Accelerated Silverlight 2

Copyright © 2008 by Jeff Scanlon

ISBN-13 (pbk): 978-1-4302-1076-4

ISBN-10 (pbk): 1-4302-1076-1

ISBN-13 (electronic): 978-1-4302-1075-7

ISBN-10 (electronic): 1-4302-1075-3

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

Lead Editor: Ewan Buckingham
Technical Reviewer: Fabio Claudio Ferracchiati
Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Richard Dal Porto
Copy Editors: Damon Larson, Ami Knox
Associate Production Director: Kari Brooks-Copony
Production Editor: Jill Ellis
Compositor: Kinetic Publishing Services, LLC
Proofreader: Linda Seifert
Indexer: Julie Grady
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

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

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

The source code for this book is available to readers at `http://www.apress.com`. You may need to answer questions pertaining to this book in order to successfully download the code.

CHAPTER 1

Introducing Silverlight

Silverlight is an exciting new technology from Microsoft for developing rich user experiences that are accessible on a variety of platforms. Stated succinctly, Silverlight is a cross-platform Common Language Runtime (CLR) with a strong presentation framework for compositing user interfaces and displaying images and video, making development of rich user experiences much easier than before. At the core of Silverlight is a new markup language called Extensible Application Markup Language, or XAML (pronounced *zammel*). XAML helps designers and developers work more effectively with each other since it is a declarative language with tools built around it. Silverlight 2.0 is a natural extension to technologies already in existence, specifically .NET and Windows Presentation Foundation (WPF). If you strip out the parts of .NET that just aren't needed or don't easily work across platforms (such as interoperating with COM), add in an implementation of XAML that is close to WPF's, and mix in a few new things such as browser interoperability and ability to execute dynamic languages such as Python (IronPython, as the .NET implementation is called), you end up with Silverlight 2.0.

Developing applications that work on multiple platforms is a difficult problem. What constitutes a platform is an important question, and for the purposes of this book, it is any unique host environment that provides an execution environment for code. If you give it some thought, it is easy to categorize Windows XP, Windows Vista, OS X, and Linux as platforms; but Firefox, Internet Explorer 6, Internet Explorer 7, Opera, and so on also count as platforms. If you've done any web development targeting multiple browsers, you're familiar with the inherent headaches in getting a web site to render and operate the same on Internet Explorer as it does on Firefox and others. Technically, this web site is a cross-platform application. The goal of Silverlight is to create a consistent execution environment across different browsers and operating systems.

There is no magical reason why a cross-platform application is automatically "good." Any responsible software engineering starts with a careful examination of the business reasons for a project. If all users are on a single platform, such as Windows, there is no reason to spend extra development time ensuring that the software also works on other platforms. Also, a significant amount of software that enables business applications (data and business logic layers) has no need to work on multiple platforms (though it can potentially be *consumed* by different platforms), and in fact benefits from platform-specific optimizations.

However, cross-platform applications are definitely important—as is best evidenced by web sites that are usable, generally, on any browser. The ability to develop cross-platform applications is of the most importance when the potential users for an application are on multiple platforms. This is a rather obvious statement, but it is important to note that development of a cross-platform application offers no inherent benefits if all users are on a single platform.

That is, unless the cross-platform aspect is obtained free or near-free (therefore helping to future-proof the application if the user base changes). This concept of "free or near-free" is important—software engineering is already a challenging endeavor, and if making software cross-platform is difficult to implement, it requires either significantly more development time for a single code base, or a second code base for a different platform that replicates the functionality of the first (not to mention a third or fourth code base if other platforms must be supported). Without question, this means more time, more money, and more development resources are needed. Optimally, we want a relatively easy way to create cross-platform applications. Fortunately, a number of frameworks have attempted to make the creation of cross-platform applications free or near-free.

Cross-Platform Frameworks

Frameworks for developing cross-platform applications are not new. Even the C language is arguably cross-platform, since the source can be written once and compiled on each target platform, thus enabling portability of projects written in C. While arguments over what truly constitutes cross-platform can be interesting, they aren't of much practical use for us here, so let's take a brief look at the serious contenders for developing cross-platform applications.

Qt

Qt (pronounced *cute*) is a cross-platform application development toolkit mainly for C++; however, it has support for other languages such as Java. The significant benefit to Qt is that programs execute natively after compilation (i.e., no new virtual machine is needed). The cross-platform nature of Qt is provided at the source level, as long as developers utilize Qt's platform-agnostic API. The major downsides to Qt are the learning curve for developers and the degree to which applications might become intertwined with Qt (though this might be acceptable to many organizations). Visit `www.trolltech.com/products/qt` for more information.

