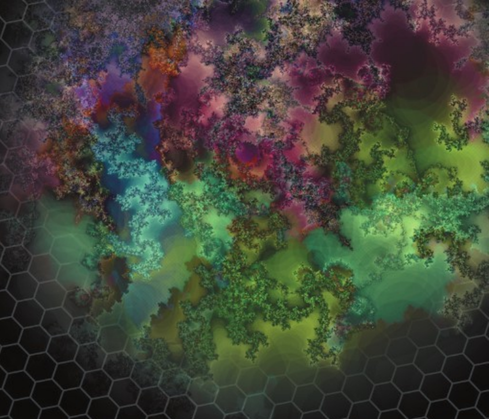
THE EXPERT'S VOICE® IN SHAREPOINT

SharePoint 2010 as a Development Platform

Discover how SharePoint is a powerful foundation for building custom collaborative business applications

Jörg Krause, Christian Langhirt, Alexander Sterff, Bernd Pehlke, Martin Döring

Apress®



SharePoint 2010 as a Development Platform

Jörg Krause, Christian Langhirt, Alexander Sterff, Bernd Pehlke, and Martin Döring

Apress®

SharePoint 2010 as a Development Platform

ISBN-13 (pbk): 978-1-4302-2706-9

ISBN-13 (electronic): 978-1-4302-2707-6

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

Contents at a Glance

Contents

About the Authors

■ **Jörg Krause** has been working with software and software technology since the early ’80s, starting with a ZX 81 using BASIC and assembler language. He studied information technology at Humboldt University Berlin but left to start his own operation in the ’90s. He has worked with Internet technology and software development since the early days of CompuServe and Fidonet. He has been working with Microsoft technologies and software since the time of Windows 95. In 1998 he worked with one of the first commercial e-commerce solutions, and during this time he also wrote his first book in Germany, *E-Commerce and Online Marketing*, published by Carl Hanser Verlag, Munich. Due to the amazing success, he decided to work as freelance consultant and author to share his experience and knowledge of technologies with others. Since then he has written for Apress, Pearson, Hanser, and other major publishers, on a total of more than 40 titles. Additionally, he has written articles for various magazines and spoken at many conferences in Germany, including BASTA, VSOne, and Prio Conference. Currently he’s working as a senior consultant for Microsoft technologies at Computacenter AG & Co. oHG in Berlin. He’s a Microsoft Certified Technology Specialist (MCTS) for SharePoint and a Microsoft Certified Professional Developer (MCPD) for ASP.NET 3.5.

In his rare spare time, Jörg enjoys reading thrillers and science fiction books, and playing badminton in the winter and golf in the summer.

■ **Christian Langhirt** has been working with information technology since the early ’90s. He began running his own software development business in 1998 during school. In addition to starting his own company, he studied information technology at the University for Applied Sciences Ravensburg-Weingarten. After that he received a master of science degree at Technische Universität München. Over the years, Christian has worked with a broad range of different technologies and is very familiar both with Java and .NET technologies. In the past years, he worked more and more with SharePoint, and he has used it as a development platform for modern intranet applications. Christian also speaks at conferences such as the German SharePoint conference. He currently works as a senior consultant at Computacenter AG and leads a highly skilled consulting team for Microsoft technologies. Christian cares deeply about his young son and his wife, and in his rare spare time he plays football passionately.

■ **Alexander Sterff** works as a consultant for Microsoft technologies at Computacenter AG. After many years of experience in object-oriented software development with Java and open source technologies, he started working with Microsoft products. Since then, he has participated in many different software projects, using his wide experience to guarantee their success. In the past years he has become an expert in building solutions that leverage the full potential of Office SharePoint Server, BizTalk, and InfoPath. Alexander received a bachelor’s degree in informatics and a master’s degree in information science, both at Technische Universität München.

■ **Bernd Pehlke** studied computer science at the University of Potsdam, Germany. He works as a software architect and technology consultant, focusing on Microsoft web technologies (ASP.NET and SharePoint). His main domain is enterprise business applications based on Windows SharePoint Services and SharePoint Server, especially conception and implementation of business process and integration scenarios.

He has spoken about business processes and integration using SharePoint and Microsoft Office technologies at several road shows and training courses.

Bernd is a Microsoft Certified Professional Developer (MCPD) for ASP.NET Developer 3.5 and Windows Developer 3.5, and a Microsoft Certified Technology Specialist (MCTS) for WSS 3.0/SharePoint 2007 configuration and application development.

■ **Martin Döringbegan** working with computers as a hobby in the mid-'90s, starting with writing code for MS-DOS in QBASIC and Turbo Pascal. Some years later he studied business computing and received a degree at the University of Applied Sciences Wildau (next to Berlin). After many experiences in software development with Java and open source technologies, his focus has moved to the .NET Framework—particularly web technologies and mobile devices.

