CHAPTER 1

A Quick Tour of Silverlight 2 Development

The Mechanics of Silverlight

This is a recipes book, which means it is prescriptive, targeting specific scenarios that developers are likely to encounter. You don't have to read the chapters in order, though we did put some thought into chapter organization for readers who want to proceed that way. Since Silverlight 2 is a relatively new technology, we first provide an overview to kick things off in this chapter. Depending on your experience level with Silverlight 2, this chapter may be essential to help you get started or a quick read.

Silverlight 2 is Microsoft's cross-browser, cross-platform, and cross-device plug-in for delivering the next generation of .NET Framework–based rich interactive applications for the Web. Silverlight runs on Windows in Internet Explorer 6 or higher as well as in the Mozilla Firefox Internet browser. Silverlight also runs on the Apple Macintosh under both Safari and Firefox as well as on Linux under Firefox as part of the Moonlight project (`http://www.mono-project.com/Moonlight`). Moonlight is a collaboration project between Novell and Microsoft to bring Silverlight to Linux. Figure 1-1 shows a Silverlight application running in Firefox.

Silverlight is a browser plug-in, meaning that the user is prompted to install the plug-in if it is not already present, which is the same behavior as similar technologies like Adobe Flash. Once the plug-in is installed, which takes about ten seconds on most Internet connections, Silverlight content can be downloaded and run.

Because Silverlight is client-side technology, Silverlight content can be hosted on any backend system—just as any other web-based content such as HTML pages and JavaScript can be hosted from a Windows server, Linux server, or any other web-serving technology.

Microsoft announced Silverlight for Windows Mobile at the MIX '08 conference. Interestingly, Nokia announced plans to port Silverlight on Symbian as well, substantially extending the reach of Silverlight.

Figure 1-1. *Firefox running Silverlight 2*

You may have heard of or worked with Silverlight 1.0, which was released in September 2007. Silverlight 1.0 consisted of primitive drawing objects such as paths, ellipses, and rectangles, with a JavaScript-based code execution model. Silverlight 2 is Silverlight 1.0 on steroids and consists of a full control model and a .NET-based execution model.

You may be asking, "Did Microsoft port the .NET Framework to additional platforms?" The answer is yes and no. The full desktop and server version of the .NET Framework runtime has not been ported to other platforms. However, the product team did encapsulate a tiny common language runtime (CLR) into the plug-in and made it cross-browser as well as cross-platform.

What makes Silverlight 2 so compelling is that much of the power and programmability of the .NET Framework is available within a relatively small 4.2MB plug-in. Here are some of your favorite namespaces available in Silverlight 2:

- `System`
- `System.Collections`

Silverlight 2 Recipes: A Problem-Solution Approach

ISBN-13 (pbk): 978-1-59059-977-8

ISBN-10 (pbk): 1-59059-977-2

ISBN-13 (electronic): 978-1-4302-0620-0

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editors: Ewan Buckingham, Matthew Moodie
Technical Reviewer: Todd Herman
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Liz Welch
Associate Production Director: Kari Brooks-Copony
Production Editor: Janet Vail
Compositor: Linda Weidemann, Wolf Creek Press
Proofreader: Kim Burton
Indexer: Ron Strauss
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at http://www.apress.com/info/bulksales.

The source code for this book is available to readers at http://www.apress.com.

- System.Collections.Generic
- System.Diagnostics
- System.Globalization
- System.IO
- System.Linq
- System.NET
- System.Reflection
- System.Runtime
- System.Security
- System.ServiceModel
- System.Text
- System.Windows
- System.XML

As you can see, Silverlight 2 packs a powerful .NET Framework base class library. From an application user interface (UI) perspective, Silverlight 2 is most similar to Windows Presentation Foundation (WPF) than to any other technology. If you understand WPF, then you pretty much will understand how to build UIs in Silverlight 2 because Silverlight 2 is a subset of WPF.

In case you are not familiar with WPF, it is the new desktop application framework that runs on top of DirectX, providing it with capabilities for building rich dynamic desktop applications. WPF can run inside a browser as an Extensible Application Markup Language (XAML) browser application. XAML browser applications (XBAPs) will run in Firefox or Internet Explorer but only on Windows because WPF requires the full desktop framework.

Both Silverlight 2 and WPF are coded in XAML with a code-behind file, which allows for interesting scenarios of sharing UI code between a desktop application and cross-platform browser application as long as you build to the Silverlight 2 programming model. Silverlight 2 is a rich subset of WPF, but not all capabilities in WPF are implemented in Silverlight 2.

Silverlight 2 can be coded with Notepad and the software development kit (SDK), but it might not be the most productive development effort. We don't see a big market for Silverlight 2 Notepad development, so we are going to spend most of our time with two tools: Visual Studio 2008 and Expression Blend 2 Service Pack 1 (SP1).

■ **Note** Silverlight 2 is not compatible with Expression Blend 2 RTM. Expression Blend 2 targets technologies released at the time of its creation—specifically Silverlight 1.0 and WPF 3.5. Silverlight 2 requires Expression Blend 2 SPI. (In beta, the tool was known as Expression Blend 2.5 June 2008 Preview.)

Visual Studio 2008 provides XAML and source code editing with excellent IntelliSense and debugging support. Expression Blend 2 SPI targets designers, though developers will find it useful as well. Expression Blend 2 SPI provides a visual design surface for Silverlight 2 UI development. In our first recipe, we cover setting up the development and designer environment; then in the subsequent recipes we expand on the capabilities each tool provides.

1-1. Setting Up the Silverlight 2 Environment

Problem

You need to set up a Silverlight 2 development and designer environment.

Solution

Uninstall all previous versions of Silverlight 2 tools, runtime, and Expression Blend 2.5 June 2008 Preview. Install Silverlight 2 Tools and SDK, Visual Studio 2008, and Expression Blend 2 SPI following the latest installation guidance.

How It Works

The steps listed at `http://Silverlight.net/GetStarted` cover the details, but the first step is to install a version of Visual Studio 2008.

■ **Note** Silverlight 2 is supported on Visual Studio 2008 Express SP1.

At `http://Silverlight.net/GetStarted` you can obtain the Silverlight Tools for Visual Studio 2008 SP1, which includes the runtime (if not already installed), the project templates, debugging support, the SDK, and documentation files. Install the Silverlight Tools to enable development of Silverlight 2 in Visual Studio 2008. The installation will want you to close any conflicting application such as Internet Explorer; otherwise, a reboot may be required.

At the same URL, another tool is available called Deep Zoom Composer. This tool allows developers to prepare images for use with the Deep Zoom feature in Silverlight 2. Deep Zoom allows users to explore collections of super-high-resolution imagery, from a 2- or 3-megapixel shot from a digital camera to gigapixel scans of museum pieces, all without waiting for huge file downloads. The simple zooming interface allows users to explore entire collections down to specific details in extreme close-up, all with fantastic performance and smooth transitions. A great implementation of the Deep Zoom is the Hard Rock Café, where users can get visually up-close to their huge collection of memorabilia. Here is the site's URL: `http://memorabilia.hardrock.com/`.

