1

Introduction to Silverlight

As software development has evolved and the process and end result become more complex and sophisticated, so too has the expectation of end-users. This not only holds true for the desktop application, but also has become increasingly evident in the area of web-based applications. A recent report (see `www.idea.org/find-information.html`) cited the fact that ‘‘designers underestimate the thresholds for an effective site,’’ thus often missing the mark for their users. Furthermore, users want ‘‘good visual design’’ in their web application experience. Thus, a technological shift was needed to bring the delivery of richer applications, similar to those that run on desktops, to the Web. Enter Silverlight.

Silverlight is a new technology from Microsoft that enables developers to build rich, interactive user experiences for web-based applications. At its essence, it is a cross-browser and cross-platform plug-in that Microsoft developed to enable a more compelling and interactive media-based application experience on the Web. Applications that you build using Silverlight give you access to a lighter version of the .NET Framework and run on an end-users’ web client through the Silverlight plug-in. Thus, Silverlight offers a rich programming model that supports not only .NET 3.5 (so technologies such as LINQ, C#, Visual Basic (VB), XML serialization, etc. are supported) and AJAX, but also more dynamic languages such as JavaScript, Ruby, and Python. Silverlight also provides an extensive library of standard controls that you can customize, support for data-binding, styling, and templates. Silverlight also ships with a comprehensive SDK.

Designer and Developer Convergence

At the core of the Silverlight development experience is the convergence of the designer and the developer. That is, with other technologies, it’s been difficult for designers and developers to simultaneously share the same project and code base with a rich set of supported designer and developer tools. Silverlight changes this. By integrating the Expression Blend designer tool and Visual Studio Developer tool, Microsoft has created a new way for designers and developers to collaboratively build applications. For example, often designers will craft and spec a user interface (UI) for an application, but they won’t have a designer tool that enables them to effectively ‘‘stub out’’ the UI for the developer — and then collaboratively iterate through application design should changes be required throughout the project life cycle. Figure 1-1 graphically illustrates an integrated experience between Expression and Visual Studio in which Julia, a UI designer for web applications, can create the UI in Expression, and Ben, a developer of web applications, can work off the same code base to implement the code-behind. The technical heart of this collaborative experience lies in the ability for the Silverlight plug-in not only to render what is called *Extensible Application Markup Language* (XAML — pronounced *zammel*), but also to integrate that XAML with the programming code that an application developer builds as the logic behind the web-based application. And while there are many features that Silverlight supports, it is these two areas that you'll focus on in this chapter and upon which you'll build in subsequent chapters.

Julia – The Designer
Microsoft Silverlight
Ben – The Developer

Figure 1-1

To better understand how the designer and developer work together, let's briefly explore XAML and how this XAML maps to the code-behind that a developer generates.

Overview of XAML

If you're familiar with Windows Presentation Foundation (WPF), one of the core features of .NET Framework 3.5, you already have come across XAML. (You should note that the Silverlight XAML is a subset of the WPF XAML, so there are some differences across them; thus, they're not directly translatable across client-based applications and web-based applications.) If you're not, XAML is essentially an XML-based, declarative language that is used to build the interface for your web application — including such things as the core UI, any animations or graphical elements within the UI, media loading, controls, textual elements, and much more. Because XAML is XML-based, you can also apply styling and templates to it.

You primarily create XAML in one of three ways:

1. A text-based tool such as Notepad (which isn't recommended given the tools' support)
2. The designer within Visual Studio (codenamed ''Cider'')
3. The Expression Blend toolset.

If you're trying to create a relatively simple or straightforward UI, then using the designer within Visual Studio may be the right option. However, we would encourage you, even as a developer, to explore Expression Blend. While it is a broad toolset with many design-specific features, the fact that you can use Expression and Visual Studio simultaneously (against the same source-code project) means that as a developer you can experiment with UI design (in a rich environment) as well as build the custom application code for your web-based application. That said, as a developer you can look to either Visual Studio or Expression, and if you're a designer, you will likely want to stick with using Expression Blend.

Using XAML, you can lay out your UI and then associate events with the controls you place within your UI. XAML provides several ''structural'' containers that you can use to position controls. For example, you can use:

- ❑ `Canvas` and then add other controls to it to build out a complete UI — or smaller part of a larger UI.
- ❑ The `Grid` object in a similar manner, although the `Grid` enables you to create a table-like structure (i.e., a grid layout) with rows and columns.
- ❑ The `StackPanel` to stack UI elements on top of one another.

With each of the above containers, you can add other controls within them and then position these controls by using coordinate properties (e.g., `Canvas.Left = "10"`). Silverlight has a good library of controls out-of-the-box, so you'll have a number of choices available to you when creating your application UI. For example, Listing 1-1 provides a fairly simple UI expressed as XAML. As you look at the XML code, notice that we're creating a `UserControl` object within the UI — the default type of control that is generated when creating a new project. Within the `UserControl`, a `Canvas` represents the host container. Note that we've also set some properties on the `Canvas`, namely, the height, horizontal and vertical alignment, and width. Also note that the example uses a Canvas gradient brush for the background (a style customization that can be applied to your application UI), and includes various properties for each of the buttons, which create the layout, textual content on the button, and the look and feel for the button. Similar to the `Canvas` object, you'll note that each of the elements on the Canvas has several possible properties that can be set, for example, height, horizontal alignment, width, and so on. As you learn XAML, you'll become very familiar with properties as all objects within the Silverlight have a range of properties that can be set. If you use Expression Blend to create your UI, the XAML is created for you so you don't need to worry about entering in the specific properties you want associated with a specific object in your UI. When in Visual Studio, you have the ability to drag-and-drop controls from the Toolbox to the XAML code window, but not to the XAML designer — so you need to enter the specific properties for an object manually.

We'll be exploring different XAML-based UIs throughout this book, but given the fact that we'll mainly be discussing the integration between SharePoint and Silverlight in this book, we assume a basic knowledge of XAML. That said, see the ''Additional References'' section in this chapter for additional references.