Currently, Martin is working as a technology specialist for Microsoft technologies at Computacenter AG & Co. oHG in Berlin, applying his know-how in many different software projects across Germany.

When he is not in front of his notebook, Martin enjoys listening music at a concert or at home, or watching a movie at the cinema.

About the Technical Reviewer

■ **Frank Binöder**, MCTS, started his own film and IT business in 2001 while studying media informatics at the University of Applied Sciences in Dresden. He currently works as a SharePoint and Project Server consultant for Computacenter AG, and has over five years of experience engineering business solutions using Microsoft technologies. Frank has been dealing with the SharePoint and Project Server platform since the 2003 release. His passion is combining the functional requirements of customers with the creativity of developers. Frank also founded some of the local SharePoint user groups in Dresden and Hamburg.

Frank lives in Radebeul, near Dresden, Germany. When not working, Frank can be found riding mountain bikes with his son in the forests of Saxony.

Acknowledgments

We'd like to mention the people who helped us create this book and make it what it is. First, we'd like to thank David White for his amazing work smoothing out our style and cajoling it into readable English. We know you had a challenging task. We can't forget the support from Jonathan Hassell for our way into the Apress world, as well as the continuing help from Anne Collett. Thanks to our technical reviewer, Frank, who had a hard task getting our samples running within such a complex environment. His work greatly improved the quality of this book.

Last but not least, we'd like to thank our families and friends for understanding when we had to go back to the computer again and again to continue writing or installing SharePoint, from the first beta to the final release candidate.

Introduction

This With SharePoint 2010, Microsoft has shifted the developer experience toward a new paradigm—the paradigm of high-level development. This allows developers to extend Microsoft's software and adapt the parts to behave exactly as you would have designed them. However, things are not that clear on a closer look. SharePoint is multifaceted—it's an application, a platform, a server, a framework, and a database.

What Does This Book Cover?

This book is for experienced .NET and ASP.NET developers. It examines the SharePoint technology in greater depth than you'll find elsewhere, and it's full of practical tips and tricks from an experienced developer crew. You'll learn not only how things work, but also why. By adopting this knowledge, you will succeed in extending and adapting highly useful functionality in your own projects.

Imagine that your next customer wishes to run a couple of intranet tools, connect to line-of-business (LOB) applications, or customize beyond the limitations of the regular web UI and SharePoint Designer. SharePoint can handle this, but its out-of-the-box features won't be adequate. This book covers situations such as these in detail and shows you what to do when the SharePoint UI reaches its limits. You'll learn how to extend, customize, and enhance this platform to get what you want. "No more compromise" is our motto. That includes looking at SharePoint not just as a tool with some customization capabilities, but as a development platform and framework in and of itself.

Hence, programmability is covered in great depth. This book is for coders and real developers; it goes far beyond tinkering with XML and clicking aimlessly through an overcrowded UI.

Conventions Used in This Book

We understand that you're eager to start reading and learning, but it is worth taking a few seconds to look over this section—it will help you to get the most out of this book. Several icons and font conventions are used throughout the book:

- Screen messages, code listings, and command samples appear in `monospace` type.
- The same `monospace` font is used for HTML, ASP.NET controls (declarative listings), and XML snippets.
- Important parts of a listing are highlighted in **`bold monospace`**.
- All code-related terms in the body text of the book, including method names, class names, namespaces, and members, as well as URLs and path names, are set are set in `monospace` font as well.

Several icons highlight important definitions, cautions, and conclusions:

■ **Tip** This is a tip.

■ **Note** This is a note that explains a topic further, but is not required for understanding the main topic.

■ **Caution** This is a warning to keep you from common pitfalls.

Who Is This Book For?

This book is intended for advanced web developers interested in learning about how to use SharePoint as a development platform. We assume that you already have some experience writing web applications, you have created some web projects, and you have a basic understanding about ASP.NET, the .NET Framework, and related technologies.

We also assume that you already have some basic knowledge of skills and technologies often required as a web developer:

- HTML, CSS, and JavaScript
- Visual Studio basics, such as creating, running, and debugging a project
- Fluency with C#, as we use this language throughout the book exclusively
- ASP.NET basics, such as putting a control onto a page, customizing a control, and creating a user or custom control
- How to obtain database access, as well as how to use LINQ to query a database and write data back
- How to use XML as either a data source or storage

Prerequisites

This book is based on SharePoint 2010. As a basic platform, we use Visual Studio 2010 Ultimate running on 64-bit Windows Server 2008 R2 with IIS 7.5. When a client is involved, and when we run SharePoint in a client environment for development purposes, we use 64-bit Windows 7 Ultimate. (64-bit Windows is a requirement; you cannot even run the installer on 32-bit Windows anymore).

