PART 1

Introduction

This section will focus on describing what this book is about, its intended audience, and the technologies we will utilize in our solutions. It will introduce the common business challenges for which we will construct solutions in the remainder of the book. Part 1 includes overviews of Microsoft Office SharePoint Server 2007, Microsoft Office 2007, and Visual Studio Tools for Office.

CHAPTER 1

Office Business Applications

It is sometimes difficult to remember the corporate office of the recent past. Think back 5 or maybe 10 years. For many of us, that isn't long ago. However, from a technology perspective, we are talking about an era that might seem as distant as the dark ages. Sure, personal computing was taking off and the Internet was in its infancy. Not every company had a web site, and the average business user in a corporate office had very little exposure to the technologies that seem commonplace today. Remember when only the technically proficient used spreadsheets? Remember when email was a productivity tool and didn't require labor-intensive filing and sorting? This trip down memory lane lends some perspective as to how far workers have come to embrace the technology solutions IT offers. Unfortunately, the amount of information that workers have to interact with increases daily, as does the proliferation of often-siloed software systems trying to provide work-related solutions. Today, many organizations find themselves in a state where information and labor are duplicated. Often, workers take more time finding and constructing information than analyzing it and making decisions. It is here that technology has an opportunity. The opportunity is to provide more-intelligent tools that focus on the work the corporate business user needs to accomplish. It is in this area that this book will explore common challenges and scenarios and their solutions.

Companies often believe the average business user isn't going to find new technologies or solutions easy enough to use. Though there is some truth here, history has shown us that the information worker will adapt if the solution delivers a true value. What is more interesting is to observe the behaviors of the next generation of workers. Recently we had the opportunity to perform some community service in the New Orleans area after Hurricane Katrina. Some of our team helped restore houses, others helped local services restore computer networks, and we got to visit area schools to do something similar to a career day for the students. Before we walked into classrooms, the principals told us that most of the families who had the means to do so didn't come back to the area and that we should not have high expectations about how much technology the students had been exposed to. Of course, we didn't listen and asked the same questions we always ask students we interact with:

1. How many of you have cell phones?

Almost every student had one. And most were comfortable discussing the phone as a multipurpose device. I quickly learned to turn off my Internet-browser access when the students were playing with my Treo. Most of the students had sent text messages and used their phones for music.

2. How many of you can type faster than you can write?

Again, almost every hand in the room was raised.

3. How many of you use a computer daily?

The hands remained raised. During our time there, many students asked us questions about their home networks. Even with middle-school kids, they were the ones setting up the networks for their families.

Our school experience is evidence that not only will the average corporate business user pick up a new technology if we can provide a solution that delivers value, but our workforce continues to get injected with users who have more exposure to technology outside of the office. However, we do have to be aware of the crowd that doesn't want to learn something completely new. Because they exist, this book will focus on solutions that customize tools that have been a staple on the corporate desktop. The solutions in this book will extend the most familiar Microsoft Office tools: Word, Excel, Outlook, and PowerPoint.

Why focus on Microsoft Office? For the information worker, Microsoft Office has a proven value that few other technologies can compare with. Word, Excel, Outlook, and PowerPoint have been on the corporate desktop for more than a decade and users are comfortable working with them. Microsoft itself is using this strength to address the needs of the enterprise around how they create, organize, and search information. At the heart of these enterprise application servers are Microsoft SharePoint Products and Technologies.

Specifically, the name SharePoint is attached to two applications: Windows SharePoint Services (WSS) is a Windows Server 2003 component (it is included in the R2 release of Windows Server 2003 and available as an add-on to the original release) that provides collaboration features focused on delivering Web-based sites for teams and groups of users. These sites provide a focal point for activities such as planning a meeting, reviewing a document, or completing a business process. Microsoft Office SharePoint Server (MOSS) extends WSS to provide enterprise-level features such as search, records management, content management, personalization, application integration, and so on. There is a reason that SharePoint carries the Microsoft Office designation—these products extend the Microsoft Office desktop applications to provide services that an organization needs beyond document creation. The server applications integrate seamlessly into the Microsoft Office desktop applications. As a result, organizations can connect information workers, service their collaboration needs, and help them locate information—all from within the same document, spreadsheet, presentation, or email that they are working on.

