Foreword by Nick Dallet and Jean Paoli, Microsoft Corporation

Designing Forms for SharePoint and InfoPath

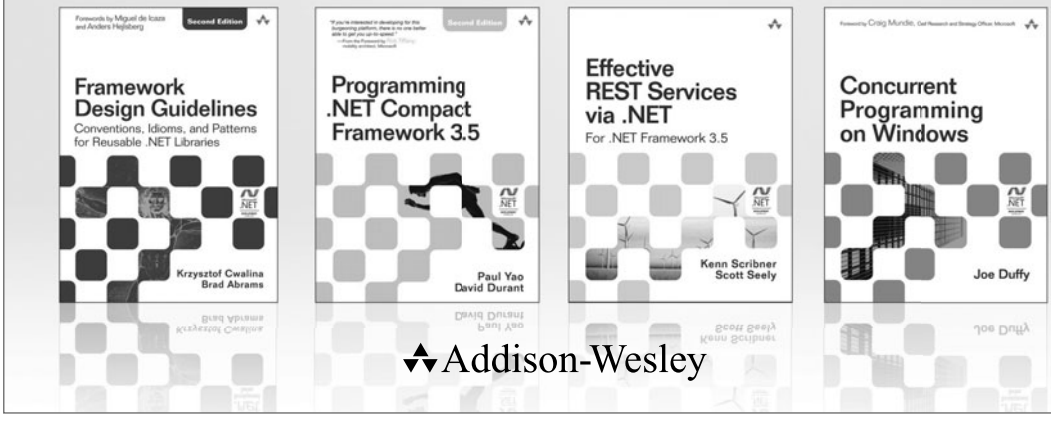
Using InfoPath Designer 2010

Microsoft .NET DEVELOPMENT SERIES

Scott Roberts
Hagen Green
Jessica Meats

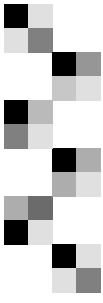
Designing Forms for SharePoint and InfoPath

Microsoft® .NET Development Series



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

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



Designing Forms for SharePoint and InfoPath

Using InfoPath Designer 2010

- Scott Roberts
- Hagen Green
- Jessica Meats

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Roberts, Scott, 1969-
Designing forms for SharePoint and InfoPath using InfoPath Designer 2010 / Scott Roberts, Hagen Green, Jessica Meats.
p. cm.
Includes index.
ISBN 978-0-321-74360-2 (pbk. : alk. paper)
1. Microsoft InfoPath. 2. Microsoft SharePoint (Electronic resource) 3. Business—Forms—Computer programs. I. Green, Hagen. II. Meats, Jessica. III. Title.

HF5371.R634 2011
651.7'4028553—dc22

2011008186

ISBN-13: 978-0-321-74360-2
ISBN-10: 0-321-74360-1
Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
First printing, May 2011

To my wife, Andrea, and my two sons, Sean and Bradley. It is due to your constant love and support that I was able to complete this project. Thank you for always being there for me no matter what project I undertake. I love you all a million times more than you could ever possibly know.
—Scott

To Jaime, my love and my life, for putting up with my two jobs and helping me stay strong through the end. And to my parents, Stuart and Christine, for their perseverance in love and for being the best parents anyone could ask for.
—Hagen

To Nick Borrett for a summer administration job that taught me the value of well-designed electronic forms. And to the friends who put up with the excuse, "Sorry, I've got a book to edit. Can we reschedule?"
—Jessica

This page intentionally left blank

Contents at a Glance

Contents

This page intentionally left blank

Figures

Tables

Foreword

Designing Forms for SharePoint and InfoPath is a hands-on introduction to Microsoft InfoPath.

Like the Web, InfoPath continues to grow and change and has evolved toward a holistic story around rapid development of workflow applications on Microsoft Office SharePoint Server. Together with Microsoft SharePoint Designer 2010, InfoPath 2010 facilitates creation of end-to-end solutions that feature powerful forms together with enterprise-scale workflow and access to key business data. InfoPath was designed, at its core, as a powerful XML editing engine that enables end users to interface easily with data.

The present book is based largely on *Designing Forms for Microsoft Office InfoPath and Forms Services 2007*, which was written by two distinguished members of the InfoPath product team who designed, implemented, and tested many of the core features of the product. In revising the book for this edition, we have tried to keep the deep encyclopedic knowledge of InfoPath embodied in that work, while also restructuring the material to focus attention on the SharePoint application development experience. The book has been updated as well to call attention to the new features introduced in InfoPath 2010 to help build more powerful SharePoint applications—features such as:

- Customizing the forms used to create, view, and edit SharePoint list items

- Dynamic queries to REST Web services
- The InfoPath Form Web part, which allows you to create powerful Web parts without writing code, and to connect them with other Web parts to create data mashups