A similar platform is required for getting all the samples running. Nevertheless, we encourage you to look to the future and work with the most current tools and platforms you can obtain.

If you travel a lot and need to take your development environment with you, we suggest using a laptop with at least 4GB memory (8GB will be best if you can afford it) and Windows Server 2008 R2 x64 with Hyper-V installed. You can run several different development environments in virtual machines on such a computer. If you need external hard disk space, it's strongly recommended to use eSATA instead of USB. Connect to this machine using your regular laptop via remote desktop.

How This Book Is Organized

There are many ways of structuring a book. From our long-term experience in writing and publishing books, we know that different people read books very differently. Some read from beginning to end, just like a novel, while others start where they find an interesting topic. There is no book that can cover all reading styles; however, this book follows the same successful strategy we've used many times before.

We start with the basics: low-level concepts and background information necessary to *really* understand a topic. Then we proceed systematically through all the topics. This allows you to read from beginning to end, or to dip into an interesting chapter and skip the others. The many references in the book, pointing to chapters or sections where related parts are described in more detail, will help you get the information you need.

This book is full of code and examples, which are all available for download from the Apress web site (`www.apress.com`). Included with the package are subfolders named after each chapter (`chapter01`, `chapter02`, etc.). These folders contain several sample solutions or web sites in separate folders. Almost all examples are fully functional. Smaller code snippets that can't run on their own aren't included, to avoid confusion.

Support

We know that such a complex subject requires continuous support. We help the community by sharing our knowledge and expertise through our web sites. Feel free to visit our sites:

- `www.sharepointdeveloper.de`: The companion web site (in English; don't get confused by the `de` top-level domain)
- `http://blog.sharepointdeveloper.de`: The team blog, in English and German
- `www.aspnetextensiblity.com`: Jörg's ASP.NET site, in English

Several more sites driven by team members, colleagues, and partners are linked from there to give you access to more information. And if you want to have a look at Microsoft's official SharePoint 2010 site, visit `http://sharepoint2010.microsoft.com`.

Welcome to SharePoint Development

In this book we treat and express SharePoint as a development platform. This includes the fact that the applications developed on top of SharePoint must run in some kind of runtime environment. Having this in mind this lets you recognize SharePoint as an application server platform, too. It is, however, not an isolated piece of software. It is built on top of a broad range of interconnected technologies. One of these technologies is ASP.NET, a platform for creating web applications, and another is the .NET Framework, the underlying application platform, which consists of both developer capabilities and a runtime environment. SharePoint adds a bunch of functionality to these basic platforms. Applications designed for SharePoint are always built using the .NET Framework and quite often ASP.NET. They also make use of several services provided exclusively by SharePoint.

For developers, a deep knowledge of ASP.NET is essential. If you already have this, you'll have a head start into learning SharePoint development. Unless your organization is forcing you to use SharePoint, you might be struggling with the question of whether to use SharePoint as an application platform or stay with the ASP.NET platform. While ASP.NET is versatile and allows you to create powerful applications, there are some reasons that make SharePoint a viable alternative.

SharePoint is a technology that lets users create their own web applications without having to understand classic web site development. With SharePoint, rather than having to seek out a developer, users can now just talk to the database and server administrators, and start creating and deploying sites themselves. SharePoint provides various templates and features for modifying and customizing almost everything, from simple layout to the data structure held in lists. For the user, SharePoint acts like an application. Several tools accompany it, including Central Administration, SharePoint Designer, and the default site settings dialogs.

Developers can do even more with the SharePoint platform. SharePoint is extensible in many ways, and this extensibility gives developers access to almost all the internal modules. You can extend

SharePoint whenever a user cannot achieve a specific task with the embedded functions. Whether you need to make a slight modification or a large-scale one, such as adding an application page, you can extend the platform endlessly.

So, SharePoint is powerful both as an application platform and a developer platform. You can understand it in greater depth by looking at its main parts:

- *SharePoint Foundation*: Along with the other foundations, including Windows Communication Foundation (WCF), Windows Presentation Foundation (WPF), and Windows Identity Foundation (WIF), SharePoint bundles a collection of class libraries, runtime environments, tools, and support applications. The various tools address different roles, such as power users being supported by SharePoint Designer.
- *SharePoint Server 2010*: This is a product built on top of SharePoint Foundation that delivers a basic stack of features required to create an intranet- or Internet-aware application with little to no coding effort. It's a classic 20:80 ratio between effort and effect. Using SharePoint Server, you can create 80 percent of what an average site requires with 20 percent of the usual cost.

Using SharePoint as a development platform primarily involves SharePoint Foundation. However, you can develop on SharePoint Server as well. SharePoint became such a success worldwide because it allows you to reduce the risk of software project drastically by using it. SharePoint products provide what you need either out of the box or by extending the platform by coding.

