Preface

It Just Makes Sense

Over the past ten years, Extensible Markup Language (XML) has become more widely used than ever before as a means of transferring data between applications and even between organizations. XML provides a standard protocol with which these applications and organizations can communicate. Using XML Schema, a company can define a standard structure for its data that can then be used across multiple departments and organizations. This structured data enables developers to easily create applications that can communicate with each other without much effort.

In addition, most organizations use forms in one way or another, whether to enter a purchase request, submit expense report information, or track weekly status. If you look at a typical form, you will notice that the form itself is structured unlike a typical freeform document created in an application such as Microsoft Office Word 2007. In these freeform documents you can type anything you like in any way that you choose. Although a form may contain sections that allow you to enter freeform text such as comments, most of your typical forms are highly structured. Fields in the form usually require you to enter specific types of data such as sales numbers or costs. Since XML defines a structured data format (which can contain some unstructured elements) and forms are highly structured with bits of freeform data, it makes sense to tie together forms and XML data. Once a user has filled out a

form that is connected to XML data, the data can easily be incorporated into back-end processes that understand the structure of the XML data for that form. So, this fits one of the main purposes of XML—tying together multiple processes using a standard protocol.

Since building forms based on XML just makes sense, many software developers want to create forms-based applications to collect data and store it as XML. However, until a few years ago, this was a tedious and time-consuming process. Developers had to use tools such as Microsoft Visual C++, C#, or Visual Basic .NET and write sometimes a tremendous amount of code to create a forms application. Often, forms applications share similar functionality, such as spell checking, calculations, and data validation. In order to share this functionality across multiple forms applications, software developers needed to create code libraries in order to reuse their code. This worked fine when sharing the code within the same department or company. However, developers across multiple companies were likely going to duplicate the same work unless, of course, companies purchased these libraries from a third-party vendor.

Developing forms applications in this way is not something that typical information workers can do. Usually this type of coding is reserved for advanced software developers. Another disadvantage of this approach is that different forms applications usually have different user interfaces. Each time a user fills out a form, he or she may need to learn a different set of commands and menu items. This learning curve costs the company time and money.

About five years ago, Microsoft recognized the need for a common tool to build forms based on XML technologies. Existing XML-based tools required a thorough understanding of XML, so most information workers had trouble understanding how to use them. Also, most information workers do not know how to write code and, therefore, could not easily use development tools such as Microsoft Visual Studio. Therefore, it just made sense to create a tool that developers and information workers could use to create forms based on XML and that users could use to fill out those forms. That tool is InfoPath. (In Chapter 1, we'll tell you exactly what InfoPath is all about and introduce you to the extensive feature set included in this application.)

Looking at the wealth of features included in InfoPath, especially those added in InfoPath 2007, it also just made sense to create this book. This book is titled *Designing Forms for Microsoft Office InfoPath and Forms Services 2007*

for a reason. It's all about designing forms using InfoPath 2007, as we're sure you have figured out by now. This book will teach you everything you need to know about creating forms using InfoPath 2007 and probably a few things you never thought you needed to know.

Who Should Read This Book

Whether you are an information worker who has created only a few forms in Word or a software developer who is familiar with more advanced coding concepts, if your intention is to learn how to design InfoPath forms, this book is for you. This book will talk about not only the basics of designing forms but also such advanced concepts as writing managed code for InfoPath. As long as you have an understanding of basic form concepts and a desire to learn, you are in the right place. If you want to learn everything you can about InfoPath 2007, you have found the right book.

How This Book Is Organized

This book contains two parts. Part I is all about designing forms in InfoPath. No prior coding experience is required to understand the concepts, so both information workers and developers can use Part I to learn the basics of InfoPath form design. Many chapters build on previous chapters and become slightly more advanced as you progress. For example, in Chapter 4 we discuss advanced controls and customization, but by Chapter 6 we show how to pull external data, such as from a Web service, into your forms. By the time you finish reading Part I, you should know everything you need to know to design an InfoPath form for the InfoPath client application or for the browser without having to write any code.