Listing 1-1: Multiple controls within XAML container

```
<UserControl
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       x:Class="ExampleGrid.Page"
       Width="650" Height="480"
 xmlns:d=http://schemas.microsoft.com/expression/blend/2008
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
mc:Ignorable="d">

            <Canvas
```

Continued

Listing 1-1: Multiple controls within XAML container *(continued)*

```
        Height="187"
        HorizontalAlignment="Left"
        VerticalAlignment="Top"
        Width="650">

    <Canvas.Background>
       <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
          <GradientStop Color="#FF000000"/>
             <GradientStop Color="#FF9C8F8F" Offset="1"/>
       </LinearGradientBrush>
    </Canvas.Background>

 <Button
       Height="53"
       HorizontalAlignment="Left"
       Margin="0,0,0,0"
       VerticalAlignment="Top"
       Width="100"
       Content="Button"
       Canvas.Top="78"
       Canvas.Left="311.966"/>

 <Button
       Height="53"
       HorizontalAlignment="Left"
       Margin="0,0,0,0"
       VerticalAlignment="Top"
       Width="98"
       Content="Button"
       Canvas.Left="415.966"
       Canvas.Top="78"/>

 <Button
       Height="53"
       HorizontalAlignment="Stretch"
       Margin="0,0,0,0"
       VerticalAlignment="Top"
       Content="Button"
       Canvas.Top="78"
       Canvas.Left="210"
       Width="97.966"/>

 <Button
       Height="53"
       HorizontalAlignment="Right"
       Margin="0,78,210,0"
       VerticalAlignment="Top"
       Content="Button"
       Width="100"/>

 <Button
       Height="53"
       HorizontalAlignment="Right"
```

```
                Margin="0,0,0,0"
                VerticalAlignment="Top"
                Content="Button"
                Width="100"
                Canvas.Top="78"
                Canvas.Left="106"/>

          <Button
                Height="53"
                Width="98"
                Content="Button"
                Canvas.Left="517.966"
                Canvas.Top="78"/>

          </Canvas>
</UserControl>
```

When you render the previous XAML, it displays a Canvas with buttons laid out horizontally — one could (loosely) argue the beginning of a custom navigation (see Figure 1-2). In a real-world scenario, a developer could associate events with each of the buttons by mapping the events to the actual controls within the XAML.

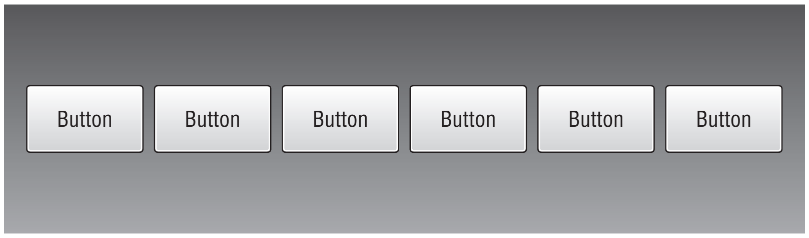


Figure 1-2

XAML also has a rich set of attributes that allow you to, for example, provide animation, shapes, and gradient brush strokes to the objects that you add to your application. Using these extended properties, you can begin to bring life to your UI in new and interactive ways and enhance the graphical display of your application UI. To get you more acquainted with XAML, we have put together another simple example — Listing 1-2. Note that this example creates a `UserControl`, within which several controls have been added to a Canvas with a white background. You'll also note the `x:Class` property; this integrates the code-behind with the XAML file in your Silverlight application. Also note the namespaces, which are standard namespaces created by default when you create a new Silverlight project and are used to render the XAML.

Within this user control are various radio buttons, text-blocks, and buttons that create a simple form. The purpose of the form is to accept some user entry, in this case, salary information, which then surfaces a calculation based on the user entry and the selected state via the radio buttons. Within each of these aforementioned controls, note the fact that they're positioned on the Canvas using the `Canvas.Left` and `Canvas.Top` properties, and where an event handler was added, you've also got an `x:Name` property associated with the XAML control. The event triggers in this simple UI are associated with the button controls, specifically a `click` event for both buttons. One of the `click` events, for example, denoted through `click_calcSalaryClick`, will call the `calcSalaryClick` method when the user clicks the button.

The `calcSalaryClick` method is where the developer enters this collaborative picture; that is, this is the method for which they will create the logic.

If you're not quite sure how you got to this point, don't worry; you'll walk through how to create a Silverlight project in Chapter 2. At this stage, we just want to familiarize you with a simple XAML example and illustrate how it hooks up to the code-behind.

