

Silverlight 5

IN ACTION

Revised Edition of
Silverlight 4 in Action

Pete Brown



MANNING



MEAP Edition
Manning Early Access Program
Silverlight 5 in Action version 4

Copyright 2011 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

©Manning Publications Co. Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=786>

Table of Contents

Part 1: Core Silverlight

- 1. Introducing Silverlight**
- 2. XAML and the Property System**
- 3. The Application Model and the Plug-in**
- 4. Working with HTML and Browsers**
- 5. Out-of-Browser Applications**
- 6. The Security Model and Elevated Trust**

Part 2: Creating the User Interface

- 7. Rendering, Layout, and Transforming**
- 8. Panels**
- 9. Human Input**
- 10. Text Fundamentals**
- 11. Editing Plain and Rich Text**

12. Control Basics and UserControls

13. Animation and Behaviors

14. Styles, Templates, and Resources

15. Extensions, Converters, Custom Controls and Panels

Part 3: Working with Data and Services

16. Binding

17. Data controls: DataGrid and DataForm

18. Input Validation

19. Networking Basics

20. Working with Soap Services

21. RESTful services

22. Working with XML, JSON, RSS and ATOM

23. Duplex, Sockets, and Local Connections

Part 4: Graphics and Media

24. Graphics and Effects

25. Working with Images

26. Introduction to 3D

27. 3D Lighting, Texturing, and Animation

28. Media Basics

29. Raw Media, Webcam, and Microphone

Part 5: Making the Most of the Platform

30. Pop-ups, Windows, and Full-Screen Applications

31. Navigation

32. Working with Files and Directories

33. Printing

34. COM, Native Extensions and p-invoke

Part 6: Techniques for Complex Applications

35. Structuring and Testing with the MVVM Pattern

36. Introduction to WCF RIA Services

37. WCF RIA Endpoints, Security, Business Logic, and Decoupling

38. WCF RIA Services and MVVM

39. Debugging Your Application

40. The Install Experience and Pre-loaders

Appendix A: Database, Connection and Data Model Setup

Part 1

Core Silverlight

When learning a new technology, it's always a good idea to start at the core: the core concepts, the core features, the core technologies, and the core skills. Taking that as a given as a good practice, the first six chapters of this book will get you the grounding you need to learn more about Silverlight and make good decisions about how you write your applications.

In the first chapter, we'll learn what Silverlight is and how it fits into the developer platforms offered by Microsoft. We'll then spend the remainder of the chapter building out our very first Silverlight application. Believe me, it won't be a boring old "Hello World".

XAML is the XML-based approach to defining the user interface for Silverlight, WPF, Windows Phone, and Windows 8 XAML applications. We'll look deeply into XAML and understand how it works, how namespaces are handled, how objects map into XAML, and much more.

Silverlight differs substantially from other client and web technologies when it comes to its application model and how the plug-in integrates with the system. It's essential and interesting to learn how all the pieces fit together, so we'll cover that next.

Building upon what we learned about the app model and web page plug-in, we next turn to how Silverlight can integrate with web applications. Silverlight has deep integration with the browser and HTML while running on a web page. We'll look at how Silverlight can interact with the web page on which it resides, even to the point of manipulating the page's DOM from within Silverlight.

From there, we'll turn to an increasingly popular model for Silverlight: the Out of Browser application. Application developers, especially those developing business applications, have used Silverlight to create rich desktop applications which install from the web, but otherwise look and behave like any other native application. It's the best of both worlds.

Finally, we'll wrap up part 1 with a look at the security model used by Silverlight, including how it determines whether or not code is safe to run, and how to get elevated permissions to escape from the sandbox.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=XYZ>

First, let's learn the basics of Silverlight and build our first Silverlight application. Come with me, I think you'll enjoy the ride.

1

Introducing Silverlight

This chapter covers

- Silverlight, the web, and WPF
- The best applications for Silverlight
- Getting started with Silverlight
- Changes in Silverlight since the previous edition of this book
- Building your first Silverlight “Hello World!” application

First of all, let me thank you for starting at chapter 1. I’m one of those people who tend to read magazines backwards and skim technology books, so I appreciate it when someone reads a book’s chapters in order. Then again, maybe you read this book backwards as well. In that case, you’ll find the “Hello World!” walkthrough in this chapter to be a refreshingly simple take on building Silverlight applications unencumbered with patterns such as Model View ViewModel (MVVM), words such as `DependencyProperty`, and technologies such as Windows Communication Foundation (WCF) Rich Internet Application (RIA) Services. For the rest of you, don’t worry—we’ll cover each of those throughout the rest of the book, steadily building our Silverlight skills as we go.

Since you’ve picked up a Silverlight book, you would probably like to know what Silverlight is. Luckily, I’m horrible at marketing, so I’ll put it simply: Silverlight is a cross-platform .NET runtime, cross-browser plug-in, and a set of Windows-based developer tools for building RIAs. At its heart, Silverlight is an implementation of the concepts and standards from Windows Presentation Foundation (WPF) such as binding, the property system, and Extensible Application Markup Language (XAML) in a cross-platform version of the .NET Common Language Runtime (CLR) and libraries.

There. I think that paragraph managed to get all of the acronyms defined for the rest of the book. Then again, this is a Microsoft technology, so expect more acronyms before we're through.

Silverlight runs on Windows and Mac through Microsoft-supplied plug-ins as well as on Linux through the Moonlight project. It is the primary development platform for Windows Phone. We've seen demos of it running on set-top boxes connected to televisions and serving up ads and content on the Xbox. Put simply, short of ASP.NET, Silverlight is the broadest reaching technology ever produced by Microsoft.

Silverlight applications work on the web as well as on the client. You can create virtually any type of application in Silverlight, from web content, to widgets, to media players to full-blown client applications.

By the end of this chapter, you'll have created your first functional Silverlight application, a Twitter search client with nice visualization of the tweets. You'll be introduced to XAML, binding, networking, controls, and much more – even a little LINQ to XML. These are all topics we'll dive deeply into in the rest of the book. First, we'll learn a bit about where Silverlight fits in to the developer ecosystem. This is a big question for many, and needs to be resolved up front. We'll follow that up with a look at the types of applications Silverlight is well-suited to. After that, we'll check out the features and capabilities that have been added since the first edition of this book.

1.1 A Silverlight Primer

Silverlight's place in the world has evolved since its original inception. Market forces and customer preferences have both led to Silverlight moving more towards private web and line-of-business applications as opposed to broad-reach public web sites and applications. Clearly both Microsoft's customers and partners have moved to a standards-based public web. Nevertheless, there are great ways and reasons to use Silverlight on the public web, especially while the HTML standards are still catching up to what plug-ins like Silverlight can accomplish. Of course, on the desktop and in the enterprise, Silverlight is as viable as ever.

In this section, we'll introduce Silverlight in context, looking at how it fits into the developer stack both on the web and on the desktop. We'll then look at some of the different types of applications Silverlight is well suited for.

