. You learned how to assign visual cues to notify the user of the progress of the operation. You also learned how to assign templates to the visual cues to allow binding and customization.

Part XV

Date, Time and Calendar

15 Date, Time and Calendar

15.1 Objectives

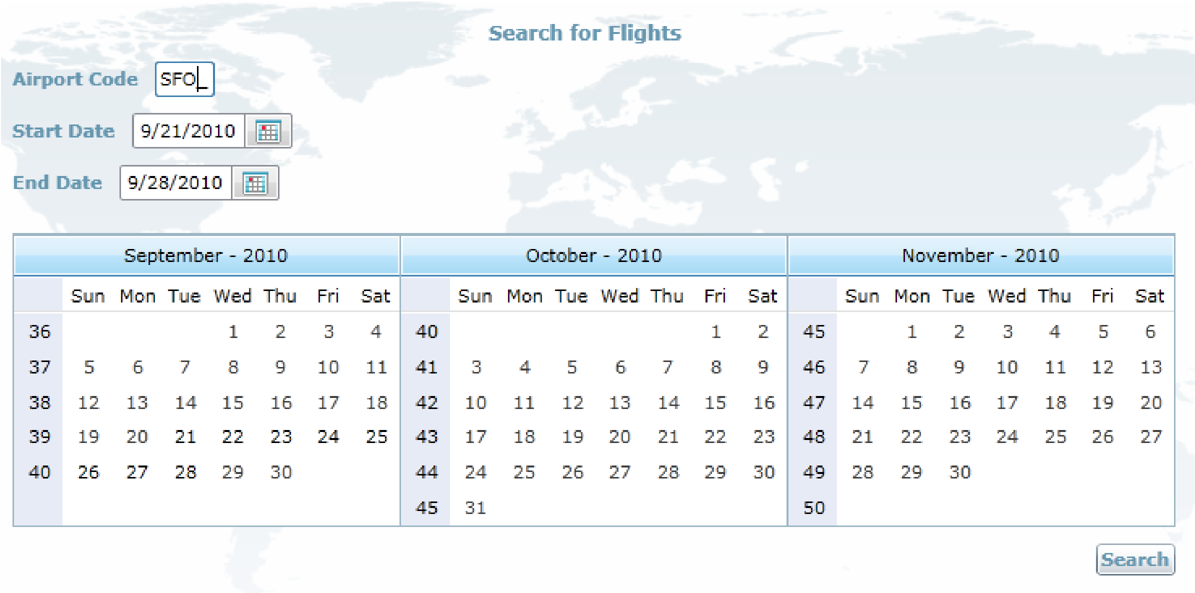
In this chapter you will learn how to use calendar, date picker, time picker and datetime picker control to collect date and time data from the user. In the Control Details section of this chapter, you will review the types of date and time controls and the appropriate uses for each control type. You will learn how to control the number of dates that can be selected at one time and the manner that they can be selected. You will also learn about the SelectedDate and SelectedDates properties for RadDatePicker and SelectedValue property for RadDateTimePicker used to store selections and the SelectionChanged event that flags when its time to look at this property. You will become aware of the properties used to limit the dates and times that can be selected or viewed. You will also look at the RadClock control used to select time values. In the Binding section of this chapter you will see how calendar and date time picker controls are bound to custom objects and will also look at a short example of binding RadTimePicker to a collection of TimeSpan objects. Finally, you will customize the background of RadCalendar using Expression Blend.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Calendar\Calendar.sln

15.2 Overview

If your user interface requires picking dates and times, use RadCalendar and RadDatePicker controls. RadCalendar looks like the page of a calendar where the user can click dates to select them. The RadDatePicker is a hybrid of masked text box/parser and a button that displays the calendar. RadDatePicker takes the minimum amount of screen real estate.



Search for Flights

Airport Code: SFO

Start Date: 9/21/2010

End Date: 9/28/2010

September - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
36				1	2	3	4
37	5	6	7	8	9	10	11
38	12	13	14	15	16	17	18
39	19	20	21	22	23	24	25
40	26	27	28	29	30		

October - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
40						1	2
41	3	4	5	6	7	8	9
42	10	11	12	13	14	15	16
43	17	18	19	20	21	22	23
44	24	25	26	27	28	29	30
45	31						

November - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
45		1	2	3	4	5	6
46	7	8	9	10	11	12	13
47	14	15	16	17	18	19	20
48	21	22	23	24	25	26	27
49	28	29	30				
50							

Calendar Features

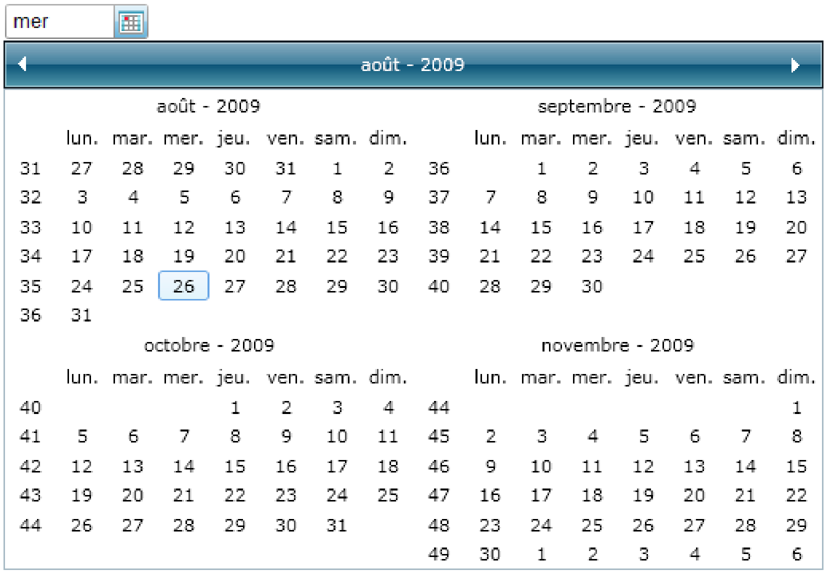
- Display modes show the calendar in Century, Decade, Year or Month views.
- Multiple months can be shown at one time.
- Three styles of date selection let the user choose only one date at a time or multiple dates. Explorer style selection using Ctrl and Shift keys is also supported.
- Viewable and selectable dates can be restricted to custom ranges.
- The calendar can be bound to Object, XML or WCF services. Two-way binding allows the user to update the data source simply by clicking the calendar.
- The calendar can be internationalized by setting a single property.

DateTime Picker Features

- All the features of RadCalendar and/or RadClock in an easy-to-use, space saving control.
- The DateTime picker control can be bound to various data sources.
- The RadDateTimePicker advanced parsing feature allow you to enter any number or string in the input field and the entered value will be transformed to a valid date, time or both.
- The RadDateTimePicker can also be internationalized by setting a single property.
- InputMode - The user can set mode for RadDateTimePicker which specifies whether the RadDateTimePicker control will allow Time input, Date Input or both.
- A PreviewToolTip - This new cool feature helps the user to preview the inputted text while typing in the RadDateTimePicker.

Date Picker Features

- All the features of RadCalendar in an easy-to-use, space saving control.
- All the features of RadDateTimePicker but selecting only dates.



mer

août - 2009

août - 2009

	lun.	mar.	mer.	jeu.	ven.	sam.	dim.
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31						

septembre - 2009

	lun.	mar.	mer.	jeu.	ven.	sam.	dim.
36		1	2	3	4	5	6
37	7	8	9	10	11	12	13
38	14	15	16	17	18	19	20
39	21	22	23	24	25	26	27
40	28	29	30				

octobre - 2009

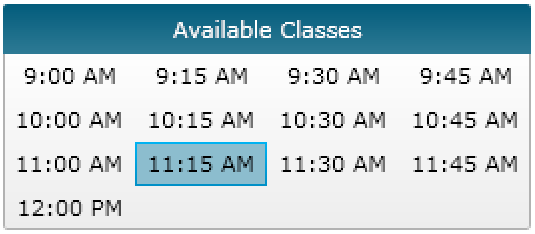
	lun.	mar.	mer.	jeu.	ven.	sam.	dim.
40				1	2	3	4
41	5	6	7	8	9	10	11
42	12	13	14	15	16	17	18
43	19	20	21	22	23	24	25
44	26	27	28	29	30	31	

novembre - 2009

	lun.	mar.	mer.	jeu.	ven.	sam.	dim.
44							1
45	2	3	4	5	6	7	8
46	9	10	11	12	13	14	15
47	16	17	18	19	20	21	22
48	23	24	25	26	27	28	29
49	30	1	2	3	4	5	6

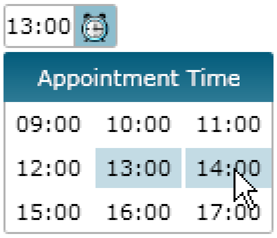
Time Picker Features

- The time picker comes either as RadTimePicker, a space saving text box/parser plus drop down or as RadClock where all the times are viewable immediately.



Available Classes

9:00 AM	9:15 AM	9:30 AM	9:45 AM
10:00 AM	10:15 AM	10:30 AM	10:45 AM
11:00 AM	11:15 AM	11:30 AM	11:45 AM
12:00 PM			



13:00

Appointment Time

09:00	10:00	11:00
12:00	13:00	14:00
15:00	16:00	17:00

- All the features of RadDateTimePicker but selecting only times.

15.3 Getting Started

In this walk through we will build a stub airline flight form that will collect a date range using RadDatePicker controls. When the "Search" button is clicked, "flight" dates within the range will be selected.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

   a) Telerik.Windows.Controls

   b) Telerik.Windows.Controls.Input

   c) Telerik.Windows.Controls.Summer

4) In the Solution Explorer, right-click the project and select **Add > New Folder** from the context menu. Rename the folder "Images".

5) Right-click the new "Images" folder and select Add > Existing Item... from the context menu. Locate the image file "map.png" from the "\courseware\images" directory. Select the file and click the **Add** button.

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace alias for the **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Input** assemblies. Also add a handler for the "Loaded" event.

   Note: Each xmlns statement should be all on one line. The example below is split up to fit the size constraints of the page in this manual.

```
<UserControl
xmlns:telerik=
"http://schemas.telerik.com/2008/xaml/presentation"
. . .
Loaded="UserControl_Loaded">
```

3) Copy the UserControl.Resources element below inside the UserControl element.

These resources are placed here primarily to let us move property settings out of the control elements they describe. We can see the structure of controls without the property settings being in the way. There are a few points of interest you should take a look at in this markup. The SolidColorBrush named "NormalBrush" uses a shade of color from the "Vista" theme. We will use it to color the titling and button text. Notice that we also have setters for "telerik:StyleManager.Theme" and that we set the theme to "Vista".

```
<UserControl.Resources>

  <SolidColorBrush x:Key="NormalBrush" Color="#FF6596AF" />

  <Style x:Key="DatePickerStyle"
      TargetType="telerik:RadDatePicker">
    <Setter Property="telerik:StyleManager.Theme"
        Value="Vista" />
    <Setter Property="Margin" Value="0,0,10,10" />
    <Setter Property="VerticalAlignment" Value="Bottom" />
  </Style>

  <Style x:Key="CalendarFontStyle" TargetType="TextBlock">
    <Setter Property="FontFamily" Value="Verdana" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Foreground"
        Value="{StaticResource NormalBrush}" />
  </Style>

  <Style x:Key="TitleStyle" TargetType="TextBlock"
      BasedOn="{StaticResource CalendarFontStyle}">
    <Setter Property="Margin" Value="0,0,10,10" />
    <Setter Property="FontSize" Value="12" />
    <Setter Property="VerticalAlignment" Value="Center" />
    <Setter Property="HorizontalAlignment" Value="Center" />
  </Style>

  <Style x:Key="HeadingStyle" TargetType="TextBlock"
      BasedOn="{StaticResource CalendarFontStyle}">
    <Setter Property="Margin" Value="0,0,10,0" />
    <Setter Property="FontSize" Value="11" />
    <Setter Property="VerticalAlignment" Value="Center" />
  </Style>

  <Style x:Key="ButtonStyle" TargetType="Button">
    <Setter Property="FontFamily" Value="Verdana" />
    <Setter Property="FontWeight" Value="Bold" />
    <Setter Property="Foreground"
        Value="{StaticResource NormalBrush}" />
    <Setter Property="HorizontalAlignment" Value="Right" />
  </Style>
  <Style x:Key="GridBackgroundStyle" TargetType="Grid">
    <Setter Property="Background">
      <Setter.Value>
        <ImageBrush ImageSource="../images/map.png"
            Stretch="Uniform" Opacity=".1">
        </ImageBrush>
      </Setter.Value>
    </Setter>
  </Style>

  <Style x:Key="MaskedTextBoxStyle" TargetType="telerik:RadMaskedTextBox">
    <Setter Property="telerik:StyleManager.Theme"
        Value="Vista"></Setter>
    <Setter Property="MaskType" Value="Standard"></Setter>
    <Setter Property="Mask" Value="LLL"></Setter>
    <Setter Property="HorizontalAlignment" Value="Left"></Setter>
  </Style>

  <Style x:Key="StackPanelStyle" TargetType="StackPanel">
    <Setter Property="Orientation" Value="Horizontal" />
    <Setter Property="Margin" Value="0,0,0,10" />
  </Style>

</UserControl.Resources>
```

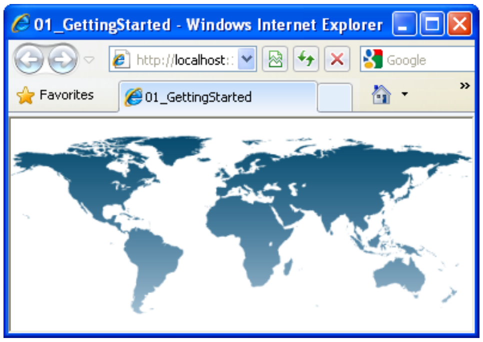
From the Forums...

How do I display a background image behind my controls? How can I make this image appear as a faint, watermark-like picture that recedes in the background and doesn't overpower the foreground?

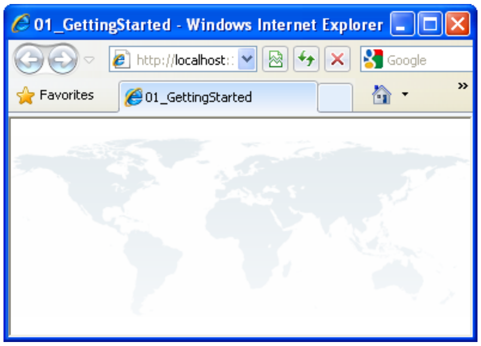
Examine the "GridBackgroundStyle" XAML below. You can set a Brush property to be an ImageBrush and assign the brush to the Grid Background. To make the image seem faint, set the Opacity property to a low value where valid values are "0" to "1".

```
<UserControl.Resources>
. . .
  <Style x:Key="GridBackgroundStyle" TargetType="Grid">
    <Setter Property="Background">
      <Setter.Value>
        <ImageBrush ImageSource="../images/map.png"
             Stretch="Uniform" Opacity=".1">
        </ImageBrush>
      </Setter.Value>
    </Setter>
  </Style>
. . .
</UserControl.Resources>
```

If Opacity is not set, the image appears like this in the browser:



With Opacity set to ".1", the image recedes and is more suitable for a background.



4) Use the XAML below to replace the main "LayoutRoot" Grid element.

This will provide the general layout where we will add additional elements later. The StackPanel inside the grid is centered both vertically and horizontally. The Orientation of this StackPanel is Vertical by default. Inside this outer StackPanel are three more StackPanel elements, each styled to arrange its children horizontally.

```
<Grid x:Name="LayoutRoot" Style="{StaticResource GridBackgroundStyle}">

  <StackPanel HorizontalAlignment="Center"
      VerticalAlignment="Center">

    <!--Title-->

    <StackPanel Style="{StaticResource StackPanelStyle}">

      <!--Airport Code Entry-->

    </StackPanel>

    <StackPanel Style="{StaticResource StackPanelStyle}">

      <!--Departure Entry-->

    </StackPanel>

    <StackPanel Style="{StaticResource StackPanelStyle}">

      <!--Arrival Entry-->

    </StackPanel>

    <!--Search Button-->


    <!--Results Calendar-->

  </StackPanel>

</Grid>
```

5) Locate the comment "<!--Title-->" and replace it with the XAML markup for the titling text below.

```
<!--Title-->
<TextBlock Text="Search for Flights"
      Style="{StaticResource TitleStyle}" />
```

6) Locate the comment "<!--Airport Code Entry-->" and replace it with the XAML markup below. *This step will add an Airport code entry using a RadMaskedTextBox. Notice the "MaskedTextBoxStyle" reference that ties the resource properties to the RadMaskedTextBox.*

```
<!--Airport Code Entry-->
<TextBlock Text="Airport Code"
      Style="{StaticResource HeadingStyle}" />

<telerik:RadMaskedTextBox x:Name="tbAirportCode"
      Style="{StaticResource MaskedTextBoxStyle}" />
```

7) Locate the comment "<!--Start Entry-->" and replace it with the XAML markup below. *This step will add a RadDatePicker and assign its style.*

```
<!--Start Entry-->
<TextBlock Text="Start Date"
      Style="{StaticResource HeadingStyle}" />

<telerik:RadDatePicker x:Name="dpStart"
      Style="{StaticResource DatePickerStyle}"
      SelectionChanged="DateChanged" />
```

8) Locate the comment "<!--End Entry-->" and replace it with the XAML markup below. *This step will add another RadDatePicker. .*

```
<!--End Entry-->
<TextBlock Text="End Date"
      Style="{StaticResource HeadingStyle}" />

<telerik:RadDatePicker x:Name="dpEnd"
      Style="{StaticResource DatePickerStyle}"
      SelectionChanged="DateChanged" />
```

9) Locate the comment "<!--Search Button-->" and replace it with the XAML markup below. *This step adds a standard Button control and attaches a "Click" event handler.*

```
<!--Search Button-->
<Button x:Name="btnSearch"
      Style="{StaticResource ButtonStyle}"
      Content="Search"
      Click="btnSearch_Click" />
```

10)Locate the comment "<!–Results-->" and replace it with the XAML markup below. *This step adds a TextBlock to display a status message and a RadCalendar named "calResults". The RadCalendar is marked read-only and invisible. Setting the Columns property to "3", displays three consecutive months in the calendar. Setting the ViewsHeaderVisibility property to "Visible" displays month titles above each calendar. HeaderVisibility set to "Collapsed" hides the top title bar graphic. Notice that SelectionMode is set to "Multiple".*

XAML

```
<!--Results -->
<TextBlock x:Name="tbResults"
      Style="{StaticResource HeadingStyle}" />

<telerik:RadCalendar x:Name="calResults"
      Margin="0,10,0,0"
      telerik:StyleManager.Theme="Vista"
      IsReadOnly="True" IsTodayHighlighted="False"
      Columns="3" Rows="1"
      ViewsHeaderVisibility="Visible"
      HeaderVisibility="Collapsed"
      SelectionMode="Multiple">
</telerik:RadCalendar>
```

Code Behind

1) In the code-behind for the page, add a private Random member.

VB

```
Dim _random As New Random()
```

C#

```
Random _random = new Random();
```

2) In the code-behind for the page, navigate to the Loaded event handler and add the code below.

Set the date picker selections to "Tomorrow" and the following week. Call the UpdateCalendar() and SetFocus() methods. Both methods will be written later.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' set the date pickers pairs to start tomorrow and
  ' end the following week.
  dpStart.SelectedDate = DateTime.Today.AddDays(1)
  dpEnd.SelectedDate = DateTime.Today.AddDays(8)
  ' set the displayable and selected dates
  UpdateCalendar()
  ' put focus on the "airport code" text box
  SetFocus(tbAirportCode)
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   // set the date pickers pairs to start tomorrow and
   // end the following week.
   dpStart.SelectedValue = DateTime.Today.AddDays(1);
   dpEnd.SelectedValue = DateTime.Today.AddDays(8);
   // set the displayable and selected dates
   UpdateCalendar();
   // put focus on the "airport code" text box
   SetFocus(tbAirportCode);
}
```

3) Add a helper method to set the focus on the "Airport Code" text box.

Note: This work-around to set focus in Silverlight was discussed briefly in the Input Controls chapter, Control Details section.

```
Private Sub SetFocus(ByVal control As Control)
  ' get focus inside the Silverlight plugin
  System.Windows.Browser.HtmlPage.Plugin.Invoke("focus")
  ' queue this call to occur after the plugin focus
  Dispatcher.BeginInvoke(Function() control.Focus())
End Sub
```

C#

```
private void SetFocus(Control control)
{
   // get focus inside the Silverlight plugin
   System.Windows.Browser.HtmlPage.Plugin.Invoke("focus");
   // queue this call to occur after the plugin focus
   Dispatcher.BeginInvoke(delegate() { control.Focus(); });
}
```

4) In the search button's Click event handler, add the code below.

This code looks at the window of available dates in the calendar and selects up to five of them randomly. To do this, first calculate the number of days between the date picker "Start" and "End" selected dates.

***Note**: Operator overloads of the DateTime class allow you to subtract one DateTime from another. The result of the operation is a TimeSpan type.*

Call the RadCalendar SelectedDates Clear() method to remove any existing collection members. Then cycle through a "For" loop and add DateTime instances to the SelectedDates collection. Use the Random class Next() method to get a number between 1 and the number of selectable days

VB

```
Private Sub btnSearch_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' calculate the number of days available between start and end
  Dim difference As Nullable(Of TimeSpan) = _
dpEnd.SelectedDate - dpStart.SelectedDate
  ' Add random selected dates
  calResults.SelectedDates.Clear()
  For i As Integer = 0 To 4
    Dim dayNumber As Integer = _random.Next(1, difference.Value.Days)
    Dim dateTime As DateTime = dpStart.SelectedDate.Value.AddDays(dayNumber)
    If (Not calResults.SelectedDates.Contains(dateTime)) Then
      calResults.SelectedDates.Add(dateTime)
    End If
  Next i

  ' display the number of selected dates
  tbResults.Text = String.Format( _
"{0} flights are available for {1} on selected dates.", _
calResults.SelectedDates.Count, tbAirportCode.Value.ToString())
End Sub
```

```
private void btnSearch_Click(object sender, RoutedEventArgs e)
{
   // calculate the number of days available between start and end
   TimeSpan? difference = dpEnd.SelectedDate - dpStart.SelectedDate;
   // Add random selected dates
   calResults.SelectedDates.Clear();
   for (int i = 0; i < 5; i++)
   {
      int dayNumber = _random.Next(1, difference.Value.Days);
      DateTime dateTime = dpStart.SelectedDate.Value.AddDays(dayNumber);
      if (!calResults.SelectedDates.Contains(dateTime))
      {
         calResults.SelectedDates.Add(dateTime);
      }
   }

   // display the number of selected dates
   tbResults.Text =
      String.Format("{0} flights are available for {1} on selected dates.",
      calResults.SelectedDates.Count, tbAirportCode.Value.ToString());
}
```

5) Both RadDatePicker controls share a common SelectionChanged event handler named "DateChanged". Create the event handler and call UpdateCalendar(), as shown below.

VB

```
Private Sub DateChanged(ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  UpdateCalendar()
End Sub
```

C#

```
private void DateChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   UpdateCalendar();
}
```

6) Use the selected dates from the two date picker controls to calculate both the "displayable" start and end dates and the "selectable" start and end dates. To do this, first set the SelectableDateStart and SelectableDateEnd from the corresponding date picker values, i.e. dpStart.SelectedDate.Value and dpEnd.SelectedDate.Value. The RadCalendar DisplayDateStart and DisplayDateEnd should range from the month previous to the date picker start date, to month-end of the date picker end date.

```
Private Sub UpdateCalendar()
 If dpStart.SelectedDate IsNot Nothing AndAlso dpEnd.SelectedDate IsNot Nothing Then
   Dim dtStart As DateTime = dpStart.SelectedDate.Value
   Dim dtEnd As DateTime = dpEnd.SelectedDate.Value

   ' displayable dates
   calResults.DisplayDateStart = New DateTime(dtStart.Year, dtStart.Month, 1)
   calResults.DisplayDateEnd = New DateTime(dtEnd.AddMonths(2).Year, dtEnd.AddMonths(2).Month, GetLa

   ' selectable dates
   calResults.SelectableDateStart = dtStart
   calResults.SelectableDateEnd = dtEnd
End If
End Sub
```

```
private void UpdateCalendar()
{
 if (dpStart.SelectedDate != null && dpEnd.SelectedDate != null)
 {
    DateTime dtStart = dpStart.SelectedDate.Value;
    DateTime dtEnd = dpEnd.SelectedDate.Value;

    // displayable dates
    calResults.DisplayDateStart =
        new DateTime(dtStart.Year, dtStart.Month, 1);
    calResults.DisplayDateEnd =
        new DateTime(dtEnd.AddMonths(2).Year, dtEnd.AddMonths(2).Month,
GetLastDayOfMonth(dtEnd.AddMonths(2)));

    // selectable dates
    calResults.SelectableDateStart = dtStart;
    calResults.SelectableDateEnd = dtEnd;
 }
}
```

7) Add a private method that calculates the last day of the month. This calculation takes a reference DateTime value passed in, moves to the following month, then subtracts all the days of that month to arrive at the last day of the month.

VB

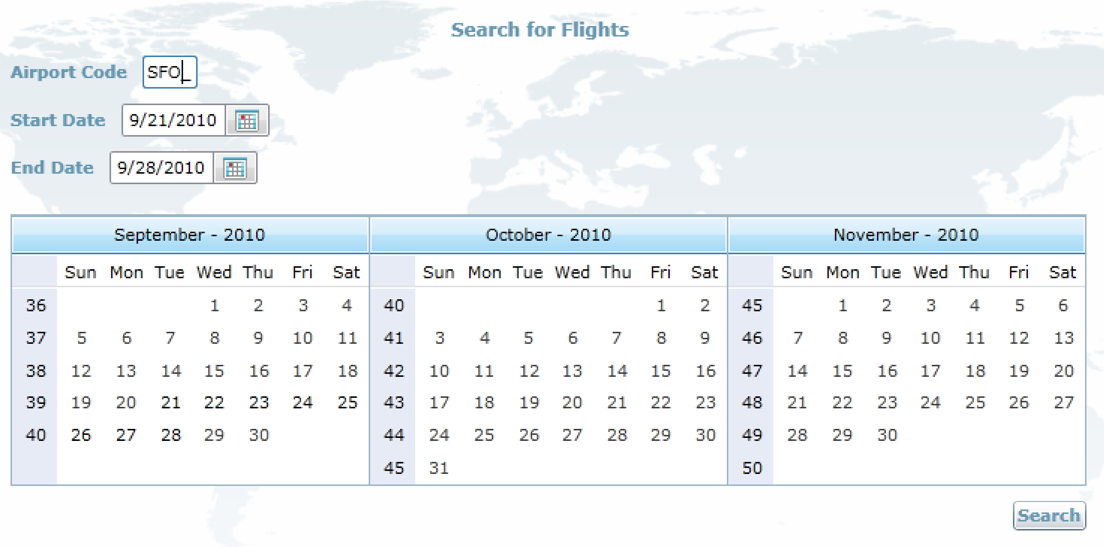
```
Private Function GetLastDayOfMonth(ByVal referenceDate As DateTime) As Integer
  Dim result As DateTime = referenceDate.AddMonths(1)
  result = result.AddDays(-result.Day)
  Return result.Day
End Function
```

C#

```
private int GetLastDayOfMonth(DateTime referenceDate)
{
   DateTime result = referenceDate.AddMonths(1);
   result = result.AddDays(-result.Day);
   return result.Day;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Search for Flights

Airport Code: SFO

Start Date: 9/21/2010

End Date: 9/28/2010

September - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
36				1	2	3	4
37	5	6	7	8	9	10	11
38	12	13	14	15	16	17	18
39	19	20	21	22	23	24	25
40	26	27	28	29	30		

October - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
40						1	2
41	3	4	5	6	7	8	9
42	10	11	12	13	14	15	16
43	17	18	19	20	21	22	23
44	24	25	26	27	28	29	30
45	31						

November - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
45		1	2	3	4	5	6
46	7	8	9	10	11	12	13
47	14	15	16	17	18	19	20
48	21	22	23	24	25	26	27
49	28	29	30				
50							

Test Application Features

- Try selecting different start and end RadDatePicker dates. The calendar should only allow selection between those dates. The displayable dates should range from a month before and the remainder of the month containing the end date. Be aware that there is no safety code to ensure that end dates are greater than start dates.

Ideas for Extending This Example

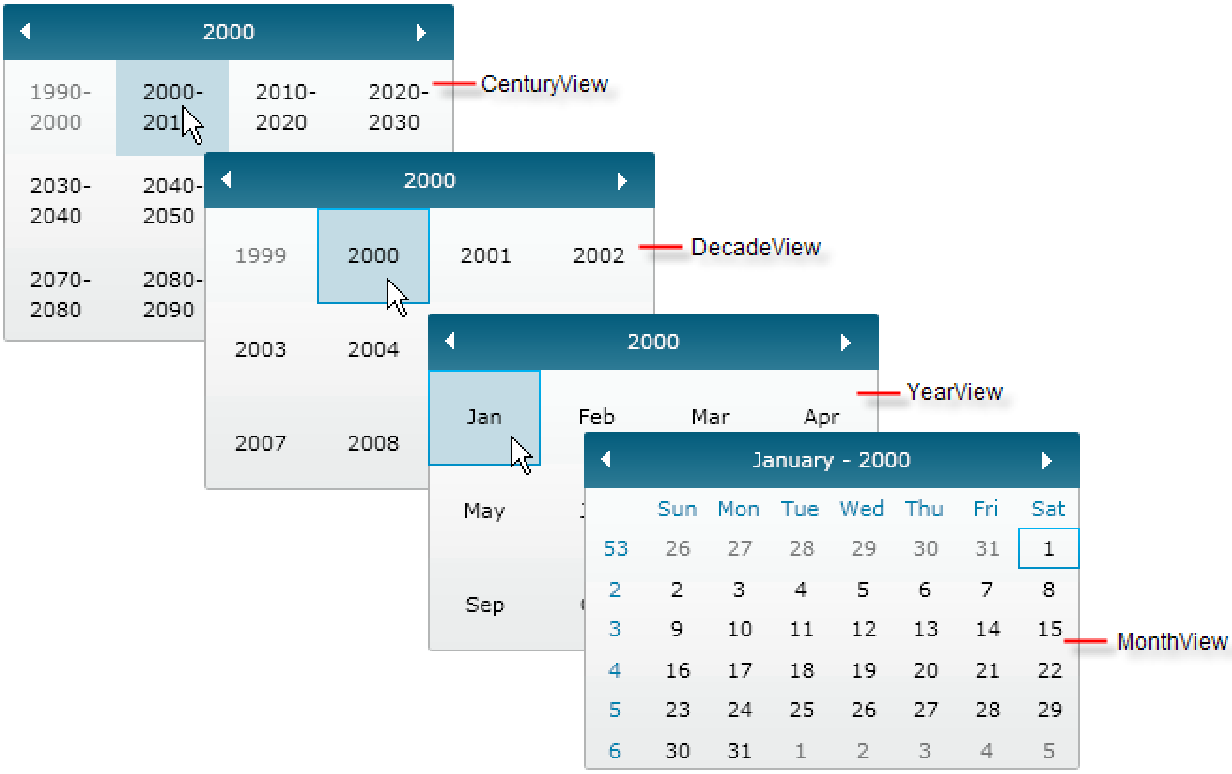
- Prevent the user from entering invalid dates, i.e. where the end date is greater than begin date.

15.4 Control Details

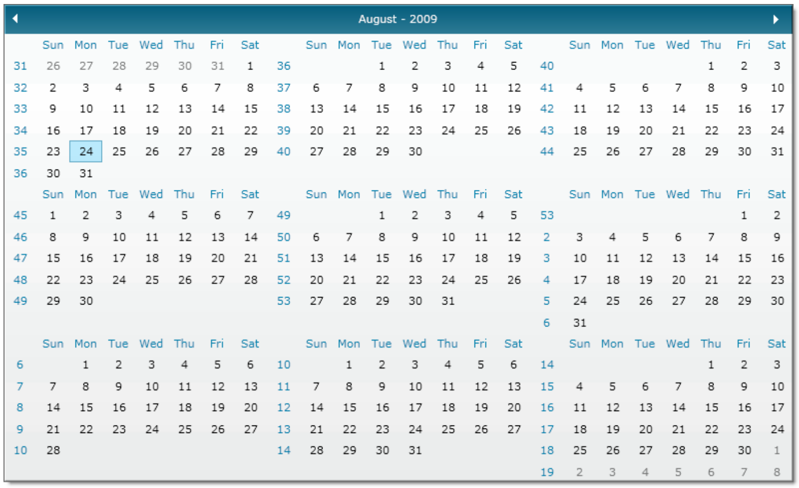
15.4.1 Calendar

Overview

RadCalendar is used for displaying and selecting dates in an easy-to-navigate calendar interface. One of your first choices is to set the **DisplayMode** property to view the calendar by century, decade, year or month (the default). The user can navigate views by clicking a calendar cell. The calendar animates the selected cell expanding to display the next view down in the hierarchy. The screenshot below shows the relationship of views.



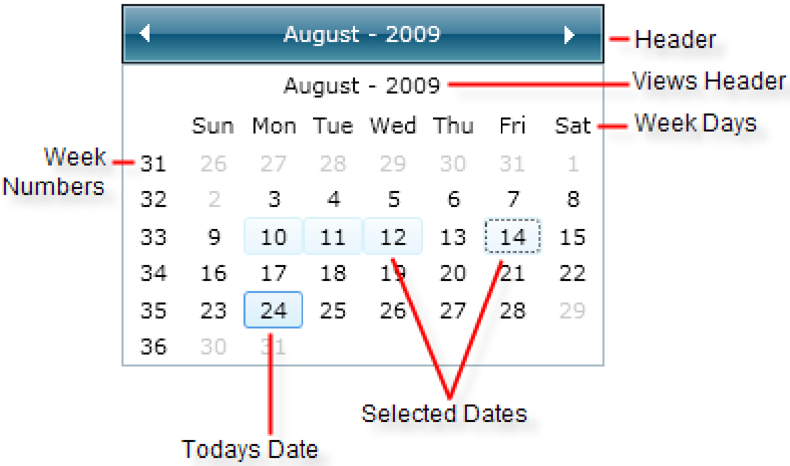
The **Columns** and **Rows** properties control how many instances of the calendar you see displayed down and across the control surface. You can have multiple columns and rows in any of the DisplayModes. The screenshot below shows the calendar in MonthView display mode where both columns and rows are set to "3".



August - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1	36			1	2	3	4	5	40					1	2	3
32	2	3	4	5	6	7	8	37	6	7	8	9	10	11	12	41	4	5	6	7	8	9	10
33	9	10	11	12	13	14	15	38	13	14	15	16	17	18	19	42	11	12	13	14	15	16	17
34	16	17	18	19	20	21	22	39	20	21	22	23	24	25	26	43	18	19	20	21	22	23	24
35	23	24	25	26	27	28	29	40	27	28	29	30				44	25	26	27	28	29	30	31
36	30	31																					
	Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat
45	1	2	3	4	5	6	7	49			1	2	3	4	5	53						1	2
46	8	9	10	11	12	13	14	50	6	7	8	9	10	11	12	2	3	4	5	6	7	8	9
47	15	16	17	18	19	20	21	51	13	14	15	16	17	18	19	3	10	11	12	13	14	15	16
48	22	23	24	25	26	27	28	52	20	21	22	23	24	25	26	4	17	18	19	20	21	22	23
49	29	30						53	27	28	29	30	31			5	24	25	26	27	28	29	30
																6	31						
	Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat		Sun	Mon	Tue	Wed	Thu	Fri	Sat
6		1	2	3	4	5	6	10		1	2	3	4	5	6	14					1	2	3
7	7	8	9	10	11	12	13	11	7	8	9	10	11	12	13	15	4	5	6	7	8	9	10
8	14	15	16	17	18	19	20	12	14	15	16	17	18	19	20	16	11	12	13	14	15	16	17
9	21	22	23	24	25	26	27	13	21	22	23	24	25	26	27	17	18	19	20	21	22	23	24
10	28							14	28	29	30	31				18	25	26	27	28	29	30	1
																19	2	3	4	5	6	7	8

The distinct areas that make up the calendar are shown in this next screenshot. The header, by default, shows a title for the calendar as a whole. The Views Header appears below the header and is most useful when there are more than one column or row. The Week Days are displayed across the top of the dates and the Week Numbers display vertically down the side. Use the **AreWeekNumbersVisible** and **AreWeekNamesVisible** properties to toggle their visibility. Selected dates and "Today's" date each have their own highlight style.



August - 2009

August - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8
33	9	10	11	12	13	14	15
34	16	17	18	19	20	21	22
35	23	24	25	26	27	28	29
36	30	31					

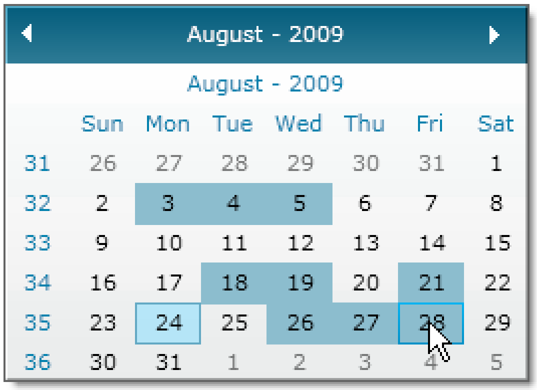
Notes

If you need full featured event calendaring, or behavior that more closely emulates Outlook, be sure to check out the Scheduler chapter.

Date Selection

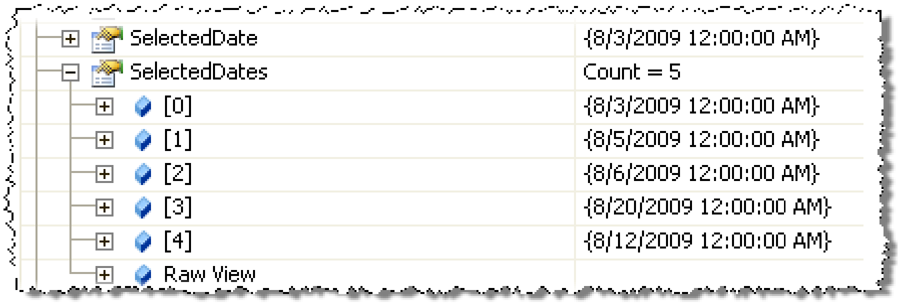
The **SelectionMode** property controls how many dates can be selected at one time and in what manner they can be selected.

- In **Single** SelectionMode (the default), only one date can be selected at a time. Click a date with the mouse or press the space bar to toggle selection. Use the arrow keys to move the selection.
- The **Multiple** SelectionMode allows any number of dates to be selected at one time. Click individual dates with the mouse or press the space bar to toggle selection. Drag with the mouse over a number of dates at one time to toggle a series of dates.
- The **Extended** SelectionMode also allows any number of dates to be selected, but the behavior is similar to Windows Explorer. Holding the Shift key down allows a range of dates to be selected with the mouse or keyboard (using the arrow keys and space bar). The Control key allows individual dates to be selected even when they are not part of a continuous range.



	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8
33	9	10	11	12	13	14	15
34	16	17	18	19	20	21	22
35	23	24	25	26	27	28	29
36	30	31	1	2	3	4	5

There are two properties that tell you what dates are selected: **SelectedDate** and **SelectedDates**. SelectedDate is a nullable DateTime object while SelectedDates is a collection of DateTime. If no dates in the calendar are selected, SelectedDate is null and SelectedDates has a Count = 0. When one or more dates are selected, SelectedDate and SelectedDates[0] are the same value. Elements of SelectedDates appear in the order they are added. The screenshot below shows the Visual Studio QuickWatch window after five dates have been selected in the calendar.



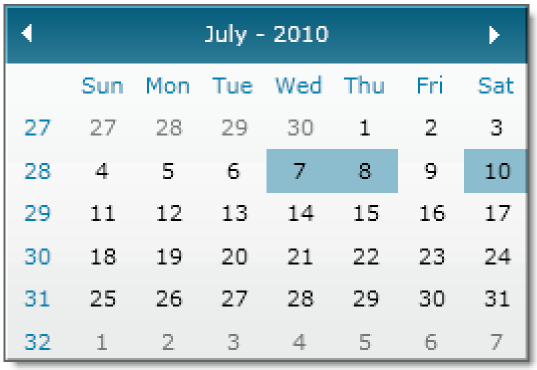
⊞ SelectedDate	{8/3/2009 12:00:00 AM}
⊟ SelectedDates	Count = 5
⊞ [0]	{8/3/2009 12:00:00 AM}
⊞ [1]	{8/5/2009 12:00:00 AM}
⊞ [2]	{8/6/2009 12:00:00 AM}
⊞ [3]	{8/20/2009 12:00:00 AM}
⊞ [4]	{8/12/2009 12:00:00 AM}
⊞ Raw View	

Set new dates in code by assigning a DateTime instance to SelectedDate or add new DateTime instances to the SelectedDates collection:

```
Private Sub UserControl_Loaded(ByVal sender As Object, _
ByVal e As RoutedEventArgs)
  calMain.SelectedDate = New DateTime(2010, 7, 7)
  calMain.SelectedDates.Add(New DateTime(2010, 7, 7))
  calMain.SelectedDates.Add(New DateTime(2010, 7, 8))
  calMain.SelectedDates.Add(New DateTime(2010, 7, 10))
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   calMain.SelectedDate = new DateTime(2010, 7, 7);
   calMain.SelectedDates.Add(new DateTime(2010, 7, 7));
   calMain.SelectedDates.Add(new DateTime(2010, 7, 8));
   calMain.SelectedDates.Add(new DateTime(2010, 7, 10));
}
```

The screenshot below shows the code example running in the browser.



July - 2010

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	27	28	29	30	1	2	3
28	4	5	6	7	8	9	10
29	11	12	13	14	15	16	17
30	18	19	20	21	22	23	24
31	25	26	27	28	29	30	31
32	1	2	3	4	5	6	7

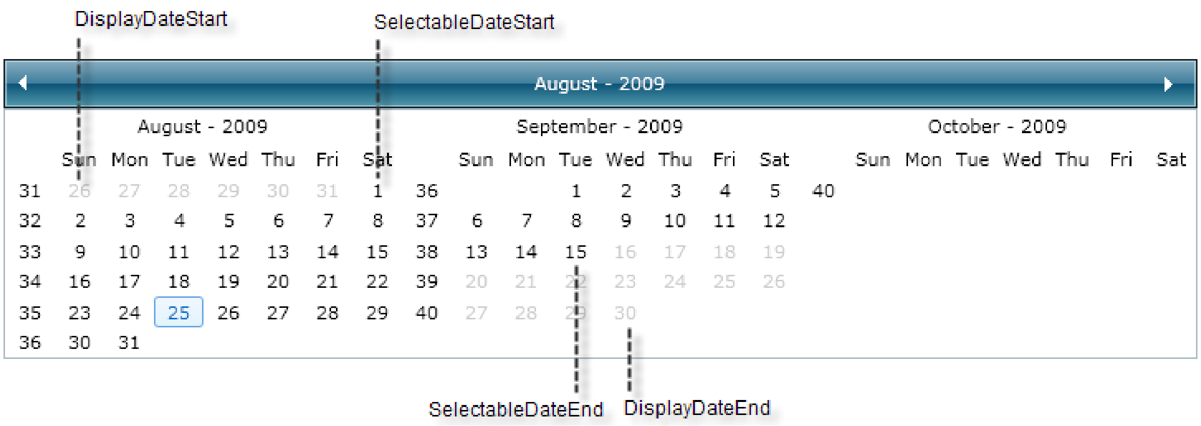
Tip!

You can set the DisplayDate property to navigate the calendar to the month or year that contains this date (depending on the current DisplayMode). Note that the DisplayDate property cannot be set to null.

Constraints

If you need to restrict the range of dates that can be selected, set the **SelectableDateStart** and **SelectableDateEnd** properties. Dates outside this range will be styled to indicate they can not be selected.

To restrict dates that can be browsed (displayed), set the **DisplayDateStart** and **DisplayDateEnd** properties. Dates outside this range will not be visible. The screenshot below shows how these properties play out in a running application. The DisplayDateStart is set to be the first date in the calendar, but is not selectable. August 1st is the SelectableDateStart and September 15th is the SelectableDateEnd. The dates between these two properties show in a darker type style and can be selected by the user. Dates after DisplayDateEnd on September 30 are not visible.



August - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8
33	9	10	11	12	13	14	15
34	16	17	18	19	20	21	22
35	23	24	25	26	27	28	29
36	30	31					

September - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
36			1	2	3	4	5
37	6	7	8	9	10	11	12
38	13	14	15	16	17	18	19
39	20	21	22	23	24	25	26
40	27	28	29	30			

October - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
40							

Tip!

RadCalendar has a set of static methods for checking date validity: IsDisplayDateEndValid(), IsDisplayDateStartValid(), IsDisplayDateValid(), IsSelectableDateEndValid() and IsSelectableDateStartValid(). Each method takes an instance of the RadCalendar to be checked against and the DateTime being checked.

Events

To be notified of date selections by the user, subscribe to the **SelectionChanged** event. You can query either SelectedDate or SelectedDates as shown in the code sample below where a LINQ statement is used to extract a collection of short date strings. The String Join() method is used to make a comma delimited list for display.

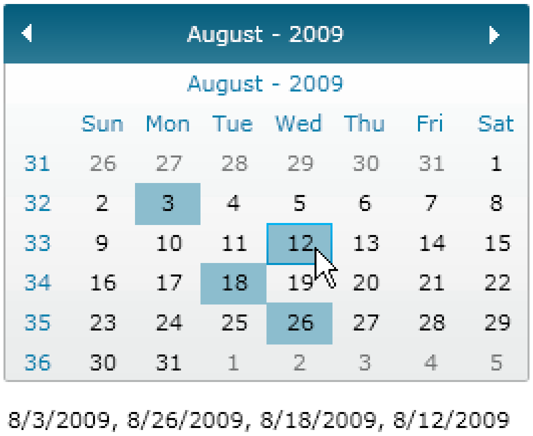
VB

```
Private Sub calMain_SelectionChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  Dim calendar As RadCalendar = (TryCast(sender, RadCalendar))
  If calendar.SelectedDates.Count > 0 Then
    ' convert the collection of dates to short date strings
    Dim dates = _
      From d As DateTime In calendar.SelectedDates _
      Select d.ToShortDateString()

    ' make a comma delimited list and display
    tbStatus.Text = String.Join(", ", dates.ToArray(Of String)())
  End If
End Sub
```

C#

```
private void calMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   RadCalendar calendar = (sender as RadCalendar);
   if (calendar.SelectedDates.Count > 0)
   {
      // convert the collection of dates to short date strings
      var dates =
      from DateTime d in calendar.SelectedDates
      select d.ToShortDateString();

      // make a comma delimited list and display
      tbStatus.Text = String.Join(", ", dates.ToArray<string>());
   }
}
```

The code example running in browser looks like this screenshot:



August - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8
33	9	10	11	12	13	14	15
34	16	17	18	19	20	21	22
35	23	24	25	26	27	28	29
36	30	31	1	2	3	4	5

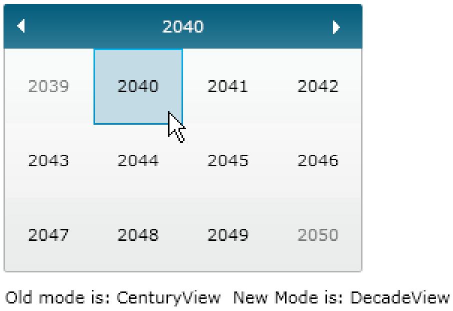
8/3/2009, 8/26/2009, 8/18/2009, 8/12/2009

RadCalendar also surfaces the **DisplayDateChanged** and **DisplayModeChanged** events. Both events pass arguments that contain the new and old DisplayDate and DisplayMode, respectively. The code example below shows a DisplayModeChanged event handler that displays the new and old modes.

```
Private Sub RadCalendar_DisplayModeChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.Calendar.CalendarModeChangedEventArgs)
  If tbStatus IsNot Nothing Then
    tbStatus.Text = _
String.Format("Old mode is: {0}  New Mode is: {1}", _
e.OldMode.ToString(), e.NewMode.ToString())
  End If
End Sub
```

```
private void RadCalendar_DisplayModeChanged(
   object sender,
   Telerik.Windows.Controls.Calendar.CalendarModeChangedEventArgs e)
{
   if (tbStatus != null)
   {
      tbStatus.Text =
         String.Format("Old mode is: {0}  New Mode is: {1}",
         e.OldMode.ToString(), e.NewMode.ToString());
   }
}
```

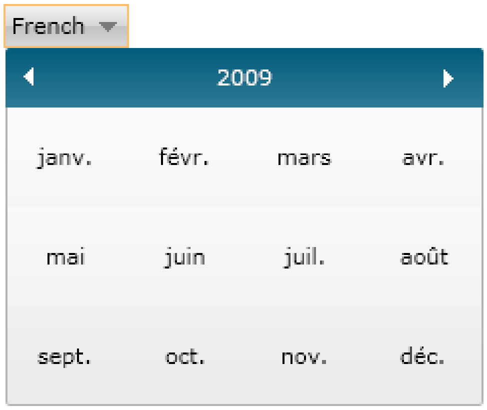
The screenshot below shows the code sample running in the browser:



Old mode is: CenturyView New Mode is: DecadeView

Internationalization

To internationalize the calendar, assign a new **CultureInfo** instance to the **Culture** property. The screenshot below shows the result of adding a number of culture codes to a combo box and using the selected culture code to create a CultureInfo.



The XAML that defines the combo box stores the culture codes, e.g. "en-us" for United States English, in the combo box item's Tag property...

```
<telerik:RadComboBox
      SelectionChanged="RadComboBox_SelectionChanged"
      HorizontalAlignment="Left" SelectedIndex="0">
   <telerik:RadComboBoxItem Content="US English"
         Tag="en-us" />
   <telerik:RadComboBoxItem Content="French"
         Tag="fr-FR" />
   <telerik:RadComboBoxItem Content="German"
         Tag="de-DE" />
</telerik:RadComboBox>
```

When a combo box item is clicked, the culture code is retrieved from the tag and fed to the CultureInfo constructor. The CultureInfo is assigned to the RadCalendar Culture property and the calendar displays in that language.

VB

```
Private Sub RadComboBox_SelectionChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  If calMain IsNot Nothing Then
    Dim item As RadComboBoxItem = _
TryCast(e.AddedItems(0), RadComboBoxItem)
    calMain.Culture = _
New System.Globalization.CultureInfo(item.Tag.ToString())
  End If
End Sub
```

C#

```
private void RadComboBox_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   if (calMain != null)
   {
      RadComboBoxItem item = e.AddedItems[0] as RadComboBoxItem;
      calMain.Culture =
         new System.Globalization.CultureInfo(item.Tag.ToString());
   }
}
```

Culture can also be set directly in the XAML:

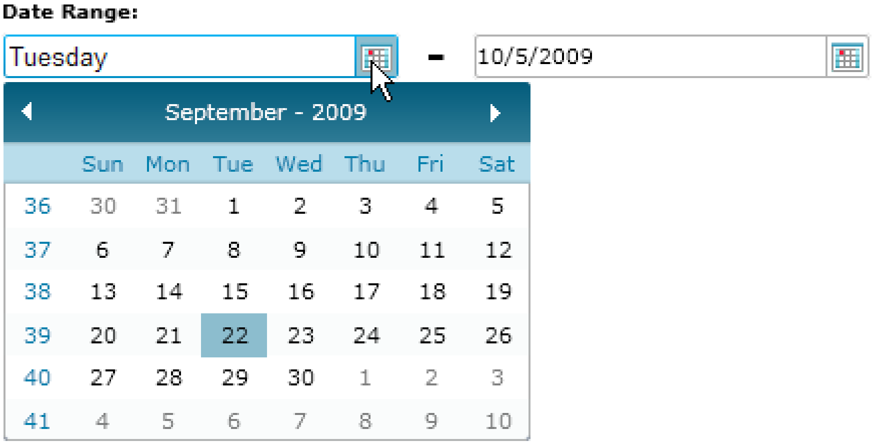
XAML

```
<telerik:RadCalendar Culture="fr-FR" />
```

15.4.2 Date Picker

Overview

If you need a calendar that takes less space, use the RadDatePicker control. RadDatePicker is essentially a text box / parser with a drop down button that displays a calendar. The advanced parsing feature allow you to enter any number or string in the input field and the entered value will be transformed to a valid date. For example, if you type “1”, the first day of the month will be shown after leaving the input field. If you type “Monday”, the corresponding date of the current week will be selected.



Date Range:

Tuesday - 10/5/2009

September - 2009

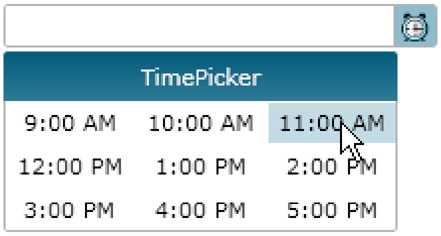
	Sun	Mon	Tue	Wed	Thu	Fri	Sat
36	30	31	1	2	3	4	5
37	6	7	8	9	10	11	12
38	13	14	15	16	17	18	19
39	20	21	22	23	24	25	26
40	27	28	29	30	1	2	3
41	4	5	6	7	8	9	10

RadDatePicker properties, for the most part, simply carry over from RadCalendar. One exception to this rule is the **IsDropDownOpen** property that reflects when the calendar portion of the control is showing.

15.4.3 Time Picker

Overview

Use RadTimePicker to select from a set of times displayed in a drop down list. The range of times displayed are bounded by the **StartTime** and **EndTime** properties. The **TimeInterval** property determines the number of minutes between selectable entries. The example below displays the hours between 9AM and 5PM with selections spaced 1 hour apart.



The XAML and code below show how to set StartTime, EndTime and TimeInterval properties. In XAML, the values are set to military (24 hour) format.

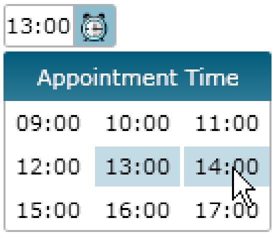
```
<telerik:RadTimePicker x:Name="tpMain" Margin="10"
    telerik:StyleManager.Theme="Summer"
    StartTime="09:00:00" EndTime="17:00:00"
    TimeInterval="01:00:00">
</telerik:RadTimePicker>
```

StartTime, EndTime and TimeInterval properties are all instances of TimeSpan. The code below produces equivalent results to the prior XAML example.

```
tpMain.StartTime = New TimeSpan(9, 0, 0)
tpMain.EndTime = New TimeSpan(17, 0, 0)
tpMain.TimeInterval = New TimeSpan(1, 0, 0)
```

```
tpMain.StartTime = new TimeSpan(9, 0, 0);
tpMain.EndTime = new TimeSpan(17, 0, 0);
tpMain.TimeInterval = new TimeSpan(1, 0, 0);
```

Use the **HeaderContent** property to place text or other content into the time picker header. In the screenshot below, HeaderContent has been assigned the simple text "Appointment Time".



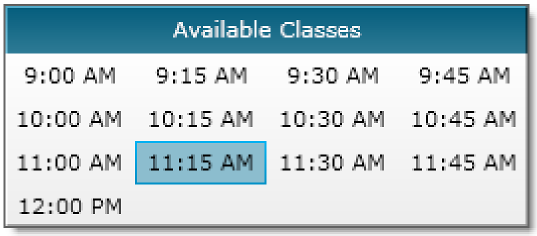
13:00

Appointment Time

09:00	10:00	11:00
12:00	13:00	14:00
15:00	16:00	17:00

RadClock

RadClock displays the grid of times without the input and button portions of a RadDatePicker. RadClock is to RadTimePicker as RadCalendar is to RadDatePicker. It simply displays all the times at once without having to click the drop down button. Set the heading content by assigning the **Header** property. In the screenshot below, Header is set to text "Available Classes".

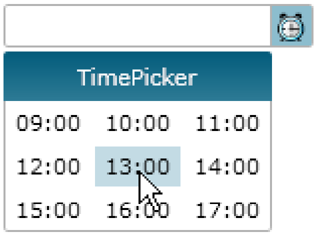


Available Classes

9:00 AM	9:15 AM	9:30 AM	9:45 AM
10:00 AM	10:15 AM	10:30 AM	10:45 AM
11:00 AM	11:15 AM	11:30 AM	11:45 AM
12:00 PM			

Internationalization

To internationalize RadTimePicker, assign the **Culture** property. For example, the RadTimePicker in the screenshot below displays time in military format because the culture is set to "en-gb", i.e. "English - Great Britain".



09:00 10:00 11:00

12:00 13:00 14:00

15:00 16:00 17:00

Here's the line of code that sets the culture:

VB

```
tpMain.Culture = New System.Globalization.CultureInfo("en-gb")
```

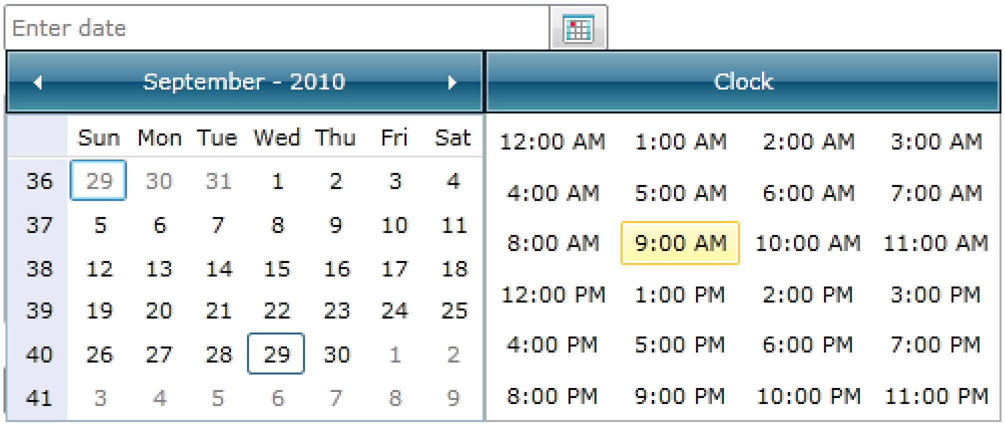
C#

```
tpMain.Culture = new System.Globalization.CultureInfo("en-gb");
```

15.4.4 DateTime Picker

Overview

If you need to select date and time together, use the RadDateTimePicker control. RadDateTimePicker is essentially a text box / parser with a drop down button that displays a calendar and/or a set of times. The advanced parsing feature allow you to enter any number or string in the input field and the entered value will be transformed to a valid date and/or time. DateTimePicker control can be used as date picker, time picker or both together when set InputMode property.



	Sun	Mon	Tue	Wed	Thu	Fri	Sat
36	29	30	31	1	2	3	4
37	5	6	7	8	9	10	11
38	12	13	14	15	16	17	18
39	19	20	21	22	23	24	25
40	26	27	28	29	30	1	2
41	3	4	5	6	7	8	9

12:00 AM	1:00 AM	2:00 AM	3:00 AM
4:00 AM	5:00 AM	6:00 AM	7:00 AM
8:00 AM	9:00 AM	10:00 AM	11:00 AM
12:00 PM	1:00 PM	2:00 PM	3:00 PM
4:00 PM	5:00 PM	6:00 PM	7:00 PM
8:00 PM	9:00 PM	10:00 PM	11:00 PM

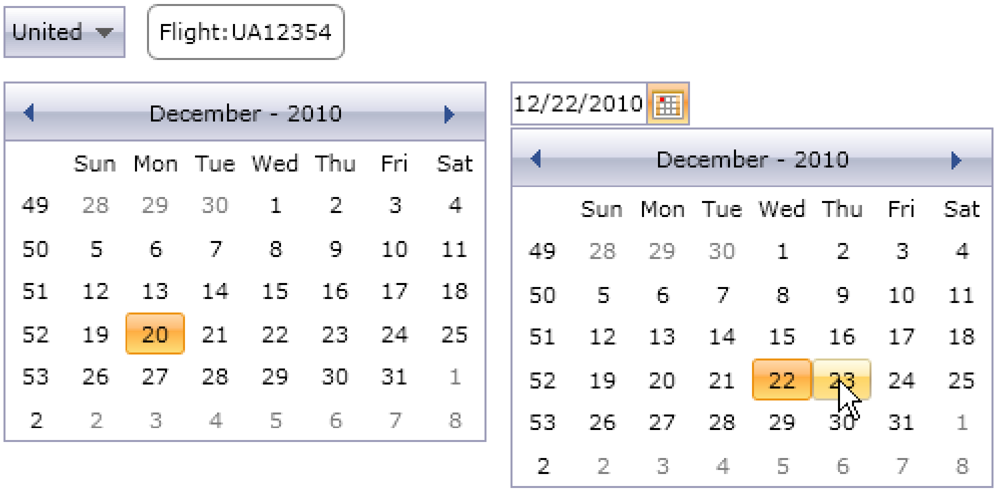
RadDateTimePicker properties, for the most part, simply put together properties from RadDatePicker and RadTimePicker. One exception to this rule is the **SelectedDate** and **SelectedTime** properties are combined in **SelectedValue** property.

15.5 Binding

Calendar and Date Picker Binding

How do I bind to the calendar and date picker controls?

The preferred approach is to use the MVVM pattern (see the Data Binding for more information on MVVM and binding in general). Your custom object exposes a property that is bound to the SelectedDate property of RadDatePicker. You can use TwoWay binding so that user selections in the calendar are propagated back to the object. The example below demonstrates binding to a RadComboBox, TextBlock, RadCalendar and RadDatePicker controls. The example has a list of "Flight" objects that have fields for "Airline", "Flight Number", "Departure Date" and "Arrival Date". The RadComboBox contains the entire list of flights. when a flight is selected from the list, the DataContext for the other objects is set to match the currently selected flight object.



	Sun	Mon	Tue	Wed	Thu	Fri	Sat
49	28	29	30	1	2	3	4
50	5	6	7	8	9	10	11
51	12	13	14	15	16	17	18
52	19	20	21	22	23	24	25
53	26	27	28	29	30	31	1
2	2	3	4	5	6	7	8

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
49	28	29	30	1	2	3	4
50	5	6	7	8	9	10	11
51	12	13	14	15	16	17	18
52	19	20	21	22	23	24	25
53	26	27	28	29	30	31	1
2	2	3	4	5	6	7	8

Tip!

The SelectedDates property is an observable collection of all the selected dates. Although it is exposed as a list, you can bind it and cast it to an observable collection if you need. Note that the SelectedDates property in the case of a Single selection mode will also contain the currently selected date, but it will be read-only.

The basic binding steps are a) build a custom "View Model" object, b) change the XAML to bind specific properties, c) add code behind. The code will populate a list of objects and set the data context for each control that needs to be bound.

For the TwoWay binding to work, your custom object needs to implement the **INotifyPropertyChanged** interface. The "Flight" object below handles firing notification if any of the properties are changed.

```
Public Class Flight
  Implements INotifyPropertyChanged
```

```
Private _airline As String

Public Property Airline() As String
  Get
    Return _airline
  End Get
  Set(ByVal value As String)
    If _airline <> value Then
      _airline = value
      OnPropertyChanged("Airline")
    End If
  End Set
End Property

Private _number As String

Public Property Number() As String
  Get
    Return _number
  End Get
  Set(ByVal value As String)
    If _number <> value Then
      _number = value
      OnPropertyChanged("Number")
    End If
  End Set
End Property

Private _departure As DateTime

Public Property Departure() As DateTime
  Get
    Return _departure
  End Get
  Set(ByVal value As DateTime)
    If _departure <> value Then
      _departure = value
      OnPropertyChanged("Departure")
    End If
  End Set
End Property

Private _arrival As DateTime

Public Property Arrival() As DateTime
  Get
    Return _arrival
  End Get
  Set(ByVal value As DateTime)
    If _arrival <> value Then
      _arrival = value
      OnPropertyChanged("Arrival")
    End If
  End Set
End Property
```

```
  Protected Overridable Sub OnPropertyChanged(ByVal propertyName As String)
    RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
  End Sub

  Public Event PropertyChanged As PropertyChangedEventHandler

End Class
```

C#

```
public class Flight : INotifyPropertyChanged
{

   private string _airline;

   public string Airline
   {
      get
      {
         return _airline;
      }
      set
      {
         if (_airline != value)
         {
            _airline = value;
            OnPropertyChanged("Airline");
         }
      }
   }

   private string _number;

   public string Number
   {
      get
      {
         return _number;
      }
      set
      {
         if (_number != value)
         {
            _number = value;
            OnPropertyChanged("Number");
         }
      }
   }

   private DateTime _departure;

   public DateTime Departure
   {
        get
        {
          return _departure;
        }
        set
        {
          if (_departure != value)
          {
            _departure = value;
            OnPropertyChanged("Departure");
          }
        }
      }

      private DateTime _arrival;

      public DateTime Arrival
      {
        get
        {
          return _arrival;
        }
        set
        {
          if (_arrival != value)
          {
            _arrival = value;
            OnPropertyChanged("Arrival");
          }
        }
      }

      protected virtual void OnPropertyChanged(String propertyName)
      {
        if (PropertyChanged != null)
        {
          PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
      }

      public event PropertyChangedEventHandler PropertyChanged;

    }
```

The XAML takes care of hooking up specific properties to the data using binding syntax, e.g. "{Binding MyProperty, Mode=TwoWay}".

```
<StackPanel Orientation="Vertical">
  <StackPanel Orientation="Horizontal">
    <telerik:RadComboBox x:Name="cbMain" Margin="5"
        telerik:StyleManager.Theme="Office_Silver"
        HorizontalAlignment="Left"
        SelectionChanged="cbMain_SelectionChanged" />
    <Border BorderBrush="Gray" BorderThickness="1"
        CornerRadius="5" HorizontalAlignment="Left"
        Padding="5" Margin="5">
      <StackPanel Orientation="Horizontal">
        <TextBlock Text="Flight:" />
        <TextBlock x:Name="tbFlightNumber"
            Text="{Binding Number, Mode=OneWay}" />
      </StackPanel>
    </Border>
  </StackPanel>

  <StackPanel Orientation="Horizontal">
    <telerik:RadCalendar x:Name="calMain" Margin="5"
        telerik:StyleManager.Theme="Office_Silver"
        SelectedDate="{Binding Departure, Mode=TwoWay}"
        HorizontalAlignment="Left" />
    <telerik:RadDatePicker x:Name="dpMain" Margin="5"
        telerik:StyleManager.Theme="Office_Silver"
        SelectedValue="{Binding Arrival, Mode=TwoWay}"
        HorizontalAlignment="Left" VerticalAlignment="Top" />
  </StackPanel>
```

The remainder of the work takes place in the code-behind. The code first populates an ObservableCollection with a series of new Flight objects. This list is assigned to the combo box ItemsSource property and the DisplayMemberPath is set to "Airline". When the SelectionChanged event fires, the selected item is cast back to a Flight object type and assigned as the DataContext for all the other controls, i.e. calendar, date picker and text block.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' populate a collection of Flight objects
  Dim flights As ObservableCollection(Of Flight) = _
New ObservableCollection(Of Flight)()
  flights.Add(New Flight() With { _
.Airline = "United", _
.Number = "UA12354", _
.Departure = New DateTime(2010, 12, 20, 9, 20, 0), _
.Arrival = New DateTime(2010, 12, 22, 11, 15, 0)})
  flights.Add(New Flight() With { _
.Airline = "Japan Airlines", _
.Number = "JP4533", _
.Departure = New DateTime(2010, 12, 21, 20, 0, 0), _
.Arrival = New DateTime(2010, 12, 25, 21, 32, 0)})
  flights.Add(New Flight() With { _
.Airline = "Southwest", _
.Number = "SW55908", _
.Departure = New DateTime(2010, 12, 23, 2, 12, 0), _
.Arrival = New DateTime(2010, 12, 24, 8, 24, 0)})

  ' assign the list of flights to the combo box
  cbMain.ItemsSource = flights
  ' display the Airline property in the combo box text
  cbMain.DisplayMemberPath = "Airline"
  ' select the first item
  cbMain.SelectedIndex = 0
End Sub

Private Sub cbMain_SelectionChanged(ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  ' get the flight object from the currently selected combo box item
  Dim flight As Flight = TryCast(cbMain.SelectedItem, Flight)

  ' assign the flight as the DataContext of the calendar,
  ' date picker and text block
  calMain.DataContext = flight
  calMain.DisplayDate = flight.Departure
  dpMain.DataContext = flight
  tbFlightNumber.DataContext = flight
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   // populate a collection of Flight objects
   ObservableCollection<Flight> flights =
      new ObservableCollection<Flight>();
   flights.Add(new Flight()
   {
      Airline = "United",
      Number = "UA12354",
      Departure = new DateTime(2010, 12, 20, 9, 20, 0),
      Arrival = new DateTime(2010, 12, 22, 11, 15, 0),
   });
   flights.Add(new Flight()
   {
      Airline = "Japan Airlines",
      Number = "JP4533",
      Departure = new DateTime(2010, 12, 21, 20, 0, 0),
      Arrival = new DateTime(2010, 12, 25, 21, 32, 0),
   });
   flights.Add(new Flight()
   {
      Airline = "Southwest",
      Number = "SW55908",
      Departure = new DateTime(2010, 12, 23, 2, 12, 0),
      Arrival = new DateTime(2010, 12, 24, 8, 24, 0),
   });

   // assign the list of flights to the combo box
   cbMain.ItemsSource = flights;
   // display the Airline property in the combo box text
   cbMain.DisplayMemberPath = "Airline";
   // select the first item
   cbMain.SelectedIndex = 0;
}

private void cbMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   // get the flight object from the currently selected combo box item
   Flight flight = cbMain.SelectedItem as Flight;

   // assign the flight as the DataContext of the calendar,
   // date picker and text block
   calMain.DataContext = flight;
   calMain.DisplayDate = flight.Departure;
   dpMain.DataContext = flight;
   tbFlightNumber.DataContext = flight;
}
```

Time Picker Binding

To bind to time picker controls, first create an ObservableCollection of TimeSpan objects and assign the collection to the RadTimePicker **ClockItemSource** property. The snippet below shows a simple example.

VB

```
Dim classTimes As ObservableCollection(Of TimeSpan) = _
New ObservableCollection(Of TimeSpan) (New TimeSpan() {New TimeSpan(9, 0, 0), _
New TimeSpan(10, 0, 0), New TimeSpan(11, 0, 0), New TimeSpan(12, 0, 0)})
tpMain.ClockItemSource = classTimes
```

C#

```
ObservableCollection<TimeSpan> classTimes =
   new ObservableCollection<TimeSpan>()
{
   new TimeSpan(9, 0, 0),
   new TimeSpan(10, 0, 0),
   new TimeSpan(11, 0, 0),
   new TimeSpan(12, 0, 0)
};
tpMain.ClockItemSource = classTimes;
```

15.6 Customization

In this example we will customize the RadCalendar control background to use the "Scoville" set of colors.

"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

1) Run Expression Blend.

2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*

3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight Application" from the right-most list. Enter a unique name for the project and click **OK**.

Edit the Page in Expression Blend

1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.

2) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu.

a) Add a reference to the **Telerik.Windows.Controls.dll** assembly.

b) Add a reference to the **Telerik.Windows.Controls.Input.dll** assembly.

3) From the Project menu select **Build Project**.

4) Add the RadCalendar to the page.

a) Open the Assets pane.

b) On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.

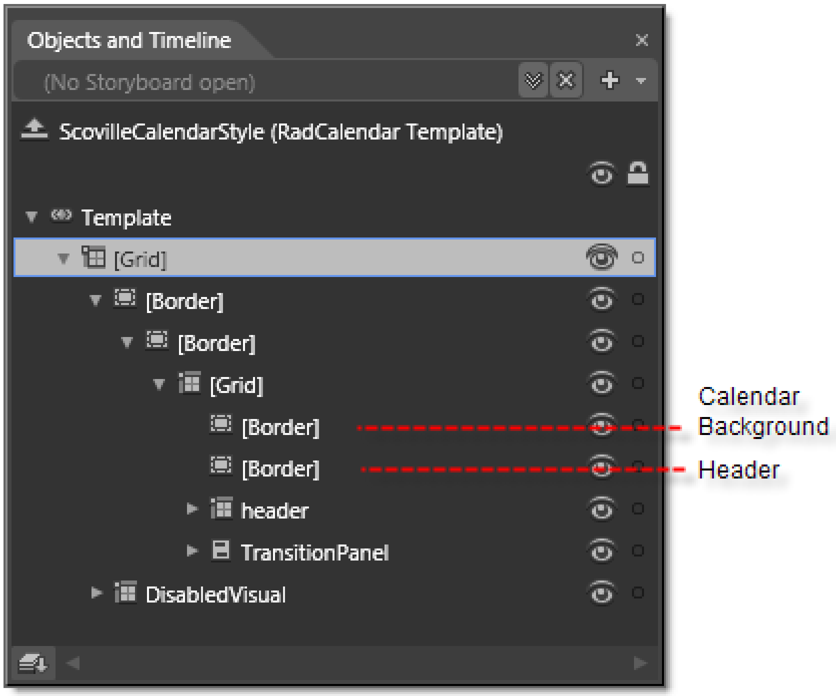
c) In the Assets pane, just above the tree view is the Assets Find entry text box.

d) Type the first few characters of "RadCalendar" into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view.

e) Locate the RadCalendar control and drag it onto the MainPage.xaml Artboard.

5) In the Objects and Timeline pane, right-click "[RadCalendar]" and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleCalendarStyle". Click **OK** to create the style resource and close the dialog.

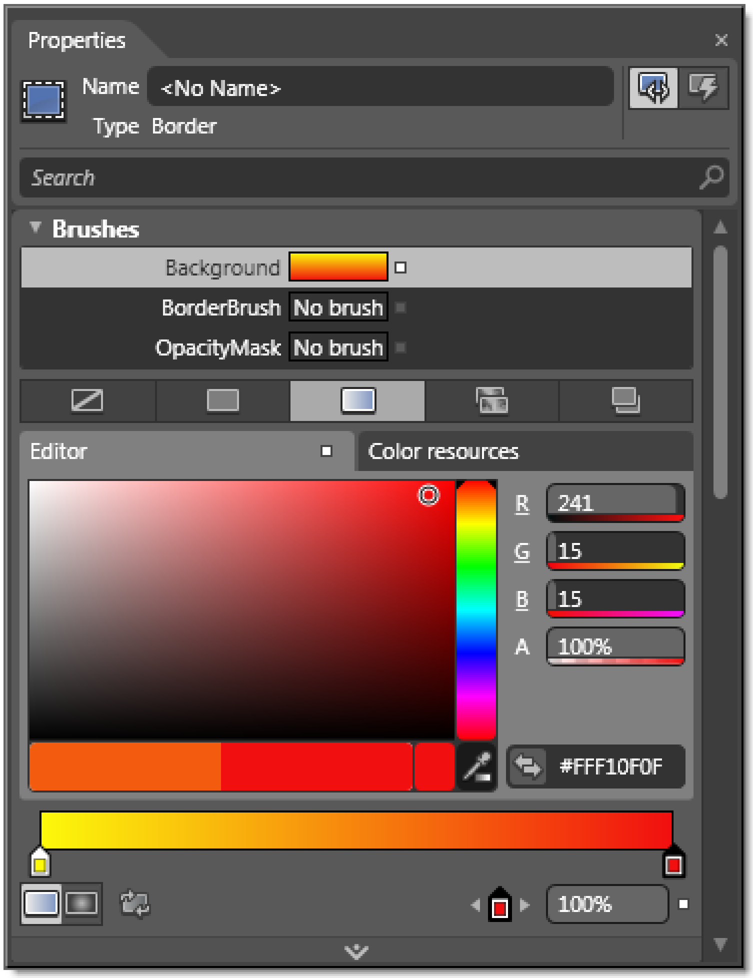
6) In the Objects and Timeline pane, open the element tree and find the top two "[Border]" elements. These two elements represent the calendar background and the calendar header background.



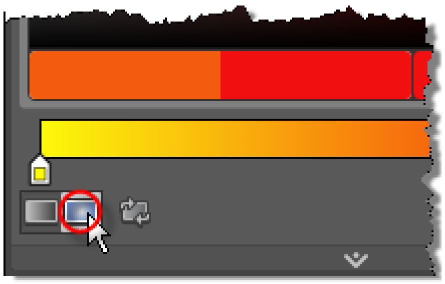
7) Click the top-most "[Border]" element to select it.

8) In the Properties pane, Brushes section, locate the **Background** property. Click the Advanced Property Options button and select **Reset** from the drop down menu.

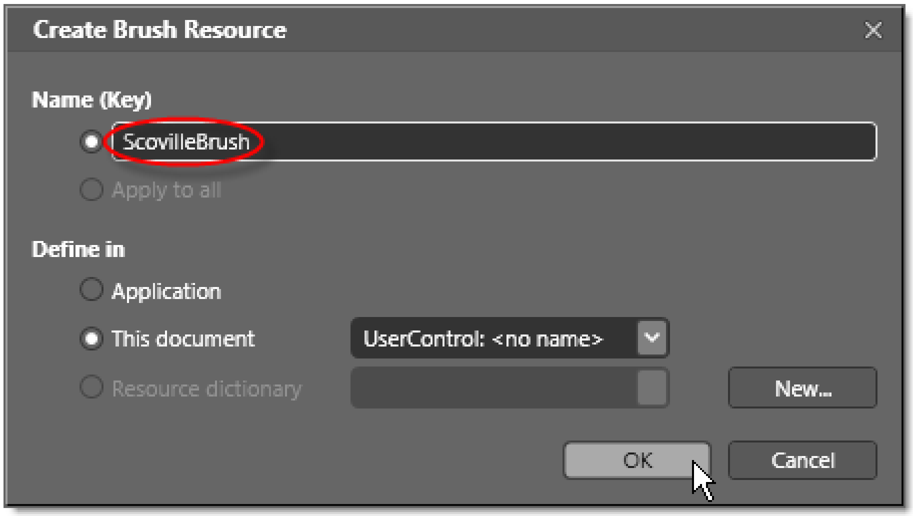
9) Also in the Brushes section of the Properties pane, select the Gradient button. Use the eye dropper tool to replace the black color of the default gradient with yellow and replace the white color with red. The Properties pane at this point should look something like the screenshot below.



10)Select the Radial Gradient button from the lower left hand corner of the Brushes section:



11)Going back to the Properties pane, Brushes section, again locate the **Background** property. Click the Advanced Property Options button and select **Convert to New Resource..** from the drop down menu. The Create Brush Resource dialog will appear. Enter "ScovilleBrush" as the name for the brush. Click the **OK** button to close the dialog and create the brush. *This will allow us to reuse the same brush later in the same project.*



Create Brush Resource

Name (Key)

● ScovilleBrush

○ Apply to all

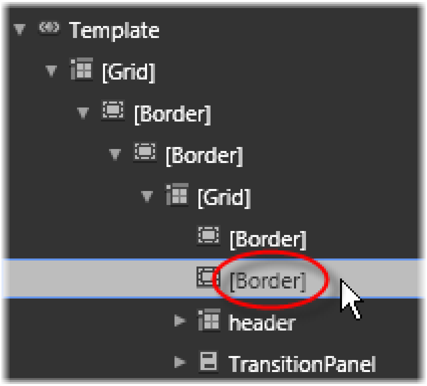
Define in

○ Application

● This document UserControl: <no name>

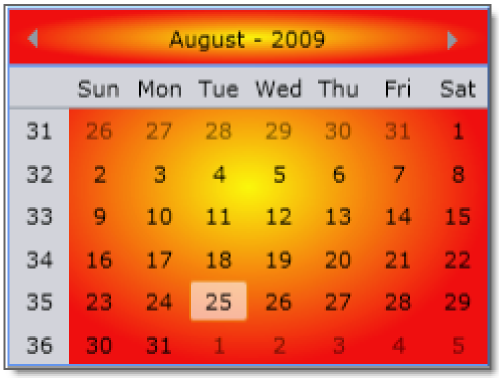
○ Resource dictionary

12)In the Objects and Timeline pane, click the second "[Border]" element to begin editing the header background.



13)In the Properties pane, Brushes section, locate the **Background** property. Click the Advanced Property Options button and select **Local Resource > ScovilleBrush** from the drop down menu.

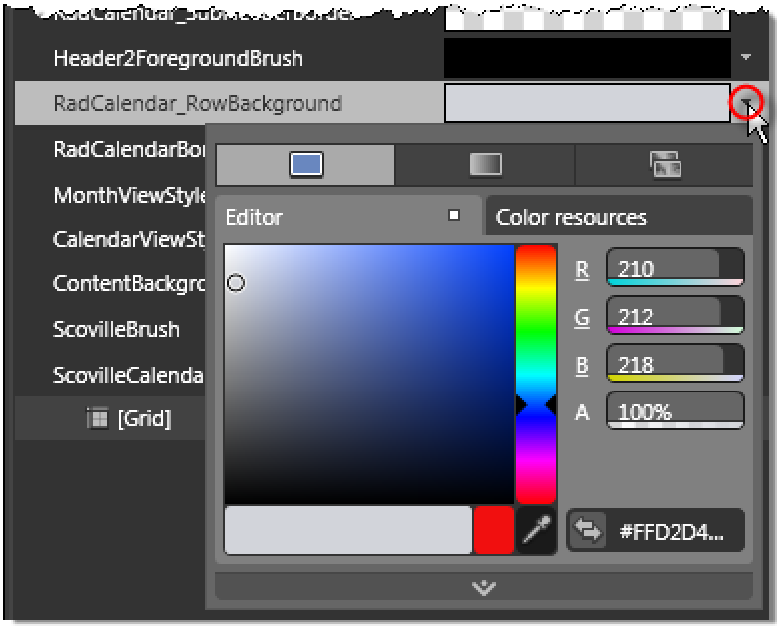
14)Looking to the Artboard, the RadCalendar should look something like the screenshot below.



August - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8
33	9	10	11	12	13	14	15
34	16	17	18	19	20	21	22
35	23	24	25	26	27	28	29
36	30	31	1	2	3	4	5

15)Navigate to the Resources pane, find the "RadCalendar_RowBackground" brush in the list and select it. Click the drop-down arrow to open the popup editor.



Header2ForegroundBrush

RadCalendar_RowBackground

RadCalendarBor

MonthViewStyle

CalendarViewSt

ContentBackgro

ScovilleBrush

ScovilleCalenda

[Grid]

Editor

Color resources

R 210

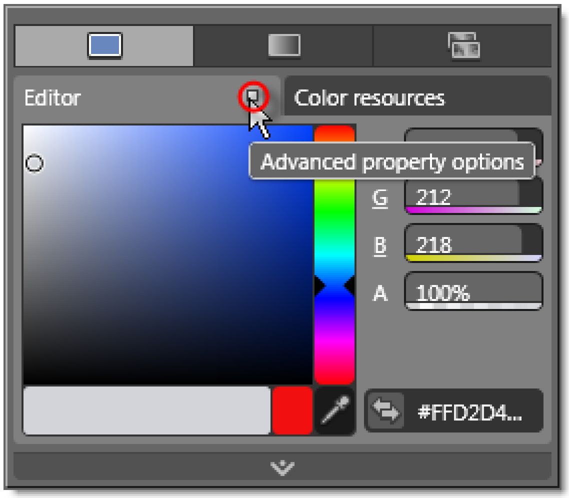
G 212

B 218

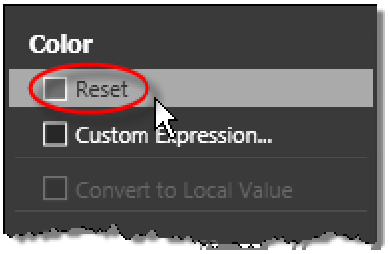
A 100%

#FFD2D4...

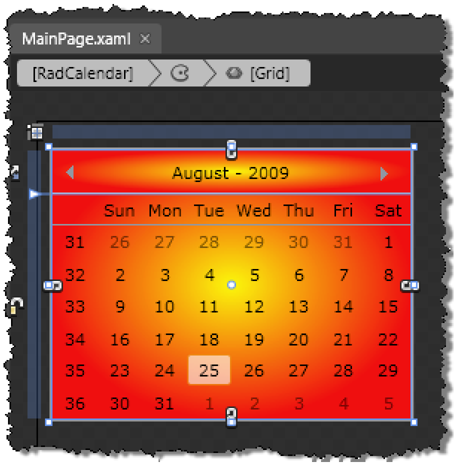
16) In the Editor, locate the Advanced Property Options button and click it.



17)Select **Reset** from the menu. This will set the Background for the rows underneath the column and row headers to Transparent

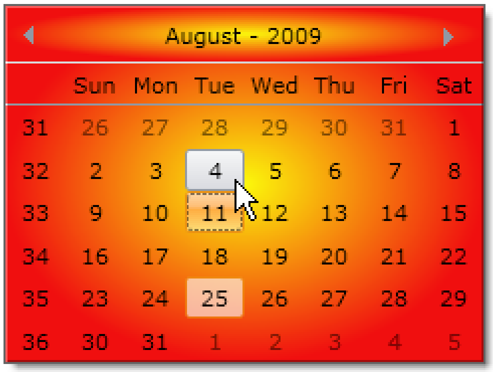


18)Checking the appearance of the calendar in the Artboard, you will see that setting the "RadCalendar_RowBackground" brush to transparent allows the calendar background color to show through.



Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.

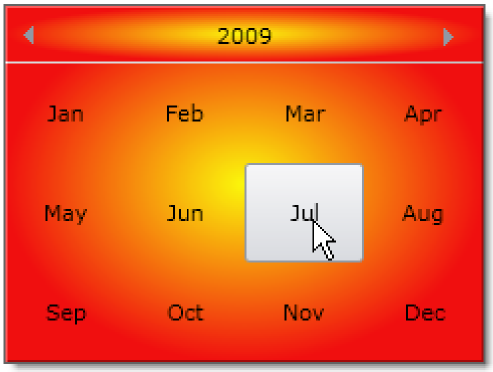


August - 2009

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	26	27	28	29	30	31	1
32	2	3	4	5	6	7	8
33	9	10	11	12	13	14	15
34	16	17	18	19	20	21	22
35	23	24	25	26	27	28	29
36	30	31	1	2	3	4	5

Test Application Features

- Functionally, the calendar should behave as it did before the styling changes were made. You should be able to use the same events, properties and methods and get the same results.
- Verify that the calendar has the same styling in each of the DisplayMode property settings.



2009

Jan	Feb	Mar	Apr
May	Jun	Jul	Aug
Sep	Oct	Nov	Dec

15.7 Wrap Up

In this chapter you learned how to use calendar, date picker, time and datetime picker controls to collect date and time data from the user. In the Control Details section of this chapter, you reviewed the types of date and time controls and the appropriate uses for each control type. You learned how to control the number of dates that can be selected at one time and the manner that they can be selected. You also learned about the SelectedDate and SelectedDates properties for RadDatePicker and SelectedValue property for RadDateTimePicker used to store selections and the SelectionChanged event that flags when its time to look at these properties. You are aware of the properties used to limit the dates that can be selected or viewed. You have also looked at the RadTimePicker and RadClock controls used to select times. In the Binding section of this chapter you saw how calendar and date picker controls are bound to custom objects and also looked at a short example of binding RadTimePicker to a collection of TimeSpan objects. Finally, you customized the background of RadCalendar using Expression Blend.

Part XVI

ComboBox

16 ComboBox

16.1 Objectives

In this chapter you will build a RadComboBox and its items using XAML and in code. You will learn how to use the Content property assignment for simple text lists and how to bind to templates for more complex layout. You will learn how to handle the SelectionChanged event of the combo box and the Selection event of the individual items. In the process, you will find out how to get and set the selected RadComboBoxItem or custom object. You will also have a brief look at events that respond to drop down opening and closing.

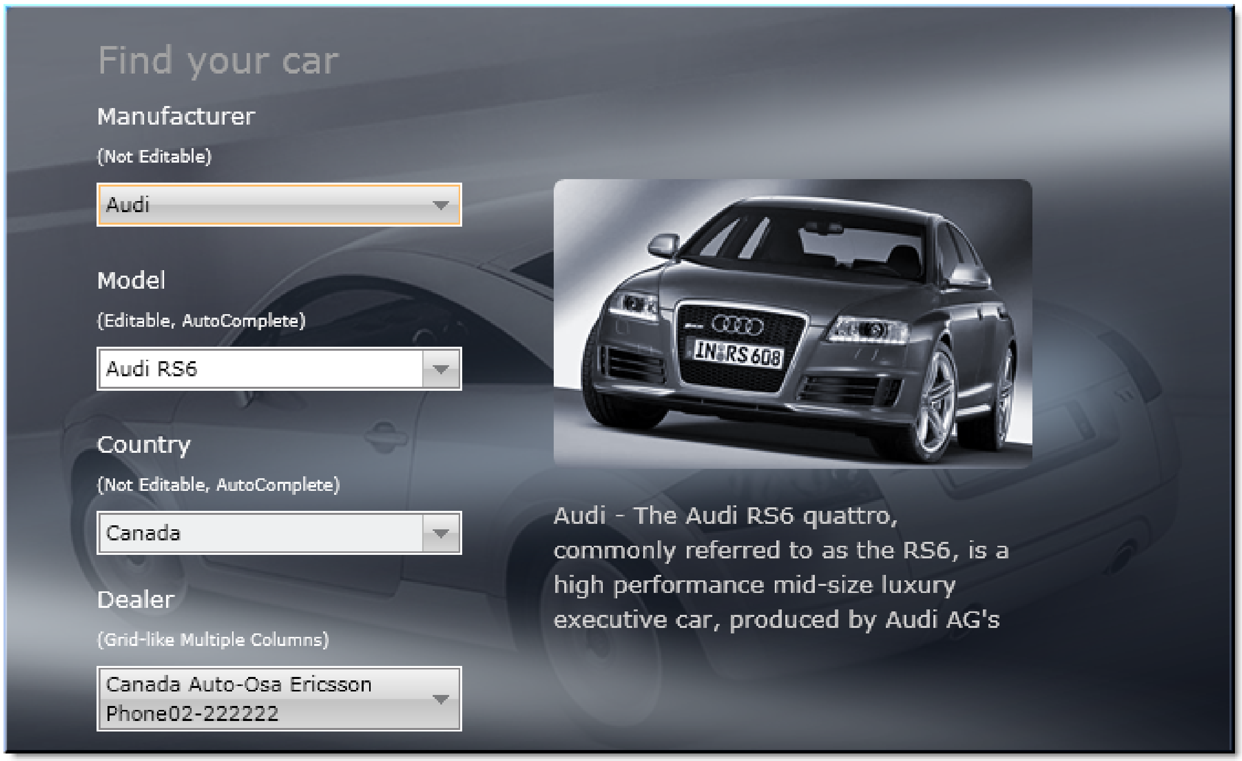
You will learn how various edit modes control the combo box Autocomplete, text search, read-only and editing behavior. You will learn how to search for items and determine if a specific item exists in a combo box. You will use ItemTemplate and SelectionBoxTemplate to build content of arbitrary complexity. You will also bind template elements to live data. Along the way you will discover a few helpful classes in the System.ServiceModel.Syndication namespace for working with RSS (Really Simple Syndication) feeds. Finally, you will use Expression Blend to style the background color of RadComboBox.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Combo\Combo.sln

16.2 Overview

RadComboBox has powerful built-in features such as Autocomplete, automatic filtering and text search. Templates allow the combo box to appear as a multiple column, grid-like list of arbitrary complexity. Using templates, you can completely customize the selected area at the top of the list and the individual list items in the drop down area. All aspects of the combo box can be customized in both editable and non-editable modes.



Find your car

Manufacturer
(Not Editable)
Audi

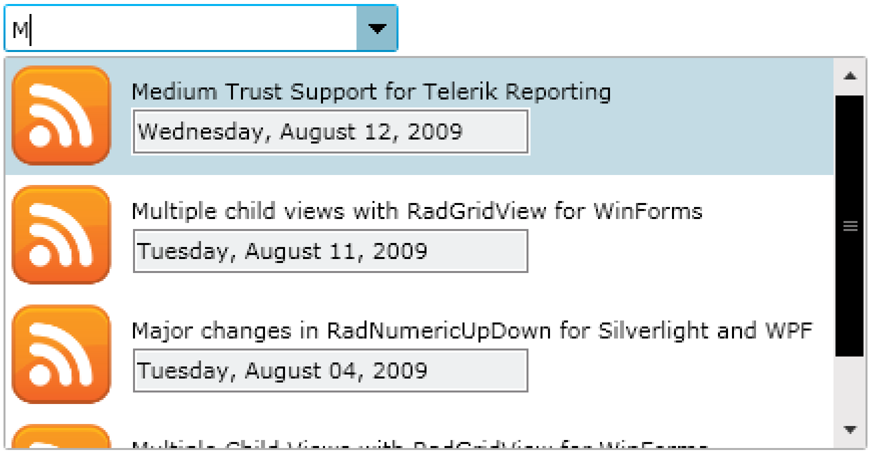
Model
(Editable, AutoComplete)
Audi RS6

Country
(Not Editable, AutoComplete)
Canada

Dealer
(Grid-like Multiple Columns)
Canada Auto-Osa Ericsson
Phone02-222222

Audi - The Audi RS6 quattro, commonly referred to as the RS6, is a high performance mid-size luxury executive car, produced by Audi AG's

The text search, Autocomplete and filtering features also work when the items are composed of templates. Filtering modes show items that either start with or contain characters entered by the user.



M

Medium Trust Support for Telerik Reporting
Wednesday, August 12, 2009

Multiple child views with RadGridView for WinForms
Tuesday, August 11, 2009

Major changes in RadNumericUpDown for Silverlight and WPF
Tuesday, August 04, 2009

RadComboBox keyboard support allows the user to press the arrow keys to move through items in the list, page up or down to move a page at a time, press Enter to select an item and Escape to close the drop down list.

16.3 Getting Started

In this walk through, you will build a simple list of items in a RadComboBox. This example will simulate a list of RSS articles. Later in this chapter we will hook up the logic to get live RSS data. For now the article titles will be hard coded in XAML.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Input**

c) **Telerik.Windows.Controls.Navigation**

d) **Telerik.Windows.Themes.Summer**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace references for **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Input**.assemblies:

Note: Each xmlns statement should be all on one line. The example below is split up to fit the size constraints of the page in this manual.

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .>
```

3) Drag a RadComboBox from the Toolbox to a point just inside the main Grid named "LayoutRoot". Set the **x:Name** property to "cbMain", the **HorizontalAlignment** to "Left", **VerticalAlignment** to "Top", **Margin** to "10", **StyleManager.Theme** to "Summer" and the **SelectionChanged** event to "cbMain_SelectionChanged".

```
<Grid x:Name="LayoutRoot">
  <telerik:RadComboBox x:Name="cbMain"
        HorizontalAlignment="Left" VerticalAlignment="Top"
        Margin="10" telerik:StyleManager.Theme="Summer"
        SelectionChanged="cbMain_SelectionChanged">

  </telerik:RadComboBox>
</Grid>
```

4) Drag a **RadComboBoxItem** from the Toolbox to a point just inside the RadComboBox element. Set the **Content** attribute to "RadControls Silverlight 3 Official".

```
<telerik:RadComboBox x:Name="cbMain"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      Margin="10" telerik:StyleManager.Theme="Summer"
      SelectionChanged="cbMain_SelectionChanged">
  <telerik:RadComboBoxItem
        Content="RadControls Silverlight 3 Official" />
</telerik:RadComboBox>
```

5) Add four more **RadComboBoxItem** elements and set the **Content** attribute for each item to the following series of strings:

- "Medium Trust Support for Telerik Reporting"
- "Telerik Extensions for ASP.NET MVC"
- "First Level Cache of Telerik OpenAccess"
- "Animating the RadWindow control for Silverlight and WPF"

The XAML should now look something like the example below:

```
<telerik:RadComboBox x:Name="cbMain"
    HorizontalAlignment="Left" VerticalAlignment="Top"
    Margin="10" telerik:StyleManager.Theme="Summer"
    SelectionChanged="cbMain_SelectionChanged">
  <telerik:RadComboBoxItem
      Content="RadControls Silverlight 3 Official" />
  <telerik:RadComboBoxItem
      Content="Medium Trust Support for Telerik Reporting" />
  <telerik:RadComboBoxItem
      Content="Telerik Extensions for ASP.NET MVC" />
  <telerik:RadComboBoxItem
      Content="First Level Cache of Telerik OpenAccess" />
  <telerik:RadComboBoxItem
      Content="Animating the RadWindow control for Silverlight and WPF" />
</telerik:RadComboBox>
```

Code Behind

1) Navigate to the code-behind and add a reference to **Telerik.Windows.Controls** in the "Imports" (VB) or "using" (C#) section of code.

2) Add a handler for the SelectionChanged event defined previously in the XAML.

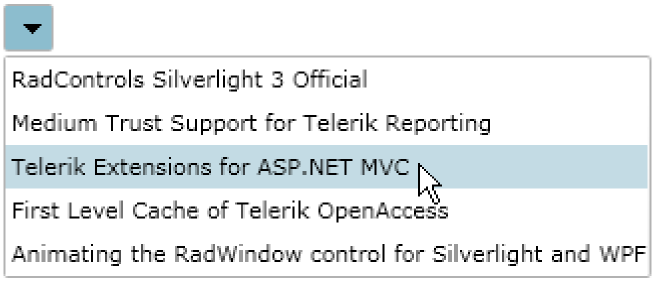
This event handler fires when the user selects another item. First the handler retrieves the arguments AddedItems property and casts the first and only element in the collection to a RadComboBoxItem type. Then the Content for the item is displayed in an Alert.

```
Private Sub cbMain_SelectionChanged(ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  Dim item As RadComboBoxItem = TryCast(e.AddedItems(0), RadComboBoxItem)
  RadWindow.Alert(item.Content.ToString())
End Sub
```

```
private void cbMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   RadComboBoxItem item = e.AddedItems[0] as RadComboBoxItem;
   RadWindow.Alert(item.Content.ToString());
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



RadControls Silverlight 3 Official
Medium Trust Support for Telerik Reporting
Telerik Extensions for ASP.NET MVC
First Level Cache of Telerik OpenAccess
Animating the RadWindow control for Silverlight and WPF

Test Application Features

1) All the strings defined as RadComboBoxItem Content should display in the list.

2) Selecting an item from the list should display the corresponding content in an alert dialog.

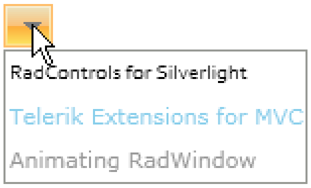
16.4 Control Details

Adding Items

To add items at design-time, include one or more **RadComboBoxItem** elements inside the RadComboBox element. Set the **Content** to the text you want to have displayed. At the same time you can set other RadComboBoxItem properties to tweak behavior and appearance.

```
<telerik:RadComboBox . . .>
  <telerik:RadComboBoxItem
      Content="RadControls for Silverlight"
      FontFamily="Corbel" />
  <telerik:RadComboBoxItem
      Content="Telerik Extensions for MVC"
      Foreground="SkyBlue" />
  <telerik:RadComboBoxItem
      Content="Animating RadWindow"
      IsEnabled="False" />
</telerik:RadComboBox>
```

The results of the XAML show in the screenshot below with item text displaying in a new FontFamily and Foreground font color. The last item is disabled.

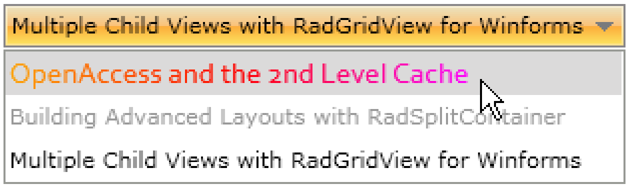

RadControls for Silverlight
Telerik Extensions for MVC
Animating RadWindow

To add items at run-time, create RadComboBoxItem instances in code, then set the Content and any properties related to behavior and appearance. Add each item to the RadComboBox Items collection. The example below configures a linear gradient brush and assigns it to the Foreground property. The second item is disabled and the last item is selected.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' setup a linear gradient brush to use in an item font
  Dim linearGradientBrush As New LinearGradientBrush()
  linearGradientBrush.GradientStops.Add( _
New GradientStop() With {.Color = Colors.Orange, .Offset = 0})
  linearGradientBrush.GradientStops.Add( _
New GradientStop() With {.Color = Colors.Red, .Offset = 0.5})
  linearGradientBrush.GradientStops.Add( _
New GradientStop() With {.Color = Colors.Magenta, .Offset = 1})

  ' add a new combo box item, change the font family, size and color
  cbMain.Items.Add(New RadComboBoxItem() _
With {
.Content = "OpenAccess and the 2nd Level Cache", _
.FontFamily = New FontFamily("Corbel"), _
.FontSize = 15, _
.Foreground = linearGradientBrush})
  ' add another item, but disable it
  cbMain.Items.Add(New RadComboBoxItem() _
With { _
.Content = "Building Advanced Layouts with RadSplitContainer", _
.IsEnabled = False})
  ' add another item and select it
  cbMain.Items.Add(New RadComboBoxItem() _
With { _
.Content = "Multiple Child Views with RadGridView for Winforms", _
.IsSelected = True})
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   // setup a linear gradient brush to use in an item font
   LinearGradientBrush linearGradientBrush = new LinearGradientBrush();
   linearGradientBrush.GradientStops.Add(new GradientStop()
   {
      Color = Colors.Orange, Offset = 0
   });
   linearGradientBrush.GradientStops.Add(new GradientStop()
   {
      Color = Colors.Red, Offset = 0.5
   });
   linearGradientBrush.GradientStops.Add(new GradientStop()
   {
      Color = Colors.Magenta, Offset = 1
   });

   // add a new combo box item, change the font family, size and color
   cbMain.Items.Add(new RadComboBoxItem()
   {
      Content = "OpenAccess and the 2nd Level Cache",
      FontFamily = new FontFamily("Corbel"),
      FontSize = 15,
      Foreground = linearGradientBrush
   });
   // add another item, but disable it
   cbMain.Items.Add(new RadComboBoxItem()
   {
      Content = "Building Advanced Layouts with RadSplitContainer",
      IsEnabled = false
   });
   // add another item and select it
   cbMain.Items.Add(new RadComboBoxItem()
   {
      Content = "Multiple Child Views with RadGridView for Winforms",
      IsSelected = true
   });
}
```

Here is a screenshot of the resulting combo box running in the browser:



Multiple Child Views with RadGridView for Winforms

OpenAccess and the 2nd Level Cache

Building Advanced Layouts with RadSplitContainer

Multiple Child Views with RadGridView for Winforms

You may need to include a "Create New" at the head of the list. Use the Items collection Insert() method, first passing the index where you want the item placed, then passing a RadComboBoxItem reference. The snippet below shows adding an item to the start of the list, hooking up its Selected event and displaying the new clicked item's content.

VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim newItem As New RadComboBoxItem() _
With {.Content = "-- New --", .Tag = "NEW"}
  AddHandler newItem.Selected, AddressOf newItem_Selected
  cbMain.Items.Insert(0, newItem)
End Sub

Private Sub newItem_Selected( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim item As RadComboBoxItem = TryCast(sender, RadComboBoxItem)
  MessageBox.Show("You clicked " & item.Content.ToString())
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadComboBoxItem newItem =
      new RadComboBoxItem() { Content = "-- New --", Tag = "NEW" };
   newItem.Selected += new RoutedEventHandler(newItem_Selected);
   cbMain.Items.Insert(0, newItem);
}

void newItem_Selected(object sender, RoutedEventArgs e)
{
   RadComboBoxItem item = sender as RadComboBoxItem;
   MessageBox.Show("You clicked " + item.Content.ToString());
}
```

Item Selection

To get or set the currently selected combo box item, use either the RadComboBox **SelectedIndex** or **SelectedItem** property. You can also set the **IsSelected** property of an individual RadComboBoxItem. Here are some examples that show all three possibilities: selecting the first item in the list by index, using a reference to the last item in the list and using the IsSelected property of a RadComboBoxItem.

VB

```
' select the first item in the list
cbMain.SelectedIndex = 0

' select the last item in the list
Dim lastItem As Object = cbMain.Items(cbMain.Items.Count - 1)
cbMain.SelectedItem = lastItem

' using the IsSelected property of an item
Dim newItem As New RadComboBoxItem() _
With {.Content = "This will be selected", .IsSelected = True}
cbMain.Items.Add(newItem)
```

C#

```
// select the first item in the list
cbMain.SelectedIndex = 0;

// select the last item in the list
object lastItem = cbMain.Items[cbMain.Items.Count - 1];
cbMain.SelectedItem = lastItem;

// using the IsSelected property of an item
RadComboBoxItem newItem = new RadComboBoxItem()
{
   Content = "This will be selected",
   IsSelected = true
};
cbMain.Items.Add(newItem);
```

From the Forums...

Forum question: "Sometimes... the SelectionBoxTemplate is not applied until the combo box is opened, even though an item is selected."

Answer:

"The problem is that the RadComboBoxItem containers are generated after the combo box drop down is opened for the first time. If the containers are not generated, the container bindings are not applied and the combo box does not 'know' which data item is selected. The correct way to initially select an item is through one of the following properties: SelectedItem, SelectedIndex."

Searching Items

You can use LINQ statements or expressions to check whether a certain RadComboBoxItem (or other object) is present a RadComboBox.Items collection. You can make variations of the LINQ statement example below. Once you have a reference to the Content, compare the Content object using the appropriate methods for the type. In this case the Content is a string so we can use the String methods such as Contains(), StartsWith() or Equals().

VB

```
Imports System.Linq
'. . .
Private exists As Boolean = cbMain.Items.Any(item => _
(TryCast(item, RadComboBoxItem)).Content.ToString().Contains("OpenAccess"))
If exists Then
  tbStatus.Text = "There are OpenAccess articles in the list"
End If
```

C#

```
using System.Linq;
//. . .
bool exists = cbMain.Items.Any(item =>
   (item as RadComboBoxItem).Content.ToString().Contains("OpenAccess"));
if (exists)
{
   tbStatus.Text = "There are OpenAccess articles in the list";
}
```

Here's another example where a LINQ expression selects all the items that contain the string "OpenAccess". The resulting collection can be enumerated, converted to a generic list or operated on by other LINQ methods.

VB

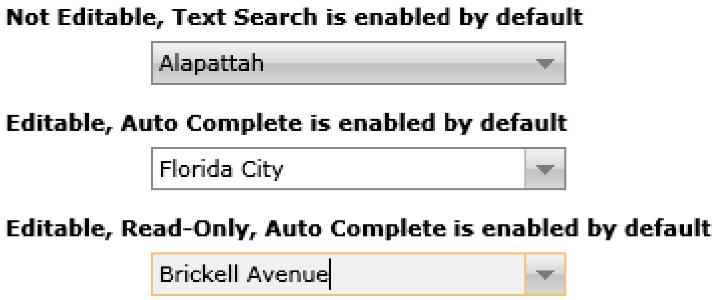
```
Dim foundItems = _
  From item In cbMain.Items _
  Where (TryCast(item, _
RadComboBoxItem)).Content.ToString().Contains("OpenAccess") _
  Select item

For Each item As RadComboBoxItem In foundItems
  MessageBox.Show(item.Content.ToString())
Next item
```

C#

```
var foundItems =
   from item in cbMain.Items
   where (item as RadComboBoxItem).Content.ToString().Contains("OpenAccess")
   select item;

foreach (RadComboBoxItem item in foundItems)
{
   MessageBox.Show(item.Content.ToString());
}
```

Edit modes

The behavior of the combo box is controlled by a combination of **IsEditable** and **IsReadOnly** property settings. RadComboBox can be editable, allowing the user to type in its text box, or non-editable, where the text box is hidden. In addition, you could make the text box read-only, in order to keep the editable look, but prevent the user from typing. The combinations are shown in the screenshot below.



Not Editable, Text Search is enabled by default

Alapattah

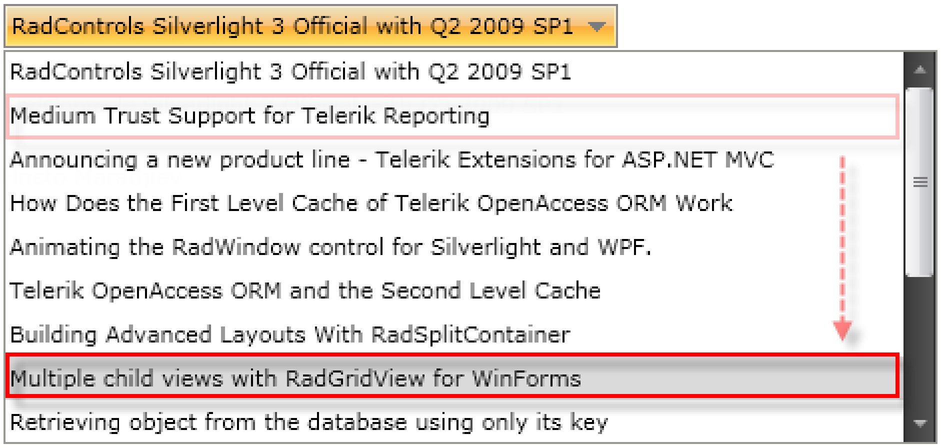
Editable, Auto Complete is enabled by default

Florida City

Editable, Read-Only, Auto Complete is enabled by default

Brickell Avenue

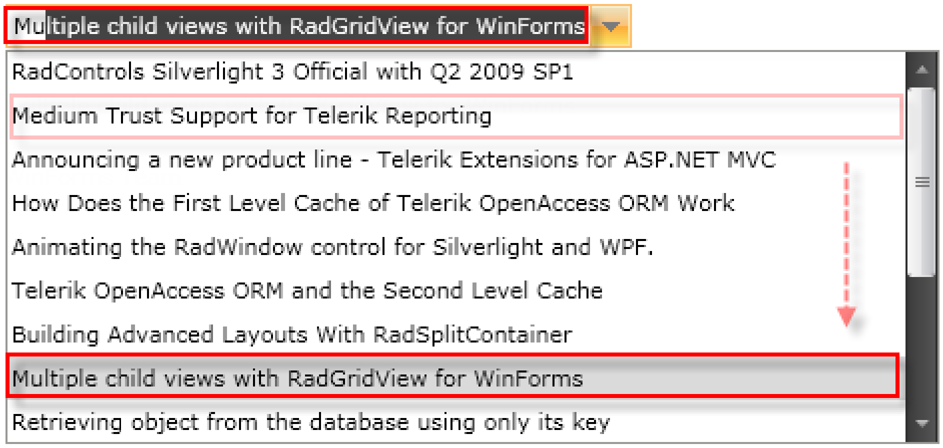
Both **IsEditable** and **IsReadOnly** properties are "False" by default. In this mode the user can perform a text search by typing a few characters in the combo box. The screenshot below shows a series of RSS feed titles. When the user types in "M", the selection travels to the 'Medium Trust Support for Telerik Reporting" item. When the user types in "u", the selection moves down to "Multiple child views with RadGridView for WinForms" item.



RadControls Silverlight 3 Official with Q2 2009 SP1

RadControls Silverlight 3 Official with Q2 2009 SP1
Medium Trust Support for Telerik Reporting
Announcing a new product line - Telerik Extensions for ASP.NET MVC
How Does the First Level Cache of Telerik OpenAccess ORM Work
Animating the RadWindow control for Silverlight and WPF.
Telerik OpenAccess ORM and the Second Level Cache
Building Advanced Layouts With RadSplitContainer
Multiple child views with RadGridView for WinForms
Retrieving object from the database using only its key

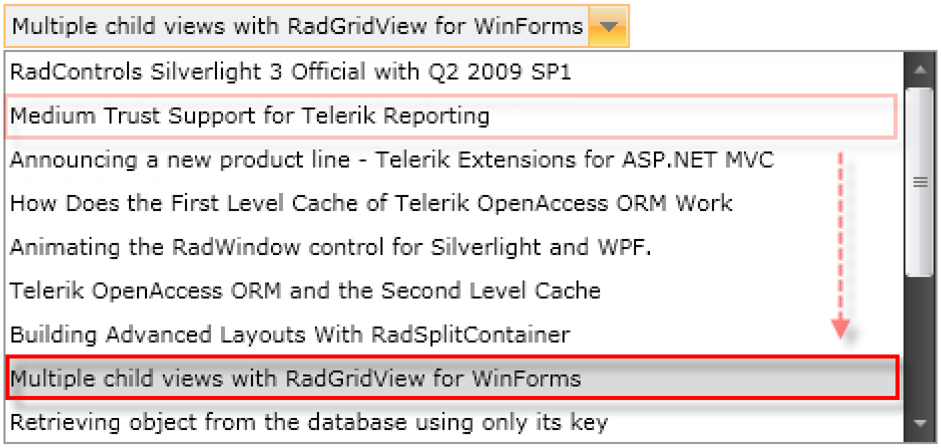
If IsEditable is "True", the combo box provides Auto Complete functionality. The selection behavior is the same as before. The difference is the "type ahead" behavior. When the user types "M" the text entry area of the combo box is automatically completed with the nearest entry starting with "M", i.e. "Medium Trust Support for Telerik Reporting". When the user types "Mu", again the entry is completed with the nearest match.



Multiple child views with RadGridView for WinForms

RadControls Silverlight 3 Official with Q2 2009 SP1
Medium Trust Support for Telerik Reporting
Announcing a new product line - Telerik Extensions for ASP.NET MVC
How Does the First Level Cache of Telerik OpenAccess ORM Work
Animating the RadWindow control for Silverlight and WPF.
Telerik OpenAccess ORM and the Second Level Cache
Building Advanced Layouts With RadSplitContainer
Multiple child views with RadGridView for WinForms
Retrieving object from the database using only its key

In this next screenshot, both IsEditable and IsReadOnly are "True".



Multiple child views with RadGridView for WinForms

RadControls Silverlight 3 Official with Q2 2009 SP1
Medium Trust Support for Telerik Reporting
Announcing a new product line - Telerik Extensions for ASP.NET MVC
How Does the First Level Cache of Telerik OpenAccess ORM Work
Animating the RadWindow control for Silverlight and WPF.
Telerik OpenAccess ORM and the Second Level Cache
Building Advanced Layouts With RadSplitContainer
Multiple child views with RadGridView for WinForms
Retrieving object from the database using only its key

By default RadComboBox uses the **DisplayMemberPath** property to determine which property of a data source to use for auto complete.

XAML

```
<telerik:RadComboBox
  DisplayMemberPath="Title.Text">
</telerik:RadComboBox>
```

If you are defining an ItemTemplate, there can be multiple items in the template. Use the **TextSearch. TextPath** attached property to specify which bound property to search on. Note: TextSearch.TextPath is in the **Telerik.Windows.Controls** namespace.

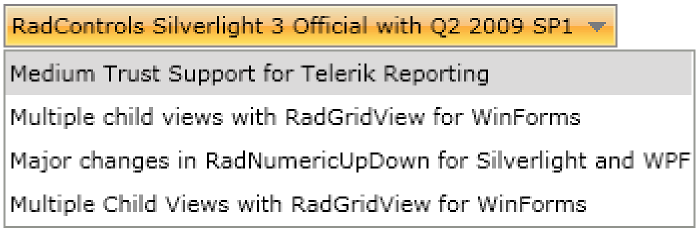
```
<telerik:RadComboBox
   telerik:TextSearch.TextPath="Title.Text">
   <telerik:RadComboBox.ItemTemplate>
      <DataTemplate>
         <TextBlock Text="{Binding Title.Text}" />
      </DataTemplate>
   </telerik:RadComboBox.ItemTemplate>
</telerik:RadComboBox>
```

Filtering

Filtering capability is built into RadComboBox. The **TextSearchMode** property can be "StartsWith", "StartsWithCaseSensitive", "Contains" or "ContainsCaseSensitive" to automatically filter the combo. The example below shows the combo where **TextSearchMode** is set to "StartsWith". To control the text filtering you can set the **IsFilteringEnabled** property. By default the **IsFilteringEnabled** is set to **False**.

```
<telerik:RadComboBox x:Name="cbMain" Margin="10"
      HorizontalAlignment="Left" VerticalAlignment="Top"
      DisplayMemberPath="Title.Text" TextSearchMode="StartsWith" IsFilteringEnabled="True">
</telerik:RadComboBox>
```

In the screenshot below the "M" character is entered. As additional characters are entered, the list narrows to meet the criteria.



RadControls Silverlight 3 Official with Q2 2009 SP1

Medium Trust Support for Telerik Reporting

Multiple child views with RadGridView for WinForms

Major changes in RadNumericUpDown for Silverlight and WPF

Multiple Child Views with RadGridView for WinForms

From the Forums...

"When TextSearchMode == Contains or StartsWith, RadComboBox searches through its items for items that match its Text. Those items that match are made visible, the rest are collapsed."

Selection Events

The most significant event to handle is **SelectionChanged** that fires when the user clicks a new item in the list or arrow keys down and hits Enter. The **Sender** parameter represents the RadComboBox and **SelectionChangedEventArgs** contains an **AddedItems** property that holds the selected item. Typically you will check that there is at least one item in AddedItems and retrieve the first (and only) item in the list. What you do from here depends on what is in the list. If you added a series of RadComboBoxItem in code or XAML, you will cast to a RadComboBoxItem type.

Here is an example that uses a collection of RSS objects (available in the **System.ServiceModel.Syndication** namespace). The code fragment at the top shows how to create a series of RadComboBoxItem objects, populate each with content and add to the RadComboBox Items collection. In the SelectionChanged event handler you can cast the first AddedItems element as a RadComboBoxItem type and work with that.

VB

```
'...
For Each item As SyndicationItem In syndicationFeed.Items
  Dim cbItem As New RadComboBoxItem() With {.Content = item.Title.Text}
  cbMain.Items.Add(cbItem)
Next item
cbMain.SelectedIndex = 0
'...

Private Sub cbMain_SelectionChanged(ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  If e.AddedItems.Count > 0 Then
    Dim item As RadComboBoxItem = TryCast(e.AddedItems(0), RadComboBoxItem)
    tbTitle.Text = item.Content.ToString()
  End If
End Sub
```

```
// . . .
foreach (SyndicationItem item in syndicationFeed.Items)
{
   RadComboBoxItem cbItem =
      new RadComboBoxItem() { Content = item.Title.Text };
   cbMain.Items.Add(cbItem);
}
cbMain.SelectedIndex = 0;
// . . .

private void cbMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   if (e.AddedItems.Count > 0)
   {
      RadComboBoxItem item = e.AddedItems[0] as RadComboBoxItem;
      tbTitle.Text = item.Content.ToString();
   }
}
```

If you bound the RadComboBox ItemsSource to a collection of some sort, then you will cast to the collection item type. One other possibility is that you could add instances of an object directly to the RadComboBox.Items collection. In both cases DisplayMemberPath indicates the property of the object to display in the list.

VB

```
'. . .
' bind to ItemsSource...
cbMain.ItemsSource = syndicationFeed.Items

' or add each object directly into the Items array
'For Each item As SyndicationItem In syndicationFeed.Items
'   cbMain.Items.Add(item)
'Next item
cbMain.SelectedIndex = 0
cbMain.DisplayMemberPath = "Title.Text"
'. . .

Private Sub cbMain_SelectionChanged(ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  If e.AddedItems.Count > 0 Then
    Dim item As SyndicationItem = TryCast(e.AddedItems(0), SyndicationItem)
    tbTitle.Text = item.Title.Text
  End If
End Sub
```

C#

```
//. . .
// bind to ItemsSource...
cbMain.ItemsSource = syndicationFeed.Items;

// or add each object directly into the Items array
//foreach (SyndicationItem item in syndicationFeed.Items)
//{
//   cbMain.Items.Add(item);
//}
cbMain.SelectedIndex = 0;
cbMain.DisplayMemberPath = "Title.Text";
//. . .

private void cbMain_SelectionChanged(object sender,
  Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
  if (e.AddedItems.Count > 0)
  {
    SyndicationItem item = e.AddedItems[0] as SyndicationItem;
    tbTitle.Text = item.Title.Text;
  }
}
```

If you want notification of individual item selection, handle the RadComboBoxItem **Selection** event. The "Sender" parameter passed to the event handler will be the RadComboBoxItem instance (or whatever item type was added to the Items collection).

VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim item As New RadComboBoxItem() With {.Content = "Test"}
  AddHandler item.Selected, AddressOf item_Selected
  cbMain.Items.Add(item)
End Sub

Private Sub item_Selected( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim item As RadComboBoxItem = TryCast(sender, RadComboBoxItem)
  MessageBox.Show("You clicked " & item.Content.ToString())
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadComboBoxItem item =
      new RadComboBoxItem() { Content = "Test" };
   item.Selected += new RoutedEventHandler(item_Selected);
   cbMain.Items.Add(item);
}

void item_Selected(object sender, RoutedEventArgs e)
{
   RadComboBoxItem item = sender as RadComboBoxItem;
   MessageBox.Show("You clicked " + item.Content.ToString());
}
```

DropDown Events

The RadComboBox class introduces the **DropDownOpened** and **DropDownClosed** events. You can use these events for any setup or tear down work just before and after the user opens the list.

The example below clears the text selection of the combo box just before the list is opened and just after it closes. The ChildrenOfType<> method (an extension method from Telerik.Windows.Controls) gets a reference to the TextBox contained by RadComboBox and sets the SelectionLength to zero.

VB

```
' Unselect the text in the RadComboBox TextBox
Private Sub ClearSelection(ByVal comboBox As RadComboBox)
  Dim textBox As TextBox = comboBox.ChildrenOfType(Of TextBox)()(0)
  textBox.SelectionStart = 0
  textBox.SelectionLength = 0
End Sub

Private Sub cbMain_DropDownClosed( _
ByVal sender As Object, ByVal e As EventArgs)
  ClearSelection(TryCast(sender, RadComboBox))
End Sub

Private Sub cbMain_DropDownOpened( _
ByVal sender As Object, ByVal e As EventArgs)
  ClearSelection(TryCast(sender, RadComboBox))
End Sub
```

C#

```
// Unselect the text in the RadComboBox TextBox
private void ClearSelection(RadComboBox comboBox)
{
   TextBox textBox =
      comboBox.ChildrenOfType<TextBox>()[0];
   textBox.SelectionStart = 0;
   textBox.SelectionLength = 0;
}

private void cbMain_DropDownClosed(object sender, EventArgs e)
{
   ClearSelection(sender as RadComboBox);
}

private void cbMain_DropDownOpened(object sender, EventArgs e)
{
   ClearSelection(sender as RadComboBox);
}
```

16.5 Binding

RadComboBox is an ItemsControl descendant and has an **ItemTemplate** that handles layout for each of the individual list items.**SelectionBoxItemTemplate** handles the layout of the text at the top of the RadComboBox. Both templates can contain static and bound elements. This walk through creates custom layouts for both templates.

In the example we will use Silverlight classes that do a nice job of parsing RSS (Really Simple Syndication) format. These classes are **SyndicationFeed** and **SyndicationItem**, both found in the **System.ServiceModel.Syndication** namespace. We will use the **WebClient** from the **System.Net** namespace to download the RSS data into a stream, then load the stream into a SyndicationFeed instance. From there, SyndicationFeed provides a collection of SyndicationItem that can be bound to the combo box.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Input**

c) **Telerik.Windows.Themes.Vista**

d) **System.ServiceModel.Syndication**

XAML Editing

1) Open MainPage.xaml for editing.

2) Verify that the XML namespaces for **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Input** exist in the UserControl element. Add them if they do not exist. Also, add a "Loaded" event handler to the UserControl element.

```
<UserControl
xmlns:telerik=
"http://schemas.telerik.com/2008/xaml/presentation"
. . .
Loaded="UserControl_Loaded">. . .
```

3) Drag a RadComboBox from the Toolbox to a point inside the main "LayoutRoot" Grid. Set the **x:Name** attribute to "cbMain", the **Margin** to "10", **MinWidth** to "200", **HorizontalAlignment** to "Left", **VerticalAlignment** to "Top" and the **StyleManager.Theme** to "Vista".

```
<Grid x:Name="LayoutRoot">
  <telerik:RadComboBox x:Name="cbMain" Margin="10"
        MinWidth="200" HorizontalAlignment="Left"
        VerticalAlignment="Top" telerik:StyleManager.Theme="Vista">
  <!--Templates go here-->
  </telerik:RadComboBox>
</Grid>
```

4) Add a RadComboBox.SelectionBoxTemplate and a RadComboBox.ItemTemplate inside the RadComboBox element. Inside these two template elements, add DataTemplate elements.

We will put our custom markup and binding expressions inside these two DataTemplate elements.

```
<telerik:RadComboBox x:Name="cbMain" Margin="10"
      MinWidth="200" HorizontalAlignment="Left"
      VerticalAlignment="Top" telerik:StyleManager.Theme="Vista">
  <telerik:RadComboBox.SelectionBoxTemplate>
    <DataTemplate>
      <!--custom markup goes here-->
    </DataTemplate>
  </telerik:RadComboBox.SelectionBoxTemplate>
  <telerik:RadComboBox.ItemTemplate>
    <DataTemplate>
      <!--custom markup goes here-->
    </DataTemplate>
  </telerik:RadComboBox.ItemTemplate>
</telerik:RadComboBox>
```

5) Add a TextBlock inside the SelectionBoxTemplate DataTemplate element. Set the **Margin** to "10", **MinWidth** to "200" and the **FontSize** to "15". Assign a binding expression to the **Text** property: "{Binding Title.Text}".

The SyndicationItem object has a Title property that in turn has a Text sub-property. It's the Text sub-property that we actually want to display in the upper portion of the RadComboBox.

```
<telerik:RadComboBox.SelectionBoxTemplate>
   <DataTemplate>
      <TextBlock Margin="10" MinWidth="200"
            FontSize="15" Text="{Binding Title.Text}" />
   </DataTemplate>
</telerik:RadComboBox.SelectionBoxTemplate>
```

6) Populate the ItemTemplate DataTemplate element. This is a more complex layout. Start by inserting a **StackPanel** with **Margin** = "10". Inside the StackPanel add a **TextBlock** and a **RadMaskedTextBox**. Set the TextBlock HorizontalAlignment to "Left" and make the binding expression "{Binding Title.Text}". Set the RadMaskedTextBox **HorizontalAlignment** to "Left", **MinWidth** to "200", **MaskType** to "DateTime", **Mask** to "D" and **IsReadOnly** to "True". Bind the RadMaskedTextBox **Value** property to "{Binding PublishDate.DateTime}".

The RadComboBox items displayed when the list is opened will show the title text and the publish date right underneath the title.

```
<telerik:RadComboBox.ItemTemplate>
   <DataTemplate>
      <StackPanel Margin="10">
         <TextBlock HorizontalAlignment="Left"
               Text="{Binding Title.Text}" />
         <telerik:RadMaskedTextBox
               HorizontalAlignment="Left"
               MinWidth="200" MaskType="DateTime"
               Mask="D" IsReadOnly="True"
               Value="{Binding PublishDate.DateTime}" />
      </StackPanel>
   </DataTemplate>
</telerik:RadComboBox.ItemTemplate>
```

Code Behind

1) Verify that namespace references below are included in the "Imports" (VB) or "using" (C#) section of code.

System.IO

System.Net

System.ServiceModel.Syndication

System.Windows

System.Windows.Controls

System.Xml

2) Handle the UserControl Loaded event. Create a new WebClient instance to download the RSS data. Assign a DownloadStringCompleted event handler that will respond once all the data has been downloaded. Call the WebClient DownloadStringAsync() method and pass a Uri of an RSS web site.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim client As New WebClient()
  AddHandler client.DownloadStringCompleted, _
AddressOf client_DownloadStringCompleted
  client.DownloadStringAsync(New Uri("http://feeds.feedburner.com/telerik"))
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   WebClient client = new WebClient();
   client.DownloadStringCompleted +=
      new DownloadStringCompletedEventHandler(client_DownloadStringCompleted);
   client.DownloadStringAsync(new Uri("http://feeds.feedburner.com/telerik"));
}
```

Notes

Be aware that this implementation is specific to this particular RSS site. This code assumes the return data will be a string and there is no safety code for null fields within the returned XML. This is to keep the code small and relevant to RadComboBox binding.

3) Handle the DownloadStringCompleted event.

The DownloadStringCompletedEventArgs contain a Result string property that holds the RSS XML data. Pass that Result string to a StringReader constructor, then pass the StringReader to a XmlReader constructor. Use the XmlReader in the static SyndicationFeed.Load() method.

Now that the SyndicationFeed object is completely initialized and loaded with RSS items, assign the SyndicationFeed Items collection to the RadComboBox ItemsSource property.

VB

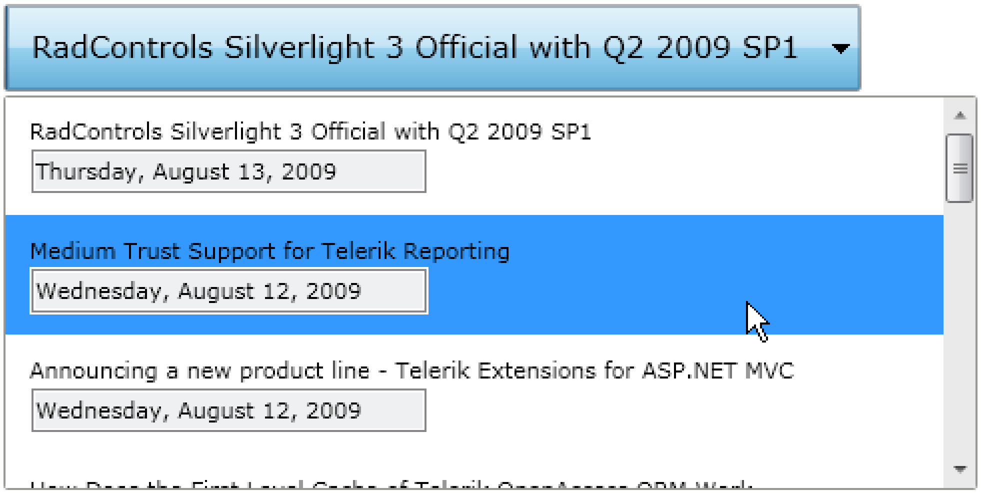
```
Private Sub client_DownloadStringCompleted( _
ByVal sender As Object, ByVal e As DownloadStringCompletedEventArgs)
  ' Create an XmlReader using a StringReader containing the result string
  Dim stringReader As New StringReader(e.Result)
  Dim xmlReader As XmlReader = XmlReader.Create(stringReader)
  ' Load and create the SyndicationFeed instance using the XmlReader
  Dim syndicationFeed As SyndicationFeed = SyndicationFeed.Load(xmlReader)

  ' bind to ItemsSource
  cbMain.ItemsSource = syndicationFeed.Items

  ' select the first item in the list
  cbMain.SelectedIndex = 0
End Sub
```

C#

```
void client_DownloadStringCompleted(
object sender, DownloadStringCompletedEventArgs e)
{
   // Create an XmlReader using a StringReader containing the result string
   StringReader stringReader = new StringReader(e.Result);
   XmlReader xmlReader = XmlReader.Create(stringReader);
   // Load and create the SyndicationFeed instance using the XmlReader
   SyndicationFeed syndicationFeed = SyndicationFeed.Load(xmlReader);

   // bind to ItemsSource
   cbMain.ItemsSource = syndicationFeed.Items;

   // select the first item in the list
   cbMain.SelectedIndex = 0;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



RadControls Silverlight 3 Official with Q2 2009 SP1

RadControls Silverlight 3 Official with Q2 2009 SP1
Thursday, August 13, 2009

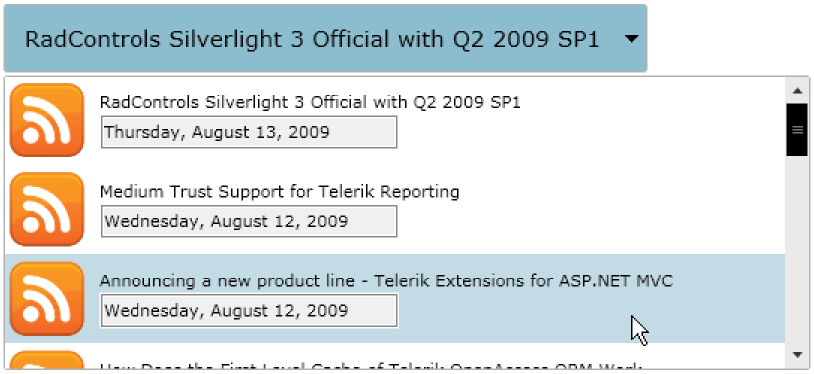
Medium Trust Support for Telerik Reporting
Wednesday, August 12, 2009

Announcing a new product line - Telerik Extensions for ASP.NET MVC
Wednesday, August 12, 2009

How Does the First Level Cache of Telerik OpenAccess ORM Work

Ideas for Extending This Example

- Disable the RadComboBox until it finishes loading.
- Point to a different RSS URL location. This will require changing the coding. RSS sites vary on what information they choose to send out, so code defensively and assume null fields.
- Add images to the template. The example below displays an Rss symbol image to the left of the other content. Also, the theme has been changed to "Summer". See below for sample XAML markup.



RadControls Silverlight 3 Official with Q2 2009 SP1

RadControls Silverlight 3 Official with Q2 2009 SP1
Thursday, August 13, 2009

Medium Trust Support for Telerik Reporting
Wednesday, August 12, 2009

Announcing a new product line - Telerik Extensions for ASP.NET MVC
Wednesday, August 12, 2009

How Does the First Level Cache of Telerik OpenAccess ORM Work

```
<telerik:RadComboBox x:Name="cbMain" Margin="10"
    MinWidth="200" HorizontalAlignment="Left"
    VerticalAlignment="Top"
    telerik:StyleManager.Theme="Summer">

  <telerik:RadComboBox.SelectionBoxTemplate>
    <DataTemplate>
      <TextBlock Margin="10" MinWidth="200"
          FontSize="15" Text="{Binding Title.Text}" />
    </DataTemplate>
  </telerik:RadComboBox.SelectionBoxTemplate>
  <telerik:RadComboBox.ItemTemplate>
    <DataTemplate>
      <StackPanel Orientation="Horizontal">
        <Image Source="../images/Rss.png" Width="50"
            Height="50" Stretch="Uniform"
            HorizontalAlignment="Left" />
        <StackPanel Margin="10">
          <TextBlock HorizontalAlignment="Left"
              Text="{Binding Title.Text}" />
          <telerik:RadMaskedTextBox
              HorizontalAlignment="Left"
              MinWidth="200" MaskType="DateTime"
              Mask="D" IsReadOnly="True"
              Value="{Binding PublishDate.DateTime}" />
        </StackPanel>
      </StackPanel>
    </DataTemplate>
  </telerik:RadComboBox.ItemTemplate>
</telerik:RadComboBox>
```

- Refactor the XAML to move the DataTemplate out into UserControl.Resources. Just relocate the DataTemplate elements to the UserControl.Resources element. Be sure to assign the "x:Key" attribute to the DataTemplate. Then, assign the RadComboBox ItemTemplate and SelectionBoxTemplate attributes. As usual, moving XAML into the Resources area really cleans up the XAML.

```
<UserControl . . .>

  <UserControl.Resources>

    <!--Template for RadComboBox ItemTemplate-->
    <DataTemplate x:Key="MyItemTemplate">
      <StackPanel Orientation="Horizontal">
        <Image Source="../images/Rss.png" Width="50"
            Height="50" Stretch="Uniform"
            HorizontalAlignment="Left" />
        <StackPanel Margin="10">
          <TextBlock HorizontalAlignment="Left"
              Text="{Binding Title.Text}" />
          <telerik:RadMaskedTextBox
              HorizontalAlignment="Left" MinWidth="200"
              MaskType="DateTime" Mask="D"
              IsReadOnly="True"
              Value="{Binding PublishDate.DateTime}" />
        </StackPanel>
      </StackPanel>
    </DataTemplate>

    <!--Template for RadComboBox SelectionBoxTemplate-->
    <DataTemplate x:Key="MySelectionBoxTemplate">
      <TextBlock Margin="10" MinWidth="200" FontSize="15"
          Text="{Binding Title.Text}" />
    </DataTemplate>

  </UserControl.Resources>


  <Grid x:Name="LayoutRoot">
    <telerik:RadComboBox x:Name="cbMain" Margin="10"
        MinWidth="200" HorizontalAlignment="Left"
        VerticalAlignment="Top"
        telerik:StyleManager.Theme="Summer"
        ItemTemplate="{StaticResource MyItemTemplate}"
        SelectionBoxTemplate=
         "{StaticResource MySelectionBoxTemplate}">
    </telerik:RadComboBox>
  </Grid>
</UserControl>
```

16.6 Customization

In this walk through we will customize the RadComboBox background. Before going into Expression Blend, you should know that RadComboBox has two control templates and that the **IsEditable** property determines which control template is shown: "NonEditableComboBox" or "EditableComboBox". We're going to edit the drop down button of the "NonEditableComboBox" to use "Scoville" colors.

"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

1) Run Expression Blend.

2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*

3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight 4" Application from the right-most list. Enter a unique name for the project and click **OK**.

4) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu.

5) Add references to **Telerik.Windows.Controls.dll** and **Telerik.Windows.Controls.Input.dll**.

Edit the Page in Expression Blend

1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.

2) From the Project menu select **Build Project**.

3) Add a **RadComboBox** to the page.

   a) Open the Assets pane.

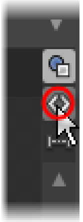
   b) On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.

   c) In the Assets pane, just above the tree view is the Assets Find entry text box.

   d) Type the first few characters of "RadComboBox**"** into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view.

   e) Locate the RadComboBox control and drag it onto the MainPage.xaml Artboard.

4) In the Artboard, select the XAML view button

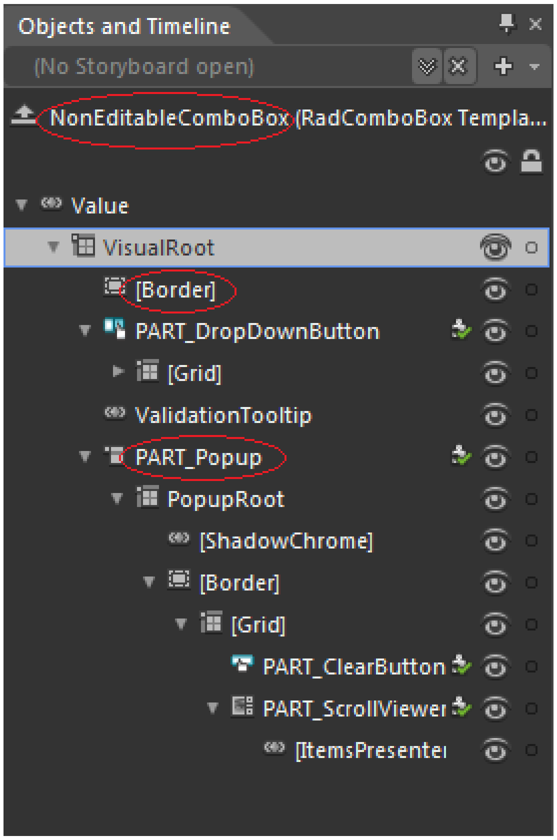


5) Replace the RadComboBox element with the markup below. This will define several sample RadComboBoxItem to populate the list.

```
<telerik:RadComboBox
  HorizontalAlignment="Left"
  Margin="8,8,0,0"
  VerticalAlignment="Top">
  <telerik:RadComboBoxItem Content="Red Savina Habanero"
    IsSelected="True" />
  <telerik:RadComboBoxItem Content="Datil pepper" />
  <telerik:RadComboBoxItem Content="Chiltepin Pepper" />
  <telerik:RadComboBoxItem Content="Jalapeño" />
  <telerik:RadComboBoxItem Content="Pimento" />
  <telerik:RadComboBoxItem Content="Pepperoncini" />
</telerik:RadComboBox>
```

6) In the Objects and Timeline pane, right-click the "[RadComboBox]" node and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleComboBoxStyle". Click **OK** to create the style resource and close the dialog.

7) In the Objects and Timeline pane, notice that we're looking at the “NonEditableCimboBox” control template. Open the tree view and review the nodes there. Within the "VisualRoot" element of this template we have the "Border" which represents the combo box in its closed state and "PART_Popup" which represents the drop down list portion of the combo box.

