# Accelerated Silverlight 2

Jeff Scanlon

Apress®

**Accelerated Silverlight 2**

**Copyright © 2008 by Jeff Scanlon**

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

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

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

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.

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

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

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.

*To Corey Chang, for showing me just how much patience can pay off.*

# Contents at a Glance

About the Author . . . . . xiii
About the Technical Reviewer . . . . . xv
Acknowledgments . . . . . xvii
Introduction . . . . . xix

**CHAPTER 1** Introducing Silverlight . . . . . 1
**CHAPTER 2** Getting to Know XAML . . . . . 13
**CHAPTER 3** Creating User Interfaces . . . . . 37
**CHAPTER 4** Network Communication . . . . . 71
**CHAPTER 5** Working with Data . . . . . 97
**CHAPTER 6** Working with Media . . . . . 117
**CHAPTER 7** Extending the User Interface . . . . . 141
**CHAPTER 8** Styling and Templating . . . . . 163
**CHAPTER 9** Animation . . . . . 183
**CHAPTER 10** Dynamic Languages and the Browser . . . . . 207
**CHAPTER 11** Security . . . . . 229
**CHAPTER 12** Testing and Debugging . . . . . 251
**CHAPTER 13** Packaging and Deploying Silverlight Applications . . . . . 271
**CHAPTER 14** Advanced Topics . . . . . 293
**CHAPTER 15** Case Study: Campus Explorer . . . . . 315

**INDEX** . . . . . 333

# Contents

About the Author . . . . . xiii
About the Technical Reviewer . . . . . xv
Acknowledgments . . . . . xvii
Introduction . . . . . xix

**CHAPTER 1 Introducing Silverlight** . . . . . 1

Cross-Platform Frameworks . . . . . 2
Qt . . . . . 2
The Java Platform . . . . . 2
Flash/Flex . . . . . 3
Silverlight . . . . . 3
The History of Silverlight . . . . . 3
Creating Your First Application . . . . . 5
Summary . . . . . 11

**CHAPTER 2 Getting to Know XAML** . . . . . 13

Introducing XAML . . . . . 13
Namespaces . . . . . 18
Dependency Property System . . . . . 18
Type Converters . . . . . 23
Markup Extensions . . . . . 24
More About Silverlight Applications . . . . . 30
Events in Silverlight . . . . . 32
Summary . . . . . 36

**CHAPTER 3 Creating User Interfaces** . . . . . 37

Building Blocks . . . . . 37
DependencyObject . . . . . 39
UIElement . . . . . 39
FrameworkElement . . . . . 41

Positioning Objects on Screen . . . 43
Canvas . . . 44
StackPanel . . . 44
Grid . . . 45
Customizing Silverlight Controls . . . 48
ContentControl . . . 48
Border . . . 49
The Button Controls . . . 50
TextBlock . . . 53
TextBox . . . 55
ItemsControl . . . 57
Popup . . . 58
ToolTipService . . . 59
RangeBase . . . 60
ScrollViewer . . . 62
Incorporating SDK Controls . . . 64
GridSplitter . . . 64
Calendar and DatePicker . . . 65
TabControl . . . 67
Implementing Navigation . . . 68
Summary . . . 70

**CHAPTER 4 Network Communication** . . . 71

Enabling Cross-Domain Communication . . . 71
Using Services . . . 74
Creating and Using a Client Service Proxy . . . 76
Communicating Over HTTP Directly . . . 81
The HttpWebRequest Class . . . 83
Raw Network Communication . . . 86
The Socket Class . . . 86
The SocketAsyncEventArgs Class . . . 87
Using the Socket Class . . . 88
Considerations for Using Networking . . . 94
Summary . . . 95

**CHAPTER 5 Working with Data** . . . 97

Displaying Data . . . 97
Data Binding . . . 97
Introducing the DataGrid . . . 105

Processing Data . . . . . 106
Parsing XML . . . . . 107
Serializing XML . . . . . 108
Using LINQ . . . . . 109
Saving State on the Client . . . . . 110
Summary . . . . . 115

