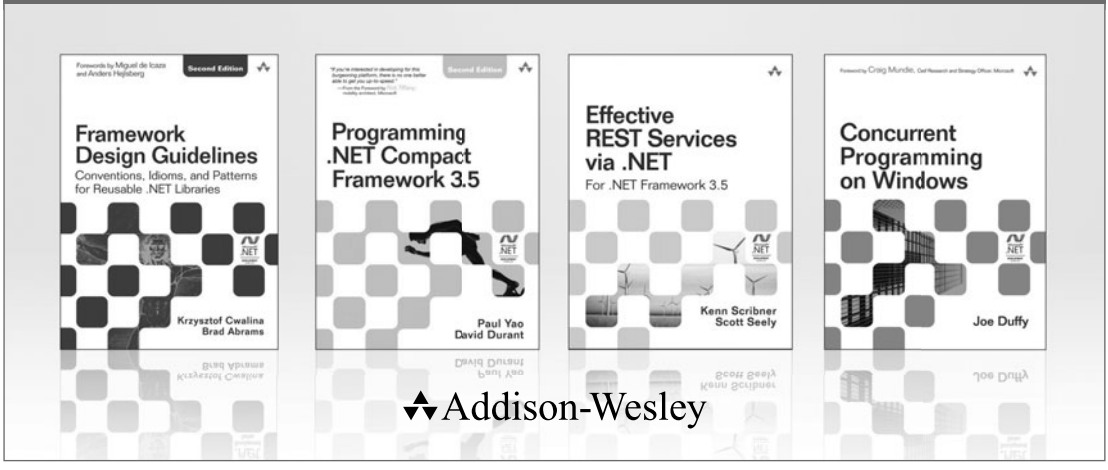
Foreword by Andrew Connell, Critical Path Training, LLC

# SharePoint® 2010 Development with Silverlight®


Bob German
Paul Stubbs

# SharePoint 2010 Development with Silverlight

# Microsoft® .NET Development Series



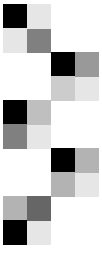

Visit **informit.com/msdotnetseries** for a complete list of available products.

---

The award-winning **Microsoft .NET Development Series** was established in 2002 to provide professional developers with the most comprehensive, practical coverage of the latest .NET technologies. Authors in this series include Microsoft architects, MVPs, and other experts and leaders in the field of Microsoft development technologies. Each book provides developers with the vital information and critical insight they need to write highly effective applications.

PEARSON

Addison-Wesley | Cisco Press | EXAM/CRAM | IBM Press | QUE | PRENTICE HALL | SAMS | Safari Books Online



# SharePoint 2010 Development with Silverlight

Bob German
Paul Stubbs

Addison-Wesley
Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

*Library of Congress Cataloging-in-Publication Data:*

German, Bob.

Sharepoint 2010 development with Silverlight / Bob German, Paul Stubbs. — 1st ed.

p. cm.

ISBN 978-0-321-76959-6 (paperwork)

1. Microsoft SharePoint (Electronic resource) 2. Silverlight (Electronic resource) 3. Intranets (Computer networks) 4. Web servers. I. Stubbs, Paul R., 1969- II. Title.

TK5105.875.I6G46 2012

004'.36—dc23

2011036853



ISBN-13: 978-0-321-76959-6
ISBN-10: 0-321-76959-7

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.

First printing November 2011

*I dedicate this book to my parents, Don German and Joan German-Grapes, who inspired and encouraged me to write.*
*—Bob*

*This book is dedicated to my brilliant friends and colleagues in the SharePoint community who inspire and encourage me every day.*
*—Paul*

*This page intentionally left blank*

# Contents at a Glance

*This page intentionally left blank*

# Contents

# Foreword

As Microsoft developed Silverlight versions 3 and 4, it enabled developers to create compelling business applications that were distributed and run in the browser with a rich, refreshing, and engaging experience. This technology was a natural addition to the SharePoint developer's toolbox, as so many companies store business data within intranets and extranets on the SharePoint platform. With the release of SharePoint 2010, Microsoft made it easier to consume and integrate data stored within SharePoint into Silverlight applications with the client object model and a new RESTful service.

While many technologies (such as HTML 5) promise and deliver, to varying degrees of success, the ability to build rich business applications in the browser, Silverlight has a proven and mature track record. It is an obvious choice when building a new business application. SharePoint serves not only as a fantastic delivery mechanism, but the application can also leverage the vast amounts of business data that is stored in corporate SharePoint deployments.

Over the years, I've been fortunate enough to know and work with both Bob German and Paul Stubbs. Bob and I have worked on other book projects, and I've worked on numerous development projects with Paul. Both have solid, real-world experience and perspectives on the SharePoint platform and both also spent a considerable amount of time with Silverlight. They have presented many informative and engaging presentations at conferences and user groups, as well as written numerous articles on the subject. Who better to collaborate on the topic!

Most SharePoint development books only touch on the client object model and how to use the Silverlight implementation or the new ListData.svc RESTful service. If you are building a Silverlight business application, you need a good resource from some trusted names to deliver solid guidance on working with both Silverlight and SharePoint together.

The authors break the learning experience into three parts. Part 1 of the book focuses on getting you up-to-speed quickly on SharePoint and Silverlight development. Part 2 dives into the fundamentals and basics you need to know, such as working with the client object model, the REST service, web services, and external data (that which SharePoint is aware of but lives in another system). Part 3 kicks into high gear, teaching you how to leverage Silverlight to create sophisticated navigation controls, utilize the emerging and ever more important cloud, and even create custom field controls.

I can't imagine two better people to collaborate and deliver a fantastic book on the subject of SharePoint 2010 and Silverlight. Consider this a must-have for your bookshelf…I do!

*Andrew Connell*
*Co-Founder, Developer, Instructor, Speaker*
*Critical Path Training, LLC*
*www.CriticalPathTraining.com*
*August 2011*

# Preface

In early versions of SharePoint, the developer experience was an afterthought at best. Microsoft finally opened up a supported way for developers to create SharePoint features in 2007. Although the tooling was still primitive, this led to an interest in developing applications on top of SharePoint. These solutions are generally cheaper and faster to build and more flexible for business users because they build on all the capabilities included in SharePoint.

Around the same time, the Internet was offering a richer user experience. Page refreshes became passé in favor of pages that were interactive. This drove a number of client-side technologies for bringing pages to life right within a web page. Silverlight was making a name for itself as a very productive way to build compelling business applications that run in a web browser.

The authors both noticed that more and more customers were asking how they could develop rich business applications on SharePoint, the kind of applications that lend themselves to a Silverlight user interface. Paul co-authored a book about SharePoint and Silverlight, which shows how to build solutions using the tools that were available at the time.

The advent of SharePoint 2010 and Visual Studio 2010 changed everything. Suddenly SharePoint wasn't just allowing applications, but it was encouraging them. Features like sandboxed solutions and client object models enabled a whole new class of light-weight applications. And the tooling in Visual Studio 2010 removed the tedious and arcane aspects of SharePoint development and seamlessly knitted in Silverlight as well.

Bob and Paul started speaking on SharePoint and Silverlight development and developed collections of sample applications. And both wanted someday to write a book (or another book!) on the topic. At one of the conferences after speaking in adjacent rooms, they decided to coauthor this book.

This book is for any .NET, SharePoint, or Silverlight developer who wants to learn how to build a new, richer class of applications. SharePoint provides a data layer, a hosting platform, and a suite of collaboration and publishing features to build on. Silverlight makes the experience richer and easier to use.

Late one night last winter, Bob's wife Kate wandered into his home office and observed how much time he was putting into this book. "But," she added, "you seem to be having fun!" It's true, programming with SharePoint and Silverlight is actually fun!

Whether you read it during your day job or late at night, may this book bring some of that fun to you, too.

# Acknowledgments

ALTHOUGH THERE ARE ONLY TWO NAMES on the cover, this book is the result of many people who contributed their time, energy and expertise to the project.

First of all, we want to thank Matt Burnett, who wrote Chapter 14 on Office 365 and Windows Azure. Matt works for Microsoft Consulting Services and has a wealth of experience making SharePoint and Silverlight work with Microsoft's cloud offerings. We were really glad he agreed to bring his knowledge and expertise to the book.

The technical review team was a cast of SharePoint luminaries, and we were very fortunate and honored to have them. The team members were: Andrew Connell and Ted Pattison, co-founders of Critical Path training; Scott Jamison, CEO of Jornata; Matt Jackson, Director at BlueMetal Architects; and Ed Hild, Architect at the Microsoft Technology Center in Reston, Virginia. Their perspectives and guidance greatly improved the quality of this book.

We'd also like to thank everyone from Addison-Wesley who contributed to this book, many of whom we never had the opportunity to meet. We'd especially like to thank Joan Murray for the opportunity to write the book, for her constant feedback and encouragement, and for deftly guiding us throughout the publishing process.

**Bob German**

I want to thank my parents, who wrote more than 35 books, for inspiring me to write and exposing me to the writing process at a young age. I remember proofreading galleys with them as soon as I learned to read.