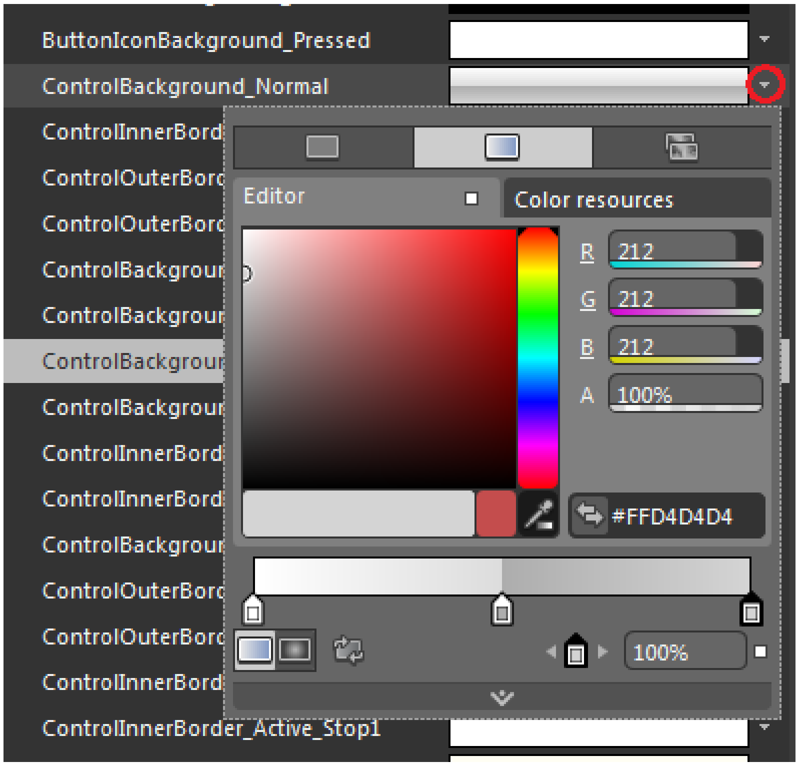


8) In the Objects and Timeline pane, locate the "PART_DropDownButton" node in the tree view, expand it and in the Grid element find “ButtonChrome” control. Right-click and select **Edit a Copy**... from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleButtonChromeStyle". Click OK to create the style resource and close the dialog.

NOTE: Make sure that all the Brushes are on top of the page, before all styles and control templates.

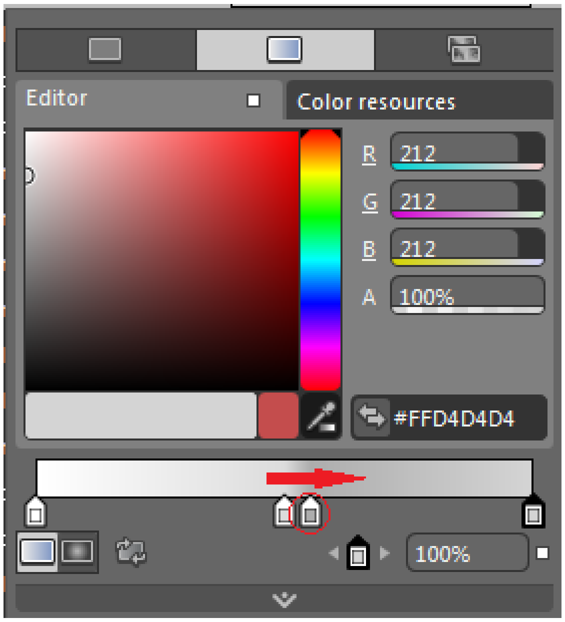
9) In the Resources pane, open the resources for the User Control and locate locate "ControlBackground_Normal". Click the downward pointing arrow button to display a color editing popup.

The color editing popup defines a gradient color with four gradient stops. The middle two gradient stops overlap each other. This gradient describes the 3-D, slightly beveled appearance of the RadComboBox when closed.



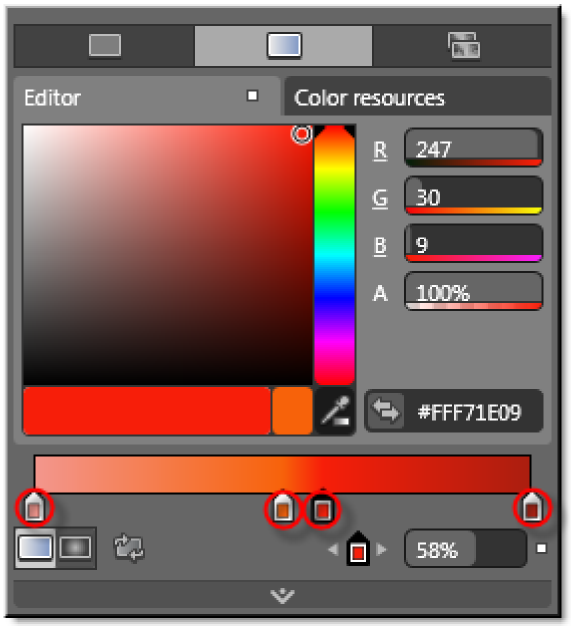
10) Move one of the middle gradient stops slightly to the right so that the two middle gradient stops do not overlap.

There is no design reason to do this. We're separating the two gradient stops merely to make it easier to talk about the process of editing the stops in Expression Blend.



11) Select and change the color for each of the gradient stops using the eye dropper tool. The general color hues, in order of gradient stop, should be pink, orange, red and red-with-black mixed.

The colors need not be exact to demonstrate customizing the RadComboBox background. Feel free to "get creative".

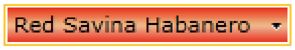


From the Forums...

People will often ask on the forums "how do I change the background color" of "X" control. Given the flexibility of Silverlight and RadControls, answering that question may not be a one-property answer, i.e. Background = "Blue". There are multiple brushes that correspond to states in the control, such as "normal", "mouse over", "selected", etc. You can edit these brushes in Expression Blend to fit your application look-and-feel. Other properties of the control may determine what set of control parts you're looking at, and these parts may use a different set of brushes. For example, RadComboBox uses two different control templates based on the setting of the IsEditable property.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.

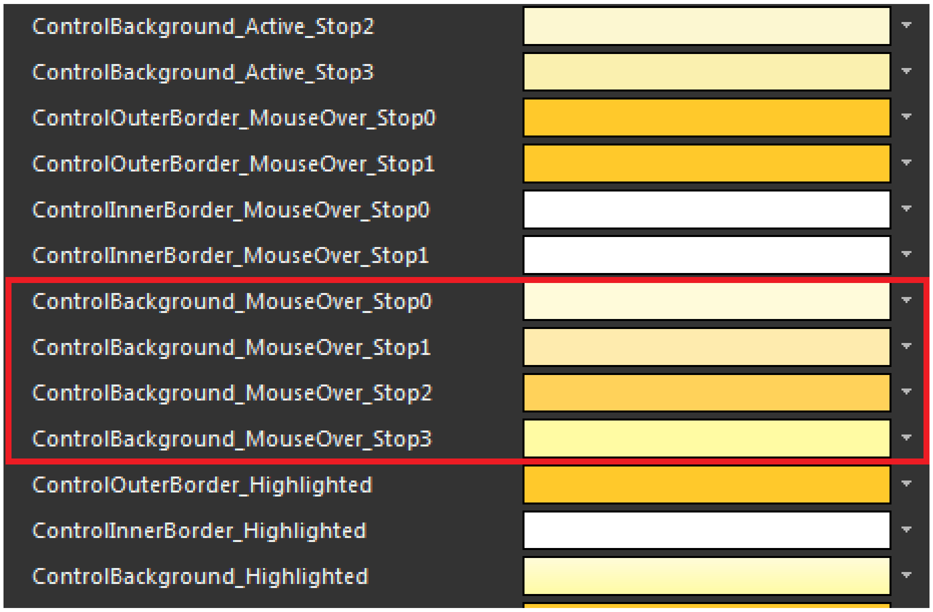


Test Application Features

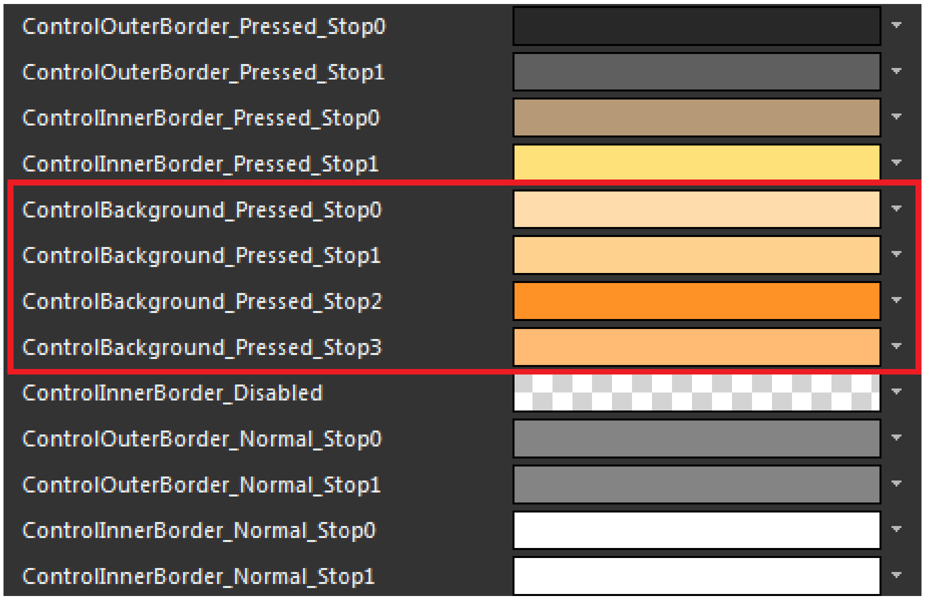
- Verify that the new "Scoville" colors are "in play" when the RadComboBox is closed and unselected.

Ideas for Extending This Example

- The Resources pane contains other brushes for "button pressed" and "mouse hovered" states. Use the same technique from the steps in this walk through to edit the "ControlBackground_Pressed" and " ControlBackground_MouseOver" brushes.



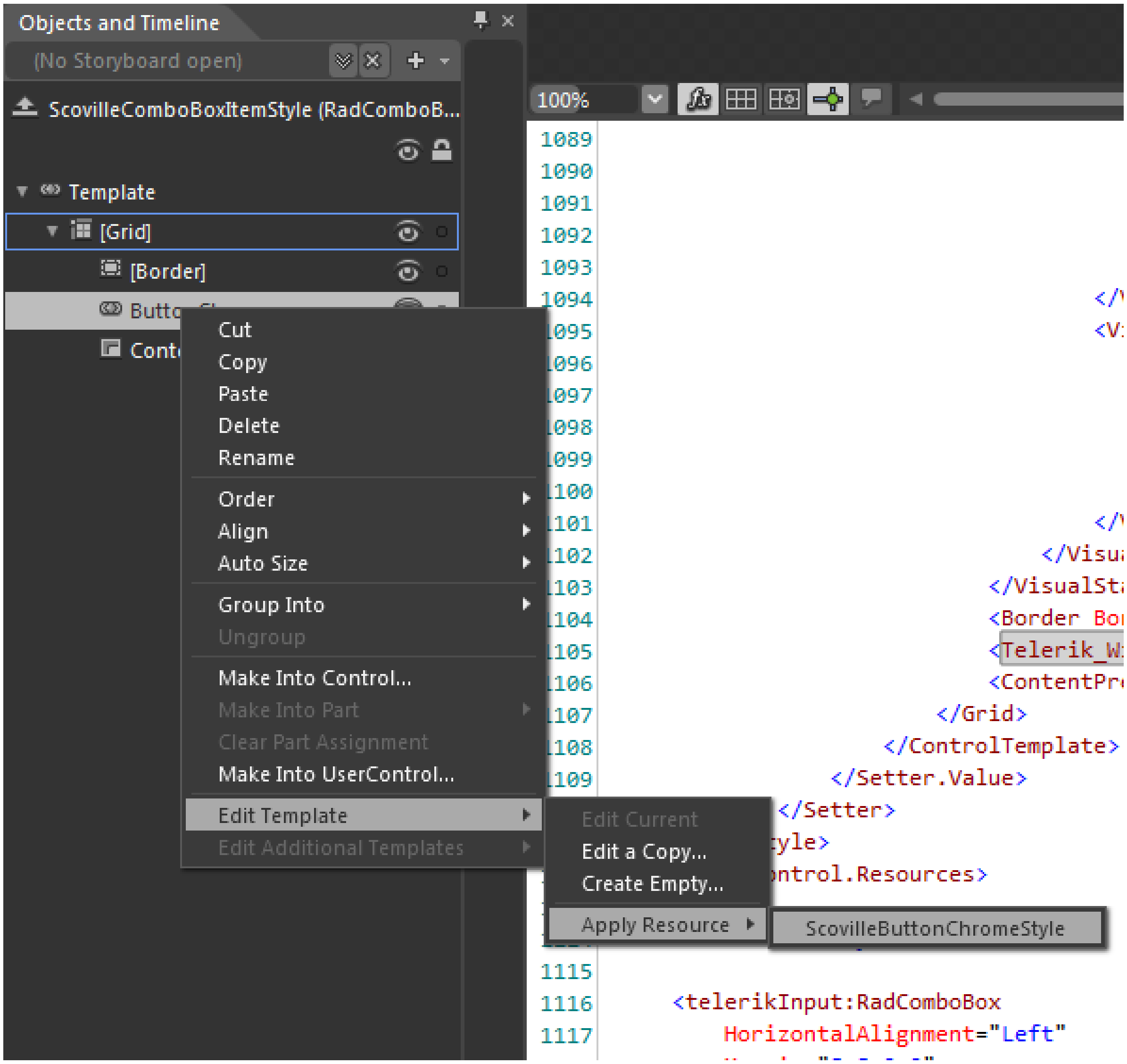
ControlBackground_Active_Stop2
ControlBackground_Active_Stop3
ControlOuterBorder_MouseOver_Stop0
ControlOuterBorder_MouseOver_Stop1
ControlInnerBorder_MouseOver_Stop0
ControlInnerBorder_MouseOver_Stop1
ControlBackground_MouseOver_Stop0
ControlBackground_MouseOver_Stop1
ControlBackground_MouseOver_Stop2
ControlBackground_MouseOver_Stop3
ControlOuterBorder_Highlighted
ControlInnerBorder_Highlighted
ControlBackground_Highlighted



ControlOuterBorder_Pressed_Stop0
ControlOuterBorder_Pressed_Stop1
ControlInnerBorder_Pressed_Stop0
ControlInnerBorder_Pressed_Stop1
ControlBackground_Pressed_Stop0
ControlBackground_Pressed_Stop1
ControlBackground_Pressed_Stop2
ControlBackground_Pressed_Stop3
ControlInnerBorder_Disabled
ControlOuterBorder_Normal_Stop0
ControlOuterBorder_Normal_Stop1
ControlInnerBorder_Normal_Stop0
ControlInnerBorder_Normal_Stop1

- Edit the border brushes.

- Change the background of the individual RadComboBoxItem instances. Here are some hints to get you started. In the Objects and Timeline pane, make a copy of the RadComboBoxItem template. It has the same structure as the one of the RadComboBox itself. Find the ButtonChrome element, right-click it and choose **Edit Template > Apply Resource > ScovilleButtonChromeStyle**. That will apply the same style of the ButtonChrome used for the ComboBox. If you want to define different look for each item you will have to generate ButtonChrome templates for each one.



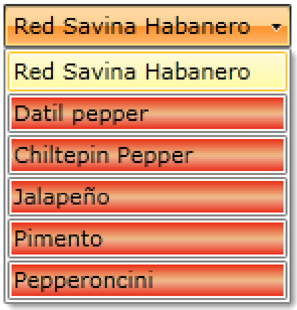
```
</\
<V:
</\
</Visu
</VisualSt
<Border Bo
<Telerik_W:
<ContentPr
</Grid>
</ControlTemplate>
</Setter.Value>
</Setter>
yle>
ntrol.Resources>

<telerikInput:RadComboBox
    HorizontalAlignment="Left"
```

NOTE: Please make sure that the ButtonChrome in the ScovilleComboBoxItemStyle is defined like so:

```
<Telerik_Windows_Controls_Chromes:ButtonChrome x:Name="ButtonChrome" CornerRadius="{StaticResource SplitButt
RenderMouseOver="{Binding IsMouseOver, ElementName=PART_DropDownButton}"
RenderFocused="{TemplateBinding IsFocused}" RenderEnabled="{TemplateBinding IsEnabled}"
```

```
telerik:StyleManager.Theme="{StaticResource Theme}" Style="{StaticResource ScovilleButtonChromeStyle}"/>
```

The resulting styled items end up looking something like the screenshot below:



16.7 Wrap Up

In this chapter you built a RadComboBox and its items using XAML and in code. You learned how to use the Content property assignment for simple text lists and how to bind to templates for more complex layout. You learned how to handle the SelectionChanged event of the combo box and the Selection event of the individual items. In the process, you found out how to get and set the selected RadComboBoxItem or custom object. You also had a brief look at events that respond to drop down opening and closing.

You learned how various edit modes control the combo box Autocomplete, text search, read-only and editing behavior. You learned how to search for items and determine if a specific item exists in a combo box. You used ItemTemplate and SelectionBoxTemplate to compose multiple Silverlight elements. You also bound template elements to live data. Along the way you discovered a few helpful classes in the System. ServiceModel.Syndication namespace for working with RSS feeds. Finally, you used Expression Blend to style the background color of a RadComboBox.

Part XVI

TreeView

17 TreeView

17.1 Objectives

In this chapter we will cover a wide range of tasks that exercise many of the RadTreeView features including defining trees manually, using the API to add/remove/enable/select nodes, locating and accessing nodes, adding images to nodes, handling node expansion and reacting to node selection. You will also define trees with mixed groups of radio buttons and checkboxes. You will learn how to work with drag-and-drop operations, both to enable simple drag-and-drop functionality and to fine-tune the behavior based on multiple conditions such as source and target nodes and the state of the data associated with a node.

You will bind the tree to a simple list of data as well as bind specific nodes to data sources. Then you will use Hierarchical Templates to organize data and present a specific appearance based on the level of data. You will learn how to use Template Selectors to choose templates on-the-fly. You will use the load-on-demand feature to quickly load only the visible nodes of the tree view.

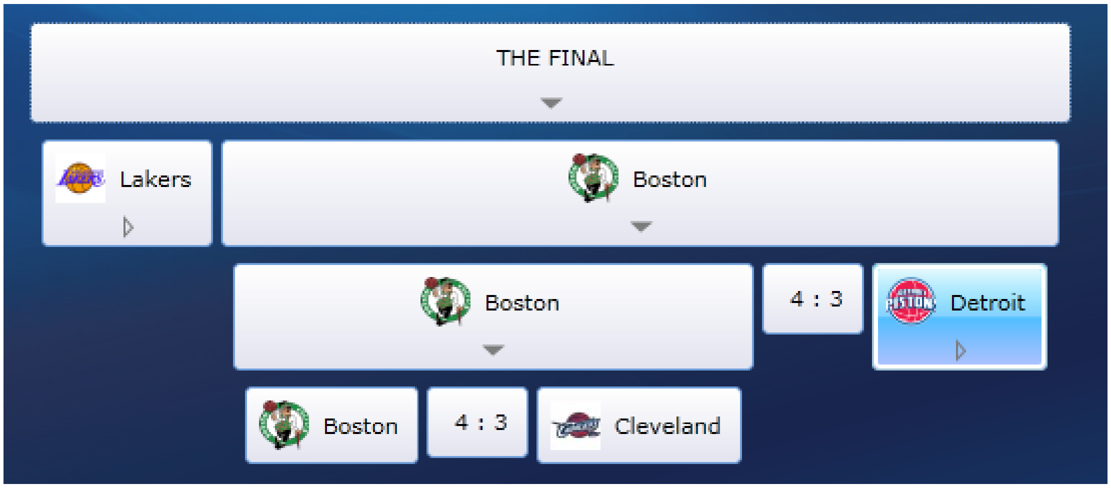
Finally, you will learn how to customize the appearance of individual nodes using Expression Blend.

Find the projects for this chapter at...

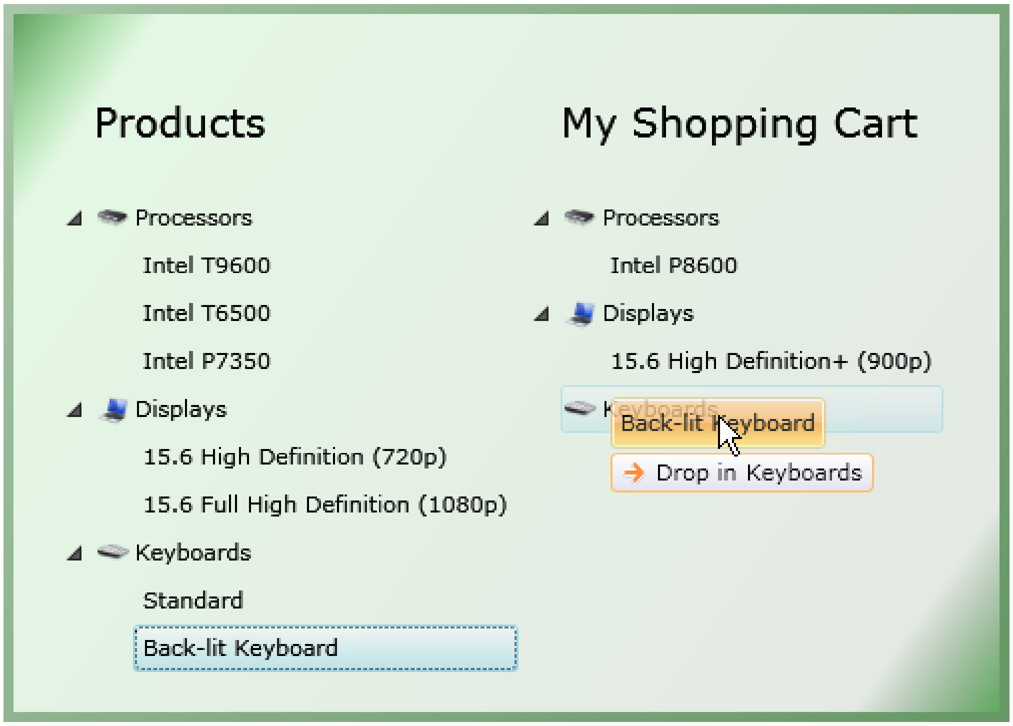
\Courseware\Projects\<CS|VB>\Treeview\Treeview.sln

17.2 Overview

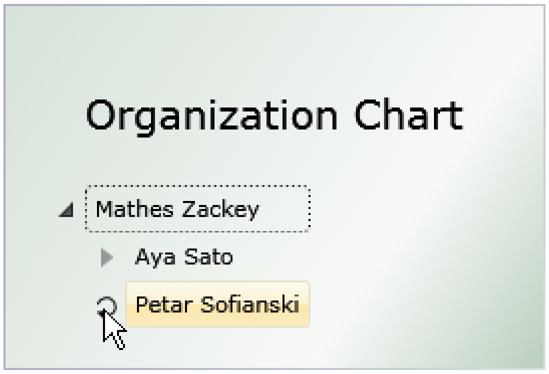
RadTreeView provides a powerful platform for building complex hierarchical navigation systems that your clients will find intuitive and fun to use. The "lookless" nature of the control lets you build unique interfaces like this team playoff example displayed in a horizontal tree.

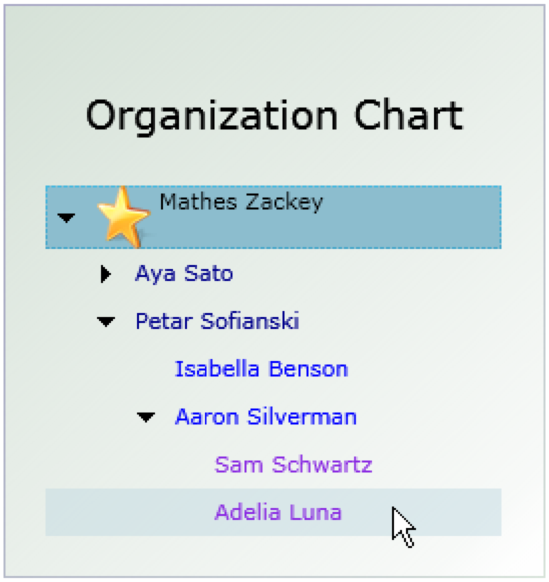


The tree view's drag-and-drop capability interacts automatically within the same tree or other tree views. The drag-and-drop functionality is based on a built-in RadDragAndDropManager, so you can drag between the tree view and any other Silverlight element.

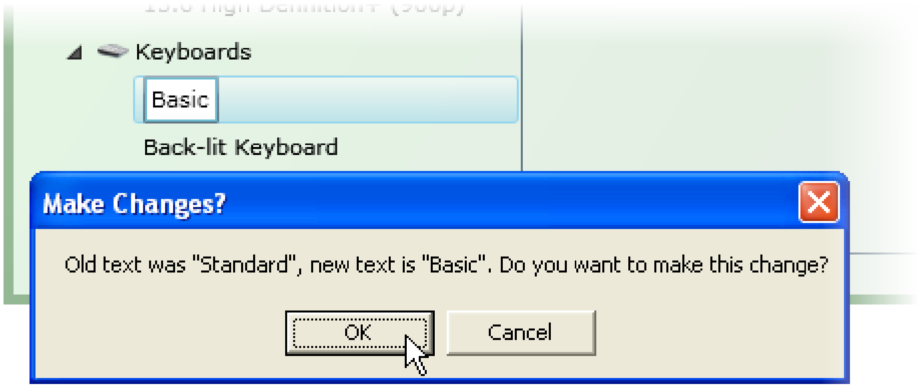


You can bind to hierarchical data such as file directories or relational data. Bind the entire tree view or mix-and-match statically defined nodes and individual nodes bound to collections of disparate data. Enable the "load-on-demand" feature for performant loading of only visible nodes. Hierarchical templates allow you to paint each level within a hierarchy using a completely unique rendering. Template selectors choose the look for each node on-the-fly at runtime based on the node's level within the hierarchy, the data in the node, or any other criteria you can think up.



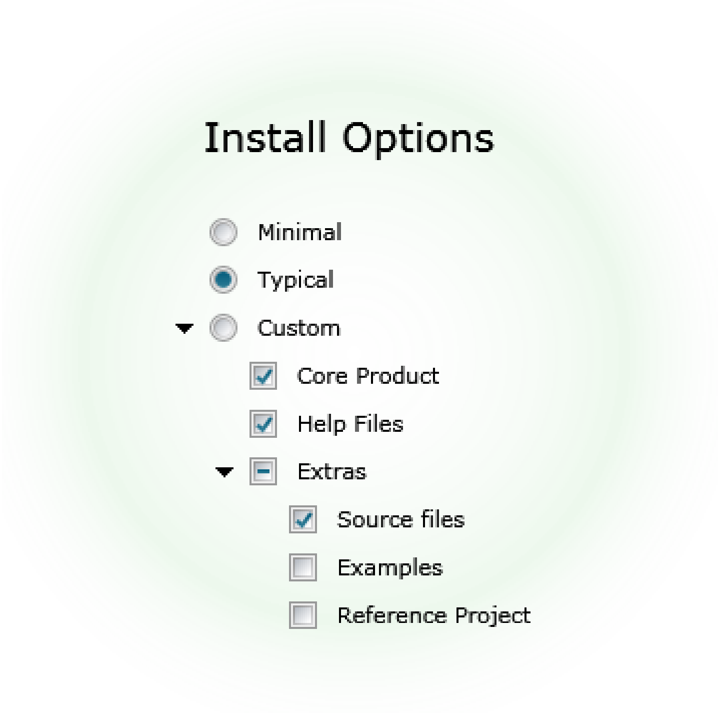


Nodes can be edited by the user or programmatically in the code behind. The rich event model lets you control the flow throughout the editing cycle.



Old text was "Standard", new text is "Basic". Do you want to make this change?

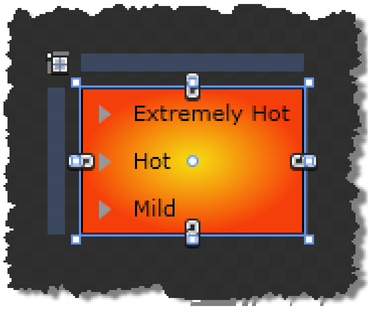
RadTreeView supports mixed collections of radio buttons and check boxes. "Tri-state" check boxes represent intermediate states where child items are both checked and unchecked.



Install Options

- ○ Minimal
- ● Typical
- ○ Custom
  - ☑ Core Product
  - ☑ Help Files
  - ⊟ Extras
    - ☑ Source files
    - □ Examples
    - □ Reference Project

As with all the RadControls for Silverlight, RadTreeView plays nicely in Expression Blend where you can make simple changes to brush colors or to make deep changes to the tree view makeup.



17.3 Getting Started

This walk through will demonstrate the basics of defining a RadTreeView with root level nodes and child nodes of the root level. The example will show how to respond to user selections and retrieve the text for a selected node.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Navigation**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags.

```
<Border Margin="10"
        CornerRadius="5"
        BorderBrush="Red"
        BorderThickness="1"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        MinHeight="500"
        MinWidth="500"
        >

  <!--Tree view goes here-->

</Border>
```

3) Drag a **RadTreeView** from the Toolbox to a point under the comment "<!--Tree view goes here-->". Set the **x:Name** attribute to "tvMain". Add a **SelectionChanged** event handler.

4) Drag a **RadTreeViewItem** from the toolbox to a point within the RadTreeView tags. Set the **Header** attribute to "Extremely Hot".

5) Add two more RadTreeViewItems below the first and set the **Header** attributes to "Hot" and "Mild", respectively.

6) Inside the first RadTreeViewItem ("Extremely Hot"), add three more RadTreeViewItems and set the **Header** attributes to

a) "Pure capsaicin"

b) "Naga Jolokia"

c) "Red Savina Habanero"

7) Inside the second RadTreeViewItem ("Hot"), add three more RadTreeViewItems and set the Header attributes to

a) "Cayenne Pepper"

b) "Tabasco"

c) "Chipotle"

8) Inside the third RadTreeViewItem ("Mild"), add three more RadTreeViewItems and set the Header attributes to

a) "Bell Pepper"

b) "Pimento"

c) "Peperoncini"

The XAML markup should now look like the example below:

```
<telerik:RadTreeView x:Name="tvMain"
                     SelectionChanged="tvMain_SelectionChanged">

  <telerik:RadTreeViewItem Header="Extremely Hot">
    <telerik:RadTreeViewItem Header="Pure capsaicin" />
    <telerik:RadTreeViewItem Header="Naga Jolokia" />
    <telerik:RadTreeViewItem Header="Red Savina Habanero" />
  </telerik:RadTreeViewItem>

  <telerik:RadTreeViewItem Header="Hot">
    <telerik:RadTreeViewItem Header="Cayenne Pepper" />
    <telerik:RadTreeViewItem Header="Tabasco" />
    <telerik:RadTreeViewItem Header="Chipotle" />
  </telerik:RadTreeViewItem>

  <telerik:RadTreeViewItem Header="Mild">
    <telerik:RadTreeViewItem Header="Bell Pepper" />
    <telerik:RadTreeViewItem Header="Pimento" />
    <telerik:RadTreeViewItem Header="Peperoncini" />
  </telerik:RadTreeViewItem>

</telerik:RadTreeView>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **Telerik.Windows.Controls**

2) Add code for the SelectionChanged event handler.

This code gets a reference to the RadTreeView and it's SelectedItem property. SelectedItem is an object type. Because we've added RadTreeViewItems directly (and haven't bound the tree view), we can cast SelectedItem as RadTreeViewItem to use its properties, e.g. Header. Note that if the TreeView is bound, the selected item will be a business object. Use the Header ToString() method and display the results in a MessageBox.

VB

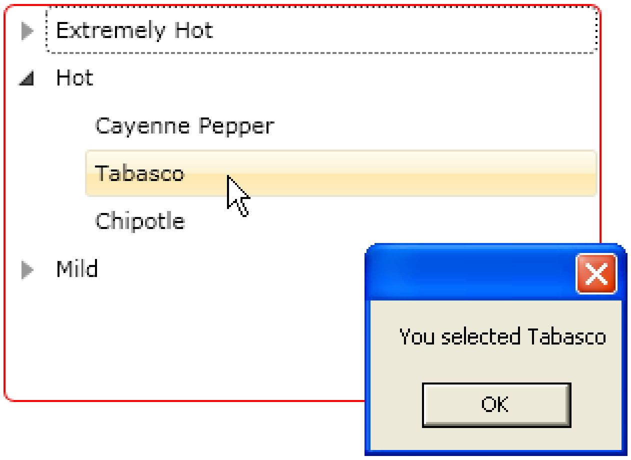
```
Private Sub tvMain_SelectionChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  Dim item As RadTreeViewItem = _
TryCast((TryCast(sender, RadTreeView)).SelectedItem, RadTreeViewItem)
  MessageBox.Show("You selected " & item.Header.ToString())
End Sub
```

C#

```
private void tvMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   RadTreeViewItem item =
      (sender as RadTreeView).SelectedItem as RadTreeViewItem;
   MessageBox.Show("You selected " + item.Header.ToString());
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



You selected Tabasco

Test Application Features

1) Clicking an item should fire the SelectionChanged event and display the MessageBox with the text for the selected item. The event should also fire when using the keyboard Arrow keys to navigate the tree view.
2) Expanding or collapsing nodes should not fire the SelectionChanged event.

17.4 Control Details

17.4.1 Working with Nodes

17.4.1.1 Adding Nodes

Adding a node to the tree view is as simple as creating a new **RadTreeViewItem** object and including it in the tree view **Items** collection. The example below creates three instances of RadTreeViewItem and adds them as root nodes to the Items collection.

VB

```
Private Sub AddNodes()
  ' add root level nodes
  tvMain.Items.Add(New RadTreeViewItem() With { _
.Name = "tviExtremelyHot", .Header = "Extremely Hot"})

  tvMain.Items.Add(New RadTreeViewItem() With {_
.Name = "tviHot", .Header = "Hot"})

  tvMain.Items.Add(New RadTreeViewItem() With {_
.Name = "tviMild", .Header = "Mild"})
End Sub
```

C#

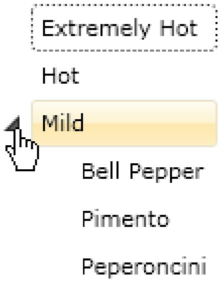
```
private void AddNodes()
{
   // add root level nodes
   tvMain.Items.Add(new RadTreeViewItem()
   {
      Name = "tviExtremelyHot",
      Header = "Extremely Hot"
   });

   tvMain.Items.Add(new RadTreeViewItem()
   {
      Name = "tviHot",
      Header = "Hot"
   });

   tvMain.Items.Add(new RadTreeViewItem()
   {
      Name = "tviMild",
      Header = "Mild"
   });
}
```

Each RadTreeViewItem has its own Items collection and can be used to create a hierarchy in code. The example below gets a reference to one of the root level nodes and adds three RadTreeViewItem instances to the Items collection.

```
Private Sub AddSubNodes()
      ' find one of the root nodes
      Dim mild As RadTreeViewItem = TryCast(tvMain.Items(2),
RadTreeViewItem)

      ' then add items to the root level node
      mild.Items.Add(New RadTreeViewItem() With { _
.Name = "tviBellPepper", .Header = "Bell Pepper"})
      mild.Items.Add(New RadTreeViewItem() With { _
.Name = "tviPimento", .Header = "Pimento"})
      mild.Items.Add(New RadTreeViewItem() With { _
.Name = "tviPeperoncini", .Header = "Peperoncini"})
End Sub
```

```
private void AddSubNodes()
{
  // find one of the root nodes
  RadTreeViewItem mild =
    tvMain.Items[2] as RadTreeViewItem;

  // then add items to the root level node
  mild.Items.Add(new RadTreeViewItem()
  {
    Name = "tviBellPepper",
    Header = "Bell Pepper"

  });
  mild.Items.Add(new RadTreeViewItem()
  {
    Name = "tviPimento",
    Header = "Pimento"
  });
  mild.Items.Add(new RadTreeViewItem()
  {
    Name = "tviPeperoncini",
    Header = "Peperoncini"
  });
}
```

The result of using code to add the root level nodes and sub-nodes looks like the screenshot below:



17.4.1.2 Locating and Accessing Nodes

You can locate and access nodes in a variety of ways including using the index of the RadTreeViewItem within the Items collection or using LINQ expressions against the Items collection. To use LINQ expressions, remember to add the **System.Linq** namespace to your "Imports" (VB) or "using" (C#) section of code.

The code below demonstrates finding a particular node.

VB

```
' locating by index
Dim hotItem As RadTreeViewItem = _
TryCast(tvMain.Items(1), RadTreeViewItem)

' using LINQ expressions
Dim mildItem As RadTreeViewItem = ( _
    From i As RadTreeViewItem In tvMain.Items _
    Where i.Name.Equals("tviMild") _
    Select i).FirstOrDefault()
```

C#

```
// locating by index
RadTreeViewItem hotItem =
   tvMain.Items[1] as RadTreeViewItem;

// using LINQ expressions
RadTreeViewItem mildItem =
   (from RadTreeViewItem i in tvMain.Items
    where i.Name.Equals("tviMild")
    select i).FirstOrDefault();
```

Gotcha!

The Items property only contains the immediate children of an items control, so you have to search recursively to handle n-level searches. The example below searches recursively, given an ItemCollection and a string to search for. This example searches on the Header property, not the Name.

```
' recursively search
Public Function FindItemByHeaderRecursive( _
ByVal nodes As ItemCollection, ByVal searchFor As String) As RadTreeViewItem
  For Each item As RadTreeViewItem In nodes
    If item.Header.Equals(searchFor) Then
      Return item
    End If

    If item.Items.Count > 0 Then
      Dim result As RadTreeViewItem = _
FindItemByHeaderRecursive(item.Items, searchFor)
      If result IsNot Nothing Then
        Return result
      End If
    End If
  Next item
  Return Nothing
End Function

'. . . calling the method
pimento = FindItemByHeaderRecursive(tvMain.Items, "Pimento")
```

```
// recursively search
public RadTreeViewItem FindItemByHeaderRecursive(
   ItemCollection nodes, string searchFor)
{
  foreach (RadTreeViewItem item in nodes)
   {
      if (item.Header.Equals(searchFor))
         return item;

      if (item.Items.Count > 0)
      {
         RadTreeViewItem result =
            FindItemByHeaderRecursive(item.Items, searchFor);
         if (result != null)
            return result;
      }
   }
   return null;
}

// . . . calling the method
pimento = FindItemByHeaderRecursive(tvMain.Items, "Pimento");
```

17.4.1.3 Path Properties and Methods

Forum question: "I want to be able to expand the tree to specific nodes."

Answer: Use the **ExpandItemByPath()** or **GetItemByPath()** methods. If we have a hierarchy like the small example below and want to expand the "Pimento" node:

Mild
 Bell Pepper
 Pimento
Hot
 Tabasco
 ...

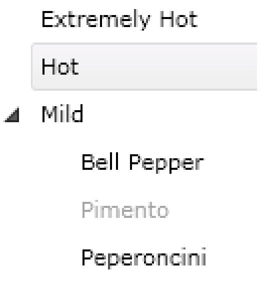
ExpandItemByPath() expands the tree view so that the item in the path is exposed. GetItemByPath() retrieves a reference to the item and also automatically expands the tree view so the item is exposed. The first parameter is the path itself, the second is the separator character.

```
Dim item As RadTreeViewItem = tvMain.GetItemByPath("Mild|Pimento", "|")
item.IsSelected = True
' or...
tvMain.ExpandItemByPath("Mild|Pimento", "|")
```

```
RadTreeViewItem item = tvMain.GetItemByPath("Mild|Pimento", "|");
item.IsSelected = true;
// or...
tvMain.ExpandItemByPath("Mild|Pimento", "|");
```

17.4.1.4 Node Properties

Once you locate a specific node, you can disable, expand, select or remove the node. The example below selects the second node of the tree, expands the third node and disables the "Pimento" node. Use the **IsExpanded**, **IsSelected** and **IsEnabled** RadTreeViewItem properties to toggle these node states.



```
' use nodes references to expand, select, disable and remove nodes
mildItem.IsExpanded = True
hotItem.IsSelected = True
pimento.IsEnabled = False
```

```
// use nodes references to expand, select,
// disable and remove nodes
mildItem.IsExpanded = true;
hotItem.IsSelected = true;
pimento.IsEnabled = false;
```

17.4.1.5 Removing Nodes

To delete a node, use the tree view **Remove()** method and pass the item to be removed or call **RemoveAt()** and pass the index of the item within the Items collection to be removed. The example below removes the same item using both methods. **Note**: This way you can remove children of the RadTreeView but not descendants. They have to be removed from their respective parents.

VB

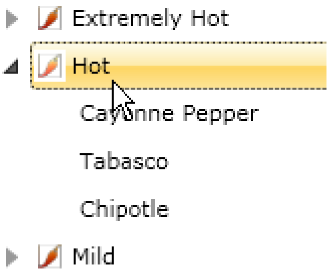
```
' remove the "Hot" node
tvMain.Items.Remove(hotItem)
' or . . .
'tvMain.Items.RemoveAt(1)
```

C#

```
// remove the "Hot" node
tvMain.Items.Remove(hotItem);
// or . . .
//tvMain.Items.RemoveAt(1);
```

17.4.1.6 Node Images

Images can display automatically in response to the default, expanded and selected states of the node. Use the **DefaultImageSrc**, **ExpandedImageSrc**, **SelectedImageSrc** to assign paths to the images using code or XAML. If all images reside in the same folder, assign the folder path to the **ImagesBaseDir** property. Then you can omit the path for your image source properties. The example below sets the default and selected images:



Extremely Hot
Hot
Cayenne Pepper
Tabasco
Chipotle
Mild

The XAML assignment sets the ImagesBaseDir to the path (notice the trailing forward slash). The **DefaultImageSrc** and **SelectedImageSrc** are assigned only the image file name.

```
<telerik:RadTreeView x:Name="tvMain"
        ImagesBaseDir="images/">
    <telerik:RadTreeViewItem Header="Extremely Hot"
            DefaultImageSrc="peppers.png"
            SelectedImageSrc="peppers_selected.png">
. . .
</telerik:RadTreeView>
```

In code, you can assign the image paths at any time. The image source properties all take either a path or a **ImageSource** instance, as shown in the example below.

VB

```
tvMain.ImagesBaseDir = "images/"

' add root level nodes
tvMain.Items.Add(New RadTreeViewItem() With { _
.Name = "tviExtremelyHot",  _
.Header = "Extremely Hot",  _
.DefaultImageSrc = "peppers.png",  _
.SelectedImageSrc = "peppers_selected.png"})

tvMain.Items.Add(New RadTreeViewItem() With { _
.Name = "tviHot",  _
.Header = "Hot",  _
.DefaultImageSrc =  _
New BitmapImage( _
New Uri("images/peppers.png", UriKind.RelativeOrAbsolute)),  _
.SelectedImageSrc =  _
New BitmapImage( _
New Uri("images/peppers_selected.png", UriKind.RelativeOrAbsolute))})
```

C#

```
tvMain.ImagesBaseDir = "images/";

// add root level nodes
tvMain.Items.Add(new RadTreeViewItem()
{
   Name = "tviExtremelyHot",
   Header = "Extremely Hot",
   DefaultImageSrc = "peppers.png",
   SelectedImageSrc = "peppers_selected.png"
});

tvMain.Items.Add(new RadTreeViewItem()
{
   Name = "tviHot",
   Header = "Hot",
   DefaultImageSrc =
      new BitmapImage(
         new Uri("images/peppers.png",
            UriKind.RelativeOrAbsolute))
      ,
   SelectedImageSrc =
      new BitmapImage(
         new Uri("images/peppers_selected.png",
            UriKind.RelativeOrAbsolute))
});
```

17.4.2 Selections

By default, you can select one node at a time in the tree view. The SelectionMode property, first introduced in the Date, Time and Calendar chapter, can be Single, Multiple or Extended.

Note: The TreeView interprets **SelectionMode** “Multiple” as “Extended”, i.e. there are just two selection modes “Single” and “Extended”. The same enum is used for convenience.

- In **Single SelectionMode** (the default), only node data can be selected at a time. Click a node with the mouse or press the space bar to toggle selection. Use the arrow keys to move the selection.
- The **Extended SelectionMode** allows any number of nodes to be selected, but the behavior is similar to Windows Explorer. Holding the Shift key down allows a range of nodes to be selected with the mouse or keyboard (using the arrow keys and space bar). The Control key allows individual nodes to be selected even when they are not part of a continuous range.

Each **RadTreeViewItem** has a **IsSelected** property that can be set programmatically or in XAML. As the user clicks nodes in the tree, the **PreviewSelected**, **Selected**, **PreviewUnselected** and **Unselected** routed events fire on the item. The "Preview" events can be canceled. For example, to make sure the user doesn't lose unsaved work when navigating between nodes, you could set the Handled argument to cancel before the selection was complete. The basic idea is sketched out in the code sample below.

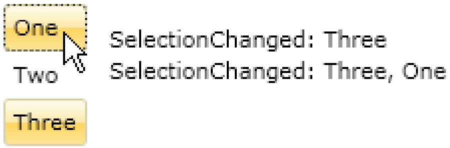
VB

```
' moving off this node
Private Sub tvMain_PreviewUnselected( _
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  ' if changes have occurred, prompt to allow
  If Me.isChanged Then
    If MessageBox.Show("You have made some changes. Do you want to continue?", _
"Alert", MessageBoxButton.OKCancel) = MessageBoxResult.Cancel Then
      ' allowed, so continue
      e.Handled = True
    Else
      ' restore any state here
    End If
  End If
End Sub
```

```
// moving off this node
private void tvMain_PreviewUnselected(object sender,
  Telerik.Windows.RadRoutedEventArgs e)
{
  // if changes have occurred, prompt to allow
  if (this.isChanged)
  {
    if (MessageBox.Show(
      "You have made some changes. Do you want to continue?",
      "Alert",
      MessageBoxButton.OKCancel) == MessageBoxResult.Cancel)
    {
      // allowed, so continue
      e.Handled = true;
    }
    else
    {
      // restore any state here
    }
  }
}
```

To reference selected nodes, use these RadTreeView properties:

- **SelectedItem**: This is the last item selected in the tree view. If you defined a series of RadTreeViewItems directly in XAML or code without binding, then SelectedItem will be a RadTreeViewItem type. If you bound the control to a collection of "MyObject" types, then SelectedItem will be a "MyObject" type.
- **SelectedItems**: A collection of all the items selected, stored in the order they were selected. Again, these will be RadTreeViewItems if not bound in XAML or code. The example below shows a tree view on the left bound to a collection of "MyObject" that contains a "Description" property. The user clicks on the "Three" node, then the "One" node. A text box to the right of the tree view shows the output.



```
SelectionChanged: Three
SelectionChanged: Three, One
```

VB

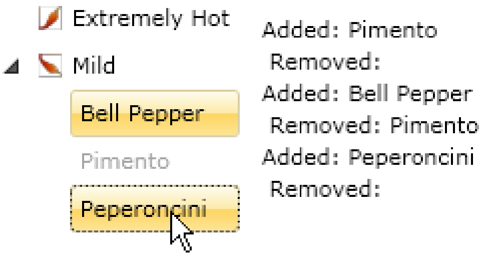
```
Private Sub tvMain_SelectionChanged( _
ByVal sender As Object,
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  ' get the tree view
  Dim treeView As RadTreeView = TryCast(sender, RadTreeView)

  ' get a collection of strings from the "Description"
  ' property of "MyObject".
  Dim descriptions = _
    From myObject As MyObject In treeView.SelectedItems _
    Select myObject.Description

  ' display the event name and list of descriptions
  tbLog.Text &= "SelectionChanged: " &  _
String.Join(", ", descriptions.ToArray()) & Environment.NewLine
End Sub
```

C#

```
void tvMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   // get the tree view
   RadTreeView treeView = sender as RadTreeView;

   // get a collection of strings from the "Description"
   // property of "MyObject".
   var descriptions = from MyObject myObject in treeView.SelectedItems
                 select myObject.Description;

   // display the event name and list of descriptions
   tbLog.Text += "SelectionChanged: " +
      String.Join(", ", descriptions.ToArray()) + Environment.NewLine;
}
```

- **SelectedContainer:** This is the RadTreeViewItem that contains a bound object.

In addition to these properties, the **SelectionChanged** event supplies a **SelectionChangedEventArgs** that contains two collections **AddedItems** and **RemovedItems**. Instead of returning all objects, these two properties keep track of that items that were just added or removed in response to the event. The example below displays the results as items are selected in the list.



```
Added: Pimento
 Removed:
Added: Bell Pepper
 Removed: Pimento
Added: Peperoncini
 Removed:
```

```
Private Sub tvMain_SelectionChanged( _
ByVal sender As Object,  _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  Dim added = _
    From item As RadTreeViewItem In e.AddedItems _
    Select item.Header.ToString()

  Dim removed = _
    From item As RadTreeViewItem In e.RemovedItems _
    Select item.Header.ToString()

  tbLog.Text &= "Added: " &  _
String.Join(", ", added.ToArray()) &  _
Environment.NewLine & " Removed: " &  _
String.Join(", ", removed.ToArray())  _
& Environment.NewLine
End Sub
```

```
void tvMain_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   var added =
      from RadTreeViewItem item in e.AddedItems
      select item.Header.ToString();

   var removed =
      from RadTreeViewItem item in e.RemovedItems
      select item.Header.ToString();

   tbLog.Text += "Added: " +
      String.Join(", ", added.ToArray()) +
      Environment.NewLine +
      " Removed: " +
      String.Join(", ", removed.ToArray()) +
      Environment.NewLine;
}
```

17.4.3 Node Expansion

Expanding an individual item is simply a matter of setting the **IsExpanded** property to true. To expand or collapse all items at all levels in the tree, call the **ExpandAll()** and **CollapseAll()** methods. ExpandAll() and CollapseAll() are available not only for the RadTreeView, but for any RadTreeViewItem so that you can expand or collapse the item and all its children recursively.

When an item expands or collapses, the **PreviewExpand**, **Expanded**, **PreviewCollapsed** and **Collapsed** events fire. The "Preview" events can be canceled by setting the **Handled** argument property to True.

VB

```
Private Sub tvMain_PreviewExpanded( _
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  If Me.role <> Role.Administrator Then
    e.Handled = True
    MessageBox.Show("You do not have permissions to this information")
  End If
End Sub
```

C#

```
private void tvMain_PreviewExpanded(object sender,
   Telerik.Windows.RadRoutedEventArgs e)
{
   if (this.role != Role.Administrator)
   {
      e.Handled = true;
      MessageBox.Show("You do not have permissions to this information");
   }
}
```

Here are a few properties that fine-tune expand behavior:

- To only allow expanding one node at a time, set the **IsSingleExpandPath** property to True. When True, the tree view will automatically collapse one branch before expanding another.
- By default you can click an item to toggle expansion. To allow expansion only when the expansion arrow is clicked, set the **IsExpandOnSingleClickEnabled** and **IsExpandOnDblClickEnabled** properties False.

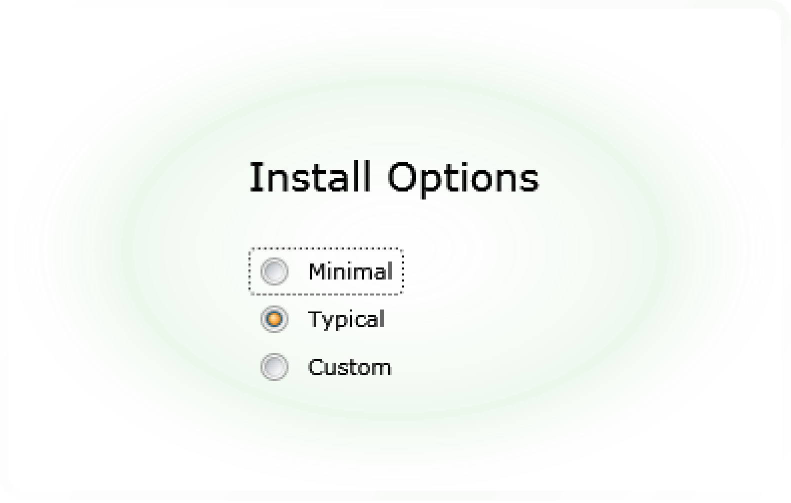
17.4.4 Checkboxes and Radiobuttons

You can build complex arrangements of checks and radio buttons, show "tri-state" check boxes that visually indicate "Indeterminate" states, and use events to completely control checking and unchecking behavior.

Start by setting the RadTreeView **IsOptionElementsEnabled** property to true. After that, you can set the **ItemsOptionListType** property for the tree view as a whole, or for any individual RadTreeViewItem, to **OptionList** or **CheckList**. **OptionList** displays radio buttons and **CheckList** displays check boxes. The minimal XAML snippet below displays a list of radio buttons and also introduces the **CheckState** property, that sets the check mark **On**, **Off** or to an **Indeterminate** state.

```
<telerik:RadTreeView
    IsOptionElementsEnabled="True"
    ItemsOptionListType="OptionList">
  <telerik:RadTreeViewItem
      Header="Minimal" />
  <telerik:RadTreeViewItem
      Header="Typical" CheckState="On" />
  <telerik:RadTreeViewItem
      Header="Custom" />
</telerik:RadTreeView>
```

The result of the XAML shows in this screenshot:



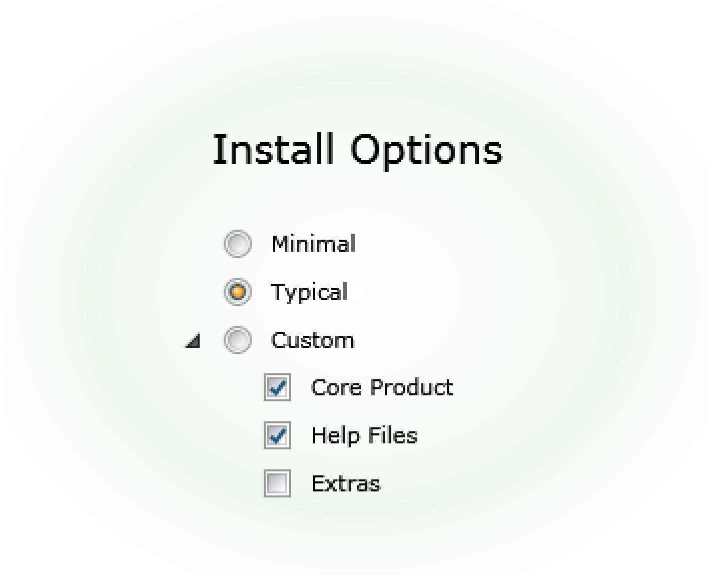
Install Options

○ Minimal
● Typical
○ Custom

The **ItemsOptionListType** property acts against all child nodes of a given node. Using this property, you can display mixed combinations of radio and check boxes. Starting from the previous example, you can add three child nodes to the "Custom" node and set the **ItemsOptionListType** to **CheckList**.

```
<telerik:RadTreeView
    IsOptionElementsEnabled="True"
    ItemsOptionListType="OptionList">
  <telerik:RadTreeViewItem
      Header="Minimal" />
  <telerik:RadTreeViewItem
      Header="Typical" CheckState="On" />
  <telerik:RadTreeViewItem Header="Custom"
      ItemsOptionListType="CheckList"
      IsExpanded="True">
    <telerik:RadTreeViewItem
        Header="Core Product" CheckState="On" />
    <telerik:RadTreeViewItem
        Header="Help Files" CheckState="On" />
    <telerik:RadTreeViewItem
        Header="Extras" CheckState="Off">
    </telerik:RadTreeViewItem>
  </telerik:RadTreeViewItem>
</telerik:RadTreeView>
```

Now you have a tree view that displays radio buttons at the root level and check boxes underneath the "Custom" node.



Install Options

○ Minimal

● Typical

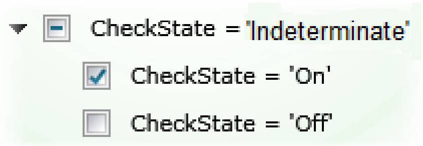
○ Custom

☑ Core Product

☑ Help Files

☐ Extras

"tri-state" refers to the ability of a check box to display not only checked and unchecked, but to show an "Indeterminate" state as well. The "Indeterminate" state indicates to the user that child nodes have a mix of checked and unchecked nodes. The specific visual cues used to represent check states vary, depending on theme and styling, but the screenshot below shows a typical example. To enable the "tri-state" behavior, you only need to set the tree view **IsTriStateMode** property to True.

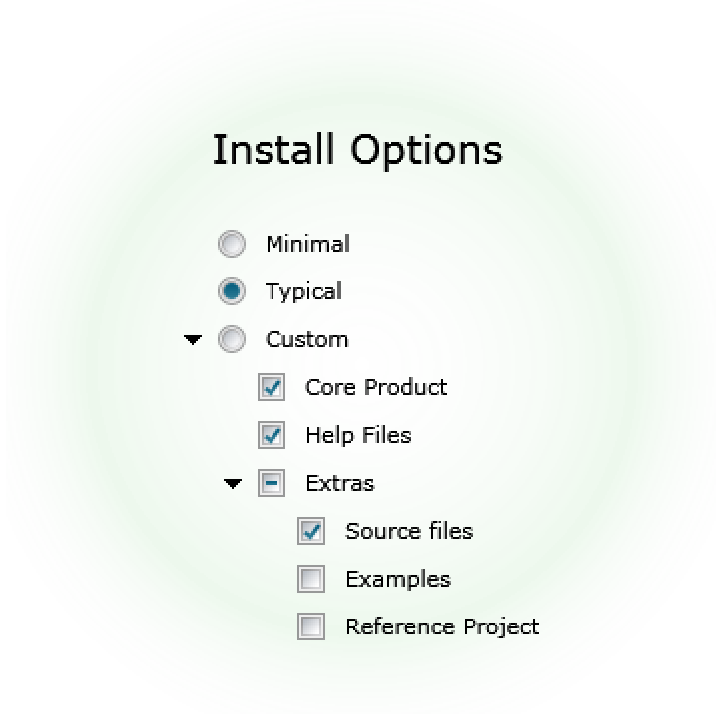


CheckState = 'Indeterminate'
☑ CheckState = 'On'
□ CheckState = 'Off'

Here is one more example with all of the previously described elements in play:

```
<telerik:RadTreeView
    IsOptionElementsEnabled="True"
    ItemsOptionListType="OptionList"
    telerik:StyleManager.Theme="Summer"
    IsTriStateMode="True">
  <telerik:RadTreeViewItem Header="Minimal" CheckState="Off" />
  <telerik:RadTreeViewItem Header="Typical" CheckState="On" />
  <telerik:RadTreeViewItem Header="Custom"
      ItemsOptionListType="CheckList" IsExpanded="True">
    <telerik:RadTreeViewItem Header="Core Product" CheckState="On" />
    <telerik:RadTreeViewItem Header="Help Files" CheckState="On" />
    <telerik:RadTreeViewItem Header="Extras" CheckState = "Indeterminate"
        ItemsOptionListType="CheckList" IsExpanded="True">
      <telerik:RadTreeViewItem Header="Source files"
        CheckState="On" />
      <telerik:RadTreeViewItem Header="Examples"
        CheckState="Off" />
      <telerik:RadTreeViewItem Header="Reference Project"
        CheckState="Off" />
    </telerik:RadTreeViewItem>
  </telerik:RadTreeViewItem>
</telerik:RadTreeView>
```

The screenshot below shows the XAML sample running in the browser:

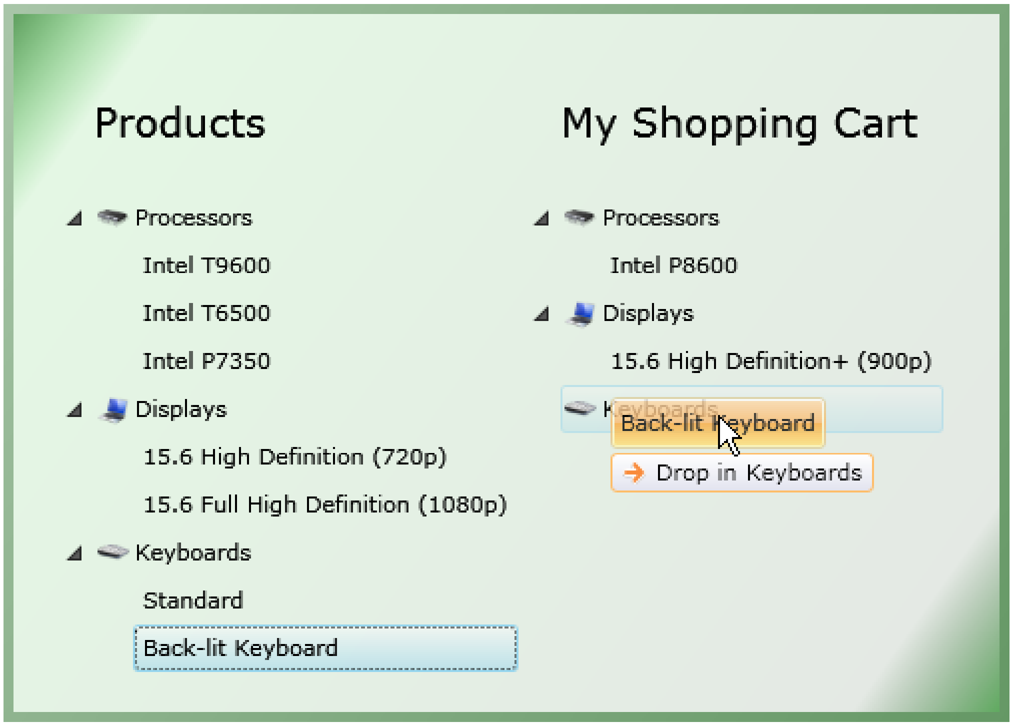


Install Options

- ○ Minimal
- ● Typical
- ▼ ○ Custom
  - ☑ Core Product
  - ☑ Help Files
  - ▼ ⊟ Extras
    - ☑ Source files
    - □ Examples
    - □ Reference Project

17.4.5 Drag-and-Drop

Users can rearrange the tree view using the built-in drag-and-drop capability. The user can drop one or more nodes onto another node, between nodes and onto other tree views. The drag-and-drop infrastructure is built using the RadDragAndDropManager, so all the RadDragAndDropManager events and properties are available.



Your first move is to set the **IsDragDropEnabled** tree view property to "True". If you want to simply move nodes around in a tree or drag them between trees without any rules, then you're done. But typically there are business rules that govern what can be dragged and where nodes can be dropped. Look at the screenshot above for example. We have a set of possible products that can be dragged to a "Shopping Cart" tree view. So right away we have several implied business rules:

- Root level nodes should not be dragged. The "Processors", "Displays" and "Keyboards" nodes should behave as if fixed into place.
- Nodes with Level == 1 should be dropped under Level 0 nodes. We should only drop the "Standard" keyboard node under one of the root level, "fixed" nodes.
- Nodes that describe particular processors should only be dragged to a root level "Processors" node. Displays should be dragged under the "Displays" node and likewise for keyboards.
- Nodes should not be dropped to the same tree.

In some other application it might be ok to drag the root level nodes around. You have to decide that for yourself and add logic to enforce the behavior. That begs the question, "ok, so how do I enforce the business logic?"

You can get some minimal control using the **IsDropAllowed** property of the **RadTreeViewItem** or take advantage of the **RadTreeView** drag/drop events.

Notes

The **RadTreeView** API offers you four events for managing the drag and drop behavior:

- PreviewDragStarted
- DragStarted
- PreviewDragEnded
- DragEnded

But to get the fine-grain control you really want in a production application, you need to handle RadDragAndDropManager events.
There are two "gotchas" you may run into right away. So, to save you some time:

1. Be sure to add **Telerik.Windows** to the "Imports" (VB) or "using" (C#) sections of code so that the AddHandler() extension method is available.
2. RadTreeView will mark all the events its responds to as "handled", so, without special measures, you will not see any events. Call the **AddHandler()** overload that includes the **handledEventsToo** parameter. Set the handledEventsToo parameter "True" when you also want the drag-and-drop events to fire, even though RadTreeView has already had a crack at them.

Here's what the code looks like for the application shown running in the screenshot above. The AddHandler() extension method registers the DropQueryEvent, then all the nodes of the tree are expanded.

VB

```
Private Sub UserControl_Loaded(_
ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' Gotcha: be sure to add Telerik.Windows to get AddHandler extension method
  ' Gotcha #2, pass "handledEventsToo" as true -- tree view already handles,
  '  so you must pass this to get the event raised.
  Me.AddHandler(RadDragAndDropManager.DropQueryEvent, _
New EventHandler(Of DragDropQueryEventArgs)(AddressOf OnDropQuery), True)

  Dispatcher.BeginInvoke(Function() { tvPossibleOptions.ExpandAll(); })
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   // Gotcha: be sure to add Telerik.Windows to get AddHandler extension method
   // Gotcha #2, pass "handledEventsToo" as true -- tree view already handles,
   //  so you must pass this to get the event raised.
   this.AddHandler(RadDragAndDropManager.DropQueryEvent,
      new EventHandler<DragDropQueryEventArgs>(OnDropQuery), true);

   Dispatcher.BeginInvoke(() =>
    {
      tvPossibleOptions.ExpandAll();
    });
}
```

The OnDropQuery event handler enforces all the business rules we first described. By getting references to the source and destination RadTreeViewItem objects, we can tell what level they belong to, if the source and destination parent tree views are the same (i.e. we're trying to drop back to the same tree we're dragging from) and the position we're attempting to drop into (i.e. inside, before, after). Not shown here is the XAML that includes a Tag property for each RadTreeViewItem where we define a "group" that the node belongs to, i.e. "Processors" or "Keyboards". All of this logic is arbitrary and must be modified to fit your particular business requirements. This example covers a lot of ground though and addresses a number of issues you may run into.

```
Private Sub OnDropQuery( _
ByVal sender As Object, ByVal e As DragDropQueryEventArgs)
  ' get references to the source and destination nodes
  Dim sourceItem As RadTreeViewItem = _
TryCast(e.Options.Source, RadTreeViewItem)
  Dim destinationItem As RadTreeViewItem = _
TryCast(e.Options.Destination, RadTreeViewItem)
  ' we should have valid source and destination items before proceeding
  If (sourceItem IsNot Nothing) AndAlso (destinationItem IsNot Nothing) Then
    ' Nodes should not be dropped to the same tree.
    Dim isSameTree As Boolean = _
sourceItem.ParentTreeView = destinationItem.ParentTreeView
    ' Nodes should only be dragged to the same "group",
    ' i.e. processors to processors, keyboards to keyboards
    Dim isSameNodeType As Boolean = sourceItem.Tag.Equals(destinationItem.Tag)
    ' Only drop a node inside the root, or before/after a level 1 node
    Dim isCorrectDropPosition As Boolean = destinationItem.Level = 0 _
AndAlso destinationItem.DropPosition = DropPosition.Inside _
OrElse destinationItem.Level = 1 _
AndAlso destinationItem.DropPosition = DropPosition.Before _
OrElse destinationItem.Level = 1 _
AndAlso destinationItem.DropPosition = DropPosition.After
    ' if all the above conditions apply and the item
    ' being dragged is Level = 1, then allow dropping
    e.QueryResult = isSameNodeType AndAlso (Not isSameTree) _
AndAlso sourceItem.Level = 1 AndAlso isCorrectDropPosition
  End If
End Sub
```

```
private void OnDropQuery(object sender, DragDropQueryEventArgs e)
{
   // get references to the source and destination nodes
   RadTreeViewItem sourceItem = e.Options.Source as RadTreeViewItem;
   RadTreeViewItem destinationItem = e.Options.Destination as RadTreeViewItem;
   // we should have valid source and destination items before proceeding
   if ((sourceItem != null) &&
      (destinationItem != null))
   {
      // Nodes should not be dropped to the same tree.
      bool isSameTree = sourceItem.ParentTreeView == destinationItem.ParentTreeView;
      // Nodes should only be dragged to the same "group",
      // i.e. processors to processors, keyboards to keyboards
      bool isSameNodeType = sourceItem.Tag.Equals(destinationItem.Tag);
      // Only drop a node inside the root, or before/after a level 1 node
      bool isCorrectDropPosition =
         destinationItem.Level == 0 && destinationItem.DropPosition == DropPosition.Inside ||
         destinationItem.Level == 1 && destinationItem.DropPosition == DropPosition.Before ||
         destinationItem.Level == 1 && destinationItem.DropPosition == DropPosition.After;
      // if all the above conditions apply and the item being dragged is Level = 1, then allow dropping
      e.QueryResult = isSameNodeType && !isSameTree && sourceItem.Level == 1 && isCorrectDropPosition
   }
}
```

You may want a small representative sample of the XAML to refer to. This XAML defines the left-most tree view shown in the screenshot.

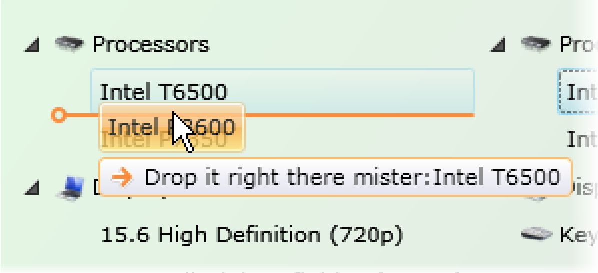
```
<telerik:RadTreeView IsDragDropEnabled="True"
      IsDragPreviewEnabled="True" IsDragTooltipEnabled="True"
      IsDropPreviewLineEnabled="True" Style="{StaticResource TreeViewStyle}" >
   <telerik:RadTreeViewItem Header="Processors"
         DefaultImageSrc="ram.png" Tag="Processor">
      <telerik:RadTreeViewItem
            Header="Intel P8600" Tag="Processor" />
. . .
```

Other properties that govern drag and drop behavior in RadTreeView:

- **IsDragPreviewEnabled**, **IsDragTooltipEnabled**, **IsDropPreviewLineEnabled**: When true, these properties provide visual cues automatically without having to handle RadDragAndDropManager events.
- **TextDropAfter**, **TextDropBefore**, **TextDropIn**, **TextDropRoot**: These are string properties that display as tool tips and are followed automatically by the node text.

```
<telerik:RadTreeView
. . .
 TextDropAfter="Drop it right there mister:"
. . . />
```

The TextDropAfter property set in the XAML above displays like this screenshot example:.



Drop it right there mister:Intel T6500

- **DropExpandDelay**: This is a TimeSpan property that determines the wait before an item is expanded when something is dragged over it.

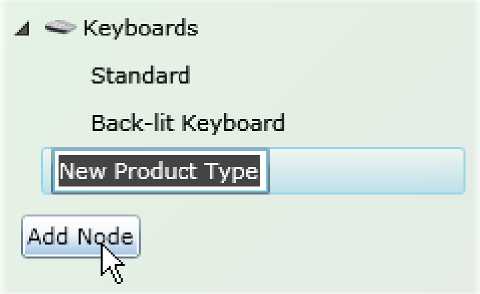
RadTreeViewItem properties that govern drag and drop behavior are:

- **IsDropAllowed**: This property allows specific nodes to refuse a drop.
- **DropPosition**: This is an enumeration that indicates the relative position of the dragged node to the node it's being dropped on and can be **Before**, **After** or **Inside**.

17.4.6 Editing

By default, the nodes of a tree view cannot be edited. Setting the tree view **IsEditable** property to True allows the user to press the **F2** key to edit tree view nodes. After changes are entered, the user can press the **Esc** key to cancel editing or **Enter** to commit their changes.

You can also set the **IsInEditMode** property to True to initiate editing programmatically. For example, if you might want the user to click a button to add a new node to the tree and automatically make that node editable:



First create the RadTreeViewItem and add it to the tree view Items collection. Then set IsInEditMode after the node has been generated.

VB

```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' create an add the item
  Dim item As New RadTreeViewItem() With {.Header = "New Product Type"}
  tvPossibleOptions.Items.Add(item)
  item.IsInEditMode = True
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   // create an add the item
   RadTreeViewItem item = new RadTreeViewItem()
      {
         Header = "New Product Type",
      };
   tvPossibleOptions.Items.Add(item);
   item.IsInEditMode = true;
}
```

Much like the tree view drag-and-drop capability, once you make editing available for the tree view, it's an "open season" and the user can edit anything in the tree without restriction. This won't do for the typical production application. Some nodes should not be editable, or not editable in certain circumstances or should not allow certain input. The **PreviewEditStarted**, **EditStarted**, **PreviewEdited**, **Edited** and **EditCanceled** routed events allow you to enforce business logic. The sequence of event is:

- User presses F2 or IsInEditMode is set to True.
- PreviewEditStarted

- EditStarted
- User presses Enter, focus is moved away from the node, or IsInEditMode is set to False
- PreviewEdited
- Edited

Once the edit is started and if the user presses the escape key or the **CancelEdit()** method is called, the EditCanceled event fires.

As is standard with all Silverlight routed events, the "Preview" events can be canceled by setting the Handled property to "True". For example, if you wanted to prevent root level nodes from being edited, you could check the Level property of the node and set Handled to true if the Level was "0". Notice in the example that you can obtain the item being edited from the parameter arguments Source property.

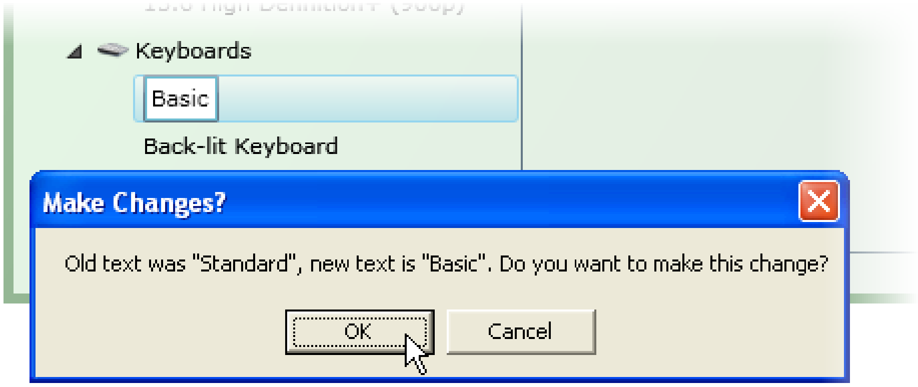
VB

```
Private Sub tvPossibleOptions_PreviewEditStarted( _
ByVal sender As Object, ByVal e As RadRoutedEventArgs)
  ' get a reference to the item being edited
  Dim item As RadTreeViewItem = TryCast(e.Source, RadTreeViewItem)
  ' disallow editing root level items
  e.Handled = item.Level = 0
End Sub
```

C#

```
private void tvPossibleOptions_PreviewEditStarted( _
object sender, RadRoutedEventArgs e)
{
   // get a reference to the item being edited
   RadTreeViewItem item = e.Source as RadTreeViewItem;
   // disallow editing root level items
   e.Handled = item.Level == 0;
}
```

The PreviewEdited and Edited events pass a **RadTreeViewItemEditedEventArgs** as a parameter that includes OldValue and NewValue object properties. The screenshot below shows what happens when you use the PreviewEdited event to display a confirmation dialog:



Old text was "Standard", new text is "Basic". Do you want to make this change?

The MessageBox takes a string containing the new and old text, then sets Handled to True if the user decides they don't want to confirm the edit. When Handled is set to True, the default behavior is that the "New" text stays in place and the node is still available for editing. If you want to restore the data to a previous state, you can set the node's Header property and call CancelEdit().

```
Private Sub tvPossibleOptions_PreviewEdited(ByVal sender As Object, _
ByVal e As RadTreeViewItemEditedEventArgs)
  Dim message As String = _
String.Format("Old text was ""{0}"", new text is ""{1}"". Do you want to make this change?", _
e.OldValue.ToString(), e.NewValue.ToString())

  ' show the user the old and new state of the edited text
  Dim cancel As Boolean = MessageBox.Show(message, "Make Changes?", _
MessageBoxButton.OKCancel) = MessageBoxResult.Cancel

  ' the user wants to cancel, so restore the header to the old text and
  ' stop the edit
  If cancel Then
    TryCast(e.Source, RadTreeViewItem).Header = e.OldText
    TryCast(e.Source, RadTreeViewItem).CancelEdit()
  End If

  e.Handled = cancel
End Sub
```

C#

```
private void tvPossibleOptions_PreviewEdited(object sender, RadTreeViewItemEditedEventArgs e)
{
   string message = String.Format(
      "Old text was \"{0}\", new text is \"{1}\". Do you want to make this change?",
      e.OldValue.ToString(), e.NewValue.ToString());

   // show the user the old and new state of the edited text
   bool cancel =
      MessageBox.Show(message,
      "Make Changes?", MessageBoxButton.OKCancel) == MessageBoxResult.Cancel;

   // the user wants to cancel, so restore the header to the old text and
   // stop the edit
   if (cancel)
   {
      (e.Source as RadTreeViewItem).Header = e.OldText;
      (e.Source as RadTreeViewItem).CancelEdit();
   }

   e.Handled = cancel;
}
```

17.4.7 Keyboard Support

RadTreeView has a rich set of keyboard commands that deserve special mention:

- **Arrow Keys** navigate, one item at a time, within the tree. The left arrow key collapses an item if expanded, otherwise it navigates one item up in the tree. The right arrow key expands an item or travels one item down in the tree.
- **PageUp, PageDown** keys page through items in the view port area.
- **End, Home** keys navigate to last or first visible item in the tree.
- **Enter** toggles the collapsed/expanded state of an item. If the tree is in edit mode, pressing the Enter key confirms the new item value.
- **Esc** cancels an edit or drag operation.
- **Add, Subtract** expands or collapses a selected item.
- **Multiply, Divide** expands or collapses all child items.
- **F2** puts an item into edit mode.
- **Shift** is used to select multiple contiguous items.
- **Ctrl** is used to select multiple items that may not be contiguous.

17.4.8 Performance

At the time of this writing, tree view aspects relating to performance are in flux, but this forum entry may help your planning.

From the Forums...

Question: Are there some tweaks I can make to help improve performance of the TreeView?

Answer: Generally, you can think of performance in the following terms:

1. A smart use of templates/control logic can achieve good performance for items in the hundreds (1-1000). Currently the Template of the TreeViewItems has a lot of elements that you may not need - for example a loading animation, disabled state visual, checkbox, radio button, lines, edit presenter, etc. The total number of visual elements for a single TreeView item is 25 and could be reduced to approx 5-6.
2. UI Virtualization can increase the usable items count to thousands - 1000-10000. See the UI Virtualization section below.
3. Data Virtualization is needed for more items, going to 1 mil. for example.

UI Virtualization

RadTreeView supports virtualization to allow large amounts of nodes to be presented without performance loss. Things to keep in mind:

- The **IsVirtualizing** property needs to be set to true.
- Containers will be generated only for the visible items.
- The **GetContainerByItem()** method becomes expensive and should not be called in a cycle. The tree view needs to do some “guessing” where the container of an item is, which may take time if the tree view has deep jagged hierarchy.
- Containers for expanded items are currently ***not*** virtualized. There are some issues in Silverlight currently that demand this, so keeping hundreds of expanded items will hurt performance.
- The virtualization has three modes: Standard, Recycling and Hierarchical. The “Recycling” is the default. It creates a visual cache of items per-ItemsControl. This way scrolling is faster but the tree view takes more memory. For “Standard” the TreeView should take less memory but scroll slower. (i.e. this is the memory-cpu tradeoff with caches). When you use "Hierarchical" mode, then all items that are out of view will be virtualized, including the expanded items. Containers are cached and reused at a TreeView level. This mode is suited for indented hierarchies and fully expanded trees. Scrolling may be slower in longer lists (collapsed trees) but faster when the tree is expanded.

The mode is set through the TreeViewPanel **VirtualizationMode** attached property found in the **Telerik.Windows.Controls.TreeView** namespace.

XAML

```
<telerik:RadTreeView
  IsVirtualizing="True"
  telerk:TreeViewPanel.VirtualizationMode="Recycling" ...
```

Animation

You can also disable animation like so:

```
<telerik:RadTreeView telerik:AnimationManager.IsAnimationEnabled="False" />
```

Where the ***telerik*** namespace is:

```
 xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
```

17.5 Binding

17.5.1 Basic Binding

Minimally, you can assign an **IEnumerable** collection to the RadTreeView **ItemsSource** property to bind it. For example, it's perfectly OK to assign a collection of strings:

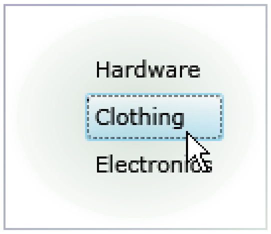
VB

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim categories As ObservableCollection(Of String) = _
 New ObservableCollection(Of String) (New String() _
{"Hardware", "Clothing", "Electronics"})
  tvMain.ItemsSource = categories
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   ObservableCollection<string> categories =
     new ObservableCollection<string>()
   {
      "Hardware", "Clothing", "Electronics"
   };
   tvMain.ItemsSource = categories;
}
```

The bound tree view might show up in the browser like the screenshot below. You will be able to select nodes, edit nodes and perform drag-and-drop operations on the bound data.

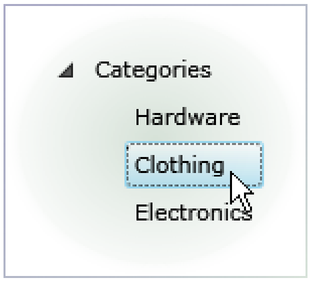


Gotcha!

"Drag-and-Drop doesn't work when I bind the tree view, why"?

This question comes up in various guises quite often. If you want the tree view to track changes in the collection, you must bind to an ObservableCollection, not a generic List or other collection. Also, if you want property changes to show up, the objects in the collection need to implement the INotifyPropertyChanged interface.

Not only can you bind the tree view, you can bind each of the nodes. The screenshot below shows a single "Categories" node added to the root of the tree view with our bound items underneath the root node.



The code snippet for this example adds a single RadTreeViewItem with header "Categories" to the Items collection of the tree view, then binds a collection underneath the programmatically defined node using the node's ItemsSource property

.

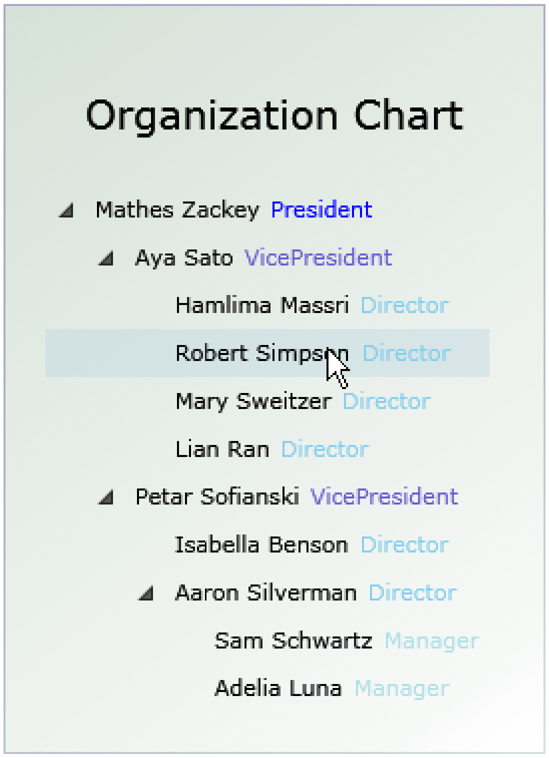
VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim item As New RadTreeViewItem() With {.Header ="Categories"}
  tvMain.Items.Add(item)

  item.ItemsSource = New ObservableCollection(Of String) _
(New String() {"Hardware", "Clothing", "Electronics"})
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   RadTreeViewItem item = new RadTreeViewItem() { Header ="Categories" };
   tvMain.Items.Add(item);

   item.ItemsSource =
      new ObservableCollection<string>()
      {
         "Hardware", "Clothing", "Electronics"
      };
}
```

17.5.2 Hierarchical Templates

Of course, it's not worthwhile to bind a tree view just to get a flat list. Tree views are all about hierarchical data. For that you need to use **HierarchicalDataTemplates** as previously discussed in the "Menu Controls" chapter. A quick reminder on hierarchical templates work: The RadTreeView ItemTemplate points to a HierarchicalDataTemplate. Each HierarchicalDataTemplate ItemTemplate attribute points up to another HierarchicalDataTemplate except the last template that has no children. This last template is a DataTemplate. Take a look at the example below that represents a corporate hierarchy with President/Vice President, Director and Manager.



Organization Chart

- ◢ Mathes Zackey President
  - ◢ Aya Sato VicePresident
    - Hamlima Massri Director
    - Robert Simpson Director
    - Mary Sweitzer Director
    - Lian Ran Director
  - ◢ Petar Sofianski VicePresident
    - Isabella Benson Director
    - ◢ Aaron Silverman Director
      - Sam Schwartz Manager
      - Adelia Luna Manager

Each level of the hierarchy has its own template. The RadTreeView ItemTemplate points at the "PresidentTemplate", the "PresidentTemplate" ItemTemplate points to the "VPTemplate" and so on until the last "ManagerTemplate".

```
<UserControl.Resources>
. . .
  <DataTemplate x:Key="ManagerTemplate">
   . . . content
  </DataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="DirectorTemplate"
      ItemTemplate="{StaticResource ManagerTemplate}">
   . . . content
  </telerik:HierarchicalDataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="VPTemplate"
      ItemTemplate="{StaticResource DirectorTemplate}">
   . . . content
  </telerik:HierarchicalDataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="PresidentTemplate"
      ItemTemplate="{StaticResource VPTemplate}">
   . . . content
  </telerik:HierarchicalDataTemplate>

</UserControl.Resources>
. . .
<telerik:RadTreeView x:Name="tvMain"
    ItemTemplate="{StaticResource PresidentTemplate}">
</telerik:RadTreeView>
```

Walk Through

This walk through will first demonstrate how to display a static hierarchy using the templates as explained above. Then a more maintainable solution will be presented that uses a single HierarchicalDataTemplate.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) Telerik.Windows.Controls

b) Telerik.Windows.Controls.Navigation

c) Telerik.Windows.Themes.Summer

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace references to **System.Windows**, **Telerik.Windows.Controls**, **Telerik.Windows.Controls.Navigation** and to the project itself. This last reference will allow access to our data source that we will write later.

```
<UserControl
xmlns:local="clr-namespace:_03B_Binding_Hierarchy"
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
xmlns:controls="clr-namespace:System.Windows.Controls;assembly=System.Windows">
```

3) Add a UserControl Resources element inside the UserControl element as shown below.

The two comments are marking where you will add resources in the following steps. The first set of resources that control appearance related properties of the page aren't important to this example and can be copied and pasted without examination. The second set of resources, i.e. "<!--data binding resources-->", is the crucial part of this example that makes hierarchical data binding work.

```
<UserControl.Resources>

  <!--resources for the appearance of the page-->


  <!--data binding resources-->

</UserControl.Resources>
```

4) Add the following appearance related XAML under the comment "<!--resources for the appearance of the page-->".

```
<!--resources for the appearance of the page-->

<Style x:Key="BorderStyle" TargetType="Border">
   <Setter Property="VerticalAlignment" Value="Center" />
   <Setter Property="HorizontalAlignment" Value="Center" />
   <Setter Property="BorderThickness" Value="1" />
   <Setter Property="BorderBrush">
      <Setter.Value>
         <LinearGradientBrush>
            <GradientStop Offset="0" Color="#55000055" />
            <GradientStop Offset="1" Color="#99BFD2C2" />
         </LinearGradientBrush>
      </Setter.Value>
   </Setter>
   <Setter Property="Background">
      <Setter.Value>
         <LinearGradientBrush>
            <GradientStop Offset="0" Color="#9FBFD2C2" />
            <GradientStop Offset=".9" Color="#1ABFD2C2" />
            <GradientStop Offset="1" Color="Transparent" />
         </LinearGradientBrush>
      </Setter.Value>
   </Setter>
</Style>

<Style x:Key="TreeStackPanel" TargetType="StackPanel">
   <Setter Property="Orientation" Value="Vertical" />
   <Setter Property="Margin" Value="20" />
</Style>

<Style x:Key="TitleStyle" TargetType="TextBlock">
   <Setter Property="Margin" Value="20" />
   <Setter Property="FontSize" Value="20" />
</Style>
```

5) *Add the following appearance related XAML under the comment "<!--data binding resources-->".*

"local:OrgChart" will point to an object we will write later to contain our data. Each of the HierarchicalDataTemplate tag contains an ItemsSource attribute that specifies a collection property in the OrgChart object and an ItemTemplate attribute that points to the next template in line. The last template in line is a DataTemplate called "ManagerTemplate".

```
<!--data binding resources-->

<local:OrgChart x:Key="OrgChart" />

<DataTemplate x:Key="ManagerTemplate">
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="{Binding FullName}" />
    <TextBlock Foreground="PowderBlue" Text="{Binding Title}" />
  </StackPanel>
</DataTemplate>

<telerik:HierarchicalDataTemplate x:Key="DirectorTemplate"
    ItemsSource="{Binding DirectReports}"
    ItemTemplate="{StaticResource ManagerTemplate}">
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="{Binding FullName}" />
    <TextBlock Foreground="SkyBlue" Text="{Binding Title}" />
  </StackPanel>
</telerik:HierarchicalDataTemplate>

<telerik:HierarchicalDataTemplate x:Key="VPTemplate"
    ItemsSource="{Binding DirectReports}"
    ItemTemplate="{StaticResource DirectorTemplate}">
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="{Binding FullName}" />
    <TextBlock Foreground="SlateBlue" Text="{Binding Title}" />
  </StackPanel>
</telerik:HierarchicalDataTemplate>

<telerik:HierarchicalDataTemplate x:Key="PresidentTemplate"
    ItemsSource="{Binding DirectReports}"
    ItemTemplate="{StaticResource VPTemplate}">
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="{Binding FullName}" />
    <TextBlock Foreground="Blue" Text="{Binding Title}" />
  </StackPanel>
</telerik:HierarchicalDataTemplate>
```

6) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags.

```
<Border Style="{StaticResource BorderStyle}">
  <StackPanel Style="{StaticResource TreeStackPanel}">
    <TextBlock Text="Organization Chart"
        Style="{StaticResource TitleStyle}" />
    <!--tree view-->
  </StackPanel>
</Border>
```

7) Drag a **RadTreeView** from the Toolbox and drop it just below the "<!--tree view-->" comment, under the TextBlock. Set the RadTreeView **ItemsSource** property to "{StaticResource OrgChart}" and the **ItemTemplate** to "{StaticResource PresidentTemplate}". The XAML should look like the example below when you're done.

```
<Border Style="{StaticResource BorderStyle}">
  <StackPanel Style="{StaticResource TreeStackPanel}">
    <TextBlock Text="Organization Chart"
        Style="{StaticResource TitleStyle}" />
    <!--tree view-->
    <telerik:RadTreeView
        ItemsSource="{StaticResource OrgChart}"
        ItemTemplate="{StaticResource PresidentTemplate}">
    </telerik:RadTreeView>
  </StackPanel>
</Border>
```

Coding the Data Source Object

In these next steps we will create an "OrgChart" object that has a collection of "Management" objects. The OrgChart object will form a corporate reporting structure where each Management object will have a "DirectReports" collection of other Management objects.

1) In the Solution Explorer, right-click the project and select **Add > Class...** from the context menu. Name the class file "Management.cs" and click **Add** to create the class file.

2) In the code-behind for the class, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

   a) **System.Collections.ObjectModel** (to support the ObservableCollection class)

   b) **System.ComponentModel** (to support the INotifyPropertyChanged interface)

3) Add a public enumeration "Title" with members "President", "VicePresident", "Director" and "Manager".

VB

```
Public Enum Title
  President
  VicePresident
  Director
  Manager
End Enum
```

C#

```
public enum Title { President, VicePresident, Director, Manager };
```

4) Add a Management class to the class file. It should implement the INotifyPropertyChanged interface. It should have a string "FullName" property, Title property, and DirectReports property. DirectReports will be an ObservableCollection of Management.

```
Public Class Management
  Implements INotifyPropertyChanged
Public Sub New()
  Me._directReports = New ObservableCollection(Of Management)()
End Sub

Private _fullName As String
Public Property FullName() As String
  Get
    Return _fullName
  End Get

  Set(ByVal value As String)
    If _fullName <> value Then
      _fullName = value
      OnPropertyChanged("FullName")
    End If
  End Set
End Property

Private _directReports As ObservableCollection(Of Management)
Public Property DirectReports() As ObservableCollection(Of Management)
  Get
    Return _directReports
  End Get

  Set(ByVal value As ObservableCollection(Of Management))
    If _directReports IsNot value Then
      _directReports = value
      OnPropertyChanged("DirectReports")
    End If
  End Set
End Property

Private _title As Title
Public Property Title() As Title
  Get
    Return _title
  End Get

  Set(ByVal value As Title)
    If _title IsNot value Then
      _title = value
      OnPropertyChanged("Title")
    End If
  End Set
End Property
```

```
Protected Overridable Sub OnPropertyChanged(ByVal propertyName As String)
  RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
End Sub

Public Event PropertyChanged As PropertyChangedEventHandler
End Class
```

```
public class Management : INotifyPropertyChanged
{
public Management()
{
   this._directReports = new ObservableCollection<Management>();
}

private string _fullName;
public string FullName
{
   get
   {
      return _fullName;
   }

   set
   {
      if (_fullName != value)
      {
         _fullName = value;
         OnPropertyChanged("FullName");
      }
   }
}

private ObservableCollection<Management> _directReports;
public ObservableCollection<Management> DirectReports
{
   get
   {
      return _directReports;
   }

   set
   {
      if (_directReports != value)
      {
         _directReports = value;
         OnPropertyChanged("DirectReports");
      }
   }
}

private Title _title;
public Title Title
{
   get
   {
      return _title;
   }

   set
   {
      if (_title != value)
      {
         _title = value;
         OnPropertyChanged("Title");
      }
   }
}

protected virtual void OnPropertyChanged(string propertyName)
{
   if (PropertyChanged != null)
   {
      PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
   }
}

public event PropertyChangedEventHandler PropertyChanged;
}
```

5) In the Solution Explorer, right-click the project and select **Add > Class...** from the context menu. Name the class file "OrgChart.cs" and click **Add** to create the class file.

6) In the code-behind for the class, add references to the "Imports" (VB) or "using" (C#) section of the code for this namespace:

a) **System.Collections.ObjectModel** (to support the ObservableCollection class)

7) Paste the code below to create the OrgChart class.

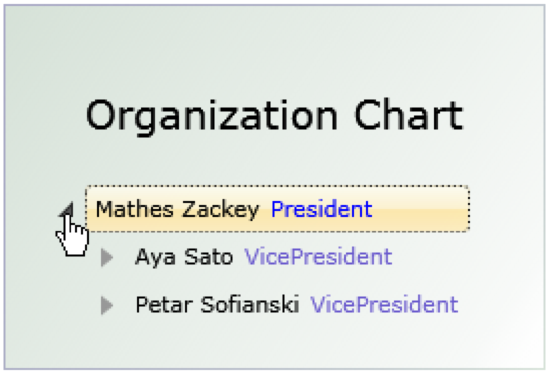
It inherits from ObservableCollection of type Management. The class simply instantiates a number of Management objects and adds them to the collection. notice that the Management objects are nested by way of the DirectReports property.

```
Public Class OrgChart
  Inherits ObservableCollection(Of Management)
  Public Sub New()
    Me.Add(New Management() With { _
.FullName = "Mathes Zackey", _
.Title = Title.President, _
.DirectReports = { New Management() With {_
.FullName = "Aya Sato", _
.Title = Title.VicePresident, _
.DirectReports = { New Management() With {_
.FullName = "Hamlima Massri", _
.Title = Title.Director}, New Management() With {_
.FullName = "Robert Simpson", _
.Title = Title.Director}, New Management() With {_
.FullName = "Mary Sweitzer", _
.Title = Title.Director}, New Management() With {_
.FullName = "Lian Ran", _
.Title = Title.Director} }}, New Management() With {_
.FullName = "Petar Sofianski", _
.Title = Title.VicePresident, _
.DirectReports = { New Management() With {_
.FullName = "Isabella Benson", _
.Title = Title.Director}, New Management() With {_
.FullName = "Aaron Silverman", _
.Title = Title.Director, _
.DirectReports = { New Management() With {_
.FullName = "Sam Schwartz", _
.Title = Title.Manager}, New Management() With {_
.FullName = "Adelia Luna", _
.Title = Title.Manager} }} }} }})
  End Sub
End Class
```

```
public class OrgChart : ObservableCollection<Management>
{
  public OrgChart()
  {
    this.Add(new Management()
    {
      FullName = "Mathes Zackey",
      Title = Title.President,
      DirectReports =
{
    new Management()
    {
        FullName = "Aya Sato",
        Title = Title.VicePresident,
        DirectReports =
        {
            new Management()
            {
                FullName = "Hamlima Massri",
                Title = Title.Director,
            },
            new Management()
            {
                FullName = "Robert Simpson",
                Title = Title.Director,
            },
            new Management()
            {
                FullName = "Mary Sweitzer",
                Title = Title.Director,
            },
            new Management()
            {
                FullName = "Lian Ran",
                Title = Title.Director,
            }
        }
    },
    new Management()
    {
        FullName = "Petar Sofianski",
        Title = Title.VicePresident,
        DirectReports =
        {
            new Management()
            {
                FullName = "Isabella Benson",
                Title = Title.Director,
            },
            new Management()
            {
                FullName = "Aaron Silverman",
                Title = Title.Director,
                DirectReports =
                {
                    new Management()
                    {
                        FullName = "Sam Schwartz",
                        Title = Title.Manager,
                    },
                    new Management()
                    {
                        FullName = "Adelia Luna",
                        Title = Title.Manager
                        }
                    }
                }

            }
        }
    }
});
}
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Organization Chart

Mathes Zackey President

Aya Sato VicePresident

Petar Sofianski VicePresident

The example so far shows how you can customize each level if you have a static hierarchy. The problem with this approach is that its inflexible and not easy to maintain. The structure of each level in the hierarchy is identical, so do I need a template for each level? No, a single template will do, as shown below. This solution will handle additional levels automatically, as long as they follow the same structure where the ItemsSource refers to the collection for the next level down. Change the example to include only a single HierarchicalDataTemplate as shown below. Notice that the ItemTemplate is not defined because we don't want to refer to another template. Also notice the "TitleToBrushConverter" that we will implement in a later step.

```
<UserControl.Resources>

  . . .
  <local:OrgChart x:Key="OrgChart" />
  <local:TitleToBrushConverter x:Name="TitleToBrushConverter" />

  <telerik:HierarchicalDataTemplate
      x:Key="DirectReportsTemplate"
      ItemsSource="{Binding DirectReports}">
    <StackPanel Orientation="Horizontal">
      <TextBlock Text="{Binding FullName}" />
      <TextBlock Text="{Binding Title}"
         Foreground="{Binding Title, Converter={StaticResource TitleToBrushConverter}}"
          Margin="5, 0, 0, 0" />
    </StackPanel>
  </telerik:HierarchicalDataTemplate>

</UserControl.Resources>

<Grid x:Name="LayoutRoot">
  <Border Style="{StaticResource BorderStyle}">
    <StackPanel Style="{StaticResource TreeStackPanel}">
      <TextBlock Text="Organization Chart"
          Style="{StaticResource TitleStyle}" />
      <!--tree view-->
      <telerik:RadTreeView
          ItemsSource="{StaticResource OrgChart}"
          ItemTemplate="{StaticResource DirectReportsTemplate}">
      </telerik:RadTreeView>
    </StackPanel>
  </Border>
</Grid>
</UserControl>
```

The TitleToBrushConverter accepts the bound object's Title enumeration property and returns a brush. The brush is applied to the text Foreground property. Here's the implementation of the new TitleToBrushConverter class.

```
Public Class TitleToBrushConverter
 Implements IValueConverter

 #Region "IValueConverter Members"

 Public Function Convert(ByVal value As Object, ByVal targetType As Type,
ByVal parameter As Object, ByVal culture As System.Globalization.CultureInfo) As Object
  Select Case CType(value, Title)
   Case Title.President
    Return New SolidColorBrush(Colors.Black)
   Case Title.VicePresident
    Return New SolidColorBrush(Colors.DarkGray)
   Case Title.Director
    Return New SolidColorBrush(Colors.Blue)
   Case Title.Manager
    Return New SolidColorBrush(Colors.LightGray)
   Case Else
    Return New SolidColorBrush(Colors.White)
  End Select
 End Function

 Public Function ConvertBack(ByVal value As Object, ByVal targetType As Type, _
ByVal parameter As Object, ByVal culture As System.Globalization.CultureInfo) As Object
  Throw New NotImplementedException()
 End Function

 #End Region
End Class
```

C#

```
public class TitleToBrushConverter : IValueConverter
{

   public object Convert(object value, Type targetType, object parameter,
    System.Globalization.CultureInfo culture)
   {
      switch ((Title)value)
      {
         case Title.President: return new SolidColorBrush(Colors.Black);
         case Title.VicePresident: return new SolidColorBrush(Colors.DarkGray);
         case Title.Director: return new SolidColorBrush(Colors.Blue);
         case Title.Manager: return new SolidColorBrush(Colors.LightGray);
         default: return new SolidColorBrush(Colors.White);
      }
   }

   public object ConvertBack(object value, Type targetType, object parameter,
    System.Globalization.CultureInfo culture)
   {
      throw new NotImplementedException();
   }

}
```

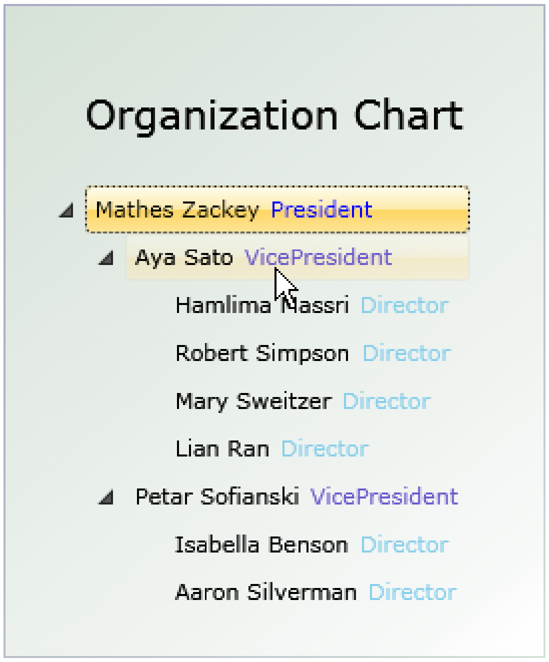
Test Application Features

1) Verify that all four templates worth of data display.

Ideas for Extending This Example

- Include additional fields in the Management object and surface them in the template.

- Change the "DirectorTemplate" from a HierarchicalDataTemplate to a DataTemplate. Remove the ItemsSource and ItemTemplate attributes from the DataTemplate. You should now see only the first three levels, even though there is more data.

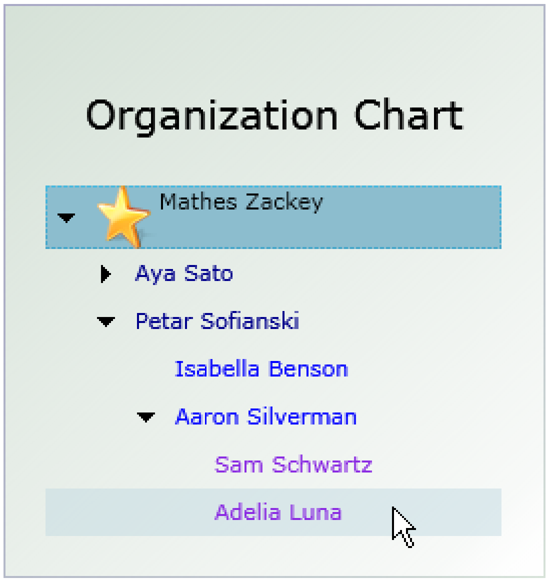


Organization Chart

Mathes Zackey President
Aya Sato VicePresident
Hamlima Nassri Director
Robert Simpson Director
Mary Sweitzer Director
Lian Ran Director
Petar Sofianski VicePresident
Isabella Benson Director
Aaron Silverman Director

17.5.3 Template Selectors

Template selectors let you choose between multiple templates at runtime. The screenshot below shows a template for the president that has an image and where the other templates also have slight differences in foreground brush colors.



Notes

Templates are selected when an item is first prepared but cannot be changed once shown.

The key piece to making template selectors work is first to inherit from the **DataTemplateSelector** class and override its **SelectTemplate()** method. Use the method's "item" parameter to reference the business object bound to the template. In this case the business object is a Management object with Name and Title properties.The example uses the Title property to determine which template should be returned. "MyTemplateSelector" defined below also has properties to store each of the types of templates it might select (these properties are set later in the XAML).

```
Public Class MyTemplateSelector
  Inherits DataTemplateSelector
  ' save off references to the page and templates on the page here
  Private privatePresidentTemplate As HierarchicalDataTemplate
  Public Property PresidentTemplate() As HierarchicalDataTemplate
    Get
      Return privatePresidentTemplate
    End Get
    Set(ByVal value As HierarchicalDataTemplate)
      privatePresidentTemplate = value
    End Set
  End Property
  Private privateVPTemplate As HierarchicalDataTemplate
  Public Property VPTemplate() As HierarchicalDataTemplate
    Get
      Return privateVPTemplate
    End Get
    Set(ByVal value As HierarchicalDataTemplate)
      privateVPTemplate = value
    End Set
  End Property
  Private privateDirectorTemplate As HierarchicalDataTemplate
  Public Property DirectorTemplate() As HierarchicalDataTemplate
    Get
      Return privateDirectorTemplate
    End Get
    Set(ByVal value As HierarchicalDataTemplate)
      privateDirectorTemplate = value
    End Set
  End Property
  Private privateManagerTemplate As DataTemplate
  Public Property ManagerTemplate() As DataTemplate
    Get
      Return privateManagerTemplate
    End Get
    Set(ByVal value As DataTemplate)
      privateManagerTemplate = value
    End Set
  End Property

  Public Overrides Function SelectTemplate(ByVal item As Object, ByVal container As DependencyObject) As
    MyBase.SelectTemplate(item, container)

    ' get the bound object
    Dim management As Management = TryCast(item, Management)

    ' use the object data to choose a template
    Select Case management.Title
      Case Title.President
        Return PresidentTemplate
      Case Title.VicePresident
        Return VPTemplate
      Case Title.Director
        Return DirectorTemplate
      Case Title.Manager
        Return ManagerTemplate
      Case Else
        Return Nothing
    End Select
```

```
public class MyTemplateSelector : DataTemplateSelector
{
  // save off references to the page and templates on the page here
  public HierarchicalDataTemplate PresidentTemplate
  {
    get;
    set;
  }
  public HierarchicalDataTemplate VPTemplate
  {
    get;
    set;
  }
  public HierarchicalDataTemplate DirectorTemplate
  {
    get;
    set;
  }
  public DataTemplate ManagerTemplate
  {
    get;
    set;
  }

  public override DataTemplate SelectTemplate(
    object item, DependencyObject container)
  {
    base.SelectTemplate(item, container);

    // get the bound object
    Management management = item as Management;

    // use the object data to choose a template
    switch (management.Title)
    {
      case Title.President: return PresidentTemplate;
      case Title.VicePresident: return VPTemplate;
      case Title.Director: return DirectorTemplate;
      case Title.Manager: return ManagerTemplate;
      default: return null;
    }
  }
}
```

Notice the XAML below:

- Appearance related styles are omitted for brevity.
- There are no ItemTemplate references as in the Hierarchical Template example because the template selector takes care of identifying the template to use.
- Each template can contain any content you care to fill it with.

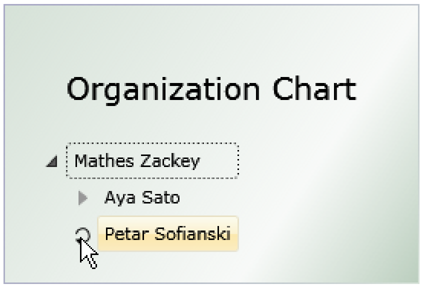
- The template selector "MyTemplateSelector" defines the template properties, "PresidentTemplate", etc, that we defined in the MyTemplateSelector code behind. Each template property is hooked up to its respective template resource.

```
<UserControl.Resources>. . .

  <local:OrgChart x:Key="OrgChartDataSource" />

  <telerik:HierarchicalDataTemplate x:Key="Manager">
    <TextBlock Text="{Binding FullName}" Foreground="{StaticResource ManagerBrush}" />
  </telerik:HierarchicalDataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="Director" ItemsSource="{Binding DirectReports}">
    <TextBlock Text="{Binding FullName}" Foreground="{StaticResource DirectorBrush}" />
  </telerik:HierarchicalDataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="VP" ItemsSource="{Binding DirectReports}">
    <TextBlock Text="{Binding FullName}" Foreground="{StaticResource VPBrush}" />
  </telerik:HierarchicalDataTemplate>

  <telerik:HierarchicalDataTemplate x:Key="President" ItemsSource="{Binding DirectReports}">
    <StackPanel Orientation="Horizontal">
      <Image Source="images/Favorites.png" />
      <TextBlock Text="{Binding FullName}" />
    </StackPanel>
  </telerik:HierarchicalDataTemplate>

  <local:MyTemplateSelector x:Key="MyTemplateSelector"
      PresidentTemplate="{StaticResource President}"
      VPTemplate="{StaticResource VP}"
      DirectorTemplate="{StaticResource Director}"
      ManagerTemplate="{StaticResource Manager}" />
</UserControl.Resources>
```

In the XAML, RadTreeView **ItemTemplateSelector** is assigned the "MyTemplateSelector" reference.

```
<telerik:RadTreeView
    ItemsSource="{StaticResource OrgChartDataSource}"
    ItemTemplateSelector="{StaticResource MyTemplateSelector}">
</telerik:RadTreeView>
```

17.5.4 Load-On-Demand



The end-user doesn't need to see every node in the tree. Nodes only need to be retrieved when they are expanded, i.e. "on demand". In the running application, a "spinny" graphic automatically displays to indicate that data is being retrieved for the expanding node.

You can mix-and-match so that some nodes are pre-loaded and expanded, while less used nodes are loaded when the user clicks the expansion button. The general steps for making load-on-demand work are the same without regard to the actual source of the data, be it from a web service, from a network source or from some other origin. The key pieces for getting load-on-demand to work are:

- Set the RadTreeView **IsLoadOnDemandEnabled** property to "True".
- Handle the **LoadOnDemand** event. In the event, get a reference to the node being expanded and assign its **ItemsSource** property. The ItemsSource property for the node supplies just the data for the node's children and no more.

```
Private Sub tvMain_LoadOnDemand( _
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  ' get a reference to the node being expanded
  Dim treeViewItem As RadTreeViewItem = TryCast(e.OriginalSource, RadTreeViewItem)
  ' you can also get a reference to a bound object
  Dim myObject As MyObject = TryCast(treeViewItem.Item, MyObject)
  treeViewItem.ItemsSource = SomeDataSource
End Sub
```

```
void tvMain_LoadOnDemand(object sender, Telerik.Windows.RadRoutedEventArgs e)
{
   // get a reference to the node being expanded
   RadTreeViewItem treeViewItem = e.OriginalSource as RadTreeViewItem;
   // you can also get a reference to a bound object
   MyObject myObject = treeViewItem.Item as MyObject;
   treeViewItem.ItemsSource = SomeDataSource;
}
```

The code below shows a modified "OrgChart" example that demonstrates the load-on-demand feature. The "DirectReports" property is assigned to each node ItemsSource property as needed. Here are some things to look for in the code:

- In the Loaded event, the **IsLoadOnDemandEnabled** property is set to "True" and the **LoadOnDemand** event is assigned an event handler. Both of these could be defined directly in the XAML. The root level object in OrgChart is assigned as the root of the tree.
- The LoadOnDemand event handler has two major aspects. Part of the code simulates an asynchronous operation with some latency and is included here just to get a feel for how the tree view behaves when loading from a remote source over the network. The significant part of the code, that you should pay attention to, gets a reference to the expanded node and its bound object, then assigns the node's ItemsSource property if there's any data. Notice that if there is no data to be assigned, the node's IsLoadOnDemandEnabled property is set "False" to remove the expansion button.

```
Private orgChart As New OrgChart()

Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' turn on the load-on-demand feature
  tvMain.IsLoadOnDemandEnabled = True
  ' handle the load on demand event
  AddHandler tvMain.LoadOnDemand, AddressOf tvMain_LoadOnDemand
  ' assign the initial node that displays at startup
  tvMain.ItemsSource = New ObservableCollection(Of Management) (New Management() {orgChart(0)})
  ' tweak load on demand related node properties based on its data
  AddHandler tvMain.ItemPrepared, AddressOf tvMain_ItemPrepared
End Sub

Private Sub tvMain_LoadOnDemand( _
ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  ' simulate asynchronous operation
    ' simulate lag time
      ' get the node being expanded
      ' get the node's bound object
      ' load the node if there's any data
  CType(New System.Threading.Thread(Function() With { _
System.Threading.Thread.Sleep(1000); Dispatcher.BeginInvoke(Function() { _
RadTreeViewItem treeViewItem = TryCast(e.OriginalSource, RadTreeViewItem); _
Management management = TryCast(treeViewItem.Item, Management); _
if (management.HasDirectReports) { _
treeViewItem.ItemsSource = management.DirectReports; } else { _
treeViewItem.IsLoadOnDemandEnabled = False; } }); }), _
System.Threading.Thread).Start()
End Sub
```

```
private OrgChart orgChart = new OrgChart();

private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    // turn on the load-on-demand feature
    tvMain.IsLoadOnDemandEnabled = true;
    // handle the load on demand event
    tvMain.LoadOnDemand +=
        new EventHandler<Telerik.Windows.RadRoutedEventArgs>(tvMain_LoadOnDemand);
    // assign the initial node that displays at startup
    tvMain.ItemsSource = new ObservableCollection<Management>() { orgChart[0] };
}

void tvMain_LoadOnDemand(object sender, Telerik.Windows.RadRoutedEventArgs e)
{
    // simulate asynchronous operation
    new System.Threading.Thread(() =>
    {
        // simulate lag time
        System.Threading.Thread.Sleep(1000);
        Dispatcher.BeginInvoke(() =>
        {
            // get the node being expanded
            RadTreeViewItem treeViewItem = e.OriginalSource as RadTreeViewItem;
            // get the node's bound object
            Management management = treeViewItem.Item as Management;
            // load the node if there's any data
            if (management.HasDirectReports)
            {
                treeViewItem.ItemsSource = management.DirectReports;
            }
            else
            {
                treeViewItem.IsLoadOnDemandEnabled = false;
            }
        });
    }).Start();
}
```

From the Forums...

Question: "If I know that a node has no children, I don't want to display the expansion button to begin with. Is there a way to hide the expansion button when the node is first displayed?"

Answer: Handle the ItemPrepared event and set the IsLoadOnDemandEnabled to "False" when there are no children available.

```vb
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  '. . .
  'tweak load on demand related node properties based on its data
  AddHandler tvMain.ItemPrepared, AddressOf tvMain_ItemPrepared
End Sub

Private Sub tvMain_ItemPrepared(ByVal sender As Object, _
ByVal e As RadTreeViewItemPreparedEventArgs)
  If TypeOf e.PreparedItem.Item Is Management Then
    'Hide the expand icon if the node has no children
    If Not(TryCast(e.PreparedItem.Item, Management)).HasDirectReports Then
      e.PreparedItem.IsLoadOnDemandEnabled = False
    End If
  End If
End Sub
```

```csharp
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
  //. . .
  //tweak load on demand related node properties based on its data
  tvMain.ItemPrepared +=
    new EventHandler<RadTreeViewItemPreparedEventArgs>(tvMain_ItemPrepared);
}

void tvMain_ItemPrepared(object sender, RadTreeViewItemPreparedEventArgs e)
{
  if (e.PreparedItem.Item is Management)
  {
    //Hide the expand icon if the node has no children
    if (!(e.PreparedItem.Item as Management).HasDirectReports)
    {
      e.PreparedItem.IsLoadOnDemandEnabled = false;
    }
  }
}
```

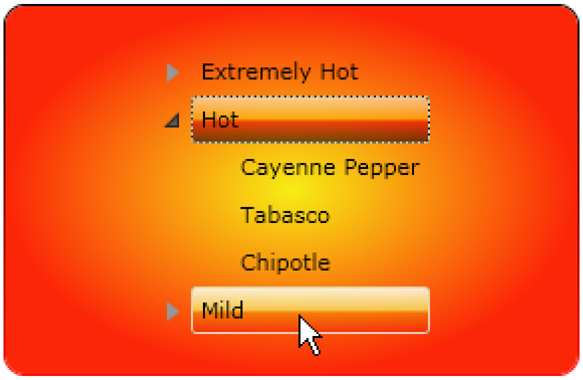
17.6 Customization

The tree view is a complex control with many templated elements that let you control the appearance of the expander button, drag-and-drop elements, header and individual nodes, just to name a few.

To change the appearance of nodes in the tree view, edit the RadTreeViewItem style and assign this new style to the tree items. Here are some of the basic approaches:

- If you code (i.e. create) tree items in XAML than you should modify the **Style** property.
- If you bind the tree's ItemsSource property you should use the tree view **ItemContainerStyle** property
- If you have different items or need to dynamically change styles at runtime than you should use the **ItemContainerStyleSelector** property.

The easiest way to get started is to edit an existing Style. Using Expression Blend, for example, you can add a RadTreeView to a page with at least one RadTreeViewItem. Select the tree item and choose to Edit Template from the context menu. This action will generate a new item style and can be modified to suit your requirements. This will allow you to change various elements of the item such as MouseOverVisual and SelectionUnfocusedVisual. The screenshot below shows the brushes used by the MouseOverVisual and SelectionUnfocusedVisual changed to fit the "Scoville" set of red, yellow, orange and black colors. The RadTreeView itself for the most part is transparent. In the example below, a Border was placed around the RadTreeView and the background was filled with a RadialGradientBrush providing the "sunburst" effect you see below.



"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Walk Through

In this example we will customize the RadTreeViewItems of a RadTreeView.

Project Setup

1) Run Expression Blend.
2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*
3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight Application" from the right-most list. Enter a unique name for the project and click **OK**.

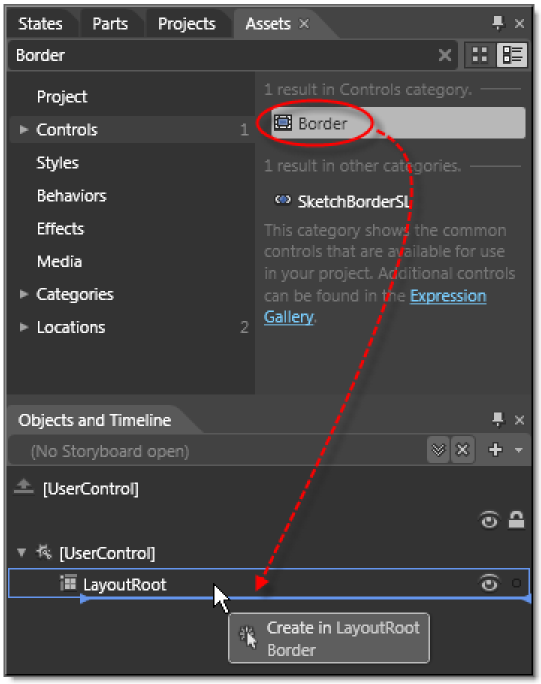
Edit the Page in Expression Blend

1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.
2) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu. Add references to the **Telerik.Windows.Controls.dll** and **Telerik.Windows.Controls.Navigation.dll** assemblies.
3) From the Project menu select **Build Project**.
4) Add a Border control to the page:

a) Open the Assets pane. On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.

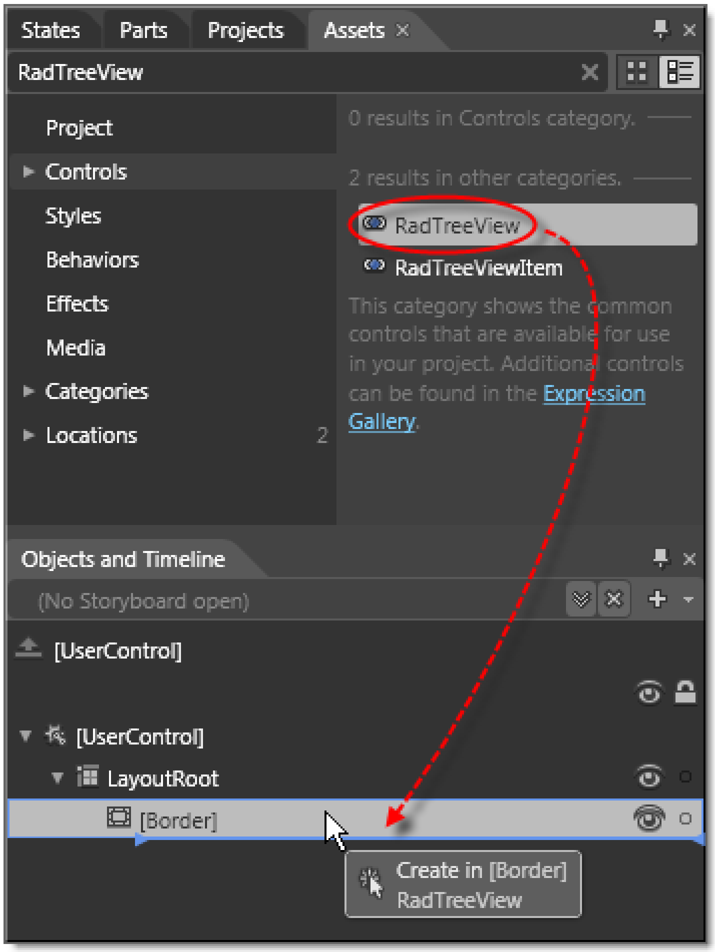
b) In the Assets pane, just above the tree view is the Assets Find entry text box. Type the first few characters of "Border" into the Assets Find entry text box. A list of matching controls will show to the right of the tree view.

c) Locate the Border control and drag it onto the "LayoutRoot" node. Notice how the tool tip reads "Create in LayoutRoot".

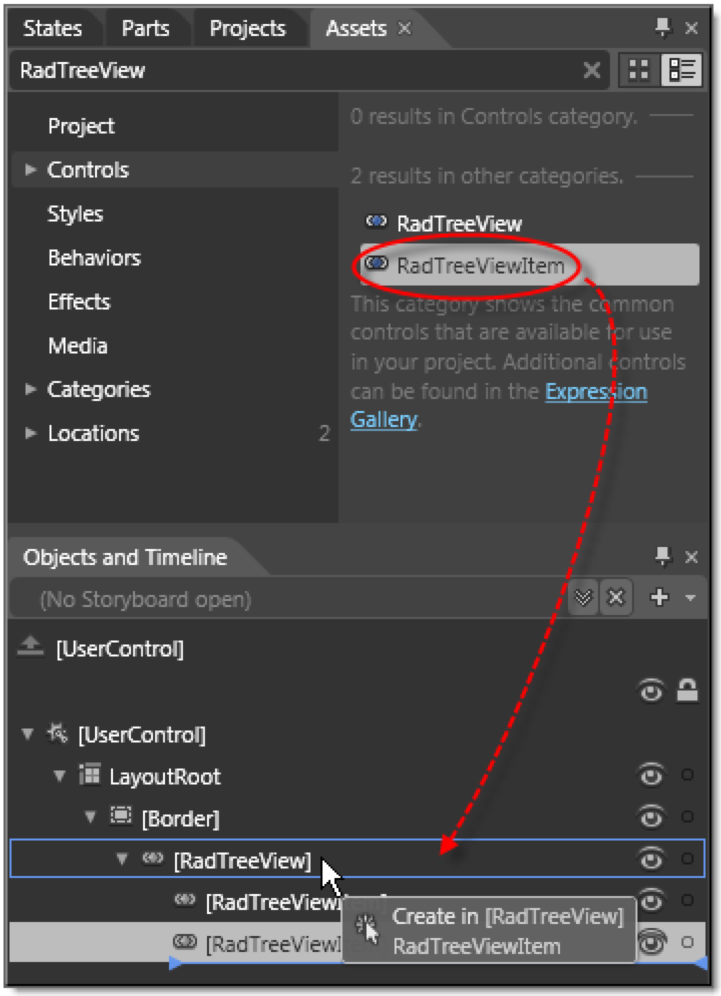


This category shows the common controls that are available for use in your project. Additional controls can be found in the Expression Gallery.

5) Add a **RadTreeView** from the Assets pane into the Border. Again, watch the tool tip to see that the tree view is created "in" the Border (i.e., not as a sibling of the Border).

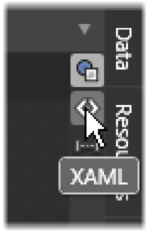


6) Drag three **RadTreeViewItem** from the Assets pane into the RadTreeView. Select each RadTreeViewItem in the Objects and Timeline pane and set the **Header** properties in the Properties pane to "Extremely Hot", "Hot" and "Mild".



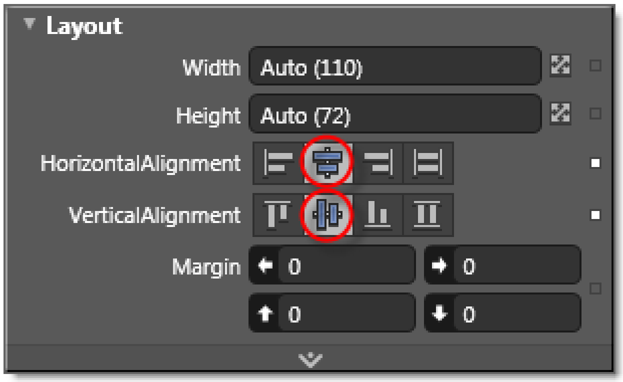
Notes

Alternatively, you can switch to the XAML view using the buttons on the upper right hand side of the Artboard and paste in the same node structure used in the Getting Started chapter. The XAML is included below for convenience.



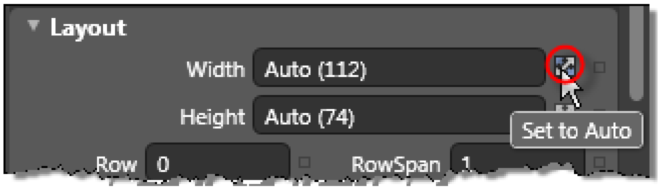
```
<telerik:RadTreeView>

  <telerik:RadTreeViewItem Header="Extremely Hot">
    <telerik:RadTreeViewItem Header="Pure capsaicin" />
    <telerik:RadTreeViewItem Header="Naga Jolokia" />
    <telerik:RadTreeViewItem Header="Red Savina Habanero" />
  </telerik:RadTreeViewItem>

  <telerik:RadTreeViewItem Header="Hot">
    <telerik:RadTreeViewItem Header="Cayenne Pepper" />
    <telerik:RadTreeViewItem Header="Tabasco" />
    <telerik:RadTreeViewItem Header="Chipotle" />
  </telerik:RadTreeViewItem>

  <telerik:RadTreeViewItem Header="Mild">
    <telerik:RadTreeViewItem Header="Bell Pepper" />
    <telerik:RadTreeViewItem Header="Pimento" />
    <telerik:RadTreeViewItem Header="Peperoncini" />
  </telerik:RadTreeViewItem>

</telerik:RadTreeView>
```

7) In the Objects and Timeline pane, select the RadTreeView to edit its properties. In the **Properties pane > Layout** panel, set the HorizontalAlignment and VerticalAlignment properties to center. This will place the tree view in the dead center of the "sunburst" inside the border.

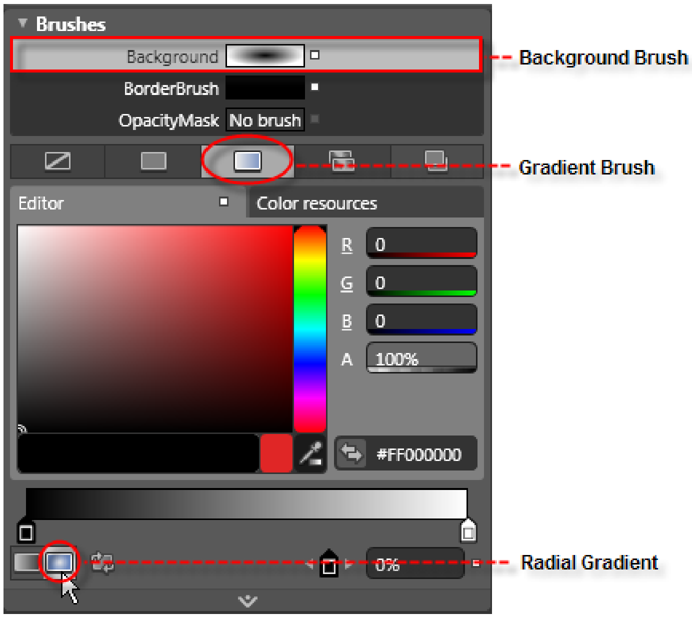

Layout
Width Auto (110)
Height Auto (72)
HorizontalAlignment
VerticalAlignment
Margin 0 0 0 0

8) In the Objects and Timeline pane, select the Border and edit its properties.

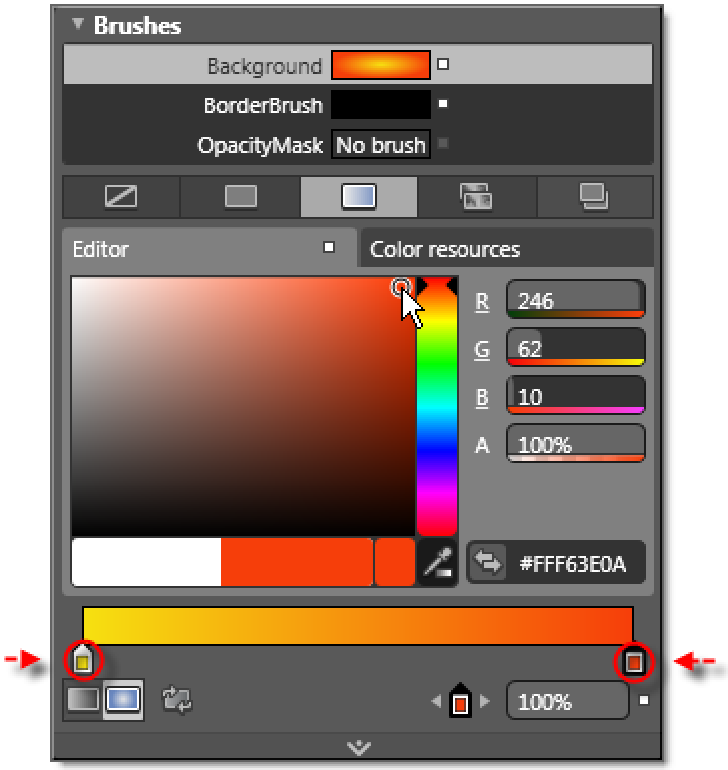
a) In the Layout panel, locate the Advanced Property Options button next to the Width and Height properties. Click the "Set to Auto" button.


Layout
Width Auto (112)
Height Auto (74)
Row 0
RowSpan 1

b) In the Properties pane, select "Background" from the Brushes panel. Click the Gradient Brush button. Click the Radial Gradient button.

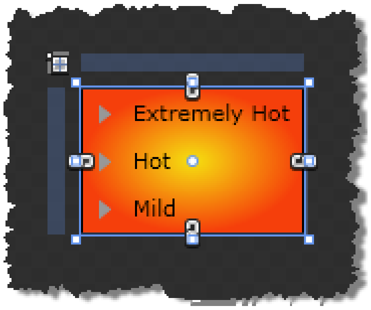

Brushes
Background
BorderBrush
OpacityMask No brush
Editor
Color resources
R 0
G 0
B 0
A 100%
#FF000000
0%

9) In the Gradient Bar, select the left-most gradient stop indicator, then choose a yellow color from the palette. Select the right-most gradient stop indicator and choose a red/orange color from the palette.



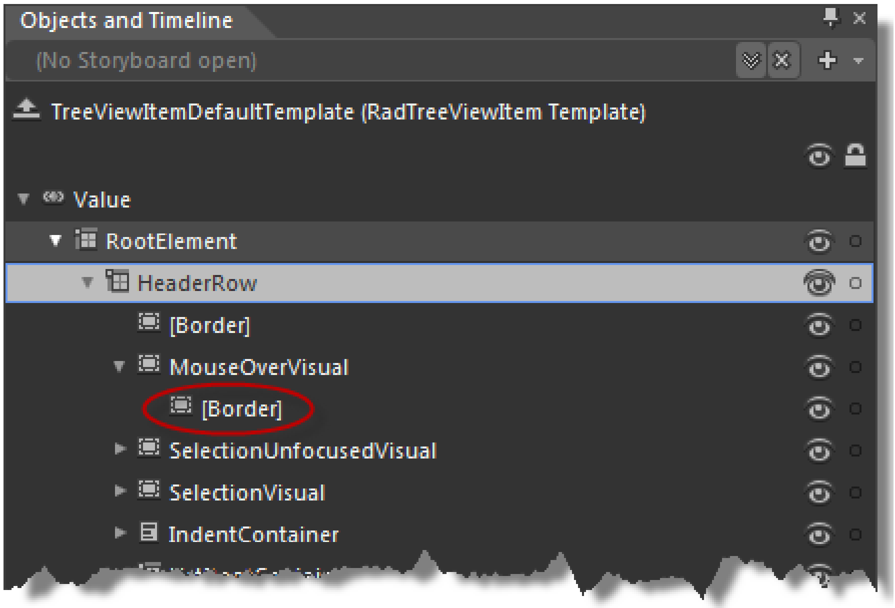
R 246
G 62
B 10
A 100%

Now the tree view in the Artboard should look something like the screenshot below:



10)Right-click the first **RadTreeViewItem** in the tree view and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleItem". Click **OK** to create the style resource and close the dialog.

11)In the Objects and Timeline pane, open up the "HeaderRow" and select the "MouseOverVisual" node. Expand it and select the inner "Border" element.

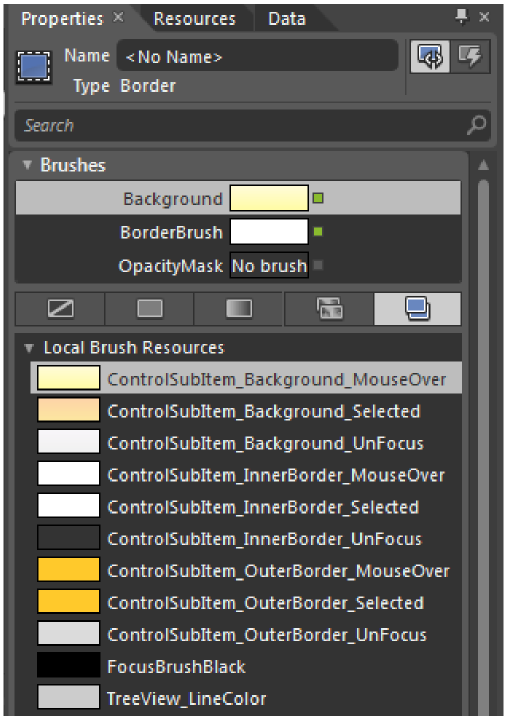


TreeViewItemDefaultTemplate (RadTreeViewItem Template)

- Value
  - RootElement
    - HeaderRow
      - [Border]
      - MouseOverVisual
        - [Border]
      - SelectionUnfocusedVisual
      - SelectionVisual
      - IndentContainer

12)Change the MouseOverVisual brush.

a) In the Properties pane, the Background property in the Brushes pane should already be selected. The "ControlSubItem_Background_MouseOver" local brush resource should also be selected. Before making changes, the Properties pane should look something like the screenshot:

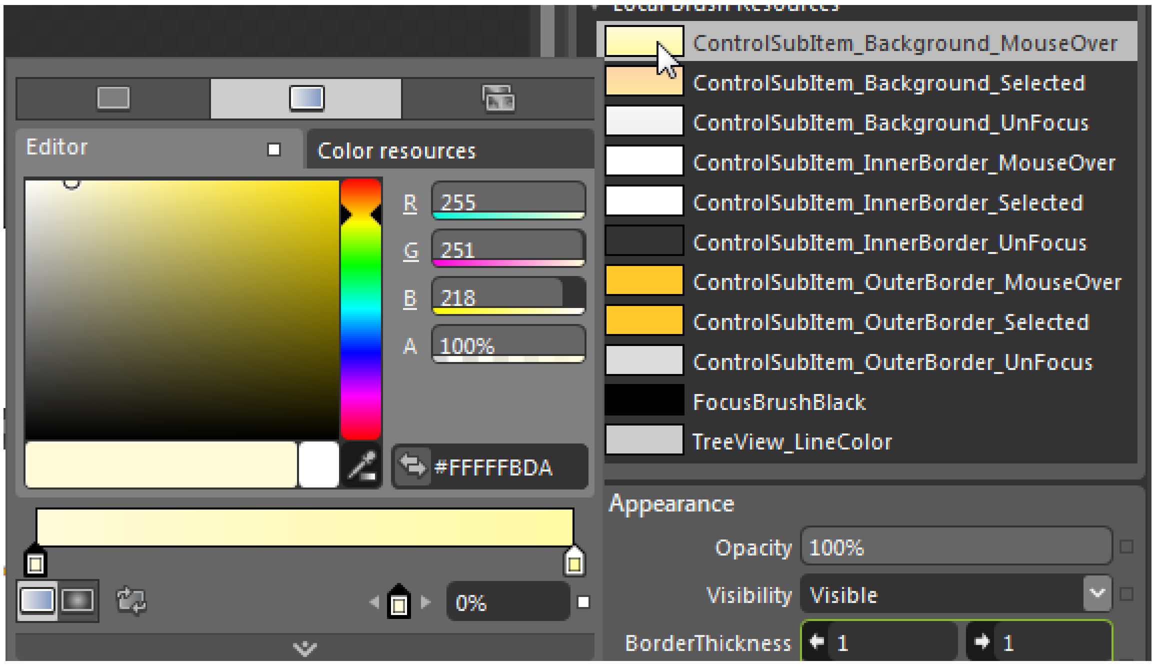


Name <No Name>
Type Border

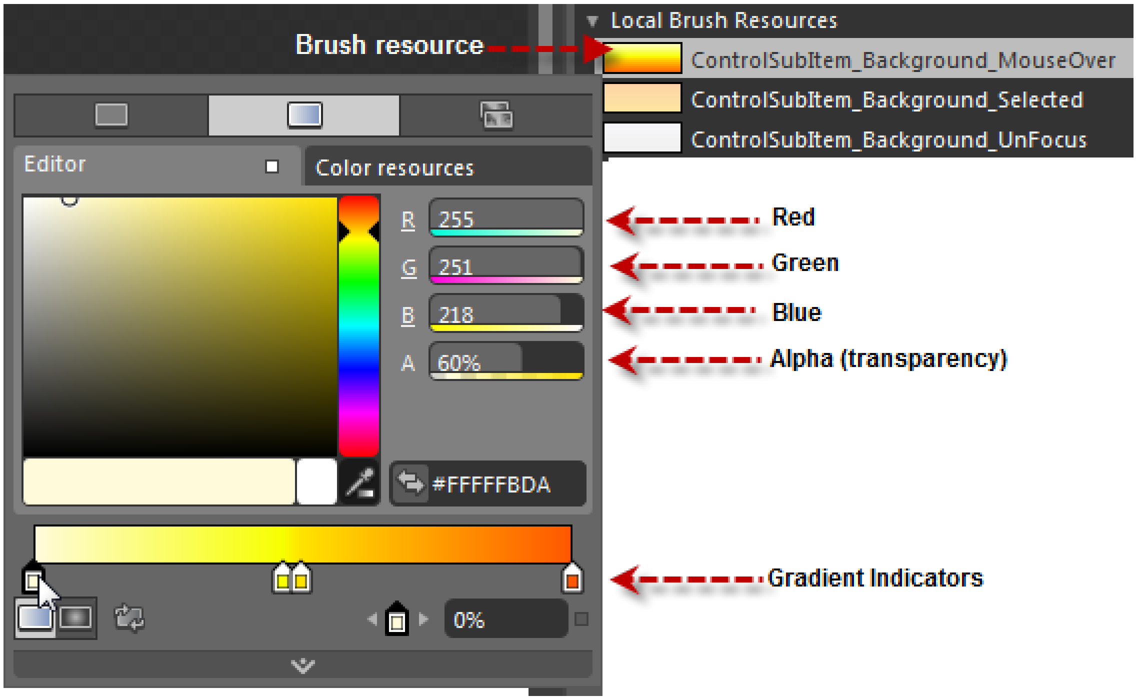
Background
BorderBrush
OpacityMask No brush

ControlSubItem_Background_MouseOver
ControlSubItem_Background_Selected
ControlSubItem_Background_UnFocus
ControlSubItem_InnerBorder_MouseOver
ControlSubItem_InnerBorder_Selected
ControlSubItem_InnerBorder_UnFocus
ControlSubItem_OuterBorder_MouseOver
ControlSubItem_OuterBorder_Selected
ControlSubItem_OuterBorder_UnFocus
FocusBrushBlack
TreeView_LineColor

b) Click the color block to the left of the "ControlSubItem_Background_MouseOver" to display the color editing drop down.


Local Brush Resources
ControlSubItem_Background_MouseOver
ControlSubItem_Background_Selected
ControlSubItem_Background_UnFocus
ControlSubItem_InnerBorder_MouseOver
ControlSubItem_InnerBorder_Selected
ControlSubItem_InnerBorder_UnFocus
ControlSubItem_OuterBorder_MouseOver
ControlSubItem_OuterBorder_Selected
ControlSubItem_OuterBorder_UnFocus
FocusBrushBlack
TreeView_LineColor
Editor
Color resources
R 255
G 251
B 218
A 100%
#FFFFFBDA
0%
Appearance
Opacity 100%
Visibility Visible
BorderThickness 1 1

c) Select the gradient stop indicators and select yellow, orange and red from the palette. The exact colors aren't critical, but could look something like the screenshot below.

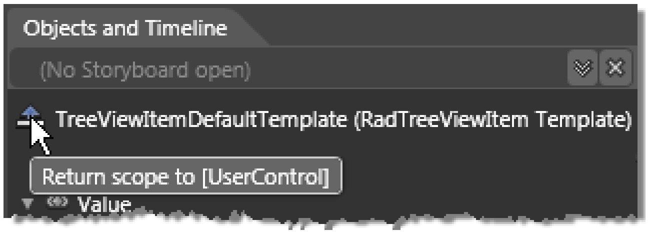


Tip!

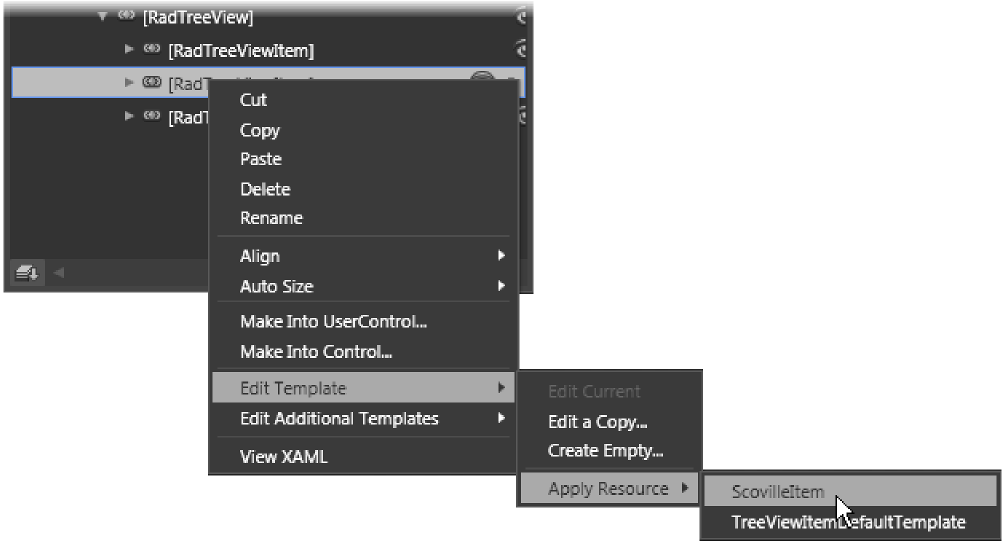
Also notice that you can set the "A" (Alpha or transparency) percentage to something less than 100%. This allows you to use stronger, darker colors without overpowering the content of the item.

13)Change the SelectionVisual brush using the same techniques you used to modify the MouseOverVisual. The brush in this case will be "ControlSubItem_Background_Selected" brush. Change the colors for this brush to use the same palette of red, orange and yellow, but make the color selection somewhat different than the MouseOverVisual.

14)In the Objects and Timeline pane, click the Return Scope button to return to the UserControl scope.

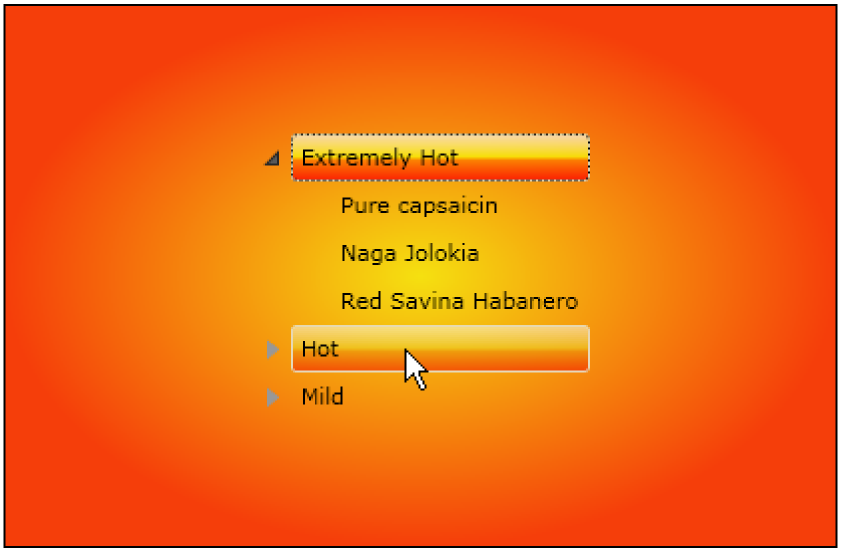

Objects and Timeline
(No Storyboard open)
TreeViewItemDefaultTemplate (RadTreeViewItem Template)
Return scope to [UserControl]
Value

15)Up to this point we have styled the first RadTreeViewItem in the tree. To apply this style to other items, right-click each RadTreeViewItem in the Objects and Timeline pane and select **Edit Template > Apply Resource > "ScovilleItem"** from the context menu.



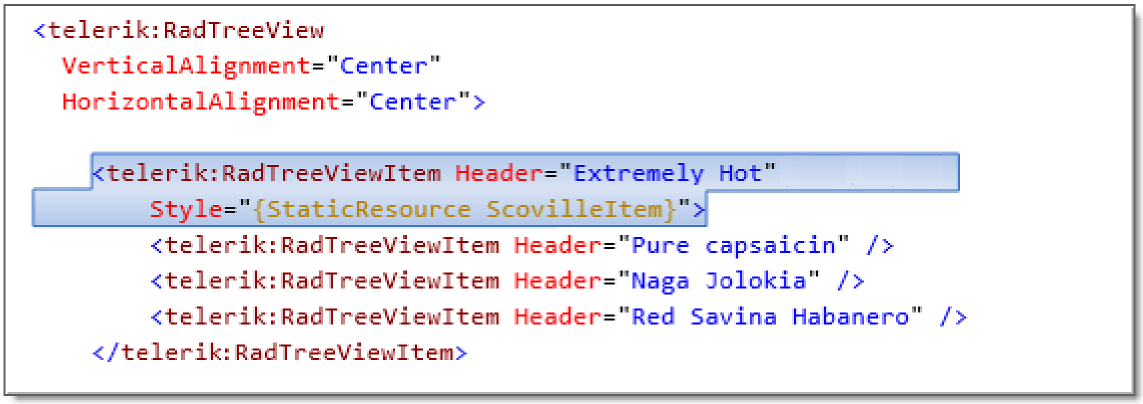
Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

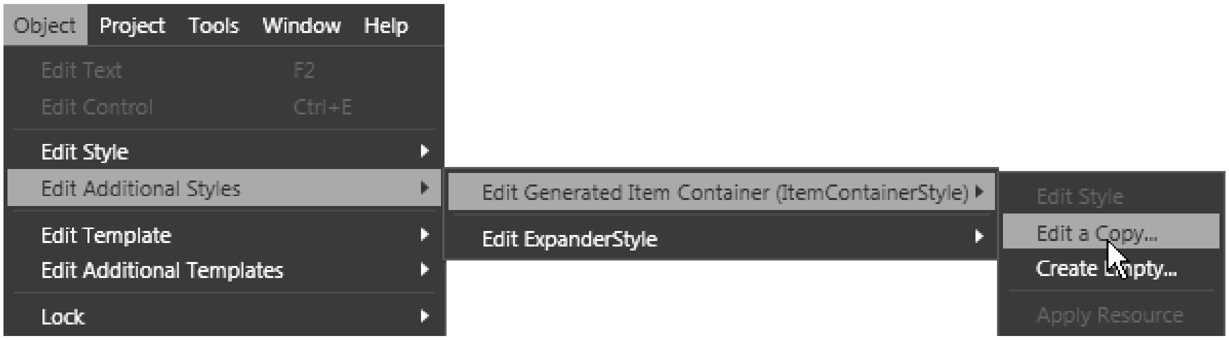
- The "sunburst" effect of the border background should take up the entire browser client area.
- Selected and hovered items should match the MouseOverVisual and SelectionVisual modified brushes.
- Look at the XAML and notice the "Style" attribute points to our "ScovilleItem" style. In this example, the style was only applied to the root items.



