Part I: Getting Started

Chapter 1: Introduction to Silverlight

Chapter 2: XAML Basics

Chapter 3: Silverlight Architectural Tour

Chapter 4: Silverlight Developer Toolbox

1

Introduction to Silverlight

Silverlight 3, the third iteration of the Silverlight platform, continues to deliver on the promise of Adobe Flash–like and Flex-like rich Internet applications (RIAs) built using a standards-based, open approach with HTML and XAML using tools like Visual Studio 2008 and Microsoft Expression Blend. Silverlight 3 adds more excitement to RIA development with the inclusion of a subset of the Base Class Libraries (BCLs) from the .NET Framework, new user interface controls, and new libraries for building line-of-business applications. The result is that not only do you have the rich, XAML markup to describe expressive user interfaces, but you also have the power of the .NET Framework and your language of choice (C#, VB, etc.) to build Silverlight applications. Even with the .NET Framework libraries, Silverlight still retains the cross-browser and cross-platform compatibility that it has had since the beginning. This includes Windows 2000, Windows XP, Windows Vista, Windows 7, Macintosh, and, through the Mono Project, various Linux distributions. You can build a Silverlight application and run it in a Safari Web browser on an Apple Macintosh, while being served up from an Apache Web Server running on Linux. There is a lot to learn about Silverlight, and you'll gain more and more insight with each chapter in this book.

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 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 an ActiveX browser plug-in that installs onto the local machine via the Web browser in the exact same manner that you install Adobe Flash to run Flash-based animations on Web pages. The Silverlight plug-in supports the entire wow factor that you'd expect from an RIA, such as vector-based graphics and animations and full video integration, including Digital Rights Management

(DRM) secured audio/video and hi-definition video, as well as the tools for building rich line-of-business applications. You can boil down the coolness of Silverlight to the following points:

- Silverlight is a cross-platform, cross-browser platform for delivering rich interactive applications.
- Silverlight 3 applications can be built using Expression Blend, Visual Studio, or Eclipse on Windows, and with Eclipse on Apple Macintosh computers.
- Silverlight supports playback of native Windows Media VC-1/WMA (with Digital Rights Management) as well as MPEG-4-based H-264 and AAC audio on PCs and Macs with no dependency on Windows Media Player.
- Silverlight supports playback of 720p+ full-screen HD Video.
- Using XAML, HTML, JavaScript, C#, VB (or your managed language of choice, including dynamic languages like Ruby and Python), Silverlight delivers rich multimedia, vector graphics, animations, and interactivity beyond what AJAX can deliver.
- With the Base Class Libraries (BCLs), you have access to common classes for generics, collections, and threading that you are accustomed to using in Windows client development.
- There are more than 60 controls in the toolbox, with many more from third-party vendors.
- You can deliver out-of-browser experiences that can run any Silverlight 3 application as a desktop application with complete network detection for graceful exception handling.
- The installation package is less than 6 MB on Windows and less than 12 MB on Macintosh.
- Almost all of 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 stand-alone environment; there is no dependency version of the .NET Framework on the client or the server to run Silverlight 3 applications. When developing applications for Silverlight 3, you are using tools (like Visual Studio 2008 or Expression Blend) that require or are based on a version of the Common Language Runtime (CLR), but the compiled Intermediate Language (IL) of your Silverlight applications that is parsed by the Silverlight 3 player is not using a specific client version of the .NET Framework. The BCL for Silverlight is entirely self-contained within the player itself. The XAML and BCL used by the Silverlight 3 player are both subsets of their counterparts that are used when building full desktop-based WPF applications.

You might ask why Microsoft is pushing out another Web-based, client-side technology when there is already ASP.NET, ASP.NET AJAX Extensions, and, with CLR 3.5 and Visual Studio 2008, specific project types that target the ASP.NET AJAX Framework. 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 Windows Presentation Foundation (WPF) on the Windows client side. WPF provides a unified approach to media, documents, and graphics in a single run time. The problem with WPF is that it is a 30-MB run time that runs only

on the Windows OS. Microsoft needed to give the same type of experience that WPF offers, only in a cross-platform, cross-browser delivery mechanism. So what Microsoft did was take the concept of a plug-in model like Adobe Flash, mix it with the .NET Framework and the WPF declarative language in XAML, and they came up with a way to develop highly rich, immersive, Web 2.0 applications.

The big picture of Silverlight from an architecture perspective is shown in Figure 1-1. Each area is covered in more detail as you read along in the book.

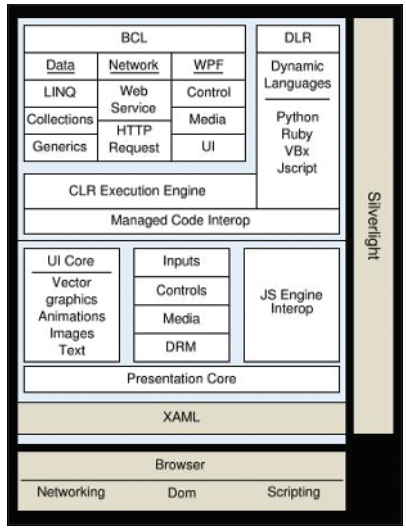

Figure 1-1

As mentioned earlier, Silverlight can conceivably be fully supported across multiple browsers and operating systems. The current status for browser and OS support is identified in the following table:

Browser	Internet Explorer 6, 7, and 8 on Windows
	Safari, Firefox on Windows and Mac
	Firefox 2 and Firefox 3 on Linux
Operating Systems	Windows 2000, Windows 2003, Windows XP, Windows Vista, Windows 7
	Mac OS 10.4/10.5 Intel
	SUSE Linux Enterprise Desktop, openSUSE 11.0, openSUSE 11.1, Ubuntu 8.04, Fedora Core 9 via the Mono Project's Moonlight implementation of the Silverlight 2 player (a Silverlight 3 implementation of the Moonlight player is not available as of this writing).

Silverlight Versions Explained

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

- **Silverlight 1.0** — This is the first version of Silverlight and supports the JavaScript programming model. 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. There is no managed language support in Silverlight 1.0, which means no BCL for Silverlight 1.0.
- **Silverlight 2** — Released in late 2008, Silverlight 2 brought the ability to create RIA applications with the familiar code-behind programming model used in Windows Forms, ASP.NET, and WPF development. Starting with Silverlight 2, you can use any CLR language to code Silverlight applications, and you have the power of the .NET Framework to interact with Silverlight objects. The ability to use the base class libraries and your .NET language of choice to build Silverlight applications truly revolutionized the way developers and designers looked at this new RIA platform.
- **Silverlight 3** — This is the third version of Silverlight and the topic of this book, following the release of Silverlight 2 in late 2008. Silverlight 3 supports the familiar code-behind programming model used in Windows Forms, ASP.NET, and WPF development. You can use any CLR language to code Silverlight applications, and you have the power of the .NET Framework to interact with Silverlight objects. Silverlight 3 includes extensive enhancements to Silverlight 2 for building line-of-business applications as well as richer support for graphics and media.

Silverlight uses an auto-update model for the player. When a new version of Silverlight is released, the player running in the browser is updated to the latest version automatically. There is also the commitment of backward compatibility, so your applications will not break when the player moves from version 1.0 to 2, or 2 to 3, and so on.