Silverlight got its start as a web page plug-in, so that's where we'll start as well.

1.1.1 Silverlight and the web

Silverlight sits in that interesting place between desktop applications and browser applications. In many ways, when in the browser, it's like a little traditional desktop application embedded in HTML. Of course, the same can be said of many JavaScript applications, themselves modeled on the code-on-the-client desktop application paradigm.

Great frameworks such as jQuery and the oft-confused HTML 5 and CSS3 further muddy the waters. Where's Silverlight's place on the web? Why should you use Silverlight instead of these other technologies?

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

I'll give you a few reasons:

- Silverlight has top-tier media support, including digital rights management (DRM), far more advanced than the proposed HTML 5 standards. It will be a while before HTML 5 catches up across all browsers
- Silverlight is a no-brainer if you're already a .NET developer looking to expand to other platforms. It's simply easier to develop with and in.
- Silverlight has best-of-class development and debugging tools

Don't get me wrong; I think HTML 5 and CSS3 and even JavaScript are a great thing for the web—exciting and capable. Having said that, Silverlight has more advanced authoring tools, faster execution, and more capabilities than HTML 5 currently has. HTML 5 will continue to raise the floor, driving the quality and experience up across the spectra of platforms and developer tools. On the public web, HTML 5 will eventually catch up to the capabilities of Silverlight for the majority of typical web scenarios and plug-ins will not be needed. For many public sites and applications, we've already reached that point due to the proliferation of plugin-hostile tablets.

I don't personally think that the code-on-the-client application development approach is going to completely disappear. Though doom has been forecast for many major development approaches over the years, few have actually declined when another rose in popularity. Silverlight and HTML 5 will just provide more options for how to implement the solution you need in the most optimal way, using the tools you're comfortable with and the skills you already have. The balance of what code you put in each technology will be the real point of debate for development teams. In addition, the skills you learn as a Silverlight developer will port quite nicely to Windows 8 XAML, once you decide to adopt that operating system.

Remember that HTML/CSS/JavaScript and Silverlight aren't mutually exclusive. Silverlight applications can happily coexist on a page with JavaScript applications, each complementing the other with features that play to their strengths.

Silverlight is far more than a web technology. Though it can live on a web page, it's also common to have out-of-browser Silverlight applications, either connected to services or simply using resources on the client. In those instances, you may wonder when to use WPF and when to use Silverlight.

1.1.2 Silverlight and WPF

Silverlight and WPF were born of the same ideas. WPF came first and broke the ground required to make XAML a UI-friendly markup language. WPF also introduced us to dependency properties and binding, storyboard-based animation, and subpixel--rendered vector UI. WPF was, and continues to be, an amazingly rich platform for developing both traditional Windows and NUI (Natural User Interface) applications.

But WPF is large and complex. It's also deeply rooted in Windows, with no good way to substitute alternate stacks for those it relies on. WPF also relies on the rather outdated and web-unfriendly code access security model for application security. So, when Microsoft

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

decided to enter the RIA space with a CLR-based vector UI technology, they took the concepts and some of the code from WPF and reimplemented them in a smaller, tighter, and more platform-independent way.

Silverlight primarily is a subset of WPF and .NET 4 with some significant additions. Some of the additions, such as the Visual State Manager, have been migrated back from Silverlight into WPF. Others, such as Deep Zoom, Media Stream Source, and the webcam and microphone APIs, are Silverlight-only features, unlikely to be ported to WPF. Others like XNA sound and 3D APIs are from entirely different technologies. Ignoring alternative solutions to the same problems, figure 1.1 shows this relationship using our friend, the Venn diagram.

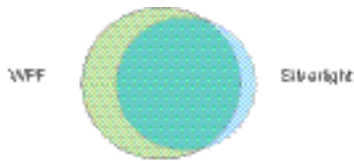


Figure 1.1 Silverlight is primarily a subset of WPF with asome extras added. Ignoring alternative solutions to the same problems, the places where WPF differs most are in the integration with the Windows OS and the access to the full .NET framework.

I recommend that developers new to both technologies learn Silverlight before learning WPF. In general, you'll find it easier to learn Silverlight first and then scale up to WPF, should your needs dictate. Silverlight is smaller, typically having a single approach to solving a given problem, whereas WPF may have several solutions for the same task. Though Silverlight doesn't have everything WPF has, Silverlight is an excellent, capable development platform and can cover many types of applications we would've previously written in Windows Forms, WPF, or even HTML.

1.1.3 Types of Silverlight applications

You can build just about anything you'd like using Silverlight. Of course, Silverlight is better suited for some types of applications over others. For example, though possible, you wouldn't necessarily want to build an entire website using Silverlight; there are better tools for the job. The most common types of applications where Silverlight is a good fit are media, business, and games.

Media: Silverlight excels at media. When Silverlight 1.0 was first introduced, one of the few capabilities it had was an excellent media stack. Silverlight through version 5 has built upon that to include new media capabilities such as smooth streaming, enhanced navigation with trick play, pluggable codecs using the Media Stream Source API, trick-play, support for remote controls and media keys, and even the DRM technologies required for the large content producers to adopt Silverlight.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

Silverlight's early focus on media was both helpful and hurtful. Video on the web is a great way to gain product adoption, especially when you have a capable high-def video technology.

Business: Early on, many potential Silverlight developers failed to see past the media roots and missed the rich business capabilities Silverlight provides. Starting with versions 3 and 4, Silverlight gained serious business capabilities. From simple things such as sync and async validation, to patterns such as MVVM and Prism, and entire middle-tier frameworks such as WCF RIA Services, Silverlight showed itself to be a mature platform, able to absorb the best practices from other areas and build upon them.

Games: Though business and media applications certainly are great staples, another fun application type is games. While HTML5 will likely turn out to be a more popular broad-reach gaming platform, silverlight has good support for casual games, including the ability to generate bitmaps on the fly, create sound from bits, loop audio in the background, play real-time sound effects, manipulate and render 3D scenes, and more. The community has successfully ported over physics and gaming engines to Silverlight, making it even easier to create complex casual games.

There are many other types of Silverlight applications ranging from ads, to photo viewers, to social media clients, to analogs for virtually every type of major desktop and web application. Some of those, such as desktop applications, weren't possible with Silverlight 2, the version used in the very first edition of this book. Let's take a high-level view of what has changed in that time. For readers of the Silverlight 4 edition, I'll mention some key new features in Silverlight 5 as well.

1.2 A Brief History of Silverlight

The first edition of this book was written for Silverlight 2. Silverlight 3, 4 and 5 have added an amazing number of new capabilities to the platform in all areas, from core capabilities, to device access, to the introduction of both trusted and sandboxed in and out-of-browser client applications. Silverlight has had 5 major releases in the past 4 years as shown in table 1.1

Table 1.1 Silverlight versions released for the desktop

Version	Release Date	A Few Key Features
1.0	Sept 5, 2007	Media, Basic graphics. JavaScript for code
2	Oct 14, 2008	First release with the .NET runtime. Rich media
3	July 9, 2009	Extensible media, Out-of-Browser, Shaders, GPU Acceleration
4	April 15, 2010	Webcam and Microphone, WCF RIA Services 1.0, Validation
5	Late 2011	3D, Vector Printing, OpenType Text Layout, TrickPlay, more

Table 1.1 shows just a few of the major enhancements delivered in each version of Silverlight over the last four years. In the remainder of this section, we'll look at the features, especially those new to Silverlight 4 and 5, in more detail. The advancements in Silverlight can be loosely grouped into four main areas: business and client applications, media and graphics, user interaction, and text.

