1

Introduction to Silverlight

Announced at the Mix '07 conference in May of 2007, Silverlight took the world by storm with a vision of Adobe Flash–like Rich Internet Applications (RIAs) built using a standards-based, open approach with HTML and XAML using tools like Visual Studio .NET and the new Microsoft Expression Blend. The idea for Silverlight is nothing new; Microsoft had been talking about a technology called WPF/e for the previous 8 or 10 months. Interestingly enough, there was so much confusion around what WPF/e meant, the name needed to change to something completely different, a name that did not associate WPF/e with the smart client, next-generation UI platform Windows Presentation Foundation (WPF). The idea was to tack the letter "e" on the end of WPF to convey Windows Presentation Foundation Everywhere. But this was simply not the case. This WPF is a core part of .NET 3.0 and has a 30MB runtime that requires a Windows-based desktop to run. WPF/e was a 2MB download, ran in the browser, and ran on multiple platforms. And multiple platforms here does not mean Windows XP and Windows Vista; it means Windows and Apple Macintosh. It means the Safari web browser on an Apple Macintosh being served up from an Apache web server running on Linux. Yes, you read that correctly, Linux, Mac, Safari, and Windows, too. There are other significant differences as well, which you'll learn about throughout the next few chapters. Needless to say, the name WPF/e was not going to adequately represent this amazing new technology, so Silverlight was announced as the new name at Mix '07. This chapter does two basic things:

- It gives you an introduction to Silverlight.
- It sets the groundwork with the essentials on creating Silverlight applications that will help you move on to the next chapter and the rest of the book.

What Is Silverlight?

Silverlight is simply a web-based platform for building and running RIAs. The web-based platform part of that equation is essentially the plug-in that runs inside the web browser. Silverlight applications execute within a browser plug-in that installs onto the local machine via the web browser in the exact same manner you install Adobe Flash to run Flash-based animations on web

pages. The Silverlight plug-in supports all of the wow factor that you'd expect from an RIA, such as vector-based graphics and animations, full video integration, and even high-definition video. You can boil down the coolness of Silverlight to the following bullets:

- It's a cross-platform, cross-browser platform for delivering RIAs.
- It supports playback of Windows Media video and audio files on PC and Mac with no dependency on Windows Media Player.
- Using XAML, HTML, and JavaScript, it delivers rich multimedia, vector graphics, animations, and interactivity beyond what AJAX can deliver.
- The installation package is less than 2MB.
- The same XAML created for Silverlight can be used in WPF applications with no changes.

The Silverlight player (or plug-in, or control — those terms are used interchangeably in the book and you will see those variances when others talk about Silverlight as well) itself is a completely standalone environment; there is no dependency on the .NET Framework on the client, on the .NET Framework on the server, or even on a specific version of the .NET Framework. The initial association to the .NET Framework comes from the fact that you are using eXtensible Application Markup Language (XAML) to build the user interface for Silverlight applications, which is of course the foundation of the rich WPF applications built on .NET 3.0. The XAML is downloaded to the browser and executes within the Silverlight runtime on the client. When the XAML is running in the Silverlight player in the web browser, you can access all of the Silverlight objects via JavaScript code that lives in your HTML page. Figure 1-1 demonstrates this interaction between the web browser, the Silverlight player, the XAML, and the HTML page.

You might be asking why Microsoft is pushing out another web-based, client-side technology. There is already ASP.NET, and the ASP.NET AJAX Extensions have been released as well. The simple answer is that users are demanding an even richer experience on the web. Even though AJAX does a lot for improved user experience — the postback nightmare of Web 1.0 is finally going away — it does not do enough. There is demand for a richer, more immersive experience on the web. This has been accomplished with WPF on the rich client side. WPF gives a unified approach to media, documents, and graphics in a single runtime. The problem with WPF, as stated earlier, is that it is a 30MB runtime that runs only on the Windows OS. Microsoft needed to give the same type of experience that WPF offers, only in a cross-platform, browser-delivery mechanism. So what Microsoft did was take the concept of a plug-in model like Adobe Flash, mixed it with the WPF declarative language in XAML, and came up with a way to develop highly rich, immersive, Web 2.0 applications.

Was the Name Silverlight a Surprise?

For the true It-Getters out there (It-Getters are the "ones who get it"), the new name Silverlight was not such a huge surprise. For the previous months of Community Technology Previews (CTPs) of WPF/e, the JavaScript file that contained the core logic to create the browser object that actually runs the WPF/e code was named `agHost.js`. For the savvy web surfers out there, you might recognize "ag" as the Internet domain suffix for Antigua and Barbuda. But in terms of `agHost.js` in a WPF/e project, the "ag" represented the atomic symbol for silver. If you are a developer and a chemist, you might have recognized this. I am not so sure anyone ever actually did, but it does show how clever these names really are.

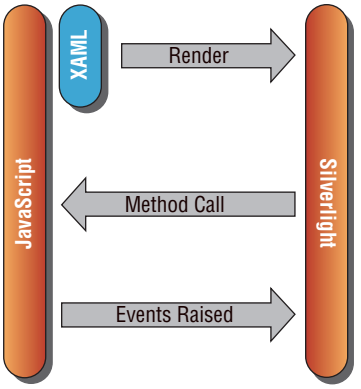

Figure 1-1

The big picture of Silverlight from an architectural perspective is in Figure 1-2. Each area is covered in more detail as you read along in the book, but the important thing to note now is that in using JavaScript you have access to the entire document object model (DOM) of the HTML page that Silverlight is running, which includes Silverlight objects themselves.

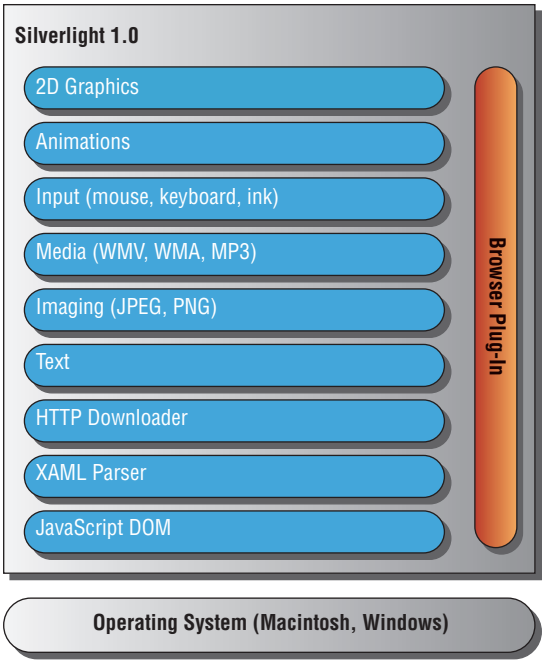

Figure 1-2

As mentioned earlier, Silverlight is fully supported across multiple browsers and operating systems. The current status for browser and OS support is identified in Table 1-1.

Table 1-1: Browser and Operating System Support for Silverlight

Browsers Note that when the Silverlight runtime is installed, a browser plug-in that exposes the Silverlight control is installed. Based on the platform, the control is packaged differently. On Windows, Internet Explorer uses an ActiveX, COM-based model to host Silverlight controls In all other combinations of browsers and platforms the Netscape plug-in technology is used to host Silverlight controls.	Internet Explorer 6 and 7 (Windows) Firefox 1.5.0.8, 2.0+ (Windows and Mac, with Linux support announced) Safari 2.0.4+ (Mac) Konqueror, WebKit, and Opera are being looked at by the Mono team and Novell.
Operating Systems	Windows XP SP2 Windows Vista Mac OS X (10.4.8+) Moonlight, a Linux-based version of Silverlight, is being created by the Mono team at Novell. BSD and Solaris are being discussed for future support, but are not currently available..