The Java Platform

The Java platform is possibly the closest comparison to Silverlight on the market. Much like .NET, the Java platform is a managed environment. Until Silverlight, though, .NET was only available on Windows. Both platforms provide the ability to compile a program and immediately execute it on multiple platforms. The Java platform and Silverlight approach this similarly: an execution environment (known as a virtual machine) is developed for each platform where programs might be run. Java source code is compiled to Java bytecode, which is then executed by the Java virtual machine. The downsides to this approach are the plethora of virtual machines that can be created, each with potential quirks that sometimes affect existing applications, and the time cost of starting up a Java virtual machine on a web site (you've no doubt seen the gray rectangle and the loading symbol on web pages). Sun also has a more direct competitor to Silverlight called JavaFX, a framework including a scripting language to more easily create Java applications. This framework makes the most sense for institutions and developers already used to working in the Java environment or needing to extend their existing Java applications. Visit `http://java.sun.com/javafx/` if you are curious about learning more.

Flash/Flex

Flash is, by far, the most popular comparison to Silverlight. A browser plug-in that enables execution of rich content for the Web—doesn't that sound familiar? This comparison is made even more explicit with Adobe releasing Flex, an environment for executing rich applications in the browser and on the desktop. While there are some feature differences between Flex and Silverlight that can make one more appealing than the other, Flex is a viable alternative to Silverlight; however, it caters to a different set of developers than Silverlight does. Flex capitalizes on the languages people already know, including JavaScript, HTML, CSS, and ActionScript. Silverlight, however, provides a brand new markup language, but is an incredibly natural platform to develop on if you're already a .NET developer. Visit `www.adobe.com/products/flex/` if you want to learn more about Flex.

Silverlight

This brings us to the subject of this book: Silverlight 2.0. The .NET 3.0 Framework included the first release of WPF, along with other key technologies. With WPF came XAML, essentially a way to create applications in markup (there is an almost one-to-one correspondence between XAML constructs and code). While XAML is not necessarily tied to presentation logic, the two most visible uses of it are in WPF and Silverlight. Silverlight's implementation of XAML is a subset of WPF's—it does not have 3D support, for example. While Silverlight does contain a CLR, it has absolutely no dependence on any of the .NET Framework versions—the Silverlight plug-in brings with it a CLR and a base class library all its own.

If you are already a .NET developer, you will be in familiar territory after learning XAML and its features. The correspondence of XAML to classes in .NET is a major strength, and the tool support built around XAML for designers and developers is strong and growing.

The History of Silverlight

Before the MIX conference in March 2007, Silverlight was known by the relatively boring but descriptive name WPF/E, which stands for Windows Presentation Foundation/Everywhere. While the details were sparse at the time, the rough goal of the technology was clear: a browser-hosted version of WPF. Silverlight 1.0 was unveiled at the conference and would no longer be known as WPF/E. This initial release of Silverlight did not have a CLR or anywhere close to the capabilities provided by 2.0. What it did have, though, is support for a small subset of XAML and a variety of capabilities that foreshadowed the future of Silverlight. Possibly the most obvious aspect of Silverlight 1.0 is that applications are written either completely in XAML or in a mix of XAML and JavaScript. Since there is no CLR, there is no compilation step, and the JavaScript is interpreted on the client. The major features supported by Silverlight 1.0 are

Core architecture: This includes `DependencyObject` at the root, and `UIElement` forming the base of user interface classes (but no `FrameworkElement` class).

Basic layout: The Canvas is the only layout component, so user interface elements can only be placed using absolute positions.

Basic controls: The TextBlock and Run controls are provided to display text. In terms of handling user input, nothing specialized is provided. This limitation extended to Silverlight 1, and the full control architecture debuted when Silverlight 2.0 was first released in beta.

2D graphics: `Geometry`-based classes (which are flexible but can't be directly placed on a user interface) and `Shape`-based classes (which can be directly placed on a user interface) provide the ability to draw 2D shapes.

Media: Many early Silverlight applications showcased the image and video support provided by Silverlight. Also included is support for easily downloading media such as images so that bandwidth could be utilized more effectively.