The next step is to install Expression Blend 2 SP1. This is an important tool in the application creation arsenal that is part of the Expression Studio product line. If you are a Microsoft Developer Network (MSDN) Premium subscriber, you can obtain Blend from MSDN downloads. If you are not an MSDN Premier subscriber, you can download a trial version at `http://www.microsoft.com/expression/try-it/Default.aspx`. As noted earlier, Silverlight 2 at released to manufacturing (RTM) requires the SP1 version of Expression Blend 2, which can be installed side by side with previous versions of Expression Studio.

■ **Note** If you are still working with the beta 2 version of Silverlight 2, the version was Expression Blend 2.5 June Preview.

Even as a developer, you will want to have Expression Blend 2 SP1 installed along with Visual Studio 2008. Designers may not want or need to have Visual Studio 2008 installed, but they may want to install the source code control client software appropriate for their company environment.

For example, if the developers keep their source code in Microsoft Team Foundation Server, the designer may want to install the Team Foundation Client in order to be able to check out and check in code directly without installing Visual Studio 2008.

1-2. Integrating the Silverlight 2 SDK Documentation

Problem

You need to add the Silverlight 2 SDK documentation to the Visual Studio 2008 combined help collection.

Solution

Manually perform the necessary steps to integrate the documentation.

How It Works

When you install the Silverlight Tools for Visual Studio 2008, a Start ➤ Programs menu item is added called Microsoft Silverlight 2 SDK, with a Welcome submenu that points to an HTML page containing a list of resources as well as a link to the VS-Help folder. Inside the VS-Help folder for your language is a `Readme.txt` file that contains steps for adding the Silverlight 2 SDK documentation to the Visual Studio integrated help environment. Here are the steps:

1. Open Visual Studio (you might have to open it as an administrator).
2. In the Help menu, choose Index. Microsoft Document Explorer displays.
3. In the Filtered By drop-down, choose (Unfiltered).
4. In the Look For field, type **Collection Manager**.
5. In the Collection Manager section, double-click Help.
6. Below Collections Available for Inclusion in VSCC, check Microsoft Silverlight 2 SDK Documentation.
7. Click Update VSCC.

The help application prompts you to close all open instances of Visual Studio. After performing these steps, open Visual Studio Help and the Contents tab will contain the Microsoft Silverlight 2 SDK.

1-3. Understanding the Structure of a Silverlight Solution

Problem

You need to understand the structure of a Silverlight 2 solution and projects.

Solution

Create a new Silverlight application in Visual Studio and review the solution and project files.

How It Works

Once the Silverlight Tools are installed and help is configured, open Visual Studio 2008, choose File ➤ New ➤ Project, and click on the Silverlight folder to see the available project templates. Two templates are available: Silverlight Application and Silverlight Class Library.

The Silverlight Class Library project generates a straightforward class library project that you can use to separate Silverlight 2 application assets into additional assemblies that can be shared by multiple applications.

The Silverlight Application project template is what you will start with to create a Silverlight application that begins initially with up to two projects, one containing the Silverlight application project and another optional project containing web pages to host the Silverlight application for testing. You can, of course, add more Silverlight applications to the solution; the Visual Studio Add Silverlight Application Wizard will automatically create test pages and add them to the existing web project if desired.

You can also create a Silverlight application that has just the Silverlight application without a separate web project. In this case, Visual Studio will dynamically create a test page for the control. We recommend starting out with an actual web project so you can see how to host Silverlight applications in web pages, but we cover both options in this recipe.

You can also add Silverlight applications to any Visual Studio solution and the project wizard will ask you whether you want to add test pages to the existing web project (if there is one), which makes it easy to add new applications to an existing Visual Studio web solution. If there isn't an existing web project in a Visual Studio solution, the project wizard will prompt you to create one.

The Code

When you create a new Silverlight application project, type a name, select the project location, and then click OK, the Add Silverlight Application dialog appears, as shown in Figure 1-2.

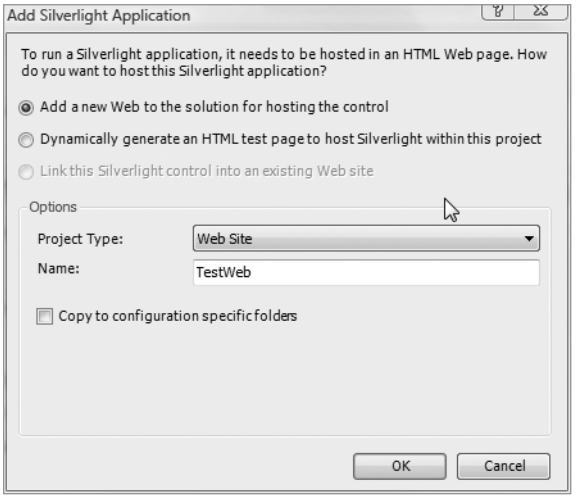

Figure 1-2. *The Add Silverlight Application dialog*

The default option is to add a new web application to the solution for hosting the control. This option creates a web project for you with two pages, an ASPX page and an HTML page that are both automatically configured to host the Silverlight application. This is a great option to start out with when creating Silverlight applications.

When you opt for adding the web site, it defaults to a web project type of Web Site, which is the simple file-based web project. If you instead choose to select Web Application Project, as we will here, it allows more project configuration, such as specifying a static port where the web site runs (because it runs in the file system). This is handy when you're creating web services and you want to keep the port number consistent. You can choose the name for the test web site as well.

There is an additional web site checkbox item called Copy to Configuration Specific Folders. This option allows you to specify if you want to copy the `.xap` file to a configuration-specific subfolder (debug/release) of the build's target folder (ClientBin) or simply copy the chosen build configuration (debug or release) into ClientBin directly. We leave this option unchecked, but this option could be handy if you are automating builds and want a set folder for each configuration—that is, debug or release.

As background, the XAP file is the output of the Silverlight application project. It contains the assemblies, the manifest, and other output files that are loaded by the Silverlight 2 plug-in to execute the application. The XAP file is actually just a ZIP file renamed with the `.xap` extension, so you can crack one open to see what is inside by simply changing the extension to `.zip`.

The other Silverlight Application option is called Dynamically Generate an HTML Test Page to Host Silverlight Within This Project. This option results in a solution with a single project containing the Silverlight control. When you run the application, an HTML page is generated that hosts the application. Our preference is to create a separate web application project and not use the dynamically generated HTML test page option, but it is good to have options.

Once you've chosen the Silverlight Application option to have a separate project-based web site in the file system and you've updated the name for the web site project, click OK to create the initial solution. Figure 1-3 shows the initial project layout.

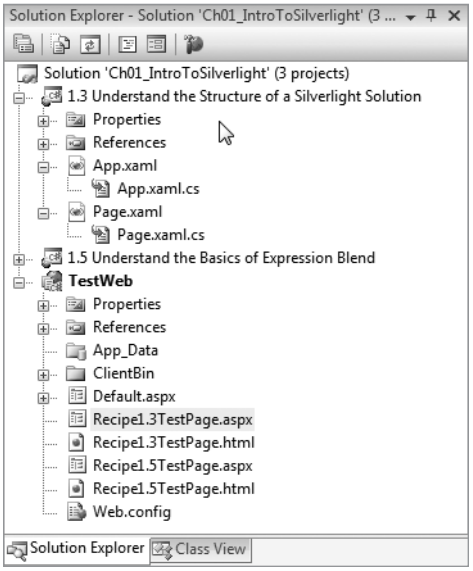

Figure 1-3. *Silverlight 2 initial project layout*

The Silverlight application project named "1.3 Understand the Structure of a Silverlight Solution" consists of two files: `App.xaml` and `Page.xaml`. There are also corresponding code-behind files: `App.xaml.cs` and `Page.xaml.cs`. The class in the `App.xaml` file serves as the start-up object for the Silverlight application. Listings 1-1 and 1-2 show the initial `App.xaml` file and its code-behind, `App.xaml.cs`, respectively.