What Does DOM Really Mean?

Those new to the web world and programming web pages might not understand what the DOM actually is. In order to effectively work with elements in HTML or XML (the document), the information in the web page needs to be represented in a way that allows programmatic access the document's contents. This document object is the root node and object structure that represents the HTML in the page, and each node or object can have its own properties, methods, and events. Using languages like JavaScript (or even VBScript if you are using Internet Explorer), you can access these nodes and objects via code to manipulate them. This is what is commonly known as *client-side scripting*.

For example, the following code shows how to use the `getElementById` method to return an HTML element that has an ID property set, and then some JavaScript updates the content of the `div` element. When accessing elements via the DOM programmatically, an element must have an ID assigned.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Page 1</title>
</head>

<body>
 <div id="message"></div>
 <script type="text/javascript">
   document.getElementById("message").innerHTML = "This is DOM access";
 </script>
</body>
```

In Chapter 4, you learn more about the DOM and interacting with page elements using JavaScript.

Silverlight Versions Explained

If you have been following Silverlight, you might be a little confused over the versions that are available:

- Currently, **Silverlight 1.0** is the first version of Silverlight and supports the JavaScript programming model that is described in the previous section. This means that your language choice is simple — JavaScript. You use JavaScript to interact with Silverlight objects that are executing within the Silverlight player in the browser.
- **Silverlight 1.1** supports the .NET Framework. This means that you can use any Common Language Runtime (CLR) language to code Silverlight applications, and you have the full power of the .NET Framework to interact with Silverlight objects.

In this book we concentrate on Silverlight 1.0, because it is in full release. Figure 1-3 compares Silverlight 1.0 and 1.1.

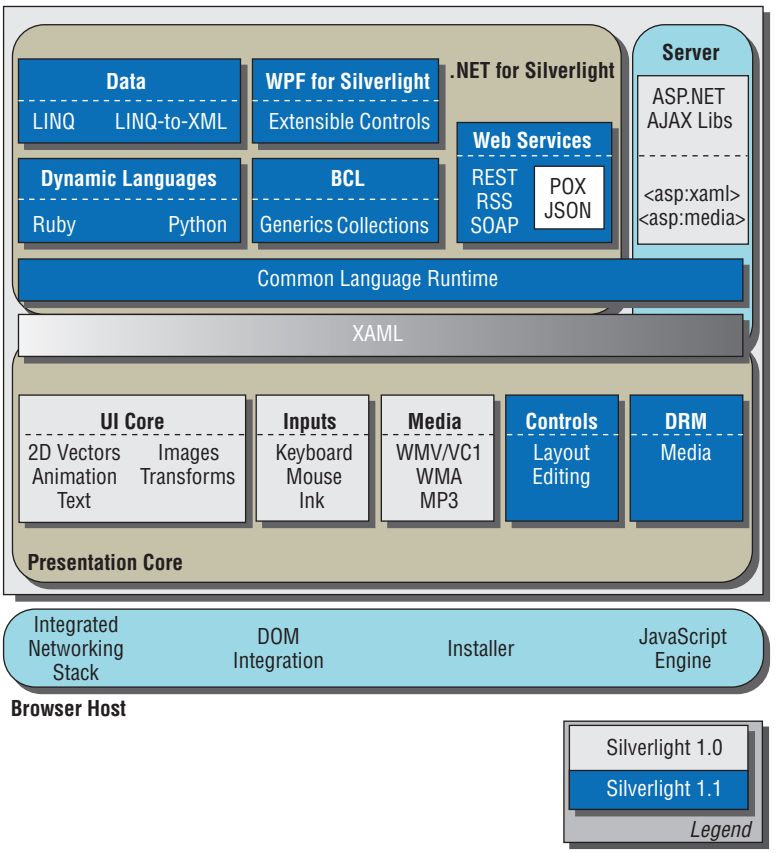

Figure 1-3

In Chapter 6 you get more insight into the next release of Silverlight and how having access to the CLR will impact your Silverlight development.

Getting the Silverlight Plug-In

The first time you navigate to a web page that contains a Silverlight application, the Silverlight player is not installed automatically; the installation is similar to the Adobe Flash experience. There is a nonintrusive image on the page where the Silverlight content is placed to run that gives a link to download the player. Silverlight has two different prompts for install — the standard install and the in-place install:

- In a standard install, the Get Microsoft Silverlight image tells you that you need to install Silverlight to complete the experience on the web page you have arrived at. Figure 1-4 demonstrates a page with the standard install images.

Figure 1-4

- Once you click the Get Microsoft Silverlight install image, you are taken to the Silverlight install page on the Microsoft site, as Figure 1-5 demonstrates.

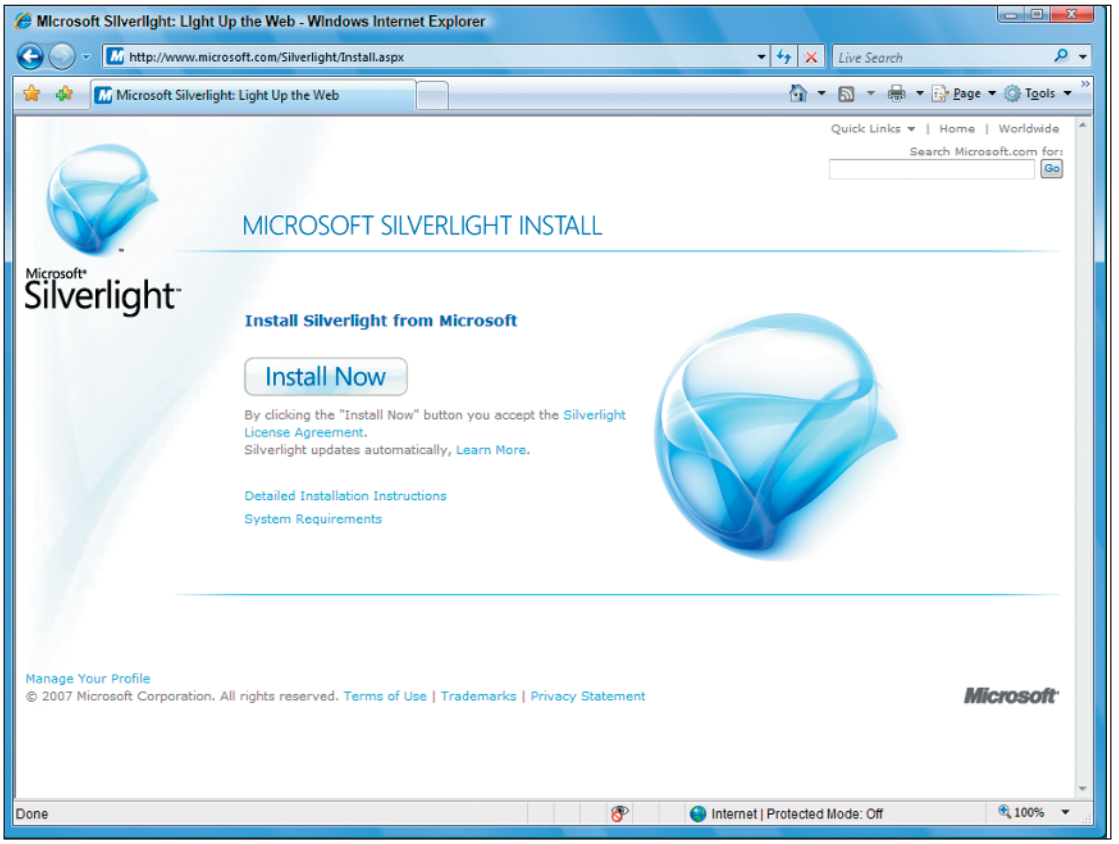

Figure 1-5

- Once you click the Install Now button, you are prompted to either download or install the Silverlight player in the same manner you would get any download from the web. Once you have successfully installed the Silverlight player, you are shown the Welcome to Silverlight page (Figure 1-6), which has a nice animation that shows off the capabilities of Silverlight. There is also a convenient link back to the page that had the Get Silverlight prompts, so the flow of your experience is fairly seamless.
- If a page is using the in-place install, the image is slightly different, and the experience is different as well. During an in-place install, the download begins without having to navigate to a special download page. Figure 1-7 shows the in-place install image. Notice the difference from the standard install image in Figure 1-4.

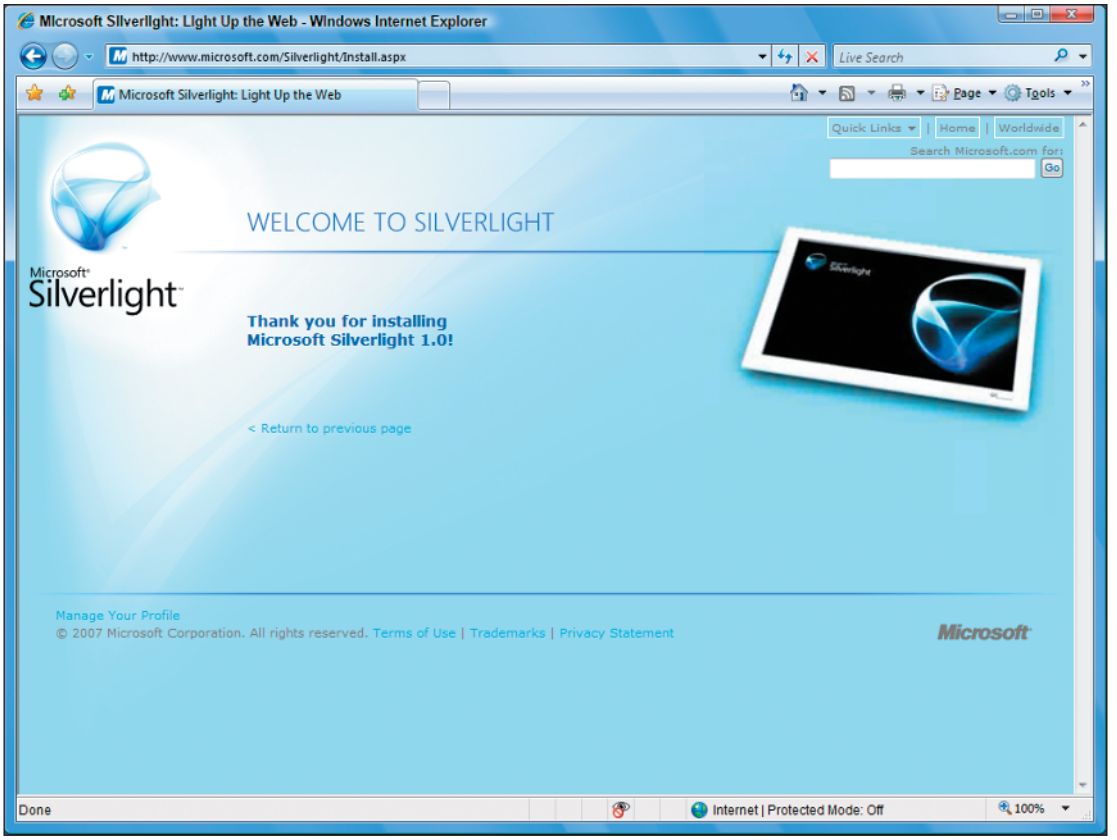

Figure 1-6

Figure 1-7

After the Silverlight player is installed, you never have to install it again. Silverlight also has built into it the knowledge of updates, so once a new version of Silverlight is available, you are asked if you would like to install the update to get the latest version of the player.

Getting the Silverlight SDK

To actually build Silverlight applications, you need more than the Silverlight player itself. If you have not arrived at a page where you are prompted to install the Silverlight runtime, you can easily get it on the Silverlight SDK page. There are also supporting files, help files, samples, and quick starts in the Silverlight Software Development Kit (SDK) that will give you the files you need to start building Silverlight applications. To get the SDK, go to `http://www.silverlight.net/getstarted/default.aspx`, shown in Figure 1-8.

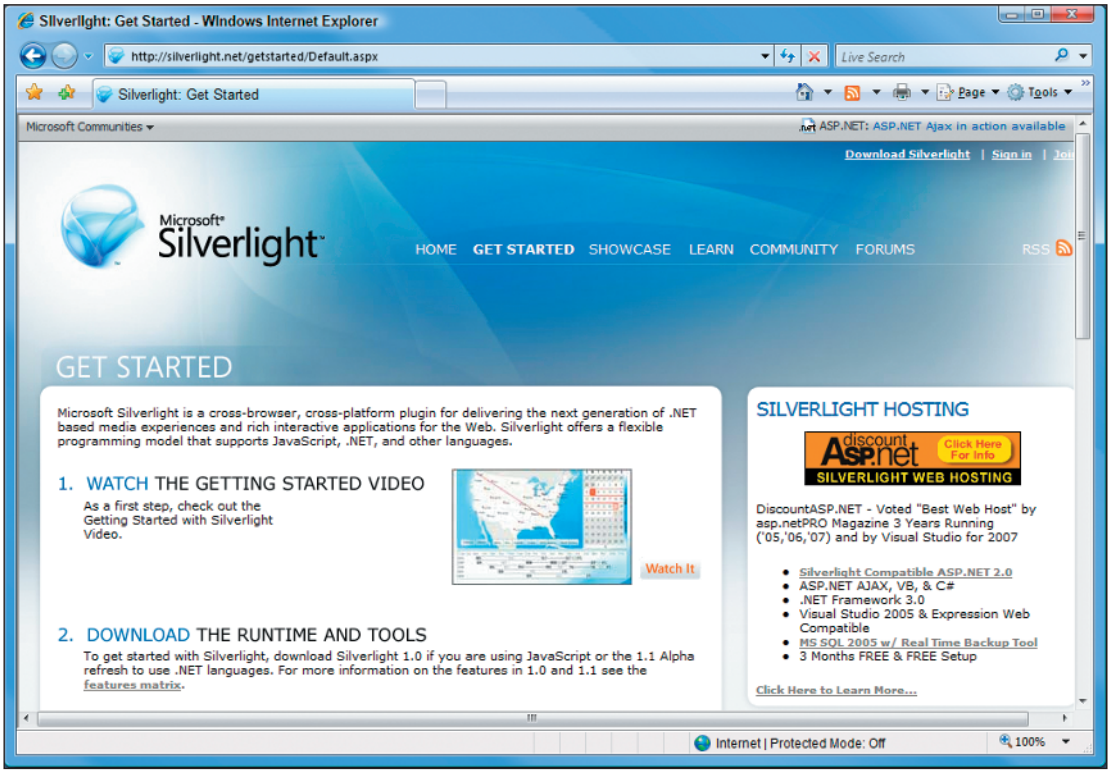

Figure 1-8

On the Silverlight SDK page, you can download all of the tools that you use to create Silverlight 1.0 or 1.1 applications:

- Runtimes:
  - Silverlight 1.0 for Mac and Windows
  - Silverlight 1.1 Alpha for Mac and Windows

- Developer Tools:
  - Microsoft Visual Studio Beta 2
  - Microsoft Silverlight Tools Alpha for Visual Studio
- Designer Tools:
  - Expression Blend
  - Expression Encoder
  - Expression Design
- SDKs:
  - Microsoft Silverlight 1.0 Software Development Kit (SDK)
  - Microsoft Silverlight 1.1 Alpha Software Development Kit (SDK)

If you need to install the 1.0 runtime, you should do that from this page. You should also download the Microsoft Silverlight 1.0 SDK. You need this to get the files and examples required to learn Silverlight. The SDK download is a standard MSI package. It installs the files necessary to create Silverlight applications, including the project templates for Visual Studio. Figure 1-9 is what the folder structure for the installed SDK looks like.