```
<telerik:RadTreeView
  VerticalAlignment="Center"
  HorizontalAlignment="Center">

    <telerik:RadTreeViewItem Header="Extremely Hot"
        Style="{StaticResource ScovilleItem}">
        <telerik:RadTreeViewItem Header="Pure capsaicin" />
        <telerik:RadTreeViewItem Header="Naga Jolokia" />
        <telerik:RadTreeViewItem Header="Red Savina Habanero" />
    </telerik:RadTreeViewItem>
```

Ideas for Extending This Example

- Bind the tree view and modify the **ItemContainerStyle** property instead of the Style property. To find the correct options in Expression Blend during design time, you need to have the tree view already bound to data. When you load a project with bound data into Expression Blend, you can select the tree view in the Objects and Timeline pane. Then, in the Expression Blend Object menu, you will be able to find the new menu item **Edit Additional Styles > ItemContainerStyle** option



17.7 Wrap Up

In this chapter we covered a wide range of tasks that exercise many of the RadTreeView features including defining trees manually, using the API to add/remove/enable/select nodes, locating and accessing nodes, adding images to nodes, handling node expansion and reacting to node selection. You also defined trees with mixed groups of radio buttons and checkboxes. You learned how to work with drag-and-drop operations, both to enable simple drag-and-drop functionality and to fine-tune the behavior based on multiple conditions such as source and target nodes and the state of the data associated with a node.

You bound the tree to simple lists of data, then bound specific nodes to data sources. Then you used Hierarchical Templates to organize the data and present a specific appearance based on the level of data. You learned how to use Template Selectors to choose templates where decisions need to be made on-the-fly at runtime. You used the load-on-demand feature to allow performant data loading restricted to nodes being expanded at any one time.

Finally, you learned how to customize the appearance of individual nodes using Expression Blend.

Part XVII

GridView

18 GridView

18.1 Objectives

In this chapter you will be introduced to RadGridView and many of its key features. Starting out you will bind the grid view to some basic data. Afterwards you will see how to expand the example to use hierarchical data and how to customize columns.

While delving into the details of RadGridView, you will work with selected rows and cells. You will handle events that notify you in case a change or selection has been made within the grid. You will learn the programming model commonalities for filtering, sorting and grouping. Furthermore, on working with groups, you will learn how to add aggregate functions for each one of them. You will be introduced to column types and learn about special columns that handle images, hyperlinks and lookups. The "Grid View Elements Visibility" section will demonstrate how to show and hide the visual elements of the grid view. Last, but not least you will be introduced to the technique of formatting and exporting the content of the grid by different clipboard capabilities.

In the Binding section of this chapter you will connect the grid view to simple .NET objects. From there you will build a REST service against the Twitter API. Then you will be prompt to build WCF and WCF RIA services. In the process you will work with inherent Silverlight security restrictions. You will also learn how to customize the layout of an entire grid view row.

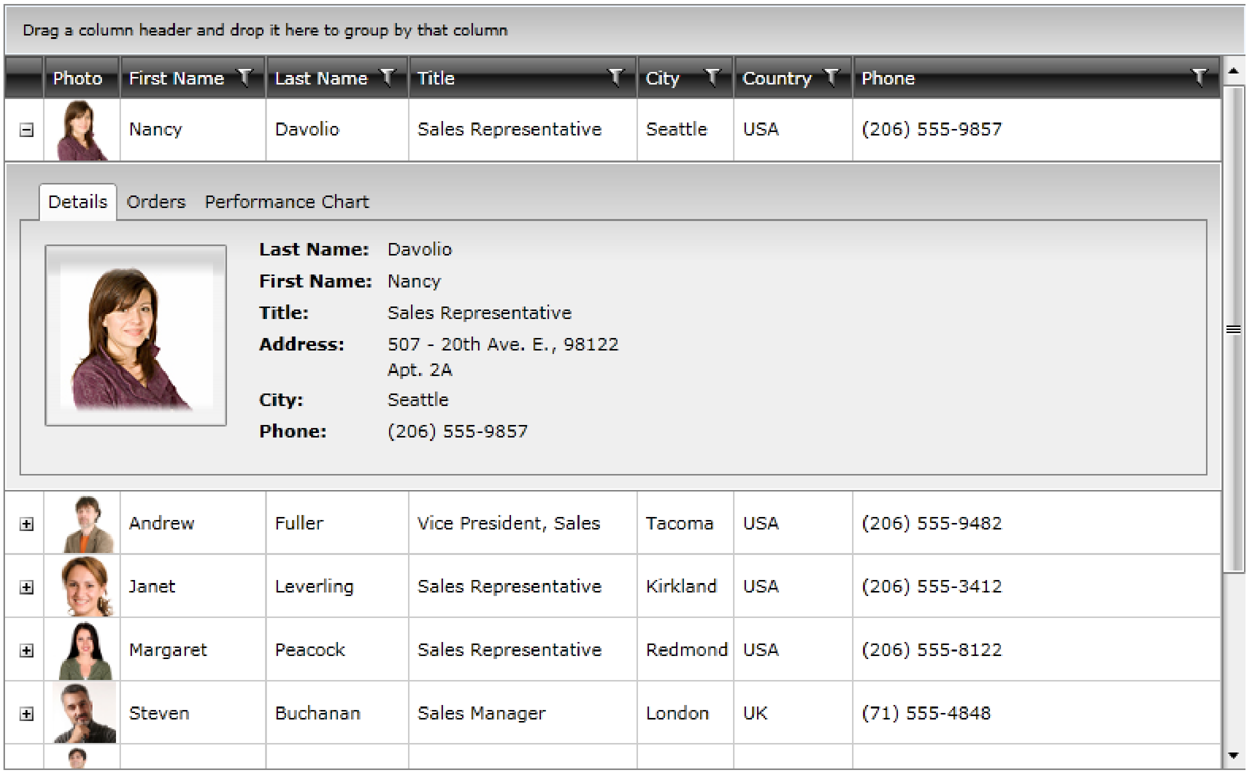
In the Customization section of this chapter you will customize grid view cells to achieve a unique look.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Gridview\Gridview.sln

18.2 Overview

RadGridView is the ultimate Silverlight grid control that features unrivalled performance through LINQ-based data engine, remarkably flexible hierarchy model, advanced features such as Excel-like filtering, row details, totals, export to Word/Excel/CSV and many more. RadGridView is a truly lookless Silverlight Grid that can be easily customized to blend perfectly into your applications.



Drag a column header and drop it here to group by that column

	Photo	First Name	Last Name	Title	City	Country	Phone
⊟		Nancy	Davolio	Sales Representative	Seattle	USA	(206) 555-9857

Details Orders Performance Chart

Last Name: Davolio
First Name: Nancy
Title: Sales Representative
Address: 507 - 20th Ave. E., 98122
Apt. 2A
City: Seattle
Phone: (206) 555-9857

	Photo	First Name	Last Name	Title	City	Country	Phone
⊞		Andrew	Fuller	Vice President, Sales	Tacoma	USA	(206) 555-9482
⊞		Janet	Leverling	Sales Representative	Kirkland	USA	(206) 555-3412
⊞		Margaret	Peacock	Sales Representative	Redmond	USA	(206) 555-8122
⊞		Steven	Buchanan	Sales Manager	London	UK	(71) 555-4848

RadGridView has a truly impressive list of features that make it a workhorse for your RIA applications:

- WPF/Silverlight Code Compatibility
- Support for WCF RIA Services
- LINQ-Based Data Engine and Native UI Virtualization
- Truly Lookless, Blend Skinnable, Completely Customizable Control
- RadCompression Module
- Direct Data Operations
- Powerful Databinding
- Asynchronous Databinding
- Data Source Updates
- Grouping and Aggregates
- Sorting
- Hierarchy
- In-place Editors and Built-in Data Validation
- Totals Row with Aggregate Functions
- Frozen Columns
- Export to Word/Excel/CSV
- Custom Layout
- Row Details
- Styling
- Gridlines Visibility
- Localization Support
- Selecting and Navigating
- Filtering and Excel-like Filtering

18.3 Getting Started

This project demonstrates binding to a grid view to simple category data being displayed in read-only mode.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.GridView**

c) **Telerik.Windows.Data**

d) **Telerik.Windows.Controls.Input.dll**

XAML Editing

1) Open MainPage.xaml for editing.

2) Drag a RadGridView control from the Toolbox to a point between the "LayoutRoot" <Grid> and </Grid> tags. Set the **x:Name** attribute to "gvMain" so the grid view can be referenced later in code. In order to work with the RadGridView for Silverlight 4 and its components, it is necessary to add a XML namespace reference to the Telerik.Windows.Controls assembly in the XAML markup. However, you may find it more convenient to use the Uri namespace that may be used for a reference to all included in the application Telerik assemblies.

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
...>
<Grid x:Name="LayoutRoot">
   <telerik:RadGridView x:Name="gvMain">
   </telerik:RadGridView>
</Grid>

</UserControl>
```

Code Behind

1) In the code-behind for the UserControl, add a Category class with a single string "Description" field. It should be defined outside and after the UserControl class.

VB

```
Public Class Category
  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
End Class
```

C#

```
public class Category
{
   public string Description { get; set; }
}
```

2) In the constructor, create a generic list of Category and assign it to the RadGridView **ItemsSource** property.

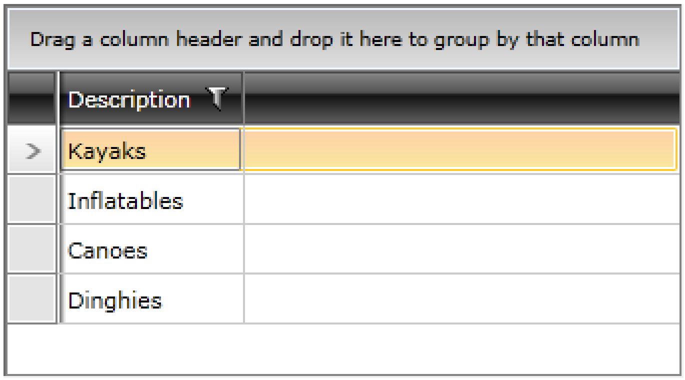
VB

```
Public Sub New()
  InitializeComponent()

  Dim categories As List(Of Category) = New List(Of Category) _
(New Category() {New Category() With { _
.Description = "Kayaks"}, New Category() With { _
.Description = "Inflatables"}, New Category() With { _
.Description = "Canoes"}, New Category() With { _
.Description = "Dinghies"}})
  gvMain.ItemsSource = categories
End Sub
```

```
public MainPage()
{
   InitializeComponent();

   List<Category> categories = new List<Category>()
   {
      new Category() { Description = "Kayaks" },
      new Category() { Description = "Inflatables" },
      new Category() { Description = "Canoes" },
      new Category() { Description = "Dinghies" }
   };
   gvMain.ItemsSource = categories;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Drag a column header and drop it here to group by that column

Description
Kayaks
Inflatables
Canoes
Dinghies

Test Application Features

The grid should display a single column of Category Description strings.

Ideas for Extending This Example

- *Display a hierarchy of data.*

For example, new "Product" objects may be shown under each Category in a master/detail relationship. In order to achieve this, add a **GridViewTableDefinition** to the grid view's **ChildTableDefinitions** collection. It specifies the collection property, containing the data about the children. In the example below, the Category class has been moved to its own class file and joined by a "Product" class.

```
Public Class Categories
  Inherits List(Of Category)
  Public Sub New()
    Me.Add(New Category() With { _
.Description = "Kayaks", . _
Products = New List(Of Product) (New Product() {New Product() With { _
.Description = "Touring Kayak"}, New Product() With { _
.Description = "Double Frame"}, New Product() With { _
.Description = "Ocean Kayak"}})})

    Me.Add(New Category() With { _
.Description = "Dinghies",  _
.Products = New List(Of Product) (New Product() {New Product() With { _
.Description = "Sport Dinghy"}, New Product() With { _
.Description = "Ribbed with Wood Floor"}, New Product() With { _
.Description = "Fishing Dinghy"}, New Product() With { _
.Description = "Yacht Tender"}})})

    Me.Add(New Category() With { _
.Description = "Canoes",  _
.Products = New List(Of Product) (New Product() {New Product() With { _
.Description = "Maine River Canoe"}, New Product() With { _
.Description = "Seattle Sport"}, New Product() With { _
.Description = "Inflatable Canoe"}})})
  End Sub
End Class

Public Class Category
  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
  Private privateProducts As List(Of Product)
  Public Property Products() As List(Of Product)
    Get
      Return privateProducts
    End Get
    Set(ByVal value As List(Of Product))
      privateProducts = value
    End Set
  End Property
End Class

Public Class Product
  Private privateDescription As String
  Public Property Description() As String
    Get
      Return privateDescription
    End Get
    Set(ByVal value As String)
      privateDescription = value
    End Set
  End Property
End Class
```

```
public class Categories : List<Category>
{
   public Categories()
   {
      this.Add(new Category()
      {
         Description = "Kayaks",
         Products = new List<Product>()
         {
            new Product() { Description = "Touring Kayak" },
            new Product() { Description = "Double Frame" },
            new Product() { Description = "Ocean Kayak" }
         }
      });

      this.Add(new Category()
      {
         Description = "Dinghies",
         Products = new List<Product>()
         {
            new Product() { Description = "Sport Dinghy" },
            new Product() { Description = "Ribbed with Wood Floor" },
            new Product() { Description = "Fishing Dinghy" },
            new Product() { Description = "Yacht Tender" }
         }
      });

      this.Add(new Category()
      {
         Description = "Canoes",
         Products = new List<Product>()
         {
            new Product() { Description = "Maine River Canoe" },
            new Product() { Description = "Seattle Sport" },
            new Product() { Description = "Inflatable Canoe" }
         }
      });

   }
}

public class Category
{
   public string Description { get; set; }
   public List<Product> Products { get; set; }
}

public class Product
{
   public Product() {}

   public string Description { get; set; }
}
```

Define a GridViewTableDefinition inside the constructor. Every such table definition has a Relation property that points to the collection of child objects.

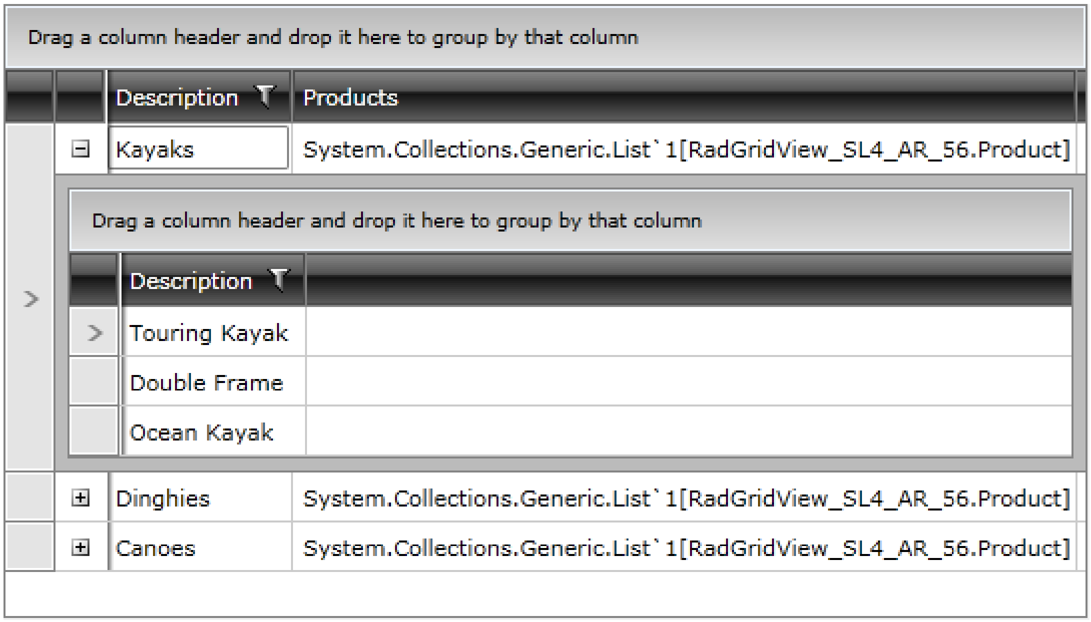
VB

```
Public Sub New()
  InitializeComponent()

  ' This class defines the field used to
  ' display a child table, in this case the
  ' "Products" collection of a "Category".
  Dim tableViewDefinition As New GridViewTableDefinition() With { _
.Relation = New PropertyRelation("Products")}
  gvMain.ChildTableDefinitions.Add(tableViewDefinition)

  ' assign the generic list of Category
  gvMain.ItemsSource = New Categories()
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();

   // This class defines the field used to
   // display a child table, in this case the
   // "Products" collection of a "Category".
   GridViewTableDefinition tableViewDefinition =
      new GridViewTableDefinition()
      {
         Relation = new PropertyRelation("Products")
      };
   gvMain.ChildTableDefinitions.Add(tableViewDefinition);

   // assign the generic list of Category
   gvMain.ItemsSource = new Categories();
}
```

Running in the browser, the example now looks like this screenshot:



Drag a column header and drop it here to group by that column

Description	Products
Kayaks	System.Collections.Generic.List`1[RadGridView_SL4_AR_56.Product]

Drag a column header and drop it here to group by that column

Description
Touring Kayak
Double Frame
Ocean Kayak

Description	Products
Dinghies	System.Collections.Generic.List`1[RadGridView_SL4_AR_56.Product]
Canoes	System.Collections.Generic.List`1[RadGridView_SL4_AR_56.Product]

- *Specify the columns to be displayed*

By default, the AutoGenerateColumns property of the grid is set to "False" and all columns in the bound object are created for you. However, you can easily control which columns should be displayed as well as their properties. For instance, the "Products" column that reads "System.Collection" at the moment can be removed. Furthermore, the Header property of the "Description" column may be modified to "Category Description". The first thing to be done is to set the **AutoGenerateColumns** property to "False", then add to the **Columns** collection.

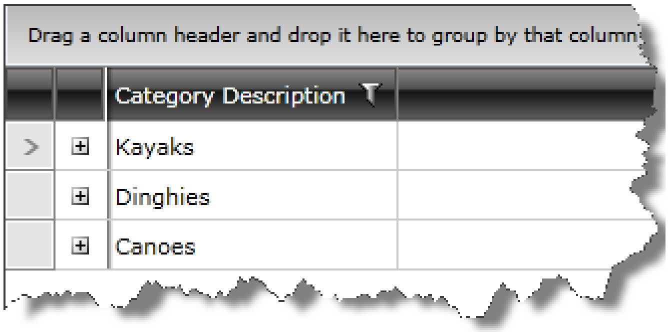
VB

```
' Only display the "Description" column, not the "Products" column
gvMain.AutoGenerateColumns = False
gvMain.Columns.Add(New GridViewDataColumn() With { _
.DataMemberBinding = New Binding("Description"), .Header = "Category Description"})
```

C#

```
// Only display the "Description" column, not the "Products" column
gvMain.AutoGenerateColumns = false;
gvMain.Columns.Add(new GridViewDataColumn()
{
   DataMemberBinding = new Binding("Description"),
   Header = "Category Description"
});
```

Now, only the new column description with its header displays in the grid.



Drag a column header and drop it here to group by that column

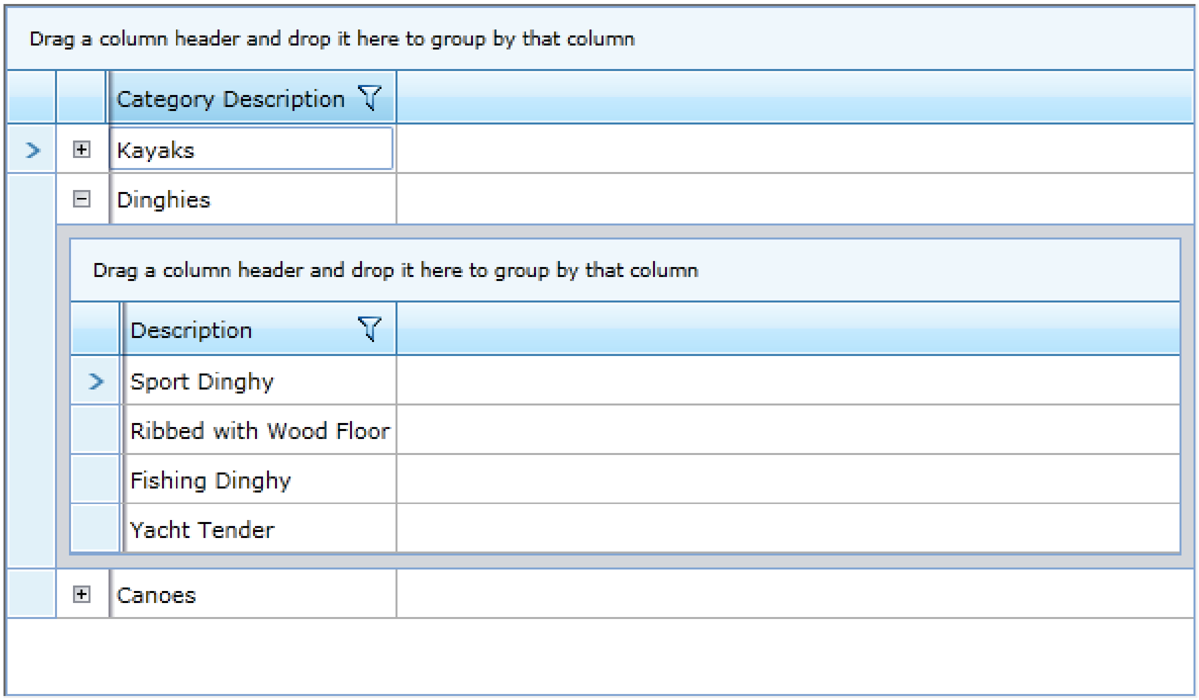
		Category Description
>	⊞	Kayaks
	⊞	Dinghies
	⊞	Canoes

- *Style the grid using one of the pre-defined themes.*

As with all the RadControls for Silverlight, you need to add the theme assembly (i.e. Telerik.Windows. Themes.Vista), add a XML namespace reference to the Telerik.Windows.Controls assembly in the XAML markup, and finally, add a StyleManager.Theme assignment to the theme. However, for facilitation you may use the common Uri namespace that gathers up all the relevant Telerik assemblies and you may use it to reference each one of them, once they are added in the applicaion. The relevant XAML markup will look like this example:

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .>

  <Grid x:Name="LayoutRoot">
    <telerik:RadGridView
      telerik:StyleManager.Theme="Vista"
      x:Name="gvMain">
     </telerik:RadGridView>
  </Grid>

</UserControl>
```

The RadGridView in the browser now displays the new theme without any other specific styling changes:



Drag a column header and drop it here to group by that column

Category Description
Kayaks
Dinghies

Drag a column header and drop it here to group by that column

Description
Sport Dinghy
Ribbed with Wood Floor
Fishing Dinghy
Yacht Tender

Category Description
Canoes

18.4 Control Details

18.4.1 Selections

The RadGridView provides you with a selection functionality, which allows the user to select one or more items from the data displayed by the control. The only thing you need to do for selecting an item is to click somewhere inside a row. Once a selection is made, the **SelectionChanged** event is fired and you are free to manipulate all selected and unselected elements. The most straight-forward way of getting the object for the selected row is to access the RadGridView **SelectedItem** property and cast the item to the appropriate type.

In the example below, the grid view is bound to a list of Category objects. The SelectedItem is retrieved and cast it to a Category type. Afterwards, you can easily use its to populate a TextBlock for instance.

```
Private Sub gvMain_SelectionChanged( _
ByVal sender As Object, ByVal e As SelectionChangeEventArgs)
  tbCurrentItem.Text = (TryCast(gvMain.SelectedItem, Category)).CategoryName
End Sub
```

C#

```
private void gvMain_SelectionChanged(
  object sender, SelectionChangeEventArgs e)
{
  tbCurrentItem.Text = (gvMain.SelectedItem as Category).CategoryName;
}
```

RadGridView provides another feature at your hands - control of the SelectionMode. It can be set by the equally named property and have the following values:

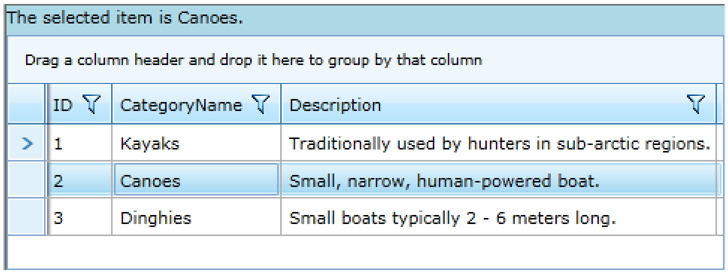
- Single - only one item can be selected at a time. (default value)
- Multiple - items are added to the selection when they get clicked and get removed when they get clicked again.
- Extended - items are added to the selection only by combining the mouse clicks with the Ctrl or Shift key.

Furthermore, as of Q2 2010, you can select not only rows, but also single or multiple cells. The desired behavior can be easily set by the SelectionUnit property, whose value may be:

- FullRow - this is the default value. Clicking within the cells will select the row
- Cell - the clicked cell is selected only. Depending on the value of the SelectionModes property you can have more than one selected cell.

Both collections for selected items (rows) and selected cells may be manipulated through the SelectedItems and SelectedCells Collections.

The example below uses LINQ to transform the SelectedItems into a collection of String. The String.Join() method turns the collection into a comma delimited list for displaying it in a TextBlock. The "CategoryName" for the three selected items is shown at the top of the screenshot.



The selected item is Canoes.

Drag a column header and drop it here to group by that column

	ID	CategoryName	Description
>	1	Kayaks	Traditionally used by hunters in sub-arctic regions.
	2	Canoes	Small, narrow, human-powered boat.
	3	Dinghies	Small boats typically 2 - 6 meters long.

VB

```
Private Sub gvMain_SelectionChanged(ByVal sender As Object, ByVal e As SelectionChangeEventArgs)
  Dim items = _
    From c As Category In gvMain.SelectedItems _
    Select c.CategoryName
  tbCurrentItem.Text = String.Join(", ", items.ToArray())
End Sub
```

C#

```
private void gvMain_SelectionChanged(
   object sender, SelectionChangeEventArgs e)
{
   var items = from Category c in gvMain.SelectedItems select c.CategoryName;
   tbCurrentItem.Text = String.Join(", ", items.ToArray());
}
```

If the selected item has scrolled away from view, you can use the **ScrollIntoView()** method and pass it the data bound to a particular row. For example, the code below gets the item for the last row, adds it to the SelectedItems collection if its not there already and then calls ScrollIntoView(), passing it the last item.

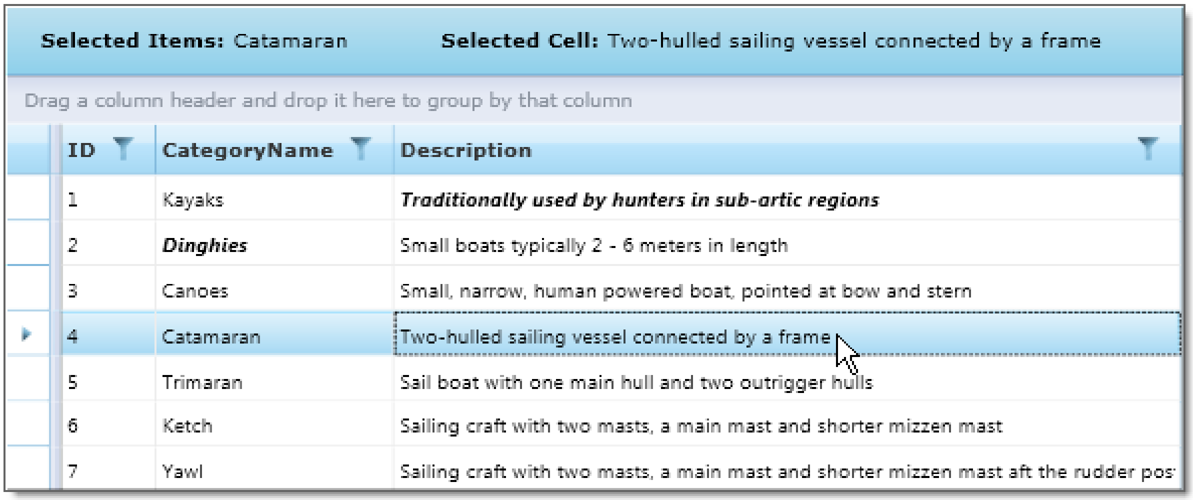
VB

```
Private Sub btnSelectLast_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim lastItem = gvMain.Items(gvMain.Items.Count - 1)
  If (Not gvMain.SelectedItems.Contains(lastItem)) Then
    gvMain.SelectedItems.Add(lastItem)
  End If
  gvMain.ScrollIntoView(lastItem)
End Sub
```

C#

```
private void btnSelectLast_Click(object sender, RoutedEventArgs e)
{
   var lastItem = gvMain.Items[gvMain.Items.Count - 1];
   if (!gvMain.SelectedItems.Contains(lastItem))
      gvMain.SelectedItems.Add(lastItem);
   gvMain.ScrollIntoView(lastItem);
}
```

You can get a selected cell by handling the **CurrentCellChanged** event. The event arguments passed in include **OldCell** and **NewCell** references. Not only can retrieve the **Content** of the cell, but you can alter all the properties of the cell. In the example below, the OldCell font is set to a bold italic and the NewCell Content is displayed in TextBox at the top of the page. **Note**: use the cell **DataContext** to get the object bound to the row.



Selected Items: Catamaran **Selected Cell:** Two-hulled sailing vessel connected by a frame

Drag a column header and drop it here to group by that column

ID	CategoryName	Description
1	Kayaks	***Traditionally used by hunters in sub-artic regions***
2	***Dinghies***	Small boats typically 2 - 6 meters in length
3	Canoes	Small, narrow, human powered boat, pointed at bow and stern
4	Catamaran	Two-hulled sailing vessel connected by a frame
5	Trimaran	Sail boat with one main hull and two outrigger hulls
6	Ketch	Sailing craft with two masts, a main mast and shorter mizzen mast
7	Yawl	Sailing craft with two masts, a main mast and shorter mizzen mast aft the rudder pos

```
Private Sub gvMain_CurrentCellChanged( _
ByVal sender As Object, ByVal e As GridViewCurrentCellChangedEventArgs)
  If e.OldCell IsNot Nothing Then
    e.OldCell.FontStyle = FontStyles.Italic
    e.OldCell.FontWeight = FontWeights.Bold
  End If
  tbCurrentCell.Text = DirectCast(e.NewCell.Content, TextBlock).Text
End Sub
```

```
private void gvMain_CurrentCellChanged(
  object sender, GridViewCurrentCellChangedEventArgs e)
{
  if (e.OldCell != null)
  {
    e.OldCell.FontStyle = FontStyles.Italic;
    e.OldCell.FontWeight = FontWeights.Bold;
  }
  tbCurrentCell.Text = ((TextBlock)e.NewCell.Content).Text;
}
```

From the Forums...

Question: How to make my first row selected initially and synchronize it with the current item?

Answer: Initially, the current item of the grid is the first one to be displayed. The current item is the one that holds the data item (business object) of the row which currently keeps the focus. However,it may or may not coincide with the selected row. What you can do in order to synchronize those two properties and make your first item selected is to use the property of the grid - IsSynchronizedWithCurrentItem. Once it is set to "True", you will get the desired result.

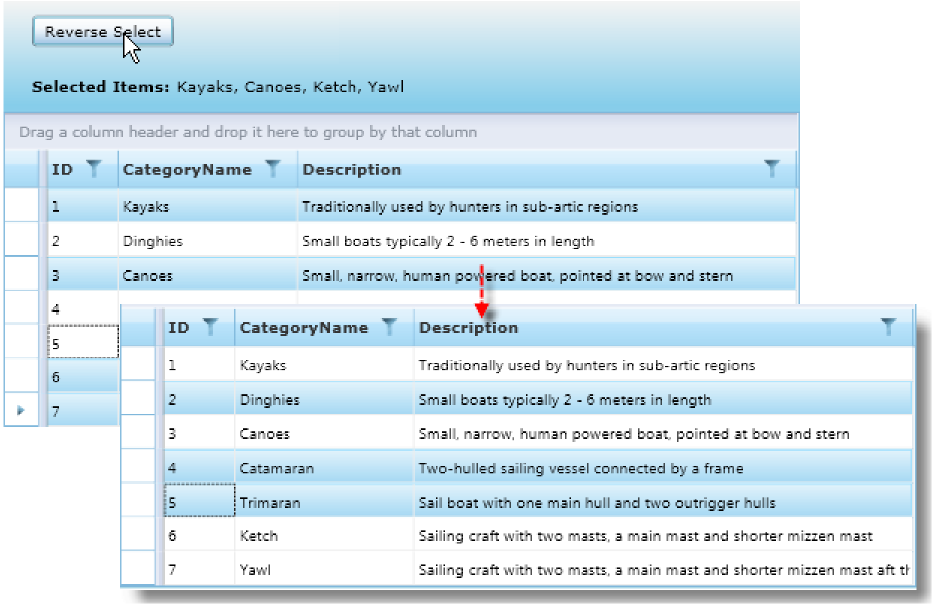
Question: I am currently writing a method to reverse the selected rows in the RadGridView. But adding items to the selected items in a "for" loop its becomes very slow. (It actually hangs the UI on the test page as there are about 5000 rows to select.) The SelectAll and UnselectAll methods on the grid are a lot quicker though (even with the 5000 of rows). So I was wondering is there a way to load the selected items collection quickly?

On a side note is it possible to display the Asynchronous animation from a call in the code behind? As I would quite like to display the "busy" message while we wait for a service call to return or when we invert the selection.

Answer: 1) You could utilize the UnselectAll() and SelectAll() methods to boost the performance when reverting the selection.

2) RadGridView has a property called **IsBusy** that controls the visibility of its loading indicator - when set to true the loading indicator is shown. You can use this property to control the loading indicator when you are waiting for a response from a service call.

You can select and de-select all items at once using the SelectAll() method and UnselectAll() methods. You can also add and remove from the SelectedItems collection programmatically, as shown in this next example where the selections for the entire grid are toggled. The code first determines if there are a greater number of selected or unselected items in order to determine the most efficient algorithm. The selected items are saved temporarily, the SelectAll()/UnselectAll() method is called, then the saved selected items are added or removed from the SelectedItems collection.


Reverse Select
Drag a column header and drop it here to group by that column

Selected Items: Kayaks, Canoes, Ketch, Yawl

ID	CategoryName	Description
1	Kayaks	Traditionally used by hunters in sub-artic regions
2	Dinghies	Small boats typically 2 - 6 meters in length
3	Canoes	Small, narrow, human powered boat, pointed at bow and stern
4		
5		
6		
7		

ID	CategoryName	Description
1	Kayaks	Traditionally used by hunters in sub-artic regions
2	Dinghies	Small boats typically 2 - 6 meters in length
3	Canoes	Small, narrow, human powered boat, pointed at bow and stern
4	Catamaran	Two-hulled sailing vessel connected by a frame
5	Trimaran	Sail boat with one main hull and two outrigger hulls
6	Ketch	Sailing craft with two masts, a main mast and shorter mizzen mast
7	Yawl	Sailing craft with two masts, a main mast and shorter mizzen mast aft th

```
Private Sub btnReverseSelect_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  gvMain.IsBusy = True

  If gvMain.SelectedItems.Count <= gvMain.Items.Count Then
    ReverseWithSelectAll()
  Else
    ReverseWithUnselectAll()
  End If

  gvMain.IsBusy = False
End Sub

Private Sub ReverseWithUnselectAll()
  var old SelectedItems = gvMain.SelectedItems.ToList()
  gvMain.UnselectAll()

  For Each item In (CType(gvMain.ItemsSource, IList))
    If (Not oldSelectedItems.Contains(item)) Then
      gvMain.SelectedItems.Add(item)
    End If
  Next item
End Sub

Private Sub ReverseWithSelectAll()
  Dim oldSelectedItems = gvMain.SelectedItems.ToList()
  gvMain.SelectAll()

  For Each item In oldSelectedItems
    gvMain.SelectedItems.Remove(item)
  Next item
End Sub
```

```
private void btnReverseSelect_Click(object sender, RoutedEventArgs e)
{
   gvMain.IsBusy = true;

   if (gvMain.SelectedItems.Count <= gvMain.Items.Count)
      ReverseWithSelectAll();
   else
      ReverseWithUnselectAll();

   gvMain.IsBusy = false;
}

private void ReverseWithUnselectAll()
{
   var old SelectedItems = gvMain.SelectedItems.ToList();
   gvMain.UnselectAll();

   foreach (var item in ((IList)gvMain.ItemsSource))
   {
      if (!oldSelectedItems.Contains(item))
         gvMain.SelectedItems.Add(item);
   }
}

private void ReverseWithSelectAll()
{
   var oldSelectedItems = gvMain.SelectedItems.ToList();
   gvMain.SelectAll();

   foreach (var item in oldSelectedItems)
   {
      gvMain.SelectedItems.Remove(item);
   }
}
```

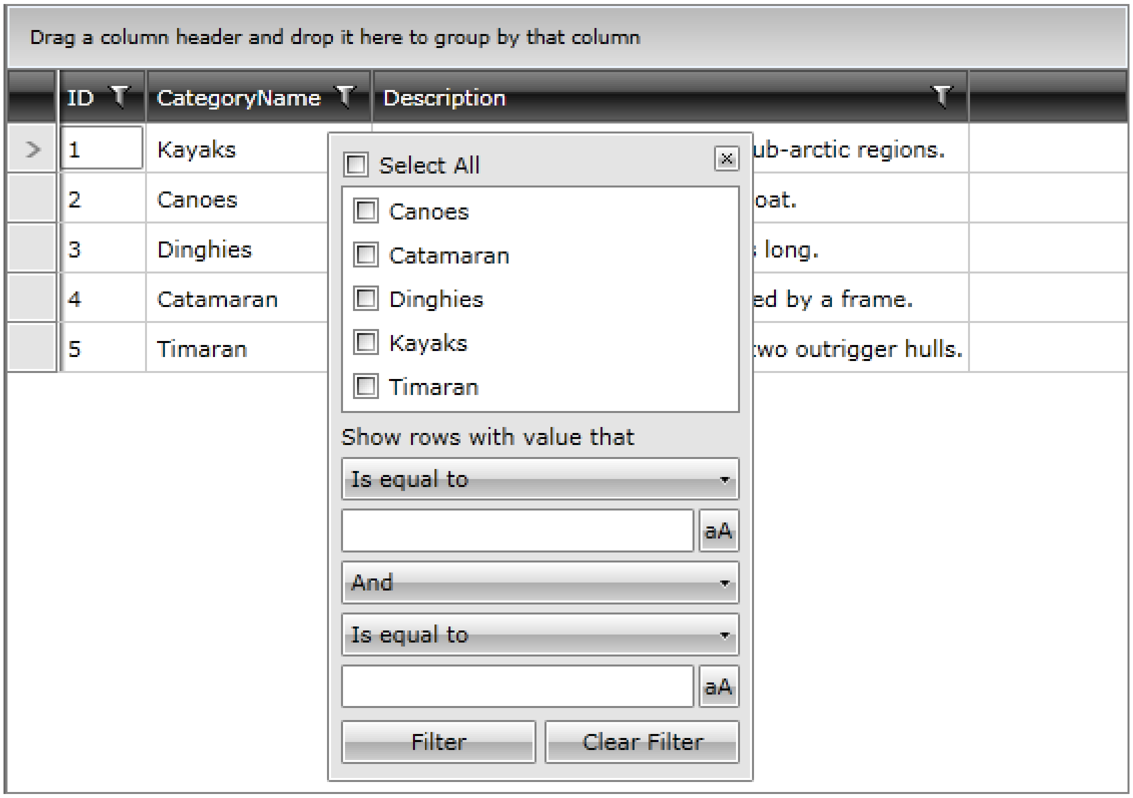
18.4.2 Filtering Sorting and Grouping

Filtering, sorting and grouping all follow a similar programming model. Each have a collection of descriptor objects. The descriptor defines the operation specifics, e.g. what column to sort/ filter/ group on. The descriptor is simply added to the collection to enable the operation. The upcoming sections demonstrate defining each kind of descriptor, adding to the collection, and viewing the result effect in a screenshots.

Parallel to this mechanism of descriptor collections, each operation also has **Filtering**, **Sorting** and **Grouping** events. These events allow you to get "down to the metal", so you can react to non-standard or more complex situations. Each event supplies "new" and "old" descriptor objects as well as a **Cancel** property to prevent the operation from occurring.

18.4.2.1 Filtering

You can filter RadGridView on the client, assuming you are not filtering on the database server, or further up, in a service before returning the data to the Silverlight client. You will need to consider the performance tradeoffs when designing your solution. By default the user can filter by clicking the "funnel" icons available on each group header. The pop-up filter dialog allows the user to check particular values or specify a filter criteria using an operation and value. Multiple filters can be "AND"-ed together. Furthermore, the user can filter multiple columns.



Drag a column header and drop it here to group by that column

ID	CategoryName	Description
1	Kayaks	ub-arctic regions.
2	Canoes	oat.
3	Dinghies	: long.
4	Catamaran	ed by a frame.
5	Timaran	:wo outrigger hulls.

□ Select All

□ Canoes
□ Catamaran
□ Dinghies
□ Kayaks
□ Timaran

Show rows with value that

Is equal to

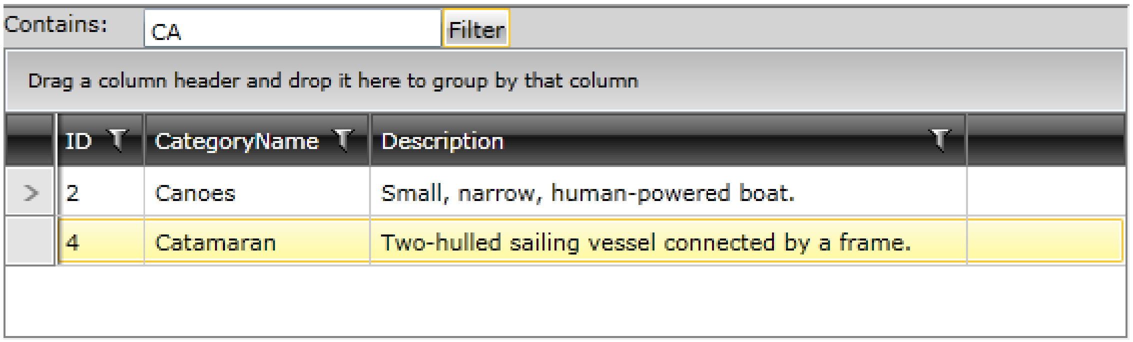
And

Is equal to

The grid view **FilterDescriptors** collection lets you add a pre-defined filter or a custom filter of your own that implements **IFilterDescriptor**. The most important classes that implement this interface directly or not are:

- **Filter Descriptor** - implements filtering property (field) name, filtering operator and value. It is used for defining filtering expressions like Country = "Germany".
- **CompositeFilterDescriptor** - a collection of multiple filter descriptors with logical operator. It is used for defining complex filtering expressions like (Country = "Germany" AND (City = "Berlin" OR City = "Aachen")).
- **ColumnFilterDescriptor** - a collection of multiple filter descriptors for single property (field). This type of object is being created and added to the RadGridView's filters collection when a filter is defined using the user interface.

The example below uses the built-in FilterDescriptor class - a basic filter that takes a data member name, an operation and value.



Contains: CA

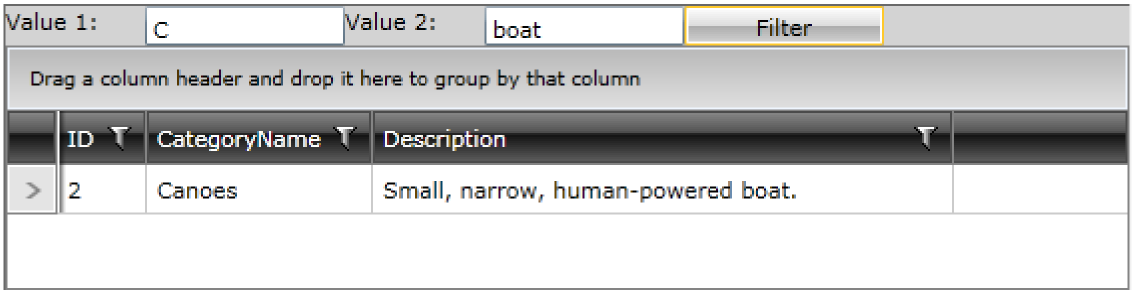
Filter
Drag a column header and drop it here to group by that column

ID	CategoryName	Description
2	Canoes	Small, narrow, human-powered boat.
4	Catamaran	Two-hulled sailing vessel connected by a frame.

```
Private Sub btnFilter_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim filter As New FilterDescriptor("CategoryName", _
FilterOperator.Contains, tbFilter.Text)
  gvMain.FilterDescriptors.Add(filter)
End Sub
```

```
private void btnFilter_Click(object sender, RoutedEventArgs e)
{
   FilterDescriptor filter =
      new FilterDescriptor("CategoryName",
        FilterOperator.Contains, tbFilter.Text);
   gvMain.FilterDescriptors.Add(filter);
}
```

You can simply add more filters and they will "AND" together like this example where only one record survives the process:


Filter
Drag a column header and drop it here to group by that column

Value 1: C
Value 2: boat

ID	CategoryName	Description
2	Canoes	Small, narrow, human-powered boat.

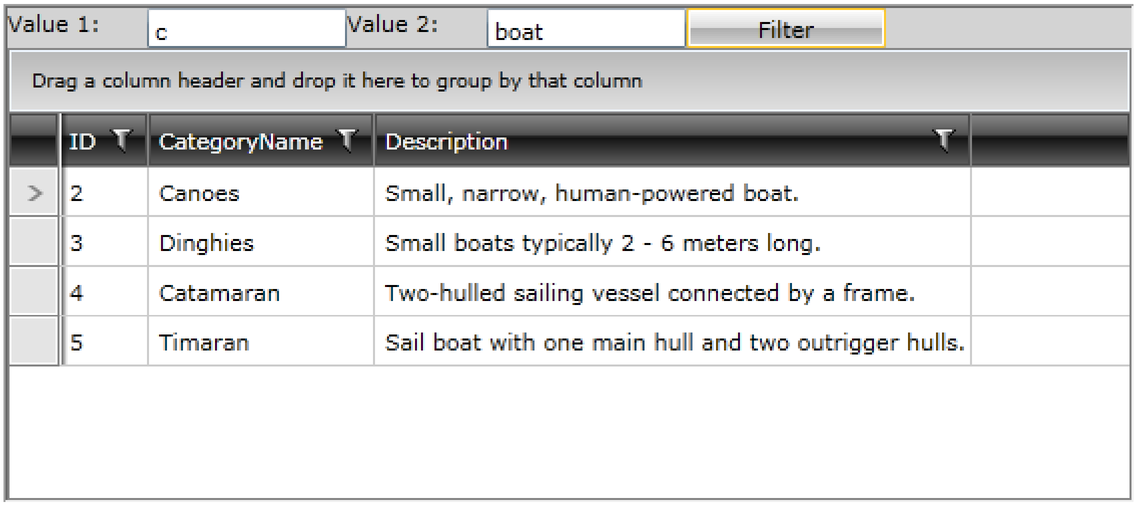
VB

```
Dim filter As New FilterDescriptor("CategoryName", _
FilterOperator.StartsWith, tbValue1.Text)
gvMain.FilterDescriptors.Add(filter)
Dim filter2 As New FilterDescriptor("Description", _
FilterOperator.Contains, tbValue2.Text)
gvMain.FilterDescriptors.Add(filter2)
```

C#

```
FilterDescriptor filter =
  new FilterDescriptor("CategoryName", FilterOperator.StartsWith, tbValue1.Text);
gvMain.FilterDescriptors.Add(filter);
FilterDescriptor filter2 =
  new FilterDescriptor("Description", FilterOperator.Contains, tbValue2.Text);
gvMain.FilterDescriptors.Add(filter2);
```

To "OR" filters together, **CompositeFilterDescriptor** lets you set a logical operator that defines the relationship between two filters. You can add any IFilterDescriptor implementation to the CompositeFilterDescriptor **FilterDescriptors** collection and define the **LogicalOperator** property as "Or" or "AND". The example below uses the same FilterDescriptors and values as the previous example, but adds those two descriptors to a ComposititeFilterDescriptor with an "Or" logical operator. Now the filter operation brings back a wider set of data.



Value 1: c Value 2: boat

Drag a column header and drop it here to group by that column

ID	CategoryName	Description
2	Canoes	Small, narrow, human-powered boat.
3	Dinghies	Small boats typically 2 - 6 meters long.
4	Catamaran	Two-hulled sailing vessel connected by a frame.
5	Timaran	Sail boat with one main hull and two outrigger hulls.

```
Dim composite As New CompositeFilterDescriptor()

Dim filter As New FilterDescriptor("CategoryName", _
FilterOperator.StartsWith, tbValue1.Text)
Dim filter2 As New FilterDescriptor("Description", _
FilterOperator.Contains, tbValue2.Text)
composite.FilterDescriptors.Add(filter)
composite.LogicalOperator = FilterCompositionLogicalOperator.Or
composite.FilterDescriptors.Add(filter2)

gvMain.FilterDescriptors.Add(composite)
```

```
CompositeFilterDescriptor composite = new CompositeFilterDescriptor();

FilterDescriptor filter =
new FilterDescriptor("CategoryName", FilterOperator.StartsWith, tbValue1.Text);
FilterDescriptor filter2 =
  new FilterDescriptor("Description", FilterOperator.Contains, tbValue2.Text);
composite.FilterDescriptors.Add(filter);
composite.LogicalOperator = FilterCompositionLogicalOperator.Or;
composite.FilterDescriptors.Add(filter2);

gvMain.FilterDescriptors.Add(composite);
```

Disabling Filtering

By default, users can filter against any column they want. The **IsFilteringAllowed** property manipulates the capability for the grid as a whole. **IsFilterable** toggles filtering only for a single column. When IsFilterable is set to "False", the filtering UI for that column is hidden from the user, but still available programmatically.

In case you want to remove the filters programmatically, you may call the FilterDescriptors **Clear()** method.

Distinct Values

You can get a list of distinct values for a column using the **GetDistinctValues()** method and passing the column where the values should come from and a Boolean that indicates if the existing filters in the other columns should be honored. The example below gets a distinct list of categories. If a CategoryName appears more than once in the column of the grid, the returned collection will only have a single occurrence of the category name.

VB

```
Dim column As GridViewDataColumn = _
TryCast(gvMain.Columns("CategoryName"), GridViewDataColumn)
lbDistinct.ItemsSource = gvMain.GetDistinctValues(column, False)
```

C#

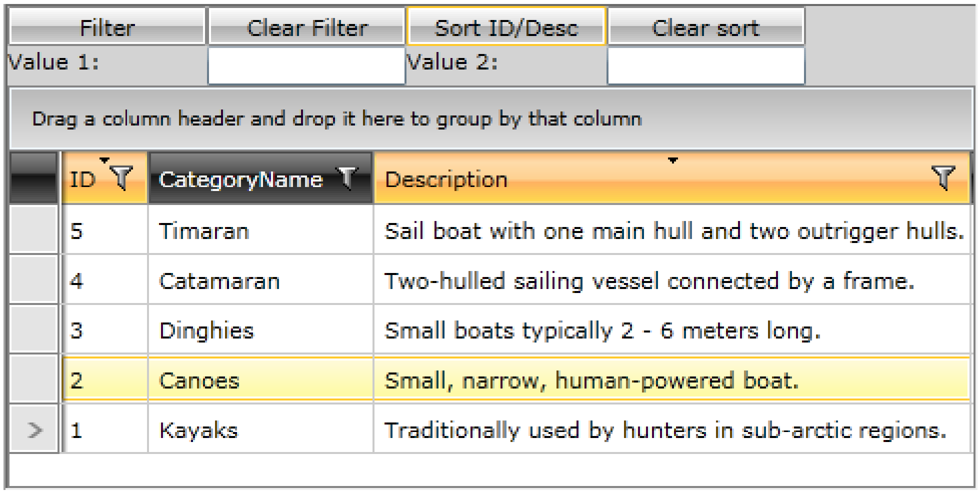
```
GridViewDataColumn column = gvMain.Columns["CategoryName"] as GridViewDataColumn;
lbDistinct.ItemsSource = gvMain.GetDistinctValues(column, false);
```

18.4.2.2 Sorting

Sorting in the RadGridView from the user perspective, simply means clicking the heading of any row and watching the sort rotate between sorted ascending, sorted descending and unsorted states. The user can also hold down the SHIFT key to sort on multiple columns.

The general model for sorting programmatically is similar to the one of filtering. RadGridView has a **SortDescriptors** collection that you can add one or more **SortDescriptor** to. The SortDescriptor has properties for the **Member** (the property towards which the sorting will be accomplished) and a **SortDirection** (it can be set to either **Ascending** or **Descending**). Furthermore, like the filtering model, you simply need to call the SortDescriptors **Clear()** method to reset them. The sorting functionality can be controlled on a grid level by setting the **CanUserSortColumns** or on a column level by setting **IsSortable** property.

The example below programmatically sorts by ID and Description in descending order. Notice the downward pointing arrows in the column headers that indicate the descending sort order.


Filter
Clear Filter
Sort ID/Desc
Clear sort
Value 1:
Value 2:
Drag a column header and drop it here to group by that column

ID	CategoryName	Description
5	Timaran	Sail boat with one main hull and two outrigger hulls.
4	Catamaran	Two-hulled sailing vessel connected by a frame.
3	Dinghies	Small boats typically 2 - 6 meters long.
2	Canoes	Small, narrow, human-powered boat.
1	Kayaks	Traditionally used by hunters in sub-arctic regions.

```
Private Sub btnSort_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  gvMain.Columns("CategoryName").IsSortable = False
  gvMain.SortDescriptors.Add(New SortDescriptor() With { _
.Member = "ID", _
.SortDirection = System.ComponentModel.ListSortDirection.Descending})
  gvMain.SortDescriptors.Add(New SortDescriptor() With { _
.Member = "Description", _
.SortDirection = System.ComponentModel.ListSortDirection.Descending})
End Sub

Private Sub btnClearSort_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  gvMain.SortDescriptors.Clear()
End Sub
```

C#

```
private void btnSort_Click(object sender, RoutedEventArgs e)
{
   gvMain.Columns["CategoryName"].IsSortable = false;
   gvMain.SortDescriptors.Add(new SortDescriptor()
   {
      Member = "ID",
      SortDirection = System.ComponentModel.ListSortDirection.Descending
   });
   gvMain.SortDescriptors.Add(new SortDescriptor()
   {
      Member = "Description",
      SortDirection = System.ComponentModel.ListSortDirection.Descending
   });
}

private void btnClearSort_Click(object sender, RoutedEventArgs e)
{
   gvMain.SortDescriptors.Clear();
}
```

In case you need to perform more complicated custom sorting, you may handle the **Sorting** or **Sorted** events of the RadGridView. The arguments of the first event handler allow you to get the sorting state (Ascending, Descending or None) by the corresponding property - **NewSortingState**. Furthermore, the sorting may be canceled at this level by using the **Cancel** property. Another important property available in both handlers is Column - it is of type GridViewColumn and may be cast to a specific column type depending on your requirements.

You may take a look at the code below for an example on the way to prevent sorting on integer columns.

VB

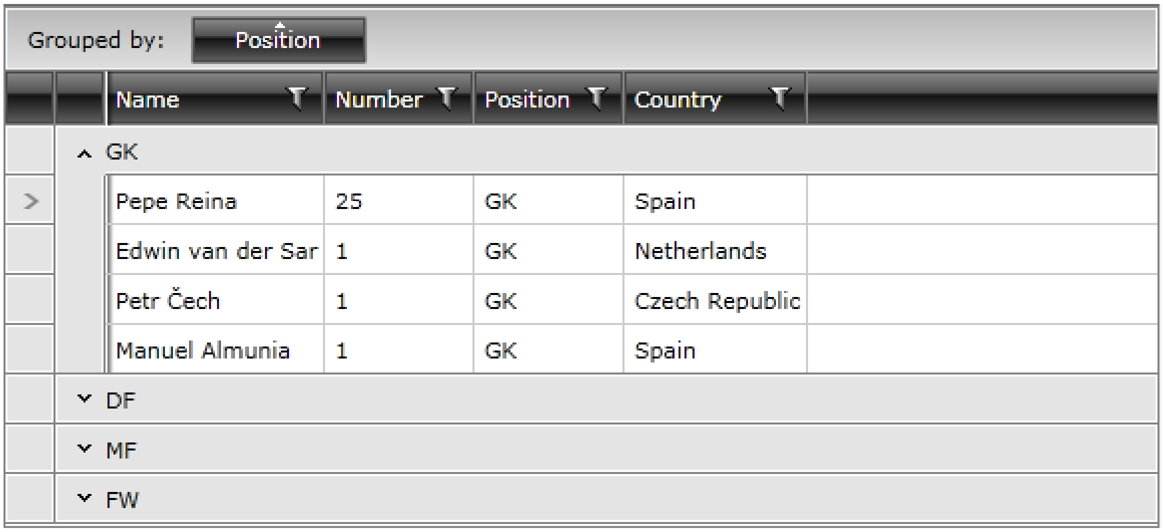
```
Private Sub gvMain_Sorting( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.GridViewSortingEventArgs)
  If TypeOf e.Column Is GridViewDataColumn Then
    Dim column As GridViewDataColumn = _
TryCast(e.Column, GridViewDataColumn)
    If e.NewSortingState = SortingState.Descending Then
      If column.DataType Is GetType(System.Int32) Then
        e.Cancel = True
      End If
    End If
  End If
End Sub
```

C#

```
private void gvMain_Sorting(object sender,
   Telerik.Windows.Controls.GridViewSortingEventArgs e)
{
   if (e.Column is GridViewDataColumn)
   {
      GridViewDataColumn column = e.Column as GridViewDataColumn;
      if (e.NewSortingState == SortingState.Descending)
      {
         if (column.DataType == typeof(System.Int32))
         {
            e.Cancel = true;
         }
      }
   }
}
```

18.4.2.3 Grouping

RadGridView allows the user to create groups depending on the values in a particular column. This can be achieved by dragging a column's header into the group panel. Consequently, the group can be removed either by dragging the group off the page or clicking the close button, or the user can click the group to cycle the sort order between ascending, descending and no sort order.



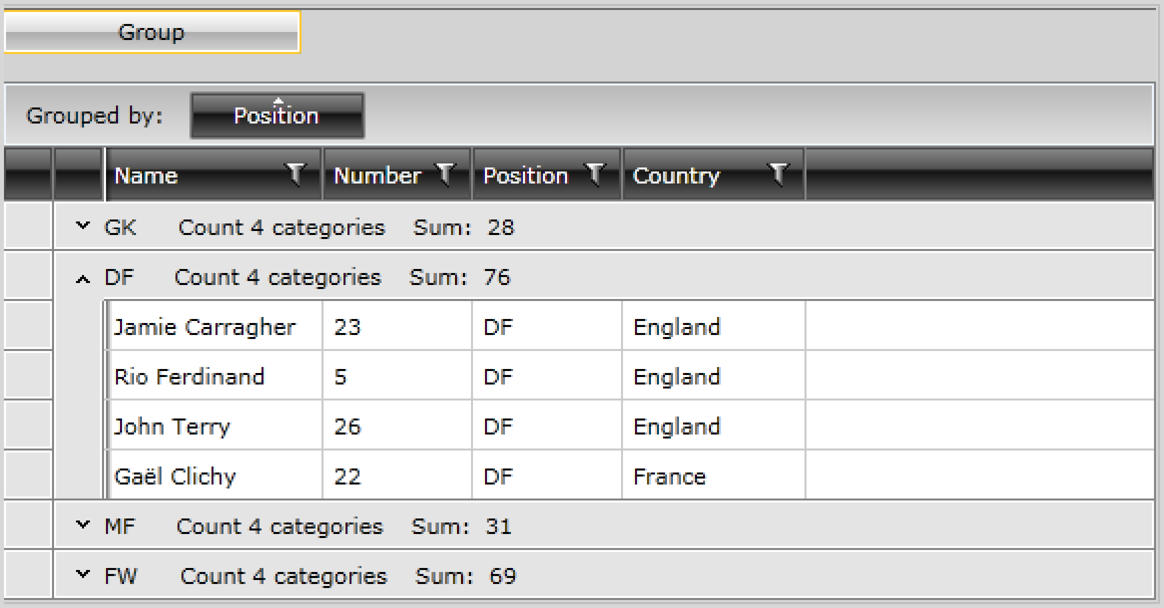
Grouped by: Position

Name	Number	Position	Country
GK			
Pepe Reina	25	GK	Spain
Edwin van der Sar	1	GK	Netherlands
Petr Čech	1	GK	Czech Republic
Manuel Almunia	1	GK	Spain
DF			
MF			
FW			

RadGridView also provides the ability of defining aggregate functions for each group. Thus you may easily display the total count of the items in each group, the minimum value, their sum, etc. Again, like the filtering and sorting models, each GroupDescriptor is held in the GroupDescriptors collection. Each GroupDescriptor defines a data **Member** property and a **SortDirection**. Then you can add to the GroupDescriptor **AggregateFunctions** collection. As mentioned previously, all the usual aggregate suspects are represented: count, sum, min, max, average, first and last. Each has a corresponding "Function" object, e. g. CountFunction, SumFunction, etc, that describes the source field to aggregate, the caption that appears before the aggregate result and a formatted string to present the result.

The example below groups on the Category column and sets the sort order to ascending. There are two aggregates defined. The CountFunction definition demonstrates how to use the **Caption** and **ResultFormatString**. The second is a SumFunction that defines the "InStock" property as the **SourceField**.

The SourceField property for the category in the CountFunction code is not defined because the field is assumed be the group column, i.e. a count of the "Category" source field.



Name	Number	Position	Country
GK Count 4 categories Sum: 28			
DF Count 4 categories Sum: 76			
Jamie Carragher	23	DF	England
Rio Ferdinand	5	DF	England
John Terry	26	DF	England
Gaël Clichy	22	DF	France
MF Count 4 categories Sum: 31			
FW Count 4 categories Sum: 69			

```
Private Sub btnGroup_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim group As New GroupDescriptor()
  group.Member = "Category"
  group.SortDirection = ListSortDirection.Ascending

  Dim countFunction As New CountFunction() With { _
.Caption = "Count", .ResultFormatString = "{0} categories"}
  group.AggregateFunctions.Add(countFunction)

  Dim sumFunction As New SumFunction() With { _
.SourceField = "InStock", .Caption = "In Stock"}
  group.AggregateFunctions.Add(sumFunction)

  gvMain.GroupDescriptors.Add(group)
End Sub
```

C#

```
private void btnGroup_Click(object sender, RoutedEventArgs e)
{
   GroupDescriptor group = new GroupDescriptor();
   group.Member = "Category";
   group.SortDirection = ListSortDirection.Ascending;

   CountFunction countFunction = new CountFunction()
   {
      Caption = "Count",
      ResultFormatString = "{0} categories"
   };
   group.AggregateFunctions.Add(countFunction);

   SumFunction sumFunction = new SumFunction()
   {
      SourceField = "InStock",
      Caption = "In Stock"
   };
   group.AggregateFunctions.Add(sumFunction);

   gvMain.GroupDescriptors.Add(group);
}
```

Expanding and Collapsing Groups

Use the **CollapseGroup()** and **ExpandGroup()** methods to toggle expansion of individual groups. Pass those methods an **IGroup** item as a parameter that you can locate using System.Linq methods. The example below uses the First() LINQ extension method. You can also call **CollapseAllGroups()** and **ExpandAllGroups()** to toggle expansion for all groups.

VB

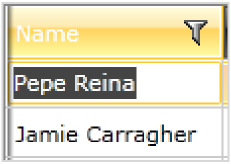
```
Dim group = gvMain.Items.Cast(Of IGroup)().First()
gvMain.CollapseGroup(group)
' or
gvMain.ExpandGroup(group)
```

C#

```
var group = gvMain.Items.Cast<IGroup>().First();
gvMain.CollapseGroup(group);
// or
gvMain.ExpandGroup(group);
```

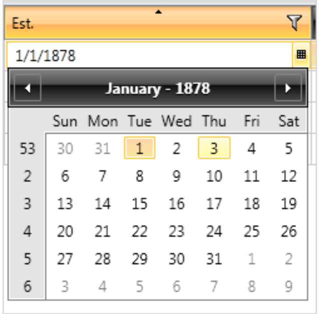
18.4.3 Editing

RadGridView can automatically supply an appropriate editor based on the data type of the column. These all use the **GridViewDataColumn** type. Here are some of the editors you will see for the GridViewDataColumn:



Name
Pepe Reina
Jamie Carragher

The most basic editor handles text input in place.



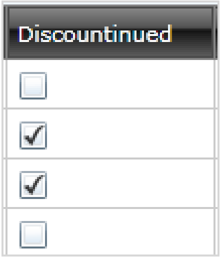
Est.

1/1/1878

January - 1878

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
53	30	31	1	2	3	4	5
2	6	7	8	9	10	11	12
3	13	14	15	16	17	18	19
4	20	21	22	23	24	25	26
5	27	28	29	30	31	1	2
6	3	4	5	6	7	8	9

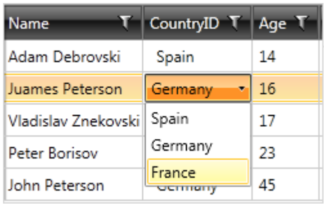
DateTime values use a built-in date picker editor.



Discountinued
☐
☑
☑
☐

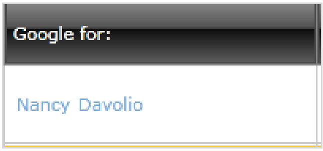
Boolean values are edited using a check box.

You can also define specific-purpose editors, such as the **GridViewComboBoxColumn** column.



Name	CountryID	Age
Adam Debrovski	Spain	14
Juames Peterson	Germany	16
Vladislav Znekovski	Spain	17
Peter Borisov	Germany	23
John Peterson	France	45

Lookups are handled using a combo box. You can define the value field that is to be updated in the grid view, the display member for the combo box and the selected value for the combo box.



Google for:
Nancy Davolio

Two hyperlink column types have been introduced: **GridViewHyperlinkColumn** and **GridViewDynamicHyperlinkColumn** that let you bind a URL. The "Dynamic" version lets you specify a format string and bind multiple columns for the values in the format string.

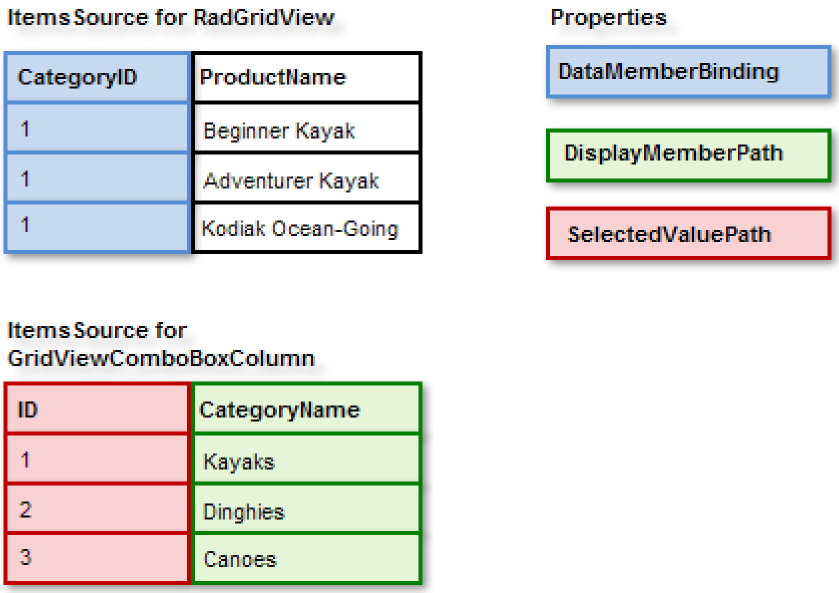
You may take a look at the code below for more details on the way those columns are defined. The GridViewDataColumns are handled automatically simply by binding to the column name. Minimally, you can set the DataMemberBinding and "you're good to go".

```
<telerik:RadGridView x:Name="gvMain"
    Margin="10" AutoGenerateColumns="False"
    ItemsSource="{StaticResource Products}">

  <telerik:RadGridView.Columns>
    . . .
    <telerik:GridViewDataColumn
        DataMemberBinding="{Binding ProductName, Mode=TwoWay}" />
    <telerik:GridViewDataColumn
        DataMemberBinding="{Binding Discontinued, Mode=TwoWay}" />
    <telerik:GridViewDataColumn
        DataMemberBinding="{Binding LastUpdated, Mode=TwoWay}" />
  </telerik:RadGridView.Columns>

</telerik:RadGridView>
```

The **GridViewComboBoxColumn** is used for lookups and requires more attention. In this example, **DataMemberBinding** points to the property in the "Products" table that is being updated after the edit, **SelectedValueMemberPath** is the value in the lookup collection and **DisplayMemberPath** is the property that will display in the combo drop down.

```
<telerik:RadGridView x:Name="gvMain"
    Margin="10" AutoGenerateColumns="False"
    ItemsSource="{StaticResource Products}">

  <telerik:RadGridView.Columns>
    <telerik:GridViewComboBoxColumn
        Header="Category"
        SelectedValueMemberPath="ID"
        DisplayMemberPath="CategoryName"
        DataMemberBinding="{Binding CategoryID, Mode=TwoWay}"
        ItemsSource="{StaticResource Categories}"
      />
    . . .
  </telerik:RadGridView.Columns>

</telerik:RadGridView>
```

Sometimes its easier to see the relationship in diagram form. A cut down representation of the data shows the RadGridView ItemsSource (Categories) and the ItemsSource for the GridViewComboBoxColumn (Products) and how each of the properties relate to the data.



ItemsSource for RadGridView

CategoryID	ProductName
1	Beginner Kayak
1	Adventurer Kayak
1	Kodiak Ocean-Going

Properties

DataMemberBinding

DisplayMemberPath

SelectedValuePath

ItemsSource for GridViewComboBoxColumn

ID	CategoryName
1	Kayaks
2	Dinghies
3	Canoes

From the Forums...

Question: I want to put a HyperlinkButton in my GridView but I want to pass in a query string to the NavigateUri from my DataSource collection. Is this even possible?

Answer: We introduced two new columns: GridViewHyperlinkColumn and GridViewDynamicHyperlinkColumn. Here is an example for both columns:

```
...
<telerik:GridViewHyperlinkColumn DataMemberBinding="{Binding Url}"
ContentBinding="{Binding CompanyName}" />
...

<telerik:GridViewDynamicHyperlinkColumn DataFormatString="Send mail to: {0}"
DataMemberBinding="{Binding CompanyName}"
NavigateUrlMemberPaths="ContactName, CompanyName"
NavigateUrlFormatString="mailto:{0}@{1}.com"/>

or

<telerik:GridViewDynamicHyperlinkColumn DataMemberBinding="{Binding MyProperty1}"
TargetName="_blank"
NavigateUrlMemberPaths="MyProperty2, MyProperty3"
NavigateUrlFormatString="http://www.myurl.com/page.aspx?q1={0}&q2={1}"/>
```

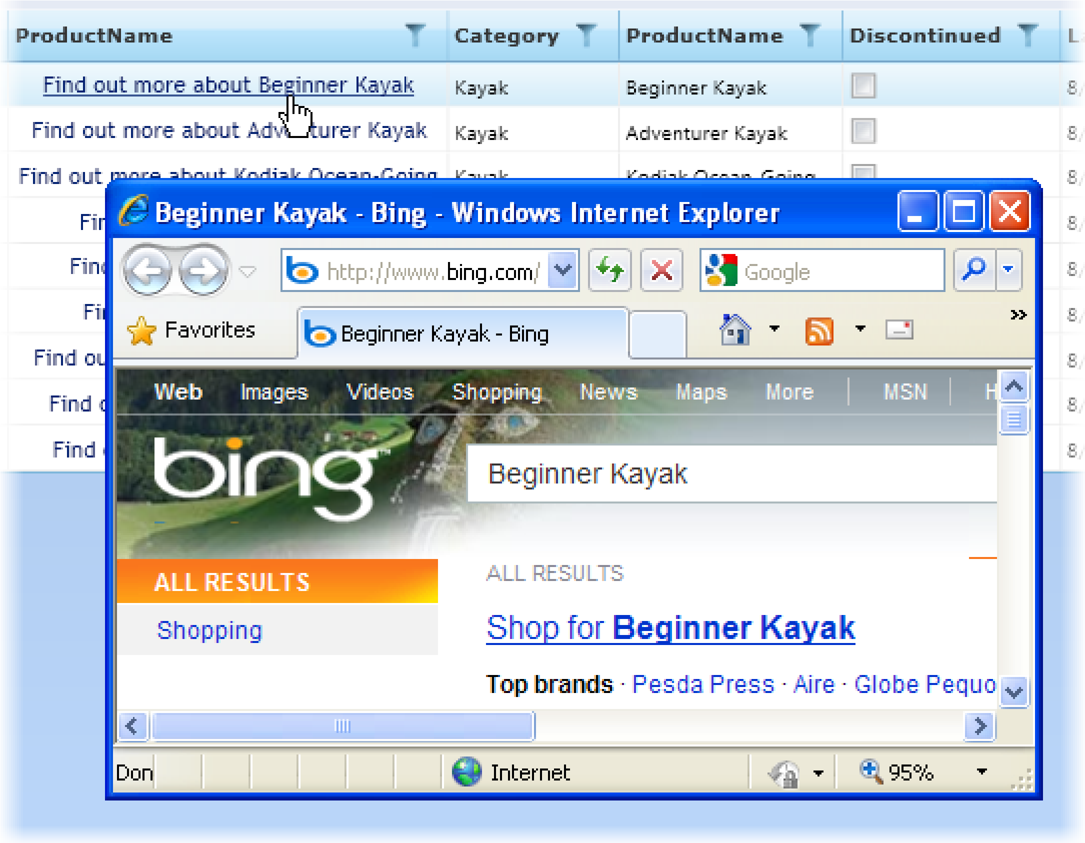
The **GridViewHyperlinkColumn** is used to define simple links where no substitutions to values in the URL need to happen. Bind **DataMemberBinding** to the column holding the URL and the **ContentBinding** to the column holding the text you want to display.

```
<telerik:GridViewHyperlinkColumn
    DataMemberBinding="{Binding InfoLink}"
    ContentBinding="{Binding ProductName}"
  />
```

The **GridViewDynamicHyperlinkColumn** is quite useful because you can merge your database data into a format string to produce searches on more than one column's worth of data. Use the **DataMemberBinding** and **DataFormatString** to build what the user sees in the column. The DataMemberBinding replaces the DataFormatString arguments, i.e. "{0}", "{1}", etc. The **NavigateUrlFormatString** and **NavigateUrlMemberPaths** produce the URL that will show when the user clicks the link.

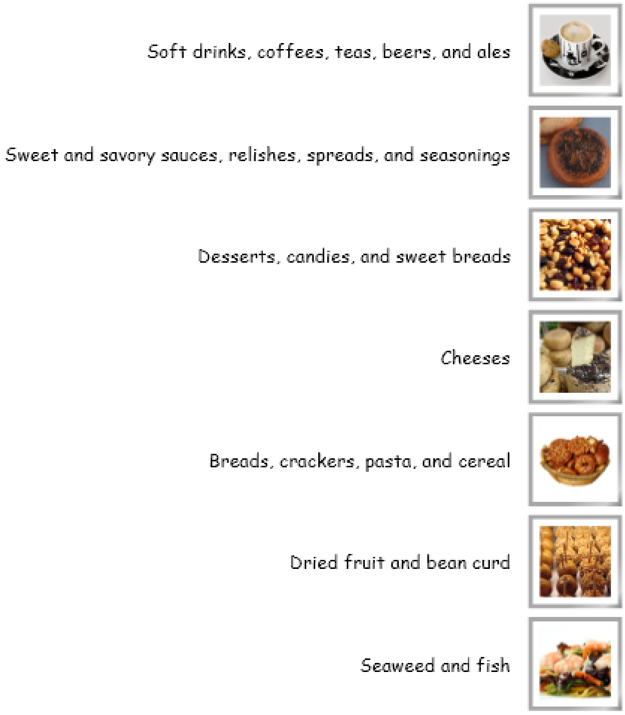
```
<telerik:GridViewDynamicHyperlinkColumn
    DataMemberBinding="{Binding ProductName}"
    DataFormatString="Find out more about {0}"
    NavigateUrlFormatString="http://www.bing.com/search?q={0}"
    NavigateUrlMemberPaths="ProductName"
    TargetName="_blank" />
```

Here is the dynamic hyperlink running in the browser.



18.4.4 Grid View Elements Visibility

Usually, every user has specific requirements for the outlook of the RadGridView and a necessity of changing some visual elements comes appears. Therefore, the RadGrodView enables you to hide show headers, footers, grid lines, row indicators and all the other spreadsheet-like trappings of a standard grid. Erasing these visual cues may be challenging unless you know where they are. Usually, those properties are not quite obvious, i.e. not always the property names end in "Visibility". The grid below has most of the visual details removed.



Soft drinks, coffees, teas, beers, and ales

Sweet and savory sauces, relishes, spreads, and seasonings

Desserts, candies, and sweet breads

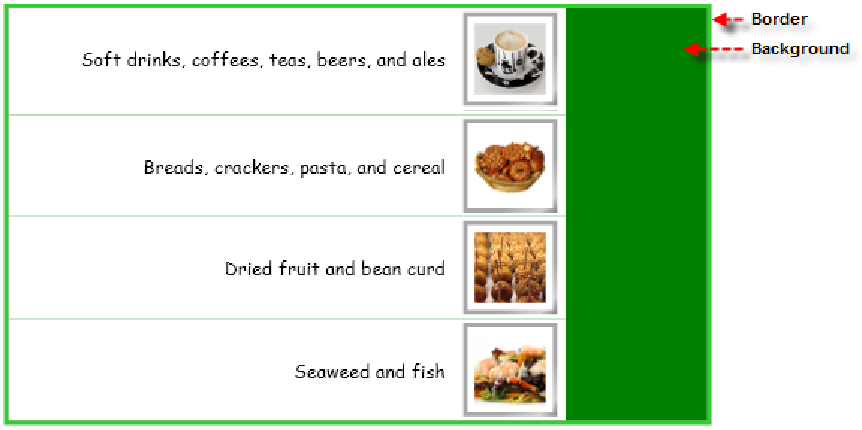
Cheeses

Breads, crackers, pasta, and cereal

Dried fruit and bean curd

Seaweed and fish

Let us turn a few of these settings back on so you can see how each property impacts the visual makeup of the grid. The **Background** and **BorderBrush** properties can be assigned "Transparent" in order to be removed. The screenshot below shows a lime green BorderBrush while the BorderThickness is set to 3 pixels and the background is green.



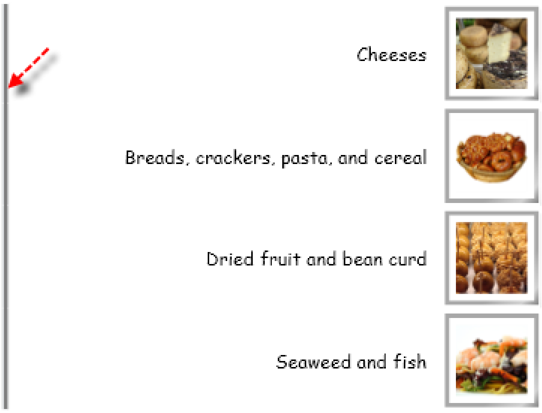
Soft drinks, coffees, teas, beers, and ales

Breads, crackers, pasta, and cereal

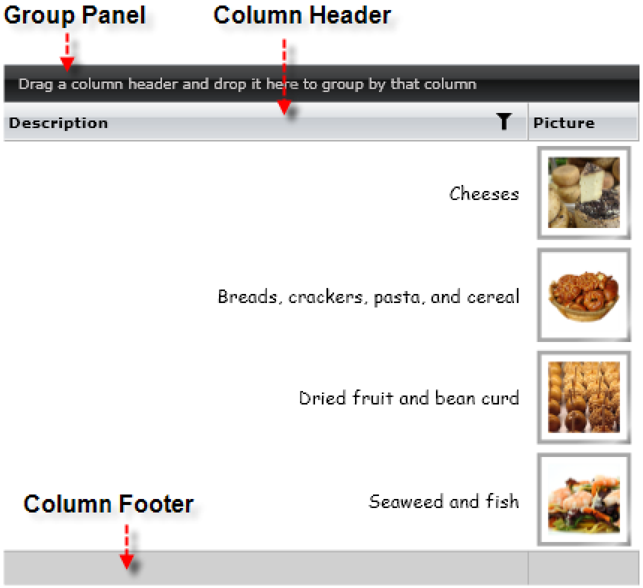
Dried fruit and bean curd

Seaweed and fish

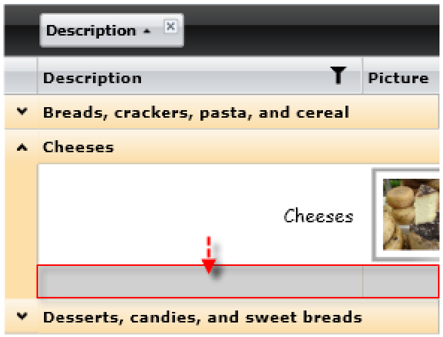
RadGridView provides a **CanUserFreezeColumns** property that enables you to freeze columns. When being set to "True", it results in some side-effect where visual artifacts are displayed to the left of the row.


Cheeses
Breads, crackers, pasta, and cereal
Dried fruit and bean curd
Seaweed and fish

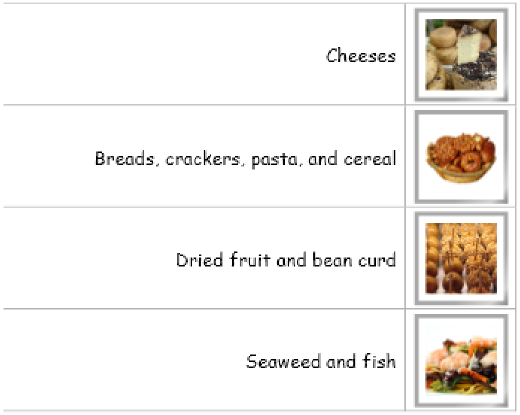
The **ShowGroupPanel, ShowColumnHeaders** and **ShowColumnFooters** properties toggle visibility for the group panel element at the top of the grid, the column headers just below the group header and above the data rows and the column footer found below the data rows.


Group Panel
Column Header
Drag a column header and drop it here to group by that column
Description
Picture
Cheeses
Breads, crackers, pasta, and cereal
Dried fruit and bean curd
Column Footer
Seaweed and fish

When grouping is applied, **ShowGroupFooters** toggles visibility for the element below a group of data rows.


Description
Description
Picture
Breads, crackers, pasta, and cereal
Cheeses
Cheeses
Desserts, candies, and sweet breads

GridLinesVisibility can be **Vertical**, **Horizontal**, **Both** or **None** (to hide grid lines altogether). The screenshot below shows the "Both" setting.



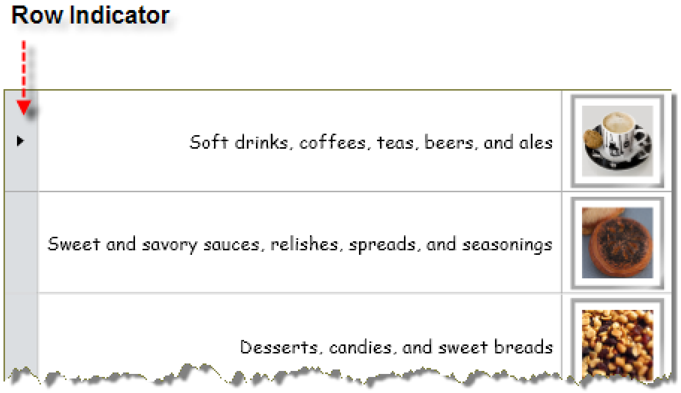
Cheeses

Breads, crackers, pasta, and cereal

Dried fruit and bean curd

Seaweed and fish

The row indicator visually flags the current row. **RowIndicatorVisibility** can be **Visible** or **Collapsed**.

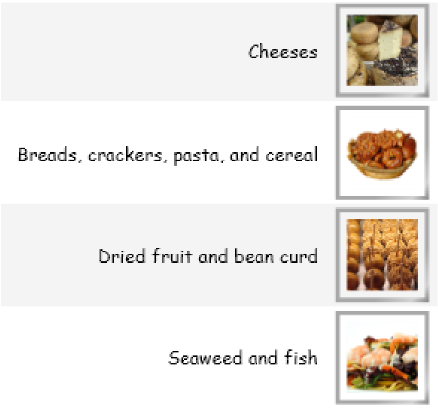


Soft drinks, coffees, teas, beers, and ales

Sweet and savory sauces, relishes, spreads, and seasonings

Desserts, candies, and sweet breads

Alternating row styles are the traditional technique for making it easier to visually differentiate between rows of complex data. They produce that "zebra" effect you see in the screenshot below. The BaseItemsControl **AlternationCount** is zero by default, so you don't need to explicitly turn this off.



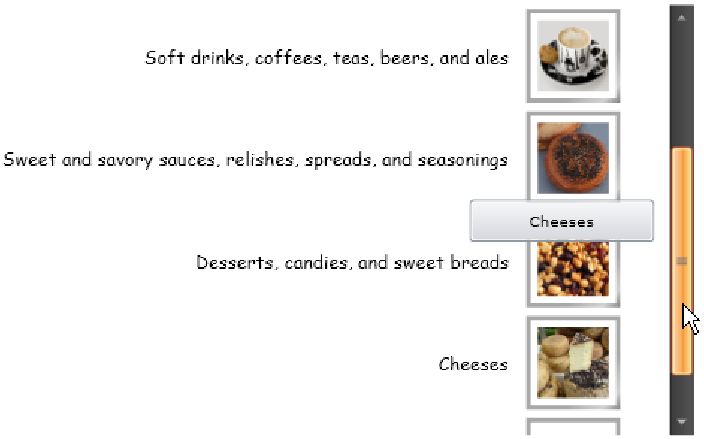
Cheeses

Breads, crackers, pasta, and cereal

Dried fruit and bean curd

Seaweed and fish

If the grid is placed in an area too small for the grid, scroll bars will display. By default, the scrolling behavior defers the actual scrolling of the grid view contents and instead displays a hint with the data for the current row. The screenshot below shows that the user has scrolled to the "Cheeses" row. The subtle side-effect is the pop-up hint itself. If you want to hide the hint, set the **ScrollMode** to **RealTime.** Otherwise, leave the property at **Deferred** (the default).


Soft drinks, coffees, teas, beers, and ales
Sweet and savory sauces, relishes, spreads, and seasonings
Cheeses
Desserts, candies, and sweet breads
Cheeses

To summarize what we've learned up to this point, here is the XAML markup that removes most of the common visual cues:

```
<telerik:RadGridView
    Background="Transparent" BorderBrush="Transparent"
    CanUserFreezeColumns="False" ShowColumnFooters="False"
    ShowColumnHeaders="False" ShowGroupFooters="False"
    ShowGroupPanel="False"
    RowIndicatorVisibility="Collapsed"
    GridLinesVisibility="None"
    ScrollMode="RealTime">
</telerik:RadGridView>
```

Notes

To configure alternate rows, set the **AlternationCount** property.

18.4.5 Accessing Elements in a Grid Row Template

When you require to customize the rows of the RadGridView - to load images for each one for example or any other changes to a row or its elements, you may handle the grid view **RowLoaded** event. Use the arguments **Row** property to find out what kind of a row this is (i.e. header, footer, "New row" element, detail row), and access the elements in the row.

Important note: make sure you add the **Telerik.Windows.Controls** namespace to the "Imports" (VB) or "using" (C#) section of code to get access to critical extension methods: **ChildrenOfType<T>()** and **ParentOfType<T>()**. ChildrenOfType<T>() is really the all-purpose "Swiss Army Knife" method that makes it easy to get at all the elements in the row. The sample below shows how you can check the Row type, then use the ChildrenOfType<T>() method to get a particular element in the row.

VB

```
Private Sub gvMain_RowLoaded( _
ByVal sender As Object, ByVal e As RowLoadedEventArgs)
  Dim isDetailRow As Boolean = _
Not(TypeOf e.Row Is GridViewNewRow OrElse _
TypeOf e.Row Is GridViewHeaderRow OrElse _
TypeOf e.Row Is GridViewFooterRow)
  If isDetailRow Then
    Dim image = e.Row.ChildrenOfType(Of Image)().FirstOrDefault()
    If image IsNot Nothing Then
      ' do something with the element
    End If
  End If
End Sub
```

C#

```
private void gvMain_RowLoaded(object sender, RowLoadedEventArgs e)
{
   bool isDetailRow = !(e.Row is GridViewNewRow ||
      e.Row is GridViewHeaderRow ||
      e.Row is GridViewFooterRow);
   if (isDetailRow)
   {
      var image = e.Row.ChildrenOfType<Image>().FirstOrDefault();
      if (image != null)
      {
       // do something with the element
      }
   }
}
```

If you need to include more information about a row, show hierarchal data or provide a rich user editing environment with RadGridView you can use the **Row Details**. **Row Details** is a **DataTemplate** defined on the grid- or row-level and it is used for displaying data without affecting the dimensions of the row and the cells within it. You can define Row Details template through the **RowDetailsTemplate** property of the grid. The display mode is specified by the **RowDetailsVisibilityMode** property and the available two options - **Visible, Collapsed** or **VisibleWhenSelected**. As the names imply the row details can be visible or collapsed for each row or only for the selected one.

You can define the row details template, like:

```
<telerikGrid:RadGridView x:Name="radGridView"
                    RowDetailsVisibilityMode="VisibleWhenSelected">
   <telerikGrid:RadGridView.RowDetailsTemplate>
      <DataTemplate>
         <StackPanel Orientation="Horizontal"
                    Margin="10,10,10,10">
            <TextBlock Text="City: " />
            <TextBlock Text="{Binding City}" />
         </StackPanel>
      </DataTemplate>
   </telerikGrid:RadGridView.RowDetailsTemplate>
   ...
</telerikGrid:RadGridView>
```

You can also change the appearance of the **Row Details** by using the **RowDetailsStyle** property of the RadGridView.

18.4.6 Sizing

The **ColumnWidth** property can be set for the RadGridView. Valid values are **Auto**(default), **Star (*)**, **SizeToHeader**, **SizeToCells** or a number of pixels. The property will set **Width** for auto-generated columns when they are first generated. Changes made to ColumnWidth after auto-generation will have no effect.

Notes

Note that if the column is set to "*" then column virtualization will be off (see the following section Performance > Virtualization for more information). Also for "*" columns to work correctly you should not place RadGridView in panels/controls that will measure its children with infinity: e.g. StackPanel, Grid panel with Column Width=Auto or Row with Height=Auto and ScrollViewer. The problem is that infinite space cannot be distributed. Also see the article at http://msdn.microsoft.com/en-us/library/cc645025(VS.95).aspx for more information about the Silverlight layout system.

```
<telerik:RadGridView x:Name="grid"  ColumnWidth="SizeToHeader">
  <telerik:RadGridView.Columns>
    <telerik:GridViewDataColumn Header="Company Name" />
    <telerik:GridViewDataColumn Header="Contact" />
    <telerik:GridViewDataColumn Header="Phone"  />
    <telerik:GridViewDataColumn Header="Ext"  />
  </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

The Width for individual columns can use a "*" notation to denote width percentages.

```
<telerik:RadGridView x:Name="grid" >
  <telerik:RadGridView.Columns>
    <telerik:GridViewDataColumn Header="Company Name" />
    <telerik:GridViewDataColumn Header="Contact" Width=".35*" />
    <telerik:GridViewDataColumn Header="Phone" Width=".5*" />
    <telerik:GridViewDataColumn Header="Ext" Width=".1*" />
  </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

18.4.7 Performance

18.4.7.1 Virtualization

Virtualization of rows and columns means that only visible rows and columns are created. Likewise, when a cell is marked with **IsVisible** = "False", the cell is not created and does not appear in the **Cells** collection. You could turn off the vertical virtualization by setting the **EnableRowVirtualization** property to false which will cause all rows to be created. However, it will also have a negative impact on the performance. You can also turn off the horizontal virtualization by means of **EnableColumnVirtualization** property. Again all elements - columns in this case will be created. It is the recommended best practice to leave virtualization in place. Furthermore, it is advisable to work with the underlying data instead of the visual elements.

> **Tip!**
>
> In general you should not use visual elements like GridViewRow. The problem with GridViewRow is that RadGridView uses virtualization and only a handful of rows are available at any time (just the visible ones).

The demo projects included with the Visual Studio installation contains a **GridView > Performance** example that loads a 500,000 rows of 100 columns each. The XAML declaration is very basic:

```
<telerik:RadGridView x:Name="RadGridView1" IsFilteringAllowed="False"
 ColumnWidth="100" ShowGroupPanel="False" IsReadOnly="True" />
```

The code-behind defines a collection of example objects populated with random data. The collection is simply assigned to the RadGridView **ItemsSource** property.

From the Forums...

Question: Performance can be slow when the RadGridView is placed inside a ScrollViewer. Why?

Answer: Some reports pointed to reduced performance of the RadGridView control when the grid is placed in a control that measures its children with infinity. Such controls are ScrollViewer, StackPanel (when vertical it measures with infinite height and when horizontal - with infinite width), and Grid panel with RowDefinition Height="Auto" or ColumnDefinition Width="Auto". When RadGridView (or any other grid) is measured with infinity virtualization is turned off which results in reduced performance. Modify your code so that RadGridView is placed in a container that will not measure it with infinity and the performance will be back to normal.

18.4.7.2 Paging

You can improve performance by simply limiting the number of records in view at any particular time. A good approach for achieving this is to add paging. What you need to do is to use a **QueryableCollectionView (or any IEnumerable)** to serve up the correct set of records, and a **RadDataPager** control on the page to navigate the data. In the XAML, make sure you have a xml namespace reference to the **Telerik.Windows.Controls.Data** assembly. Add a RadDataPager control and set its **Source** attribute to bind to the RadGridView element **Items** property.

```
<UserControl
    xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .>
  <Grid x:Name="LayoutRoot">
    <telerik:RadGridView x:Name="gridView" />
    <telerik:RadDataPager Source="{Binding Items, ElementName=gridView}" PageSize="10" />
  </Grid>
</UserControl>
```

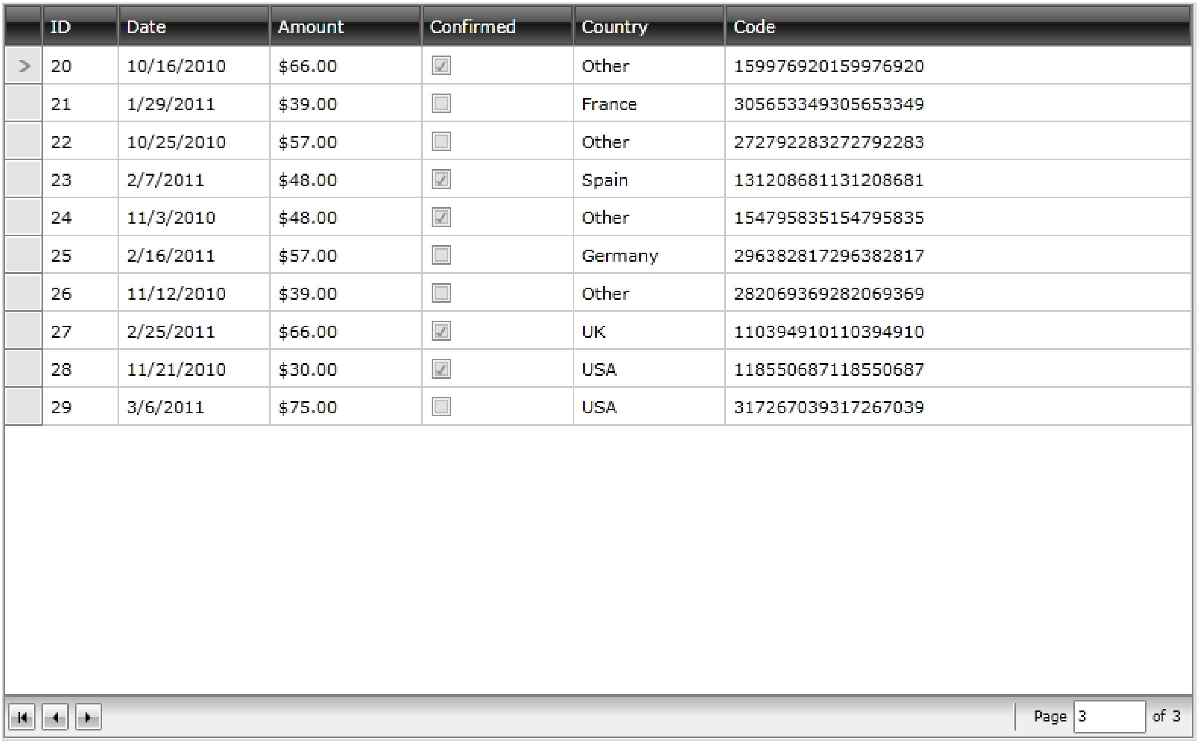
Instead of assigning data directly to the RadGridView ItemsSource, assign a **QueryableCollectionView**. The QueryableCollectionView constructor consumes a source collection of IEnumerable and its **PageSize** property can be set to the number of records that should appear at any one time.

The example uses a source collection called "BusinessObjects" (not shown) that holds random data and can output a given number of sample data rows using its "GetData()" method. You can use any IEnumerable in place of the "BusinessObjects" collection.

```
Dim businessObjects As New MyBusinessObjects()
Dim view As New QueryableCollectionView(businessObjects.GetData(100))
view.PageSize = 10
Me.gridView.ItemsSource = view
```

```
MyBusinessObjects businessObjects = new MyBusinessObjects();
QueryableCollectionView view =
   new QueryableCollectionView(businessObjects.GetData(100));
view.PageSize = 10;
this.gridView.ItemsSource = view;
```

When you run the application, the RadDataPager will appear, allowing you to move through the data.



	ID	Date	Amount	Confirmed	Country	Code
>	20	10/16/2010	$66.00	☑	Other	159976920159976920
	21	1/29/2011	$39.00	□	France	305653349305653349
	22	10/25/2010	$57.00	□	Other	272792283272792283
	23	2/7/2011	$48.00	☑	Spain	131208681131208681
	24	11/3/2010	$48.00	☑	Other	154795835154795835
	25	2/16/2011	$57.00	□	Germany	296382817296382817
	26	11/12/2010	$39.00	□	Other	282069369282069369
	27	2/25/2011	$66.00	☑	UK	110394910110394910
	28	11/21/2010	$30.00	☑	USA	118550687118550687
	29	3/6/2011	$75.00	□	USA	317267039317267039

18.4.8 Print and Export

Exporting and printing from RadGridView are supported by the **Export** method or the extension methods in static classes **ExportExtensions** and **PrintExtensions**, where both classes can be found in the **Telerik.Windows.Controls** namespace.

18.4.8.1 Exporting

To export the RadGridView's data use the **Export** method of the control. The method expects two parameters:

1. **Stream** - usually the file stream which you are exporting data to.

2. **GridViewExportOptions** or **GridViewCsvExportOptions** object - use it to set the following export options:

- Format - the possible formats are defined in the ExportFormat enumeration: Csv, ExcelML, Html or Text
- Encoding - the possible values are Encoding.Unicode, Encoding.UTF8, etc.
- ShowColumnHeaders - determines whether to export the column headers
- ShowColumnFooters - determines whether to export the column footers
- ShowGroupFooters - determines whether to export the group footers
- ColumnDelimiter - determines the string that will separate the cells of the exported data. Default is comma ",". Available in GridViewCsvExportOptions only.
- RowDelimiter - determines the string that will separate the rows of the exported data. Default is new line. Available in GridViewCsvExportOptions only.
- UseSystemCultureSeparator - if set, the RadGridView will use the system List Separator string, specified in Control Panel's Regional Options, to separate cells. This property overrides the ColumnDelimiter property. Available in GridViewCsvExportOptions only.

The following example shows how to show a save file dialog asking the user to save the file in excel format:

VB

```
Public Sub New()
 InitializeComponent()
 AddHandler btnExport.Click, AddressOf btnExport_Click
End Sub
Private Sub btnExport_Click(sender As Object, e As RoutedEventArgs)
 Dim extension As String = "xls"
 Dim dialog As New SaveFileDialog() With { _
  .DefaultExt = extension, _
  .Filter = [String].Format("{1} files (*.{0})|*.{0}|All files (*.*)|*.*", extension, "Excel"), _
  .FilterIndex = 1 _
 }
If dialog.ShowDialog() = True Then
  Using stream As Stream = dialog.OpenFile()
   gridViewExport.Export(stream, New GridViewExportOptions() With { _
    .Format = ExportFormat.Html, _
    .ShowColumnHeaders = True, _
    .ShowColumnFooters = True, _
    .ShowGroupFooters = False _
   })
  End Using
 End If
End Sub
```

```
public MainPage()
{
InitializeComponent();
btnExport.Click += new RoutedEventHandler(btnExport_Click);
}
void btnExport_Click(object sender, RoutedEventArgs e)
{
string extension = "xls";
SaveFileDialog dialog = new SaveFileDialog()
{
 DefaultExt = extension,
 Filter = String.Format("{1} files (*.{0})|*.{0}|All files (*.*)|*.*", extension, "Excel"),
 FilterIndex = 1
};
if (dialog.ShowDialog() == true)
{
 using (Stream stream = dialog.OpenFile())
 {
  gridViewExport.Export(stream,
   new GridViewExportOptions()
   {
    Format = ExportFormat.Html,
    ShowColumnHeaders = true,
    ShowColumnFooters = true,
    ShowGroupFooters = false,
   });
 }
}
}
```

In addition, you can use the following extension methods which allow RadGridView to export tab delimited text, comma delimited text, XML and HTML:

- **ToText()**: Returns a string with tab delimited text.
- **ToHtml()**: Returns an HTML string that can be saved as both "doc" and "xls" files.
- **ToCsv()**: Returns a string with comma delimited text.
- **ToExcelML()**: Returns an XML string.

The contents for all formats can all be written to disk using the same general technique shown below. This particular example writes an HTML file. As we are in a Silverlight environment, you will need to use the SaveFileDialog to access local file storage.

VB

```
Dim dialog As New SaveFileDialog() With {.Filter = "HTML files (*.html)|*.html|All files (*.*)|*.*"}
If dialog.ShowDialog() = True Then
 Using stream As Stream = dialog.OpenFile()
   Using writer As New StreamWriter(stream, Encoding.UTF8)
     writer.Write(gridView.ToHtml())
   End Using
   stream.Close()
 End Using
End If
```

C#

```
SaveFileDialog dialog = new SaveFileDialog()
{
   Filter = "HTML files (*.html)|*.html|All files (*.*)|*.*"
};
if (dialog.ShowDialog() == true)
{
 using (Stream stream = dialog.OpenFile())
 {
    using (StreamWriter writer =
       new StreamWriter(stream, Encoding.UTF8))
    {
       writer.Write(gridView.ToHtml());
    }
    stream.Close();
 }
}
```

18.4.8.2 Formatting

Formatting the elements of the grid is available through subscribing to the **ElementExporting** event. Each element is passed in the GridViewElementExportingEventArgs. You can find the usage of the element by comparing to the ExportElement enumeration (i.e. if its a Cell, Row, HeaderRow, etc) and the elements value. The example below simply bolds the header row text.

VB

```
Private Sub gridView_Exporting(ByVal sender As Object, ByVal e As GridViewExportEventArgs)
  If e.Element Is ExportElement.HeaderRow Then
    e.FontWeight = FontWeights.Bold
  End If
End Sub
```

C#

```
private void gridView_Exporting(object sender, GridViewExportEventArgs e)
{
   if (e.Element == ExportElement.HeaderRow)
   {
      e.FontWeight = FontWeights.Bold;
   }
}
```

18.4.8.3 Printing

Printing the contents of the grid view using the host browser is enabled by calling the **PrintToHtml()** method. Use the same **ElementExporting** event to format the HTML content while printing.

VB

```
Private Sub Print(ByVal sender As Object, ByVal e As Telerik.Windows.RadRoutedEventArgs)
  gridView.PrintToHtml()
End Sub
```

C#

```
private void Print(object sender, Telerik.Windows.RadRoutedEventArgs e)
{
   gridView.PrintToHtml();
}
```

18.5 Binding

18.5.1 .NET Objects

As mentioned previously in the Getting Started section of this chapter, you can easily bind a simple generic list if the user does not need to change the data. The typical grid is not used as a list, but still it allows the user to change its data. For this you must descend any collections from **ObservableCollection** and implement **INotifyPropertyChanged** for each object in a collection. No additional coding is needed for the grid view, only the bound object class definitions change. For example, the code below will run but not display a new category object in the grid.

VB

```
Private Sub AddCategoriesClick( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim categories As Categories = _
TryCast(gvMain.ItemsSource, Categories)
  categories.Add(New Category() With {_
.Description = "Life Rafts", .Products = New List(Of Product)()})
End Sub
```

C#

```
private void AddCategoriesClick(object sender, RoutedEventArgs e)
{
   Categories categories = gvMain.ItemsSource as Categories;
   categories.Add(new Category()
   {
      Description = "Life Rafts",
      Products = new List<Product>()
   });
}
```

To make the button click code above work, the generic List<> has to be changed to an **ObservableCollection<>**. Note that ObservableCollection is found in the **System.ComponentModel** namespace.

VB

```
' change this...
Public Class Categories
  Inherits ObservableCollection(Of Category)

' to this...
Public Class Categories
  Inherits List(Of Category)
```

C#

```
// change this...
public class Categories : List<Category>
// to this...
public class Categories : ObservableCollection<Category>
```

You also need to change the Products property from a List<> to ObservableCollection<>.

Then implement the **INotifyPropertyChanged** interface in each of the objects and sub-objects in the collection. Implementing the interface simply surfaces the PropertyChanged event. Trigger PropertyChanged after you assign each new property value. The sample below shows the Product class with a INotifyPropertyChanged interface implementation. Use this same pattern to implement the interface in the Category class.

VB

```
Public Class Product
  Implements INotifyPropertyChanged
  Public description_Renamed As String
  Public Property Description() As String
    Get
      Return description_Renamed
    End Get
    Set(ByVal value As String)
      description_Renamed = value
      OnPropertyChanged("Description")
    End Set
  End Property

  Public Sub OnPropertyChanged(ByVal propertyName As String)
    RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
  End Sub

  Public Event PropertyChanged As PropertyChangedEventHandler
End Class
```

```
public class Product : INotifyPropertyChanged
{
   public string description;
   public string Description
   {
      get { return description; }
      set
      {
         description = value;
         OnPropertyChanged("Description");
      }
   }

   public void OnPropertyChanged(string propertyName)
   {
      if (PropertyChanged != null)
      {
         PropertyChanged(this,
            new PropertyChangedEventArgs(propertyName));
      }
   }

   public event PropertyChangedEventHandler PropertyChanged;
}
```

Gotcha!

The grid may still refresh at times even if all the plumbing described here is not done completely. For example, if you neglect to change the Products property from List<> to ObservableCollection, you would see a refresh of the Products when a new category is added. This could cause hard-to-debug behavior as the project becomes more complex.

Property Paths

How can you flatten out a nested object hierarchy? For example, if our Category object has a Buyer object that in turn has properties for "First", "Last" and "Home Phone", how do we display this on one line? The Property Paths feature allows you to drill down and include properties from sub-objects. The example below shows Category.Description and Category.Buyer.FirstName, both in the same row.

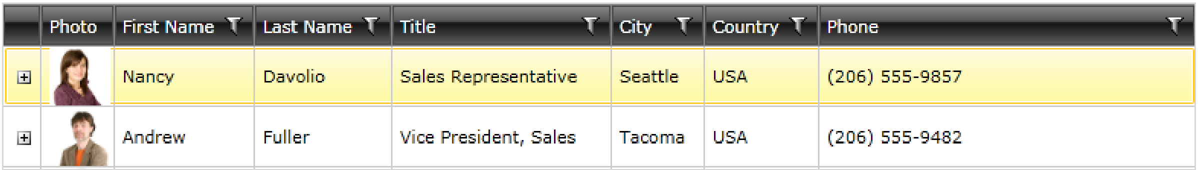


	Photo	First Name	Last Name	Title	City	Country	Phone
⊞		Nancy	Davolio	Sales Representative	Seattle	USA	(206) 555-9857
⊞		Andrew	Fuller	Vice President, Sales	Tacoma	USA	(206) 555-9482

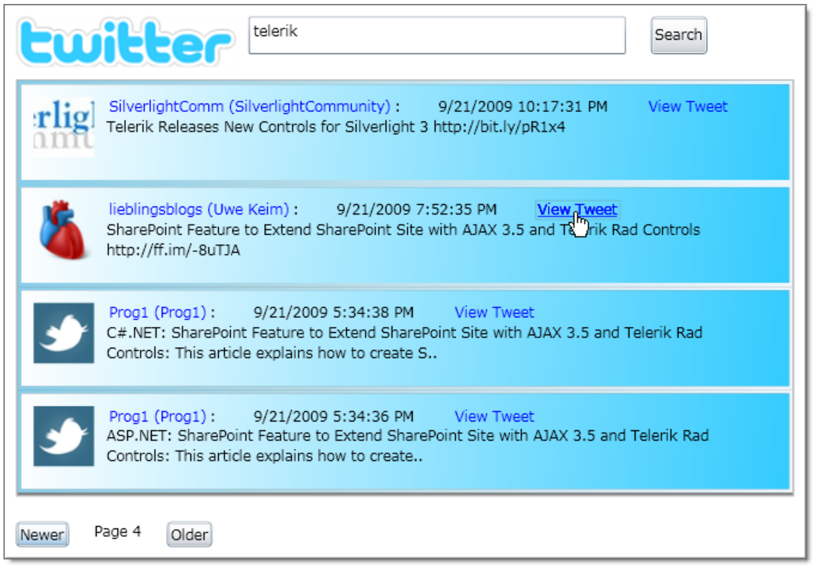
To display sub-property values, set the **AutoGenerateColumns** property to "False", add **GridViewDataColumn** instances to the **Columns** collection and assign a new **Binding** (using the property path) to the **DataMemberBinding** property of the column. Use the dot notation of the full path when creating the Binding.

```
gvMain.AutoGenerateColumns = False
gvMain.Columns.Add(New GridViewDataColumn() With {. _
DataMemberBinding = New Binding("Description")})
gvMain.Columns.Add(New GridViewDataColumn() With {. _
DataMemberBinding = New Binding("Buyer.FirstName")})
gvMain.Columns.Add(New GridViewDataColumn() With {. _
DataMemberBinding = New Binding("Buyer.LastName")})
gvMain.Columns.Add(New GridViewDataColumn() With {. _
DataMemberBinding = New Binding("Buyer.HomePhone")}
```

```
gvMain.AutoGenerateColumns = false;
gvMain.Columns.Add(new GridViewDataColumn()
{
   DataMemberBinding = new Binding("Description")
});
gvMain.Columns.Add(new GridViewDataColumn()
{
   DataMemberBinding = new Binding("Buyer.FirstName")
});
gvMain.Columns.Add(new GridViewDataColumn()
{
   DataMemberBinding = new Binding("Buyer.LastName")
});
gvMain.Columns.Add(new GridViewDataColumn()
{
   DataMemberBinding = new Binding("Buyer.HomePhone")
}
```

18.5.2 REST

Twitter exposes a REST service that searches for a string and returns an XML document. The results presented in the RadGridView look like the screenshot below.



SilverlightComm (SilverlightCommunity) : 9/21/2009 10:17:31 PM
Telerik Releases New Controls for Silverlight 3 http://bit.ly/pR1x4

lieblingsblogs (Uwe Keim) : 9/21/2009 7:52:35 PM
SharePoint Feature to Extend SharePoint Site with AJAX 3.5 and Telerik Rad Controls http://ff.im/-8uTJA

Prog1 (Prog1) : 9/21/2009 5:34:38 PM
C#.NET: SharePoint Feature to Extend SharePoint Site with AJAX 3.5 and Telerik Rad Controls: This article explains how to create S..

Prog1 (Prog1) : 9/21/2009 5:34:36 PM
ASP.NET: SharePoint Feature to Extend SharePoint Site with AJAX 3.5 and Telerik Rad Controls: This article explains how to create..

This particular example covers a wide range of techniques, including:

- How to use WebClient to return data from a REST service.
- How to use templates to freely format a row in the grid view.
- How to handle the grid view RowLoaded event.
- How to programmatically access elements in the template from the RowLoaded event.
- How to download and display images from an external site.
- How to work with image related issues such as image type and security concerns.
- Use LINQ to parse an XML document.

18.5.2.1 Project Setup

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.GridView**

c) **Telerik.Windows.Data**

d) **Telerik.Windows.Controls.Input.dll**

18.5.2.2 XAML Editing

1) Add XML namespace references for **Telerik.Windows.Controls** and **Telerik.Windows.Controls.GridView** in the **Telerik.Windows.Controls.GridView** assembly and to the **Telerik.Windows.Controls** namespace in the **Telerik.Windows.Controls** assembly. Also add a Loaded event handler for the UserControl.

```
<UserControl
xmlns:telerik=
"http://schemas.telerik.com/2008/xaml/presentation"
Loaded="UserControl_Loaded">
```

2) Add a UserControl.Resources element and a Grid inside the UserControl tags.

This step will replace the existing "LayoutRoot" Grid element. In following steps we will replace the comments with working XAML. The grid defines three rows of that will contain the search controls at the top, the grid view in the middle and the paging controls of the bottom.

```
<UserControl.Resources>
 <!--brushes-->
 <!--styles-->
 <!--custom row template-->
</UserControl.Resources>

<Grid x:Name="LayoutRoot">
   <Grid.RowDefinitions>
      <RowDefinition Height="50" />
      <RowDefinition />
      <RowDefinition Height="40" />
   </Grid.RowDefinitions>
   <!--search controls row-->
   <!--search results-->
   <!--pager-->
</Grid>
```

3) Replace the "<!--brushes-->" comment with the XAML below. *These will color the background and border to harmonize with the Twitter logo color.*

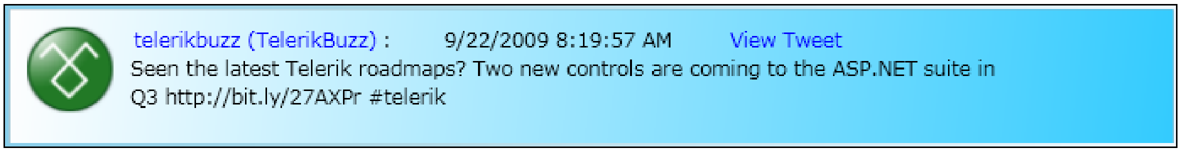
```
<!--brushes-->
<LinearGradientBrush x:Key="RowBackgroundBrush">
   <GradientStop Color="White" Offset="0" />
   <GradientStop Color="#FF33CCFF" Offset="1" />
</LinearGradientBrush>

<LinearGradientBrush x:Key="RowBorderBrush">
   <GradientStop Color="SkyBlue" Offset="0" />
   <GradientStop Color="LightBlue" Offset=".3" />
   <GradientStop Color="AliceBlue" Offset="1" />
</LinearGradientBrush>
```

4) Replace the "<!--custom row template-->" comment with the XAML below.

```
<!--custom row template-->
<ControlTemplate x:Key="MyCustomRowTemplate"
    TargetType="telerik:GridViewRow">
  <Border Background="{StaticResource RowBackgroundBrush}"
      BorderBrush="{StaticResource RowBorderBrush}"
      BorderThickness="3">
    <StackPanel Margin="10" VerticalAlignment="Center"
        Orientation="Horizontal"
        HorizontalAlignment="Left">
      <Image VerticalAlignment="Top" />
      <StackPanel Orientation="Vertical"
          VerticalAlignment="Center"
          Margin="10,0,0,0">
        <StackPanel Orientation="Horizontal"
            VerticalAlignment="Center">
          <HyperlinkButton
              Content="{Binding Author.Name}"
              Foreground="Blue"
              TargetName="_blank"
              NavigateUri="{Binding Author.Url}" />
          <TextBlock Text=":" />
          <StackPanel Orientation="Horizontal"
              HorizontalAlignment="Right"
              VerticalAlignment="Center">
            <TextBlock Margin="30,0,0,0"
                Text="{Binding Published}" />
            <HyperlinkButton Margin="30,0,0,0"
                Content="View Tweet"
                Foreground="Blue"
                TargetName="_blank"
                NavigateUri="{Binding Url}" />
          </StackPanel>
        </StackPanel>
        <TextBlock Width="500" Height="40"
            Text="{Binding Title}"
            TextWrapping="Wrap" />
      </StackPanel>
    </StackPanel>
  </Border>
</ControlTemplate>
```

Notes

The XAML creates the layout for each GridViewRow. The screenshot below is a sample of a single row. The outer-most Border sets the border and the background color of the row as a whole. Inside that, a StackPanel arranges groups of elements from left to right. The left-most element is an Image that contains the "Avatar" or image representing each member making a Twitter post (or "Tweet"). To the right of the image, at the top is a HyperlinkButton that points to the author of the tweet with a link to their page. Then, moving to the right is the publish date and time, followed by a "View Tweet" HyperlinkButton that links to a URL for that specific post. Underneath all of this is the content of the tweet displayed in a TextBlock.



telerikbuzz (TelerikBuzz) : 9/22/2009 8:19:57 AM View Tweet

Seen the latest Telerik roadmaps? Two new controls are coming to the ASP.NET suite in Q3 http://bit.ly/27AXPr #telerik

5) Replace the "<!--styles-->" comment with the XAML below.

Take a moment to review the style called "GridViewStyle" that will be applied to the RadGridView. The XAML here is mainly concerned with removing the lines, headers and other style details that might get in the way of the custom row style we apply later.

Also take a careful look at "RowStyle". This style points to a ControlTemplate that we shall add later. The ControlTemplate defines the arrangement of elements for each grid view row.

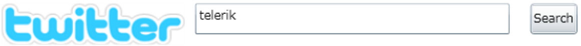
```
<!--styles-->
<Style x:Key="TextBlockStyle" TargetType="TextBlock">
  <Setter Property="Margin" Value="10" />
</Style>
<Style x:Key="TextBoxStyle" TargetType="TextBox">
  <Setter Property="Margin" Value="10" />
</Style>
<Style x:Key="ButtonStyle" TargetType="Button">
  <Setter Property="Margin" Value="10" />
</Style>
<Style x:Key="GridViewStyle"
    TargetType="telerikControlsGridView:RadGridView">
  <Setter Property="Margin" Value="10" />
  <Setter Property="ShowGroupPanel" Value="False" />
  <Setter Property="ShowColumnHeaders" Value="False" />
  <Setter Property="ScrollMode" Value="RealTime" />
  <Setter Property="VerticalGridlinesVisibility"
      Value="Collapsed" />
  <Setter Property="VerticalGridlinesBrush"
      Value="Transparent" />
</Style>
<Style x:Key="StackPanelStyle" TargetType="StackPanel">
  <Setter Property="Orientation" Value="Horizontal" />
  <Setter Property="HorizontalAlignment" Value="Left" />
</Style>
<Style x:Key="RowStyle"
    TargetType="telerik:GridViewRow">
  <Setter Property="Template"
      Value="{StaticResource MyCustomRowTemplate}" />
</Style>
```

6) Replace the <!--search controls row--> comment with the XAML below.

```
<!--search controls row-->
<StackPanel Grid.Row="0"
      Style="{StaticResource StackPanelStyle}">
   <Image x:Name="imageTwitterLogo" Stretch="None"
         Margin="10, 10, 0, 0" />
   <TextBox x:Name="tbSearch"
         Style="{StaticResource TextBoxStyle}" Width="300" />
   <Button x:Name="btnSearch"
         Style="{StaticResource ButtonStyle}"
         Content="Search" Click="btnSearch_Click" />
</StackPanel>
```

Notes

This step defines the Twitter logo image, the search text box and button. The screenshot below shows the arrangement of controls at runtime.



7) Replace the <!--search results--> comment with the XAML below.

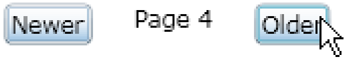
This step defines the RadGridView itself. The grid view is named "gvMain" so we can refer to it in code. The Style points back to the "GridViewStyle" we defined earlier and "RowStyle" points back to the GridViewRow style definition. The RowLoaded event handler is defined so we can populate the images as we go.

```
<!--search results-->
<telerik:RadGridView x:Name="gvMain"
      Grid.Row="1" Style="{StaticResource GridViewStyle}"
      RowStyle="{StaticResource RowStyle}"
      RowLoaded="gvMain_RowLoaded">
</telerik:RadGridView>
```

8) Replace the <!--pager--> comment with the XAML below.

Notes

This step defines a "pager", actually an arrangement of controls at the bottom of the page where two Buttons flank a TextBlock. The screenshot below shows the pager controls in action.



```
<!--pager-->
<StackPanel x:Name="Pager" Grid.Row="2"
      Style="{StaticResource StackPanelStyle}"
      Visibility="Collapsed">
   <Button x:Name="btnNewer"
         Style="{StaticResource ButtonStyle}"
         Content="Newer" Click="Button_Click" />
   <TextBlock x:Name="PageInfo"
         Style="{StaticResource TextBlockStyle}" />
   <Button x:Name="btnOlder"
         Style="{StaticResource ButtonStyle}"
         Content="Older" Click="Button_Click" />
</StackPanel>
```

18.5.2.3 Code Behind

1) From the Solution Explorer, right-click the project and select **Add > Class...** from the context menu. Name the class file "Twitter" and click the **Add** button.

2) Add the code below to the class file.

The code here defines "TwitterEntry" and "TwitterAuthor" classes that store information returned from the service. These two classes have no logic of their own.

```
Public Class TwitterEntry
  Private privateID As String
  Public Property ID() As String
    Get
      Return privateID
    End Get
    Set(ByVal value As String)
      privateID = value
    End Set
  End Property
  Private privatePublished As DateTime
  Public Property Published() As DateTime
    Get
      Return privatePublished
  End Get
  Set(ByVal value As DateTime)
    privatePublished = value
  End Set
End Property
Private privateUrl As String
Public Property Url() As String
  Get
    Return privateUrl
  End Get
  Set(ByVal value As String)
    privateUrl = value
  End Set
End Property
Private privateTitle As String
Public Property Title() As String
  Get
    Return privateTitle
  End Get
  Set(ByVal value As String)
    privateTitle = value
  End Set
End Property
Private privateContent As String
Public Property Content() As String
  Get
    Return privateContent
  End Get
  Set(ByVal value As String)
    privateContent = value
  End Set
End Property
Private privateUpdated As DateTime
Public Property Updated() As DateTime
  Get
    Return privateUpdated
  End Get
  Set(ByVal value As DateTime)
    privateUpdated = value
  End Set
End Property
Private privateImageUrl As String
Public Property ImageUrl() As String
  Get
    Return privateImageUrl
  End Get
  Set(ByVal value As String)
    privateImageUrl = value
  End Set
End Property
Private privateAuthor As TwitterAuthor
Public Property Author() As TwitterAuthor
  Get
    Return privateAuthor
  End Get
```

```
    Set(ByVal value As TwitterAuthor)
      privateAuthor = value
    End Set
  End Property
End Class

Public Class TwitterAuthor
  Private privateName As String
  Public Property Name() As String
    Get
      Return privateName
    End Get
    Set(ByVal value As String)
      privateName = value
    End Set
  End Property
  Private privateUrl As String
  Public Property Url() As String
    Get
      Return privateUrl
    End Get
    Set(ByVal value As String)
      privateUrl = value
    End Set
  End Property
End Class
```

C#

```
public class TwitterEntry
{
    public string ID { get; set; }
    public DateTime Published { get; set; }
    public string Url { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }
    public DateTime Updated { get; set; }
    public string ImageUrl { get; set; }
    public TwitterAuthor Author { get; set; }
}

public class TwitterAuthor
{
    public string Name { get; set; }
    public string Url { get; set; }
}
```

3) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **System.Windows.Media.Imaging** (supports BitmapImage)

b) **Telerik.Windows.Controls** (supports ChildrenOfType<T>())

c) **Telerik.Windows.Controls.GridView** (supports GridViewRow types, RowLoadedEventArgs)

4) Define a private integer member "currentPageIndex" to track the location of paging through multiple pages of data.

VB

```
Private currentPageIndex As Integer = 1
```

C#

```
private int currentPageIndex = 1;
```

5) Define a private method "DownloadTwitterPage" as shown in the code below.

It should take a search string as a parameter. Create a new WebClient instance, hook up a DownloadStringCompleted event handler and call the WebClient DownloadStringAsync() method. The string passed to DownloadStringAsync() is defined in the "urlFormat" constant below. The format of the string is Twitter service specific, so if you use a REST service from some other site you will need to research the appropriate format. Before calling DownloadStringAsync() you should check that the search string is not empty and that a WebClient request is not already in process. Show the button labeled "Newer" based on the position of the current page index.

```
Private Sub DownloadTwitterPage(ByVal searchString As String)
  Const pageSize As Integer = 4
  ' 0 = search string, 1 = page size, 2 = current page index
  Const urlFormat As String = _
"http://search.twitter.com/search.atom?q={0}&rpp={1}&page={2}"

  Dim webClient As New WebClient()
  AddHandler webClient.DownloadStringCompleted, _
AddressOf client_DownloadStringCompleted

  If (Not String.IsNullOrEmpty(searchString)) Then
    If (Not webClient.IsBusy) Then
      webClient.DownloadStringAsync( _
New Uri(String.Format(urlFormat, searchString, pageSize, currentPageIndex)))
    End If
  End If
  btnNewer.Visibility = _
If(currentPageIndex > 1, Visibility.Visible, Visibility.Collapsed)
End Sub
```

```
private void DownloadTwitterPage(string searchString)
{
   const int pageSize = 4;
   // 0 = search string, 1 = page size, 2 = current page index
   const string urlFormat =
      "http://search.twitter.com/search.atom?q={0}&rpp={1}&page={2}";

   WebClient webClient = new WebClient();
   webClient.DownloadStringCompleted += new DownloadStringCompletedEventHandler(
        client_DownloadStringCompleted);

   if (!String.IsNullOrEmpty(searchString))
   {
      if (!webClient.IsBusy)
         webClient.DownloadStringAsync(
            new Uri(String.Format(urlFormat, searchString, pageSize,
                  currentPageIndex)));
   }
   btnNewer.Visibility = currentPageIndex > 1 ?
      Visibility.Visible : Visibility.Collapsed;
}
```

6) Handle the DownloadStringCompleted event as shown in the code below.

The purpose of this handler is to parse the XML returned in the Result parameter and load new instances of the TwitterEntry object. The collection of TwitterEntry objects is assigned to the grid view ItemsSource property. This event handler also sets the pager visibility, grid view visibility and pager text.

VB

```
Private Sub client_DownloadStringCompleted( _
ByVal sender As Object, ByVal e As DownloadStringCompletedEventArgs)
  Pager.Visibility = Visibility.Visible
  gvMain.Visibility = Pager.Visibility
  PageInfo.Text = String.Format("Page {0}", currentPageIndex)
  Dim atomNamespace As XNamespace = "http://www.w3.org/2005/Atom"
  Dim xDocument As XDocument = XDocument.Parse(e.Result)
  Dim twitterData = _
    From item In xDocument.Descendants(atomNamespace + "entry") _
    Select New TwitterEntry
item.Descendants(atomNamespace + _
"link").Last().Attribute("href").Value, _
Author = New TwitterAuthor() With {
.Name = (CType(item.Element(atomNamespace + _
"author").FirstNode, XElement)).Value, _
.Url = (CType(item.Element(atomNamespace + _
"author").LastNode, XElement)).Value}
DateTime.Parse(item.Element(atomNamespace + _
"updated").Value), _
ImageUrl = _
item.Descendants(atomNamespace + _
"link").Last().Attribute("href").Value, Author
item.Element(atomNamespace + "content").Value, _
Updated = DateTime.Parse(item.Element(atomNamespace + _
"updated").Value), ImageUrl
item.Element(atomNamespace + "title").Value, _
Content = item.Element(atomNamespace + _
"content").Value, Updated
item.Element(atomNamespace + "link").Attribute("href").Value, _
Title = item.Element(atomNamespace + "title").Value, Content
DateTime.Parse(item.Element(atomNamespace + _
"published").Value), _
Url = item.Element(atomNamespace + _
"link").Attribute("href").Value, Title
item.Element(atomNamespace + _
"id").Value, _
Published = DateTime.Parse(item.Element(atomNamespace + _
"published").Value), Url
ID = item.Element(atomNamespace + "id").Value, Published
  gvMain.ItemsSource = twitterData
End Sub
```

C#

```
void client_DownloadStringCompleted(object sender,
    DownloadStringCompletedEventArgs e)
{
    gvMain.Visibility = Pager.Visibility = Visibility.Visible;
    PageInfo.Text = String.Format("Page {0}", currentPageIndex);

    XNamespace atomNamespace = "http://www.w3.org/2005/Atom";

    XDocument xDocument = XDocument.Parse(e.Result);

    var twitterData =
        from item in xDocument.Descendants(atomNamespace + "entry")
        select new TwitterEntry
        {
            ID = item.Element(atomNamespace + "id").Value,
            Published =
              DateTime.Parse(item.Element(atomNamespace + "published").Value),
            Url =
              item.Element(atomNamespace + "link").Attribute("href").Value,
            Title =
              item.Element(atomNamespace + "title").Value,
            Content =
              item.Element(atomNamespace + "content").Value,
            Updated =
              DateTime.Parse(item.Element(atomNamespace + "updated").Value),
            ImageUrl =
              item.Descendants(atomNamespace +
                "link").Last().Attribute("href").Value,
            Author = new TwitterAuthor()
            {
                Name = ((XElement)item.Element(atomNamespace +
                  "author").FirstNode).Value,
                Url = ((XElement)item.Element(atomNamespace +
                  "author").LastNode).Value
            }
        };
    gvMain.ItemsSource = twitterData;
}
```

7) Add a new method "LoadImage()" that will take a string parameter that defines the URL where the image is located, and an Image object that will be loaded with the new image.

Check that the URL points to a ".png" or ".jpg" file. These are the only supported image types in Silverlight at the time of this writing. Create a new WebClient instance, hook up a OpenReadCompleted event handler and call the OpenReadAsync() method. OpenReadAsync() should take a URI object that points to the image URL and pass a reference to the Image to be loaded.

VB

```
Private Sub LoadImage(ByVal imageUrl As String, ByVal image As Image)
  Dim isValidSilverlightImageType As Boolean = imageUrl.EndsWith(".png") OrElse imageUrl.EndsWith(".jpg'

  If (Not isValidSilverlightImageType) Then
    Return
  End If

  Dim webClient As New WebClient()
  AddHandler webClient.OpenReadCompleted, AddressOf webClient_OpenReadCompleted
  webClient.OpenReadAsync(New Uri(imageUrl), image)
End Sub
```

C#

```
private void LoadImage(string imageUrl, Image image)
{
   bool isValidSilverlightImageType =
      imageUrl.EndsWith(".png") || imageUrl.EndsWith(".jpg");

   if (!isValidSilverlightImageType)
      return;

   WebClient webClient = new WebClient();
   webClient.OpenReadCompleted +=
      new OpenReadCompletedEventHandler(webClient_OpenReadCompleted);
   webClient.OpenReadAsync(
      new Uri(imageUrl), image);
}
```

8) Handle the OpenReadCompleted event to retrieve the streamed image and assign it to an Image object.

Create a BitmapImage and use the SetSource method to assign the stream held in Result. Then assign the BitmapImage to the source of the Image. Remember that Image will be passed as the UserState argument when OpenReadAsync() is first called. To handle security restrictions where no ClientAccessPolicy.xml exists on the server where the image resides, trap for the TargetInvocationException. Inside the "Catch", verify that the InnerException is a SecurityException. The code below ignores the security exception, assuming that we're trying to access an image across domains but without the permissions granted by the presence of a ClientAccessPolicy.xml file. If there is any other kind of exception, re-throw the exception.

VB

```
Private Sub webClient_OpenReadCompleted( _
ByVal sender As Object, ByVal e As OpenReadCompletedEventArgs)
  Dim bitmap As New BitmapImage()
  Try
    bitmap.SetSource(e.Result)
    TryCast(e.UserState, Image).Source = bitmap
  Catch ex As System.Reflection.TargetInvocationException
    If Not(TypeOf ex.InnerException Is _
System.Security.SecurityException) Then
      MessageBox.Show(ex.Message)
      Throw
    End If
  End Try
End Sub
```

C#

```
void webClient_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
   BitmapImage bitmap = new BitmapImage();
   try
   {
      bitmap.SetSource(e.Result);
      (e.UserState as Image).Source = bitmap;
   }
   catch (System.Reflection.TargetInvocationException ex)
   {
      if (!(ex.InnerException is System.Security.SecurityException))
      { MessageBox.Show(ex.Message); throw; }
   }
}
```

9) Handle the grid view RowLoaded event as shown in the code below.

This is your opportunity to alter elements of the row programmatically. First check that this is not a header, footer or "new row" using the Row property of the event argument. Use the ChildrenOfType<T>() method to get a reference to the Image object in the grid row template. There is only one Image object in the template, so we can call FirstOrDefault() to get the instance. Use the DataElement property of the event argument to get the bound TwitterEntry object. Finally, call the private LoadImage() method you created earlier and pass the ImageUrl of the TwitterEntry and the instance of the Image object that you retrieved from the template.