What's interesting and unique about InfoPath is the type of information it allows people to gather. InfoPath lets organizations design and edit "semistructured" documents, or documents that have regions of meaning, in the same way that columns in a database have meaning. While the program provides great design and editing capabilities for traditional forms such as purchase orders and equipment requests, InfoPath innovatively yet squarely targets information that historically has been more difficult to capture, such as business-critical data contained in sales reports, inventory updates, project memos, travel itineraries, and performance reviews. InfoPath was born as a tool for editing XML, and XML is about creating documents in which the content is delimited, or set apart, by tags that explain the meaning of each piece of content. With XML, documents can become a source of information as rich as a database, enabling search, processing, and reuse. The underlying structure of the information in an InfoPath template is described using a **schema.** A schema describes how the data is constructed, in the same way that a blueprint describes how a building is constructed. In SharePoint, these schemas are represented as content types in lists and libraries, and InfoPath provides a consistent way to author forms and logic that turn these lists and libraries into powerful applications that automate processes that previously required many manual steps.

Microsoft's long-term vision for InfoPath has always been about more than the ability to rapidly create forms: It is about building complete end-to-end applications using the power of XML and workflow. Together with the powerful collaboration features of SharePoint, InfoPath is a key part of the toolset you need to rapidly create applications that meet your enterprise application needs. InfoPath 2010 and InfoPath Forms Services in SharePoint 2010 empower business users to automate their own business processes that collect, manage, and share information. IT, developers, and power users can create powerful business applications on the SharePoint

platform using InfoPath forms to interact with external data, to drive workflow, and to enhance Web pages.

—*Jean Paoli*
General Manager, Interoperability and XML Architecture, Microsoft Corporation
Co-creator of the W3C XML 1.0 Recommendation
Co-creator of Microsoft Office InfoPath

—*Nick Dallett*
Program Manager, InfoPath 2003, InfoPath 2007, InfoPath 2010

This page intentionally left blank

Preface

It Just Makes Sense

Over the past decade or so, 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 2010. 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 eight 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 2010, it also made sense to create this book. This book is titled *Designing Forms for InfoPath and SharePoint* for a reason. It's all about designing forms using InfoPath Designer 2010, 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 Designer 2010 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 2010, 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 Designer. It introduces the concepts and explains most of the principles needed to design rich forms. 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 5 we discuss advanced controls and customization, but by Chapter 7 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 Filler application or for SharePoint Server 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 13), advanced topics regarding InfoPath Forms Services (Chapter 14), and ways to host InfoPath (Chapter 15).

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 Designer 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. 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 choose the form template, click the checkbox *Always use this form template*

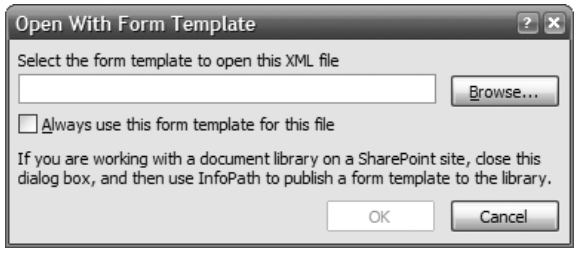


FIGURE P.1: Open with Form Template Dialog

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 later 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 that 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, 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 *Microsoft Visual Studio Tools for Applications* window appears, hit F5 to

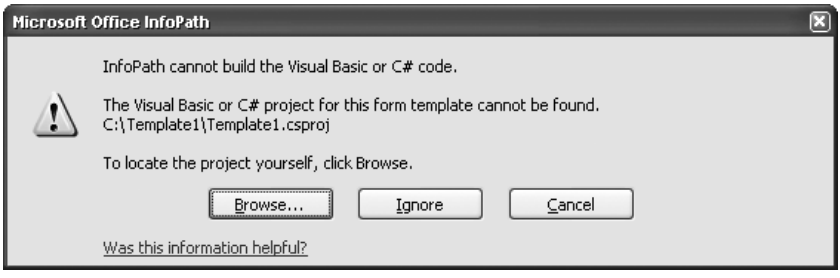


FIGURE P.2: Dialog shown when InfoPath cannot find the Visual C# project with the form code

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 2010. (This is usually C:\Program Files\Microsoft Office\Office14.) 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 2010.

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.

This page intentionally left blank

Acknowledgments

The process of writing a book is never a one-person job (or three-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: Curt Johnson and Eric Garulay. We also thank our copy editor, Kelli Brooks, who did an outstanding job, and our project editor, Anna Popick.

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.

Jessica Meats would like to thank Nick Dallett and his team, as well as others at Microsoft for their comments and feedback throughout the writing process. Thanks goes out to Christopher Brotsos, Bojana Duke, Laura Harrington, Anson Hidajat, Silviu Ifrim, Petru Moldovanu, Philip Newman, Shirish Pulikkal, Roberto Taboada, Andrew Whitechapel, Daniel Witriol, and Richard Witte. She would also like to thank Chris Marchal for his help around the new authentication options in SharePoint 2010. Also to Michael Lotter for his contribution of content around the SharePoint and InfoPath integration content that's included in Appendix A of the book.