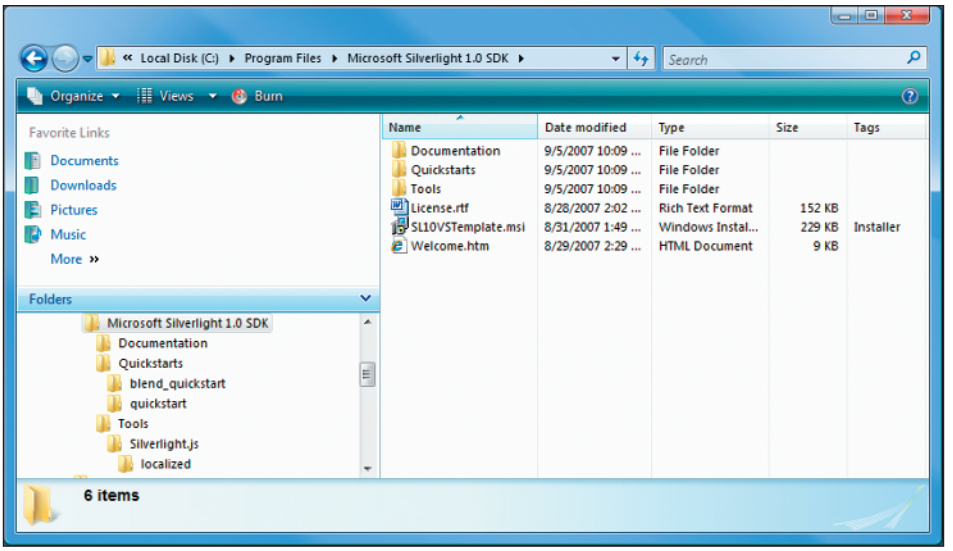

Figure 1-9

To view the samples that ship with Silverlight, you can browse to the community gallery at `http://silverlight.net/community/communitygallery.aspx`. Here you can view and download all of the samples built on Silverlight. Most of the samples are built by Microsoft, but there are also community contributions as well. Because Silverlight 1.0 is based on HTML, JavaScript, and XAML,

you do not need to execute the samples within the context of a web server like IIS. So when you download a sample and run it locally, in each of the project folders there is a Default.html file that you can simply double-click to run the sample. (Note that the Internet Explorer security warning will appear because you are running pages with ActiveX on them locally.)

In Figures 1-10, 1-11, and 1-12, I have highlighted some of the more impressive examples of what you can do with Silverlight. The Page Turner example in Figure 1-10 demonstrates the ability to flip and skew images to mimic the real-life behavior of turning pages in a book.

Figure 1-10

The Grand Piano Player example shown in Figure 1-11 mimics the playing of a keyboard by mapping keys of the piano to their corresponding keyboard values.

Shown in Figure 1-12, SilverPad, an interactive application, lets you view the results of XAML that you add dynamically to the editor area of the application.

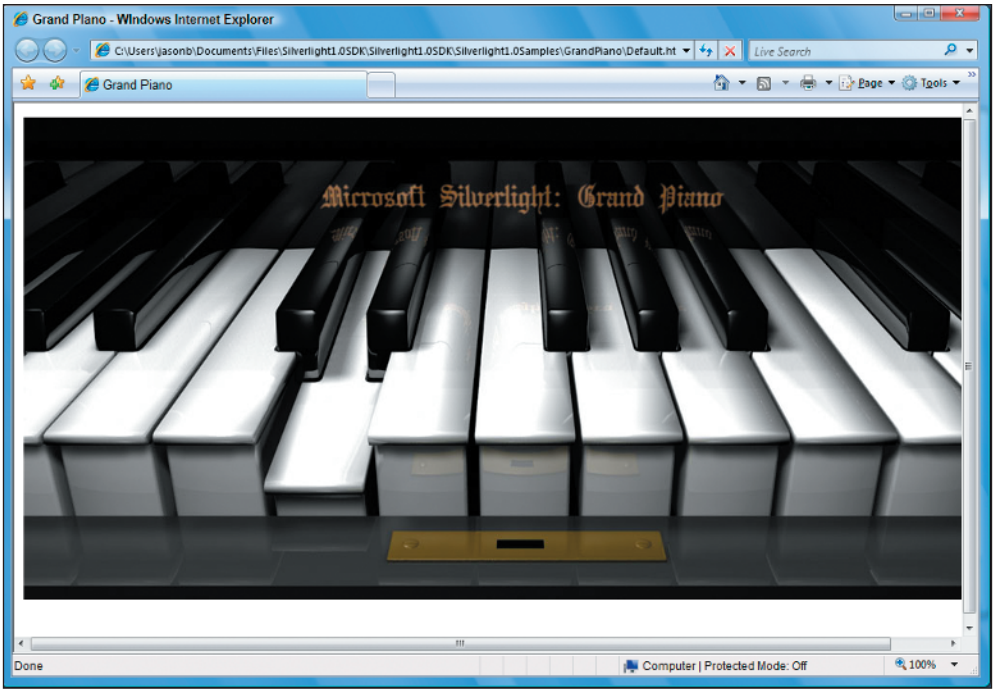

Figure 1-11

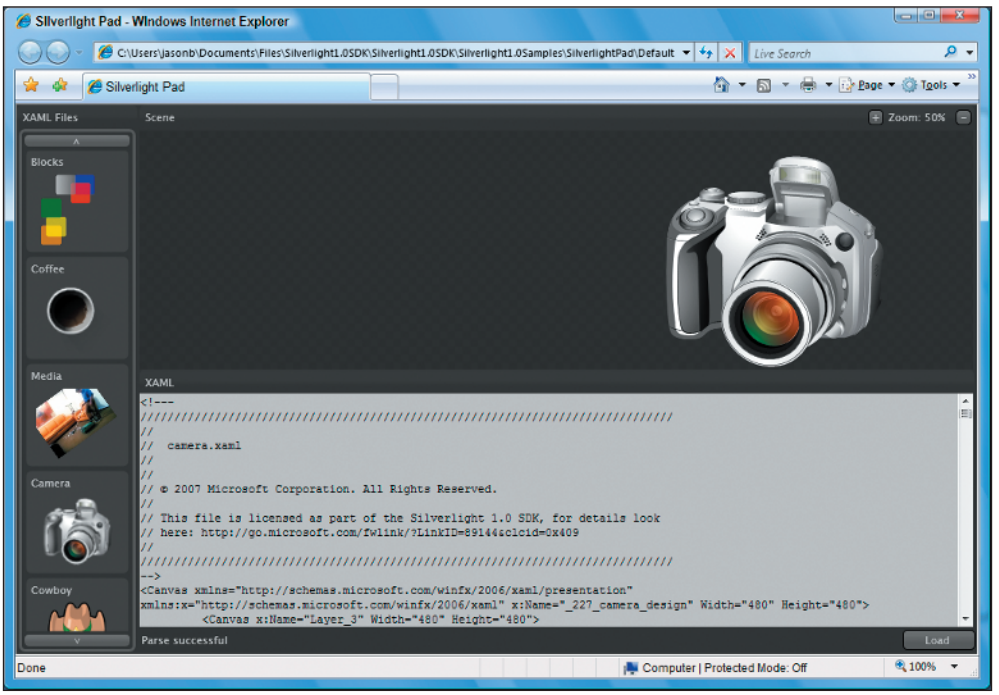

Figure 1-12

Now that you have seen some real-life examples of how powerful Silverlight is, you can start to learn more about building Silverlight applications in the next sections and, subsequently, the next chapters.

Building Silverlight Applications

Now that you have the Silverlight player installed and you have the SDK, you can start building Silverlight applications. There are several ways to create Silverlight applications:

- Visual Studio 2008 Silverlight project template. (These templates target Silverlight 1.1 only.)
- Visual Studio 2005 with the Silverlight 1.0 project templates installed from the Silverlight 1.0 SDK
- Expression Blend 2.0 using the Silverlight 1.0 and Silverlight 1.1 project templates
- Using any text editor to create the HTML, XAML, and JavaScript needed to run a Silverlight 1.0 application

Because this book is focused on Silverlight 1.0, we are going to look at building Silverlight applications using Visual Studio 2005 with the project templates from the SDK. The project template is installed when you run the initial SDK install. If you chose not to install the project template during the install, there is a package you can run from the Microsoft Silverlight 1.0 SDK from the Start Menu that will install it for you.