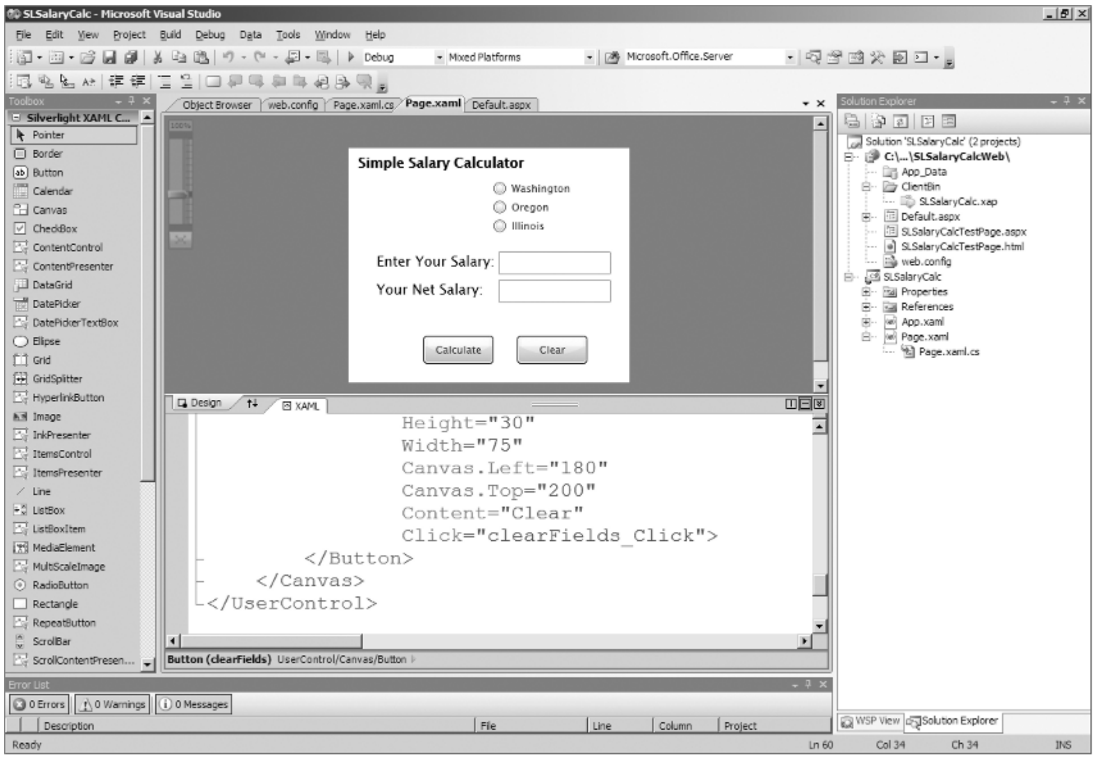
Listing 1-2: Simple user control

```
<UserControl x:Class="SLSalaryCalc.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="300" Height="250">

    <Canvas Background="White">
        <TextBlock
                    Canvas.Left="10"
                    Canvas.Top="5"
                    FontWeight="Bold">
                    Simple Salary Calculator
        </TextBlock>
        <RadioButton
                    x:Name="WAState"
                    Canvas.Left="155"
                    Canvas.Top="30"
                    Content="Washington">
        </RadioButton>
        <RadioButton
                    x:Name="ORState"
                    Canvas.Left="155"
                    Canvas.Top="50"
                    Content="Oregon">
        </RadioButton>
        <RadioButton
                    x:Name="ILState"
                    Canvas.Left="155"
                    Canvas.Top="70"
                    Content="Illinois">
        </RadioButton>


        <TextBlock
                 x:Name="lbl"
                 HorizontalAlignment="Left"
                 Foreground="Black"
                 Canvas.Top="110"
                 Canvas.Left="30">Enter Your Salary:
        </TextBlock>
        <TextBox
                 x:Name="myTextBox"
                 VerticalAlignment="Center"
                 Height="25"
                 Width="120"
                 Canvas.Left="160"
```

```
                  Canvas.Top="110">
          </TextBox>
            <TextBlock
                  x:Name="total"
                  HorizontalAlignment="Left"
                  Foreground="Black"
                  Canvas.Top="140"
                  Canvas.Left="30">Your Net Salary:
          </TextBlock>
            <TextBox
                  x:Name="netSalary"
                  HorizontalAlignment="Left"
                  VerticalAlignment="Center"
                  Canvas.Top="140"
                  Canvas.Left="160"
                  Height="25"
                  Width="120">
          </TextBox>
            <Button
                 x:Name="calcSalary"
                 Height="30"
                 Width="75"
                 Canvas.Left="80"
                 Canvas.Top="200"
                 Content="Calculate"
                 Click="calcSalary_Click">
            </Button>
            <Button x:Name="clearFields"
                 Height="30"
                 Width="75"
                 Canvas.Left="180"
                 Canvas.Top="200"
                 Content="Clear"
                 Click="clearFields_Click">
            </Button>
        </Canvas>
    </UserControl>
```

When you're building your XAML in Visual Studio, Figure 1-3 shows you the view you can expect to work with. The view is split into a designer and a XAML view. You can drag-and-drop Silverlight controls from the Toolbox to the XAML view, and then you add the properties manually in the XAML view. For the full drag-and-drop experience, as mentioned earlier, you'd need to use Expression Blend — which also provides you with a very rich set of tools to manipulate your Silverlight controls. In Figure 1-3, note that you've got the controls available in the Toolbox on the left-hand side, the XAML designer and code view in the middle, and, of course, the Solution Explorer to the right-hand side of the project view. If you're familiar with Visual Studio, then all of this is old news to you; if you're new to Visual Studio, you'll want to make sure you ramp up on Visual Studio to understand your available options (beyond that of Silverlight which are covered in this book).

Because Expression Blend and Visual Studio are integrated, you can right-click on the Page.xaml file in your Solution Explorer and select ''Open in Expression Blend'' (see Figure 1-4). This will open Expression Blend and automatically load the same Silverlight project, opening the XAML view by default.



```
                Height="30"
                Width="75"
                Canvas.Left="180"
                Canvas.Top="200"
                Content="Clear"
                Click="clearFields_Click">
        </Button>
    </Canvas>
</UserControl>
```