Listing 1-1. *Recipe 1-3 App.xaml File*

```
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             x:Class="Ch01_IntroToSilverlight.Recipe1_3.App"
             >
    <Application.Resources>

    </Application.Resources>
</Application>
```

Listing 1-2. *Recipe 1-3 Initial App.xaml.cs Class File*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace Ch01_IntroToSilverlight.Recipe1_3
{
  public partial class App : Application
  {

    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 Page();
    }

    private void Application_Exit(object sender, EventArgs e)
    {

    }
    private void Application_UnhandledException(object sender,
    ApplicationUnhandledExceptionEventArgs e)
    {
      // If the app is running outside of the debugger then report the exception
      //using the browser's exception mechanism. On IE this will display it a
      //yellow alert icon in the status bar and Firefox will display a script error.
      if (!System.Diagnostics.Debugger.IsAttached)
      {
```

```
        // NOTE: This will allow the application to continue running after an
        //exception has been thrown but not handled.
        // For production applications this error handling should be replaced with
       //something that will report the error to the website and stop
       //the application.
        e.Handled = true;

        try
        {
          string errorMsg = e.ExceptionObject.Message + e.ExceptionObject.StackTrace;
          errorMsg = errorMsg.Replace('"', '\'').Replace("\r\n", @"\n");

          System.Windows.Browser.HtmlPage.Window.Eval("throw new Error(\
          "Unhandled Error in Silverlight 2 Application " + errorMsg + "\");");
        }
        catch (Exception)
        {
        }
      }
    }
  }
}
```

Both files contain partial classes for the `Ch01_IntroToSilverlight.Recipe1_3.App` class that inherits from the `System.Windows.Application` class. Notice at the top of `App.xaml` in the `<Application>` element two namespaces are declared that are required for Silverlight 2 development. Also, in the `<Application>` element tag is an `x:Class` attribute linking the `App.xaml` markup file to the `App.xaml.cs` code-behind file with the related partial class. The `App.xaml` file is a good place to store application-wide resources such as styles and templates, which we cover in Chapters 4 and 5.

The `App.xaml.cs class` file implements methods of the `Application` class, such as the constructor where events are wired up and `InitializeComponent()` is called to process the XAML markup. The `App_Startup` event sets the visual root to the `Page` class defined in `Page.xaml` and `Page.xaml.cs`. `Application_Exit` is implemented as a placeholder for the developer to add logic needed to handle the exit process. Finally, `Application_UnhandledException` is implemented to provide a basic top-level exception handler event.

All of the UI and execution logic for the Silverlight 2 application exists in the `Page` class XAML and code-behind class file. Listings 1-3 and 1-4 show the `Page.xaml` file and its code-behind, `Page.xaml.cs`.

Listing 1-3. *Recipe 1-3 Initial Page.xaml File*

```
<UserControl x:Class="Ch01_IntroToSilverlight.Recipe1_4.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">

    </Grid>
</UserControl>
```

Listing 1-4. *Recipe 1-3 Initial Page.xaml.cs Class File*

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

```
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace Ch01_IntroToSilverlight.Recipe1_3
{
  public partial class Page : UserControl
  {
    public Page()
    {
      InitializeComponent();
    }
  }
}
```

Both files contain partial classes for the `Ch01_IntroToSilverlight.Recipe1_3.Page` class that inherits from the `System.Windows.Controls.UserControl` class. Notice at the top of `Page.xaml` in the `<UserControl>` element two namespaces are declared that are required for Silverlight 2 development. Also, in the `<UserControl>` element tag is an `x:Class` attribute linking the `Page.xaml` markup file to the `Page.xaml.cs` code-behind file with the related partial class. The `Page.xaml` file is a good place to store page-wide resources such as styles and templates, which we cover in Chapter 2.

The project settings for the Silverlight application have a Silverlight tab, as shown in Figure 1-4.

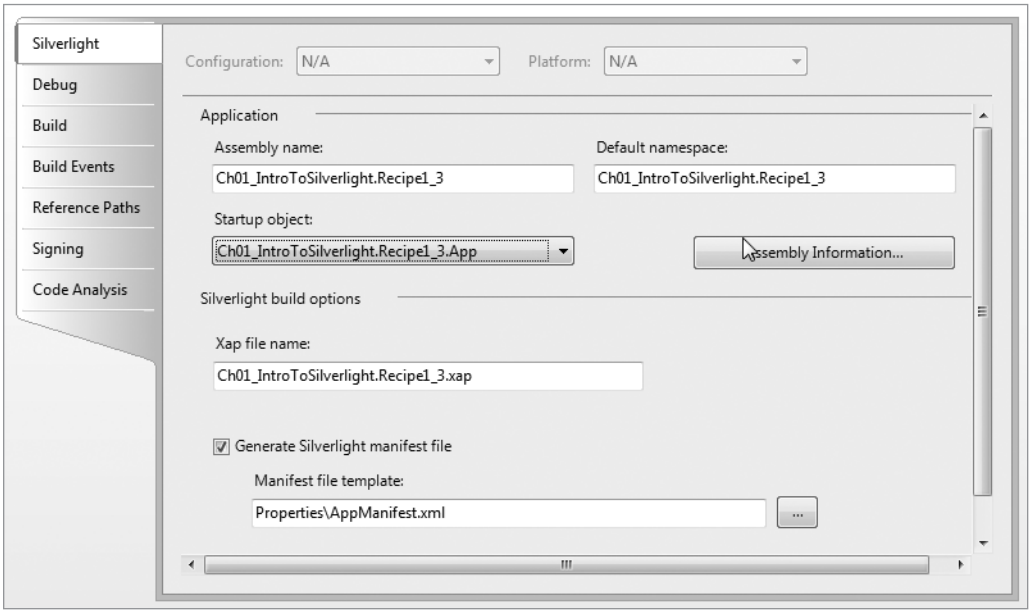

Figure 1-4. *The Silverlight application settings tab in the Silverlight project settings*

In addition to the properties that are typically available for a .NET application are the XAP file name and an option that lets you specify whether you want to generate a Silverlight manifest file.

The other project in the solution is the `TestWeb` web project. The web project consists of two pages per Silverlight application. One file is an ASPX page and the other is an HTML page. So the corresponding files for the Silverlight application we just covered are `Recipe1.3TestPage.aspx` and `Recipe1.3TestPage.html`. Both pages instantiate the Silverlight application. The ASPX page uses an instance of the ASP.NET server control `System.Web.UI.SilverlightControls.Silverlight` to create the application within the browser. Here is an example of the markup from the ASPX page:

```
<asp:Silverlight ID="Xaml1" runat="server"
Source="~/ClientBin/Ch01_IntroToSilverlight.Recipe1_3.xap"
MinimumVersion="2.0.30523" Width="100%" Height="100%" />
```

You can see the reference to the XAP file in the ASP.NET markup. The HTML page manually creates the Silverlight application using an `<object>` tag. Here is an example of the HTML markup from the HTML page:

```
<object data="data:application/x-silverlight," type="application/x-silverlight-2-b2"
width="100%" height="100%">
  <param name="source" value="ClientBin/Ch01_IntroToSilverlight.Recipe1_3.xap"/>
  <param name="onerror" value="onSilverlightError" />
  <param name="background" value="white" />
  <a href="http://go.microsoft.com/fwlink/?LinkID=115261"
    style="text-decoration: none;">
   <img src="http://go.microsoft.com/fwlink/?LinkId=108181"
    alt="Get Microsoft Silverlight" style="border-style: none"/></a>
</object>
```

The `<object>` tag shown here is essentially what the ASP.NET server control renders at runtime. There is the reference to the XAP file as well as some client-side JavaScript events, such as `onSilverlightError`, wired in.

Figure 1-5 shows the project settings for the `TestWeb` web project.

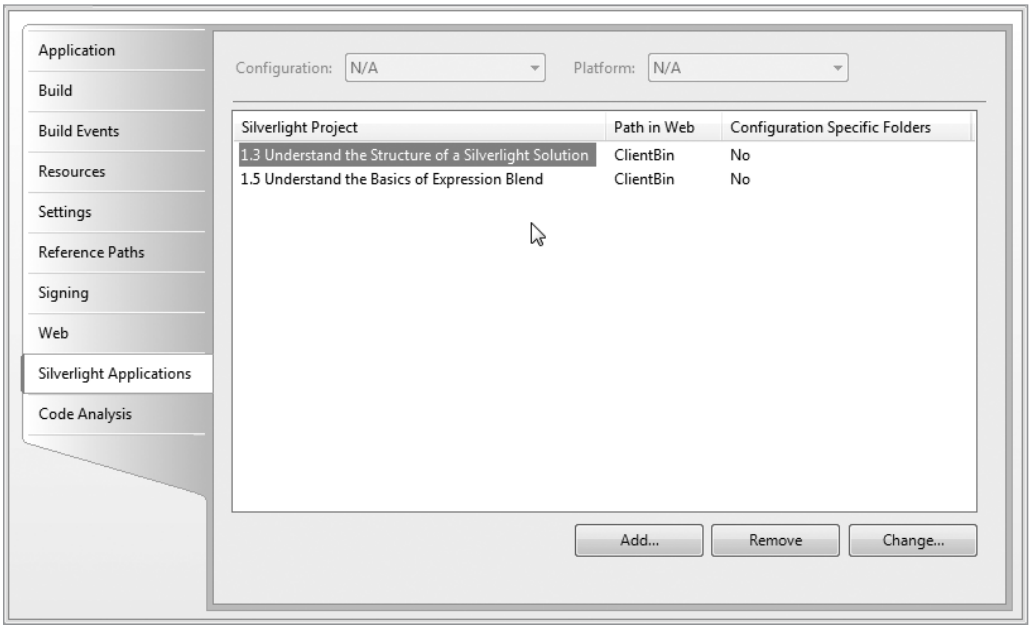

Figure 1-5. *The Silverlight Applications tab in the web project settings*

The tab shown in Figure 1-5 lists the Silverlight projects available in the solution, where the XAP file should be copied to (the ClientBin folder), as well as whether there should be configuration specific folders under the ClientBin folder for debug or release versions of the Silverlight application.

Notice also in Figure 1-5 the Add, Remove, and Change buttons. The Add button allows you to easily add a Silverlight 2 application to a web project. Let's say you have an existing ASP.NET application that you now want to add to a Silverlight 2 application. Open the project settings for the web project and click the Add button to display the Add Silverlight Application dialog shown in Figure 1-6.

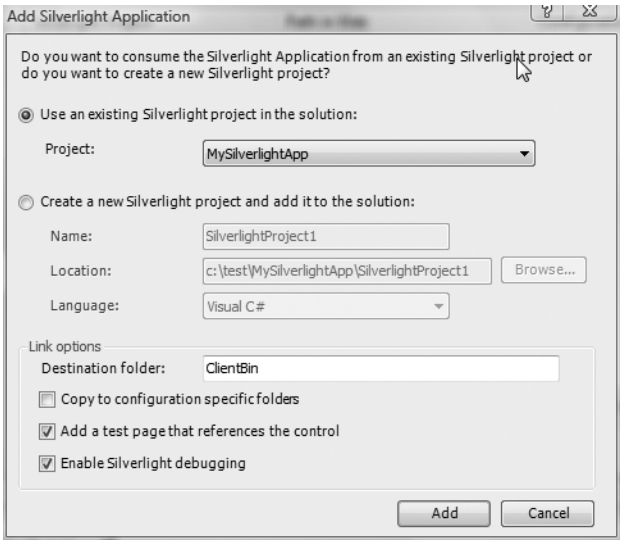

Figure 1-6. *The Add Silverlight Application dialog*

You have the option to create a new Silverlight 2 project or add an existing project form the prepopulated Project combo box as well as configure related settings, such as the destination folder, whether to have configuration specific folders, whether to add test pages, as well as whether to enable Silverlight debugging.

When you click the Change button on the Silverlight Applications tab, a dialog displays with the message shown in Figure 1-7.

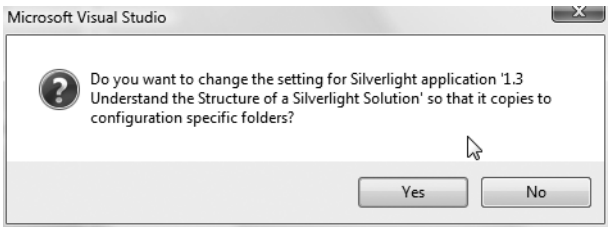

Figure 1-7. *This dialog lets you switch to using configuration files for debug/release versions.*

This dialog allows you to switch to using configuration files for debug/release versions. When you click the Remove button on the Silverlight Applications tab, you'll see the dialog shown in Figure 1-8.

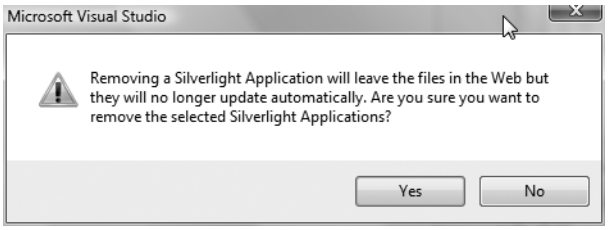

Figure 1-8. *This dialog lets you remove a Silverlight application from a web project.*

1-4. Understanding the Developer/ Designer Workflow

Problem

You need to understand the process for creating Silverlight applications.

Solution

Learn the capabilities of the Visual Studio 2008 and Expression Blend 2 SP1 environments. Depending on the type of application, determine whether a dedicated UI designer is required for the project or whether the developer will handle the UI development and the coding. If the application requires a dedicated designer due to UI requirements, introduce the designer to Expression Blend 2 SP1.

How It Works

With any application development effort, many roles, such as project manager, architect, developer, tester, and designer, are involved. Depending on the target application, the role of the designer can greatly vary in the amount of effort required. For an intranet line-of-business (LOB) application, the designer may not have much more of a role than letting developers know where the required corporate application standard resources (such as Cascading Stylesheet [CSS] styles, images, and so forth) are located. In a public-facing rich media application, a designer may be heavily involved, from conception of the application all the way through development and user experience testing.

For Silverlight, the same generalizations apply. You can build powerful desktop-like applications within Silverlight for intranet use that may not require a dedicated designer. Or, you can build rich Internet applications (RIAs) that require dedicated designer skills from start to finish.

The Tools

Silverlight 2 developers do not have to become full-fledged designers, but from a practical standpoint developers will want to become familiar with Expression Blend 2 SP1. As of this writing, the Visual Studio 2008 Silverlight 2 designer surface is a read-only rendering of the edited markup. The Visual Studio 2008 Silverlight 2 designer does not offer the same drag-and-drop support provided for WPF. So, while Visual Studio 2008 includes first-class IntelliSense support, using it can be

tedious for complex UI layout. Expression Blend 2 SPI is a great tool for designing Silverlight 2 UIs, though it does not provide IntelliSense support when viewing the XAML markup.

Also, since the designer is simply a read-only rendering of the markup, the Visual Studio 2008 properties window is nonfunctional as well. Figure 1-9 shows the beginnings of a Silverlight 2 media player in Visual Studio 2008, designed to play the video named `lake.wmv` that ships with Windows Vista.

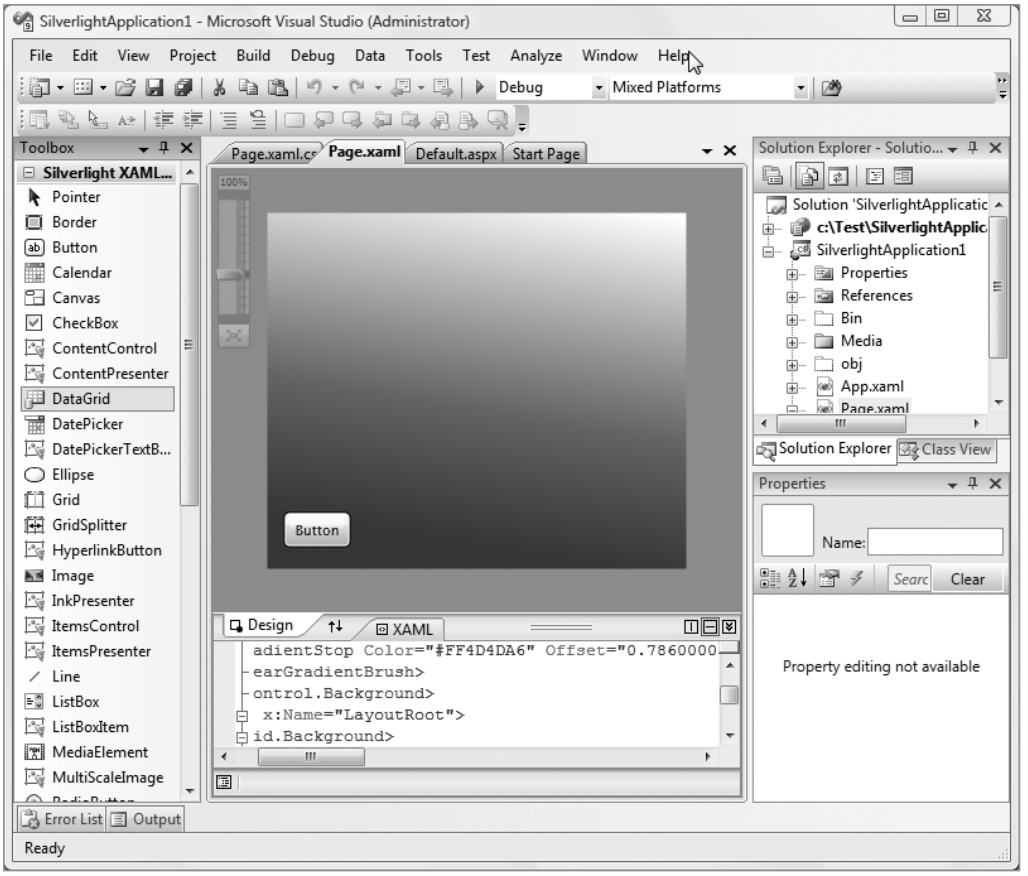

Figure 1-9. *A Silverlight 2 application in Visual Studio 2008*

The simple application has a gradient background, a media element, and a button. Notice that there isn't any design-time rendering of the media element configured to play the video. If you tried to drag and drop a `DataGrid` onto the surface, you would get the "not allowed" mouse symbol. If you tried to select the button to see its properties, they would not show up in the properties window. However, because Visual Studio 2008 is a coding environment, the XAML editing in Visual Studio makes it easy to add markup as well as event handlers to XAML elements and then edit the code.

Expression Blend 2 SP1 provides rich designer support that includes drag-and-drop control editing, visual creation of animation timelines, as well as a rich properties window. Figure 1-10