Part II is about advanced form design. In this part of the book we talk about using more advanced form design techniques, including how to write code for InfoPath. These chapters are geared mainly toward software developers who have some basic coding experience. However, if you are an information worker and you have completed Part I of this book, the second part may interest you as well. In Part II, we talk about such topics as the InfoPath object model (Chapter 15), advanced topics regarding InfoPath Forms Services (Chapter 17), and ways to host InfoPath (Chapter 18).

Conventions Used in This Book

We use a few typographical conventions throughout this book. **Bold** text indicates key topics or terms. The names of features shown in the user interface, such as menu items, appear in *italic* text.

Information that pertains to InfoPath Forms Services is clearly displayed as features in the text. These tips will let you know when certain InfoPath features work differently in browser forms or don't work at all.

Samples

Almost every chapter in this book has one or more samples, which you can download from the Addison-Wesley Web site for this book. Sometimes the samples are InfoPath form templates (.xsn) files, which is the case throughout Part I. In order to use these form templates, you first need to open them in InfoPath design mode and resave them to a local folder. (This will make more sense after you start reading Part I.) Trying to open the form template in order to fill it out without first saving it will result in an error.

Some samples include form (.xml) files in addition to form templates. To open the forms, first open InfoPath, and then open the XML file using the standard *Open* dialog (i.e., click on the *On My Computer* link from the *Getting Started* dialog.) The first time you open one of the sample forms, the dialog shown in Figure P.1 will be displayed. This dialog allows you to choose the form template associated with the form you are trying to open. The text of the chapter indicates the correct form template to use. After you

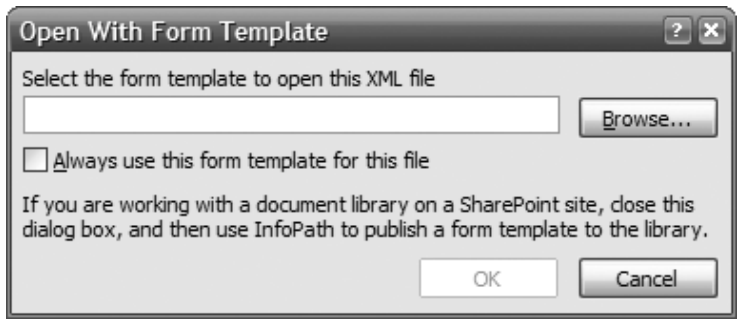

FIGURE P.1 Open With Form Template dialog

choose the form template, click the checkbox *Always use this form template for this file.* After the form is opened, immediately save it. This will prevent you from having to choose the form template each time you open the form.

Some sample form templates define one or more data connections. For these samples to work properly, the external data source must exist. To see if a form template depends on a data connection, go to the *Data Connections* menu item under the *Tools* menu while in design mode. Since there are many types of data connections, we'll describe how to set up each one to successfully preview the form.

- XML document: If the XML (.xml) file exists within the form template (under the *Resource Files* menu item on the *Tools* menu), there is nothing you need to do. If the XML file is external to the form template, you will need to click the *Modify* button on the *Data Connections* dialog to point to your copy of the XML file. You can find the file within the samples for a given chapter.
- Database: A database connection depends on a SQL Server or Access database. If the sample uses a SQL Server database, you must have SQL Server installed and have administrative rights to the SQL Server instance. For an Access database, the chapter will include an Access database (.mdb) file. For either case, click the *Modify* button on the *Data Connections* dialog to update the data connection to point to your database to restore the connection.
- Web service: To use a Web service, you must have Internet Information Services (IIS) 6.0 or higher installed on your computer. The ASP.NET 2.0 ISAPI Web extension must be enabled, and ASP.NET should be configured to render .aspx pages. Copy the Web service code from the sample and paste it into a new ASP.NET Web service project in Visual Studio 2005. If you don't have Visual Studio, you can still create the Web service by creating a text file with extension .aspx within an IIS virtual directory. Check to see if the Web service code requires read or write access to specific directories; you can grant access to those directories or simply update the code to use directories of your choice. Before using the Web service with InfoPath, try navigating to the Web service by using a Web browser on the local machine. Once the Web service works outside of

InfoPath, you can click the *Modify* button in the *Data Connections* dialog to change the Web service connection from the sample to point to your own Web service.