Application Development Scenarios

When building Silverlight applications, there are two basic scenarios that occur:

- Your entire application is written in Silverlight, the player takes up 100 percent of the height and width of the browser, and all UI interaction is done through Silverlight.
- You implement an "Islands of Richness" scenario, in which your application is an ASP.NET application (or any other type of HTML-rendered application), and you build islands of your UI with Silverlight. Thus, you are adding richness to your web applications but not building the entire interaction using Silverlight.

I see the "Islands of Richness" scenario as being a very common way for Silverlight to find its way into most applications. Silverlight is a simple way to add audio, video, or interactive data visualization to a Web page without having to rebuild or re-design existing applications on a new platform. The area surrounded with the red box in Figure 1-2 is an example of an "Islands of Richness" scenario in which Silverlight has been added to an existing web application. In this case, the image strip is a Silverlight control that will play a video in-page when an item is clicked on. Silverlight 3 enhances the "Islands of

Richness" scenarios by allowing multiple Silverlight plug-ins and an easy way to communicate with each other in the browser. This also works across browsers; for example, a Silverlight 3 application running in a Firefox browser can talk to a Silverlight 3 application running in Internet Explorer 8 on the same machine.

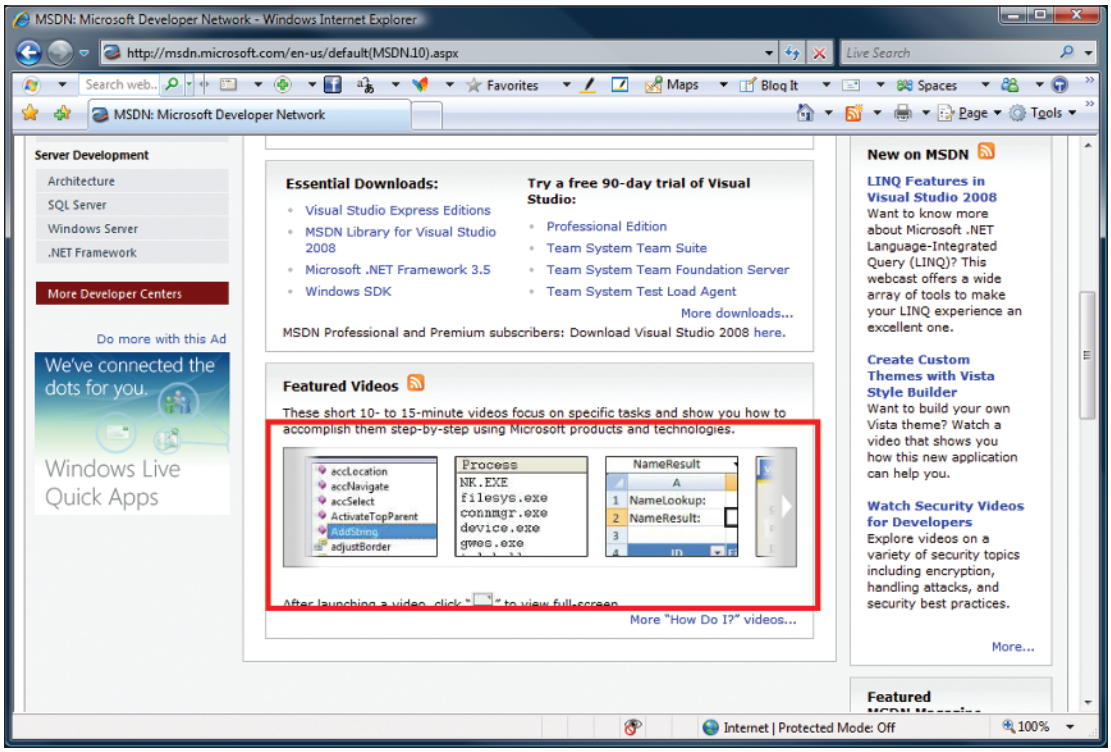

Figure 1-2

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 installation — 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 1-3 illustrates a page with the standard install images.

Once you click on the Get Microsoft Silverlight Installation image, one of two scenarios takes place. You are taken to the Silverlight Installation page on the Microsoft site, as Figure 1-4 demonstrates.

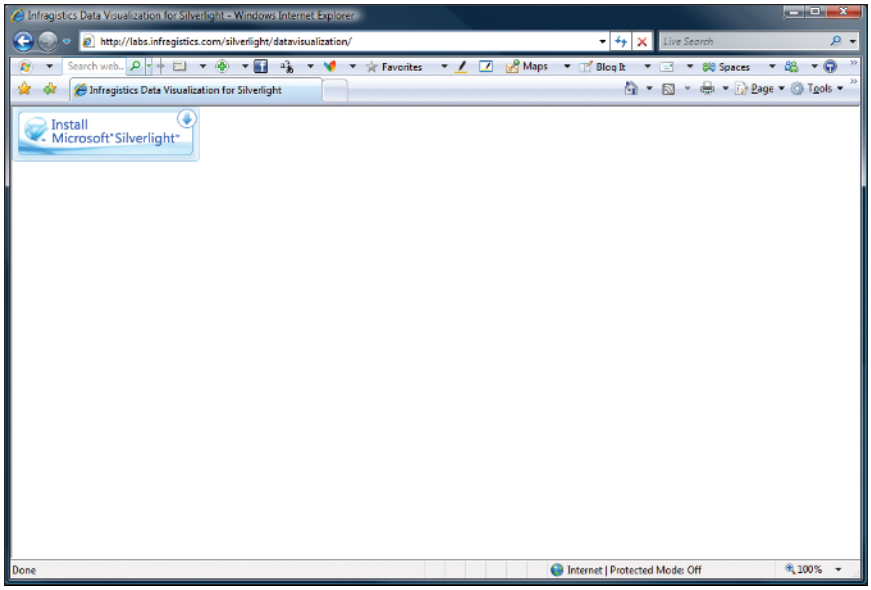

Figure 1-3

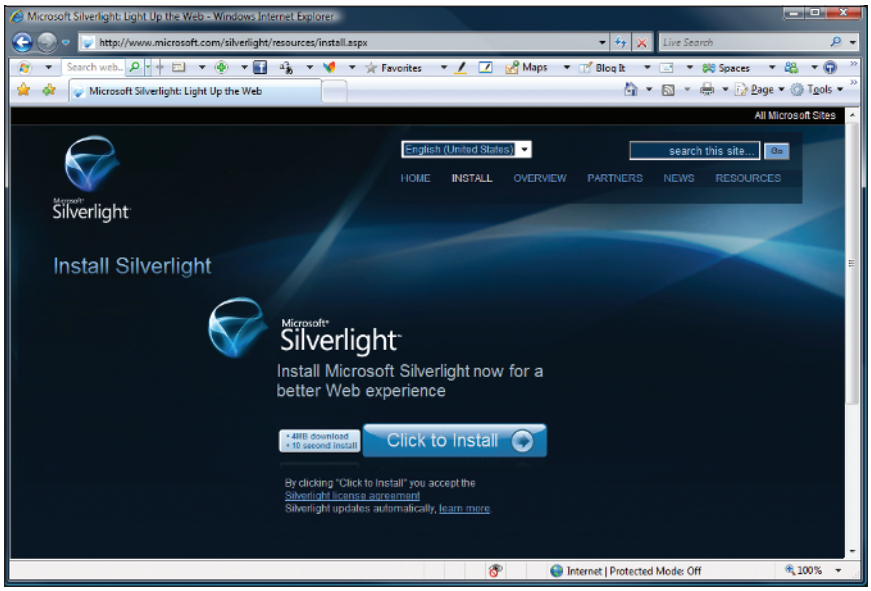

Figure 1-4

Or you are prompted to install Silverlight *in place* with a download prompt, as shown in Figure 1-5.