CHAPTER 6 **Working with Media** . . . . . 117

Images . . . . . 117
Multiscale Images (Deep Zoom) . . . . . 121
Media (Video and Audio) . . . . . 125
Timeline Markers . . . . . 131
Silverlight Streaming . . . . . 132
Preparing an Application . . . . . 134
Packaging Images and Media . . . . . 138
Summary . . . . . 140

CHAPTER 7 **Extending the User Interface** . . . . . 141

2D Graphics . . . . . 141
Using Geometries . . . . . 142
Using Shapes . . . . . 147
Transforms . . . . . 151
Translation . . . . . 151
Rotation . . . . . 152
Skewing . . . . . 153
Scaling . . . . . 153
Arbitrary Linear Transforms . . . . . 154
Combining Multiple Transformations . . . . . 156
Brushes . . . . . 157
The SolidColorBrush . . . . . 158
The Tile Brushes . . . . . 158
The Gradient Brushes . . . . . 159
Summary . . . . . 162

**CHAPTER 8 Styling and Templating** . . . . . 163

Using Styles . . . . . 163
Using Control Templates . . . . . 169
Creating a Control Template . . . . . 169
Control Templates for Other Controls . . . . . 178
Developing a Templated Control . . . . . 179
Summary . . . . . 181

**CHAPTER 9 Animation** . . . . . 183

Animation Basics . . . . . 183
Timelines . . . . . 184
Storyboards and Animation . . . . . 189
From/To/By Animations . . . . . 190
Keyframe Animations . . . . . 197
Animating with Expression Blend . . . . . 201
Summary . . . . . 205

**CHAPTER 10 Dynamic Languages and the Browser** . . . . . 207

Introducing Dynamic Languages . . . . . 207
The DynamicApplication Class . . . . . 209
Creating a Dynamic Language Application . . . . . 209
Executing a Dynamic Language Application . . . . . 211
Developing with Dynamic Languages . . . . . 212
Interoperating with the Browser . . . . . 216
Summary . . . . . 227

**CHAPTER 11 Security** . . . . . 229

Security in the CoreCLR . . . . . 229
Application-Level Security . . . . . 232
Securing Information in Transit . . . . . 232
Securing Information with Cryptography . . . . . 232
Division of Responsibility . . . . . 244
Summary . . . . . 249

**CHAPTER 12 Testing and Debugging** . . . . . 251

Testing . . . . . 251
Unit Testing . . . . . 251
Automated User Interface Testing . . . . . 258
Debugging . . . . . 260
The Debugging Process . . . . . 260
Conditional Compilation . . . . . 261
Debugging with Visual Studio . . . . . 263
Handling Unhandled Exceptions . . . . . 267
Summary . . . . . 270

**CHAPTER 13 Packaging and Deploying Silverlight Applications** . . . . . 271

Client Considerations . . . . . 271
Silverlight Deployment Packages . . . . . 275
Hosting Silverlight on a Web Page . . . . . 278
Silverlight Versioning . . . . . 280
Custom Initialization Parameters . . . . . 281
Resources . . . . . 282
Silverlight and the Build Process . . . . . 284
Silverlight Assemblies . . . . . 290
Summary . . . . . 291

**CHAPTER 14 Advanced Topics** . . . . . 293

Threading . . . . . 293
The Thread Class . . . . . 295
Creating and Managing Threads . . . . . 297
The Dispatcher . . . . . 299
The BackgroundWorker Class . . . . . 300
Working with Shared Data . . . . . 304
Using Timers . . . . . 308
The DispatcherTimer . . . . . 309
The System.Threading Timer . . . . . 309
Dynamically Loading Applications . . . . . 311
Summary . . . . . 313

**CHAPTER 15 Case Study: Campus Explorer** . . . . . 315

Application Features . . . . . 316
Design of the Application . . . . . 316
User Interface Design . . . . . 316
Data Representation . . . . . 317
Packaging of the Application . . . . . 320
Application Implementation . . . . . 320
Helper Methods . . . . . 320
XAML Organization . . . . . 321
MainPage . . . . . 321
Map.xaml . . . . . 323
Summary . . . . . 331