To start, open Visual Studio 2005 and hit Ctrl+Shift+N to open the New Project dialog. Select Visual C# Projects, because there is no template for Visual Basic in the SDK. Even if you are a hard-core VB developer, it does not really matter what language the template is under because you are only using HTML, JavaScript, and XAML to build Silverlight 1.0 applications. The New Project dialog should look something like Figure 1-13, with the Silverlight Javascript Application template under My Templates. This is the template extracted from the zip file you copied to the templates' folder previously.

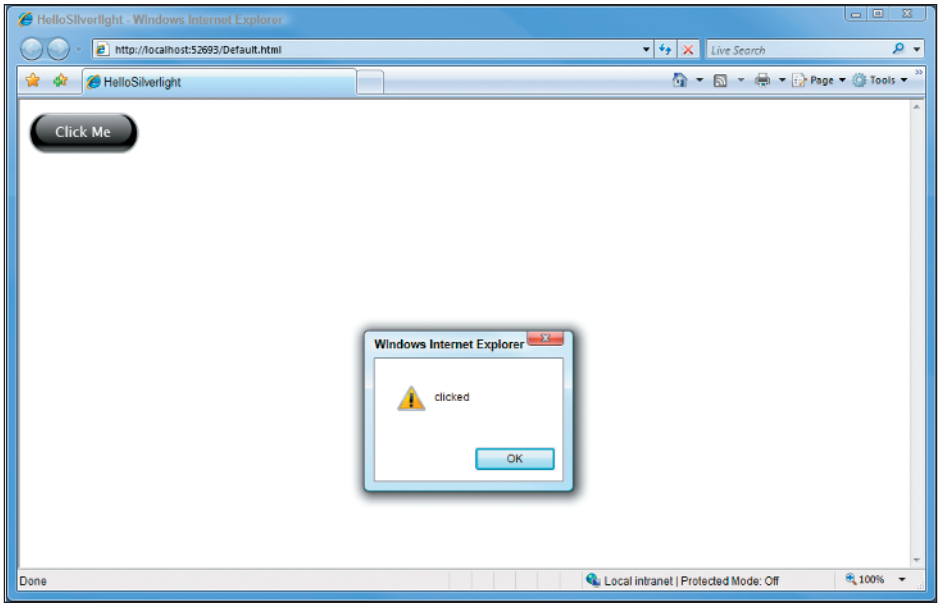

Figure 1-13

Change the Name of the project to HelloSilverlight, as Figure 1-13 shows, and click the OK button to create the solution. You should be looking at Visual Studio now, with a Solution Explorer that looks similar to Figure 1-14.

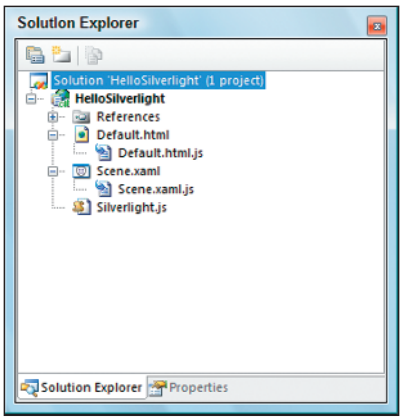

Figure 1-14

Table 1-2 outlines the files in the solution, along with their definitions.

Table 1-2: Default Files in a Silverlight Project and Their Descriptions

File	Description
`Default.html`	Default HTML file that hosts the Silverlight player and contains references to the JavaScript files required by the Silverlight player.
`Default.html.js`	The JavaScript file associated with the `Default.html` file.
`Scene.xaml`	The default XAML file that contains the objects and elements that make up the user interface that plays in the Silverlight player that is embedded in `Default.html`.
`Scene.xaml.js`	The JavaScript file for the control events specified in the XAML file. This can be likened to the code behind file of a `.CS` file.
`Silverlight.js`	The core JavaScript library that ensures the Silverlight player is on the client machine and then creates the Silverlight object that runs in the page.

Now that you have a little bit of an understanding of the files in the solution, go ahead and run the project to see how the default Silverlight template works. You should see a web page with a button on it. If you click the button, you should see something similar to Figure 1-15.

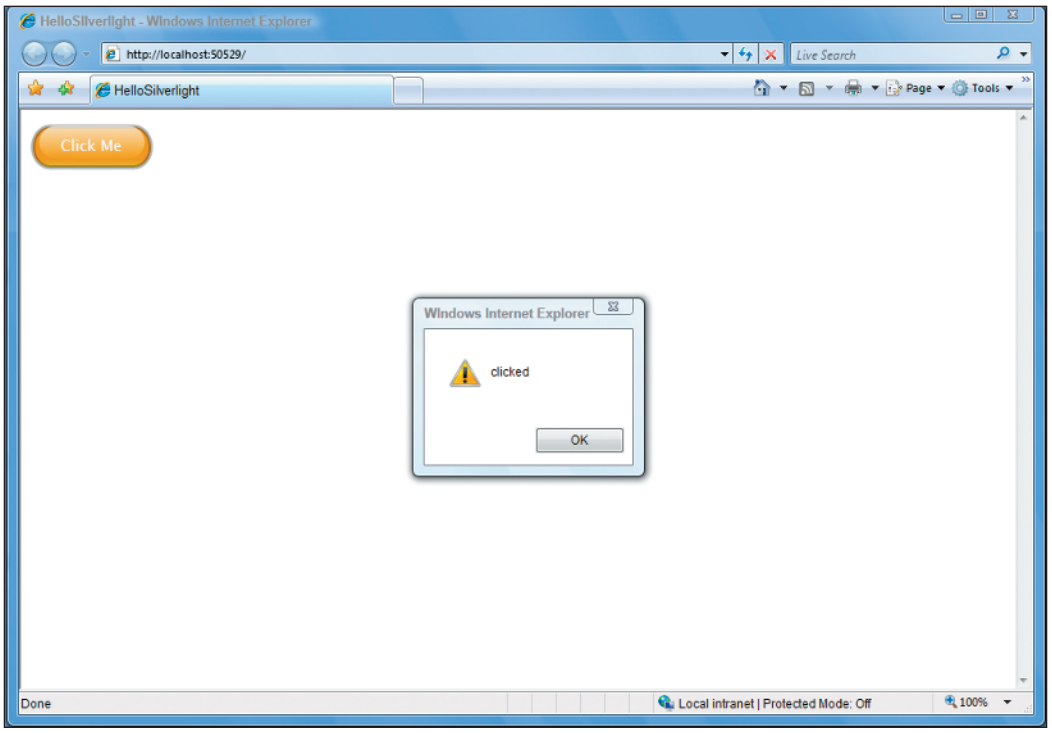

Figure 1-15

If you look at the HTML (Listing 1-1) in the `Default.htm` file, you will notice that there is no ActiveX control on the page. There are, however, several references to JavaScript files that contain the code that is needed to create the instance of the Silverlight player in the HTML page.

Listing 1-1: HTML from the Default.htm File That Hosts the Silverlight Control

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
    Transitional//EN" "http://www.w3c.org/TR/1999/REC-html401-19991224/loose.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>HelloSilverlight</title>

    <script type="text/javascript" src="Silverlight.js"></script>
    <script type="text/javascript" src="Default.html.js"></script>
    <script type="text/javascript" src="Scene.xaml.js"></script>
```