Figure 1-3

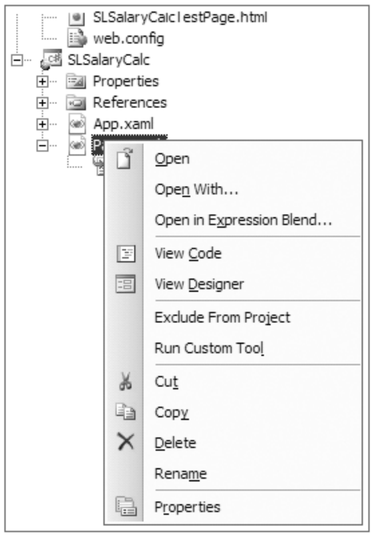
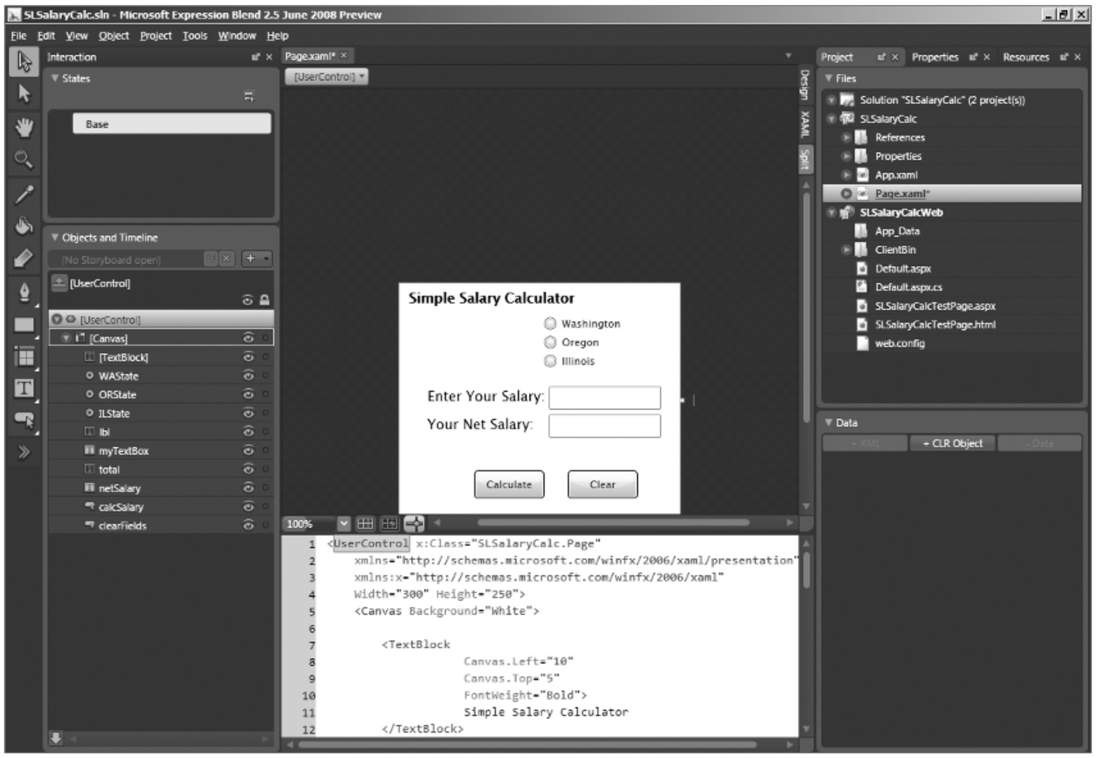


Figure 1-4

When Expression Blend opens (Figure 1-5 shows a simple form in Blend), you can select the Split view, which shows both the user control and the XAML. Note that in Expression Blend, you have a similar view of the project files on the right side of the main view. You also have a Properties and Resources tab, which you use to graphically enhance your XAML control or add additional resources to it. On the left side of the main view, you have your tool palette menu, where you can select specific options such as Paint, Select, Magnify, and so on. You can also select controls and add them to the designer. As mentioned previously, this is where you would drag the controls into a specific position. The build and test experience is consistent across Visual Studio and Expression Blend; that is, when you want to test your application, you press [F5] to test it



```
<UserControl x:Class="SLSalaryCalc.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="300" Height="250">
    <Canvas Background="White">

        <TextBlock
                    Canvas.Left="10"
                    Canvas.Top="5"
                    FontWeight="Bold">
                    Simple Salary Calculator
        </TextBlock>
```

Figure 1-5

Silverlight Code-Behind

Now that you've seen some XAML, which in the example represents a basic form, you're probably wondering what the code-behind looks like. Listing 1-3 shows both the C3 and the VB code for this simple form. Note that in this code-behind, there are two methods that are mapped to the XAML UI: `calcSalary_Click` and `clearFields_Click`. These two methods were added when you created your XAML, but they provide the link between the user controls as objects and the events that are mapped to those objects. For example, the `calcSalary_Click` event is mapped to the Calculate button, which calls the `calculateMyTax` method, which returns a net salary based on the user input. The sample code also checks to see what radio button the user has selected (the state within which they live), which sets the tax rate, which is further used in the calculation of the net salary. You can see that in the sample code you're checking to see if either the `WAState`, `ORState`, or `ILState` radio buttons are checked, and whichever one is selected, you associate with a specific tax rate. To provide some additional guidance on the code, we have added comments inline to both the C# and the VB sample code.

Listing 1-3: Simple form code-behind

C#

```
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 SLSalaryCalc
{
    public partial class Page : UserControl
    {
        public Page()
        {
            InitializeComponent();
        }

        private void calcSalary_Click(object sender, RoutedEventArgs e)
        {
            //Variables used for calculating tax and salary.
            double stateTax = 0.00;
            string userText = "";
            int salary = 0;

            //Checking what state is entered and then setting the tax rate.
            if (WAState.IsChecked == true)
            {
                stateTax = .10;
            }
            else if (ORState.IsChecked == true)
            {
                stateTax = .13;
            }
            else if (ILState.IsChecked == true)
            {
                stateTax = .15;
            }

            //Getting the user entry information (their salary).
            userText = myTextBox.Text;
            salary = Int32.Parse(userText);

            //Calling the method to calculate the tax rate.
                netSalary.Text = calculateMyTax(salary, stateTax);
        }

        private void clearFields_Click(object sender, RoutedEventArgs e)
        {
            myTextBox.Text = "";
            netSalary.Text = "";
        }

        //Method to calculate the tax rate.
        private string calculateMyTax(int salary, double stateTax)
        {
            double tax;
            double totalSalary;
            tax = salary * .28 * stateTax;
            totalSalary = salary - tax;
            return totalSalary.ToString();
        }



    }
}
```

VB

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Net
Imports System.Windows
Imports System.Windows.Controls
Imports System.Windows.Documents
Imports System.Windows.Input
Imports System.Windows.Media
Imports System.Windows.Media.Animation
Imports System.Windows.Shapes
Partial Public Class Page
    Inherits UserControl

    Public Sub New()
        InitializeComponent()
    End Sub


    Private Sub calcSalary_Click(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs)
'Variables used for calculating tax and salary
        Dim stateTax As Double = 0.0
        Dim userText As String = ""
```

Continued

Listing 1-3: Simple form code-behind *(continued)*