I also want to thank my teachers: John Campbell, for introducing me to programming as a child, and my many excellent college professors, especially Mark Seiden and the late Anita Goldner. I thank Scott Jamison for my first serious education in SharePoint on a project in 2002 and Ted Pattison for sharing his development wizardry and exposing the magic that makes it all work.

I'm thankful to Paul Stubbs for being a great and experienced coauthor and helping me with this, my first book project, with lots of ongoing technical and writing advice. Also his chapters are great!

Andrew Connell has been a great friend and mentor throughout the project. He gave me the opportunity to write two chapters in his Web Content Management book, which was an extremely valuable experience. He also gave me a huge amount of encouragement and guidance.

Ed Hild was also an invaluable advisor and sounding board. He shared a great deal of helpful experience from his own book writing and was equally helpful in working out technical problems and digging deeply into issues while reviewing the book.

My most heartfelt thanks goes to my wife, Kate Severinsen, who supported and encouraged me throughout the project. She cut me endless slack while I was working nights and weekends on the book, and she reminded me to stop and laugh along the way.

**Paul Stubbs**

First I want to thank Bob German for being a great coauthor. This may be Bob's first book, but he was the one that held it all together and went above and beyond to see this book to completion. Bob is going to have a bright future in writing more books, and I look forward to doing more projects with Bob in the future.

I want to thank Matt Burnett for being a good friend to me over the years and listening to all of my crazy ideas. Matt is always ready to help me

solve the tough problems that come up when developing SharePoint solutions.

I also want to thank Steve Fox. Steve has been a good friend and was the co-author of my first SharePoint and Silverlight book years ago. Steve has also been a real motivation for me in writing. He is a writing machine, cranking out multiple books a year. This has driven me to try and keep up and finish projects that I never would have even started in the past.

Last, but certainly not least, I want to thank my wife Rosa for allowing me the time required to write yet another book.

*This page intentionally left blank*

# About the Authors

**Bob German** is an architect at the Microsoft Technology Center (MTC) near Boston, Massachusetts, where he helps customers create and prove out solutions that fit their business and technology needs. Bob works on SharePoint solutions for customers in a wide range of industries and technology environments. He also advises independent software vendors who are looking to build products on, or integrate them with, SharePoint technologies.

Bob's career began as a systems programmer in the minicomputer industry. Eventually he became a project leader and architect specializing in network protocols and distributed systems. In 1995, he took his networking and development experience to Microsoft Consulting Services. This soon led to web development engagements, including a knowledge management web site for a major industry analyst. The site was based on Site Server 3.0, a precursor to SharePoint.

In 2000, Bob joined the very first Microsoft Technology Center and provided consulting services in an incubation environment to the burgeoning dot-com industry. This involved quite a bit of performance and scalability testing and plenty of troubleshooting because most of the applications crashed under load testing. It also involved helping out with some pretty cool web sites, although not all of them saw the light of day.

Bob has specialized in SharePoint technologies since a major project in 2002 threw him head-first into the SharePoint 2003 beta. He's helped many customers get started and regularly develops SharePoint and Silverlight solutions for proof of concept and demonstrations. Bob is a frequent

speaker at conferences such as TechEd North America, the Microsoft SharePoint Conference, and MIX, as well as at user groups and SharePoint Saturdays.

**Paul Stubbs** is a Microsoft Technical Evangelist for SharePoint and Office, where he focuses on information worker development community around SharePoint and Office, Silverlight, and Web 2.0 social networking. He has authored several books on solution development using Microsoft Office, SharePoint, and Silverlight, several articles for *MSDN Magazine*, and has also spoken at Microsoft Tech-Ed, PDC, SharePoint Conference, DevConnections and MIX conferences.

Paul has also worked as a Senior Program Manager on Visual Studio in Redmond, Washington. Paul is a Microsoft Certified Trainer (MCT) and frequently participates in the developer community on the Microsoft forums. Visit Paul's blog at blogs.msdn.com/pstubbs for deep SharePoint developer information.

# PART I

## Getting Started

*This page intentionally left blank*

# 1

# Getting Started with SharePoint and Silverlight

SHAREPOINT AND SILVERLIGHT are a great combination of technologies for building great web applications. Users can create and configure their own web-based collaboration and publishing solutions with SharePoint and can incorporate richer user interface components with Silverlight. Further, Silverlight extends the things that SharePoint's user-installable "sandboxed solutions" can do, such as reaching across SharePoint site collections and line of business systems and integrating multi-media features.

For developers, SharePoint provides an easily-packaged data layer, and Silverlight allows rich display and interaction with that data. SharePoint and Silverlight offer a unified development experience based on Visual Studio 2010, a consistent runtime environment (.NET) on both client and server, and extensive client-side APIs for accessing SharePoint in Silverlight.

This book began at Microsoft technology conferences such as TechEd, the Microsoft SharePoint Conference, MIX, and other venues where the authors delivered a variety of talks on the subject of SharePoint and Silverlight. Attendance was high, and feedback was positive, revealing a lot of interest in this combination of technologies. The authors of this book dive much more deeply than a conference talk or boot camp would allow, however, showing you all the tricks and techniques for being successful with this strong combination of technologies.

## Why SharePoint?

A great struggle for control has been underway since the first business-oriented personal computer, the IBM model 5150, was introduced in 1981. Finally business managers could thumb their noses at lengthy IT backlogs and take direct control over computing tasks by purchasing a PC and using the simple word processors, spreadsheets, and other applications that were available at the time.

Although this independence led to business innovation and empowerment, it also led to a number of unanticipated problems. The IT people, despite their backlogs and sometimes conflicting priorities, had been securing and backing up their software and data, ensuring it complied with relevant laws and policies, and planning for contingencies in case anything went wrong. The newly empowered PC users often skipped over such concerns, unwittingly adding huge business risks. Meanwhile, data proliferated in companies, leading to confusion when a dozen variations of the same spreadsheet all yielded different results, with no way to know which was the right one.

Over time, personal computing has become ubiquitous, and IT has found ways to manage their companies' personal computers. Although some of these risks can be mitigated by, for example, a group policy that forces everyone to encrypt their data, other risks still remain.

SharePoint is one of a new breed of application environments that balances the needs of business users and those of IT. With SharePoint, business users can innovate and build simple solutions on their own while IT ensures that the environment is secure and backed up. The central idea of "sharing" eliminates or reduces the proliferation problem, so users all work on one common set of data, conflicts don't arise, and IT can govern the environment to encourage business users to comply with legal and company policies.

All this has led to SharePoint being a huge success in the marketplace. It's a platform that allows business and IT to work together rather than at cross purposes.

To a business user, SharePoint is a place to collaborate and publish information. Many simple business solutions can be created directly by savvy business users, with no need to involve IT in the details. It is mainly browser based; however, rich applications also integrate with SharePoint so users can share directly from tools such as Microsoft Word and Microsoft Outlook.

One of SharePoint's strengths is its extensibility. A developer can add functionality to the palette of available features, and business users can then use these extensions to build richer solutions. The most common extension by far is to add custom "web parts," which are small application components that appear on the screen as part of a SharePoint solution. (Web parts are similar to "portlets" or "widgets" used by other portal platforms.) However this is really only the beginning, as developers can also extend SharePoint workflows, add custom application and administrative pages, connect to line-of-business data, and more.

SharePoint's popularity, along with this extensibility, has led to a whole marketplace of independent software vendors who provide add-ons to SharePoint. SharePoint integration has become a critical component of many business applications, which can then be combined in the SharePoint user interface for simple, one-stop access by business users.

From a technical point of view, SharePoint has another strong advantage, which really amounts to code re-use. Why reinvent site provisioning or document management when SharePoint has both? Why create a new security model or rendering framework when you can build on an already established one? Why spend resources figuring out how to package your application or host your workflows when the SharePoint team already made the same investment? The list goes on and on, and since the entry level product, SharePoint Foundation, is free with the Windows Server operating system, it need not add to your cost of entry.

All in all, SharePoint saves developers work and comes with a large marketplace of customers who have already adopted SharePoint and want to extend its capabilities. This is why so many developers have gone beyond ASP.NET and are developing on SharePoint as a platform.

**TIP**

Shocking as it might seem, there is no product called SharePoint! SharePoint is a family of products that build on one another, each adding more capabilities and features. Throughout this book, the word "SharePoint" refers to the family of products because it's a lot shorter than spelling out "Microsoft SharePoint 2010 Products and Technologies."

The base product, Microsoft SharePoint Foundation 2010, is a free download and includes basic document management and collaboration. Microsoft Share-Point Server 2010 comes in Standard and Enterprise editions, each adding more features.

## Why Silverlight?

It wasn't long after the introduction of NCSA Mosaic, the first graphical web browser, that it was dubbed a "killer application." Instead of a hodgepodge of tools such as WAIS, FTP, and Gopher, the web browser provided universal access to Internet resources in a way that was easy enough for any computer user.

Yet the standardization that made the World Wide Web possible has also been a limiting factor. Standardization takes time, and interoperability is tricky. To this day, web developers need to test on a variety of target web browsers to ensure their sites look right, and they need to be aware of quirks that can affect the behavior and rendering in one browser versus another. This makes web development inherently more difficult and less flexible than other development environments.