**INDEX** . . . . . 333

# About the Author

**JEFF SCANLON** is an independent Microsoft consultant with extensive experience developing software using a variety of technologies. He designs and implements software across all layers with a focus on web-based applications. Jeff has lead developer training sessions on software engineering practices and introductions to new technologies at nearly every company he has worked with. He is the author of *Professional Java Programming with JDK 5*, and has written a number of articles on .NET for *Software Development* magazine. He holds several Microsoft certifications and a bachelor's degree in computer science from George Mason University.

# About the Technical Reviewer

**FABIO CLAUDIO FERRACCHIATI** is a senior consultant and a senior analyst/developer. He works for Brain Force (`www.brainforce.com/`) in its Italian branch (`www.brainforce.it/`). He is a Microsoft Certified Solution Developer for .NET, a Microsoft Certified Application Developer for .NET, and a Microsoft Certified Professional.

Fabio is a prolific author and technical reviewer. Over the past ten years, he has written articles for Italian and international magazines, and has coauthored more than ten books on a variety of computer-related topics. You can read his LINQ blog at `www.ferracchiati.com/`.

# Acknowledgments

Thank you for deciding to buy this book. Without readers, a book is nothing more than an actor on an empty stage. I hope this book successfully helps you gain proficiency with Silverlight.

I had several publishers in mind when I first decided to pitch a book on Silverlight. After sending out several e-mails, I received a response from Ewan Buckingham in a mere matter of hours. This fast response encouraged me, since Apress was my top choice. Ewan impressed me from the beginning by seeing where this book could fit and helping get the project off the ground. He also saw this book through to the end, despite an issue during the middle of the project. I can't thank him enough for bringing me into the Apress author family and sticking with me on this project.

There are many other people at Apress that helped create this book and each deserves a great deal of appreciation. I couldn't have asked for a better project manager than Richard Dal Porto. Despite deadline slips and repeated schedule revisions that I'm sure would stress out any PM, Richard gave me plenty of space to work without constantly asking for status reports. Ewan and Richard trusted me to get the work done and I'm indebted to them for seeing this project through to the end.

Damon Larson was the copy editor for this book. Without his feedback, many ideas in this book would be unclear and the structure of the language might seem inconsistent. I think I gave him a lot of work to do, but the book is much better for his careful combings-through of my writing.

Jill Ellis worked as production editor. She helped the book through its final stages, ensuring the book looked how it should before it was sent to the printer. She was helpful in putting the extra polish on the book that will be evident when you read its pages.

Other people from Apress who helped along the way (in no particular order) are Joohn Choe, Ami Knox, Linda Seifert, Nancy Wright, and Dominic Shakeshaft. Compared to the hard work of all the people at Apress, I feel like my job was minor!

Fabio Claudio Ferracchiati, technical editor, was also of vital importance to this book. He offered a significant amount of feedback that helped improve the chapters. Based on his feedback, some sections were completely rewritten to improve structure and quality of information.

Any imperfections in this book are solely my responsibility, not Apress' or my technical editor's. I encourage you to leave feedback online or contact me directly so they can be addressed if necessary.

Claudia Holland from George Mason University helped ensure I had the required permissions to use the name of the university and video/pictures of the Fairfax campus.

I would also like to thank the people in my life who helped make this book possible. My parents deserve credit for letting me pursue what I want in life, regardless of the craziness of some pursuits. My closest friends deserve endless appreciation for their support: Corey Chang, for inspiring me and supporting me despite being the only person busier than I am; Phill Alexander, for his continual support and help with random snippets of language, despite a busy schedule as a PhD student; and Dave Nelson, for getting me into this intense, incredible field all those years ago.