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. Once you refresh the browser, the Silverlight content will be rendered correctly in the browser, as Figure 1-6 shows.

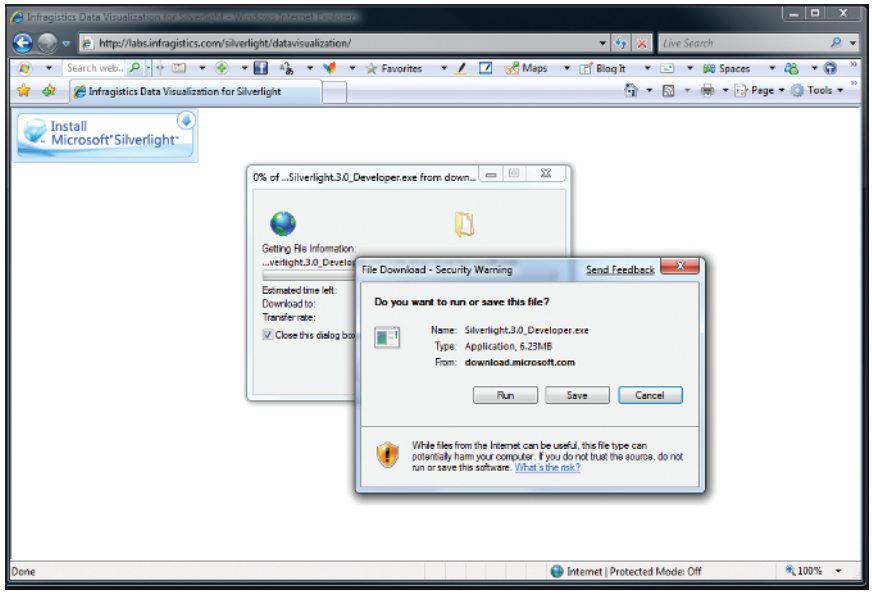

Figure 1-5

Figure 1-6

Getting the Silverlight SDK

To build Silverlight applications, you need more than the Silverlight player. If you have not arrived at a page where you are prompted to install the Silverlight run time, 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), which will give you the files you need to start building Silverlight applications. To get the SDK, go to `www.silverlight.net/getstarted/default.aspx`, as shown in Figure 1-7.

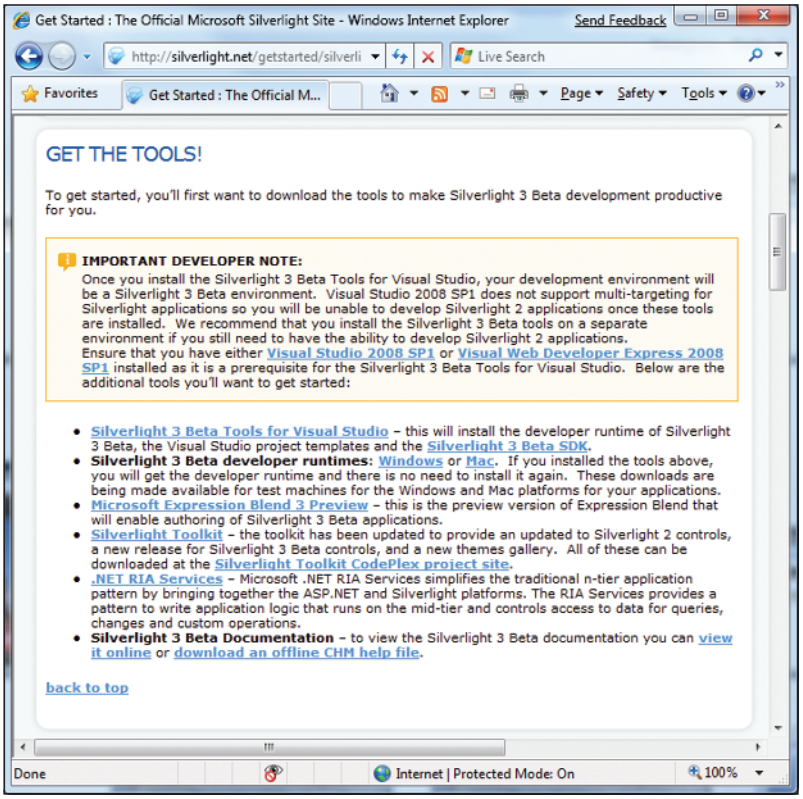

Figure 1-7

On the Get Started page, you can download all of the tools that you need to create Silverlight 3 applications:

- Silverlight run times for Mac and Windows operating systems
- Silverlight tools for Visual Studio 2008
- The latest version of Microsoft Expression Blend
- A trial version of Visual Studio 2008

More importantly, this page has links to dozens of videos, tutorials, and samples that will help you learn Silverlight.

Building Silverlight Applications

Now that you have the Silverlight player installed and you know how to get the tools for Visual Studio that will give you the project templates, you can start building Silverlight applications. There are several ways to create Silverlight applications:

- Visual Studio 2008 Silverlight project templates, which include Silverlight Application, Silverlight Navigation Application, and Silverlight Class Library, as well as Silverlight Business Application if you have .NET RIA Services installed
- Expression Blend 3
- Eclipse using the Eclipse plug-in. There is an Eclipse plug-in for both Windows- and Apple Macintosh–based operating systems.

In the next chapter, you will get a better understanding of the details for how to build applications using Visual Studio 2008 and Expression Blend.

Silverlight 3 Tour

Silverlight 3 continues the improvements that Silverlight 2 delivered over Silverlight 1.0. In the next section, we'll look at some of the more important features of Silverlight 3, including:

- XAML
- .NET Framework support
- Graphics and animations
- Page layout and design
- User interface controls
- Audio and video
- Local data storage
- Out-of-browser capability
- Navigation Framework
- Ink support
- Network access
- Data binding
- Deep Zoom technology

Throughout the book, you will learn about each of the items listed in much more detail. The following sections are designed to set the stage for what's to come as you explore the full capability of Silverlight 3.

XAML

If you are not familiar with WPF, you are probably not familiar with XAML. Since the dawn of Visual Studio, there has always been 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 had never been realized, mostly because developers and designers were always using different tools and different languages. With the introduction of XAML, however, 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 mockup 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 2008 allow you to separate the code logic from the layout and control definitions, using XAML gives you the opportunity to have this separation of the design from the code.