- SharePoint library or list: The prerequisite to using samples with a SharePoint library or list connection is a Microsoft Office SharePoint server. If you have a SharePoint server, ensure you have at least reader rights if the connection is only reading data. Likewise, if the sample form template submits to a SharePoint library, you must have contributor or higher privilege. Ensure the library or list upon which the form template depends actually exists on the server. To use your library or list, edit the connection to point to your server and select the appropriate library or list.

In Part II of this book, most of the samples include code. Those samples that require you to perform special actions (or actions in a specific order) to build the samples include ReadMe.txt files that explain what you need to do.

For sample form templates that include form code, you must do the following. First, find the sample form template (.xsn) and archive (.zip) files with the same name. Extract the archive to a location on your computer by right-clicking on the .zip file and selecting *Extract All.* Next, open the form template in design mode as you would for samples without form code. Select the *Microsoft Visual Studio Tools for Applications* (VSTA) menu item under *Tools* and then *Programming*. If InfoPath cannot find the VSTA project, the dialog shown in Figure P.2 appears. Click the *Browse* button to navigate to the *Visual C# Project* (.csproj) file within the extracted folder. Once the

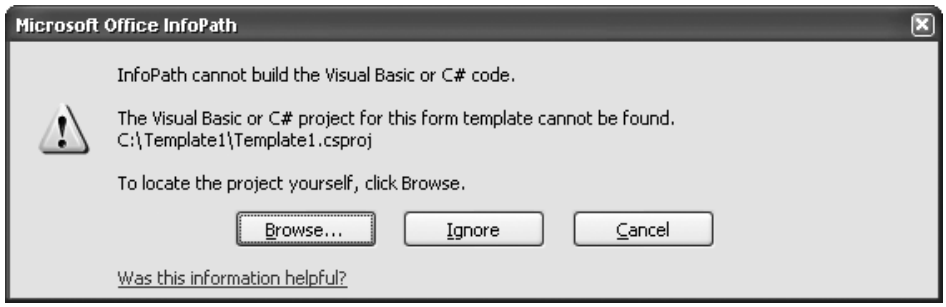

Figure P.2 Dialog shown when InfoPath cannot find the Visual C# project with the form code

Microsoft Visual Studio Tools for Applications window appears, hit F5 to fill out the form while previewing it. The debugger will be automatically attached, so any breakpoints or unhandled exceptions halt form execution.

Some samples in Part II require references to the interop assemblies for InfoPath. In order to set a reference to the correct interop assemblies, open the *Add Reference* dialog in Visual Studio and browse to the install location for Microsoft Office 2007. (This is usually C:\Program Files\Microsoft Office\Office12.) Then, locate the ipeditor.dll file and select it. This will include the Microsoft.Office.Interop.InfoPath interop assembly that you need as well as a few assemblies that aren't needed. In order to be able to install the samples, you will need to remove the references to the ADODB, MSHTML, and MSXML2 assemblies. Some samples use the Microsoft.Office.Interop.InfoPath.Xml assembly. You can also locate this assembly in the install location for Office 2007.

Some code snippets that you see within chapters may differ from the code in the sample. Due to space constraints, brevity in code may have resulted in reformatting or removal of comments or error-handling code that are not required to understand the sample. Regardless, the functionality of the code itself remains unaffected. Note that the code included with the sample form templates is not considered production quality. The code samples, not to mention the form templates themselves, have not been subjected to the rigorous testing you would expect from a company such as Microsoft.

Acknowledgments

The process of writing a book is never a one-person job (or two-person job, in this case). There are always many different people involved in the writing of a book, from the publisher to the reviewers to the product team whose product we are writing about. Therefore, we extend heartfelt thanks to the following people.