VB

```
Private Sub gvMain_RowLoaded(ByVal sender As Object, ByVal e As RowLoadedEventArgs)
  Dim isDetailRow As Boolean = Not(TypeOf e.Row Is GridViewNewRow OrElse _
TypeOf e.Row Is GridViewHeaderRow OrElse TypeOf e.Row Is GridViewFooterRow)
  If isDetailRow Then
    Dim image = e.Row.ChildrenOfType(Of Image)().FirstOrDefault()
    If image IsNot Nothing Then
      Dim twitterEntry As TwitterEntry = TryCast(e.DataElement, TwitterEntry)
      LoadImage(twitterEntry.ImageUrl, image)
    End If
  End If
End Sub
```

C#

```
private void gvMain_RowLoaded(object sender, RowLoadedEventArgs e)
{
   bool isDetailRow = !(e.Row is GridViewNewRow ||
      e.Row is GridViewHeaderRow ||
      e.Row is GridViewFooterRow);
   if (isDetailRow)
   {
      var image = e.Row.ChildrenOfType<Image>().FirstOrDefault();
      if (image != null)
      {
         TwitterEntry twitterEntry = e.DataElement as TwitterEntry;
         LoadImage(twitterEntry.ImageUrl, image);
      }
   }
}
```

10)Handle the Button Click events for the "Newer" and "Older" buttons as well as the "Search" button.

The "Search" button click simply calls the private DownloadTwitterPage() method you wrote earlier and passes the text entered into the search text box. The "Button_Click" event handler is triggered by both the "Newer" and "Older" buttons. Depending on the identity of the button, the current page index is advanced or decreased. Then, like the search button, the DownloadTwitterPage() method is called, passing the search text.

VB

```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  If sender Is btnNewer Then
    If currentPageIndex >= 2 Then
      currentPageIndex -= 1
    End If
  ElseIf sender Is btnOlder Then
    currentPageIndex += 1
  End If

  DownloadTwitterPage(tbSearch.Text)
End Sub

Private Sub btnSearch_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  DownloadTwitterPage(tbSearch.Text)
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   if (sender == btnNewer)
   {
      if (currentPageIndex >= 2)
         currentPageIndex--;
   }
   else if (sender == btnOlder)
      currentPageIndex++;

   DownloadTwitterPage(tbSearch.Text);
}

private void btnSearch_Click(object sender, RoutedEventArgs e)
{
   DownloadTwitterPage(tbSearch.Text);
}
```

11)Handle the UserControl Loaded event. *Here we simply load the Twitter logo to the top of the page.*

VB

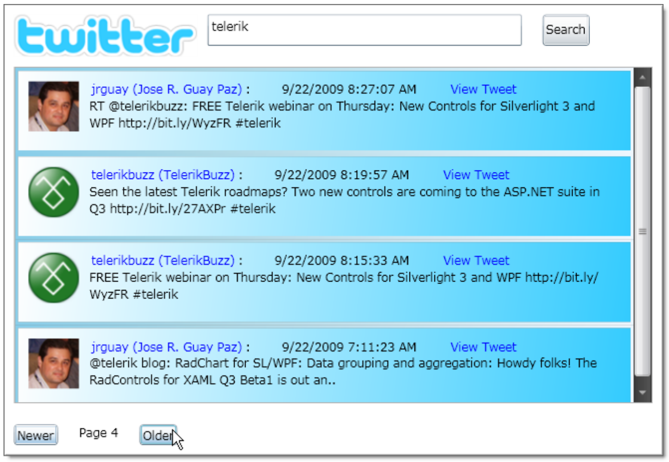
```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Const twitterLogoUrl As String = _
"http://search.twitter.com/images/search/twitter-logo-small.png"

  LoadImage(twitterLogoUrl, imageTwitterLogo)
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   const string twitterLogoUrl =
      "http://search.twitter.com/images/search/twitter-logo-small.png";

   LoadImage(twitterLogoUrl, imageTwitterLogo);
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



jrguay (Jose R. Guay Paz) : 9/22/2009 8:27:07 AM View Tweet
RT @telerikbuzz: FREE Telerik webinar on Thursday: New Controls for Silverlight 3 and WPF http://bit.ly/WyzFR #telerik

telerikbuzz (TelerikBuzz) : 9/22/2009 8:19:57 AM View Tweet
Seen the latest Telerik roadmaps? Two new controls are coming to the ASP.NET suite in Q3 http://bit.ly/27AXPr #telerik

telerikbuzz (TelerikBuzz) : 9/22/2009 8:15:33 AM View Tweet
FREE Telerik webinar on Thursday: New Controls for Silverlight 3 and WPF http://bit.ly/WyzFR #telerik

jrguay (Jose R. Guay Paz) : 9/22/2009 7:11:23 AM View Tweet
@telerik blog: RadChart for SL/WPF: Data grouping and aggregation: Howdy folks! The RadControls for XAML Q3 Beta1 is out an..

Test Application Features

1) Enter a search string and click the Search button. Depending on the number of hits for the data, you should see the grid fill up with new rows.
2) Each row should conform to the grid row template you defined.
3) Images should appear for the Twitter logo (assuming Twitter doesn't move the image) and next to each of the "Tweets". Some or even many rows may not show an image if they were not a supported image type or if there was no ClientAccessPolicy.xml on the server.
4) Click the link for the author to navigate to their Twitter page.

5) Click the "View Tweet" link to navigate to the page for that specific tweet.

6) Use the "Newer" and "Older" buttons to page through the available data.

Ideas for Extending This Example

- Display a default image when the image is not presented due to security issues or is not supported by Silverlight.
- Allow the user to navigate to the main Twitter page by clicking the logo. Wrap the logo image in a HyperlinkButton.
- Visit the wiki at http://apiwiki.twitter.com for additional Twitter REST services that can be called.
- Google for other REST services that could be adapted.

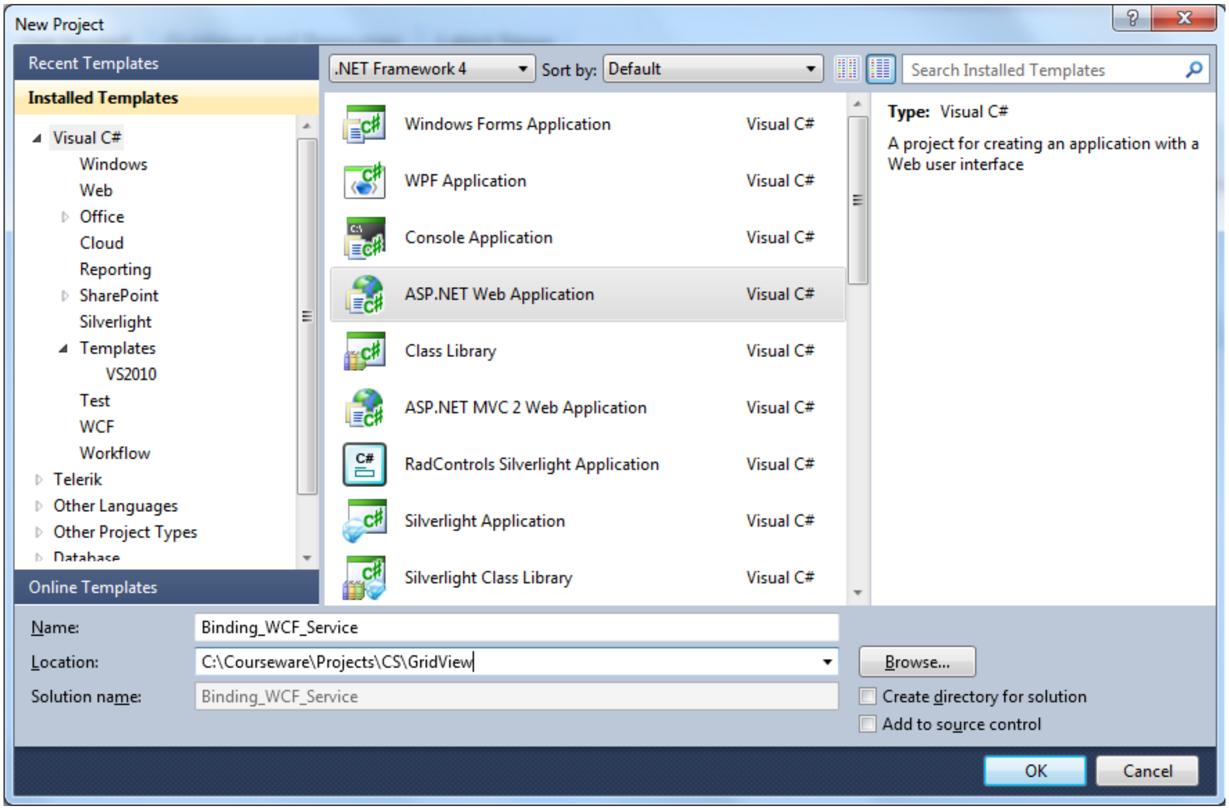
18.5.3 WCF

The walk through that follows demonstrates building a simple WCF service that supplies a list of customers and populates a RadGridView in a Silverlight client.

18.5.3.1 Building the WCF Service

1) Create a new ASP.NET Web Application project.

This application will host the WCF service.



New Project
Recent Templates
Installed Templates
Visual C#
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Templates
VS2010
Test
WCF
Workflow
Telerik
Other Languages
Other Project Types
Database
Online Templates
.NET Framework 4
Sort by: Default
Search Installed Templates
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
ASP.NET Web Application Visual C#
Class Library Visual C#
ASP.NET MVC 2 Web Application Visual C#
RadControls Silverlight Application Visual C#
Silverlight Application Visual C#
Silverlight Class Library Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating an application with a Web user interface

Name: Binding_WCF_Service

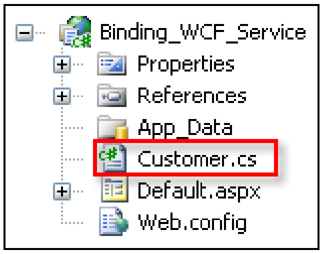
Location: C:\Courseware\Projects\CS\GridView

Solution name: Binding_WCF_Service

□ Create directory for solution

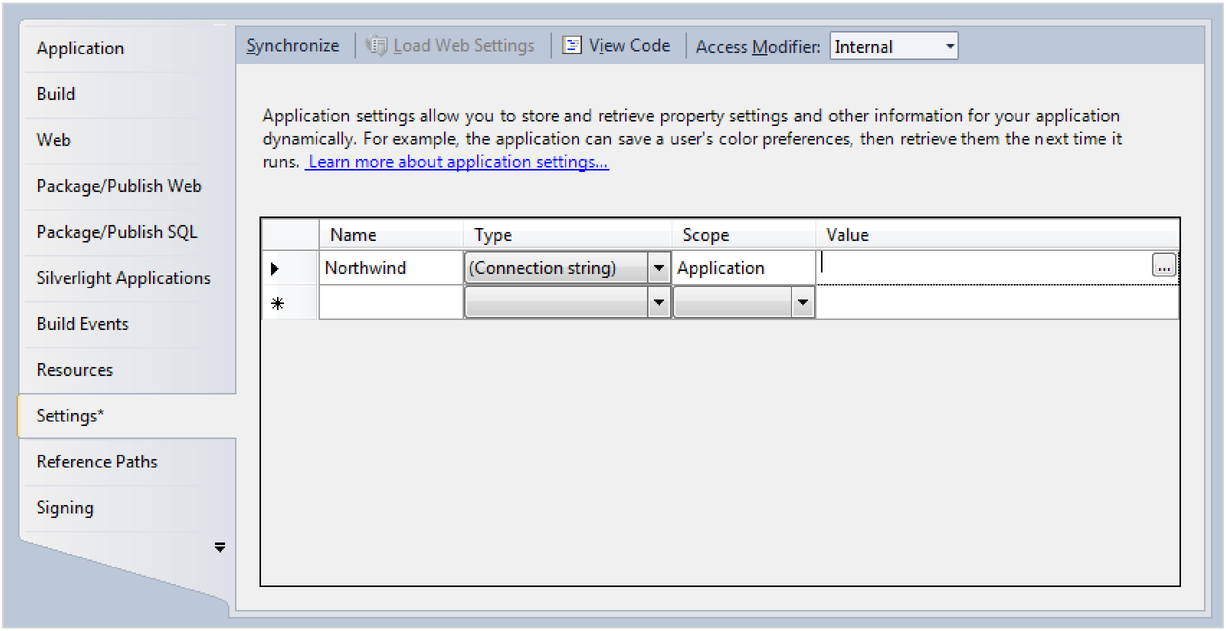
□ Add to source control

2) Navigate to the RadControls for Silverlight installation directory and find the Customer.cs class file in the "\Examples\GridView" directory (there is also a copy for your convenience located in the "\courseware\datasources" directory). Drag this file to the Solution Explorer and drop it in the root of the project.



3) In the Solution Explorer, double click the Properties node of the project, then click the Settings tab. You will see a message "This project does not contain a default settings file. Click here to create one". Click the link to create the settings file.

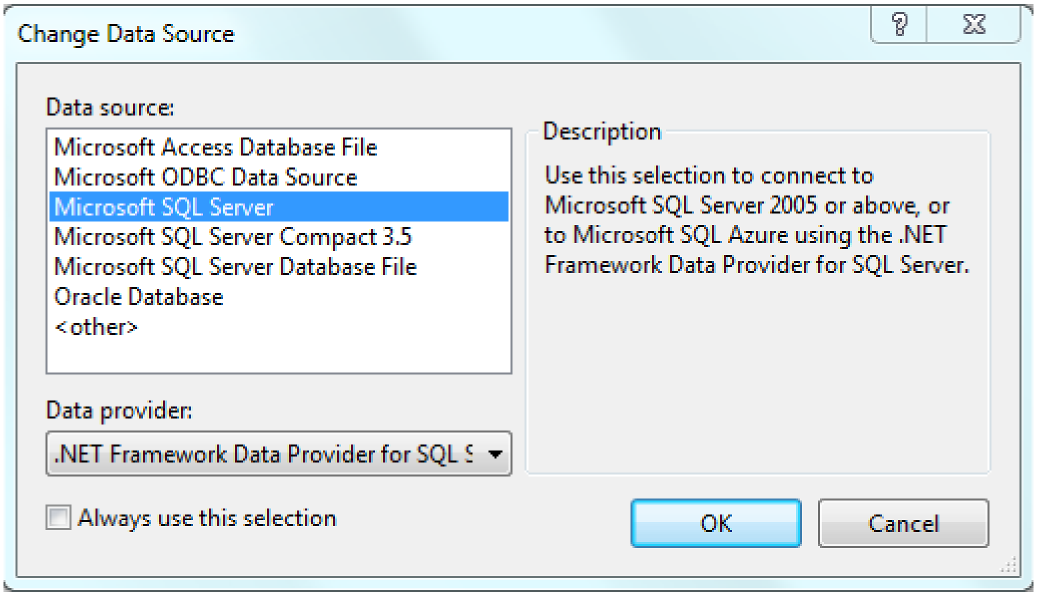
4) In the Settings, configure the first setting so that **Name** = "Northwind", **Type** = "(Connection string) and **Scope** = "Application". Click in the **Value** entry box, then click the ellipses to edit the connection. This will display the Connection Properties window.



Application settings allow you to store and retrieve property settings and other information for your application dynamically. For example, the application can save a user's color preferences, then retrieve them the next time it runs. Learn more about application settings...

	Name	Type	Scope	Value
▶	Northwind	(Connection string)	Application	
*				

5) In the Connection Properties window, click the **Change...** button to display the **Change Data Source** dialog. Select "Microsoft SQL Server Database File" from the list and click the **OK** button to close the dialog, returning you to the Connection Properties window.



Change Data Source

Data source:

Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Compact 3.5
Microsoft SQL Server Database File
Oracle Database
<other>

Data provider:

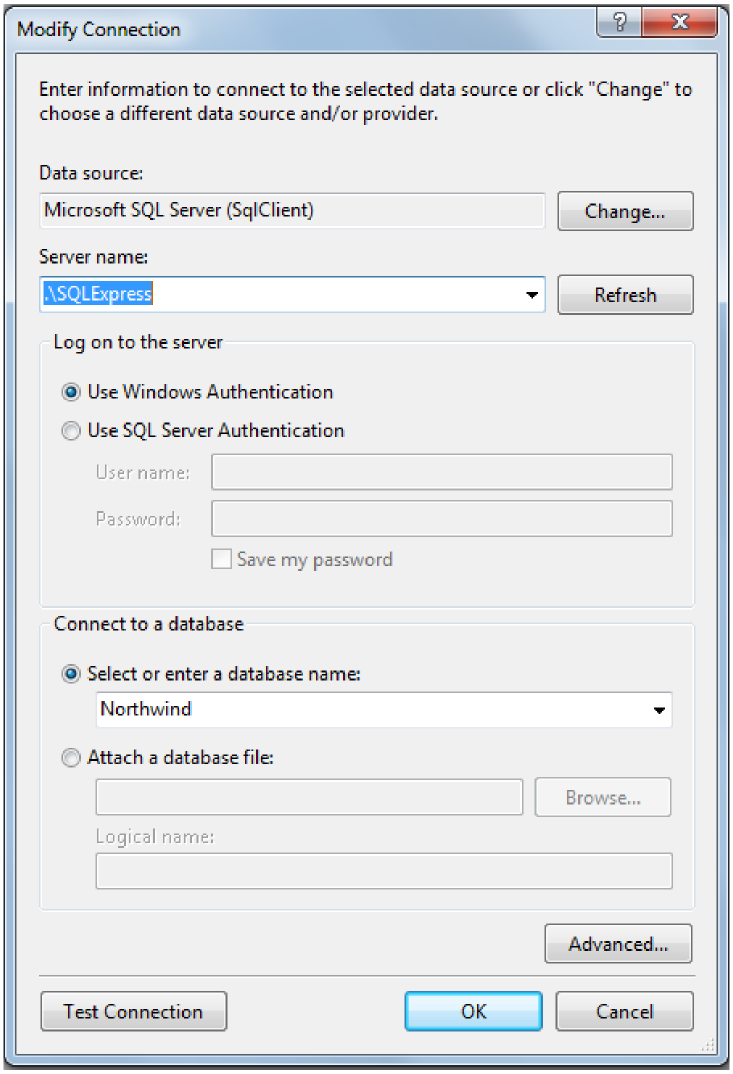
.NET Framework Data Provider for SQL S

Description

Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

□ Always use this selection

6) Click the Browse... button. Navigate to the "Northwind.mdf" file located in the RadControls for Silverlight installation directory under the "\Examples\DataSources" path (there is also a copy for your convenience located in the "\courseware\datasources" directory). Click the **OK** button to close the dialog and create the connection string.



Modify Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:

Microsoft SQL Server (SqlClient)

Server name:

.\SQLExpress

Log on to the server

● Use Windows Authentication

○ Use SQL Server Authentication

User name:

Password:

□ Save my password

Connect to a database

● Select or enter a database name:

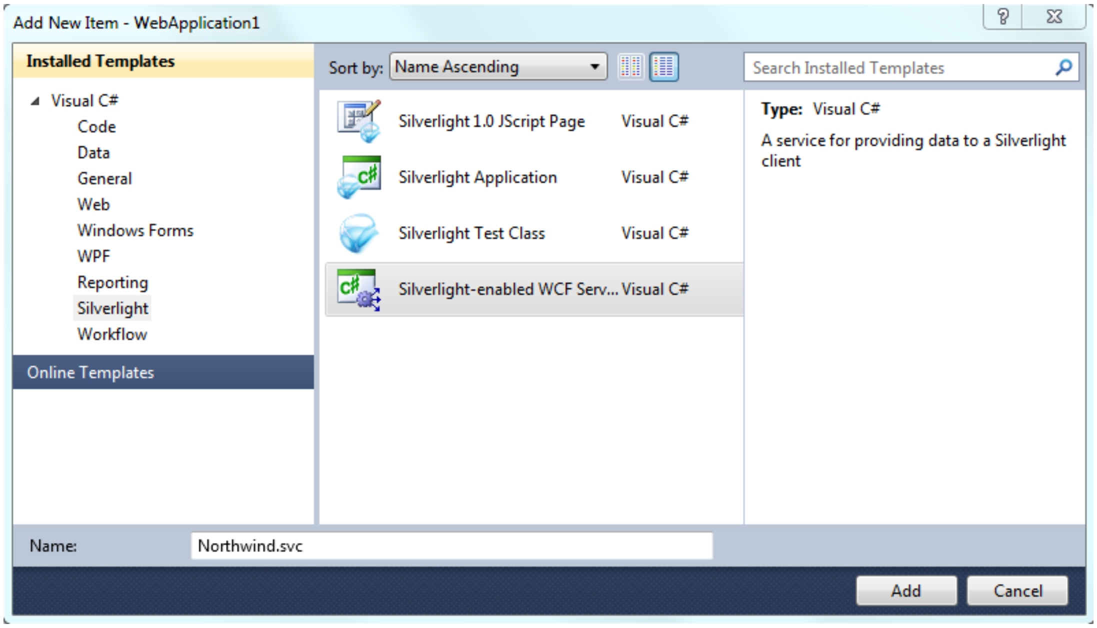
Northwind

○ Attach a database file:

Logical name:

7) In the Solution Explorer, right-click and select **Add > New Item...** from the context menu.

8) Select the "Silverlight-enabled WCF Service" template, name it "Northwind.svc" and click the **Add** button to create the service and close the dialog.



Add New Item - WebApplication1

Sort by: Name Ascending

Type: Visual C#

A service for providing data to a Silverlight client

Name: Northwind.svc

Installed Templates
Visual C#
Code
Data
General
Web
Windows Forms
WPF
Reporting
Silverlight
Workflow
Online Templates
Search Installed Templates
Silverlight 1.0 JScript Page Visual C#
Silverlight Application Visual C#
Silverlight Test Class Visual C#
Silverlight-enabled WCF Serv... Visual C#
Add
Cancel

9) Open the "Northwind.svc.cs" file for editing.

10)Verify that the following namespaces are added to the "Imports" (VB) or "using" (C#) clauses.

System.Collections.Generic
System.Configuration
System.Data
System.Data.SqlClient
System.ServiceModel
System.ServiceModel.Activation
Telerik.Windows.Examples (supports the Customer.cs class you copied to this project)

11)Replace the default "DoWork()" method that already exists in the service with the GetCustomers() method using the code below. **Important note:** Be sure to replace the string "<project name>" passed to connection strings, with the actual name of your project. So, for a project named "Binding_WCF_Service" and connection string named "Northwind", the connection string is named:

"Binding_WCF_Service.Properties.Settings.Northwind"

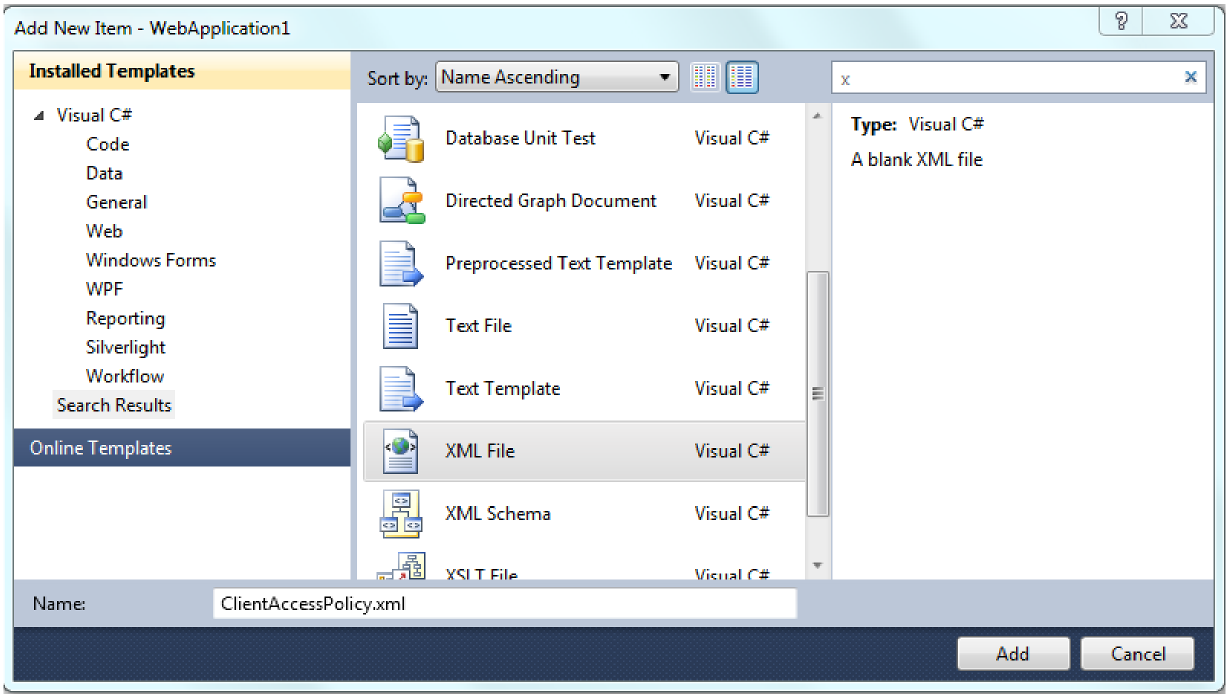
In this example a SqlCommand is used to select a set of customers and a SqlDataReader populates customer objects. Customer instances are added to a generic list and passed back as the result of the service method.

You could use a variety of other databases and data retrieval mechanisms here. As long as the contract is satisfied and the method returns a generic list of Customer, how you fill the list can vary according to your situation.

```
<OperationContract> _
Public Function GetCustomers() As List(Of Customer)
  Dim connectionString As String = _
ConfigurationManager.ConnectionStrings( _
"<project name>.Properties.Settings.Northwind").ConnectionString
  Dim result = New List(Of Customer)()
  Using conn As New SqlConnection(connectionString)
    Const sql As String = "SELECT Top 10 Address, Bool, City, " & _
"CompanyName, ContactName, ContactTitle, " & _
"Country, CustomerID, Fax, Phone, PostalCode FROM Customers"
    conn.Open()
    Using command As New SqlCommand(sql, conn)
      Dim reader As SqlDataReader = _
command.ExecuteReader(CommandBehavior.CloseConnection)
      Do While reader.Read()
        Dim customer = New Customer With { _
.Address = reader.GetValue(0).ToString(), _
.Bool = reader.GetBoolean(1), _
.City = reader.GetValue(2).ToString(), _
.CompanyName = reader.GetValue(3).ToString(), _
.ContactName = reader.GetValue(4).ToString(), _
.ContactTitle = reader.GetValue(5).ToString(), _
.Country = reader.GetValue(6).ToString(), _
.CustomerID = reader.GetValue(7).ToString(), _
.Fax = reader.GetValue(8).ToString(), _
.Phone = reader.GetValue(9).ToString(), _
.PostalCode = reader.GetValue(10).ToString()}
        result.Add(customer)
      Loop
      Return result
    End Using
  End Using
End Function
```

```
[OperationContract]
public List<Customer> GetCustomers()
{
    string connectionString =
        ConfigurationManager.ConnectionStrings[
"<project name>.Properties.Settings.Northwind"].ConnectionString;
    var result = new List<Customer>();
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        const string sql = @"SELECT Top 10 Address, Bool, City, " +
            "CompanyName, ContactName, ContactTitle, " +
            "Country, CustomerID, Fax, Phone, PostalCode FROM Customers";
        conn.Open();
        using (SqlCommand command = new SqlCommand(sql, conn))
        {
            SqlDataReader reader =
                command.ExecuteReader(CommandBehavior.CloseConnection);
            while (reader.Read())
            {
                var customer = new Customer
                {
                    Address = reader.GetValue(0).ToString(),
                    Bool = reader.GetBoolean(1),
                    City = reader.GetValue(2).ToString(),
                    CompanyName = reader.GetValue(3).ToString(),
                    ContactName = reader.GetValue(4).ToString(),
                    ContactTitle = reader.GetValue(5).ToString(),
                    Country = reader.GetValue(6).ToString(),
                    CustomerID = reader.GetValue(7).ToString(),
                    Fax = reader.GetValue(8).ToString(),
                    Phone = reader.GetValue(9).ToString(),
                    PostalCode = reader.GetValue(10).ToString()
                };
                result.Add(customer);
            }
            return result;
        }
    }
}
```

12)In the Solution Explorer, right-click the project and select **Add > New Item...** from the context menu. Name the file "ClientAccessPolicy.xml", then click the **Add** button to create the file and close the dialog.



Name: ClientAccessPolicy.xml

13)Replace the contents of "ClientAccessPolicy.xml" with the following XML.

This XML allows requests from all domains to get resources from all locations on the server.

```
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
 <cross-domain-access>
  <policy>
   <allow-from http-request-headers="*">
    <domain uri="*"/>
   </allow-from>
   <grant-to>
    <resource path="/" include-subpaths="true"/>
   </grant-to>
  </policy>
 </cross-domain-access>
</access-policy>
```

14)In the Solution Explorer, right-click the "ClientAccessPolicy.xml" file and select "Properties" from the context menu. Set the "Copy to Output Directory" property to "Copy if Newer". *This step will make sure that policy file ends up in the \bin directory, i.e. the root directory for the service. When Silverlight tries to access the service, it will find the policy file there and can continue interacting with the service.*

18.5.3.2 Building the WCF Silverlight Client

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.GridView**

c) **Telerik.Windows.Data**

d) **Telerik.Windows.Controls.Input.dll**

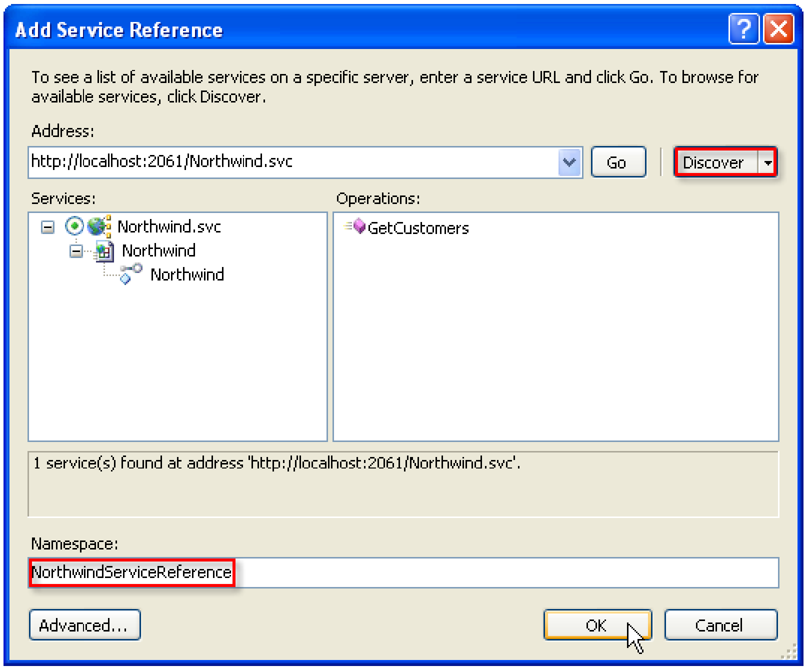
XAML Editing

1) Drag a RadGridView from the Toolbox to a point between the main "LayoutRoot" grid. Set the "x:Name" attribute to "gvMain" so we can reference it later in code.

Reference The WCF Service

1) In the Solution Explorer, right-click the References node and select **Add Service Reference...** This will display the "Add Service Reference" dialog.

2) In the "Add Service Reference" dialog, click the **Discover** button. The Northwind.svc server will display. Enter "NorthwindServiceReference" as the Namespace and click **OK** to create the client proxy object.



To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:2061/Northwind.svc

Services:
Northwind.svc
Northwind
Northwind

Operations:
GetCustomers

1 service(s) found at address 'http://localhost:2061/Northwind.svc'.

Namespace: NorthwindServiceReference

Code Behind

1) Add a namespace reference for the proxy in the "Imports" (VB) or "using" (C#) section of code. This will be the name of your project + "." + "NorthwindServiceReference", i.e. "MyProject. NorthwindServiceReference".

2) Handle the UserControl Loaded event.

   The client proxy methods are asynchronous, so this will be very much like working with the WebClient against a REST service. The proxy will have a "Completed" event, i.e. "GetCustomersCompleted" and an "Async" method, i.e. "GetCustomersAsync()". Create an instance of the client proxy object, hook up the "GetCustomersCompleted" event, then call the "GetCustomersAsync()" method.

   VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim client As New NorthwindClient()
  AddHandler client.GetCustomersCompleted, _
AddressOf client_GetCustomersCompleted
  client.GetCustomersAsync()
End Sub
```

   C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   NorthwindClient client = new NorthwindClient();
   client.GetCustomersCompleted +=
      new EventHandler<GetCustomersCompletedEventArgs>(
        client_GetCustomersCompleted);
   client.GetCustomersAsync();
}
```

3) Handle the GetCustomersCompleted event. The result of the method is passed back in e.Result. Simply assign the result to the grid view **ItemsSource** property. *Note: The result, even though it left the service as a List<Customer>, ends up in Silverlight as ObservableCollection<Customer>.*

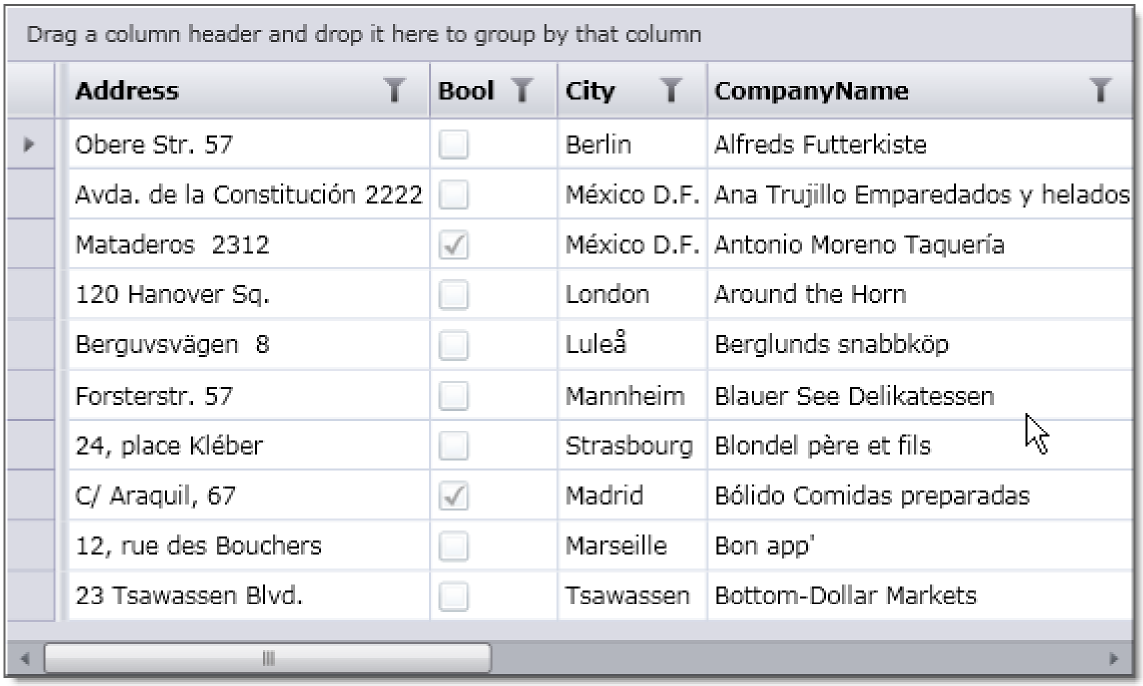
   VB

```
Private Sub client_GetCustomersCompleted(ByVal sender As Object, _
ByVal e As GetCustomersCompletedEventArgs)
  gvMain.ItemsSource = e.Result
End Sub
```

```
void client_GetCustomersCompleted(object sender,
GetCustomersCompletedEventArgs e)
{
   gvMain.ItemsSource = e.Result;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Drag a column header and drop it here to group by that column

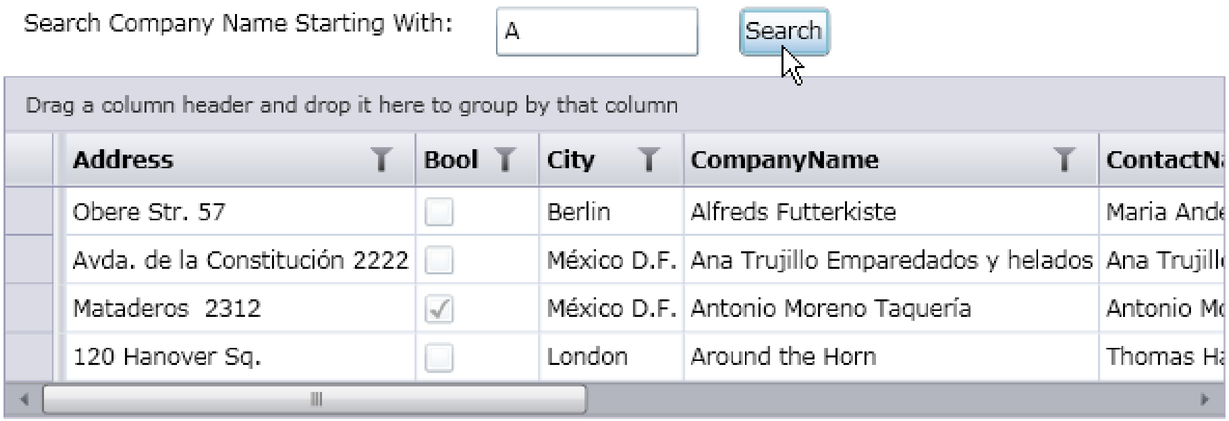
Address	Bool	City	CompanyName
Obere Str. 57	☐	Berlin	Alfreds Futterkiste
Avda. de la Constitución 2222	☐	México D.F.	Ana Trujillo Emparedados y helados
Mataderos 2312	☑	México D.F.	Antonio Moreno Taquería
120 Hanover Sq.	☐	London	Around the Horn
Berguvsvägen 8	☐	Luleå	Berglunds snabbköp
Forsterstr. 57	☐	Mannheim	Blauer See Delikatessen
24, place Kléber	☐	Strasbourg	Blondel père et fils
C/ Araquil, 67	☑	Madrid	Bólido Comidas preparadas
12, rue des Bouchers	☐	Marseille	Bon app'
23 Tsawassen Blvd.	☐	Tsawassen	Bottom-Dollar Markets

Test Application Features

1) Ten rows of data with all columns from the SQL should show up in the grid.

Ideas for Extending This Example

- Change the service method or add a new method that allows parameters. You can either parameterize the query itself or use LINQ to filter the contents. Add a text box and search button to the client. You can send the search text when calling GetCustomersAsync() using one of the overloads that allows an "userState" object parameter.



Search Company Name Starting With: A

Drag a column header and drop it here to group by that column

Address	Bool	City	CompanyName	ContactN
Obere Str. 57	□	Berlin	Alfreds Futterkiste	Maria Ande
Avda. de la Constitución 2222	□	México D.F.	Ana Trujillo Emparedados y helados	Ana Trujill
Mataderos 2312	☑	México D.F.	Antonio Moreno Taquería	Antonio Mo
120 Hanover Sq.	□	London	Around the Horn	Thomas Ha

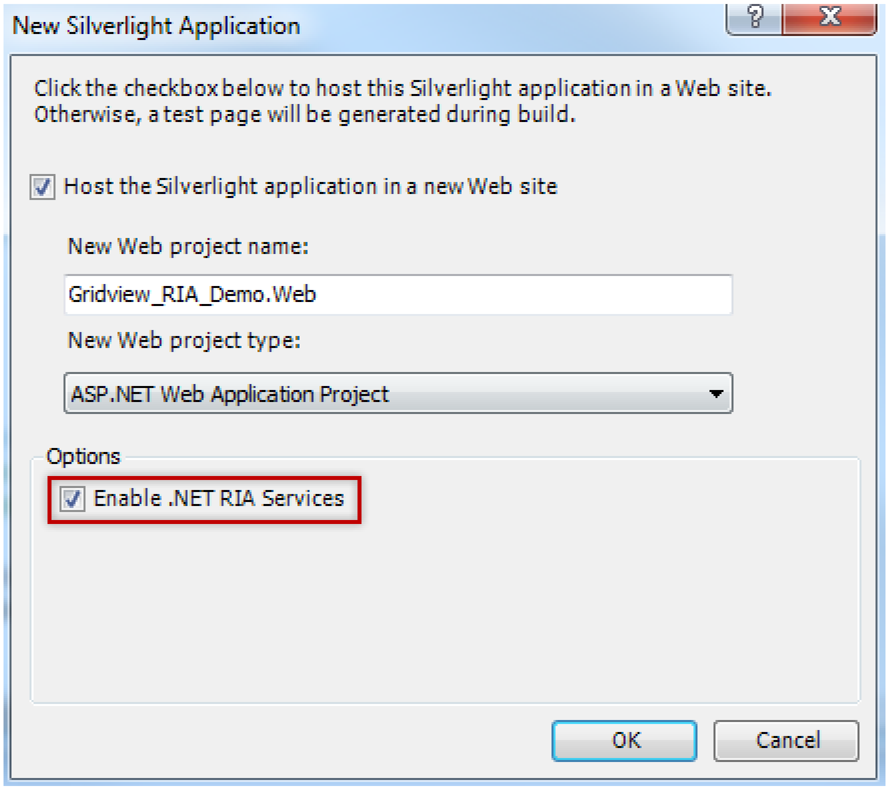
18.5.4 WCF RIA Services

The example service consumes the Northwind "Categories" table using LINQ to Entities. The Silverlight client instantiates a proxy client, retrieves a list of entities and binds them to the grid view.

18.5.4.1 Project Setup

1) In Visual Studio, create a new Silverlight Application. This will display the **New Silverlight Application** dialog. Check the "Enable WCF RIA Services" option. Leave the other defaults and click **OK** to close the dialog and create the projects.

The RIA service layer will be created in the ASP.NET host application, in this case, the Gridview_RIA_Demo.Web project.



New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site.
Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new Web site

New Web project name:

Gridview_RIA_Demo.Web

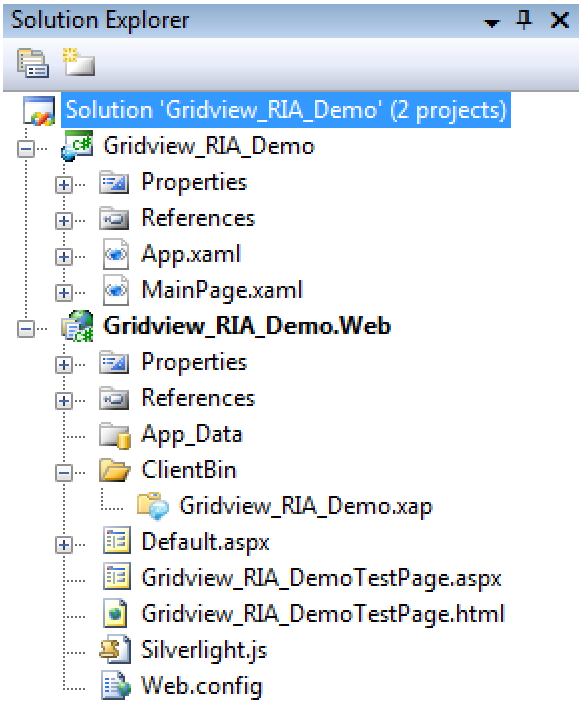
New Web project type:

ASP.NET Web Application Project

Options

☑ Enable .NET RIA Services

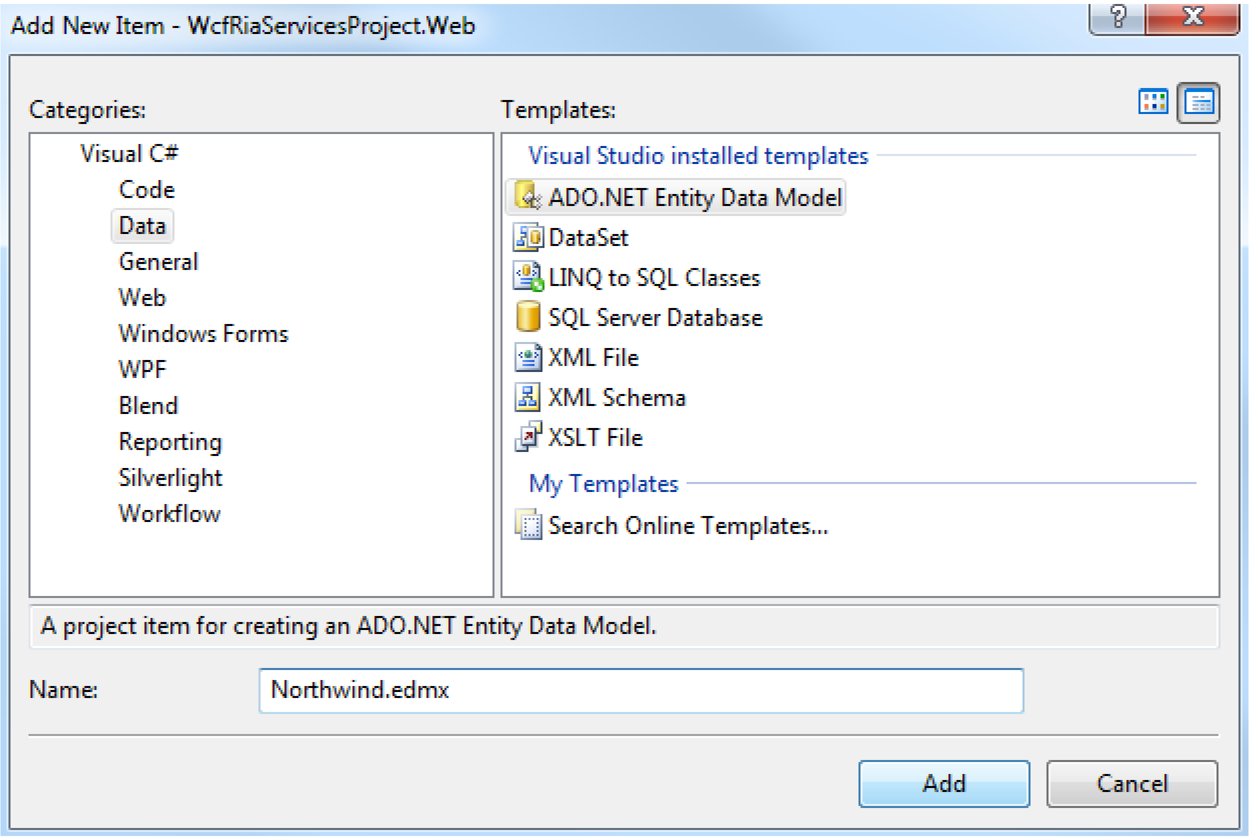
2) Take a look at the Solution Explorer and notice that you now have two projects, the ASP.NET host project ("Gridview_RIA_Demo.Web in the screenshot below) that will contain the RIA service and the Silverlight client application ("Gridview_RIA_Demo").



18.5.4.2 Building the RIA Service

1) Add a "ADO.NET Entity Data Model" item to the ASP.NET project. Name the data model "Northwind.edmx" and click the **Add** button.

This will display the Entity Data Model Wizard dialog.



A project item for creating an ADO.NET Entity Data Model.

Name: Northwind.edmx

2) In the first page of "Entity Data Model Wizard", select the "Generate from Database" icon and click the **Next** button.

3) Create a connection to the Northwind database file that ships with RadControls for Silverlight.

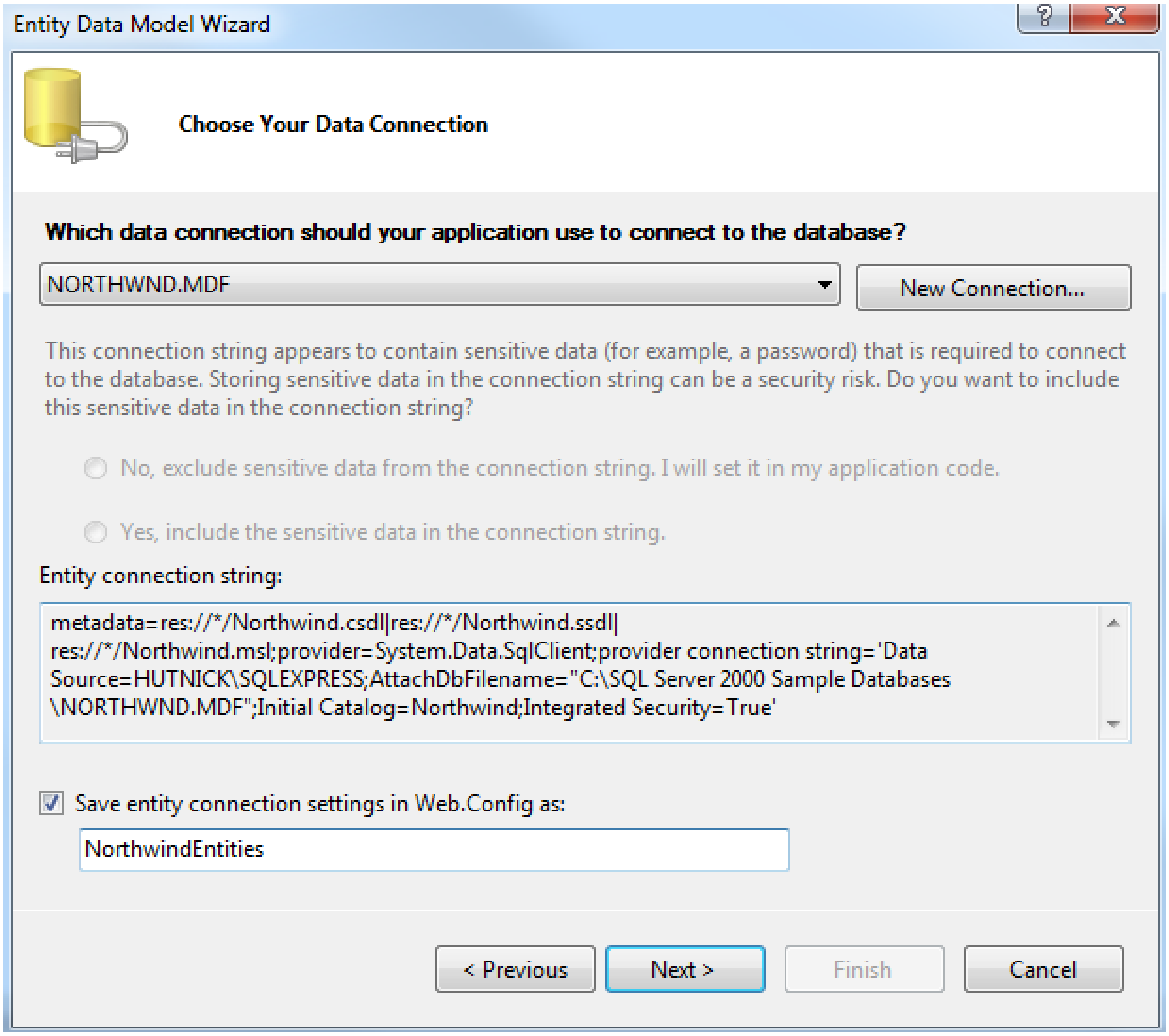
a) In the "Choose your Data Connection" page of the wizard, click the **New Connection** button. This will display the Connection Properties dialog.

b) Click the **Change** button to show the "Change Data Source" dialog, select the "Microsoft SQL Server Database File" option and click the **OK** button to return to the "Change Data Source" dialog.

c) Click the **Browse** button, locate the file "Northwind.mdf" file in the RadControls for Silverlight installation directory under \Examples\DataSources.

d) Click **OK** to create the connection and return to the "Entity Data Model Wizard" "Choose Your Data Connection" page.

4) In the "Choose Your Data Connection" page of the wizard, enter "NorthwindEntities" in the "Save entity connection settings in Web.Config" text box and click the **Next** button.



Choose Your Data Connection

Which data connection should your application use to connect to the database?

NORTHWND.MDF

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

○ No, exclude sensitive data from the connection string. I will set it in my application code.

○ Yes, include the sensitive data in the connection string.

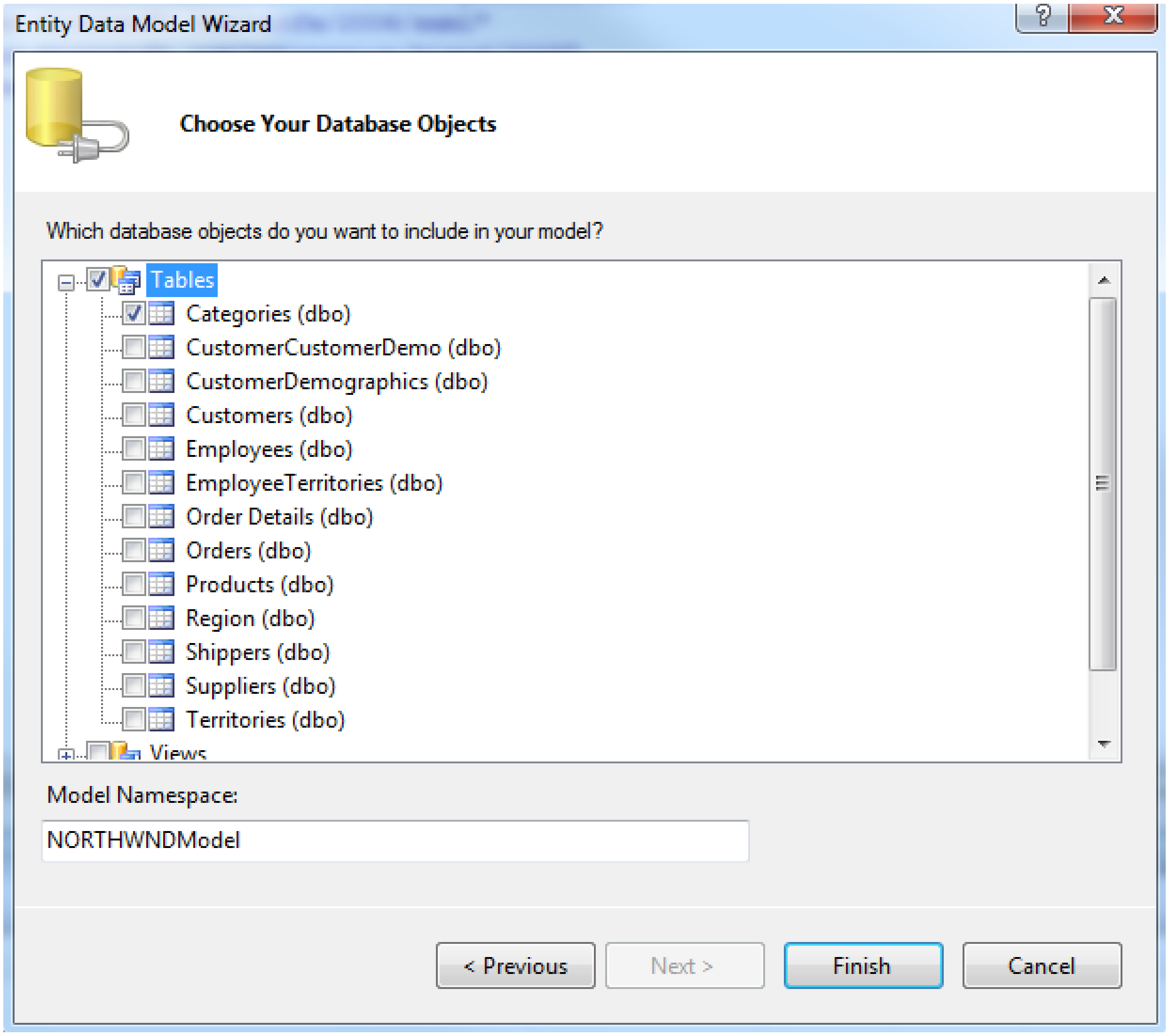
Entity connection string:

```
metadata=res://*/Northwind.csdl|res://*/Northwind.ssdl|
res://*/Northwind.msl;provider=System.Data.SqlClient;provider connection string='Data
Source=HUTNICK\SQLEXPRESS;AttachDbFilename="C:\SQL Server 2000 Sample Databases
\NORTHWND.MDF";Initial Catalog=Northwind;Integrated Security=True'
```

☑ Save entity connection settings in Web.Config as:

NorthwindEntities

5) In the "Choose Your Database Objects" page of the wizard, expand the "Tables" node in the tree view and select the "Categories" table check box.



Entity Data Model Wizard

Choose Your Database Objects

Which database objects do you want to include in your model?

- ☑ Tables
  - ☑ Categories (dbo)
  - □ CustomerCustomerDemo (dbo)
  - □ CustomerDemographics (dbo)
  - □ Customers (dbo)
  - □ Employees (dbo)
  - □ EmployeeTerritories (dbo)
  - □ Order Details (dbo)
  - □ Orders (dbo)
  - □ Products (dbo)
  - □ Region (dbo)
  - □ Shippers (dbo)
  - □ Suppliers (dbo)
  - □ Territories (dbo)
- □ Views

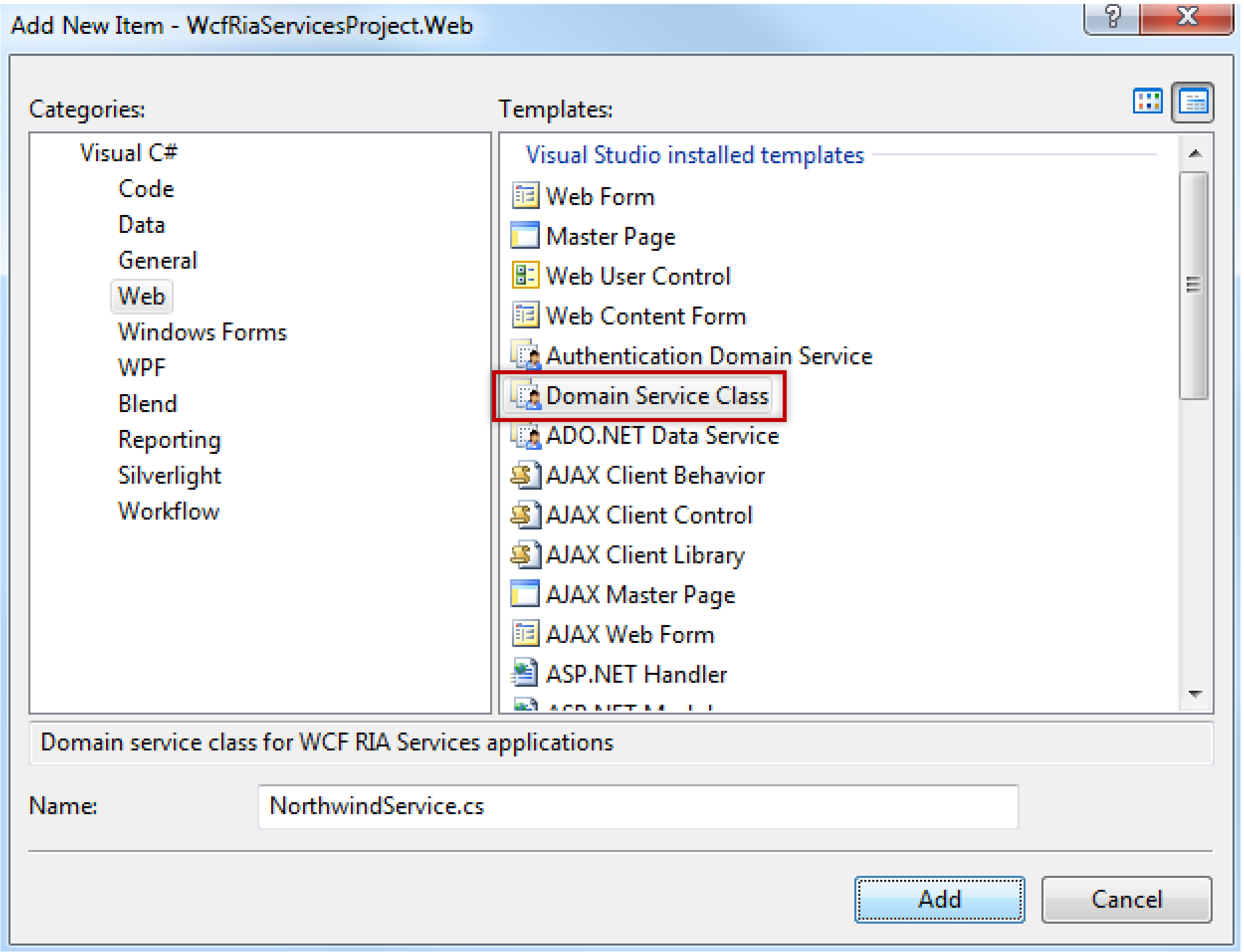
Model Namespace:

NORTHWNDModel

6) Click the **Finish** button to create the data model.

7) Build the ASP.NET project. **This step is important**: the following step where you create the Domain Service Class will not see the entity data model information without building the project.

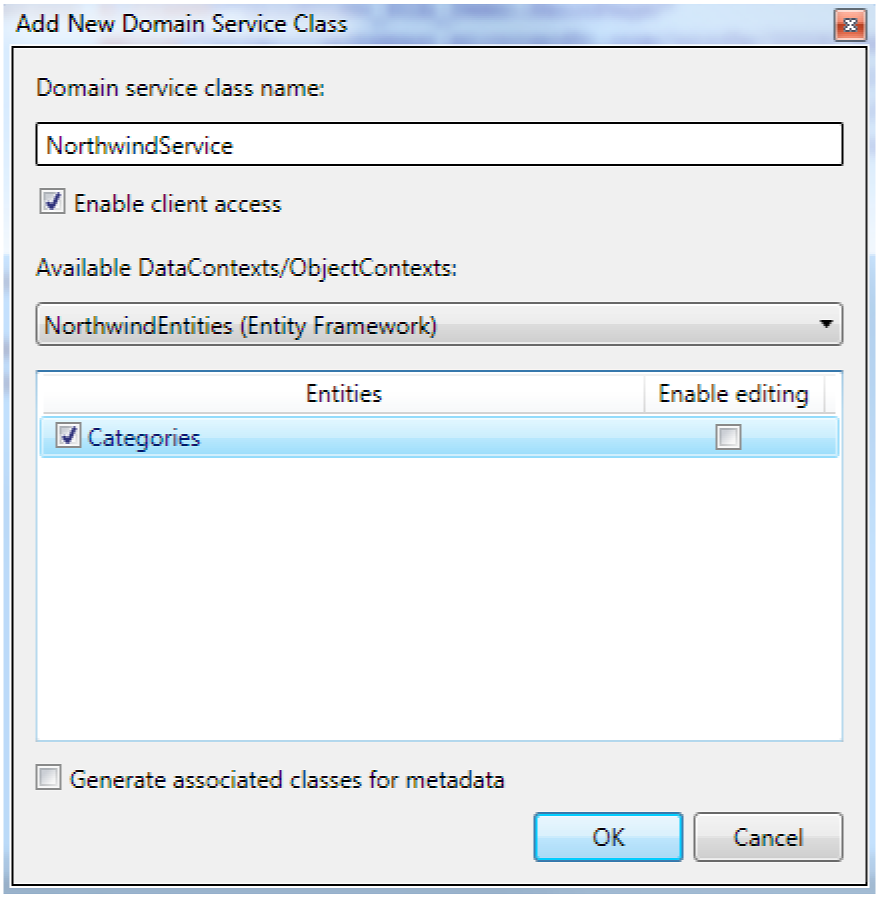
8) Add a "Domain Service Class" item to your project. Name it "NorthwindService" and click the **Add** button to create the service and close the dialog. This step will display the "Add New Domain Service Class" dialog.



Domain service class for WCF RIA Services applications

Name: NorthwindService.cs

9) In the "Add New Domain Service Class" dialog, name the Domain Service Class "NorthwindService", make sure that the "Enable client access" option is checked (this allows the client proxy code to be generated), select "NorthwindEntities" from the drop down list of available context objects and check the "Categories" entity check box. Click the **OK** button to create the domain service class and close the dialog.



Add New Domain Service Class

Domain service class name: NorthwindService

☑ Enable client access

Available DataContexts/ObjectContexts: NorthwindEntities (Entity Framework)

Entities	Enable editing
☑ Categories	□

□ Generate associated classes for metadata

10)Build the project.

11)Review the generated NorthwindService code. Notice that a GetCategories() method has been created that returns an IQueryable<> of Categories.

18.5.4.3 Building the RIA Client

1) In the Solution Explorer, open the Silverlight client application node.
2) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add references to these assemblies:
   a) **Telerik.Windows.Controls**
   b) **Telerik.Windows.Data**
   c) **Telerik.Windows.Controls.GridView**
   d) **Telerik.Windows.Controls.Input.dll**
3) Open "MainPage.xaml" for editing.
4) Drag a RadGridView from the Toolbox into the main "LayoutRoot" grid element of the page. Set the "x:Name" attribute to "gvMain" so that we can refer to it later in code.
5) Open "MainPage.xaml.cs" for editing.
6) Add a namespace reference to the ASP.NET service project "Imports" (VB) or "using" (C#) portion of code.
7) In the constructor for the UserControl, get the data from the domain service by way of the context object and bind it to the grid view using the code below:
   a) Create the "context" object (the generated client counterpart to the domain service).
   b) Assign the context "Categories" property to the grid view ItemsSource property.
   c) Call the context object Load() method and pass the context "GetCategoriesQuery()" method.

VB

```
Public Sub New()
  InitializeComponent()

  Dim context As New NorthwindContext()
  gvMain.ItemsSource = context.Categories
  context.Load(context.GetCategoriesQuery())
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();

   NorthwindContext context = new NorthwindContext();
   gvMain.ItemsSource = context.Categories;
   context.Load(context.GetCategoriesQuery());
}
```

Run The Application

Press **F5** to run the application.

18.6 Customization

The "A RESTful Walk Through" section of this chapter demonstrated customizing an entire row for that free form "Card" look. In this section we will look at individual customization of a column using a cell template. This example extends the "WCF RIA Services Walk Through" by displaying the "Picture" column. In the process we will cover the following:

- Format a CellTemplate of a GridViewDataColumn. The first column will contain a TextBlock bound to the category description. It will use a variation on the "shadow" technique discussed in the ToolBar chapter to place a second TextBlock "reflection" of the first. The second column will contain an Image surrounded by a Border.
- Use an IValueConverter to convert the binary picture data to a BitmapImage. **Note:** Be sure to check the latest features of RadControls for Silverlight for the new **GridViewImageColumn** type that replaces the need for converters in most cases.
- Hide all visual clues that we're working with a grid except the category description and the picture.

We will look at just those key parts that have changed from the original RIA Services Walk Through project, starting with the cell template. First, to get oriented in the XAML before we dig down into the cell template, take a look at the RadGridView element. It contains a Columns element and within that, multiple GridViewDataColumn elements, one for each column.

```
<telerik:RadGridView . . .">
  <telerik:RadGridView.Columns>
    <telerik:GridViewDataColumn
        DataMemberBinding="{Binding Description}">
     <!--template here-->
    </telerik:GridViewDataColumn>
    <telerik:GridViewDataColumn
        DataMemberBinding="{Binding Picture}" >
     <!--template here-->
    </telerik:GridViewDataColumn>
  </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

Now we can look at the detail of the "Description" column and see how the template is put together. Inside the GridViewDataColumn, add a GridViewDataColumn CellTemplate element. Inside the CellTemplate is a DataTemplate that holds whatever elements suit your purpose. Here we are adding a StackPanel to organize the two TextBlocks. The Text property of both TextBlock elements is bound to the "Description" property.

```
<telerik:GridViewDataColumn
    DataMemberBinding="{Binding Description} ">
  <telerik:GridViewDataColumn.CellTemplate>
    <DataTemplate>
      <StackPanel>
        <TextBlock HorizontalAlignment="Right"
            Text="{Binding Description}"
            Style="{StaticResource TextStyle}" />
        <TextBlock HorizontalAlignment="Right"
            Text="{Binding Description}"
            Style="{StaticResource ShadowTextStyle}" />
      </StackPanel>
    </DataTemplate>
  </telerik:GridViewDataColumn.CellTemplate>
</telerik:GridViewDataColumn>
```

The "Picture" column follows the same pattern, but the DataTemplate contains a Border surrounding an Image element. One difference to this column is that we are using an IValueConverter called "ImageConverter" to create a BitmapImage from the raw database data.

```
<telerik:GridViewDataColumn
    DataMemberBinding="{Binding Picture}">
  <telerik:GridViewDataColumn.CellTemplate>
    <DataTemplate>
      <Border Style="{StaticResource BorderStyle}">
        <Image Margin="5"
          Source="{Binding Picture,
           Converter={StaticResource ImageConverter}}"
          Width="50" Height="50" />
      </Border>
    </DataTemplate>
  </telerik:GridViewDataColumn.CellTemplate>
</telerik:GridViewDataColumn>
```

We looked briefly at IValueConverter in the Input Controls chapter to demonstrate converting a number to a string displayed on a Slider control. Refer back to that chapter for a more detailed explanation of IValueConverter. The "value" parameter passed to Convert is the byte array from the "Picture" column in the database. Use that byte array to populate a new MemoryStream and then, set the source of a BitmapImage to be the MemoryStream. Finally, the BitmapImage is returned from the method.

VB

```
Public Class ImageConverter
  Implements IValueConverter
  Public Function Convert(ByVal value As Object, ByVal targetType As Type, _
 ByVal parameter As Object, ByVal culture As CultureInfo) As Object
    ' convert the "Picture" byte array to a memory stream
    Dim memoryStream As New MemoryStream((CType(value, Byte())))
    Dim image As New BitmapImage()
    Try
      image.SetSource(memoryStream)
    Catch ' ignore invalid arrays
    End Try
    Return image
  End Function

  Public Function ConvertBack(ByVal value As Object, ByVal targetType As Type, _
ByVal parameter As Object, ByVal culture As CultureInfo) As Object
    Throw New NotImplementedException()
  End Function
End Class
```

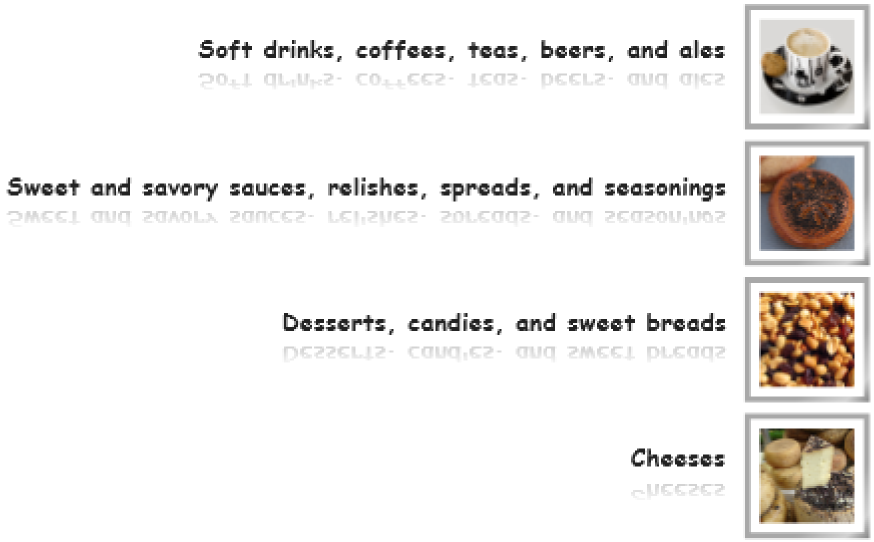
C#

```
public class ImageConverter: IValueConverter
{
   public object Convert(
      object value, Type targetType, object parameter, CultureInfo culture)
   {
      // convert the "Picture" byte array to a memory stream
      MemoryStream memoryStream = new MemoryStream(((byte[])value));
      BitmapImage image = new BitmapImage();
      try { image.SetSource(memoryStream); }
      catch {} // ignore invalid arrays
      return image;
   }

   public object ConvertBack(object value, Type targetType,
     object parameter, CultureInfo culture)
   { throw new NotImplementedException(); }
}
```

For a reference, here are the resources used in the project. Remember that you will need to add an XML namespace reference to the project that contains the IValueConverter implementation (shown below as "local"). The other resources are styles that you can copy or change as you wish.

```
<UserControl.Resources>
  <LinearGradientBrush x:Key="BorderBrush">
    <GradientStop Color="DarkGray" Offset=".7" />
    <GradientStop Color="Silver" Offset=".8" />
    <GradientStop Color="LightGray" Offset=".9" />
    <GradientStop Color="Gray" Offset="1" />
  </LinearGradientBrush>
  <Style x:Key="BorderStyle" TargetType="Border">
    <Setter Property="BorderBrush" Value="{StaticResource BorderBrush}" />
    <Setter Property="BorderThickness" Value="3" />
    <Setter Property="Margin" Value="3" />
  </Style>
  <Style x:Key="TextStyle" TargetType="TextBlock">
    <Setter Property="FontFamily" Value="Comic Sans MS" />
    <Setter Property="Margin" Value="0" />
    <Setter Property="FontSize" Value="13" />
    <Setter Property="Foreground" Value="#FF222222" />
    <Setter Property="FontWeight" Value="Bold" />
  </Style>
  <Style x:Key="ShadowTextStyle" TargetType="TextBlock"
      BasedOn="{StaticResource TextStyle}">
    <Setter Property="OpacityMask">
      <Setter.Value>
        <LinearGradientBrush StartPoint="0,0" EndPoint="0,1">
          <GradientStop Color="Transparent" Offset="0" />
          <GradientStop Color="#33000000" Offset="1" />
        </LinearGradientBrush>
      </Setter.Value>
    </Setter>
    <Setter Property="RenderTransformOrigin" Value="0,0.5" />
    <Setter Property="RenderTransform">
      <Setter.Value>
        <TransformGroup>
          <ScaleTransform ScaleY="-.5"></ScaleTransform>
          <SkewTransform AngleX="0.1"></SkewTransform>
        </TransformGroup>
      </Setter.Value>
    </Setter>
  </Style>
  <local:ImageConverter x:Key="ImageConverter" />
</UserControl.Resources>
```

The finished project will look something like the screenshot below:



Soft drinks, coffees, teas, beers, and ales

Sweet and savory sauces, relishes, spreads, and seasonings

Desserts, candies, and sweet breads

Cheeses

18.7 Wrap Up

In this chapter you were introduced to RadGridView and many of its key features. You started out by binding the grid view to some basic data. Then you saw how to expand that example to use hierarchical data and how to customize the columns.

While delving into the details of RadGridView you worked with selected rows and cells. You handled events that notified you when the user was moving within the grid view. You learned the programming model commonalities for filtering, sorting and grouping. While working with groups, you learned how to add aggregate functions for each group. You were introduced to column types and learned about special columns that handle images, hyperlinks and lookups. "Grid View Elements Visibility" demonstrated how to show and hide the visual elements of the grid view.

In the Binding section of this chapter you bound the grid view to simple .NET objects. From there you built a REST service that queried Twitter. Then you built WCF and WCF RIA services. In the process you also learned about how to work with inherent Silverlight security restrictions. You also learned how to customize the layout of an entire grid view row.

In the Customization section of this chapter you customized grid view cells to achieve a unique look.

Part XIX

Scheduler

19 Scheduler

19.1 Objectives

This chapter starts out by using RadScheduler in a simple project to create a single appointment. Then you will be introduced to view modes that allow you switch between month, week and day views. You will create and configure appointments and also learn role of IAppointment and AppointmentBase in building custom appointment classes. You will select appointments programmatically and respond to user selections in the scheduler. You will use RecurrenceRule and RecurrencePattern classes to specify the recurrence behavior of an appointment. We will take a brief look at scheduler commands and how they can be used programmatically and declaratively within XAML. RadDragAndDropManager will be used to drag-and-drop ListBox items into time slots. You will localize the scheduler using the LocalizationManager with pre-defined cultures and custom resource files.

You will perform basic data binding to a collection of Appointments and also work with building and binding to custom appointment objects. Finally, you will create a custom Theme for RadScheduler where the Appointment Editing dialog user interface is modified to include a custom appointment property.

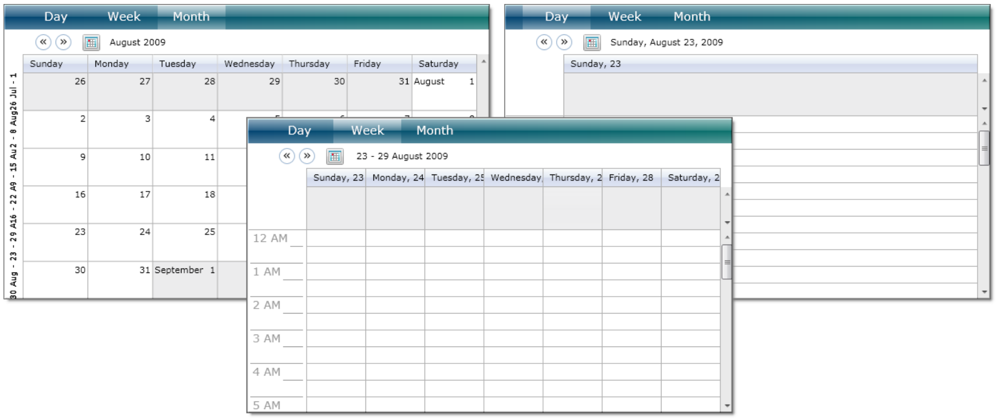
Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Scheduler\Scheduler.sln

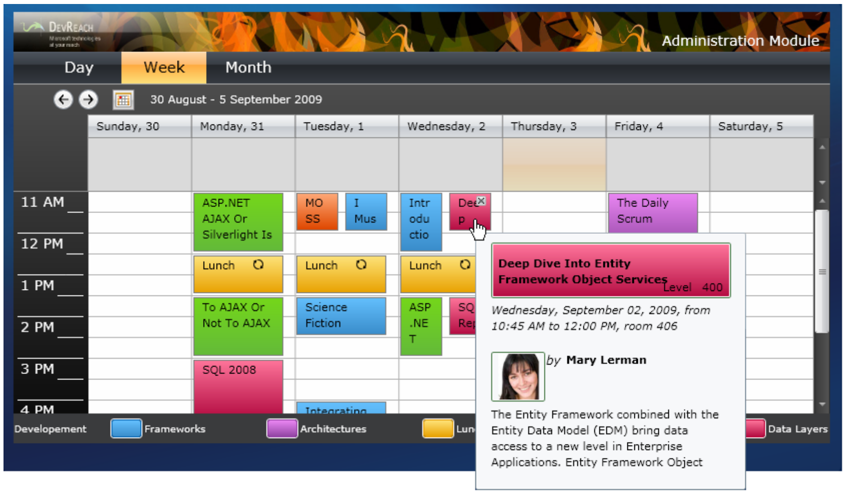
19.2 Overview

RadScheduler serves up Office-like appointment scheduling packed with features:

- Day, Week, Month and configurable multi-day views



- Binds to collections of pre-defined Appointment objects or to a custom classes you define for your specific business requirements.
- Fully templated to allow customization at any level. Alter the appearance of any view, time slots, appointments and appointment editing dialog.



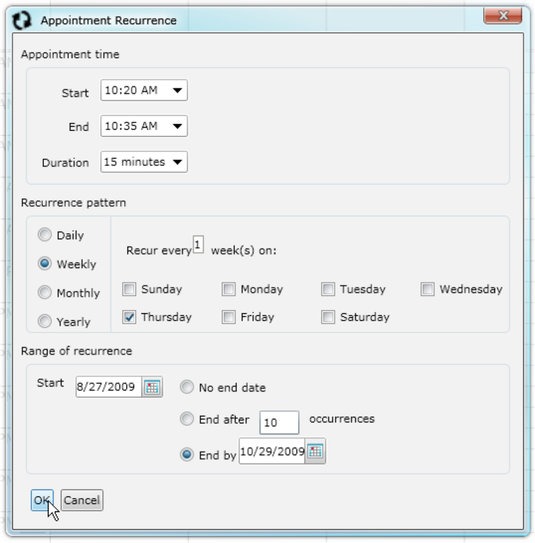
Deep Dive Into Entity Framework Object Services Level 400

Wednesday, September 02, 2009, from 10:45 AM to 12:00 PM, room 406

by **Mary Lerman**

The Entity Framework combined with the Entity Data Model (EDM) bring data access to a new level in Enterprise Applications. Entity Framework Object

- Recurrent appointment behavior handles appointments that repeat within a time range or fit a pattern, e. g. "every Thursday, weekly".



Appointment Recurrence

Appointment time

Start 10:20 AM

End 10:35 AM

Duration 15 minutes

Recurrence pattern

○ Daily
● Weekly
○ Monthly
○ Yearly

Recur every 1 week(s) on:

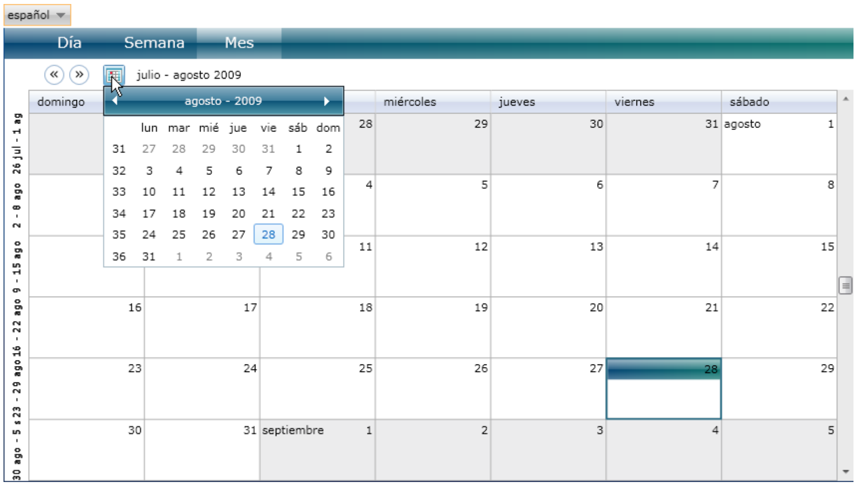
□ Sunday □ Monday □ Tuesday □ Wednesday
☑ Thursday □ Friday □ Saturday

Range of recurrence

Start 8/27/2009

○ No end date
○ End after 10 occurrences
● End by 10/29/2009

- RadScheduler can easily adjust to any culture or use resources to completely modify the terminology used in the scheduler.



agosto - 2009

	lun	mar	mié	jue	vie	sáb	dom
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31	1	2	3	4	5	6

19.3 Getting Started

In this walk through we will use only the very basic capabilities of RadScheduler. We will set the view of the scheduler and add a single appointment in code.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Scheduler**

c) **Telerik.Windows.Controls.Input.dll**

d) **Telerik.Windows.Controls.Navigation.dll**

e) **Telerik.Windows.Data.dll**

f) **Telerik.Windows.Themes.Vista**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace references to the **Telerik** assemblies.

```
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
```

3) Add a "Loaded" event to the UserControl element.

```
<UserControl
. . .
    Loaded="UserControl_Loaded">
```

4) Drag a RadScheduler from the Toolbox to a point inside the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. Name the RadScheduler "schTasks". Add attributes to the RadScheduler so that the **ViewMode**="Week" and **telerik:StyleManager.Theme**="Vista".

```xml
<telerik:RadScheduler x:Name="schTasks"
      ViewMode="Week"
      telerik:StyleManager.Theme="Vista">
</telerik:RadScheduler>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these name spaces:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Scheduler**

2) Handle the Loaded event of the UserControl.

a) Create a new **Appointment** instance and add it to the scheduler's **Appointments** collection.

b) Set the **Start** property to the current date and time using the **DateTime.Now** property.

c) Set the **End** property to 15 minutes after the current time using the **AddMinutes()** method of DateTime.

d) Set the **Subject** property to "Backup servers"

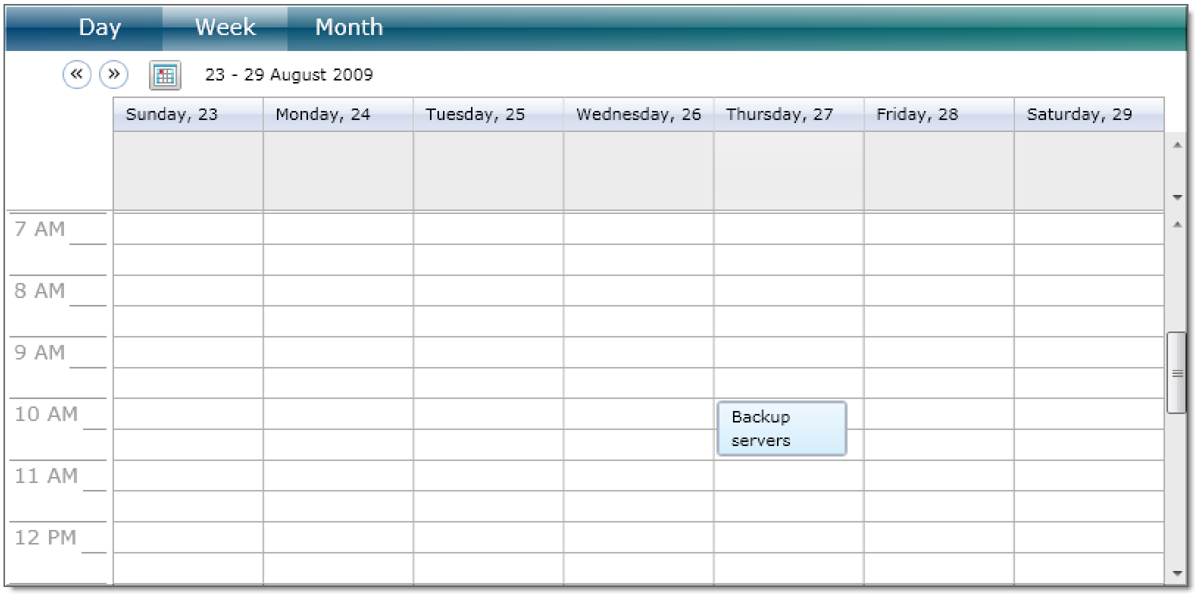
e) Set the **Body** to "Backup server contents to network storage".

```vb
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  schTasks.Appointments.Add(New Appointment() _
With {.Start = DateTime.Now, .End = DateTime.Now.AddMinutes(15), _
.Subject = "Backup servers", .Body = "Backup server contents to network storage"})
End Sub
```

```csharp
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   schTasks.Appointments.Add(new Appointment()
   {
      Start = DateTime.Now,
      End = DateTime.Now.AddMinutes(15),
      Subject = "Backup servers",
      Body = "Backup server contents to network storage",
   });
}
```

Run The Application

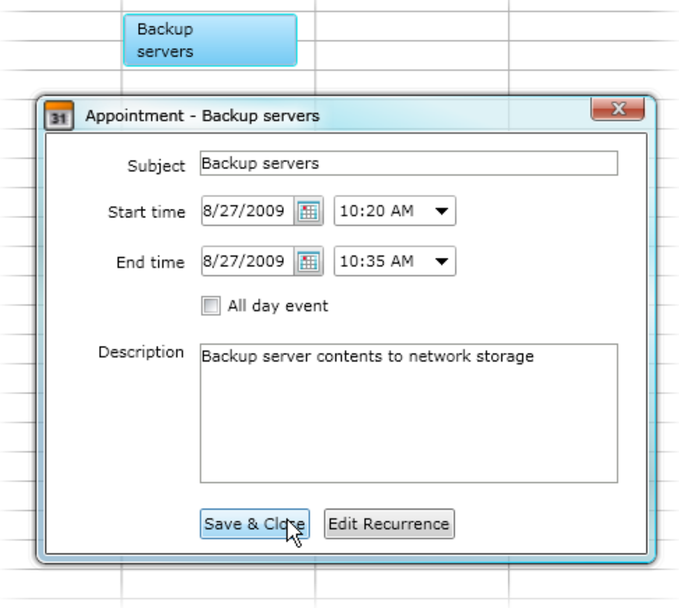
Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

1) The "Backup servers" appointment should be placed on the current time and date.

2) Double-click the "Backup servers" appointment to edit it. The Start, End, Subject and Description should match the corresponding properties set in the UserControl Loaded event.



Subject: Backup servers

Start time: 8/27/2009 10:20 AM

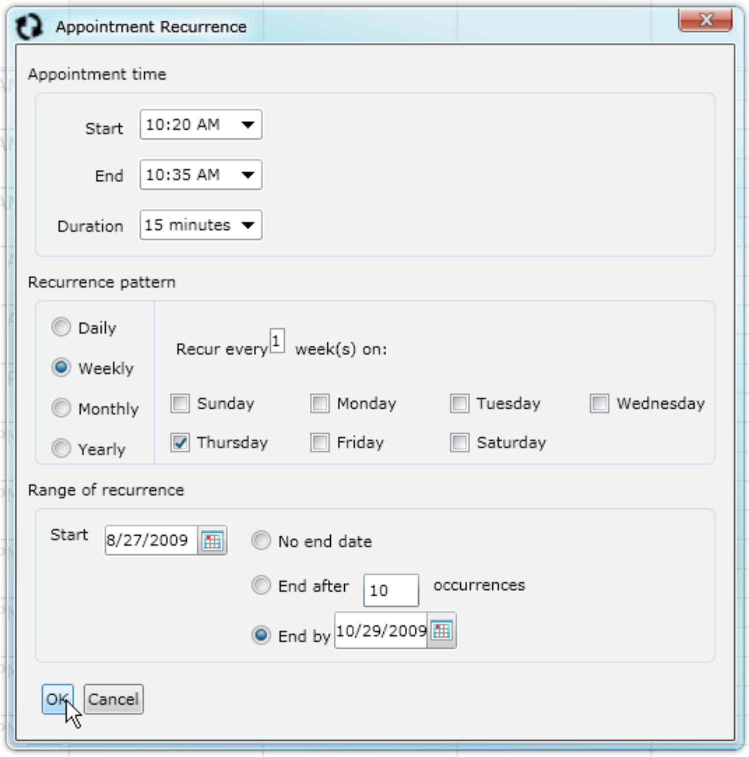
End time: 8/27/2009 10:35 AM

□ All day event

Description: Backup server contents to network storage

3) Edit and save the appointment. Open it again and verify the changes have persisted.

4) From the edit dialog, click the **Edit Recurrence** button and make changes to the Appointment Recurrence dialog. When you're through making edits, Click **OK.** Then click the **Save & Close** button of the edit dialog.



Appointment Recurrence

Appointment time

Start 10:20 AM

End 10:35 AM

Duration 15 minutes

Recurrence pattern

○ Daily
● Weekly
○ Monthly
○ Yearly

Recur every 1 week(s) on:

☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday
☑ Thursday ☐ Friday ☐ Saturday

Range of recurrence

Start 8/27/2009

○ No end date
○ End after 10 occurrences
● End by 10/29/2009

19.4 Control Details

19.4.1 Time Slots

Time slots are the areas of time in the scheduler that appointments can be placed in. You can select a time slot of any duration by assigning the scheduler **SelectedTimeSlot** property. SelectedTimeSlot is a **TimeSlot** defined in the **Telerik.Windows.Controls.Scheduler** namespace.

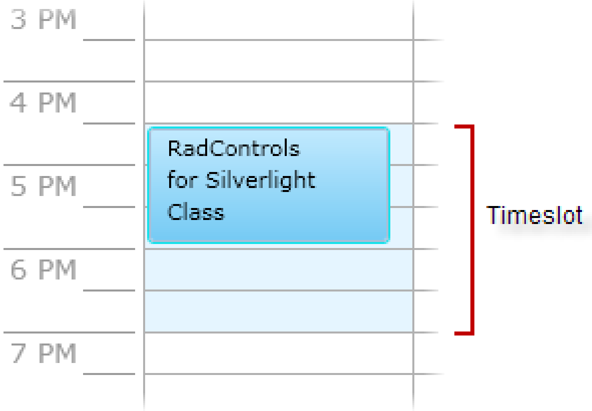
VB

```
schTasks.SelectedTimeSlot = _
New TimeSlot(appointment1.Start, appointment1.End.AddHours(1))
```

C#

```
schTasks.SelectedTimeSlot =
  new TimeSlot(appointment1.Start, appointment1.End.AddHours(1));
```

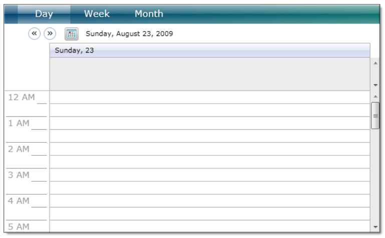
Running in the browser, the TimeSlot selection appears underneath an Appointment that happens to fall within the slot.



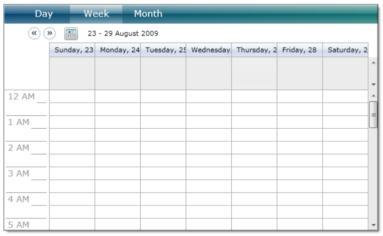
19.4.2 Views

RadScheduler has views for Day, Week and Month controlled by the **ViewMode** property.

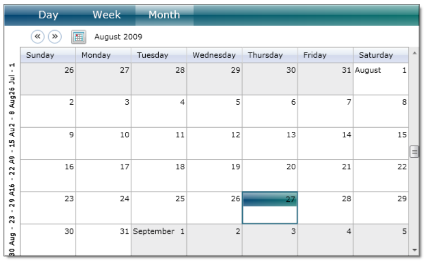
Day



Week



Month



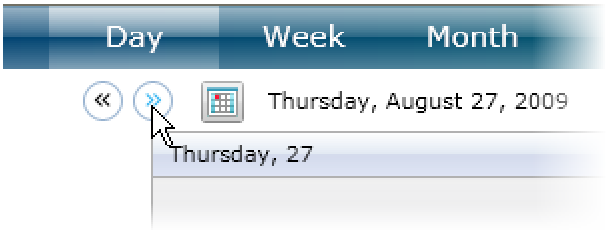
The ViewMode can be set in XAML or in code to show a particular view when your page starts up.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  schTasks.ViewMode = Telerik.Windows.Controls.SchedulerViewMode.Day
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
  schTasks.ViewMode = Telerik.Windows.Controls.SchedulerViewMode.Day;
}
```

The ViewMode also controls the **ActiveViewDefinition** property. ActiveViewDefinition controls how the times and dates are arranged in the view and how the view behaves. The instance contained in ActiveViewDefinition may be DayViewDefinition, WeekViewDefinition or MonthViewDefinition types. All three descend from the abstract **ViewDefinitionBase** and implement some of these properties:

- **VisibleDays**: The number of days that show in the view. You can use this property to have multiple days in Day mode or have a week of any length in Week mode.
- **LargeChangeInterval**: The amount of time to move in response to the forward and backward buttons. By default, the amount is one month for MonthViewDefinition, seven days for WeekViewDefinition and a single day for DayViewDefinition. The screenshot below shows the navigation buttons being used to move between days in the Day view.



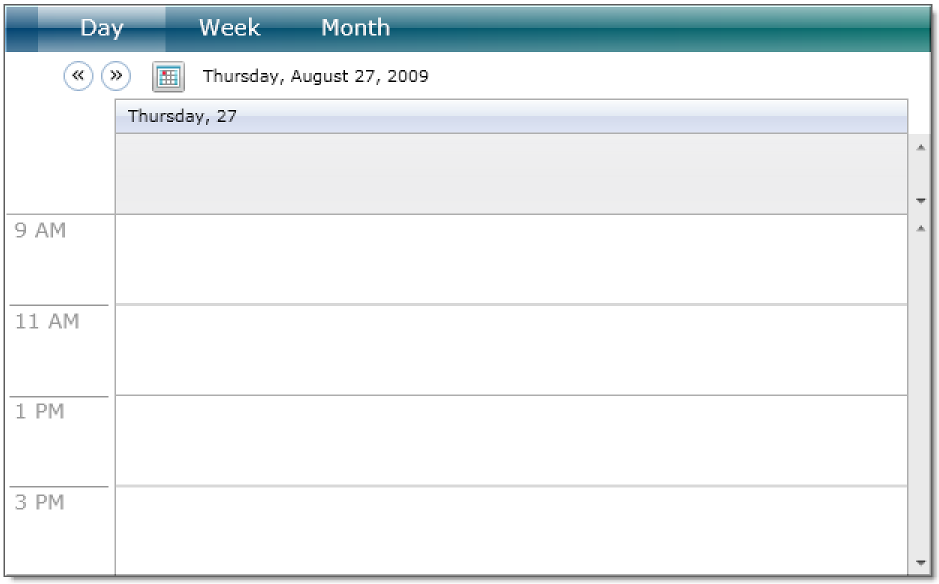
- **TimeSlotLength:** A TimeSpan that sets the length of each time slot.
- DayViewDefinition and MonthViewDefinition both have additional **DayStartTime** and **DayEndTime** TimeSpan properties that limit the length of the visible day. You could use these properties to display only business hours.

The XAML below shows how DayViewDefinition and WeekViewDefinition are configured. Take a look at the DayViewDefinition markup:

- Only one day is shown at a time
- Time slots are two hours long
- Large changes, triggered from the forward and back navigation buttons, move two days at a time
- The day starts at 9AM and ends at 5PM.

```
<telerik:RadScheduler x:Name="schTasks"
    telerik:StyleManager.Theme="Vista">
  <telerik:RadScheduler.DayViewDefinition>
    <telerik:DayViewDefinition VisibleDays="1"
        TimeSlotLength="2:0:0"
        LargeChangeInterval="2d" DayStartTime="9:0:0"
        DayEndTime="17:0:0">
    </telerik:DayViewDefinition>
  </telerik:RadScheduler.DayViewDefinition>
  <telerik:RadScheduler.WeekViewDefinition>
    <telerik:WeekViewDefinition VisibleDays="5"
        TimeSlotLength="1:0:0"
        LargeChangeInterval="7d">
    </telerik:WeekViewDefinition>
  </telerik:RadScheduler.WeekViewDefinition>
</telerik:RadScheduler>
```

Running in the browser, the day view looks like the screenshot below.



19.4.3 Appointments

RadScheduler can consume any appointment classes that implement **IAppointment**. You can implement IAppointment yourself, descend from the abstract **AppointmentBase** or use the **Appointment** class right-out-of-the-box. The easiest route of course is to create instances of Appointment and add them to the Appointments collection:

VB

```
schTasks.Appointments.Add( _
New Appointment() With { _
.Start = DateTime.Now, _
.End = DateTime.Now.AddHours(1), _
.Subject = "RadControls for Silverlight Class", _
.Body = "A three day, on-site course with mixed _
lecture and hands-on practical labs", _
.IsAllDayEvent = False, _
.Location = "Capitola, California", _
.Url = "http://www.telerik.com", _
.TimeZone = TimeZoneInfo.Local, _
.UniqueId = "1234"})
```

C#

```
schTasks.Appointments.Add(new Appointment()
{
   Start = DateTime.Now,
   End = DateTime.Now.AddHours(1),
   Subject = "RadControls for Silverlight Class",
   Body = "A three day, on-site course with mixed lecture" +
     "and hands-on practical labs",
   IsAllDayEvent = false,
   Location = "Capitola, California",
   Url = "http://www.telerik.com",
   TimeZone = TimeZoneInfo.Local,
   UniqueId = "1234"
});
```

Removing appointments is relatively straightforward and can be accomplished in several ways. You can use RemoveAt(index) to remove the appointment at a given index, Remove(IAppointment) takes an instance of the appointment to remove and RemoveAll() lets you remove every appointment that fits a certain filter. The example below uses a Lambda expression in the RemoveAll() parameter list to filter for all appointments where IsAllDayEvent is true.

VB

```
' remove the appointment at index "0"
schTasks.Appointments.RemoveAt(0)
' remove an IAppointment instance
schTasks.Appointments.Remove(TryCast(schTasks.Appointments(0), IAppointment))
' remove all matching appointments
schTasks.Appointments.RemoveAll(Function(app) app.IsAllDayEvent = True)
```

C#

```
// remove the appointment at index "0"
schTasks.Appointments.RemoveAt(0);
// remove an IAppointment instance
schTasks.Appointments.Remove(schTasks.Appointments[0] as IAppointment);
// remove all matching appointments
schTasks.Appointments.RemoveAll(app => app.IsAllDayEvent == true);
```

Much like RadCalendar or selection in a ListBox control, appointments can be selected by assigning the SelectedAppointment property or adding to the SelectedAppointments collection.

VB

```
schTasks.SelectedAppointment = appointment1
schTasks.SelectedAppointments.Add(appointment2)
```

C#

```
schTasks.SelectedAppointment = appointment1;
schTasks.SelectedAppointments.Add(appointment2);
```

Tip!

By default, the scheduler displays today's date. Use the scheduler's SetFirstVisibleDate() method to bring a specific day into view.

VB

```
schTasks.SetFirstVisibleDate(appointment1.Start)
```

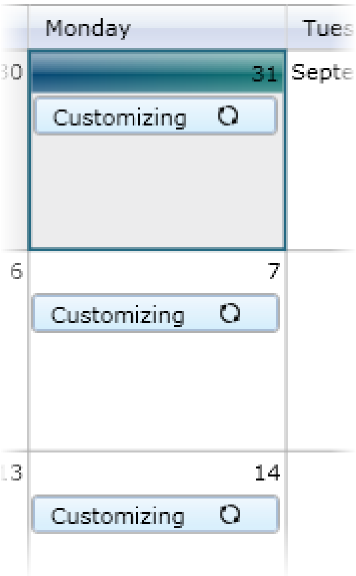
C#

```
schTasks.SetFirstVisibleDate(appointment1.Start);
```

19.4.4 Recurrence

To make an appointment recur programmatically, create a **RecurrenceRule** instance and assign it to an Appointment RecurrenceRule. The key RecurrenceRule property is **RecurrencePattern.** RecurrencePattern answers questions about when the rule will fire including: how frequently (monthly, weekly, hourly?), must the appointment occur on specific days of the week?, when does the appointment first occur after the initial appointment?, how long does the appointment keep recurring?.

The code example below defines an appointment that recurs on Mondays after the initial appointment.



The code example demonstrates how the RecurrencePattern is setup, assigned to the RecurrenceRule and finally the RecurrenceRule is assigned to the Appointment.

VB

```
' Create a RecurrenceRule. The RecurrencePattern
' assigned to this rule establishes that the appointment
' will recur on Mondays after the initial appointment
Dim pattern As New RecurrencePattern(Nothing, RecurrenceDays.Monday, _
RecurrenceFrequency.Monthly, 1, Nothing, Nothing)
Dim rule As New RecurrenceRule(pattern)

Dim appointment As New Appointment() With { _
.Subject = "Customizing RadScheduler", _
.Start = DateTime.Today.AddHours(3), _
.End = DateTime.Today.AddHours(4), _
.RecurrenceRule = rule}
```

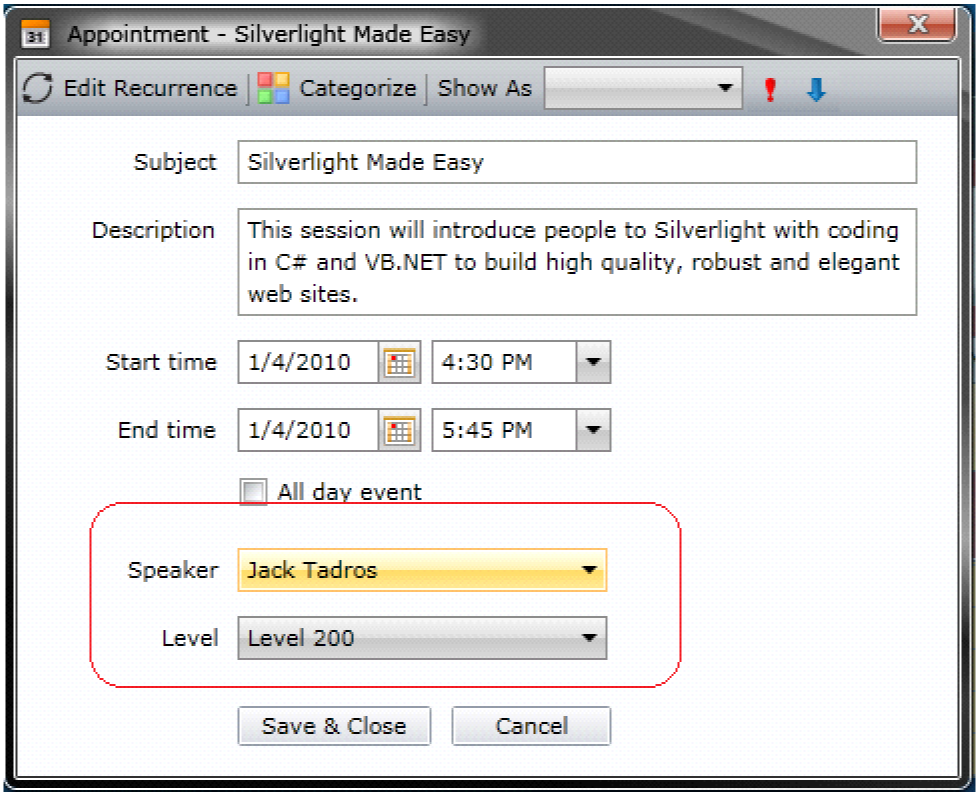
```
// Create a RecurrenceRule. The RecurrencePattern
// assigned to this rule establishes that the appointment
// will recur on Mondays after the initial appointment
RecurrencePattern pattern =
   new RecurrencePattern(null, RecurrenceDays.Monday,
      RecurrenceFrequency.Monthly, 1, null, null);
RecurrenceRule rule = new RecurrenceRule(pattern);

Appointment appointment = new Appointment()
{
   Subject = "Customizing RadScheduler",
   Start = DateTime.Today.AddHours(3),
   End = DateTime.Today.AddHours(4),
   RecurrenceRule = rule
};
```

19.4.5 Resources

RadScheduler provides the ability to define different resource types and also group the appointments by these resource types. The code example below shows how to add resources to the RadScheduler's **ResourceTypes** collection:

```
<telerik:RadScheduler x:Name="scheduler" Grid.Row="1"  DisplayEmptyGroup="True"
        OpenModalDialogs="True" AppointmentTemplate="{StaticResource AppointmentTemplate}"
        AppointmentCreating="scheduler_AppointmentCreating"  ViewMode="Week" MonthViewScrollBarVisibi
          <telerik:RadScheduler.ResourceTypes>
          <telerik:ResourceType Name="Speaker" >
                <telerik:ResourceType.Resources>
                       <telerik:Resource ResourceName="Shoun Petrik"  />
                        <telerik:Resource ResourceName="Jack Tadros" />
                        <telerik:Resource ResourceName="Vladi Pelev"/>
               </telerik:ResourceType.Resources>
          </telerik:ResourceType>
          <telerik:ResourceType Name="Level" >
               <telerik:ResourceType.Resources>
                  <telerik:Resource ResourceName="100" DisplayName="Level 100"  />
                  <telerik:Resource ResourceName="200" DisplayName="Level 200"  />
                  <telerik:Resource  ResourceName="300" DisplayName="Level 300"  />
                  <telerik:Resource  ResourceName="400" DisplayName="Level 400"  />
               </telerik:ResourceType.Resources>
            </telerik:ResourceType>
          </telerik:RadScheduler.ResourceTypes>
</telerik:RadScheduler>
```

As a result to the above code the edit dialog will look like the image shown below. The predefined resources are "Speaker" and "Level". To assign the appointment to certain resources you should select the respective "Speaker" and/or "Level".



Subject: Silverlight Made Easy

Description: This session will introduce people to Silverlight with coding in C# and VB.NET to build high quality, robust and elegant web sites.

Start time: 1/4/2010 4:30 PM

End time: 1/4/2010 5:45 PM

☐ All day event

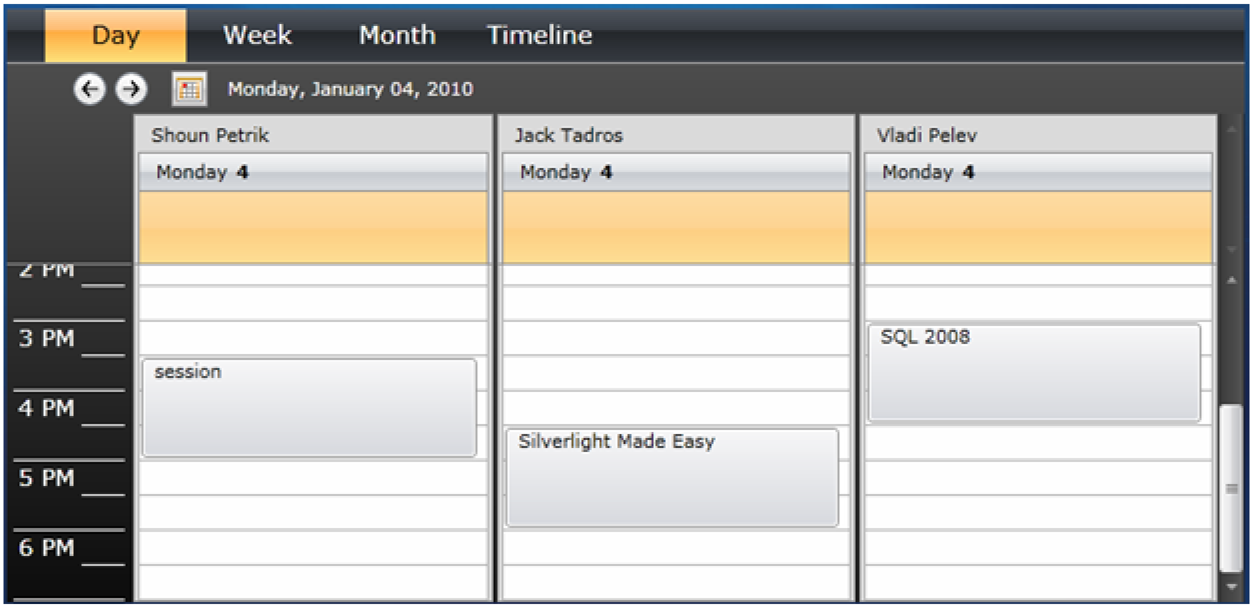
Speaker: Jack Tadros

Level: Level 200

After you have assigned an appointment to certain resources you can group appointments by the resources, to which they have been assigned. The code example below demonstrates how to group the appointments by "Speaker".

```
<telerik:RadScheduler x:Name="scheduler" GroupBy="Speaker" >.
  ...
</ telerik:RadScheduler>
```

As a result, the grouped appointments will appear as shown in the image below:



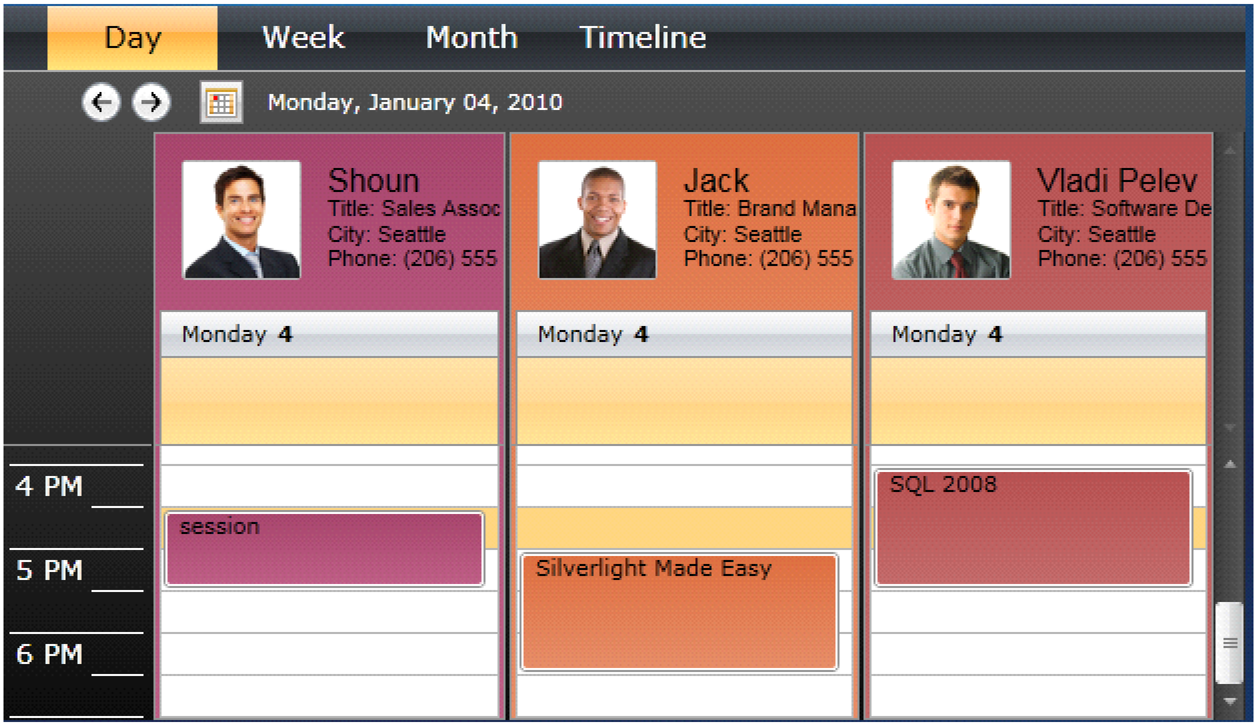
In addition you can customize the look and feel of each resource group. To achieve this you need to create a **ResourceStyleMapping** object for every resource group and set the following properties of this object:

- **ResourceName** - defines the name of the resource that should be associated with this style
- **ResourceBrush** - sets the color of the header of the resource group and to the background of the appointments in this group
- **AppointmentBrush** - sets the color of the appointments in this group. The color is different from the background of the group's header
- **MouseOverAppointmentBrush** - sets the color of the appointments on mouse-over
- **SelectedAppointmentBrush** - sets the color of the appointments when they have been clicked
- **HeaderTemplate** - used to customize the header of the resource group. The value of this property should be of **DataTemplate** type

The code example below shows how to use all the aforementioned properties to customize the appearance of the appointments and the resource groups.

```
<telerik:RadScheduler x:Name="scheduler" GroupBy="Speaker" >
    ...
    <telerik:RadScheduler.ResourceStyleMappings>
      <telerik:ResourceStyleMapping ResourceType="Speaker" ResourceName="Shoun Petrik" Resour
      <telerik:ResourceStyleMapping ResourceType="Speaker" ResourceName="Jack Tadros" Resour
      <telerik:ResourceStyleMapping ResourceType="Speaker" ResourceName="Vladi Pelev" Resource
      <telerik:ResourceStyleMapping ResourceType="Level" ResourceName="200"   ResourceBrush="#
      <telerik:ResourceStyleMapping ResourceType="Level" ResourceName="300"  ResourceBrush="#
      <telerik:ResourceStyleMapping ResourceType="Level" ResourceName="400"  ResourceBrush="#
    </telerik:RadScheduler.ResourceStyleMappings>
</telerik:RadScheduler>
```

The result will be looking like:



19.4.6 Events

RadScheduler has a full set of events to cover all the basic CRUD (Create, Read, Update, Delete) operations and then some. The events are paired so that they occur just before and after the actual operation. The "before" events end in the postfix "ing", e.g. "AppointmentCreating". The "after" events end in the postfix "ed", e.g. "AppointmentCreated".

For example, when an appointment about to be added, the **AppointmentAdding** event fires. Based on the content of the appointment, you can decide to cancel the event. Once the Appointment is added, the **AppointmentAdded** event fires, again passing a reference to the Appointment.

VB

```
Private Sub schTasks_AppointmentAdding(_
ByVal sender As Object, ByVal e As AppointmentAddingEventArgs)
  If e.Appointment.Start <= DateTime.Today Then
    MessageBox.Show("You can only add dates in the future")
    e.Cancel = True
  End If
End Sub

Private Sub schTasks_AppointmentAdded(_
ByVal sender As Object, ByVal e As AppointmentAddedEventArgs)
  If e.Appointment.Subject.StartsWith("UC Class Schedule") Then
    Dim message As String = _
"A reminder email for ""{0}"" will be sent one week before class"
    MessageBox.Show(String.Format(message, e.Appointment.Subject))
  End If
End Sub
```

C#

```
private void schTasks_AppointmentAdding(object sender,
AppointmentAddingEventArgs e)
{
   if (e.Appointment.Start <= DateTime.Today)
   {
      MessageBox.Show("You can only add dates in the future");
      e.Cancel = true;
   }
}

private void schTasks_AppointmentAdded(object sender,
AppointmentAddedEventArgs e)
{
   if (e.Appointment.Subject.StartsWith("UC Class Schedule"))
   {
      string message =
        "A reminder email for \"{0}\" will be sent one week before class";
      MessageBox.Show(string.Format(message, e.Appointment.Subject));
   }
}
```

There are similar event pairs for creating, deleting and editing appointments. Along with these event pairs is a lone **AppointmentSaving** event that fires after pressing the "Save & Close" button of the appointment editing dialog.

19.4.7 Commands

To act on the scheduler programmatically, or to implement your own custom templates and bind to actions directly in XAML, use methods of the **RadSchedulerCommands** class. Each command is a RoutedCommand type and so has an Execute() method. The signature for Execute() includes "parameter" that can be null or another value appropriate to the command. The second parameter is "target", a UIElement that the command executes against. Here is the Execute() method signature:

VB

```
public void Execute(Object parameter, UIElement target)
```

C#

```
public void Execute(object parameter, UIElement target);
```

In the case of RadScheduler commands, the parameter may be an Appointment or a TimeSlot, depending on the context. The UIElement is the scheduler itself. For example, if you want a button click to trigger a new appointment you could call it in code like this:

VB

```
Private Sub Add_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadSchedulerCommands.CreateAppointment.Execute(Nothing, schTasks)
End Sub
```

C#

```
private void Add_Click(object sender, RoutedEventArgs e)
{
   RadSchedulerCommands.CreateAppointment.Execute(null, schTasks);
}
```

The button click fires the command and the Appointment dialog displays, all without directly touching the scheduler. When the first parameter is null, the SelectedTimeSlot is used to set the Start and End time. You can add a TimeSlot instance as the first parameter instead:

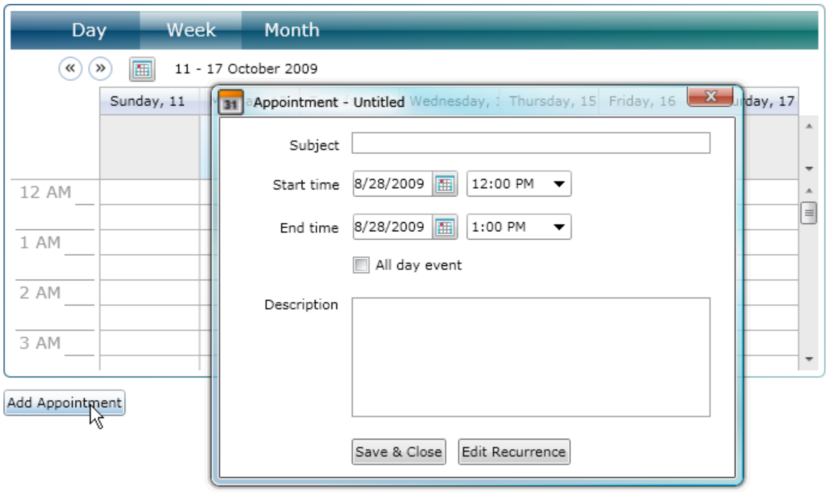
VB

```
Private Sub Add_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim startTime As New DateTime( _
DateTime.Today.Year, DateTime.Today.Month, DateTime.Today.Day, 12, 0, 0)
  Dim timeSlot As New TimeSlot(startTime, startTime.AddHours(1))
  RadSchedulerCommands.CreateAppointment.Execute(timeSlot, schTasks)
End Sub
```

C#

```
private void Add_Click(object sender, RoutedEventArgs e)
{
   DateTime startTime = new DateTime(DateTime.Today.Year,
         DateTime.Today.Month, DateTime.Today.Day, 12, 0, 0);
   TimeSlot timeSlot = new TimeSlot(startTime, startTime.AddHours(1));
   RadSchedulerCommands.CreateAppointment.Execute(timeSlot, schTasks);
}
```

Running this code displays the dialog with the Start and End time set to match the TimeSlot parameter:



Appointment - Untitled

Subject:

Start time: 8/28/2009 12:00 PM

End time: 8/28/2009 1:00 PM

□ All day event

Description:

The EditAppointment(), DeleteAppointment() and SaveAppointment() methods all follow the same format as CreateAppointment().

VB

```
Private Sub Save_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  If schTasks.SelectedAppointment IsNot Nothing Then
    TryCast(schTasks.SelectedAppointment, Appointment).Body &= _
" " & DateTime.Now.ToLongTimeString()

    RadSchedulerCommands.SaveAppointment.Execute( _
schTasks.SelectedAppointment, schTasks)
  End If
End Sub
```

C#

```
private void Save_Click(object sender, RoutedEventArgs e)
{
  if (schTasks.SelectedAppointment != null)
  {
    (schTasks.SelectedAppointment as Appointment).Body +=
      " " + DateTime.Now.ToLongTimeString();

    RadSchedulerCommands.SaveAppointment.Execute(
      schTasks.SelectedAppointment, schTasks);
  }
}
```

Here are the list of all the possible commands you can invoke as of this writing:

- ChangeRecurrenceState
- ChangeTimePickersVisibility
- DecreaseVisibleDateLarge
- DecreaseVisibleDateSmall
- DeleteRecurrenceRule
- EditParentAppointment
- EditRecurrenceRule
- IncreaseVisibleDateLarge
- IncreaseVisibleDateSmall
- SaveRecurrenceRule
- SetDayViewMode
- SetMonthViewMode
- SetWeekViewMode

These commands are especially useful when you want to customize the scheduler. When you add new elements to customized templates, commands allow you to bind functionality to the new elements. The snippet below is from the resources that define a RadScheduler theme. The snippet defines the "Save" button. Notice how the MouseBinding Command attribute is assigned the SaveAppointment command.

```
<Button
    telerik:StyleManager.Theme="{StaticResource SchedulerSystemControlsTheme}"
    telerik:LocalizationManager.ResourceKey="SaveAndCloseCommandText">
  <telerik:CommandManager.InputBindings>
    <telerik:InputBindingCollection>
      <telerik:MouseBinding
          Command="telerik:RadSchedulerCommands.SaveAppointment"
          Gesture="LeftClick" />
    </telerik:InputBindingCollection>
  </telerik:CommandManager.InputBindings>
</Button>
```

See the Customization section of this chapter for more information about customizing scheduler and using themes.

Notes

A RoutedCommand is a WPF construct where a command source, e.g. a Button, can be bound to a command and the command is executed when the command source is activated, e.g. the button is clicked. RoutedCommands are executed from code, but also can be bound to a command source in XAML. RoutedCommand doesn't exist in Silverlight yet, but Telerik has built a RoutedCommand implementation in the Telerik.Windows.Controls namespace.

19.4.8 Drag-and-Drop

19.4.8.1 Drag-and-Drop Overview

If you remember from the "Drag and Drop" chapter, the basic steps for implementing drag-and-drop are:

- Set the RadDragAndDropManager.AllowDrag attached property to True for any "Source" elements that you want to drag.
- Set the RadDragAndDropManager.AllowDrop attached property to True for any "Destination" elements that should receive the dragged elements.
- Handle DragQueryEvent, DropQueryEvent and DropInfoEvent events.

If you want to drag items from some collection container to the scheduler, the steps are the same. To drag from a ListBox, you need to set the RadDragAndDropManager AllowDrag = "True" for every item in the list. ListBox items have an **ItemContainerStyle** property that handles this automatically without having to assign the property to each item individually. Inside the ItemContainerStyle, just add a Style with "ListBoxItem" as its TargetType. Include a single Setter element where the property is RadDragAndDropManager.AllowDrag and the Value is "True". The relevant parts of the XAML are shown below:

```
<ListBox x:Name="listBox">
  <ListBox.ItemContainerStyle>
    <Style TargetType="ListBoxItem">
      <Setter
          Property="dragDrop:RadDragAndDropManager.AllowDrag"
          Value="True" />
    </Style>
  </ListBox.ItemContainerStyle>
  . . .
</ListBox>
```

You also need to set the RadDragAndDropManager **AllowDrop** property for every time slot in the current scheduler view. This is a little more involved because AllowDrop is set programmatically and must be reset every time the view changes. The view can change for a variety of reasons, including the initial load of the scheduler, if the visible range dates change or if the active view changes. When any of these occur, you need to roll through the current set of time slots and set the AllowDrop property. InitializeTimeSlotItems() is a private method that sets the AllowDrop property and will be discussed momentarily.

VB

```
Private Sub OnSchedulerViewPropertyChanged( _
ByVal sender As Object, ByVal e As PropertyChangedEventArgs)
  If e.PropertyName = "VisibleRangeStart" OrElse _
e.PropertyName = "VisibleRangeEnd" Then
    Me.viewChanged = True
  End If
End Sub

Private Sub Scheduler_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Me.viewChanged = True
End Sub

Private Sub Scheduler_ActiveViewDefinitionChanged( _
ByVal sender As Object, ByVal e As EventArgs)
  Me.viewChanged = True
End Sub

Private Sub Scheduler_LayoutUpdated( _
ByVal sender As Object, ByVal e As EventArgs)
  If Me.viewChanged Then
    Me.viewChanged = False
    Me.InitializeTimeSlotItems()
  End If
End Sub
```

C#

```
private void OnSchedulerViewPropertyChanged(object sender, PropertyChangedEventArgs e)
{
   if (e.PropertyName == "VisibleRangeStart" || e.PropertyName == "VisibleRangeEnd")
   {
      this.viewChanged = true;
   }
}

private void Scheduler_Loaded(object sender, RoutedEventArgs e)
{
   this.viewChanged = true;
}

private void Scheduler_ActiveViewDefinitionChanged(object sender, EventArgs e)
{
   this.viewChanged = true;
}

private void Scheduler_LayoutUpdated(object sender, EventArgs e)
{
   if (this.viewChanged)
   {
      this.viewChanged = false;
      this.InitializeTimeSlotItems();
   }
}
```

Setting AllowDrop on each item is made easier by the ChildrenOfType() extension method that collects all of the TimeSlotItem objects from the scheduler. Once you have the TimeSlotItems, iterate and call the SetValue() method to assign AllowDrop.

VB

```
Private Sub InitializeTimeSlotItems()
  Me.timeSlotItems = Me.Scheduler.ChildrenOfType(Of TimeSlotItem)()
  For Each item As TimeSlotItem In Me.timeSlotItems
    item.SetValue(RadDragAndDropManager.AllowDropProperty, True)
  Next item
End Sub
```

C#

```
private void InitializeTimeSlotItems()
{
   this.timeSlotItems = this.Scheduler.ChildrenOfType<TimeSlotItem>();
   foreach (TimeSlotItem item in this.timeSlotItems)
   {
      item.SetValue(RadDragAndDropManager.AllowDropProperty, true);
   }
}
```

19.4.8.2 Drag-and-Drop Walk Through

Now we're going to walk through the complete steps, including the event handlers for the Drag-and-Drop operation. The application simulates a series of servers to be scheduled for backup.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Scheduler**

c) **Telerik.Windows.Controls.Input.dll**

d) **Telerik.Windows.Controls.Navigation.dll**

e) **Telerik.Windows.Data.dll**

f) **Telerik.Windows.Themes.Summer**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the XML name space references below to the UserControl element to support the scheduler, Drag-and-Drop and themes.

```
<UserControl . . .
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation">
```

3) Add the XAML below to replace the main "LayoutRoot" Grid element. This XAML defines the ListBox and the scheduler. The ListBox has the ItemContainerStyle defined as discussed in the previous "Drag-and-Drop Overview" section. The ListBox also contains a relatively simple ItemTemplate that has a single bound TextBlock that will display an ObservableCollection of strings.

```
<Grid x:Name="LayoutRoot">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="100" />
    <ColumnDefinition />
  </Grid.ColumnDefinitions>
  <ListBox x:Name="listBox">
    <ListBox.ItemContainerStyle>
      <Style TargetType="ListBoxItem">
        <Setter
          Property="dragDrop:RadDragAndDropManager.AllowDrag"
          Value="True" />
      </Style>
    </ListBox.ItemContainerStyle>
    <ListBox.ItemTemplate>
      <DataTemplate>
        <TextBlock Text="{Binding}" />
      </DataTemplate>
    </ListBox.ItemTemplate>
  </ListBox>
  <telerik:RadScheduler Grid.Column="1" ViewMode="Day"
      telerik:StyleManager.Theme="Summer" Name="Scheduler">
  </telerik:RadScheduler>
</Grid>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **System.Collections.ObjectModel**

b) **Telerik.Windows.Controls**

c) **Telerik.Windows.Controls.DragDrop**

d) **Telerik.Windows.Controls.Scheduler**

2) Add two private members as shown in the code below: A Boolean "viewChanged" variable to signal between event handlers when the time slots need to be re-initialized with the AllowDrop property setting, and "listData" that contains a collection of strings to display in the ListBox.

VB

```
Private viewChanged As Boolean

Private listData As ObservableCollection(Of String) = _
New ObservableCollection(Of String){ _
"Backup Houdini", _
"Backup Kirby", _
"Backup Maxine", _
"Backup Smoky", _
"Backup Kiwi", _
"Backup Bobo" _
}
```

C#

```
private bool viewChanged;

private ObservableCollection<string> listData = new ObservableCollection<string>{
      "Backup Houdini", "Backup Kirby", "Backup Maxine", "Backup Smoky",
      "Backup Kiwi", "Backup Bobo"
   };
```

3) Change the constructor for the page to add event handlers for the scheduler, add drag-and-drop routed event handlers and assign ListBox data. *Note: be sure to leave the call to InitializeComponent() in place.*

VB

```
Public Sub New()
  InitializeComponent()

  ' add scheduler events
  Me.Scheduler.Loaded += Me.Scheduler_Loaded
  Me.Scheduler.ActiveViewDefinitionChanged += _
Me.Scheduler_ActiveViewDefinitionChanged
  Me.Scheduler.LayoutUpdated += Me.Scheduler_LayoutUpdated
  Me.Scheduler.View.PropertyChanged += Me.OnSchedulerViewPropertyChanged

  ' handle drag-and-drop routed events
  RadDragAndDropManager.AddDragQueryHandler(Me, OnDragQuery)
  RadDragAndDropManager.AddDropQueryHandler(Me, OnDropQuery)
  RadDragAndDropManager.AddDropInfoHandler(Me, OnDropInfo)

  ' assign the ListBox data
  Me.listBox.ItemsSource = Me.listData
End Sub
```

C#

```
public MainPage()
{
   InitializeComponent();

   // add scheduler events
   this.Scheduler.Loaded += this.Scheduler_Loaded;
   this.Scheduler.ActiveViewDefinitionChanged +=
      this.Scheduler_ActiveViewDefinitionChanged;
   this.Scheduler.LayoutUpdated += this.Scheduler_LayoutUpdated;
   this.Scheduler.View.PropertyChanged += this.OnSchedulerViewPropertyChanged;

   // handle drag-and-drop routed events
   RadDragAndDropManager.AddDragQueryHandler(this, OnDragQuery);
   RadDragAndDropManager.AddDropQueryHandler(this, OnDropQuery);
   RadDragAndDropManager.AddDropInfoHandler(this, OnDropInfo);

   // assign the ListBox data
   this.listBox.ItemsSource = this.listData;
}
```

4) Add a private method to assign AllowDrop to each scheduler time slot for the current view.