Animation: The `Storyboard` class known from WPF became part of the XAML implementation in this first release of Silverlight, providing the ability to animate different user interface elements in a variety of ways.

Brushes and transforms: Brushes such as the image brush, video brush, and color brushes (solid colors and gradients) have been in Silverlight since this initial release.

Silverlight 1.0 does require a plug-in on the client side, and in the spirit of Microsoft's commitment to backward compatibility, Silverlight 1.0 applications still work on Silverlight 2.0. Two of the most important parts of the latest release of Silverlight that are not present in Silverlight 1.0 are a rich set of controls and performance advantages due to compiled code.

Soon after Silverlight 1.0 was released, the next version of Silverlight was released in preview form. This preview release was known as Silverlight 1.1, the most significant aspect of which is the cross-platform CLR. While Silverlight 1.0 could be used to develop some impressive media-based applications, the possibilities greatly expand with the ability to target the .NET platform and know that the application will run on multiple host platforms. The biggest missing feature from Silverlight 1.1 was a set of standard controls. This made developing useful user interfaces difficult. Handling input events was also difficult since events could only be captured on the root container. You then had to manually propagate the events to child objects. Input focus was also tricky.

After several months, as it got closer to the MIX08 conference in March 2007, Microsoft revealed that Silverlight 1.1 would actually be released as Silverlight 2.0 since the feature set grew so much. Fortunately, the 2.0 release of Silverlight includes a standard control set (probably everything you would want except for a tree control and a combo box control) and an input event system that saves Silverlight developers the tedium of handling input events manually. Silverlight 2.0 comes with much more than just these important additions. We get strong networking support, even including the ability to communicate over sockets. We get the `System.Xml` classes, though they are a subset of the same classes in the .NET Framework on Windows. We get the ability to develop in any .NET language we want—including dynamic languages such as compiled JavaScript and IronPython. This book will cover Silverlight 2.0 in detail and help you quickly get up to speed on this new technology.

Creating Your First Application

Since Visual Studio 2008 supports .NET 3.0 and 3.5, WPF application support is already built in. However, since the release of Visual Studio 2008 preceded Silverlight 2.0, Silverlight support is not provided out of the box. After you install the Silverlight 2.0 SDK, Visual Studio 2008 gains support for building Silverlight 2.0 applications and class libraries, and adds a read-only design surface and appropriate IntelliSense in the XAML editor. While Visual Studio is an established tool targeted to developers, tool support for WPF and Silverlight for both designers and developers is necessary. This need is satisfied by the Expression suite of products from Microsoft. Let's install the Silverlight 2.0 SDK and briefly explore it and one of the Expression tools.

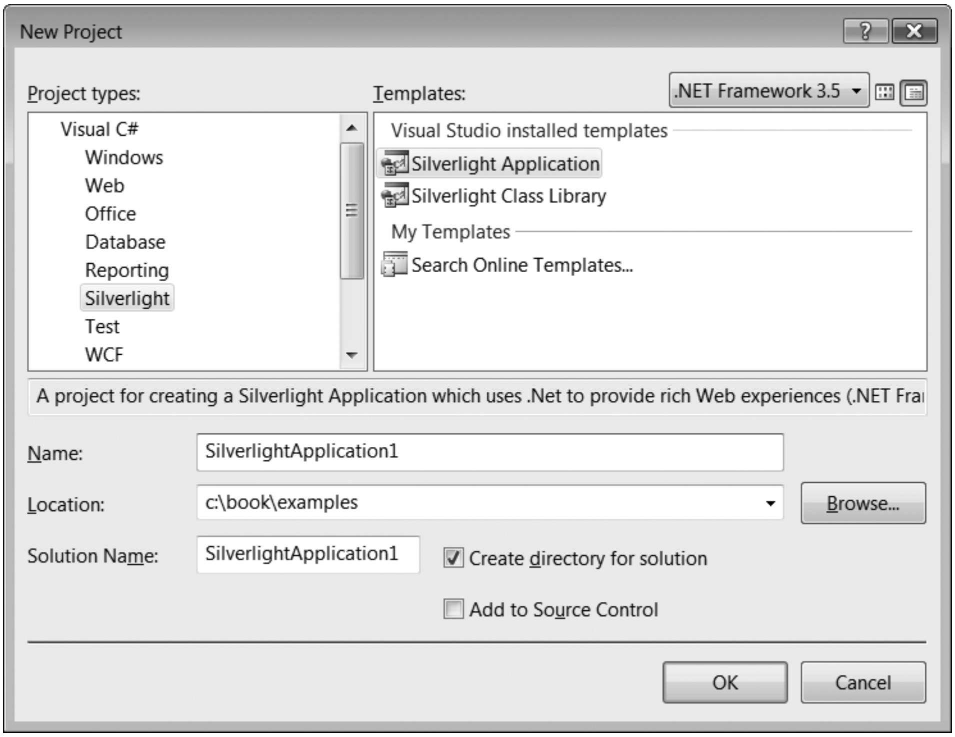
Visit `http://silverlight.net/GetStarted/` and download Microsoft Silverlight Tools for Visual Studio 2008. This single download includes the SDK for Visual Studio 2008 (Standard edition and above) and the runtime for Silverlight 2.0.