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 the three of them, 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, we 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.

About the Authors

Scott Roberts is a Principal Development Lead in the SQL Server group at Microsoft Corporation. Previously, he was a Development Lead on the InfoPath team, and was involved with InfoPath since its inception. While in the InfoPath team, he led the development of many features for designing and editing InfoPath forms. Scott is also the author of *Programming Microsoft Internet Explorer 5* (Microsoft Press, 1999) and numerous technical articles and publications.

Hagen Green is a Senior Program Manager at Microsoft Corporation, working on the Windows Phone Communications team. He previously led the User Experience Platform test team in SharePoint Foundation. He contributed chapters on InfoPath to *Visual Studio Tools for Office: Using Visual Basic 2005 with Excel, Word, Outlook, and InfoPath* (Addison-Wesley, 2006), and *Visual Studio Tools for Office: Using C# with Excel, Word, Outlook, and InfoPath* (Addison-Wesley, 2005).

Jessica Meats is a Partner Technology Advisor for Microsoft Corporation specializing in Microsoft SharePoint Server, Office and Business Intelligence. She joined Microsoft in 2008, straight out of university, as a Partner Technology Specialist and chose InfoPath as one of her specialities, wishing that some of the companies she'd done temp work for as a student

had used such a product. Her job involves helping partner organizations work with these products and aiding new partners quickly gain the skills needed to build a practice around delivering these solutions. Jessica is the author of a science-fiction thriller, *Child of the Hive* (Book Guild Publishing Ltd., 2009). In her spare time, Jessica writes fiction, juggles fire, and studies kung fu.

This page intentionally left blank

1
Introduction to InfoPath 2010

What Is InfoPath?

The rapid adoption of XML-based technologies over the past decade or so has precipitated the need for a tool that helps end users interact with and share XML data. On August 19, 2003, as part of the Microsoft Office System 2003, Microsoft shipped a new application created to fill that need—Microsoft Office InfoPath.

InfoPath may be one of the least known yet most appropriate platforms for gathering data in the Microsoft Office suite of applications. But InfoPath's popularity has been on a sharp rise. Anyone from an information technology (IT) manager tracking a purchase request workflow to little Johnny tracking his music collection can benefit from what InfoPath offers. Before InfoPath, programs such as Word, Excel, or Access may have been used for these tasks. There is nothing wrong with using any of these other programs to build forms. However, none of these applications were built with forms in mind, so they don't provide the ease of use and power that InfoPath has when it comes to creating a form based on XML data. The goal of this book is to make you an expert InfoPath form designer. Whether you are a novice, an information worker, an experienced developer, or anyone in between, you have found the right book to help you get the job done with InfoPath 2010.

You may be wondering why Microsoft created this new tool. Doesn't Microsoft already have other form design tools such as Word, Excel, and Visual Basic .NET, to name a few? While it is true that you can build forms with these tools, none of them are based solely on XML technologies, nor were they created exclusively to edit XML data. Both Word and Excel can interact with XML data, but they do not offer structural editing of the XML data to the extent that InfoPath does. Visual Basic .NET requires you to have at least a basic understanding of application development. InfoPath does not require you to be a developer in order to build a form. Anyone who has experience using Word can create a form in InfoPath Designer.

Thanks to the familiar Microsoft Office user interface (UI), ramp-up time to productivity is minimal. Users with little or no programming experience can use InfoPath Designer to create form templates. InfoPath Filler users can also fill out forms based on those templates. Since InfoPath was designed (no pun intended) to be used by almost anyone with little or no training, form designers have all the tools needed to deliver form templates that are easy to use. Of course, if you are a developer, you may also want to customize your form templates, for example, by writing script or managed code using InfoPath's extensive object model (OM).

Form Templates and Forms

Form template: The blueprint for a form, which you create in InfoPath Designer. Many forms that are unrelated in content can be created with the same form template. The form template defines the look-and-feel and functionality of a form.

Form: An instance of a form template that users fill out by using InfoPath Filler or an Internet browser connected to InfoPath Forms Services.

Using InfoPath Designer, you can rapidly build form templates that interact with XML data and XML Schema (XSD) with little or no code required. Then users can open InfoPath Filler to fill out forms based on the form templates you created. Because InfoPath is a Microsoft Office

product, the UI is as familiar to users as Word. In fact, InfoPath provides a lot of the same features as Word (e.g., spell checking, rich text editing), so filling out forms in InfoPath is as easy as drafting a document in Word. The main difference is that when users are filling out an InfoPath form, they are actually editing XML data.

However, the value of InfoPath goes way beyond its ability to edit XML data directly. In fact, its usefulness as a data-gathering and management tool is one of the most powerful aspects of InfoPath. Also, because the InfoPath form templates you create can be based on industry-standard XML Schemas and integrated with Web services, data stored in InfoPath forms can be integrated with existing processes in your organization and across other organizations.