VB

```
Private Sub InitializeTimeSlotItems()
  Dim timeSlots As IList(Of TimeSlotItem) = _
Me.Scheduler.ChildrenOfType(Of TimeSlotItem)()
  For Each timeSlot As TimeSlotItem In timeSlots
    timeSlot.SetValue(RadDragAndDropManager.AllowDropProperty, True)
  Next timeSlot
End Sub
```

C#

```
private void InitializeTimeSlotItems()
{
   IList<TimeSlotItem> timeSlots = this.Scheduler.ChildrenOfType<TimeSlotItem>();
   foreach (TimeSlotItem timeSlot in timeSlots)
   {
      timeSlot.SetValue(RadDragAndDropManager.AllowDropProperty, true);
   }
}
```

5) Add the scheduler event handlers.

```
Private Sub OnSchedulerViewPropertyChanged( _
ByVal sender As Object, ByVal e As PropertyChangedEventArgs)
  If e.PropertyName = "VisibleRangeStart" OrElse _
e.PropertyName = "VisibleRangeEnd" Then
    Me.viewChanged = True
  End If
End Sub

Private Sub Scheduler_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Me.viewChanged = True
End Sub

Private Sub Scheduler_ActiveViewDefinitionChanged( _
ByVal sender As Object, ByVal e As EventArgs)
  Me.viewChanged = True
End Sub

Private Sub Scheduler_LayoutUpdated( _
ByVal sender As Object, ByVal e As EventArgs)
  If Me.viewChanged Then
    Me.viewChanged = False
    Me.InitializeTimeSlotItems()
  End If
End Sub
```

C#

```
private void OnSchedulerViewPropertyChanged(object sender, PropertyChangedEventArgs e)
{
   if (e.PropertyName == "VisibleRangeStart" || e.PropertyName == "VisibleRangeEnd")
   {
      this.viewChanged = true;
   }
}

private void Scheduler_Loaded(object sender, RoutedEventArgs e)
{
   this.viewChanged = true;
}

private void Scheduler_ActiveViewDefinitionChanged(object sender, EventArgs e)
{
   this.viewChanged = true;
}

private void Scheduler_LayoutUpdated(object sender, EventArgs e)
{
   if (this.viewChanged)
   {
      this.viewChanged = false;
      this.InitializeTimeSlotItems();
   }
}
```

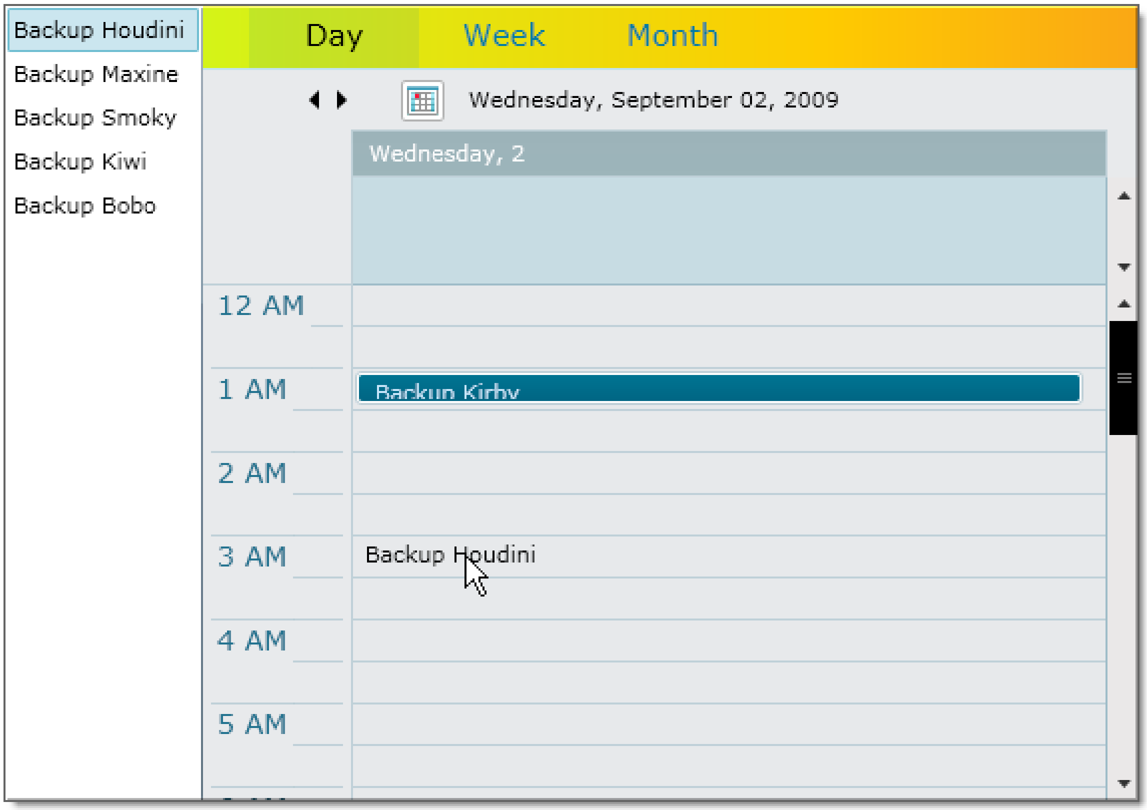
6) Add the drag-and-drop event handlers.

```
Private Sub OnDropInfo(ByVal sender As Object, ByVal e As DragDropEventArgs)
  ' get references to the source list box item and
  ' destination timeslot item
  Dim sourceListBoxItem = TryCast(e.Options.Source, _
System.Windows.Controls.ListBoxItem)
  Dim timeSlotItem = TryCast(e.Options.Destination, TimeSlotItem)

  ' if the drop is complete and we have a valid timeslot
  If e.Options.Status = DragStatus.DropComplete AndAlso _
timeSlotItem IsNot Nothing Then
    ' add a new appointment that matches the timeslot duration
    ' and copy in the listbox item text as the appointment subject
    Me.Scheduler.Appointments.Add(New Appointment() With { _
.Start = timeSlotItem.TimeSlot.Start, _
.End = timeSlotItem.TimeSlot.End, _
.Subject = sourceListBoxItem.FindChildByType(Of TextBlock)().Text})
    ' remove the item from the listbox
    Me.listData.Remove(sourceListBoxItem.Content.ToString())
  End If
End Sub
```

```
private void OnDropInfo(object sender, DragDropEventArgs e)
{
   // get references to the source list box item and
   // destination timeslot item
   var sourceListBoxItem =
     e.Options.Source as System.Windows.Controls.ListBoxItem;
   var timeSlotItem = e.Options.Destination as TimeSlotItem;

   // if the drop is complete and we have a valid timeslot
   if (e.Options.Status == DragStatus.DropComplete && timeSlotItem != null)
   {
      // add a new appointment that matches the timeslot duration
      // and copy in the listbox item text as the appointment subject
      this.Scheduler.Appointments.Add(
         new Appointment()
         {
            Start = timeSlotItem.TimeSlot.Start,
            End = timeSlotItem.TimeSlot.End,
            Subject = sourceListBoxItem.FindChildByType<TextBlock>().Text
         }
      );
      // remove the item from the listbox
      this.listData.Remove(sourceListBoxItem.Content.ToString());
   }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

1) Drag items from the list on the left onto scheduler time slots.
2) When the item is dropped, an appointment should be created automatically, using the text of the ListBox item.
3) Items added to the scheduler should disappear from the ListBox.

Ideas for Extending This Example

- Try dragging from another type of list object, e.g. combo box or tab strip.

19.4.9 Internationalization

19.4.9.1 Using Predefined Cultures

The scheduler **Culture** property has been deprecated. In its place use the **LocalizationManager DefaultCulture** property. You also need to have the scheduler recognize the changes. This currently involves a workaround where the scheduler's template is set to null, then reassigned. For the sake of example, we can bind a List of CultureInfo objects to a RadComboBox ItemsSource.

VB

```
cbCultures.ItemsSource = New List(Of CultureInfo) ( _
New CultureInfo() { CultureInfo.InvariantCulture, _
New CultureInfo("es"), _
New CultureInfo("de"), _
New CultureInfo("fr"), _
New CultureInfo("it"), _
New CultureInfo("tr")})
```

C#

```
cbCultures.ItemsSource = new List<CultureInfo>() {
    CultureInfo.InvariantCulture,
    new CultureInfo("es"),
    new CultureInfo("de"),
    new CultureInfo("fr"),
    new CultureInfo("it"),
    new CultureInfo("tr")
};
```

The combo box DisplayMemberPath points to the CultureInfo DisplayName property.

XAML

```
<telerik:RadComboBox
    x:Name="cbCultures"
    DisplayMemberPath="DisplayName"
    SelectionChanged="cbCultures_SelectionChanged"
    HorizontalAlignment="Left" />
```

When the user clicks on a culture name, the SelectionChanged event handler picks up the selected culture and assigns it to the LocalizationManager.DefaultCulture. The ResetTemplate() method takes care of the workaround that allows the new culture to be displayed.

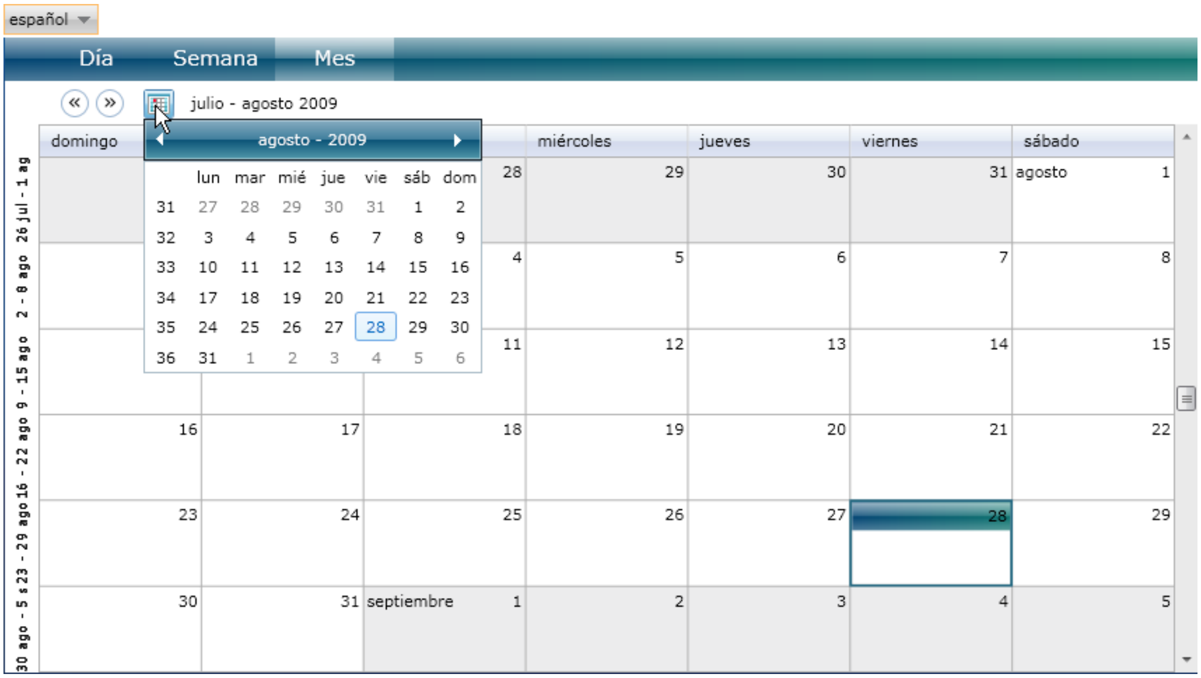
VB

```
Private Sub cbCultures_SelectionChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  LocalizationManager.DefaultCulture = _
CType(cbCultures.SelectedItem, CultureInfo)
  ResetTemplate()
End Sub

Private Sub ResetTemplate()
  Dim template As ControlTemplate = schTasks.Template
  schTasks.Template = Nothing
  schTasks.Template = template
End Sub
```

C#

```
private void cbCultures_SelectionChanged(
   object sender, Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   LocalizationManager.DefaultCulture = (CultureInfo)cbCultures.SelectedItem;
   ResetTemplate();
}

private void ResetTemplate()
{
   ControlTemplate template = schTasks.Template;
   schTasks.Template = null;
   schTasks.Template = template;
}
```

When selected from the combo box, the new culture is reflected in the scheduler and all its child elements, including the drop down date picker.



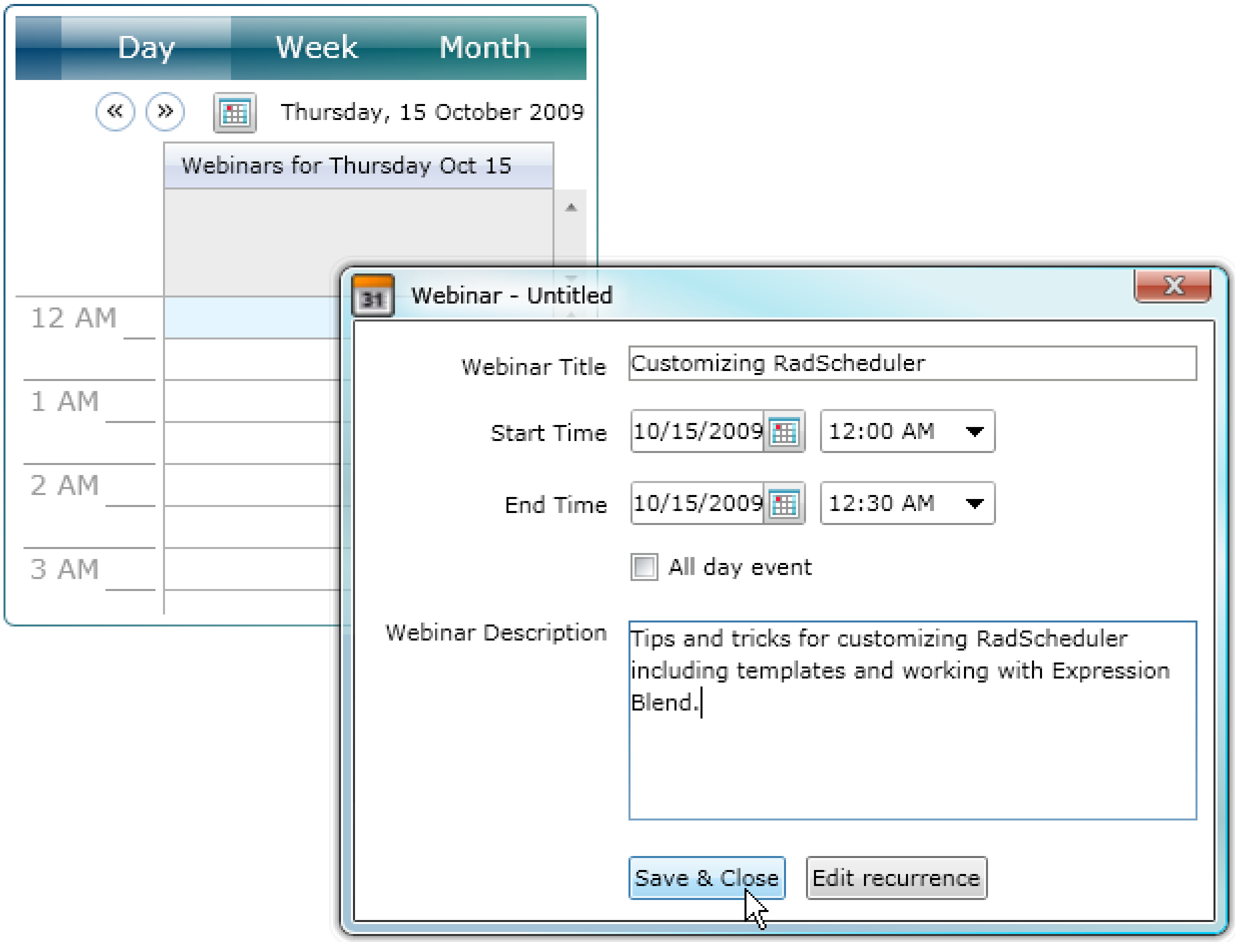
agosto - 2009

	lun	mar	mié	jue	vie	sáb	dom
31	27	28	29	30	31	1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30
36	31	1	2	3	4	5	6

	domingo			miércoles	jueves	viernes	sábado
26 jul - 1 ag			28	29	30	31	agosto 1
2 - 8 ago			4	5	6	7	8
9 - 15 ago			11	12	13	14	15
16 - 22 ago	16	17	18	19	20	21	22
23 - 29 ago	23	24	25	26	27	28	29
30 ago - 5 s	30	31	septiembre 1	2	3	4	5

19.4.9.2 Custom Translations

If there is no predefined translation for the culture you want to use or if you want the scheduler to reflect some particular language, dialect or professional terminology, use a custom resource file and assign it to the scheduler. The screenshot below shows the scheduler language customized for online webinars.



Webinar Title: Customizing RadScheduler

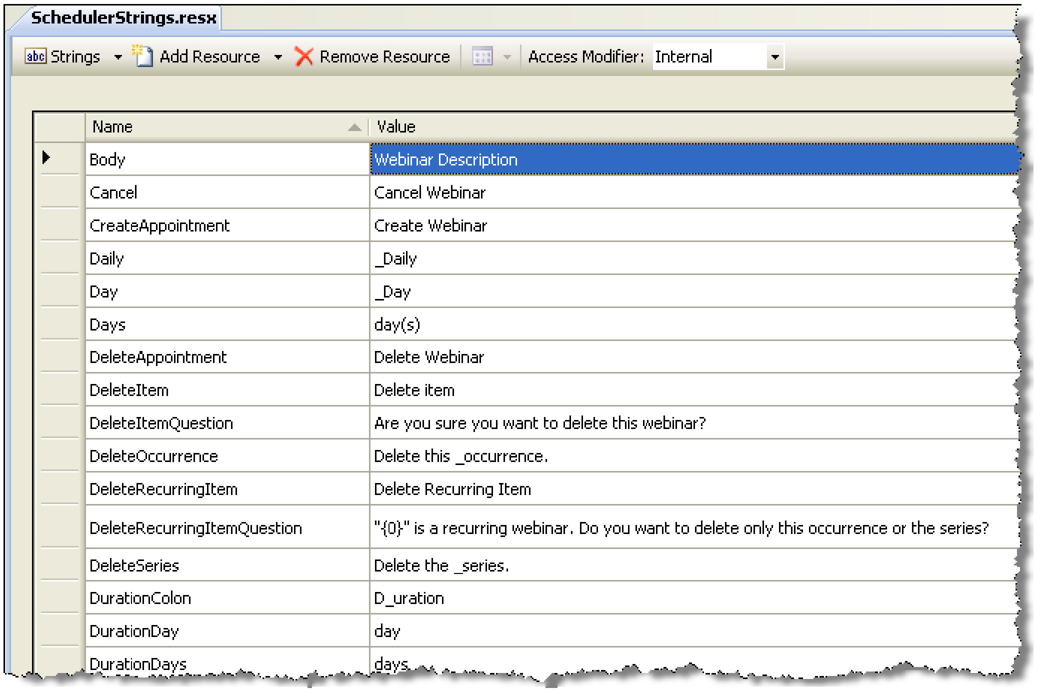
Start Time: 10/15/2009 12:00 AM

End Time: 10/15/2009 12:30 AM

□ All day event

Webinar Description: Tips and tricks for customizing RadScheduler including templates and working with Expression Blend.

To customize the language, you need a resource file populated with specific names that correspond to strings in the scheduler. You can use the SchedulerStrings.resx file found at "\courseware\resources" directory. Open the resource file in Visual Studio and edit the Value columns to use the specific language or terminology.



Access Modifier: Internal

Name	Value
Body	Webinar Description
Cancel	Cancel Webinar
CreateAppointment	Create Webinar
Daily	_Daily
Day	_Day
Days	day(s)
DeleteAppointment	Delete Webinar
DeleteItem	Delete item
DeleteItemQuestion	Are you sure you want to delete this webinar?
DeleteOccurrence	Delete this _occurrence.
DeleteRecurringItem	Delete Recurring Item
DeleteRecurringItemQuestion	"{0}" is a recurring webinar. Do you want to delete only this occurrence or the series?
DeleteSeries	Delete the _series.
DurationColon	D_uration
DurationDay	day
DurationDays	days

In the code behind, you need to assign the resource file's ResourceManager to the LocalizationManager. DefaultResourceManager. Currently, you also need to perform the work-around explained earlier in this chapter to reset the scheduler Template.

VB

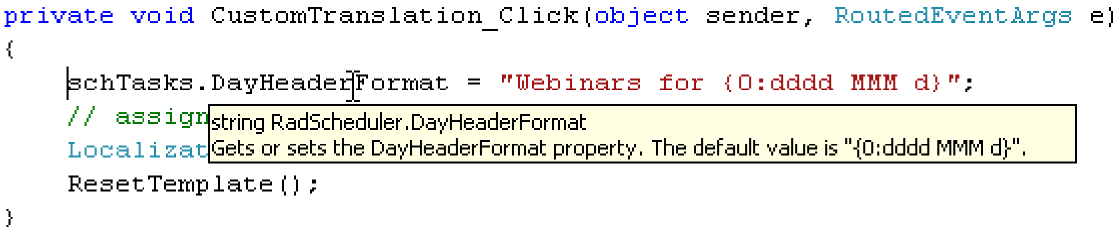
```
Private Sub CustomTranslation_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  schTasks.DayHeaderFormat = "Webinars for {0:dddd MMM d}"
  ' assign the custom resource file to the LocalizationManager
  LocalizationManager.DefaultResourceManager = SchedulerStrings.ResourceManager
  ResetTemplate()
End Sub

Private Sub ResetTemplate()
  Dim template As ControlTemplate = schTasks.Template
  schTasks.Template = Nothing
  schTasks.Template = template
End Sub
```

C#

```
private void CustomTranslation_Click(object sender, RoutedEventArgs e)
{
   schTasks.DayHeaderFormat = "Webinars for {0:dddd MMM d}";
   // assign the custom resource file to the LocalizationManager
   LocalizationManager.DefaultResourceManager = SchedulerStrings.ResourceManager;
   ResetTemplate();
}

private void ResetTemplate()
{
   ControlTemplate template = schTasks.Template;
   schTasks.Template = null;
   schTasks.Template = template;
}
```

Tip!

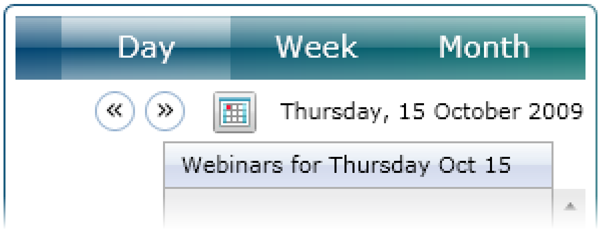
Notice that we're using the scheduler DayHeaderFormat property to set the language at the top of each day. There are several similar properties appended with "Format" that you can use to customize parts of the scheduler. The tool tip for each property provides the default format you can use as a starting point. The screenshot below shows the tool tip displayed in the Visual Studio editor.



```
private void CustomTranslation_Click(object sender, RoutedEventArgs e)
{
    schTasks.DayHeaderFormat = "Webinars for {0:dddd MMM d}";
    // assign
    Localizat
    ResetTemplate();
}
```

string RadScheduler.DayHeaderFormat
Gets or sets the DayHeaderFormat property. The default value is "{0:dddd MMM d}".

This particular property setting results in the heading shown below that reads "Webinars for Thursday Oct 15".



19.5 Binding

19.5.1 Basic Binding

RadScheduler can bind to any **IAppointment** implementation. Telerik provides an **Appointment** class that serves nicely where you don't need any custom fields. To setup binding, add Appointment instances to an ObservableCollection, then assign the collection to the scheduler **AppointmentsSource** property. The walk through below demonstrates just that.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Scheduler**

c) **Telerik.Windows.Controls.Input.dll**

d) **Telerik.Windows.Controls.Navigation.dll**

e) **Telerik.Windows.Data.dll**

f) **Telerik.Windows.Themes.Vista**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace references to the Telerik.Windows.Controls and Telerik.Windows.Controls.Scheduler assemblies. Also add a Loaded event handler.

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .
Loaded="UserControl_Loaded">
. . .
</UserControl>
```

3) Drag a RadScheduler from the Toolbox to a point inside the main "LayoutRoot" grid. Set the name to "schTasks" and the Theme to "Vista".

```
<Grid x:Name="LayoutRoot">
  <telerik:RadScheduler
    x:Name="schTasks"
    telerik:StyleManager.Theme="Vista" />
</Grid>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these name spaces:

a) **System.Collections.ObjectModel** (Supports the ObservableCollection class)

b) **Telerik.Windows.Controls.Scheduler**

2) Add the code below to the Loaded event handler.

This code creates the series of Appointment objects and adds them to the ObservableCollection. The populated collection is assigned to the RadScheduler AppointmentsSource.

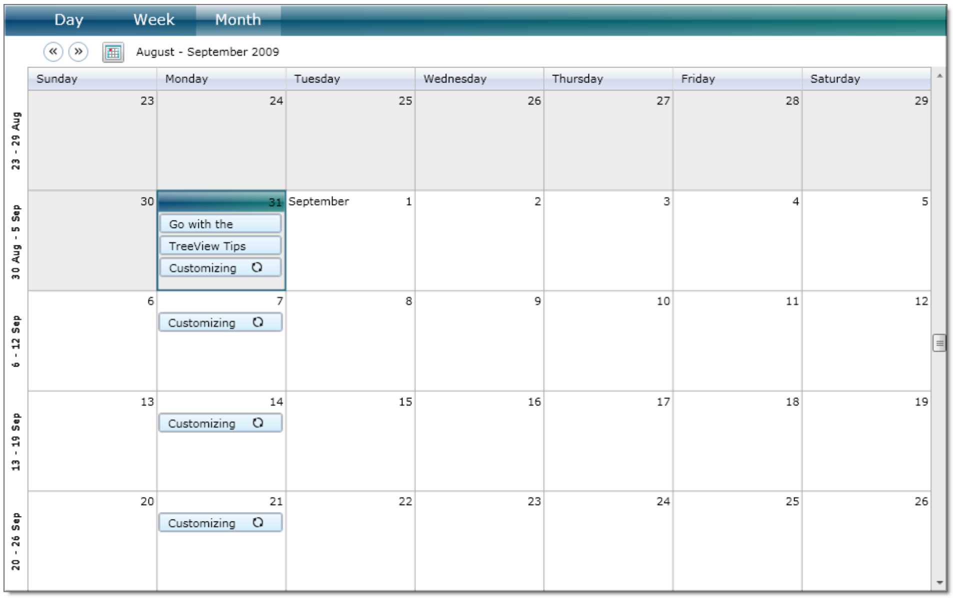
```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim webinars As ObservableCollection(Of Appointment) = _
New ObservableCollection(Of Appointment)()

  webinars.Add(New Appointment() With { _
.Subject = "Go with the CoverFlow", _
.Start = DateTime.Today, _
.End = DateTime.Today.AddHours(1)})

  webinars.Add(New Appointment() With { _
.Subject = "TreeView Tips and Tricks", _
.Start = DateTime.Today.AddHours(1), _
.End = DateTime.Today.AddHours(2)})

  Dim pattern As New RecurrencePattern(Nothing, _
RecurrenceDays.Monday, RecurrenceFrequency.Monthly, 1, Nothing, Nothing)
  Dim rule As New RecurrenceRule(pattern)

  webinars.Add(New Appointment() With { _
.Subject = "Customizing RadScheduler", _
.Start = DateTime.Today.AddHours(3), _
.End = DateTime.Today.AddHours(4), _
.RecurrenceRule = rule})

  schTasks.AppointmentsSource = webinars
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   ObservableCollection<Appointment> webinars =
      new ObservableCollection<Appointment>();

   webinars.Add(new Appointment()
   {
      Subject = "Go with the CoverFlow",
      Start = DateTime.Today,
      End = DateTime.Today.AddHours(1)
   });

   webinars.Add(new Appointment()
   {
      Subject = "TreeView Tips and Tricks",
      Start = DateTime.Today.AddHours(1),
      End = DateTime.Today.AddHours(2)
   });

   RecurrencePattern pattern =
      new RecurrencePattern(null, RecurrenceDays.Monday,
         RecurrenceFrequency.Monthly, 1, null, null);
   RecurrenceRule rule = new RecurrenceRule(pattern);

   webinars.Add(new Appointment()
   {
      Subject = "Customizing RadScheduler",
      Start = DateTime.Today.AddHours(3),
      End = DateTime.Today.AddHours(4),
      RecurrenceRule = rule
   });

   schTasks.AppointmentsSource = webinars;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



August - September 2009

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
23 - 29 Aug	23	24	25	26	27	28	29
30 Aug - 5 Sep	30	31 Go with the TreeView Tips Customizing	September 1	2	3	4	5
6 - 12 Sep	6	7 Customizing	8	9	10	11	12
13 - 19 Sep	13	14 Customizing	15	16	17	18	19
20 - 26 Sep	20	21 Customizing	22	23	24	25	26

Test Application Features

1) Verify that you can change the appointment information and that the changes persist.

2) Verify that the recurring appointment is listed for every Monday after the initial appointment.

19.5.2 Custom Appointments

Telerik provides an abstract **AppointmentBase** class that includes all the basics, i.e. "Subject", "Start", "End". AppointmentBase also has INotifyPropertyChanged support baked in. Once you have an AppointmentBase descendant, you can create an ObservableCollection of your custom appointment objects.

Here's a sample AppointmentBase descendant called WebinarAppointment that adds a single new property "Presenter". Notice that plumbing for Start, End, Subject, Recurrence, IsAllDayEvent and TimeZone properties is already taken care of by the AppointmentBase class. Also notice that the call to the OnPropertyChanged() method inside the "Presenter" property setter. Likewise, the CopyFrom() IAppointment method calls the base method and only adds the new "Presenter" property assignment.

```
Public Class WebinarAppointment
  Inherits AppointmentBase
  #Region "properties"

  Private _presenter As String
  Public Property Presenter() As String
    Get
      Return _presenter
    End Get

    Set(ByVal value As String)
      If _presenter <> value Then
        _presenter = value
        OnPropertyChanged("Presenter")
      End If
    End Set
  End Property

  #End Region

  #Region "IAppointment methods"

  ' makes a new appointment with all the properties of this
  ' appointment
  Public Overrides Function Copy() As IAppointment
    Dim app As IAppointment = New WebinarAppointment()
    app.CopyFrom(Me)
    Return app
  End Function

  ' copy all the properties from another appointment
  Public Overrides Sub CopyFrom(ByVal other As IAppointment)
    MyBase.CopyFrom(other)
    Dim appointment = TryCast(other, WebinarAppointment)
    If appointment IsNot Nothing Then
      Presenter = appointment.Presenter
    End If
  End Sub
  #End Region

End Class
```

C#

```
public class WebinarAppointment : AppointmentBase
{
    #region properties

    private string _presenter;
    public string Presenter
    {
        get
        {
            return _presenter;
        }

        set
        {
            if (_presenter != value)
            {
                _presenter = value;
                OnPropertyChanged("Presenter");
            }
        }
    }

    #endregion

    #region IAppointment methods

    // makes a new appointment with all the properties of this
    // appointment
    public override IAppointment Copy()
    {
        IAppointment app = new WebinarAppointment();
        app.CopyFrom(this);
        return app;
    }

    // copy all the properties from another appointment
    public override void CopyFrom(IAppointment other)
    {
        base.CopyFrom(other);
        var appointment = other as WebinarAppointment;
        if (appointment != null)
        {
            Presenter = appointment.Presenter;
        }
    }
    #endregion

}
```

Like the previous example from the "Basic Binding" section that binds Appointment class instances, WebinarAppointment instances are created and added to an ObservableCollection, then assigned to the scheduler's AppointmentsSource. The major difference here, other than the use of WebinarAppointment instead of Appointment, is the inclusion of the new "Presenter" property.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' create WebinarAppointment custom objects and add
  ' them to the observable collection
  Dim webinars As ObservableCollection(Of WebinarAppointment) = _
New ObservableCollection(Of WebinarAppointment)()

  webinars.Add(New WebinarAppointment() With { _
.Subject = "Go with the CoverFlow", _
.Presenter = "Hristo Borisov", _
.Start = DateTime.Today, _
.End = DateTime.Today.AddHours(1)})

  webinars.Add(New WebinarAppointment() With { _
.Subject = "TreeView Tips and Tricks", _
.Presenter = "Valeri Hristov", _
.Start = DateTime.Today.AddHours(1), _
.End = DateTime.Today.AddHours(2)})

  ' Create a RecurrenceRule. The RecurrencePattern
  ' assigned to this rule establishes that the appointment
  ' will recur on Mondays after the initial appointment
  Dim pattern As New RecurrencePattern(Nothing, _
RecurrenceDays.Monday, RecurrenceFrequency.Monthly, 1, Nothing, Nothing)
  Dim rule As New RecurrenceRule(pattern)

  webinars.Add(New WebinarAppointment() With { _
.Subject = "Customizing RadScheduler", _
.Presenter = "Rosi F", _
.Start = DateTime.Today.AddHours(3), _
.End = DateTime.Today.AddHours(4), _
.RecurrenceRule = rule})

  schTasks.AppointmentsSource = webinars
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   // create WebinarAppointment custom objects and add
   // them to the observable collection
   ObservableCollection<WebinarAppointment> webinars =
      new ObservableCollection<WebinarAppointment>();

   webinars.Add(new WebinarAppointment()
   {
      Subject = "Go with the CoverFlow",
      Presenter = "Hristo Borisov",
      Start = DateTime.Today,
      End = DateTime.Today.AddHours(1)
   });

   webinars.Add(new WebinarAppointment()
   {
      Subject = "TreeView Tips and Tricks",
      Presenter = "Valeri Hristov",
      Start = DateTime.Today.AddHours(1),
      End = DateTime.Today.AddHours(2)
   });

   // Create a RecurrenceRule. The RecurrencePattern
   // assigned to this rule establishes that the appointment
   // will recur on Mondays after the initial appointment
   RecurrencePattern pattern =
      new RecurrencePattern(null, RecurrenceDays.Monday,
         RecurrenceFrequency.Monthly, 1, null, null);
   RecurrenceRule rule = new RecurrenceRule(pattern);

   webinars.Add(new WebinarAppointment()
   {
      Subject = "Customizing RadScheduler",
      Presenter = "Rosi F",
      Start = DateTime.Today.AddHours(3),
      End = DateTime.Today.AddHours(4),
      RecurrenceRule = rule
   });

   schTasks.AppointmentsSource = webinars;
}
```

19.6 Customization

In the previous Binding section of this chapter we created a new bit of data called "Presenter". Its all very good that the data is somehow available, but how do you get this data to show in the scheduler? As with all the Silverlight controls, control templates let you compose arbitrary arrangements of elements and bind them to data. The scheduler provides access to the control template using a custom theme. Custom themes are implemented using XAML resource files stored in a separate assembly. The general steps that display a new bound data property in a custom theme are:

- Create a new Silverlight Class Library project. The class library will hold a XAML file that contains all the resources for the scheduler and a Telerik.Windows.Controls.Theme descendant.
- Edit the XAML to alter the specific template resource you're interested in.
- Reference the new theme in a Silverlight application.

You would use these steps for any customization of the scheduler, such as including additional properties or changing the layout and appearance of any scheduler aspect.

In this next walk through, you will create a new "MyCustomTheme" class. By altering the "EditAppointmentTemplate" of the theme XAML file, you will include an additional row containing a TextBox bound to the "Presenter" property.

Build the Silverlight Class Library

1) From the Visual Studio menu choose File > New > Project..., select the Silverlight project type, then select the Silverlight Class Library template. Name the project "CustomTheme" and click the **OK** button.

2) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add references to these assemblies:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Scheduler**

c) **Telerik.Windows.Controls.Input.dll**

d) **Telerik.Windows.Controls.Navigation.dll**

e) **Telerik.Windows.Data.dll**

3) In the Solution Explorer, rename the default "Class1" file to "MyCustomTheme".

4) Open the class file for editing and make the following changes:

a) Add the **Telerik.Windows.Controls** namespace to the "Imports" (VB) or "using" (C#) section of code.

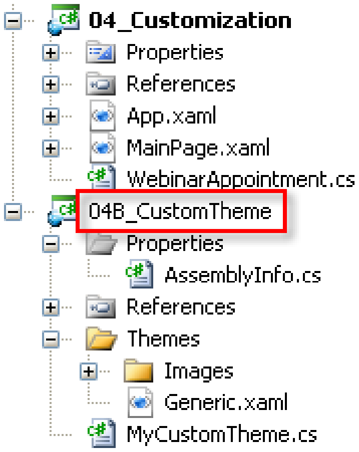
b) Replace "Class1" with the code below. Also replace the "path" string "<My Project Name>" with the name of your Silverlight Class Library project.

```
Public Class MyCustomTheme
  Inherits Theme
  Public Sub New()
    'path the XML that defines this theme
    Dim path As String = "/<My Project Name>;component/themes/Generic.xaml"

    'assign the source file for this theme
    Me.Source = New Uri(path, UriKind.RelativeOrAbsolute)
  End Sub
End Class
```

```
public class MyCustomTheme : Theme
{
   public MyCustomTheme()
   {
      // path the XML that defines this theme
      string path =
         "/<My Project Name>;component/themes/Generic.xaml";

      // assign the source file for this theme
      this.Source =
         new Uri(path, UriKind.RelativeOrAbsolute);
   }
}
```

Notes

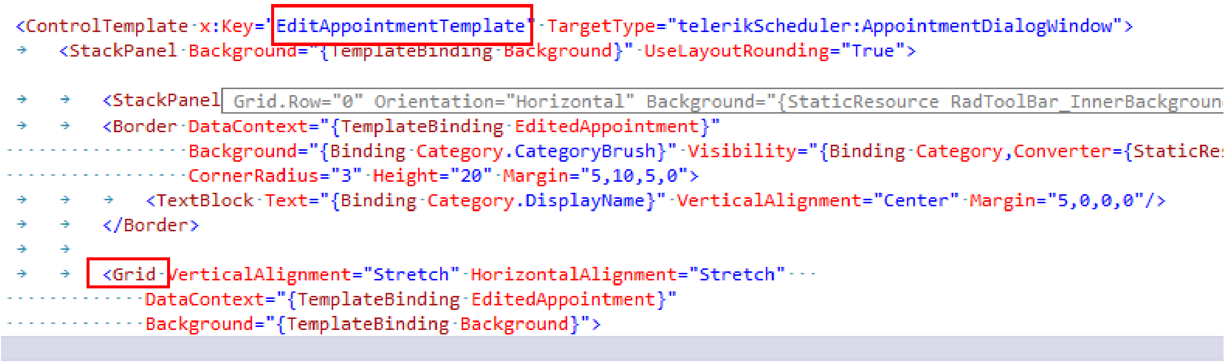
In the sample projects that accompany this courseware, the theme project is called "04B_CustomTheme", so the correct path for the XAML resource file is "04B_CustomTheme; component/themes/Generic.xaml". See the screenshot of the Solution Explorer below for reference.



5) Drag a copy of the "\Themes" folder and its contents from the "\courseware\resources" directory into your Silverlight Class Library. Note that this file can also be obtained from the project source available from download page of your Telerik account.

6) Open the Generic.xaml file for editing.

7) Locate the "EditAppointmentTemplate" ControlTemplate. We will be editing the grid within this element.



```
<ControlTemplate x:Key="EditAppointmentTemplate" TargetType="telerikScheduler:AppointmentDialogWindow">
    <StackPanel Background="{TemplateBinding Background}" UseLayoutRounding="True">

        <StackPanel Grid.Row="0" Orientation="Horizontal" Background="{StaticResource RadToolBar_InnerBackgroun
        <Border DataContext="{TemplateBinding EditedAppointment}"
                Background="{Binding Category.CategoryBrush}" Visibility="{Binding Category,Converter={StaticRe
                CornerRadius="3" Height="20" Margin="5,10,5,0">
            <TextBlock Text="{Binding Category.DisplayName}" VerticalAlignment="Center" Margin="5,0,0,0"/>
        </Border>

        <Grid VerticalAlignment="Stretch" HorizontalAlignment="Stretch"
            DataContext="{TemplateBinding EditedAppointment}"
            Background="{TemplateBinding Background}">
```

8) Set the Grid **Height** attribute to "350". This will grow the edit appointment dialog slightly to accommodate the new row you are about to add.

9) Find the **Grid.RowDefinitions** element within the grid. Replace the row definitions with the XAML below: This will provide one extra row.

```
<Grid.RowDefinitions>
   <RowDefinition Height="Auto" />
   <RowDefinition Height="Auto" />
   <RowDefinition Height="Auto" />
   <RowDefinition Height="Auto" />
   <RowDefinition Height="Auto" />
   <RowDefinition Height="Auto" />
   <RowDefinition Height="Auto" />
</Grid.RowDefinitions>
```

10)Locate the **TextBox** element named "Subject", i.e., 'x:Name="Subject"'. Add the following XAML just below this TextBox element.

Notice in the XAML below that the TextBox is bound to the Presenter property.

```
<!--new controls to represent the "Presenter" property-->
<TextBlock Text="Presenter:" Grid.Row="1"
      Style="{StaticResource FormElementTextBlockStyle}"
      telerik:StyleManager.Theme="{StaticResource Theme}" />
<TextBox
      telerik:StyleManager.Theme="{StaticResource Theme}"
      x:Name="Presenter" Grid.Row="1" Grid.Column="1"
      Text="{Binding Presenter, Mode=TwoWay}"
      MaxLength="255" Margin="10, 10, 20, 2" />
```

11)Build the Silverlight Class Library project.

Build the Silverlight Application

Project Setup

1) In the Solution Explorer, right-click the solution and select **Add > New Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

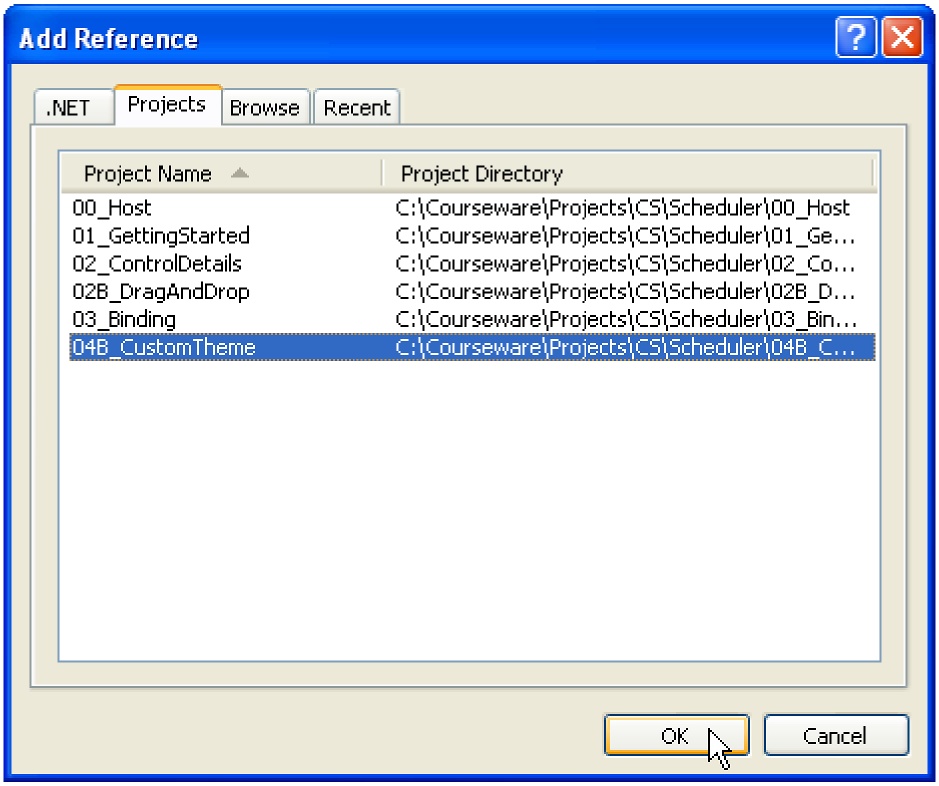
2) In the "New Silverlight Application" dialog, make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web project Type option is checked. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add references to these assemblies:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Scheduler**

c) Your Silverlight Class Library assembly. You can find your custom theme project in the Projects tab of the Add Reference dialog.



Project Name	Project Directory
00_Host	C:\Courseware\Projects\CS\Scheduler\00_Host
01_GettingStarted	C:\Courseware\Projects\CS\Scheduler\01_Ge...
02_ControlDetails	C:\Courseware\Projects\CS\Scheduler\02_Co...
02B_DragAndDrop	C:\Courseware\Projects\CS\Scheduler\02B_D...
03_Binding	C:\Courseware\Projects\CS\Scheduler\03_Bin...
04B_CustomTheme	C:\Courseware\Projects\CS\Scheduler\04B_C...

4) In the Solution Explorer, create a WebinarAppointment class file and copy the code below into the file. Note: This is the exact same WebinarAppointment file created in the Binding section of this chapter. Feel free to copy this file over to save time.

```
Imports System
Imports Telerik.Windows.Controls.Scheduler

Namespace _04_Customization
  Public Class WebinarAppointment
    Inherits AppointmentBase

    Private _presenter As String
    Public Property Presenter() As String
      Get
        Return _presenter
      End Get

      Set(ByVal value As String)
        If _presenter <> value Then
          _presenter = value
          OnPropertyChanged("Presenter")
        End If
      End Set
    End Property

    ' makes a new appointment with all the properties of this
    ' appointment
    Public Overrides Function Copy() As IAppointment
      Dim app As IAppointment = New WebinarAppointment()
      app.CopyFrom(Me)
      Return app
    End Function

    ' copy all the properties from another appointment
    Public Overrides Sub CopyFrom(ByVal other As IAppointment)
      MyBase.CopyFrom(other)
      Dim appointment = TryCast(other, WebinarAppointment)
      If appointment IsNot Nothing Then
        Presenter = appointment.Presenter
      End If
    End Sub

  End Class
End Namespace
```

```
using System;
using Telerik.Windows.Controls.Scheduler;

namespace _04_Customization
{
   public class WebinarAppointment : AppointmentBase
   {
      private string _presenter;
      public string Presenter
      {
         get { return _presenter; }

         set
         {
            if (_presenter != value)
            {
               _presenter = value;
               OnPropertyChanged("Presenter");
            }
         }
      }

      // makes a new appointment with all the properties of this
      // appointment
      public override IAppointment Copy()
      {
         IAppointment app = new WebinarAppointment();
         app.CopyFrom(this);
         return app;
      }

      // copy all the properties from another appointment
      public override void CopyFrom(IAppointment other)
      {
         base.CopyFrom(other);
         var appointment = other as WebinarAppointment;
         if (appointment != null)
         {
            Presenter = appointment.Presenter;
         }
      }
   }
}
```

XAML Editing

1) Open MainPage.xaml for editing.

2) Add the XML namespace references below to the UserControl element to support the scheduler and your custom theme. Also add a Loaded event handler.

   Important Note: In the XML namespace "customTheme", replace the portion to the right of the "=" with your own custom theme assembly reference. Put the cursor to the right of the equal sign and press Ctrl-Spacebar to get a list of available assemblies. Select your custom theme assembly reference from the list.

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
Loaded="UserControl_Loaded">
```

3) Add a UserControl.Resources element. In the Resources element, add a reference to your custom theme.

```
<UserControl.Resources>
   <customTheme:MyCustomTheme x:Key="theme" />
</UserControl.Resources>
```

4) Drag a RadScheduler control from the Toolbox to a point inside the main "LayoutRoot" grid element. Name the scheduler "schTasks" and set the Theme to "{StaticResource theme}".

Just to recap, "{StaticResource theme}" points back to the UserControl.Resources element with key="theme". That element points back to "customTheme", a reference to your Silverlight Class Library and its "MyCustomTheme" class.

XAML

```
<telerik:RadScheduler x:Name="schTasks"
 telerik:StyleManager.Theme="{StaticResource theme}" />
```

Notes

An alternative to the last two steps would be to assign the theme in code using the StyleManager.SetTheme() method. SetTheme() takes a reference to the scheduler and an instance of the custom theme class.

VB

```
' alternative to defining in XAML
Telerik.Windows.Controls.StyleManager.SetTheme(schWebinars, New MyCustomTheme())
```

C#

```
// alternative to defining in XAML
Telerik.Windows.Controls.StyleManager.SetTheme(schWebinars, new MyCustomTheme());
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **System.Collections.ObjectModel**

b) **Telerik.Windows.Controls.Scheduler**

2) Create an ObservableCollection of WebinarAppointment objects and assign it to the scheduler AppointmentsSource.

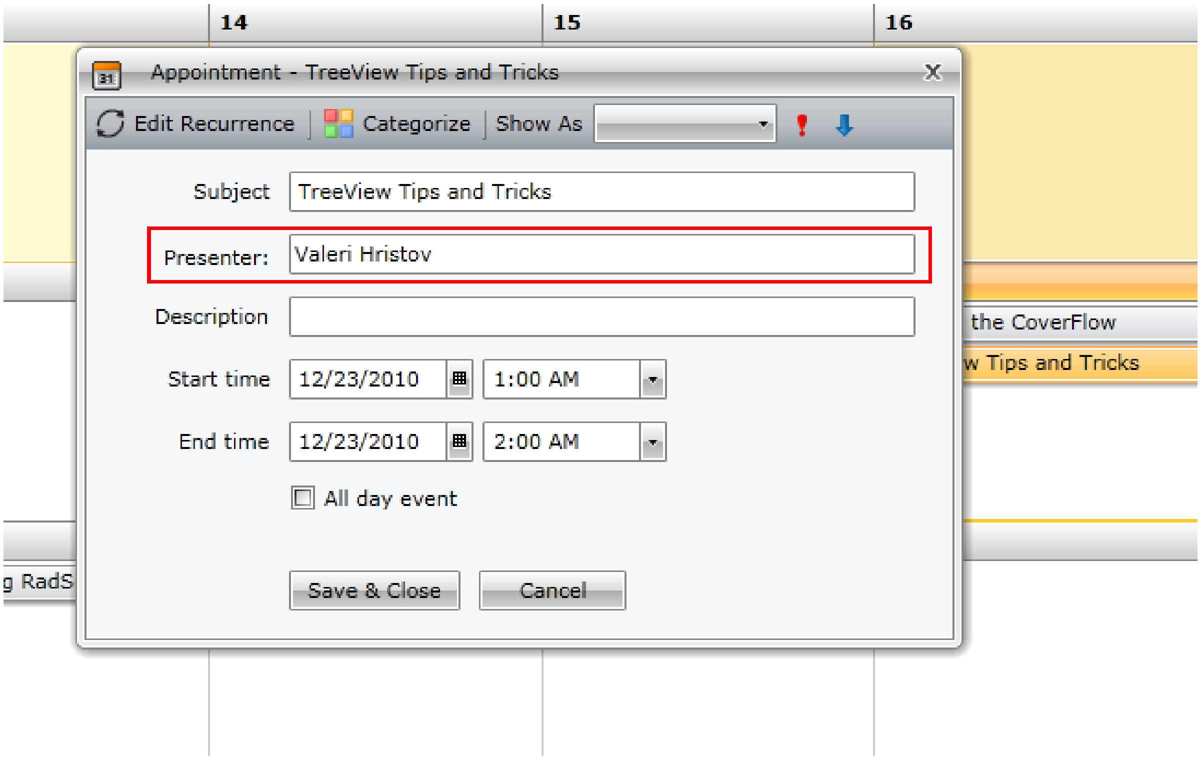
This code is replicated from the Binding section regarding Custom Appointments.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' create WebinarAppointment custom objects and add
  ' them to the observable collection
  Dim webinars As ObservableCollection(Of WebinarAppointment) = _
New ObservableCollection(Of WebinarAppointment)()

  webinars.Add(New WebinarAppointment() With { _
.Subject = "Go with the CoverFlow", _
.Presenter = "Hristo Borisov", _
.Start = DateTime.Today, _
.End = DateTime.Today.AddHours(1)})

  webinars.Add(New WebinarAppointment() With { _
.Subject = "TreeView Tips and Tricks", _
.Presenter = "Valeri Hristov", _
.Start = DateTime.Today.AddHours(1), _
.End = DateTime.Today.AddHours(2)})

  ' Create a RecurrenceRule. The RecurrencePattern
  ' assigned to this rule establishes that the appointment
  ' will recur on Mondays after the initial appointment
  Dim pattern As New RecurrencePattern( _
Nothing, RecurrenceDays.Monday, RecurrenceFrequency.Monthly, 1, Nothing, Nothing)
  Dim rule As New RecurrenceRule(pattern)

  webinars.Add(New WebinarAppointment() With { _
.Subject = "Customizing RadScheduler", _
.Presenter = "Rosi F", _
.Start = DateTime.Today.AddHours(3), _
.End = DateTime.Today.AddHours(4), _
.RecurrenceRule = rule})

  schTasks.AppointmentsSource = webinars
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
  // create WebinarAppointment custom objects and add
  // them to the observable collection
  ObservableCollection<WebinarAppointment> webinars =
    new ObservableCollection<WebinarAppointment>();

  webinars.Add(new WebinarAppointment()
  {
    Subject = "Go with the CoverFlow",
    Presenter = "Hristo Borisov",
    Start = DateTime.Today,
    End = DateTime.Today.AddHours(1)
  });

  webinars.Add(new WebinarAppointment()
  {
    Subject = "TreeView Tips and Tricks",
    Presenter = "Valeri Hristov",
    Start = DateTime.Today.AddHours(1),
    End = DateTime.Today.AddHours(2)
  });

  // Create a RecurrenceRule. The RecurrencePattern
  // assigned to this rule establishes that the appointment
  // will recur on Mondays after the initial appointment
  RecurrencePattern pattern =
    new RecurrencePattern(null, RecurrenceDays.Monday,
      RecurrenceFrequency.Monthly, 1, null, null);
  RecurrenceRule rule = new RecurrenceRule(pattern);

  webinars.Add(new WebinarAppointment()
  {
    Subject = "Customizing RadScheduler",
    Presenter = "Rosi F",
    Start = DateTime.Today.AddHours(3),
    End = DateTime.Today.AddHours(4),
    RecurrenceRule = rule
  });

  schTasks.AppointmentsSource = webinars;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Subject: TreeView Tips and Tricks

Presenter: Valeri Hristov

Description:

Start time: 12/23/2010 1:00 AM

End time: 12/23/2010 2:00 AM

□ All day event

The edit appointment dialog appears with the new "Presenter:" title TextBlock and TextBox bound to the Presenter property of the WebinarAppointment object.

Test Application Features

1) Verify that the new Presenter fields appear on the appointment editing dialog.
2) Make changes to the presenter, save, then re-open the dialog to verify the changes.

19.7 Wrap Up

In this chapter you covered a wide range of material in the process of learning how to use RadScheduler to best effect. You started out by using RadScheduler in a simple project that created a single appointment. You learned how to switch the view between month, week and day. You also learned how to create and configure appointments and the role of IAppointment and AppointmentBase in building custom appointment classes. You selected appointments programmatically and responded to user selections in the scheduler. You worked with RecurrenceRule and RecurrencePattern classes to specify the recurrence behavior of an appointment. You looked briefly at scheduler commands and how they can be used programmatically or to add functionality to controls defined in XAML. You learned how to drag-and-drop ListBox items into time slots using the RadDragAndDropManager control. You also learned how to localize the scheduler using the LocalizationManager and you also saw how to provide a custom translation using resource files.

You performed basic data binding to a collection of the supplied Appointment class and you also worked with building and binding to custom appointment objects. Finally, you built a custom Theme for RadScheduler and modified the Appointment Editing dialog user interface.

Part XX

Gauges

20 Gauges

20.1 Objectives

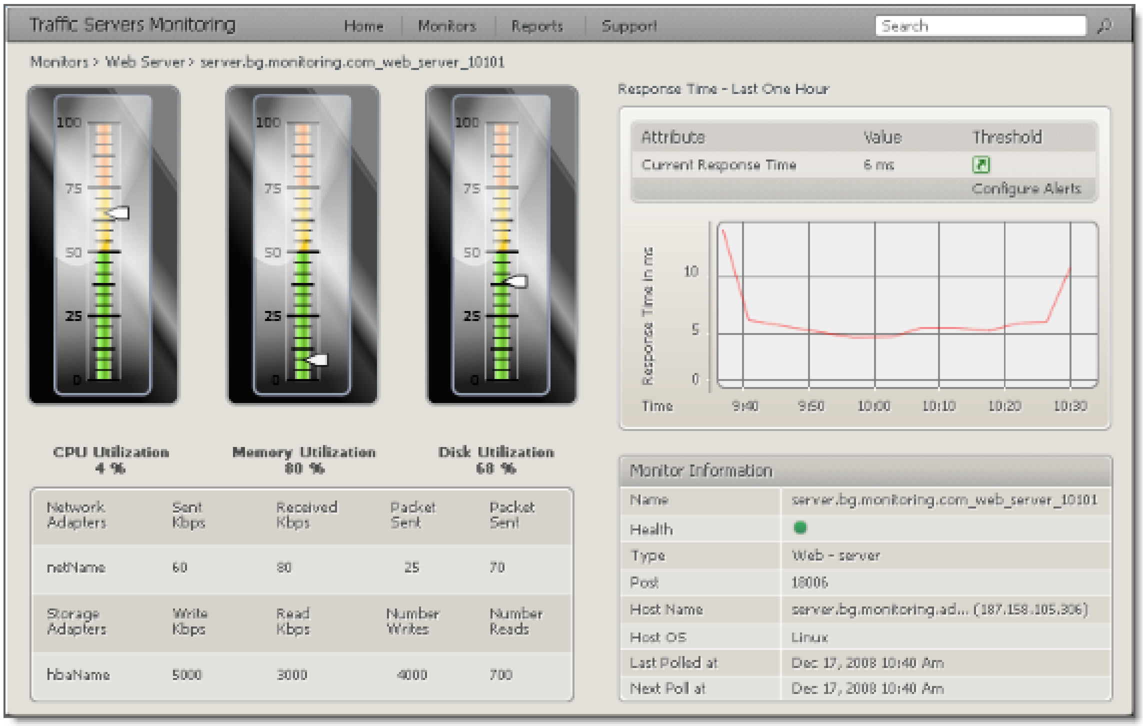
This chapter demonstrates how to use gauges to support dynamic data visualization. "First up", you will learn how to build a gauge starting with RadGauge and adding its constituent elements. You'll learn how to work with radial and numeric gauges and how these two element can be swapped without affecting application behavior. You will drill down into the gauge elements to see how scales, indicators and ranges work together and how some of the important properties are used to tweak behavior and appearance. Next, you'll learn how to bind individual indicators to data. Finally, you will use Expression Blend to customize gauge appearance.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Gauge\Gauge.sln.

20.2 Overview

Use Telerik Gauge controls to build dashboards for business, science, electronics or any purpose where dynamic data visualization is required.


Traffic Servers Monitoring
Home
Monitors
Reports
Support
Search
Monitors > Web Server > server.bg.monitoring.com_web_server_10101
100
75
50
25
0
Response Time - Last One Hour
Response Time in ms
10
5
0
Time
9:40
9:50
10:00
10:10
10:20
10:30

CPU Utilization
4 %

Memory Utilization
80 %

Disk Utilization
68 %

Attribute	Value	Threshold
Current Response Time	6 ms	↗
		Configure Alerts

Network Adapters	Sent Kbps	Received Kbps	Packet Sent	Packet Sent
netName	60	80	25	70
Storage Adapters	Write Kbps	Read Kbps	Number Writes	Number Reads
hbaName	5000	3000	4000	700

Monitor Information

Name	server.bg.monitoring.com_web_server_10101
Health	●
Type	Web - server
Post	18006
Host Name	server.bg.monitoring.ad... (187.158.105.306)
Host OS	Linux
Last Polled at	Dec 17, 2008 10:40 Am
Next Poll at	Dec 17, 2008 10:40 Am

RadGauge has circular, linear and numeric gauges that work with a many indicator types, such as needle, bar, state and marker. RadGauge comes with several predefined themes that provide visual "pop" at the cost of a single property assignment. The gauge and its elements can be composed and styled in endless ways to provide a unique, striking appearance for any purpose. For example, a car dashboard can be assembled using multiple radial and numeric gauges combined with any other Silverlight elements you might care to use.

120
90
60
oil temp
200
250
300
350
150
100
50
km/h
0.0
3
4
5
6
7
8
2
1
0
RPM
S W R 3
0 . 0
40
80
120
Water
1.0
0.5
0.0
fuel
17:53
25.5°c
3
2
1
0
oil pressure

Features

- **Radial Gauge**: a circular scale with numbers and tick marks. You can adjust the radius, center point, sweep angle, start angle and end angle of the scale.
- **Linear Gauge**: a rectangular scale that can be used for many familiar interfaces such as a thermometer or graphic equalizer.
- **Indicators**: There are five types of indicators that you can mix and match: marker, needle, bar, state indicator and numeric indicator. "State" indicators help you signal a change in the status of the data, e.g. "over temperature threshold" or "stock sell price".
- **Ranges**: Use ranges to stake out areas along the scale for special attention, particularly thresholds or other areas that may call for some decision to be made in response to the current data. The "State" indicator automatically mirrors the background of the corresponding range.
- **Rich Customization Capabilities**: The look-and-feel of the gauge can be articulated separately from the gauge behavior. The gauge and its constituent parts can be styled for a completely custom look. Each of the gauge elements, i.e. indicators, ranges, etc., has a rich set of properties and events to handle the most demanding requirements.
- **Animations**: gauges are smoothly animated, right out of the box.
- **Events**: indicators and ranges generate events to alert you to value changes, when indicator values are entering or leaving a range or when a value stays in a range for a given time period.

20.3 Getting Started

In this walk through you will build a simple linear gauge with a single indicator. You will also implement a timer that changes the indicator value to a random value every half second.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Gauge**

c) **Telerik.Windows.Themes.Vista**

d) **Telerik.Windows.Themes.Summer**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add an event handler to the UserControl element for the "Loaded" event.

3) Add XML namespace references for the **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Gauges** namespaces found in the Telerik.Windows.Controls.Gauge assembly. Also add an XML namespace reference to the **Telerik.Windows.Controls** namespace found in the Telerik.Windows.Controls assembly.

Note: Each xmlns statement should be all on one line. The example below is split up to fit the size constraints of the page in this manual.

```
xmlns:control=
"clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls.Gauge"
xmlns:gauge=
"clr-namespace:
Telerik.Windows.Controls.Gauges;assembly=Telerik.Windows.Controls.Gauge"
xmlns:telerik=
"clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls"
```

4) Drag a **RadGauge** control to a point inside the main "LayoutRoot" Grid element.

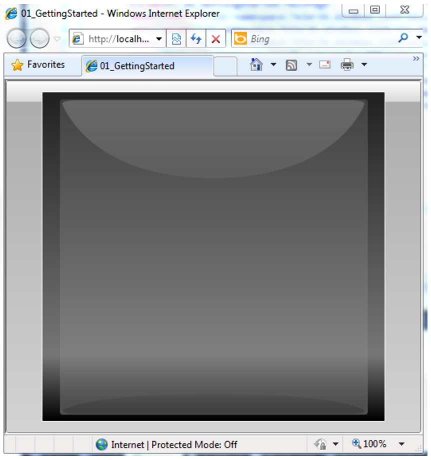
```
<UserControl . . .>
   <Grid x:Name="LayoutRoot">
      <control:RadGauge>

      </control:RadGauge>
   </Grid>
</UserControl>
```

If you run the application now, the web page will be blank. RadGauge is a ContentControl descendant, whose purpose is to hold other Silverlight controls. In particular, RadGauge can contain RadialGauge, LinearGauge and NumericScale controls.

5) Add a **LinearGauge** inside the RadGauge element.

```
<Grid x:Name="LayoutRoot">
   <control:RadGauge>
      <gauge:LinearGauge></gauge:LinearGauge>
   </control:RadGauge>
</Grid>
```

6) Press **F5** to run the application. Now the web page will display the default background of a LinearGauge. As yet, there is no other gauge functionality added, just the background appearance:

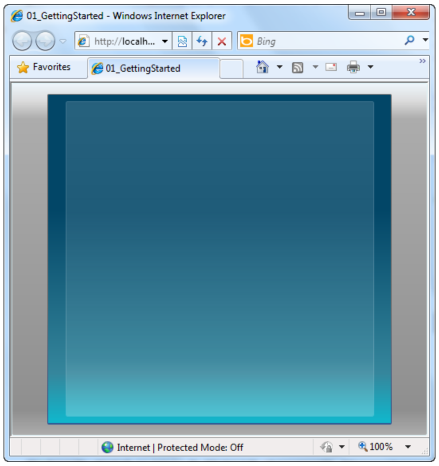

01_GettingStarted - Windows Internet Explorer
http://localh...
Bing
Favorites
01_GettingStarted
Internet | Protected Mode: Off
100%

7) Close the browser to end the application.

8) Add a **StyleManager.Theme** attribute and set it to "Vista".

```
<gauge:LinearGauge telerik:StyleManager.Theme="Vista">
</gauge:LinearGauge>
```

9) Press **F5** to run the application again to see the Vista theme in action.

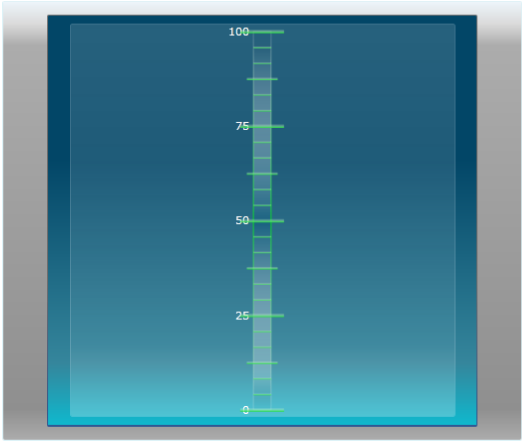

01_GettingStarted - Windows Internet Explorer
http://localh...
Bing
Favorites
01_GettingStarted
Internet | Protected Mode: Off
100%

10)Close the browser to end the application.

11)Add a **LinearScale** element to the LinearGauge. Set the **Min** attribute to "0" and the **Max** attribute to "100".

```xml
<control:RadGauge>
  <gauge:LinearGauge telerik:StyleManager.Theme="Vista">
    <gauge:LinearScale Min="0" Max="100"></gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```

12)Press **F5** to run the application and see the scale.



13)Close the browser to end the application.

14)Inside the LinearScale element, add an **IndicatorList**. Inside the IndicatorList, add a **LinearBar** element and name it "linearBar" so we can access it in code later.

```xml
<control:RadGauge>
  <gauge:LinearGauge telerik:StyleManager.Theme="Vista">
    <gauge:LinearScale Min="0" Max="100">
      <gauge:IndicatorList>
        <gauge:LinearBar x:Name="linearBar"></gauge:LinearBar>
      </gauge:IndicatorList>
    </gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```

Code Behind

1) In the code-behind for the page, add references to the **System.Windows.Threading** namespace in the "Imports" (VB) or "using" (C#) section of code.

2) Add a **DispatcherTimer** and a **Random** object. Instantiate both objects.

VB

```
Private _timer As New DispatcherTimer()
Private _random As New Random()
```

C#

```
private DispatcherTimer _timer = new DispatcherTimer();
private Random _random = new Random();
```

3) In the UserControl Loaded event handler, set the DispatcherTimer **Interval** to a half second time span, add a **Tick** event handler and call the **Start()** method.

VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  _timer.Interval = New TimeSpan(0, 0, 0, 0, 500)
  AddHandler _timer.Tick, AddressOf _timer_Tick
  _timer.Start()
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   _timer.Interval = new TimeSpan(0, 0, 0, 0, 500);
   _timer.Tick += new EventHandler(_timer_Tick);
   _timer.Start();
}
```

4) Add a Tick event handler. Inside the new handler, set the LinearBar Value property using the Random. Next() method. Pass the Next() method the minimum and maximum values it can return.

VB

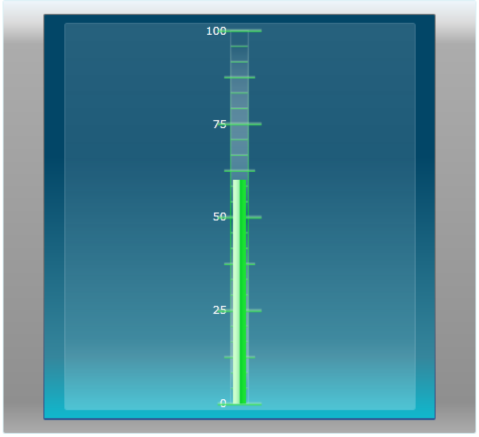
```
Private Sub _timer_Tick(ByVal sender As Object, ByVal e As EventArgs)
  linearBar.Value = _random.Next(1, 100)
End Sub
```

C#

```
void _timer_Tick(object sender, EventArgs e)
{
   linearBar.Value = _random.Next(1, 100);
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



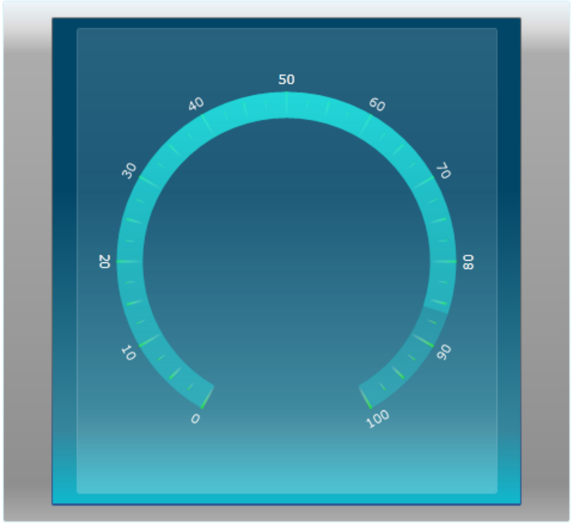
Test Application Features

- Verify that the linear bar animates smoothly up and down as the values change every half second.

Ideas for Extending This Example

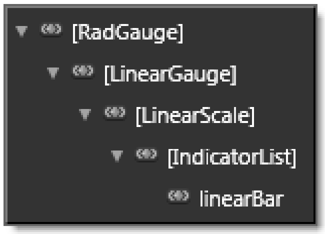
- Try styling the LinearGauge with the "Summer" theme.
- Change the LinearGauge to be a RadialGauge.

- Change the LinearScale to RadialScale and LinearBar to RadialBar. The screenshot below shows the resulting RadialGauge/RadialScale/RadialBar combination. You can make these changes to the XAML alone. No other code changes are required.



20.4 Control Details

In the "Getting Started" section, you created a simple RadGauge example that shows the typical hierarchy of elements that make up a gauge. If we open up the project in Expression Blend and look at the Objects and Timeline pane, the tree of elements shows how the RadGauge contains gauge controls, scale and a list of indicators.



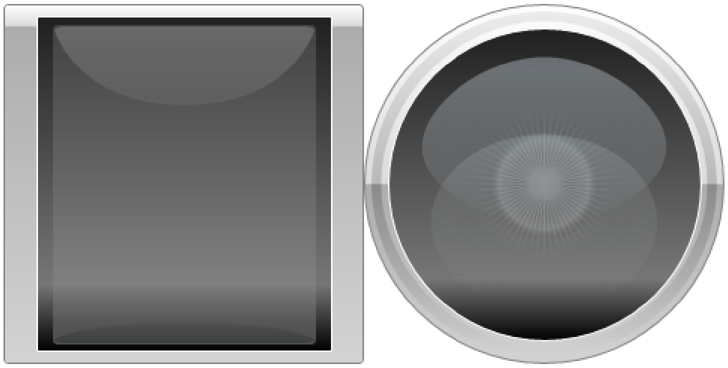
```
▼ [RadGauge]
    ▼ [LinearGauge]
        ▼ [LinearScale]
            ▼ [IndicatorList]
                linearBar
```

RadGauge

The RadGauge control can contain all gauge types (radial, linear or numeric). Layout is completely flexible and any Panel object (StackPanel, Grid, etc.) can be used as content. RadGauge will typically contain one or more gauge objects such as RadialGauge or LinearGauge. RadGauge is themeable and can set the style for all contained gauges. In the XAML below, the RadGauge has a "Summer" theme and contains a StackPanel as its content. Inside the StackPanel a LinearGauge and a RadialGauge are displayed side-by-side.

```
<control:RadGauge telerik:StyleManager.Theme="Summer">
   <StackPanel Orientation="Horizontal">
      <gauge:LinearGauge>
      </gauge:LinearGauge>
      <gauge:RadialGauge>
      </gauge:RadialGauge>
   </StackPanel>
</control:RadGauge>
```

The result of the markup shows the two gauges with the Summer theme applied:



Gauges

RadialGauge and **LinearGauge** classes provide a visual background for radial, linear and numeric scales. Both classes have a similar construction shown in the simplified XAML control template below. The template contains a ContentControl for the background, an ItemsPresenter to contain scales or indicators and another ContentControl for the foreground. You can use the predefined themes to handle the background and foreground or you can design your own unique look for the control.

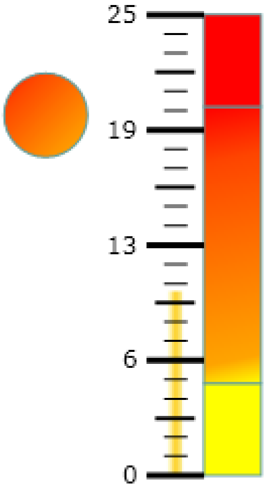
```xml
<ControlTemplate x:Key="RadialGaugeTemplate" TargetType="local:RadialGauge">
  <Border . . .>
    <Grid>
      <ContentControl Template="{StaticResource RadialGaugeBackground}"/>
      <ItemsPresenter />
      <ContentControl Template="{StaticResource RadialGaugeForeground}"/>
    </Grid>
  </Border>
</ControlTemplate>
```

Tip!

While the typical layout of a gauge is gauge\scale\indicators, there is some latitude to mix-and-match. Scale objects for example, can be contained directly in the RadGauge or in any Panel. At minimum you need a scale object to render your indicators, ranges, etc. The XAML below uses a StackPanel to contain a LinearScale.

```
<UserControl.Resources>
  <LinearGradientBrush x:Key="OrangeBrush">
    <GradientStop Color="Yellow" Offset="1" />
    <GradientStop Color="Orange" Offset="0.9" />
    <GradientStop Color="OrangeRed" Offset="0.2" />
    <GradientStop Color="Red" Offset="0" />
  </LinearGradientBrush>
</UserControl.Resources>

<Grid x:Name="LayoutRoot">

  <StackPanel Orientation="Vertical">
            <gauge:LinearScale Min="0" Max="25">
                <gauge:LinearScale.MajorTick>
                    <gauge:MajorTickProperties Location="OverCenter" Background
="Black"/>
                </gauge:LinearScale.MajorTick>
                <gauge:LinearScale.MiddleTick>
                    <gauge:MiddleTickProperties Location="OverCenter" Length="0.07"
Background="Black"/>
                </gauge:LinearScale.MiddleTick>
                <gauge:LinearScale.MinorTick>
                    <gauge:MinorTickProperties Location="OverCenter" Length="0.055"
Background="Black"/>
                </gauge:LinearScale.MinorTick>
                <gauge:LinearScale.Label>
                    <gauge:LabelProperties FontSize="9" Offset="0.02" Foreground
="Black" Location="Outside" />
                </gauge:LinearScale.Label>
                <gauge:IndicatorList>
                    <gauge:LinearBar Value="10" />
                    <gauge:StateIndicator Value="8" Top="0.2" Left="0.2"/>
                </gauge:IndicatorList>
                <gauge:RangeList>
                    <gauge:LinearRange Background="Yellow"
                    Min="0" Max="5" StartWidth="0.1" EndWidth="0.1" />
                    <gauge:LinearRange Background="{StaticResource OrangeBrush}"
                    Min="5" Max="20" StartWidth="0.1" EndWidth="0.1" />
                    <gauge:LinearRange Background="Red"
                    Min="20" Max="25" StartWidth="0.1" EndWidth="0.1" />
                </gauge:RangeList>
                </gauge:LinearScale>

            </StackPanel>

    </Grid>
```

The XAML rendered in the browser is shown in the screenshot below.



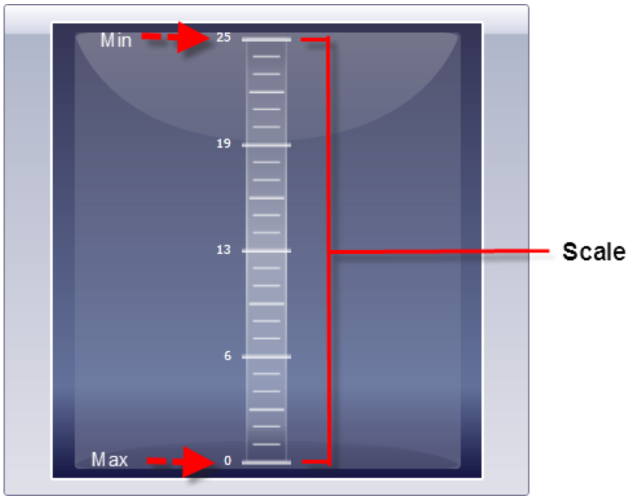
Scales

LinearScale and RadialScale are used to control the overall layout of tick marks, labels, indicators, ranges and an optional scale bar. Both of these classes descend from ScaleBase (in the Telerik.Windows. Controls.Gauges namespace). Starting with a LinearScale, lets walk through some of the rich set of properties available in ScaleBase.

The first task should be to set the **Min** and **Max** values. In this example the Min is "0" and Max is "25". This will automatically place a set of tick marks and labels in the scale area. MajorTick, MinorTick, MiddleTick and Label Properties are used to set correct location for ticks and Labels ; otherwise they will be shown shifted.

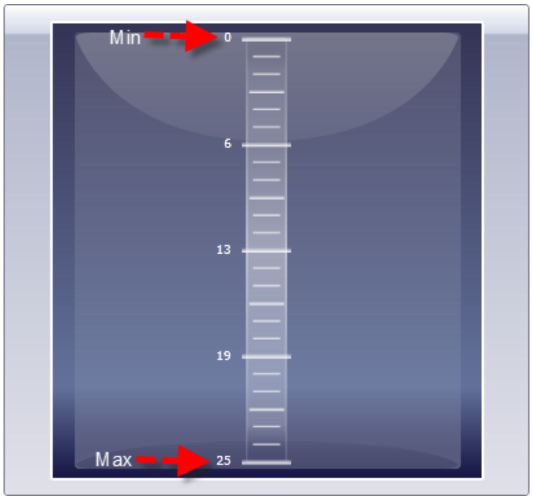
```
<control:RadGauge telerik:StyleManager.Theme="Office_Silver">
  <gauge:LinearGauge>
    <gauge:LinearScale Min="0" Max="25">
        <gauge:LinearScale.MajorTick>
             <gauge:MajorTickProperties Location="OverCenter" />
        </gauge:LinearScale.MajorTick>
        <gauge:LinearScale.MiddleTick>
             <gauge:MiddleTickProperties Location="OverCenter" Length="0.07" />
        </gauge:LinearScale.MiddleTick>
        <gauge:LinearScale.MinorTick>
              <gauge:MinorTickProperties Location="OverCenter" Length="0.055" />
         </gauge:LinearScale.MinorTick>
              <gauge:LinearScale.Label>
                 <gauge:LabelProperties FontSize="9" Offset="0.02" Location="Outside" />
             </gauge:LinearScale.Label>
    </gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```

Now the output looks like this screenshot:

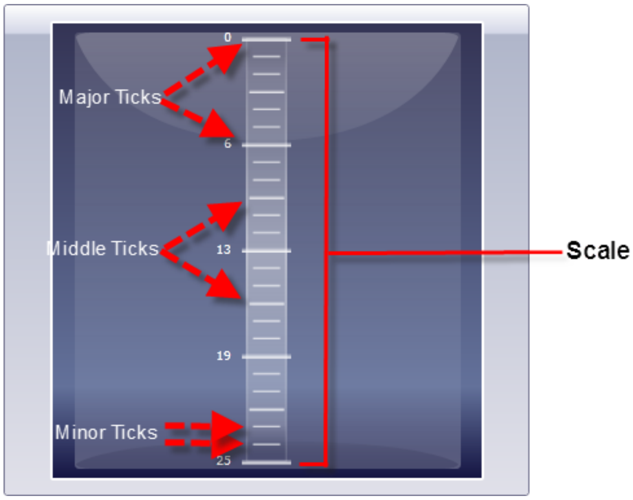


Set the **IsReversed** attribute "True" to swap the Min and Max locations:

```
<control:RadGauge telerik:StyleManager.Theme="Office_Silver">
  <gauge:LinearGauge>
    <gauge:LinearScale Min="0" Max="25" IsReversed="True">
 <gauge:LinearScale.MajorTick>
            <gauge:MajorTickProperties Location="OverCenter" />
       </gauge:LinearScale.MajorTick>
       <gauge:LinearScale.MiddleTick>
            <gauge:MiddleTickProperties Location="OverCenter" Length="0.07" />
       </gauge:LinearScale.MiddleTick>
       <gauge:LinearScale.MinorTick>
             <gauge:MinorTickProperties Location="OverCenter" Length="0.055" />
        </gauge:LinearScale.MinorTick>
             <gauge:LinearScale.Label>
                <gauge:LabelProperties FontSize="9" Offset="0.02" Location="Outside" />
            </gauge:LinearScale.Label>
    </gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```



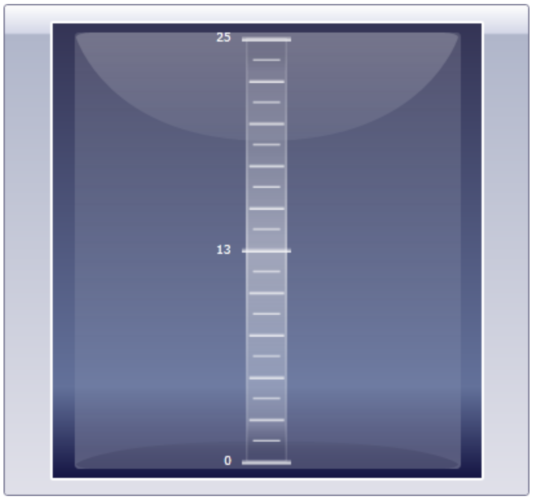
Values along the scale are marked with "Major", "Middle" and "Minor" ticks. The "Major" ticks are labeled.



You can control the number of ticks by setting the **MajorTicks**, **MiddleTicks** and **MinorTicks** attributes to integer values.

```
<control:RadGauge telerik:StyleManager.Theme="Office_Silver">
  <gauge:LinearGauge>
    <gauge:LinearScale Min="0" Max="25"
      MajorTicks="2" MiddleTicks="5" MinorTicks="2">
        <gauge:LinearScale.MajorTick>
             <gauge:MajorTickProperties Location="OverCenter" />
        </gauge:LinearScale.MajorTick>
        <gauge:LinearScale.MiddleTick>
             <gauge:MiddleTickProperties Location="OverCenter" Length="0.07" />
        </gauge:LinearScale.MiddleTick>
        <gauge:LinearScale.MinorTick>
              <gauge:MinorTickProperties Location="OverCenter" Length="0.055" />
         </gauge:LinearScale.MinorTick>
              <gauge:LinearScale.Label>
                 <gauge:LabelProperties FontSize="9" Offset="0.02" Location="Outside" />
             </gauge:LinearScale.Label>
    </gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```

In the screenshot below, there are two major ticks, five middle ticks and two minor ticks.

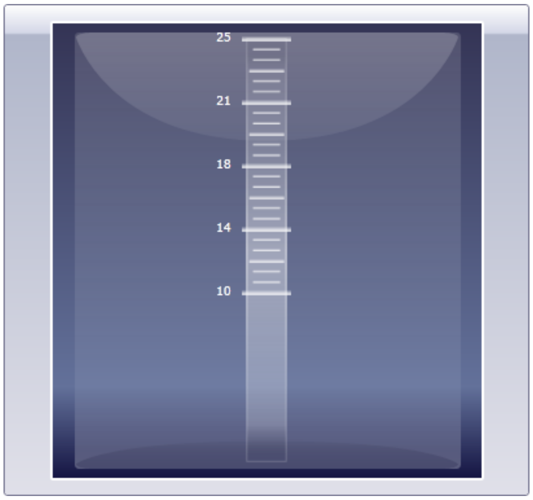


Note: The **ShowFirstLabel** and **ShowLastLabel** attributes control visibility of the first and last major ticks. Labels are only shown for the major ticks.

You can actually have the ticks start at a particular value by using the **StartTickOffset** to set the starting point for drawing ticks. In this example, the ticks are drawn beginning with the StartTickOffset value of "10".

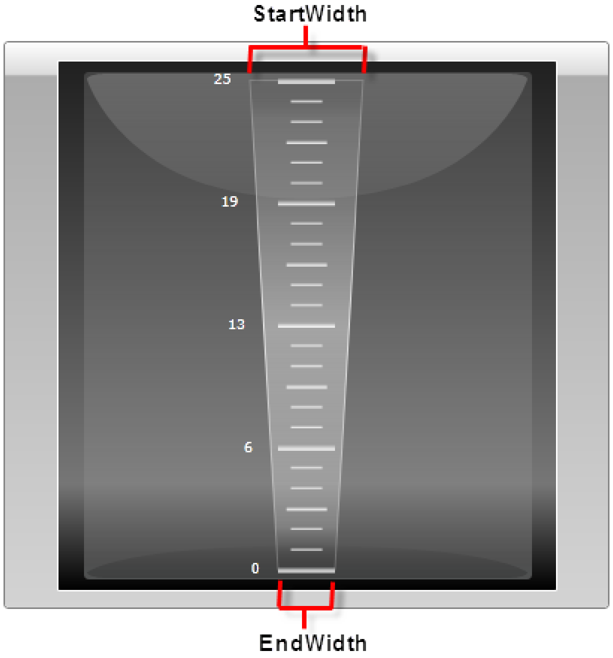
XAML

```
<control:RadGauge telerik:StyleManager.Theme="Office_Silver">
  <gauge:LinearGauge>
    <gauge:LinearScale Min="0" Max="25"
        StartTickOffset="10">
        <gauge:LinearScale.MajorTick>
             <gauge:MajorTickProperties Location="OverCenter" />
        </gauge:LinearScale.MajorTick>
        <gauge:LinearScale.MiddleTick>
             <gauge:MiddleTickProperties Location="OverCenter" Length="0.07" />
        </gauge:LinearScale.MiddleTick>
        <gauge:LinearScale.MinorTick>
              <gauge:MinorTickProperties Location="OverCenter" Length="0.055" />
         </gauge:LinearScale.MinorTick>
              <gauge:LinearScale.Label>
                 <gauge:LabelProperties FontSize="9" Offset="0.02" Location="Outside" />
             </gauge:LinearScale.Label>
    </gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```

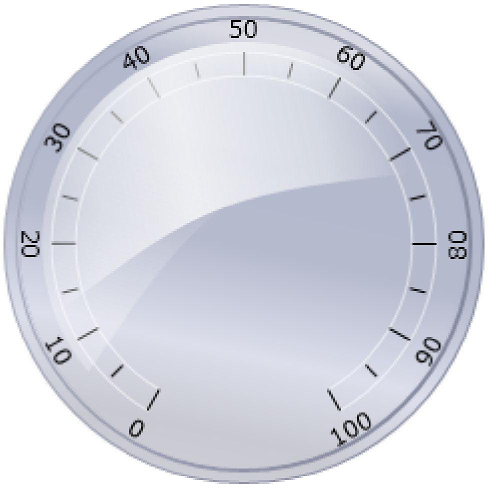

25
21
18
14
10

The scale width can be wider at one end using the **StartWidth** and **EndWidth** properties. In this example the EndWidth is slightly wider than the StartWidth. The "Office_Black" theme is being used here for better contrast so you can see the scale outline.

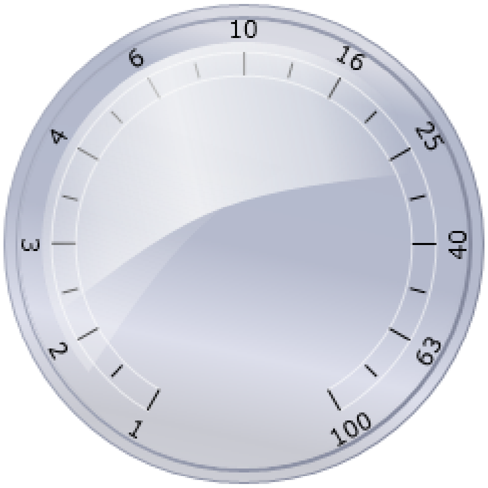
```
<control:RadGauge telerik:StyleManager.Theme="Office_Black">
  <gauge:LinearGauge>
    <gauge:LinearScale Min="0" Max="25"
      StartWidth=".1" EndWidth=".2">
        <gauge:LinearScale.MajorTick>
             <gauge:MajorTickProperties Location="OverCenter" />
        </gauge:LinearScale.MajorTick>
        <gauge:LinearScale.MiddleTick>
             <gauge:MiddleTickProperties Location="OverCenter" Length="0.07" />
        </gauge:LinearScale.MiddleTick>
        <gauge:LinearScale.MinorTick>
              <gauge:MinorTickProperties Location="OverCenter" Length="0.055" />
         </gauge:LinearScale.MinorTick>
              <gauge:LinearScale.Label>
                 <gauge:LabelProperties FontSize="9" Offset="0.02" Location="Outside" />
             </gauge:LinearScale.Label>
    </gauge:LinearScale>
  </gauge:LinearGauge>
</control:RadGauge>
```



By default, the numbers on the scale are distributed in a linear fashion. Distribute the numbers using a logarithmic scale by setting **IsLogarithmic** to "True". Base 10 is used by default, but the **LogarithmicBase** property can be set to a custom value. The screenshot below shows the same scale with and without IsLogarithmic enabled.



IsLogarithmic = False



IsLogarithmic = True

Indicators

You will want to populate your scale objects with "Indicators", i.e. visual objects that report a value or state to the user. These indicators can be bars, needles, markers or state indicators. Most of the indicators descend from **IndicatorBase** (except for NumericIndicator which is an ItemsControl). IndicatorBase has several important common properties:

- **Value** is the key property that drives everything. In a "Bar" indicator, the Value is the location of the top of the bar in the scale. For a "Needle" indicator, Value is the position of the needle along the scale.
- **Refresh Behavior**: **RefreshMode** can be **Average**, **Last**, **Min** or **Max**. If the user is clicking the indicator with a mouse, RefreshMode determines the new value for the indicator. **RefreshRate** is a TimeSpan and determines when to supply a new value.
- **Animation**: By default, the **IsAnimated** property is true, the indicators are animated and move smoothly between points on the scale. **Duration** controls the amount of time the animation takes to complete.
- **Snap Behavior**: **SnapType** can be **None**, **ToGrid** and **ToInterval**. "None" turns off snapping. "ToGrid" causes the indicator to snap to the nearest tick mark. "ToInterval" causes the indicator to snap to the next value that is **SnapInterval** away from the starting value.

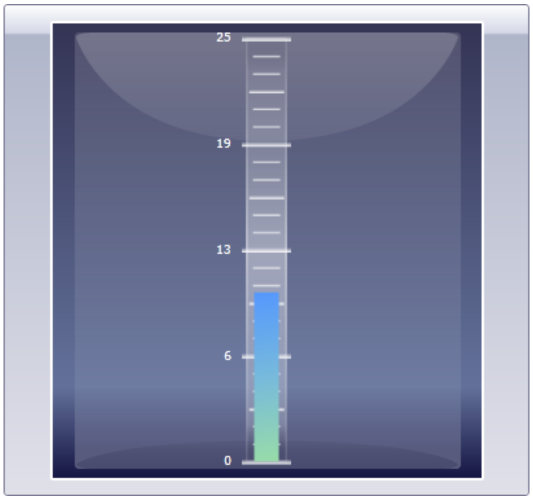
IndicatorBase descends from a **ScaleBase** object that adds a few more important properties including:

- **Location** is the position of the indicator relative to parts of the scale and can be **Outside**, **OverOutside** , **OverCenter**, **OverInside**, **CenterOutside**, **CenterInside** and **Inside**. The exact meaning of these settings is dependant on the type of scale (radial or linear) that the indicator is part of. See the online help for detailed information.
- **Offset** is the distance of the indicator from the scale.

A scale bar is an indicator that is rendered as a continuous band spanning the gauge and stopping at a point matching its Value property setting. The scale bar is also used as a platform for the placement of child elements, such as tick marks and labels. To add an indicator to your scale element, first add an IndicatorList element, then one or more indicators inside this list element. In the example below there is a single LinearBar added to the list with its value set at "10". The Background brush is defined in a resource not shown here in the interests of brevity.

```
 <control:RadGauge telerik:StyleManager.Theme="Office_Silver">
            <gauge:LinearGauge>
                <gauge:LinearScale Min="0" Max="25">
                    <gauge:LinearScale.MajorTick>
                        <gauge:MajorTickProperties Location="OverCenter" />
                    </gauge:LinearScale.MajorTick>
                    <gauge:LinearScale.MiddleTick>
                        <gauge:MiddleTickProperties Location="OverCenter" Length="0.07"
/>
                    </gauge:LinearScale.MiddleTick>
                    <gauge:LinearScale.MinorTick>
                        <gauge:MinorTickProperties Location="OverCenter" Length="0.055"
/>
                    </gauge:LinearScale.MinorTick>
                    <gauge:LinearScale.Label>
                        <gauge:LabelProperties FontSize="9" Offset="0.02" Location
="Outside" />
                    </gauge:LinearScale.Label>
                    <gauge:IndicatorList>
                        <gauge:LinearBar Value="10"
                        Background="{StaticResource IndicatorBrush}" />
                    </gauge:IndicatorList>
                </gauge:LinearScale>
            </gauge:LinearGauge>
        </control:RadGauge>
```

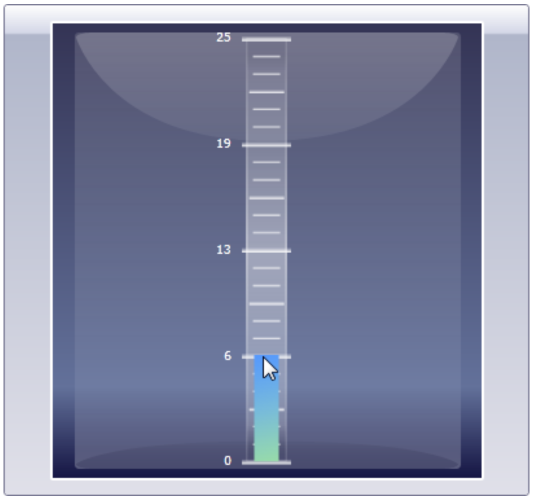
The output for the XAML in the screenshot shows the bar ending at "10".


25
19
13
6
0

If you want the user to set the values of the indicators using the mouse, set the scale **IsInteractive** property to "True".

```
  <control:RadGauge telerik:StyleManager.Theme="Office_Silver">
            <gauge:LinearGauge>
                <gauge:LinearScale Min="0" Max="25"
                IsInteractive="True">
                    <gauge:LinearScale.MajorTick>
                        <gauge:MajorTickProperties Location="OverCenter" />
                    </gauge:LinearScale.MajorTick>
                    <gauge:LinearScale.MiddleTick>
                        <gauge:MiddleTickProperties Location="OverCenter" Length="0.07"
/>
                    </gauge:LinearScale.MiddleTick>
                    <gauge:LinearScale.MinorTick>
                        <gauge:MinorTickProperties Location="OverCenter" Length="0.055"
/>
                    </gauge:LinearScale.MinorTick>
                    <gauge:LinearScale.Label>
                        <gauge:LabelProperties FontSize="9" Offset="0.02" Location
="Outside" />
                    </gauge:LinearScale.Label>
                    <gauge:IndicatorList>
                        <gauge:LinearBar
                        Background="{StaticResource IndicatorBrush}" />
                    </gauge:IndicatorList>
                </gauge:LinearScale>
            </gauge:LinearGauge>
        </control:RadGauge>
```

In the screenshot below, the mouse sets the LinearBar Value at "6".



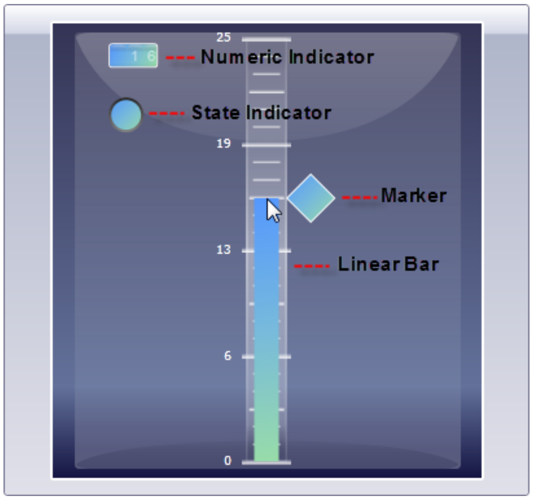
In this next example we add a **Marker**, a **NumericIndicator** and a **StateIndicator** to the indicator list.

Notice that the NumericIndicator has several NumberPosition elements defined in order to show up in the scale, one for each digit that will be displayed. Both the Marker and NumericIndicator will respond to mouse clicks when the IsInteractive property is set to "True". You can set the Format attribute if you have a specific number of decimals to display or want to display as currency {C} or percentage {P}. This example uses the floating point format.

The StateIndicator will be used a little later when we discuss ranges. For now, know that it signals some state of the data. For example, if you had a "danger zone" temperature in a scale, the StateIndicator might change to a red color to alert the user.

```
 <control:RadGauge telerik:StyleManager.Theme="Office_Silver">
            <gauge:LinearGauge>
                <gauge:LinearScale Min="0" Max="25"
                IsInteractive="True">
                    <gauge:LinearScale.MajorTick>
                        <gauge:MajorTickProperties Location="OverCenter" />
                    </gauge:LinearScale.MajorTick>
                    <gauge:LinearScale.MiddleTick>
                        <gauge:MiddleTickProperties Location="OverCenter" Length="0.07"
/>
                    </gauge:LinearScale.MiddleTick>
                    <gauge:LinearScale.MinorTick>
                        <gauge:MinorTickProperties Location="OverCenter" Length="0.055"
/>
                    </gauge:LinearScale.MinorTick>
                    <gauge:LinearScale.Label>
                        <gauge:LabelProperties FontSize="9" Offset="0.02" Location
="Outside" />
                    </gauge:LinearScale.Label>
                    <gauge:IndicatorList>
                        <gauge:LinearBar Value="10"
                        Background="{StaticResource IndicatorBrush}" />
                        <gauge:Marker Value="13"
                        Background="{StaticResource IndicatorBrush}" />
                        <gauge:NumericIndicator Format="{}{0:F0}"
                        Left="0.2" Top="0.08"
                        RelativeWidth="0.1"
                        RelativeHeight="0.05" Value="10"
                        Background="{StaticResource IndicatorBrush}">
                            <gauge:NumberPosition />
                            <gauge:NumberPosition />
                            <gauge:NumberPosition />
                        </gauge:NumericIndicator>
                        <gauge:StateIndicator Value="10" Top="0.19"
                        Left="0.2" RelativeWidth="0.07"
                        RelativeHeight=".07"
                        Background="{StaticResource IndicatorBrush}" />
                    </gauge:IndicatorList>
                </gauge:LinearScale>
            </gauge:LinearGauge>
        </control:RadGauge>
```

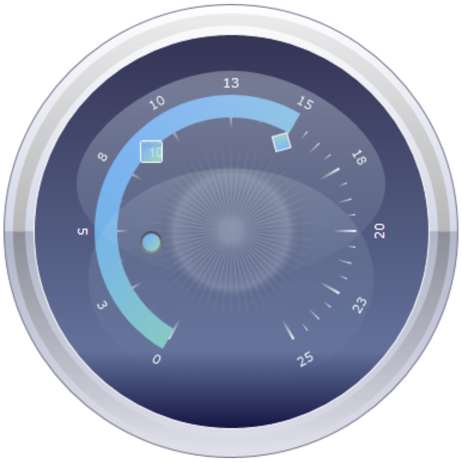
Running in the browser, you can see that the numeric indicator shows in the upper left hand corner. The marker shows to the right of the bar. IsInteractive is set to "True", so the value selected by the mouse is reflected by bar, marker and numeric indicators.



Everything you've learned up to this point is also valid for the "Radial", versions of the gauge elements. The XAML below is essentially the same as the previous example except that RadialGauge, RadialScale and RadialBar have replaced their "Linear" counterparts. A few minor tweaks to size and location make it easier to see the numeric and state indicators.

```
<control:RadGauge telerik:StyleManager.Theme="Office_Silver">
          <gauge:RadialGauge>
              <gauge:RadialScale Min="0" Max="25"
              IsInteractive="True">
                  <gauge:IndicatorList>
                      <gauge:RadialBar Value="10"
                      Background="{StaticResource IndicatorBrush}" />
                      <gauge:Marker Value="13"
                      Background="{StaticResource IndicatorBrush}" />
                      <gauge:NumericIndicator Format="{}{0:F0}"
                      Left="0.3" Top="0.3"
                      RelativeWidth="0.1"
                      RelativeHeight="0.1" Value="10"
                      Background="{StaticResource IndicatorBrush}">
                          <gauge:NumberPosition />
                          <gauge:NumberPosition />
                          <gauge:NumberPosition />
                      </gauge:NumericIndicator>
                      <gauge:StateIndicator Value="10" Top="0.5"
                      Left="0.3" RelativeWidth="0.1"
                      RelativeHeight=".1"
                      Background="{StaticResource IndicatorBrush}" />
                  </gauge:IndicatorList>
              </gauge:RadialScale>
          </gauge:RadialGauge>
      </control:RadGauge>
```

Running in the browser, the gauge performs in the same manner as the "Linear" version.



Ranges

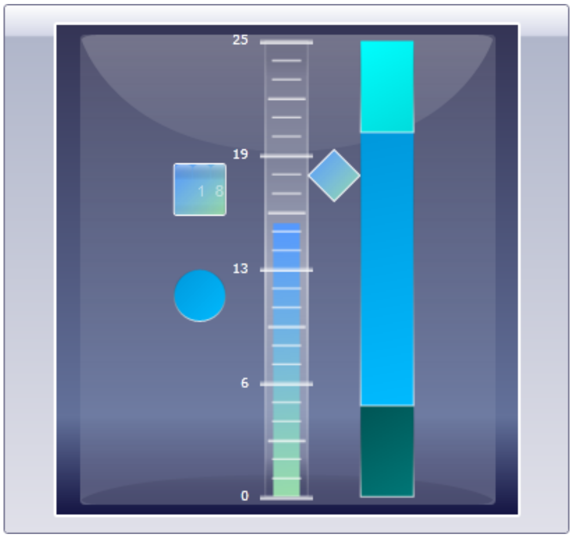
Ranges represent a continuous set of values along a scale. Every range has minimum/maximum values and can fire events when an indicator enters, leaves or stays in a range for a specified time span. You can define multiple ranges. When a StateIndicator value falls within a range, the StateIndicator fill color matches the range color.

To create a range, first add a **RangeList** element at the same level within a scale as an IndicatorList. Greater **Offset** numbers moves the range away from the center of the scale. **StartWidth** and **EndWidth** can be equal to make the range appear as an even bar, or can be asymmetrical to display as a bevel (perhaps to communicate that values farther up the range are larger).

There are three ranges in the example below that divide the scale between 0..5, 5..20 and 20..25.

```
. . .
<gauge:RangeList>
  <gauge:LinearRange Min="0" Max="5"
      Background="{StaticResource BottomRangeBrush}"
      Offset="0.1" StartWidth="0.1"
      EndWidth="0.1" />
  <gauge:LinearRange Min="5" Max="20"
      Background="{StaticResource MiddleRangeBrush}"
      Offset="0.1" StartWidth="0.1"
      EndWidth="0.1" />
  <gauge:LinearRange Min="20" Max="25"
      Background="{StaticResource TopRangeBrush}"
      Offset="0.1" StartWidth="0.1"
      EndWidth="0.1" />
</gauge:RangeList>
. . .
```

Notice that when the application runs, the StateIndicator color matches the range that the user clicks in.



Range Events

Respond to indicator activity within a range using the **EnterRange**, **LeaveRange** and **RangeTimeout** events. The **RangeTimeout** event fires based on the setting of the **Timeout** property of the range. You can assign the Timeout at the same time you subscribe to the event. The event arguments for all three events pass a reference to the **Indicator** object that's generating the activity and the **Range** object that is responding to the activity. The sample code below shows some of the possibilities.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  AddHandler bottomRange.LeaveRange, AddressOf bottomRange_LeaveRange
  AddHandler bottomRange.EnterRange, AddressOf bottomRange_EnterRange
  bottomRange.Timeout = New TimeSpan(0, 30, 0)
  AddHandler bottomRange.RangeTimeout, AddressOf bottomRange_RangeTimeout
End Sub

Private Sub bottomRange_EnterRange( _
ByVal sender As Object, ByVal eventArgs As RoutedRangeEventArgs)
  If (TypeOf eventArgs.Indicator Is LinearBar) _
AndAlso (eventArgs.Range.Name.Equals("bottomRange")) Then
    RadWindow.Alert("Price has dropped below the threshold")
  End If
End Sub

Private Sub bottomRange_RangeTimeout( _
ByVal sender As Object, ByVal eventArgs As RoutedRangeEventArgs)
  RadWindow.Alert("Indicator has remained in the bottom range for 30 minutes")
End Sub

Private Sub bottomRange_LeaveRange( _
ByVal sender As Object, ByVal eventArgs As RoutedRangeEventArgs)
  If (TypeOf eventArgs.Indicator Is LinearBar) _
AndAlso (eventArgs.Range.Name.Equals("bottomRange")) Then
    RadWindow.Alert("Price has risen above the threshold")
  End If
End Sub
```

C#

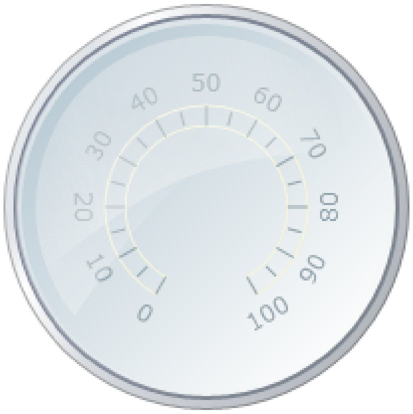
```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    bottomRange.LeaveRange += new RoutedRangeEventHandler(bottomRange_LeaveRange);
    bottomRange.EnterRange += new RoutedRangeEventHandler(bottomRange_EnterRange);
    bottomRange.Timeout = new TimeSpan(0, 30, 0);
    bottomRange.RangeTimeout += new RoutedRangeEventHandler(bottomRange_RangeTimeout);
}

void bottomRange_EnterRange(object sender, RoutedRangeEventArgs eventArgs)
{
    if ((eventArgs.Indicator is LinearBar) &&
        (eventArgs.Range.Name.Equals("bottomRange")))
    {
        RadWindow.Alert("Price has dropped below the threshold");
    }
}

void bottomRange_RangeTimeout(object sender, RoutedRangeEventArgs eventArgs)
{
    RadWindow.Alert("Indicator has remained in the bottom range for 30 minutes");
}

void bottomRange_LeaveRange(object sender, RoutedRangeEventArgs eventArgs)
{
    if ((eventArgs.Indicator is LinearBar) &&
        (eventArgs.Range.Name.Equals("bottomRange")))
    {
        RadWindow.Alert("Price has risen above the threshold");
    }
}
```

LinearScale Specifics

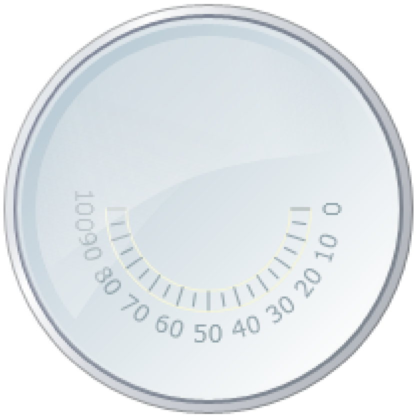
Orientation can be **Horizontal** or **Vertical**. You can set the location of the scale relative to its container using the **Left** and **Top** properties. Use the **RelativeHeight** property to set the height of the scale relative to its container.

RadialScale Specifics

By default the scale is automatically centered within the gauge with a **Center** property of "0.5,0.5". That works out nicely because the gauge graphics for a given style form a frame around the scale. You can use the Center property to move the scale to any Point. Set the **Radius** property to a value between 0 and 1. A Radius of "0.5", for instance, renders the scale at the half way point of the container.



The origin and length of a radial scale are controlled by **StartAngle** and **SweepAngle** properties. These properties are in degrees and follow Silverlight standards where zero degrees points due east and positive angles result in a clockwise rotation. The screenshot below shows the result when StartAngle = "0" and SweepAngle = "180".



20.5 Binding

To bind data using the current edition of RadGauge you actually bind the individual indicator elements. In this walk through you will bind a series of "Stock" market objects to linear bar elements and update them periodically.

Gotcha!

At the time of this writing there is a Silverlight issue where animation and binding conflict. As a workaround you can leave animation off or create your own external animation.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

   a) **Telerik.Windows.Controls**

   b) **Telerik.Windows.Controls.Gauge**

Create the View Model Object

1) In the Solution Explorer, right-click the project and select **Add > Class...** Rename the class file "Stock.cs". Verify that a reference to the **System.ComponentModel** namespace is included in the "Imports" (VB) or "using" (C#) section of code. Copy and paste the code below.

   This class will represent a single stock market symbol and "quote" or price. Stock implements the INotifyPropertyChanged interface so that changes to the Quote property will be reflected automatically in a bound LinearBar Value property.

VB

```
Public Class Stock
  Implements INotifyPropertyChanged
  Public Event PropertyChanged As PropertyChangedEventHandler

  Private _quote As Double
  Private _symbol As String

  Public Sub New(ByVal symbol As String, ByVal quote As Double)
    Me.Quote = quote
    Me.Symbol = symbol
  End Sub

  Public Property Quote() As Double
    Get
      Return _quote
    End Get
    Set(ByVal value As Double)
      _quote = value
      NotifyPropertyChanged("Quote")
    End Set
  End Property

  Public Property Symbol() As String
    Get
      Return _symbol
    End Get
    Set(ByVal value As String)
      _symbol = value
      NotifyPropertyChanged("Symbol")
    End Set
  End Property

  Private Sub NotifyPropertyChanged(ByVal propertyName As String)
    RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
  End Sub
End Class
```

```
public class Stock : INotifyPropertyChanged
{
   public event PropertyChangedEventHandler PropertyChanged;

   private double _quote;
   private string _symbol;

   public Stock(string symbol, double quote)
   {
      this.Quote = quote;
      this.Symbol = symbol;
   }

   public double Quote
   {
      get
      {
         return _quote;
      }
      set
      {
         _quote = value;
         NotifyPropertyChanged("Quote");
      }
   }

   public string Symbol
   {
      get
      {
         return _symbol;
      }
      set
      {
         _symbol = value;
         NotifyPropertyChanged("Symbol");
      }
   }

   private void NotifyPropertyChanged(string propertyName)
   {
      if (PropertyChanged != null)
      {
         PropertyChanged(this,
            new PropertyChangedEventArgs(propertyName));
      }
   }
}
```

2) In the Solution Explorer, right-click the project and select **Add > Class...** Rename the class file "Stocks.cs". Verify that references to the **System.Collections.Generic** and **System.Collections.ObjectModel** namespaces are included in the "Imports" (VB) or "using" (C#) section of code. Copy and paste the code below.

This class will represent a series of Stock objects that will be bound to gauge controls.

VB

```
Public Class Stocks
  Inherits List(Of Stock)
  Public Sub New()
    Dim symbols As List(Of String) = _
New List(Of String) (New String() {"MSFT", "GOOG", "YHOO", "IBM", "AAPL"})
    Dim random As New Random()
    For Each symbol As String In symbols
      Me.Add(New Stock(symbol, random.Next(0, 100)))
    Next symbol
  End Sub
End Class
```

C#

```
public class Stocks : List<Stock>
{
   public Stocks()
   {
      List<string> symbols = new List<string>()
      { "MSFT", "GOOG", "YHOO", "IBM", "AAPL" };

      Random random = new Random();
      foreach(string symbol in symbols)
      {
         this.Add(new Stock(symbol, random.Next(0, 100)));
      }
   }
}
```

XAML Editing

1) Open MainPage.xaml for editing.
2) Verify that the XML namespaces for **Telerik.Windows.Controls** exist in the UserControl element. Add them if they do not exist. Also, add a "Loaded" event handler to the UserControl element.

```
<UserControl
. . .
xmlns:telerik=
"clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls"
. . .
Loaded="UserControl_Loaded">. . .
```

3) Add a **UserControl.Resources** element. Add a **LinearGradientBrush** to the resources using the XAML below.

```
<UserControl.Resources>
    <LinearGradientBrush x:Key="BackgroundBrush">
        <GradientStop Color="LightBlue" Offset="0.01" />
        <GradientStop Color="SkyBlue" Offset="0.02" />
        <GradientStop Color="SlateBlue" Offset="0.8" />
        <GradientStop Color="SkyBlue" Offset="0.99" />
        <GradientStop Color="LightBlue" Offset="1" />
    </LinearGradientBrush>
</UserControl.Resources>
```

4) Drag a RadGauge control from the Toolbox to a point inside the main "LayoutRoot" grid element. Inside the RadGauge element, add two nested StackPanel controls using the XAML below.

   The first stack panel provides the background. The second stack panel contains a series of text blocks and linear scale elements, centered in the browser.

XAML

```
<telerik:RadGauge>
    <StackPanel
        Background="{StaticResource BackgroundBrush}"
        HorizontalAlignment="Stretch">
        <StackPanel x:Name="spScales"
            Orientation="Horizontal"
            HorizontalAlignment="Center"
            Margin="10" />
    </StackPanel>
</telerik:RadGauge>
```

Code Behind

1) Navigate to the code-behind for the page. Verify that references to the **System.Windows.Controls**, **System.Windows.Data**, **System.Windows.Media**, **System.Windows.Threading** and **Telerik. Windows.Controls.Gauges** namespaces are included in the "Imports" (VB) or "using" (C#) section of code.

2) Add private members using the code below.

   The Random object will generate random numbers for stock quotes and linear bar colors. The DispatcherTimer will update the quotes every half second.

VB

```
Dim _random As New Random()
Dim _timer As New DispatcherTimer()
Dim _stocks As New Stocks()
```

C#

```
Random _random = new Random();
DispatcherTimer _timer = new DispatcherTimer();
Stocks _stocks = new Stocks();
```

3) Navigate to the code for the UserControl Loaded event handler. Add the code below to create a stack panel for each stock and populate each stack panel with a text label and linear bar.

   The code iterates the collection of Stock objects and generates a text block label and linear bar element for each stock. The GetLinearBar() method takes a Stock reference, creates the linear bar and binds the Value property to the Stock object's Quote property. At the end of this event handler, a timer is initialized and started. The timer will take care of updating the data.

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  ' Create a stack panel for each stock and populate
  ' with text label and linear bar
  For Each stock As Stock In _stocks
    ' initialize a linear scale
    Dim scale As New LinearScale()
    scale.Min = 0
    scale.Max = 100
    scale.Width = 70

    ' add a linear bar, already bound to the stock
    scale.Indicators.Add(GetLinearBar(stock))

    ' create a label to display stock symbol
    Dim tb As New TextBlock()
    tb.FontWeight = System.Windows.FontWeights.Bold
    tb.Text = stock.Symbol

    ' initialize a stack panel to contain a text label
    ' and scale. Add the stack panel to the stack panel
    ' already in the xaml markup
    Dim stackPanel As New StackPanel()
    stackPanel.VerticalAlignment = VerticalAlignment.Stretch
    stackPanel.Children.Add(tb)
    stackPanel.Children.Add(scale)
    spScales.Children.Add(stackPanel)
  Next stock

  ' initialize and start a timer to update the stocks
  AddHandler _timer.Tick, AddressOf _timer_Tick
  _timer.Interval = New TimeSpan(0, 0, 0, 1)
  _timer.Start()
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
  // Create a stack panel for each stock and populate
  // with text label and linear bar
  foreach (Stock stock in _stocks)
  {
    // initialize a linear scale
    LinearScale scale = new LinearScale();
    scale.Min = 0;
    scale.Max = 100;
    scale.Width = 70;

    // add a linear bar, already bound to the stock
    scale.Indicators.Add(GetLinearBar(stock));

    // create a label to display stock symbol
    TextBlock tb = new TextBlock();
    tb.FontWeight = System.Windows.FontWeights.Bold;
    tb.Text = stock.Symbol;

    // initialize a stack panel to contain a text label
    // and scale. Add the stack panel to the stack panel
    // already in the xaml markup
    StackPanel stackPanel = new StackPanel();
    stackPanel.VerticalAlignment = VerticalAlignment.Stretch;
    stackPanel.Children.Add(tb);
    stackPanel.Children.Add(scale);
    spScales.Children.Add(stackPanel);
  }

  // initialize and start a timer to update the stocks
  _timer.Tick += new EventHandler(_timer_Tick);
  _timer.Interval = new TimeSpan(0, 0, 0, 1);
  _timer.Start();
}
```

4) Add a private method to create a LinearBar element and bind it to the Stock Quote property.

"Binding" is System.Windows.Data object that connects the properties of binding targets and data sources. Because we're getting our data from a "Stock" object, the Stock object is assigned as the Binding Source. PropertyPath describes the property that we're binding to in the source. In this case, PropertyPath is the "Quote" property of the Stock object. BindingMode describes when and how the data will propagate. In this case, we want the data to be travel "one way" from the Stock object to the target. The SetBinding() method associates the binding information with the LinearBar Value property. When changes occur to the Stock Quote property, the new data propagates to the bound LinearBar' Value property automatically.

VB

```
' generate a linear bar and bind Value to the Stock "Quote" property
Private Function GetLinearBar(ByVal stock As Stock) As LinearBar
  Dim linearBar As New LinearBar()
  linearBar.Value = stock.Quote
  linearBar.Background = New SolidColorBrush(GetRandomColor())
  linearBar.StartWidth =.1
  linearBar.EndWidth =.1
  linearBar.RelativeHeight = 0.9

  ' Bind linear bar to the Quote property of the data object.
  Dim binding As New Binding()
  binding.Source = stock
  binding.Path = New PropertyPath("Quote")
  binding.Mode = BindingMode.OneWay
  linearBar.SetBinding(LinearBar.ValueProperty, binding)
  Return linearBar
End Function
```

```
// generate a linear bar and bind Value to the Stock "Quote" property
private LinearBar GetLinearBar(Stock stock)
{
   LinearBar linearBar = new LinearBar();
   linearBar.Value = stock.Quote;
   linearBar.Background = new SolidColorBrush(GetRandomColor());
   linearBar.StartWidth = .1;
   linearBar.EndWidth = .1;
   linearBar.RelativeHeight = 0.9;

   // Bind linear bar to the Quote property of the data object.
   Binding binding = new Binding();
   binding.Source = stock;
   binding.Path = new PropertyPath("Quote");
   binding.Mode = BindingMode.OneWay;
   linearBar.SetBinding(LinearBar.ValueProperty, binding);
   return linearBar;
}
```

5) Add a private method to generate a semi-transparent random color.

This method is used by the GetLinearBar() method to set the Background brush for the linear bar element.

VB

```
Private Function GetRandomColor() As Color
  Return Color.FromArgb(150, CByte(_random.Next(0, 255)), _
   CByte(_random.Next(0, 255)), CByte(_random.Next(0, 255)))
End Function
```

C#

```
private Color GetRandomColor()
{
   return Color.FromArgb(150, (byte)_random.Next(0, 255),
      (byte)_random.Next(0, 255), (byte)_random.Next(0, 255));
}
```

6) Add the Tick event handler for the timer. Iterate the Stock objects in the Stocks collection and refresh the Quote property using the Random.Next() method.

VB

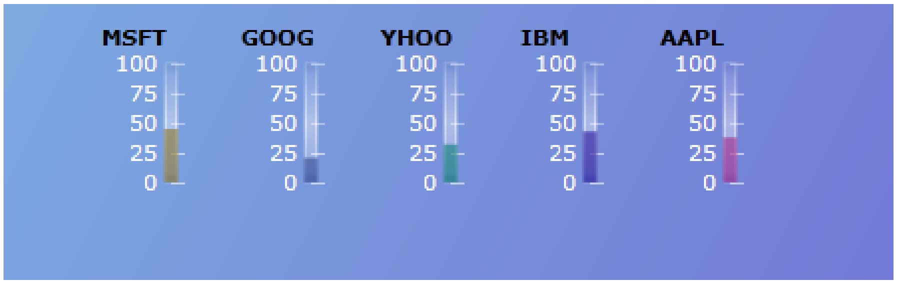
```
Private Sub _timer_Tick(ByVal sender As Object, ByVal e As EventArgs)
  For Each stock As Stock In _stocks 'update the symbols
    stock.Quote = _random.Next(0, 100)
  Next stock
End Sub
```

C#

```
void _timer_Tick(object sender, EventArgs e)
{
   foreach (Stock stock in _stocks) // update the symbols
   {
      stock.Quote = _random.Next(0, 100);
   }
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Ideas for Extending This Example

- Change the type of object from Stock to some other entity.
- Change the method of updating the object, using a web service for example.
- Use another type of scale object, i.e. radial or numeric.

20.6 Customization

In this example we will customize the RadGauge control. Specifically, we will change the background of a LinearGauge to fit with the "Scoville" style.

"Scoville" Styles

We will use a set of colors that include black, red, yellow and orange in many of the style related topics and prefix the style names with "Scoville". The "Scoville scale" measures the spiciness of peppers and other culinary irritants.

Project Setup

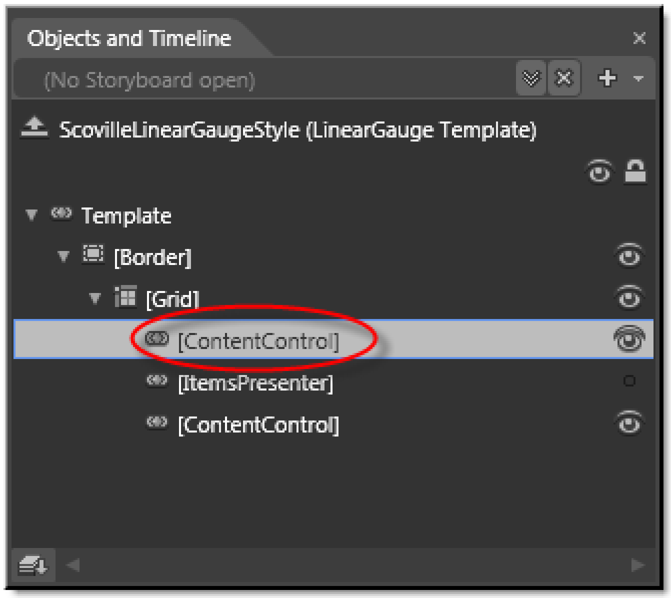
1) Start with the "Getting Started" project or a copy. Open the project in Expression Blend.

Edit the Page in Expression Blend

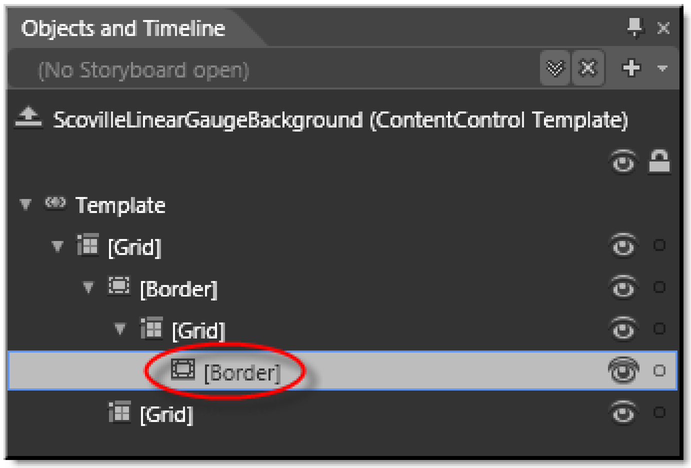
1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.

2) In the Objects and Timeline pane, right-click the "[LinearGauge]" element and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleLinearGaugeStyle". Click **OK** to create the style resource and close the dialog.

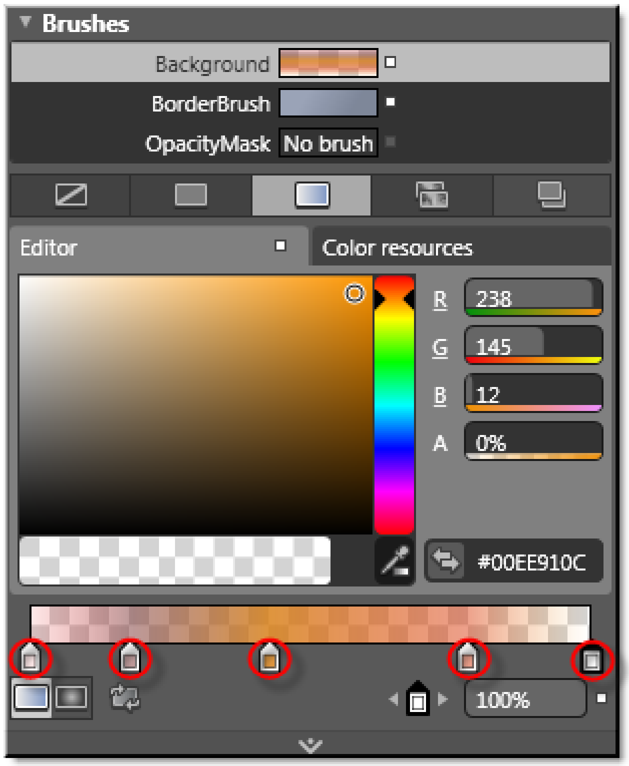
3) In the Objects and Timeline pane, the template contains a border with a grid. Inside the grid you find a "[ContentControl]" that you may remember from the Control Details section as the " LinearGaugeBackground". Right-click the [ContentControl] and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "Scoville LinearGaugeBackground". Click **OK** to create the style resource and close the dialog.



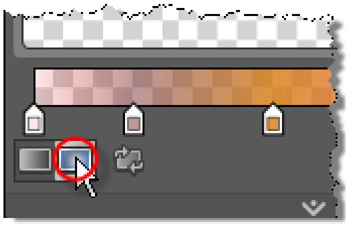
4) In the Objects and Timeline pane, open the object tree and select the innermost "[Border]" object.



5) In the Properties pane, select the **Brushes > Fill** property. The Border object will already have a gradient brush assigned. Notice the series of existing gradient stop indicators at the bottom of the Brushes pane. Select each gradient stop indicator and choose a color (you can use the eye dropper tool or the color slider just above the eye dropper). Choose between shades of black, red and orange.

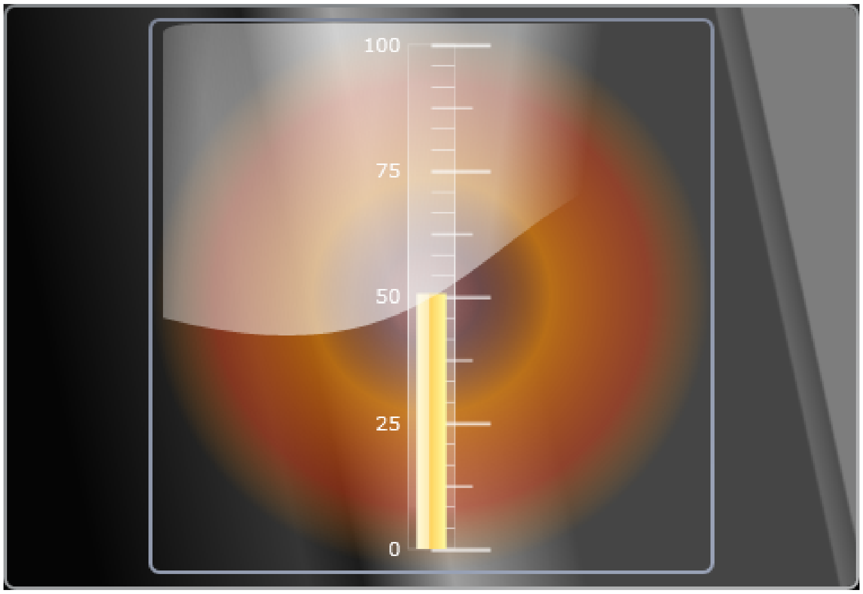


6) Select the Radial Gradient button from the lower left of the Brushes pane.



Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

- The animation for the application should run with the same functionality as when tested during the "Getting Started" walk through.

Ideas for Extending This Example

- Modify the foreground [ContentControl] element.
- Animate the background gradient.
- Modify the LinearBar indicator appearance.

20.7 Wrap Up

This chapter demonstrated how to use gauges to support dynamic data visualization. "First up", you learned how to build a gauge starting with RadGauge and adding its constituent elements. You learned how to work with radial and numeric gauges and how these two element can be swapped without affecting the applications behavior. You drilled down into the gauge elements to see how scales, indicators and ranges work together and how some of the important properties are used to tweak behavior and appearance. Next, you learned how to bind individual indicators to data. Finally, you used Expression Blend to customize gauge appearance.

Part XXI

ProgressBar

21 ProgressBar

21.1 Objectives

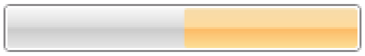
In this chapter you will use the RadProgressBar control to visually notify users about processes of a known or indeterminate length. You will learn how to work with the progress bar orientation, minimum/maximum values and how to update the progress bar.

Find the projects for this chapter at...

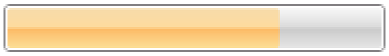
\Courseware\Projects\<CS|VB>\ProgressBar\ProgressBar.sln.

21.2 Overview

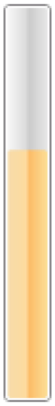
RadProgressBar, like the slider and up-down controls, is a RangeBase descendant and so has **Minimum**, **Maximum** and **Value** properties. RadProgressBar also uses the RangeBase **ValueChanged** event to notify your application when the progress bar value moves. Typically you won't code the ValueChanged event but instead code an event handler for some other control that acts *on* the progress bar. RadProgressBar includes a **SkipValue** property that starts the progress indicator at some midpoint between Minimum and Maximum. The image shows a SkipValue of "50" where the Minimum is "1" and the Maximum is "100".



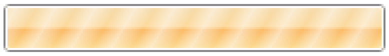
By default the progress bar has an **Orientation** of **Horizontal**...



... but can also be set **Vertical**:



The **IsIndeterminate** property, when true, animates a striped "barbershop" background indicating that the operation length is unknown.



21.3 Getting Started

The following walk through uses a timer to update the progress bar Value property. A label above the progress bar displays the Value property as a percentage of completion. Buttons are used to start, stop and reset the progress bar.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:
   a) **Telerik.Windows.Controls**

XAML Editing

1) Open MainPage.xaml for editing.
2) In the UserControl tag, add a "Loaded" event handler. *We will code this event later to initialize the timer that updates the progress bar.*

```
<UserControl
...
   Loaded="UserControl_Loaded">
```

3) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the **<Grid>** and **</Grid>** tags. *This XAML defines a basic structure of StackPanel elements where labels, buttons and the progress bar will be placed. The comments mark locations where you will add XAML in later steps.*

```
<StackPanel
   HorizontalAlignment="Center">

   <StackPanel
      Orientation="Horizontal"
      HorizontalAlignment="Center"
      Margin="20">

      <!--Loading text-->

   </StackPanel>

   <StackPanel
      Width="Auto"
      Orientation="Horizontal"
      VerticalAlignment="Center"
      HorizontalAlignment="Center"
      Height="30">

      <!--Buttons-->

   </StackPanel>

   <!--ProgressBar-->

</StackPanel>
```

4) Locate the comment "<!--Loading text-->" and replace it with the XAML below. This step adds two TextBlocks that will display the "Loading..." and current progress bar percentage value.

```
<!--Loading text-->

<TextBlock
   x:Name="loadingText"
   HorizontalAlignment="Center"
   Text="Loading... " />

<TextBlock
   x:Name="loadingPercentage"
   HorizontalAlignment="Center"
   FontWeight="Bold"
   Text="" />
```

5) Locate the comment "<!--Buttons-->" and replace it with the XAML below. This step will add the "Start", "Stop" and "Reset" buttons. Notice that the Click event has already been defined for each button. We will define the actual event handling code in a later step.

```
<!--Buttons-->

<Button
   x:Name="btn_start"
   Margin="0 0 10 0"
   Click="Btn_start_Click"
   VerticalAlignment="Top"
   Content="  Start  " />

<Button
   x:Name="btn_stop"
   Margin="0 0 10 0"
   Click="Btn_stop_Click"
   VerticalAlignment="Top"
   Content="  Stop  " />

<Button
   x:Name="btn_restart"
   Margin="0 0 10 0"
   Click="Btn_restart_Click"
   VerticalAlignment="Top"
   Content=" Restart " />
```

6) Drag a **RadProgressBar** from the Toolbox to a point just under the <!--ProgressBar--> comment. Edit the RadProgressBar tag to add the following properties.

a) **Name** = "pb"

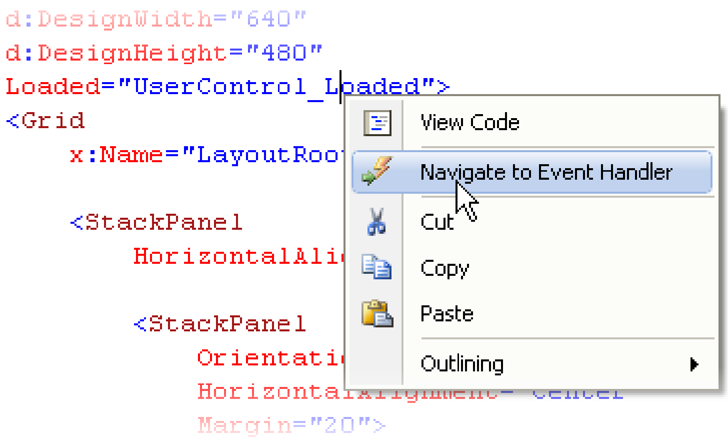
b) **Minimum** = "1"

c) **Maximum** = "100"

d) **MinHeight** = "25"

Code Behind

1) Locate the <UserControl> tag at the top of MainPage.xaml. Right-click the Loaded event and select "Navigate to Event Handler" from the context menu.



```
d:DesignWidth="640"
d:DesignHeight="480"
Loaded="UserControl_Loaded">
<Grid
    x:Name="LayoutRoo

    <StackPanel
        HorizontalAli

        <StackPanel
            Orientati
            HorizontalAlignment="Center"
            Margin="20">
```

2) Leave the Loaded event handler empty. Above the Loaded event handler, add a private member for the timer.

```
Private timer As System.Windows.Threading.DispatcherTimer
```

```
private System.Windows.Threading.DispatcherTimer timer;
```

3) Add the code below to the Loaded event handler. *This code creates an instance of a timer and sets the timer Interval to fire a Tick event every 10 milliseconds***.**

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Me.timer = New System.Windows.Threading.DispatcherTimer()
  Me.timer.Interval = TimeSpan.FromMilliseconds(10.0)
  AddHandler timer.Tick, AddressOf timer_Tick
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   this.timer = new System.Windows.Threading.DispatcherTimer();
   this.timer.Interval = TimeSpan.FromMilliseconds(10.0);
   this.timer.Tick += new EventHandler(timer_Tick);
}
```

4) Add a private helper method GetValuePercentage(). *This method calculates the Value as a percentage, given a RangeBase Minimum and Maximum properties.*

```
' Get a percentage based on the Value and its
' relation to the Minimum and Maximum properties.
Private Function GetValuePercentage(ByVal rb As RangeBase) As Integer
  Dim value As Double = rb.Value - rb.Minimum
  Dim [end] As Double = rb.Maximum - rb.Minimum
  Return Convert.ToInt32(value / [end] * 100)
End Function
```

C#

```
// Get a percentage based on the Value and its
// relation to the Minimum and Maximum properties.
private int GetValuePercentage(RangeBase rb)
{
   double value = rb.Value - rb.Minimum;
   double end = rb.Maximum - rb.Minimum;
   return Convert.ToInt32(value / end * 100);
}
```

5) Verify that a reference to the **System.Windows.Controls.Primitives** namespace is included in the "Imports" (VB) or "using" (C#) section of code.

6) Handle the timer Tick event to update the progress bar. *This event handler receives the current percentage of the progress bar, updates the "loadingPercentage" TextBlock to reflect the percentage and "bumps" the progress bar Value property.*

VB

```
Private Sub timer_Tick(ByVal sender As Object, ByVal e As EventArgs)
  Dim value As Integer = GetValuePercentage(pb)
  loadingPercentage.Text = value.ToString() & "%"
  pb.Value += 1
End Sub
```

C#

```
void timer_Tick(object sender, EventArgs e)
{
   int value = GetValuePercentage(pb);
   loadingPercentage.Text = value.ToString() + "%";
   pb.Value++;
}
```

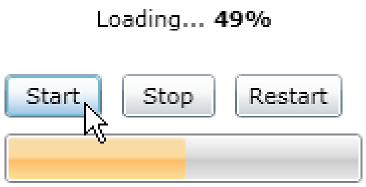
7) Add event handlers for each of the button Click events. *These handlers start, stop and reset the timer and progress bar.*

```
Private Sub Btn_start_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Me.timer.Start()
End Sub

Private Sub Btn_stop_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Me.timer.Stop()
End Sub

Private Sub Btn_restart_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Me.timer.Stop()
  pb.Value = pb.Minimum
  Me.timer.Start()
End Sub
```

```
private void Btn_start_Click(object sender, RoutedEventArgs e)
{
   this.timer.Start();
}

private void Btn_stop_Click(object sender, RoutedEventArgs e)
{
   this.timer.Stop();
}

private void Btn_restart_Click(object sender, RoutedEventArgs e)
{
   this.timer.Stop();
   pb.Value = pb.Minimum;
   this.timer.Start();
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.


Loading... 49%

Test Application Features

- Click the Start, Stop and Restart buttons.

Ideas for Extending This Example

- Change the RadProgressBar in MainPage.xaml to use different Minimum and Maximum values. Be aware that there is no safety code checking that Minimum is not greater than or equal to the Maximum value.
- In the code behind for the UserControl Loaded event, change the Interval so that updates are more or less frequent.
- Set the **SkipValue** property to a value between Minimum and Maximum.
- Turn on the **IsIndeterminate** property and observe the effect on the Value property and on the progress indicator.

21.4 Wrap Up

In this chapter you used RadProgressBar to visually notify the user about processes of known and indeterminate length. You worked with the progress bar orientation, minimum/maximum values and also learned how to update the progress bar value.

Part XXI

Charting

22 Charting

22.1 Objectives

In this chapter you will first build a chart declaratively using default settings wherever possible. In the process you will learn the differences between default vs. custom layouts. You will define a series with individual data points and specify the legend and chart title. You will also learn about how the series definition displays the data in a particular arrangement, i.e. bar, pie, 3D line, etc. You will enable the interactivity feature of the chart.

During the exploration of chart control details you will take a tour of the many chart series types that include the standard bar/line/pie, stacked versions that compare contributions of values across categories, stacked bar 100% that show the stacked values as percentages, special purpose charts like "candlestick" for showing stock price and currency changes, and many 3D chart series types as well. You will learn how to create chart series and data points programmatically. If you need to integrate the chart with existing ASP. NET applications, the section on "Integration with ASP.NET" will walk you through how this is done.

As part of the binding chapter you will first learn about series and item mappings that route data to various elements in the chart, including the data point and axis labels. You will display tool tips as simple text, use special formatting tokens to show data as currency or as percentages of all categories and you will learn how to display a tool tip that shows another chart as a drill-down.

You will apply styles to customize elements of the chart. You will use the MVVM pattern to display chart elements in colors corresponding to data in the model.

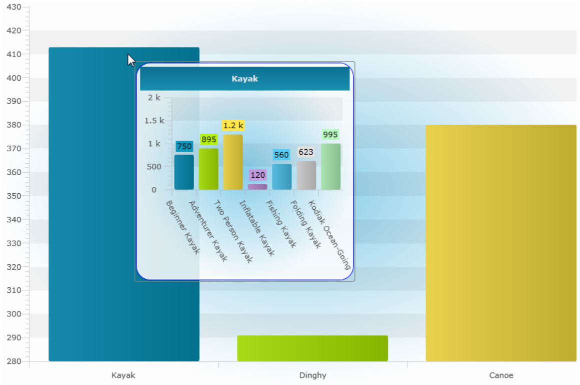
Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Chart\Chart.sln.

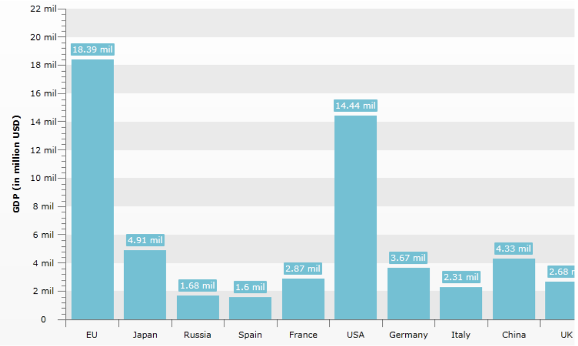
22.2 Overview

With RadChart you can transform business scenarios into interactive, rich, animated charts. RadChart takes data visualization to another dimension and is the first commercial 3D Chart for Silverlight. RadChart features include:

- 20 2D Charts Types for Silverlight
- 9 3D Chart Types for Silverlight
- Rich Data Binding Support
- Automated Data Binding to Nested Collections
- Tooltip Support, from simple text, formatted data or drill-downs into charts or other Silverlight elements.



- Flexible API
- MVVM Support
- Advanced X Axis Capabilities
- Strict Mode for X Axis
- Easily Customizable Series
- Flexible Layout
- Axis AutoRange Functionality
- Axis AutoStep Functionality
- Styling and Appearance
- Animations and Interactivity
- Categorical Axis



22.3 Getting Started

In this walk through you will build a simple chart declaratively.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Charting**

c) **Telerik.Windows.Data**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add references to **Telerik.Windows.Controls** and **Telerik.Windows.Controls.Charting** XML namespaces, both from the **Telerik.Windows.Controls.Charting** assembly.

```
<UserControl
xmlns:control=
"clr-namespace:Telerik.Windows.Controls;assembly=Telerik.Windows.Controls.Charting"
xmlns:chart=
"clr-namespace:Telerik.Windows.Controls.Charting;assembly=Telerik.Windows.Controls.Charting"
. . .>
```

3) Drag a RadChart from the Toolbox to a point between the main "LayoutRoot" grid begin and end tags.

4) Setup the basic chart structure.

a) Inside the RadChart element, add a RadChart.**DefaultView**. Notice that DefaultView uses the "control" XML namespace reference.

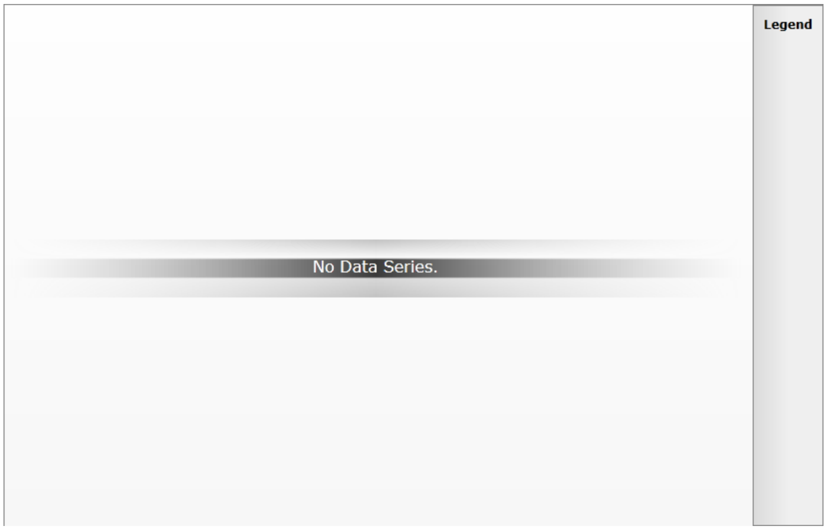
b) Inside the DefaultView, add a **ChartDefaultView**. Notice that ChartDefaultView uses the "chart" XML namespace reference.

c) Add the comments shown below to indicate where we will place the ChartArea, ChartLegend and ChartTitle elements.

The XAML markup should look like the example below.

```
<control:RadChart>
  <control:RadChart.DefaultView>
    <chart:ChartDefaultView>

      <!--ChartArea-->

      <!--ChartLegend-->

      <!--ChartTitle-->

    </chart:ChartDefaultView>
  </control:RadChart.DefaultView>
</control:RadChart>
```

If you ran the application right now, it would report "No Data Series.", as shown in the screenshot below.



Notes

RadChart has a **UseDefaultLayout** property. By default, this property is true and the chart expects a single ChartArea, ChartLegend and ChartTitle. When UseDefaultLayout is false RadChart can contain an arbitrary arrangement of any number of chart elements (analogous to how gauges allow any number of gauge types - see the Gauges chapter). UseDefaultLayout is a handy way to avoid completely overriding a control template. For example, you could have a pie chart along side another chart showing a bar graph, and place the legend centered below the two chart areas and leave the chart title off altogether. The arrangement can be defined solely by your requirements.

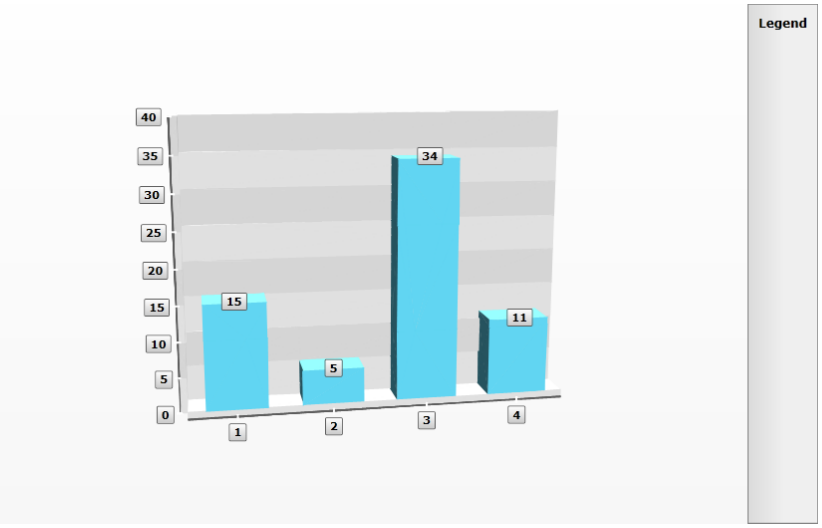
5) Replace the "<!--ChartArea-->" comment with the XAML below.

The ChartDefaultView contains a ChartArea and within the ChartArea, a DataSeries. The DataSeries. Definition element determines the type of series it will be, i.e. pie, line, 3D bar, etc. Then a collection of DataPoint elements list the data that will display in the series. Bar charts are relatively simple and only need a single "Y" value for each data point. Note that the properties you fill out for each DataPoint vary depending on the type of series and the amount of data it requires. For example, a "Candlestick" series require four different values for each data point.

The series below defines a Bar3DSeriesDefinition with four data points, each with a YValue defined.

```
<!--ChartArea-->
<chart:ChartDefaultView.ChartArea>
   <chart:ChartArea>
      <chart:ChartArea.DataSeries>
         <chart:DataSeries>
            <chart:DataSeries.Definition>
               <chart:Bar3DSeriesDefinition />
            </chart:DataSeries.Definition>
            <chart:DataPoint YValue="15" />
            <chart:DataPoint YValue="5" />
            <chart:DataPoint YValue="34" />
            <chart:DataPoint YValue="11" />
         </chart:DataSeries>
      </chart:ChartArea.DataSeries>
   </chart:ChartArea>
</chart:ChartDefaultView.ChartArea>
```

If you ran the project, now you would see the data displayed as a 3D bar chart. There is a default "Legend" label in the upper right corner and no title as yet.

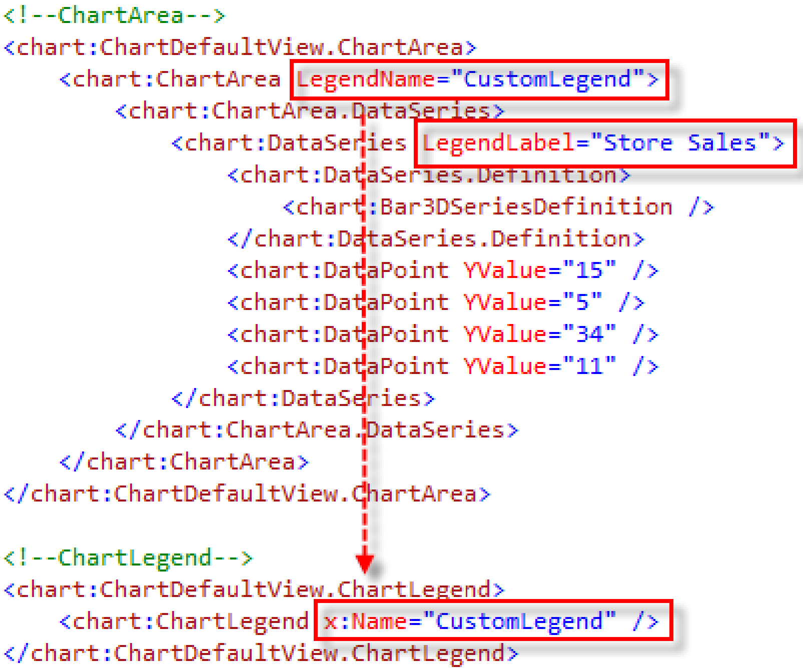


6) Replace the "<!--ChartLegend-->" comment with the XAML below to define the ChartLegend and use its defaults.

In a moment we will go back and point the LegendName property of the ChartArea at this element.