Two more tools are available at this page: Expression Blend and the Deep Zoom Composer. If you have seen the Hard Rock Memorabilia site, you have seen a product of the Deep Zoom Composer. This technology will be discussed when we take a closer look at media support in Silverlight in Chapter 5. For now, just download and install Expression Blend 2.5 Preview.

When you edit a XAML file in a WPF application using Visual Studio, you have access to a toolbox of controls, a design surface onto which you can drag and drop controls, and a text editor view of the XAML code. When you edit a XAML file in a Silverlight application, you still have these three elements, but the design surface is read-only. This is probably a result of the Silverlight package being an add-on to Visual Studio. One thing you can do, though, is drag and drop controls from the toolbox onto the text editor. This can help a lot when you want to work with XAML exclusively in Visual Studio.

You can use Expression Blend if you want a full drag-and-drop user interface construction tool for Silverlight. It's possible to use Expression Blend simultaneously with Visual Studio. Modifications to both XAML files and the project/solution file are fine, since when you switch from one tool to the other, the tool will reload the updated files.

Start by loading Visual Studio 2008 and creating a new project (see Figure 1-1).



A project for creating a Silverlight Application which uses .Net to provide rich Web experiences (.NET Fra

Name: SilverlightApplication1

Location: c:\book\examples

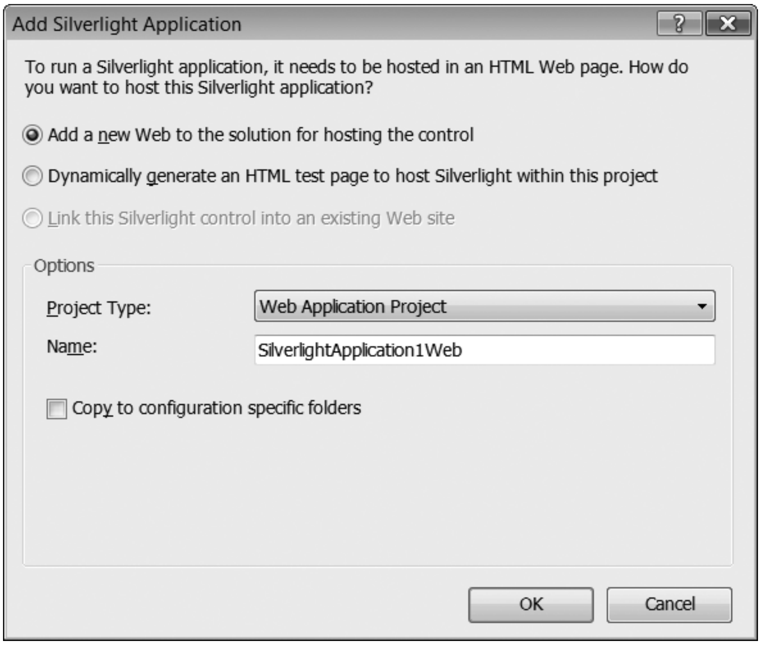
Solution Name: SilverlightApplication1

[✓] Create directory for solution

[ ] Add to Source Control

Figure 1-1. *The New Project dialog in Visual Studio 2008*

After you click OK, the next dialog allows you to create a web site/web application project that hosts the Silverlight application (see Figure 1-2).


Add Silverlight Application

To run a Silverlight application, it needs to be hosted in an HTML Web page. How do you want to host this Silverlight application?