Let's look at an example. Say that you run a sales organization. You have an existing database that contains customer and sales data. Your sales manager wants you to create two different forms—one for salespeople to input customer information and another for them to input sales data. Before InfoPath, there were many ways to create these forms. However, since the data comes from a back-end database, all of these methods involved writing some code. And you would typically have to duplicate this code from one form to the next.

With InfoPath, creating these forms is much easier. InfoPath allows you to create form templates that connect to existing databases or Web services, thus removing the need to write code to access the data. This part is done for you. InfoPath also allows you to easily merge the data gathered in these forms so that you can create aggregate reports for upper management. Again, InfoPath enables you to do this without writing any code at all, unlike other form tools. InfoPath includes an extensive OM, so it's highly customizable. Using InfoPath, you can enable almost any data collection scenario you can dream up. The scenarios can be extended by combining InfoPath with SharePoint workflows to rapidly develop applications to automate business processes in the enterprise. Deeper integration with SharePoint is a key development point in the latest release, which makes it easier to link your data collection forms to the processes they drive.

As your business needs change, so should your software. InfoPath is designed to cater to these ever-changing data collection needs. Built on

World Wide Web Consortium (W3C) standards, XML and its associated technologies help InfoPath deliver on its promises. A few of the greater benefits of XML include the following:

- Provides a semistructured data format
- Is human readable
- Is easy for programs to process
- Works across disparate systems

Technologies encompassing XML include Extensible Stylesheet Language Transformations (XSLT) for transforming the XML to another format such as HTML, XML Path Language (XPath) for selecting specific parts of the XML data for processing, and XSD for defining precise rules for how the XML structure and data should look. InfoPath seamlessly incorporates each of these technologies into the experience of designing and filling out forms. For example, thanks to XSLT, one of the most important benefits of using InfoPath is the ability to separate data from how it is displayed. With the drag-and-drop UI of the InfoPath Designer, you can completely change the look-and-feel of a form without changing the underlying data that is saved when a user fills it out. Another advantage is being able to merge data from multiple forms. This is especially useful when creating a report, for example.

As we mentioned, InfoPath offers a plethora of built-in features geared toward collecting data that would require custom code in other form design tools. With InfoPath, for example, you can use data validation and conditional formatting to ensure that users enter valid data before the form is submitted to your back-end databases or Web services. You can also design your forms to collect only the data you need. Traditional forms include many parts that don't apply to everyone. For example, employment applications often contain sections relevant only to people with college degrees. However, these forms have to include sections for this information since the forms are static by nature. These forms may contain instructional text such as "If you chose Yes for question 5, please enter more information here. Otherwise, skip to question 6." With InfoPath, you can design dynamic forms that contain the additional sections only if needed.

Using our example, when a user chooses Yes to question 5, a section of the form that wasn't visible suddenly becomes available. Also, the static employment application may not provide enough room to fill in a complete employment history. In this case, users must either not include the additional information or provide it on an additional form. With InfoPath forms, your users can fill in as much data as they need since your forms can be built to grow dynamically. When your users need more space to fill in additional data, the form can expand to meet their requirements.

These are just a few aspects of InfoPath that makes it a more powerful tool for XML-based forms than other applications. We will talk about these benefits and more throughout this book. First, however, we'd like to give you an overview of the evolution of InfoPath through four successive versions—InfoPath 2003, InfoPath 2003 Service Pack 1, InfoPath 2007, and InfoPath 2010. This will not only give you some insight into the product's background but also introduce you to several of its features.

InfoPath 2003

To provide reasonable context for discussing new InfoPath features, let's look at the different versions of the product and what each brings to the table. Version 1, officially released as InfoPath 2003 on August 19, 2003, is Microsoft's inception into the XML forms and data collection markets. As such, it bundles competitive features to allow for a rich form experience. This section recaps the highlights of InfoPath 2003.

The InfoPath design mode is all about creating nice-looking forms that include basic functionality. The design surface, or view, is similar to that of Microsoft Word. The content of the view flows from left to right, top to bottom, and serves as the presentation layer for the form's data. Absolute positioning, such as what PowerPoint offers, would make InfoPath form design cumbersome when accommodating different resolutions and monitor sizes. Therefore, **layout tables** are used instead as the positioning paradigm of choice—equivalent to that of many professional Web sites. Tables offer a familiar and flexible option when demanding pixel-by-pixel, picture-perfect parity.

For those of us with a less than acute sense for visual design, **color schemes** spice up even the most boring forms. A single click on the *Bright Blue* color scheme adds a subtle blue color theme to parts of your form template's controls and layout tables. If you don't care for the effects of *Bright Blue*, you can choose another scheme at your convenience at any time. Color schemes more dramatically affect Repeating Table controls more than other controls. The header and footer backgrounds, text, and table borders comply with the current theme.