SharePoint Applications

Ordinary users will be able to do lot of things with SharePoint by themselves (e.g., creating web sites, modifying the look and feel, adding certain features, and entering data), and developers equipped with at least basic SharePoint knowledge will be able to customize SharePoint further. This includes things such as adding a new menu item in site menus, creating Web Parts, adding code that invokes custom actions, and creating workflows beyond the built-in three-state limit.

Imagine that you're supposed to write a web-based application using ASP.NET, IIS, and SQL Server (as you may have done many times). Instead, you can use SharePoint Foundation to create your application, and you'll still have ASP.NET, IIS, and SQL Server at your disposal. Considering SharePoint as a development platform can only strengthen your development portfolio.

Figure 1 illustrates a general overview of SharePoint and its related technologies.

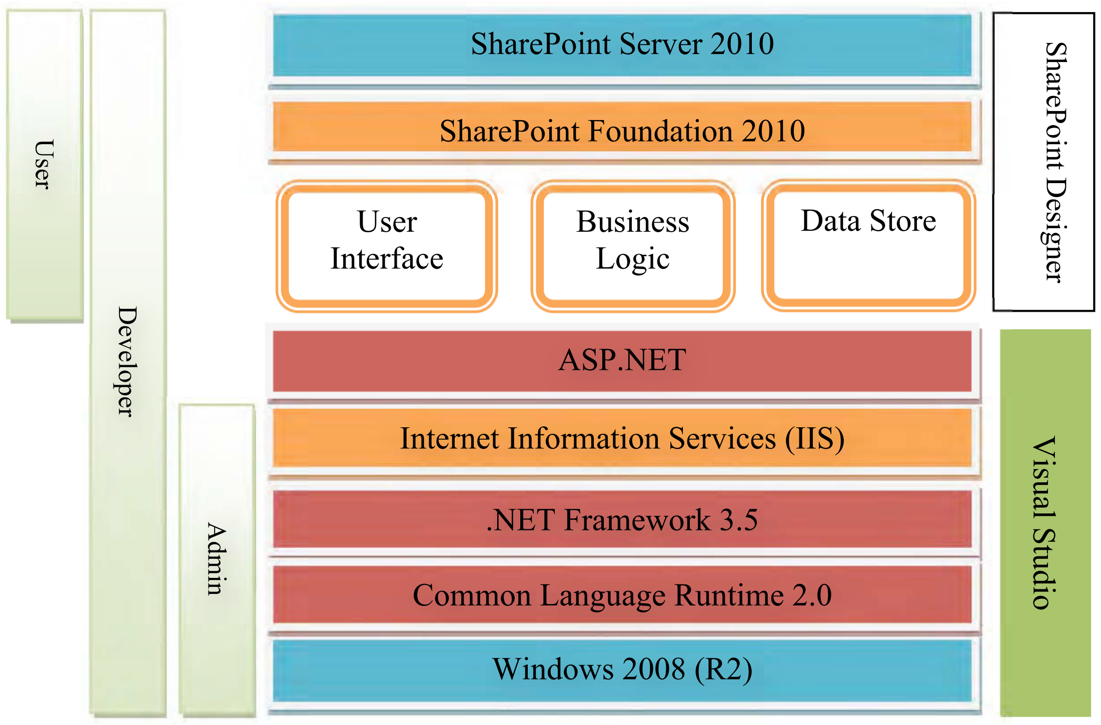


Figure 1*. A typical SharePoint application within the Windows Server stack*

The figure shows what a SharePoint application typically includes and how it relates to the Windows Server components. Users interact with some UI, the behavior is controlled by some business logic, and data is stored somewhere. Some development tools are used to empower both developers and power users. Additionally, SharePoint allows you to define three different roles: users, administrators, and developers. As you can see, the developer role spans all parts and is indeed the most demanding.

SharePoint is a platform that supports all of the following:

- The ability for administrators to maintain any installation, from a single server to a hierarchical farm
- A built-in way to work with data, including schemas created by end users
- The ability to create and execute business logic, including workflows
- A basic, easy-to-use UI, along with sophisticated customization features
- Visual Studio 2010 and a set of development tools that support everything from simple customization to huge team-based projects

While the developer support is not SharePoint specific, the other features are enhancements of the existing infrastructure. This is an important factor in considering SharePoint as a development platform.

The SharePoint Community

However, keep in mind that SharePoint is not just a huge piece of software. It is also part of an ecosystem of communities, information sources, consultant companies, third-party developers, forums, and books that provide support, help, training, and ideas. You can find people to communicate with, and meet to exchange ideas or discuss issues. There is a developer community, conferences exclusively devoted to SharePoint, MVPs, and authors you can hire. There are patterns and best practices available from Microsoft. While SharePoint may have a steep learning curve, it will be much easier if you don't walk alone.