(●) Add a new Web to the solution for hosting the control

( ) Dynamically generate an HTML test page to host Silverlight within this project

( ) Link this Silverlight control into an existing Web site

Options

Project Type: Web Application Project

Name: SilverlightApplication1Web

☐ Copy to configuration specific folders

Figure 1-2. *The Add Silverlight Application dialog in Visual Studio 2008*

For the purpose of the examples in this book, it does not matter if you use a web site or a web application project; however, web application projects are better for eventual deployment since they contain a project file suitable for MSBuild.

Click OK, and the Solution Explorer will show two projects: the Silverlight application (SilverlightApplication1) and the web site supporting it (SilverlightApplication1_Web). If you now build the application, the Silverlight application is built to a XAP file that is automatically copied to the `ClientBin` folder within the web site. This XAP file contains the Silverlight application and will be downloaded by the client when it visits the web site.

If you now start the development server in Visual Studio (by pressing F5 or Ctrl+F5), you will see the Silverlight application start. If, however, you create a new web site in IIS, point the document root to SilverlightApplication1_Web, and navigate to this site, you will get a 404 error when trying to load the Silverlight application in your browser. What's going on? IIS must know about the new file extension `.xap`. You accomplish this by adding a new MIME type to either the root of IIS or to the specific web site you created. The file extension is `.xap` and the MIME type is `application/x-silverlight-app`.

Now let's take a look at Expression Blend, a tool used to lay out user interface controls and create animations in WPF and Silverlight. Without closing Visual Studio, start Blend, go to File ➤ Open ➤ Project/Solution, and navigate to the solution file created in Visual Studio (in `C:\book\examples\SilverlightApplication1` if you used the same directory structure).

The top-right portion of Blend is devoted to managing the project files (like the Solution Explorer in Visual Studio); properties for various user interface elements; and resources, which include style templates, and animation storyboards, stored in XAML. Double-click Page.xaml to open this XAML page in the designer (see Figure 1-3).

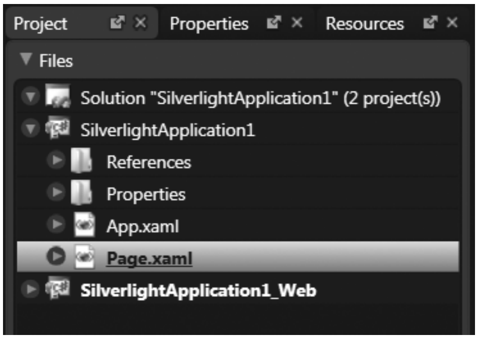


Figure 1-3. *The Project Property pane in Expression Blend*

Along the left side of the Blend screen is the toolbox. This provides access to both layout and input controls, and several tools used to modify the user interface, such as a paint bucket and a transform tool for brushes. Hold down the left mouse button when selecting any icon with a white triangle in the lower-right-hand corner and more tools will expand from it. Figure 1-4 shows an example when clicking the Button icon (which looks like a mouse cursor hovering over a rounded rectangle).

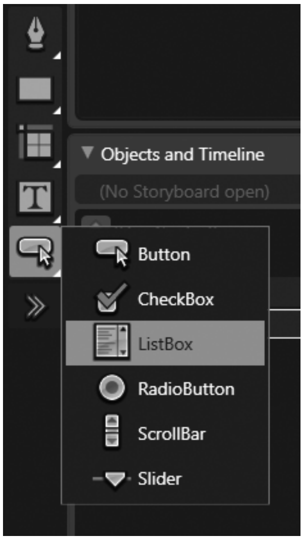


Figure 1-4. *The control toolbox in Expression Blend*

The Objects and Timeline area to the immediate right of the toolbox provides a place to create and manage animation storyboards, but more importantly for us right now, it shows the object hierarchy in XAML. After creating our application, we see [UserControl] and

LayoutRoot. Click [UserControl] to highlight it and then click Properties in the top-right portion of the screen. The control with the gray highlight is the control that shows up in the Properties pane (see Figure 1-5).

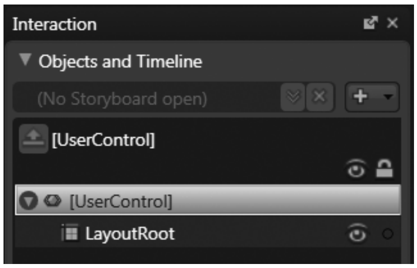
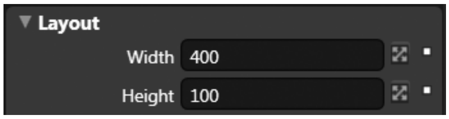


Figure 1-5. *The Objects and Timeline pane in Expression Blend*

Go to the Properties pane and set the width and height of the UserControl to 400 and 100, respectively, as shown in Figure 1-6.