InfoPath 2003 introduced a general set of controls for inclusion in forms. Figure 1.1 shows the *Controls* task pane. As you would expect, the InfoPath design mode includes controls such as Text Box, List Box, Drop-Down List Box, Check Box, Bulleted List, Numbered List, Plain List, Option Button, and Button. The Section, Optional Section, Repeating Section, and Repeating Table controls support InfoPath's structural editing features. These

Figure 1.1: Controls task pane in InfoPath 2003

editing features allow the end user, while filling out the form, to modify the structure of the XML data as the XML Schema allows. Structural editing is what makes InfoPath such an attractive offering compared with its alternatives. The Hyperlink, Expression Box, Picture, and Rich Text Box controls round out the controls available in InfoPath 2003.

We mentioned earlier that the data and the presentation layers are two disparate concepts. We've already discussed the presentation layer, but we haven't yet focused on the data. When we refer to "data," we really mean "data and structure." XML consists of hierarchical elements that may or may not contain text content. The definition for the XML structure is bounded by the rules of the XML Schema. (We will talk more about XML Schema in Chapter 4.) InfoPath abstracts away the XML Schema details through the *Data Source* task pane (DSP; see Figure 1.2). The DSP is a design-mode feature that shows what the XML Schema allows. If you don't have

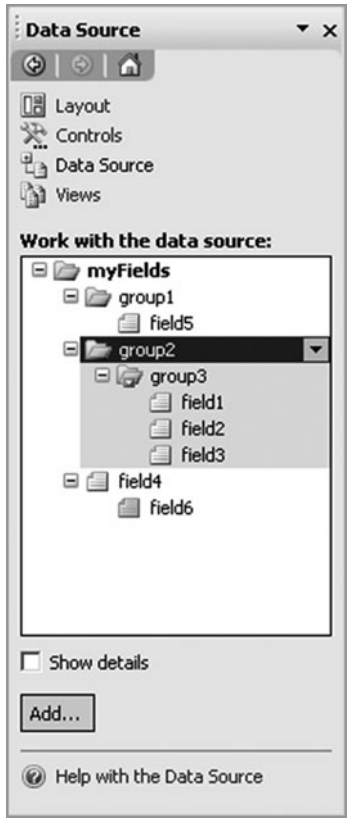


FIGURE 1.2: Data Source task pane in InfoPath 2003

an XML Schema lying around, there's no need to worry. InfoPath automatically builds the XML Schema as you insert new controls onto the designer view. Inserting a control into the form may create either a **field** or a **group** in the DSP. This process populates the DSP based on which controls are selected. The newly inserted controls are **bound** to the newly created data source. The word **binding** describes the connection between what the user sees and what data is stored behind the scenes. We'll talk more about binding in Chapter 4.

Binding sounds easy, right? It is—for most situations. But due to the flexibility of hooking up any data source to any controls, getting into odd binding dilemmas is inevitable. **Design-time visuals** are icons displayed on your form controls as you design your form template. These visuals serve as handy reminders to a designer that the form may be broken. We will discuss this versatile feature in more detail in Chapter 4.

Besides the visual zest, let's look at the different ways to pull external data into an InfoPath 2003 form. **Secondary data sources** are used to connect to databases, SharePoint libraries or lists, and Web services. Say your car registration form template has a Drop-Down List Box control containing all 50 states in the United States. When creating your form template, you could enter each state manually when creating the list items for the control, but it's probably easier to point the Drop-Down List Box control to an existing database. SharePoint servers are somewhat ubiquitous in high-tech small business. Pulling a real-time list of manager names or part numbers into your form allows it to always be up to date with no intervention on your behalf.

Web services have created a growing sensation throughout the Internet community for the last several years. Any popular search engine will yield innumerable results when you query for Web services. Ironically, many search engines also expose a Web service for you to do the searching. InfoPath supports connecting to these Web services without you, as the form designer, writing a single line of code. All you need to know is where the Web service resides. So forget about details like the Web Service Definition Language (WSDL), input parameters, and output parameters that you would need to understand if using other form tools. However, if you wish to deal with those aspects of the Web service using InfoPath 2003, you can access them by writing script code against the InfoPath OM.

The InfoPath OM in version 2003 is designed as a programmatic extension to many features provided in the InfoPath UI and some features that are not. The OM is not a superset but rather a supplement to the functionality available from the UI. To access the OM in InfoPath 2003, you could choose to write either JScript or VBScript in the Microsoft Script Editor (MSE). MSE is the Microsoft Office–bundled development environment of choice for scripting languages. InfoPath even has an extensive help system built into MSE that provides on-demand, context-sensitive help while scripting InfoPath objects.

So let's say that we've made the final touches on our new InfoPath form template by adding some JScript code using MSE. We're all ready to roll the form template out to our team. It's time to **publish** our form template. Publishing is a short process to deploy the form template to a location accessible by users. A few steps through a wizard is all it takes to properly put your template on a network file share, SharePoint site, Web server, or other shared location.