Windows SharePoint Foundation for Developers

Before you start coding, you should understand what the pieces of the puzzle are. From one perspective, you can consider SharePoint to be built with tiers (although it's not really a multitier architecture behind the scenes, and the levels aren't as loosely coupled as you might like). But treating such a complex system as a collection of parts provides a well-structured way for you to understand it. From a developer perspective, we can identify six layers:

- The execution environment
- The data layer
- The business logic layer
- The UI layer
- The security layer
- The developer toolbox

Each of these layers is described in the following sections.

The Execution Environment

Executing a SharePoint application means using both the Common Language Runtime (CLR) and the ASP.NET engine. SharePoint itself comes with a bunch of services, along with tight integration with IIS using ASP.NET. In real-life applications, the complete scenario might appear more complicated. For example, in a farm with multiple servers, the execution environment will span several machines. While its logical structure still goes from farm to sites grouped into site collections, its physical appearance might be different. From a developer's perspective, the logical structure determines the objects you work with.

SharePoint provides a *site collection* as a container for sites. A site collection must have at least one site, the root site, which in turn can have child sites that run the applications. A site collection forms a multilevel hierarchy of instances that hold their own pages, lists, libraries, and individual users. SharePoint provides an inheritance mechanism that simplifies management of the sites. You can understand each site as a container of data, pages, and users working with that data. Each site can have its own administrator responsible for it and all its deriving sites.

The Data Layer

The sites that function as logical containers for data store that data in lists and libraries. Libraries are lists that have the ability to store documents or files. Libraries are the basic instances that make SharePoint act as a document management system.

Lists function similarly to relational database tables, but are much more versatile. While a table is a strongly typed container for rows of data, a list can be used more flexibly. Lists are, however, not containers for storing any sort of object regardless of its structure. Using content types to describe a lists schema, you can define the structure of the objects stored in that list. A list can be bound to many content types and therefore store different kinds of structured objects, the so-called list items. In real-life projects, this results in an object schema that imitates the real-world counterparts of the objects. For instance, when you need to handle offers, invoices, and packing slips for customers, a list called Customer Documents could keep all three document types together in one place. The variations in metadata between the individual document types would be too unwieldy to store in a relational database table. A database developer would suggest using four tables—three for the different types and one that joins them together. A SharePoint developer, on the other hand, would use just one list, extended with three content types. The relation is implicitly defined by this with no additional effort.

Additionally, lists are available through a UI that's powerful for both developers and end users. As a list's schema supports a data model that's closer in nature to an object than a relational database, it's easy for nondevelopers to use lists to create sophisticated structured data stores. Even working with list data is quite easy, because the UI for viewing, sorting, filtering, and manipulating data is available out of the box. Relationships between lists can be quickly and simply defined with a lookup feature, which forms a many-to-many connection. Complex fields can hold predefined sets of data contained in drop-down controls in forms. And the access control mechanism is available at the list and item stages, with no code required.

Internally, list querying is based on CAML (Collaborative Application Markup Language). The generated LINQ to SharePoint layer internally translates queries into CAML. Using CAML, both developers and users can create powerful selections of complex data.

The data layer is not limited to internal lists. You can even connect to external data from SQL server databases, web services, XML, or similar sources. Internally, that data appears in a similar format to lists. The external data is bound using Business Connectivity Services (BCS). However, this service is not a component of SharePoint Foundation; it's available in SharePoint Server only.

Exposing data to the outer world is another subject that's well addressed in SharePoint. SharePoint 2010 supports WCF Data Services for common REST-based access to any data stored internally.

The Business Logic Layer

If you use SharePoint as an application development platform, it's likely that you'll work on some sort of application logic as part of a development project. Such logic is usually encapsulated in modules that form a business logic layer. Technically, the logic makes use of the various built-in features, such as ASPX pages, web services, workflows, event receivers, and timers. Along with the .NET Framework, a huge toolkit is available to form the logic.

In simple projects it might be acceptable to spread the logic through the instances SharePoint provides—Web Parts, application pages, workflows, and event receivers. Such projects generally have only one or two of these instances and remain easy to maintain. However, from the perspective of professional software development and application life cycle management, a different strategy is needed. It makes sense to create a unique business layer that handles all relevant logic in one or more assemblies. The instances where the logic gets triggered should have interfaces only. That means you have to take care of your project's structure and the ways the data flows between the instances.

SharePoint itself does not come with a part that is dedicated to the logic. Moreover, the internal logic is spread over several parts, and cannot be found in one place. You should be aware of this, and instead create a dedicated business logic layer for your SharePoint application.

The User Interface Layer

The most sophisticated business logic is unusable if the user cannot exploit it easily. The UI layer is responsible for providing a modern and accessible way to interact with your application. SharePoint includes a browser-based engine and a web-based application. This has some nice implications, such as zero deployment and a broad range of clients, but it has some disadvantages compared with Windows applications as well (mostly concerning the limitations of HTML and browser environments). SharePoint defines basic UI elements and conventions for how to use these elements.