XAML elements are objects that map to classes in the Silverlight run time. So when you declare a XAML `TextBlock` like this:

```
<TextBlock />
```

you are actually creating a new instance of the `TextBlock` class like this:

```
TextBlock t = new TextBlock();
```

The following code demonstrates a XAML snippet from a Silverlight application that shows *Hello World* in a `TextBlock`:

```
<Canvas>
    <TextBlock>Hello World</TextBlock>
</Canvas>
```

The next code listing shows how the XAML can get more complex, demonstrating adding animations to the `TextBlock` element. In this example, a `RotateTransform` is being applied to a `TextBlock` control via a `DoubleAnimation` in a `StoryBoard` object. This action is triggered when the `UserControl` loads, through the `RoutedEvent Canvas.Loaded`. If you run the XAML, you will see that the text *Hello World* rotates in a 360-degree circle.

> *In Chapter 9, you will learn how animations work in Silverlight and how they are used to bring your application to life in the Silverlight player.*

```
<StackPanel Margin="4"
            HorizontalAlignment="Center"
            Orientation="Horizontal">
    <TextBlock Width="200" Height="150"
               FontSize="24">Hello World

        <TextBlock.Triggers>
            <EventTrigger RoutedEvent="Canvas.Loaded">
                <EventTrigger.Actions>
                    <BeginStoryboard>
                        <Storyboard BeginTime="0"
                                    RepeatBehavior="Forever">
                            <DoubleAnimation
```

```
                                    Storyboard.TargetName="rotate"
                                    Storyboard.TargetProperty="Angle"
                                    To="360"
                                    Duration="0:0:10"/>
                            </Storyboard>
                        </BeginStoryboard>
                    </EventTrigger.Actions>
                </EventTrigger>
            </TextBlock.Triggers>

            <TextBlock.RenderTransform>
                <RotateTransform x:Name="rotate"
                                 Angle="0"
                                 CenterX="300"
                                 CenterY="200"/>
            </TextBlock.RenderTransform>
        </TextBlock>
    </StackPanel>
```

In Chapter 2, you will get 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 and applications that we have created. Tools like Microsoft Expression Blend and Visual Studio 2008 are all Rapid Application Development (RAD) tools that you can use to create your Silverlight applications. As you will learn when you start building Silverlight applications using Visual Studio 2008, Microsoft Expression Blend is 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. Visual Studio 2010 will have rich support for RAD development of Silverlight 3 applications. At the time of this writing, Visual Studio 2010 is still in beta. Besides using Expression Blend or Visual Studio 2008, you can look to other XAML tools like XAMLPad or Kaxaml to help you learn XAML. In Chapter 4, you will learn more of the specifics on building Silverlight 3 applications using Visual Studio.

.NET Framework Support

Two key aspects of Silverlight 3, and probably the most exciting aspects of this technology, are its support for the CLR and BCL of the .NET Framework. Although they are not the exact set of class libraries you are familiar with using on the desktop and the CLR might handle memory management and optimizations slightly differently than on the desktop or server, the fundamental capabilities of the .NET Framework do exist for you to use to build rich Silverlight applications.

Execution of content targeting the Silverlight player is handled by the CoreCLR. The CoreCLR is a smaller, refactored version of the CLR used in full .NET desktop applications. Although the Microsoft Intermediate Language (MSIL) is exactly the same between the CLRs, the CoreCLR is stripped of the unnecessary scenarios that are not needed for Silverlight 3 development. The CLR is still responsible for managing memory in Silverlight applications, as well as enforcing the common type system (CTS). Some examples of the differences in the CoreCLR versus the full CRL are:

- The JIT Compiler in the CoreCLR is enhanced for fast startup time, while the full CLR is enhanced for more complex optimizations.
- In ASP.NET applications, the garbage collection mode is tuned for multiple worker threads, while the CoreCLR is tuned for interactive applications.

Both the CoreCLR and CLR can run in the same process; therefore, for example, you can have an embedded Silverlight player running in an Office Business application that also includes a full .NET 3.5 plug-in. The isolation of the CoreCLR is why you can run Silverlight applications on machines that do not have any versions of the .NET Framework installed; this is further highlighted by the fact that Silverlight can run on Macintosh operating systems.

The namespaces that contain all of the classes that you interact with in your Code window are the Base Class Libraries, as you have learned. The Silverlight BCL does not contain namespaces and classes that do not make sense for client development, such as code-access security, ASP.NET Web Server–specific classes, and many others.

Chapter 3 delves into the specifics of the CoreCLR.

Graphics and Animations

A big part of why Silverlight is an exciting technology is that it provides a rich, vector-based drawing system as well as support for complex animations. Some of the new additions in Silverlight 3 include:

- Perspective 3D graphics
- Pixel-Shader effects, including `Blur` and `DropShadow`
- Bitmap Caching to increase the rendering performance
- Animation effects like `Spring` and `Bounce`
- Local font usage for rendering text

For vector-based drawing, Silverlight supports `Geometry` and `Shape` objects that include support for rendering shapes, such as ellipse, line, path, polygon, polyline, and rectangle. These classes give you the ability to render any type of visual display. For example, the following XAML displays an image in its normal, square shape:

```
<Canvas>
     <Image
          Source="Images/elk.jpg"
        Width="200" Height="150">
     </Image>
 </Canvas>
```

Using the `EllipseGeomtry` class, you can clip the image into whatever shape you desire. This XAML clips the image into an oval:

```
<Canvas>
    <Image
       Source="Images/elk.jpg"
       Width="200" Height="150">
        <Image.Clip>
            <EllipseGeometry
                RadiusX="100"
                RadiusY="75"
                Center="100,75"/>
```

```
            </Image.Clip>
        </Image>
    </Canvas>
```

with the results displayed in Figure 1-8.

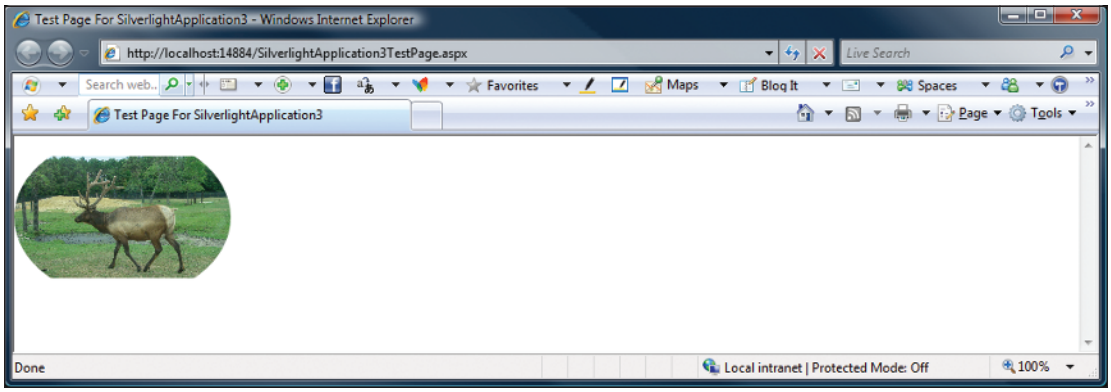

Figure 1-8

Once you render your geometries or shapes into something meaningful, you can use `Brushes`, `VideoBrushes`, or `Transforms` to further give life to your UI rendering. The following XAML takes a basic `TextBlock` and adds a `LinearGradientBrush` for some nice special effects:

```
<TextBlock
  Canvas.Top="100"
  FontFamily="Verdana"
  FontSize="32"
  FontWeight="Bold">
    Linear Gradient Brush
    <TextBlock.RenderTransform>
        <ScaleTransform ScaleY="4.0" />
    </TextBlock.RenderTransform>
    <TextBlock.Foreground>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
            <GradientStop Color="Red" Offset="0.0" />
            <GradientStop Color="Blue" Offset="0.2" />
            <GradientStop Color="Green" Offset="0.4" />
            <GradientStop Color="Olive" Offset="0.6" />
            <GradientStop Color="DodgerBlue" Offset="0.8" />
            <GradientStop Color="OrangeRed" Offset="1.0" />
        </LinearGradientBrush>
    </TextBlock.Foreground>
</TextBlock>
```