Listing 1-1: *(continued)*

```
</head>

<body>
    <div id="SilverlightControlHost">
        <script type="text/javascript">
            createSilverlight();
        </script>
    </div>
</body>
</html>
```

The embedded script in the `<body>` tag contains a call to the JavaScript function `createSilverlight()`, which is located in the `default.html.js` file, is listed in Listing 1-2, and is responsible for creating the Silverlight object in the HTML page.

Listing 1-2: The createSilverlight Function

```
function createSilverlight()
{
    var scene = new HelloSilverlight.Scene();
    Sys.Silverlight.createObjectEx({
       source: "Scene.xaml",
        parentElement:
        document.getElementById("SilverlightControlHost"),
        id: "SilverlightControl",
        properties: {
            width: "400",
            height: "400",
            version: "0.9"
        },
        events: {
            onLoad:
            Sys.Silverlight.createDelegate
            (scene, scene.handleLoad)
        }
    });
}
```

The core code in the JavaScript file is the call to `Sys.Silverlight.createObjectEx`. This function, shown in Figure 1-16, is responsible for creating the actual Silverlight object and returning it to the page. It also makes sure that the Silverlight control is installed on the client computer, so when it creates the object and accepts the parameters passed in, there are no errors.

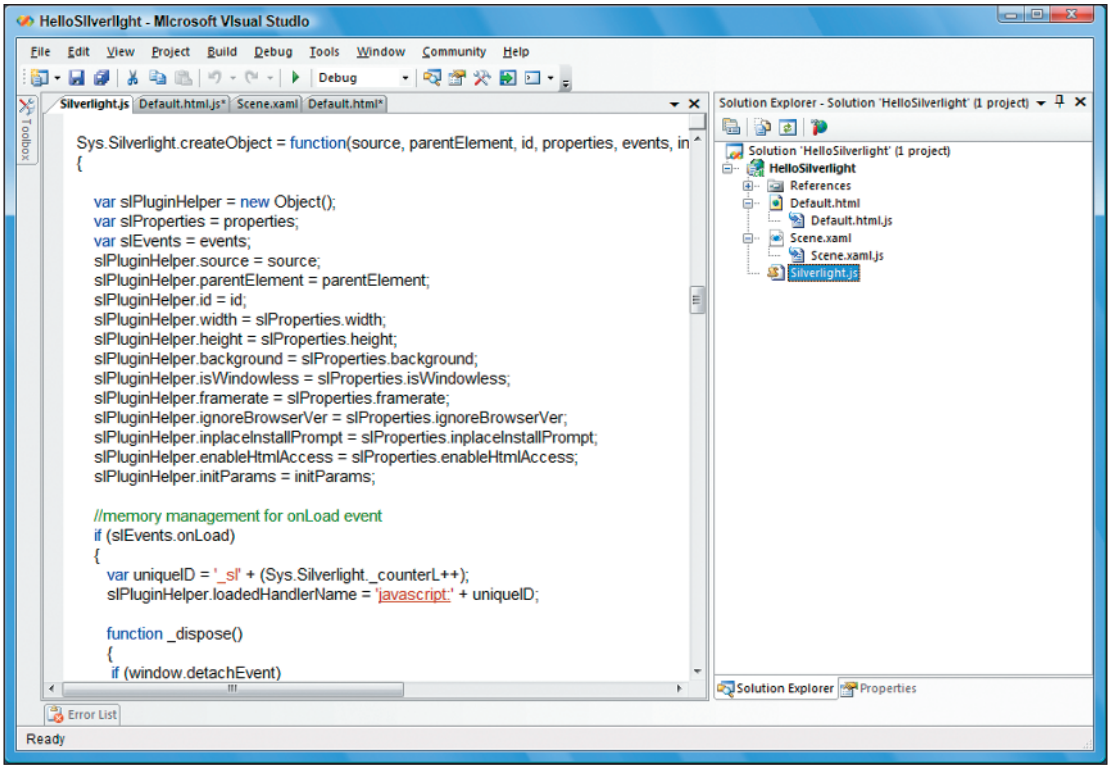

Figure 1-16

The `Sivlerlight.js` file is the core JavaScript needed to instantiate and run the Silverlight player in the browser. It is required in any Silverlight applications you create.

Creating the Silverlight Object in HTML

There are other ways to create the Silverlight player; you do not have to use the method that the default template uses. You can simply use the `object` or `embed` tag in your HTML page to get a Silverlight player. The HTML in Listing 1-3 demonstrates using the `object` tag to create an instance of a Silverlight player named `silverlight1`.

Listing 1-3: Creating a Silverlight Control Using the object Tag in HTML

```
<object
  type="application/ag-plugin"
  id="silverlight1"
  width="350"
  height="350">
  <param name="background" value="#ffebcd" />
  <param name="enableFramerateCounter" value="true" />
```

Listing 1-3: *(continued)*

```
    <param name="enableHtmlAccess" value="true" />
    <param name="initParams" value="paramValue1, paramValue2" />
    <param name="maxFrameRate" value="30" />
    <param name="onError" value="myErrorHandler" />
    <param name="onLoad" value="onLoad" />
    <param name="source" value="HelloSilverlight.xaml" />
    <param name="windowless" value="true" />
  </object>
```

The `object` tag works only in Internet Explorer browsers, whereas the `embed` tag is used for Firefox and Macintosh browsers. The HTML listed in Listing 1-4 is an example of creating a Silverlight player instance using the `embed` tag.

Listing 1-4: Creating a Silverlight Control Using the embed Tag in HTML

```
<embed
  type="application/ag-plugin"
  id="silverlight1"
  width="350"
  height="350"
  background="#ffebcd"
  enableFramerateCounter="true"
  enableHtmlAccess="true"
  initParams="paramValue1, paramValue2"
  maxFrameRate="30"
  OnError="myErrorHandler"
  OnLoad="onLoad"
  source="HelloWorld.xaml"
  windowless="false"
 />
```

The obvious observation is that you would probably not want to do this yourself, because you are limiting yourself to a specific browser type by using `embed` or `object` tags. Your best bet is to use the technique in the default Silverlight project template, which uses generic JavaScript to call the `createObjectEx` function in the `Silverlight.js` file. This ensures you are not making any mistakes in how the Silverlight player is created in the client browser.

Silverlight Object Attributes

In the various examples you have just read about that create the Silverlight player in the browser, you've seen that you can also pass parameters to the object during its creation to set various properties on the control. Table 1-3 (which paraphrases a table that can be found online at `http://msdn2.microsoft.com/library/en-us/bb412394.aspx`) lists the attributes you can use on the Silverlight control when you are creating it.