1.2.1 Features for business and client applications

Back in 2008 and 2009, Silverlight 2 was just starting to gain adoption. It was a brand new technology from Microsoft (the managed code version was, anyway), one with strong competition from Flash/Flex and even Java. Though Silverlight 2 could have been used to build rich business applications, it didn't yet have the chops to be a strong contender in that space. Three versions later, that story has certainly changed. Many of the features in this section are useful in applications of all sorts; I hate to classify them under the heading of "business," but that's the largest consumer of these features.

Input validation, covered in chapter 18, was one of the biggest new features for business applications. Silverlight didn't add just validation but included support for validation through attributes, validation through exceptions, and even asynchronous validation, all of which work with the Silverlight controls. Silverlight even made it possible to completely customize the style of the validation information provided to the end-user.

Validation relies very heavily on the binding system in Silverlight. The teams added a number of new features to binding including a `DataContextChanged` event, finer control over when the data source is updated, and the ability to bind to properties added to objects at runtime. I cover all these new features in chapter 16.

One technology that builds heavily on the validation stack is WCF RIA Services (chapters 36, 37 and 38). A good bit of the validation functionality rolled into the Silverlight runtime actually came from that project. WCF RIA Services provides a way to share validation and logic between the client and server as well as a framework for validation, data access, and security, shareable between Silverlight and other clients.

WCF RIA Services builds upon the WCF stack, but it's not the only enhancement there. The Silverlight networking stack, described in chapters 19 through 23, was greatly enhanced to support in-browser and out-of-browser operation, as well as SOAP 1.2. When combined with technologies such as the WCF Web API (chapter 21), networking has gotten both more flexible and more interesting.

We've also done a lot to optimize networking performance and reduce latency in Silverlight 5. These changes make it easier to use Silverlight behind a firewall where the services often have different requirements than those on the Internet. Of course, the addition of technologies like tasks from the Task Parallel Library (TPL) and the availability of Reactive Extensions (Rx) all help to make network operations simpler. You'll find out more about those in chapter 19 as well.

Despite the promises of a paperless office, printing (covered in chapter 33) is still a staple of business applications everywhere. Printing in Silverlight 4 was optimized for relatively

short reports or documents, as well as for the equivalent of print-screen operations. Silverlight 5 adds a new Postscript or true vector printing mode to the existing bitmap approach, making serious printing and print-preview now possible.

Silverlight has supported out-of-browser applications since version 3, with trusted out-of-browser applications added to Silverlight 4. Silverlight 4 provided COM automation support for calling compatible APIs on Windows. Silverlight 5 expands on the capabilities of trusted applications by giving them more system and file access, even p-invoke access to call APIs. You'll learn about out-of-browser applications in chapter 5. Then, because Silverlight 5 enables trusted applications to be hosted in-browser as well, we'll take a deeper dive into system integration in chapter 34, and file access in chapter 32.

Finally, for really big applications, or those which must process a lot of data, Silverlight now supports 64 bit browsers with a special 64 bit version of the plug-in for Windows. 64 bit support enables applications to use more memory and resources, and avoid the (transparent to you) thunking and virtualization required on a 32 bit browser, often resulting in a decent performance increase.

One of the next major areas of enhancement for Silverlight is media.

1.2.2 Media and graphics enhancements

Silverlight was first and best known for its media capabilities. The Silverlight media team didn't rest on that success, instead pumping out enormous advances in media in each update to Silverlight.

Silverlight 2 included a Media Stream Source API for pushing media through the pipeline. But that API required that the bits be preencoded into one of the formats natively understood at the time. Though useful, this could lead to double-encoding and made transcoding even more difficult.

Silverlight 3 added support for pushing raw video and audio out of custom Media Stream Source implementations, as covered in chapter 29. As a result, you can write a managed codec for any type of media or even do something crazy like I did and use it to generate audio and video in real time for an emulator. Another option for generating video or at least images in real-time is the bitmap API covered in chapter 25.

Speaking of codecs, one of the new codecs added to Silverlight 4 was H.264 for video. H.264 has emerged as one of the most popular codecs for TV and video for devices. It was a logical choice for an additional native Silverlight format because now content producers can use even more of their content without reencoding. Silverlight 5 expanded on the H.264 support by adding hardware decoding to improve performance, especially on lower-power devices. To appeal to the same audience, Silverlight also continued to improve DRM capabilities, including the addition of offline DRM and seamless switching between DRM media sources.

One common request in the media space was for the ability to play video at 1.x or even double speed. Commonly called "Trick Play", this has been added to Silverlight 5, and is covered in chapter 28. Now you can watch my tutorial videos in 20 minutes instead of half

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

an hour, with automatic pitch correction so I don't sound like a chipmunk (although I may inhale some helium before my next recording just to mess with you).

Would you prefer to watch Silverlight video on your TV in your living room? Silverlight 5 adds support for remote controls and the media keys so many of us have on our desktop and laptop keyboards. This new feature is covered in chapter 28.

Another exciting feature introduced in Silverlight 4 was built-in support for video and audio capture devices or, specifically, webcams and microphones. Though not yet quite at the level that would allow you to create a real-time video chat application, the support does open up a number of new possibilities for application development. Webcam and microphone support are both covered in chapter 29.

Silverlight 4 added support for all formats of portable network graphics (PNG), something that was only partially supported in previous versions. The same release also introduced support for pixel shaders and a set of built-in performance-tuned effects such as drop-shadow and blur, covered in chapter 25.

Easily the most anticipated feature in Silverlight 5, the addition of the new GPU-accelerated 3D programming interface will enable all sorts of scenarios from games to data visualization to custom third-party 3d rendering and scene management systems. It even let me create a nice animated retro demo as you'll see in chapters 26 and 27.

With all of these advancements plus a number of performance optimizations and even additions such as the Microsoft Media Platform Player Framework, Silverlight continues its leadership in the media space, offering everything you need to build rich media-centric applications.

Sometimes, what you want is more than just a media experience; you want an application that can be truly interactive. Silverlight has your back there, too.

1.2.3 User interaction

Since Silverlight 2, user interaction has received a number of important enhancements. Two of the most requested features, mouse scroll wheel and right-click mouse support (both covered in chapter 8), are baked into the Silverlight core runtime. Silverlight 5 builds on that with the addition of mouse multi-click support (think double-click), also covered in the same chapter.

