1

Anatomy of an Office Business Application

Over the past twenty-odd years, companies have spent considerable amounts of money (in the billions of dollars) installing and maintaining line-of-business (LOB) systems to manage all types of data, including customer data, sales and finance data, and human resource information. In many cases, these business data represent the backbone of the organization. One of the issues with these LOB systems, though, has been accessibility; the systems have been accessible to only a subset of people within the organizations. Though in some cases this accessibility might be by design, for the most part organizations can benefit by exposing much of the business data within the LOB system to their employees. For example, if salespeople have direct access to updating their sales data within the LOB system, they rely less on people who have direct access to the system and subsequently have to copy and paste data out of the system. For those that do have access to the system, there are a number of costs to the organization that are quite significant. For example, in a study entitled "The Financial Impact of Packaged Applications" (Forrester, 07/11/2006), a number of key costs were outlined such as training, temporary business backfill, or change management — all direct costs to the company resulting from an enterprise-wide LOB system implementation (see Figure 1-1). When you consider many LOB system implementations run in the twenty-five million plus region, these cost hits are no small matter.

With the growth of LOB system installations, there have also been some other changes taking place in the workplace. To name a few, many of us engage in an abundance of email communication, sending attached documents like fiscal plans created in Excel spreadsheets "across the wire" instead of posting them on a central document share; many people keep information local to their PCs (think of the problem with intellectual property leakage when these people leave the organization and IT staff wipe the drives of their machines — this information is now lost); the organization has evolved into a more team-based environment, so sharing information and knowledge has become a critical part of our daily collaborative processes; and many people are generally unhappy with the current LOB systems that are costing companies so much to implement today (some studies have found these system have user adoption issues in the neighborhood of 45% of the total user base). With these changes have arrived many new productivity tools

to help manage our processes and behaviors within the workplace; in fact, it's hard to think what our lives would be like without these tools in place. Though there are many different productivity technologies on the market today, this book deals primarily with those tools, servers, and applications found within the Office platform and how these tools can integrate with LOB systems to not only mitigate the cost burdens just described (and shown in Figure 1-1), but also to begin to take better and more discrete use of the business data that lives in these LOB systems. That said, OBAs help mitigate the issues within these scenarios by leveraging the Office platform to:

- Provide a central interface into the business data within a LOB system
- Use existing technology (assuming an organization has deployed Office) to build OBAs
- Improve user adoption issues with users interacting with the business data within a comfortable environment
- Increase cost savings in, for example, training and productivity

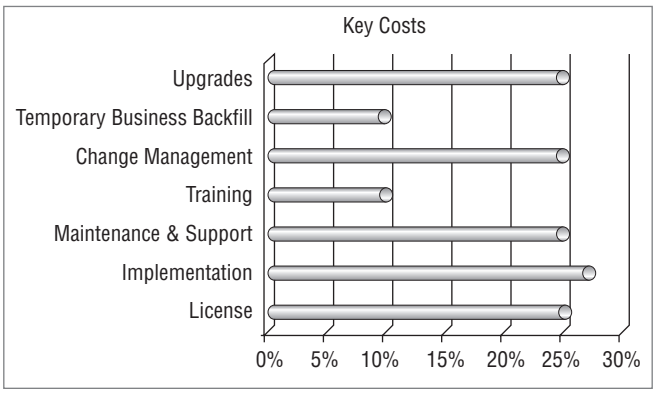

Figure 1-1

Now that you understand the high-level problem, you're probably asking yourself…

What Is an Office Business Application?

OBAs are in essence composite applications that leverage the Microsoft Office platform (or OBA Framework) to bridge the "results gap," the gap that exists when information workers don't have access to critical business data that reside in these LOB systems. OBAs are, for the most part, service-oriented, so they will require a service-oriented architecture (SOA) — or what some in the IT industry are also referring to as Software plus Services (S+S). (S+S is a broad term for integrating software that exists on the desktop with a variety of services that exist in the 'cloud.') OBAs provide access to the LOB system via a service proxy so information workers can use up-to-date business data in their everyday business lives within an environment that is both comfortable and familiar to them: the Office environment. Examples of service proxies might be Web services, Windows Communication Foundation (WCF) services, or even hosted services.

Examples of OBAs range from solutions that are specific to sales, finance, and human resources, to customer relationship management. Typically, these types of solutions are tied to some process or

workflow (for example, approval of budget or interview evaluation forms) and are larger in scale — in other words, enterprise-level solutions. You can also build smaller, more discretely targeted OBAs that, for example, process annual performance review forms or generate expense forms. OBA solution design, as you'll see throughout this chapter, can also include smart-client applications (for example, documents with custom task panes or custom ribbons) to SharePoint customizations (for example, custom Web parts). It is no coincidence that these OBAs are tied to key business data that resides in the LOB system; it is the purpose of OBAs to expose this data to the organization to improve process, productivity, and business.

At a high level, we can roughly define OBAs as being composed of four main pieces:

1. The LOB system
2. Client-side customizations
3. SharePoint (or other server-side) customizations
4. The services (for example, Web services, WCF, or BizTalk adapters) that bind the LOB system to the Office client and SharePoint

Figure 1-2 illustrates these four parts of the OBA. One thing to note is that although these are the main components of many OBAs that are being built today, it is possible to build an OBA that has just a client-to-LOB system integration or SharePoint-to-LOB system integration.