# Introduction

Silverlight is an exciting technology. There are technological aspects to Silverlight that convince me it will be a viable platform in the years ahead. I think Silverlight and its Linux implementation, Moonlight, will prove interesting to all developers, regardless of whether they have .NET experience. Silverlight brings the advantages of XAML and a CLR, and a set of useful platform libraries, to operating systems other than Windows. Someone once commented that Silverlight is where I placed my bet, and this is definitely true. I've upped the ante by investing the time and energy to write this book. *Accelerated Silverlight 2* aims to get you up to speed as efficiently as possible on Silverlight, and I hope you find what you're looking for within its pages. If you have any questions or problems, please get in touch with me using the contact information at the end of the Introduction, and I'll help. I also have a site specifically devoted to this book, `www.acceleratedsilverlight.net/`, which provides a place for readers to get questions answered and extend some of the topics discussed in this book.

## Who This Book Is For

This book assumes you have a reasonable degree of familiarity with .NET, such as understanding what assemblies are and how to develop on the .NET platform using C#. The goal of this book is to get you up to speed on Silverlight as efficiently as possible. Although Windows Presentation Foundation (WPF) also uses XAML (which you'll learn about in Chapter 2, in case you're unfamiliar with this term), you do not need to be familiar with WPF.

## How This Book Is Structured

This book covers a significant amount of Silverlight, from the new Extensible Application Markup Language (XAML), to creating user interfaces, to the building of a real world–style application. The following subsections more specifically detail what is covered in each chapter.

## Chapter 1, "Introducing Silverlight"

This chapter discusses some of the background of cross-platform applications to help you understand where Silverlight fits into the overall technological picture. There may not be much history to Silverlight, but it did start as a version 1.0 product that featured a basic XAML parser (but no managed execution engine). This chapter concludes with using Visual Studio 2008 and Expression Blend to create your first Silverlight application.

## Chapter 2, "Getting to Know XAML"

XAML is a new declarative language. It provides an easy way to create and configure object hierarchies and relationships in markup. This chapter introduces important concepts, such as markup extensions to support handling resources and data binding, type converters for interpreting property values, dependency properties, attached properties, events, and other important aspects of XAML and Silverlight.

## Chapter 3, "Creating User Interfaces"

Silverlight provides important controls for organizing user interfaces, displaying information, and receiving user input. After discussing the important aspects of the Silverlight object hierarchy, we get right into creating user interfaces. The major layout controls are explored—the Canvas for absolute positioning, the StackPanel for organizing controls horizontally or vertically, and the Grid for placing controls in HTML-like tables. Next, all the standard user interface controls are covered, including those for creating text entry boxes, check boxes, radio buttons, and list boxes.

## Chapter 4, "Network Communication"

An online application that does not talk to other systems (or even back to its hosting server) is a rare case, so Silverlight must provide ways to interact with other systems. Unsurprisingly, Silverlight provides functionality to invoke web services and download data (such as ZIP files) from a web server. However, it might surprise you that Silverlight includes support for raw network communication, though it is subject to security restrictions.

## Chapter 5, "Working with Data"

Communicating over the network is important for getting data—but once you have data, what do you do with it? This chapter details how to connect data from a data source to the user interface using the data binding architecture. Data can be stored in a collection in the code-behind or in XML. Silverlight provides the ability to use LINQ expressions (introduced in .NET 3.5—but don't worry, Silverlight is still completely separate from the

.NET Framework) in the code-behind, as well as support for both reading and writing XML files and serialization to and from objects. This chapter concludes with a look at how to save state on the client, mainly through the use of isolated storage—a private, secure area on disk for Silverlight applications.

## Chapter 6, "Working with Media"

Silverlight makes it easy to create rich user interfaces involving images, audio, and video. This chapter details how to access and utilize these media elements. Silverlight can be used to create sites that manage video—such as YouTube (`www.youtube.com/`)—or sophisticated image-browsing sites like Hard Rock Memorabilia (`http://memorabilia.hardrock.com/`). This chapter details the various media controls, including Image, MediaElement, and MultiScaleImage (also known as Deep Zoom, the MultiScaleImage control was used to create the Hard Rock Memorabilia site). The chapter concludes with a look at Silverlight Streaming, a service Microsoft provides to host both Silverlight applications and videos for streaming.