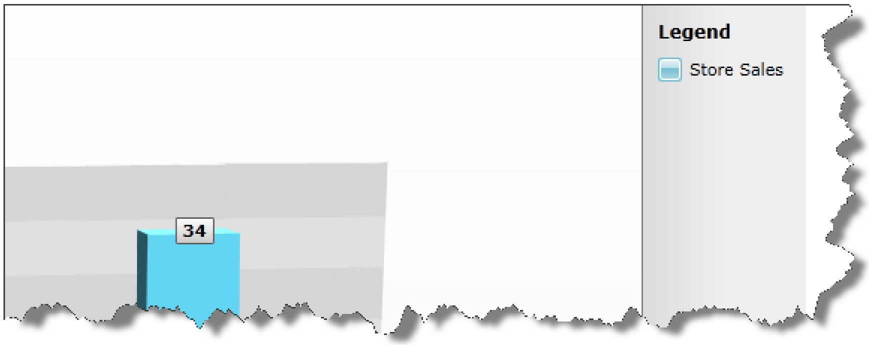
```
<!--ChartLegend-->
<chart:ChartDefaultView.ChartLegend>
   <chart:ChartLegend x:Name="CustomLegend" />
</chart:ChartDefaultView.ChartLegend>
```

7) Go back to the ChartArea element and add a **LegendName** attribute with a value of "CustomLegend". Also, add a **LegendLabel** attribute to the DataSeries element and set the value to "Store Sales".



```
<!--ChartArea-->
<chart:ChartDefaultView.ChartArea>
    <chart:ChartArea LegendName="CustomLegend">
        <chart:ChartArea.DataSeries>
            <chart:DataSeries LegendLabel="Store Sales">
                <chart:DataSeries.Definition>
                    <chart:Bar3DSeriesDefinition />
                </chart:DataSeries.Definition>
                <chart:DataPoint YValue="15" />
                <chart:DataPoint YValue="5" />
                <chart:DataPoint YValue="34" />
                <chart:DataPoint YValue="11" />
            </chart:DataSeries>
        </chart:ChartArea.DataSeries>
    </chart:ChartArea>
</chart:ChartDefaultView.ChartArea>

<!--ChartLegend-->
<chart:ChartDefaultView.ChartLegend>
    <chart:ChartLegend x:Name="CustomLegend" />
</chart:ChartDefaultView.ChartLegend>
```

*You can see that the "Store Sales" label shows up in the legend. Note: The ChartLegend element has a **UseAutoGeneratedItems** property ("True" by default) that makes all DataSeries labels display automatically in the legend.*

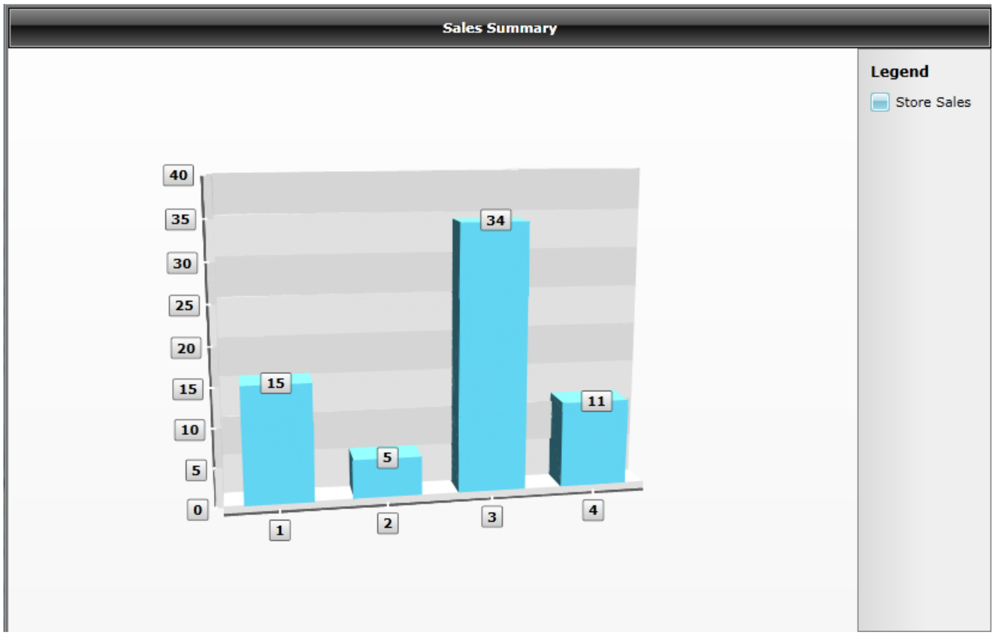


8) Replace the "<!--ChartTitle-->" comment with the XAML below. *This new element will display a chart title "Sales Summary" at the top of the page.*

```
<!--ChartTitle-->
<chart:ChartDefaultView.ChartTitle>
   <chart:ChartTitle Content="Sales Summary" />
</chart:ChartDefaultView.ChartTitle>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

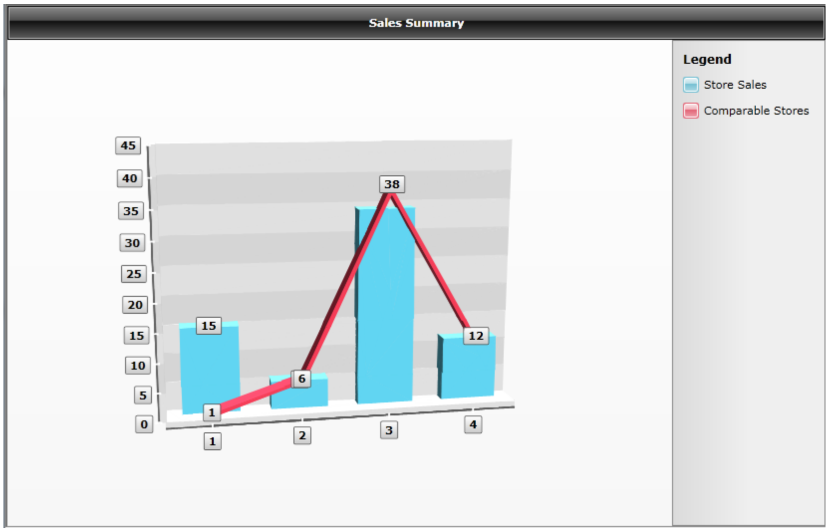
1) The application should display four data points in 3D bar format, a legend that reads "Store Sales" and a title, "Sales Summary".

Ideas for Extending This Example

- Try this: copy the DataSeries element labeled "Store Sales". Change the copied DataSeries Label to "Comparable Stores" and change the DataSeries.Definition to **Line3DSeriesDefintion**. Change the DataPoint YValue attributes to different values.

```
<chart:ChartArea.DataSeries>
  <chart:DataSeries LegendLabel="Store Sales">
    <chart:DataSeries.Definition>
      <chart:Bar3DSeriesDefinition />
    </chart:DataSeries.Definition>
    <chart:DataPoint YValue="15" />
    <chart:DataPoint YValue="5" />
    <chart:DataPoint YValue="34" />
    <chart:DataPoint YValue="11" />
  </chart:DataSeries>
  <chart:DataSeries LegendLabel="Comparable Stores">
    <chart:DataSeries.Definition>
      <chart:Line3DSeriesDefinition/>
    </chart:DataSeries.Definition>
    <chart:DataPoint YValue="1" />
    <chart:DataPoint YValue="6" />
    <chart:DataPoint YValue="38" />
    <chart:DataPoint YValue="12" />
  </chart:DataSeries>
</chart:ChartArea.DataSeries>
```

The second series should display over the first as a 3D line chart and add a new "Comparable Stores" entry in the legend.

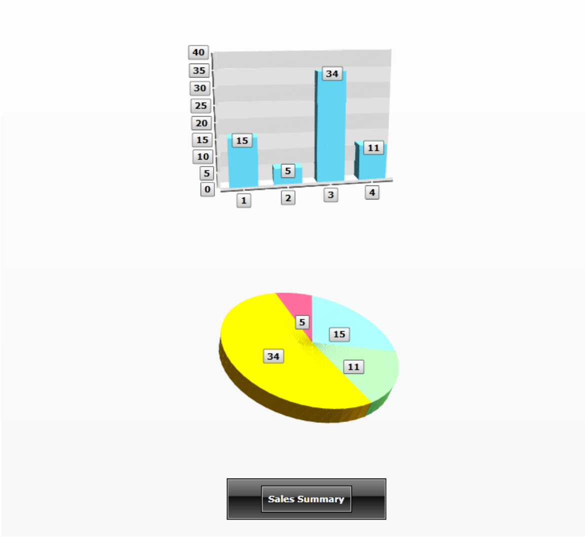


- Try this: arrange the elements of the chart to make your own custom layout.

  Set the chart UseDefaultLayout property to "False". Then remove all the references to ChartDefaultView. The example below uses a StackPanel to arrange the chart elements vertically. There are two ChartArea elements, where the second is a 3D pie chart, the Legend is removed altogether and the ChartTitle appears at the bottom of the page.

```
<control:RadChart UseDefaultLayout="False">

  <StackPanel>
    <!--ChartArea-->
    <chart:ChartArea MaxHeight="300" MaxWidth="300">
      <chart:ChartArea.DataSeries>
        <chart:DataSeries>
          <chart:DataSeries.Definition>
            <chart:Bar3DSeriesDefinition />
          </chart:DataSeries.Definition>
          <chart:DataPoint YValue="15" />
          <chart:DataPoint YValue="5" />
          <chart:DataPoint YValue="34" />
          <chart:DataPoint YValue="11" />
        </chart:DataSeries>
      </chart:ChartArea.DataSeries>
    </chart:ChartArea>

    <!--ChartArea-->
    <chart:ChartArea MaxHeight="300" MaxWidth="300">
      <chart:ChartArea.DataSeries>
        <chart:DataSeries>
          <chart:DataSeries.Definition>
            <chart:Pie3DSeriesDefinition/>
          </chart:DataSeries.Definition>
          <chart:DataPoint YValue="15" />
          <chart:DataPoint YValue="5" />
          <chart:DataPoint YValue="34" />
          <chart:DataPoint YValue="11" />
        </chart:DataSeries>
      </chart:ChartArea.DataSeries>
    </chart:ChartArea>

    <!--ChartTitle-->
    <chart:ChartTitle MaxHeight="100" MaxWidth="100">
      <chart:ChartTitle Content="Sales Summary" />
    </chart:ChartTitle>

  </StackPanel>

</control:RadChart>
```

The XAML above results in a page layout similar to the screenshot below.



From the Forums...

Question: Is RadChart interactive out of the box? Can I interact with the chart, i.e rotate in 3d space, using my mouse pointer?

Answer: RadChart supports mouse interactivity through instantiating a **CameraExtension** instance.

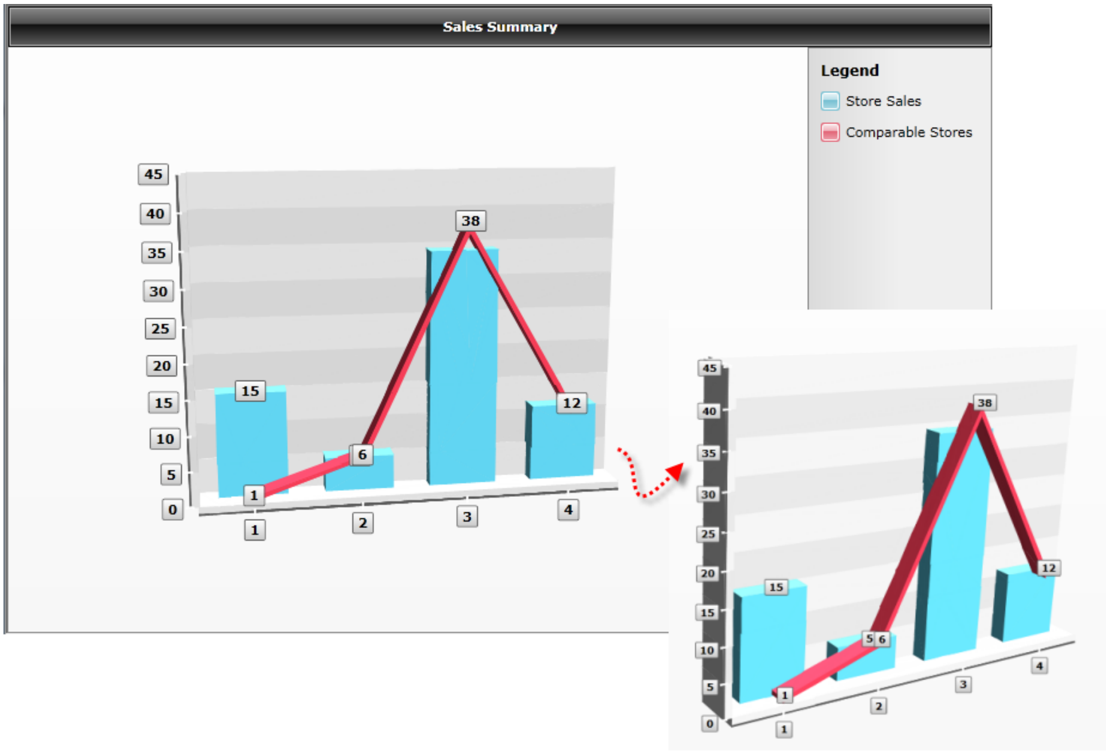
VB

```
'needed for the 3D zoom & interactivity out-of-the-box
chart.DefaultView.ChartArea.Extensions.Add(New CameraExtension())
```

C#

```
//needed for the 3D zoom & interactivity out-of-the-box
chart.DefaultView.ChartArea.Extensions.Add(new CameraExtension());
```

Now the user can use the mouse to drag across the chart and rotate the chart image in all directions.



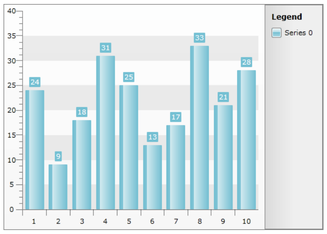
RadChart supports extensibility for the creation of custom chart extensions via the Extensible Object Pattern. This pattern enables an object to participate in custom behavior, such as registering for events, or watching state transitions. The camera extension is included "out of the box". See the online help article "Chart Extensions" for more information.

22.4 Control Details

22.4.1 Chart Series Types

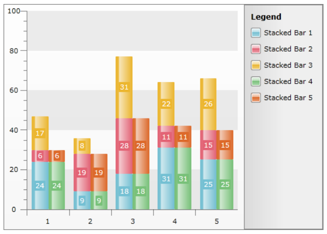
At the time of this writing there are twenty-four 2D charts and ten 3D charts. The following is a quick reference to the available chart types and some of the typical uses. Be sure to check www.telerik.com for the latest developments!

Bar



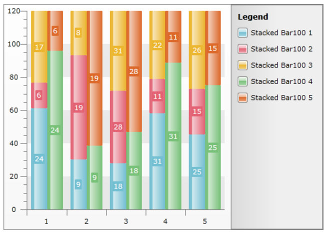
Bar charts graphically display values in vertical and horizontal bars across categories. Bar charts are useful for comparing multiple series of data (i.e. providing snapshots of data at particular points in time).

Stacked Bar



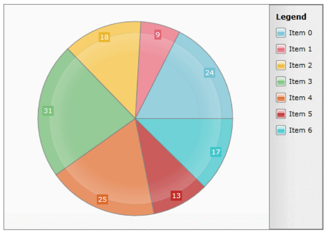
Stacked Bar charts are used to compare contributions of values to a total across categories. Use the Stacked Bar chart when you need visibility to the combined values for each category.

Stacked Bar 100%



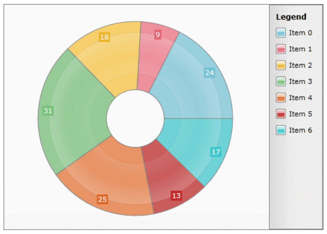
Stacked Bar 100% shows the combined contribution of values as percentages where the combined total for each category is 100 percent. Use when the relationship between values in a category is more significant than the amounts.

Pie



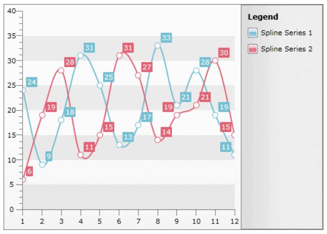
The Pie chart shows slices representing fractional parts of a whole. Use when you need to display the contribution of fractional parts to a whole.

Doughnut



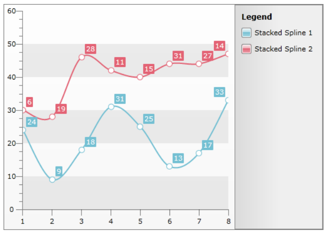
The Doughnut chart is very close to the Pie chart. The only difference is that it uses a doughnut shape instead of the solid pie. Use when you need to display the contribution of fractional parts to a whole.

Spline



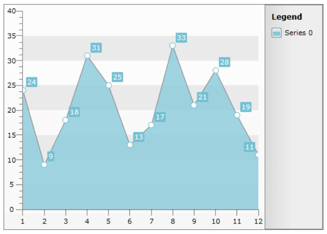
Spline charts allow you to take a limited set of known data points and approximate intervening values. The Spline chart is often used for data modeling by taking a limited number of data points and interpolating or estimating the intervening values.

Stacked Spline



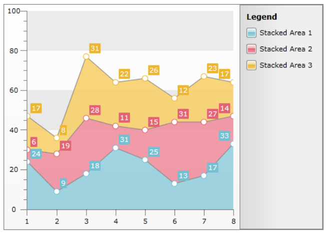
Use the Stacked Spline when you need to show the correlation between two or more series of data visualized as splines.

Spline Area



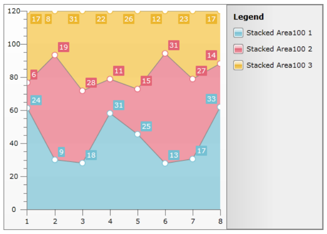
The Spline Area chart type defines one or more spline curves and fills in the area defined by the spline. Can be used for data modeling in that it takes a limited number of data points and interpolates the intervening values. This chart type also emphasizes the area between the spline curve and a mid-point of the spline.

Stacked Spline Area



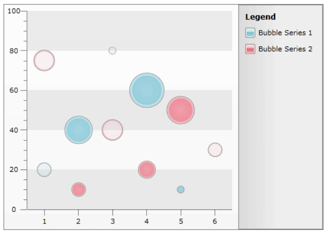
The Stacked Spline Area chart is a variation of the Spline Area chart. The areas are stacked so that each series adjoins but does not overlap the preceding series. Can be used for data modeling in that it takes a limited number of data points and interpolates the intervening values. This chart type allows the entire surface area for all sequences to be displayed at one time.

Stacked Spline Area 100%



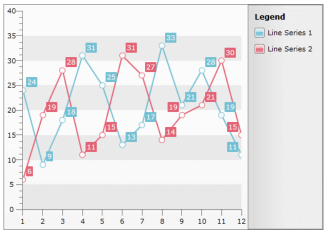
The Stacked Spline Area 100% chart is a variation of the Spline Area chart. The areas are stacked so that each series adjoins but does not overlap the preceding series and where the combined total for each category is 100 percent. Can be used for data modeling in that it takes a limited number of data points and interpolates the intervening values. This chart type allows the entire surface area for all sequences to be displayed at one time. Use this chart type when the relationship between values in a category is more significant than the amounts.

Bubble



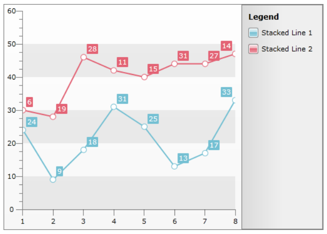
The Bubble chart show correlations between sets of values. The bubble size is used to convey larger values. The Bubble chart is often used for scientific data modeling or financial data.

Line



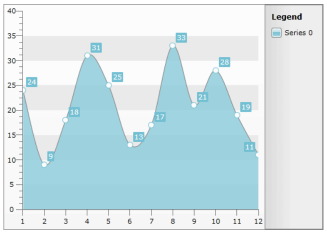
This chart type displays a set of data points connected by a line. A common use for the line chart is to show trends over a period of time.

Stacked Line



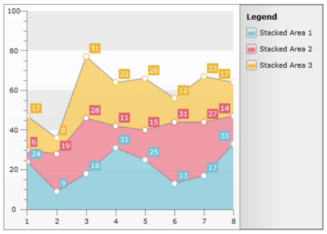
Use the Stacked Line when you need visibility to the combined values of two or more series.

Area



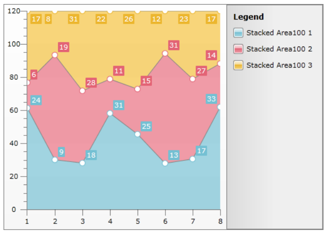
The Area chart consists of a series of data points joined by a line where the area below the line is filled. Area charts are appropriate for visualizing data that fluctuates over a period of time and can be useful for emphasizing trends.

Stacked Area



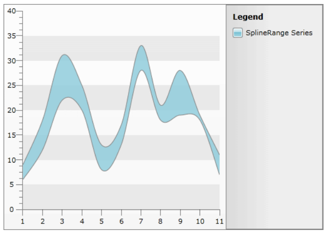
The Stacked Area chart is a variation of the Area chart that displays trends of the contribution of each value over time (or across categories). The areas are stacked so that each series adjoins but does not overlap the preceding series. Stacked Area charts are appropriate for visualizing data that fluctuates over a period of time and where the entire area for all series data must be visible at one time.

Stacked Area 100%



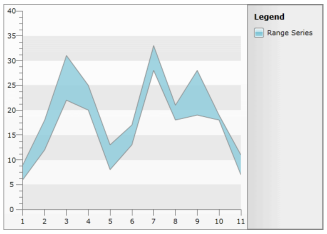
Stacked Areas 100% charts are a variation of Stacked Area charts that present values for trends as percentages, totaling to 100% for each category. Use this chart type to visualize data that fluctuates over a period of time and where the relationship between values in a category is more significant than the amounts.

Range



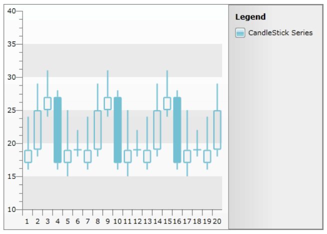
A range chart type displays a set of data points that are each defined by multiple values for the same category. Values are represented by the height of the marker as measured by the value axis. The range chart fills in the area between the top and bottom value for each data point. Range charts are often used to graph data that contains minimum and maximum values for each category group in the dataset.

SplineRange



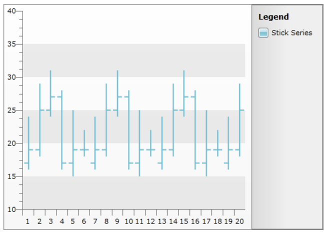
A spline range chart type displays a set of data points that are each defined by multiple values for the same category. Values are represented by the height of the marker as measured by the value axis. The spline range chart fills in the area between the top and bottom value for each data point. Spline range charts are often used to graph data that contains minimum and maximum values for each category group in the dataset.

CandleStick



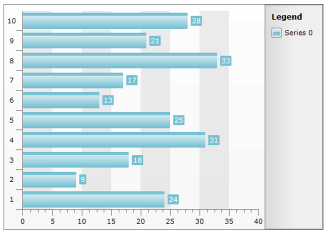
The CandleStick chart combines bar and line chart styles to show a range of value movement over time. Dark colored bars show downward trends, light colored bars show upward trends and the line through the center (the "wick") shows the extreme high and low values. Use this chart type to visualize price or currency fluctuations. Typically this chart is used to analyze stock prices or currency changes.

Stick



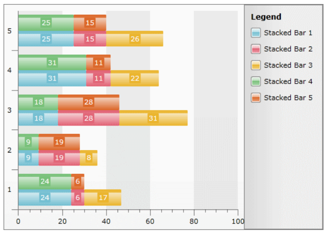
The Stick, like the CandleStick chart shows a range of value movement over time. It uses only lines to do so. The left tick shows the "open" value, while the right one represents the "close" value. The vertical line shows the extreme high and low values. Use this chart type to visualize price or currency fluctuations. Typically this chart is used to analyze stock prices or currency changes.

Horizontal Bar



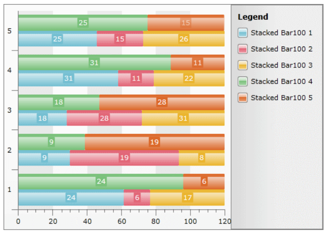
Horizontal bar charts are typically used to compare two or more series of data. Also, for categorical charts, you have more space for labels on the Y-Axis. The labels read as a list from top to bottom. These charts might be used to represent data that has a defined start and an end date.

Horizontal Stacked Bar



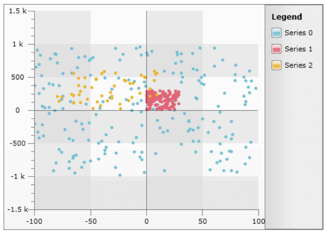
Use the Horizontal Stacked Bar chart when you need visibility to the combined values for each category. Suitable when you have more than three data series.

Horizontal Stacked Bar 100%



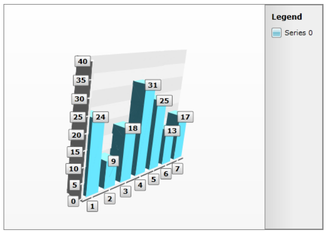
Use when the relationship between values in a category is more significant than the amounts. Suitable when you have more than three data series.

Scatter



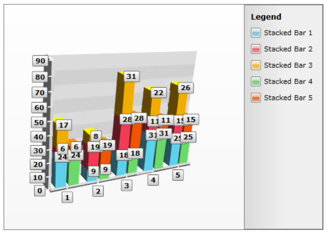
Use when you have to compare aggregated data across categories.

Bar 3D



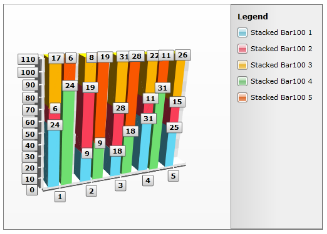
As Bar charts do, the Bar3D charts graphically display values in vertical and horizontal bars across categories. Bar3D charts are useful for comparing multiple series of data (i.e. providing snapshots of data at particular points in time).

Stacked Bar 3D

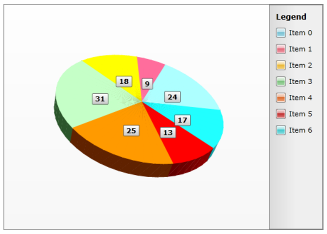
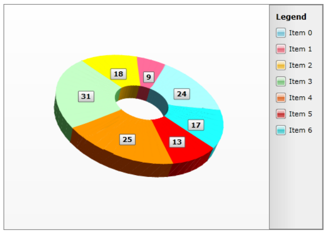
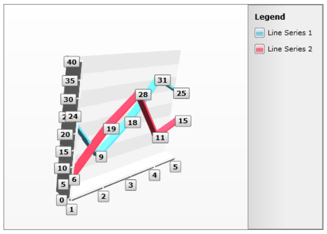
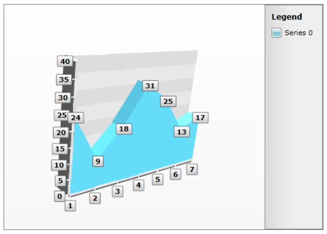
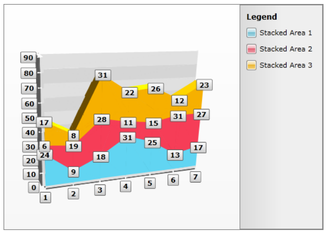
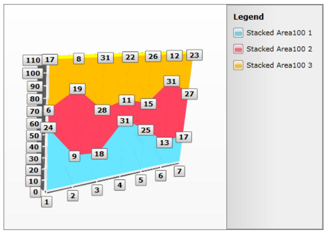


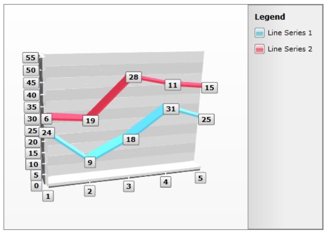
Similar to the Stacked Bar chart, StackedBar3D is used to compare contributions of values to a total across categories. Use the StackedBar3D chart when you need visibility to the combined values for each category.

Stacked Bar 100% 3D



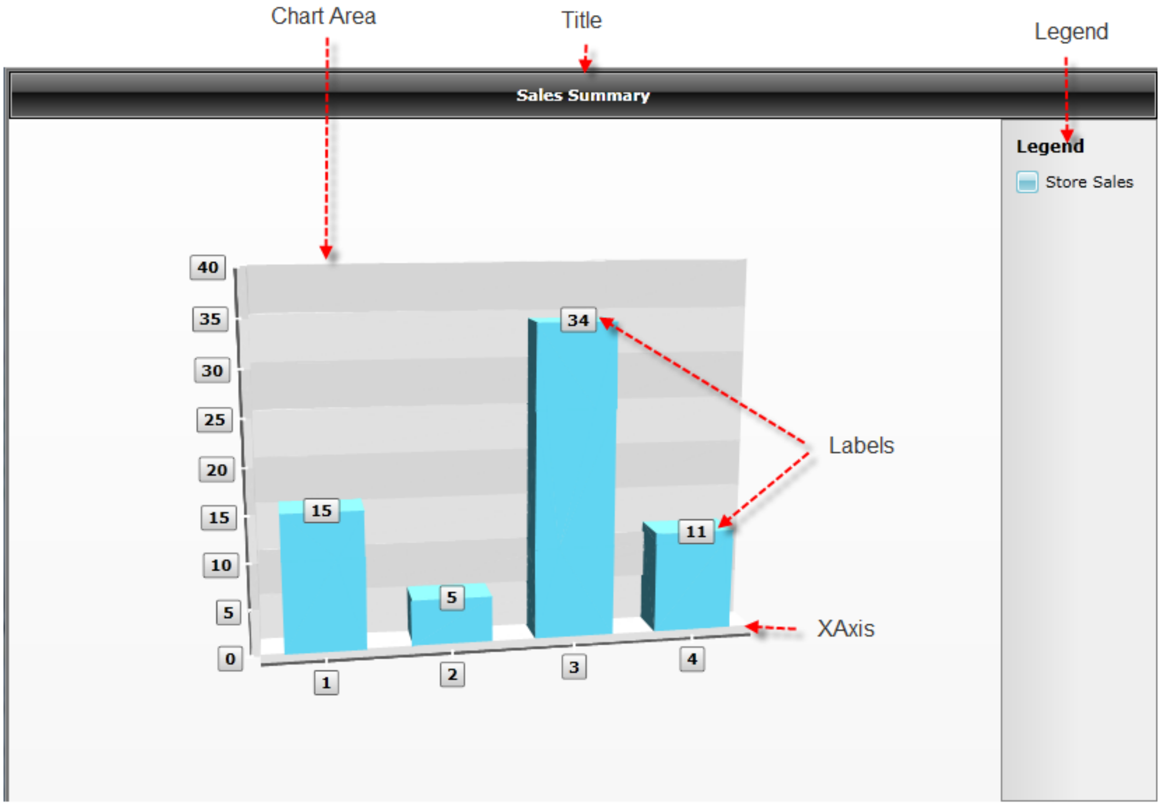
Similar to the Stacked Bar 100% chart, Stacked Bar 100% 3D shows the combined contribution of values as percentages where the combined total for each category is 100 percent. Use when the relationship between values in a category is more significant than the amounts.

Pie 3D	 	The Pie3D chart shows slices representing fractional parts of a whole. Use when you need to display the contribution of fractional parts to a whole.
Doughnut 3D	 	Same as Pie3D chart, but leaving the center empty (for additional Pie3D/Doughnut3D series). Use when you need to display the contribution of fractional parts to a whole.
Line 3D	 	Line3D chart type displays a set of data points connected by a line in a 3D scene. A common use for the line chart is to show trends over a period of time.
Area 3D	 	The Area 3D chart consists of a series of data points joined by a line where the area below the line is filled. Area charts are appropriate for visualizing data that fluctuates over a period of time and can be useful for emphasizing trends.
Stacked Area 3D	 	The Stacked Area 3D chart is a variation of the Area 3D chart that displays trends of the contribution of each value over time (or across categories). The areas are stacked so that each series adjoins but does not overlap the preceding series. Stacked Area charts are appropriate for visualizing data that fluctuates over a period of time and can be useful for emphasizing trends.
Stacked Area 100% 3D	 	The Stacked Area 100% 3D charts charts are a variation of Stacked Area charts that present values for trends as percentages, totaling to 100% for each category. Use this chart type to visualize data that fluctuates over a period of time and where the relationship between values in a category is more significant than the amounts.

Stacked Line 3D		The Stacked Line 3D chart is a variation of the Stacked Line Chart that displays a set of data points connected by a line, but the lines are stacked so that each series adjoins but does not overlap the preceding series.

22.4.2 Chart Elements

The three main areas of the chart are **ChartTitle**, **ChartArea** and **ChartLegend**. Inside the ChartArea are the chart series that in turn contain data points. At the outer edge of the ChartArea you can find the AxisY and AxisX objects.



22.4.2.1 Series and DataPoints

You can replicate everything you did declaratively during the Getting Started walk through, in code. If you have a chart with essentially no content...

```
<control:RadChart x:Name="chart" />
```

You can build up the individual elements that parallel the XAML. You still need to create a DataSeries, assign its Definition and fill it with DataPoints. You create a ChartLegend and add the DataSeries, create a ChartLegend and assign its Name and finally, create a ChartTitle and assign its Content. A few notes to remember here:

- The ChartArea LegendName needs to match the ChartLegend's Name property.
- Be aware that the ChartArea DataSeries is a DataSeriesCollection while the stand-alone DataSeries type is a collection of DataPoint.

VB

```
Public Sub New()
  InitializeComponent()
  ' create a series and set its definition
  Dim series As New DataSeries() With { _
.Definition = New Bar3DSeriesDefinition(), .LegendLabel = "Store Sales"}

  ' add data to the series
  series.Add(New DataPoint(15))
  series.Add(New DataPoint(5))
  series.Add(New DataPoint(34))
  series.Add(New DataPoint(11))

  ' Create and populate the chart area
  chart.DefaultView.ChartArea = New ChartArea() With { _
.LegendName = "CustomLegend"}

  ' add series to the area
  chart.DefaultView.ChartArea.DataSeries.Add(series)

  ' create the populate legend
  chart.DefaultView.ChartLegend = New ChartLegend() With { _
.Name = "CustomLegend"}

  ' create and populate the chart title
  chart.DefaultView.ChartTitle = New ChartTitle() With { _
.Content = "Sales Summary"}
End Sub
```

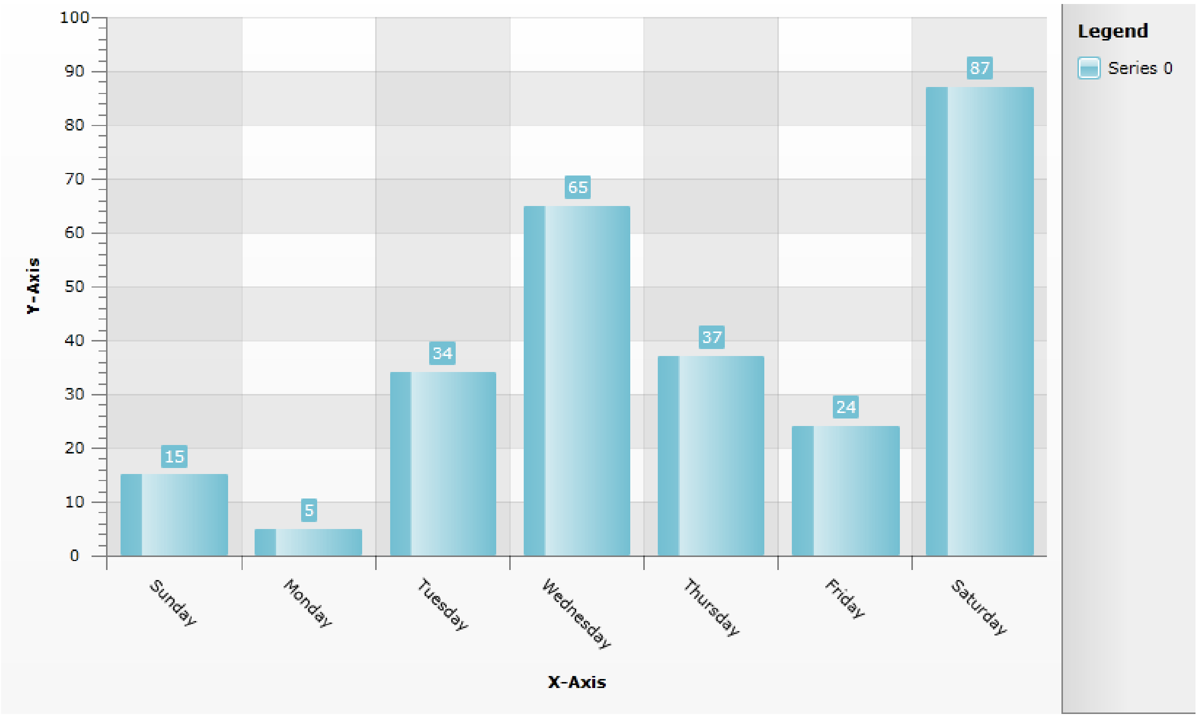
```
public MainPage()
{
    InitializeComponent();
  // create a series and set its definition
  DataSeries series = new DataSeries()
  {
    Definition = new Bar3DSeriesDefinition(),
    LegendLabel = "Store Sales"
  };

  // add data to the series
  series.Add(new DataPoint(15));
  series.Add(new DataPoint(5));
  series.Add(new DataPoint(34));
  series.Add(new DataPoint(11));

  // Create and populate the chart area
  chart.DefaultView.ChartArea = new ChartArea()
  {
    LegendName = "CustomLegend"
  };

  // add series to the area
  chart.DefaultView.ChartArea.DataSeries.Add(series);

  // create the populate legend
  chart.DefaultView.ChartLegend = new ChartLegend()
  {
    Name = "CustomLegend"
  };

  // create and populate the chart title
  chart.DefaultView.ChartTitle = new ChartTitle()
  {
    Content = "Sales Summary"
  };
}
```

22.4.2.2 Axis Elements

Use the Axis properties to control the axis Title, visibility of the axis itself, the grid lines and the strip lines (strip lines are the alternating bands of color on the background). These properties are controlled by the Telerik.Windows.Controls.Charting **Axis** class. Properties specific to each axis, i.e. X or Y, are controlled by the corresponding **AxisX** and **AxisY** classes. AxisY for instance has an **ExtendDirection** property that controls the addition of "head room" above or below the chart area. AxisX has a **LabelRotationAngle** used to spin the label such that more data points can fit

The example below sets the titles for both axis. The visibility for the axis, strip lines and grid lines are enabled explicitly for your reference.

Set the **IsDateTime** property "True" to display DateTime values on the X-axis. You can also use the **DefaultLabelFormat** to display dates according to a date and time format. In the example, the format is "dddd" and displays the day names.



```
chart.DefaultView.ChartArea.AxisX.Title = "X-Axis"
chart.DefaultView.ChartArea.AxisX.Visibility = Visibility.Visible
chart.DefaultView.ChartArea.AxisX.StripLinesVisibility = _
Visibility.Visible
chart.DefaultView.ChartArea.AxisX.MajorGridLinesVisibility =  _
Visibility.Visible
chart.DefaultView.ChartArea.AxisX.AutoRange = True
chart.DefaultView.ChartArea.AxisX.IsDateTime = True
chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45
chart.DefaultView.ChartArea.AxisX.DefaultLabelFormat = "dddd"

chart.DefaultView.ChartArea.AxisY.Title = "Y-Axis"
chart.DefaultView.ChartArea.AxisY.Visibility = Visibility.Visible
chart.DefaultView.ChartArea.AxisY.StripLinesVisibility =  _
Visibility.Visible
chart.DefaultView.ChartArea.AxisY.MajorGridLinesVisibility =  _
Visibility.Visible
chart.DefaultView.ChartArea.AxisY.ExtendDirection =  _
AxisExtendDirection.Smart
```

```
chart.DefaultView.ChartArea.AxisX.Title = "X-Axis";
chart.DefaultView.ChartArea.AxisX.Visibility = Visibility.Visible;
chart.DefaultView.ChartArea.AxisX.StripLinesVisibility = Visibility.Visible;
chart.DefaultView.ChartArea.AxisX.MajorGridLinesVisibility = Visibility.Visible;
chart.DefaultView.ChartArea.AxisX.AutoRange = true;
chart.DefaultView.ChartArea.AxisX.IsDateTime = true;
chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45;
chart.DefaultView.ChartArea.AxisX.DefaultLabelFormat = "dddd";

chart.DefaultView.ChartArea.AxisY.Title = "Y-Axis";
chart.DefaultView.ChartArea.AxisY.Visibility = Visibility.Visible;
chart.DefaultView.ChartArea.AxisY.StripLinesVisibility = Visibility.Visible;
chart.DefaultView.ChartArea.AxisY.MajorGridLinesVisibility = Visibility.Visible;
chart.DefaultView.ChartArea.AxisY.ExtendDirection = AxisExtendDirection.Smart;
```

22.4.3 Animations

For perceived performance or aesthetic reasons you may need to disable animation altogether. Set the EnableAnimations property of the ChartArea:

```
RadChart1.DefaultView.ChartArea.EnableAnimations = False
```

C#

```
RadChart1.DefaultView.ChartArea.EnableAnimations = false;
```

Animation timings can be modified for the entire chart or for an individual series. The settings for the example below cause the animation for a SeriesDefinition to take 10 seconds. The spline series is drawn slowly, as if being traced on a white board. This setting isn't performant of course, but you may want to display some series more slowly to point out the relationship of data. Quicker animation settings provide visual "pop" without noticeable pause in the rendering. The **ItemAnimationDuration** property controls the time span for the animation of each individual item in the chart. **TotalSeriesAnimationDuration** controls the time span for the animation of the entire series. **ItemDelay** and **DefaultSeriesDelay** delay the the item and series animation.

VB

```
chart.DefaultSeriesDefinition.AnimationSettings = _
New AnimationSettings() With { _
.TotalSeriesAnimationDuration = TimeSpan.FromSeconds(10)}
```

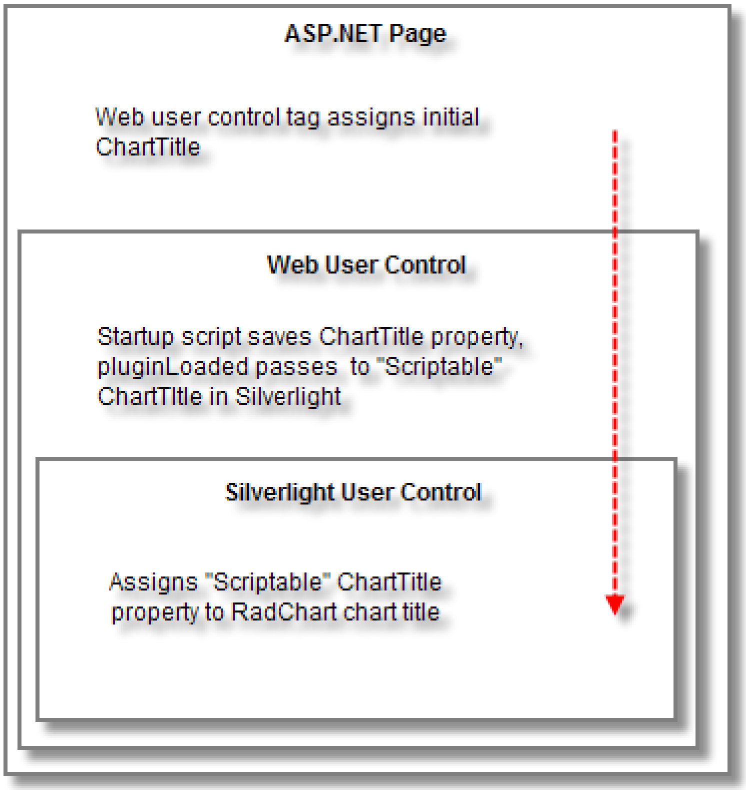
C#

```
chart.DefaultSeriesDefinition.AnimationSettings =
  new AnimationSettings()
  {
    TotalSeriesAnimationDuration = TimeSpan.FromSeconds(10)
  };
```

22.4.4 Integration with ASP.NET AJAX

You may not have the luxury of building Silverlight-only solutions for your business. But you can introduce Silverlight elements into existing ASP.NET applications. Silverlight RadChart, RadGridView or any of the other RadControls can be including as-needed. The technique is to wrap a Silverlight control in a web user control, then supply the plumbing to communicate from the ASP.NET web page that contains the user control, all the way back to the Silverlight control. In the example below, we pass a chart title string all the way from an attribute on the ASP.NET page back to a RadChart on the Silverlight user control.

The key steps are described in the diagram below where the ASP.NET page sets the "ChartTitle" property of a web user control. The web user control uses a startup script to save the ChartTitle value, then, when the Silverlight plugin loads, the saved value is passed to a "Scriptable" property in a Silverlight user control ("Scriptable" means that the Silverlight user control property is available to Javascript). The "Scriptable" ChartTitle is then assigned when the RadChart first loads.


ASP.NET Page
Web user control tag assigns initial ChartTitle
Web User Control
Startup script saves ChartTitle property, pluginLoaded passes to "Scriptable" ChartTItle in Silverlight
Silverlight User Control
Assigns "Scriptable" ChartTitle property to RadChart chart title

All the pieces described below have to be in place for the technique to work, but we will follow the process sequentially from Silverlight page, through the web user control and finally, to the ASP.NET page.

Before getting started with the steps below, first create a Silverlight Application along with a host ASP.NET application.

Silverlight Page

The Silverlight XAML page can be any arbitrary set of Silverlight elements. In this example we will re-use the XAML from the RadChart Getting Started walk through.

1) In the code-behind for the page add a reference to the **System.Windows.Browser** namespace to support HtmlPage and ScriptableMember objects.

2) In the constructor for the page, call **HtmlPage.RegisterScriptableObject()** and pass a string "slChartPage" and a reference to the page. This line of code registers the page as an object that can be accessed in Javascript. Also in the constructor, add a Loaded event handler to the RadChart.
3) In the Loaded event handler, assign the Content for the chart title. The content will come from a string property of the page to be defined called "ChartTitle". Its important to put this assignment after the chart has been loaded, and we can make that explicit by putting the assignment in the chart's Loaded event handler.
4) Include a string property called "ChartTitle" and decorate it with the ScriptableMember attribute. The combination of RegisterScriptableObject to surface the page and the ScriptableMember attribute to expose the ChartTitle will make both objects accessible later in Javascript from the ASP.NET page.

   The code for the page should look like the example below:

```
Partial Public Class MainPage
  Inherits UserControl
  Public Sub New()
    InitializeComponent()
    ' "slChartPage" makes ChartTitle available to script
    ' in the user control
    HtmlPage.RegisterScriptableObject("slChartPage", Me)
    AddHandler chart.Loaded, AddressOf chart_Loaded
  End Sub

  Private Sub chart_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
    ' !! Gotcha: chart must be loaded before assigning content
    chart.DefaultView.ChartTitle.Content = Me.ChartTitle
  End Sub

  Private privateChartTitle As String
  <ScriptableMember> _
  Public Property ChartTitle() As String
    Get
      Return privateChartTitle
    End Get
    Set(ByVal value As String)
      privateChartTitle = value
    End Set
  End Property
End Class
```

```
public partial class MainPage : UserControl
{
   public MainPage()
   {
      InitializeComponent();
      // "slChartPage" makes ChartTitle available to script
      // in the user control
      HtmlPage.RegisterScriptableObject("slChartPage", this);
      chart.Loaded += new RoutedEventHandler(chart_Loaded);
   }

   void chart_Loaded(object sender, RoutedEventArgs e)
   {
      // !! Gotcha: chart must be loaded before assigning content
      chart.DefaultView.ChartTitle.Content = this.ChartTitle;
   }

   [ScriptableMember]
   public string ChartTitle
   { get; set; }
}
```

User Control

1) In the Solution Explorer, References node of the ASP.NET application, add a reference to the **System.Web.Silverlight** assembly.

2) In the host ASP.NET application, create a Web User Control.

3) Configure the web user control in the designer:

a) Add a reference to System.Web.Silverlight in the markup of the Web User Control.

```
<%@ Register Assembly="System.Web.Silverlight"
   Namespace="System.Web.UI.SilverlightControls"
   TagPrefix="asp" %>
```

b) Add a block of script to the user control with two functions. The global variable "chartTitle" at the top of the script block is used by both functions.

The pluginLoaded() function is triggered from an event of the Silverlight plugin. "sender" is the Silverlight control. The content for the Silverlight control holds a reference to the "slChartPage" object we defined earlier and through that we can also reference the "ChartTitle" property. When we assign the slChartPage.ChartTitle, we're actually setting the chart title of the RadChart back on the Silverlight page.

The second function, setTitleValue(), will be called later from a startup script when the user control first loads.

```
<script type="text/javascript">
  var chartTitle;

  function pluginLoaded(sender) {
    // get reference to the silverlight control on the page
    var silverlightControl = sender.get_element();

    // call scriptable method in silverlight application
    // we call it here to ensure silverlight controls is fully loaded.
    if (chartTitle != null) {
      silverlightControl.content.slChartPage.ChartTitle = chartTitle;
    }
  }

  function setTitleValue(title) {
    chartTitle = title;
  }

</script>
```

c) Add a Silverlight server control inside the web user control. Point the Source property at the path of the "xap" file in the ClientBin directory. Also be sure to hookup the OnPluginLoaded event to the pluginLoaded() function you created in the last step.

```
<asp:Silverlight ID="Xaml1" runat="server"
 Source="~/ClientBin/02B_ControlDetails_Integration.xap"
 MinimumVersion="2.0.31005.0" OnPluginLoaded="pluginLoaded" />
```

4) Add the web user control code-behind.

a) Add a string property that surfaces ChartTitle. This property will be set later from the ASP.NET page as an attribute of the web user control.

b) In Page_Load, if the ChartTitle property has been assigned a value, register a startup script that calls the Javascript setTitleValue() function. The RegisterStartupScript takes the type of the page, an arbitrary string that acts as a key for the script, the Javascript itself and a Boolean indicating that Script tags should be included automatically.

VB

```
Partial Public Class ChartUserControl
  Inherits System.Web.UI.UserControl
  Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
    ' pass values to silverlight in HTML
    If (Not String.IsNullOrEmpty(Me.ChartTitle)) Then
      Dim script As String = _
String.Format("setTitleValue('{0}');", Me.ChartTitle)
      Page.ClientScript.RegisterStartupScript(GetType(Page), _
"SLPROXY_TITLE", script, True)
    End If
  End Sub

  Private privateChartTitle As String
  Public Property ChartTitle() As String
    Get
      Return privateChartTitle
    End Get
    Set(ByVal value As String)
      privateChartTitle = value
    End Set
  End Property
End Class
```

C#

```
public partial class ChartUserControl : System.Web.UI.UserControl
{
   protected void Page_Load(object sender, EventArgs e)
   {
      // pass values to silverlight in HTML
      if (!string.IsNullOrEmpty(this.ChartTitle))
      {
         string script =
            string.Format("setTitleValue('{0}');", this.ChartTitle);
         Page.ClientScript.RegisterStartupScript(
            typeof(Page), "SLPROXY_TITLE", script, true);
      }
   }

   public string ChartTitle
   {
      get;
      set;
   }
}
```

ASP.NET Page

1) Add a reference to the web user control.

```
<%@ Register Src="ChartUserControl.ascx" TagName="ChartUserControl" TagPrefix="uc1" %>
```

2) Add a ScriptManager to the page if one doesn't already exist and add the web user control to the page. Set the ChartTItle property of the web user control to "Category Sales".

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<div>
  <uc1:ChartUserControl ID="ChartUserControl1" runat="server"
   ChartTitle="Category Sales" />
</div>
```

Run the Application

Set the web application as the Startup Application and the ASP.NET page as the Start Page. When you run the application, the RadChart should display the ChartTitle "Category Sales" assigned in the ASP.NET markup.

Here's a summary of the sequence:

1. ASP.NET page markup sets the web user control ChartTitle property.
2. When the web user control page loads on the server, the startup script inserts Javascript to call setTitleValue() and pass the web user control ChartTitle property value.
3. When the web user control loads on the client, the startup script stores the chart title in a global chartTitle variable. Inside the web user control, the Silverlight control triggers the PluginLoaded event. The PluginLoaded event handler gets a reference to the Silverlight control content, gets a reference to the Silverlight user control ChartTitle and assigns the global chartTitle.
4. When the RadChart finishes loading, the Silverlight user control ChartTitle property is assigned to the RadChart.

22.5 Binding

22.5.1 Binding Basics

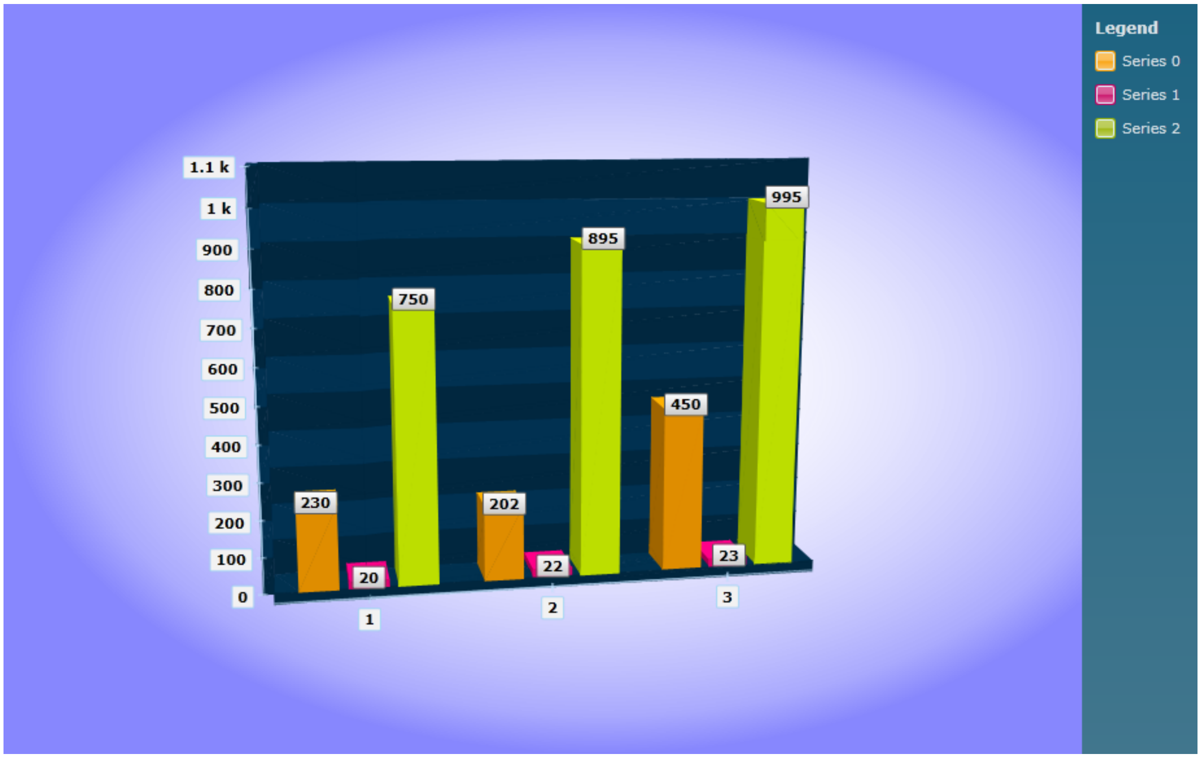
Binding has several components:

1. How do I assign a data source?
2. What columns or properties from the data source should be used?
3. How do the properties from the data source relate to properties on the control being bound?

For a chart series the first question is answered by the **ItemsSource** property. Questions 2 & 3 are answered using "Series Mapping" and "Item Mapping that define binding between data source properties, the series and each data point.

Series mapping can be automatic if you don't need much control and just want to throw data onto the chart. The numeric value properties in a data source are each displayed in their own series. The example below uses a collection of "Product" objects. "Product" has numeric properties for "InStock", "OnOrder" and "Cost". "Product" also has non-numeric columns that are ignored.

The example assigns the chart **DefaultSeriesDefinition** property. The assignment simply takes a specific series definition type, in this case a Bar3DSeriesDefinition.



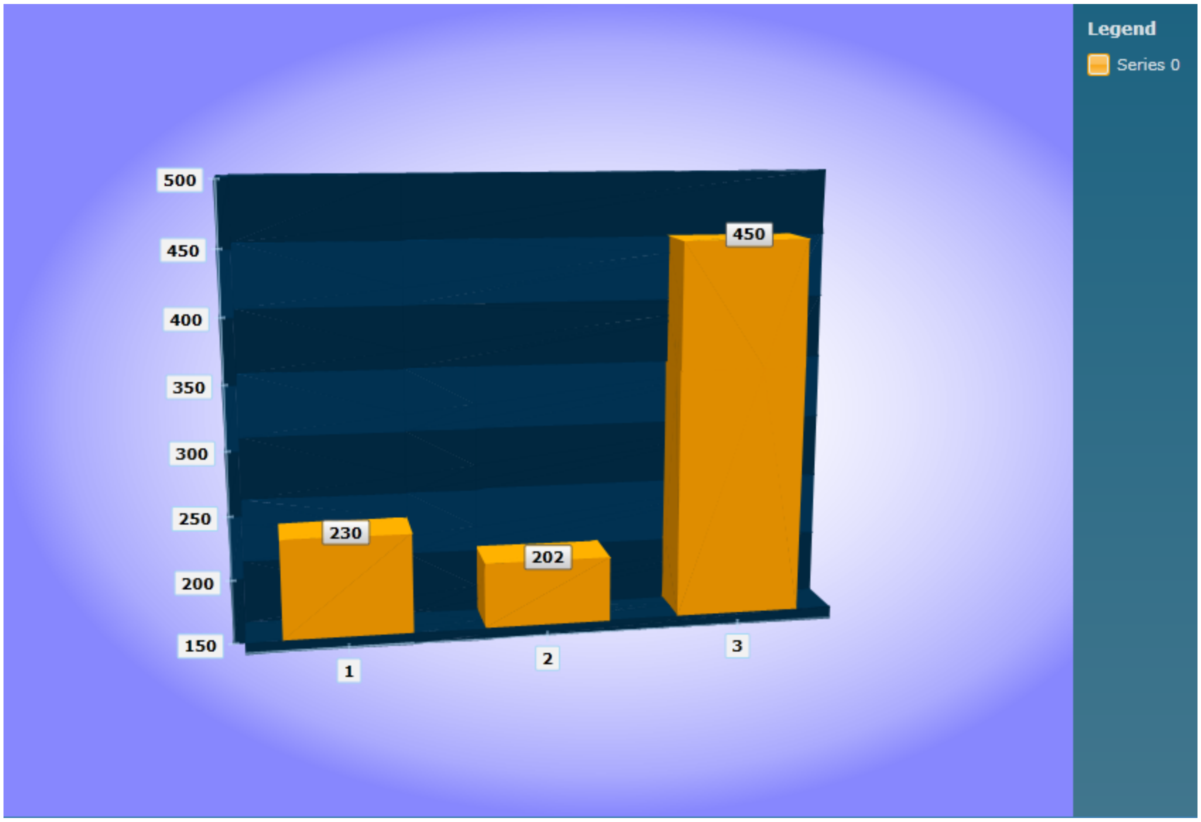
```vb
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim categories As New Categories()
  Dim kayaks As Category = _
categories.FirstOrDefault(Function(c) c.CategoryName.Equals("Kayak"))

  chart.ItemsSource = kayaks.Products
  chart.DefaultSeriesDefinition = New Bar3DSeriesDefinition()
End Sub
```

```csharp
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   Categories categories = new Categories();
   Category kayaks =
      categories.FirstOrDefault(c => c.CategoryName.Equals("Kayak"));

   chart.ItemsSource = kayaks.Products;
   chart.DefaultSeriesDefinition = new Bar3DSeriesDefinition();
}
```

Its more likely we will want control of where the data goes. Each **SeriesMapping** has an **ItemMappings** collection that holds **DataPoint** objects. The relationship is roughly:

```
RadChart.SeriesMappings
  SeriesMapping
    ItemMappings
      ItemMapping
        DataPointMember
```

Now we can tell the chart to bind just the "InStock" data to the **YValue** property of each DataPoint and the chart now looks something like the screenshot below.



VB

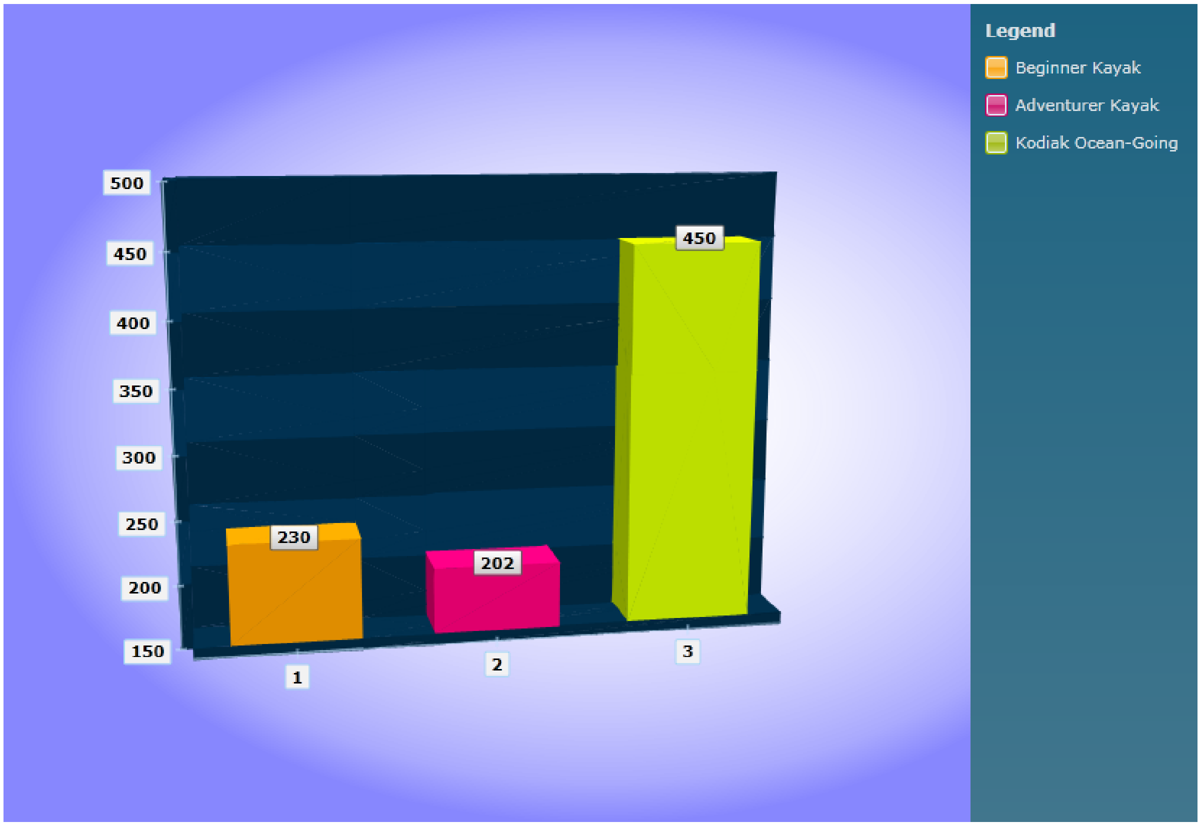
```
chart.ItemsSource = kayaks.Products

Dim seriesMapping As New SeriesMapping() With { _
.SeriesDefinition = New Bar3DSeriesDefinition()}

seriesMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.LegendLabel, .FieldName = "ProductName"})

seriesMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.YValue, .FieldName = "InStock"})

chart.SeriesMappings.Add(seriesMapping)
```

C#

```
chart.ItemsSource = kayaks.Products;

SeriesMapping seriesMapping = new SeriesMapping()
{
 SeriesDefinition = new Bar3DSeriesDefinition()
};

seriesMapping.ItemMappings.Add(new ItemMapping()
{
 DataPointMember = DataPointMember.LegendLabel,
 FieldName = "ProductName"
});

seriesMapping.ItemMappings.Add(new ItemMapping()
{
 DataPointMember = DataPointMember.YValue,
 FieldName = "InStock"
});

chart.SeriesMappings.Add(seriesMapping);
```

Notice that in the last bit of code we attempt to bind both "InStock" to the **YValue** property and "ProductName" to the **LegendLabel** property. But LegendLabel doesn't show up in the running example. To fix that, set the series definition **LegendDisplayMode** property to **DataPointLabel** instead of the default **SeriesLabel**. Now, each element of the "Products" collection shows up both as a data point and a legend element. Because we bound the "ProductName" to the LegendLabel, each legend element display the appropriate product name.

We can polish the results by positioning the Header text to the top of the legend. You can access this through the chart **DefaultView.ChartLegend.Header** property.



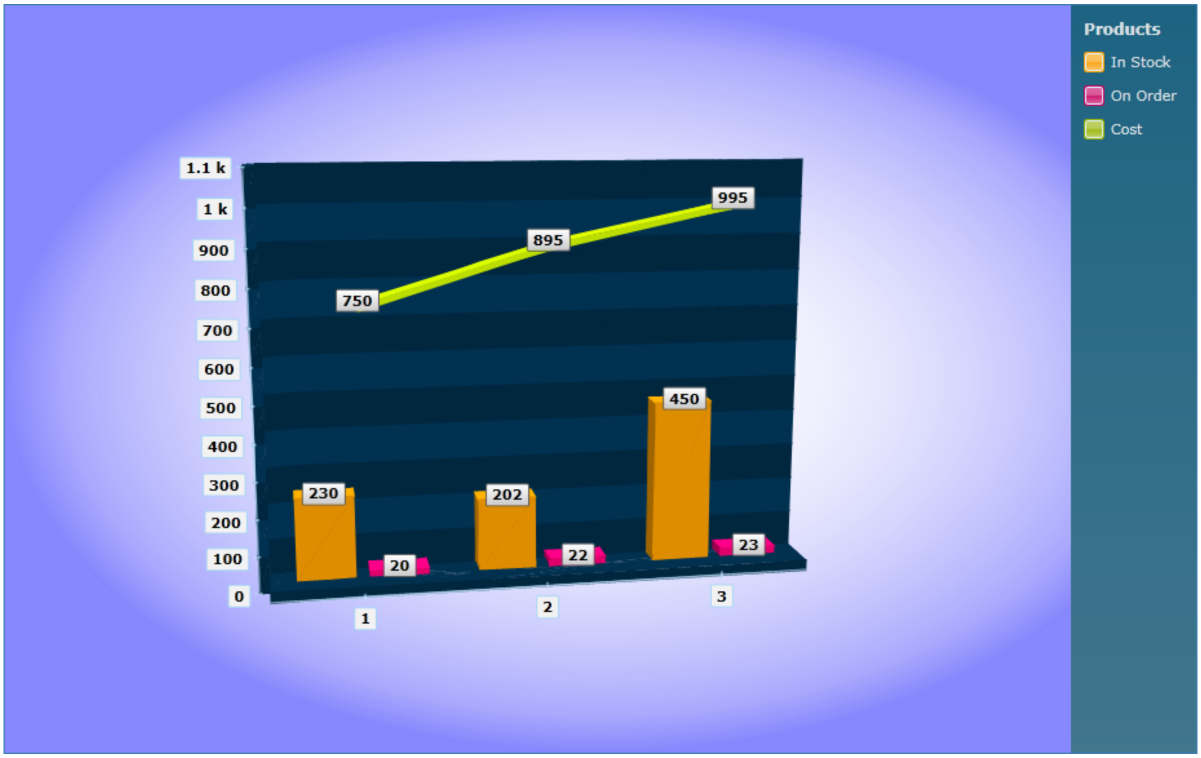
VB

```
chart.ItemsSource = kayaks.Products
chart.DefaultView.ChartLegend.Header = "Products"

Dim seriesMapping As New SeriesMapping() With {. _
SeriesDefinition = New Bar3DSeriesDefinition() With { _
.LegendDisplayMode = LegendDisplayMode.DataPointLabel}}
```

C#

```
chart.ItemsSource = kayaks.Products;
chart.DefaultView.ChartLegend.Header = "Products";

SeriesMapping seriesMapping = new SeriesMapping()
{
   SeriesDefinition = new Bar3DSeriesDefinition()
   {
      LegendDisplayMode = LegendDisplayMode.DataPointLabel,
   }
};
```

Let's look at three series together, displaying the "In Stock", "On Order" and "Cost" data. The example below refactors the code involved with creating a series mapping to a new method "AddSeriesMapping()". This new method also reverts to using the LegendLabel LegendDisplayMode instead of the DataPointLabel because we're going to show multiple series of data and it will be easier to display in the legend this way, i. e. one legend label per series.



```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim categories As New Categories()
  Dim kayaks As Category = _
categories.FirstOrDefault(Function(c) c.CategoryName.Equals("Kayak"))

  chart.ItemsSource = kayaks.Products
  chart.DefaultView.ChartLegend.Header = "Products"
  chart.SeriesMappings.Add(AddSeriesMapping( _
New Bar3DSeriesDefinition(), "In Stock", "InStock"))
End Sub

Private Function AddSeriesMapping( _
ByVal seriesDefinition As ISeriesDefinition, _
ByVal legendLabel As String, ByVal fieldName As String) As SeriesMapping
  ' add mapping for the series
  Dim mapping As New SeriesMapping() With { _
.SeriesDefinition = seriesDefinition, _
.LegendLabel = legendLabel}

  ' add item mapping for each property in the item
  mapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.LegendLabel})

  mapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.YValue, _
.FieldName = fieldName})

  Return mapping
End Function
```

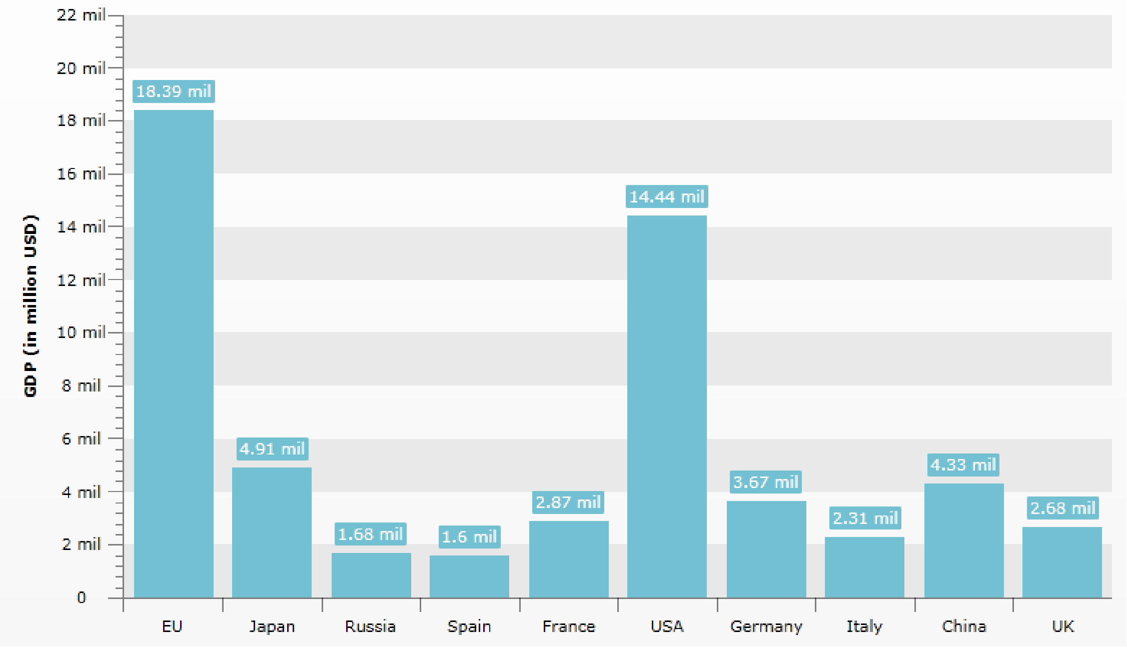
```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
    Categories categories = new Categories();
    Category kayaks =
        categories.FirstOrDefault(c => c.CategoryName.Equals("Kayak"));

    chart.ItemsSource = kayaks.Products;
    chart.DefaultView.ChartLegend.Header = "Products";
    chart.SeriesMappings.Add(
        AddSeriesMapping(new Bar3DSeriesDefinition(), "In Stock", "InStock"));
}

private SeriesMapping AddSeriesMapping(ISeriesDefinition seriesDefinition,
  string legendLabel, string fieldName)
{
    // add mapping for the series
    SeriesMapping mapping = new SeriesMapping()
    {
        SeriesDefinition = seriesDefinition,
        LegendLabel = legendLabel
    };

    // add item mapping for each property in the item
    mapping.ItemMappings.Add(new ItemMapping()
    {
        DataPointMember = DataPointMember.LegendLabel
    });

    mapping.ItemMappings.Add(new ItemMapping()
    {
        DataPointMember = DataPointMember.YValue,
        FieldName = fieldName
    });

    return mapping;
}
```

22.5.2 Binding Axis Labels

From the Forums...

Question: I can get the data to appear as a bar series with the correct values but how do I put labels on the x axis?

Answer: One of the DataPointMember enumeration values is **XCategory**. Bind the property that you want to use for labels to the ItemMapping FieldName and set the DataPointMember to DataPointMember.XCategory.



VB

```
seriesMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.XCategory, .FieldName = "ProductName"})
```

C#

```
seriesMapping.ItemMappings.Add(new ItemMapping()
{
   DataPointMember = DataPointMember.XCategory,
   FieldName = "ProductName"
});
```

22.5.3 Tooltips

There are several methods that populate tool tips, each with tradeoffs to consider. First, set the SeriesDefinition **ShowItemToolTips** property "True" before any of the methods will work.

When you create ItemMappings, you can bind the DataPointMember to the **DataPointMember.ToolTip** enumeration member. Assign **FieldName** to the name of the property that will supply the value.

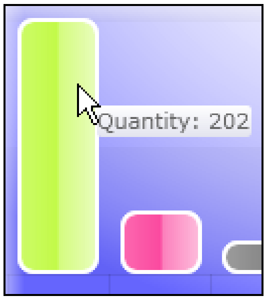
VB

```
mapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.Tooltip, _
.FieldName = "OnOrder"})
```

C#

```
mapping.ItemMappings.Add(new ItemMapping()
{
   DataPointMember = DataPointMember.Tooltip,
   FieldName = "OnOrder"
});
```

Instead of using an ItemMapping, set the ItemToolTipFormat property to a Label Format Expression (see the upcoming Label Format Expressions section for more detail on syntax). In the example below, ItemToolTipFormat displays the "Y" value in a formatted string.



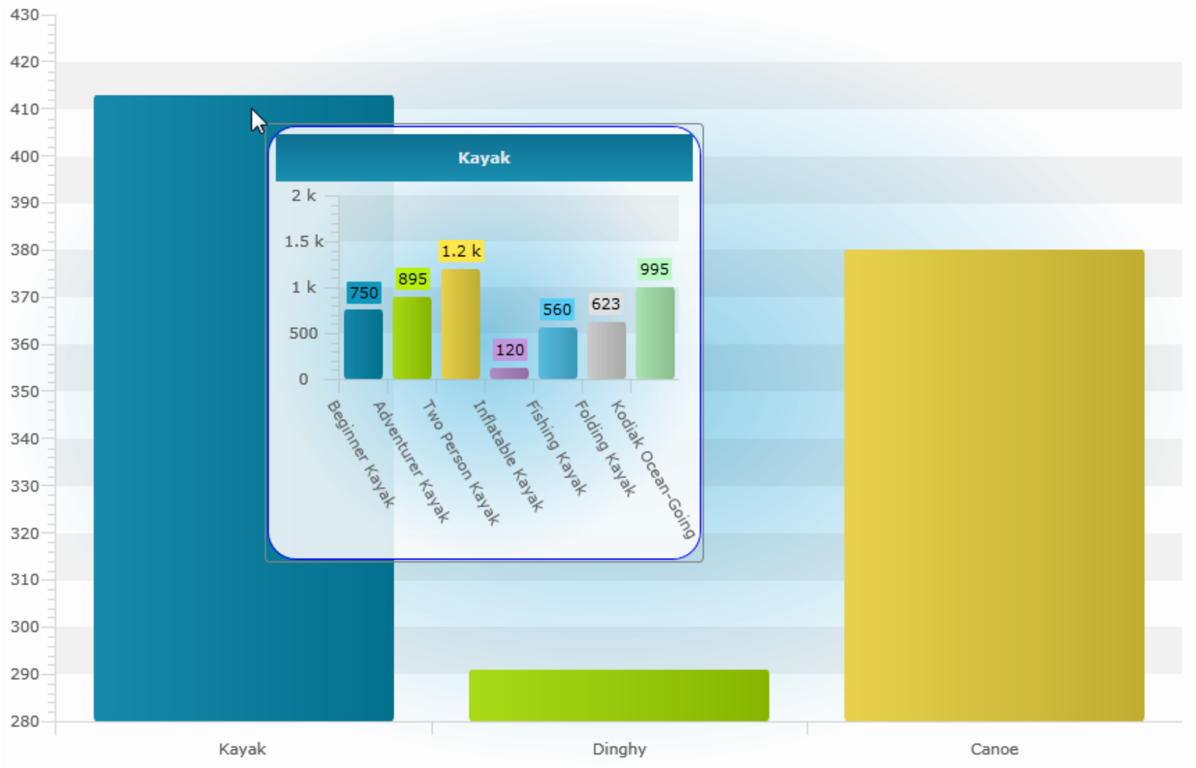
VB

```
Dim definition As ISeriesDefinition = New BarSeriesDefinition() With { _
.LegendDisplayMode = LegendDisplayMode.DataPointLabel, _
.ShowItemLabels = False, _
.ShowItemToolTips = True, _
.ItemToolTipFormat = "Quantity: #Y{0}"}
```

```
ISeriesDefinition definition = new BarSeriesDefinition()
{
   LegendDisplayMode = LegendDisplayMode.DataPointLabel,
   ShowItemLabels = false,
   ShowItemToolTips = true,
   ItemToolTipFormat = "Quantity: #Y{0}"
};
```

To get fine-grained control over tool tips, handle the **ItemToolTipOpening** event. The event passes the ItemToolTip2D that encapsulates the tool tip element and a ItemToolTipEventArgs that contains additional information about where the tool tip opened from, including the DataPoint, DataSeries, ItemIndex and MouseData. You have unlimited latitude to make tooltip Content be just about anything, from simple text to complex layouts including grids or even other RadCharts. For example, you can display a drill down chart showing detail for a data point under the mouse.

The example below displays a Border with a detail RadChart. You can see in the code that we can access the bound data using the DataPoint.DataItem property. In this case, the DataItem is a "Category" object that contains a "Products" collection that can then be bound to the detail chart. We can even apply the same theme to the detail chart for a consistent look and feel. This example only shows a Border and chart, but by adding any container, such as a Grid or StackPanel to the tooltip.Content, you can build tool tips of arbitrary complexity.



```
Private Sub ChartArea_ItemToolTipOpening( _
ByVal tooltip As ItemToolTip2D, ByVal e As ItemToolTipEventArgs)
  Dim category As Category = TryCast(e.DataPoint.DataItem, Category)

  Dim detailChart As New RadChart()
  detailChart.SetValue(StyleManager.ThemeProperty,  _
ThemeManager.FromName("Summer"))
  detailChart.Background = New SolidColorBrush(Colors.Transparent)
  detailChart.DefaultView.ChartTitle.Content = category.CategoryName
  detailChart.DefaultView.ChartLegend.Visibility = Visibility.Collapsed
  detailChart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 60
  detailChart.Width = 300
  detailChart.Height = 300
  detailChart.ItemsSource = category.Products

  Dim definition As ISeriesDefinition = New BarSeriesDefinition() With { _
.LegendDisplayMode = LegendDisplayMode.DataPointLabel,  _
.ShowItemLabels = True,  _
.ShowItemToolTips = True}

  Dim mapping As New SeriesMapping() With {.SeriesDefinition = definition}

  mapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.XCategory,  _
.FieldName = "ProductName"})

  mapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.YValue,  _
.FieldName = "Cost"})

  detailChart.SeriesMappings.Add(mapping)

  Dim border As New Border() With { _
.Background = chart.Background,  _
.BorderBrush = New SolidColorBrush(Colors.Blue),  _
.BorderThickness = New Thickness(1),  _
.CornerRadius = New CornerRadius(20),  _
.Padding = New Thickness(5)}
  border.Child = detailChart
  tooltip.Background = New SolidColorBrush(Colors.Transparent)
  tooltip.Content = border
End Sub
```

```
void ChartArea_ItemToolTipOpening(ItemToolTip2D tooltip, ItemToolTipEventArgs e)
{
   Category category = e.DataPoint.DataItem as Category;

   RadChart detailChart = new RadChart();
   detailChart.SetValue(StyleManager.ThemeProperty, ThemeManager.FromName("Summer"));
   detailChart.Background = new SolidColorBrush(Colors.Transparent);
   detailChart.DefaultView.ChartTitle.Content = category.CategoryName;
   detailChart.DefaultView.ChartLegend.Visibility = Visibility.Collapsed;
   detailChart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 60;
   detailChart.Width = 300;
   detailChart.Height = 300;
   detailChart.ItemsSource = category.Products;

   ISeriesDefinition definition = new BarSeriesDefinition()
   {
      LegendDisplayMode = LegendDisplayMode.DataPointLabel,
      ShowItemLabels = true,
      ShowItemToolTips = true
   };

   SeriesMapping mapping = new SeriesMapping()
   {
      SeriesDefinition = definition
   };

   mapping.ItemMappings.Add(new ItemMapping()
   {
      DataPointMember = DataPointMember.XCategory,
      FieldName = "ProductName"
   });

   mapping.ItemMappings.Add(new ItemMapping()
   {
      DataPointMember = DataPointMember.YValue,
      FieldName = "Cost"
   });

   detailChart.SeriesMappings.Add(mapping);

   Border border = new Border()
   {
      Background = chart.Background,
      BorderBrush = new SolidColorBrush(Colors.Blue),
      BorderThickness = new Thickness(1),
      CornerRadius = new CornerRadius(20),
      Padding = new Thickness(5)
   };
   border.Child = detailChart;
   tooltip.Background = new SolidColorBrush(Colors.Transparent);
   tooltip.Content = border;
}
```

22.5.4 Format Expressions

You can format an entire series by defining the SeriesDefinition ItemLabelFormat...

VB

```
inStockMapping.SeriesDefinition.ItemLabelFormat = "#Y"
```

C#

```
inStockMapping.SeriesDefinition.ItemLabelFormat = "#Y";
```

or you can set the label format for each DataPoint:

VB

```
Dim point As New DataPoint()
point.LabelFormat = "#Y"
```

C#

```
DataPoint point = new DataPoint();
point.LabelFormat = "#Y";
```

Formats are made up of a pound sign ("#") + a predefined token + an optional format string. Here are the defined tokens and how they relate to DataPoint or TickPoint properties.

Token	Use For	Value
#Y	Series Items Label, Tooltip	DataPoint.YValue
#X	Series Items Label, Tooltip	DataPoint.XValue
#XCAT	Series Items Label, Tooltip	DataPoint.XCategory
#HIGH	Series Items Label, Tooltip	DataPoint.High
#LOW	Series Items Label, Tooltip	DataPoint.Low
#OPEN	Series Items Label, Tooltip	DataPoint.Open

#CLOSE	Series Items Label, Tooltip	DataPoint.Close
#BUBBLESIZE	Series Items Label, Tooltip	DataPoint.BubbleSize
#LABEL	Series Items Label, Tooltip	DataPoint.Label
#LEGENDLABEL	Series Items Label, Tooltip	DataPoint.LegendLabel
#TOOLTIP	Series Items Label, Tooltip	DataPoint.Tooltip
#SERIESLABEL	Series Items Label, Tooltip	DataSeries.LegendLabel
#%	Series Items Label, Tooltip	DataPoint.YValue / (The sum of all YValues in the current data series)
#STSUM	Series Items Label, Tooltip	Represents the sum of all stacked items for a given index
#STPERCENT	Series Items Label, Tooltip	The percent representation of the value of a given item with respect to all stacked items for the respective index.
#DATAITEM. <PropertyName>	Series Items Label, Tooltip	Used to access the DataPoint.DataItem and read the value from a property of the underlying business object.
#VAL	X-Axis, Y-Axis	TickPoint.Value. This will work only when formatting axis labels.

.

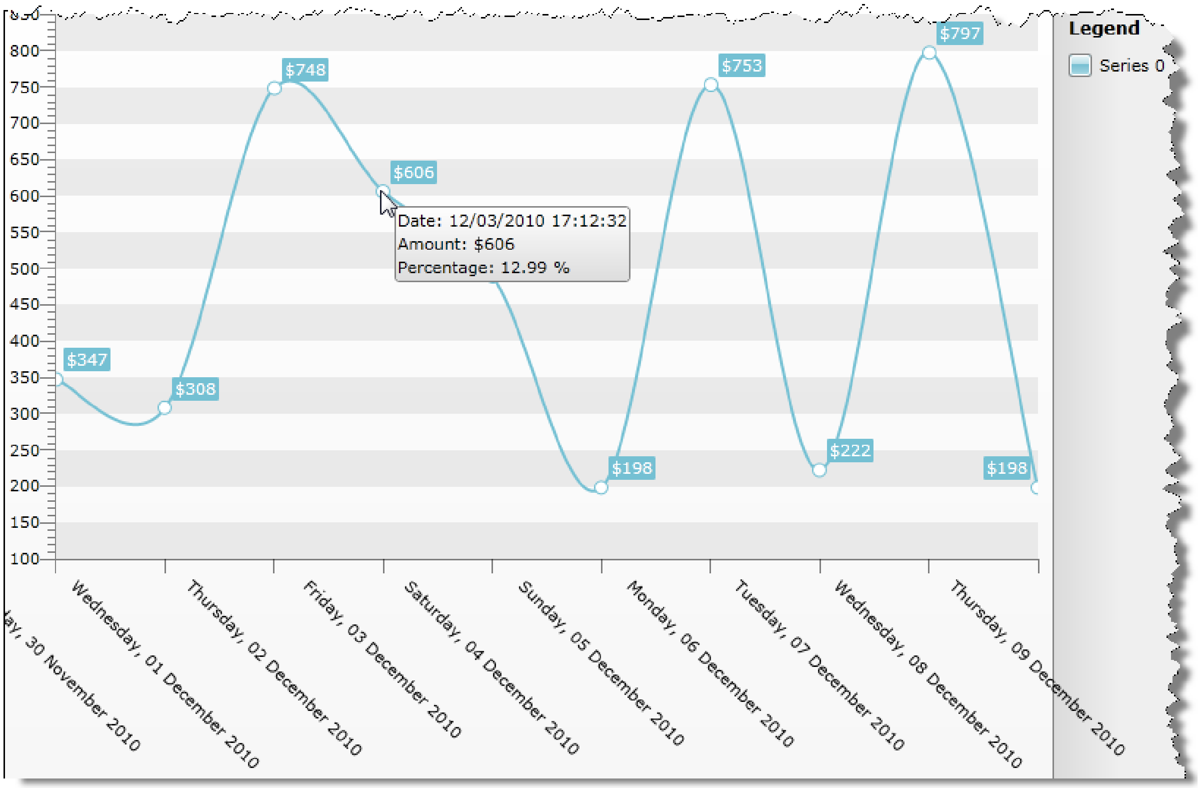
Here's a slightly larger example that demonstrates several formatting expressions at once. Both the label and the tool tip are formatted.

The example uses a "Sales" class that knows how to create a set of DateTime values and random "Amounts".

```
Public Class Sales
  Private privateAmount As Double
  Public Property Amount() As Double
    Get
      Return privateAmount
    End Get
    Set(ByVal value As Double)
      privateAmount = value
    End Set
  End Property
  Private privateSalesDate As DateTime
  Public Property SalesDate() As DateTime
    Get
      Return privateSalesDate
    End Get
    Set(ByVal value As DateTime)
      privateSalesDate = value
    End Set
  End Property

  Public Shared Function Fill(ByVal count As Integer) _
As IEnumerable(Of Sales)
    Dim random As New Random()
    Dim now As DateTime = DateTime.Now
    Dim max As Integer = count * 100

    For i As Integer = 0 To count - 1
      Dim sales As New Sales() With { _
.Amount = random.Next(max), .SalesDate = now.AddDays(i)}
            Return sales
    Next i
  End Function
End Class
```

```
public class Sales
{
   public double Amount { get; set; }
   public DateTime SalesDate { get; set; }

   public static IEnumerable<Sales> Fill(int count)
   {
      Random random = new Random();
      DateTime now = DateTime.Now;
      int max = count * 100;

      for (int i = 0; i < count; i++)
      {
         Sales sales = new Sales()
         {
            Amount = random.Next(max),
            SalesDate = now.AddDays(i)
         };

         yield return sales;
      }
   }
}
```

While configuring the series mappings, the **ItemToolTipFormat** and the **ItemLabelFormat** are assigned. The ItemToolTipFormat lets you try out several formats all at once. Here we're showing the "XCategory", the "Y" value as a currency with zero decimal places and a percentage of the all the values with two decimal places. The Item label format also displays the YValue as a currency with no decimal places.



VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  chart.DefaultSeriesDefinition = New SplineSeriesDefinition()
  chart.DefaultSeriesDefinition.ShowItemToolTips = True
  chart.DefaultSeriesDefinition.ItemToolTipFormat = _
"Date: #XCAT" & Constants.vbCrLf & "Amount: #Y{C0}" & _
Constants.vbCrLf & "Percentage: #%{P2}"
  chart.DefaultSeriesDefinition.ItemLabelFormat = "#Y{C0}"
  chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45
  Dim mapping As New SeriesMapping()
  mapping.ItemMappings.Add(New ItemMapping("SalesDate", _
DataPointMember.XCategory))
  mapping.ItemMappings.Add(New ItemMapping("Amount", _
DataPointMember.YValue))
  chart.SeriesMappings.Add(mapping)
  chart.ItemsSource = Sales.Fill(10)
End Sub
```

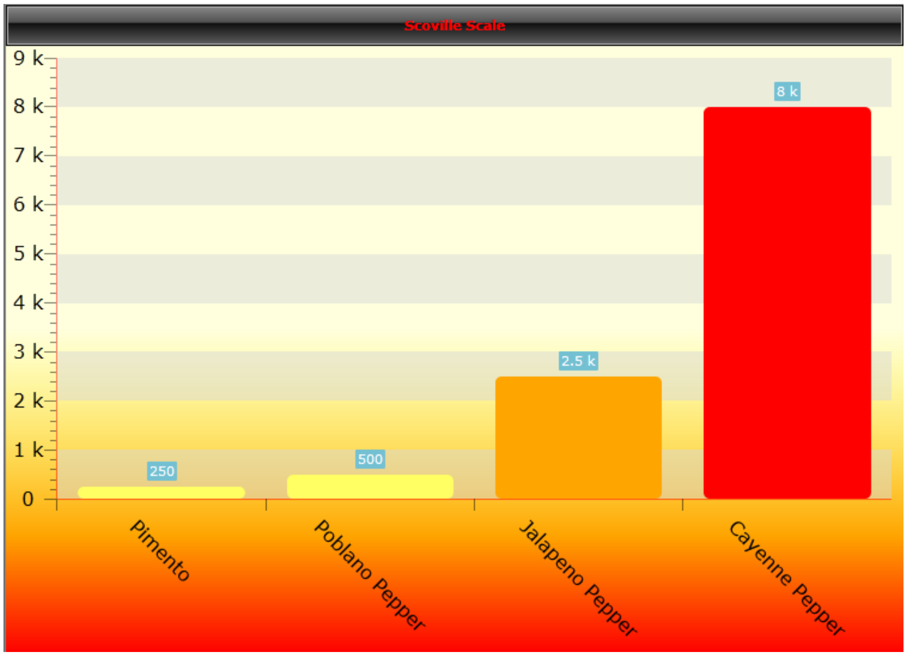
C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   chart.DefaultSeriesDefinition = new SplineSeriesDefinition();
   chart.DefaultSeriesDefinition.ShowItemToolTips = true;
   chart.DefaultSeriesDefinition.ItemToolTipFormat =
      "Date: #XCAT\r\nAmount: #Y{C0}\r\nPercentage: #%{P2}";
   chart.DefaultSeriesDefinition.ItemLabelFormat =
      "#Y{C0}";
   chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45;
   SeriesMapping mapping = new SeriesMapping();
   mapping.ItemMappings.Add(
      new ItemMapping("SalesDate", DataPointMember.XCategory));
   mapping.ItemMappings.Add(
      new ItemMapping("Amount", DataPointMember.YValue));
   chart.SeriesMappings.Add(mapping);
   chart.ItemsSource = Sales.Fill(10);
}
```

22.6 Customization

22.6.1 Coloring Chart Elements

A very common issue discussed in the forums is the need to color series items based on data values. The recommended technique employs the MVVM pattern. The "View Model" layer translates some property of the "Model" into a Brush. The chart as a whole is bound to the view model and a custom template element for a given series item type, e.g. "Bar", is bound to the brush. For example, we can have a series of "Pepper" objects that have a "Scoville" scale rating that measures relative spiciness. The "PepperViewModel" object translates the scale rating into a brush that represents the Scoville scale rating. The brush is used in the binding of the bar element in the template.



Let's take a look at the relevant pieces that make this work.

Model

The Model defines the data you want to see in the chart. This may be your own object or data returned by a service. In this example we have the Pepper class that defines a name, a numeric "Scoville" scale rating and a Spiciness enumeration value that tells us the range of the scale rating. The specifics of the class implementation are less important than the fact that this model has a property that will be translated by the view model into a visually relevant property.

```
Public Enum Spiciness
  Mild
  Medium
  Hot
  Extreme
End Enum

Public Class Pepper
  Private _name As String
  Private privateName As String
  Public Property Name() As String
    Get
      Return privateName
    End Get
    Set(ByVal value As String)
      privateName = value
    End Set
  End Property

  Private privateScovilleRating As Integer
  Public Property ScovilleRating() As Integer
    Get
      Return privateScovilleRating
    End Get
    Set(ByVal value As Integer)
      privateScovilleRating = value
    End Set
  End Property

  Public ReadOnly Property Spiciness() As Spiciness
    Get
      If (Me.ScovilleRating > 0) AndAlso _
(Me.ScovilleRating <= 500) Then
        Return Spiciness.Mild
      ElseIf (Me.ScovilleRating > 500) AndAlso _
(Me.ScovilleRating <= 2500) Then
        Return Spiciness.Medium
      ElseIf (Me.ScovilleRating > 2500) AndAlso _
(Me.ScovilleRating < 100000) Then
        Return Spiciness.Hot
      Else
        Return Spiciness.Extreme
      End If
    End Get
  End Property
End Class
```

```
public enum Spiciness { Mild, Medium, Hot, Extreme };

public class Pepper
{
   private string _name;
   public string Name
   {
      get; set;
   }

   public int ScovilleRating
   {
      get; set;
   }

   public Spiciness Spiciness
   {
      get
      {
         if ((this.ScovilleRating > 0) &&
            (this.ScovilleRating <= 500))
            return Spiciness.Mild;
         else if ((this.ScovilleRating > 500) &&
               (this.ScovilleRating <= 2500))
            return Spiciness.Medium;
         else if ((this.ScovilleRating > 2500) &&
               (this.ScovilleRating < 100000))
            return Spiciness.Hot;
         else return Spiciness.Extreme;
      }
   }
}
```

View Model

The view model wraps the model in a nice tidy package suitable for direct use by the view. The example below defines a PepperViewModel class. Notice that the constructor takes an instance of the model.

The purpose of the view model here is to translate the Spiciness property value on the model to a Brush that can be used in the View. Look at the implementation of the ScovilleColor Brush property. The Spiciness property is used to return an appropriate Brush color.

```
Public Class PepperViewModel
  Public Sub New(ByVal pepper As Pepper)
    Me.Pepper = pepper
  End Sub

  Private privatePepper As Pepper
  Public Property Pepper() As Pepper
    Get
      Return privatePepper
    End Get
    Set(ByVal value As Pepper)
      privatePepper = value
    End Set
  End Property

  Public ReadOnly Property ScovilleColor() As Brush
    Get
      Dim lightYellow As New SolidColorBrush(Color.FromArgb(255, 255, 255, 100))
      Dim darkOrange As New SolidColorBrush(Color.FromArgb(255, 250, 175, 0))
      Dim darkRed As New SolidColorBrush(Color.FromArgb(255, 140, 0, 0))

      Select Case Me.Pepper.Spiciness
        Case Spiciness.Mild
          Return lightYellow
        Case Spiciness.Medium
          Return New SolidColorBrush(Colors.Orange)
        Case Spiciness.Hot
          Return New SolidColorBrush(Colors.Red)
        Case Else
          Return darkRed
      End Select
    End Get
  End Property
End Class
```

C#

```
public class PepperViewModel
{
   public PepperViewModel(Pepper pepper)
   {
      this.Pepper = pepper;
   }

   public Pepper Pepper
   {
      get; set;
   }

   public Brush ScovilleColor
   {
      get
      {
         SolidColorBrush lightYellow =
            new SolidColorBrush(Color.FromArgb(255, 255, 255, 100));
         SolidColorBrush darkOrange =
            new SolidColorBrush(Color.FromArgb(255, 250, 175, 0));
         SolidColorBrush darkRed =
            new SolidColorBrush(Color.FromArgb(255, 140, 0, 0));

         switch (this.Pepper.Spiciness)
         {
            case Spiciness.Mild: return lightYellow;
            case Spiciness.Medium: return new SolidColorBrush(Colors.Orange);
            case Spiciness.Hot: return new SolidColorBrush(Colors.Red);
            default: return darkRed;
         }
      }
   }
}
```

Template

We use a custom template to allow binding view model properties to elements arranged in a container (e.g. Canvas, Grid, etc). In this case we are templating the Bar series item. The Bar contains a Rectangle with a Fill property that can be bound to DataItem.ScovilleColor. DataItem in this case is a PepperViewModel. Remember that the style is called "ScovilleStyle". We will assign the style later in code.

```
<Style x:Name="ScovilleStyle" TargetType="chart:Bar">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="telerikCharting:Bar">
        <Canvas x:Name = "PART_MainContainer">
          <Rectangle x:Name="PART_DefiningGeometry"
              . . .
              Fill="{Binding DataItem.ScovilleColor}" />
          . . .
        </Canvas>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

For reference, here's the entire control template for the Bar series item. You can also get at this information using Expression Blend.

```
<Style x:Name="ScovilleStyle"
      TargetType="telerikCharting:Bar">
      <Setter Property="Template" >
         <Setter.Value>
            <ControlTemplate TargetType="telerikCharting:Bar">
                <Canvas x:Name="PART_MainContainer">
                  <Rectangle x:Name="PART_DefiningGeometry"

                    Height="{TemplateBinding ItemActualHeight}"
                    Width="{TemplateBinding ItemActualWidth}"
                    Style="{TemplateBinding ItemStyle}"
                    StrokeThickness="2"
                    RadiusX="5"
                    RadiusY="5"
                    Fill="{Binding DataItem.ScovilleColor}" />
                 <Canvas.RenderTransform>
                   <ScaleTransform x:Name="PART_AnimationTransform" ScaleY="0" />
                 </Canvas.RenderTransform>
                  <Canvas.Triggers>
                 <EventTrigger
                     RoutedEvent="Rectangle.Loaded">
                  <EventTrigger.Actions>
                        <BeginStoryboard>
                            <Storyboard
                             x:Name="PART_Storyboard"
                             BeginTime="00:00:00.5">
                             <DoubleAnimation
                                To="1"
                                Storyboard.TargetName="PART_AnimationTransform"
                                Storyboard.TargetProperty="ScaleY"
                                Duration="00:00:00.25"
                                BeginTime="00:00:00.2">
                             </DoubleAnimation>
                            </Storyboard>
                        </BeginStoryboard>
                   </EventTrigger.Actions>
                   </EventTrigger>
                   </Canvas.Triggers>
                 </Canvas>
             </ControlTemplate>
            </Setter.Value>
        </Setter>
</Style>
```

Code Behind

The first tasks is to create a collection of model "Pepper" objects. Then create a collection of matching view model "PepperViewModel" objects. For your own purposes, the "model" objects may be coming from another source, such as a web, RIA or REST service.

VB

```
' create the model
Dim peppers As List(Of Pepper) = New List(Of Pepper) (New Pepper() { _
New Pepper() With {.Name = "Pimento", .ScovilleRating = 250},  _
New Pepper() With {.Name = "Poblano Pepper", .ScovilleRating = 500},  _
New Pepper() With {.Name = "Jalapeno Pepper", .ScovilleRating = 2500},  _
New Pepper() With {.Name = "Cayenne Pepper", .ScovilleRating = 10000}})

' create the view model and copy model into it
Dim viewModel As List(Of PepperViewModel) = New List(Of PepperViewModel)()
For Each pepper As Pepper In peppers
  viewModel.Add(New PepperViewModel(pepper))
Next pepper
```

C#

```
// create the model
List<Pepper> peppers = new List<Pepper>()
{
   new Pepper() { Name = "Pimento", ScovilleRating = 250 },
   new Pepper() { Name = "Poblano Pepper", ScovilleRating = 500 },
   new Pepper() { Name = "Jalapeno Pepper", ScovilleRating = 2500 },
   new Pepper() { Name = "Cayenne Pepper", ScovilleRating = 10000 }
};

// create the view model and copy model into it
List<PepperViewModel> viewModel = new List<PepperViewModel>();
foreach (Pepper pepper in peppers)
{
   viewModel.Add(new PepperViewModel(pepper));
}
```

During assignment of the series definition and item mapping, you have the opportunity to assign ItemStyle to "ScovilleStyle" and to tie view model properties to elements on the chart. In this example, the Pepper. ScovilleRating is bound to the YValue and the Pepper.Name is bound to the XCategory that displays along the X-Axis of the chart.

VB

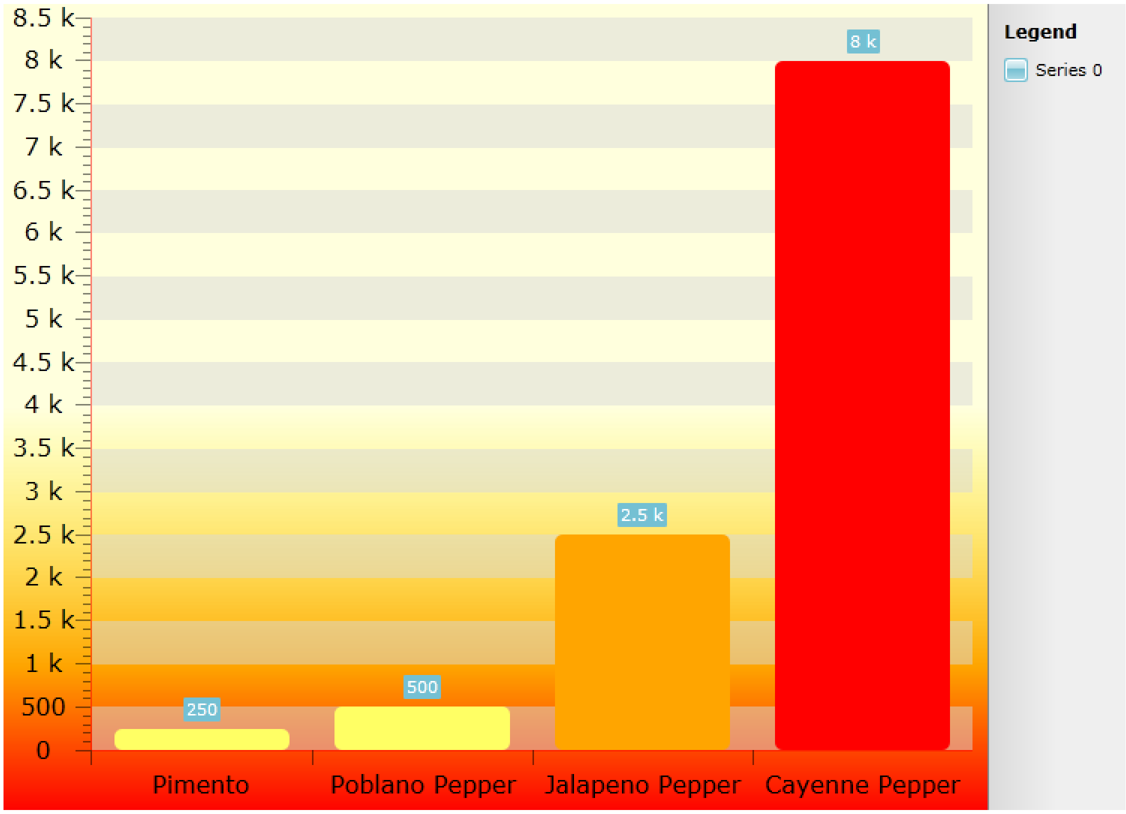
```
' setup the grid to display name and rating
chart.DefaultSeriesDefinition = New BarSeriesDefinition() With { _
.ItemStyle = Me.ScovilleStyle}
Dim mapping As New SeriesMapping()
mapping.ItemMappings.Add( _
New ItemMapping("Pepper.ScovilleRating", DataPointMember.YValue))
mapping.ItemMappings.Add( _
New ItemMapping("Pepper.Name", DataPointMember.XCategory))
chart.SeriesMappings.Add(mapping)
' bind view model to chart
chart.ItemsSource = viewModel
```

C#

```
// setup the grid to display name and rating
chart.DefaultSeriesDefinition = new BarSeriesDefinition()
{ ItemStyle = this.ScovilleStyle };
SeriesMapping mapping = new SeriesMapping();
mapping.ItemMappings.Add(
new ItemMapping("Pepper.ScovilleRating", DataPointMember.YValue));
mapping.ItemMappings.Add(
new ItemMapping("Pepper.Name", DataPointMember.XCategory));
chart.SeriesMappings.Add(mapping);
// bind view model to chart
chart.ItemsSource = viewModel;
```

22.6.2 Styling the Chart

You may have noticed from the last example that the axis lines were red axis text has a larger-than-default size. The chart surfaces many styles that make it possible to change the appearance of chart elements without completely re-templating the entire control. The appearance of the chart below is disorganized. The title is missing and legend text is not legible, the axis lines are barely visible and the X axis text is horizontal.



There are a few tweaks you can make without having to style the chart. The code snippet below sets properties for the ChartLegend, ChartTitle and ChartArea AxisX properties.

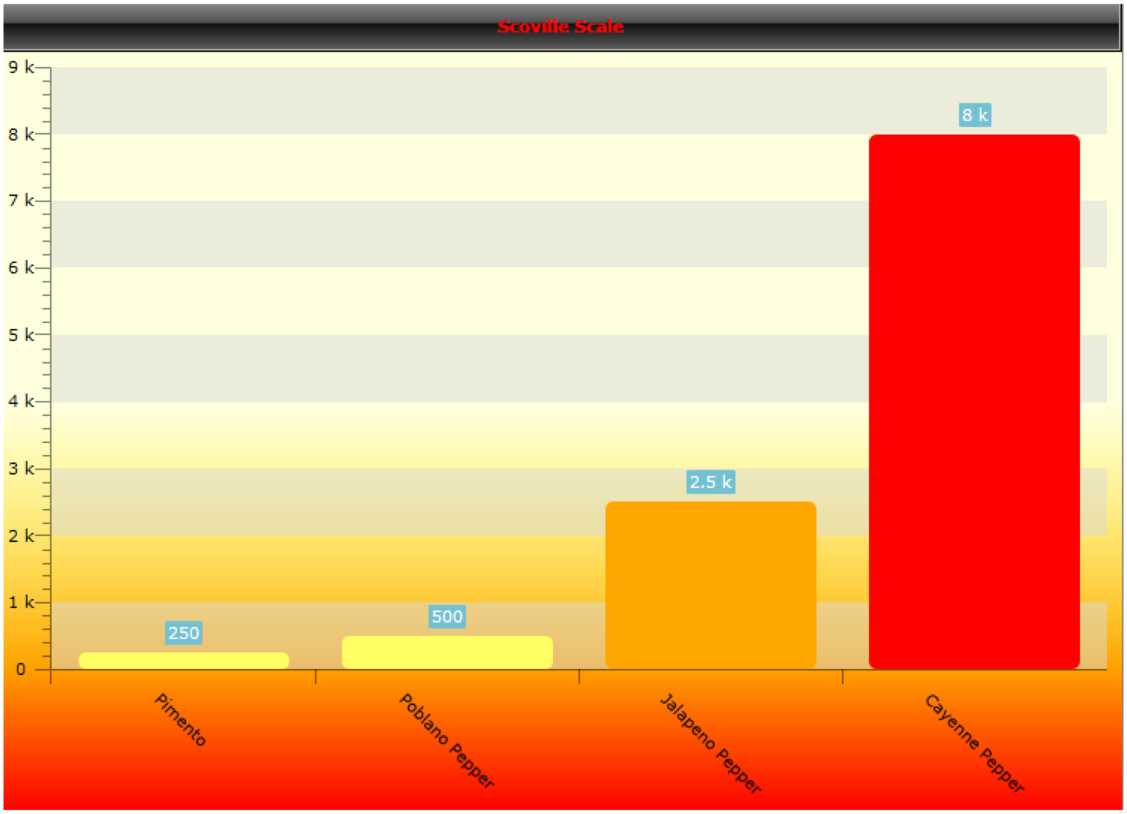
VB

```
chart.DefaultView.ChartLegend.Visibility = Visibility.Collapsed
chart.DefaultView.ChartTitle.Content = "Scoville Scale"
chart.DefaultView.ChartTitle.Foreground = New SolidColorBrush(Colors.Red)
chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45
```

C#

```
chart.DefaultView.ChartLegend.Visibility = Visibility.Collapsed;
chart.DefaultView.ChartTitle.Content = "Scoville Scale";
chart.DefaultView.ChartTitle.Foreground = new SolidColorBrush(Colors.Red);
chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45;
```

These changes go a long way to cleaning up the appearance. Removing the legend gets rid of the legend key artifact and lets the chart take up more room. The X Axis labels are rotated and this also relieves crowding on the page. But we still can't see the axis lines text or markings very well.



We can define styles that apply to elements of the chart and apply them in XAML or later, in code. Two styles are defined below that set the look for an axis line and label text.

```
<Style x:Key="CustomAxisStyle" TargetType="Line">
  <Setter Property="Stroke" Value="Red" />
  <Setter Property="StrokeThickness" Value="1" />
</Style>

<Style x:Key="CustomItemLabelStyle" TargetType="TextBlock">
  <Setter Property="Foreground" Value="Black" />
  <Setter Property="FontSize" Value="16" />
</Style>
```

These can be applied against both X and Y axis elements as shown in the code snippet below.

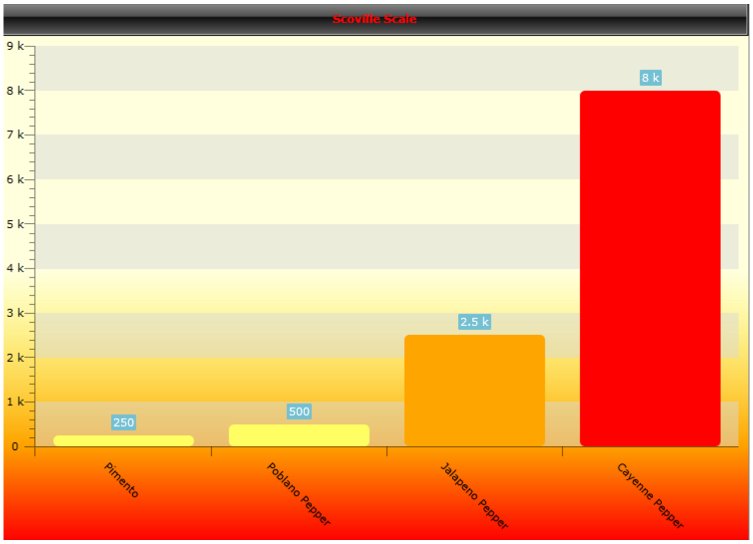
VB

```
chart.DefaultView.ChartArea.AxisY.AxisStyles.AxisLineStyle = _
TryCast(Me.Resources("CustomAxisStyle"), Style)
chart.DefaultView.ChartArea.AxisY.AxisStyles.ItemLabelStyle = _
TryCast(Me.Resources("CustomItemLabelStyle"), Style)
chart.DefaultView.ChartArea.AxisX.AxisStyles.AxisLineStyle = _
TryCast(Me.Resources("CustomAxisStyle"), Style)
chart.DefaultView.ChartArea.AxisX.AxisStyles.ItemLabelStyle = _
TryCast(Me.Resources("CustomItemLabelStyle"), Style)
```

C#

```
chart.DefaultView.ChartArea.AxisY.AxisStyles.AxisLineStyle =
   this.Resources["CustomAxisStyle"] as Style;
chart.DefaultView.ChartArea.AxisY.AxisStyles.ItemLabelStyle =
   this.Resources["CustomItemLabelStyle"] as Style;
chart.DefaultView.ChartArea.AxisX.AxisStyles.AxisLineStyle =
   this.Resources["CustomAxisStyle"] as Style;
chart.DefaultView.ChartArea.AxisX.AxisStyles.ItemLabelStyle =
   this.Resources["CustomItemLabelStyle"] as Style;
```

Now all visual aspects of the chart are clear and legible.



22.7 Wrap Up

In this chapter you built a chart declaratively using default settings wherever possible. In the process you learned the differences between default vs. custom layouts. You defined a series with individual data points and specified the legend and chart title. You also learned about how the series definition displays the data in a particular arrangement, i.e. bar, pie, 3D line, etc. You enabled the interactivity feature of the chart.

During the exploration of chart control details you toured chart series types that include the standard bar/line/pie, stacked versions that compare contributions of values across categories, stacked bar 100% that show the stacked values as percentages, special purpose charts like "candlestick" for showing stock price and currency changes, and many 3D chart series types as well. You learned how to create chart series and data points programmatically. The section on "Integration with ASP.NET" walked you through how to include RadChart with existing ASP.NET applications.

As part of the binding chapter you first learned about series and item mappings that route data to various elements in the chart, including the data point and axis labels. You displayed tool tips as simple text, used special formatting tokens to show data as currency or as percentages of all categories and you learned how to display a tool tip that shows another chart as a drill-down.

You applied styles to customize elements of the chart. You used the MVVM pattern to display chart elements in colors corresponding to data in the model.

Part XII

Docking

23 Docking

23.1 Objectives

In this chapter you will learn how to create flexible layouts using the RadDocking control. You will start by creating a layout entirely in XAML. During this initial project you will learn how to control user interaction with panes including floating, closing and pinning behaviors.

In the Control Details section of the chapter you will learn how to create split containers, groups and panes all in code, how to size and position floating panes, how to save and load panes, how to control pinning behavior, how to hide and show panes and how to use "Preview" events.

During the Binding section of this chapter you will learn how to build a Silverlight "mashup" client application with docking that aggregates various REST services. The service output will be bound to both the pane content and the pane title area.

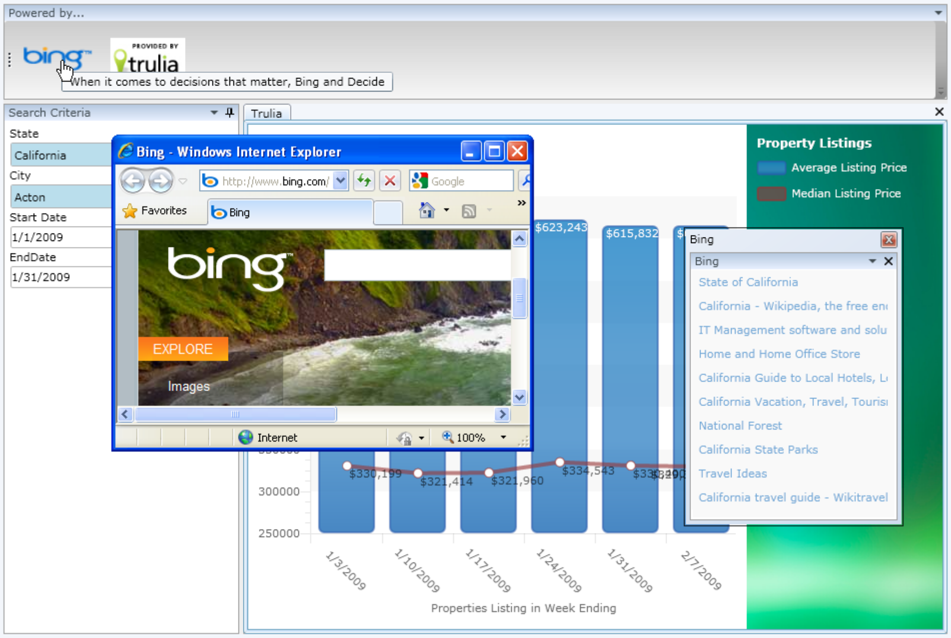
During the Customization section of the chapter you will modify the pane title area to include a button and image.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Dock\Dock.sln.

23.2 Overview

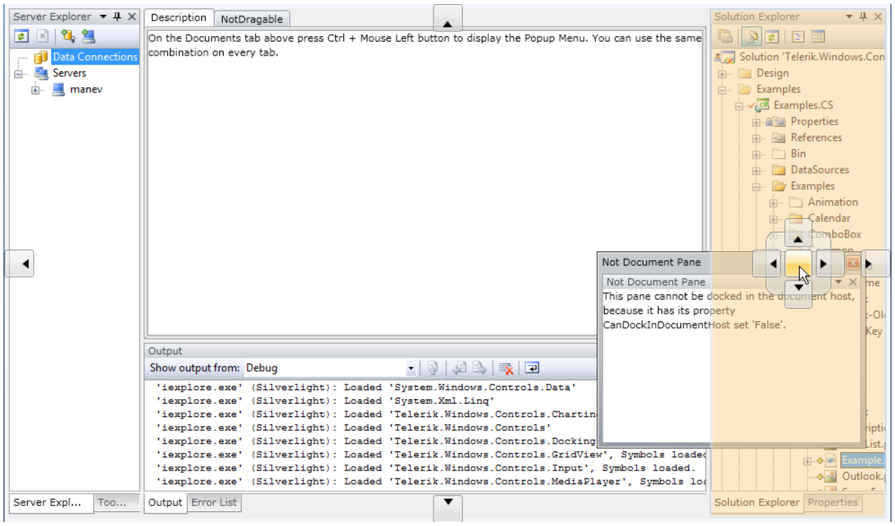
Handling multiple dockable windows is a no-hassle operation with the RadDocking control. Incorporate splitters, tabbed documents, float and auto-hide panes to your application. RadDocking maintains a clear separation between window management and content.



State
California
City
Acton
Start Date
1/1/2009
EndDate
1/31/2009

RadDocking includes these features:

- Dockable Windows



On the Documents tab above press Ctrl + Mouse Left button to display the Popup Menu. You can use the same combination on every tab.

This pane cannot be docked in the document host, because it has its property CanDockInDocumentHost set 'False'.

Show output from: Debug

```
'iexplore.exe' (Silverlight): Loaded 'System.Windows.Controls.Data'
'iexplore.exe' (Silverlight): Loaded 'System.Xml.Linq'
'iexplore.exe' (Silverlight): Loaded 'Telerik.Windows.Controls.Chartin
'iexplore.exe' (Silverlight): Loaded 'Telerik.Windows.Controls'
'iexplore.exe' (Silverlight): Loaded 'Telerik.Windows.Controls.Docking
'iexplore.exe' (Silverlight): Loaded 'Telerik.Windows.Controls.GridView', Symbols loaded
'iexplore.exe' (Silverlight): Loaded 'Telerik.Windows.Controls.Input', Symbols loaded.
'iexplore.exe' (Silverlight): Loaded 'Telerik.Windows.Controls.MediaPlayer', Symbols loa
```

- Floating Windows
- ToolWindow Control and Behavior
- Pin, Unpin and Hide
- Tabbed Documents

23.3 Getting Started

In this walk through you will build a page with multiple panes that can be floated, docked and pinned. The example also includes a document area for tabbed documents.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

   a) **Telerik.Windows.Controls**

   b) **Telerik.Windows.Controls.Docking**

   c) **Telerik.Windows.Controls.Navigation**

XAML Editing

1) Open MainPage.xaml for editing.
2) Drag a **RadDocking** control from the Toolbox to a point between the "LayoutRoot" <Grid> and </Grid> tags.

```
<Grid x:Name="LayoutRoot">
   <telerik:RadDocking>
   </telerik:RadDocking>
</Grid>
```

3) Drag a **RadSplitContainer** from the Toolbox to a point inside the RadDocking element. Name the element "TopContainer". Set the **InitialPosition** property to "DockedTop".

```
<telerik:RadDocking>
   <telerik:RadSplitContainer x:Name="TopContainer"
         InitialPosition="DockedTop">
   </telerik:RadSplitContainer>
</telerik:RadDocking>
```

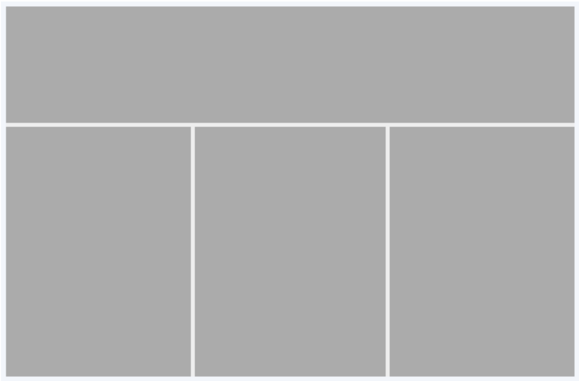
4) Create two more new **RadSplitContainer** elements just under the first RadSplitContainer. Name the elements "LeftContainer" and "RightContainer" respectively. Set the **InitialPosition** properties to "DockedLeft" and "DockedRight". The XAML should now look like the example below.

```
<telerik:RadDocking>

  <telerik:RadSplitContainer x:Name="TopContainer"
      InitialPosition="DockedTop">
  </telerik:RadSplitContainer>

  <telerik:RadSplitContainer x:Name="LeftContainer"
      InitialPosition="DockedLeft">
  </telerik:RadSplitContainer>

  <telerik:RadSplitContainer x:Name="RightContainer"
      InitialPosition="DockedRight">
  </telerik:RadSplitContainer>

</telerik:RadDocking>
```

5) Below the first three RadSplitContainer elements and within the RadDocking element, add a **RadDocking.DocumentHost** element. Inside the DocumentHost element place one more RadSplitContainer. Name the container "CenterContainer".

```
<telerik:RadDocking.DocumentHost>
  <telerik:RadSplitContainer x:Name="CenterContainer">
  </telerik:RadSplitContainer>
</telerik:RadDocking.DocumentHost>
```

If you were to run the application now, the page would look something like this screenshot:



6) Drag a new RadPaneGroup from the Toolbox to a point inside each of the RadSplitContainer elements. The XAML should now look something like the example below.

```
<telerik:RadDocking>
   <telerik:RadSplitContainer x:Name="TopContainer"
         InitialPosition="DockedTop">
      <telerik:RadPaneGroup></telerik:RadPaneGroup>
   </telerik:RadSplitContainer>

   <telerik:RadSplitContainer x:Name="LeftContainer"
         InitialPosition="DockedLeft">
      <telerik:RadPaneGroup></telerik:RadPaneGroup>
   </telerik:RadSplitContainer>

   <telerik:RadSplitContainer x:Name="RightContainer"
         InitialPosition="DockedRight">
      <telerik:RadPaneGroup></telerik:RadPaneGroup>
   </telerik:RadSplitContainer>

   <telerik:RadDocking.DocumentHost>
      <telerik:RadSplitContainer x:Name="CenterContainer">
         <telerik:RadPaneGroup></telerik:RadPaneGroup>
      </telerik:RadSplitContainer>
   </telerik:RadDocking.DocumentHost>

</telerik:RadDocking>
```

7) Navigate back to the "TopContainer" element. Add a **RadPane** from the Toolbox to a point inside the RadPaneGroup element. Set RadPane properties as follows:

a) **Header**="Powered by..."

b) **CanUserClose**="False"

c) **CanUserPin**="False"

d) **CanFloat**="False"

e) **CanDockInDocumentHost**="False"

The markup should now look like the example below:

```
<telerik:RadSplitContainer x:Name="TopContainer"
      InitialPosition="DockedTop">
   <telerik:RadPaneGroup>
      <telerik:RadPane Header="Powered by..."
            CanUserClose="False" CanUserPin="False"
            CanFloat="False"
            CanDockInDocumentHost="False">
      </telerik:RadPane>
   </telerik:RadPaneGroup>
</telerik:RadSplitContainer>
```

8) Navigate to the "LeftContainer" element. Add a **RadPane** from the Toolbox to a point inside the RadPaneGroup element. Set RadPane properties as follows:

a) **Header**="Search Criteria"

b) **CanUserClose**="False"

c) **CanUserPin**="True"

d) **CanFloat**="False"

e) **CanDockInDocumentHost**="False"

The markup should now look like the example below:

```
<telerik:RadSplitContainer x:Name="LeftContainer"
      InitialPosition="DockedLeft">
   <telerik:RadPaneGroup>
      <telerik:RadPane Header="Search Criteria"
            CanDockInDocumentHost="False"
            CanFloat="False" CanUserClose="False"
            CanUserPin="True">
      </telerik:RadPane>
   </telerik:RadPaneGroup>
</telerik:RadSplitContainer>
```

9) Navigate to the "RightContainer" element. Add two **RadPane** controls from the Toolbox to the RadPaneGroup element. Set RadPane **Header** properties to "Chart" and "Data" respectively.

The markup should now look like the example below:

```
<telerik:RadSplitContainer x:Name="RightContainer"
      InitialPosition="DockedRight">
   <telerik:RadPaneGroup>
      <telerik:RadPane Header="Chart">
      </telerik:RadPane>
      <telerik:RadPane Header="Data">
      </telerik:RadPane>
   </telerik:RadPaneGroup>
</telerik:RadSplitContainer>
```

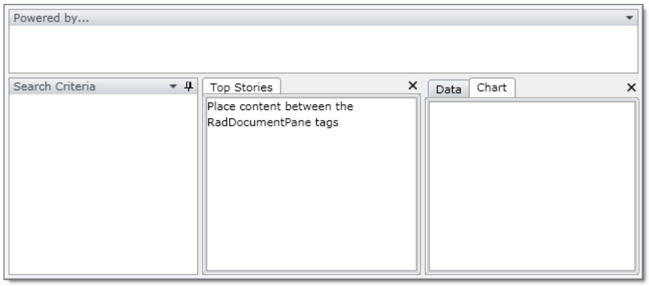
10)Navigate to the "CenterContainer" element. Add a **RadDocumentPane** from the Toolbox to a point inside the RadPaneGroup element. Set RadDocumentPane **Header** property to "Top Stories" and **CanFloat** to "False". Inside the RadDocumentPane add a TextBlock with some arbitrary text.

The markup should now be similar to the example below:

```
<telerik:RadDocking.DocumentHost>
  <telerik:RadSplitContainer
      x:Name="CenterContainer">
    <telerik:RadPaneGroup>
      <telerik:RadDocumentPane
          Header="Top Stories" CanFloat="False">
        <TextBlock
            Text="Place content between the RadDocumentPane tags"
            TextWrapping="Wrap" />
      </telerik:RadDocumentPane>
    </telerik:RadPaneGroup>
  </telerik:RadSplitContainer>
</telerik:RadDocking.DocumentHost>
```

Run The Application

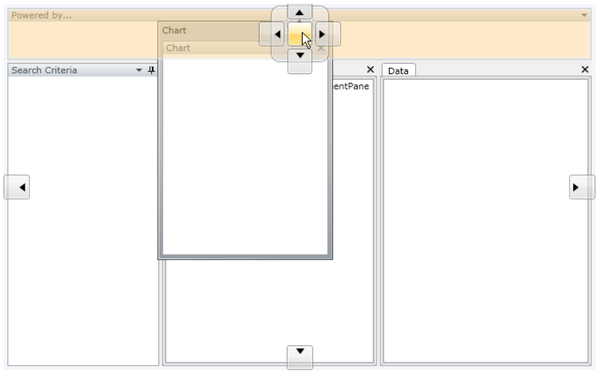
Press **F5** to run the application. The web page should look something like the screenshot below.



Place content between the RadDocumentPane tags

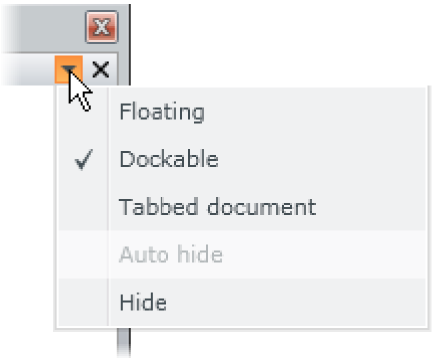
Test Application Features

1) Try dragging each of the panels or tabbed documents. The "Powered by...", "Search Criteria" and "Top Stories" should not allow dragging because their CanFloat properties are set to "False". The "Data" and "Chart" panes can be dragged and docked with any other pane. Also notice that once docked, they can be dragged back out to another location.

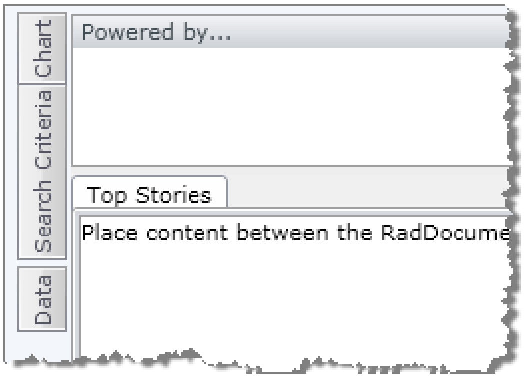


2) Test the behavior of each pane relative to its **CanUserClose**, **CanUserPin** and **CanDockInDocumentHost** property settings.

3) Open the drop-down menus in the panes. The menus control the current state of the pane and also reflect the property settings of each pane.



4) Find a pane with the "pin" icon and click it. The "pin" icon controls the "Auto hide" functionality and causes the pane to become a button on the side of the RadDocking area.



Ideas for Extending This Example

- Add content to each of the panes, either as text directly in the Content attribute, or within the RadPane or RadDocumentPane element tags.
- Change the **CanFloat**, **CanUserClose**, **CanUserPin** and **CanDockInDocumentHost** of a pane and see how the behavior changes.

23.4 Control Details

23.4.1 Creating Containers in Code

The "Getting Started" example shows a RadDocking with both panes and tabbed documents. Using RadPane inside RadDocking forms this basic hierarchy:

```
RadDocking
  RadSplitContainer
    RadPaneGroup
      RadPane
      RadPane
      RadPane
        <your content>
```

Tabbed documents have a similar structure, but are wrapped by a DocumentHost element and have a RadDocumentPane instead of a RadPane.

```
RadDocking
  RadDocking.DocumentHost
    RadSplitContainer
      RadPaneGroup
        RadDocumentPane
          <your content>
```

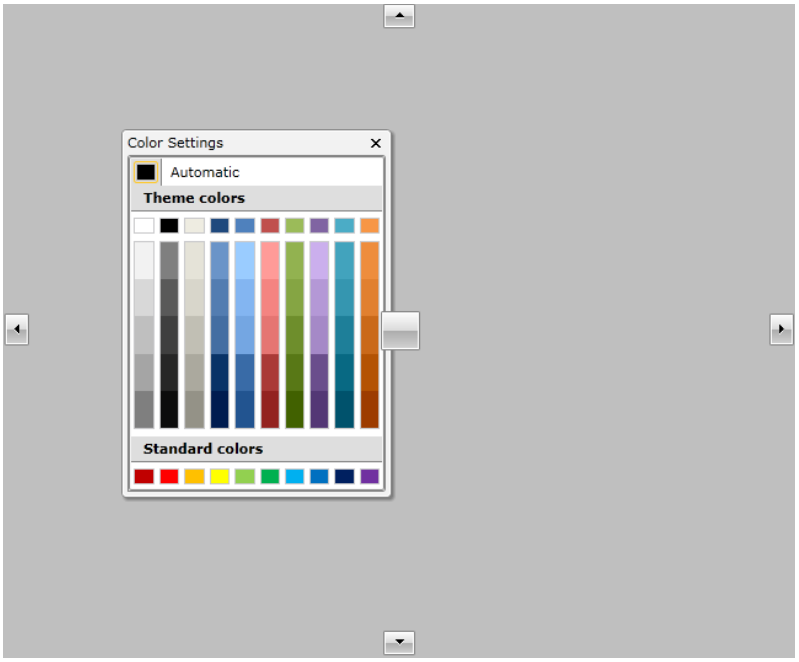
Adding Panes Programmatically

RadDocking, RadSplitContainer and RadPaneGroup all have Items collections, so adding a pane to a RadDocking control is as simple as creating instances of each and adding to the Items collection in order of parentage, i.e. docking adds container, container adds group, group adds pane. The only other property set in the following example is the InitialPosition that sets the state of the pane as FloatingDockable.

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim container = New RadSplitContainer() With { _
.InitialPosition = DockState.FloatingDockable}

  Dim paneGroup = New RadPaneGroup()
  Dim pane = New RadPane() With { _
.Header = "Color Settings", _
.Content = New RadColorSelector()}

  paneGroup.Items.Add(pane)
  container.Items.Add(paneGroup)
  docking.Items.Add(container)
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   var container = new RadSplitContainer()
   {
      InitialPosition = DockState.FloatingDockable
   };

   var paneGroup = new RadPaneGroup();
   var pane = new RadPane()
   {
      Header = "Color Settings",
      Content = new RadColorSelector()
   };

   paneGroup.Items.Add(pane);
   container.Items.Add(paneGroup);
   docking.Items.Add(container);
}
```

The example running in the browser looks like the screenshot below.



Add Tabbed Documents Programmatically

To add a tabbed document, the container isn't added to the Items collection but to the RadDocking control **DocumentHost** property. The example below demonstrates creating a series of document panes for a series of stock market symbols.

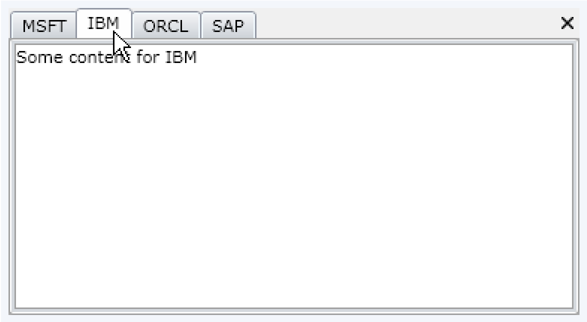
VB

```
Dim symbols() As String = { "MSFT", "IBM", "ORCL", "SAP" }
Dim container = New RadSplitContainer()
Dim paneGroup = New RadPaneGroup()

For Each symbol As String In symbols
  Dim pane = New RadDocumentPane() With { .Header = symbol,  _
.Content = "Some content for " & symbol}
  paneGroup.Items.Add(pane)
Next symbol

container.Items.Add(paneGroup)
docking.DocumentHost = container
```

C#

```
string[] symbols = new string[] { "MSFT", "IBM", "ORCL", "SAP" };
var container = new RadSplitContainer();
var paneGroup = new RadPaneGroup();

foreach (string symbol in symbols)
{
   var pane = new RadDocumentPane()
   {
      Header = symbol,
      Content = "Some content for " + symbol
   };
   paneGroup.Items.Add(pane);
}

container.Items.Add(paneGroup);
docking.DocumentHost = container;
```

The example running in the browser looks like the screenshot below.

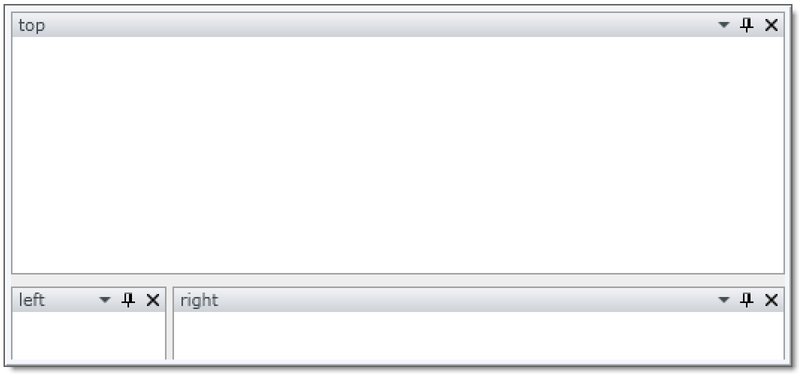


Some content for IBM

23.4.2 Sizing and Positioning

Relative Sizing

Docking panes sizes are dynamic and so its doesn't make sense to set absolute size. Instead, you can set the **RelativeSize** attached property of any two RadSplitContainer or RadPaneGroup elements. The layout in the screenshot below was achieved by setting RelativeSize of RadSplitContainer elements docked top and bottom, and setting RelativeSize of RadPaneGroup elements docked side by side.

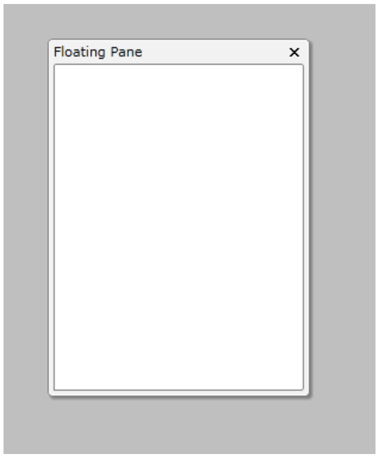


Examine the XAML below to see how the RelativeSize property of the ProportionalStackPanel is applied. The relative vertical sizes for the containers are "70" and "30". Relative horizontal sizes for the two RadPaneGroup elements are "20" and "80". You can use any set of numbers and the relative proportions are calculated automatically.

```
<telerik:RadDocking x:Name="docking">

  <telerik:RadSplitContainer
      InitialPosition="DockedTop"
      telerikDocking:ProportionalStackPanel.RelativeSize="0, 70">
    <telerik:RadPaneGroup>
      <telerik:RadPane Header="top" />
    </telerik:RadPaneGroup>
  </telerik:RadSplitContainer>

  <telerik:RadSplitContainer
      InitialPosition="DockedBottom"
      telerikDocking:ProportionalStackPanel.RelativeSize="0, 30">
    <telerik:RadPaneGroup
        telerikDocking:ProportionalStackPanel.RelativeSize="20, 0">
      <telerik:RadPane Header="left" />
    </telerik:RadPaneGroup>
    <telerik:RadPaneGroup
        telerikDocking:ProportionalStackPanel.RelativeSize="80, 0">
      <telerik:RadPane Header="right" />
    </telerik:RadPaneGroup>
  </telerik:RadSplitContainer>

</telerik:RadDocking>
```

Floating Pane Size and Position

A floating pane doesn't have to be proportionally sized to another window, so we should be able to assign a specific size. While we're at it, can we place the pane in a specific location? The example below places a floating pane 50 pixels from the top left of the page. The pane is sized 300x300 pixels.



Again, this is done through attached properties. See how the XAML below assigns the RadDocking **FloatingLocation** and **FloatingSize** attached properties to a RadSplitContainer.

```
<telerik:RadDocking x:Name="docking">

  <telerik:RadSplitContainer
      InitialPosition="FloatingDockable"
      telerikDocking:RadDocking.FloatingLocation="50, 50"
      telerikDocking:RadDocking.FloatingSize="300, 300">
    <telerik:RadPaneGroup>
      <telerik:RadPane Header="Floating Pane" />
    </telerik:RadPaneGroup>
  </telerik:RadSplitContainer>

</telerik:RadDocking>
```

From the Forums...

Question: The SizeChanged event is not raised, when a pane is resized while floating.

Answer:

Here are a few details about the architecture of the Docking control that will help you better understand when each event is fired and why.

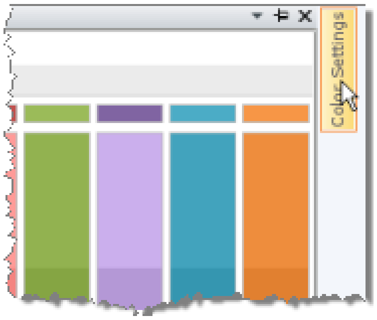
The Docking control contains SplitContainers and they contain PaneGroups. PaneGroups inherit from TabControl and they contain Panes that inherit from TabControlItem. The Pane has Header and Content part (like the TabControlItem) and the Header part is displayed in the TabStrip panel at the top of the TabControl. The actual Content of the TabItem is displayed in an area in the TabControl, because only one Pane's content is visible in any moment. As the TabControl hosts the Content of the TabItem, it is in the TabControl's visual tree.

The TabItem's visual representation is only the button with the header. It is not resized with the whole TabControl. This is the reason why the SizeChanged event of Pane is not fired when you resize the PaneGroup (as the PaneGroup is a TabControl and the Pane is a TabItem). The SizeChaged event is fired on the PaneGroup and on the Content of the currently selected Pane. If you need to handle this event we would recommend you to use the event of the Content of the Pane instead of the event of the PaneGroup as the groups are created and destroyed dynamically.

23.4.3 Pane Pinning and Visibility

Pinning and Auto-Hide Behavior

Set the **IsPinned** property false to have the RadPane "Auto Hide" against one of the edges of the docking area. The pane is collapsed to the side of the docking area where it is docked. The InitialPosition of the RadPane in the screenshot below is DockedRight. When the mouse moves over the unpinned pane button, the pane expands. Use the **AutoHideHeight** and **AutoHideWidth** properties to set the dimensions the pane will occupy when the pane expands. Only the AutoHideWidth property will be taken into account if the pane is docked right or left and only the AutoHideHeight will be taken into account if the pane is docked top or bottom.



Hiding and Showing

Use the RadPane **IsHidden** property to toggle the pane's visibility. You can also use the RadPaneGroup **HideAllPanes()** method to make the panes of the group invisible, all in one shot.