We've exhausted our brief talk of the InfoPath design mode for now. Let's peek at some of the important features you can find while filling out a form. As mentioned earlier, the UI is instantly familiar. You can expect the standard Microsoft Office set of features, such as cut, copy, and paste; a dizzying array of font formatting options; spell checking; find and replace; and clip art (to name a few).

What's so special about InfoPath? It sounds a lot like Word in many ways. However, InfoPath has two key advantages: An InfoPath form uses XML data, and InfoPath offers data validation, which is not found in Word. If you are an application developer, you will appreciate the tight integration between XML data type validation and the InfoPath UI. You can think of data validation as a way to guide your users to format their input as you determine. For example, if you want a phone number, you could easily use a text box that accepts any text the user types. But with data validation, you can disallow a phone number if it's not ten digits or has non-numeric characters. InfoPath's validation model is also extensible by using the InfoPath OM and writing custom form code.

A popular way to make your form template more interactive is to use **conditional formatting.** Take an expense report form that adds up prices

from a list of items purchased during a business trip. You can design the form template so that if the sum of expenses is more than $1,000, the text for the total value will change from a calm black to an attention-getting bold red. Conditional formatting can be used throughout your form template to not only change fonts but to show and hide regions of your form as well.

It's great to have your users happily filling out your forms. But how do you make sense of the data saved in XML files? InfoPath has convenient, built-in reporting features to bring together data for you and your users. The first of these reporting functions is form merge. When a user selects multiple saved forms, InfoPath aggregates the data from each saved form into a single new one. A common use for this feature is rolling up status reports for the boss. InfoPath also exposes custom form merging for advanced developers by allowing them to write their own custom XSLT. InfoPath 2007 made some great improvements to this customization experience by supporting custom merge actions in design mode. Another way to report InfoPath form data is to export it to Excel. Sometimes InfoPath may not be the best environment to do complex spreadsheet functions like pivot tables. Exporting the data to Excel is a simple click away, and you can customize the process by exporting specific parts of the data. Once data is exported to Excel, you can create pivot tables and charts to help you analyze the data. Since InfoPath is a powerful data-gathering tool, reporting options such as form merging and export to Excel help round out the story for bringing data together in a useful way.

InfoPath 2003 Service Pack 1

Even before InfoPath 2003 was out the door, InfoPath 2003 Service Pack 1 (SP1) was under construction; it was finally released on July 27, 2004. Convenience features, such as a table-drawing tool and the Format Painter, were added in SP1. Integration was introduced or enhanced, such as Web service support for ADO.NET DataSet objects. New controls, such as the File Attachment and Master/Detail controls, were added. Finally, InfoPath was also made more robust by adding features like crash recovery while filling out a form. Even if the computer doesn't crash, there's always the

possibility that a power outage may occur (or the battery may die on your laptop). If you didn't think it could get any better, InfoPath 2003 SP1 was available as a free upgrade on the Web and fully backward compatible with 2003.

In tune with the numerous enhancements you would expect from a new release, InfoPath 2003 SP1 provided a much richer design experience. A lot of features that were already available in other Microsoft Office applications were integrated into InfoPath 2003 SP1. The new *Insert Layout Table* button (available on the *Standard* toolbar, as shown in Figure 1.3) enables you to quickly insert a table with as many as five columns and four rows.

Another tool that was added in SP1 to make table creation easier is the new *Tables* toolbar (see Figure 1.4). This toolbar, which includes a table-drawing tool that can be turned on by clicking the *Draw Table* button, simplifies the process of creating tables tremendously. Using the table-drawing tool, you can easily draw a table any way you want without having to use the *Insert Table* dialog. The *Tables* toolbar also allows you to quickly and easily change various properties of any table, such as borders and shading. Considering that most InfoPath forms rely heavily on table layout, these tool improvements were welcome additions.

Another very useful tool added in SP1 is the **Format Painter.** You may already be familiar with this tool since it's also available in other Microsoft

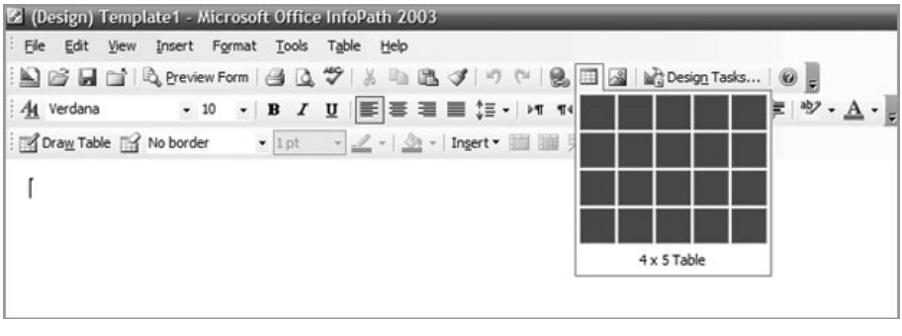


FIGURE 1.3: Insert Layout Table toolbar item

FIGURE 1.4: Tables toolbar

Office applications, such as Word. This tool allows you to quickly copy formatting from one part of the form template's view to another. You can select text or a control and copy its formatting to other text or another control in the form template.