There are at least two ways to address these issues. Runtime environments such as JQuery on the client side or ASP.NET on the server try to hide the browser-specific quirks from developers so they can focus on their applications. These environments work pretty well, but cross-browser testing is still advised, and the applications are still functionally limited to rendering what the target browsers can support.

Another approach is to use a browser plug-in, such as Oracle Java, Adobe Flash, or Microsoft Silverlight. With this approach, a trusted third-party builds a plug-in that runs in multiple browsers, and applications run inside the plug-in. These applications may appear to be part of a web page, yet the plug-in can go beyond what the web browser can do. For example, a plug-in can display streaming video even in browsers that don't have any video features by bypassing the browser and accessing the native operating system.

This architecture generally includes some kind of "sandbox" to protect end-users from malicious or poorly constructed applications. If the user trusts the plug-in, which comes from a major, established software vendor, he knows that the plug-in will limit what applications can do to his computer when they run.

Silverlight is a .NET-based plug-in that runs in Firefox, Internet Explorer, and Safari on Windows, Macintosh, and on Linux desktops as well through the "Moonlight" project. Silverlight adds a lot of functionality to the browsers it supports, including

- Consistent rendering on all supported platforms, using an extensive set of reusable controls from Microsoft and other software vendors
- A strongly-typed object-oriented development environment based on the popular .NET framework
- Effective separation of visual design and code, which, along with advanced data binding technology, allows designers and developers to work more independently and greatly facilitates automated testing
- 2D and 3D vector animation and graphics
- Video (up to 720p high definition) and audio streaming in a number of standard formats
- Isolated storage for saving state on the client
- Easy access to web services and network sockets, with support for advanced scenarios such as multicast networking
- Access to client devices such as webcam and microphone

- Access to the web browser for tight integration with JavaScript and dynamic HTML
- Support for theming, localization, visual state management, multi-threading, accessibility, and other attributes that are useful in many applications

At this writing, the RIA (Rich Internet Application) Statistics web site at http://riastats.com/ reports that Silverlight is installed on about 70% of client computers on the public Internet. Most SharePoint sites, however, are not on the public Internet but are used within enterprises as "intranets" for employee use or as "extranets" for working with business partners. In these environments it's easier to ensure the Silverlight plug-in is available; indeed, installation across an enterprise can be automated using Windows Update Services. This means that Silverlight is likely to be available or could be made available to users of most SharePoint sites.

There's been a lot of excitement in the industry lately about the forthcoming HTML 5 standard, which will provide a number of features such as 2D vector graphics and video support that were previously only available using browser plug-ins. At the time of this writing, HTML 5 is in Working Draft stage, and features based on the draft are beginning to show up in new versions of web browsers such as Internet Explorer, Mozilla Firefox, and Google Chrome.

When the new standard is complete, it will be implemented in incremental releases by browser vendors, continuing to complicate compatibility testing. Script libraries like JQuery and KnockOut can help by offering features such as cross-browser consistency and data binding to the browser programming experience. Other libraries like Modernizr can check to see what browser features are available so the developer can adapt the user interface accordingly. Many web developers hope that these advances will finally make developing browser-based code as easy and productive as other modern development environments.

In the meantime, developers need to choose between a browser plug-in such as Silverlight, grappling with the emerging HTML 5 draft implementations, or sticking with more mature but functionally limited web

standards such as HTML 4. There is no one-size-fits-all answer to this decision, and in some cases it can be a tough one to make.

The key is to focus, as developers have always done, on the target for the application. If the application must run on devices that don't run Silverlight, then clearly it's not an option. But if the application is targeted toward computers running Windows or Mac OS, or mobile devices running Windows Phone 7 (which runs Silverlight natively), then developers can take advantage of all that Silverlight offers. In addition to providing a richer user experience, Silverlight can reduce development and testing time by providing a strongly-typed object-oriented development environment that works consistently across platforms. It's also a good approach for developers who know .NET because they will be able to leverage their knowledge in Silverlight.

HTML has a rich future for sure, and Silverlight will be there as well. Browser technology will continue to advance, reducing the need for plug-ins, and plug-ins will advance as well to fill gaps in the new browsers. If you're working in an environment where you can ensure Silverlight is available, and want to take advantage of its consistency, productivity, and features, then go for it! If you're not sure but want some of the advantages of Silverlight anyway, then selective use of Silverlight within an otherwise HTML UI might be advised. It's a balancing act that everyone needs to be aware of as the technology evolves. This may not make the decision obvious, but hopefully it can help with the thought process.

## Why SharePoint and Silverlight Together?

The phrase "better together" has become almost a cliché at Microsoft, as it engineers its products for easier integration with one another. SharePoint and Silverlight are indeed better together for a number of reasons.

First and foremost, both are based on the .NET framework, and both share a common development tool (Visual Studio 2010), which in addition to reducing the learning curve for developers, generally simplifies development. A Visual Studio 2010 solution can contain both SharePoint and Silverlight projects, and the output of the Silverlight projects can be automatically included in the SharePoint deployment package. Debugging

is also a unified experience; a developer can set and hit breakpoints in both the client and server-side code when troubleshooting code.

In addition, SharePoint provides a client object model for Silverlight to allow easy access to SharePoint content. This is also true for JavaScript, but not for other browser plug-ins such as Adobe Flash. Developers for Flash or Java could consume SharePoint's SOAP web services or RESTful OData interface, but the level of difficulty could increase dramatically.

Another important consideration is the emergence of sandboxed solutions in SharePoint 2010. Many people think of a development or testing environment when they hear the term "sandboxed solutions" but this is something different.

Sandboxed solutions provide an isolated environment for running applications that are only partially trusted, whether in development, testing, or in production. This allows end-users to upload SharePoint web solution packages they have written and purchased and run them without putting the SharePoint installation at risk. The sandbox means that whoever is hosting SharePoint, be it the local IT department or an online service such as Microsoft Office 365, can allow the code to run without worrying about security breaches, memory leaks, or other issues that could affect the overall SharePoint environment.

Sandboxed solutions are, by necessity, limited in nature. They can declare workflows, lists and library structures, and they can include .NET code, but the code runs with very restricted privileges. It cannot, for example, make any kind of network or database call, nor can it access the SharePoint object model outside of the site collection where it is installed.

Silverlight is a natural complement to sandboxed solutions because it can access resources directly from the client that would be outside of the reach of the SharePoint sandbox. For example, Silverlight can easily call a web service or another SharePoint site collection using the client object model. Because the Silverlight application can be deployed right in the SharePoint web solution package, end users can install it like any other SharePoint solution and need not be bothered with the details of deploying the Silverlight application or embedding the Silverlight plug-in on the page.

Finally, using SharePoint with Silverlight can simplify Silverlight applications while giving the user more flexibility. Rather than using a framework such as the Microsoft Extensibility Framework (MEF) or PRISM, SharePoint and Silverlight follow a similar pattern by allowing users to dynamically add web parts without recompiling the application. The assemblies just reside in a library as .xap files.

All in all, Silverlight can make SharePoint solutions richer and more powerful for end users while making the development experience simpler as well.

## Who Should Read This Book

This book is written for developers, architects, and application designers who want to build solutions using SharePoint and Silverlight. It assumes you have a working knowledge of .NET programming, especially ASP.NET, which is the basis for SharePoint.

The book is focused on where SharePoint and Silverlight meet and shows you how to use the two technologies in concert. Although it doesn't offer comprehensive coverage of either SharePoint or Silverlight, it does provide a sufficient introduction to allow someone new to either or both technologies to understand what's in the book.

There is a code download to accompany the book, which is located at www.informit.com/title/0321769597. Most chapters include code samples to illustrate the concepts, and all the code is available at this location so you can try it out in your own environment. The code listings in the book are intended to illustrate concepts, but supporting code and packaging isn't always shown. If you want the complete solutions, they're in the download.

## How to Use This Book

This book is organized so you can read it from end-to-end or in pieces according to your needs and interests.

The first few chapters are introductory in nature, and you might choose to skip over them. If you already have a SharePoint and Silverlight

development environment set up, you don't need to read "Creating a Development Environment" later in this chapter. Chapter 2, "Introduction to SharePoint Development," provides an introduction to SharePoint development targeted at the ASP.NET developer; Chapter 3, "Introduction to Silverlight Development," does the same for Silverlight. If you already know the basics, you can skip over these chapters.

Chapters 4 through 11 form the core of SharePoint and Silverlight development. Although each chapter can stand on its own, any given chapter might refer back to concepts from an earlier one. Chapter 4, "A First Look at Silverlight in SharePoint," explains the Silverlight features that are built into SharePoint 2010, and Chapter 5, "Web Part Development," gets you started developing Silverlight web parts for SharePoint. Chapter 6, "Expression Blend, Data Binding, and Sample Data," explains how to use Expression Blend with SharePoint and Silverlight. Expression Blend is a design tool for Silverlight that makes it easy to prototype and visually design Silverlight applications. Then Chapters 7 through 11 focus on various ways of accessing SharePoint content in Silverlight, ranging from the new client object model to web services and OData access.