The latest versions of the Microsoft Office desktop tools and application servers provide an incredible number of features and functionality out of the box. They integrate more strongly than any previous release. However, as organizations apply these technologies to their business processes or to solve specific problems, there may still be some gaps because of the nuances inherent in the way a company wants to use them. This is where the application developer enters center stage. The mistake often made here is for the application developer to build a solution completely removed from the environment that the user is familiar with. Any solution needs to be highly integrated with the Office desktop tools as well as leverage SharePoint's services. There would be no reason to build a custom database application that stores documents and their metadata, and provides versioning capabilities. Such features are available from any SharePoint document library. In terms of interface, why ask the user to close the document they are working on to go to some other thick Windows client or custom web page?

Historically, developers have avoided customizing the Office tools for several reasons. The first is that such solutions are notoriously difficult to construct. This is in large part due to the lack of a sophisticated development environment. Developers focused on C++, Visual Basic, or C# are typically not exposed to VBA, the development model within Office, and therefore lack the comfort level needed to build effective solutions. Microsoft Visual Studio Tools for Office (VSTO) bridges this chasm. VSTO is a runtime library that provides a layer of abstraction between the .NET developer and the Microsoft Office primary interop assemblies. This layer helps hide some of the COM particulars from the .NET developer and enables them to build .NET solutions for Office using Visual Studio and focusing on their solution, not on interop plumbing. In addition to making the code simpler, VSTO provides a designer experience. For example, when building an Excel-based solution, Excel is loaded into Visual Studio and the spreadsheet becomes the design surface. This means the developer can drag and drop their well-known Windows forms controls right onto the spreadsheet, set their properties, and add a code-behind. This is the expected experience for the Windows or Web developer. Most of the examples in this book will utilize Visual Studio Tools for Office.

Another recent milestone that promotes new development opportunities is the switch from Office documents that rely on proprietary binary file formats to formats that are open and built on XML. The new Microsoft Office 2007 desktop tools rely on an Open XML format. Very often, developers find themselves in a situation where a solution requires the generation of a document, spreadsheet, or presentation based on data in a SQL database, web service, or other external application. Previously, most solutions relied on automating Office, which required the application to be installed on the server. Not only were these solutions extremely difficult to scale, but in most circumstances they were not supported by Microsoft. With the Open XML file format, developers will be able to build server-side document-generation solutions without having the Office application even present on the server.

So this book is about building solutions on top of the Microsoft Office platform. This means that the solutions will incorporate SharePoint, Office, and VSTO. This is a book for the developer community. We assume an average level of experience building .NET applications and some familiarity with Office and SharePoint.

The three chapters following this one (Chapters 2, 3, and 4) provide an overview of SharePoint, Office, and VSTO, as well as new features and enhancements present in the latest versions of these technologies. Almost all of our chapters have a “Further Reading” section at the end in case you want more information on the topics covered there. This book is not meant to be a reference manual that teaches you every feature of these technologies; instead it shows you common solution patterns through scenarios that could apply to any organization. If you are an expert in these technologies, feel free to skim the overview chapters or even skip them and jump straight to the scenario/solution ones (Chapter 5 and onward). We set out to make each of our solution chapters capable of standing on their own so you can read them in any order, focusing on the scenarios that most interest you.

You might think with all of this technology that we will be building solutions never dreamed of before. However, that really isn’t the case. The solutions we will construct are ones that developers have been struggling to deliver for some time. The biggest differences are the ease with which we will construct them and the reduction in the amount of code required.

The solutions we will construct have their humble beginnings in custom VBA applications. Many VBA solutions are brought into businesses by a technology-savvy business user who loves to record macros and see the code they emit. Such users are able to put together a document or spreadsheet that turns into a mission-critical application. Unfortunately for the organization, this application becomes difficult to manage and deploy. Performance and security are hardly ever presented as a benefit. For the developer, debugging has always been largely a process of trial and error. Even though the VBA applications are rather primitive, these applications tend to be a huge success. The reason is that they are targeted to make the information worker's job easier and their time to deployment is rather quick. They are integrated into the Office applications and were built by someone intimately involved with the challenges faced in the workplace.