shows the same application rendered in Expression Blend 2 SP1. You can open a solution in Expression Blend 2 SP1 from within Visual Studio by right-clicking on `Page.xaml` and selecting Open in Expression Blend from the context menu.

Figure 1-10. *A Silverlight 2 application in Expression Blend 2 SP1*

Notice that the media element and video render at design-time in Expression Blend 2 SP1. Also, the properties window provides full access to an element's properties. As in Visual Studio 2008, there is an XAML tab that lets you view the underlying markup in Expression Blend 2 SP1, but there is no IntelliSense support as with Visual Studio 2008.

The Process

At this point, the developer/designer workflow should start to take shape in your mind. UI development is primarily done in Expression Blend 2 SP1 and the coding in Visual Studio. Both Expression Blend 2 SP1 and Visual Studio 2008 can create the initial Silverlight 2 project, but the UI design will most likely start out in wireframe diagrams realized in Expression Design initially and then exported to XAML so that it can be added to the application in Expression Blend 2 SP1. Figure 1-11 provides a visual representation of this iterative process.

For a highly interactive, rich UI, the designer role may want to perform initial layout of the application, as shown in Figure 1-11, developing the initial UI concepts. Unlike when building mockups in an image-editing tool, what is great about Silverlight 2 and Expression Blend 2 SP1 is that the mockup of a visually compelling UI can become the foundation of the actual UI. As Figure 1-11 shows, designers can focus on UI design and usability in the outer loop while the developers focus on writing the code behind the UI as well as the rest of the application. Periodic synchronization points allow the application to be fully integrated between the developer and designer workflows with minimal overhead because of the common underlying markup.

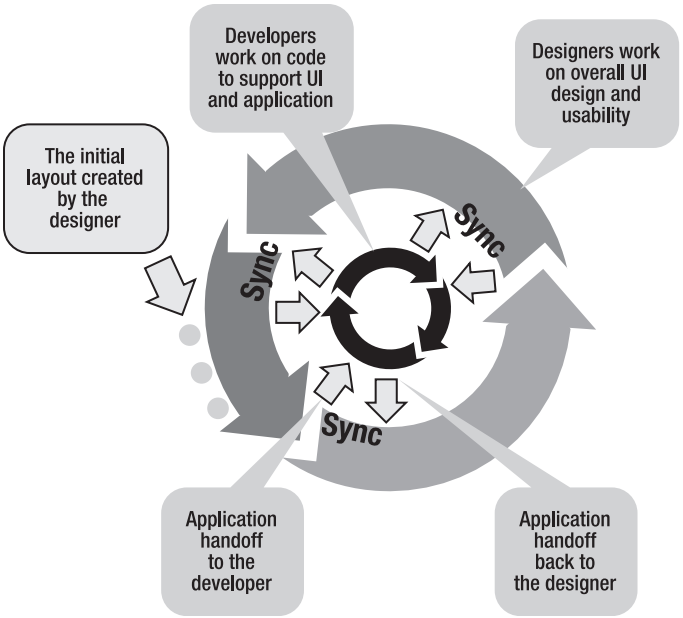

Figure 1-11. *The developer/designer workflow*

Otherwise, a developer can start to build the basic UI with Visual Studio 2008 and then hand off the UI to a designer, who then refines the application's layout, adding animation and control templates. This workflow would make sense if you are migrating an existing .NET application to Silverlight and the developer must first get the code working within the Silverlight programming model, make adjustments, and lay out a basic UI. Developers can also open a Silverlight application in Expression Blend 2 SP1 from Visual Studio 2008 by right-clicking on the `Page.xaml` file and selecting Open in Expression Blend from the context menu. Doing so opens the entire solution in Expression Blend 2 SP1.

A point to emphasize is that unlike with other technologies, the output from the rich design tool, XAML, is what actually is compiled into the application. For comparison purposes, in Windows 32-bit or Windows Forms development a designer cannot create a rich, highly visual control in the technology used by the developer directly. Instead, the designer might use a drawing tool such as Adobe Photoshop to create a mockup. The developer starts from scratch using separate tools and technology and attempts to create a custom control that renders like the mockup. This creates a developer-designer disconnect, or lag, between design changes and coding implementation because the designer and developer work in separate toolsets.

XAML technology enables you to use a wide range of tools since it is well-formed XML. The fact that XAML can be compiled directly permits the developer to take the output from the design team and directly utilize it in the application, completely removing the lag between the designer and the developer.

1-5. Understanding the Basics of Expression Blend 2 SP1

Problem

You need to understand how to create a UI in Expression Blend 2 SP1.

Solution

Learn the basics of the Expression Blend 2 SP1 environment.

How It Works

As mentioned previously, Expression Blend 2 SP1 is a visual design tool that generates XAML. It is a powerful tool that is worthy of a book dedicated to completely understanding the environment. While this book is not exclusively about Expression Blend 2 SP1, we will cover the basics of the environment to help communicate steps when performing tasks visually.

Visual Studio developers may find Expression Blend 2 SP1 to be a dramatic departure from what they are familiar with in Visual Studio 2008. However, developers will want to know how to work in Expression Blend 2 SP1 for maximum productivity. Figure 1-12 shows the Expression Blend UI with major elements pointed out that we cover next.

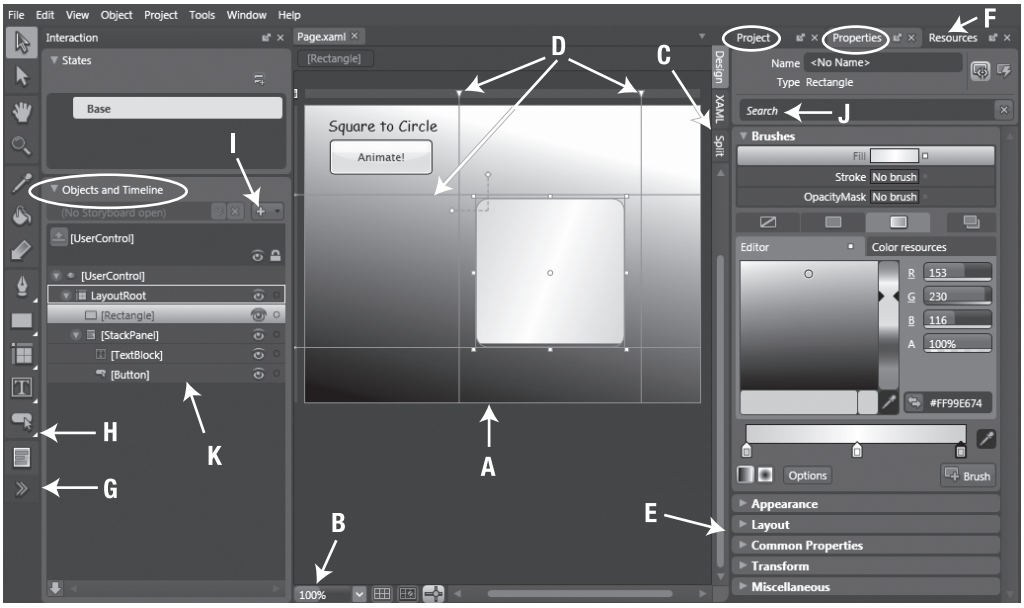

Figure 1-12. *Navigating Expression Blend 2 SP1*

Figure 1-12 shows Expression Blend 2 SP1 with a simple project opened. The project is contrived but suits our purpose of providing an overview of the tool's major features. When the button is clicked, an animation is kicked off that turns the square into a circle and then back into a square again. Table 1-1 provides a quick description of the annotated points.

Table 1-1. *Expression Blend 2 SP1 Features*

Annotation	Description
A	The designer surface, also known in the documentation as the Artboard, which supports drag-and-drop editing.
B	Zoom the designer surface as needed. Zoom out to see the entire application, or zoom in close to perform precise visual editing.
C	Tabs allow you to switch between the design surface, the XAML markup, or split view to see both the design surface and XAML.
D	Represent grid lines for laying out controls in the UI. When you move the mouse over the edge of the Grid control, the UI provides a visual cue that you can add a grid line.
E	The properties window, with several sections collapsed so that they fit in the view.
F	The Resources view, which lists available resources such as styles and templates, We cover these resources throughout the book, starting in Chapter 2 and then in Chapters 4 and 5.