The last four chapters focus on specific situations. You learn how to work with SharePoint in Windows Phone 7 applications and how to develop for the new hosted Office 365. You also learn how to use Silverlight in site navigation and how to create field controls that use Silverlight to render and edit new kinds of data in SharePoint.

### SILVERLIGHT VERSIONS IN THIS BOOK

Silverlight 5 was in beta testing while this book was being written. Although most of the programming examples work with Silverlight 4 or 5, some chapters include special sections to show how you can take the solutions a step further with new Silverlight features. These sections are shaded for easy identification.

## Creating a Development Environment

SharePoint development took a big leap forward in the 2010 version due to greatly improved tooling built into Visual Studio 2010. All you have to do is press the F5 key, and Visual Studio will build your projects, package them, and deploy them to a local SharePoint server for debugging. However, this experience assumes that there is a local SharePoint server running on the same computer as Visual Studio. So for practical purposes, every developer will need his own copy of SharePoint in his development environment.

Most of the material in this book works with the free SharePoint Foundation 2010; the examples in Chapters 10, 13, and 15 require the full SharePoint Server 2010 product.

For development purposes, SharePoint 2010 will run on the following operating systems:

- Windows Server 2008 R2 x64
- Windows Server 2008 x64
- Windows 7 x64
- Windows Vista SP3 or greater, x64

Notice that all the choices are x64 because SharePoint can't run in a 32-bit environment. This can present a challenge if your development environment is 32-bit today. Virtualization can help, as can the new boot to VHD option in Windows 7. Some SharePoint development shops host virtual servers and allow developers to connect with remote desktops. There are several options, but they all lead to the same place: x64 is mandatory.

Table 1.1 shows an inventory of tools to be installed in a SharePoint and Silverlight development environment. All of these are available either for free or as trial versions; those that aren't free are available under some MSDN subscriptions. Please note that the download links were current as of this writing but could change over time; the URL shortening service bit.ly does not allow updating the links.

**TABLE 1.1: Tools for a SharePoint and Silverlight Development Environment**

| Needed for | Tool | Download |
|---|---|---|
| Both | **Visual Studio 2010** Visual Studio is the common development tool for both SharePoint and Silverlight. Be sure to install at least Service Pack 1 as well. | http://bit.ly/SPSL_VS2010 |
| SharePoint | **SharePoint Server 2010 or SharePoint Foundation 2010** Note that SharePoint Foundation is a subset of SharePoint server. The examples in Chapters 10, 13, and 15 depend on the full SharePoint Server. | http://bit.ly/SPSL_SharePoint |
| SharePoint | **Visual Studio 2010 SharePoint Power Tools** These tools make it easier to develop sandboxed solutions for SharePoint. Some of the examples in this book make use of the Visual Web Part (Sandboxed) template that is included. | http://bit.ly/SPSL_PowerTools |
| Both | **Silverlight Web Parts for SharePoint (VSIX)** This adds two templates for building Silverlight web parts and requires the Visual Studio 2010 SharePoint Power Tools. Some of the examples in this book make use of the Silverlight Web Parts for SharePoint. | http://bit.ly/SPSL_VSIX_WebParts |
| Silverlight | **Silverlight 4.0 Tools for Visual Studio or Silverlight 5.0 Tools for Visual Studio** Visual Studio 2010 ships with Silverlight 3.0 support; the Silverlight Tools add support for later versions and are required to run the examples in this book. | http://bit.ly/SPSL_GettingStarted |

| Needed for | Tool | Download |
|---|---|---|
| Silverlight | **Expression Blend 4.0 or Expression Blend 5.0** Expression Blend is a visual designer for Silverlight and is used in several chapters. Blend makes some tasks much easier in Silverlight and is well worth using. | http://bit.ly/SPSL_GettingStarted |
| SharePoint | **SharePoint Designer 2010** SharePoint Designer is a tool for editing SharePoint content and is a free download. Some chapters assume you have SharePoint Designer available, although strictly speaking it doesn't need to run on your development machine. | http://bit.ly/SPSL_SPDesigner64 |
| SharePoint | **ADO.NET Data Services Update for .NET 3.5** This add-on is required to access SharePoint's RESTful interface as explained in Chapter 9. | http://bit.ly/SPSL_DataServicesUpdate |
| (optional) | A web tracing tool such as the "F12" developer tool included with Internet Explorer 9, or a download such as Fiddler or Nikhil's Web Development Helper. These tools allow tracing all requests coming from a web browser, including Silverlight network calls. Having one of them handy can help with troubleshooting. | http://bit.ly/SPSL_Fiddler or http://bit.ly/SPSL_WebDevHelper |

## Setting Up Your Environment

There are a few strategies for setting all this up in a development environment.

The first and surely the easiest approach is to simply download the whole environment with everything preinstalled as trial versions, which can be updated with your product keys from MSDN or elsewhere to allow

them to run indefinitely. This is available at http://bit.ly/SPSL_2010VM. With this approach, all you need is the ability to run a Hyper-V virtual machine image. All the examples in this book have been tested on this virtual machine.

If you're interested in doing your development on a Windows 7 machine, you might want to start with the SharePoint 2010 Easy Setup Script, which can be downloaded at http://bit.ly/SPSL_EasySetup. This amazing script will download and install on a Windows 7 machine trial versions of nearly everything on the list just provided; the only missing items are the Silverlight Web Parts, ADO.NET Data Services, and a network tracing tool such as Fiddler or Nikhil's Web Development Helper. It's easy to configure the script to plug in your license keys if you don't want to run trial versions, and it can be adapted to crank out custom developer workstations for your shop.

The Easy Setup Script can install these tools directly on a Windows 7 x64 machine, or it will set up a Boot to VHD environment for you. Windows 7 is capable of booting directly to a virtual hard disk (.vhd) file; if you provide a .vhd with Windows 7 x64 installed, the script will install all the developer tools on that as an option.

Please note that the Easy Setup Script expects a clean and fully patched version of Windows 7. Before running the script, ensure none of the tools it installs are already installed and that you've run Windows Update to install all service packs and updates for Windows.

### Installing SharePoint "From Scratch"

If you want to build a SharePoint development environment on your own, it can be a helpful learning experience and give you complete control over the configuration. For example, you might want to run SharePoint as part of an Active Directory domain; this should be done before installing SharePoint. This is helpful for working with user profiles, which can be synchronized with Active Directory, and is the most secure and easiest way to set up a multi-server environment. The virtual machine download mentioned earlier is already configured as its own domain controller; you can do this as well or join an existing domain prior to installing SharePoint.

All the products in Table 1.1 are easy to install and require no special instructions, except one, and that's SharePoint itself. This section shows you how to set up SharePoint 2010 on a clean Windows Server 2008 R2 machine as a single-server SharePoint "farm." Your production and staging environments may well have several servers, but that's beyond the scope of this book.

Begin with a clean Windows Server 2008 R2 computer and install all available service packs and Windows updates. It is not necessary to configure any roles on the server; SharePoint's Prerequisite Installer will take care of that and will correctly set up Internet Information Services (IIS) and other dependencies for you.

It's best to set up at least one service account to run SharePoint services; many developers run SharePoint under the Administrator account, which can lead to problems in production if they write code that depends on administrative privileges. If your SharePoint server will be part of a Windows domain, make the service account be an ordinary domain user. In production and staging, many service accounts can be used to create a "least privilege" set-up; here we are content to avoid running everything under administrator privileges.

In addition to a service account, you also need an installation account that has administrator rights on the SharePoint server. This can be the same as your developer account, which needs server administrator rights to run the debugger.

SharePoint 2010 needs SQL Server 2005 SP3 or later in order to run, and the SharePoint installer will set up SQL Server Express for you unless you install your own SQL server. If you prefer the improved management features and storage capacity of real SQL Server, the developer edition is a better choice. So before installing SharePoint, install SQL Server yourself, ideally running under its own service account (which can be another ordinary domain user). You also have the option to use an existing SQL Server instance on another server in your environment—just remember that you need to give the SharePoint service account some pretty high privileges, and that SharePoint creates a number of databases, so a separate SQL instance for each developer is often preferred.

Open SQL Server Management Studio to give the SharePoint service account the permissions it needs. Under Security right-click Logins and add a new user. The new login dialog box is displayed, as shown in Figure 1.1.