A layer above VBA applications are solutions developed using the COM interfaces exposed by Office. These tend to be application add-ins that run within an Office tool or applications that automate Office. An add-in is essentially a custom library that Office will load as the application launches. This add-in can then extend the Office tool to provide additional functionality. An example could be an Excel add-in that adds a button onto the toolbar that, when pressed, loads data from an external source onto the spreadsheet. The add-in model is a powerful one that continues into the managed code solutions created by VSTO. The biggest differences between COM and VSTO add-ins are the improvements in the development experience and the benefits of .NET code bases over COM.

Developers have also built solutions that rely on automating an Office tool. An example of this would be custom code that loads the application of Word (Word.Application) and automatically begins to construct the document based on a data source. These applications tend to be deployed on servers where this work can be done on behalf of the user. The main problem with these solutions is that they will not scale since automating Word in this manner is equivalent to each user logging on to the server and opening the application directly. Not to mention, the server-based automation of Office is something Microsoft has never been willing to support.

Smart documents were an application type that appeared with Microsoft Office 2003. The idea of a smart document was to solve the information-integration/duplication challenge. Smart documents took advantage of the task pane and Office 2003's support for XML. An example of such an application would be a Word document that, through a custom task pane, allows a user to select data from another application. That data is then placed appropriately into the document based on the document's XML schema. Smart documents were first presented as COM applications in which the task pane was HTML with an XML file detailing the controls that were to be loaded. Later the first version of VSTO for Visual Studio 2003 provided a .NET programming option, but only for Word, Excel, Outlook, and InfoPath. These types of applications got "smart" in their name because of their similarity to smart clients. In *smart client applications*, a thick client relies on external resources such as web services. This model has many appealing advantages, including deployment scenarios and support for disconnected clients. Visual Studio 2005 Tools for the 2007 Microsoft Office System continues to evolve solutions of this type and exposes this model to many more Office applications.

Solutions developed on the Microsoft Office platform are termed *Office Business Applications*, or *OBA*. The OBA team at Microsoft has documented common patterns that solutions incorporate to deliver integration, a rich experience, and data reduction for the information worker. These patterns can be grouped into the categories displayed in Table 1-1. As we introduce you to the scenarios in the book, we will describe which of these patterns the solutions implement. (Often there is not a single pattern used, but rather a combination.) For more information on these patterns and OBA, visit the OBA developer portal at `http://msdn2.microsoft.com/en-us/office/aa905528.aspx`.

Table 1-1. *Categories of Office Business Application Patterns*

Pattern Category	Description
Office Applications as a Reach Channel	Using Office to present data from other systems in an effort to simplify access or reduce duplication of effort
Document Integration	Automating the generation of documents with data from another system or processing the documents to extract data
Composite User Interface	Bringing together data from disparate resources into a single tool for the end user
Complementary Document Workflow	Providing the ability to incorporate ad-hoc workflow into other line-of-business processes
Discovery Navigation	Providing the ability to search and navigate through data of other systems
Collaborative Site	Using a SharePoint site to represent an instance of a structured process
Application Generated Tasks and Notifications	Consolidating task requests from systems into Microsoft Outlook

You've just read about the need for these solutions, the technologies we have access to, and how we may have built them in the past. Now let's focus on the information-worker problems we will solve in this book. We have arranged the solutions in this book by the Office application that we will leverage, and we've designed each solution not to rely on code presented in another chapter. Therefore, you will be able to read these chapters in any order depending on how appealing you find the problems.

Part 2: Microsoft Excel Solutions

Chapter 5: Maintaining Offline List Content from Multiple Sites—Windows SharePoint Services provides team sites to support collaboration for a particular team of users or in support of a business process. Often, users are members of several of the same type of site. As their membership to these sites increase, so does the burden of maintaining the site content. The user must visit each of these sites to make their content changes. In this chapter we will consolidate the maintenance of this content into a single Microsoft Excel workbook. A spreadsheet-based application will pull the data from the user's sites, present a single interface to make edits, and post changes back to the server. In addition, this application will support offline storage for working on the content when the user does not have access to the SharePoint sites.