G	This bar, called the Asset Library, is similar to the Visual Studio toolbar area where controls are listed. Clicking on the chevron pointed to by G brings up the Asset Library, where you can search for a control if you are not sure what the icon is or whether it is visible.
H	The little arrow in the lower-right corner under some of the controls shown in the Asset Library is a visual cue that related controls are available for quick access. Clicking and holding on the arrow brings up a small window listing the related controls. Click on the control, and it becomes the visual control for that section of the Asset Library.
I	Clicking this button creates a new `Storyboard` object. Storyboards are where you design animations, such as turning the rectangle into a circle and then back again. We talk more about storyboards later in this chapter.
J	The extremely useful Search text box. Type in a property name and Expression Blend 2 SP1 will search the list of properties available for the control and bring the property into view for easy viewing. Be sure to clear the Search text box when you've finished. Otherwise, it can be confusing when you switch objects and the filter entered in the Search text box does not apply, resulting in a blank properties window.
K	The XAML visual tree is listed in this area of Expression Blend 2 SP1. The yellow frame around the `LayoutRoot` control indicates that the `LayoutRoot` control is the active element. This means that double-clicking on a control in the Asset Library will insert the control as a child to the `LayoutRoot` control. Double-clicking on another control, such as the `StackPanel`, would make it the active element and the insertion point for child controls dragged on the visual design surface.

The Code

At first glance, Expression Blend 2 SP1 looks a lot like Visual Studio with a Project tab (circled in Figure 1-12) that lists the solution, project, and files as in Visual Studio (see Figure 1-13).

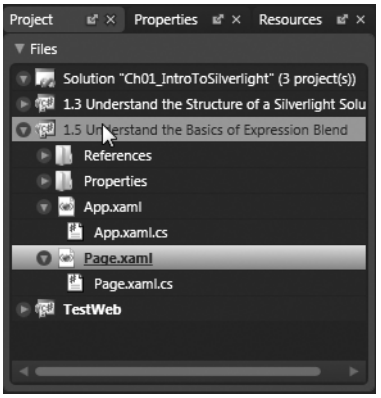

Figure 1-13. *Expression Blend 2 SP1's Project tab*

In Figure 1-12, the "I" points to a button that lets you create a new `Storyboard` object. When you click that button, you are prompted to provide a name or key for the storyboard in the Create Storyboard Resource dialog. Click OK to put Expression Blend 2 SP1 into timeline recording mode, as shown in Figure 1-14.

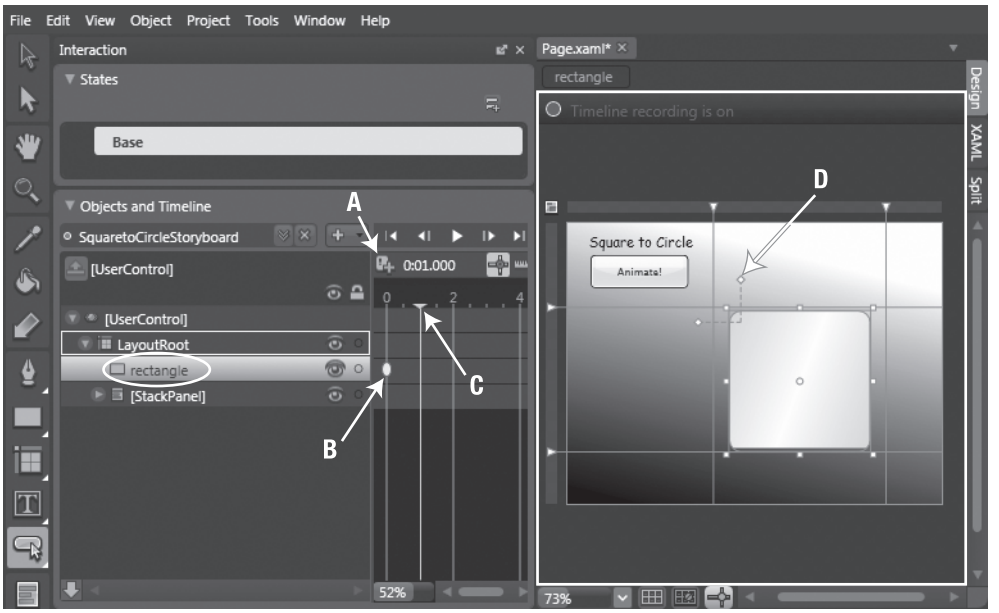

Figure 1-14. *Expression Blend 2 SP1 with timeline recording on*

When Expression Blend 2 SP1 is in timeline recording mode, you can visually create animations. We are now going to create an animation that has four keyframes. We animate a `Rectangle` object in the shape of a square that will transition from a square appearance to a circle appearance between the first and second keyframe. The animation will keep the circle appearance between

the second and third keyframe. Finally, the animation will transition from the circle appearance to a square appearance between the third and fourth keyframe.

To create this animation, click the Record Keyframe button that the letter A points to in Figure 1-14. This creates a keyframe wherever the yellow vertical line in the timeline is located. The letter B in Figure 1-14 points to the keyframe we created at the start time of zero seconds on the timeline. We then drag the yellow vertical timeline pointed to by the letter C to 1 second. Clicking the Record Keyframe button creates a keyframe at that point in the timeline where the yellow vertical line sits, as shown in Figure 1-15.

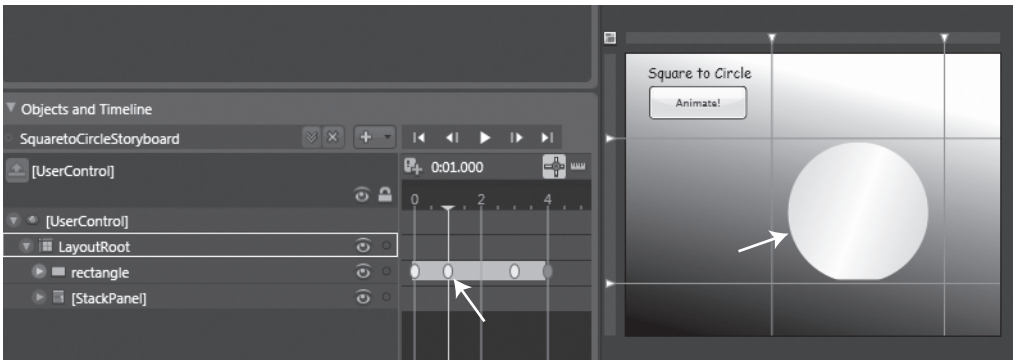

Figure 1-15. *Adding a keyframe to create an animation*

We then adjusted the square to make it look like a circle by dragging the handles pointed to in Figure 1-14 with the letter "D" to create the circle shown in Figure 1-15 at time of 1 second. This results in an animation transitioning from a square to a circle over a period of 1 second. We want the circle appearance to last for 2 more seconds, so we copy the keyframe at 1 second and paste it at a time of 3 seconds on the timeline. This results in the appearance not changing from 1 second to 3 seconds, remaining a circle. We now want the animation to transition back to a square. At a time of 4 seconds on the timeline, we add a copy of the original keyframe at 0 seconds, which is a square. This results in an animation that transitions back to the square appearance between a time of 3 and 4 seconds on the timeline.

A great technique to adopt when you need an animation to "go back" is the use of copy and paste. Notice in Figure 1-15 there are four keyframes for the `rectangle` object in the visual tree. The first keyframe is set at 0 seconds to represent the initial state. At one second, a keyframe is created, as shown in Figure 1-15, with the square now looking like a circle. When this animation runs, the square will smoothly transition into a circle.

The third keyframe shown in Figure 1-15 is a copy of the second `Keyframe`, so that from one second to three seconds, the circle shape is maintained. To copy a `Keyframe`, simply right-click on it and select Copy from the context menu. To paste a copy at 3 seconds, move the yellow vertical timeline to 3 seconds and then press Ctrl+V to paste. The paste location for the keyframe is wherever the yellow vertical line is located along the timeline.

For the fourth `Keyframe`, copy the first `Keyframe` as before, move the yellow timeline to 4 seconds, and then press Ctrl+V to paste. Click the VCR-like play button at the top of the timeline window to test the animation and fine-tune as desired. We cover animations in more detail in Chapter 3, but we wanted to provide an introduction here as part of learning Expression Blend 2 SP1.

The last step is to add code to `Page.xaml.cs` that kicks off the storyboard. To do this, switch to the same solution opened in Visual Studio 2008. Locate the `Button` XAML and type a space inside the first part of the `<Button>` element tag to invoke IntelliSense, as shown in Figure 1-16.

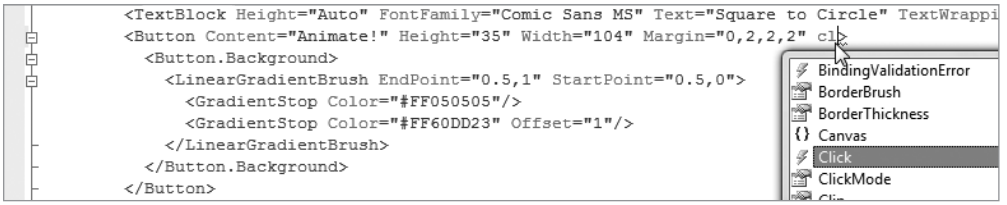

Figure 1-16. *Adding an event in Visual Studio 2008*

It takes one line of code to launch the animation when the button is clicked:

```
SquaretoCircleStoryboard.Begin();
```

Listings 1-5 and 1-6 show the `Page.xaml` and `Page.xaml.cs` files, respectively.

Listing 1-5. *Recipe 1-5 Page.xaml File*

```
<UserControl x:Class="Ch01_IntroToSilverlight.Recipe1_5.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300" xmlns:d=
"http://schemas.microsoft.com/expression/blend/2008" xmlns:mc=
"http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d">
  <UserControl.Resources>
    <Storyboard x:Name="SquaretoCircleStoryboard">
      <DoubleAnimationUsingKeyframes BeginTime="00:00:00"
        Storyboard.TargetName="rectangle"
        Storyboard.TargetProperty="(Rectangle.RadiusX)">
        <SplineDoubleKeyFrame KeyTime="00:00:00" Value="12"/>
        <SplineDoubleKeyFrame KeyTime="00:00:01" Value="75"/>
        <SplineDoubleKeyFrame KeyTime="00:00:03" Value="75"/>
        <SplineDoubleKeyFrame KeyTime="00:00:04" Value="12"/>
      </DoubleAnimationUsingKeyframes>
      <DoubleAnimationUsingKeyframes BeginTime="00:00:00"
        Storyboard.TargetName="rectangle"
        Storyboard.TargetProperty="(Rectangle.RadiusY)">
        <SplineDoubleKeyFrame KeyTime="00:00:00" Value="12"/>
        <SplineDoubleKeyFrame KeyTime="00:00:01" Value="75"/>
        <SplineDoubleKeyFrame KeyTime="00:00:03" Value="75"/>
        <SplineDoubleKeyFrame KeyTime="00:00:04" Value="12"/>
      </DoubleAnimationUsingKeyframes>
    </Storyboard>
  </UserControl.Resources>
  <Grid x:Name="LayoutRoot">