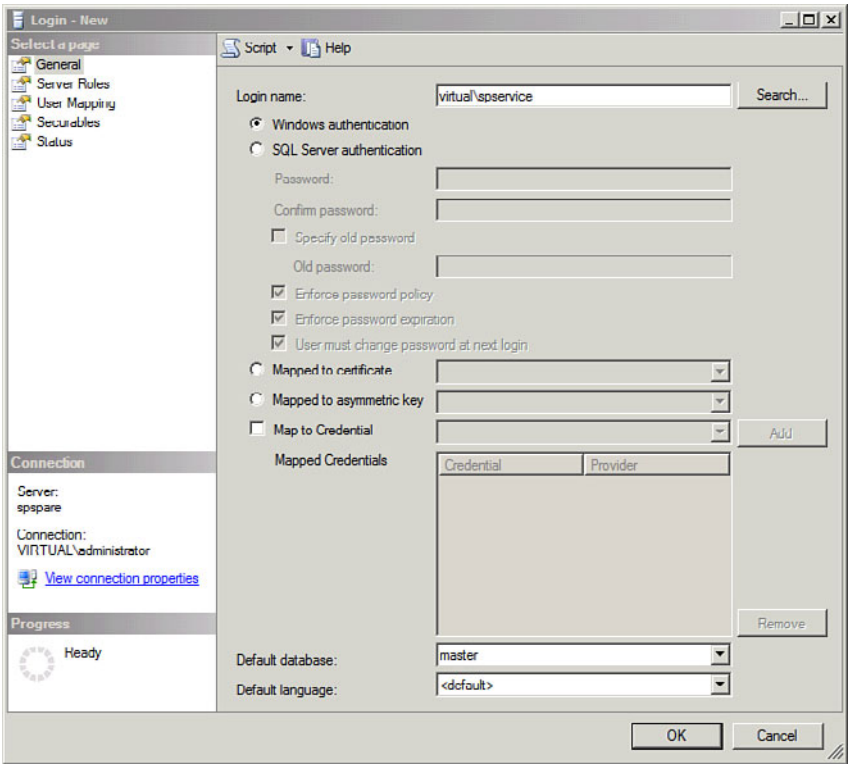



**FIGURE 1.1: Adding a login for the SharePoint Service account in SQL Server**

In the screen shots, the service account is called SPService and is in the Virtual domain. Before clicking OK to create the new login, click the Server Roles option on the left and grant your service account the dbcreator and securityadmin roles so it can create and secure new databases when the configuration runs, and when running the SharePoint Central Administration web site. This is shown in Figure 1.2.

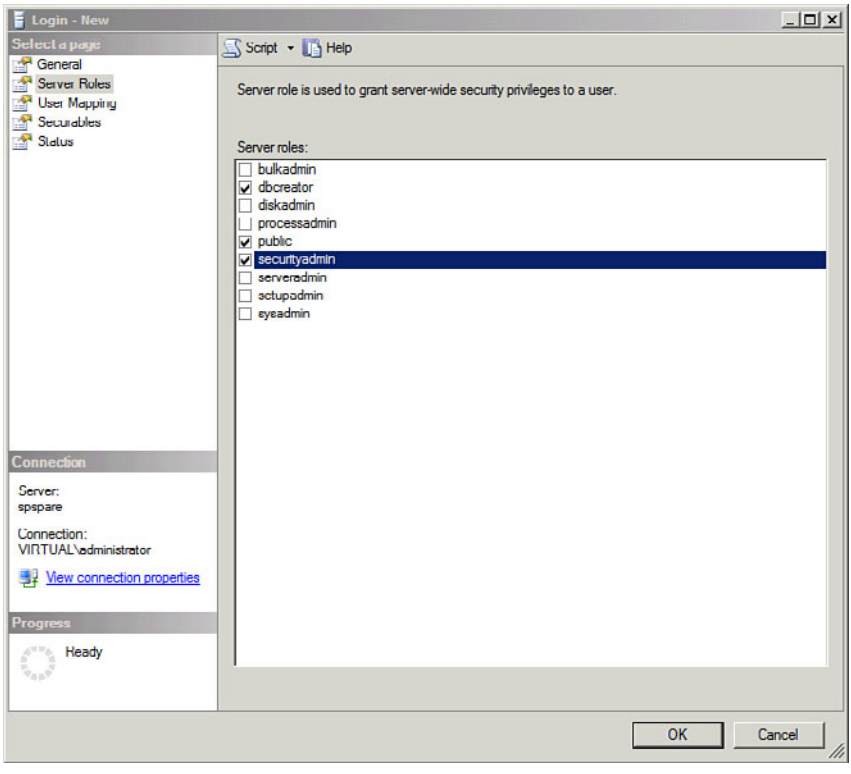



**FIGURE 1.2: Granting server roles to a SharePoint service account**

With SQL Server set up, it's time to install SharePoint. When you install (or virtually mount) the installation DVD, you see the splash screen shown in Figure 1.3. The screen shots are for full SharePoint Server 2010; the SharePoint Foundation installation is similar.


**Figure 1.3: The SharePoint installation splash screen**

If you like, you can review the links under Prepare to learn more about SharePoint installation. When you're ready to install, click Install software prerequisites under the Install heading. This leads you through a wizard that shuts down any conflicting services, asks you to agree to license terms, and then installs SharePoint's prerequisites. In addition to server roles, this includes the Windows Identity Foundation (for claims security), chart controls (for rendering reports), and even the Microsoft Speech Server runtime (for phonetic name searching). There isn't much you have to do here except provide an Internet connection because the prerequisite installer will download the latest versions of the prerequisites as it runs.

When the prerequisite installer completes, reboot the server if you're directed to do so and then begin the installation again. This time click Install SharePoint Server.

You are then prompted for a product key. The product key determines whether you're using the trial or full version of the product and also

whether you have the Standard or Enterprise edition. The Standard edition is enough for the examples in this book, but the Enterprise edition unlocks several great features you might want to try. You can always change to the Enterprise edition or graduate from trial to full product by entering a new key later in SharePoint Central Administration. If you need a trial product key, it's given right on the SharePoint download page.

After accepting license terms, you are offered a choice of a Standalone or Server Farm installation as shown in Figure 1.4.

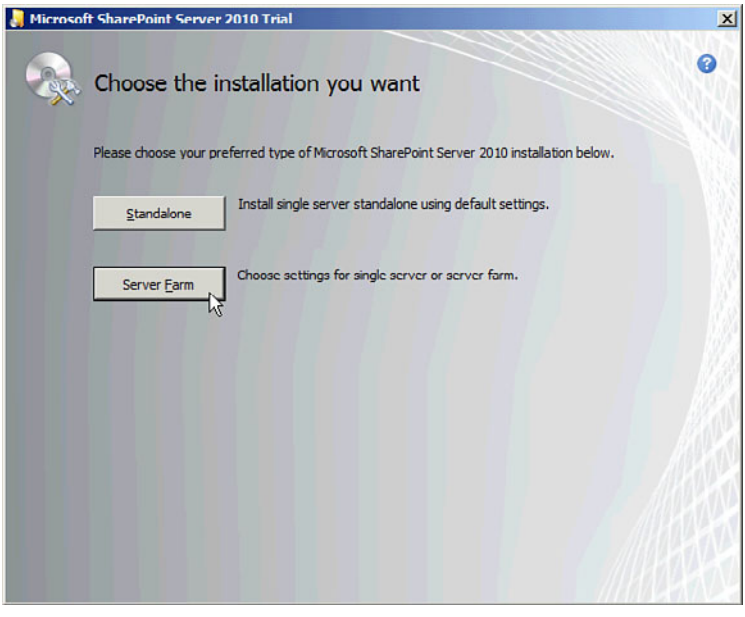



**FIGURE 1.4: Choosing the SharePoint installation**

Although you might be tempted to click the Standalone option, this will install SQL Server Express for you. If you want to use full SQL Server, either on your development machine or elsewhere, you need to select the Server Farm option. When you do this, you are asked to select the Server Type as shown in Figure 1.5.

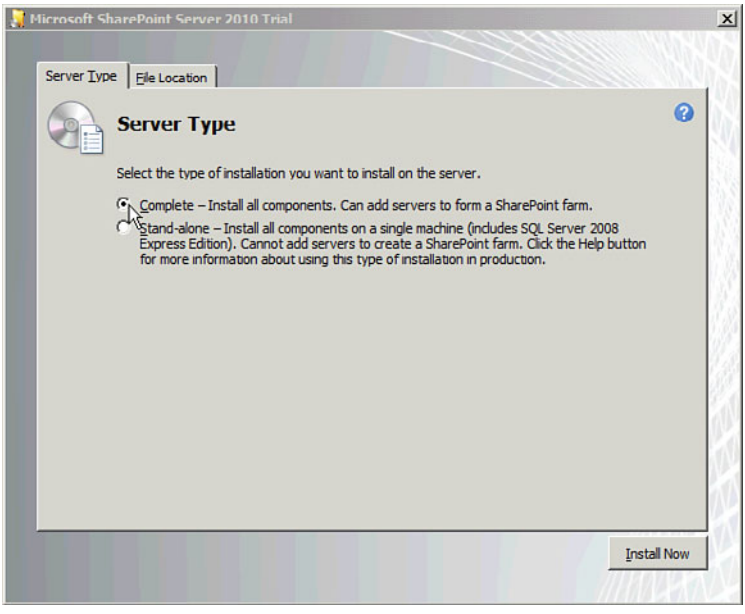



**FIGURE 1.5: Selecting the Server Type**

Because this is a single server farm, you need to select Complete so all of the SharePoint components are installed. Again, the Stand-alone option installs SQL Server Express Edition. Click the Install Now button and wait while SharePoint is installed.

When the installation completes, you see a checkbox to Run the SharePoint Products Configuration Wizard now. Go ahead and let it run, or if you want to do it later, you can find it on the Start menu.

The SharePoint Products Configuration Wizard creates the SharePoint databases and configures the server to run SharePoint. After a warning about restarting services, you are presented with the opportunity to connect to an existing server farm or to create a new one, as shown in Figure 1.6. For a single-server farm, of course, you want to create a new server farm.