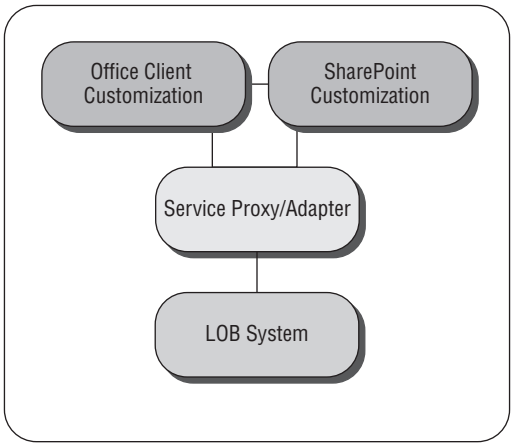

Figure 1-2

Note that in the diagram, we've made the line between the Office client and SharePoint a dotted line. This is to indicate that you could integrate these two parts of the OBA, however, it's not necessary for the solution to be considered an OBA. An example of this type of integration would be mapping a Visual Studio Tools for Office (VSTO) document-level customization to a SharePoint content type (an example that is explored in Chapter 10, "Deploying Your Client Components").

OBAs provide deep value in a number of ways. First, OBAs unlock key business data for team use through technologies such as the Business Data Catalog, Excel Services, Visual Studio Tools for Office (VSTO) customizations, and so on. Second, OBAs leverage existing functionality that is native to

Office, thus alleviating the need to reproduce this functionality in your solution design. An example of this would be building a client-side customization for Excel 2007, which would enable your end-users to take advantage of the functionality native to Excel, such as charting or functions. Another example might be building an OBA within Outlook and then taking advantage of the underlying messaging and calendaring functionality within Outlook. Third, OBAs are about enabling process, so you can customize your OBA solutions centric to a particular human workflow within the organization. The example we use throughout this book is a sales forecast approval solution (we talk more about this in Chapter 2, "Architecture Guidance and Design Patterns for Office Business Applications"), where a manager can *approve*, *reject*, or *amend* a salesperson's forecast through the custom workflow built into SharePoint and associated with the sales forecast document. Fourth, OBAs are about sharing information and collaboration — whether it be through document libraries, exchanging or sharing information via blogs, wikis, or MySites (where individuals can provide personalized SharePoint sites with searchable information), OBAs are all about enabling the collective — thus enabling community-level and Web 2.0 technology around Office integrations with the LOB system. This is where you'll truly begin to see the strength of the OBA technologies (which we discuss in the next section of this chapter). Finally, OBAs are about keeping information workers inside the context of their everyday environments while providing them access to the information that they need. It is here that we can distill much of the benefit of building OBAs and meet real cost and productivity savings. This cost-savings is not only realized in the context of keeping information workers within a familiar environment (thus mitigating some of the costs listed in Figure 1-1), but also alleviates engineering costs by eradicating the need for additional tools generation, support, and ongoing maintenance costs.

Types of Office Business Applications

Though the scenarios around designing and deploying OBAs can vary, there are essentially three different "types" of OBAs:

1. Type 1: Client to LOB system architecture. This represents a smart client that integrates via a service (for example, Web service, WCF service, and so on) such as a custom task pane or custom ribbon in Excel, Word, or Outlook.
2. Type 2: Client or Server to LOB system architecture. This represents the combination of a smart client and SharePoint server customization integrated, for example, via a service to the LOB system. We've discussed an example of a smart client, but the SharePoint customization might be a Business Data Catalog (BDC) Web part that consumes a service to render data.
3. Type 3: Client and Server to multiple LOB systems architecture. This represents a smart client and SharePoint server customization integrated with multiple LOB systems.

Although SharePoint is the primary server-side platform on which OBAs are built, you can also use other servers in the Office platform — for example, Exchange Server or Communications Server.

Though not mainstream, there is also the architecture where you could use hosted services that integrate with your LOB system. This architecture arguably could represent a fourth type of OBA.

Across these types, you can also have permutations of OBAs, depending on what designs you choose and what technologies you use to design your OBAs. To make this a little more real, take a look at a problem scenario and apply an OBA design to it.

Suppose you're a product manager, and you're trying to identify business data that helps justify the creation of a set of features for a new software application. You do not have access to the LOB system where there is key data to justify the features, so you request from your co-worker (call her Sandy), who does, to send you some data that will help in your quest. Sandy searches for the appropriate data for you and copies and pastes the data in an Excel spreadsheet. She then saves it locally in a *Temp* folder and sends the spreadsheet to you via Outlook. You are now happy that you have some data to justify the features you're going to spec, and you build your case for the new product. This is great, right? Well, not really.

First, in organizations today there are many people looking for business data that helps them create competitive products or features and our LOB systems contain much of that needed data. Second, not having direct access to the business data (and sending it around via email) is problematic. That is, if the day after the data is sent to you the data does not justify the features, all future work on that product becomes a cost, both real and competitive, to the organization. Because of the disconnect, however, you, the product manager, may not realize this until too late, or worse, not at all. Third, the process of product development is typically not done in isolation; it's a team process where virtual teams convene and agree upon the next foot forward. The costs of creating this feature set around stale data are not isolated to one person; they extend out to virtual teams and to the wider organization. Figure 1-3 illustrates this process. Note that the diagram also includes this idea of moving from the *structured* realm (where an OBA manages direct integration with the business data the processes around that data) to the *unstructured* realm (where there exists no OBA and where the business data moves further away from the source affecting more people and business decisions along the way — the results gap).