    <Grid.Background>
      <LinearGradientBrush EndPoint="0.810999989509583,0.18299999833107"
        StartPoint="0.630999982357025,1.15100002288818">
        <GradientStop Color="#FF000000"/>
        <GradientStop Color="#FFFFFFFF" Offset="1"/>
      </LinearGradientBrush>
    </Grid.Background>
```

```
    <Grid.RowDefinitions>
      <RowDefinition Height="0.3*"/>
      <RowDefinition Height="0.54*"/>
      <RowDefinition Height="0.16*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="0.39*"/>
      <ColumnDefinition Width="0.461*"/>
      <ColumnDefinition Width="0.149*"/>
    </Grid.ColumnDefinitions>
    <Rectangle  Margin="17.2000007629395,4,17.2000007629395,8" Height="150"
      Width="150" Grid.Column="1" Grid.Row="1" RadiusX="12" RadiusY="12"
               x:Name="rectangle">
      <Rectangle.Fill>
        <LinearGradientBrush EndPoint="1.32400000095367,0.783999979496002"
          StartPoint="-0.310999989509583,0.172000005841255">
          <GradientStop Color="#FF99E674" Offset="0.004"/>
          <GradientStop Color="#FFFFFFFF" Offset="0.504"/>
          <GradientStop Color="#FF99E674" Offset="0.97299998998641968"/>
        </LinearGradientBrush>
      </Rectangle.Fill>
    </Rectangle>
    <StackPanel Margin="8,8,8,8" Grid.Column="0" Grid.Row="0">
      <TextBlock Height="Auto" FontFamily="Comic Sans MS" Text="Square to Circle"
        TextWrapping="Wrap" Width="150" Margin="15,2,2,2"/>
      <Button Content="Animate!" Height="35" Width="104" Margin="0,2,2,2"
        Click="Button_Click">
        <Button.Background>
          <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="#FF050505"/>
            <GradientStop Color="#FF60DD23" Offset="1"/>
          </LinearGradientBrush>
        </Button.Background>
      </Button>
    </StackPanel>

  </Grid>
</UserControl>
```

Listing 1-6. *Recipe 1-5 Page.xaml.cs File*

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

namespace Ch01_IntroToSilverlight.Recipe1_5
{
  public partial class Page : UserControl
  {
    public Page()
    {
      InitializeComponent();
    }
```

```
    private void Button_Click(object sender, RoutedEventArgs e)
    {
      SquaretoCircleStoryboard.Begin();
    }
  }
}
```

This recipe covers the basics to help you get started. We cover Expression Blend 2 SP1 in Chapter 3 as well, but for the most up-to-date information, visit this web site for self-study tutorials, starter kits, training videos, virtual labs, and webcasts: `http://expression.microsoft.com/en-us/cc136522.aspx`.

1-6. Accessing Source Control

Problem

You need to understand how a non–Visual Studio user such as a designer can access a Silverlight 2 project from Team Foundation Server (TFS).

Solution

Use the stand-alone Team Foundation Client Windows application or the TFS Web Access client to connect to TFS and check source code in and out.

How It Works

Given the highly iterative nature that Silverlight development can entail, designers will most likely be accessing the application source code more frequently throughout the development timeline, so it is important that source code integrity be maintained no matter who is working on the application.

Designers will generally spend their time in Expression Blend 2 SP1designing and building Silverlight 2 applications. For most "real" Silverlight applications, developers will want to store source code within TFS or another source code application.

Most if not all source code control applications have stand-alone clients that do not require Visual Studio to access source code. Designers can use the stand-alone client access tools appropriate for their environment. Designers should work with their development team counterparts to obtain the appropriate client for their system.

1-7. Running Silverlight 2 on a Mac

Problem

You need to run Silverlight 2 on a Mac.

Solution

On your Mac, navigate to a web site running Silverlight 2 to automatically download the plug-in or go to `http://www.microsoft.com/silverlight/resources/install.aspx?v=2.0`.

How It Works

Silverlight 2 is a cross-platform, cross-browser plug-in that is designed to automatically install when the web browser accesses a site running Silverlight 2. Note that Silverlight 2 works on Intel-based Mac systems, not PowerPC.

1-8. Running Silverlight 2 on Linux

Problem

You need to run Silverlight 2 applications on a Linux system.

Solution

Download the Moonlight plug-in from `http://www.go-mono.com/moonlight/`.

To access the Moonlight Getting Started page at the Mono project, go to `http://www.mono-project.com/Moonlight#Getting_Started`.

How It Works

In partnership with Microsoft, Novell is providing an implementation of Silverlight 2 for Linux called Moonlight. Moonlight will be available for the major Linux distributions, with support for Firefox, Konqueror, and Opera browsers.

The goal of the implementation is to allow Moonlight to run any Silverlight 2 application without having to recompile the Silverlight 2 application. To view screenshots of Moonlight running existing Silverlight 2 demos, go to `http://www.mono-project.com/MoonlightShots`.