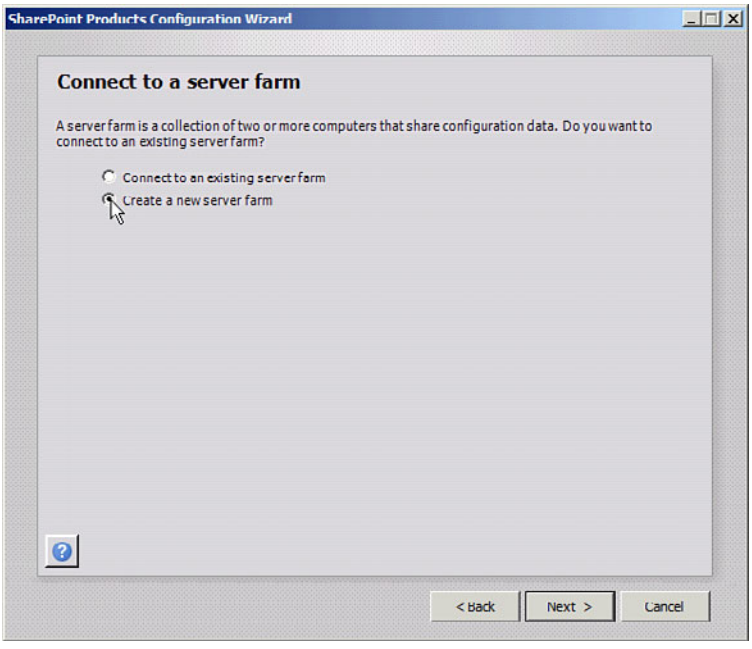



FIGURE 1.6: Creating a new server farm

Next, you are prompted to enter the database server name and the name and access accounts for the configuration database. The configuration database is at the center of the SharePoint configuration and contains information about all the other farm databases, server roles, web application, and other farm settings. Enter your SharePoint service account name and password as the access account; because you already gave it dbcreate and security admin rights on the database server, it is able to create the configuration and other databases. It is not easy to change the location of the configuration database after it's been set up.

Next you are asked to choose a Passphrase. This phrase is used to unlock configuration information, including service account passwords. Select a secure phrase you can remember.

After this, you are invited to create the web application for the SharePoint Central Administration web application. The wizard chooses a random port number; you can override it with a number you find easy to remember if you wish. Normally you want the default choice of NTLM authentication, but if you need to authenticate with Kerberos, that option is also available. Keep in mind this is only for the Central Administration web application and that you can select other types of authentication for your own web applications.

The wizard confirms your choices and then performs a number of configuration tasks, which can take a while to run. At the end, you should see a success screen and can click Finish to close the wizard.

You now have a working SharePoint farm with a Central Administration site but no web applications or site collections for users (or development testing). To set that up, visit Central Administration; a link can be found under Microsoft SharePoint 2010 Products on the start menu. The first time you visit Central Administration another wizard runs and offers to walk you through setting up the farm. Decide if you want to send feedback to Microsoft, and then click the Start the Wizard button to begin this last phase of setup.

Figure 1.7 shows the next screen, which allows you to set up the available SharePoint services. If you don't want to use a separate account, click the Use existing managed account radio button and select the same service account you've been using. If for some reason you don't want to run some services, uncheck them and click the Next button.

This might take a while; eventually the wizard offers to set up a site collection for you, which is your farm's first SharePoint site (other than Central Administration)! Give the site a name and description and select a site template. The standard Team Site template is fine for most tasks; Chapters 13 and 15 require the Publishing Portal template on the Publishing tab.

Your reward, as shown in Figure 1.8 is a working SharePoint site, ready for you to start testing your development work!

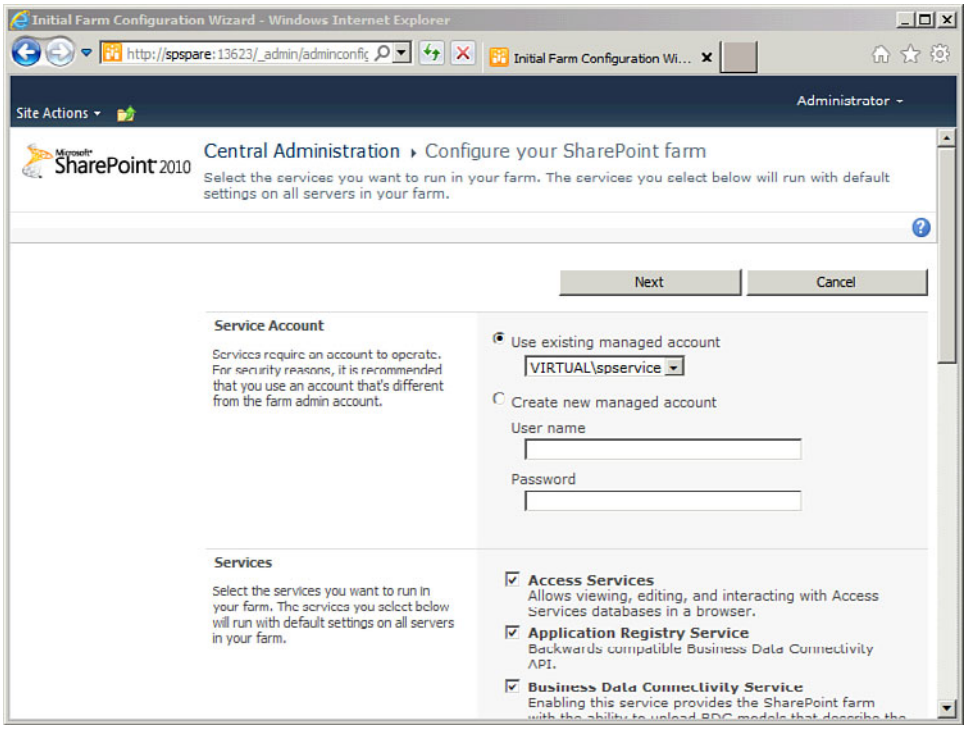



FIGURE 1.7: The Central Administration Wizard configures SharePoint services

Now that SharePoint is set up, go down the list in Table 1.1 and install everything else. The other product installations are straight-forward; just be sure you include the SharePoint and Silverlight portions of Visual Studio 2010 when you set it up.

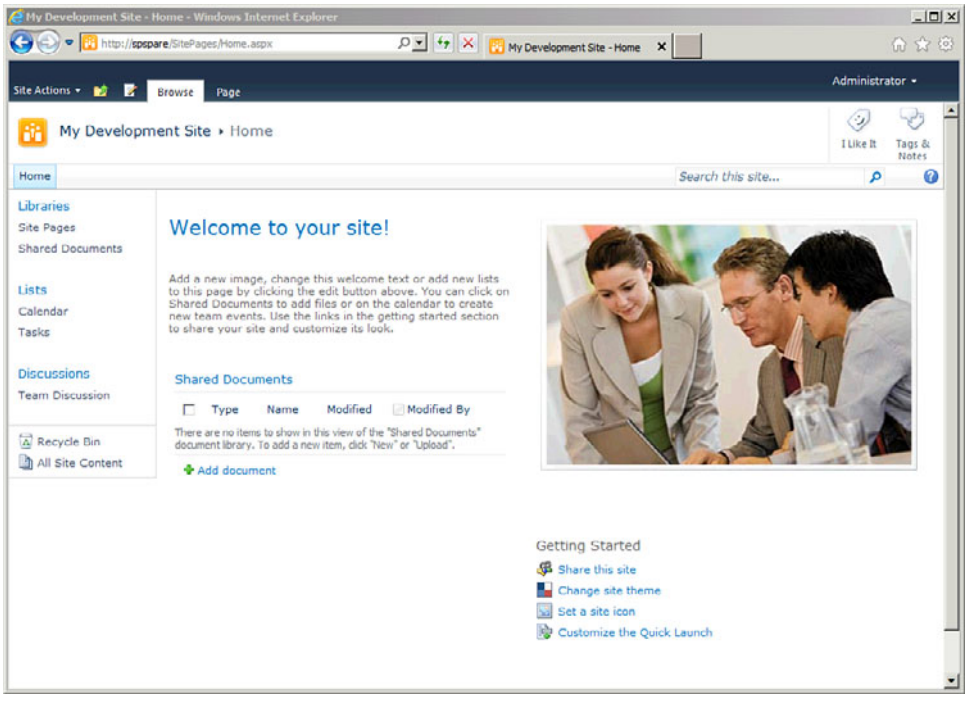



FIGURE 1.8: Your SharePoint farm's first team site

## Summary

Whether you downloaded the pre-built virtual machine, used the Easy Setup Script, or built it all by hand, you should now be ready to begin developing SharePoint solutions. The chapters that follow show you how to do quite a lot with SharePoint and Silverlight in your new development environment.

Have fun with it!

# 2

# Introduction to SharePoint Development

With the introduction of SharePoint 2010 and Visual Studio 2010, SharePoint development is easier than ever. This chapter introduces the concepts you need to get started with SharePoint development and leads you through creating your first SharePoint web part in Visual Studio. It is intended to help new SharePoint developers be successful with this book, rather than as a comprehensive guide to SharePoint development.