While you may be tempted to customize SharePoint's default design, we recommend against departing from it. Consider that the preferred way to extend SharePoint is through Web Parts. Web Parts can only appear in certain zones within a Web Part page, and users can arrange, close, or resize them. The available options don't change the UI—and neither should you while creating it. Treat regular application pages as part of an existing system.

Microsoft also recommends against creating new master pages, and instead recommends that you only change the basic layout and/or replace the company logo. This is something of a paradigm shift, since ASP.NET developers are used to having total freedom when designing their application's UI. But why should you accept these restrictions?

Primarily, many users have experience with the standard UI, and changing it could be potentially confusing for them. Likewise, you should anticipate new employees to arrive from other companies with SharePoint experience. If your intranet looks radically different from the standard, their prior experience won't be of use in your organization.

Silverlight Integration

The power of the SharePoint UI comes through its amazing collection of controls. You can use these to create similar pages with little effort. However, keep in mind that even the most sophisticated controls render as HTML and JavaScript, which have some limitations and pitfalls. In order to mitigate these, Microsoft introduced another paradigm shift with Silverlight—a new, small, autonomous framework that is mostly compatible with (but not identical to) the .NET Framework, and is available as a browser plug-in. Developers can create a GUI with animations and vector objects, based on the well-known XAML language that drives WPF (Windows Presentation Foundation) already. Using Silverlight, you can create ambitious applications that match users' needs without bending the existing UI. Silverlight's tight integration with SharePoint via the client object model makes it a serious alternative to HTML.

The Security Layer

Any security layer consists of two basic actions: authentication and authorization. Authentication identifies users by some combination of name and password, a token, or a smart card with a PIN (personal identification number). SharePoint has a comprehensive security model, and there is no need to reinvent the wheel. SharePoint makes use of IIS for authentication purposes, which in turn uses common data stores such as Active Directory. The authorization module itself is based on ASP.NET membership providers that can be replaced, customized, or enhanced using common techniques.

Once the user is authenticated, the security layer is responsible for informing other modules what the user is authorized to do or what data he or she is allowed to access. SharePoint comes with three predefined groups to handle users in categories. These groups can be extended by assigning common roles for data access. The security settings inherit from one level to another, such as from a site collection to its child sites. However, you can break the inheritance and choose another security model on any level; none of this configuration requires writing code. The security model works down from the farm level to the individual list items. For coders, the security model is available through particular classes. Specific base classes for application pages are available for use with the built-in security model or for allowing anonymous access explicitly.

Whatever you need to implement, you can extend and customize the ASP.NET security model that SharePoint is built on.

The Developer Toolbox

The toolbox is filled with everything you need to customize, extend, and adopt almost any part of SharePoint Foundation. This includes dedicated and exclusive tools such as SharePoint Designer, as well as common tools such as Visual Studio 2010, which comes with a couple of preinstalled project and item templates. There are several tools, notably `stsadm` and the PowerShell cmdlets, which support scripted administration and deployment processes. Also included is the SDK (software development kit), which consists of manuals, guides, step-by-step-instructions, samples, and references.

Because of SharePoint's two-pronged appearance, it's not a pure developer environment. While some other platforms may put all the power into the developer's hands, SharePoint also caters to end users and administrators with its tools. Consequently, for certain tasks, developers will need to use SharePoint Designer, the web UI, or Central Administration, as there may be no developer-specific alternative. In this way, your toolbox is not limited to Visual Studio and its relatives—it encompasses everything that comes with the standard installation. We strongly recommend that you treat SharePoint not just as couple of classes and controls, but also as your favorite platform for accomplishing daily tasks. SharePoint is used by Microsoft as a platform for Team Foundation Server administration, as the interface used by Project Server, and as the heart of the Office platform. This indicates that you're already in two roles—developer and user.

Microsoft SharePoint Server for Developers

SharePoint Server 2010 is everything SharePoint Foundation is. You can see SharePoint Foundation as a functional subset of SharePoint Server. Additionally, many high-level features have been added to SharePoint Server. The following list shows the areas that SharePoint server deals with exclusively:

- Content (document related)
- Search (mostly document related)
- Dashboards (data related)
- Forms and workflows (mostly data related)
- Community (people related)
- Content publishing (mostly people related)

We explain these subjects in the next few sections in more detail.

Content

Enterprise content management (ECM) is in the 21st century what file shares were in the past. The overwhelming quantity of documents and the various workflows that manage the ways people create, approve, edit, and update these documents create challenges for ECM systems. Content management systems consist of various built-in services, storage capabilities, and features to help developers focus on business logic and workflow creation, leaving infrastructure tasks to be completed by SharePoint.