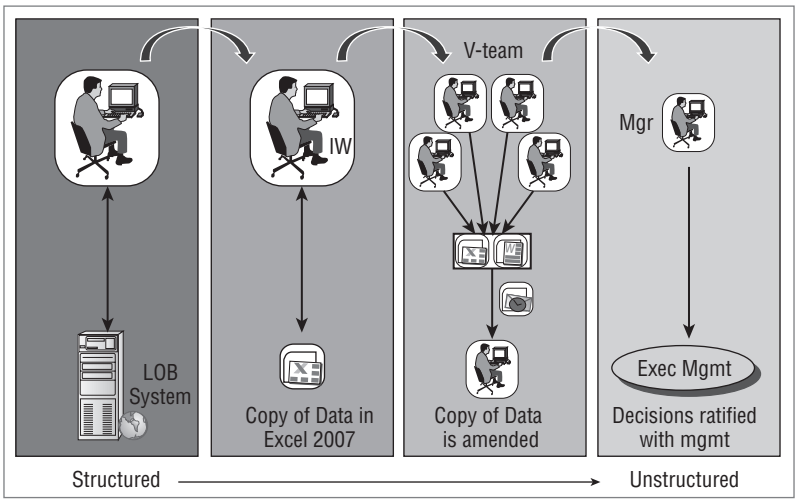

Figure 1-3

If you were to build an OBA to solve this problem, what would that OBA look like?

For this type of scenario, you might want to build a fairly simple OBA (that is, a Type 1): a smart-client Excel spreadsheet that is connected to the appropriate data source via a Web service to the LOB system. So, when you open Excel, the managed code assembly would load a custom task pane that further loads the appropriate data using the Web service connection to your LOB system. You could then select the data that you needed and populate the custom Excel template (or workbook) with the data. You might

also use a custom task pane in a Word document; especially if you don't require any calculations but just require the creation of a report or inclusion of data to justify your business case within a specification template. This design, as opposed to, say, binding the LOB system data to a ListObject, might require instead using a binding method to Word content controls or OpenXML. Don't worry if any of this doesn't spring forth elaborate visions; we explore a lot of this throughout this book, so if it's not familiar now, it will be soon enough.

The OBA we explore in this book is one that is based around a sales forecast scenario; that is, a sales team that needs to submit their forecasts, have their managers review them, and then have them subsequently approved. During this process, we'll explore many different technologies that you can incorporate in the design of your own Office solution. Unlike other books that discuss related technologies using many different scenarios, we wanted to build the solution from the ground up. The reason we chose the path of building an end-to-end solution and then discussing all of the facets of releasing this solution "to market" (or to your enterprise) is that you can get a sense for all of the things that are possible when releasing an OBA to the enterprise. Thus, this book provides you with some information on the How, in both the contexts of the technology and deployment and security — from both a smart-client perspective (which you learn about in Chapter 4, "Customizing the Office Fluent Ribbon and Task Pane") and a SharePoint perspective (which you learn about in Chapters 5 through 10).

Before we dive into the solution and the rest of the book, though, let's first discuss the technologies that are available for you today to build your OBA. We say today because, as we write, Visual Studio 2008 is the tool released to market, and when this book is published you'll likely hear whispers of Visual Studio "10" and SharePoint/Office "14" futures, as well.

OBA Technologies

OBAs can be relatively straightforward in their design; conversely, they can also involve many moving parts. As I'm sure you can extrapolate from the different types of OBAs mentioned earlier in this chapter (Types 1, 2, and 3), they can range from a simple smart-client (for example, a client customization that includes a custom task pane) that uses a Web service to load data into a Word document to more complex solutions that involve custom workflow, multiple smart-client elements, consumption of SharePoint services, and so on. As an interesting example, I see the Word-based OBAs quite a lot when working with the public sector. IT departments will make custom task panes with tabs that manage data into a document that creates boilerplate forms that information workers can then fill out — thus increasing productivity and centrally storing boilerplate form data. This is, obviously, only one design for OBAs. You could, for example, use other, "thinner" presentation layers for a service, such as an InfoPath form within a SharePoint environment. InfoPath forms are great for integrating with data or Web services. Though your presentation layer can vary depending on your requirements and design, remember that OBAs are not about Office technology alone; they're also about the LOB system with which you're integrating. In large part, you'll be designing and developing the services part of the solution first, which will likely mean using the native tools within your LOB system.

This book does not cover individual LOB systems; rather, it abstracts the services away from the other layers of the OBA, which include a presentation layer, productivity layer, and services layer. Thus, the assumption is that if you're going to build your own OBA, you'll design and build your LOB system services first and then use the OBA technologies to integrate with the Office system.

That said, Figure 1-4 provides an overview of the different layers of an OBA. In effect, it shows the different technologies that exist within each and more broadly within the Office platform. Now, building an OBA does not necessarily mean using all of these technologies in your design. Use this diagram to help you understand what options are available to you when designing your OBA solution. For example, you may design your OBA using only a smart client or you may also have another Web-based Intranet presentation layer that uses SharePoint Web parts. Each presentation component would serve a different purpose, but keep in mind that the OBA would grow in complexity and support with each new moving part you add to it. The following sections take a deeper look at each of the layers in Figure 1-4.

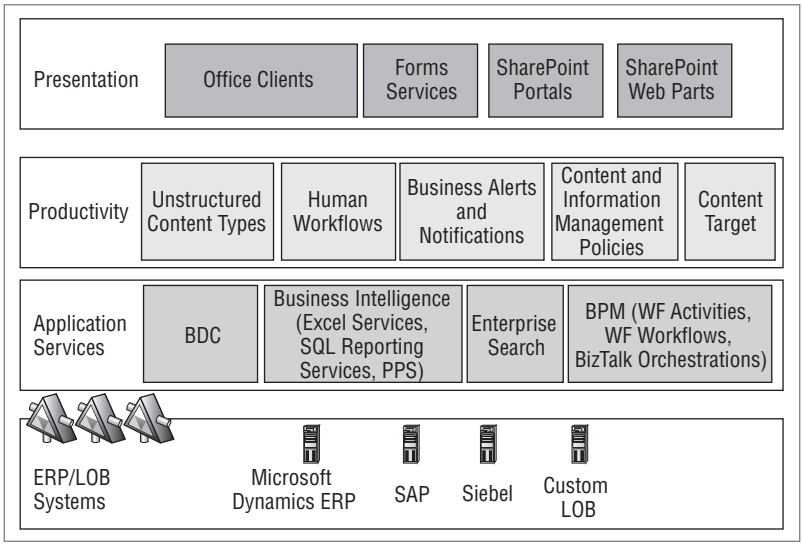

Figure 1-4

Presentation Layer

When building the presentation layer for your OBA, you have a number of options depending on your design. Your design might include an Office smart client, InfoPath forms, SharePoint-centric design, or all of the above.