Width 400
Height 100

Figure 1-6. *The size properties for a control in Expression Blend*

You can also click XAML or Split along the right side of the design surface and view and edit the XAML directly. However, as interfaces get more complex, Blend becomes an invaluable design tool for working with the XAML indirectly. Hand-editing XAML should generally be used for tweaking some XAML instead of creating full-blown user interfaces.

Next, right-click LayoutRoot in the Objects and Timeline pane and select Delete. This removes the default Grid layout control. While you can go to the toolbox and select the Canvas control (it's in the group four controls up from the bottom), let's view the XAML and create a Canvas control by hand. Click Split alongside the design surface to see the design surface simultaneously with the XAML. Edit the XAML to look like the following (reintroducing the `close` tag to the UserControl and dropping in the `Canvas` tag):

```
<UserControl
   xmlns="http://schemas.microsoft.com/client/2007"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   x:Class="SilverlightApplication1.Page"
   Width="400" Height="100">
   <Canvas Height="Auto" Width="Auto" Background="White"/>
</UserControl>
```

Now go to the toolbox and select the TextBlock control, as shown in Figure 1-7.

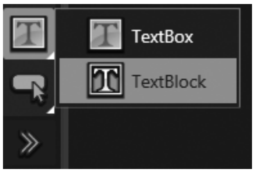


Figure 1-7. *Choosing the TextBlock control from the toolbox*

This control is used to place text on a user interface, much like a label in Windows Forms or ASP.NET. Click the design surface and hold the mouse button down, and then drag right and down to create a rectangle describing the area for the TextBlock. Now the TextBlock should appear as a child of the Canvas in the Objects and Timeline pane. Make sure the TextBlock is selected, and go to Properties.

If you've read even just one other programming book, you know what's coming next. Scroll down the properties until you see the Common Properties area, and set the text to "Hello World!" as shown in Figure 1-8.

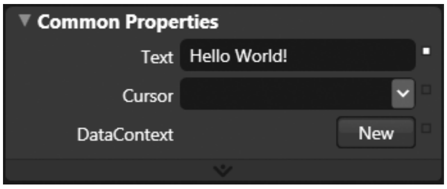

Common Properties
Text: Hello World!
Cursor:
DataContext: New

Figure 1-8. *Setting the Text property of a TextBlock in Expression Blend*

If you now switch back to Visual Studio, it will ask to reload `Page.xaml`. Go ahead and reload. Press F6 to build the application and then Ctrl+F5 to start the application without debugging. You should see something similar to Figure 1-9 in your browser.

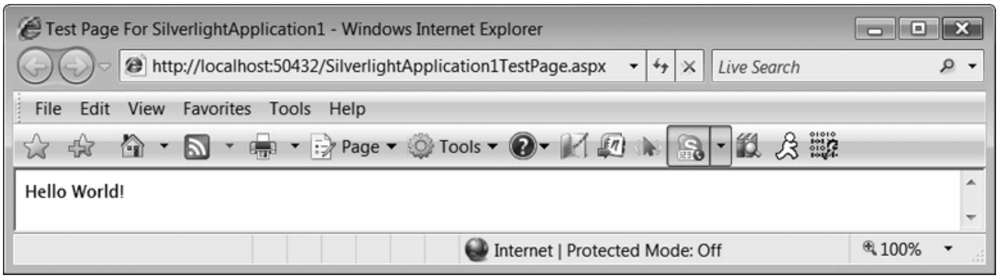

Test Page For SilverlightApplication1 - Windows Internet Explorer
http://localhost:50432/SilverlightApplication1TestPage.aspx
Hello World!
Internet | Protected Mode: Off

Figure 1-9. *The Hello World application as viewed in Internet Explorer 7*

Congratulations, you have now created your first Silverlight application using both Expression Blend and Visual Studio!

Summary

This chapter began with a discussion of Silverlight and its major competitors. Next, it covered how to create a new Silverlight application in Visual Studio with a supporting web site, how to modify the user interface in Expression Blend, and finally, how to build and execute an application in Visual Studio. The next stop on our journey through practical Silverlight development takes us to XAML. Many of the core concepts needed to understand how Silverlight works are covered in the next chapter, including markup extensions, dependency properties, and previews of features such as data binding and styling applications.