You can also use an `ImageBrush` to paint an image on your `TextBlock`, as the following code demonstrates:

```
<StackPanel>
    <!--TextBlock without an ImageBrush -->
```

```
        <TextBlock
            FontSize="72"
            FontFamily="Verdana"
            FontStyle="Italic"
            FontWeight="Bold">
                Rhino Image
            </TextBlock>

        <!--TextBlock with an ImageBrush -->
        <TextBlock
            FontSize="72"
            FontFamily="Verdana"
            FontStyle="Italic"
            FontWeight="Bold">
                Rhino Image
            <!-- Add an Image as the foreground -->
            <TextBlock.Foreground>
            <ImageBrush ImageSource="Images/rhino.jpg"
                    Stretch="Fill"/>
            </TextBlock.Foreground>
        </TextBlock>
    </StackPanel>
```

The resulting content looks like Figure 1-9.

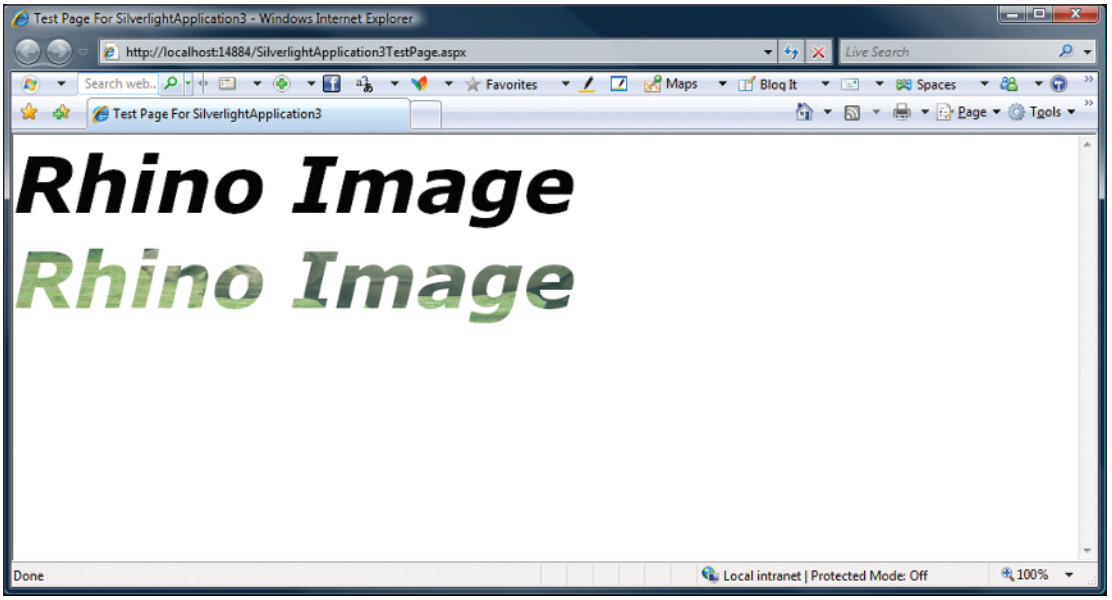

Figure 1-9

Later in this section, you will see a `VideoBrush` applied to text. In Chapter 9, we'll cover graphics and animations in full detail.

Page Layout and Design

Silverlight 3 includes several options for doing rich, resolution independent layout using a `Canvas`, `DockPanel`, `Grid`, `StackPanel`, and `WrapPanel` element. These five major layout panels can be described as:

- `Canvas` — An absolute positioning panel that gives you an area within which you can position child elements by coordinates relative to the `Canvas` area. A `Canvas` can parent any number of child `Canvas` objects.
- `DockPanel` — Is used to arrange a set of objects around the edges of a panel. You specify where a child element is located in the `DockPanel` with the `Dock` property.
- `Grid` — Similar to an HTML table, a `Grid` is a set of columns and rows that can contain child elements.
- `StackPanel` — A panel that automatically arranges its child elements into horizontal or vertical rows
- `WrapPanel` — Allows the arrangement of elements in a vertical or horizontal list and have elements automatically wrap to the next row or column when the height or width limit of the panel is reached.

One you decide how you are going to lay out your page using one of the layout types, you can use other means of positioning individual elements as well. For example, you can change margins, set the `ZOrder` or `Border` of an object, or perform `RotateTranforms` to change the position of an object. Chapter 7 covers all layout options in greater detail. Here we'll look at the `Canvas` object and how it behaves.

The `Canvas` essentially becomes the container for other child elements, and all objects are positioned using their X- and Y-coordinates relative to their location in the parent canvas. This is done with the `Canvas.Top` and `Canvas.Left` attached properties, which provide the resolution-independent pixel value of a control's X- and Y-coordinates. The following code shows a `Canvas` object with several child elements absolutely positioned within the `Canvas`.

```
<Canvas>
    <Rectangle
    Canvas.Top ="30"
    Canvas.Left="30"
    Fill="Blue"
    Height="100" Width="100"/>

    <Rectangle
    Canvas.Top ="75"
    Canvas.Left="130"
    Fill="Red"
    Height="100" Width="100"/>

    <Ellipse
    Canvas.Top ="100"
    Canvas.Left="30"
    Fill="Green"
    Height="100" Width="100"/>
</Canvas>
```

This XAML is explained in more detail in Figure 1-10, which shows the location of the objects in the canvas.