```
        Dim salary As Integer = 0

'Checking what state is entered and then setting the tax rate.
        If WAState.IsChecked = True Then
            stateTax = 0.1
        ElseIf ORState.IsChecked = True Then
            stateTax = 0.13
        ElseIf ILState.IsChecked = True Then
            stateTax = 0.15
        End If

'Getting the user entry information (their salary).
        userText = myTextBox.Text
        salary = Int32.Parse(userText)

'Calling the method to calculate the tax rate.
        netSalary.Text = calculateMyTax(salary, stateTax)

    End Sub

    Private Sub clearFields_Click(ByVal sender As System.Object, _
ByVal e As System.Windows.RoutedEventArgs)

        myTextBox.Text = ""
        netSalary.Text = ""

    End Sub

    'Method to calculate the tax rate
    Private Function calculateMyTax(ByVal salaryParam As Integer, _
ByVal stateTaxParam As Double)
        Dim tax As Double = 0.0
        Dim totalSalary As Double = 0.0
        Dim returnValue As Double = 0.0

        tax = salaryParam * 0.28 * stateTaxParam
        totalSalary = salaryParam - tax
        returnValue = totalSalary.ToString
        Return returnValue
    End Function
End Class
```

The great thing is, when you build and deploy this simple form, and — in the context of this book — integrate with SharePoint, Silverlight executes the applications within SharePoint using the Silverlight plug-in that the user downloads and installs on his or her client. However, what's really nice is that it does so without needing any additional hookup code; that is, all of the code that you require to run your application is built and deployed as a XAP (pronounced *zap*) Package, which can be unzipped and inspected when trying to understand what files are packaged for deployment. Thus, when integrating Silverlight into your SharePoint sites, you have the option to deploy these applications to your site very easily, and they can act as independent components within your overall SharePoint site. Depending on the purpose of your Silverlight application, this may be a desired outcome; else you may want to tie Silverlight to specific SharePoint objects (e.g., skinning a list with Silverlight). We will discuss both of these scenarios throughout this book.

Developer Environment and Tools

One of the key things you'll need to understand before you get under way is how to set up a development sandbox environment so that you can build and integrate Silverlight applications into SharePoint. To do this, you'll want to make sure you have the following installed:

- ❑ Windows Server (2003 or 2008)
- ❑ Windows SharePoint Services 3.0 and Windows SharePoint Services Service Pack 1
- ❑ Microsoft Office SharePoint Server 2007 and 2007 Microsoft Office SharePoint Servers Service Pack 1 (SP1)
- ❑ Visual Studio 2008 Professional Edition (or above)
- ❑ Silverlight Tools Beta 2 for Visual Studio 2008
- ❑ Silverlight Beta 2 Runtime
- ❑ Windows SharePoint Services 3.0 Tools: Visual Studio Extensions, Version 1.2
- ❑ Expression Blend

Note that within this book, we'll assume that you're running Windows Server 2008. The links to some of the software listed above (where they are free downloads) are in the section ''Additional References'' at the end of this chapter.

After you've installed each of these, you now have to ensure that your development environment is configured for SharePoint and Silverlight integration. To do this, you must do the following:

1. Change the MIME type in Internet Information Services (IIS) to support Silverlight applications.
2. Change the web.config file in your root SharePoint direction to support Silverlight applications.
3. Ensure that the System.Web.Silverlight.dll is in your global assembly cache (GAC).

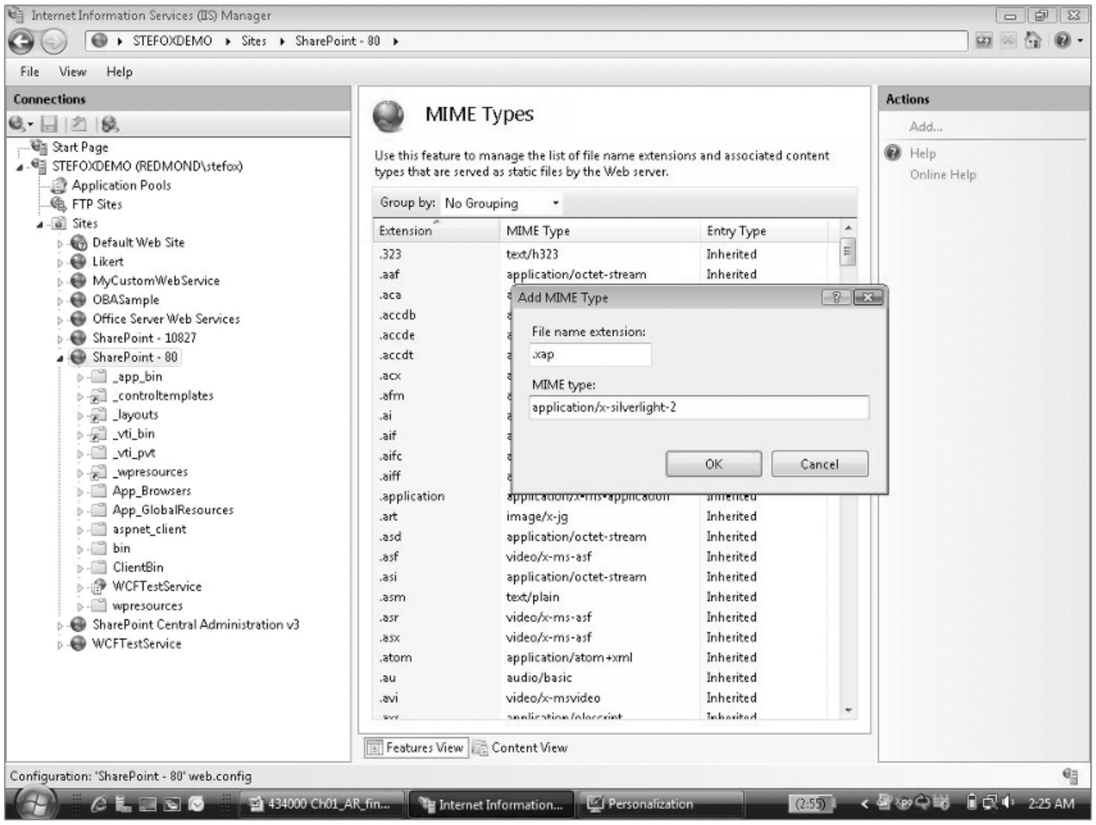
Let's examine each of these steps.

Changing the MIME Type

You must add MIME support for Silverlight applications that you build and deploy to your SharePoint server. If you do not, when trying to integrate your Silverlight application with SharePoint, it will not render properly. To add MIME support for Silverlight applications, do the following:

1. Click Start ➢ Administrative Tools, and then select Internet Information Services (IIS) Manager. This will open IIS.
2. Under Web Sites in the folder structure, click on the web application that represents your SharePoint site (e.g., SharePoint ➢ –80).
3. In the main IIS window, click ''MIME Types,'' and then click ''Open Feature.''

4. In the Mime Type view, click Add, add the file-type extension, and type **.xap** as the ‘‘File name extension’’ and **application/x-silverlight-2** as the ‘‘MIME type’’ (see Figure 1-6).



Use this feature to manage the list of file name extensions and associated content types that are served as static files by the Web server.

Extension	MIME Type	Entry Type
.323	text/h323	Inherited
.aaf	application/octet-stream	Inherited
.aca		
.accdb		
.accde		
.accdt		
.acx		
.afm		
.ai		
.aif		
.aifc		
.aiff		
.application	application/x-ms-application	Inherited
.art	image/x-jg	Inherited
.asd	application/octet-stream	Inherited
.asf	video/x-ms-asf	Inherited
.asi	application/octet-stream	Inherited
.asm	text/plain	Inherited
.asr	video/x-ms-asf	Inherited
.asx	video/x-ms-asf	Inherited
.atom	application/atom+xml	Inherited
.au	audio/basic	Inherited
.avi	video/x-msvideo	Inherited
.axs	application/olescript	Inherited

File name extension: .xap

MIME type: application/x-silverlight-2

Configuration: 'SharePoint - 80' web.config

Figure 1-6

5. Click OK to add the new MIME type to IIS.

Editing the web.config File

A web.config file houses all of the configuration settings that are required by your web application. By default, the web.config file that resides in the root folder of your SharePoint directory does not support the rendering of Silverlight applications. Thus, you need to ensure that all of the necessary elements are added to enable Silverlight support within your SharePoint server.

You can discover what you need to add to the web.config file through a couple of ways. The first is to do a search on the Web and find examples. The second, and easier, way is if you create a new project and select the option to build the Silverlight application using a web site instead of a dynamic HTML page as your test environment, the Visual Studio project creates a web.config file for you. If you open this file within Visual Studio, you can see all of the necessary elements that need to be added to your SharePoint web.config file. For example, Listing 1-4 shows the web.config file for the example we showed you earlier in this chapter.

Listing 1-4: Sample web.config file generated by Visual Studio