The new table tools and the Format Painter are just two of the many features added to InfoPath 2003 SP1 in response to customer feedback. There are many more features that were added as well. Another feature added in response to feedback from international markets is support for complex script and right-to-left (RTL) languages. With InfoPath SP1, you can create forms that work with languages requiring complex script and/or RTL reading such as Arabic, Hebrew, Chinese, and so on.

InfoPath 2003 SP1 also added a slew of new controls to the *Controls* task pane. As with the other features added to improve the design experience, the decision about which controls to add in SP1 was based solely on customer feedback. (We will talk about each of these controls in more detail in Chapters 2 and 4.) The following controls were added in InfoPath 2003 SP1:

- Master/Detail
- File Attachment
- Choice Group
- Repeating Choice Group
- Choice Section
- Recursive Section controls
- Scrolling Region
- Vertical Label
- Ink Picture
- Custom controls

One of the most exciting additions to the *Controls* task pane was the support for custom controls. As of InfoPath 2003 SP1, you can use ActiveX technologies to build your own custom controls for InfoPath. Support for custom controls expanded in InfoPath 2007 and further expanded 2010 to include the ability to build controls without code.

Data source enhancements, such as support for choice and recursion, were added in conjunction with the new Choice Group and Recursive Section controls. Customers clearly use industry-standard XML Schemas with choice constructs. As a result, InfoPath 2003 SP1 fully supports the XML Schema choice element and the Choice control that binds to it. Recursion is also very common in real-world XML Schemas. Recursion is when a schema element can contain itself. For example, a schema describing a hierarchy of employees can be described only by using recursion (since an employee can have employees under him or her). The Recursive Section control naturally supports recursion by allowing you to insert the same item as a child of itself. In SP1 there is no extra burden on the form designer to get this working properly, but in InfoPath 2003 it was a bit of a technical hurdle.

One of the most useful features introduced for nondevelopers in SP1 was the **Rules** feature. Overwhelming customer feedback was clear: Too many seemingly simple things required writing script. For example, showing a dialog or changing views required a few lines of script. With rules, setting up your form template to show a dialog is a few button clicks away. A more complex scenario would be to switch views, submit to a database, and set the value of a field—all in one rule and without code! Introducing rules to the masses meant less programming and more form function.

Creating a rich, functional form template is sufficient until you realize that your one form template does not fit all users. Sometimes data in the form shouldn't be shown to specific users. Imagine a form that tracks top issues, and each issue has a checkbox to set the entry as private. With InfoPath 2003 SP1, you can assign users to private and public **roles** and, in combination with conditional formatting, decide whether or not to show private data based on the current user role.

When it comes to trusting the data in a form, little can supersede the security offered by **digital signatures.** Highlights such as nonrepudiable partial signatures, cosigning, and countersigning allow the form template designer a greater level of freedom while not sacrificing the concept of "secure by default." Factor in the extended programmability support for digitally signed forms, and you have a complete solution for protecting form data.

Arguably one of the most anticipated features in the SP1 release was the introduction of **managed code** support. With the world buzzing about .NET, customers could not wait to get their hands on the free Visual Studio Toolkit plug-in. Along with managed code support, a revised and expanded programming interface, or object model was included. In Chapter 13 we will introduce the OM, which was introduced in InfoPath 2007.

Not only can InfoPath simplify the creation of forms but, as mentioned earlier, since InfoPath forms are based on XML, they can be integrated easily into existing business processes and workflows. By using InfoPath with Microsoft Office SharePoint Server or Outlook, you can quickly build workflow processes in a fraction of the time it would take with other forms development tools. We will talk in more detail about workflow in Chapter 10.

With the growing popularity of Tablet PCs, it seems only natural that InfoPath would include support for these devices. InfoPath 2003 included limited support for Tablet PCs that was expanded in SP1. In design mode, the Ink Picture control enables you to include sections in your form template that are exclusively for handwriting with the Tablet PC pen. When editing the form, all controls support an ink-entry mode that allows you to write in them with your Tablet PC pen. Your writing will then be converted to text. While in ink-entry mode, InfoPath supports many of the same gestures to edit and correct text you may already be familiar with.

SP1 also brought enhancements in form template deployment. InfoPath 2003 made it very difficult to send form templates by e-mail for users to fill out. Form template designers had to jump through a few technical hoops to get it to work properly. It was also inconvenient for recipients as well. The addition of **e-mail publishing** in SP1 opened up the potential for anyone with an e-mail address and InfoPath to fill out a form. Deploying a form template via e-mail is only a quick click away. Once a template is deployed via e-mail, users can open the e-mail message in Outlook and start filling out the form.