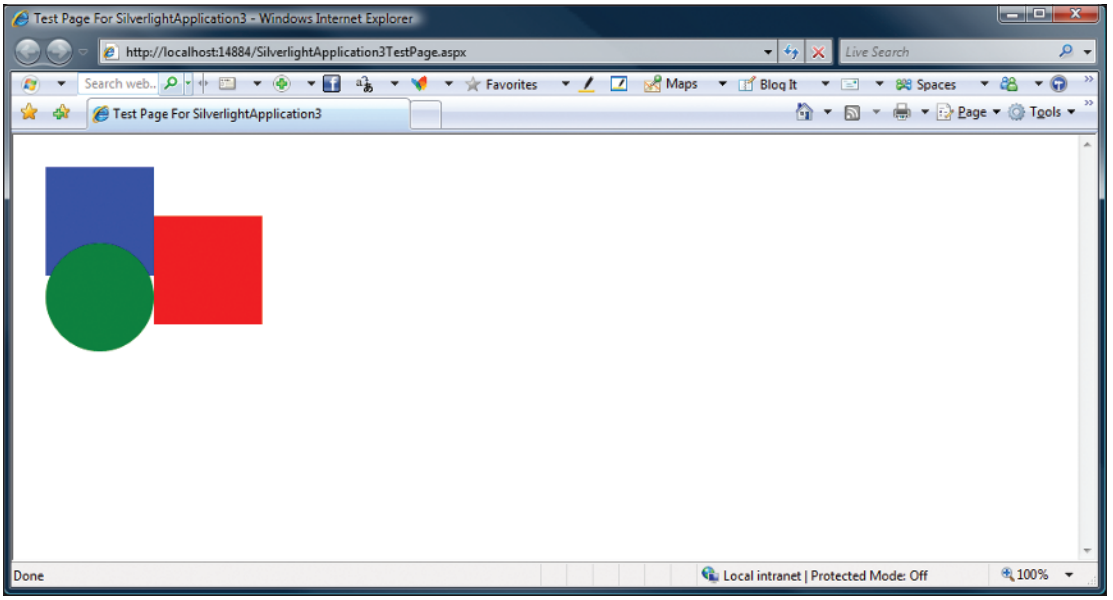

Figure 1-10

In the following example from the SDK, you can see how a `DockPanel` can be configured to return the results shown in Figure 1-11.

```
<StackPanel x:Name="LayoutRoot" Background="White">
    <TextBlock Margin="5" Text="Dock Panel" />
    <Border BorderBrush="Red" BorderThickness="2" >
        <controls:DockPanel LastChildFill="true"
                            Height="265">
            <Button Content="Dock: Left"
                    controls:DockPanel.Dock ="Left" />
            <Button Content="Dock: Right"
                    controls:DockPanel.Dock ="Right" />
            <Button Content="Dock: Top"
                    controls:DockPanel.Dock ="Top" />
            <Button Content="Dock: Bottom"
                    controls:DockPanel.Dock ="Bottom" />
            <Button Content="Last Child" />
        </controls:DockPanel>
    </Border>
</StackPanel>
```

In Figure 1-11, notice the position of the elements based on the `TextBlock` and `Border` controls that wrap the `DockPanel` in the XAML.

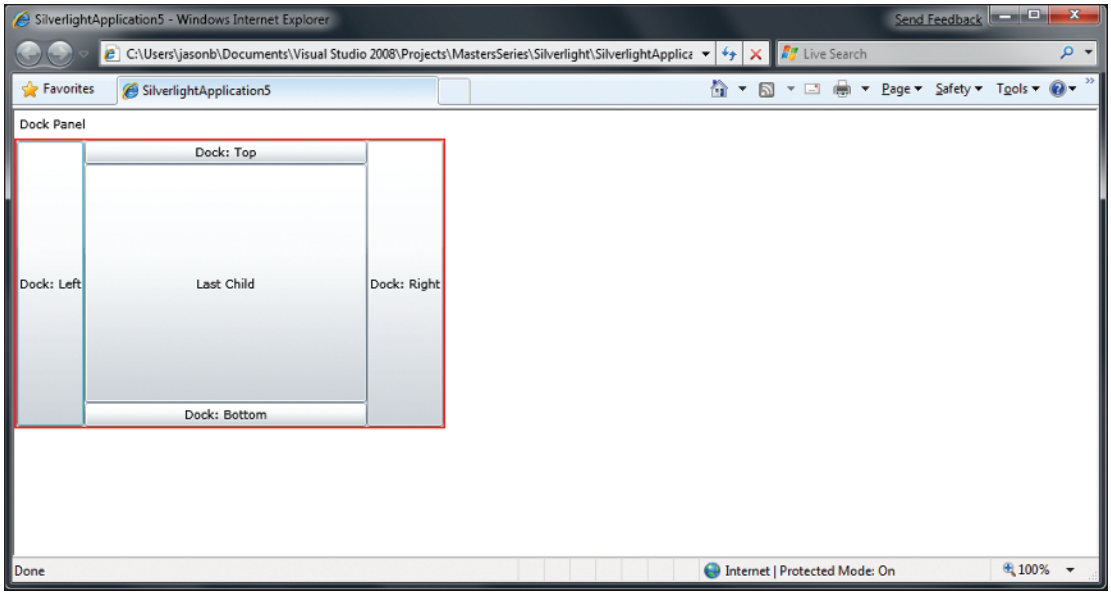

Figure 1-11

User Interface Controls

Silverlight 3 adds an even greater number of controls to the Toolbox for creating user interfaces. The Toolbox in Visual Studio 2008 is now filled with controls that can be dragged onto forms to build the user interface. The following controls are included for use by the core Silverlight 3 player:

AutoCompleteBox	DatePicker	Label	StackPanel
Border	DockPanel	ListBox	TabControl
Button	Expander	MediaElement	TextBlock
Calendar	Grid	MultiScaleImage	TextBox
Canvas	GridSplitter	Popup	TreeView
CheckBox	HeaderedContentControl	RadioButton	ViewBox
ContentControl	HeaderedItemsControl	RepeatButton	WrapPanel
DataForm	HyperlinkButton	ScrollBar	
DataGrid	Image	ScrollViewer	
DataPager	InkPresenter	Slider	

In addition to the aforementioned controls, the Silverlight Toolkit, which is a separate download from CodePlex, contains several very useful additions to the core list.

When working with any of the controls, remember they are just like any other control model: The XAML controls in Silverlight can be instantiated in code, and properties can be retrieved or set on them. Over the next several chapters, you will learn about the controls in more detail, as well as how they can be used with Visual Studio 2008 or Expression Blend.

Using Media in Silverlight

One could argue that the entire reason for Silverlight was to provide rich, multimedia experiences on Web pages, which essentially means audio and video on Web pages. If you take a look at the top 100 trafficked web sites on the Internet, almost all of them have video playing on the home page or use video prevalently throughout. Silverlight 3 continues to add first-class media capability to the player.

Adding Video to Web Pages

To add video or audio to a Web page, you set the `Source` property on the `MediaElement` object. The following code demonstrates playing the video file car.wmv automatically when the canvas is loaded:

```
<UserControl x:Class="SilverlightApplication3.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="600" Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <MediaElement Source="Images/video1.wmv" />
    </Grid>
</UserControl>
```

The `Source` property is the URI of a valid video or audio file. In the preceding code example, the source file is located in the deployment directory of your Silverlight application. Your media files can be located in various locations, including the web-site folder structure you are running the page from, or from a remote site. In either case, in order to maintain cross-platform support, you must use "/" in place of "\" in your URIs. For example:

```
<MediaElement Source="..\..\car.wmv"></MediaElement>
```

should read:

```
<MediaElement Source="../../car.wmv"></MediaElement>
```

If the `Source` property is pointing to a file on a Windows Media Server using the MMS protocol, the player will automatically attempt to stream the video down to the client. The default behavior is a progressive download, which means that the audio or video will begin playing immediately and background-load as you are playing the media. The drawback to progressive downloads is that even if you pause the video, it still downloads the media file, even if you never intended to continue playing it. With streaming media, the only data that is downloaded is the data that you actually play, which is a more efficient use of network resources.

> *Microsoft Live Services offers a free media streaming service for Silverlight applications, named* Silverlight Streaming Services. *Using Silverlight Streaming Services, anyone can upload up to 4 GB of Silverlight content to stream to their pages. To get a free account for this service, visit* `https://silverlight.live.com`.