```
<?xml version="1.0"?>

<configuration>
      <configSections>
            <sectionGroup name="system.web.extensions"
type="System.Web.Configuration.SystemWebExtensionsSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
                  <sectionGroup name="scripting"
type="System.Web.Configuration.ScriptingSectionGroup, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35">
                        <section name="scriptResourceHandler"
type="System.Web.Configuration.ScriptingScriptResourceHandlerSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
requirePermission="false" allowDefinition="MachineToApplication"/>
                        <sectionGroup name="webServices"
type="System.Web.Configuration.ScriptingWebServicesSectionGroup,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35">
                             <section name="jsonSerialization"
type="System.Web.Configuration.ScriptingJsonSerializationSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35" requirePermission="false"
allowDefinition="Everywhere"/>
                             <section name="profileService"
type="System.Web.Configuration.ScriptingProfileServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
requirePermission="false" allowDefinition="MachineToApplication"/>
                             <section name="authenticationService"
type="System.Web.Configuration.ScriptingAuthenticationServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
requirePermission="false" allowDefinition="MachineToApplication"/>
                             <section name="roleService"
type="System.Web.Configuration.ScriptingRoleServiceSection,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"
requirePermission="false" allowDefinition="MachineToApplication"/>
                        </sectionGroup>
                  </sectionGroup>
            </sectionGroup>
      </configSections>
      <appSettings/>
      <connectionStrings/>
      <system.web>
```

Continued

Listing 1-4: Sample web.config file generated by Visual Studio *(continued)*

```
            <compilation debug="true">
                  <assemblies>

                             <add assembly="System.Core, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                             <add assembly="System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                             <add assembly="System.Data.DataSetExtensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                             <add assembly="System.Xml.Linq, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
                            <add assembly="System.Web.Silverlight, Version=2.0.5.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/></assemblies>
            </compilation>
           <authentication mode="Windows"/>
           <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm" />
        </customErrors>
        -->
            <pages>
                  <controls>
                        <add tagPrefix="asp"
namespace="System.Web.UI" assembly="System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                        <add tagPrefix="asp" namespace="System.Web.UI.WebControls"
assembly="System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                  </controls>
            </pages>
            <httpHandlers>
                  <remove verb="*" path="*.asmx"/>
                  <add verb="*" path="*.asmx" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                  <add verb="*" path="*_AppService.axd" validate="false"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                  <add verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35" validate="false"/>
            </httpHandlers>
            <httpModules>

                  <add name="ScriptModule" type="System.Web.Handlers.ScriptModule,
System.Web.Extensions, Version=3.5.0.0, Culture=neutral,
PublicKeyToken=31BF3856AD364E35"/>
            </httpModules>
      </system.web>
      <system.codedom>
            <compilers>
                <compiler language="c#;cs;csharp" extension=".cs" warningLevel="4"
type="Microsoft.CSharp.CSharpCodeProvider, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089">