Visual Studio Tools for Office (Smart Client)

An Office "smart" client (at least as defined in this book) uses the Visual Studio Tools for Office (VSTO) technology (now included in Visual Studio 2008 Professional and above) to build managed code (that is, Visual Basic and C#) assemblies (or add-ins) that you deploy to the client. These add-ins then show up when you open up the application with which the assembly is associated. These add-ins typically manifest through an extension in the custom toolbar menu (in Office 2003 client applications), custom Fluent (the branding for the Office 2007 user interface) ribbon extensions, custom task/actions panes, or other types of customization embedded within the document or triggered by an event. Figure 1-5 provides an example of an Excel 2007–based document-level VSTO solution. The solution includes a custom ribbon and custom task pane to manage data input to the Excel spreadsheet.

The add-ins that you create with VSTO come in two flavors: document-level solutions and application-level add-ins. The document-level solution is tied to a specific document (so is only loaded when you open that particular document), and the application-level add-in is tied to a specific Office application (so loads every time that host application loads). The example in Figure 1-5 is a document-level solution, so would load only when you load the particular document with which the solution is associated.

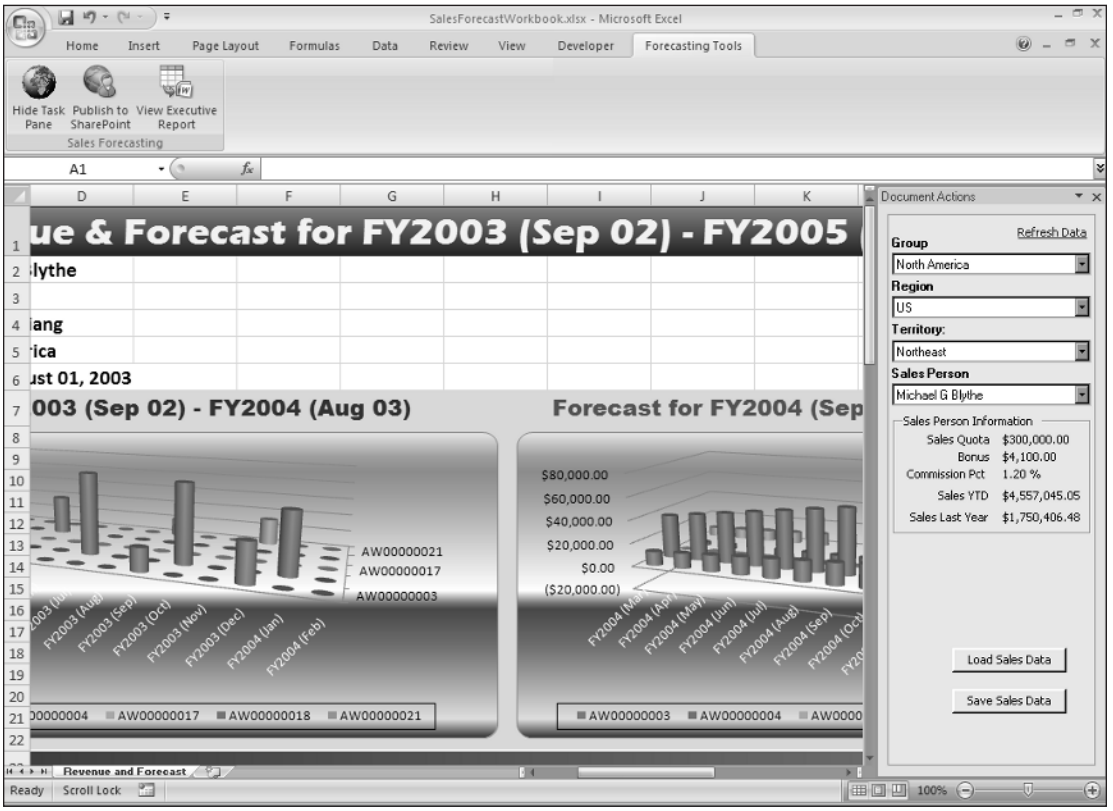

Figure 1-5

Application-level add-ins, though, are also very powerful. For example, mentioned earlier in the chapter was the fact that you could build OBAs by extending Outlook. Many companies are taking advantage of this technology to extend Outlook in many exciting ways. Creating an Outlook add-in using VSTO, you can customize the menu toolbar, customize the Outlook ribbon, create custom task panes, create custom Outlook form regions, and so on. All the while, you can integrate these objects with data sources or Web services. Figure 1-6 illustrates a custom Outlook form region (a custom contact form) that has altered the contact form in two ways. First, a data grid is linked to an external data source; second, custom logic that sits behind the form calls a service that maps the selected address of the customers in a web browser pane.