Supported Audio and Video Formats

The `MediaElement` supports the Advanced Stream Redirector (ASX) playlist file format, as well as the audio and video formats listed in the following table:

Video Formats	Audio formats
WMV1: Windows Media Video 7	WMA 7: Windows Media Audio 7
WMV2: Windows Media Video 8	WMA 8: Windows Media Audio 8
WMV3: Windows Media Video 9	WMA 9: Windows Media Audio 9
WMVA: Windows Media Video Advanced Profile, non-VC-1	WMA 10: Windows Media Audio 10
WMVC1: Windows Media Video Advanced Profile, VC-1	AAC: Advanced Audio Coding — Can only be used for progressive download, smooth streaming, and adaptive streaming. AAC is the LC variety and supports sampling frequencies up to 48 kHz.
H.264 — Can only be used for progressive download, smooth streaming, and adaptive streaming. Supports Base, Main, and High Profiles.	MP3: ISO/MPEG Layer-3 with the following features: ❑ Input — ISO/MPEG Layer-3 data stream ❑ Channel configurations — Mono, stereo ❑ Sampling frequencies — 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, and 48 kHz ❑ Bitrates — 8–320 Kbps, variable bitrate ❑ Limitations — "Free format mode" (ISO/IEC 11172-3, subclause 2.4.2.3) is not supported.

Local Data Storage

Using the *isolated storage* concept in the full .NET Framework, you can use a client-side cache location to store data. This means that you can take commonly needed data, and, instead of always having to go back to the server to retrieve it, you can store it locally and access it locally. Examples might be a list of states or countries, or buddy lists for instant messenger clients. This data is commonly needed for fast access but does not change often enough to warrant constant round-trips back to the server to retrieve it.

By default, Silverlight gives you 1 MB of local storage. This can be increased by prompting the user to allow for more local storage or can be accessed via the Silverlight Configuration screen. As its name implies, this is isolated storage, so you cannot access the end-user's filesystem or do anything that would break the partial trust sandbox that Silverlight runs in. Storage is granted per application, so, for example, you might have `www.someapp.com`, which is using 10 MB of storage, and another application running on the same client computer from a different domain that has its own 20 MB of isolated storage. The storage areas are independent of each other; there is no limit to the number of applications that can have isolated storage on a client machine.

Out-of-Browser Experiences

With the new Out-of-Browser capability of any Silverlight application, an end-user can save your application to the desktop on their Windows or Apple Macintosh computer. There is no need to install any special assemblies or controls to make this work — it is part of the native Silverlight 3 experience. With the new network detection APIs in Silverlight 3, an out-of-browser application can intelligently determine if it is connected to the network and react accordingly. In Chapter 11, you'll learn how easy it is to actually create this out-of-browser experience.

Navigation Framework

Silverlight 3 adds two new controls that enable complete browser-journal back/forward integration with your application. Using the new `Frame` and `Page` controls, you can partition your views into separate XAML files (instead of separate `UserControl` objects as you did in Silverlight 2) and navigate to each view as simply as you would previously a Web page. The Navigation Framework also allows you to implement deep linking support in your Silverlight application, which builds on the SEO (Search Engine Optimization) enhancements added in Silverlight 3.

The following XAML shows the navigation control added to a `UserControl`:

```
<navigation:Frame x:Name="Frame"
                  Source="/Views/HomePage.xaml"
                  HorizontalContentAlignment="Stretch"
                  VerticalContentAlignment="Stretch"
                  Padding="15,10,15,10"
                  Background="White"/>
```

And the following code demonstrates the `Navigate` method of the `Frame` class, which is how you move from `Page` to `Page`.

```
private void NavButton_Click(object sender, RoutedEventArgs e)
{
    Button navigationButton = sender as Button;
    String goToPage = navigationButton.Tag.ToString();
    this.Frame.Navigate(new Uri(goToPage, UriKind.Relative));
}
```

As well as `Navigate`, the `Frame` class includes other useful methods such as `Navigated`, `NavigationFailed`, and `NavigationStopped` that give you complete control over the navigation life cycle of your `Page` object. Chapter 11 talks more about the `Navigation` and `Frame` classes.

Annotation and Ink

Like WPF, Silverlight has full support for ink input in the player. Using the `InkPresenter` object, you can give users an input area where they can use the mouse or an input device to handwrite. Using the application interface for the `InkPresenter` object, the application developer collects the `Stroke` objects that are written and persists them to a location on the server for later use. An example of where ink might be cool on a Web page is a simple blog, where text and ink can combine to create a great visual

output for whatever the blog is about. The XAML in the following code shows how to create an `InkPresenter` object:

```
<InkPresenter x:Name="inkInput" Cursor="Stylus"
    MouseLeftButtonDown="inkInput_MouseLeftButtonDown"
    MouseMove="inkInput_MouseMove"
    MouseLeftButtonUp="inkInput_MouseLeftButtonUp"/>
```

Notice that events are wired up for the various mouse behaviors. Each action of the mouse — the `Move`, `LeftButtonUp`, and `LeftButtonDown` — has a method in the code-behind that acts on the strokes of the input device. The following code gives an example of how you would collect the strokes from the `InkPresenter`:

```
private Stroke MyStroke = null;

private void inkInput_MouseLeftButtonDown
    (object sender, MouseButtonEventArgs e)
{
    inkInput.CaptureMouse();
    StylusPointCollection
        MyStylusPointCollection = new StylusPointCollection();
    MyStylusPointCollection.Add
        (e.StylusDevice.GetStylusPoints(inkInput));
    MyStroke = new Stroke(MyStylusPointCollection);
    inkInput.Strokes.Add(MyStroke);
}

private void inkInput_MouseMove
    (object sender, MouseEventArgs e)
{
    if (MyStroke != null)
    {
        MyStroke.StylusPoints.Add
            (e.StylusDevice.GetStylusPoints(inkInput));
        txtBlock.Text =
            "" + e.StylusDevice.GetStylusPoints(inkInput)[0].X;
        txtBlock.Text =
            "" + e.StylusDevice.GetStylusPoints(inkInput)[0].Y;
    }

}

private void inkInput_MouseLeftButtonUp
    (object sender, MouseButtonEventArgs e)
{
    MyStroke = null;
}
```

Once you have the ink data collected, you can store it locally on the client machine, put it into a database, or even save the ink as an image.

Accessing the Network

To access network resources in Silverlight, you have the classes in the `System.Net` namespaces and the `System.Net.Sockets` namespace. The namespace you choose would depend on the type of network access you are trying to achieve. For basic HTTP or HTTPS access to URI-based resources, you can use the `WebClient` class in the `System.Net` namespace. Some examples of this type of network access are:

- Retrieving XML, JSON, RSS, or Atom data formats from a URI then parsing it on the client
- Downloading resources such as media or data to the browser cache

Using `WebClient`, you can perform the types of asynchronous operations that are common in browser-based applications. The following code demonstrates a simple method that grabs an image file from a network resource and downloads it to the browser cache:

```
void DownloadFile(string imgPart)
{
    WebClient wc = new WebClient();
    wc.OpenReadCompleted +=
       new OpenReadCompletedEventHandler
       (wc_OpenReadCompleted);
    wc.OpenReadAsync(new Uri("imgs.zip",
       UriKind.Relative), imgPart);
}
```

If you need more flexibility in how you access HTTP or HTTPS resources, you can use the `HttpWebRequest` and `HttpWebResponse` classes.