## Chapter 7, "Extending the User Interface"

Although Chapter 2 detailed many controls useful for building Silverlight applications, it only showed one aspect of Silverlight's support for building user interfaces. This chapter returns to building user interfaces. Silverlight has support for 2D graphics, such as lines and ellipses, and even complex geometrical shapes. Almost anything that can be drawn on a user interface (such as 2D graphics or controls) can be transformed (e.g., rotated or scaled down). These transforms are discussed along with performing custom transformations by using a transformation matrix. This chapter concludes with a look at the various brushes provided by Silverlight, useful for painting colors, images, video, or even color gradients onto foregrounds or backgrounds of elements in a user interface.

## Chapter 8, "Styling and Templating"

Silverlight provides the ability to centrally manage styles that control the appearance of elements on a user interface, such as those for font face, font size, and color. It also supports the ability to completely replace the visual representation of controls using control templates. Both of these mechanisms are explored in this chapter.

## Chapter 9, "Animation"

Animation provides the ability to change the properties of user interface elements over time. This chapter discusses the support Silverlight provides for animation, beginning with an explanation of a timeline and continuing with an exploration of storyboards and the different ways to animate elements of a user interface. The chapter concludes with a look at animating using Expression Blend, an invaluable tool for easily developing and previewing animation.

## Chapter 10, "Dynamic Languages and the Browser"

A big aspect of Silverlight that is currently not officially available in .NET on Windows is the Dynamic Language Runtime (DLR). This enables the smooth execution of dynamic languages such as IronPython, IronRuby, and Managed JScript within Silverlight. After showing how to utilize dynamic languages in Silverlight applications, this chapter switches gears to the support Silverlight provides for interoperating with the browser. Silverlight provides the ability to send and receive data from the hosting browser, including invoking JScript and accessing the DOM.

## Chapter 11, "Security"

Silverlight can interact with the host operating system—for example, isolated storage ultimately writes files to disk. This direct access is impossible from your application code because all application code is considered unsafe. This forms the core of the security model for executable code in Silverlight. Beyond the security of executable code, there are other aspects at an application level that contribute to sound security in Silverlight applications. These aspects include authentication/authorization to control access, communicating over SSL, and using cryptography to protect sensitive data. This chapter explores all of these, along with how to design a Silverlight application with security in mind.

## Chapter 12, "Testing and Debugging"

Applications must be tested to prove, as best as possible, that they are bug free and work as designed. This chapter primarily focuses on unit testing—testing Silverlight applications from the perspective of a developer. A strong set of unit tests can prove a useful part of the build and verification process. When bugs are found, during development or from testing, the root cause must be discovered. This is where debugging proves useful. Debugging is more than simply attaching a debugger to a Silverlight application and tracing execution. Both proactive and reactive debugging measures are discussed.

### Chapter 13, "Packaging and Deploying Silverlight Applications"

Silverlight is a client-side technology. A Silverlight application can be placed on any web server (e.g., IIS, Apache, etc.); however, there are some benefits to deploying Silverlight on IIS 7 (primarily in the handling of video). This chapter will discuss how Silverlight applications are packaged and deployed on web servers, how they are embedded in HTML/ASPX pages, and also what is necessary to support building Silverlight applications using MSBuild.

### Chapter 14, "Advanced Topics"

One of the most frustrating things for users of an application is a frozen user interface. Long-running operations should never occur on the user interface thread, and you should be well aware of this if you've done any Windows Forms development. Silverlight supports several techniques to improve responsiveness of user interfaces, including asynchronous communication and threading. This chapter explores techniques to create responsive user interfaces by looking at both explicit and implicit ways of leveraging multiple threads. Silverlight also provides several timer-related classes useful for certain periodic tasks, such as providing a time signature for a video that is playing.

### Chapter 15, "Case Study: Campus Explorer"

The book concludes with the design and development of an example application that uses many aspects of Silverlight. The application provides an interactive map of a university campus and displays images/video linked to buildings on campus to give visitors to the application a good idea of what the campus is like. The key features of this application include images, video, control templating and styling, data binding, and various controls.