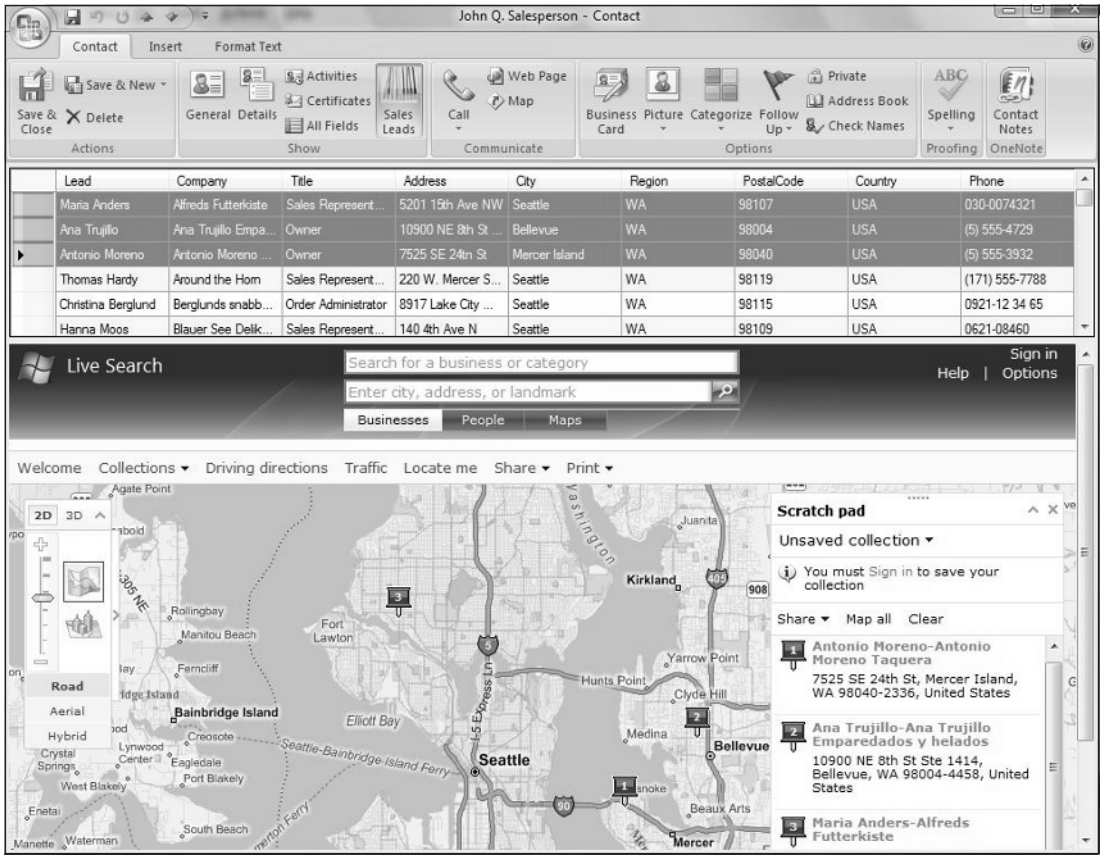

Figure 1-6

In short order, creating OBA solutions using VSTO is a great way to take advantage of many of the new features of the .NET Framework 3.5 and Visual Studio 2008 (for example, Windows Presentation Foundation [WPF], Windows Communication Foundation [WCF], Windows Workflow Foundation [WF], Language Integrated Query [LINQ], and so on) and build these into your smart-client solutions. You'll see more of VSTO in Chapter 4 and more about the other technologies throughout the book.

OpenXML

VSTO is not the only client-side technology that is conversant with Office 2007. OpenXML, a culmination of XML efforts that began in earlier versions of Office, enables server-side processing of the information that exists within a document by storing the document in XML format. This not only allows for dynamic document creation and manipulation, but it also exposes the underlying content of a document to other Office (and custom) services and processes; thus, you can program against a document without having to open it — that is, you open it programmatically without needing the host application to manage the opening of the document for you. For programmatic document manipulation, this is a huge step forward with support for the OpenXML cutting across Word, Excel, and PowerPoint. We discuss OpenXML in greater detail in Chapter 7, "Using OpenXML and Business Data."

InfoPath Forms

InfoPath 2007 is the way in which many developers are creating forms for Microsoft Office SharePoint Server 2007. InfoPath is a forms-based technology that enables you to build thin-client forms quickly, embed controls on the forms, and then tie those controls to data sources or to services, making it an easy-to-use, thin-client technology. In SharePoint 2007, there exists a forms publishing capability that enables developers to build and deploy forms direct to a MOSS 2007 site. You can further integrate workflow (as the InfoPath form exists as a document object within SharePoint), which allows you to use InfoPath forms for a wide variety of business processes. This ability to integrate with workflow makes InfoPath a technology that is doubly useful for creating OBAs. Assuming that you're building them within the InfoPath Forms Services framework for SharePoint, InfoPath forms add quite a lot of power when thinking about process-based and collaborative designs for your solution. Figure 1-7 illustrates an InfoPath form that is tied to a loan application processing workflow. In this example, the form retrieves data using the MLS (Multiple Listing Service) Web service (via the Go button) that passes the MLS number entered as a parameter and then loads property information into the form. For more information on this specific InfoPath example, you can download the OBA Loan Origination Reference Application Pack at `http://msdn2.microsoft.com/en-us/architecture/bb265266.aspx`.

Though we won't cover InfoPath in great depth in this book, you can look at a number of resources to learn more about this technology. See the "Further Reading" section of this chapter for more information. The reason we didn't use InfoPath for our OBA solution was because we wanted to build an OBA that used the Office client — and functionality of the Office client — as opposed to a thin-client solution.