Records management is another advanced subject in the field of ECM; it addresses the need for a reliable and auditable document store. Using ECM records management, it's possible to create custom solutions on top of the basic modules SharePoint Server provides.

Search

Document and enterprise management systems have been used to organize content in various ways, including taxonomies, category trees, and tagged documents. Since Google's success, we know that full-text search is a very user-friendly organization strategy. With the increasing number of documents found on file shares in companies around the world, search capabilities have become standard.

SharePoint's search capabilities are on par with this standard, offering you highly customizable and extensible ways to create sophisticated full-text search functionality.

Dashboards

The ability to access timely, pertinent data from many sources has become a necessity for company leaders. Knowing what's really going on in an enterprise is essential for managing a company well. To address this, SharePoint provides Web Parts that display key performance indicators, chart controls that express complex data simply, and business connectivity services that gain access to LOB systems quickly. SharePoint now acts as business dashboard.

Particularly impressive in this regard is the Excel services feature—available as a browser application—which extends the Web Part–based view into a full-blown spreadsheet environment. Spreadsheets are stored in document libraries, making them available under central rights management to the targeted audience.

Forms and Workflows

Forms management is part of almost every enterprise's intranet. From simple employee self-service to absence management, travel expense reimbursement, and internal orders for equipment, there are many applications that benefit from easy-to-create, flexible forms. Combined with workflows, forms management allows you to express internal processing through the SharePoint technology and automate daily tasks. Both forms and workflows in SharePoint are highly customizable, programmable, and well supported by additional tools.

Community

Blogs and wikis are available in SharePoint Foundation already. For most applications, these are the entry points into community support within an enterprise. SharePoint server adds some features that address a broader audience. They make social computing available to teams as well as entire enterprises. One core community-based feature is MySites, which allows employees to introduce themselves and express their thoughts in a managed and centralized way.

Content Publishing

Blogging allows users to express their ideas quickly in a relatively unstructured way. It complements the full-text search engine paradigm described previously. Adding and updating articles on a wiki is more structured, and complements the link paradigm that the whole Web is built on. Giving the ordinary user the power to create pages without knowing anything about HTML empowers common people to take part in creating content in a more powerful, less structured way. SharePoint's ability to allows users to create sites, blogs, and wikis has in part pushed it to its current level of popularity. While SharePoint Foundation offered these capabilities already, SharePoint Server's Publishing feature takes them even further. The larger an enterprise's sites and the greater its activity, the more it will benefit from content-publishing modules.

Applications Make the World Go Round

SharePoint has been recognized and widely accepted for its support for certain applications, including

- Business collaboration applications
- Portals for LOB applications
- Web Parts solutions

Collaboration features are an essential part of SharePoint—it only takes a few clicks to add collaboration capabilities to your application. Workflows and forms bring things together with little effort, and business applications with some specific SharePoint features added (what we call *business collaboration applications*) are commonly used.

Creating a portal for a LOB application is just like creating other types of applications. The use of built-in features such as the Dashboard Web Part, Business Connectivity Services, and integrated web services are common in such types of applications. Your application can add the requisite data collection services, filtering and sorting capabilities, and smart data displays using chart controls, and can employ rights management.

SharePoint also supports the creation of Web Part solutions, in which you create various Web Parts that you deploy to an existing SharePoint environment and let people use your modules to extend their pages. Web Parts can be complex, and the data connections between them can empower you to build a universe of solutions.

SharePoint brings developers to a new level of programming with a high-level foundation, an overflowing toolkit, and a base application that's already there before you start coding. Some people see SharePoint as the first platform of the future of coding. While it's not the only one out there, it's one that fits well into the existing development landscape.

PART 1

Basic Tasks

CHAPTER 1

Developer Basics

In this chapter you'll learn everything you need to start developing on the new SharePoint 2010 platform. In particular this chapter will explore in detail the primary development tool: Visual Studio 2010. You'll learn the strengths and weaknesses of SharePoint Designer 2010 and how it compares with Visual Studio.

The topics covered in this chapter are

- SharePoint developer support
- Debugging your code
- Installation scenarios and how to set up your development environment
- Visual Studio 2010 and the SharePoint tools
- Developing on 64-bit computers

Not all developer tasks can be accomplished using visual tools. The SharePoint SDK (software development kit—for SharePoint it's a collection of help files, manuals, and examples) is a great collection of tools, code examples, and documentation. This chapter will introduce the SDK and highlight the most useful tools.

SharePoint projects tend to become large. Visual Studio Team System and the Team Foundation Server are good bases for team development. We recommend, for optimal output from multiple developers, that you establish a shared environment based on a SharePoint farm. You'll learn how to set up and use such an environment and how to incorporate desirable software development practices such as continuous integration and rapid deployment.