This chapter shows you

- How content is organized in SharePoint and how SharePoint combines web server files with content and customizations stored in SharePoint content databases
- How to build and deploy SharePoint web parts, both as "Visual" Web Parts which are based on an ASP.NET User Control, and code-based web parts based on ASP.NET Web Part objects
- How to create SharePoint lists and libraries in Visual Studio for use in your project

- How to access lists and libraries using the SharePoint API and LINQ to SharePoint
- How to intercept changes to lists and libraries using event receivers
- How to package your projects with SharePoint Solutions and Features

The first thing to remember is that SharePoint is an ASP.NET application. The web parts you see on the page are ASP.NET web parts, and they are part of an .aspx page just like any other ASP.NET application. The biggest difference is that whereas an ASP.NET site stores its pages in the web server file system, SharePoint stores pages in content databases.

## Understanding SharePoint Content

Figure 2.1 shows a SharePoint content database. One or more *site collections* are inside, each with a top-level *site* and possibly a hierarchy of child sites, grandchild sites, and so on. As the name suggests, a site collection is a collection of SharePoint sites that are stored and managed together. Each site contains *lists* and *libraries* which hold all sorts of information. SharePoint announcements, blog entries, contacts, and tasks are all stored as lists. A threaded discussion is a list of postings, and a calendar is a list of events. Libraries are lists that include a binary file integrated with each item, so for example there are document libraries, picture libraries, and media asset libraries. The web pages you see are stored as content too, either in a Pages or SitePages library or as free-floating .aspx files, depending on the features enabled in your site.

Placing all the content in databases makes it easier to manage and update and to share the content among a group of web servers. SharePoint supports flexible "farms" of servers working together, and traffic is often load balanced across multiple web servers.

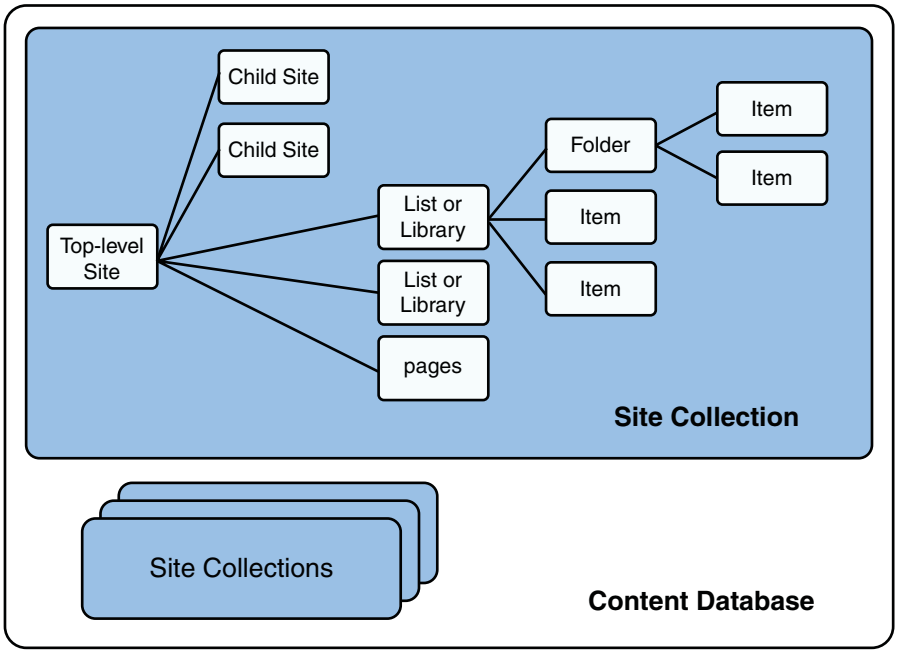



**Figure 2.1: SharePoint content**

There's a problem, however, with simply storing all the pages and files in every SharePoint site within a content database: Much of the content in each site is redundant, such as the pages used for site administration, to display and manipulate lists, and so on. Storing all this duplicate data for every site would be wasteful and would make it hard to update the pages globally. On the other hand, if a single set of pages were shared by all sites, it would be hard to customize them for any individual site.

SharePoint manages to balance both of these, sharing central copies of page and file templates among many sites unless they are customized using SharePoint Designer 2010, a free SharePoint editing tool. You might hear this sharing called *ghosting*, a term that dates back to earlier versions of SharePoint. A physical file was "ghosted" into the site URL space until it was "unghosted" (or "customized"). This haunting terminology was deemed confusing to end-users, so now we say that a file is uncustomized until it is customized; you can decide which is clearer!

Figure 2.2 shows page and file templates stored on the file system of each web server. SharePoint ghosts them into an apparent file structure accessible by a URL within each site. For example, given two newly created sites from the same site definition, http://server1/sites/site1/default.aspx is the same file as http://server1/sites/site2/default.aspx.

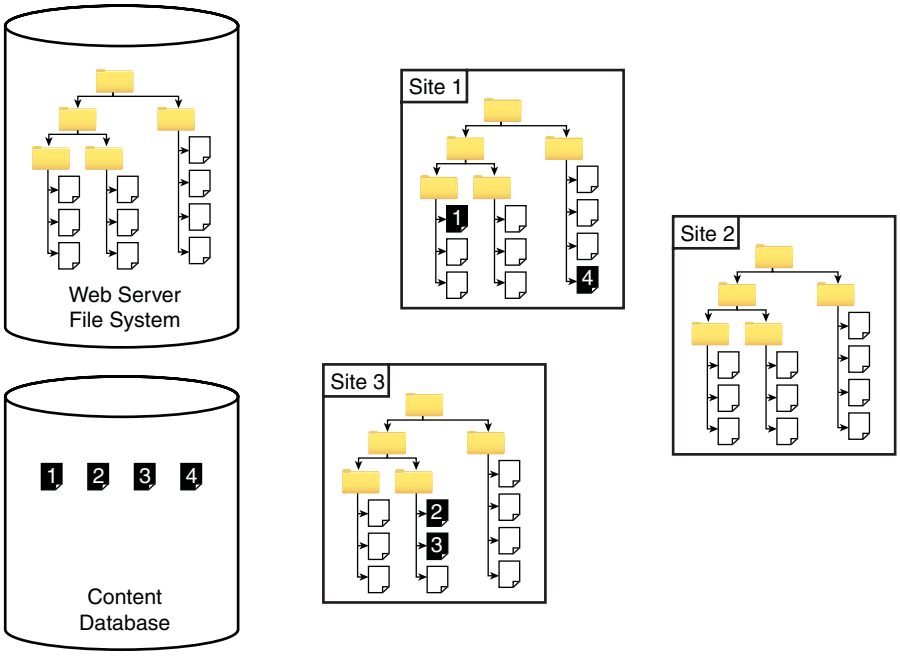



**Figure 2.2: SharePoint customization**

When users edit SharePoint pages in the web browser, they don't really edit the pages but only the content in the pages' web part zones, wiki fields, or other page fields. This avoids having to edit the original pages all together but doesn't allow changing their structure or editing their HTML.

To do that, SharePoint Designer 2010 is needed. In this case, the file is customized, and the changed copy is saved in the site collection's content database. The customized pages are shown as numbered pages in Figure 2.2. You can tell if a page has been customized in SharePoint Designer by the blue i icon that appears next to customized files as shown in Figure 2.3 and the Customization Status shown on the file information summary. If you want to revert to the initial file contents, simply right-click and select

Reset to Site Definition to uncustomize the file. A site definition is a set of files that is used to create new sites; this is where the original page templates are stored. Uncustomizing a file removes the customizations and allows the original template to show through.

By overlaying customized files from the content database over the baseline files on the web servers, SharePoint achieves the best of both worlds: It's possible to customize pages as needed without repeatedly storing the boilerplate files that are used most of the time.

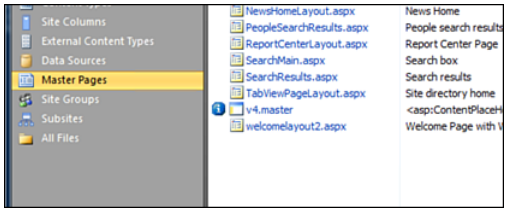



**Figure 2.3: A customized file in SharePoint Designer 2010**

## TIP

The best way to inspect the content in SharePoint is to use SharePoint Designer 2010, which is a free download available from http://bit.ly/SPSL_SPDesigner64. Use the file menu to open a SharePoint site. You will see more content if you choose the top-level site at the root of a SharePoint site collection, considering these sites store shared content that is used by all the sites in the collection, such as the Master Page Gallery.

SharePoint Designer presents you with a site summary page showing the name and location of the site, site permissions, and a list of child or subsites (if any). To the left of this, you can see a Navigation panel with shortcuts to the content most frequently edited with SharePoint designer. Click the All Files shortcut at the bottom of the list to see all the content in the site, with shared content from the file system seamlessly overlaid with content from the content database. Many of the files are set to use SharePoint's versioning system, with checkouts turned on and sometimes approvals as well.

This is the same content that your code sees when it runs in SharePoint and uses the SharePoint API. SharePoint Designer only works at the content layer; it never changes the file system.