                        <providerOption name="CompilerVersion" value="v3.5"/>
                        <providerOption name="WarnAsError" value="false"/>
                  </compiler>
                  <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb"
warningLevel="4" type="Microsoft.VisualBasic.VBCodeProvider, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089">
                        <providerOption name="CompilerVersion" value="v3.5"/>
                        <providerOption name="OptionInfer" value="true"/>
                        <providerOption name="WarnAsError" value="false"/>
                  </compiler>
            </compilers>
      </system.codedom>
      <system.webServer>
            <validation validateIntegratedModeConfiguration="false"/>
            <modules>
                  <remove name="ScriptModule"/>
                  <add name="ScriptModule"
preCondition="managedHandler"
type="System.Web.Handlers.ScriptModule, System.Web.Extensions, Version=3.5.0.0,
Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
            </modules>
            <handlers>
                  <remove name="WebServiceHandlerFactory-Integrated"/>
                  <remove name="ScriptHandlerFactory"/>
                  <remove name="ScriptHandlerFactoryAppServices"/>
                  <remove name="ScriptResource"/>
                  <add name="ScriptHandlerFactory" verb="*" path="*.asmx"
preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                  <add name="ScriptHandlerFactoryAppServices" verb="*"
path="*_AppService.axd" preCondition="integratedMode"
type="System.Web.Script.Services.ScriptHandlerFactory, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
                  <add name="ScriptResource" preCondition="integratedMode"
verb="GET,HEAD" path="ScriptResource.axd"
type="System.Web.Handlers.ScriptResourceHandler, System.Web.Extensions,
Version=3.5.0.0, Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
            </handlers>
      </system.webServer>
      <runtime>
            <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
                  <dependentAssembly>
                       <assemblyIdentity
```

Continued

Listing 1-4: Sample web.config file generated by Visual Studio *(continued)*

```
name="System.Web.Extensions" publicKeyToken="31bf3856ad364e35"/>
                       <bindingRedirect oldVersion="1.0.0.0-1.1.0.0"
newVersion="3.5.0.0"/>
                   </dependentAssembly>
                   <dependentAssembly>
                       <assemblyIdentity
name="System.Web.Extensions.Design" publicKeyToken="31bf3856ad364e35"/>
                       <bindingRedirect oldVersion="1.0.0.0-1.1.0.0"
newVersion="3.5.0.0"/>
                   </dependentAssembly>
            </assemblyBinding>
      </runtime>
</configuration>
```

Although this will get you most of the way there toward ensuring that you have the appropriate support for your Silverlight applications in SharePoint, one final addition is required: Add the Silverlight application as a safe control within your SharePoint web.config file. This is often overlooked and can cause some headaches when trying to troubleshoot Silverlight applications not rendering. For example, the following line of code shows the `SafeControl` entry for the Silverlight application discussed earlier in the chapter:

```
<SafeControl Assembly="SimpleSalary, Version=1.0.0.0, Culture=neutral,
PublicKeyToken=9f4da00116c38ec5" Namespace="SimpleSalary"
TypeName="SimpleSalaryCalc" Safe="True"/>
```

After you've completed these amendments to the web.config file in your root SharePoint folder, you should be ready to move to the last step in configuring your environment.

Copying the Silverlight DLL to Your GAC

The last step in configuring your server is simple: Make sure that the System.Web.Silverlight.dll exists within your global assembly cache (GAC). The GAC will typically reside in the root Windows directory under the assembly folder. Also, if you've previously been using earlier versions of Silverlight, make sure that the version is correct — the DLL is named the same across versions, but Silverlight 2 applications won't render using the Silverlight 1 (or Silverlight Beta 1) DLL. Therefore, delete the old version in your GAC and then recopy the newer Silverlight 2 DLL to the assembly folder.

Make sure that you're running as the administrator on the machine, or else you'll get an ''Access Denied'' message. You can get around this by right-clicking on the Visual Studio 2008 tools and running as administrator and then using the gacutil tool to copy files to the GAC. Also, as a general rule when using Visual Studio 2008, right-click and select ''Run as Administrator''; otherwise, you will not have full access to your machine when building and deploying applications.

At this point, your SharePoint server will be configured to support Silverlight applications and, assuming you've installed all of the applications listed at the beginning of this section, your sandbox environment will be ready to go. Before moving on to the next chapter and building your first Silverlight and SharePoint integration, let's talk a bit about why you would want to integrate SharePoint and Silverlight.

Why Silverlight and SharePoint?

To return to the beginning of the chapter for a brief minute, we repeat that increasingly, the user expectation is to demand a richer user experience when interacting with web-based applications. Furthermore, the demands of these web-based applications go beyond just supporting a single browser technology or platform — thus the need for Silverlight to support not only a richer, more compelling user experience, but also one that supports the delivery of these applications to multiple platforms, browsers, and audiences. In this chapter, you've seen a simple Silverlight example, but there are many possibilities when building Silverlight applications. For example, Figure 1-7 illustrates a Silverlight-enabled web application that allows users to drill into showcase Silverlight examples through an interactive and dynamic map. This is a great example of what Silverlight can do and is available for download off the Silverlight.net web site — along with many other samples and training.

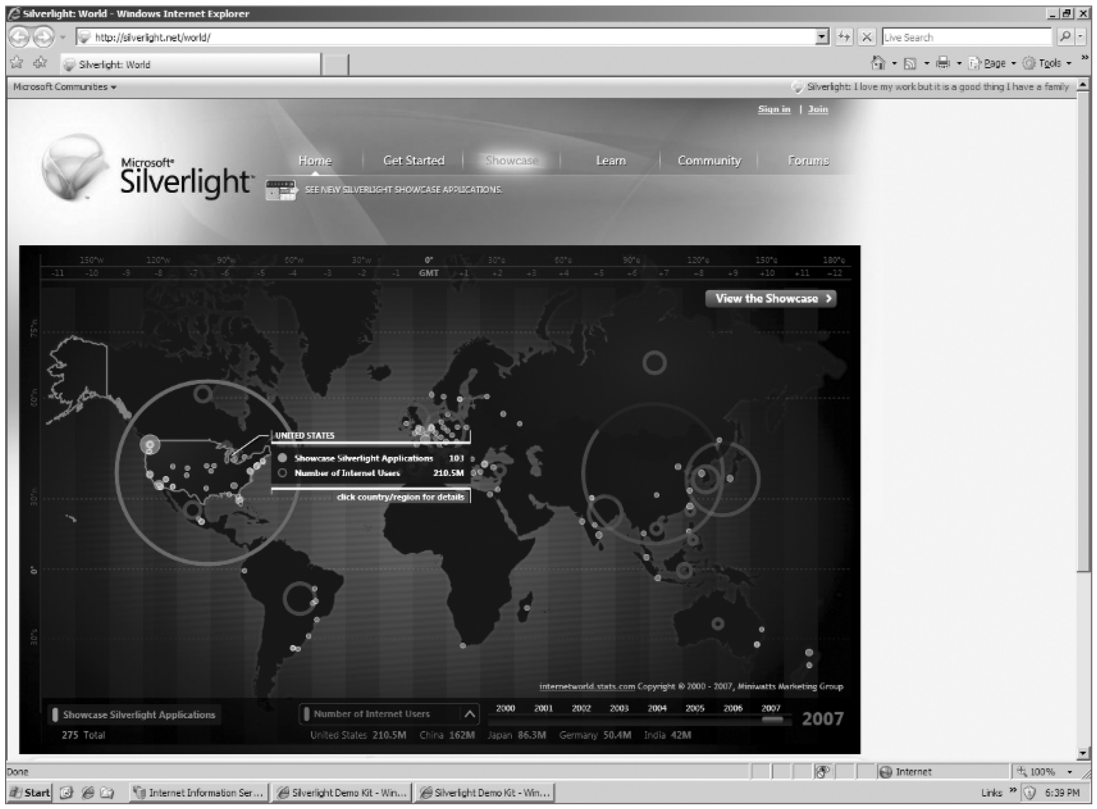

Silverlight: World - Windows Internet Explorer
http://silverlight.net/world/
Microsoft Silverlight
Home
Get Started
Showcase
Learn
Community
Forums
SEE NEW SILVERLIGHT SHOWCASE APPLICATIONS.
View the Showcase
UNITED STATES
Showcase Silverlight Applications 103
Number of Internet Users 210.5M
click country/region for details
Showcase Silverlight Applications
275 Total
Number of Internet Users
United States 210.5M China 162M Japan 86.3M Germany 50.4M India 42M
2007

Figure 1-7

While Silverlight can create some powerful applications, why integrate it with SharePoint? The primary reasons are twofold:

- ❑ While there are some great templates that ship with SharePoint, they don't go all the way to customize the site in the way that many organizations want, so Silverlight is a natural extension for

SharePoint to really take the user experience to the next level while still using the SharePoint infrastructure — especially when building on SharePoint for the wider Web.

- As more companies more fully utilize their SharePoint investments (both as intranets and externally facing web sites), they are looking for tools that can take these sites to the next level. Faced with the same pressures to build richer user experiences for their web-based applications, many companies are using Microsoft Silverlight 2 to reach new user experience levels in SharePoint — thus the importance of the integration between SharePoint and Silverlight.

Beyond these two reasons, Silverlight offers a lot to the developer:

- As discussed, it offers a great integration for the designer to work with the developer with rich tools' support for both.
- It ships with many supported features such as managed code, .NET Framework 3.5, dynamic languages, a rich SDK, data-binding, and more. These are not only great for the aspiring Silverlight developer, but for the .NET developer the transition from your current developer skills is natural because you're still leveraging many of the core .NET features; you're just building the application for a different delivery mechanism.
- As a developer, you're also more able to rapidly build and deploy self-contained applications that have less ''hookup'' code than traditional script-based web applications.
- There's also an added layer of security given the fact that you can develop your applications with a managed-code environment.
- The support network for Silverlight is strong and growing quickly. The attention being paid to this technology is high, and the technology is very hot. Combining it with SharePoint, whose reach is also very strong, is something you're going to hear a lot about moving forward.

Summary

In this chapter, you were introduced to what Silverlight is and answered the question of why (at least at a high level) Silverlight is a great fit for SharePoint — while much of this you'll determine on your own throughout this book. This chapter contained a couple of simple XAML examples to get you familiar with the declarative language that describes the Silverlight application UI, as well as providing you with the C# and VB code-behind for this simple Silverlight application. That said, you can walk away from this chapter knowing you've learned three key things:

- XAML is the declarative language that defines the UI for Silverlight applications, and you have different options for building out the code-behind, which include managed code and dynamic languages.
- Designers and developers have rich tools' support in not only Visual Studio 2008, but also in the Expression Blend suite of tools to aid in UI design.
- Silverlight applications are not difficult to build; however, there is some setup that is required when setting up your sandbox development environment — this is often the leading cause of issues with applications not rendering.

In the next chapter, you'll walk through the creation of your first Silverlight application, so you can begin thinking about how you can begin your own integrations with SharePoint and put what you've learned in this chapter to practice.

Additional References

2007 Microsoft Office SharePoint Servers Service Pack 1 (SP1): `www.microsoft.com/downloads/details.aspx?FamilyId=AD59175C-AD6A-4027-8C2F-DB25322F791B&displaylang=en`.

Laurence Moroney, *Introducing Silverlight 2* (Microsoft Press, 2008).

Patrick Tisseghan's Blog: `www.u2u.info/Blogs/Patrick/default.aspx`.

Silverlight Web Site: `www.silverlight.net`.

Silverlight Beta 2 Runtime: `www.silverlight.net`.

Silverlight Tools Beta 2 for Visual Studio 2008: `www.microsoft.com/downloads/details.aspx?FamilyId=50A9EC01-267B-4521-B7D7-C0DBA8866434&displaylang=en`.

Windows SharePoint Services 3.0 and Windows SharePoint Services Service Pack 1: `www.microsoft.com/downloads/details.aspx?familyid=4191A531-A2E9-45E4-B71E-5B0B17108BD2&displaylang=en`.

Windows SharePoint Services 3.0 Tools: Visual Studio Extensions, Version 1.2: `www.microsoft.com/downloads/details.aspx?familyid=7BF65B28-06E2-4E87-9BAD-086E32185E68&displaylang=en`.