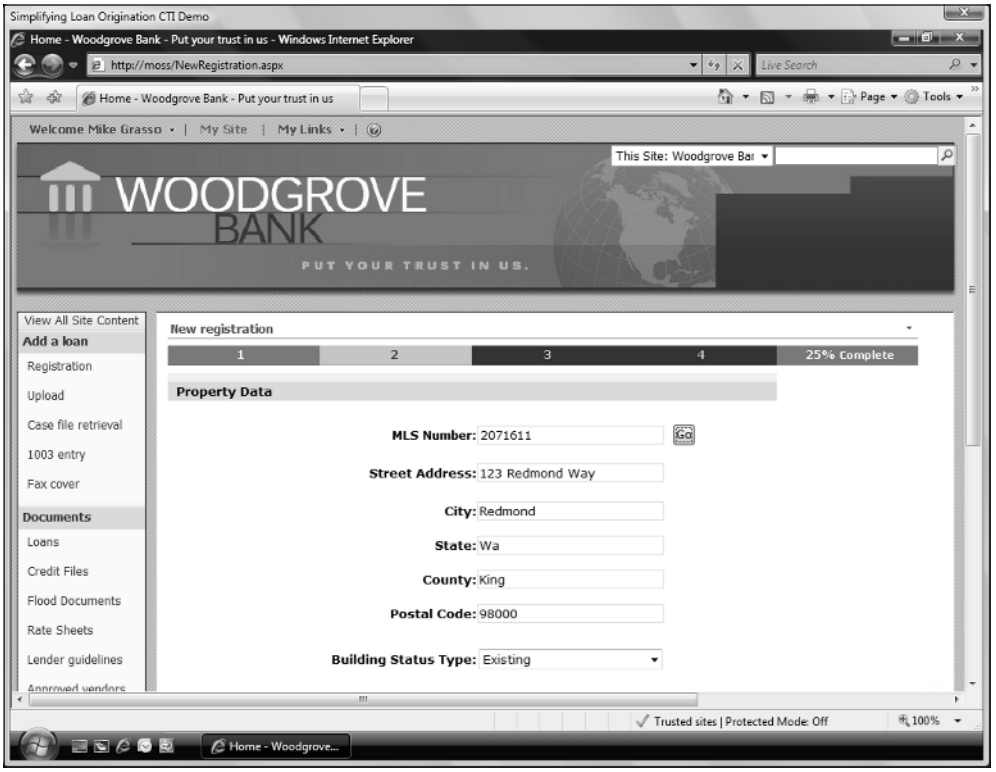

Figure 1-7

SharePoint

SharePoint is a technology that has grown incredibly over the past five years into a mature platform that enables developers to build and deploy applications not only for the enterprise but also for publicly facing sites as well. The SharePoint technology (which is referred to as SharePoint in this book) comprises two major parts: Windows SharePoint Services (WSS) and Microsoft Office SharePoint Server (MOSS). WSS is essentially the infrastructure that you can use to build sites, lists, and Web parts, for example. There is a powerful object model that lies behind WSS, and the software to build solutions based on WSS is free. MOSS is a server-based program that provides a number of out-of-the-box enterprise features that support collaboration, manage content and processes, and access information and people across your organization. You might think of MOSS as an out-of-the-box SharePoint site and tools that enable you to build, customize, and extend your SharePoint team sites and portals. When you further combine SharePoint with technologies such as Silverlight, AJAX, or other Web 2.0 technologies, you can build some very compelling user experiences on top of an already robust platform.

SharePoint portals represent a great way to provide team-based views, and subsequent role-based drill-down views, to manage process, documents, data, and information (to name a few). In this book, you build a sales forecasting portal to manage your document library, workflow, dashboard views of business intelligence, and search results — see Figure 1-8.

Figure 1-8

When building SharePoint portals and sites, you'll need to become familiar with what is known as a Web part. You can think of Web parts as one of the essential building blocks for the SharePoint site, and as such are a critical component to the design and deployment of an OBA. The Web part consists of a title bar, a frame, and content (which changes depending on which Web part you select from the Web Part Gallery); see Figure 1-9. You can choose from two primary types of Web parts: a standard Web part that comes out of the box with MOSS 2007, and a custom Web part.