## Contacting the Author

You can contact the author by visiting his site at `www.artofcoding.net/`, or via this book's site, at `www.acceleratedsilverlight.net/`. Comments on this book can be sent directly to the author at `feedback@acceleratedsilverlight.net`.

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).

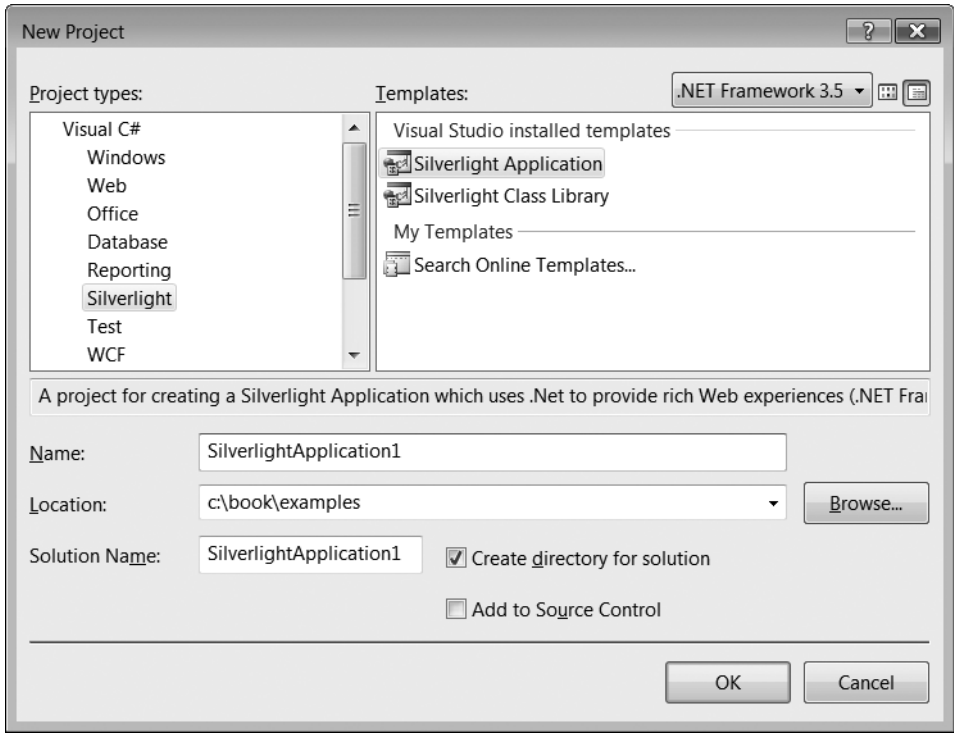


**Figure 1-1.** *The New Project dialog in Visual Studio 2008*

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).

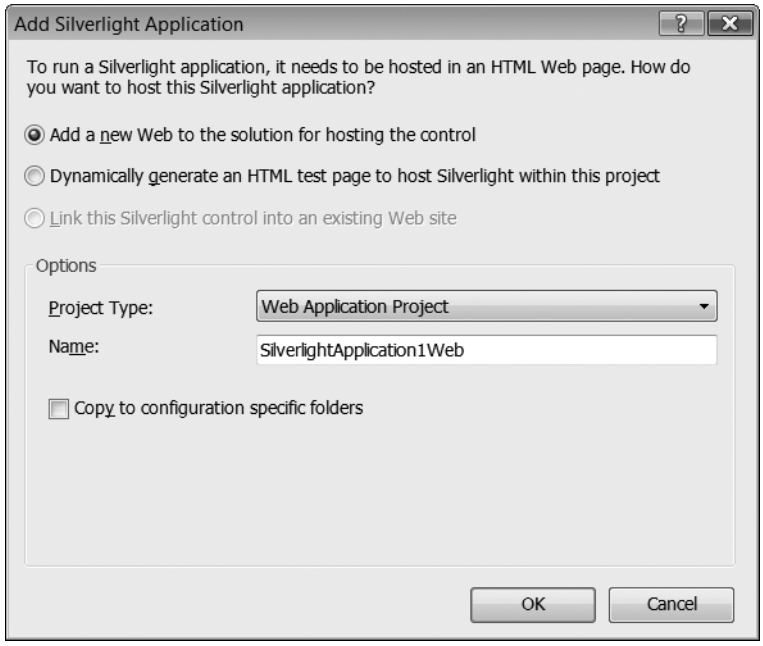


**Figure 1-2.** *The Add Silverlight Application dialog in Visual Studio 2008*

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).