One of the newer and hotter user interaction mechanisms is multi-touch, also covered in chapter 9. The ability to support multipoint interaction with the user interface, especially in kiosk and handheld/tablet scenarios, is quickly becoming a requirement for many applications. Silverlight includes core runtime support for multipoint touch interaction with Silverlight application.

Another user interaction piece missing from Silverlight 2 was the easy ability to show dialogs and pop-up windows (simulated) within your applications. Silverlight has not only has those (covered in chapter 30) but also notification toast, covered in chapter 5. Also covered in chapter 30, Silverlight 5 adds in a new option – real operating system windows – which

enables you to create windows which can run on different displays, or simply overlap and act like normal windows.

Finally, all the interaction in the world has no value if your user can't read the text on the screen. Happily, Silverlight includes plenty of improvements in text as well.

1.2.4 Text

By far, the biggest improvement to text since Silverlight 2 is proper ClearType font rendering. Silverlight 2 performed only grayscale rendering, giving text a fuzzy appearance unless you carefully picked your color schemes. Silverlight 5 has expanded on this to provide even better and faster text rendering, with performance and clarity-targeted options much like we did with WPF 4. Chapter 10 explains how to use these new features.

While ClearType may be important for font rendering in general, right-to-left or bidirectional (BiDi) text is something that's absolutely essential for the correct rendering of many non-European languages. Silverlight supports not only BiDi text but also input method editors (IMEs) for complex composite characters for many languages, especially eastern languages.

Silverlight 5 added enhancements to better improve layout and typography. For example, multi-column and linked text (chapter 11) now enables magazine-like layouts with simulated fluid text flow around other elements. Tracking and leading, always important to the typography-aware (you know, the people who go around wearing shirts that say "Helvetica"), was also added.

When it comes to typography, my favorite addition to Silverlight 5 is the excellent support for OpenType. The text team added support for ligatures, alternates, style sets and much more. They even expanded on the capabilities available for eastern languages. Chapter 10 goes into detail on how to use many of these new features.

Finally, one great improvement to text rendering and entry is the inclusion of the rich text box control and the new read-only rich text block. These controls allows you to display or edit text that includes multiple fonts and styles. The controls can even embed other elements that can be interactive when the control is in read-only mode.

ClearType, BiDi and IME text, as well as the new text layout functionality and the rich text box are all covered in chapter 10, along with insight into the text rendering stack in general and how to apply these new features to text throughout Silverlight.

Those are the major items. Of course, there are many more improvements sprinkled throughout. In addition to capturing the major items in this book, I've also added information based on the experience gained from working with Silverlight since its inception as well as knowledge gained from working closely with the Silverlight and WPF product teams. In important areas, such as layout and rendering, I've gone deeper than needed by the average developer to provide some insight into the inner workings of Silverlight.

That was a lot to cover. I hope you enjoy reading it as much as I enjoyed writing it. Before we start covering individual feature areas, we'll need to get our development environment set up and build a small "Hello World!" application.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

1.3 Getting started with Silverlight development

If you're a .NET developer, you're already well on your way to becoming a Silverlight developer. Silverlight builds on the .NET framework and uses the same tools as other .NET framework applications. You'll use Visual Studio and, optionally, Expression Blend to build your applications. You'll be able to turn to CodePlex, GitHub, and other open-source sites for sample code to use. And, of course, you'll have a huge community of peers to lean on when trying to figure out those hard problems.

Before you can do any of that, though, you need to make sure your development environment is set up.

1.3.1 Setting up your development environment

Silverlight 5 requires Visual Studio 2010, at a minimum, to work with projects and build the solutions. The multitargeting support of Visual Studio 2010 means that your applications can target Silverlight 3 4, or 5, once you have the Silverlight 5 tools installed.

If you don't already have a version of Visual Studio 2010, you can get the free Visual Web Developer 2010 Express from Microsoft at <http://www.microsoft.com/express/Web/>. The free web developer tools will enable you to create Silverlight 5 applications as well as ASP.NET applications. If you want additional features and tools as well as the ability to create more than just web applications, upgrade to Visual Studio 2010 Pro or higher.

Once you have installed Visual Studio 2010, visit <http://silverlight.net/getstarted/> and use the Web Platform Installer or manual installer to install the Silverlight 5 tools and SDK as well as any optional components.

The Silverlight tools for Visual Studio 2010 and the SDK contain everything you need to develop Silverlight 5 applications, including WCF RIA Services 1.0 SP2.

Optionally, you may want to install Microsoft Expression Blend for Silverlight 5. The link for that is also available on the Get Started page on Silverlight.net. Expression Blend provides a designer-friendly set of tooling that makes creating complex animations, behaviors, and layouts a snap.

Microsoft and the community have created a number of helpful sites that will make your learning process go smoothly.

1.3.2 Helpful sites

The main MSDN home page at <http://msdn.microsoft.com> is a great starting point for both web-based Silverlight and desktop Silverlight applications. You'll find videos and tutorials (some of which I wrote/recorded) there as well as pointers to additional resources.

One of those additional resources is the official Microsoft Silverlight developer at <http://silverlight.net>. There you'll find videos, sample applications, tutorials, add-ons and the community forums, all designed to help you be the best and most efficient Silverlight developer you can be.

In addition to Silverlight.net, <http://channel9.msdn.com> includes interviews with community and product team members, as well as tutorials.

Also, as a completely shameless plug, you may want to subscribe to my own blog at <http://10rem.net>. You can also follow me on twitter; my id is @pete_brown.

Finally, one other place you'll want to visit is Dave Campbell's Silverlight Cream: <http://bit.ly/SilverlightCream>. Dave has done a spectacular job, daily compiling the best Silverlight posts on the web. From Dave's link blog, you'll get an idea of what other community member blogs to subscribe to.

At this point, your developer machine is set up, you've subscribed to a few blogs, created an account at Silverlight.net, and maybe even poked around a little on the sites. Before we get into the features in detail in the rest of the book, I thought it would be good to see just how easy it is to build your first Silverlight "Hello World!" application.

1.4 Building your first Silverlight web application

Expectations have come a long way since the days of C, where just getting "Hello World!" to compile and output to the screen was considered a great accomplishment. If we were talking assembly, I'd go with that, but we're not. Silverlight is so easy to use, we need a much higher bar for our first application.

Rather than rehash the tired "Hello World" example, I think it would be neat if our first application actually did something interesting-like hit a public service on the web. Twitter is the ubiquitous example, and far be it for me to buck a trend.

This application will be a Twitter search application. We'll hard code a search string and use that to hit a network service. We'll then take the results that come back and parse them using LINQ to XML. When displaying the data, we'll use some `ListBox` templating and binding to show the power of the lookless UI and XAML. The end result will look like figure 1.1.