Table 1-3: Object Attributes for the Silverlight Control

Tag	Type	Description
`Type`	String	The MIME type for the installed plug-in.
`Id`	String	Gives the Silverlight control a unique identifier, because there can be multiple controls on a page.
`width`	String	The rectangular width that displays the Silverlight content in pixels. You can also specify the width as a percentage of the displayable area of the tag containing the Silverlight control — for example, `"40%"`.
`Height`	String	The rectangular region that displays the Silverlight content in pixels. You can also specify the height as a percentage of the displayable area of the tag containing the Silverlight control — for example, `"50%"`.

In addition to the object tag parameters in Table 1-3, the `object` tag also supports a set of optional parameter attributes for the control. Table 1-4 (which paraphrases a table that can be found online at `http://msdn2.microsoft.com/library/en-us/bb412394.aspx`) lists the optional parameter tags that map to properties on the Silverlight control.

Table 1-4: Optional Parameter Tags for the Silverlight Control

Parameter Tag	Type	Description
`background`	String or hexadecimal	The background color of the rectangular region that displays XAML content. The default value is `null`, which is equivalent to the color white.
`enableFramerateCounter`	Boolean	Indicates whether to display the current frame rate in the hosting browser's status bar. The default value is `false`.
`enableHtmlAccess`	Boolean	Determines whether the hosted content in the Silverlight control has access to the browser DOM. The default value is `true`.
`initParams`	String	Specifies an optional set of user-defined initialization parameters.
`maxFramerate`	Integer	Specifies the maximum number of frames to render per second, which defaults to `24`.

Table 1-4: *(continued)*

Parameter Tag	Type	Description
`onError`	String	Specifies the JavaScript error handling function for the `OnError` event. The default value is `default_error_handler`.
`onLoad`	String	Specifies the JavaScript event handling function for the `OnLoad` event. `null` is the default.
`source`	String	Specifies the XAML content that is rendered.
`windowless`	Boolean	Determines whether the Silverlight control displays as a windows-less control. The default value is `false`.

Adding Multiple Silverlight Objects to a Page

Just like any other element or object on an HTML page, you can run multiple Silverlight players on a page as well. To enable multiple Silverlight players on a page, make sure there is a unique `<div>` tag with a unique ID representing the player you are creating.

```
<div id="SilverlightControlHost">
    <script type="text/javascript">
        createSilverlight();
    </script>
</div>
```

In the default `createSilverlight` function, the name of the `<div>` element is hard coded, as demonstrated in Listing 1-5.

Listing 1-5: createSilverlight Function Showing the createObjectEx Method Call

```
function createSilverlight()
{
    var scene = new HelloSilverlight.Scene();
    Sys.Silverlight.createObjectEx({
        source: "Scene.xaml",
        parentElement:
        document.getElementById("SilverlightControlHost"),
        id: "SilverlightControl",
        properties: {
            width: "400",
            height: "400",
            version: "0.9"
        },
        events: {
            onLoad:
```

Notice the XAML file is also hard coded, as is the control host you are attempting to pass the Silverlight player back to. To ensure there are no errors, you may want to consider creating a generic `createSilverlight` function that accepts parameters for the information that you need to make unique, as the code in Listing 1-6 demonstrates.

Listing 1-6: Sample of Parameterized createSilverlight Function

```
function createSilverlight(xamlFile, hostId, controlId, height, width)
{
    var scene = new HelloSilverlight.Scene();
    Sys.Silverlight.createObjectEx({
        source: xamlFile,
        parentElement:
        document.getElementById(hostId),
        id: controlId,
        properties: {
            width: height,
            height: width,
            version: "0.9"
        },
        events: {
            onLoad:
```

As you can see, adding multiple Silverlight players to a page is pretty simple, and in many cases, will be the default scenario.

Resizing a Silverlight Control

The Silverlight control provides both resize capabilities and full screen mode. When the control is displayed in embedded mode, it is playing in the default height and width specified when it was created. In embedded mode, the `OnResize` event fires, which allows you to write code that resizes the player and the elements in the player.

The JavaScript in Listing 1-7 shows how to define an `OnResize` event for a Silverlight control.

Listing 1-7: Resizing a Control Using the OnResize Event

```
// function call from the OnResize event
function ResizeControl(width, height)
{
    // Reference the control in the DOM
    var control = document.getElementById("silverlight1");

    // Set the size
    control.width = width;
    control.height = height;
}
```

Whenever the `actualHeight` or `actualWidth` property of the Silverlight control changes, the `OnResize` event occurs. An effective way to assign a function to this event is during the control's creation. For example, you can assign the `ResizeControl` function in the `createSilverlight` function described earlier in this section:

```
silverlightObject.content.onResize = "ResizeControl"
```

When resizing a Silverlight control or when switching to full screen mode, the performance of any running video or animations should not be affected.

In full screen mode, the player is displayed in the full height and width of the screen's resolution on top of all other running applications. When the player goes from embedded mode to full screen mode, the `OnResize` event does not fire, but the `ActualHeight` and `ActualWidth` properties do change. They will reflect the height and width of your screen's resolution.

Figure 1-17 demonstrates a Silverlight player embedded in the browser, and Figure 1-18 shows the same Silverlight player in full screen mode.

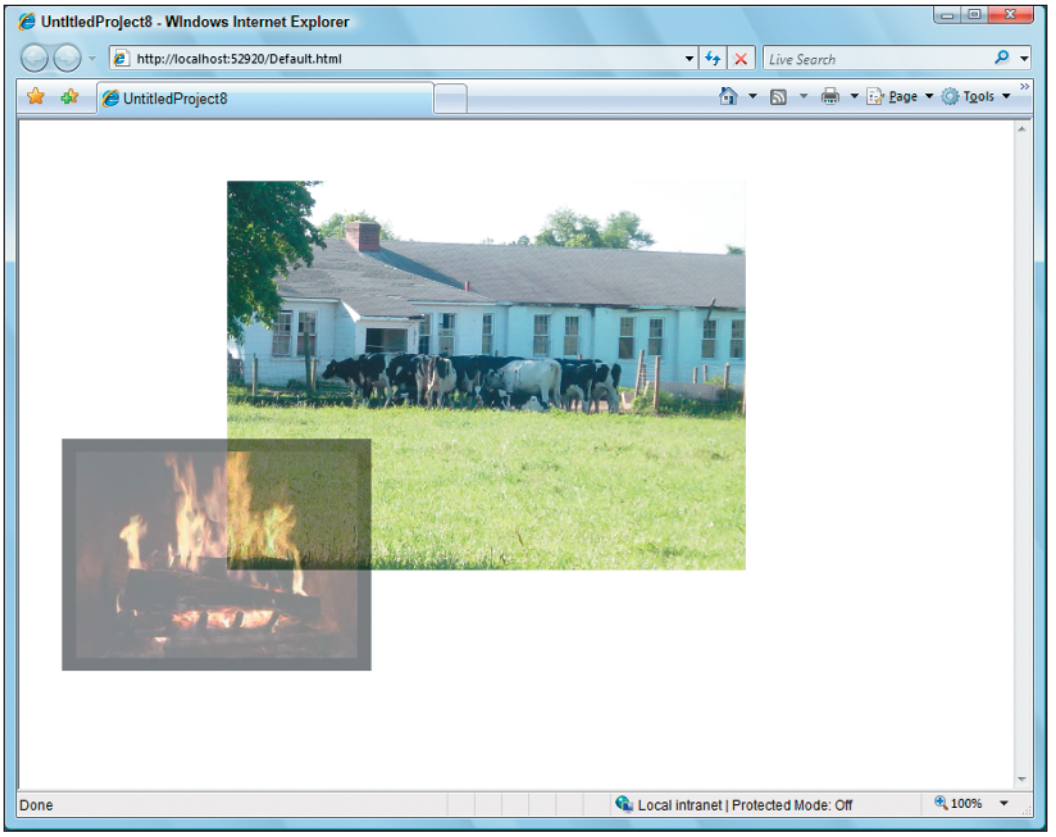

Figure 1-17

Figure 1-18

Displaying in Full Screen Mode

To display a Silverlight player in full screen mode, the `fullScreen` property must be set to `true`. If the `fullScreen` property is not set to `true`, the Silverlight control assumes its normal height and width and displays in embedded mode. Note that only one Silverlight player can be in full screen mode at a time; if a control is displayed in full screen mode, it must be returned to embedded mode before another control can go to full screen. Listing 1-8 shows how to use JavaScript to get the Silverlight player into full screen mode.