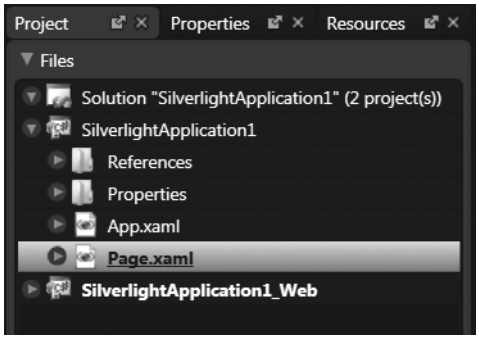


**Figure 1-3.** *The Project Property pane in Expression Blend*

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).

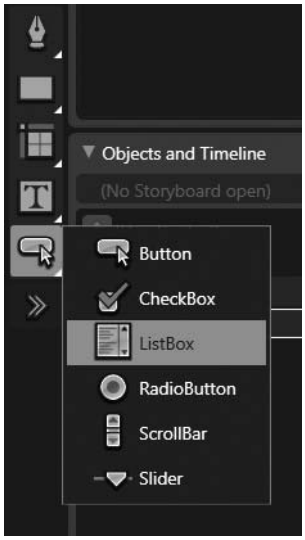


**Figure 1-4.** *The control toolbox in Expression Blend*

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).

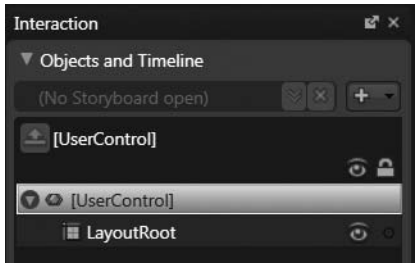


**Figure 1-5.** *The Objects and Timeline pane in Expression Blend*

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.

**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.

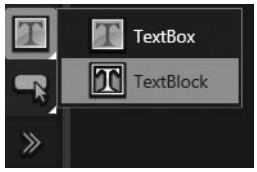


**Figure 1-7.** *Choosing the TextBlock control from the toolbox*

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.

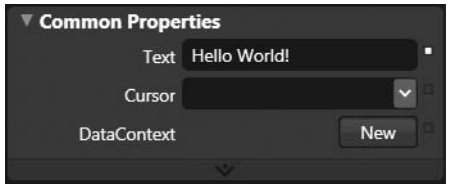


**Figure 1-8.** *Setting the Text property of a TextBlock in Expression Blend*

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.

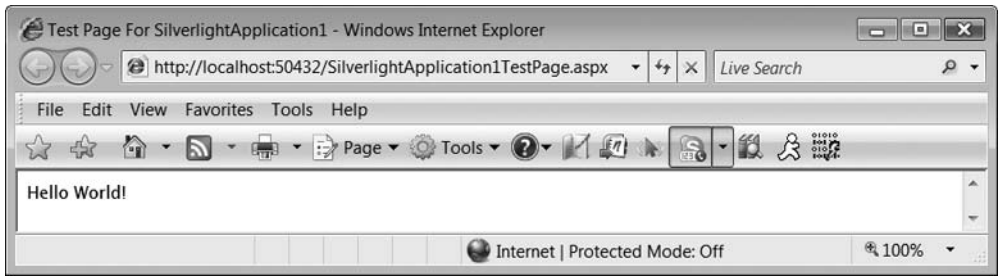


**Figure 1-9.** *The Hello World application as viewed in Internet Explorer 7*

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.