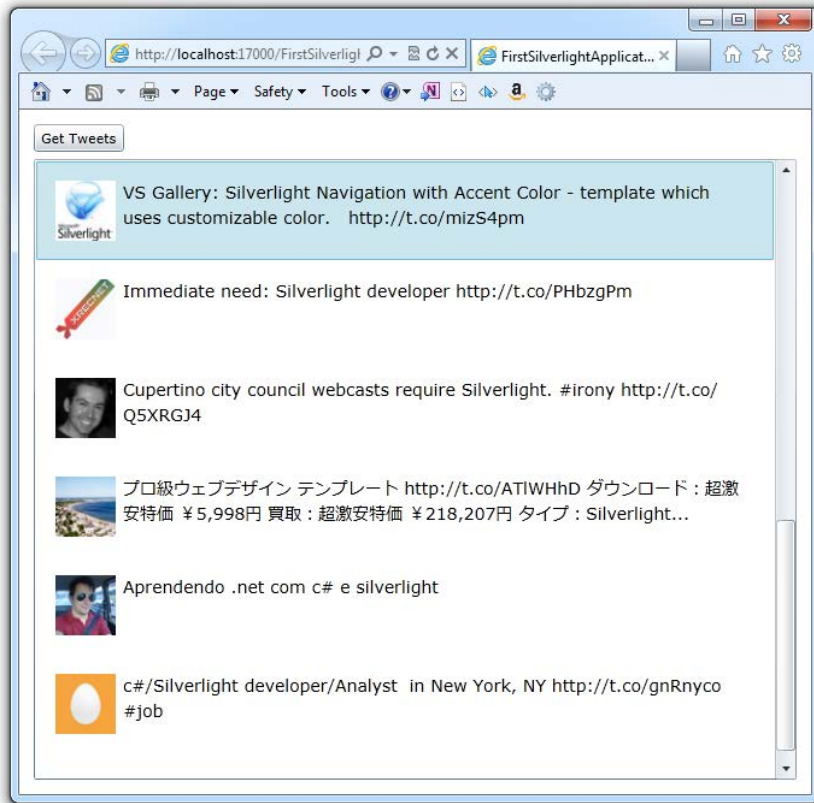


Figure 1.1 The end result of the Twitter search “Hello World!” example. That is a plain old ListBox with our own template. The data came from Twitter and was parsed using LINQ to XML.

This is a surprisingly functional application for our first application. You’ll also find it extremely easy to create using your current C# skills plus the new Silverlight skills you’ll start building in this chapter and continue to build through the rest of this book.

1.4.1 Project setup

Open Visual Studio 2010. Choose File > New Project and create a new Silverlight Application project. The name isn’t important but I chose *FirstSilverlightApplication* for mine. Figure 1.2 shows the dialog with the correct project type selected and named.

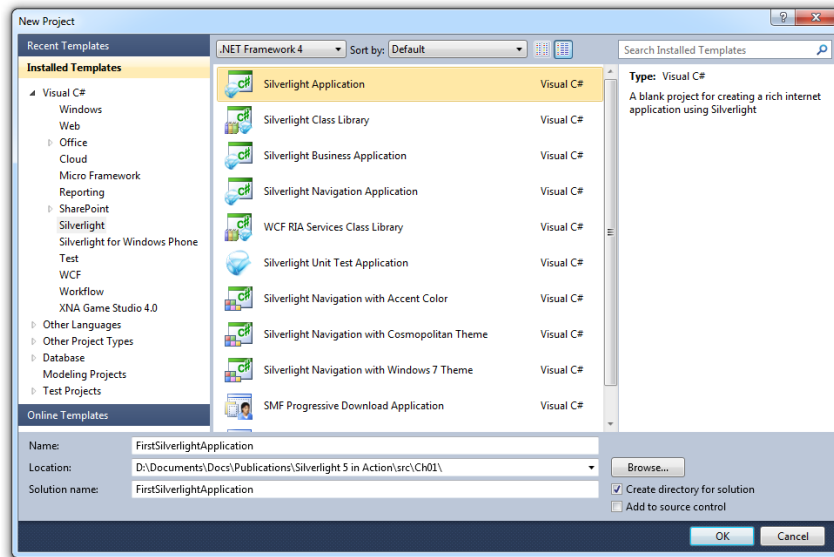


Figure 1.3 Visual Studio 2010 New Project dialog with the correct project type selected and named

Once you click OK, you'll be presented with another dialog. This dialog provides options specific to the Silverlight project. Figure 1.3 shows the dialog.

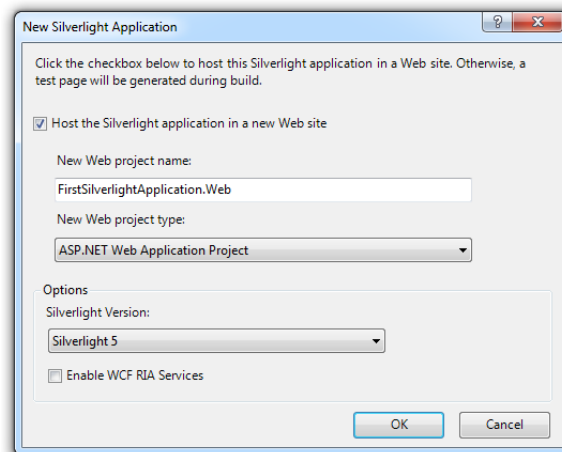


Figure 1.4 The New Silverlight Application options dialog

Typically, you'll leave the options at their default values and just click through this dialog. But it's important to understand what's available to you. Table 1.2 describes each of the options presented in this dialog.

Table 1 2 The New Silverlight Application dialog options

Option	Description
Host in a new website	Silverlight applications, even out-of-browser apps, are served from a website. You can also serve them from a static HTML page on the file system but this is a limiting option as we'll see in chapter 4. You'll typically want to leave this checked, unless you have an existing website you want to use when building your application.
New Web Project Name	Provide a project name for the website. The default is usually sufficient.
New Web Project Type	If you're an ASP.NET programmer and have a preference as to the ASP.NET project type, set it here. Otherwise, leave at the default.
Silverlight Version	This allows you to select Silverlight 3, Silverlight 4 or Silverlight 5. For this book, every example will assume Silverlight 5. You only have one plug-in installed, but if you target an older version, Silverlight will run in quirks mode to ensure compatibility.
Enable WCF RIA Services	Check this if you want to link the web project to the Silverlight project as a WCF RIA Services endpoint. This enables additional compile-time tooling.

Once the new solution is created, you'll see two projects. The first one is the Silverlight application; the second is the website. The website project contains a folder ClientBin, which will contain the compiled and packaged output (.xap file) from your Silverlight application. That .xap file is what the web page will download and load into the browser plug-in.

It also contains two test pages that may be used to test your Silverlight application. By default, the .aspx page is set as the startup page but you may use the HTML page if you later plan to host on a non-.NET server. (Yes, Silverlight applications may be hosted by any HTTP server and not just Internet Information Services [IIS] running ASP.NET.)

If you run the application at this point, you'll get a blank page in the browser. At first glance, it'll look like Silverlight didn't load. However, right-click on that empty space, and you'll get the Silverlight menu which shows you that you are, indeed, running a Silverlight application.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

With the project open and ready, it's time to turn to the user interface.