Related OBA patterns: Composite User Interface, Collaborative Site

Chapter 6: Integrating Spreadsheets into the Enterprise—Information workers have become accustomed to modeling business calculations with spreadsheets in Microsoft Excel. Often developers take these spreadsheets and recode their logic into custom applications or line-of-business (LOB) systems. This usually removes information workers completely from the picture, leaving them unable to have any influence on the calculation without making changes to the application. In this chapter we focus on techniques to extend Excel so that it can participate with other enterprise systems; we do this by adding .NET code exposed as new Excel functions that can call web services, query databases, or perform calculations that are normally not available. In addition, we show you how the Excel Services feature in Microsoft Office SharePoint Server allows you to reuse the spreadsheet's calculation logic in your own applications while still allowing the information worker the ability to alter it.

Related OBA pattern: Office Applications as a Reach Channel

Part 3: Microsoft Word Solutions

Chapter 7: Merging SharePoint List Data into Word Documents—Organizations often have sets of document templates that are used throughout their enterprise. It is not unusual for several of these templates to share common data elements. At the same time, the list structures in SharePoint sites are increasingly being used to store records of data that might have previously been in local spreadsheets or Access databases. In this chapter we show you how to leverage the Open XML file format to insert a list item into a Word document so that its fields are displayed in designated locations.

Related OBA pattern: Document Integration

Chapter 8: Working Collaboratively with Document Fragments—By storing a document in a SharePoint library, users are able to collaborate and benefit from features such as versioning, metadata, and security. However, the document can be checked out by only one user at a time. What if an organization had a document that contained standard sections that needed to be completed by different users concurrently? In this chapter we will build a SharePoint library that is capable of breaking apart a document into separate files for each of its sections. Once these fragments have been generated, they can each be modified independently. Each fragment can also leverage the metadata and versioning features of the library. Once complete, our library will reassemble the sections into a single document.

Related OBA pattern: Document Integration

Part 4: Microsoft PowerPoint Solutions

Chapter 9: Extending PowerPoint to Build a Presentation Based on Site Content—Windows SharePoint Services provides team sites to support collaboration for a particular team of users or in support of a business process. Often the users involved with this process have to present their progress to the organization or to upper management. In this chapter we will extend Microsoft PowerPoint with a wizard that allows the user to have presentation slides containing site content inserted into their presentation automatically.

Related OBA patterns: Document Integration, Composite User Interface

Chapter 10: Building a Presentation Server-Side within a Web Part—When an organization uses a team site to represent an instance of a business process, it often has to report the status of that process in the form of a presentation. That presentation usually includes content from the lists within the site. For types of presentations that happen frequently, the organization has likely adopted a presentation template that is always used and ordered the slides and information in a specific manner. In this chapter we will build a web part that takes a PowerPoint template and populates the slides with content from a SharePoint site. This all happens server-side with a simple click of a button.

Related OBA pattern: Document Integration

Part 5: Microsoft Outlook Solutions

Chapter 11: Working with Email Messages and SharePoint—Microsoft Outlook already includes functionality to dissuade users from sending copies of a document as attachments in an effort to collaborate and gather feedback. When the authoring user sends the email with the attachment, Outlook asks them if they would rather create a collaboration site and send a link to their recipients. However, there is no such integration for an email a user *receives*. What if a user receives an email with attachments from an outside organization that warrants an internal site for processing? In this chapter we will extend Microsoft Outlook to enable these users to easily persist these messages to SharePoint repositories.