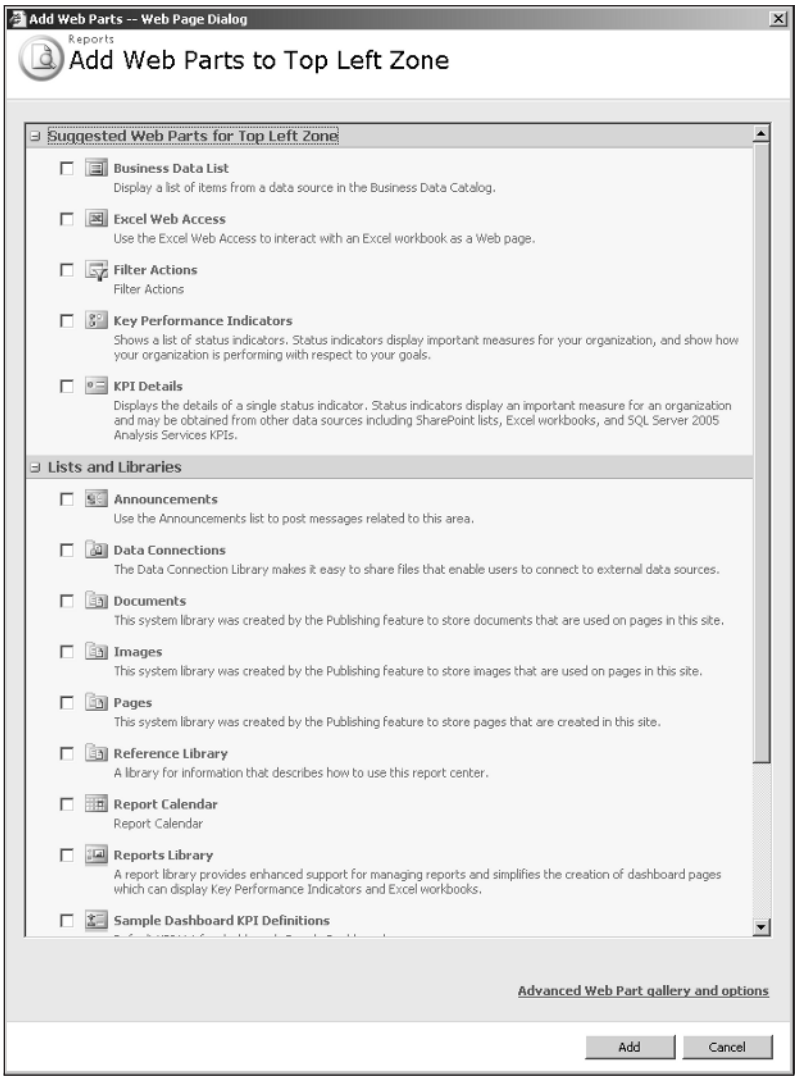

Figure 1-9

You can create a custom Web part and embed your own custom code to accomplish very specific tasks. These custom Web parts implement ASP.NET code, and developers typically use Visual Studio 2005 or 2008, SharePoint Designer, Visual Studio Extensions for Windows SharePoint Services, Silverlight, or Windows SharePoint Services to create them. Alternatively, you can use an out-of-the-box Web part (for example, Excel Services Web part) to create a business intelligence view for your OBA. Through the connection of different Web parts on one page, you can also integrate data views to make your Web parts more dynamic. For example, you can use two types of Business Data Catalog Web parts, one for a summary view and another for a line-item view, so both Web parts work together to give you high-level and detailed information.

Productivity Layer

Beneath the presentation layer lies the productivity layer, or essentially, the parts of the OBA that manage process, security, information, and so on, within your OBA solution. In Figure 1-4, we highlight a number of areas that are important when it comes to productivity. For example, when thinking about integrating smart clients into SharePoint, one of the key features in SharePoint to consider is the ability to create custom content types (an object that allows you to set properties and map the content type to a specific document or template) and then map those content types to client-side customizations. It is this type of client-server integration that begins to strike home at the power and potential of OBA solution development. That is, when you integrate the customized Excel document, for example, and a SharePoint content type, you are now integrating the Office client application features within your OBA solution — think of all the calculations and native spreadsheet functions that are now tied into your application. We've taken this approach with the sales forecast OBA in this book, so you'll learn more about this throughout the book, but the specific integration piece is covered in Chapter 10, "Deploying Your Client Components."

There's another important aspect to making the custom, smart-client document an artifact of SharePoint; you've now provided an object against which you can build custom SharePoint workflow. As our organizations become ever-more driven toward processes and accountability through those processes, creating software solutions that manage process through workflow is becoming increasingly more useful and pervasive. In the case of the integrated document, you now have the ability to tie the document into a workflow that further ties into the messaging of alerts and notifications to ensure that the appropriate people are completing the appropriate tasks. This is an important aspect of the sales forecast OBA; that is, the fact that when we take a custom document and integrate that document with SharePoint we can now manage processes around that document through custom workflow.

Application Services Layer

The LOB system business data is processed and managed through a number of key services, shown in the Application Services layer in Figure 1-4. For example, the BDC can provide a business intelligence view that can directly link to SQL Server (or other ADO.NET-based data source) or to a LOB system through a Web service. Figure 1-10 illustrates sample BDC Web parts that are interconnected and surfacing data through a Web service. Further, you use the Excel Services to create Web-based views of Excel spreadsheets — either the entire worksheet or objects within the worksheet. Key Performance Indicators (KPIs) can also be used to create score-card views to provide "quick-checks" on status for project/business health metrics. We discuss business intelligence in greater detail in Chapter 8, "Adding Business Intelligence through Excel Web Services and Key Performance Indicators."