1.4.2 User interface

Open the `MainPage.xaml` file; it's usually open by default when you create a new Silverlight project. `MainPage.xaml` is the start page for your application, set as such by a single line of code inside `App.xaml.cs`. XAML (Extensible Application Markup Language) is the markup language used by Silverlight. We'll cover the basics of XAML in chapter 2, but throughout the book you'll learn about additional features of XAML.

One key thing to know right up front is that XAML is a representation of CLR objects. Each tag you see in XAML has an equivalent CLR object behind it.

Inside the opening and closing `Grid` tags, add the markup from listing 1.1 to add two objects to our UI, specified as elements in XAML.

Listing 1.1 XAML Markup for the Hello World Twitter UI

```
<Button x:Name="GetTweets" Content="Get Tweets" #A
    Height="23" Width="75"
    Margin="12,12,0,0"
    HorizontalAlignment="Left"
        VerticalAlignment="Top" />
<ListBox x:Name="TweetList" Margin="12,41,12,12"/> #B
```

#A Button Control

#B ListBox Control

That markup creates two elements on the page: a `Button` and a `ListBox`. In the design view, you should end up with a form that looks like figure 1.4.

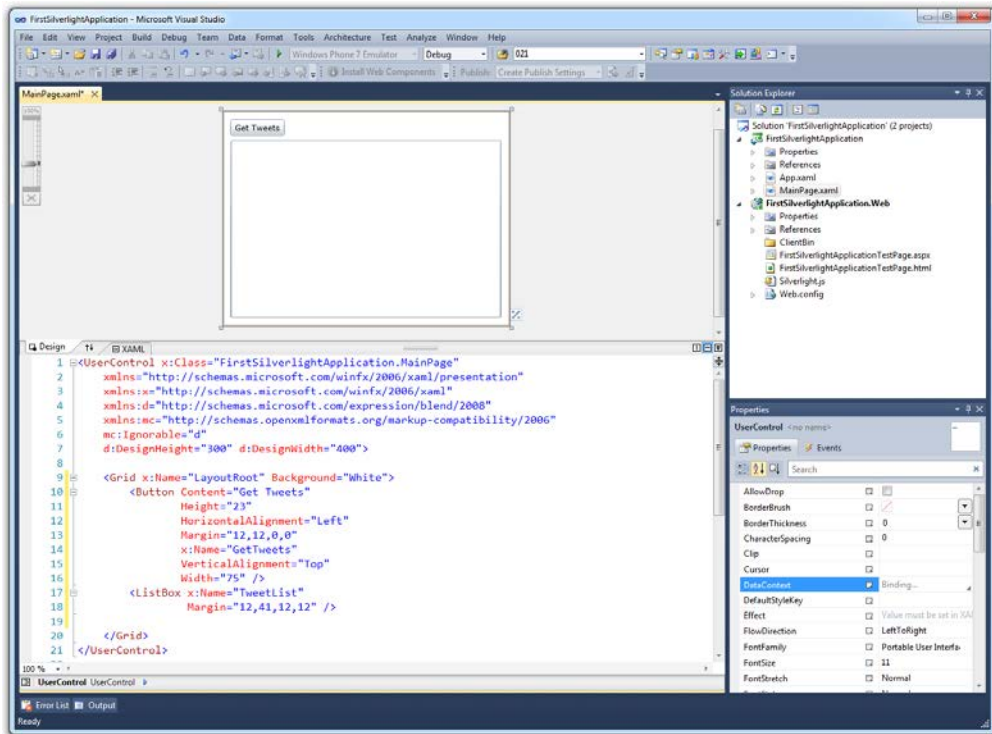


Figure 1.5 The Visual Studio 2010 IDE showing the markup correctly entered for MainPage.xaml

The editor you see is split into two parts: the top part is the XAML design surface and preview pane. The bottom part is the XAML markup. Depending upon your settings, these may actually be two different tabs in the editor.

You can drag controls into either the markup, or directly on to the design surface. The approach you take will depend upon what is more comfortable to you and which gets the job done more efficiently. For obvious reasons, the source listings in this book will list the XAML markup rather than a myriad of drag and drop instructions.

If you run the application at this point, you'll have a simple Silverlight UI with a button and an unpopulated `ListBox`. If you get any compile errors, ensure that you pasted the content into the grid as mentioned, and that your tags are all matched and correctly closed following normal XML rules.

That's about as basic an application as you can get. If you want, you can even change the button `Content` to say "Hello World" instead of "Get Tweets".

TIP

The code we've used for the Hello World example, like all code in Silverlight, will run on the client inside the Silverlight plug-in. In this case, it's also running inside the browser. The web server code is simply serving up the containing page, and the .xap it references. If you're used to coding ASP.NET applications, that's a bit of a shift. In that case, think of Silverlight apps more like the HTML and JavaScript on the client rather than like the server-side code.

If you ran the application, close the browser and stop debugging. Back in Visual Studio, double-click the "Get Tweets" button on the design surface to create an event handler in the code-behind, and automatically navigate to that code. (You could also do this by typing the event name into the XAML, but we'll look more at that in later chapters.) The event handler will be used in the next section, where we make a call to the Twitter search API.

1.4.3 Calling Twitter search

The next step is to call out to the Twitter search API. Fill out the event handler we just created in the code-behind to include the code in listing 1.2.

Listing 1.2 The event handler for the GetTweets button Click event

```
private void GetTweets_Click(object sender, RoutedEventArgs e)
{
    WebClient client = new WebClient();
    client.DownloadStringCompleted += (s,ea) =>
    {
        System.Diagnostics.Debug.WriteLine(ea.Result); #A
    };
    client.DownloadStringAsync(
        new Uri("http://search.twitter.com/search.atom?q=silverlight"));
}
#A Temporary Debug Code
```

The code here does a few interesting things. First, it creates an instance of `WebClient`, one of the easiest to use network clients in Silverlight. You'll learn about the `WebClient` in detail in chapter 18. It then sets up an event handler using a lambda expression to respond to the results. Finally, it asynchronously calls the method to download the result string from `search.twitter.com`. The search is for tweets mentioning "silverlight".

TIP

The seemingly strange lambda expression approach used here, signified by the "s,ea => {...}" simply uses an anonymous delegate (an unnamed function) as the event handler. The beauty of this approach is that it doesn't clutter up your code with tons of event handlers that are really part of discrete processes. I'll use these in a number of places throughout the book. You can learn more about lambda expressions in the C# language on MSDN at <http://bit.ly/CSharpLambda>.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

The network call is asynchronous because all network calls in Silverlight are asynchronous. This can take a little getting used to at first but is easy to deal with once you've done it a few times. Chapter 18 goes into detail on how to use the asynchronous methods as well as the reasons behind them. It also discusses using the Task class and Reactive Extensions to manage your async code.

If you run the application, click the "Get Tweets" button, and view the output window (Control-W,O if it isn't visible), you'll see that you've already built enough to call Twitter and pull back the results in XML format. Not bad for a few lines of code! Our next step is to parse the results and display them in the `ListBox` control.