Before You Start

This book assumes that you're already familiar with SharePoint—at least SharePoint 2007—and its underlying technologies and platform. However, it begins by clarifying some matters that might be unclear even for an experienced developer.

First and foremost, SharePoint is an *application platform*. An application platform is a software development foundation that consists of an operating system, one or more frameworks, and interfaces that applications use to accomplish tasks. Primarily, an application platform has the user in mind. Microsoft SharePoint is an example of a very good application platform.

The platform is a reliable, reusable, and well-documented set of products, technologies, and tools. While some of these modules are highly usable out of the box, others may be replaced, customized, or modified according to specific needs. The platform components are dedicated to specific services they provide and that other components can consume.

An application provides business capabilities to its users via its components. A platform is classified as having a service-oriented architecture (SOA) if the components provide a standard way to communicate via services. The services use providers that act as a transparent layer between the underlying data source and the service. SOA has evolved predominantly by using XML-based web services. However, these services are not limited to XML, and modern platforms may support many other data transmission standards. An application platform provides many ways to access the services, and this is true for SharePoint. As a developer you not only consume such services, but also create services, customize existing ones, and install providers appropriate to the business cases.

The production environment for the SharePoint application platform consists of the following:

- Windows Server as the common server platform. For SharePoint 2010, Windows Server 2008 (64 bit) is the minimum requirement. As parts of the operating system, you have to have access to the following:
    - Microsoft Management Console (MMC) as the common server interface.
    - The .NET Framework and the underlying Common Language Runtime (CLR), as well as the additional libraries such as Windows Workflow Foundation (WF).
- SQL Server 2005, SQL Server 2008/2008 R2, or a corresponding version of SQL Server Express edition. SQL Server must be a 64-bit platform regardless of the actual version.

SharePoint and SQL Server

It's worth thinking about the usage of SQL Server in your SharePoint installation. SharePoint makes use of any SQL Server database server found while installing. If there is none present, it uses a SQL Server embedded database or SQL Server Express. If you install a standalone version and there is no database preset, setup chooses the embedded version. If you choose a farm installation—which is always recommended—setup will choose the SQL Server Express version. For any other serious production installation, you should have a SQL server somewhere else up and running.

Setting Up a Development Environment for SharePoint

There are several ways to set up a development environment. It depends on whether you're on a team or you're a single developer, and whether you work for an enterprise or a small shop. First, consider your own development machine. The whole process of coding, packaging, installing, and running a SharePoint component takes time.

Working Outside a Server

When developing for SharePoint, the first impression is often that a new 64-bit development machine is required to run the server and the development environment together. We'll examine this scenario and its advantages and disadvantages in the next sections. The simplest option is to keep your client machine and work against a remote server running SharePoint. It makes you independent of local configuration requirements and reduces your cost outlay, particularly for your first, simpler projects. In Chapter 2, we explain what types of development are possible with SharePoint. As a preview, the different outputs are

- Web Parts
- Application pages
- Custom fields
- Controls

A legitimate concern as a developer on a disconnected machine is, “Won’t my SharePoint projects require dependencies from SharePoint assemblies?” The most common development targets, Web Parts, have no such dependency. You simply derive your control from the ASP.NET `WebPart` class and use the SharePoint web services to access data. While this is not the most powerful tool set, it’s easy to set up and maintain.

For a more professional and versatile environment, consider using a remote development configuration.

Considering Remote Development

Whatever configuration you choose, you should try to develop on your own server machine. That’s the only way to get the Visual Studio 2010 *F5 deployment* feature. It makes your development cycles shorter and increases productivity.

For a team-working arrangement there are several extra prerequisites. SharePoint 2010 is a 64-bit–only product. That means your whole environment must run on 64 bits. If this is overkill, there are alternatives. We strongly recommend setting up a virtual server on a physical machine, such as Hyper-V or VMware on Windows Server 2008. Create your virtual development machines there and access them remotely. A cheap Windows 7 machine will suffice as such a client computer. SharePoint projects occasionally crash the server during heavy development. Re-creating a virtual machine is much easier than losing your whole personal computer.

Installation Scenarios

The following installation scenarios list the options you currently have to install SharePoint as a core component of a development environment. The 64-bit prerequisite limits the choice of operating system to the following:

- Windows Server 2008 x64
- Windows Server 2008 R2 x64
- Windows Vista x64
- Windows 7 x64

These descriptions are two-pronged, including instructions for the clients and the SharePoint server.

Developer Workstation on Windows Server 2008

The following list is the minimum needed for a working SharePoint development system. Depending on your specific needs, you may need to install additional components. The order is obligatory—meaning that SharePoint must be installed before Visual Studio. The help files are, of course, optional.

1. Install Windows Server 2008 x64.
2. Configure, at a minimum, the Web Server role.