23.4.4 Prevent Docking

If you want to examine the conditions of the docking layout before an action occurs, use one of the "Preview" events: **PreviewShow**, **PreviewPin**, **PreviewClose**, **PreviewUnPin**, **PreviewCloseWindow** and **PreviewShowCompass**. You can set the **Canceled** property based on the arguments passed to the event handler. The last event listed here, PreviewShowCompass, can be used to prevent docking in a particular area. The example below checks if the group being dragged to has a SerializationTag property of "DocumentGroup" and prevents the compass from showing.

```
Private Sub docking_PreviewShowCompass(ByVal sender As Object, ByVal e As PreviewShowCompassEve
  Dim tag As String = e.TargetGroup.GetValue(RadDocking.SerializationTagProperty).ToString()
  e.Canceled = tag.Equals("DocumentGroup")
End Sub
```

C#

```
void docking_PreviewShowCompass(object sender,
  PreviewShowCompassEventArgs e)
{
  string tag =
    e.TargetGroup.GetValue(RadDocking.SerializationTagProperty).ToString();
  e.Canceled = tag.Equals("DocumentGroup");
}
```

23.4.5 Saving and Loading

Each docking group or pane can be identified with a **SerializationTag** property. You can persist the layout using the RadDocking **SaveLayout(stream)** method. Only those panes and groups with the SerializationTag will be saved. Later, when you want to restore the layout, call the **LoadLayout(stream)** method.

The SerializationTag is an attached property that can be added to groups or panes as shown in the XAML below:

XAML

```
<telerik:RadPaneGroup telerikDocking:RadDocking.SerializationTag="Group1">
```

Or you can assign the SerializationTag programmatically using the SetValue() method.

VB

```
Dim symbols() As String = { "MSFT", "IBM", "ORCL", "SAP" }

Dim container = New RadSplitContainer()
Dim paneGroup = New RadPaneGroup()
paneGroup.SetValue(RadDocking.SerializationTagProperty, "DocumentGroup")

For Each symbol As String In symbols
  Dim pane = New RadDocumentPane() With { .Header = symbol,  _
.Content = "Some content for " & symbol}
  pane.SetValue(RadDocking.SerializationTagProperty, symbol & "pane")
  paneGroup.Items.Add(pane)
Next symbol

container.Items.Add(paneGroup)
docking.DocumentHost = container
```

C#

```
string[] symbols = new string[] { "MSFT", "IBM", "ORCL", "SAP" };

var container = new RadSplitContainer();
var paneGroup = new RadPaneGroup();
paneGroup.SetValue(RadDocking.SerializationTagProperty,
   "DocumentGroup");

foreach (string symbol in symbols)
{
   var pane = new RadDocumentPane()
   {
      Header = symbol, Content = "Some content for " + symbol
   };
   pane.SetValue(RadDocking.SerializationTagProperty, symbol + "pane");
   paneGroup.Items.Add(pane);
}

container.Items.Add(paneGroup);
docking.DocumentHost = container;
```

To use the SaveLayout() method, you need to pass it a stream that contains layout information. Any accessible storage mechanism will do, but you may want to consider Isolated Storage. Isolated Storage is a convenient bucket to drop user specific information that may be larger than cookie size. The example below obtains a reference to a IsolatedStorageFileStream that's passed to the SaveLayout() and LoadLayout() methods.

VB

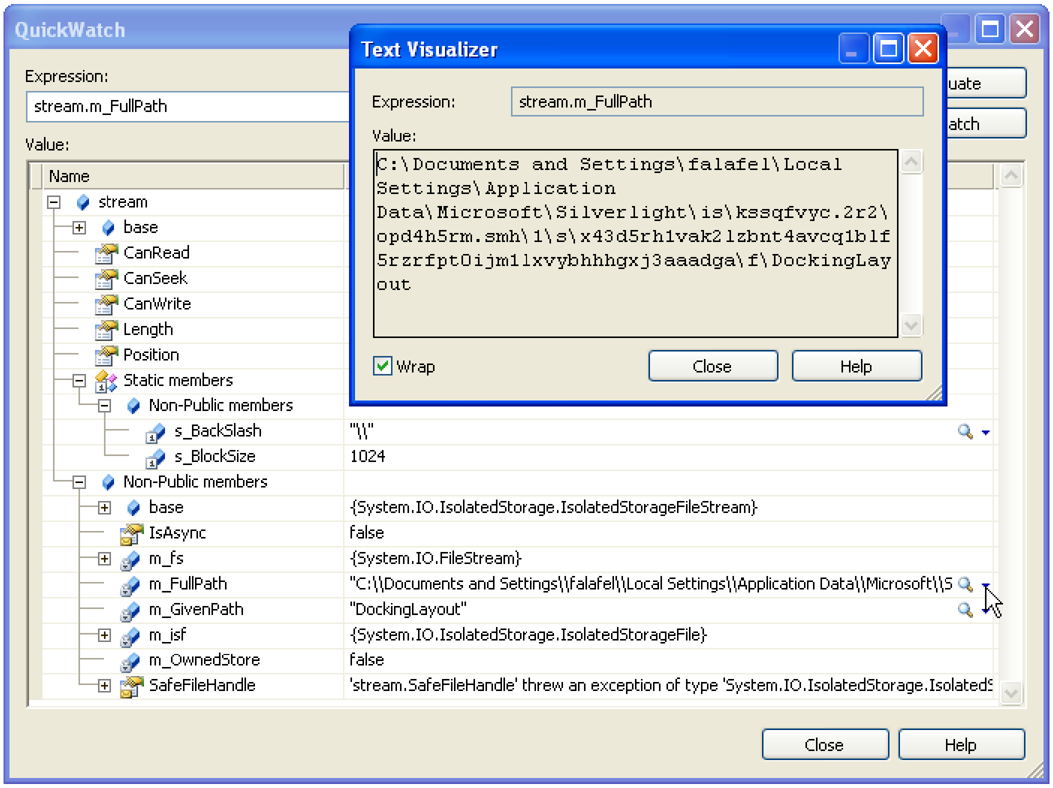
```
Private Const DockingLayoutFileName As String = "DockingLayout"

Private Sub btnSave_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Using isf As IsolatedStorageFile =  _
IsolatedStorageFile.GetUserStoreForApplication()
    Using stream As New IsolatedStorageFileStream( _
DockingLayoutFileName, FileMode.Create, isf)
      docking.SaveLayout(stream)
    End Using
  End Using
End Sub

Private Sub btnLoad_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Using isf As IsolatedStorageFile =  _
IsolatedStorageFile.GetUserStoreForApplication()
    Using stream As New IsolatedStorageFileStream( _
DockingLayoutFileName, FileMode.Open, isf)
      docking.LoadLayout(stream)
    End Using
  End Using
End Sub
```

C#

```
private const string DockingLayoutFileName = "DockingLayout";

private void btnSave_Click(object sender, RoutedEventArgs e)
{
  using (IsolatedStorageFile isf =
       IsolatedStorageFile.GetUserStoreForApplication())
  {
    using (IsolatedStorageFileStream stream =
      new IsolatedStorageFileStream(
       DockingLayoutFileName, FileMode.Create, isf))
    {
      docking.SaveLayout(stream);
    }
  }
}

private void btnLoad_Click(object sender, RoutedEventArgs e)
{
  using (IsolatedStorageFile isf =
       IsolatedStorageFile.GetUserStoreForApplication())
  {
    using (IsolatedStorageFileStream stream =
      new IsolatedStorageFileStream(
        DockingLayoutFileName, FileMode.Open, isf))
    {
      docking.LoadLayout(stream);
    }
  }
}
```

Notes

The isolated storage file is hidden deep within the local documents. You can find the actual path for the document by setting a breakpoint and examining the IsolatedStorageFileStream object in the QuickWatch window.



Expression: stream.m_FullPath

Value:

```
C:\Documents and Settings\falafel\Local
Settings\Application
Data\Microsoft\Silverlight\is\kssqfvyc.2r2\
opd4h5rm.smh\1\s\x43d5rh1vak2lzbnt4avcq1b1f
5rzrfpt0ijm1lxvybhhhgxj3aaadga\f\DockingLay
out
```

☑ Wrap

Name	Value
stream	
base	
CanRead	
CanSeek	
CanWrite	
Length	
Position	
Static members	
Non-Public members	
s_BackSlash	"\\"
s_BlockSize	1024
Non-Public members	
base	{System.IO.IsolatedStorage.IsolatedStorageFileStream}
IsAsync	false
m_fs	{System.IO.FileStream}
m_FullPath	"C:\\Documents and Settings\\falafel\\Local Settings\\Application Data\\Microsoft\\S
m_GivenPath	"DockingLayout"
m_isf	{System.IO.IsolatedStorage.IsolatedStorageFile}
m_OwnedStore	false
SafeFileHandle	'stream.SafeFileHandle' threw an exception of type 'System.IO.IsolatedStorage.IsolatedS

The actual saved layout is in XML form and looks something like the fragment below. Notice the inclusion of the SerializationTag.

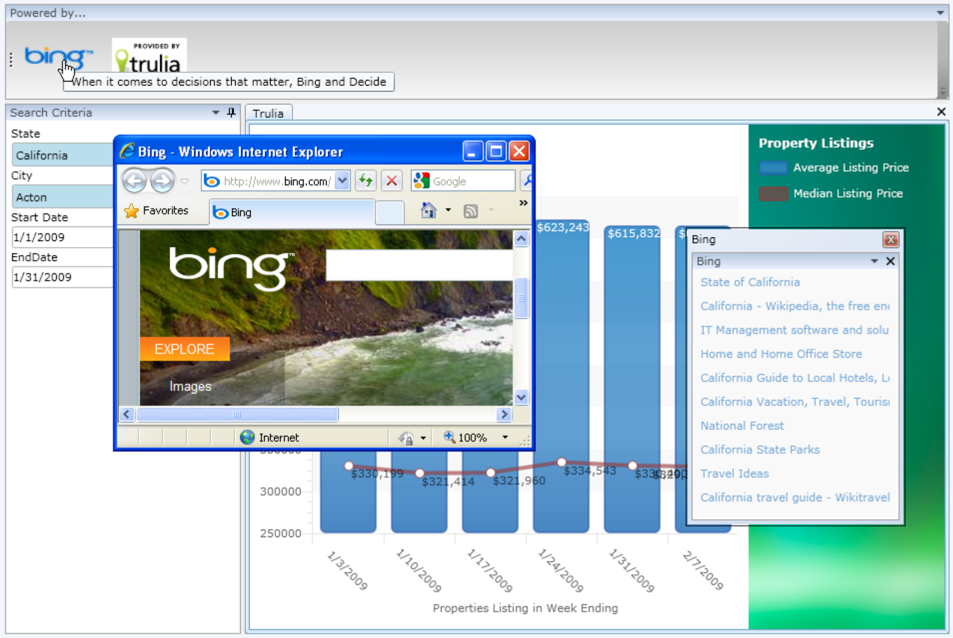
```
<?xml version="1.0" encoding="utf-8"?>
<RadDocking>
 <DocumentHost>
  <RadSplitContainer>
   <Items>
    <RadPaneGroup SerializationTag="DocumentGroup" SelectedIndex="0">
    <Items>
      <RadDocumentPane SerializationTag="MSFTpane" IsDockable="True" Title="MSFT" Header="MSFT" />
      . . .
```

23.5 Binding

RadDocking is a good fit where you are aggregating services from various sources. For example, you could pull in data from related REST based services and display them as one coherent application. A real-estate application can draw on data from sites that expose REST services such as zillow.com or trulia.com. Google and Bing also provide REST API's that return articles about just about any subject. RadGridView and RadChart can be placed in panes and tabbed documents to assemble a high-quality UI.

This section discusses the major tasks you would need to get this done, including:

- Providing logos and links to service provider sites.
- Working around security issues for otherwise "trustworthy" sites that do not have client access policy files.
- Binding to content controls within panes as well as binding to the panes themselves.
- Re-using techniques from earlier chapters including "Input Controls", "ToolBar", "GridView" and "Charting".



State: California
City: Acton
Start Date: 1/1/2009
EndDate: 1/31/2009

State of California
California - Wikipedia, the free enc
IT Management software and solu
Home and Home Office Store
California Guide to Local Hotels, Lo
California Vacation, Travel, Touris
National Forest
California State Parks
Travel Ideas
California travel guide - Wikitravel

Notes

Important note! You will need to visit the developer API sites for any services you use, read any agreements and typically, apply for a user and application key. Many REST API's require the application key to be sent with each call to the service. The application keys have been removed from these examples. The configuration file for the WCF service in the sample solution has a "Mashups" section ("Mashup" being a presentation of aggregate services) that has the developer site urls.

Also be aware that services can change their API without notice. You may need to research and alter the code to keep up with these changes. The larger, more stable sites can be expected to keep their API more consistent over time.

23.5.1 Building the WCF Service

What happens when a REST service or other external resource has information you need, but does not have a ClientAccessPolicy.xml or Crossdomain.xml file? For example, the "Bing" web search API has a ClientAccessPolicy.xml in place (as you might expect from a Microsoft site), but the "Trulia" real estate search site does not have a client access file, so attempts to access from Silverlight fall in a hail of security exceptions. You can step around the issue by wrapping all your REST calls in a WCF service and include the policy file in the root of the service. By accessing external resources through WCF, even from sites without policy files, security exceptions are avoided.

Our WCF service will have a series of calls that total no more than a page of code. The other pieces we need to build to support the service calls are:

- A set of web.config entries to describe the REST API's we intend to consume. The config entries will hold the application keys, paths to logo images, paths to the main site and the developer API reference site and a tool tip.
- A custom ConfigurationSection object to parse the web.config entries. A custom ConfigurationSection allows us to describe N number of services without having to add more code later.
- A ClientAccessPolicy.xml file.
- A set of objects to wrap various REST services: a "RestServiceBase" base class to handle tasks common to all services, such as reading the configuration file and retaining the basic information about the service, "Logo" to contain the services logo image and a link to the site and two RestServiceBase implementations "Bing" and "Trulia".

Notes

This section will not completely reiterate all details on how to build a WCF service. Please refer to the "Data Binding" and "GridView chapters for more complete information.

1) Create a new ASP.NET web site to host the WCF service. **Note:** Remember that this project must be in the same solution with the Silverlight client application that consumes it.

2) Add a "Silverlight-enabled WCF Service" item to the project. **Note:** Be sure to create a "Silverlight-enabled" WCF service item and not a general "WCF service application". The "Silverlight-Enabled" WCF item brings along the proper attributes and configuration entries to work with a Silverlight client application.

3) Add a ClientAccessPolicy.xml file to the project and place the following XML in it:

```xml
<?xml version="1.0" encoding="utf-8"?>
<access-policy>
 <cross-domain-access>
  <policy>
   <allow-from http-request-headers="*">
    <domain uri="*"/>
   </allow-from>
   <grant-to>
    <resource path="/" include-subpaths="true"/>
   </grant-to>
  </policy>
 </cross-domain-access>
</access-policy>
```

4) Add a new section to the web.config file (being sure not to overwrite any existing configuration sections.

The "<section>" element describes the "mashupsSection" to follow and points to a class that will read and parse the custom section. We will write the class in a following step. The format of the "type" section attribute is "<name space>.<class name>, assembly name". Notice that the custom section, "mashupsSection", contains a <services> element and N number of "<add>" elements. Each "<add>" describes a single service.

```
<configuration>
 <configSections>
  <!--custom section-->
  <section
   name="mashupsSection"
   <!--name space + "." + custom section class name, assembly name-->
   type="_03_Binding_Service_Host.MashupsSection, 03_Binding_Service_Host"/>
 </configSections>

 <mashupsSection>
  <services>
   <add
    name="Trulia"
    applicationKey="yourapplicationkeyhere"
    logoUrl="http://images.trulia.com/images/logos/trulia_logo_100x60.jpg"
    linkUrl="http://www.trulia.com"
    developerUrl="http://developer.trulia.com/"
    toolTip="The best place to start your real estate search"
    />
   <add name="Bing"
    applicationKey="yourapplicationkeyhere"
    logoUrl="http://www.bing.com/siteowner/s/siteowner/Logo_63x23_Dark.png"
    linkUrl="http://www.live.com"
    developerUrl="http://www.bing.com/developers"
    toolTip="When it comes to decisions that matter, Bing and Decide"
    />
  </services>
 </mashupsSection>
```

5) Create a new class file called "MashupsSection.cs" and place the code below in it.

The file contains three class implementations: ConfigurationSection, ConfigurationElementCollection and ConfigurationElement. The ConfigurationElement encapsulates an single "<add>" element. This is a minimal custom configuration implementation that we're using to retrieve N number of services (as opposed to application settings that work better for individual bits of information). Search the web on "ConfigurationSection" for more details.

VB

```
Public Class MashupsSection
  Inherits ConfigurationSection
  ' only use the static method GetSection()
  Private Sub New()
  End Sub
  Public Shared Function GetSection() As MashupsSection
    Return TryCast(ConfigurationManager.GetSection("mashupsSection"), _
MashupsSection)
  End Function

  <ConfigurationProperty("services", IsDefaultCollection := False), _
 ConfigurationCollection(GetType(ServiceCollection))> _
  Public ReadOnly Property Services() As ServiceCollection
    Get
      Dim services_Renamed As ServiceCollection = _
TryCast(MyBase.Item("services"), ServiceCollection)
      Return services_Renamed
    End Get
  End Property
End Class

Public Class ServiceCollection
  Inherits ConfigurationElementCollection
  Protected Overrides Function CreateNewElement() As ConfigurationElement
    Return New ServiceElement()
  End Function

  Protected Overrides Function GetElementKey( _
ByVal element As ConfigurationElement) As Object
    Return (CType(element, ServiceElement)).Name
  End Function
End Class

Public Class ServiceElement
  Inherits ConfigurationElement
  <ConfigurationProperty("name", IsRequired := True)> _
  Public ReadOnly Property Name() As String
    Get
      Return TryCast(Me("name"), String)
    End Get
  End Property

  <ConfigurationProperty("applicationKey", IsRequired := True)> _
  Public ReadOnly Property ApplicationKey() As String
    Get
      Return TryCast(Me("applicationKey"), String)
    End Get
  End Property

  <ConfigurationProperty("logoUrl", IsRequired := True)> _
  Public ReadOnly Property LogoUrl() As String
    Get
      Return TryCast(Me("logoUrl"), String)
    End Get
  End Property

  <ConfigurationProperty("linkUrl", IsRequired := True)> _
  Public ReadOnly Property LinkUrl() As String
    Get
      Return TryCast(Me("linkUrl"), String)
    End Get
  End Property

  <ConfigurationProperty("developerUrl", IsRequired := True)> _
  Public ReadOnly Property DeveloperUrl() As String
    Get
      Return TryCast(Me("developerUrl"), String)
    End Get
  End Property

  <ConfigurationProperty("toolTip", IsRequired := True)> _
  Public ReadOnly Property ToolTip() As String
    Get
      Return TryCast(Me("toolTip"), String)
    End Get
  End Property
End Class
```

```
public class MashupsSection : ConfigurationSection
{
    // only use the static method GetSection()
    private MashupsSection() { }

    public static MashupsSection GetSection()
    {
        return ConfigurationManager.GetSection("mashupsSection") as
            MashupsSection;
    }

    [ConfigurationProperty("services", IsDefaultCollection = false)]
    [ConfigurationCollection(typeof(ServiceCollection))]
    public ServiceCollection Services
    {
        get
        {
            ServiceCollection services = base["services"] as ServiceCollection;
            return services;
        }
    }
}

public class ServiceCollection : ConfigurationElementCollection
{
    protected override ConfigurationElement CreateNewElement()
    {
        return new ServiceElement();
    }

    protected override object GetElementKey(ConfigurationElement element)
    {
        return ((ServiceElement)element).Name;
    }
}

public class ServiceElement : ConfigurationElement
{
   [ConfigurationProperty("name", IsRequired = true)]
   public string Name { get { return this["name"] as string; } }

   [ConfigurationProperty("applicationKey", IsRequired = true)]
   public string ApplicationKey
   {
      get { return this["applicationKey"] as string; }
   }

   [ConfigurationProperty("logoUrl", IsRequired = true)]
   public string LogoUrl
   {
      get { return this["logoUrl"] as string; }
   }

   [ConfigurationProperty("linkUrl", IsRequired = true)]
   public string LinkUrl
   {
      get { return this["linkUrl"] as string; }
   }

   [ConfigurationProperty("developerUrl", IsRequired = true)]
   public string DeveloperUrl
   {
      get { return this["developerUrl"] as string; }
   }

   [ConfigurationProperty("toolTip", IsRequired = true)]
   public string ToolTip
   {
      get { return this["toolTip"] as string; }
   }
}
```

6) Create a "Logo.cs" class file and populate with the code below. The "Logo" class is used to pass the service branding information back to the client application for display.

```
Public Class Logo
  Private privateName As String
  Public Property Name() As String
    Get
      Return privateName
    End Get
    Set(ByVal value As String)
      privateName = value
    End Set
  End Property
  Private privateImage As Byte()
  Public Property Image() As Byte()
    Get
      Return privateImage
    End Get
    Set(ByVal value As Byte())
      privateImage = value
    End Set
  End Property
  Private privateLinkUri As Uri
  Public Property LinkUri() As Uri
    Get
      Return privateLinkUri
    End Get
    Set(ByVal value As Uri)
      privateLinkUri = value
    End Set
  End Property
  Private privateLogoUri As Uri
  Public Property LogoUri() As Uri
    Get
      Return privateLogoUri
    End Get
    Set(ByVal value As Uri)
      privateLogoUri = value
    End Set
  End Property
  Private privateToolTip As String
  Public Property ToolTip() As String
    Get
      Return privateToolTip
    End Get
    Set(ByVal value As String)
      privateToolTip = value
    End Set
  End Property
End Class
```

C#

```
public class Logo
{
   public string Name { get; set; }
   public byte[] Image { get; set; }
   public Uri LinkUri { get; set; }
   public Uri LogoUri { get; set; }
   public string ToolTip { get; set; }
}
```

7) Create a "RestServiceBase" class and populate it with the code below.

The class holds the basic service information consumed from the custom configuration section object. The LINQ statement in the constructor filters for the "<add>" element that matches a service name, e.g. "Trulia". The class also has a separate GetLogo() method that downloads the logo image byte array on-demand.

```
Public Class RestServiceBase
  Public Sub New()
  End Sub

  Public Sub New(ByVal name As String)
    Dim mashupSection As MashupsSection = MashupsSection.GetSection()
    Dim element = ( _
        From s As ServiceElement In mashupSection.Services _
        Where s.Name.Equals(name) _
        Select s).First()
    Me.Name = name
    Me.ApplicationKey = element.ApplicationKey
    Me.Logo = New Logo() With {.Name = name, .LinkUri = New Uri(element.LinkUrl), .LogoUri = New Uri(elei
  End Sub

  Private privateName As String
  Public Property Name() As String
    Get
      Return privateName
    End Get
    Set(ByVal value As String)
      privateName = value
    End Set
  End Property
  Private privateApplicationKey As String
  Protected Property ApplicationKey() As String
    Get
      Return privateApplicationKey
    End Get
    Set(ByVal value As String)
      privateApplicationKey = value
    End Set
  End Property
  Private privateLogo As Logo
  Private Property Logo() As Logo
    Get
      Return privateLogo
    End Get
    Set(ByVal value As Logo)
      privateLogo = value
    End Set
  End Property
  Public Function GetLogo() As Logo
    Using client As New WebClient()
      Me.Logo.Image = client.DownloadData(Me.Logo.LogoUri)
    End Using
    Return Me.Logo
  End Function
End Class
```

```
public class RestServiceBase
{
   public RestServiceBase() { }

   public RestServiceBase(string name)
   {
      MashupsSection mashupSection = MashupsSection.GetSection();
      var element = (from ServiceElement s in mashupSection.Services
                  where s.Name.Equals(name)
                  select s).First();
      this.Name = name;
      this.ApplicationKey = element.ApplicationKey;
      this.Logo = new Logo()
      {
         Name  = name,
         LinkUri = new Uri(element.LinkUrl),
         LogoUri = new Uri(element.LogoUrl),
         ToolTip = element.ToolTip
      };
   }

   public string Name { get; set; }
   protected string ApplicationKey { get; set; }
   private Logo Logo { get; set; }
   public Logo GetLogo()
   {
      using (WebClient client = new WebClient())
      {
         this.Logo.Image = client.DownloadData(this.Logo.LogoUri);
      };
      return this.Logo;
   }
}
```

8) Create a "Bing" class and populate it with the code below.

The class descends from our RestServiceBase object and passes the name of the service, e.g. "Bing", to the constructor. Much like the example in the "GridView" chapter section on "REST" binding, this class sets up a URL used to make an API call to a REST service. A WebClient object is used to download XML as a string, the results are parsed and returned from GetResults() as a collection of XElement. The collection of XElement is filtered through a LINQ statement and output placed into new "Result" objects, one for each search result.

Notes

This set of tasks can be similar but not identical from one service to another. Many REST services have a specific URL format published on their developer site that wants your application ID (you need to apply for your own application ID from the vendor site), and some set of parameters. The return value format for many sites usually includes XML but may also be ATOM, JSON, etc.. The XDocument.Parse() method makes short work of converting raw XML into object form where you can then divide and conquer. Typically the developer site will show sample XML output so that you can make some intelligent guesses about what elements to access. Be aware that you will need to see the actual data returned from the service to be sure of the format.

LINQ allows you to filter and convert collections of XElement objects output from the XDocument Parse() into your own custom objects.

```
Public Class Bing
  Inherits RestServiceBase
  ' pass service name to constructor
  Public Sub New()
    MyBase.New("Bing")
  End Sub

  ' url for API calls
  Private Const urlFormat As String = "http://api.search.live.net/xml.aspx?Appid={0}&query={1}&sources={2}"

  ' namespace element for this particular XML
  Private nsWeb As XNamespace = "http://schemas.microsoft.com/LiveSearch/2008/04/XML/web"

  ' format the url, call the api, return "<results>" elements
  Private Function GetResults(ByVal searchString As String) As IEnumerable(Of XElement)
    Using webClient As New WebClient()
      Dim url As String = String.Format(urlFormat, Me.ApplicationKey, searchString, "web")
      Dim xml As String = webClient.DownloadString(New Uri(url))
      If (Not xml.Equals(String.Empty)) Then
        Dim document As XDocument = XDocument.Parse(xml)
        Dim webElement As XElement = document.Root.Element(nsWeb + "Web")
        Dim resultsElement As XElement = webElement.Element(nsWeb + "Results")
        If resultsElement IsNot Nothing Then
          Return resultsElement.Elements()
        End If
      End If
    End Using
    Return Nothing
  End Function

  ' filter and use result elements to return List of "Result"
  Public Function Search(ByVal searchString As String) As List(Of Bing.Result)
    Dim results = GetResults(searchString)
    If results IsNot Nothing Then
      Return ( _
          From r In results _
          Select New Result() With {.Title = r.Element(nsWeb + "Title").Value, .Description = r.Element(nsWeb
    End If
    Return Nothing
  End Function

  ' wraps a single search result
  Public Class Result
    Private privateTitle As String
    Public Property Title() As String
      Get
        Return privateTitle
      End Get
      Set(ByVal value As String)
        privateTitle = value
      End Set
    End Property
    Private privateDescription As String
    Public Property Description() As String
      Get
        Return privateDescription
      End Get
      Set(ByVal value As String)
        privateDescription = value
      End Set
    End Property
    Private privateUrl As String
    Public Property Url() As String
      Get
        Return privateUrl
      End Get
      Set(ByVal value As String)
        privateUrl = value
      End Set
    End Property
    Private privateDateTime As DateTime
    Public Property DateTime() As DateTime
      Get
        Return privateDateTime
      End Get
      Set(ByVal value As DateTime)
        privateDateTime = value
      End Set
    End Property
  End Class
End Class
```

```
public class Bing : RestServiceBase
{
   // pass service name to constructor
   public Bing() : base("Bing") { }
```

```
// url for API calls
private const string urlFormat =
  "http://api.search.live.net/xml.aspx?Appid={0}&query={1}&sources={2}";

// namespace element for this particular XML
private XNamespace nsWeb =
  "http://schemas.microsoft.com/LiveSearch/2008/04/XML/web";

// format the url, call the api, return "<results>" elements
private IEnumerable<XElement> GetResults(string searchString)
{
   using (WebClient webClient = new WebClient())
   {
      string url = String.Format(
         urlFormat, this.ApplicationKey, searchString, "web");
      string xml = webClient.DownloadString(new Uri(url));
      if (!xml.Equals(String.Empty))
      {
         XDocument document = XDocument.Parse(xml);
         XElement webElement =
            document.Root.Element(nsWeb + "Web");
         XElement resultsElement =
            webElement.Element(nsWeb + "Results");
         if (resultsElement != null)
         {
            return resultsElement.Elements();
         }
      }
   }
   return null;
}

// filter and use result elements to return List of "Result"
public List<Bing.Result> Search(string searchString)
{
   var results = GetResults(searchString);
   if (results != null)
   {
      return (from r in results
            select new Result()
            {
               Title = r.Element(nsWeb + "Title").Value,
               Description = r.Element(nsWeb + "Title").Value,
               Url = r.Element(nsWeb + "Url").Value,
               DateTime =
                 DateTime.Parse(r.Element(nsWeb + "DateTime").Value)
            }).ToList();
   }
   return null;
}

// wraps a single search result
public class Result
{
   public string Title { get; set; }
```

```
        public string Description { get; set; }
        public string Url { get; set; }
        public DateTime DateTime { get; set; }
    }
}
```

9) Create a "Trulia" class. The class file is available from the reference projects for this chapter and is omitted here for the sake of brevity.

   The Trulia class also descends from RestServiceBase. The class is slightly more involved than the Bing class but performs the same basic steps, i.e. formats a URL that includes the application key and some set of parameters. Trulia has two sets of API functions, one for "LocationInfo" that returns US states and cities, and "Stats" that return statistics about a location. Again, the results are filtered using LINQ and placed into generic lists of simple Location and Stat objects.

10) Implement the WCF service using the code below. Each method to be used by the Silverlight client is surfaced by a method marked with the OperationContract attribute. Each contract simply surfaces Bing or Trulia object functionality.

```
<ServiceContract(Namespace := ""), _
AspNetCompatibilityRequirements(RequirementsMode := _
AspNetCompatibilityRequirementsMode.Allowed)> _
Public Class MashupService

  <OperationContract> _
  Public Function GetLogos() As List(Of Logo)
   Return New List(Of Logo)()
     CType(New Bing(), Bing).GetLogo(), New Trulia().GetLogo()
  End Function

  <OperationContract> _
  Public Function GetStates() As List(Of Trulia.Location)
    Return New Trulia().GetStates()
  End Function

  <OperationContract> _
  Public Function GetCities(ByVal state As String) As List(Of Trulia.Location)
    Return New Trulia().GetCities(state)
  End Function

  <OperationContract> _
  Public Function GetStateStats(ByVal state As String, _
ByVal start As DateTime, ByVal [end] As DateTime) As List(Of Trulia.Stat)
    Return New Trulia().GetStateStats(state, start, [end])
  End Function

  <OperationContract> _
  Public Function GetCityStats(ByVal state As String, _
ByVal city As String, ByVal start As DateTime,  _
ByVal [end] As DateTime) As List(Of Trulia.Stat)
    Return New Trulia().GetCityStats(state, city, start, [end])
  End Function
  <OperationContract> _
  Public Function GetBingSearchResults( _
ByVal searchString As String) As List(Of Bing.Result)
     Return New Bing().Search(searchString)
  End Function
End Class
```

C#

```
[ServiceContract(Namespace = "")]
[AspNetCompatibilityRequirements(RequirementsMode =
 AspNetCompatibilityRequirementsMode.Allowed)]
public class MashupService
{

   [OperationContract]
   public List<Logo> GetLogos()
   {
     return new List<Logo>()
     {
        new Bing().GetLogo(), new Trulia().GetLogo()
     };
   }

   [OperationContract]
   public List<Trulia.Location> GetStates()
   {
      return new Trulia().GetStates();
   }

   [OperationContract]
   public List<Trulia.Location> GetCities(string state)
   {
      return new Trulia().GetCities(state);
   }

   [OperationContract]
   public List<Trulia.Stat> GetStateStats(
     string state, DateTime start, DateTime end)
   {
      return new Trulia().GetStateStats(state, start, end);
   }

   [OperationContract]
   public List<Trulia.Stat> GetCityStats(
     string state, string city, DateTime start, DateTime end)
   {
      return new Trulia().GetCityStats(state, city, start, end);
   }

   [OperationContract]
   public List<Bing.Result> GetBingSearchResults(string searchString)
   {
```

```
            return new Bing().Search(searchString);
        }
    }
```

23.5.2 Building the Docking Client Application

Notes

This section will not completely reiterate all details on how to build a WCF Silverlight client. Please refer back to the "Data Binding" and "GridView for more complete information.

23.5.2.1 Project Setup

1) Create a new Silverlight application.

2) Add a reference to the WCF service.

3) In the Solution Explorer, add references to the following assemblies:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Charting**

c) **Telerik.Windows.Controls.Docking**

d) **Telerik.Windows.Controls.GridView**

e) **Telerik.Windows.Controls.Input**

f) **Telerik.Windows.Controls.Navigation**

g) **Telerik.Windows.Data**

h) **Telerik.Windows.Themes.Summer**

23.5.2.2 Silverlight Client Code Behind

1) Add an ImageConverter class. This is the same code used in the "GridView" chapter in the "Customization" section. The converter will be used when binding the logo images.

VB

```
Public Class ImageConverter
  Implements IValueConverter
  Public Function Convert(ByVal value As Object, _
ByVal targetType As Type, ByVal parameter As Object, _
ByVal culture As CultureInfo) As Object
    ' convert the "Picture" byte array to a memory stream
    Dim memoryStream As New MemoryStream((CType(value, Byte())))
    Dim image As New BitmapImage()
    Try
      image.SetSource(memoryStream)
    Catch ' ignore invalid arrays
    End Try
    Return image
  End Function

  Public Function ConvertBack(ByVal value As Object, _
ByVal targetType As Type, ByVal parameter As Object, _
ByVal culture As CultureInfo) As Object
    Throw New NotImplementedException()
  End Function
End Class
```

C#

```
public class ImageConverter : IValueConverter
{
   public object Convert(
      object value, Type targetType, object parameter, CultureInfo culture)
   {
      // convert the "Picture" byte array to a memory stream
      MemoryStream memoryStream = new MemoryStream(((byte[])value));
      BitmapImage image = new BitmapImage();
      try { image.SetSource(memoryStream); }
      catch { } // ignore invalid arrays
      return image;
   }

   public object ConvertBack(object value, Type targetType,
     object parameter, CultureInfo culture)
   { throw new NotImplementedException(); }
}
```

2) In the code behind for the page, add references to the following namespaces:

a) **System.Windows.Media**

b) **System.Windows.Media.Imaging**

c) **Telerik.Windows.Controls**

d) **Telerik.Windows.Controls.Charting**

3) Add a reference to the namespace for the WCF proxy client. This will be the name of the client project + "." + the reference name for the proxy client. If you have trouble finding the name through IntelliSense, be sure to rebuild the project.

4) Create a new **SummerTheme** object in the constructor before the call to InitializeComponent() and set the **IsApplicationTheme** property to "True". This will apply the Summer theme to all RadControls in the application. **Note:** Be sure to leave the call to InitializeComponent in the constructor.

VB

```
Public Sub New()
  CType(New SummerTheme(), SummerTheme).IsApplicationTheme = True
  InitializeComponent()
End Sub
```

C#

```
public MainPage()
{
   new SummerTheme().IsApplicationTheme = true;
   InitializeComponent();
}
```

5) Add a reference to the WCF client proxy object so that we can reference it throughout the code.

VB

```
Private client As New MashupServiceClient()
```

C#

```
private MashupServiceClient client = new MashupServiceClient();
```

6) Add a method to hook up all the WCF client proxy events and add the events handlers.

Notice that most of the "Completed" events simply assign the Result property of the event arguments to the ItemsSource of a RadControl. The GetLogosCompleted() method uses LINQ to locate a Logo object for one of the services, i.e. "Bing" or "Trulia", and assigns the object to the DataContext of a RadPane where the service is being used. The asynchronous events that are used to retrieve data are discussed in the "Data Binding" and "GridView" chapters.

VB

```
Private Sub HookupServiceEvents()
  AddHandler client.GetLogosCompleted, AddressOf client_GetLogosCompleted
  AddHandler client.GetStatesCompleted, AddressOf client_GetStatesCompleted
  AddHandler client.GetCitiesCompleted, AddressOf client_GetCitiesCompleted
  AddHandler client.GetStateStatsCompleted,
AddressOf client_GetStateStatsCompleted
  AddHandler client.GetCityStatsCompleted,  _
AddressOf client_GetCityStatsCompleted
  AddHandler client.GetBingSearchResultsCompleted, _
AddressOf client_GetBingSearchResultsCompleted
End Sub

Private Sub client_GetCityStatsCompleted( _
ByVal sender As Object, ByVal e As GetCityStatsCompletedEventArgs)
  chart.ItemsSource = e.Result
End Sub

Private Sub client_GetStateStatsCompleted( _
ByVal sender As Object, ByVal e As GetStateStatsCompletedEventArgs)
  chart.ItemsSource = e.Result
End Sub

Private Sub client_GetCitiesCompleted( _
ByVal sender As Object, ByVal e As GetCitiesCompletedEventArgs)
  cbCity.ItemsSource = e.Result
End Sub

Private Sub client_GetStatesCompleted( _
ByVal sender As Object, ByVal e As GetStatesCompletedEventArgs)
  cbState.ItemsSource = e.Result
End Sub

Private Sub client_GetBingSearchResultsCompleted( _
ByVal sender As Object, ByVal e As GetBingSearchResultsCompletedEventArgs)
  gvResults.ItemsSource = e.Result
End Sub

Private Sub client_GetLogosCompleted( _
ByVal sender As Object, ByVal e As GetLogosCompletedEventArgs)
  tbLogos.ItemsSource = e.Result
  paneChart.DataContext =  _
e.Result.FirstOrDefault(Function(logo) logo.Name.Equals("Trulia"))
  paneTopStories.DataContext =  _
e.Result.FirstOrDefault(Function(logo) logo.Name.Equals("Bing"))
End Sub
```

C#

```
private void HookupServiceEvents()
{
   client.GetLogosCompleted += new EventHandler<GetLogosCompletedEventArgs>(
         client_GetLogosCompleted);
   client.GetStatesCompleted += new EventHandler<GetStatesCompletedEventArgs>(
         client_GetStatesCompleted);
   client.GetCitiesCompleted += new EventHandler<GetCitiesCompletedEventArgs>(
         client_GetCitiesCompleted);
   client.GetStateStatsCompleted +=
      new EventHandler<GetStateStatsCompletedEventArgs>(
         client_GetStateStatsCompleted);
```

```
    client.GetCityStatsCompleted +=
      new EventHandler<GetCityStatsCompletedEventArgs>(
        client_GetCityStatsCompleted);
    client.GetBingSearchResultsCompleted +=
      new EventHandler<GetBingSearchResultsCompletedEventArgs>(
        client_GetBingSearchResultsCompleted);
  }

  void client_GetCityStatsCompleted(object sender,
    GetCityStatsCompletedEventArgs e)
  {
    chart.ItemsSource = e.Result;
  }

  void client_GetStateStatsCompleted(object sender,
    GetStateStatsCompletedEventArgs e)
  {
    chart.ItemsSource = e.Result;
  }

  void client_GetCitiesCompleted(object sender,
  GetCitiesCompletedEventArgs e)
  {
    cbCity.ItemsSource = e.Result;
  }

  void client_GetStatesCompleted(object sender,
    GetStatesCompletedEventArgs e)
  {
    cbState.ItemsSource = e.Result;
  }

  void client_GetBingSearchResultsCompleted(object sender,
    GetBingSearchResultsCompletedEventArgs e)
  {
    gvResults.ItemsSource = e.Result;
  }

  void client_GetLogosCompleted(object sender,
    GetLogosCompletedEventArgs e)
  {
    tbLogos.ItemsSource = e.Result;
    paneChart.DataContext =
     e.Result.FirstOrDefault(logo => logo.Name.Equals("Trulia"));
    paneTopStories.DataContext =
     e.Result.FirstOrDefault(logo => logo.Name.Equals("Bing"));
  }
```

7) Add a private method to setup the chart.

The code is very similar to the code in the "Charting" chapter, "Binding" section dealing with "Binding Basics". The code sets up two series mappings, one for "average listing price" and the second for "median listing price". Notice that the "Week ending date" data is mapped to the XCategory data point member so that the dates are listed along the bottom of the chart. See the "Charting" chapter for more information on working with series mapping, axis and legends.

```
Private Sub SetupChart()

 ' format as currency
  Dim averageMapping As New SeriesMapping() With { _
.LegendLabel = "Average Listing Price", _
.SeriesDefinition = New BarSeriesDefinition() With { _
.DefaultLabelFormat = "#Y{C0}"}}

  averageMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.XCategory, _
.FieldName = "WeekEndingDate"})

  averageMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.YValue, _
.FieldName = "AverageListingPrice"})

  chart.SeriesMappings.Add(averageMapping)

 ' format as currency
  Dim medianMapping As New SeriesMapping() With { _
.LegendLabel = "Median Listing Price", _
.SeriesDefinition = New LineSeriesDefinition() With { _
.DefaultLabelFormat = "#Y{C0}"}}

  medianMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.XCategory, _
.FieldName = "WeekEndingDate"})

  medianMapping.ItemMappings.Add(New ItemMapping() With { _
.DataPointMember = DataPointMember.YValue, _
.FieldName = "MedianListingPrice"})

  chart.SeriesMappings.Add(medianMapping)

  chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45
  chart.DefaultView.ChartArea.AxisX.IsDateTime = True
  chart.DefaultView.ChartArea.AxisX.Title = "Properties Listing in Week Ending"
  chart.DefaultView.ChartArea.AxisX.DefaultLabelFormat = "d"
  chart.DefaultView.ChartLegend.Header = "Property Listings"
End Sub
```

```
private void SetupChart()
{
   SeriesMapping averageMapping = new SeriesMapping()
   {
      LegendLabel = "Average Listing Price",
      SeriesDefinition = new BarSeriesDefinition()
      { DefaultLabelFormat = "#Y{C0}"  // format as currency }
   };
   averageMapping.ItemMappings.Add(new ItemMapping()
   {
      DataPointMember = DataPointMember.XCategory,
      FieldName = "WeekEndingDate"
   });

   averageMapping.ItemMappings.Add(new ItemMapping()
   {
      DataPointMember = DataPointMember.YValue,
      FieldName = "AverageListingPrice",
   });

   chart.SeriesMappings.Add(averageMapping);

   SeriesMapping medianMapping = new SeriesMapping()
   {
      LegendLabel = "Median Listing Price",
      SeriesDefinition = new LineSeriesDefinition()
      { DefaultLabelFormat = "#Y{C0}" // format as currency }
   };

   medianMapping.ItemMappings.Add(new ItemMapping()
   {
      DataPointMember = DataPointMember.XCategory,
      FieldName = "WeekEndingDate"
   });

   medianMapping.ItemMappings.Add(new ItemMapping()
   {
      DataPointMember = DataPointMember.YValue,
      FieldName = "MedianListingPrice",
   });

   chart.SeriesMappings.Add(medianMapping);

   chart.DefaultView.ChartArea.AxisX.LabelRotationAngle = 45;
   chart.DefaultView.ChartArea.AxisX.IsDateTime = true;
   chart.DefaultView.ChartArea.AxisX.Title = "Properties Listing in Week Ending";
   chart.DefaultView.ChartArea.AxisX.DefaultLabelFormat = "d";
   chart.DefaultView.ChartLegend.Header = "Property Listings";
}
```

8) Add a private method to kick off the Bing search.

The method takes the state and city, formats a search string and calls the asynchronous Bing search method. The search is hard coded to bring back results about "Real Estate" in a particular state and city, i.e. "San Francisco, California Real Estate".

VB

```
Private Sub BingSearch(ByVal state As TruliaLocation, ByVal city As TruliaLocation)
  Dim searchString As String = String.Empty
  Const RealEstate As String = " Real Estate"

  If city IsNot Nothing Then
    searchString = city.Name & "," & state.Name & " " & RealEstate
  ElseIf state IsNot Nothing Then
    searchString = state.Name & searchString
  End If

  client.GetBingSearchResultsAsync(searchString)
End Sub
```

C#

```
private void BingSearch(TruliaLocation state, TruliaLocation city)
{
   string searchString = String.Empty;
   const string RealEstate = " Real Estate";

   if (city != null)
   {
      searchString =
         city.Name + "," + state.Name + " " + RealEstate;
   }
   else if (state != null)
   {
      searchString = state.Name + searchString;
   }

   client.GetBingSearchResultsAsync(searchString);
}
```

9) Handle the Loaded event of the page.

The event handler calls the HookupServiceEvents() method so that the WCF client proxy events will fire, calls client proxy methods to get the logos and load up the "State" combo box. The selected dates for the date picker controls are hard coded to a date range known to have data at the time of this writing. Finally, the SetupChart() method is called to setup the series and items mappings.

VB

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  HookupServiceEvents()

  client.GetLogosAsync()
  client.GetStatesAsync()

  dpStart.SelectedDate = New DateTime(DateTime.Today.Year, 1, 1)
  dpEnd.SelectedDate = dpStart.SelectedDate.Value.AddDays(30)

  SetupChart()
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   HookupServiceEvents();

   client.GetLogosAsync();
   client.GetStatesAsync();

   dpStart.SelectedDate = new DateTime(DateTime.Today.Year, 1, 1);
   dpEnd.SelectedDate = dpStart.SelectedDate.Value.AddDays(30);

   SetupChart();
}
```

10)Handle the Click event for the Search button.

This event handler starts the Bing search and calls a WCF client proxy method to get state or city statistics depending on user selection in the state and city combo boxes.

VB

```
Private Sub btnSearch_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim state = If(cbState.SelectedValue Is Nothing, Nothing, TryCast(cbState.SelectedValue, TruliaLocation))
  Dim city = If(cbCity.SelectedValue Is Nothing, Nothing, TryCast(cbCity.SelectedValue, TruliaLocation))

  BingSearch(state, city)

  If city IsNot Nothing Then
    client.GetCityStatsAsync(state.ID, city.Name, CDate(dpStart.SelectedDate), CDate(dpEnd.SelectedDate))
  ElseIf state IsNot Nothing Then
    client.GetStateStatsAsync(state.ID, CDate(dpStart.SelectedDate), CDate(dpEnd.SelectedDate))
  End If
End Sub
```

C#

```
private void btnSearch_Click(object sender, RoutedEventArgs e)
{
   var state = cbState.SelectedValue == null ? null :
      cbState.SelectedValue as TruliaLocation;
   var city = cbCity.SelectedValue == null ? null :
      cbCity.SelectedValue as TruliaLocation;

   BingSearch(state, city);

   if (city != null)
   {
      client.GetCityStatsAsync(
         state.ID, city.Name,
         (DateTime)dpStart.SelectedDate, (DateTime)dpEnd.SelectedDate);
   }
   else if (state != null)
   {
      client.GetStateStatsAsync(
         state.ID, (DateTime)dpStart.SelectedDate, (DateTime)dpEnd.SelectedDate);
   }
}
```

11)Handle the SelectionChanged event for the state combo box.

This event handler calls a WCF client proxy method to get cities for the currently selected state.

VB

```
Private Sub cbState_SelectionChanged( _
ByVal sender As Object, _
ByVal e As Telerik.Windows.Controls.SelectionChangedEventArgs)
  Dim state As TruliaLocation = _
TryCast(cbState.SelectedValue, TruliaLocation)
  client.GetCitiesAsync(state.ID)
End Sub
```

C#

```
private void cbState_SelectionChanged(object sender,
   Telerik.Windows.Controls.SelectionChangedEventArgs e)
{
   TruliaLocation state = cbState.SelectedValue as TruliaLocation;
   client.GetCitiesAsync(state.ID);
}
```

23.5.2.3 Silverlight Client XAML

1) Add XML namespace references for

a) **Telerik.Windows.Controls.Navigation**

b) **Telerik.Windows.Controls.GridView**

c) **Telerik.Windows.Controls.Charting**

d) **Telerik.Windows.Controls.Input**

e) **Telerik.Windows.Controls.Docking**

f) A reference to the namespace where the image IValueConverter lives. The xmlns alias in this example will be named "local".

```
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
```

2) Add a Loaded event handler to the UserControl that points at the Loaded event handler we added earlier to the code behind.

3) Add a UserControl.Resources element with a style for the grid view and a reference to the ImageConverter.

The grid view style uses the settings described in the "GridView" chapter "Control Details" section dealing with "Grid View Elements Visibility". The style turns off the usual visual accoutrements of the grid view so that we will only see a list of hyperlinks to search results.

```
<UserControl.Resources>
  <Style x:Key="GridlessStyle"
      TargetType="telerik:RadGridView">
    <Setter Property="Background" Value="Transparent" />
    <Setter Property="BorderBrush" Value="Transparent" />
    <Setter Property="CanUserFreezeColumns" Value="False" />
    <Setter Property="ShowColumnFooters" Value="False" />
    <Setter Property="ShowColumnHeaders" Value="False" />
    <Setter Property="ShowGroupPanel" Value="False" />
    <Setter Property="ShowGroupFooters" Value="False" />
    <Setter Property="RowIndicatorVisibility"
        Value="Collapsed" />
    <Setter Property="GridLinesVisibility" Value="None" />
    <Setter Property="ScrollMode" Value="RealTime" />
    <Setter Property="ColumnWidth" Value="SizeToHeader" />
  </Style>

  <local:ImageConverter x:Key="ImageConverter" />

</UserControl.Resources>
```

4) Add the XAML below to the main "LayoutRoot" grid element. This will setup the basic RadDocking structure. The setup is similar to the "Getting Started" walk through. Take a minute to review the XAML and locate the comments where RadControls will be placed, e.g. "<!--add tool bar here-->". Also notice the RadPane settings.

```
<telerik:RadDocking>
  <!--top panel-->
  <telerik:RadSplitContainer
      InitialPosition="DockedTop" Orientation="Vertical"
      MaxHeight="100" MinHeight="100">
    <!--logo area-->
    <telerik:RadPaneGroup>
      <telerik:RadPane Header="Powered by..."
          CanUserClose="False" CanUserPin="False"
          CanFloat="False"
          CanDockInDocumentHost="False">
        <!--add tool bar here-->
      </telerik:RadPane>
    </telerik:RadPaneGroup>
  </telerik:RadSplitContainer>
  <!--search panel-->
  <telerik:RadSplitContainer
      InitialPosition="DockedLeft">
    <telerik:RadPaneGroup>
      <telerik:RadPane Header="Search Criteria"
          CanDockInDocumentHost="False"
          CanFloat="True" CanUserClose="False"
          CanUserPin="True">
        <!--add search controls here-->
      </telerik:RadPane>
    </telerik:RadPaneGroup>
  </telerik:RadSplitContainer>
  <!--center, document area-->
  <telerik:RadDocking.DocumentHost>
    <telerik:RadSplitContainer>
      <telerik:RadPaneGroup>
        <telerik:RadPane
            x:Name="paneTopStories"
            Header="{Binding Name, Mode=OneTime }">
          <!--add grid view here-->
        </telerik:RadPane>
        <telerik:RadPane x:Name="paneChart"
            Header="{Binding Name, Mode=OneTime }">
          <!--add chart here-->
        </telerik:RadPane>
      </telerik:RadPaneGroup>
    </telerik:RadSplitContainer>
  </telerik:RadDocking.DocumentHost>
</telerik:RadDocking>
```

5) Locate the "<!–add tool bar here–>" comment and add the following XAML to define a RadToolBar in the top pane.

Here we define the RadToolBar ItemTemplate that contains a HyperlinkButton surrounding an Image for each Logo we return from the WCF service. Notice the bindings for the NavigateUri, ToolTip and Source. See the "ToolBar" Binding section for more information.

```
<telerik:RadToolBar x:Name="tbLogos">
  <telerik:RadToolBar.ItemTemplate>
    <DataTemplate>
      <HyperlinkButton
          NavigateUri="{Binding LinkUri, Mode=OneTime }"
          ToolTipService.ToolTip="{Binding ToolTip, Mode=OneTime }"
          TargetName="_blank"
          Margin="5">
        <Image
          Source="{Binding Image, Converter={StaticResource ImageConverter}}"
          Width="75" Height="75"
          Stretch="Uniform"></Image>
      </HyperlinkButton>
    </DataTemplate>
  </telerik:RadToolBar.ItemTemplate>
</telerik:RadToolBar>
```

6) Locate the "<!--add search controls here-->" comment and add the following XAML.

Here we define a series of input controls in a StackPanel that will allow the user to define search criteria. Notice that the DisplayMemberPath for both combo boxes is the "Name" property of "Location" objects returned from the WCF service. See the "HTMLPlaceholder", "Date, Time and Calendar" and "ComboBox" chapters for more information.

```
<StackPanel Margin="5">
  <TextBlock Text="State"></TextBlock>
  <telerik:RadComboBox x:Name="cbState"
      DisplayMemberPath="Name"
      HorizontalAlignment="Stretch"
      SelectionChanged="cbState_SelectionChanged" />
  <TextBlock Text="City"></TextBlock>
  <telerik:RadComboBox x:Name="cbCity"
      DisplayMemberPath="Name"
      HorizontalAlignment="Stretch" />
  <TextBlock Text="Start Date"></TextBlock>
  <telerik:RadDatePicker
      x:Name="dpStart"
      HorizontalAlignment="Stretch" />
  <TextBlock Text="EndDate"></TextBlock>
  <telerik:RadDatePicker x:Name="dpEnd"
      HorizontalAlignment="Stretch" />
  <Button x:Name="btnSearch"
      HorizontalAlignment="Right"
      Margin="5" Content="Search"
      Click="btnSearch_Click" />
</StackPanel>
```

7) Locate the "<!--add grid view here-->" comment and add the following XAML.

The grid view uses the "GridlessStyle" defined in the UserControl Resources. AutoGenerateColumns is turned off and a single hyperlink column is bound to the results of the Bing search where the link is to the "Url" property and the visible content displays the "Title" property. See the "GridView" chapter "Binding" section for more information.

```
<telerik:RadGridView
      x:Name="gvResults"
      Style="{StaticResource GridlessStyle}"
      AutoGenerateColumns="False">
   <telerik:RadGridView.Columns>
      <telerik:GridViewHyperlinkColumn
            DataMemberBinding="{Binding Url}"
            ContentBinding="{Binding Title}"
            TargetName="_blank" />
   </telerik:RadGridView.Columns>
</telerik:RadGridView>
```

8) Locate the "<!--add chart here-->" comment and add the following XAML.

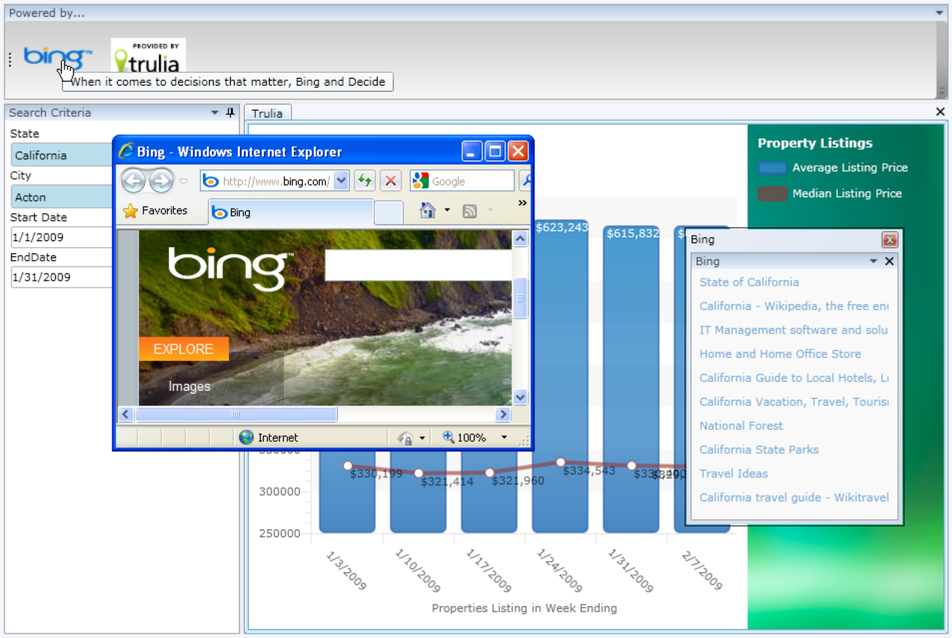
The chart is named so we can access it and setup series mappings in code. See the "Charting" chapter "Binding" section for more information.

```
<telerik:RadChart x:Name="chart"></telerik:RadChart>
```

23.5.2.4 Run and Test the Application

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



State: California
City: Acton
Start Date: 1/1/2009
EndDate: 1/31/2009

Test Application Features

1) When you first run the application, the Bing and Trulia logos should appear. Tool tips for each logo should appear as you pass the mouse over. Expect the tool tip text to match the WCF web.config file entries for each service. Click each logo image. The web site for each service should appear in a separate browser window.

2) Drop down the State combo box. The state names should appear there.

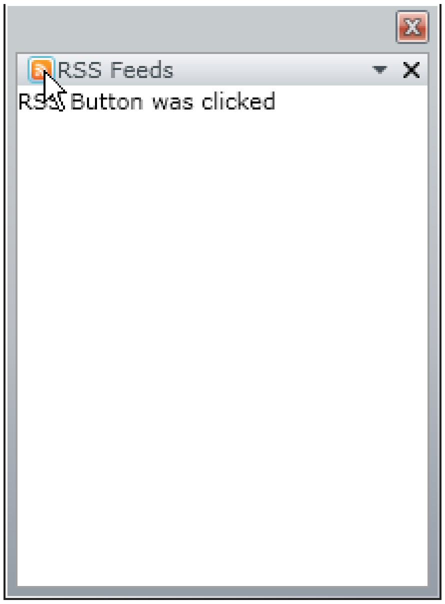
3) Select a state and click the "Search" button. The Trulia and Bing panes should populate with data. The Trulia pane should display a chart with two series and the Bing pane should display a grid view list of links.

Ideas for Extending This Example

- Research and add a new service.
- Read the Customization section of this chapter and customize the RadPane header to include the service logo.
- Define other attributes of each service. For example, you could extract the colors from the service logos, e.g. the Bing blue color or the Trulia green color and add them to the configuration file. Add plumbing to make the color available in the application and use it to color the custom RadPane header background.
- RadDocking is a ItemsControl descendant. Bind to the entire RadDocking control instead of individual windows.

23.6 Customization

One question that comes up often in the forums is how to get a button in the title area of a pane. RadPane has a number of templates that you can override to put your own stamp on the control. Depending on the circumstances of the pane, different templates will be applied. For instance, the **DocumentHostTemplate** is applied when the RadPane is placed in the DocumentHost. Depending on the docked position of the pane, the **BottomTemplate**, **LeftTemplate**, **RightTemplate** or **TopTemplate** might be in play. We will define the **TitleTemplate** and add a button that will act on the content of the pane.



RSS Button was clicked

Inside the RadPane element, place the TitleTemplate element followed by the DataTemplate element. Inside that you can place whatever arbitrary markup suits your purpose. Here we add a StackPanel to contain a **Button**, **Image** inside the button and a **TextBlock** for the title text. Notice the Click event handler hooked up to the Button element.

XAML

```
<telerik:RadDocking>
   <telerik:RadSplitContainer>
      <telerik:RadPaneGroup>
         <telerik:RadPane x:Name="paneRSS">
            <telerik:RadPane.TitleTemplate>
               <DataTemplate>
                  <StackPanel Orientation="Horizontal">
                     <Button Click="Button_Click"
                           Width="16" Height="16">
                        <Image Source="Rss.png"
                              Stretch="Uniform" />
                     </Button>
                     <TextBlock Text="RSS Feeds" />
                  </StackPanel>
               </DataTemplate>
            </telerik:RadPane.TitleTemplate>
         </telerik:RadPane>
      </telerik:RadPaneGroup>
   </telerik:RadSplitContainer>
</telerik:RadDocking>
```

The Click event handler simply assigns some test content to the panel Content property.

VB

```
Private Sub Button_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  paneRSS.Content = "RSS Button was clicked"
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   paneRSS.Content = "RSS Button was clicked";
}
```

23.7 Wrap Up

In this chapter you learned how to create flexible layouts using the RadDocking control. You started by creating a layout entirely in XAML. During the initial project you learned how to control user interaction with panes including floating, closing and pinning behaviors.

In the Control Details section of the chapter you learned how to create split containers, groups and panes all in code, how to size and position floating panes, how to save and load panes, how to control pinning behavior, how to hide and show panes and how to use various "Preview" events.

During the Binding section of this chapter you learned how to build a Silverlight "mashup" client application with docking that aggregated various REST services. The service output was bound to both the pane content and the pane title area.

During the Customization section of the chapter you customized the pane title area to include a button and image.

Part XIV

Windows

24 Windows

24.1 Objectives

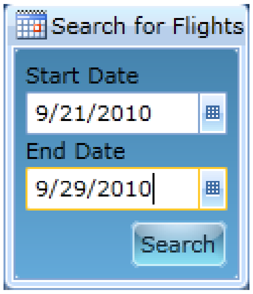
In this chapter you will learn how to build RadWindows with free-form content and control them as a group with RadWindowManager. You will work with predefined Alert, Confirm and Prompt dialogs. You will learn how to control window appearance, both with simple background and title color modification as well as customization using Expression Blend. You will react to windows opening, closing, state change and moving using RadWindow events. You will learn how to alter the initial state of a window. You will also how to bind data RadWindow properties.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Window\Window.sln.

24.2 Overview

Use RadWindow to frame important information or collect user input. RadWindow is a themed, floating container that can be displayed free floating or as a modal dialog. The event model lets you keep track of window movement and state changes, allowing you to intercede with custom logic. Alerts, Confirmation dialogs and Prompts are all built in. The RadWindowManager provides access and control to all windows as a collection and PopupManager can be used for special situations that require tweaking the Z-Order.


Search for Flights

Start Date: 9/21/2010

End Date: 9/29/2010

RadWindow includes these features:

- Child Windows Across All Platforms and Scenarios
- Predefined Dialogs
- Styling and Appearance
- Customizable Behavior: choose window size, state and position parameters
- Customizable Content

24.3 Getting Started

You can define a window in XAML and open it from code or you may need to generate windows from scratch dynamically. The following walk through shows you how to build windows on-the-fly, entirely in code.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Navigation**

c) **Telerik.Windows.Themes.Summer**

XAML Editing

1) Open MainPage.xaml for editing.
2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. This will define a RadToolBar with two buttons and their respective Click handlers.

```
<telerik:RadToolBar VerticalAlignment="Top">
  <telerik:RadToolBar.Items>
    <Button x:Name="btnNewWindow" Content="New Window"
        Click="btnNewWindow_Click" />
    <Button x:Name="btnCloseAll" Content="Close All"
        Click="btnCloseAll_Click" />
  </telerik:RadToolBar.Items>
</telerik:RadToolBar>
```

Code Behind

1) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for the **Telerik.Windows.Controls** namespace.
2) In the constructor, create a new **SummerTheme** instance and set the **IsApplicationTheme** property to "True". This will style both the tool bar and any windows that are created.

VB

```
Public Sub New()
  CType(New SummerTheme(), SummerTheme).IsApplicationTheme = True
  InitializeComponent()
End Sub
```

C#

```
public MainPage()
{
    new SummerTheme().IsApplicationTheme = true;
    InitializeComponent();
}
```

3) Handle the Click event for the "New Window" button.

The code creates a RadWindow instance and calls the Show() method. Here we also set the window Header, Content, starting position and dimensions.

VB

```
Private Sub btnNewWindow_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim window As New RadWindow()
  window.Header = "This is the header"
  window.Content = "Content goes here"
  window.SetValue(RadWindow.WidthProperty, 300R)
  window.SetValue(RadWindow.HeightProperty, 300R)
  window.WindowStartupLocation = WindowStartupLocation.CenterScreen
  window.Show()
End Sub
```

C#

```
private void btnNewWindow_Click(object sender, RoutedEventArgs e)
{
    RadWindow window = new RadWindow();
    window.Header = "This is the header";
    window.Content = "Content goes here";
    window.SetValue(RadWindow.WidthProperty, 300d);
    window.SetValue(RadWindow.HeightProperty, 300d);
    window.WindowStartupLocation = WindowStartupLocation.CenterScreen;
    window.Show();
}
```

4) Handle the Click event for the "Close All" button. This event handler calls the **RadWindowManager CloseAllWindows()** method of the singleton RadWindowManager.

```
Private Sub btnCloseAll_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  RadWindowManager.Current.CloseAllWindows()
End Sub
```

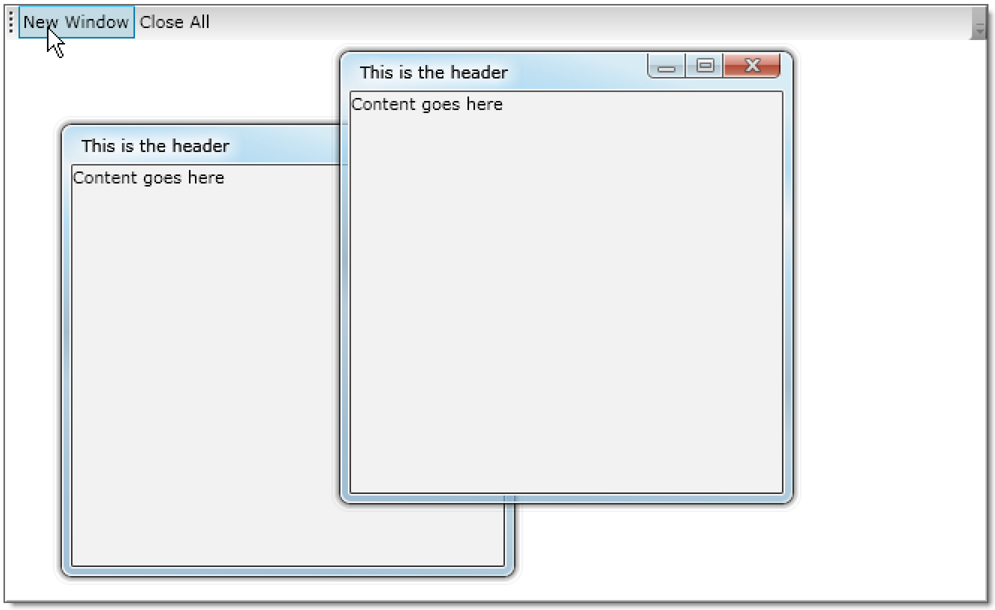
```
private void btnCloseAll_Click(object sender, RoutedEventArgs e)
{
   RadWindowManager.Current.CloseAllWindows();
}
```

Gotcha!

RadWindowManager has several deprecated methods that are static. You may get a semi-cryptic message to "use RadWindowManager instance methods". By this, the message means to use the singleton "Current" property of the RadWindowManager as shown in the code snippet above.

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

1) Open multiple windows using the "New Window" button.

2) Close all the windows at once using the "Close All" button.

Ideas for Extending This Example

- Add additional content to the windows.
- Use the other methods of the RadWindowManager such as **MaximizeAllWindows()**, **MinimizeAllWindows()** and **NormalAllWindows()**.

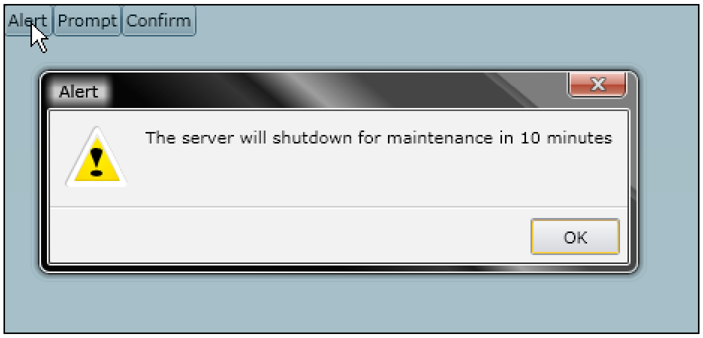
24.4 Control Details

24.4.1 Predefined Dialogs

RadWindow comes with predefined **Alert()**, **Prompt()** and **Confirm()** dialog methods. These "workhorse" static methods of RadWindow can be very simple or fully tailorable.

24.4.1.1 Alert

For quick-and-dirty alerts, simply call the the **Alert()** method and pass the content.



The server will shutdown for maintenance in 10 minutes

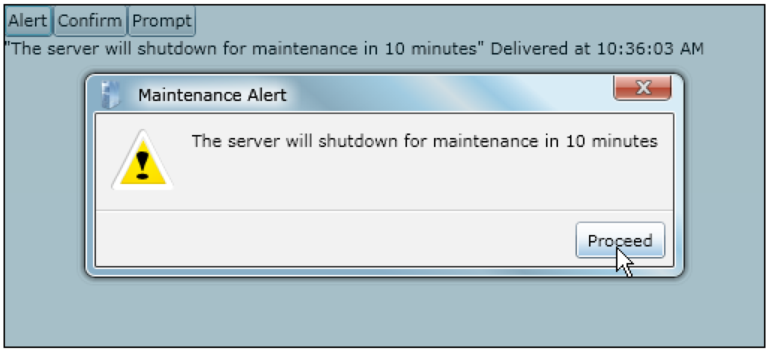
VB

```
RadWindow.Alert("The server will shutdown for maintenance in 10 minutes")
```

C#

```
RadWindow.Alert(
  "The server will shutdown for maintenance in 10 minutes");
```

To get specific control over all aspects of the dialog including the modal background, icon and content of each part of the dialog, pass **DialogParameters**. This example changes the theme of the dialog, adds an icon to the left corner of the header, adds text to the header and changes the content of the OK button and handles the **Opened** and **Closed** events.



"The server will shutdown for maintenance in 10 minutes" Delivered at 10:36:03 AM

Maintenance Alert

The server will shutdown for maintenance in 10 minutes

```
Private Sub btnAlert_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim icon As New Image() With { _
.Source = New BitmapImage(New Uri("VPN.png", UriKind.RelativeOrAbsolute)), _
.Margin = New Thickness(1)}

  Dim message As String = _
"The server will shutdown for maintenance in 10 minutes"

  RadWindow.Alert(New DialogParameters() With { _
.Content = message, _
.Theme = New SummerTheme(), _
.IconContent = icon, _
.OkButtonContent = "Proceed", _
.Header = "Maintenance Alert", _
.Closed = _
New EventHandler(Of WindowClosedEventArgs)(AddressOf OnDialogClosed), _
.Opened = New EventHandler(AddressOf OnDialogOpened)})
End Sub

Private Sub OnDialogOpened( _
ByVal sender As Object, ByVal e As EventArgs)
  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadAlert)
  tbStatus.Text = """" & _
dialog.Content.ToString() & _
"""" & " Delivered at " & _
DateTime.Now.ToLongTimeString()
End Sub

Private Sub OnDialogClosed( _
ByVal sender As Object, ByVal e As WindowClosedEventArgs)
  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadAlert)
  tbStatus.Text = tbStatus.Text _
& Environment.NewLine & _
"User acknowledges at " & _
DateTime.Now.ToLongTimeString()
End Sub
```

C#

```
private void btnAlert_Click(object sender, RoutedEventArgs e)
{
    Image icon = new Image()
    {
        Source = new BitmapImage(new Uri("VPN.png", UriKind.RelativeOrAbsolute)),
        Margin = new Thickness(1)
    };

    string message = "The server will shutdown for maintenance in 10 minutes";

    RadWindow.Alert(new DialogParameters()
    {
        Content = message,
        Theme = new SummerTheme(),
        IconContent = icon,
        OkButtonContent = "Proceed",
        Header = "Maintenance Alert",
        Closed = new EventHandler<WindowClosedEventArgs>(OnDialogClosed),
        Opened = new EventHandler(OnDialogOpened)
    });
}

private void OnDialogOpened(Object sender, EventArgs e)
{
    var dialog = (sender as RadWindow).Content as RadAlert;
    tbStatus.Text =
        "\"" +
        dialog.Content.ToString() +
        "\"" +
        " Delivered at " +
        DateTime.Now.ToLongTimeString();
}

private void OnDialogClosed(object sender, WindowClosedEventArgs e)
{
    var dialog = (sender as RadWindow).Content as RadAlert;
    tbStatus.Text =
        tbStatus.Text +
        Environment.NewLine +
        "User acknowledges at " +
        DateTime.Now.ToLongTimeString();
}
```

Notes

You can also set the DialogParameters **ModalBackground** property. ModalBackground is a Brush that sets the background *behind* the dialog, not the background of the dialog itself.

VB

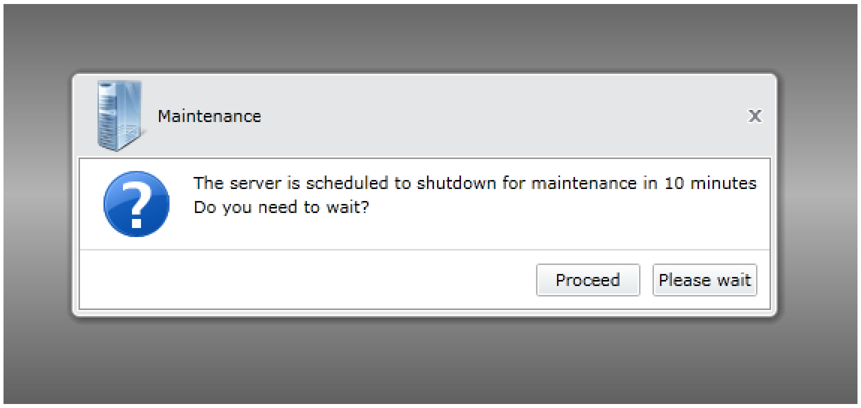
```
RadWindow.Alert(New DialogParameters() With { _
.Content = "The background behind the dialog is light gray", _
.ModalBackground = New SolidColorBrush(Colors.LightGray)})
```

C#

```
RadWindow.Alert(new DialogParameters()
{
   Content = "The background behind the dialog is light gray",
   ModalBackground = new SolidColorBrush(Colors.LightGray)
});
```

24.4.1.2 Confirm

The next step up is the static **Confirm()** method where we can find out if the user clicked the OK or Cancel buttons. The code is very similar to the Alert() example except for the addition of assigning the **CancelButtonContent.** Now the Closed event becomes much more important because the Boolean **DialogResult** is assigned. Notice that DialogResult is a nullable value so you will have to cast it to Boolean.



Maintenance

The server is scheduled to shutdown for maintenance in 10 minutes
Do you need to wait?

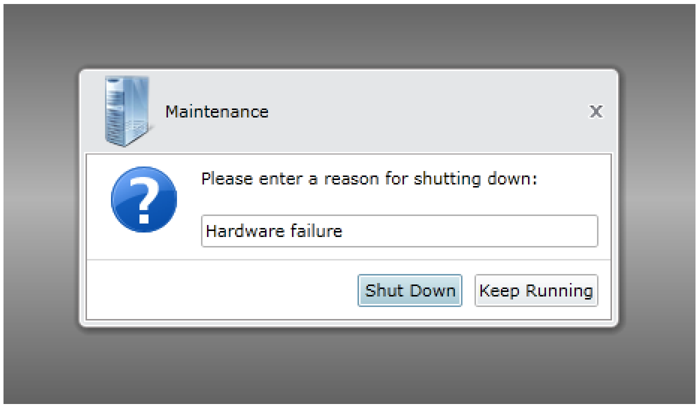
```
Private Sub btnConfirm_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim icon As New Image() With { _
.Source = New BitmapImage(New Uri("VPN.png", UriKind.RelativeOrAbsolute)), _
 .Margin = New Thickness(1)}

  Dim message As String =  _
"The server is scheduled to shutdown for maintenance in 10 minutes" &  _
Environment.NewLine & "Do you need to wait?"

  RadWindow.Confirm(New DialogParameters() With { _
.Content = message,  _
.Theme = New SummerTheme(),  _
.IconContent = icon,  _
.OkButtonContent = "Proceed",  _
.CancelButtonContent = "Please wait",  _
.Header = "Maintenance",  _
.Opened = New EventHandler(AddressOf OnConfirmOpened),  _
.Closed =  _
New EventHandler(Of WindowClosedEventArgs)(AddressOf OnConfirmClosed)})
End Sub

Private Sub OnConfirmOpened( _
ByVal sender As Object, ByVal e As EventArgs)
  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadConfirm)
  tbStatus.Text = """" &  _
dialog.Content.ToString() &  _
"""" & " Delivered at " &  _
DateTime.Now.ToLongTimeString()
End Sub

Private Sub OnConfirmClosed( _
ByVal sender As Object, ByVal e As WindowClosedEventArgs)
  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadConfirm)
  tbStatus.Text =  _
If(CBool(e.DialogResult), "User confirms", "User requests wait")
End Sub
```

C#

```
private void btnConfirm_Click(object sender, RoutedEventArgs e)
{
   Image icon = new Image()
   {
      Source = new BitmapImage(
        new Uri("VPN.png", UriKind.RelativeOrAbsolute)),
      Margin = new Thickness(1)
   };

   string message =
      "The server is scheduled to shutdown for maintenance in 10 minutes" +
      Environment.NewLine +
      "Do you need to wait?";

   RadWindow.Confirm(new DialogParameters()
   {
      Content = message,
      Theme = new SummerTheme(),
      IconContent = icon,
      OkButtonContent = "Proceed",
      CancelButtonContent = "Please wait",
      Header = "Maintenance",
      Opened = new EventHandler(OnConfirmOpened),
      Closed = new EventHandler<WindowClosedEventArgs>(OnConfirmClosed)
   });
}

private void OnConfirmOpened(Object sender, EventArgs e)
{
   var dialog = (sender as RadWindow).Content as RadConfirm;
   tbStatus.Text =
      "\"" +
      dialog.Content.ToString() +
      "\"" +
      " Delivered at " +
      DateTime.Now.ToLongTimeString();
}

private void OnConfirmClosed(object sender, WindowClosedEventArgs e)
{
   var dialog = (sender as RadWindow).Content as RadConfirm;
   tbStatus.Text =
     (bool)e.DialogResult ? "User confirms" : "User requests wait";
}
```

24.4.1.3 Prompt

From Confirm() we move onto the static **Prompt()** method where the user enters some text. You can enter a **DefaultPromptResultValue** that will appear in the text entry box when the dialog first shows. If the DialogResult is true you can retrieve the user entry from the Closed event **PromptResult** argument


Maintenance

Please enter a reason for shutting down:

Hardware failure

Gotcha!

Be aware that PromptResult is ***only*** assigned when the DialogResult is true, i.e. when the user has clicked the "OK" button. This is the observed behavior at the time of this writing. That is, if you assigned the text "Shut Down" to the OkButtonContent, prompted the user to enter a shut down reason, and the user clicked the "Shut Down", i.e. Cancel button, the PromptResult would be empty. You will have to arrange your prompt to work with these semantics.

VB

```
Private Sub btnPrompt_Click( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim icon As New Image() With { _
.Source = New BitmapImage(New Uri("VPN.png", UriKind.RelativeOrAbsolute)),  _
.Margin = New Thickness(1)}

  Dim message As String = "Please enter a reason for shutting down: "

  RadWindow.Prompt(New DialogParameters() With {.Content = message,  _
.Theme = New SummerTheme(),  .IconContent = icon,
.OkButtonContent = "Shut Down", .CancelButtonContent = "Keep Running",  _
.DefaultPromptResultValue = "Monthly Maintenance",  .Header = "Maintenance",  _
.Opened = New EventHandler(AddressOf OnPromptOpened),  _
.Closed = New EventHandler(Of WindowClosedEventArgs)(AddressOf OnPromptClosed)})
End Sub

Private Sub OnPromptOpened(ByVal sender As Object, ByVal e As EventArgs)
  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadPrompt)
  tbStatus.Text = """" & dialog.Content.ToString() &  _
"""" & " Delivered at " & DateTime.Now.ToLongTimeString()
End Sub

Private Sub OnPromptClosed( _
ByVal sender As Object, ByVal e As WindowClosedEventArgs)
  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadPrompt)
  tbStatus.Text = "Wait reason: " & e.PromptResult
End Sub
```

C#

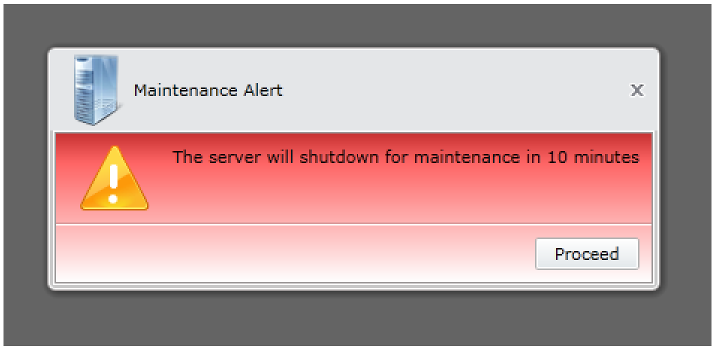
```
private void btnPrompt_Click(object sender, RoutedEventArgs e)
{
   Image icon = new Image()
   {
      Source = new BitmapImage(
         new Uri("VPN.png", UriKind.RelativeOrAbsolute)),
      Margin = new Thickness(1)
   };

   string message = "Please enter a reason for shutting down: ";

   RadWindow.Prompt(new DialogParameters()
   {
      Content = message,
      Theme = new SummerTheme(),
      IconContent = icon,
      OkButtonContent = "Shut Down",
      CancelButtonContent = "Keep Running",
      DefaultPromptResultValue = "Monthly Maintenance",
      Header = "Maintenance",
      Opened = new EventHandler(OnPromptOpened),
      Closed = new EventHandler<WindowClosedEventArgs>(OnPromptClosed)
   });
}

private void OnPromptOpened(Object sender, EventArgs e)
{
   var dialog = (sender as RadWindow).Content as RadPrompt;
   tbStatus.Text = "\"" + dialog.Content.ToString() +
      "\"" + " Delivered at " + DateTime.Now.ToLongTimeString();
}

private void OnPromptClosed(object sender, WindowClosedEventArgs e)
{
   var dialog = (sender as RadWindow).Content as RadPrompt;
   tbStatus.Text = "Wait reason: " + e.PromptResult;
}
```

24.4.2 Brushes

RadWindow surfaces three brushes to allow changing the basic dialog appearance easily:

- **Background**: The background brush of the window.
- **ModalBackground**: The background behind/underneath the window.
- **BorderBackground**: The title area of the window.

The example shows the BorderBackground in a light gray, the ModalBackground in a darker gray and the Background of the window as a linear gradient brush. This particular example uses code from within a RadAlert Opened event handler, and gets a reference to the underlying RadWindow.



The server will shutdown for maintenance in 10 minutes

```
Private Sub OnDialogOpened(ByVal sender As Object, ByVal e As EventArgs)
  Dim backgroundBrush As New LinearGradientBrush() With { _
.StartPoint = New Point(0, 1), .EndPoint = New Point(0, 0)}
  backgroundBrush.GradientStops.Add(New GradientStop() With { _
.Color = Colors.White, .Offset = 0})
  backgroundBrush.GradientStops.Add(New GradientStop() With { _
.Color = Color.FromArgb(255, 255, 100, 100), .Offset = 0.8})
  backgroundBrush.GradientStops.Add(New GradientStop() With { _
.Color = Color.FromArgb(255, 200, 50, 50), .Offset = 1})

  Dim titleBrush As New SolidColorBrush( _
Color.FromArgb(255, 200, 200, 200))
  Dim modalBackgroundBrush As New SolidColorBrush( _
Color.FromArgb(255, 100, 100, 100))

  Dim window As RadWindow = TryCast(sender, RadWindow)
  window.Background = backgroundBrush
  window.BorderBackground = titleBrush
  window.ModalBackground = modalBackgroundBrush

  Dim dialog = TryCast((TryCast(sender, RadWindow)).Content, RadAlert)
  tbStatus.Text = """" & dialog.Content.ToString() & """" & _
" Delivered at " & DateTime.Now.ToLongTimeString()
End Sub
```

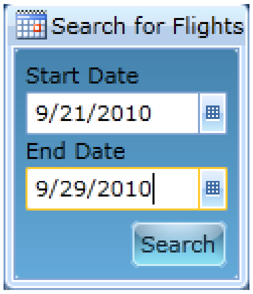
```
private void OnDialogOpened(Object sender, EventArgs e)
{
   LinearGradientBrush backgroundBrush = new LinearGradientBrush()
   {
      StartPoint = new Point(0, 1),
      EndPoint = new Point(0, 0)
   };
   backgroundBrush.GradientStops.Add(new GradientStop()
   {
      Color = Colors.White, Offset = 0
   });
   backgroundBrush.GradientStops.Add(new GradientStop()
   {
      Color = Color.FromArgb(255, 255, 100, 100),  Offset = 0.8
   });
   backgroundBrush.GradientStops.Add(new GradientStop()
   {
      Color = Color.FromArgb(255, 200, 50, 50), Offset = 1
   });

   SolidColorBrush titleBrush =
      new SolidColorBrush(Color.FromArgb(255, 200, 200, 200));
   SolidColorBrush modalBackgroundBrush =
      new SolidColorBrush(Color.FromArgb(255, 100, 100, 100));

   RadWindow window = sender as RadWindow;
   window.Background = backgroundBrush;
   window.BorderBackground = titleBrush;
   window.ModalBackground = modalBackgroundBrush;

   var dialog = (sender as RadWindow).Content as RadAlert;
   tbStatus.Text = "\"" + dialog.Content.ToString() +
      "\"" + " Delivered at " + DateTime.Now.ToLongTimeString();
}
```

24.4.3 Events

You can monitor the state of your windows using events surfaced by RadWindow. We already worked briefly with the Opened and Closed events. The **Activated** event fires when a window is brought to the front. You can also use:

- **LocationChanged**: Fires when the window Top or Left changes.
- **WindowStateChanged**: Fires when WindowState changes between Normal, Minimized and Maximized.
- **PreviewClosed**: Fires just before the window closes.

The PreviewClosed event allows you to prevent a window from closing. In the example below, the Click event for the "Search" button calls the window Close() method. The PreviewClosed event fires and Cancel is set "True" if there is no value in either date picker.



Search for Flights

Start Date

9/21/2010

End Date

9/29/2010

```
Private Sub RadWindow_PreviewClosed(sender As Object, e As WindowPreviewClosedEventArgs)
        e.Cancel = Not dpStart.SelectedDate.HasValue OrElse Not dpEnd.SelectedDate.HasValue
End Sub

Private Sub btnSearch_Click(sender As Object, e As RoutedEventArgs)
        Me.Close()
End Sub
```

```
private void RadWindow_PreviewClosed(object sender, WindowPreviewClosedEventArgs e)
{
    e.Cancel =
       !dpStart.SelectedDate.HasValue
       || !dpEnd.SelectedDate.HasValue;
}

private void btnSearch_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}
```

24.4.4 Window State and Z-Order

Window State

The **CanClose** property when false removes the close button from the window but still allows the window to be closed programmatically. Likewise, the **CanMove** can be set false to prevent the window from being dragged. **ResizeMode** can be set to **CanResize** (the default), **CanMinimize**, **CanMaximize** or **NoResize** .

Z-Order

From the Forums...

Question: "How do I handle relative Z-Index positioning of windows and other controls that popup?"

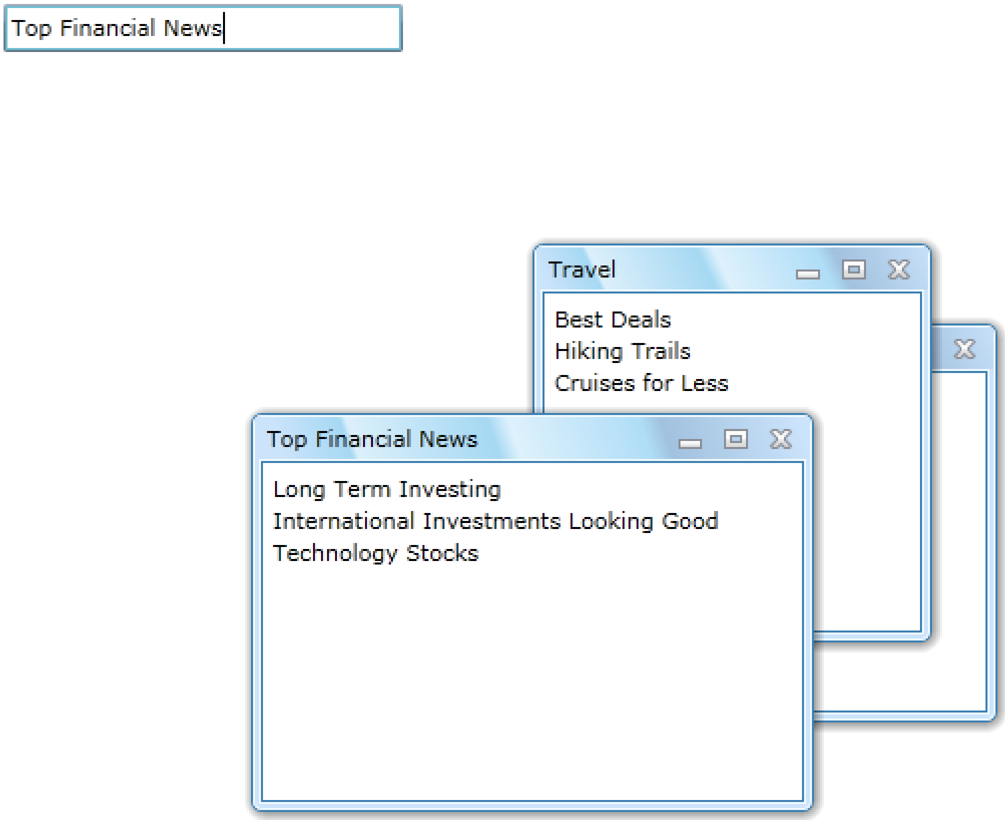
Answer: "We introduced a PopupManager that handles Z-Index on all popups (RadWindow, RadWindows with TopMost=true, and RadPopup - used in combobox, menuitem, datepicker, timepicker).The PopupManager exposes four zones: Window, TopWindow, Popup and DockWindow (for future use). If you want your popup (not RadPopup) to show on top you can set its Child Z-Index to be bigger then 300,000 (this is the starting Z-Index for Popup zone)."

PopupManager has static methods for setting Z-Index:

- **DockWindowStartingZIndex**: starting index for RadDockWindow controls. Should be bigger than Popups ZIndex and smaller then 1000000.
- **PopupStartingZIndex**: starting index for Popup controls. Should be bigger then TopWindows ZIndex and smaller than DockWinows ZIndex.
- **TopWindowStartingZIndex**: starting index for TopMost RadWindow controls. Should be bigger then Windows ZIndex and smaller than Popups ZIndex.
- **WindowStartingZIndex**: starting index for RadWindow controls. Should be bigger then zero and smaller then TopWindows ZIndex.

24.5 Binding

You can bind the properties of RadWindow in a similar manner to other Silverlight controls. The example below binds the **Header** of each window to the "Title" property of a "NewsCategory" object. Each RadWindow Activated event has a handler that assigns the DataContext of the top-most window to a TextBox and also binds the "Title" property. Changes to the text box show up right away in the top window. Bringing other windows to the top changes the title shown in the text box. The underlying NewsDataSource object is an ObservableCollection<NewsCategory> where NewsCategory is an INotifyPropertyChanged implementation with two properties: a "Title" and a List<string> of "Articles".



Top Financial News

Travel

Best Deals
Hiking Trails
Cruises for Less

Top Financial News

Long Term Investing
International Investments Looking Good
Technology Stocks

```
Private Sub UserControl_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim dataSource As New NewsDataSource()
  For Each category As NewsCategory In dataSource
    Dim window As New RadWindow()
    window.DataContext = category
    window.SetBinding(RadWindow.HeaderProperty,  _
New Binding("Title") With {.Mode = BindingMode.TwoWay})
    Dim listBox As New Telerik.Windows.Controls.ListBox() With { _
.Margin = New Thickness(5)}
    listBox.ItemsSource = category.Articles
    window.Content = listBox
    window.WindowStartupLocation = WindowStartupLocation.CenterScreen
    window.SetValue(RadWindow.WidthProperty, 200R)
    window.SetValue(RadWindow.HeightProperty, 200R)
    AddHandler window.Activated, AddressOf window_Activated
    window.Show()
  Next category
End Sub

Private Sub window_Activated(ByVal sender As Object, ByVal e As EventArgs)
  tbTitle.DataContext = (TryCast(sender, RadWindow)).DataContext
  tbTitle.SetBinding(TextBox.TextProperty, New Binding("Title") With { _
.Mode = BindingMode.TwoWay})
End Sub
```

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   NewsDataSource dataSource = new NewsDataSource();
   foreach (NewsCategory category in dataSource)
   {
      RadWindow window = new RadWindow();
      window.DataContext = category;
      window.SetBinding(RadWindow.HeaderProperty,
         new Binding("Title") { Mode = BindingMode.TwoWay });
      Telerik.Windows.Controls.ListBox listBox =
         new Telerik.Windows.Controls.ListBox()
         {
            Margin = new Thickness(5)
         };
      listBox.ItemsSource = category.Articles;
      window.Content = listBox;
      window.WindowStartupLocation = WindowStartupLocation.CenterScreen;
      window.SetValue(RadWindow.WidthProperty, 200d);
      window.SetValue(RadWindow.HeightProperty, 200d);
      window.Activated += new EventHandler(window_Activated);
      window.Show();
   }
}

void window_Activated(object sender, EventArgs e)
{
   tbTitle.DataContext = (sender as RadWindow).DataContext;
   tbTitle.SetBinding(TextBox.TextProperty,
      new Binding("Title") { Mode = BindingMode.TwoWay });
}
```

Gotcha!

Are changes in the text box not showing up in the window? Be sure that you have implemented INotifyPropertyChanged in the bound object and that the collection is ObservableCollection<> and not List<>.

24.6 Customization

Walk Through

In this example we will customize the RadWindow control border and background.

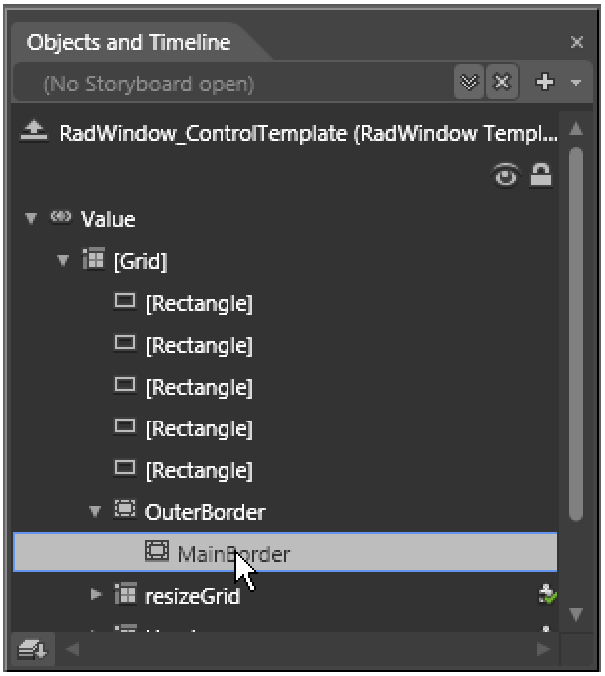
Project Setup

1) Run Expression Blend.
2) From the **File** menu select **New Project**. *Note: If you have an existing solution open, right-click the solution and select Add New Project... from the context menu instead.*
3) In the New Project dialog, select "Silverlight" from "Project types" and "Silverlight Application" from the right-most list. Enter a unique name for the project and click **OK**.

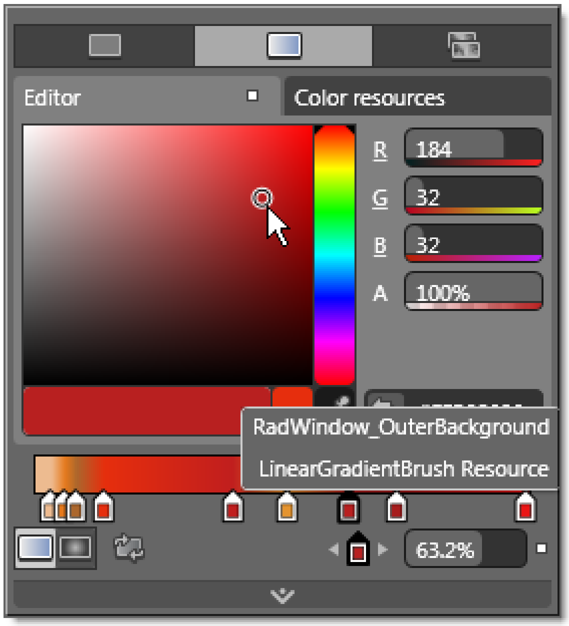
Edit the Page in Expression Blend

1) MainPage.xaml should already be open for editing. If not, locate MainPage.xaml in the Projects pane and double-click to open the page.
2) In the Projects pane, right-click the References node and select **Add Reference...** from the context menu.
3) Add a reference to the **Telerik.Windows.Controls.dll** and **Telerik.Windows.Controls.Navigation.dll** assemblies.
4) From the Project menu select **Build Project**.
5) Add a **RadWindow** to the page.
   a) Open the Assets pane.
   b) On the left side of the Assets pane is a tree view. Locate and select the "Controls" node.
   c) In the Assets pane, just above the tree view is the Assets Find entry text box.
   d) Type the first few characters of "RadWindow" into the Assets Find entry text box. A list of all matching controls will show to the right of the tree view.
   e) Locate the RadWindow control and drag it onto the MainPage.xaml Artboard.
6) In the Objects and Timeline pane, double-click "RadWindow" in the tree view. Enter a new name "winInfo".
7) Optionally, you can add a TextBox or other controls to the content area of the window.
8) Right-click the RadWindow and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "ScovilleWindowStyle". Click **OK** to create the style resource and close the dialog.

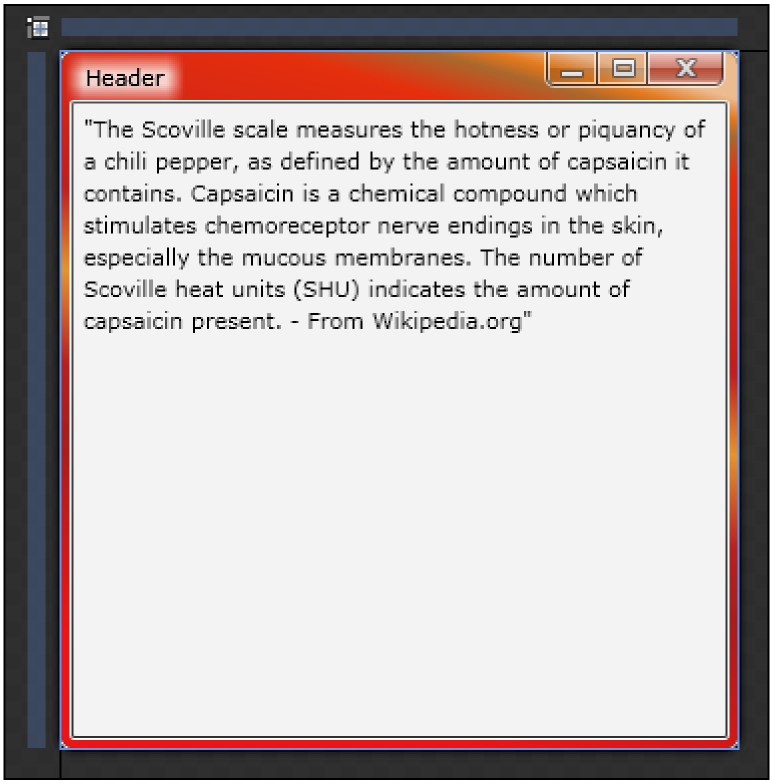
9) In the Objects and Timeline pane, locate the "MainBorder" node and click it.



10)In the Resources pane, locate the "RadWindow_Outerbackground" resource and click the drop down arrow to edit the brush. Click each of the gradient stop indicators, then use the color editor above the gradient stop indicators to change the color for each.



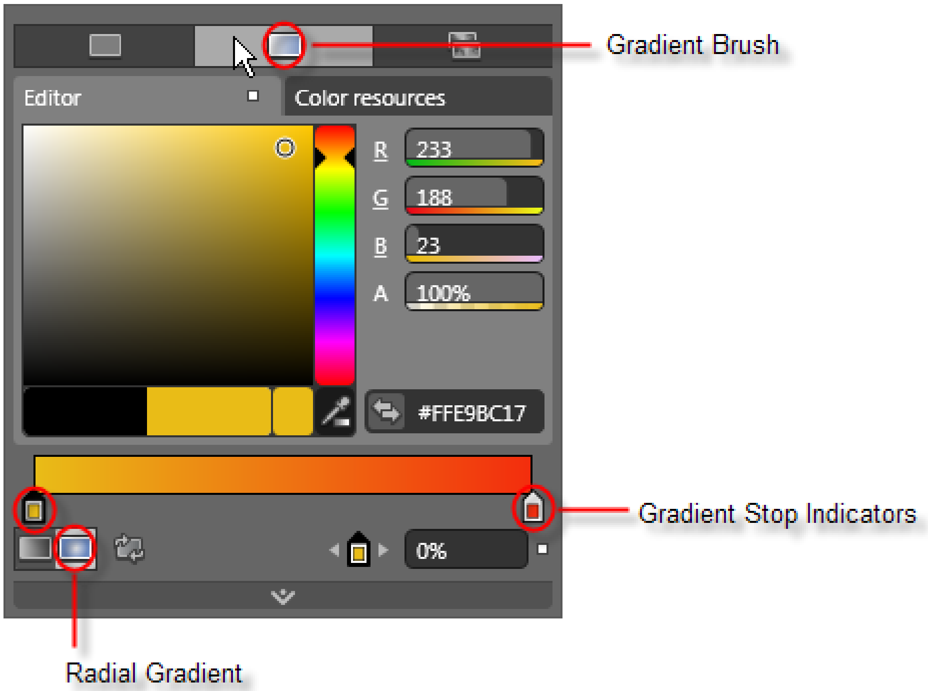
The window border may look something like the screenshot below. The actual colors chosen will determine the final visual makeup of the outer window border area.



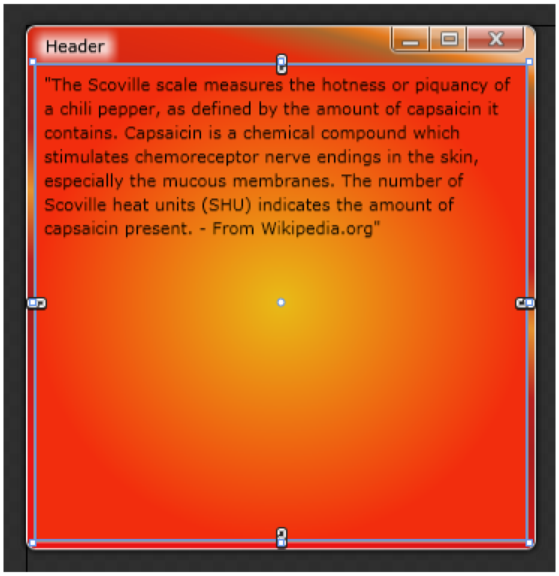
"The Scoville scale measures the hotness or piquancy of a chili pepper, as defined by the amount of capsaicin it contains. Capsaicin is a chemical compound which stimulates chemoreceptor nerve endings in the skin, especially the mucous membranes. The number of Scoville heat units (SHU) indicates the amount of capsaicin present. - From Wikipedia.org"

11)Also in the Resources pane, locate the "RadWindow_InnerBackground" resource and click the drop down arrow to edit the brush.

12)Click the "Gradient Brush" button at the top of the editor. Click on both gradient stop indicators and select a yellow and a red color, respectively. Click the "Radial Gradient" button at the bottom of the editor.



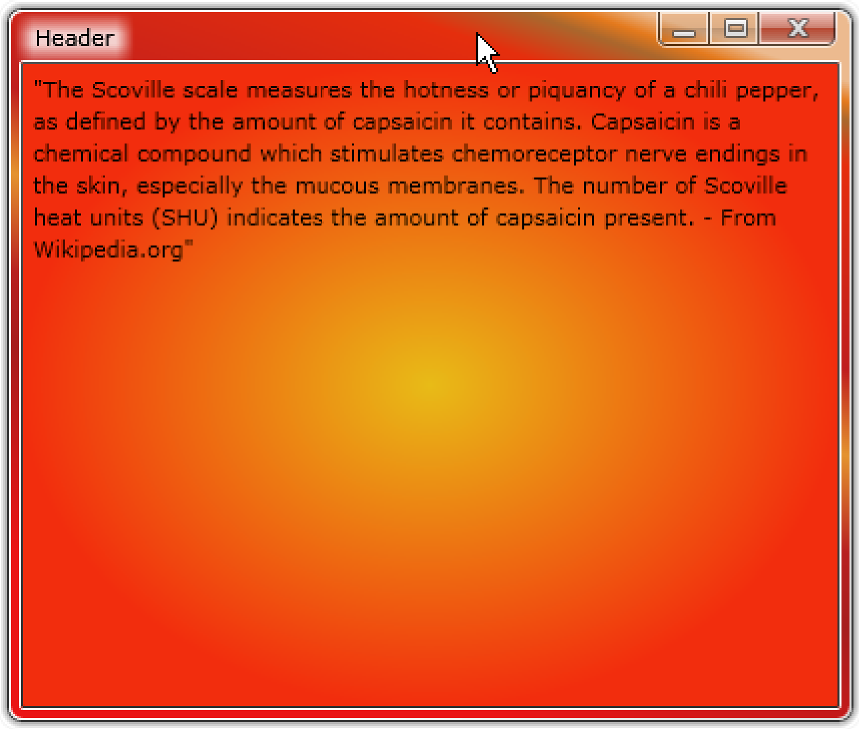
In the Artboard, the window may look something like this, depending on your color choices.



"The Scoville scale measures the hotness or piquancy of a chili pepper, as defined by the amount of capsaicin it contains. Capsaicin is a chemical compound which stimulates chemoreceptor nerve endings in the skin, especially the mucous membranes. The number of Scoville heat units (SHU) indicates the amount of capsaicin present. - From Wikipedia.org"

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



"The Scoville scale measures the hotness or piquancy of a chili pepper, as defined by the amount of capsaicin it contains. Capsaicin is a chemical compound which stimulates chemoreceptor nerve endings in the skin, especially the mucous membranes. The number of Scoville heat units (SHU) indicates the amount of capsaicin present. - From Wikipedia.org"

24.7 Wrap Up

In this chapter you learned how to build RadWindows with free-form content and control them as a group with RadWindowManager. You worked with the predefined dialogs Alert, Confirm and Prompt. You learned how to control window appearance, both for simple modifications like background and title colors as well as customization using Expression Blend. You worked with events that react to windows opening, closing, state changes and moving. You learned how to alter the initial state of a window. You learned how to bind data to a control.

Part XXV

HTMLPlaceholder

25 HTMLPlaceholder

25.1 Objectives

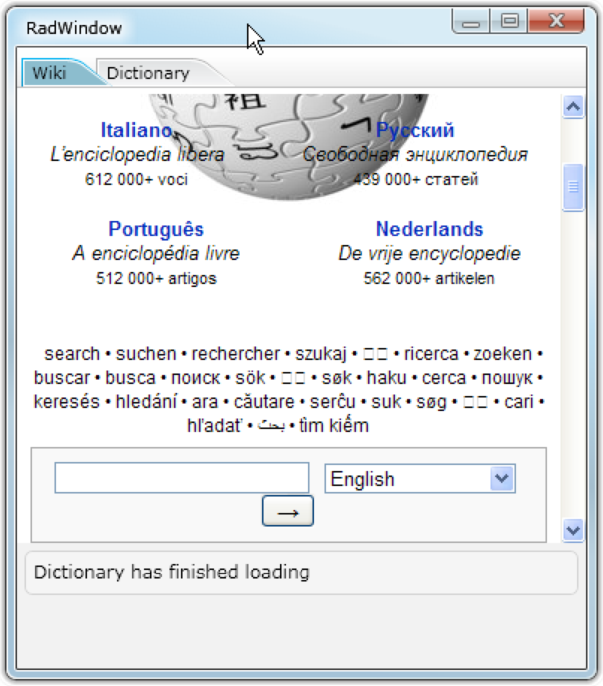
In this chapter you will learn how RadHtmlPlaceholder is used to host HTML content in Silverlight applications. First you will add custom HTML content to a place holder to see how it renders, then you will point to an external web page. You will learn how to host place holders in the page, in other controls and within RadWindows. You will learn how to respond to events when the content page is loaded. You will learn how to interact between Javascript and managed code behind. Finally, you will learn how to bind a series of place holder controls within an ItemsControl to custom objects.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\HtmlPlaceholder\HtmlPlaceholder.sln.

25.2 Overview

RadHtmlPlaceholder is a powerful tool that lets you blend standard HTML, ASP.NET and Silverlight elements. You can load content by assigning custom HTML or by loading external web pages. RadHtmlPlaceholder can be added as content to standard Silverlight and RadControls including tab controls and windows.


RadWindow
Wiki
Dictionary
Italiano
L'enciclopedia libera
612 000+ voci
Русский
Свободная энциклопедия
439 000+ статей
Português
A enciclopédia livre
512 000+ artigos
Nederlands
De vrije encyclopedie
562 000+ artikelen
search • suchen • rechercher • szukaj • □□ • ricerca • zoeken • buscar • busca • поиск • sök • □□ • søk • haku • cerca • пошук • keresés • hledání • ara • căutare • serĉu • suk • søg • □□ • cari • hľadať • بحث • tìm kiếm
English
→

Dictionary has finished loading

25.3 Getting Started

In this walk through you will use two HTML place holder controls, one that renders a literal HTML string, and a second place holder that points to an external URL. During this walk through you will learn about the special conditions that govern use of the RadHtmlPlaceholder control.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:
   a) **Telerik.Windows.Controls**

XAML Editing

1) Open MainPage.xaml for editing.
2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags.

```
<StackPanel>
  <!--place holder that uses literal html-->

  <!--place holder that uses external site-->
</StackPanel>
```

3) Drag a **RadHtmlPlaceholder** from the Toolbox to a point just under the comment "<!--place holder that uses literal html-->". Name the place holder "phHtml" so we can access it later in code.

```
<!--uses literal html-->
<telerik:RadHtmlPlaceholder x:Name="phHtml" />
```

4) Drag a **RadHtmlPlaceholder** from the Toolbox to a point just under the comment "<!--place holder that uses external site-->". Name the place holder "phUrl". Set the **SourceUrl** property to "http://www.telerik.com".

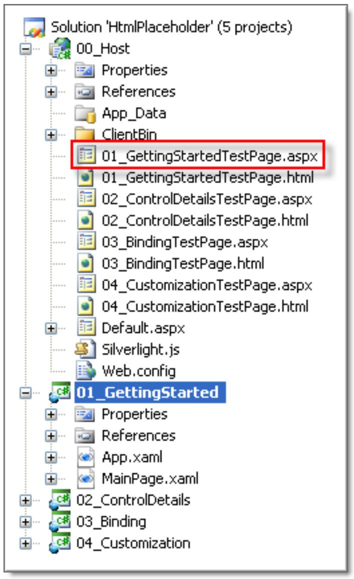
```
<!--uses external site-->
<telerik:RadHtmlPlaceholder x:Name="phUrl"
     SourceUrl="http://www.telerik.com" />
```

5) In the Solution Explorer, navigate to the host ASP.NET application and locate the startup page for the project. Open the startup page for editing. Add a "windowless" parameter to the SilverlightControlHost and set it to "true" (Look at the last parameter in the example below).

```
<body>
  <form id="form1" runat="server" style="height:100%">
  <div id="silverlightControlHost">
    <object data="data:application/x-silverlight-2,"
        type="application/x-silverlight-2"
        width="100%" height="100%">
    <param name="source" value="ClientBin/01_GettingStarted.xap"/>
    <param name="onError" value="onSilverlightError" />
    <param name="background" value="white" />
    <param name="minRuntimeVersion" value="3.0.40624.0" />
    <param name="autoUpgrade" value="true" />
    <param name="windowless" value="true" />
    . . .
```

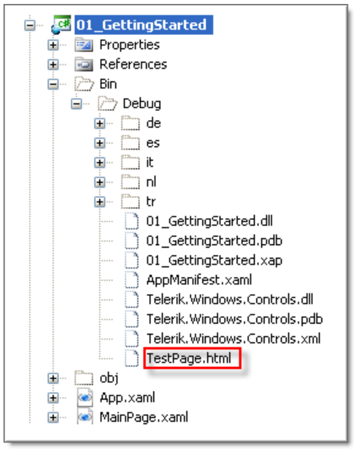
Gotcha!

"RadHtmlPlaceholder requires the "Windowless" parameter of the Silverlight application to be True." This message is probably the most common question raised in the forums for this control. The message appears if you don't set the "windowless" parameter as shown above.



```
Solution 'HtmlPlaceholder' (5 projects)
  00_Host
    Properties
    References
    App_Data
    ClientBin
    01_GettingStartedTestPage.aspx
    01_GettingStartedTestPage.html
    02_ControlDetailsTestPage.aspx
    02_ControlDetailsTestPage.html
    03_BindingTestPage.aspx
    03_BindingTestPage.html
    04_CustomizationTestPage.aspx
    04_CustomizationTestPage.html
    Default.aspx
    Silverlight.js
    Web.config
  01_GettingStarted
    Properties
    References
    App.xaml
    MainPage.xaml
  02_ControlDetails
  03_Binding
  04_Customization
```

Here's the gotcha: the key is to set the "windowless" parameter in the silverlight control ***in the page that hosts the Silverlight application***. The hosting page may not always be consistent. Consider the solution in this screenshot that has a host application and several Silverlight projects. If you set the "windowless" parameter in the "01_GettingStartedTestPage. aspx" of the host project, set the host project to be the startup project and set that page to be the startup page, the error message will not appear.



```
01_GettingStarted
  Properties
  References
  Bin
    Debug
      de
      es
      it
      nl
      tr
      01_GettingStarted.dll
      01_GettingStarted.pdb
      01_GettingStarted.xap
      AppManifest.xaml
      Telerik.Windows.Controls.dll
      Telerik.Windows.Controls.pdb
      Telerik.Windows.Controls.xml
      TestPage.html
  obj
  App.xaml
  MainPage.xaml
```

But if you set the "01_GettingStarted" project to be the startup project, a test page will be automatically generated that will not contain the parameter. You can edit the TestPage.html and add the "windowless" parameter there to prevent the error, but the page will be regenerated.

The main point is to be aware of the origin of the hosting page and make sure the windowless parameter exists there. If you're unsure of the host page origin, check the URL in the browser when project runs.

Code Behind

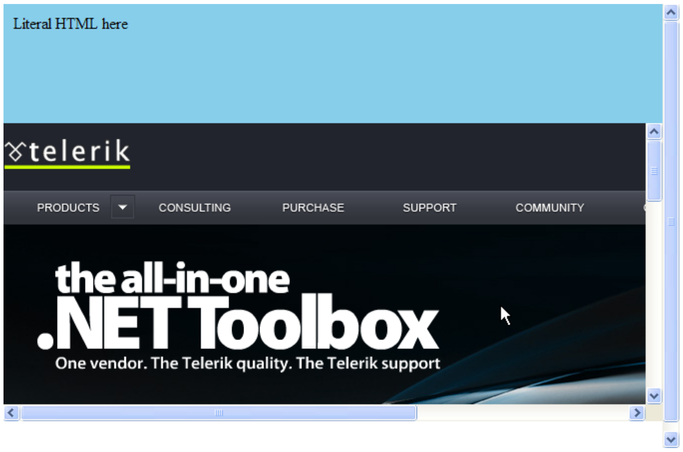
1) Add the code below to the UserControl constructor that assigns the **HtmlSource** property. Optionally, you can place some other arbitrary HTML here.

```
Public Sub New()
  InitializeComponent()
  phHtml.HtmlSource = "<div style=" & _
    """width:100%;height:100px;padding:10px;background-color:SkyBlue;""" & _
    ">Literal HTML here</div>"
End Sub
```

```
public MainPage()
{
   InitializeComponent();
   phHtml.HtmlSource =
       "<div style=" +
       "\"width:100%;height:100px;padding:10px;background-color:SkyBlue;\"" +
       ">Literal HTML here</div>";
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.


Literal HTML here

Ideas for Extending This Example

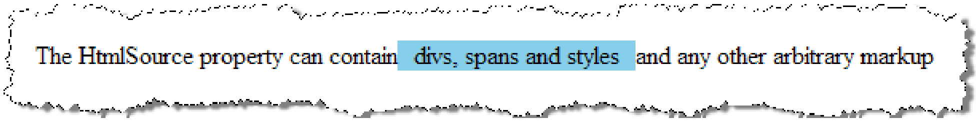
- Change the HtmlSource of the html place holder control.
- Change the SourceUrl of the html place holder control.

25.4 Control Details

25.4.1 Loading Content

In a Silverlight Page

As you saw in the "Getting Started" walk through, you can assign markup directly to the **HtmlSource** property. This might be useful if you have HTML content stored in a database returned from a service or saved to isolated storage locally. The example below shows that as expected, divs, spans and styles can be placed within the HTML.


The HtmlSource property can contain divs, spans and styles and any other arbitrary markup

VB

```
Const tag As String = "<span style=" & _
"""width:100%height:100px;padding:10px;background-color:SkyBlue;""">" & _
"{0}</span>"

phHtml.HtmlSource = _
"The HtmlSource property can contain" & _
String.Format(tag, "divs, spans and styles") & _
"and any other arbitrary markup"
```

C#

```
const string tag = "<span style=" +
    "\"width:100%height:100px;padding:10px;background-color:SkyBlue;\">" +
    "{0}</span>";

phHtml.HtmlSource = "The HtmlSource property can contain" +
   string.Format(tag, "divs, spans and styles") +
    "and any other arbitrary markup";
```

The alternative is to assign the **SourceUrl** property. SourceUrl takes a Uri object that can point to an external site.

VB

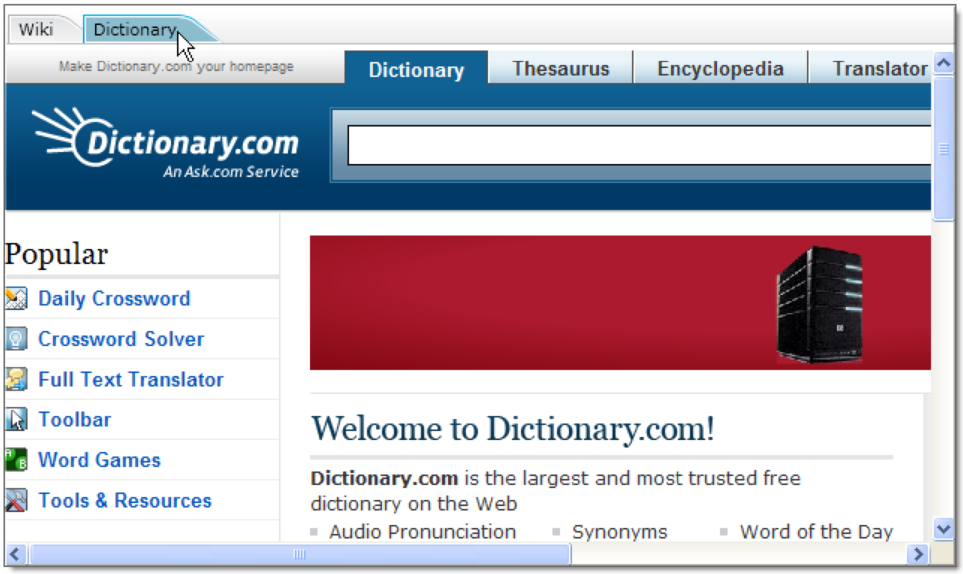
```
phHtml.SourceUrl = New System.Uri("http://www.telerik.com")
```

C#

```
phHtml.SourceUrl = new System.Uri("http://www.telerik.com");
```

In RadTabControl

RadHtmlPlaceholder can be used as content for other Silverlight and RadControls, such as a RadTabItem. In the example below, the place holder is included in the Content portion of the RadTabItem element.



Welcome to Dictionary.com!

Dictionary.com is the largest and most trusted free dictionary on the Web

```
<telerik:RadTabControl>
  <telerik:RadTabItem Header="Wiki">
    <telerik:RadHtmlPlaceholder
        SourceUrl="http://www.wikipedia.org" />
  </telerik:RadTabItem>
  <telerik:RadTabItem Header="Dictionary">
    <telerik:RadHtmlPlaceholder
        SourceUrl="http://www.dictionary.com" />
  </telerik:RadTabItem>
</telerik:RadTabControl>
```

In RadWindow

When RadHtmlPlaceholder is placed in a RadWindow, and the window is dragged, the RadHtmlPlaceholder does not redraw quickly enough, showing an unpleasant artifact.

The solution is subscribe to the **LocationChanged** event of the window and call the RadHtmlPlaceholder **InvalidateArrange()** method. The XAML for the sample below simply wraps the previous RadTabControl example in a RadWindow, names it "winHtml" and hooks up a LocationChanged event handler. Notice that the **UrlLoaded** event is used to notify us that all content is available in particular place holder.

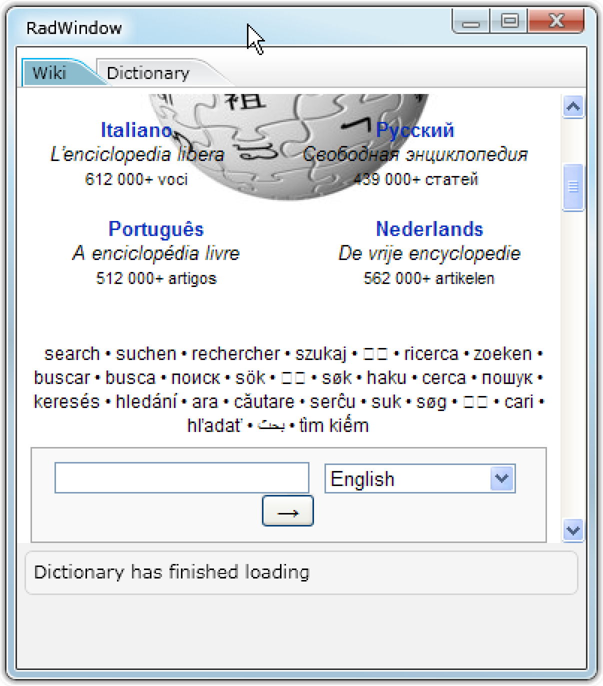
```
<telerik:RadWindow x:Name="winHtml"
  LocationChanged="winHtml_LocationChanged">
  <StackPanel>
    <telerik:RadTabControl>
      <telerik:RadTabItem Header="Wiki">
        <telerik:RadHtmlPlaceholder Tag="Wiki"
            SourceUrl="http://www.wikipedia.org"
            UrlLoaded="RadHtmlPlaceholder_UrlLoaded" />
      </telerik:RadTabItem>
      <telerik:RadTabItem Header="Dictionary">
        <telerik:RadHtmlPlaceholder Tag="Dictionary"
            SourceUrl="http://www.dictionary.com"
            UrlLoaded="RadHtmlPlaceholder_UrlLoaded" />
      </telerik:RadTabItem>
    </telerik:RadTabControl>
    <Border Background="WhiteSmoke" CornerRadius="5"
        Margin="5" Padding="5" BorderThickness="1"
        BorderBrush="LightGray">
      <TextBlock x:Name="tbStatus" FontSize="12" />
    </Border>
  </StackPanel>
</telerik:RadWindow>
```

In this example, the code-behind has to deal with multiple place holders, so a little extra code is required. We first need to get a RadHtmlPlaceholder reference before calling InvalidateArrange(). The handler gets a reference to the RadWindow, then drills down to its content (a StackPanel), then uses the ChildrenOfType() extension method to get the RadTabControl. Then the event handler finally gets the RadHtmlPlaceholder from the RadTabControl SelectedContent property. Your coding requirements may or may not be as involved, but the key point is to get a reference to the place holder and call the InvalidateArrange() method.

```
Private Sub winHtml_LocationChanged( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim stackPanel As StackPanel = _
TryCast((TryCast(sender, RadWindow)).Content, StackPanel)
  Dim tabControl As RadTabControl = _
stackPanel.ChildrenOfType(Of RadTabControl)().FirstOrDefault()
  If tabControl IsNot Nothing Then
    Dim placeHolder As RadHtmlPlaceholder = _
TryCast(tabControl.SelectedContent, RadHtmlPlaceholder)
    placeHolder.InvalidateArrange()
  End If
End Sub
```

```
private void winHtml_LocationChanged(object sender, RoutedEventArgs e)
{
  StackPanel stackPanel = (sender as RadWindow).Content as StackPanel;
  RadTabControl tabControl =
   stackPanel.ChildrenOfType<RadTabControl>().FirstOrDefault();
  if (tabControl != null)
  {
    RadHtmlPlaceholder placeHolder = tabControl.SelectedContent as RadHtmlPlaceholder;
    placeHolder.InvalidateArrange();
  }
}
```

Now when you move the window, the contents appear to track correctly.


RadWindow
Wiki
Dictionary
Italiano
L'enciclopedia libera
612 000+ voci
Русский
Свободная энциклопедия
439 000+ статей
Português
A enciclopédia livre
512 000+ artigos
Nederlands
De vrije encyclopedie
562 000+ artikelen
search • suchen • rechercher • szukaj • □□ • ricerca • zoeken • buscar • busca • поиск • sök • □□ • søk • haku • cerca • пошук • keresés • hledání • ara • căutare • serĉu • suk • søg • □□ • cari • hľadať • بحث • tìm kiếm
English
→

Dictionary has finished loading

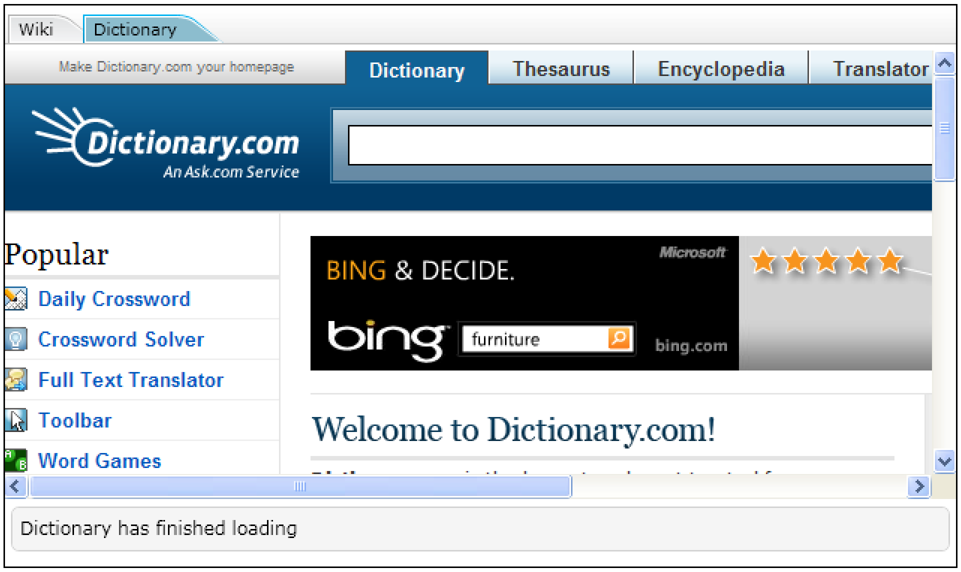
From the Forums...

Question: The RadHtmlPlaceholder is always on top of everything. What problems might this cause and can this be changed?

Answer: This question has multiple permutations, such as why doesn't the place holder follow the window, or why doesn't the place holder handle z-order relative to other Silverlight controls as I expect, or why doesn't a container for a place holder resize as expected. The cause for all of these behaviors has the same root. The content of the RadHtmlPlaceholder is rendered in an IFrame element, which floats above everything else on the page. If the RadHtmlPlaceholder content overlaps with another element, that element will always be covered. The IFrame can also prevent Silverlight controls that it covers from receiving mouse events.

25.4.2 Events

RadHtmlPlaceholder has a single event, UrlLoaded that fires, not when the page first shows, but when all the content has loaded to the page. The example below has two RadHtmlPlaceholder controls hooked up to the same UrlLoaded event. Sender is the RadHtmlPlaceholder that triggered the event. In this example the title is placed in the Tag property of the RadHtmlPlaceholder.



Dictionary has finished loading

```
<StackPanel>
  <telerik:RadTabControl>
    <telerik:RadTabItem Header="Wiki">
      <telerik:RadHtmlPlaceholder Tag="Wiki"
          SourceUrl="http://www.wikipedia.org"
          UrlLoaded="RadHtmlPlaceholder_UrlLoaded" />
    </telerik:RadTabItem>
    <telerik:RadTabItem Header="Dictionary">
      <telerik:RadHtmlPlaceholder Tag="Dictionary"
          SourceUrl="http://www.dictionary.com"
          UrlLoaded="RadHtmlPlaceholder_UrlLoaded" />
    </telerik:RadTabItem>
  </telerik:RadTabControl>
  <Border Background="WhiteSmoke" CornerRadius="5" Margin="5" Padding="5"
      BorderThickness="1" BorderBrush="LightGray">
    <TextBlock x:Name="tbStatus" FontSize="12" />
  </Border>
</StackPanel>
```

VB

```
Private Sub RadHtmlPlaceholder_UrlLoaded( _
ByVal sender As Object, ByVal e As EventArgs)
  Dim title As String = _
(TryCast(sender, RadHtmlPlaceholder)).Tag.ToString()
  tbStatus.Text = title & " has finished loading"
End Sub
```

C#

```
private void RadHtmlPlaceholder_UrlLoaded(object sender, EventArgs e)
{
   string title = (sender as RadHtmlPlaceholder).Tag.ToString();
   tbStatus.Text =  title + " has finished loading";
}
```

25.4.3 Sizing and Positioning

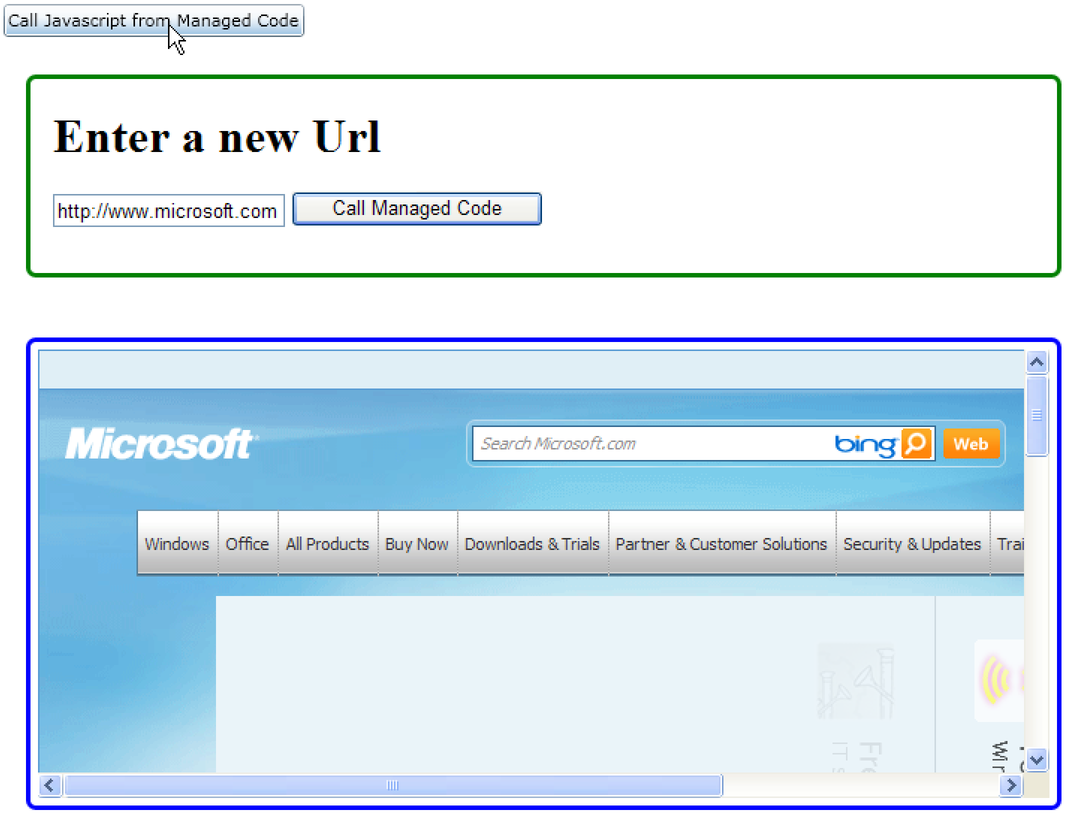
Use standard Silverlight properties to position and size the place holder control: Height, Width, MaxWidth, MaxHeight, MinWidth, MinHeight, VerticalAlignment and HorizontalAlignment. Scrollbars are added automatically for content that does not fit the control.

25.4.4 Interaction with the Page

You can make calls in both directions between the page that hosts the Silverlight plugin and the Silverlight code itself. From the Silverlight client you can trigger Javascript functions on a page from managed code. Javascript functions can also *call* managed code in your Silverlight application. This is especially powerful because you get the full might of managed code accessed from Javascript, without a trip to the server.

The example explained below has a Silverlight button "Call Javascript from Managed Code" at the top of the page. Below that are two place holder controls. The top place holder, bordered in green in the screenshot below, loads a local page "MyPage.htm" stored with the host ASP.NET application. It contains the banner "Enter a new Url", a text input element and a button input element "Call Managed Code".

When the "Call Javascript from Managed Code" button is clicked, code from inside the Silverlight application calls a Javascript function that sets the text input element. When the "Call Managed Code" button is clicked, a Javascript function is fired that gets a reference to a "Scriptable" method in the Silverlight application, that in turn sets the URL of the second place holder control (bordered in blue). Using these two basic techniques you can interact between the page, the managed code and ultimately, between HTML elements in multiple place holders.



Enter a new Url

http://www.microsoft.com

25.4.4.1 Calling Javascript from Managed Code

The general steps to call Javascript from managed code in your Silverlight client application are:

1. Get the **HtmlPresenter** property from the place holder control and index into the first element of the **Children** collection. This will be your reference to the "iframe" that contains the HTML displayed in the place holder.
2. Set the Id attribute of the iframe element so we can get at it easily from a Javascript statement.
3. Use **HtmlPage.Window.Eval()** from managed code to call Javascript.

The "MyPage.html" file lives at the root of the host ASP.NET application. For the managed-code-to-Javascript piece, we're interested in the setUrlText() Javascript method and the "tbUrl" input element. Walk through this markup briefly before reading about how to call the Javscript function from managed code.

```
<html>
<head>
   <title>Test Page</title>

   <script type="text/javascript">
      function setUrlText(text) {
         var tbUrl = document.getElementById("tbUrl");
         if (tbUrl) {
            tbUrl.value = text;
         }
      }

      //. . .
   </script>

</head>
<body>
   <h1>
      Enter a new Url</h1>
   <input id="tbUrl" />
   . . .
</body>
</html>
```

The example below calls a custom Javascript method added to the page called "setUrlText()".

VB

```
' Get the IFrame from the HtmlPresenter
Dim iframe As HtmlElement = _
CType(phUrl.HtmlPresenter.Children(0), HtmlElement)
' Set an ID to the IFrame so that can be used when calling the javascript
iframe.Id = "myIFrame"
' Format the code to be executed, i.e. call javascript setUrlText()
' and pass the place holder source url
Dim format As String = _
"document.getElementById('myIFrame').contentWindow.{0}('{1}');"
Dim code As String = _
String.Format(format, "setUrlText", phTarget.SourceUrl.OriginalString)
' call the javascript code
HtmlPage.Window.Eval(code)
```

C#

```
// Get the IFrame from the HtmlPresenter
HtmlElement iframe = (HtmlElement)phUrl.HtmlPresenter.Children[0];
// Set an ID to the IFrame so that can be used when calling the javascript
iframe.Id = "myIFrame";
// Format the code to be executed, i.e. call javascript setUrlText()
// and pass the place holder source url
string format =
   "document.getElementById('myIFrame').contentWindow.{0}('{1}');";
string code =
   String.Format(format, "setUrlText", phTarget.SourceUrl.OriginalString);
// call the javascript code
HtmlPage.Window.Eval(code);
```

Notes

The "iframe" variable is an **HtmlElement** that represents an HTML element in the Document Object Model (DOM) of the page. The HtmlElement class lets you get at the parent HtmlElement, CssClass, Id, tag name, and has a series of methods that match counterparts you would normally call from Javascript directly. SetAttribute() and SetStyleAttribute() methods are particularly useful for setting all the properties of an element. For example, you could call SetStyleAttribute() to set scroll bar settings for the place holder control:

VB

```
' Get the IFrame from the HtmlPresenter
Dim iframe As HtmlElement = CType(phUrl.HtmlPresenter.Children(0), HtmlElement)

' set the style that controls scroll bar visibility
iframe.SetStyleAttribute("overflow", "auto")
```

C#

```
// Get the IFrame from the HtmlPresenter
HtmlElement iframe = (HtmlElement)phUrl.HtmlPresenter.Children[0];

// set the style that controls scroll bar visibility
iframe.SetStyleAttribute("overflow", "auto");
```

25.4.4.2 Calling Managed Code from Javascript

This second example reverses the direction of communication. Now we're going to kick off a managed "Scriptable" method from Javascript. The basic steps are:

1) In managed code, identify one or more methods as "scriptable"
2) Register the managed object for access in Javascript.
3) In the ASP.NET host application, add an "id" attribute to the Silverlight host control object so we can access it in Javascript.
4) Add a Javascript method that gets references to the Silverlight control and the registered managed object. Use the Javascript managed object reference to call one of the Scriptable methods.

Mark Methods as Scriptable

Mark the methods you will need to access from Javascript with the **ScriptableMember** attribute. You will need to add a reference to the **System.Windows.Browser** namespace to support ScriptableMember usage. The method below takes a string and sets the SourceUrl of a place holder.

VB

```
<ScriptableMember()> _
Public Sub SetPlaceholderUrl(ByVal url As String)
  phTarget.SourceUrl = New Uri(url, UriKind.RelativeOrAbsolute)
End Sub
```

C#

```
[ScriptableMember()]
public void SetPlaceholderUrl(string url)
{
   phTarget.SourceUrl = new Uri(url, UriKind.RelativeOrAbsolute);
}
```

> **Gotcha!**
>
> Don't forget to make your scriptable methods public or you will get the error "Object does not have a ScriptableAttribute or any scriptable members".

Register the Managed Object for Access in Javascript

Call the **HtmlPage.RegisterScriptableObject()** method, pass it a name that can be used later in Javascript and an instance of the object it represents. Note the "MySilverlightPage" name for use later in Javascript.

```
Public Sub New()
      InitializeComponent()
      ' register a managed object for access by javascript
      HtmlPage.RegisterScriptableObject("MySilverlightPage", Me)
End Sub
```

```
public MainPage()
{
   InitializeComponent();
   // register a managed object for access by javascript
   HtmlPage.RegisterScriptableObject("MySilverlightPage", this);
}
```

Identify the Silverlight Control Object

Locate the host ASP.NET application (the same place where you set the "windowless" parameter that enables place holders), and add an "id" attribute to the Silverlight control object. Here it's named "silverlightControl".

```
<body>
   <form id="form1" runat="server" style="height:100%">
   <div id="silverlightControlHost">
      <object id="silverlightControl" <-- . . .
```

Use Javascript to Call Scriptable Methods

As demonstrated in the code below, create a Javascript function that uses the scriptable method.

Get a reference to the Silverlight control object, using the "id" attribute you set in the previous step. From there, drill down through the content to get the object registered as "MySilverlightPage" and call the scriptable method from it.

The example below also hooks up the Javascript function to the "onclick" event of a button.

```
<html>
<head>
  <title>Test Page</title>

  <script type="text/javascript">
    //. . .
    function callManagedCode() {
      // get the textbox text from this page
      var tbUrl = document.getElementById("tbUrl");
      var text = tbUrl.value;

      // get the silverlight control object from the host page
      // drill down through the content to get the registered object
      // "MySilverlightPage" and call its scriptable methods
      var silverlightControl =
        parent.document.getElementById("silverlightControl");
      silverlightControl.Content.MySilverlightPage.SetPlaceholderUrl(text);
    }
  </script>

</head>
<body>
  <h1>
    Enter a new Url</h1>
  <input id="tbUrl" />
  <input id="btnUrl" type="button" onclick="javascript:callManagedCode()"
   value="Call Managed Code" />
</body>
</html>
```

25.5 Binding

In this walk through you will bind a RadTabControl to an ObservableCollection of "Article", where each article will have a "Title" and "SourceUri". The content of each tab will contain a RadHtmlPlaceHolder bound to "SourceUri".

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.
2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.
3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Navigation**

c) **Telerik.Windows.Themes.Vista**

XAML Editing

1) Open MainPage.xaml for editing.
2) Add XML namespace references for **Telerik.Windows.Controls** and **Telerik.Windows.Navigation** namespaces. Also add a "Loaded" event handler to the UserControl element.

```
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
```

3) Add a UserControl.Resources section that defines the RadTabControl layout so that each tab will contain a bound RadHtmlPlaceholder. Name the resource "TabHtmlHolderStyle" so we can reference it later from the RadTabControl definition.

Most of the work is in this style definition, so take a minute to review how this is assembled. The HeaderTemplate contains only a text block bound to the "Title" property of our custom "Article" object. The ContentTemplate is the area below the tab where the RadHtmlPlaceholder lives. The place holder SourceUrl property is bound to the "Article" object "SourceUri" property.