1.4.4 Parsing the results and binding the `ListBox`

If you look in the output window from your last run, you'll see that the result format is an Atom-formatted XML document with an `entry` node for each of the results. In Silverlight, you can parse Atom a couple ways: you can use the built-in parsing of the `SyndicationFeed` class or you can use LINQ to XML to parse the results. Both approaches are covered in detail in chapter 21.

LINQ to XML is a great technology and has many uses above and beyond Atom document parsing, and is used far more often, so I'm going to go that route. We'll end up with a little more code than the alternative approach, but I think it's worth it for this example.

TWEET CLASS

Before we do the actual parsing, we'll need to create a simple class to hold the content we're interested in. In the Visual Studio Solution Explorer pane, right-click the Silverlight project and choose Add > Class. Name the class `Tweet.cs` and fill it out so it looks like this:

Listing 1.3 The Tweet Class

```
using System;

namespace FirstSilverlightApplication
{
    public class Tweet
    {
        public string Message { get; set; }
        public Uri Image { get; set; }
    }
}
```

The `Tweet` class will be used to contain a tweet in our application. In Silverlight, you are encouraged to use strongly-typed classes to represent all your entities, or model objects. If you're familiar with using recordsets or similar loosely-typed approaches, you'll find they either don't work well in Silverlight, or don't work at all.

Save that class and move back to `MainPage.xaml.cs` code-behind file. Somewhere inside the `MainPage` class definition, add the following collection variable. Above the `GetTweets_Click` method would be a perfect location:

```
private ObservableCollection<Tweet> _tweets =
```

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

```
new ObservableCollection<Tweet>();
```

Be sure to right-click the `ObservableCollection` type name and choose `Resolve` to add the appropriate using `System.Collections.ObjectModel` statement to your code. This collection will be the location where we place all of the parsed tweets. It's also what we'll bind the `ListBox` to. We'll use the `ObservableCollection` class in chapter 15 when we cover binding.

PARSING WITH LINQ TO XML

LINQ is something you may have used on other .NET projects. If so, you'll feel right at home because it's supported in Silverlight as well. If not, it's pretty easy to pick up. Think of it almost like SQL but in code and working on objects. Oh...and written backwards, with no database in sight. Okay, it's not exactly like SQL, but it's a great query language that lets you perform iterations and filters in a single line of code. In any case, you won't need to be a LINQ expert for this example.

Right-click the project and choose `Add Reference`; add a reference to `System.Xml.Linq`. Figure 1.5 shows the dialog with the correct reference selected. Note the location is the Silverlight 5 SDK folder.

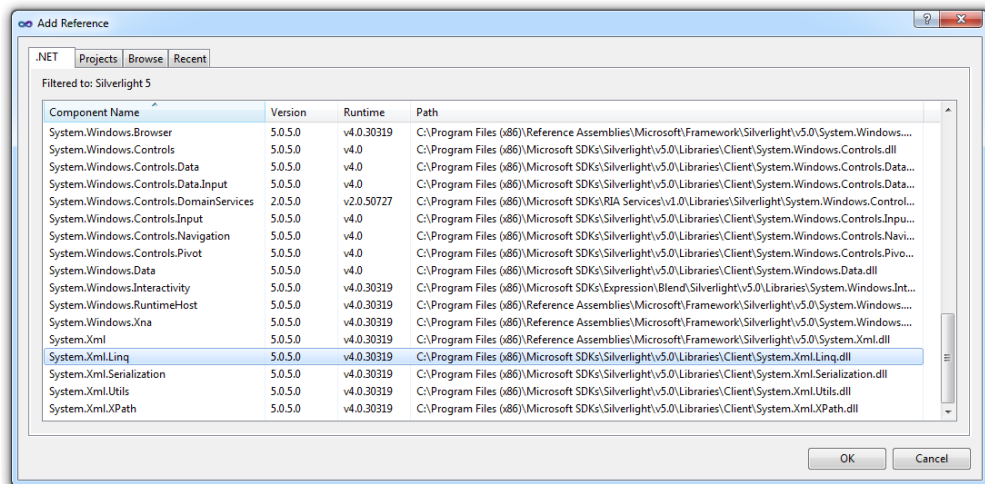


Figure 1.6 The Add Reference dialog with `System.Xml.Linq` selected for LINQ to XML functionality

Once the reference is added, replace the temporary `Debug.WriteLine` statement and the event handler declaration in the code-behind with the code inside the braces in listing 1.4. This code performs the actual parsing of the XML document returned by Twitter search and loads the `_tweets` collection with the processed results.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=786>

Listing 1.4 Processing the Twitter search results using LINQ to XML

```

client.DownloadStringCompleted += (s, ea) =>
{
    XDocument doc = XDocument.Parse(ea.Result);
    XNamespace ns = "http://www.w3.org/2005/Atom";           #A
    var items = from item in doc.Descendants(ns + "entry")
                select new Tweet()
                {
                    Message = item.Element(ns + "title").Value,
                    Image = new Uri((
                        from XElement xe in item.Descendants(ns + "link")
                        where xe.Attribute("type").Value == "image/png"
                        select xe.Attribute("href").Value
                        ).First<string>()),
                };
    foreach (Tweet t in items)
    {
        _tweets.Add(t);
    }
};

```

A Atom namespace

Be sure to right-click and resolve the `XDocument` class in order to add the correct using `System.Xml.Linq` statement to the top of your code.

The code does some interesting processing. It first loads the results into an `XDocument` so that it may be processed using LINQ statements. It then goes through the document selecting each `entry` element and creating a new `Tweet` object from each. The `Tweet` object itself is filled out by first grabbing the title element's value and assigning that to the `Message` and then doing another LINQ query to find the link element that has a type of `image/png` and assigning that to the `Image` property. Finally, the code loops through each of the results and adds them to the `tweets` collection.

The namespace declaration at the top is necessary because the Atom namespace is the default `xmlns` in the document. When parsing XML, you need to have the default namespace declared or the results will be empty. Some people like it, but that's my least favorite thing about XML.

With the parsing out of the way, the next step is to bind the `ListBox` to the `_tweets` collection so that it has a place to pull the data from.

BINDING THE LISTBOX

Silverlight is all about binding data. Chapter 11 goes into detail on how binding works and how to use it. For now, it's important to understand that rarely in Silverlight will you find yourself assigning data directly to controls. Instead, you'll set up binding relationships and let the elements pull the data as it becomes available.

In this case, we want to set the `ListBox`'s `ItemsSource` property to our collection, so that it knows to load its individual items from the collection when the collection is updated. Since we're using an `ObservableCollection`, the `ListBox` will be alerted whenever an item is added to or removed from that collection.

Add the following line of code to the MainPage constructor, under the InitializeComponent call:

```
TweetList.ItemsSource = _tweets;
```

The resulting complete code-behind should look like listing 1.5.