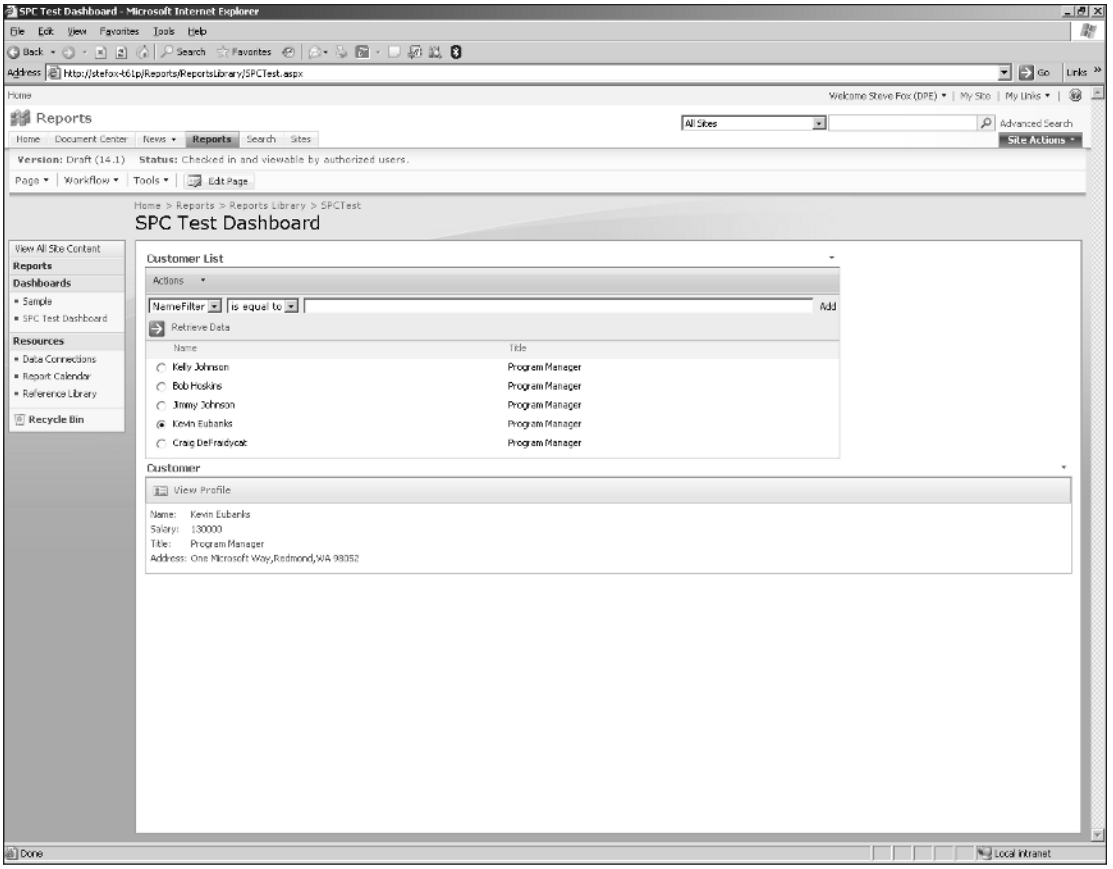

Figure 1-10

Although business intelligence is an important part of the SharePoint services, data discovery is also important, which brings us to search: a central part of any organizational portal. SharePoint's search is a shared service that provides extensible and rich information discovery through its querying and indexing engines. It supports full-text and keyword search, and it can be extended to search and index LOB system business data through the BDC. For the sales forecast solution, we felt that search was a very important component, and so we've dedicated an entire chapter (Chapter 9, "Integrating Your LOB System using the Business Data Catalog, and Extending Search into Your LOB System") to discussing how we extended search into the LOB system to query and index LOB system data.

We think you'll find this book extremely useful. There is a lot of good content that provides developers with key guidance on developing and deploying OBAs using both the client-side customization (that is, the smart client) and SharePoint customizations. We hope you enjoy reading and learning from this book as much as we enjoyed writing it. As an overview of what you'll find in the rest of the book, the next section provides a brief summary of each chapter.

What You'll Find in This Book

- Chapter 2: Architecture Guidance and Design Patterns for Office Business Applications
  - This chapter introduces the OBA solution patterns, and then applies the appropriate pattern to the sales forecast scenario we'll use throughout this book. The chapter also discusses considerations and decisions for our design and for alternative designs that you might consider when building your OBA solutions.
- Chapter 3: Installing and Configuring MOSS
  - One of the key aspects of the sales forecast solution is the team portal, through which all team members have access to the documents, information, and sites. This chapter provides a summary of how we created, designed, and deployed the site for team consumption.
- Chapter 4: Customizing the Office Fluent Ribbon and Task Pane
  - This chapter introduces you to Visual Studio 2008 and Visual Studio Tools for Office 3.0. It provides an overview of how, for example, to customize the Office Fluent ribbon and create a custom task pane and then tie these user interface elements into core parts of the sales forecast solution.
- Chapter 5: Creating and Deploying a Custom MOSS 2007 Workflow Using Visual Studio 2008
  - Visual Studio 2008 (and the Visual Studio Tools for Office component technology) is a powerful tool that provides the ability to create custom workflows for SharePoint. This chapter shows you how to create the custom workflow that was used in the sales forecast solution. It also shows you how to debug, test, and deploy the workflow in the Visual Studio IDE against and within the SharePoint environment.
- Chapter 6: Creating a Custom Outlook Form Region
  - Custom Outlook form regions are a great way to extend Outlook into an OBA solution. This chapter shows you how you can create a custom Outlook form region that is bound to data and uses WPF to enhance data visualization. It also discusses how we tied the Outlook form region to custom SharePoint workflow.
- Chapter 7: Using OpenXML and Business Data
  - Part of the document-creation process is using OpenXML to parse and generate server-side documents. This chapter focuses on using OpenXML to demonstrate how we did this in the sales forecast solution.
- Chapter 8: Adding Business Intelligence through Excel Web Services and Key Performance Indicators
  - Business intelligence is a key aspect of OBAs, and it is very important in the sales forecast solution. This chapter shows you how you can create Excel services and KPI Web parts that provide business intelligence views into the sales forecast data.
- Chapter 9: Integrating Your LOB System using the Business Data Catalog, and Extending Search into Your LOB System
  - One of the key ways of integrating your LOB systems with SharePoint is through the Business Data Catalog (BDC). This chapter walks you through how you can both integrate the BDC with databases *and* Web services using the BDC Definition Editor, a tool

that shipped with the SharePoint Server SDK. This chapter also shows you how to integrate LOB system search and indexing into your SharePoint search to enhance data and information discoverability.

- Chapter 10: Deploying Your Client Components
  - One of the critical pieces when building OBAs is the integration of the client-side customizations that you build using VSTO and SharePoint. This chapter walks you through how you integrate the sales forecast customized template with a content type within SharePoint.
- Chapter 11: Deploying and Securing Your OBA Server Components
  - Building your OBA is one thing, but deploying and securing it is yet another. This chapter describes how you deploy both the server components for your OBA and also provides guidance on how you can secure your OBA as part of your enterprise solution infrastructure.

Further Reading

At the end of each chapter, we've provided some (of what we feel are) useful links for learning more about the areas of discussion in this book. We've taken the approach of creating an end-to-end solution for this book and walking you through the entire solution from top to bottom. We cannot, unfortunately, cover every technology that we'd like to cover that is related to OBA development (because we chose a specific design).

- VSTO Developer Center: `http://msdn2.microsoft.com/en-us/office/aa905533.aspx`
- SharePoint (WSS) Developer Center: `http://msdn2.microsoft.com/en-us/sharepoint/default.aspx`
- Office Developer Center: `http://msdn2.microsoft.com/en-us/office/default.aspx`
- OBA Developer Portal: `http://msdn2.microsoft.com/en-us/office/aa905528.aspx`
- InfoPath Developer Portal: `http://msdn2.microsoft.com/en-us/office/aa905434.aspx`