Related OBA pattern: Collaborative Site

Chapter 12: Surfacing Data from Line-of-Business Applications—Information workers are often put into an environment where they need several tools that contain silos of duplicate data. They often have to jump in and out of these tools, copying and pasting data from one screen to the next. In this chapter we will show you how to leverage the Business Data Catalog (BDC) functionality of Microsoft Office SharePoint Server to seamlessly integrate a line-of-business system with SharePoint. This will allow the data to be used in web parts and columns of lists, and will support search indexing. We will also show you how to extend the BDC's integration with a web service so that another application (Microsoft Outlook) can consume it. This limits the integration to a single authoritative point for the data.

Related OBA patterns: Office Applications as a Reach Channel, Discovery Navigation, Composite User Interface

Part 6: Microsoft InfoPath Solutions

Chapter 13: Taking InfoPath Forms to the Web—InfoPath was introduced with Office 2003 and information workers in the enterprise quickly realized its value in easily replacing the collection of data traditionally done with paper forms. However, the use of InfoPath was usually relegated to simple, departmental forms that never left the boundary of the company. This limitation was largely due to the fact that user who would need to fill out the form also needed to have InfoPath. In this chapter we will show you how the Form Services functionality of Microsoft Office SharePoint Server enables you to take InfoPath-designed forms and present them to a user in a browser-only interface. We will also tackle strategies for incorporating these forms into the enterprise, such as how to connect to data sources, security implications, and how to host the forms in your own application.

Related OBA pattern: Office Applications as a Reach Channel

Chapter 14: Incorporating Workflow into Forms Processing—Traditionally forms are used to collect data for a line-of-business application. The user fills out a form and the data is submitted into another system. However, many scenarios require a complementary workflow process to intervene before the data is stored in the LOB system. These workflow processes could be ad-hoc, requiring different routes of approval. In this chapter we will show you how to leverage SharePoint's workflow capabilities to complement the processing of forms to enterprise systems.

Related OBA Pattern: Complementary Document Workflow

Part 7: Conclusion

Chapter 15: Realizing the Vision—Finally, Chapter 15 will sum things up, revisiting the importance of the solutions and taking a look into the future of developing on the Office platform.

Development-Environment Requirements

There are many options for a development environment, but we are assuming that you are using either Microsoft Virtual PC or Microsoft Virtual Server to provide you with a sandbox to implement the solutions presented in the book. Our core development environment for the book consisted of two Windows Server 2003 R2 virtual machines running under Virtual Server. This does create quite the hardware requirement. If you are going to match our configuration, you should have a recent machine with at least 2GB of RAM, and an external drive for one of the virtual machines is recommended. One virtual machine was our domain controller (sample.com) and supported Microsoft Exchange 2003 and Microsoft SQL Server 2005. The second virtual machine was the Microsoft Office SharePoint Server and our development environment. The following products were installed there: Microsoft Office Enterprise 2007, Microsoft Office SharePoint Server 2007 (enterprise features enabled), Microsoft Office SharePoint Designer 2007, and Microsoft Visual Studio 2005 Team Edition. In MOSS we used separate web applications for the shared service provider, the My Sites (my.sample.com), and the portal (portal.sample.com). In addition to these core products there were many starter kits, SDKs, toolkits, and things we installed. Here is a summary:

- Visual Studio Tools for Office 2005 (included in VS.NET Team Edition)
- Visual Studio Tools for Office 2005 SE
- The .NET Framework (versions 2.0 and 3.0)
- 2007 Office System XML code snippets
- MSXML 6.0 parser
- Office SharePoint Server 2007 SDK (including ECM Starter Kit)
- Visual Studio 2005 extensions for .NET Framework (WCF & WPF) November 2006 CTP
- Visual Studio 2005 extensions for Windows SharePoint Services November 2006 CTP
- Visual Studio 2005 extensions for Windows Workflow Foundation
- Windows SharePoint Services 3.0 SDK

It is possible to do everything on one virtual machine, but in our experience it really does not lessen the hardware requirements of the machine you run on.

■ **Note** There is one solution (Chapter 5) that required us to have an Office 2003 Professional–based environment. For that solution, we had a Windows XP virtual machine with Microsoft Office 2003 Professional, Visual Studio 2005 Team Edition, and both Visual Studio Tools for Office editions. Since Chapter 5 is different from the others, we will repeat this information in a compatibility section in the chapter.