Listing 1-8: Using JavaScript to Get the Silverlight Player into Full Screen Mode

```
function GoFullScreen(sender, eventArgs)
{
        host = sender.getHost();
        host.content.fullscreen = true;
}
```

To use the function, you would need to specify an event on an XAML element. For example, because the user needs to trigger the action to go into full screen, you might have a `TextBlock` or a `Rectangle` element that indicates to users that if they click on the element, the browser goes into full screen mode. Notice in the following XAML the call to the `GoFullScreen` function on the `MouseLeftButtonUp` event.

```
<Rectangle Height="100" Width="100"
           Canvas.Top="100" Canvas.Left="100"
           Fill="Red"
           MouseLeftButtonUp="GoFullScreen" />
```

Note that you cannot set the mode to full screen in the `Loaded` event of the control. It must be set in response to user actions, such as the `MouseLeftButtonDown`, `MouseLeftButtonUp`, `KeyDown`, and `KeyUp` events. The following code toggles the screen mode based on the `MouseLeftButtonUp` event:

```
function GoFullScreen(sender, eventArgs)
{
        host = sender.getHost();
        host.content.fullscreen = !host.content.fullscreen;
}
```

When a Silverlight player switches to full screen, the message demonstrated in Figure 1-19 is displayed for several seconds, giving the user instructions on how to return to embedded mode.

Figure 1-19

Once a Silverlight player is in full screen mode, keyboard events are prevented from being passed on to keyboard event handlers in the application. The only valid keyboard input that is acted upon is the set of keystrokes that return the Silverlight control to embedded mode. This limitation of keyboard input during full screen mode is a security feature and is intended to minimize the possibility of unintended information being entered by a user. The other way to return from full screen mode to embedded mode is to simply use the following keystroke combinations:

- **Windows** — Esc, Ctrl+W, or Alt+F4
- **Macintosh** — Esc

Understanding XAML

The one file we have not touched on yet is the `Scene.xaml` file from the HelloSilverlight sample. The XAML file is the most important part of the Silverlight application. It is the markup that defines what should actually get displayed in the Silverlight player itself. The XAML in the `Scene.xaml` file is fairly complex for your first Silverlight application, so we are not going to delve into the details of what the XAML is actually doing. Over the next several chapters, you will get a better understanding of what XAML can do for you in Silverlight, the various ways you can get XAML into the Silverlight player, and how you can create very interactive user interfaces with not so complex XAML. For now, you'll just get an understanding of XAML and look at a couple of examples of how it can be used in Silverlight.

If you are not familiar with WPF, you are probably not familiar with XAML. Since the dawn of Visual Studio, way back around 1997 or so when when we used its precursor Visual InterDev, there was always the promise of code and user interface design separation. This means that a developer can write code, while a designer just works on the design and layout aspects of an application. This has never been realized, mostly because a developer and a designer are always using different tools and different languages. With the introduction of XAML, there was finally a unified markup that could not only describe what a control is and how it fits into a page, but also how layout, and more importantly, the overall look and feel of the controls on a page are defined. A designer can use XAML to create a mock-up of a page or an application, and a developer can take that XAML markup and use it directly in his project files. Because partial classes and code behind files in Visual Studio .NET allow you to separate the code logic from the layout and control definitions, using XAML gives the opportunity to have this separation of the design from the code. To give you an idea of how XAML works, look at the code in Listing 1-9. This is an XAML snippet from a Silverlight application that shows Hello World in a `TextBlock`.

Listing 1-9: Basic XAML to Display Hello World in a Silverlight Player

```
<Canvas x:Name="parentCanvas"
        xmlns="http://schemas.microsoft.com/client/2007"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Width="640"
        Height="480"
        Background="White"
        >
    <TextBlock>Hello World</TextBlock>
</Canvas>
```

Listing 1-10 shows how the XAML can get more complex, demonstrating adding various animations to the text block control. In this example, four different transforms are occurring:

- `ScaleTransform` — Stretches or shrinks an object horizontally or vertically.
- `SkewTransform` — Creates the illusion of three-dimensional depth in a two-dimensional object.
- `RotateTransform` — Rotates an object by a specified angle around the CenterX and CenterY of an object.
- `TranslateTransform` — Translates, or moves, an object in the two-dimensional x-y coordinate system.

Listing 1-10: XAML Showing a Custom Transform on a TextBlock Element

```
<Canvas
    xmlns="http://schemas.microsoft.com/client/2007"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="640" Height="480"
    Background="White"
    >
    <Canvas.Triggers>
        <EventTrigger RoutedEvent="Canvas.Loaded">
            <BeginStoryboard>
                <Storyboard x:Name="Timeline1"/>
            </BeginStoryboard>
        </EventTrigger>
    </Canvas.Triggers>
    <TextBlock Width="349" Height="67"
        Canvas.Left="150"
        Canvas.Top="140" Text="Hello World"
        TextWrapping="Wrap"
        RenderTransformOrigin="0.5,0.5"
        x:Name="textBlock">
        <TextBlock.RenderTransform>
            <TransformGroup>
                <ScaleTransform ScaleX="1" ScaleY="1"/>
                <SkewTransform AngleX="0" AngleY="0"/>
                <RotateTransform Angle="0"/>
                <TranslateTransform X="0" Y="0"/>
            </TransformGroup>
        </TextBlock.RenderTransform>
    </TextBlock>
</Canvas>
```

Appendix B gives a more in-depth explanation of XAML and how you can use it to define and create your Silverlight applications. You will also be getting your fair share of XAML throughout the book, because it is how you will create most of the examples that we have defined. Tools like Microsoft Expression Blend 2.0, Visual Studio 2005 with the Silverlight project support, and Visual Studio 2008 are all Rapid Application Development (RAD) tools that you can use to create your Silverlight applications. The problem right now is that Microsoft Expression Blend 2.0 is really the only tool that gives you a nice design-time experience, where you can drag and drop controls onto the design surface and switch between the designer view and the XAML view. We know these problems will work themselves out. As Silverlight matures and goes into a release stage, and as Visual Studio 2008 nears release, the appropriate tools will be released that will let you design applications in a RAD fashion. The good thing about not having the perfect tool right now to create Silverlight applications is that you are forced to type XAML, which will give you the understanding you need to know what is happening behind the scenes of the runtime player.

Summary

This chapter gave you the groundwork you need to move forward in the book. You learned that Silverlight is a rich platform for delivering next-generation RIAs based on XAML, HTML, and JavaScript. Using tools like the Silverlight project template in the Silverlight 1.0 SDK, it is easy to get started writing Silverlight applications. In the next chapter, you continue where you left off here with more information on creating the user interface using XAML. If you ever need a reference on XAML, refer to Appendix B to find out additional information on XAML.