```
<UserControl.Resources>
  <Style x:Key="TabHtmlHolderStyle"
      TargetType="telerik:RadTabItem">
    <Setter Property="HeaderTemplate">
      <Setter.Value>
        <DataTemplate>
          <TextBlock Text="{Binding Title}" Margin="5" />
        </DataTemplate>
      </Setter.Value>
    </Setter>
    <Setter Property="ContentTemplate">
      <Setter.Value>
        <DataTemplate>
          <telerik:RadHtmlPlaceholder x:Name="phArticle"
              SourceUrl="{Binding SourceUri}" />
        </DataTemplate>
      </Setter.Value>
    </Setter>
  </Style>
</UserControl.Resources>
```

4) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. The RadTabControl added to the grid should be named "tbArticles" so we can reference it in code later when we want to assign data. Set the VerticalAlignment property to "Top" and point the ItemContainerStyle back to our "TabHtmlHolderStyle".

```
<Grid x:Name="LayoutRoot">
  <telerik:RadTabControl x:Name="tbArticles"
      VerticalAlignment="Top"
      ItemContainerStyle="{StaticResource TabHtmlHolderStyle" />
</Grid>
```

Code Behind

1) In the Solution Explorer, add a new class file and populate it with the code below. This will define the "Article" INotifyPropertyChanged implementation and the "Articles" ObservableCollection.

```
Public Class Articles
  Inherits ObservableCollection(Of Article)
  Public Sub New()
    Me.Add(New Article() With {.Title = "Wikipedia", .SourceUri = New Uri("http://www.wikipedia.org")})
    Me.Add(New Article() With {.Title = "Dictionary", .SourceUri = New Uri("http://www.dictionary.com")})
    Me.Add(New Article() With {.Title = "Google Translation Services", .SourceUri = New Uri("http://translate.go
    Me.Add(New Article() With {.Title = "Thesaurus", .SourceUri = New Uri("http://thesaurus.reference.com/")})
  End Sub
End Class
```

```
Public Class Article
  Implements INotifyPropertyChanged
  Private title_Renamed As String
  Public Property Title() As String
    Get
      Return title_Renamed
    End Get

    Set(ByVal value As String)
      If title_Renamed <> value Then
        title_Renamed = value
        OnPropertyChanged("Title")
      End If
    End Set
  End Property

  Private sourceUri_Renamed As Uri
  Public Property SourceUri() As Uri
    Get
      Return sourceUri_Renamed
    End Get

    Set(ByVal value As Uri)
      If sourceUri_Renamed <> value Then
        sourceUri_Renamed = value
        OnPropertyChanged("SourceUri")
      End If
    End Set
  End Property

  Public Sub OnPropertyChanged(ByVal propertyName As String)
    RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
  End Sub

  Public Event PropertyChanged As PropertyChangedEventHandler

End Class
```

C#

```
public class Articles: ObservableCollection<Article>
{
   public Articles()
   {
      this.Add(new Article()
      {
         Title = "Wikipedia",
         SourceUri = new Uri("http://www.wikipedia.org")
      });
      this.Add(new Article()
      {
         Title = "Telerik",
         SourceUri = new Uri("http://www.telerik.com")
```

```
        });
        this.Add(new Article()
        {
            Title = "Google Translation Services",
            SourceUri = new Uri("http://translate.google.com/?hl=en#")
        });
        this.Add(new Article()
        {
            Title = "Thesaurus",
            SourceUri = new Uri("http://thesaurus.reference.com/")
        });
    }
}

public class Article : INotifyPropertyChanged
{
    private string title;
    public string Title
    {
        get
        {
            return title;
        }

        set
        {
            if (title != value)
            {
                title = value;
                OnPropertyChanged("Title");
            }
        }
    }

    private Uri sourceUri;
    public Uri SourceUri
    {
        get
        {
            return sourceUri;
        }

        set
        {
            if (sourceUri != value)
            {
                sourceUri = value;
                OnPropertyChanged("SourceUri");
            }
        }
    }

    public void OnPropertyChanged(string propertyName)
    {
        if (PropertyChanged != null)
        {
          PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
      }

      public event PropertyChangedEventHandler PropertyChanged;

    }
```

2) In the code-behind for the page, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **Telerik.Windows.Controls**

3) In the constructor, before the InitializeComponent() method call, create a new **VistaTheme** instance and set the **IsApplicationTheme** property to True. Be sure to leave the InitializeComponent() method call in place.

VB

```
Public Sub New()
  CType(New VistaTheme(), VistaTheme).IsApplicationTheme = True
  InitializeComponent()
End Sub
```

C#

```
public MainPage()
{
   new VistaTheme().IsApplicationTheme = true;
   InitializeComponent();
}
```

4) In the UserControl Loaded event handler, create an instance of the Articles object and assign it to the tab control **ItemsSource** property.

VB

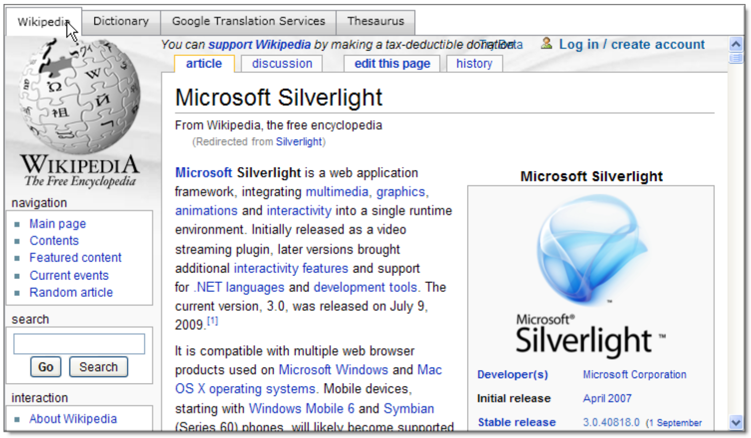
```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  tbArticles.ItemsSource = New Articles()
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   tbArticles.ItemsSource = new Articles();
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



You can support Wikipedia by making a tax-deductible donation.

Microsoft Silverlight

From Wikipedia, the free encyclopedia

(Redirected from Silverlight)

Microsoft Silverlight is a web application framework, integrating multimedia, graphics, animations and interactivity into a single runtime environment. Initially released as a video streaming plugin, later versions brought additional interactivity features and support for .NET languages and development tools. The current version, 3.0, was released on July 9, 2009.[1]

It is compatible with multiple web browser products used on Microsoft Windows and Mac OS X operating systems. Mobile devices, starting with Windows Mobile 6 and Symbian (Series 60) phones, will likely become supported

Microsoft Silverlight

Developer(s)	Microsoft Corporation
Initial release	April 2007
Stable release	3.0.40818.0 (1 September

25.6 Wrap Up

In this chapter you learned how RadHtmlPlaceholder is used to host HTML content in Silverlight applications. First you added custom HTML content to a place holder to see how it renders, then you used an external web page. You learned how to host place holders in the page, in other controls and within RadWindows. You also learned how to respond to events when a page loads into the place holder. You learned about interaction between Javascript and managed code. Finally, you learned how to bind a series of place holder controls within an ItemsControl to custom objects.

Part XV

MediaPlayer

26 MediaPlayer

26.1 Objectives

In this chapter you will learn how to incorporate media into your Silverlight applications. First you will build a play list using static XAML and in the process learn how to define RadMediaItem elements. You will learn about the media types supported by the media player, how to work with video size and full screen, how to add chapters to a media item and about the available events for the media player. You will construct an application where the media player consumes data from an RSS service including the titles, descriptions, images and the video itself. Finally, you will learn how to create a play list with a unique appearance using a custom template.

Find the projects for this chapter at...

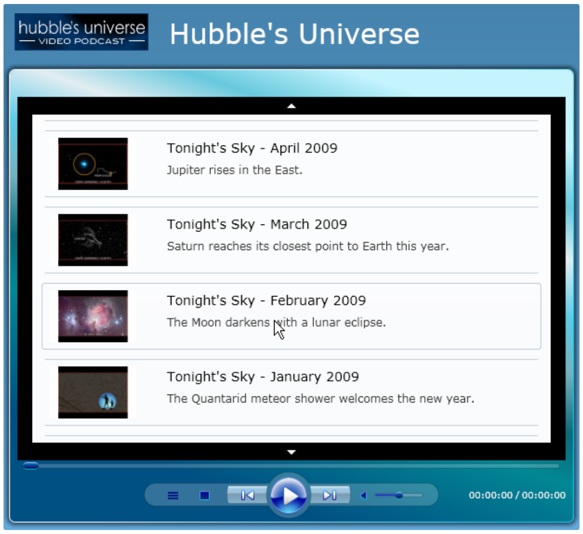
\Courseware\Projects\<CS|VB>\MediaPlayer\MediaPlayer.sln.

26.2 Overview

RadMediaPlayer makes it easy to incorporate rich media content to your web site. Create your own fixed play lists, create play lists in code or bind the play list for complete flexibility. You can set chapters at custom intervals for intuitive navigation. Multiple size and stretch options let you control media proportions while the built-in full-screen mode lets you make maximum use of the screen real-estate.

RadMediaPlayer features include:

- Set Chapters
- Create Playlists
- Set Video Size
- Full Screen Mode

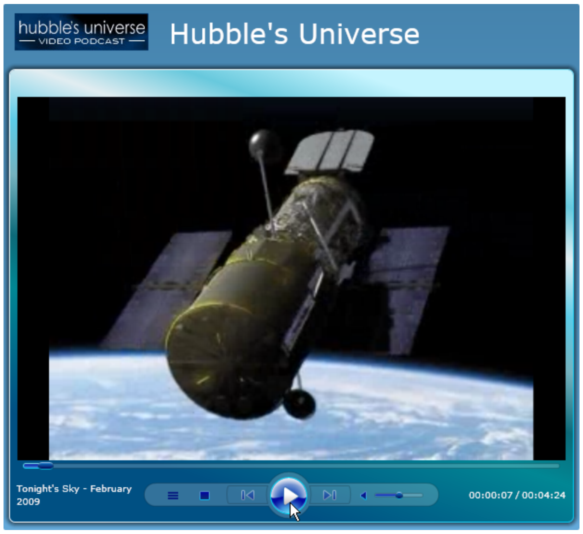


Tonight's Sky - April 2009
Jupiter rises in the East.

Tonight's Sky - March 2009
Saturn reaches its closest point to Earth this year.

Tonight's Sky - February 2009
The Moon darkens with a lunar eclipse.

Tonight's Sky - January 2009
The Quantarid meteor shower welcomes the new year.



26.3 Getting Started

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.MediaPlayer**

c) **Telerik.Windows.Themes.Vista**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace references to the Telerik.Windows.Controls namespace in both the **Telerik.Windows.Controls** and **Telerik.Windows.Controls.MediaPlayer** assemblies.

```
<UserControl
xmlns:telerik="xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .>
```

3) Add a RadMediaPlayer from the Toolbox to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. Set the **StyleManager.Theme** to "Vista" and the **IsPlaylistVisible** to "True".

```
<telerik:RadMediaPlayer
      telerik:StyleManager.Theme="Vista"
      IsPlaylistVisible="True">
</telerik:RadMediaPlayer>
```

4) Add a RadMediaItem from the Toolbox to a point inside the RadMediaPlayer element begin and end tags and set properties:

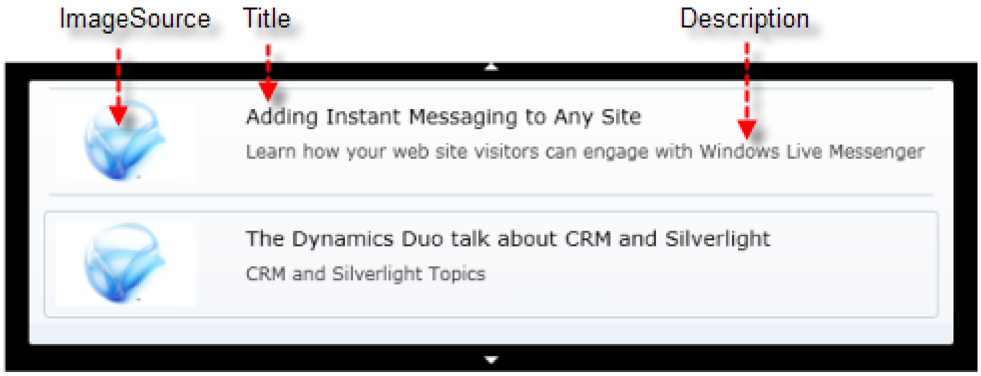
a) **ImageSource** = "http://neosmart.net/blog/wp-content/uploads/microsoft-silverlight.png"

b) **Title** = "Adding Instant Messaging to Any Site"

c) **Source** = "http://msstudios.vo.llnwd.net/o21/mix08/08_WMVs/T03.wmv"

Notes

The ImageSource is the path to the image on the left. Title and Description are the text labels displayed top and bottom right, respectively. Source is the path to the video that will play when one of the items is clicked.



Adding Instant Messaging to Any Site
Learn how your web site visitors can engage with Windows Live Messenger

The Dynamics Duo talk about CRM and Silverlight
CRM and Silverlight Topics

5) Add a second RadMediaItem from the Toolbox to a point just below the first RadMediaItem.

a) **ImageSource** = "http://neosmart.net/blog/wp-content/uploads/microsoft-silverlight.png"

b) **Title** = "The Dynamics Duo talk about CRM and Silverlight"

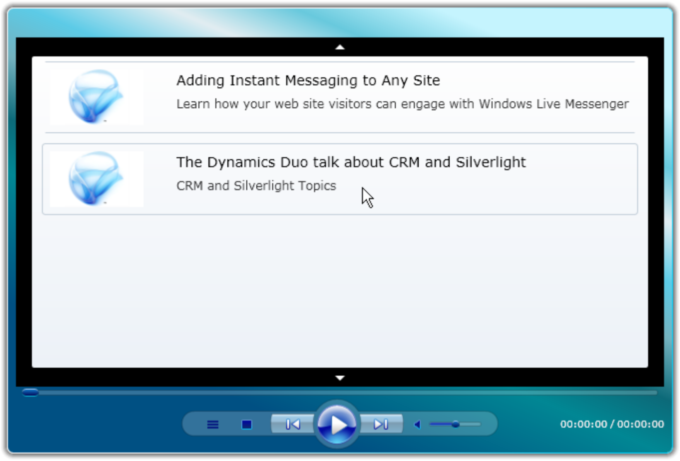
c) **Source** = "http://mschnlnine.vo.llnwd.net/d1/ch9/7/1/5/1/2/4/DynamicsDuoCRMSilverlight_ch9.wmv"

The RadMediaPlayer and RadMediaItem elements together should now look like the example below.

```
<telerik:RadMediaPlayer x:Name="mediaPlayer"
    telerik:StyleManager.Theme="Vista"
    IsPlaylistVisible="True" >
  <telerik:RadMediaItem
        ImageSource="http://neosmart.net/blog/wp-content/uploads/microsoft-silverlight.png"
        Title="Adding Instant Messaging to Any Site"
        Source="http://msstudios.vo.llnwd.net/o21/mix08/08_WMVs/T03.wmv">
  </telerik:RadMediaItem>
  <telerik:RadMediaItem
        ImageSource="http://neosmart.net/blog/wp-content/uploads/microsoft-silverlight.png"
        Title="The Dynamics Duo talk about CRM and Silverlight"
        Source="http://mschnlnine.vo.llnwd.net/d1/ch9/7/1/5/1/2/4/DynamicsDuoCRMSilverlight_ch9.wmv">
  </telerik:RadMediaItem>
</telerik:RadMediaPlayer>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Adding Instant Messaging to Any Site
Learn how your web site visitors can engage with Windows Live Messenger

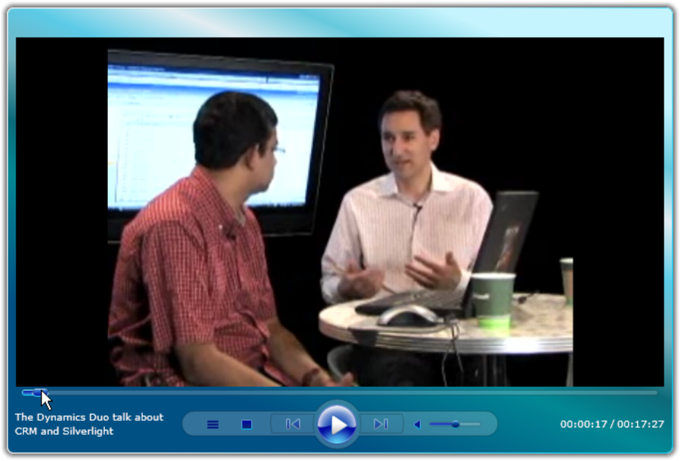
The Dynamics Duo talk about CRM and Silverlight
CRM and Silverlight Topics

Gotcha!

Be sure to run the application from the ASP.NET host application. In other words, the startup project should be the ASP.NET host application, not the Silverlight application itself. If you run with the Silverlight application, the application will run, but the images will not display and the WMV files will not show.

Test Application Features

1. The play list should display automatically.
2. The images and titles should display in the play list.
3. Clicking an item in the play list should trigger the media player to stream and show the video.



26.4 Control Details

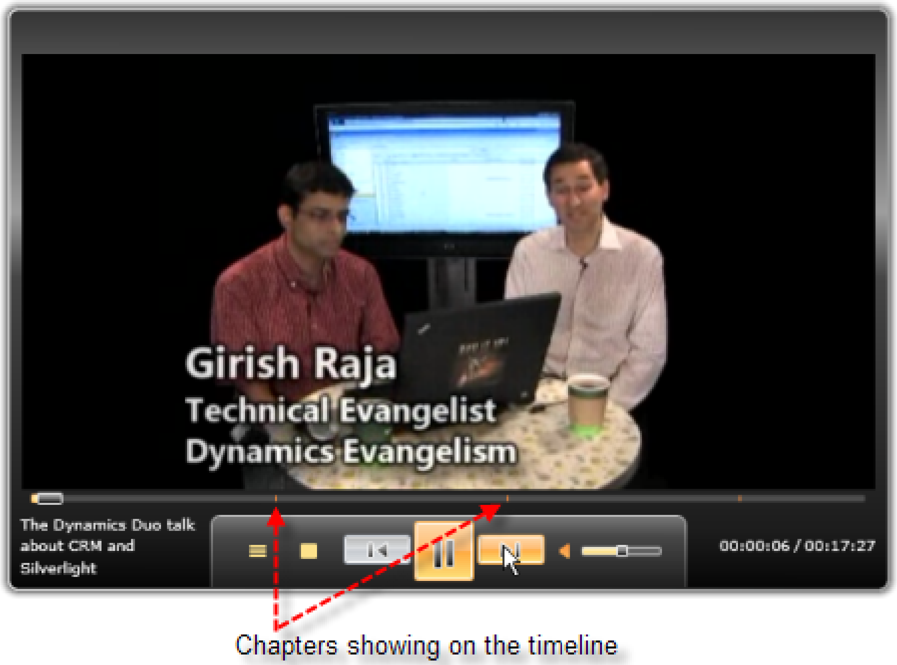
Supported Media Types

RadMediaPlayer wraps the Silverlight MediaElement control. The following is a list of supported video media types. See http://msdn.microsoft.com/en-us/library/cc189080(VS.95).aspx for more complete information.

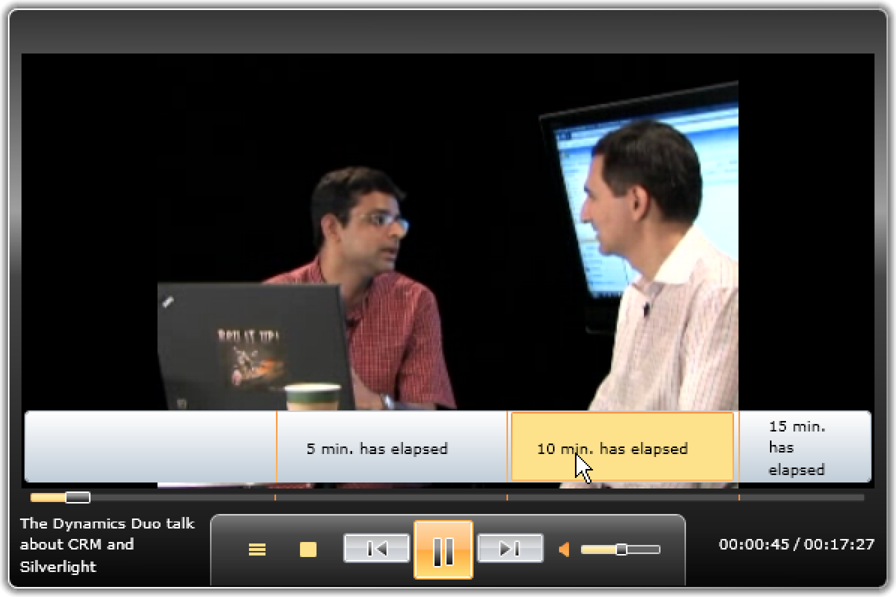
- Raw Video
- YV12 - YCrCb(4:2:0)
- RGBA - 32 bit Alpha Red, Green, Blue
- WMV1: Windows Media Video 7
- WMV2: Windows Media Video 8
- WMV3: Windows Media Video 9
- Supports Simple and Main Profiles.
- Supports only progressive (non-interlaced) content.
- WMVA: Windows Media Video Advanced Profile, non-VC-1
- WVC1: Windows Media Video Advanced Profile, VC-1
- Supports Advanced Profile.
- Supports only progressive (non-interlaced) content.
- H264 (ITU-T H.264 / ISO MPEG-4 AVC)

Chapters

Videos can be split up into arbitrary time periods called "Chapters". Each chapter has a **Position** and **Title**. The position is a string property in the format of "00:10:00" in the format of hours : minutes : seconds. Title is a string that is displayed to describe the chapter.


Girish Raja
Technical Evangelist
Dynamics Evangelism
The Dynamics Duo talk about CRM and Silverlight
00:00:06 / 00:17:27
Chapters showing on the timeline

Moving the mouse over the timeline causes a panel with the chapter titles to display. Clicking on a title moves the current marker on the timeline to the beginning of that chapter. You can also navigate chapters by clicking the forward and back buttons.

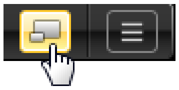


The markup for chapters shown in the screenshots is defined below.

```
<telerik:RadMediaPlayer Width="600" Height="400"
    Margin="20" telerik:StyleManager.Theme="Office_Silver">
  <telerik:RadMediaItem
 Source="http://mschnlnine.vo.llnwd.net/d1/ch9/7/1/5/1/2/4/DynamicsDuoCRMSilverlight_ch9.wmv"
 ImageSource="http://mschnlnine.vo.llnwd.net/d1/ch9/7/1/5/1/2/4/DynamicsDuoCRMSilverlight_small_ch9.j
 Title="The Dynamics Duo talk about CRM and Silverlight">
    <telerik:RadMediaChapter Position="00:05:00"
        Title="5 min. has elapsed" />
    <telerik:RadMediaChapter Position="00:10:00"
        Title="10 min. has elapsed" />
    <telerik:RadMediaChapter Position="00:15:00"
        Title="15 min. has elapsed" />
  </telerik:RadMediaItem>
</telerik:RadMediaPlayer>
```

Video Size and Proportion

RadMediaPlayer allows you to display the media content in fullscreen mode by internally changing the application fullscreen mode. You can easily toggle the mode by pressing the FullScreen button or changing the IsFullScreen property.



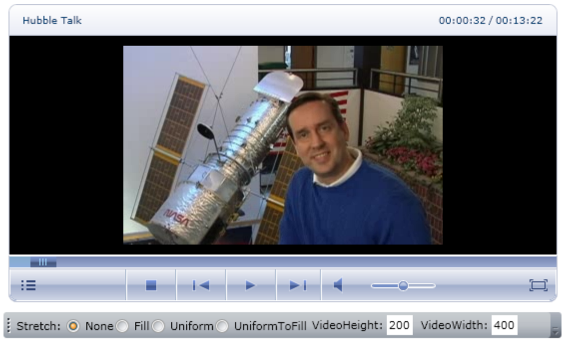
Gotcha!

The fullscreen mode can only be entered after a user triggered action.

Since the fullscreen functionality may be application dependant, you could customize it by handling the FullScreenChanged event, that is raised when the RadMediaPlayer changes its fullscreen mode.

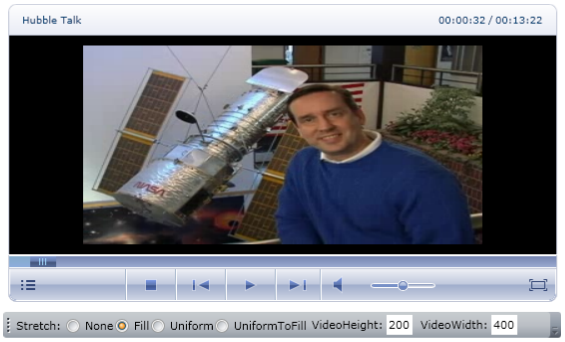
Another way to get at the dimensions of the area where the video plays is the **VideoStretch**, **VideoWidth** and **VideoHeight** properties. Here are a few screenshots that will give you an idea of how these settings interact. Be aware that these are properties of the RadMediaItem, not the player. You can use the RadMediaPlayer.**SelectedItem** to get access to the selected MediaItem.

None



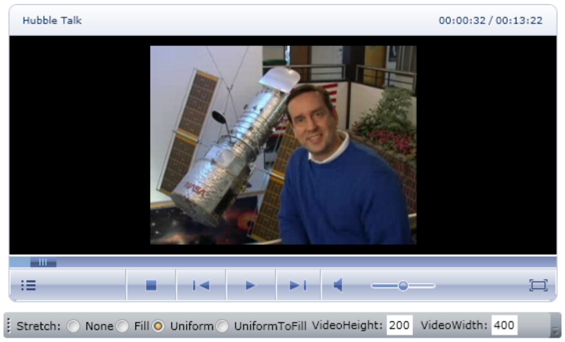
Stretch: ● None ○ Fill ○ Uniform ○ UniformToFill VideoHeight: 200 VideoWidth: 400

Fill



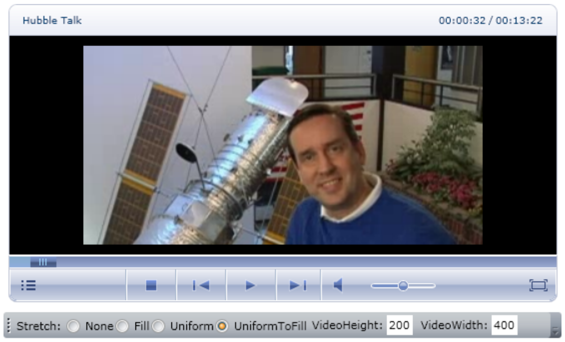
Stretch: ○ None ● Fill ○ Uniform ○ UniformToFill VideoHeight: 200 VideoWidth: 400

Uniform



Stretch: ○ None ○ Fill ● Uniform ○ UniformToFill VideoHeight: 200 VideoWidth: 400

UniformToFill



Stretch: ○ None ○ Fill ○ Uniform ● UniformToFill VideoHeight: 200 VideoWidth: 400

Events

- **CurrentStateChanged**: Use this event to find out when the media player **CurrentState** property changes. CurrentState is a MediaElementState enumeration and can be **Closed**, **Opening**, **Buffering**, **Playing**, **Paused**, **Stopped**, **Individualizing** or **AquiringLicense**.
- **ChapterReached**: This event fires when the timeline progress to a new chapter. Use the **ChapterTitle** property to get the text for the current media item and chapter. You can also use the **MediaElementTotalSeconds** property to get the current progress through the timeline.
- **MediaOpened** fires when the video first opens, **DownloadProgressChanged** then fires as the current media item progressively downloads and finally, the **MediaEnded** event fires as the timeline completes.

26.5 Binding

In this walk through we will use an RSS feed to populate screen elements, the play list text, images and the individual videos. We will not need to provide any custom templating because we will transform our data to a collection of RadMediaItem before binding to the media player **ItemsSource** property. This particular site, HubbleSite.org, at the time of this writing has a Crossdomain.xml policy file in place, so we should be able to access all media directly.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.MediaPlayer**

c) **Telerik.Windows.Themes.Vista**

XAML Editing

1) Open MainPage.xaml for editing.

2) Add XML namespace references for the following assemblies to the UserControl element.

```
<UserControl
xmlns:telerik="http://schemas.telerik.com/2008/xaml/presentation"
. . .>
```

3) Locate the **TwilightBlueTheme** control from the "Silverlight XAML Controls" tab of the Toolbox and drop it between the begin and end tags of the main "LayoutRoot" grid.

This will style the background and other elements with a gradient blue that should blend nicely with the Telerik "Vista" theme of the media player to be added later.

```
<Grid x:Name="LayoutRoot">
  <twilightBlue:TwilightBlueTheme>
  </twilightBlue:TwilightBlueTheme>
</Grid>
```

4) Inside the "TwilightBlueTheme" element, add the XAML below. This XAML contains a stack panel arranged vertically. We will use the XAML comments in following steps to include additional elements.

```
<StackPanel>
  <!--logo and title-->
  <!--media player-->
</StackPanel>
```

5) Replace the "<!--logo and title-->" comment with the XAML below. The XAML includes a new StackPanel element arranged horizontally. Inside the stack panel, is a HyperlinkButton that navigates to the site and a tool tip that displays a short introduction to the site. Inside the HyperlinkButton, an Image control displays the logo image for the site. Below the HyperlinkButton a text block displays the title of the site.

```
<!--logo and title-->
<StackPanel x:Name="spTitle" Orientation="Horizontal">
  <HyperlinkButton Margin="10"
      NavigateUri="{Binding LinkUrl}"
      ToolTipService.ToolTip="{Binding Description}"
      TargetName="_blank">
    <Image Source="{Binding LogoUrl}"></Image>
  </HyperlinkButton>
  <TextBlock Text="{Binding Title}" FontSize="30"
      Margin="10" />
</StackPanel>
```

6) Drag a RadMediaPlayer control from the Toolbox to a point below the "<!--media player-->" comment. Set the name to be "mediaPlayer" so that we can access the control later in code. Set the **IsPlaylistVisible** property to "True" and **Height** to "500". Add an event handler for the **Loaded** event.

```
<!--media player-->
<telerik:RadMediaPlayer x:Name="mediaPlayer"
    Loaded="mediaPlayer_Loaded"
    IsPlaylistVisible="True" Height="500">
</telerik:RadMediaPlayer>
```

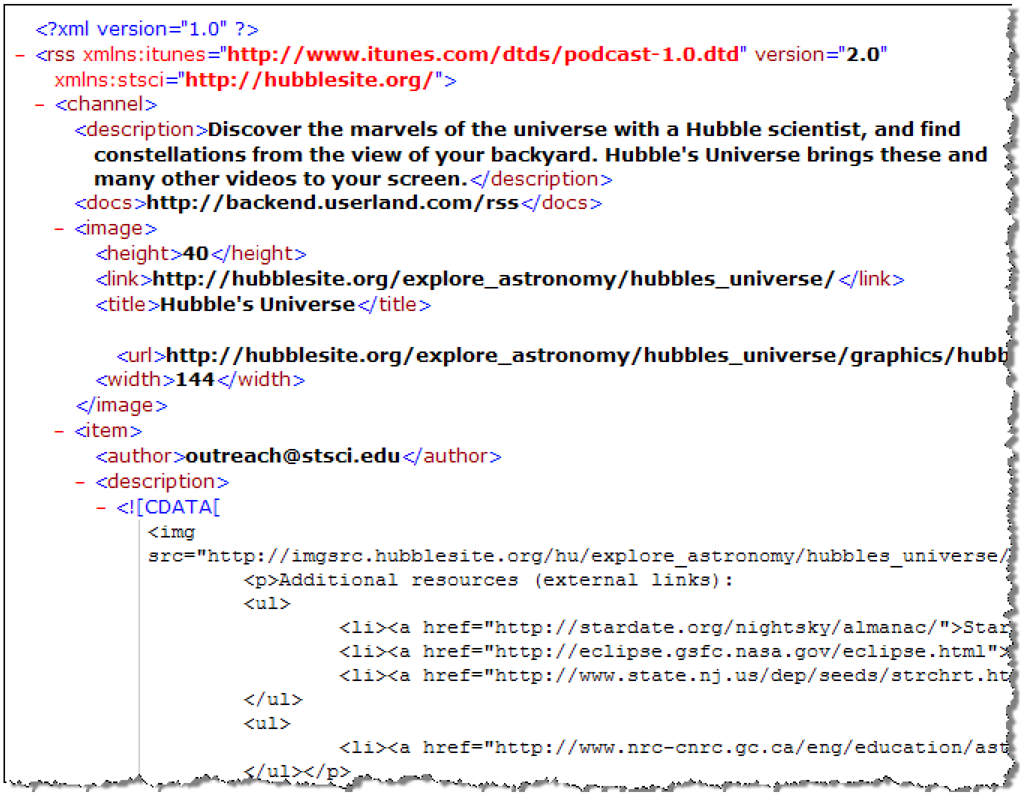
Understanding the RSS Data

1) To understand what's happening during the coding phase of this project, you should take a look at the RSS feed XML and understand the structure of the data. Using Internet Explorer, navigate to the page below and save the page as XML:

http://hubblesite.org/explore_astronomy/hubbles_universe/rss.php?feed=windows-320

2) The screenshot below shows a small sample from the top of the file. There's a node at the top called "channel" that contains a description for the site, a "title" and an "image" that contains a path to the logo for the site. Below that are "N" number of "item" tags, each containing one article's worth of information in a "description" tag. Each article has text describing the article, a link to an image and a link to a video in "*.wmv" format.

   The "gotcha" is that the description node contains "CDATA", i.e. free formatted data. In this case the CDATA is HTML. We need to parse the html to get the description, image and video. We will use regular expressions to make this job relatively painless, but will not go into detail on how regular expressions work.



```
  <?xml version="1.0" ?>
- <rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd" version="2.0"
    xmlns:stsci="http://hubblesite.org/">
  - <channel>
      <description>Discover the marvels of the universe with a Hubble scientist, and find
        constellations from the view of your backyard. Hubble's Universe brings these and
        many other videos to your screen.</description>
      <docs>http://backend.userland.com/rss</docs>
    - <image>
        <height>40</height>
        <link>http://hubblesite.org/explore_astronomy/hubbles_universe/</link>
        <title>Hubble's Universe</title>

          <url>http://hubblesite.org/explore_astronomy/hubbles_universe/graphics/hubb
        <width>144</width>
      </image>
    - <item>
        <author>outreach@stsci.edu</author>
      - <description>
        - <![CDATA[
            <img
            src="http://imgsrc.hubblesite.org/hu/explore_astronomy/hubbles_universe/
                    <p>Additional resources (external links):
                    <ul>
                            <li><a href="http://stardate.org/nightsky/almanac/">Star
                            <li><a href="http://eclipse.gsfc.nasa.gov/eclipse.html">
                            <li><a href="http://www.state.nj.us/dep/seeds/strchrt.ht
                    </ul>
                    <ul>
                            <li><a href="http://www.nrc-cnrc.gc.ca/eng/education/ast
                    </ul></p>
```

Extract RSS Data

1) Add a new class file to the project and name the file "HubbleItem.cs". Define the class using the code below.

   This class will be a simple container for each that will be shown later in the play list of the media player.

VB

```
Public Class HubbleItem
  Private privateTitle As String
  Public Property Title() As String
    Get
      Return privateTitle
    End Get
    Set(ByVal value As String)
      privateTitle = value
    End Set
  End Property
  Private privateVideoUri As Uri
  Public Property VideoUri() As Uri
    Get
      Return privateVideoUri
    End Get
    Set(ByVal value As Uri)
      privateVideoUri = value
    End Set
  End Property
  Private privateImageUri As Uri
  Public Property ImageUri() As Uri
    Get
      Return privateImageUri
    End Get
    Set(ByVal value As Uri)
      privateImageUri = value
    End Set
  End Property
End Class
```

C#

```
public class HubbleItem
{
   public string Title { get; set; }
   public Uri VideoUri { get; set; }
   public Uri ImageUri { get; set; }
}
```

2) Add a new class file to the project and name the file "HubbleInfo.cs".

This class will contain header information about the Hubble site and a collection of "HubbleItem".

3) In the "HubbleInfo.cs" class file, add references to the "Imports" (VB) or "using" (C#) section of the code for these namespaces:

a) **System.Collections.Generic**

b) **System.Linq**

c) **System.Net**

d) **System.Text.RegularExpressions**

e) **System.Xml.Linq** (supports XDocument, XElement)

4) Add a delegate "LoadedEventHandler" to the top of the class file. *This will be used later to define a "Loaded" event.*

VB

```
Public Delegate Sub LoadedEventHandler( _
ByVal sender As Object, ByVal e As EventArgs)
```

C#

```
public delegate void LoadedEventHandler(object sender, EventArgs e);
```

5) Add the code below to the "HubbleInfo" class to define a series of string constants.

The constants include a URL for the Hubble site, the Url for the RSS feed on the Hubble site and two patterns to be used later in regular expressions.

VB

```
Private Const hubbleUrl As String = "http://hubblesite.org"
Private Const feedUrl As String = hubbleUrl & _
"/explore_astronomy/hubbles_universe/rss.php?feed=windows-320"
Private Const urlTagPattern As String = _
"http://([\w+?\.\w+])+([a-zA-Z0-9\~\!\@\#\$\%\^\&amp;" & _
"\*\(\)_\-\=\+\\\/\?\.\:\;\'\,]*)?"
Private Const tagPattern As String = "<p>\s*(.+?)\s*</p>"
```

C#

```
private const string hubbleUrl = "http://hubblesite.org";
private const string feedUrl = hubbleUrl +
   "/explore_astronomy/hubbles_universe/rss.php?feed=windows-320";
private const string urlTagPattern =
   "http://([\\w+?\\.\\w+])+([a-zA-Z0-9\\~\\!\\@\\#\\$\\%\\^\\&amp;" +
   "\\*\\(\\)_\\-\\=\\+\\\\\\/\\?\\.\\:\\;\\'\\,]*)?";
private const string tagPattern = @"<p>\s*(.+?)\s*</p>";
```

6) Add the code below to the HubbleInfo class. It should include an IEnumerable collection of HubbleItem, a title, description and Uri's to the site and site logo.

```
Private privateItems As IEnumerable(Of HubbleItem)
Public Property Items() As IEnumerable(Of HubbleItem)
  Get
    Return privateItems
  End Get
  Set(ByVal value As IEnumerable(Of HubbleItem))
    privateItems = value
  End Set
End Property
Private privateTitle As String
Public Property Title() As String
  Get
    Return privateTitle
  End Get
  Set(ByVal value As String)
    privateTitle = value
  End Set
End Property
Private privateDescription As String
Public Property Description() As String
  Get
    Return privateDescription
  End Get
  Set(ByVal value As String)
    privateDescription = value
  End Set
End Property
Private privateLogoUrl As Uri
Public Property LogoUrl() As Uri
  Get
    Return privateLogoUrl
  End Get
  Set(ByVal value As Uri)
    privateLogoUrl = value
  End Set
End Property
Private privateLinkUrl As Uri
Public Property LinkUrl() As Uri
  Get
    Return privateLinkUrl
  End Get
  Set(ByVal value As Uri)
    privateLinkUrl = value
  End Set
End Property
```

C#

```
public IEnumerable<HubbleItem> Items { get; set; }
public string Title { get; set; }
public string  Description  { get; set; }
public Uri LogoUrl { get; set; }
public Uri LinkUrl { get; set; }
```

7) Declare a Loaded event and OnLoaded() method to trigger the event.

VB

```
Public Event Loaded As LoadedEventHandler

Protected Overridable Sub OnLoaded()
  RaiseEvent Loaded(Me, New EventArgs())
End Sub
```

C#

```
public event LoadedEventHandler Loaded;

protected virtual void OnLoaded()
{
   if (Loaded != null)
      Loaded(this, new EventArgs());
}
```

8) Define a Load() method using the code below. The method creates a WebClient instance, sets up a DownloadStringCompleted event handler and calls the DownloadStringAsync() method. Pass the "feedUrl" constant to the Uri constructor. The call will download the RSS XML from the Hubble site.

VB

```
Public Sub Load()
  Dim client As New WebClient()
  AddHandler client.DownloadStringCompleted, _
AddressOf client_DownloadStringCompleted
  client.DownloadStringAsync(New Uri(feedUrl))
End Sub
```

C#

```
public void Load()
{
   WebClient client = new WebClient();
   client.DownloadStringCompleted +=
      new DownloadStringCompletedEventHandler(
         client_DownloadStringCompleted);
   client.DownloadStringAsync(new Uri(feedUrl));
}
```

9) Define the DownloadStringCompleted event handler using the code below.

The "e.Result" here is the RSS feed as an XML string. Use the XDocument Parse() method to create an XDocument. From there, get the top level elements "channel", "image" and "description". Populate HubbleInfo properties using element values. Call the GetHubbleItems private method (to be written later). Finally, trigger the Loaded event by calling the OnLoaded() method.

VB

```
Private Sub client_DownloadStringCompleted(ByVal sender As Object, _
ByVal e As DownloadStringCompletedEventArgs)
   Dim document As XDocument = XDocument.Parse(e.Result)

   ' get the top level nodes of the document
   Dim channel = document.Root.Element("channel")
   Dim image = channel.Element("image")
   Dim description = channel.Element("description")

   ' populate HubbleInfo properties
   Me.LogoUrl = New Uri(image.Element("url").Value, UriKind.Absolute)
   Me.LinkUrl = New Uri(hubbleUrl, UriKind.Absolute)
   Me.Title = image.Element("title").Value.ToString()
   Me.Description = description.Value.Replace(".", "." + Environment.NewLine).Trim()

   ' populate items collection
   Me.Items = GetHubbleItems(channel)

   ' notify consumer that data is ready
   OnLoaded()

End Sub
```

```
void client_DownloadStringCompleted(object sender,
DownloadStringCompletedEventArgs e)
{
   XDocument document = XDocument.Parse(e.Result);

  // get the top level nodes of the document
  var channel = document.Root.Element("channel");
  var image = channel.Element("image");
  var description = channel.Element("description");

  // populate HubbleInfo properties
  this.LogoUrl = new Uri(image.Element("url").Value, UriKind.Absolute);
  this.LinkUrl = new Uri(hubbleUrl, UriKind.Absolute);
  this.Title = image.Element("title").Value.ToString();
  this.Description = description.Value.Replace(".", "." + Environment.NewLine).Trim();

  // populate items collection
   this.Items = GetHubbleItems(channel);

  // notify consumer that data is ready
  OnLoaded();
}
```

10)Create a new method GetHubbleItems that takes an XElement as a parameter and returns an IEnumerable of HubbleItem.

This method is primarily a LINQ expression that extracts "item" elements and populates HubbleItem objects. The title is taken from the element value of the same name. A series of LINQ "let" statements extract "CData" data from the "description" element, then uses Regex to get the remaining detail. Regex returns the "<p>" tag if it exists and we also strip the tags themselves and assign it to the temporary "description" variable. Regex is also used to get the image URL. The video url is taken from the "<enclosure><url>" elements. Finally, all this data is used to populate new HubbleItem instances.

```
Private Function GetHubbleItems(ByVal channel As XElement) _
As IEnumerable(Of HubbleItem)

  Dim regxUrls As New Regex(urlTagPattern, RegexOptions.IgnoreCase)
  Dim regxPTag As New Regex(tagPattern, RegexOptions.IgnoreCase)
  Dim items = _
    From i In channel.Elements("item")_
    Let title = i.Element("title").Value_
    Let cData = TryCast(i.Element("description").FirstNode, XCData).Value_
    Let tempMatches = regxPTag.Matches(cData)_
    Let descTags = If(tempMatches.Count > 0, tempMatches(0).ToString(), [String].Empty)_
    Let description = Regex.Replace(descTags, "<.*?>", String.Empty)_
    Let imageUri = New Uri(regxUrls.Matches(cData)(0).ToString())_
    Let videoUri = New Uri(i.Element("enclosure").Attribute("url").Value)_
   Select New HubbleItem() With { _
          .Title = title, _
          .VideoUri = videoUri, _
          .ImageUri = imageUri _
  Return items

End Function
```

```
private IEnumerable<HubbleItem> GetHubbleItems(XElement channel)
{
        Regex regxUrls = new Regex(urlTagPattern, RegexOptions.IgnoreCase);
        Regex regxPTag = new Regex(tagPattern, RegexOptions.IgnoreCase);
        // populate HubbleInfo Items collection
        var items =

                from i in channel.Elements("item")
                let title = i.Element("title").Value
                // get the CData section of the xml
                let cData = (i.Element("description").FirstNode as XCData).Value
                // find the description "<p>" tag in the CData
                let tempMatches = regxPTag.Matches(cData)
                // only retain if there are "<p>" tags
                let descTags =
                        tempMatches.Count > 0 ? tempMatches[0].ToString() : String.Empty
                // remove the <p> tags from the description
                let description = Regex.Replace(descTags, "<.*?>", string.Empty)
                // find the image url in the CData
                let imageUri = new Uri(regxUrls.Matches(cData)[0].ToString())
                // get the image url that will appear in the playlist
                let videoUri = new Uri(i.Element("enclosure").Attribute("url").Value)

                // add the info to a RadMediaItem
                select new HubbleItem()
                {
                        Title = title,
                        VideoUri = videoUri,
                        ImageUri = imageUri
                };
        return items;
}
```

Code Behind

1) In the code behind for the main page, verify that these namespace references exist and add them if necessary:

a) **System.Linq**

b) **System.Windows**

c) **System.Windows.Controls**

d) **System.Windows.Media.Imaging**

e) **Telerik.Windows.Controls**

2) In the constructor for the page, create a new VistaTheme instance and set its IsApplicationTheme property to "True". This should be done before the InitializeComponent() method call.

VB

```
Public Sub New()
  CType(New VistaTheme(), VistaTheme).IsApplicationTheme = True
  InitializeComponent()
End Sub
```

C#

```
public MainPage()
{
   new VistaTheme().IsApplicationTheme = true;
   InitializeComponent();
}
```

3) In the media player Loaded event, create a new **HubbleInfo** instance, hook up its **Loaded** event and call the **Load()** method.

Indirectly, the Load() method call will kick off an asynchronous call from WebClient to retrieve the RSS XML and populate the HubbleInfo object. When that completes, the custom HubbleInfo Loaded event will fire.

VB

```
Private Sub mediaPlayer_Loaded( _
ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim hubbleInfo As New HubbleInfo()
  AddHandler hubbleInfo.Loaded, AddressOf hubbleInfo_Loaded
  hubbleInfo.Load()
End Sub
```

C#

```
void mediaPlayer_Loaded(object sender, RoutedEventArgs e)
{
   HubbleInfo hubbleInfo = new HubbleInfo();
   hubbleInfo.Loaded += new LoadedEventHandler(hubbleInfo_Loaded);
   hubbleInfo.Load();
}
```

4) In the Loaded event handler of the HubbleInfo object set the DataContext of the title StackPanel to the HubbleInfo instance. Use a LINQ statement to transform the collection of HubbleItem to a collection of RadMenuItem and assign the lot to the media player ItemsSource property.

VB

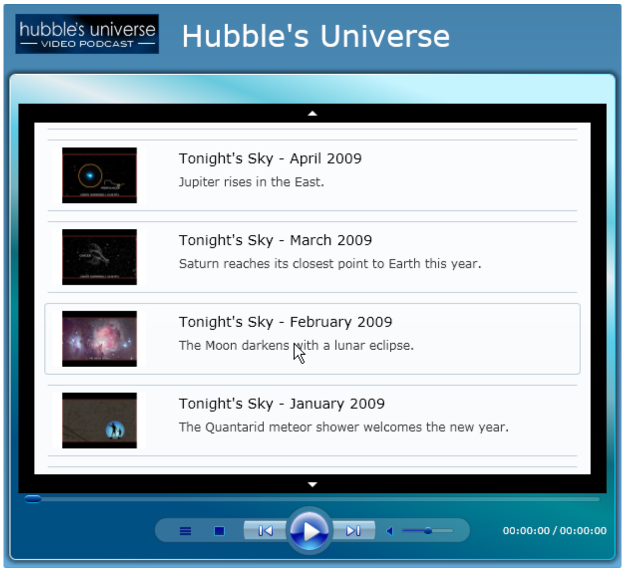
```
Private Sub hubbleInfo_Loaded(ByVal sender As Object, ByVal e As EventArgs)
  Dim info As HubbleInfo = TryCast(sender, HubbleInfo)
  spTitle.DataContext = info
  mediaPlayer.ItemsSource = _
    From i In info.Items _
    Select New RadMediaItem()
      Dim TempBitmapImage As BitmapImage = _
New BitmapImage(i.ImageUri), Source = i.VideoUri, ImageSource = New BitmapImage(i.ImageUri),  Title = i.T
End Sub
```

C#

```
void hubbleInfo_Loaded(object sender, EventArgs e)
{
   HubbleInfo info = sender as HubbleInfo;
   spTitle.DataContext = info;
   mediaPlayer.ItemsSource =
      from i in info.Items
      select new RadMediaItem()
      {
         Title = i.Title,
         ImageSource = new BitmapImage(i.ImageUri),
         Source = i.VideoUri
      };
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Tonight's Sky - April 2009
Jupiter rises in the East.

Tonight's Sky - March 2009
Saturn reaches its closest point to Earth this year.

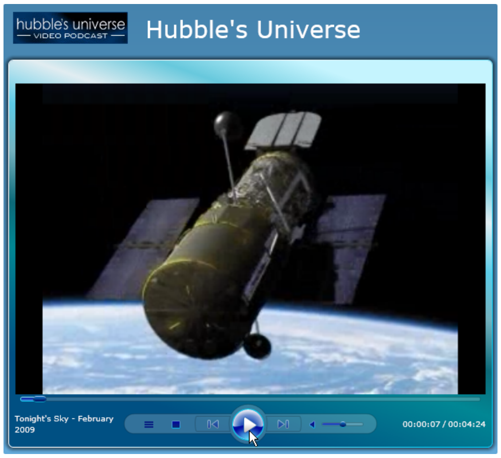
Tonight's Sky - February 2009
The Moon darkens with a lunar eclipse.

Tonight's Sky - January 2009
The Quantarid meteor shower welcomes the new year.

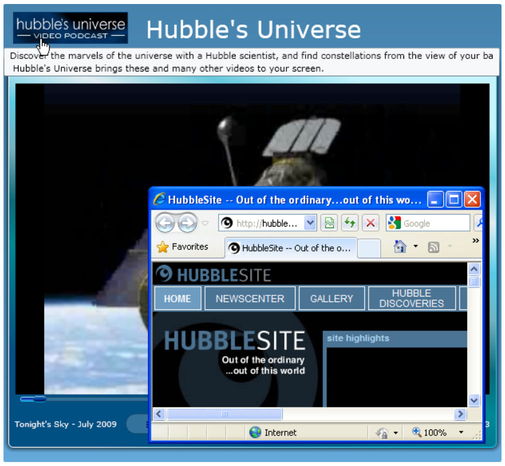
Test Application Features

1) Each of the play list items should display an image on the left, and a title to the right.

2) Clicking one of the play list entries should stream and play the video.



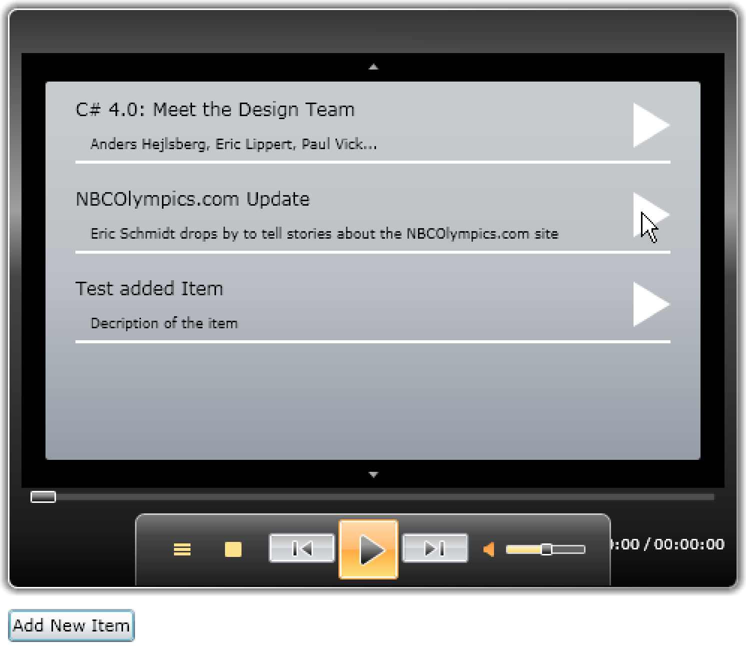
3) The site logo and title should appear at the head of the page. Passing the mouse over the logo image should display a tool tip. Clicking the logo should bring up a separate browser with the Hubble site.



Discover the marvels of the universe with a Hubble scientist, and find constellations from the view of your ba
Hubble's Universe brings these and many other videos to your screen.

26.6 Customization

You can customize the player list by overriding the ControlTemplate of the RadMediaItem. The screenshot below shows a slightly different arrangement of elements with a triangular "Play" button at the right of each item.



C# 4.0: Meet the Design Team

Anders Hejlsberg, Eric Lippert, Paul Vick...

NBCOlympics.com Update

Eric Schmidt drops by to tell stories about the NBCOlympics.com site

Test added Item

Decription of the item

The XAML for the template can be defined in the User.Resources or other resource area. Here the template is named "MediaItemTemplate" for reference later in the RadMediaItem itself. The ControlTemplate contains a series of Grids that arrange bound elements. TextBlock elements are bound to the Title and Description. The button is named "PlayButton" to correspond to the media player element of the same name. In its own ControlTemplate, the ContentPresenter surfaces the original button functionality so that when its clicked, the video plays. The Path element simply creates the triangular area for the button.

```
<ControlTemplate TargetType="telerik:RadMediaItem"
       x:Key="MediaItemTemplate">
  <Grid x:Name="LayoutRoot"
      MinHeight="{TemplateBinding MinHeight}">
    <VisualStateManager.VisualStateGroups>
      <VisualStateGroup x:Name="CommonStates">
        <VisualState x:Name="Normal" />
        <VisualState x:Name="MouseOver">
          <Storyboard>
            <ObjectAnimationUsingKeyFrames BeginTime="00:00:00"
                            Duration="00:00:00.0010000"
                            Storyboard.TargetProperty="(UIElement.Visibility)"
                            Storyboard.TargetName="MouseOverVisual">
              <DiscreteObjectKeyFrame KeyTime="0"
                          Value="Visible" />
            </ObjectAnimationUsingKeyFrames>
          </Storyboard>
        </VisualState>
        <VisualState x:Name="Disabled">
          <Storyboard Duration="0">
            <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Visibility"
                            Storyboard.TargetName="DisabledVisual">
              <DiscreteObjectKeyFrame KeyTime="0"
                          Value="Visible" />
            </ObjectAnimationUsingKeyFrames>
          </Storyboard>
        </VisualState>
      </VisualStateGroup>
      <VisualStateGroup x:Name="SelectionStates">
        <VisualState x:Name="NotSelected" />
        <VisualState x:Name="Selected">
          <Storyboard>
            <ObjectAnimationUsingKeyFrames BeginTime="00:00:00"
                            Duration="00:00:00.0010000"
                            Storyboard.TargetProperty="(UIElement.Visibility)"
                            Storyboard.TargetName="SelectedVisual">
              <DiscreteObjectKeyFrame KeyTime="0"
                          Value="Visible" />
            </ObjectAnimationUsingKeyFrames>
          </Storyboard>
        </VisualState>
      </VisualStateGroup>
    </VisualStateManager.VisualStateGroups>
    <ItemsPresenter x:Name="itemsPresenter"
            Margin="10 0"
            Opacity="0" />
    <Border BorderBrush="{TemplateBinding BorderBrush}"
        BorderThickness="{TemplateBinding BorderThickness}"
        Background="{TemplateBinding Background}"
        CornerRadius="{StaticResource MediaItem_CornerRadius}" />
    <Border x:Name="MouseOverVisual"
        BorderBrush="{StaticResource MediaItem_Border_MouseOver}"
        BorderThickness="{TemplateBinding BorderThickness}"
        Background="{StaticResource MediaItem_Background_MouseOver}"
        CornerRadius="{StaticResource MediaItem_CornerRadius}"
        Visibility="Collapsed" />
    <Border x:Name="SelectedVisual"
        BorderBrush="{StaticResource MediaItem_Border_Selected}"
        BorderThickness="{TemplateBinding BorderThickness}"
```

When declaring each of the RadMediaItem elements in the RadMediaPlayer, the Template property simply needs to be pointed at the ControlTemplate "MediaItemTemplate" resource you declared earlier.

XAML

```
<telerik:RadMediaItem
      Template="{StaticResource MediaItemTemplate}"
. . ./>
```

You can also add new RadMediaItem instances to the collection. The additional step is to extract the ControlTemplate from the resource file and assign it to the Template property in code, as shown below.

VB

```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim mediaSource As String = "http://msstudios.vo.llnwd.net/o21/mix08/08_WMVs/BCT06.wmv"
  Dim mediaItemCustomTemplate As ControlTemplate = _
TryCast(Me.Resources("MediaItemTemplate"), ControlTemplate)
  Dim item = New RadMediaItem()
  item.Template = mediaItemCustomTemplate
  item.Source = New Uri(mediaSource, UriKind.RelativeOrAbsolute)
  item.Title = "Test added Item"
  item.Description = "Decription of the item"
  radMediaPlayer1.Items.Add(item)
End Sub
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
   string mediaSource =  @"http://msstudios.vo.llnwd.net/o21/mix08/08_WMVs/BCT06.wmv";
   ControlTemplate mediaItemCustomTemplate =  this.Resources["MediaItemTemplate"] as ControlTemplate
   var item = new RadMediaItem();
   item.Template = mediaItemCustomTemplate;
   item.Source = new Uri(mediaSource, UriKind.RelativeOrAbsolute);
   item.Title = "Test added Item";
   item.Description = "Decription of the item";
   radMediaPlayer1.Items.Add(item);
}
```

26.7 Wrap Up

In this chapter you learned how to incorporate media into your Silverlight applications. You first built a play list using static XAML and in the process learned how to define RadMediaItem elements. You learned about the media types supported by the media player, how to work with video size and full screen, how to add chapters to a media item and about the available events for the media player. You constructed an application where the media player consumed data from an RSS service including the titles, descriptions, images and the video itself. Finally, you learned how to create a play list with a unique appearance using a custom template.

Part XV

CoverFlow

27 CoverFlow

27.1 Objectives

In this chapter you will learn how to configure RadCoverFlow to include a set of items. In the process you will use properties to control the coverflow position, the position of the "camera" in relation to the items item rotation and item scale. You will learn how to bind lists of images, videos and Silverlight elements to the coverflow control. Finally, you will customize the coverflow navigation panel to display a RadSlider instead of a scrollbar.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Coverflow\Coverflow.sln.

27.2 Overview

RadCoverFlow turns media navigation into a dazzling visual experience. RadCoverFlow uses real 3D transitions to navigate through the items. Users can flip through the list of images intuitively by selecting images with the mouse, rolling the mouse wheel or using arrow and page keys. RadCoverFlow can display a series of images, videos or any Silverlight element. Feel free to configure camera position, item rotations or position.

RadCoverFlow features include:

- Item Source and Databinding
- Configure Items Reflection and Camera Position
- Real 3D Rotation
- Customizable Navigation Template

27.3 Getting Started

Although the visual effect is stunning, RadCoverFlow is essentially a ListBox with a special ItemsPanel. RadCoverFlow can automatically handle a series of images or videos as well as any Silverlight element. In this walk through we will add a series of images and set the camera viewpoint.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

 a) **Telerik.Windows.Controls**

 b) **Telerik.Windows.Controls.Navigation**

4) From the Solution Explorer, right-click the project and select **Add > New Folder** from the context menu. Name the folder "Images".

5) Add five images named "wave1.jpg", "wave2.jpg", "wave3.jpg", "wave4.jpg" and "wave5.jpg" to the "Images" folder. These images can be found in the "\courseware\images" directory.

XAML Editing

1) Open MainPage.xaml for editing.

2) Add a **UserControl.Resources** element with a **Style** defined against the Image type. The style will be applied to all images within the cover flow.

```
<UserControl.Resources>
  <Style x:Key="ImageStyle" TargetType="Image">
    <Setter Property="Width" Value="150" />
    <Setter Property="Height" Value="100" />
    <Setter Property="Stretch" Value="Uniform" />
  </Style>
</UserControl.Resources>
```

3) From the Toolbox, drag a **RadCoverFlow** control to a point within the main "LayoutRoot" Grid element. Set the **CameraViewpoint** property to "Top"

4) Inside the RadCoverFlow element tag, add five Image elements that use "ImageStyle" and with the following Source paths:

 a) "images/wave1.jpg"

 b) "images/wave2.jpg"

 c) "images/wave3.jpg"

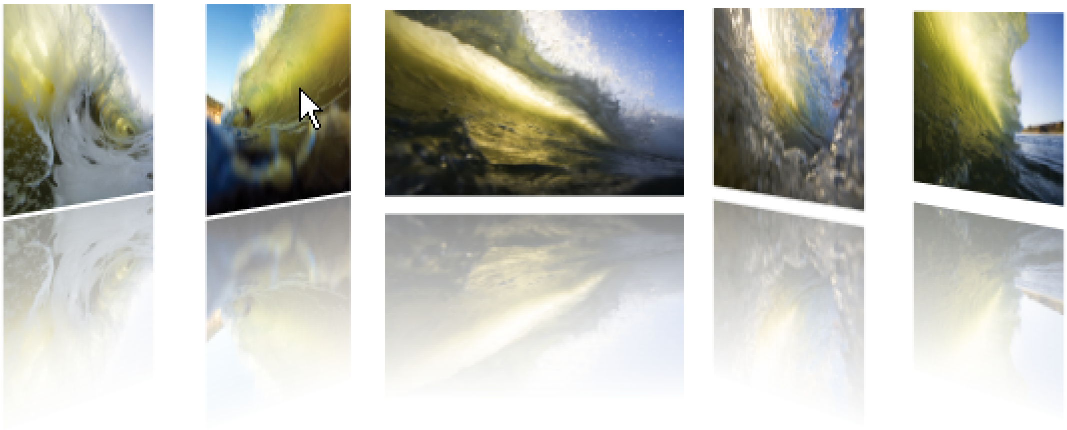
 d) "images/wave4.jpg"

 e) "images/wave5.jpg"

```
<telerik:RadCoverFlow CameraViewpoint="Top">
   <Image Source="images/wave1.jpg" Style="{StaticResource ImageStyle}" />
   <Image Source="images/wave2.jpg" Style="{StaticResource ImageStyle}" />
   <Image Source="images/wave3.jpg" Style="{StaticResource ImageStyle}" />
   <Image Source="images/wave4.jpg" Style="{StaticResource ImageStyle}" />
   <Image Source="images/wave5.jpg" Style="{StaticResource ImageStyle}" />
</telerik:RadCoverFlow>
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

1) All five photos should be displayed.

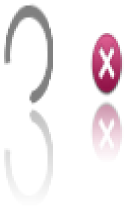
2) Click the background photos or use the mouse wheel to navigate between images.

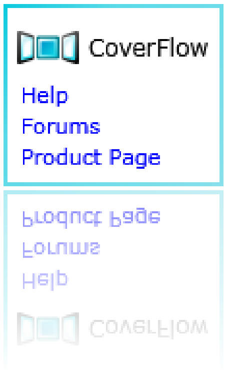
27.4 Control Details

27.4.1 Items

RadCoverflowItem

RadCoverflowItem is a ListBoxItem descendant that has Boolean properties that signal if the item **IsLoading** (animates the image on the left in the screenshot) and **IsContentValid** (displays the image on the right).



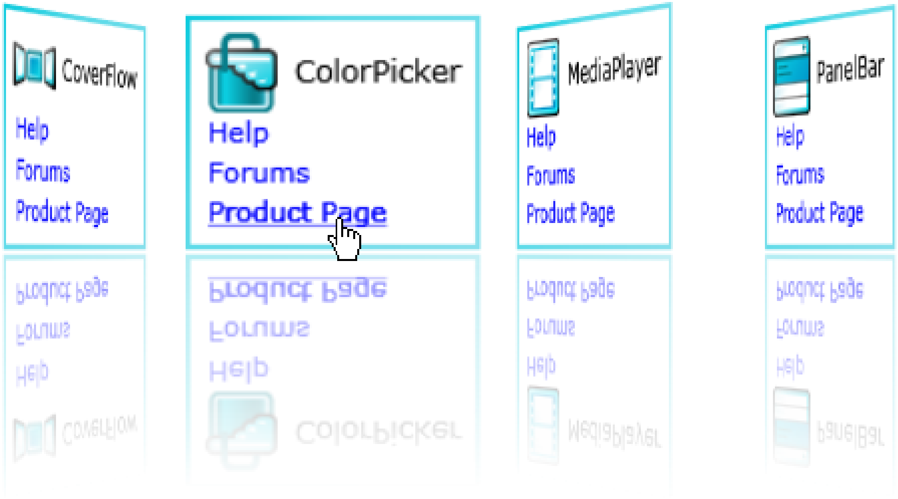


Other than that you can work with RadCoverflowItem in much the same way as ListBoxItem including assigning content and binding to items. RadCoverFlowItem content can contain arrangements of elements of any arbitrary complexity. The example here has a single RadCoverFlowItem that contains a StackPanel in its Content element. The StackPanel holds images, text blocks and hyperlinks.

Note that the styles for this example are not included in the listing. The idea here is to show the flexibility of RadCoverFlowItem and that you're not limited as to amount or type of content.

```
<telerik:RadCoverFlow>
  <telerik:RadCoverFlowItem Width="200" Height="120"
      Background="{StaticResource BackgroundBrush}">
    <telerik:RadCoverFlowItem.Content>
      <StackPanel Margin="5" HorizontalAlignment="Center" VerticalAlignment="Center">
        <StackPanel Orientation="Horizontal">
          <Image Source="images/CoverFlow.png" Style="{StaticResource ImageStyle}"
           telerik:RadCoverFlow.EnableLoadNotification="True"/>
          <TextBlock Text="RadCoverFlow" Style="{StaticResource TitleStyle}" />
        </StackPanel>
        <HyperlinkButton Content="Help"
            NavigateUri="http://www.telerik.com/help/silverlight/introduction.html"
            Style="{StaticResource HyperlinkStyle}" />
        <HyperlinkButton Content="Forums"
            NavigateUri="http://www.telerik.com/community/forums"
            Style="{StaticResource HyperlinkStyle}" />
        <HyperlinkButton Content="Product Page"
            NavigateUri="http://www.telerik.com/products/silverlight"
            Style="{StaticResource HyperlinkStyle}" />
      </StackPanel>
    </telerik:RadCoverFlowItem.Content>
  </telerik:RadCoverFlowItem>
</telerik:RadCoverFlow>
```

What if I want to add items for "RadColorPicker" or "RadMediaPlayer" content? To scale this up nicely you can bind data to the cover flow and use binding expressions in an ItemTemplate.

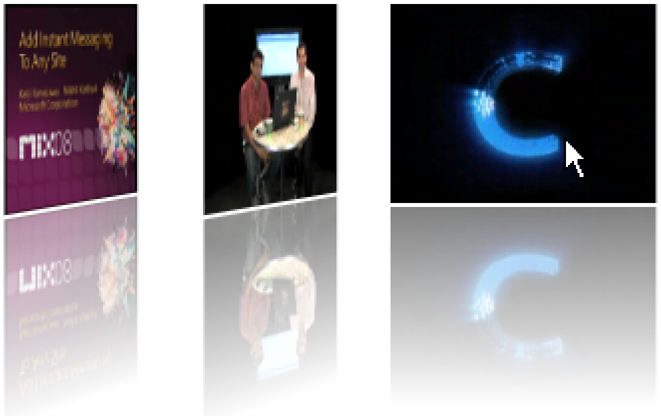


The example uses a collection of "ControlInfo" objects that contain the name, help path, logo image path, etc. and binds the collection to the RadCoverFlow ItemsSource. These are bound to the title TextBlock, logo image and HyperlinkButton controls in the template. For a walk through on binding to the RadCoverFlow, see the Binding section of this chapter.

```
<telerik:RadCoverFlow
    ItemsSource="{StaticResource ControlInfoList}">
  <telerik:RadCoverFlow.ItemTemplate>
    <DataTemplate>
      <Border BorderBrush="{StaticResource BorderBrush}" BorderThickness="2">
        <StackPanel Style="{StaticResource PanelStyle}">
          <StackPanel Orientation="Horizontal">
            <Image Source="{Binding Logo}" Style="{StaticResource ImageStyle}" />
            <TextBlock Text="{Binding Name}" Style="{StaticResource TitleStyle}" />
          </StackPanel>
          <HyperlinkButton Content="Help" NavigateUri="{Binding Help}"
              Style="{StaticResource HyperlinkStyle}" />
          <HyperlinkButton Content="Forums" NavigateUri="{Binding Forums}"
              Style="{StaticResource HyperlinkStyle}" />
          <HyperlinkButton Content="Product Page" NavigateUri="{Binding Product}"
              Style="{StaticResource HyperlinkStyle}" />
        </StackPanel>
      </Border>
    </DataTemplate>
  </telerik:RadCoverFlow.ItemTemplate>
</telerik:RadCoverFlow>
```

Video

In the Getting Started section you saw that images can be added as items in the RadCoverFlow markup. You can also load video content using the standard **MediaElement** as items. The MediaElement can be played in response to selecting the item. When the video plays, even the reflection stays in sync with the content.



The example below includes several MediaElement items with **AutoPlay** properties turned off. Because CoverFlow is a ListBox descendant, you can use the **SelectionChanged** event to know when the user navigates through the items.

```
<UserControl.Resources>
   <Style x:Key="MediaElementStyle" TargetType="MediaElement" >
      <Setter Property="Width" Value="150" />
      <Setter Property="Height" Value="100" />
      <Setter Property="AutoPlay" Value="False" />
   </Style>
</UserControl.Resources>

<StackPanel x:Name="LayoutRoot">

   <telerik:RadCoverFlow x:Name="coverFlow"
         CameraViewpoint="Top" SelectionChanged="coverFlow_SelectionChanged">
      <MediaElement Style="{StaticResource MediaElementStyle}"
       Source="http://msstudios.vo.llnwd.net/o21/mix08/08_WMVs/T03.wmv" />
      <MediaElement Style="{StaticResource MediaElementStyle}"
       Source="http://mschnlnine.vo.llnwd.net/d1/ch9/7/1/5/1/2/4/DynamicsDuoCRMSilverlight_ch9.wmv" />
      <MediaElement Style="{StaticResource MediaElementStyle}"
        Source="http://mschnlnine.vo.llnwd.net/d1/ch9/6/7/3/8/1/4/BTCRebeccaNorlander_ch9.wmv" />
   </telerik:RadCoverFlow>

</StackPanel>
```

The SelectionChanged event handler stops any currently playing video, starts the currently selected item and stores the current element so we can stop the video playing the next time the event fires.

VB

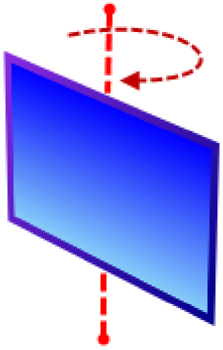
```
Private currentElement As MediaElement = Nothing

Private Sub coverFlow_SelectionChanged(ByVal sender As Object, _
 ByVal e As SelectionChangedEventArgs)
   ' stop the last video
   If currentElement IsNot Nothing Then
      currentElement.Stop()
   End If
   '  play the selected video
   currentElement = TryCast(coverFlow.SelectedItem, MediaElement)
   currentElement.Play()
End Sub
```

C#

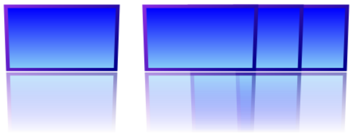
```
private MediaElement currentElement = null;

private void coverFlow_SelectionChanged(object sender,
    SelectionChangedEventArgs e)
{
    // stop the last video
    if (currentElement != null)
    {
        currentElement.Stop();
    }
    // play the selected video
    currentElement =
        coverFlow.SelectedItem as MediaElement;
    currentElement.Play();
}
```

27.4.2 Item Properties

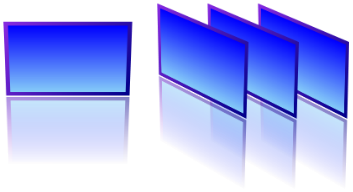
The **RotationY** property rotates each non-selected item around the Y-axis.



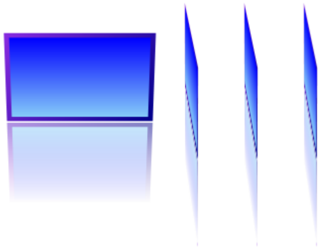
The series of screenshots below show the effect of increasing RotationY values when there are several items present.



RotationY = "0"

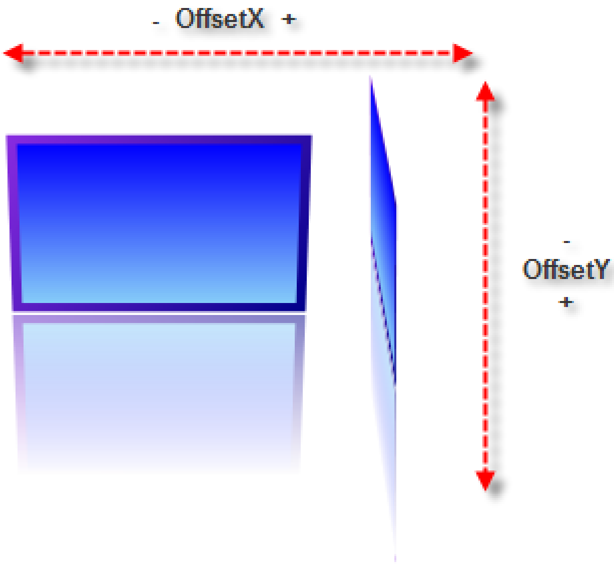


RotationY = "45"

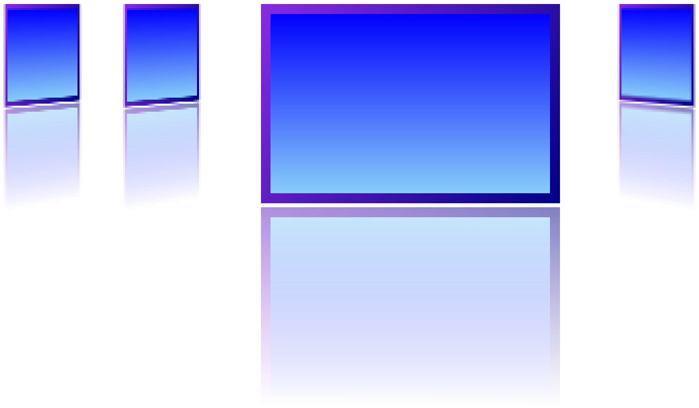


RotationY = "85"

Use **OffsetX** and **OffsetY** properties to move cover flow items horizontally and vertically. OffsetX moves cover flow items horizontally; with larger values shifting the items to the right. OffsetY moves items vertically with larger values pushing the items down.

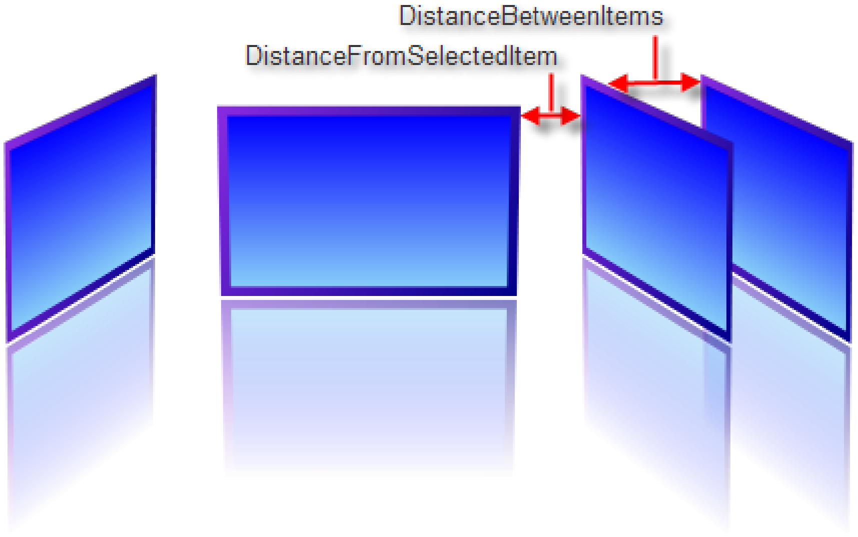


ItemScale scales non-selected items. The screenshot shows the result when ItemScale is "0.5", i.e. 50% of the original size.



27.4.3 Distance

Control the spacing between items using the **DistanceFromSelectedItem** (distance between the selected item and non-selected items) and the **DistanceBetweenItems** properties.

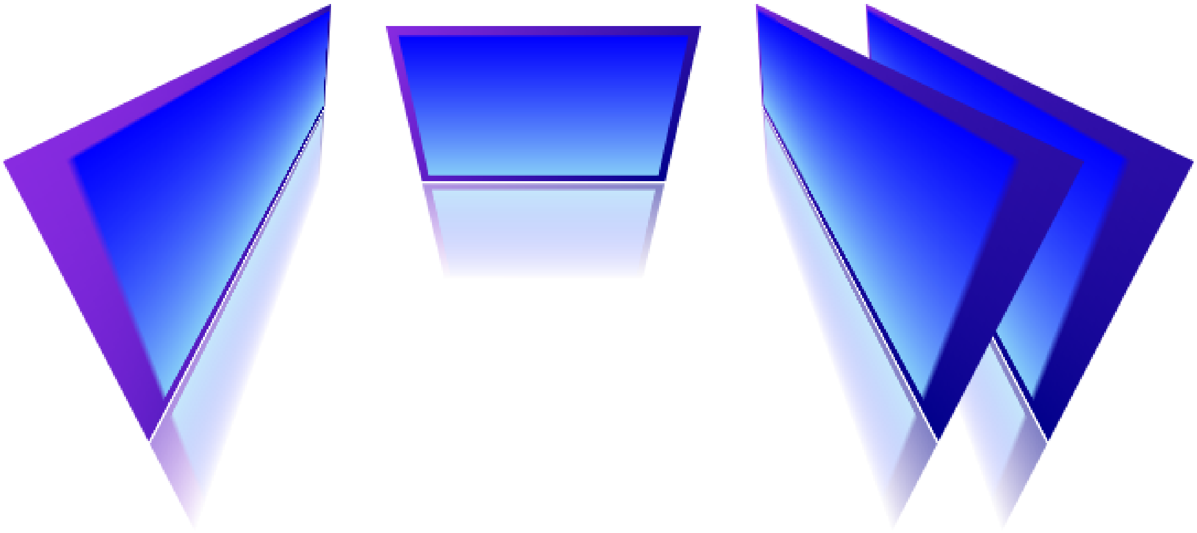


The screenshot above was produced using the settings in the XAML below:

```
<telerik:RadCoverFlow DistanceBetweenItems="60" DistanceFromSelectedItem="30" . . ./>
```

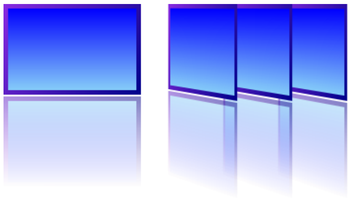
27.4.4 Camera

The "camera" is the viewpoint of the observer when looking at cover flow items in 3D space. The **CameraDistance** is the simulated "Z" position of the camera where smaller values bring the camera closer to the items and larger values move the camera farther away. As the camera draws closer to the cover flow, the perspective effect on the items becomes more exaggerated. By default, the CameraDistance is "1000" (at the time of this writing). The screenshot below actually has the same settings as the previous example for "Distance" but where the CameraDistance property value is "100".

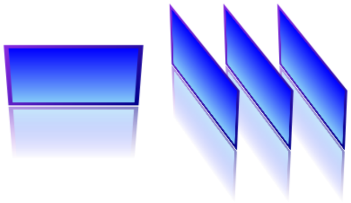


```
<telerik:RadCoverFlow CameraDistance="100" CameraViewpoint="Top"  ... >
```

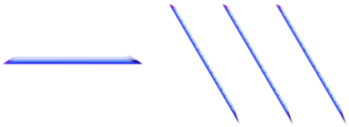
CameraRotation determines the view angle toward the control where a "0" angle represents the camera pointed straight on and larger values roll the camera angle. The screenshots below should give you an idea of how this works.



CameraRotation = "0"



CameraRotation = "45"



CameraRotation = "90"

27.4.5 Animation

When an item is selected in the coverflow, an animation plays that rearranges the items and the selected item is moved to the front. The speed of the animation is controlled by the **ItemChangeDelay**, a TimeSpan property that is currently "600" milliseconds by default. The **EasingFunction** provides a profile for the animation to follow, giving the animation a more realistic feel. At the time of this writing, the EasingFunction defaults to a "Circle".

27.4.6 Reflection

IsReflectionEnabled toggles the appearance of reflection below the cover flow items.

27.5 Binding

This next walk through demonstrates retrieving a series of images from the "Flickr" REST API and binding the images to the coverflow.

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) Telerik.Controls

b) Telerik.Controls.Navigation

4) Using the Solution Explorer, right-click the project and select **Add > New Folder** from the context menu. Name the folder "Images".

5) Drag an image file "search.png" to the "Images" folder. You can find this file in the "\courseware\images" directory.

XAML Editing

1) Open MainPage.xaml for editing.

2) Add a UserControl.Resources element just above the main "LayoutRoot" Grid element. These resources style text, borders and panels, but are not central to learning about binding RadCoverFlow. Feel free to copy and paste this XAML and review it at your leisure.

```
<UserControl.Resources>
  <!--colors-->
  <Color x:Key="FadedWhite">#AAFFFFFF</Color>
  <Color x:Key="FadedBlue">#AA0000FF</Color>
  <Color x:Key="FadedLightBlue">#AA000099</Color>
  <!--brushes-->
  <LinearGradientBrush x:Key="SkyBrush" StartPoint="0, 0"
      EndPoint="0, 1">
    <GradientStop Color="SkyBlue" Offset="0" />
    <GradientStop Color="{StaticResource FadedLightBlue}" Offset="1" />
  </LinearGradientBrush>
  <LinearGradientBrush x:Key="SkyBrushStreak">
    <GradientStop Color="SkyBlue" Offset="0" />
    <GradientStop Color="{StaticResource FadedWhite}" Offset="0.2" />
    <GradientStop Color="{StaticResource FadedBlue}" Offset=".5" />
    <GradientStop Color="{StaticResource FadedLightBlue}" Offset="1" />
  </LinearGradientBrush>
  <Style x:Key="CaptionStyle" TargetType="TextBlock">
    <Setter Property="VerticalAlignment" Value="Center" />
    <Setter Property="FontFamily" Value="Comic Sans MS" />
    <Setter Property="FontSize" Value="12" />
  </Style>
  <!--styles-->
  <Style x:Key="HyperlinkStyle" TargetType="HyperlinkButton">
    <Setter Property="TargetName" Value="_blank" />
    <Setter Property="Foreground" Value="Blue" />
  </Style>
  <Style x:Key="ImageStyle" TargetType="Image">
    <Setter Property="Stretch" Value="Uniform" />
    <Setter Property="VerticalAlignment" Value="Center" />
    <Setter Property="Width" Value="150" />
    <Setter Property="Height" Value="150" />
    <Setter Property="Margin" Value="5" />
  </Style>
  <Style x:Key="PictureFrameStyle" TargetType="Border">
    <Setter Property="BorderBrush" Value="{StaticResource SkyBrush}" />
    <Setter Property="BorderThickness" Value="1" />
    <Setter Property="Background" Value="{StaticResource SkyBrushStreak}"/>
    <Setter Property="Padding" Value="10" />
  </Style>
</UserControl.Resources>
```

3) Replace the main "LayoutRoot" Grid with the XAML below.

This will setup our basic layout. The main Grid has two rows configured so that the first row will size itself to the elements it contains and the second row will expand to take any available space.

The top "tool bar" row will have its own grid with two columns. The first column on the left will size itself to the elements it contains. This first column will hold a HyperlinkButton with an Image of the site logo. The button will navigate to the main Flickr site and have a "powered by Flickr" tool tip. The rightmost column will take up the remaining width that contains a TextBox for search criteria and a search button. Underneath the "tool bar, the RadCoverFlow will take must of the client area.

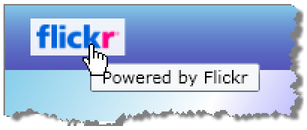
```
<Grid x:Name="LayoutRoot">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>

  <!--top tool bar-->
  <Border Background="{StaticResource SkyBrush}" Padding="10"
      Grid.Row="0">
    <Grid>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
      </Grid.ColumnDefinitions>

      <!--flickr logo-->

      <!--search panel-->

    </Grid>
  </Border>

   <!--cover flow-->

</Grid>
```

4) Replace the "<!--flickr logo-->" comment with the XAML below.

```
<!--flickr logo-->
<HyperlinkButton Grid.Column="0" Margin="5 0 0 0"
      NavigateUri="http://www.flickr.com"
      ToolTipService.ToolTip="Powered by Flickr"
      TargetName="_blank">
   <Image Stretch="Uniform" Width="64"
         Source="http://l.yimg.com/g/images/logo_home.png.v2" />
</HyperlinkButton>
```

Notes

The XAML defines a HyperlinkButton that in turn holds a logo image. The HyperlinkButton NavigateUri points to the main Flickr site and will open in a new window by virtue of the TargetName = "_blank" property setting. The HyperlinkButton also has the tooltip "Powered by Flickr" defined. The Image Source is hard coded.

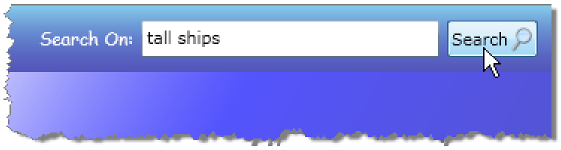


5) Replace the "<!--search panel-->" comment with the XAML below.

```
<!--search panel-->
<StackPanel Orientation="Horizontal"
      HorizontalAlignment="Right" Grid.Column="1">
   <TextBlock Text="Search On:"
         Style="{StaticResource CaptionStyle}"
         Foreground="Black" />
   <TextBox x:Name="tbSearch" MinWidth="200"
         Margin="5, 0, 5, 0" />
   <Button x:Name="btnGo" Click="btnGo_Click">
      <Button.Content>
         <StackPanel Orientation="Horizontal">
            <TextBlock Text="Search" />
            <Image Source="images/search.png"
                  Width="16" Height="16"></Image>
         </StackPanel>
      </Button.Content>
   </Button>
</StackPanel>
```

Notes

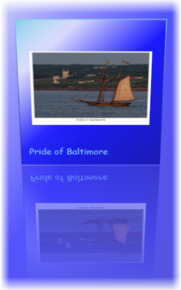
The XAML defines a horizontal StackPanel with all the elements used for searching. Points to notice: The "Search On" TextBlock uses one of the Styles we defined earlier. The magnifying glass image uses "search.png" that we added to the project early on. A handler is defined for the button's Click event where the main search logic takes place.


Search On: tall ships

6) Replace the "<!--cover flow-->" comment with the XAML below.

```
<!--cover flow-->
<Border Background="{StaticResource SkyBrushStreak}"
    Grid.Row="1">
  <telerik:RadCoverFlow x:Name="coverFlow" OffsetY="60" RotationY="45"
    CameraRotation="30" DistanceBetweenItems="30">
    <telerik:RadCoverFlow.ItemTemplate>
      <DataTemplate>
        <Border Style="{StaticResource PictureFrameStyle}">
          <StackPanel>
            <Image Source="{Binding ImageUrl}" Style="{StaticResource ImageStyle}" />
            <TextBlock Text="{Binding ImageTitle}" Style="{StaticResource CaptionStyle}"
                Foreground="LightSkyBlue" />
          </StackPanel>
        </Border>
      </DataTemplate>
    </telerik:RadCoverFlow.ItemTemplate>
  </telerik:RadCoverFlow>
</Border>
```

Notes



The coverflow is contained within a Border element to provide the gradient background. The coverflow ItemTemplate provides the layout for coverflow item. Each item is bounded by a slender border that contains a StackPanel. The StackPanel holds the Flickr Image and the TextBlock below displays the Flickr Title. We will take care of the binding in upcoming steps.

Code Behind

1) In the Solution Explorer, right-click the project and select **Add > Class...** from the context menu. Name the class file "FlickrItem.cs". Replace the code for the Flickr class with the code below.

This class will encapsulate a single image returned from the Flickr REST service. The "ImageUrl" will be added to a collection and assigned to the coverflow ItemsSource property.

```
Public Class FlickrItem
  Implements INotifyPropertyChanged
  Public Event PropertyChanged As PropertyChangedEventHandler

  Private imageTitle_Renamed, imageUrl_Renamed As String

  Public Property ImageTitle() As String
    Get
      Return Me.imageTitle_Renamed
    End Get
    Set(ByVal value As String)
      If value <> Me.imageTitle_Renamed Then
        Me.imageTitle_Renamed = value
        Me.FirePropertyChanged("ImageTitle")
      End If
    End Set
  End Property

  Public Property ImageUrl() As String
    Get
      Return Me.imageUrl_Renamed
    End Get
    Set(ByVal value As String)
      If value <> Me.imageUrl_Renamed Then
        Me.imageUrl_Renamed = value
        Me.FirePropertyChanged("ImageUrl")
      End If
    End Set
  End Property

  Private Sub FirePropertyChanged(ByVal propertyName As String)
    If Me.PropertyChangedEvent IsNot Nothing Then
      RaiseEvent PropertyChanged(Me, New PropertyChangedEventArgs(propertyName))
    End If
  End Sub
End Class
```

```
public class FlickrItem : INotifyPropertyChanged
{
   public event PropertyChangedEventHandler PropertyChanged;

   private string imageTitle, imageUrl;

   public string ImageTitle
   {
      get { return this.imageTitle; }
      set
      {
         if (value != this.imageTitle)
         {
            this.imageTitle = value;
            this.FirePropertyChanged("ImageTitle");
         }
      }
   }

   public string ImageUrl
   {
      get { return this.imageUrl; }
      set
      {
         if (value != this.imageUrl)
         {
            this.imageUrl = value;
            this.FirePropertyChanged("ImageUrl");
         }
      }
   }

   private void FirePropertyChanged(string propertyName)
   {
      if (this.PropertyChanged != null)
      {
         this.PropertyChanged(this,
            new PropertyChangedEventArgs(propertyName));
      }
   }
}
```

2) Add a second class file and name it "Flickr.cs". Add namespace references to the "Imports" (VB) or "using" (C#) section of code for these assemblies:

a) **System.Collections.Generic**

b) **System.Linq**

c) **System.Net**

d) **System.Xml.Linq**

3) Replace the "Flickr.cs" class definition with the code below.

```
Public Class Flickr
  Public Delegate Sub LoadedEventHandler(ByVal sender As Object, ByVal e As FlickrLoadedEventArgs)

  Public Event Loaded As LoadedEventHandler

  Public Sub Load(ByVal searchString As String)
    Const url As String = "http://api.flickr.com/services/feeds/photos_public.gne?tags="

    Dim client As New WebClient()
    AddHandler client.DownloadStringCompleted, AddressOf client_DownloadStringCompleted
    client.DownloadStringAsync(New Uri(url & searchString))
  End Sub

  Private Sub client_DownloadStringCompleted( _
ByVal sender As Object, ByVal e As DownloadStringCompletedEventArgs)
    RaiseEvent Loaded(Me, New FlickrLoadedEventArgs() With {.Items = GetFlickrItems(e.Result)})
  End Sub

  Public Function GetFlickrItems(ByVal xml As String) As IEnumerable(Of FlickrItem)
    Dim atomNameSpace As XNamespace = "http://www.w3.org/2005/Atom"
    Dim feed As XDocument = XDocument.Parse(xml)
    Dim result = _
      From e In feed.Root.Elements(atomNameSpace + "entry") _
      Let link = ( _
        From l In e.Elements(atomNameSpace + "link") _
        Where l.Attribute("rel").Value.Equals("enclosure") _
        Select l).FirstOrDefault() _
      Select New FlickrItem()
        e.Element(atomNameSpace + "title").Value, ImageUrl = link.Attribute("href").Value
        ImageTitle = e.Element(atomNameSpace + "title").Value, ImageUrl
    Return result.Take(4)
  End Function
End Class

Public Class FlickrLoadedEventArgs
  Inherits EventArgs
  Private privateItems As IEnumerable(Of FlickrItem)
  Public Property Items() As IEnumerable(Of FlickrItem)
    Get
      Return privateItems
    End Get
    Set(ByVal value As IEnumerable(Of FlickrItem))
      privateItems = value
    End Set
  End Property
End Class
```

C#

```
public class Flickr
{
   public delegate void LoadedEventHandler(object sender, FlickrLoadedEventArgs e);
   public event LoadedEventHandler Loaded;

   public void Load(string searchString)
   {
      const string url = "http://api.flickr.com/services/feeds/photos_public.gne?tags=";

      WebClient client = new WebClient();
      client.DownloadStringCompleted +=
         new DownloadStringCompletedEventHandler(
            client_DownloadStringCompleted);
      client.DownloadStringAsync(new Uri(url + searchString));
   }

   void client_DownloadStringCompleted(object sender,
      DownloadStringCompletedEventArgs e)
   {
      if (Loaded != null)
      {
         Loaded(this, new FlickrLoadedEventArgs()
         { Items = GetFlickrItems(e.Result) });
      }
   }

   private IEnumerable<FlickrItem> GetFlickrItems(string xml)
   {
      XNamespace atomNameSpace = "http://www.w3.org/2005/Atom";
      XDocument feed = XDocument.Parse(xml);
      var result  =
         from e in feed.Root.Elements(atomNameSpace + "entry")
         let link =
            (from l in e.Elements(atomNameSpace + "link")
            where l.Attribute("rel").Value.Equals("enclosure")
            select l).FirstOrDefault()
         select new FlickrItem()
         {
            ImageTitle = e.Element(atomNameSpace + "title").Value,
            ImageUrl = link.Attribute("href").Value
         };
      return result.Take(4);
   }
}

public class FlickrLoadedEventArgs : EventArgs
{
   public IEnumerable<FlickrItem> Items    {get; set;}
}
```

Notes

Most of the work is being done in the "Flickr" class. The point of entry for the Flickr class is the Load() method where a WebClient downloads an XML string from the Flickr REST service. The URL for the DownloadStringAsync() method call is formatted to include whatever search criteria the user has entered. When the call returns in the DownloadStringCompleted event handler a custom Loaded event is fired. The XML is converted to a collection of FlickrItem instances. We have custom arguments, FlickrLoadedEventArgs, that have a single property that contains the collection. The conversion from XML to collection occurs in the private GetFlickrItems() method where the XDocument Parse() method converts the raw XML into objects that can be sliced-and-diced in a LINQ statement. Notice that the Take() extension method returns just the top set of items.

4) In the code-behind for the main page, add a namespace reference to **Telerik.Windows.Controls**.

5) In the code-behind for the main page add the code below.

The Click event for the Search button, "btnGo_Click", creates a new Flickr instance, hooks up the custom Loaded event to a handler and calls Load(), passing the search text. After the Flickr class gets done with retrieving XML from the REST service and converting to a collection of FlickrItem instances, the Flickr class fires its Loaded event where the collection is assigned to the cover flow ItemsSource property.

VB

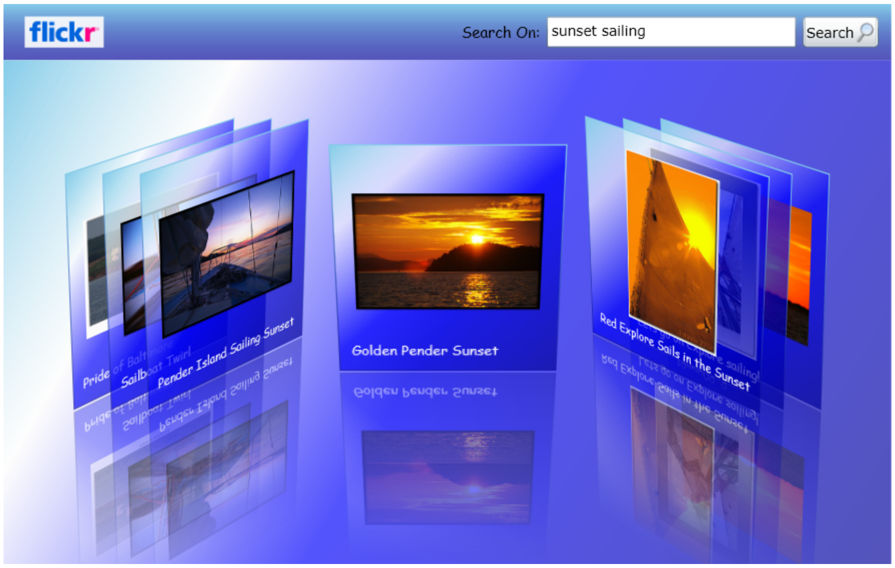
```
Private Sub btnGo_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
  Dim flickr As New Flickr()
  AddHandler flickr.Loaded, AddressOf flickr_Loaded
  flickr.Load(tbSearch.Text)
End Sub

Private Sub flickr_Loaded(ByVal sender As Object, ByVal e As FlickrLoadedEventArgs)
  coverFlow.ItemsSource = e.Items
End Sub
```

C#

```
private void btnGo_Click(object sender, RoutedEventArgs e)
{
   Flickr flickr = new Flickr();
   flickr.Loaded += new Flickr.LoadedEventHandler(flickr_Loaded);
   flickr.Load(tbSearch.Text);
}

void flickr_Loaded(object sender, FlickrLoadedEventArgs e)
{
   coverFlow.ItemsSource = e.Items;
}
```

Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



Test Application Features

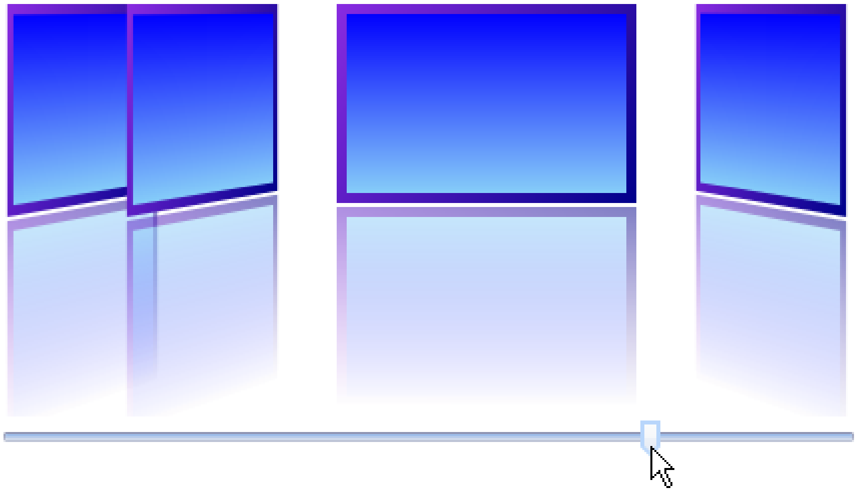
1) Enter search text and click the "Search" button.

2) Pass the mouse over the logo to see the tool tip and click to show the main site in a separate browser.

27.6 Customization

27.6.1 Navigation

RadCoverFlow does not have built-in navigation controls, but using Silverlight 3 element binding you can hook up your own using a slider, up-down, scroll bar or other element that navigates through a set of values. The screenshot below shows a RadSlider bound two-way with RadCoverFlow so that clicking the slider navigates the cover flow and clicking items in the cover flow reflects in the slider.



Navigation is implemented in the XAML below. Pay particular attention to the RadSlider **Value** and **Minimum** and **Maximum** properties. The key is to bind the navigation Value property to the RadCoverFlow **SelectedIndex** property. To completely "dial-in" the behavior you must set the **Minimum** and **Maximum** properties to match the count of items in the cover flow. The Minimum should be "0" and Maximum should be one less than the count of items in the cover flow. To assign the correct Maximum, you will need a simple value converter that takes the Count of items in the cover flow and subtracts one (code for the converter follows). To use the converter, be sure to add an XML namespace that references the assembly that contains the converter, add a resource that points to the converter and finally, use the resource reference in the binding statement for the Maximum attribute.

```
<UserControl x:Class="_02_ControlDetails.MainPage"
      xmlns:local="clr-namespace:<Your project assembly>" . . .>
   <UserControl.Resources>
      <local:IntToIntValueConverter x:Key="IntToIntValueConverter" /> . . .
   </UserControl.Resources>
   <StackPanel x:Name="LayoutRoot">
      <telerik:RadCoverFlow x:Name="coverFlow" >
         <telerik:RadCoverFlowItem  . . ./>
         <telerik:RadCoverFlowItem  . . ./>
         <telerik:RadCoverFlowItem  . . ./>
         <telerik:RadCoverFlowItem  . . ./>
         <telerik:RadCoverFlowItem  . . ./>
      </telerik:RadCoverFlow>
      <telerik:RadSlider
            Value="{Binding SelectedIndex, ElementName=coverFlow, Mode=TwoWay}"
            Minimum="0"
            Maximum="{Binding Items.Count, ElementName=coverFlow,
      Converter={StaticResource IntToIntValueConverter}, ConverterParameter=-1}" />
   </StackPanel>
</UserControl>
```

The IValueConverter implementation code adds the parameter passed in ("-1") and returns the result.

```
Public Class IntToIntValueConverter
  Implements IValueConverter
  Private Function IValueConverter_Convert(ByVal value As Object, ByVal targetType As Type, _
ByVal parameter As Object, _
ByVal culture As System.Globalization.CultureInfo) _
As Object Implements IValueConverter.Convert
    Dim val As Integer
    Dim param As Integer
    If (Integer.TryParse(value.ToString(), val) AndAlso _
(Integer.TryParse(parameter.ToString(), param))) Then
      Return val + param
    End If
    Return value
  End Function
 '. . .
End Class
```

C#

```
public class IntToIntValueConverter : IValueConverter
{
   object IValueConverter.Convert(object value, Type targetType,
     object parameter, System.Globalization.CultureInfo culture)
   {
      int val;
      int param;
      if ((int.TryParse(value.ToString(), out val) && (int.TryParse(parameter.ToString(), out param))))
      {
         return val + param;
      }
      return value;
   }
   //. . .
}
```

See the article "Navigation for Coverflow" at http://blogs.telerik.com/hristoborisov/posts/09-10-06/navigation_for_coverflow.aspx for more information.

27.7 Wrap Up

In this chapter you learned how to define RadCoverFlow to include a set of items. In the process you used properties to control the coverflow position, the position of the "camera" in relation to the items and item properties. You learned how to bind lists of images, videos and Silverlight elements to the coverflow control. Finally, you customized the coverflow navigation panel to use a RadSlider for navigation.

Part XVI

Upload

28 Upload

28.1 Objectives

In this chapter you will learn the minimal configuration needed to upload files from a Silverlight client to a server. You will learn the basics for creating the server upload handler along with some of the "Gotchas" that could cause the upload to fail.

In the section on "Control Details" you will become familiar with controlling file access on the client, including properties that limit file types and limiting the number of files and bytes allowed. You will also learn about the properties that show or hide buttons in the UI. You will create your own custom upload handler and learn how to pass parameters to and from the client. You will learn about the events that cover the entire upload life-cycle and the methods that can be used to trigger the file operations.

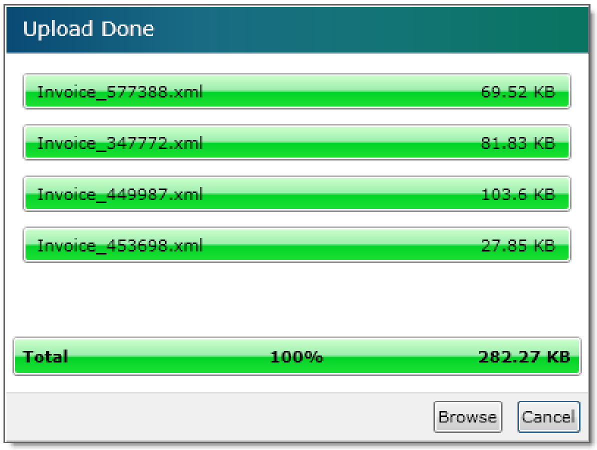
During the "Customization" section you will walk through using Expression Blend to modify the RadUpload dialog layout.

Find the projects for this chapter at...

\Courseware\Projects\<CS|VB>\Upload\Upload.sln.

28.2 Overview

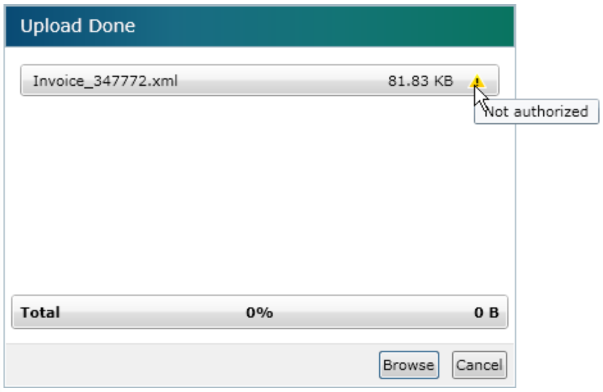
RadUpload saves end-user time and effort by allowing multiple files and automatic uploads. This dedicated file-upload control is a fast performer that allocates a minimum of server memory, while enabling optimized and fully configurable single and multi-file uploads. The server can automatically save to a folder in your project, or customize the server handler for fine-grained control, allowing you to add compression, additional security and saving to database or other persistence medium.



Upload Done

Invoice_577388.xml 69.52 KB
Invoice_347772.xml 81.83 KB
Invoice_449987.xml 103.6 KB
Invoice_453698.xml 27.85 KB

Total 100% 282.27 KB



Upload Done

Invoice_347772.xml 81.83 KB

Not authorized

Total 0% 0 B

RadUpload includes the following features:

- Multiple Files Upload
- Styling and Appearance
- Extension Filters
- Automatic Upload
- Files Count and Size Limitation

- Customizable Upload Handler

28.3 Getting Started

Project Setup

1) From the Visual Studio menu choose **File > New > Project...**, select the Silverlight project type, then select the Silverlight Application template. Provide a unique name for the project and click the **OK** button.

2) In the "New Silverlight Application" dialog make sure that the "Host the Silverlight application in a new Web site" option is checked, give the project a unique name and verify that the "ASP.NET Web Application Project" New Web Project Type option is selected. Click **OK** to close the dialog and create the project.

3) In the Solution Explorer, right-click the References node and select **Add References...** from the context menu. Add Assembly references:

a) **Telerik.Windows.Controls**

b) **Telerik.Windows.Controls.Input**

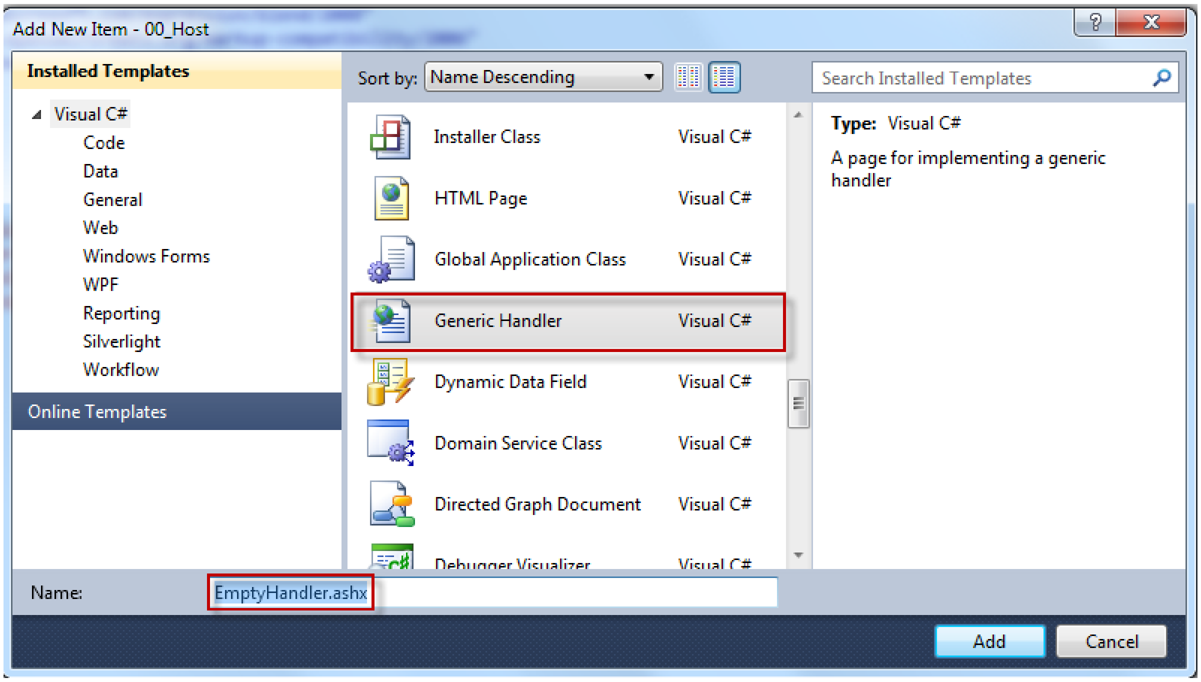
Create Upload Handler

1) In Solution Explorer, navigate to the host application project.

2) In the Solution Explorer, right-click the references node and add a reference to the **Telerik.Windows.RadUploadHandler** assembly.

3) In the Solution Explorer, right-click the project and select **Add > New Folder** from the context menu. Name the folder "MyFolder".

4) In the Solution Explorer, right-click and **Add > New Item...** Select the "Generic Handler" template and name it "EmptyHandler.asxh".


Add New Item - 00_Host
Installed Templates
Visual C#
Code
Data
General
Web
Windows Forms
WPF
Reporting
Silverlight
Workflow
Online Templates
Sort by: Name Descending
Search Installed Templates
Installer Class Visual C#
HTML Page Visual C#
Global Application Class Visual C#
Generic Handler Visual C#
Dynamic Data Field Visual C#
Domain Service Class Visual C#
Directed Graph Document Visual C#
Debugger Visualizer Visual C#
Add
Cancel

Type: Visual C#

A page for implementing a generic handler

Name: EmptyHandler.ashx

5) Navigate to the code-behind for the handler and replace the class declaration with the example that follows.

Be sure not to eliminate the namespace surrounding the class. The fully qualified class name must match the markup for the handler. Accidentally deleting the namespace is one way to generate a "Handler not found" error.

VB

```
Public Class EmptyHandler
  Inherits Telerik.Windows.RadUploadHandler
End Class
```

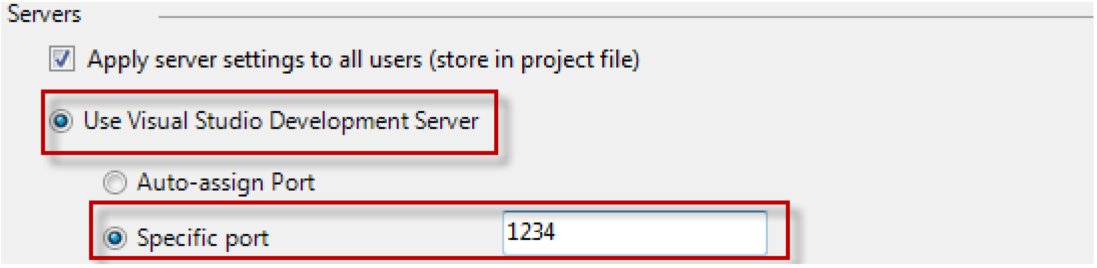
C#

```
public class EmptyHandler : Telerik.Windows.RadUploadHandler
{
}
```

Notes

You can code the upload handler to perform more specific tasks later, but for now, descending from RadUploadHandler is sufficient to get the base uploading functionality.

6) In the Solution Explorer, right-click the project node and select **Properties**. Select the **Web** tab. In the Servers section, make sure the "Use Visual Studio Development Server" radio button is selected and that the "Specific port" button is checked. Enter a specific port of "1234".



Servers

☑ Apply server settings to all users (store in project file)

● Use Visual Studio Development Server

○ Auto-assign Port

● Specific port 1234

Notes

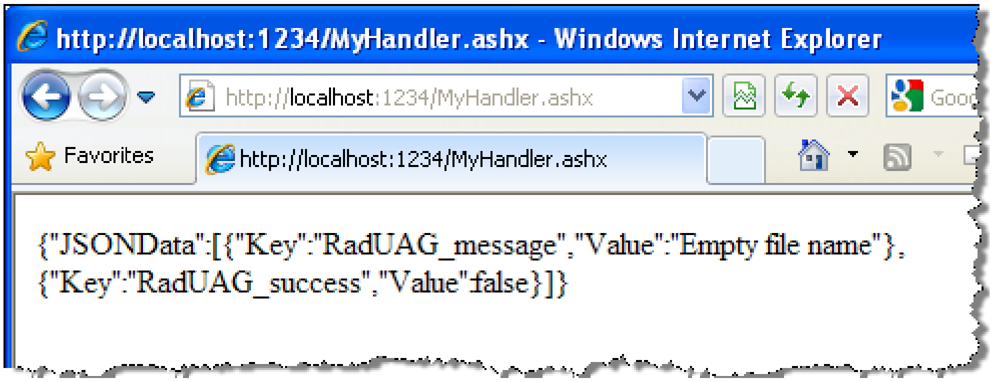
You could also select the "Use Local IIS Web Server" or host the handler in some other web server. The important point is to know the specific location of the handler so that we can refer to it in XAML markup later when we define the RadUpload control.

7) Test the handler:

a) Set the host project to be the startup project and set the page to be the default page for the project.

b) Run the host project.

c) In a new browser window, point the URL of another browser to "http://localhost:1234/EmptyHandler.ashx". You should see a JSON string as shown in the screenshot below. If you see an error message, you need to go back and check the steps leading up to this.


http://localhost:1234/MyHandler.ashx - Windows Internet Explorer
http://localhost:1234/MyHandler.ashx
Favorites
http://localhost:1234/MyHandler.ashx

{"JSONData":[{"Key":"RadUAG_message","Value":"Empty file name"},
{"Key":"RadUAG_success","Value"false}]}

XAML Editing

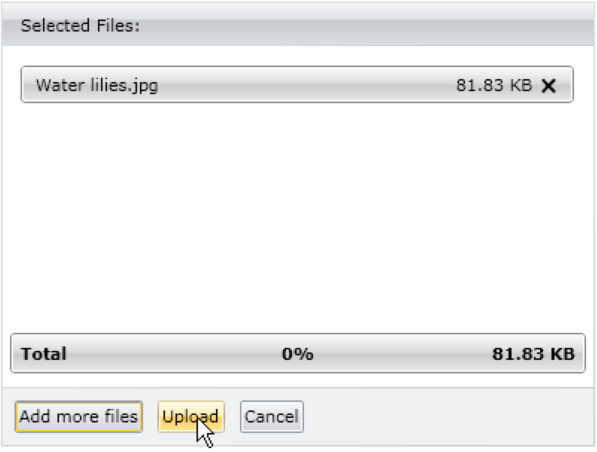
1) Open the Silverlight project MainPage.xaml for editing.

2) Add the XAML below to the main "LayoutRoot" Grid element. The XAML should be added between the <Grid> and </Grid> tags. The **UploadServiceUrl** must point to the exact path of the handler you defined in previous steps. **TargetFolder** must point to the exact name of the folder in the project you created earlier.

```
<telerik:RadUpload
    UploadServiceUrl="http://localhost:1234/EmptyHandler.ashx"
    TargetFolder="MyFolder"
    OverwriteExistingFiles="True"
    IsAutomaticUpload="false">
</telerik:RadUpload>
```

Run the Application

1) Press **F5** to run the application. The web page should look something like the screenshot below.

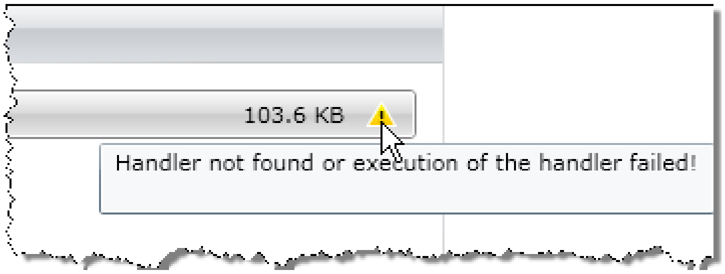


Gotcha!

"Handler not found or execution of the handler failed!"

This error and errors with similar wording can occur if...

- The handler address doesn't match the UploadServiceUrl
- The class name of the handler in code doesn't match the class name described in the handler's markup.
- The host application housing the handler is not running.



103.6 KB

Handler not found or execution of the handler failed!

Test Application Features

1) Click the "Browse" button and select a file to upload.

2) Click the "Upload" button and verify that the upload occurs without error.

3) Verify that the file you uploaded appears in "\MyFolder".

4) Click the "Add more files" button and add additional files.

5) Add a file, then click the "Cancel" button to return to the initial "Browse" state.

Ideas for Extending This Example

- Set the theme for the application, e.g., add Telerik.Windows.Controls and Telerik.Windows.Controls.<theme name>, then adding this code to the constructor:

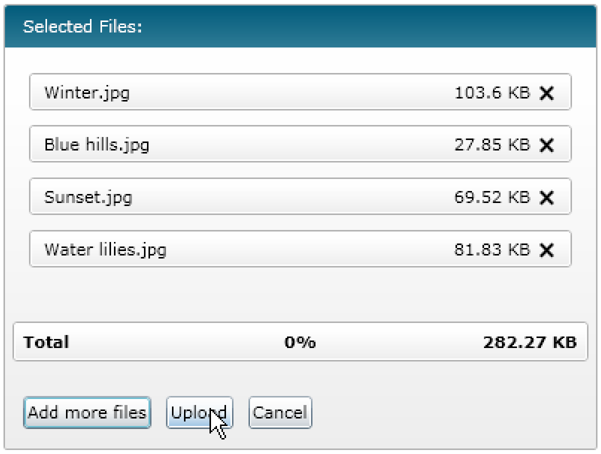
VB

```
Public Sub New()
  CType(New SummerTheme(), SummerTheme).IsApplicationTheme = True
  InitializeComponent()
End Sub
```

C#

```
public MainPage()
{
   new SummerTheme().IsApplicationTheme = true;
   InitializeComponent();
}
```

The upload dialog will be styled with the theme:


Selected Files:

Winter.jpg 103.6 KB

Blue hills.jpg 27.85 KB

Sunset.jpg 69.52 KB

Water lilies.jpg 81.83 KB

Total 0% 282.27 KB

28.4 Control Details

28.4.1 Controlling Upload Access

You can limit the files being uploaded based on several criteria and also receive notification from events:

- **File Size**: You can limit the size of a single file to a specific number of bytes by setting the **MaxFileSize** property. If the size is exceeded, the **FileTooLarge** event fires. The FileTooLarge event passes a parameter containing the name, size and all the file system information about the file in question.
- **Total File Size**: To limit the maximum total upload size in bytes, use the **MaxUploadSize** property. If this size is exceeded, the **UploadSizeExceeded** event fires.
- **Number of Files**: Limit the number of files using **MaxFileCount** properties. The **FileCountExceeded** event fires when the user attempts to upload more than MaxFileCount files.

You can also limit what files can be sent to the server by defining extension filters. **Filter** is a string property that determines the choices that appear in the "Open File Dialog" box. For each filtering option, the filter string contains a description of the filter, followed by the vertical bar (|) and the filter pattern. The strings for different filtering options are separated by the vertical bar. You can add several filter patterns to a filter by separating the file types with semicolons. Use the **FilterIndex** property to set which filtering option is shown first to the user. By default all file extensions are allowed.

Note: See http://msdn.microsoft.com/en-us/library/system.windows.controls.openfiledialog.filter(VS.95).aspx for specifics on filter syntax.

Hide buttons in the UI by setting the **IsAppendFilesEnabled**, **IsDeleteEnabled**, and **IsPauseEnabled** properties that control the corresponding named buttons. **IsMultiselect,** when true, allows more than one file to be selected in the "Open File Dialog" box. Turn on the **IsAutomaticUpload** property if you want the upload to begin immediately after the user selects a file.

28.4.2 Working with the Upload Handler

Even though the upload handler is an *.ashx and the RadUpload control is working from a Silverlight client, you can communicate both ways. You can also add custom logic to the handler itself, allowing you an opportunity to save files to a data base or other storage and to perform any other operations with the file stream that suit your purpose.

Sending Parameters to the Upload Handler

You can send parameters that travel to the upload handler on every request and can also be sent along with specific files. In both cases you can use Dictionaries made available by RadUpload to store keys and values.

The example below adds a custom parameter called "CompanyID" that will be sent with every upload. Custom parameters are added to the RadUpload **AdditionalPostFields** dictionary.

Use the **FileUploadStarting** event when you want to send parameters along with individual files. The arguments to this event include a **FileParameters** dictionary. The example sends a parameter with key "InvoiceNumber". For the sake of brevity, an invoice number is extracted from the file name where the format is similar to "Invoice_12345.xml".

VB

```
Private Sub UserControl_Loaded(ByVal sender As Object, ByVal e As RoutedEventArgs)
  InvoiceUploader.AdditionalPostFields.Add("CompanyID", "55892")
End Sub

Private Sub RadUpload_FileUploadStarting(ByVal sender As Object, _
 ByVal e As FileUploadStartingEventArgs)
  ' remove file extension
  Dim fileName As String = e.SelectedFile.Name.Split("."c)(0)
  'extract invoice number
  Dim invoiceNumber As String = fileName.Split("_"c)(1)
  e.FileParameters.Add("InvoiceNumber", invoiceNumber)
End Sub
```

C#

```
private void UserControl_Loaded(object sender, RoutedEventArgs e)
{
   InvoiceUploader.AdditionalPostFields.Add("CompanyID", "55892");
}

private void RadUpload_FileUploadStarting(object sender, FileUploadStartingEventArgs e)
{
   // remove file extension
   string fileName = e.SelectedFile.Name.Split('.')[0];
   //extract invoice number
   string invoiceNumber = fileName.Split('_')[1];
   e.FileParameters.Add("InvoiceNumber", invoiceNumber);
}
```

Custom Processing in the Upload Handler

To perform your own custom processing in the upload handler, override the **SaveChunkData()** method. The brief example below shows parameters from the Silverlight client being passed in, some mock processing, then parameters are passed back. Parameters can be sent back to the client using one of two methods:

- The **AddReturnFileParam()** method. The key to each Dictionary element is a predefined key from RadUploadContants, e.g. RadUploadContants.ParamNameMessage. The AddReturnParam() method can be called during the ProcessStream() method.
- Overriding the **GetAssociatedData()** method. This method runs before the ProcessStream() method and can contain sets of keys and values.

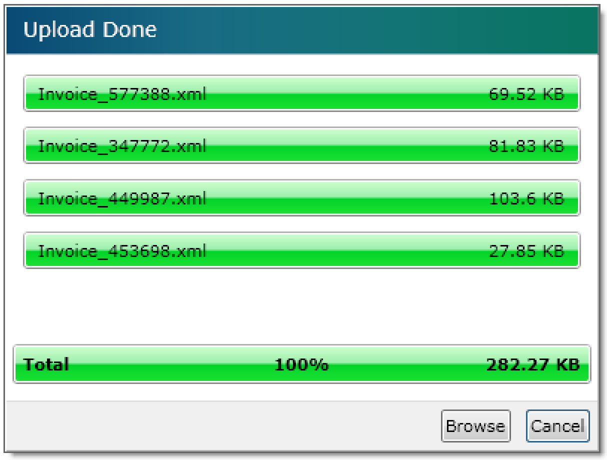
Here are some of the key points to the example below:

- Parameters coming from the client "AdditionalPostFields" and "FileParameters" are retrieved by **this**. GetQueryParameter().
- A check against **IsFinalFileRequest()** lets us know that the entire file has been downloaded. **Note**: Larger files are broken into "chunks", so the SaveChunkData() method may be called multiple times for a single file.
- After processing the file with custom logic, parameters are returned to the client using **AddReturnFileParam()**. The predefined **ParamNameMessage** key has a value "Processed invoice #12345" and the **ParamNameSuccess** value is "True". If our custom "IsAuthorized" member is false, the ParamNameSuccess value is "False" and the ParamNameMessage value is "Not authorized". The important issue here isn't the specific program logic, but how the **RadUploadConstants** are used along with the AddReturnParam() method.
- In the **GetAssociatedData()** method we get the "CompanyID" sent from the client with every request and do a primitive check against a hard coded company number and store the result in a private "IsAuthorized" member. "IsAuthorized" is included in the dictionary and passed back to the client.

```
Private IsAuthorized As Boolean = False

Public Overrides Function SaveChunkData(filePath As String, position As Long, buffer As Byte(), contentLeng
  Dim invoiceNumber As Integer = Integer.Parse(Me.GetQueryParameter("InvoiceNumber").ToString())
  Dim result As Boolean = False
  ' is entire file uploaded?
  If Me.IsFinalFileRequest() Then
    If IsAuthorized Then
      ' do some mock processing
      ProcessInvoice()
      Dim message As String = [String].Format("Processed invoice #{0}", invoiceNumber)
      Me.AddReturnFileParam(RadUploadConstants.ParamNameMessage, message)
      Me.AddReturnFileParam(RadUploadConstants.ParamNameSuccess, True)

        'return result;
      result = MyBase.SaveChunkData(filePath, position, buffer, contentLength, savedBytes)
    Else
      Me.AddReturnFileParam(RadUploadConstants.ParamNameMessage, "Not authorized")
      Me.AddReturnFileParam(RadUploadConstants.ParamNameSuccess, False)
      savedBytes = 0
      result = False
    End If
  End If

  savedBytes = 0
  Return result
End Function

Public Overrides Function GetAssociatedData() As Dictionary(Of String, Object)
  Dim companyID As Integer = Integer.Parse(Request.Form("CompanyID").ToString())
  IsAuthorized = companyID.Equals(55892)

  Dim dictionary As Dictionary(Of String, Object) = New Dictionary(Of String, Object)()
  dictionary.Add("IsAuthorized", IsAuthorized)
  Return dictionary
End Function

Public Sub ProcessInvoice()
  ' mock processing
End Sub
```

```
bool IsAuthorized = false;

public override bool SaveChunkData(string filePath, long position, byte[] buffer, int contentLength, out int sav
{
  int invoiceNumber = int.Parse(this.GetQueryParameter("InvoiceNumber").ToString());
  bool result = false;
  // is entire file uploaded?
  if (this.IsFinalFileRequest())
  {
    if (IsAuthorized)
    {
      // do some mock processing
      ProcessInvoice();
      string message = String.Format("Processed invoice #{0}", invoiceNumber);
      this.AddReturnFileParam(RadUploadConstants.ParamNameMessage, message);
      this.AddReturnFileParam(RadUploadConstants.ParamNameSuccess, true);

      result = base.SaveChunkData(filePath, position, buffer, contentLength, out savedBytes);
      //return result;
    }
    else
    {
      this.AddReturnFileParam(RadUploadConstants.ParamNameMessage, "Not authorized");
      this.AddReturnFileParam(RadUploadConstants.ParamNameSuccess, false);
      savedBytes = 0;
      result = false;
    }
  }

  savedBytes=0;
  return result;
}

public override Dictionary<string, object> GetAssociatedData()
{
   int companyID = int.Parse(Request.Form["CompanyID"].ToString());
   IsAuthorized = companyID.Equals(55892);

   Dictionary<string, object> dictionary =
      new Dictionary<string, object>();
   dictionary.Add("IsAuthorized", IsAuthorized);
   return dictionary;
}

public void ProcessInvoice()
{
   // mock processing
}
```

If everything runs without exception, then the RadUpload control shows that all the selected files have been uploaded.



Upload Done

Invoice_577388.xml 69.52 KB

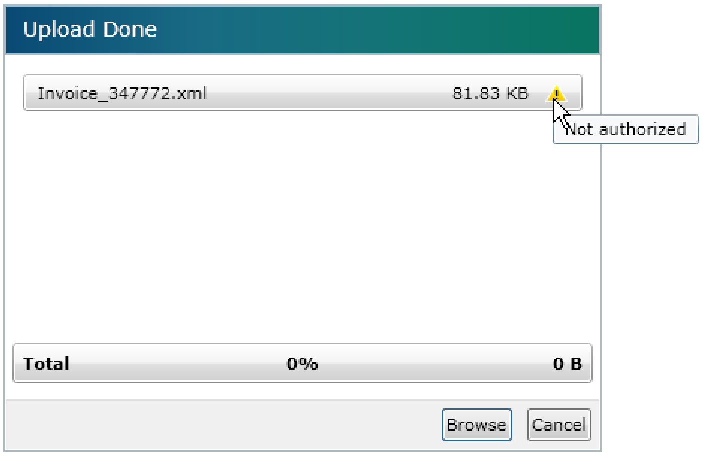
Invoice_347772.xml 81.83 KB

Invoice_449987.xml 103.6 KB

Invoice_453698.xml 27.85 KB

Total 100% 282.27 KB

If we send the wrong company ID from the client, ParamNameSuccess gets a value of "False" and ParamNameMessage is set to "Not authorized". As a result, RadUpload displays a warning icon and a tool tip with the ParamNameMessage value.



Upload Done

Invoice_347772.xml 81.83 KB ⚠

Not authorized

Total 0% 0 B

From the Forums...

Question: Is it possible to just have the byte array and not save the file to disk? Is there a switch to turn off the save to disk?

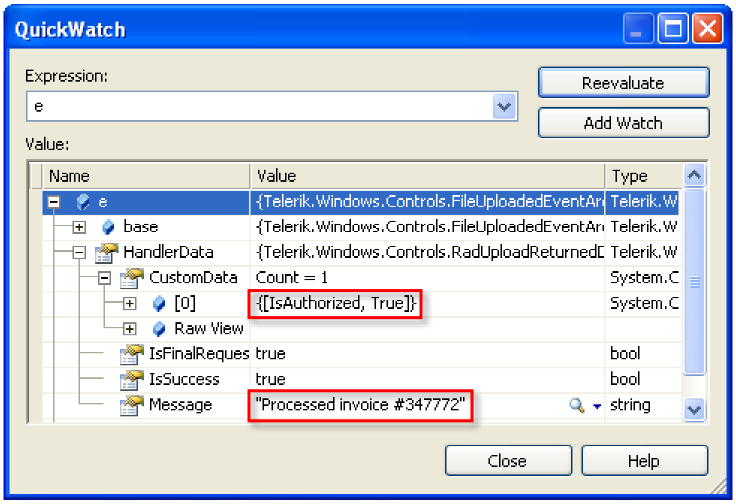
Answer: Yes - it is possible. If you override the SaveChunkData() method but don't call the base. SaveChunkData() method, you will have full control over the upload process. Please note that this will not stop the upload by chunks. If you want to stop the upload by chunks you can set a bigger BufferSize, and to set a MaxFileSize smaller than the BufferSize.

```
byte[] array = File.ReadAllBytes(filePath);
```

Returning Parameters from the Upload Handler

The parameters from the handler can be received during the **FileUploaded** event. Look for the arguments **HandlerData** property. This will contain the **IsSuccess** and the **Message** from the handler. In HandlerData, the **CustomData** property holds the values added during the GetAssociatedData() method. See the screenshot below to see the data returned in **FileUploadedEventArgs**.

```
private void RadUpload_FileUploaded(object sender,
    FileUploadedEventArgs e)
{

}
```



Expression: e

Value:

Name	Value	Type
e	{Telerik.Windows.Controls.FileUploadedEventAr	Telerik.W
base	{Telerik.Windows.Controls.FileUploadedEventAr	Telerik.W
HandlerData	{Telerik.Windows.Controls.RadUploadReturnedD	Telerik.W
CustomData	Count = 1	System.C
[0]	{[IsAuthorized, True]}	System.C
Raw View		
IsFinalReques	true	bool
IsSuccess	true	bool
Message	"Processed invoice #347772"	string

28.4.3 Events and Methods

The event model of RadUpload lets you track every phase of the selecting and uploading process. "Upload" events track the state of the upload as a whole and include **UploadStarted**, **UploadPaused**, **UploadResumed** and **UploadCanceled and UploadFinished**. UploadStarted passes an argument with a **SelectedFiles** property that lists all the files being uploaded. You can also handle the **FilesSelected** event when the user closes the "Open File" dialog and this will also pass back the SelectedFiles property.

When a file doesn't upload for some reason, **FileUploadFailed** will fire and pass back arguments that include the error message and the selected file.

The **ProgressChanged** event doesn't pass any interesting arguments, but instead you can use the RadUpload **CurrentSession** property to get the **CurrentProgress** for the entire session, **CurrentFileProgress** for the progress of the file uploading and **CurrentFile** for all the information about the file being uploaded. **Note:** CurrentSession also has collections of **FailedFiles**, **SelectedFiles**, **TooLargeFiles**, **UploadedFiles** and **ValidFiles**.

RadUpload methods complement these events by letting you control the uploading process completely from code-behind: **ShowFileDialog()**, **StartUpload()**, **PauseUpload()** and **CancelUpload()**. You can in fact use the RadUpload as a "silent" control without the UI portion by following these steps:

1) Add reference to the **Telerik.Windows.Controls.Input** assembly;

2) Add a member of type **RadUpload** to your class. Do not initialize or create the RadUpload in XAML.

3) In the XAML code, add a place holder control to host the hidden RadUpload.

4) Initialize the upload control. The Page.Load event would be an appropriate spot for this logic.

   a) Set the addresses for the **TargetFolder** and the **UploadServiceUrl**;

   b) Set the values to determine RadUpload behavior, i.e. buffer, file and upload sizes, multi-selection, etc.;

   c) To allow interaction, implement handlers for **FilesSelected**, **UploadStarted**, **UploadFinished**, **UploadCanceled**, **UploadPaused** and **UploadResumed**.

   d) To hide the RadUpload, set the **Opacity** property to Zero.

5) Add the RadUpload control as a child of its place holder.

6) Add Browse and Upload buttons to the page. These buttons will be used to call ShowFileDialog() and StartUpload() methods.

28.5 Customization

In this example we will customize RadUpload to be more compact.

Project Setup

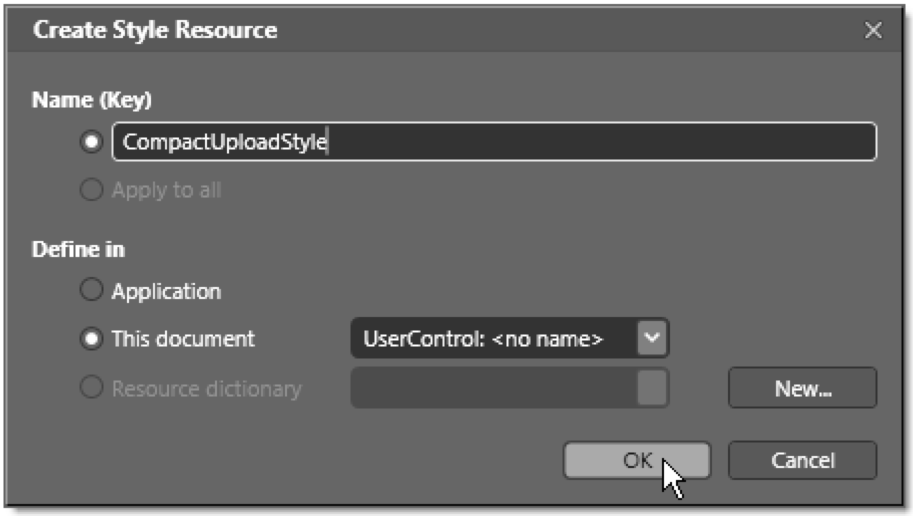
1) Run Expression Blend.

2) Open the "Getting Started" project or a copy.

Edit the Page in Expression Blend

1) Locate MainPage.xaml in the Projects pane and double-click to open the page.

2) Right-click the RadUpload control in the Objects and Timeline pane and select **Edit Template > Edit a Copy** from the context menu. In the "Create Style Resource" dialog, set the Name (Key) to "CompactUploadStyle". Click **OK** to create the style resource and close the dialog.



Create Style Resource

Name (Key)

● CompactUploadStyle

○ Apply to all

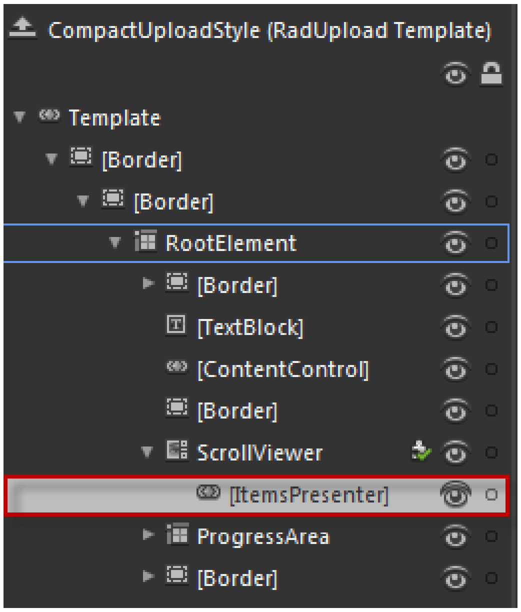
Define in

○ Application

● This document UserControl: <no name>

○ Resource dictionary

3) In the Objects and Timeline pane, open the tree view and locate the "[ItemsPresenter]" and select it. You should find it as a child of the ScrollViewer under the RootElement.



4) In the Properties pane, Layout section, set the **Width** of the ItemsPresenter to "220".

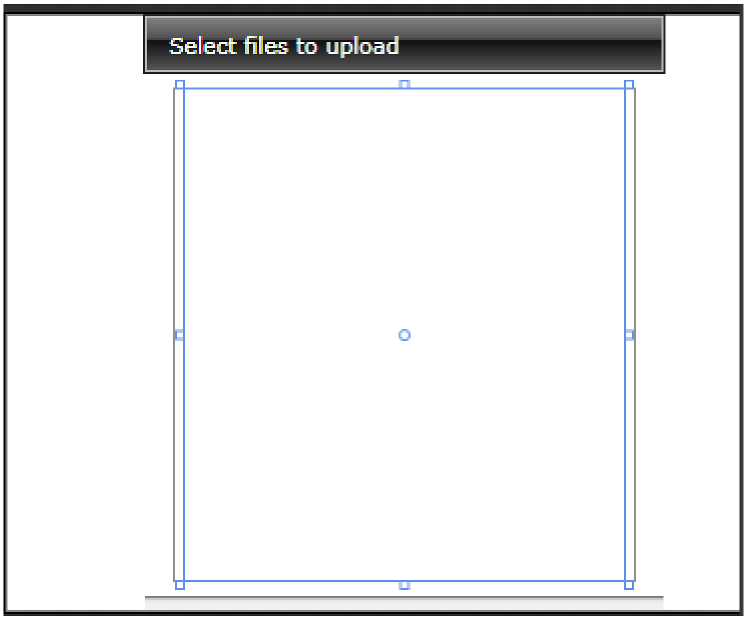
5) Select the ScrollViewer. Use the Properties pane to set the **Width** property at "230".

6) Select the "[Border]" node located just above the ScrollViewer. Use the Properties pane to set the **Width** property at "230".

7) Selected the top "[Border]" element. Use the Properties pane to set the **Width** property at "280".

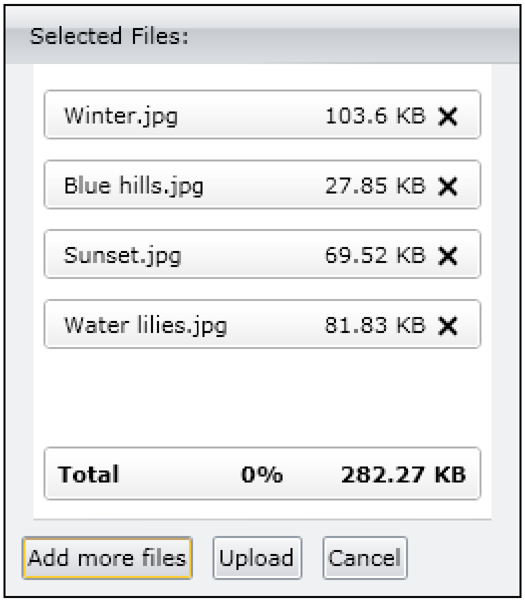
8) Below the ScrollViewer, find the "ProgressArea" and select it. Use the Properties pane to set the **Width** property at "230".

The RadUpload in the Artboard should look something like the screenshot below.



Run The Application

Press **F5** to run the application. The web page should look something like the screenshot below.



28.6 Wrap Up

In this chapter you used the minimal configuration required to upload files from a Silverlight client to a server. You learned the basics for creating the server upload handler along with some of the "Gotchas" that could cause an upload to fail.

In the section on "Control Details" you became familiar with how to control access to files during the upload including properties that limit file types and limiting the number of files and bytes allowed. You also learned about the properties that control the operations allowed by showing or hiding buttons in the UI. You created your own upload handler and learned how to pass parameters to and from the client. You learned about the events that cover the entire upload life-cycle and the methods that can be used to trigger the file operations.

During the "Customization" section you walked through using Expression Blend to modify RadUpload into a more compact dialog.

Index

www.telerik.com