A SharePoint developer needs to be aware of these two layers: the files installed on each web server and database contents that overlay them. In many cases, a given change can be made on either layer, with different consequences.

For example, suppose a developer wants to create a new master page to brand SharePoint's look and feel. Master pages in SharePoint are just like they are in other ASP.NET applications: they provide a common outer HTML structure used to brand a site, and they include elements that are common to all web pages. Master pages are stored in the Master Page Gallery, which is a document library in every SharePoint top-level site.

If the developer creates the master page in SharePoint Designer or uploads it into the Master Page Gallery using a web browser, it is created directly in the content database and is only visible to the site collection where it was created. If another site collection wants to use the master page, it needs to be copied into place, and two independent copies will be stored in the content database.

On the other hand, the master page could be created in Visual Studio 2010 as part of a SharePoint web solution package (.wsp file). Web solution packages are the vehicle for deploying customizations to SharePoint, and they correspond one-to-one with Visual Studio solutions. If the master page is in a farm-level web solution package, it can be installed on each web server's file system so all site collections share the same copy of the master page. Those copies can all be changed by updating the solution package—except in sites where someone used SharePoint designer to customize the master page (and thus saved a one-off copy in the content database).

Similarly, web page content is layered in SharePoint. Special pages, such as web part pages, wiki pages, and page layouts used in web publishing, are files ending in .aspx in SharePoint's content system. Whether ghosted from the file system or retrieved from a SharePoint content database, they are passed to ASP.NET for rendering.

These pages contain special ASP.NET controls that emit dynamic content on the page, such as the placement of web parts, text, and other elements that are edited in the web browser. The web part settings can be personalized by each user; for example, if it's enabled in a web part zone, users can rearrange and reconfigure web parts to personalize their view of

a SharePoint site. Thus, when you look at a typical SharePoint page, you're actually seeing the web page (including its master page), overlaid with web parts and other content residing in the content database. Although the web part code might be installed on each web server, the web part placement and metadata are treated as content. The user sees a seamless blend of web pages and controls that are installed on the web server, overlaid with controls placed on the page as content, possibly with customizations, as shown in Figure 2.4.

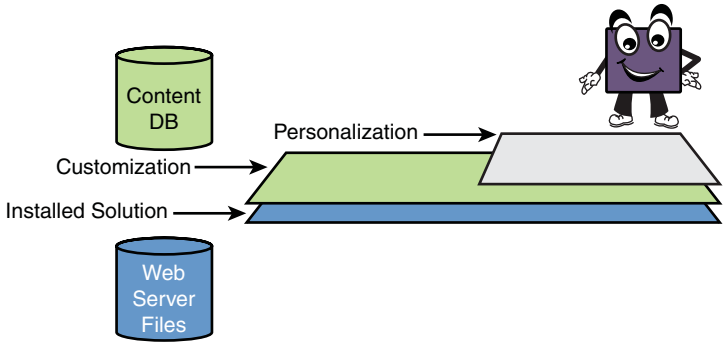



**Figure 2.4: Customization and personalization in SharePoint**

For more detail on SharePoint customization see http://bit.ly/SPSL_SPCustomization.

## Building a Web Part

A good first project for SharePoint development is to build a web part. Begin by setting up your development environment as described in Chapter 1, "Getting Started with SharePoint and Silverlight." Log in to your development machine as a server and SharePoint administrator and then open a web browser and visit the test SharePoint site you created at the end of the SharePoint installation. Then open Visual Studio 2010 and be sure to give it administrative privileges by right-clicking the icon and selecting Run as Administrator.

You might want to download the finished solution from http://code.msdn.com/SPSL/. If you're starting from scratch, click New Project … in the Visual Studio 2010 start screen, and the New Project

dialog box presents itself as in Figure 2.5. On the left, under Installed Templates click the SharePoint project template.

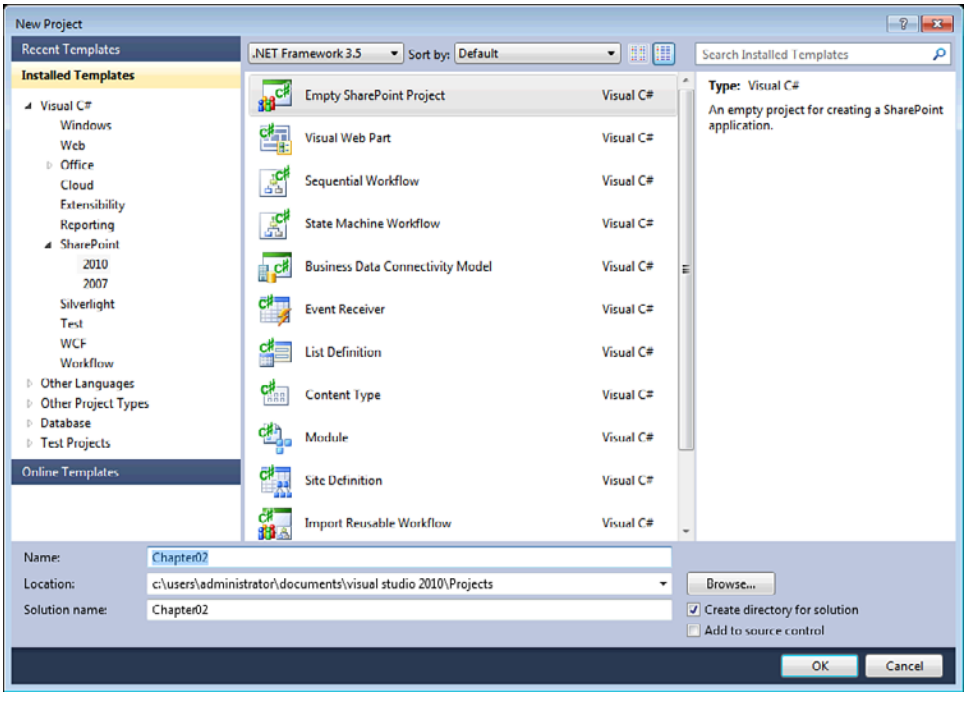



**Figure 2.5: Visual Studio 2010 New Project dialog box**

SharePoint 2010 uses .NET Framework 3.5, so be sure that's the version that's selected. For this project, start with an Empty SharePoint Project and enter a project name before pressing the OK button.

The next dialog box, shown in Figure 2.6, asks for the site you wish to use for debugging; enter your test site URL. It also asks you if you want a sandboxed or farm solution. This web part is a sandboxed solution, which means the entire solution is to be installed into the content database, and in many situations it can be installed by a business user. Because they're so easy to deploy and don't require the approval of a high-level administrator, sandboxed solutions run with limited trust. They can only use a subset of

the SharePoint API within the site collection where they are installed, and they can't call out over the network or gain access to the servers where they run.

Farm solutions, on the other hand, require the full trust of the SharePoint farm administrator because they can place files on the web server file systems and can install .dlls into the web application's bin directory or the global assembly cache.

Sandboxed solutions are the recommended choice when it's possible to work within their limitations. They can be deployed by a larger audience, and developers will appreciate that the debugger starts much faster after each build of your project, given Visual Studio only has to restart the sandboxed solution service rather than the whole web application as in a farm solution. After ensuring the Deploy as a Sandboxed Solution radio button is selected, click Finish to close the wizard.

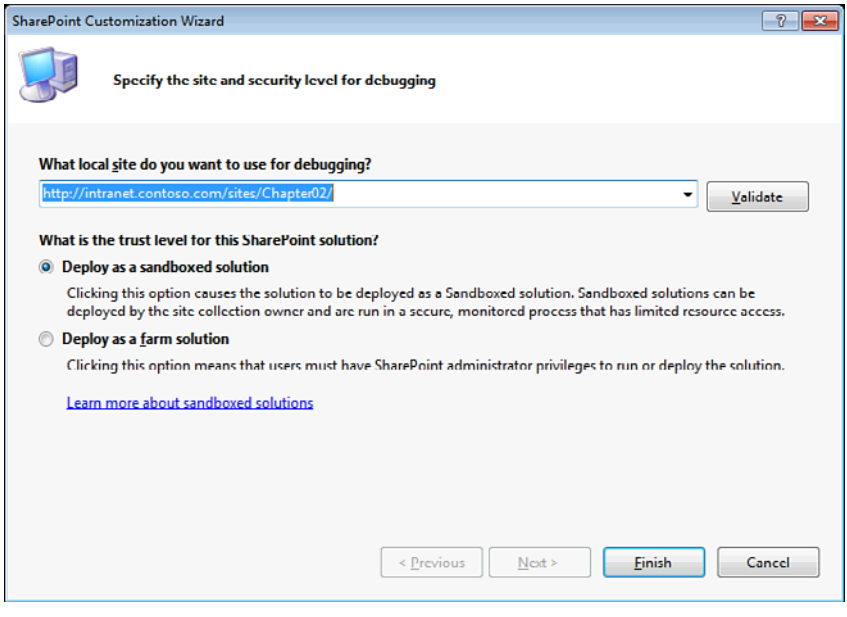



FIGURE 2.6: SharePoint Customization Wizard