First, we want to thank the entire InfoPath product team. Without the InfoPath team and the wonderful application that is InfoPath, this book wouldn't exist. Next, we thank our editor, Joan Murray, who has guided us throughout the 15 months it took to make this project a reality. Also, the following people from Addison-Wesley played a key role in the process: Kim Boedigheimer, Curt Johnson, Eric Garulay, and Jessica D'Amico. We also

thank our copyeditor, Chrysta Meadowbrooke, who did an outstanding job; our project manager, Kathy Glidden; and our project editor, Tyrrell Albaugh.

Next, from the InfoPath team, we thank our Development Manager, Paul Lorimer, for coordinating the InfoPath review process and for providing general support and guidance. Thanks to our Test Manager, Brad Thompson, and our managers Laurent Mollicone (development) and Rodrigo Lode (test) for their support throughout this project. To Jean Paoli, whose vision made InfoPath a reality, we extend our gratitude.

We thank our reviewers on the InfoPath team: Dragos Barac, Andrew Begun, Ned Friend, Jun Jin, Nathaniel Stott, Mike Palmer, Balbir Singh, and Willson Raj David. Thanks for ensuring that the technical content was accurate. We also thank our external reviewers (those who are not members of the InfoPath team): Pedro Serrano, Joe Kunk, and Susan Ramlet.

In addition, many other people helped us along the way, whether with technical information or advice on what topics would be most useful to readers.

- Dragos Barac helped us with user roles, workflow, and hosting. We extend a special thank you to Dragos, who went out of his way on many occasions to help us. In fact, he reviewed most of our chapters and provided extremely valuable feedback. Dragos, we thank you very much for the contribution you have made to the success of this book.
- Eilen Hao helped us understand workflows in Microsoft Office SharePoint Server.
- We extend our thanks to the following people for their help with InfoPath hosting: Petru Moldovanu, David Airapetyan, Yael Peled, DeVere Dyett, Gary Hsu, Balbir Singh, and Alnur Ali.
- Nathaniel Stott gave us his help with InfoPath e-mail forms.
- For providing us with information about workflow, we thank Ned Friend, Noah Edelstein, and Michael Dalton.
- For helping us with COM and managed add-ins, we thank Petru Moldovanu, Frank Mueller, and David Vierzba.
- We thank Mike Palmer, Andrew Begun, Silviu Ifrim, Eric Korn, and Nima Mirzad for their help with the InfoPath object model.
- Jeff Bienvenu and Aparna Suripeddi shared their knowledge of Information Rights Management.

- Mary Smith provided pointers to the Office Customization Tool.
- Without Travis Rhodes and Nick Dallett, we couldn't have included topics on the data connection library, centrally managed connection library, and Universal Data Connection (UDC) files.
- Natalie Eason helped set up and convey all sorts of information on Visual Studio Tools for Office (VSTO).
- The InfoPath XML Schema guru, Phyllis Lai, offered her insight on data source intricacies.

Hagen would like to thank Scott for his mentorship throughout writing this book. He is not only a role model but also a great personal friend. I could not have asked for a better coauthor as well as coworker. Scott, thank you for all you have taught me in authorship.

Scott would like to thank Hagen for his dedication and persistence throughout the long and difficult process of completing this project. Your hard-working attitude and easygoing nature make you a joy to work with. We started this project as close personal friends and made it through all the tough times unscathed.

Last, but certainly not least, we'd like to thank our families. Scott thanks his wife, Andrea, and his two sons, Sean and Bradley, for being so supportive during the 15 months it took to write this book. Without you three, life would have been much more difficult during this time. Hagen thanks his girlfriend, Jaime, for being by his side throughout one of the biggest undertakings of his life. Hagen also thanks his parents, Christine and Stuart, for their constant encouragement and support, especially his dad for being his biggest fan and the first to preorder this book!

Feedback

After you read this book, we would love to hear what you think about it—both positive and negative. This will help us improve future revisions. Also, feel free to send us any questions you may have. We would be happy to help. To contact us, simply send us an e-mail at DesigningIPForms@hotmail.com. If you have a question, either Scott or Hagen will reply as quickly as possible. We hope that you enjoy the book and that it makes the process of designing InfoPath form templates much easier.