Listing 1.5 The complete code-behind for MainPage

```
using System;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Collections.ObjectModel;
using System.Xml.Linq;

namespace FirstSilverlightApplication
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            TweetList.ItemsSource = _tweets; #A
        }

        private ObservableCollection<Tweet> _tweets =
            new ObservableCollection<Tweet>(); #B

        private void GetTweets_Click(object sender, RoutedEventArgs e)
        {
            WebClient client = new WebClient();

            client.DownloadStringCompleted += (s, ea) =>
            {
                XDocument doc = XDocument.Parse(ea.Result);
                XNamespace ns = "http://www.w3.org/2005/Atom";
                var items = from item in doc.Descendants(ns + "entry")
                    select new Tweet()
                    {
                        Message = item.Element(ns + "title").Value,
                        Image = new Uri((
                            from XElement xe in item.Descendants(ns + "link")
                                where xe.Attribute("type").Value == "image/png"
                                select xe.Attribute("href").Value
                            ).First<string>()),
                    };
                foreach (Tweet t in items)
                {
                    _tweets.Add(t);
                }
            };
        }
    }
};
```

```

        client.DownloadStringAsync(new #C
        Uri("http://search.twitter.com/search.atom?q=silverlight"));
    }
}
}

```

#A Bind ListBox
#B ObservableCollection
#C Network call

That's all you need to do to set up the binding relationship for the `ListBox`. Run the application and retrieve the tweets. You should end up with something that looks like figure 1.6.

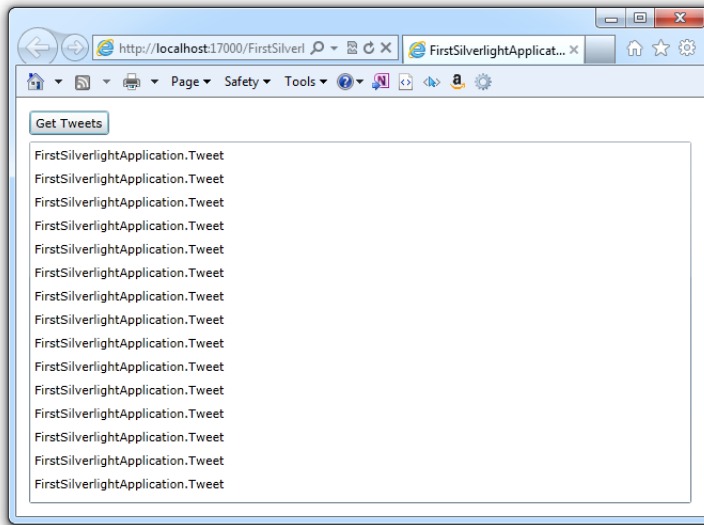


Figure 1.7 The default presentation for the `ListBox` items leaves something to be desired. It looks like WinForms or something! I demand more from our first Silverlight example!

That's not really what we want, though. All we see are a bunch of type names. We want to display images and text. The reason you see the type name is because this is the default item template behavior. By default, the individual items are presented as their `ToString` call. This works fine for a string or numbers or similar, but with complex types? Not so much. Chapter 15 talks more about item templates and how to use them with the `ListBox` control.

Our final step in this walkthrough is to pretty up the `ListBox` results so we can see something more meaningful.

1.4.5 Making the ListBox contents more meaningful

To make the `ListBox` present items using a format of our own choosing, we need to use a `DataTemplate`. `DataTemplates` are covered in detail in section 11.4. For now, understand that they're a chunk of XAML that'll be used to format each item in the list.

The `DataTemplate` for this `ListBox` will contain two columns for each row. The first column will contain the picture of the tweeter; the second will contain the body of the tweet.

Open `MainPage.xaml` and replace the entire `ListBox` declaration (not the `Button`, just the `ListBox`) with the XAML from listing 1.6.

Listing 1.6 DataTemplate to format the tweets

```
<ListBox x:Name="TweetList"
    HorizontalContentAlignment="Stretch"
    ScrollViewer.HorizontalScrollBarVisibility="Disabled"
    Margin="12,41,12,12">
  <ListBox.ItemTemplate>
    <DataTemplate>
      <Grid Margin="10">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="Auto" /> #A
          <ColumnDefinition Width="*" /> #B
        </Grid.ColumnDefinitions>
        <Image Source="{Binding Image}"
            Grid.Column="0"
            Margin="3" Width="50" Height="50"
            Stretch="UniformToFill"/>
        <TextBlock Text="{Binding Message}"
            FontSize="14" Margin="3"
            Grid.Column="1" TextWrapping="Wrap" />
      </Grid>
    </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>
#A Autosized Column
#B Full-Width Column
```

In this markup, we first tell the `ListBox` that we want its content to take up the full width of the `ListBox`, without any horizontal scrolling. The `HorizontalContentAlignment` and `HorizontalScrollBarVisibility` elements do that.

Then, we break out the `ItemTemplate` property, and supply it with a `DataTemplate`. Inside the `DataTemplate`, we define the grid, with an autosized first column and a full-width second column. We'll cover data templates more in chapter 16 when we cover binding, but for now understand that the template is applied to each item loaded into the `ListBox`, almost like a macro expanded for each Tweet. Then, we bind an `Image` element to the `Image` property of the `Tweet` class and a `TextBlock` to the `Message` property of the same.

The end result of the work we've done, including this fine `ListBox DataTemplate`, is shown at the start of this section in figure 1.1.

I've been working with Silverlight and WPF for well over half a decade now, but it never fails to impress me just how easy it is to have complete control over what your application displays. I remember the days when you had to purchase specialty controls to do something as simple as display an image inside a `ListBox`. Now, all you need to do is a little XAML. And, if you don't feel like typing in XAML, you can crack open Expression Blend and use it to design the `DataTemplate` interactively on the design surface. As a famous dark lord of the Sith once said, "Impressive...most impressive."

1.5 Summary

Silverlight is one of the most promising development platforms to come from Microsoft since the original release of .NET a decade ago. Silverlight fills a niche that sits solidly between traditional desktop applications and web applications, while offering capabilities that both lack. It does all this via a small plug-in that takes only minutes to install and runs on different browsers and different operating systems.

The code you write and the skills you gain are portable between the desktop and the web, devices in your pocket, game consoles in your living room, and the set-top box on your TV. That's a pretty good return on your investment.

Silverlight has come a long way since the Silverlight 2 version covered in the original edition of this book. It's amazing just how much the product teams have been able to pack into the product in those few years. In fact, even in between Silverlight 4 and Silverlight 5, we've seen a huge number of great features and improvements added. Before I joined Microsoft, I heard rumors about people with sleeping bags in their offices and coffee delivered by the gallon. I suspect I now know which team they work for, and I have to say that I'm "super" impressed with the results.

Your environment is all set up, and you've whet your appetite by building a basic yet non-trivial "Hello World!" Twitter application in Silverlight 5. In the next chapter, we'll dive right into the meat of what makes Silverlight UI work: XAML. From there, we'll take a tour of all the features this platform has to offer. By the end of this book, you'll have all the knowledge you need to build awesome Silverlight applications.