If you need more direct and constant access to network resources or if you are working in a situation in which multiple clients are "listening" for the same server data, you would choose to use the classes in the `System.Net.Sockets` namespace. Although both Sockets and WebClient allow asynchronous communication using the TCP protocol, Sockets gives you the ability to write "push-style" applications, where the server can communicate with the client in a more client-server manner. Imagine the unnecessary overhead when using basic AJAX timers (polling) to look for updated data on the server. If you were using sockets instead of this type of timer-based polling, you would reduce the amount of wasted bandwidth and would achieve tighter control of the data passing between the client and the server.

No matter how you choose to work with the network, both the `System.Net` and `System.Net.Sockets` namespaces support the ability to access network resources from other URIs than the originating domain. By default, a Silverlight 3 application can always access resources from its originating domain. Using a policy file, an application can access resources from different domains from the one containing its original URL. This cross-domain access is controlled by policy files that dictate the type of network domain access an application has. For `WebClient` requests, the same format used by Adobe Flash is supported. The next code is an example of a crossdomain.xml file:

```
<?xml version="1.0"?>
<! DOCTYPE cross-domain-policy
   SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
    <allow-access-from domain="*" />
</cross-domain-policy>
```

In Chapters 11 and 13, you will be fully exposed to various ways of accessing network resources.

Data Binding

Similarly to the data-binding features in WPF, Silverlight supports data-bound controls, XAML markup extensions, and support for data context binding. Most of the time, your bindings will be set up in XAML, which is where the markup extensions come into play. In the following XAML, the `Text` property of the `TextBlock` element is using the `Binding` markup extension to bind the `Title` field from the data source:

```
<TextBlock x:Name="Title"
 Text="{Binding Title, Mode=OneWay}" />
```

The field `Title` from the original data source is retrieved from the data content of the control's parent element; in this case, the `TextBlock` could be contained in a `Canvas` or `Grid` object. Once you set the `DataContext` property for the parent element, the data contained in that object is available for binding to anything it contains. A more complete example of this data binding looks like this:

```
<Canvas x:Name="rootCanvas" Background="White" >
    <TextBlock x:Name="Title"
    Text="{Binding Title, Mode=OneWay }" />

    <TextBlock x:Name="Name"
    Text="{Binding Title, Mode=OneWay }" />
</Canvas>
```

You would then set the context in the code like this:

```
LayoutRoot.DataContext = dataList;
```

where the `dataList` object is an object that contains the data you are binding to the controls. In the case of simple `TextBlock` objects, you would have to handle the navigation between elements yourself. If you want a richer, tabular data display, you could use the `Grid` that is included with Silverlight. The XAML for the `DataGrid` control looks like this:

```
<data:DataGrid x:Name="dataGrid1"
    Height="120" Width="450"
    AutoGenerateColumns="True" />
```

The same `dataList` object can be bound to the grid in code like this:

```
dataGrid1.ItemsSource = dataList;
```

All of the binding could be accomplished in code, but using the combination of XAML and code gives you greater flexibility when building Silverlight applications. An interesting area of data binding in Silverlight is where the data actually comes from. Since the Silverlight player is a complete client-side solution, you are not creating connections to SQL Server or other data sources, then dumping that data into a data set in your code-behind. You are going to be using technologies like WCF to access services on the Internet and then putting the data you retrieve into objects that are bound to controls in Silverlight. In Chapter 14, you

will learn about the various types of data access, how to interact with different data formats, and how the data-binding mechanism works in Silverlight.

Deep Zoom Graphics

Deep Zoom is a multi-scale image-rendering technology that partitions a very large image, or set of images, into smaller *tiles* that are rendered on demand to the Silverlight player. When an image is first loaded, it is in the lowest-resolution tiles. As the user zooms into the image using the mouse wheel or keyboard, higher-resolution images are loaded based on the area that is being zoomed into. The wow factor of Deep Zoom was shown off at Mix '08 in April 2008. The Hard Rock Cafe created a Deep Zoom collection of their memorabilia. You can explore the site yourself at `http://memorabilia.hardrock.com`. In Figure 1-12, you can see the initial page loaded into the browser.

Once you start zooming in with the mouse wheel, you can get from the lower-resolution images to jthe higher-resolution images. Figure 1-13 shows the detail of a portion of the larger image seen in Figure 1-12.

To build a Deep Zoom application, you don't even need Visual Studio or Expression Blend. You can do it all with the Deep Zoom Composer tool. To get a complete tutorial on how to use the Deep Zoom Composer and integrate your own images into Deep Zoom, visit this URL: `http://community.infragistics.com/redirects/silverlight/deepzoom.aspx`.

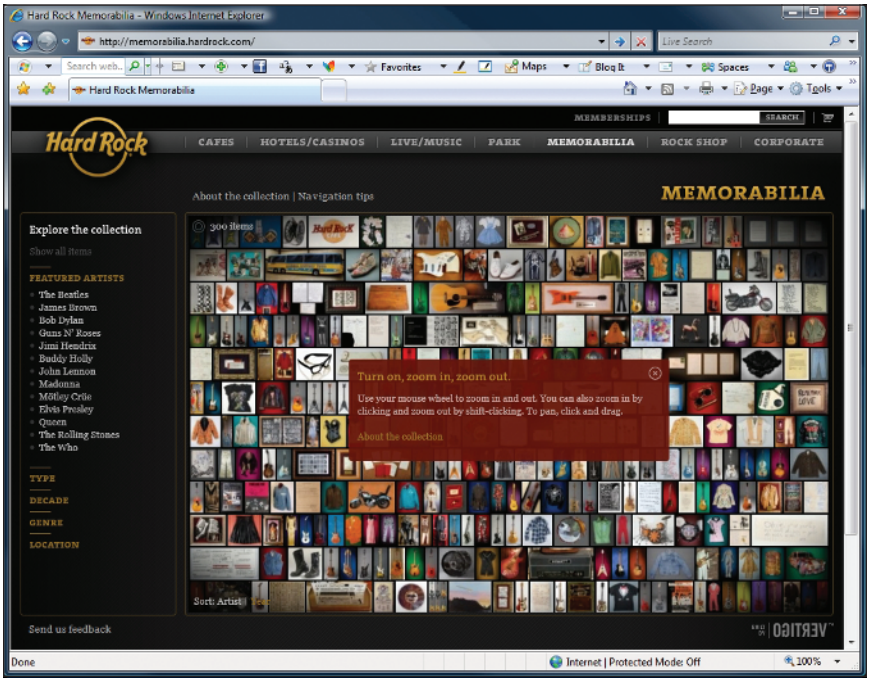

Figure 1-12

Figure 1-13

Summary

Silverlight brings a lot to the table for rich Internet application development. It has progressed from its original release into much more than a simple media player. Silverlight is a platform for developing rich line-of-business applications that have the data and input capability of ASP.NET with the media and interactive capabilities usually reserved for Adobe Flex applications.