In addition to enabling you to build more elaborate form templates, new features also enhance the process of filling out forms. One very useful new feature is the *Fill Out a Form* dialog. This dialog, shown in Figure 1.5, gives you a starting point for editing forms and a central location for managing them. (These capabilities have now been included in the backstage

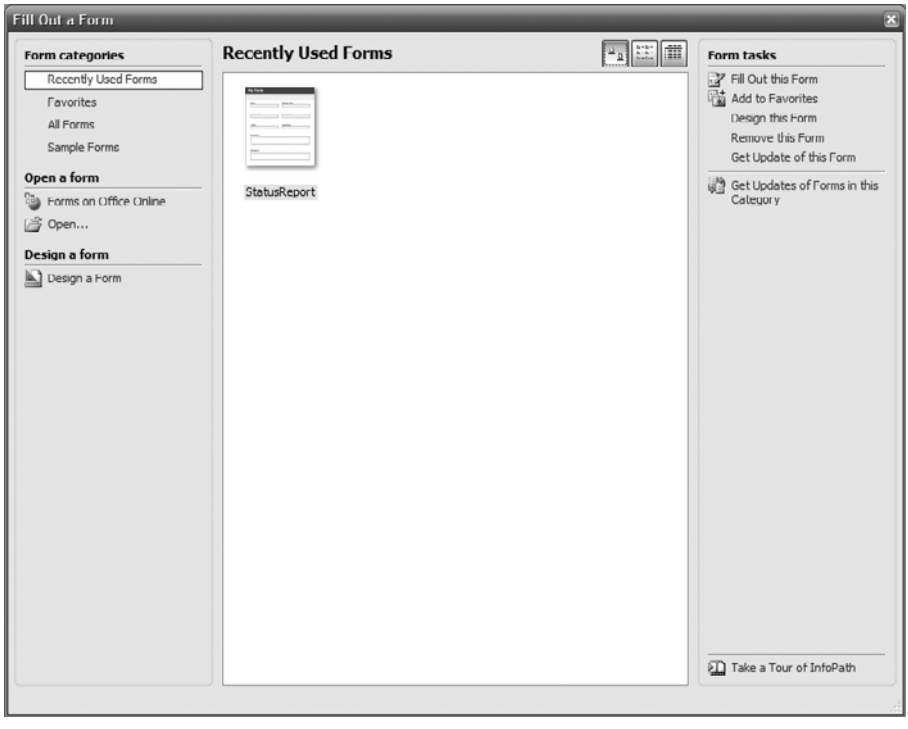


FIGURE 1.5: Fill Out a Form dialog in InfoPath 2003 SP1

view available from the *File* tab.) Using the dashboard, you have easy access to forms you've filled out before by clicking the *Recently Used Forms* link in the left-hand side of the dialog. You can also add forms to your list of favorites by clicking the *Add to Favorites* link on the right-hand side and keep track of your favorites via the *Favorites* link on the left. You can download forms from Office Online by clicking the *Forms on Office Online* link or design a new or existing form template by clicking either the *Design a Form* or *Design this Form* links. This new dialog greatly simplifies the process of creating, filling out, and managing forms.

InfoPath 2007

The next version of InfoPath was InfoPath 2007, released on November 30th, 2006. As with previous versions, all feature additions in InfoPath

2007 were based on customer feedback. InfoPath 2007 not only included improvements in the core client application but also finally satisfied the number one customer request—the ability to fill out forms without having to install the InfoPath client application on every user's computer. To satisfy this request, InfoPath 2007 introduced a sister product called InfoPath Forms Services, included as a feature of Microsoft Office SharePoint Server 2007 Enterprise.

InfoPath Forms Services brought reality to the ultimate deployment story: publishing your InfoPath form template to the Web. Anyone anywhere, on any device, could now fill out your form. Feedback dictated that the InfoPath Forms Services would need to reach UNIX users running Mozilla, Mac OS folks with Safari, and even always-on-the-go businesspeople with their HTML-enabled PDAs and smartphones. Forms were filled out in full fidelity with data validation, conditional formatting, and even custom form code, to name a few features. (Figure 1.6 shows an InfoPath form being filled out in a browser.)

The key to making InfoPath Forms Services successful was to maintain as much compatibility as possible with the InfoPath client application. As a result, most form templates created with the InfoPath client can simply be reused "as is" in Forms Services. The InfoPath 2007 client application supports the few limitations of InfoPath Forms Services by offering two modes: InfoPath-only or browser-enabled form templates. In the browser-enabled mode, some features that are not supported (either in full or in part) are disabled or removed from the InfoPath design mode UI. When designing a new form template, you choose whether to design a client or browser-enabled form template, with the ability to change it at any time. To support switching from an InfoPath-only to a browser-enabled form template, the design mode offers the **Design Checker,** which reports form features added during InfoPath-only mode that are incompatible with InfoPath Forms Services. The Design Checker also helps maintain compatibility with previous versions of InfoPath.

You can expect your browser-enabled forms to behave similarly in InfoPath Forms Services as you would with the InfoPath client application. All conditional formatting, data validation, rules, multiple views, digital signatures, security settings, and most controls are respected. There are so

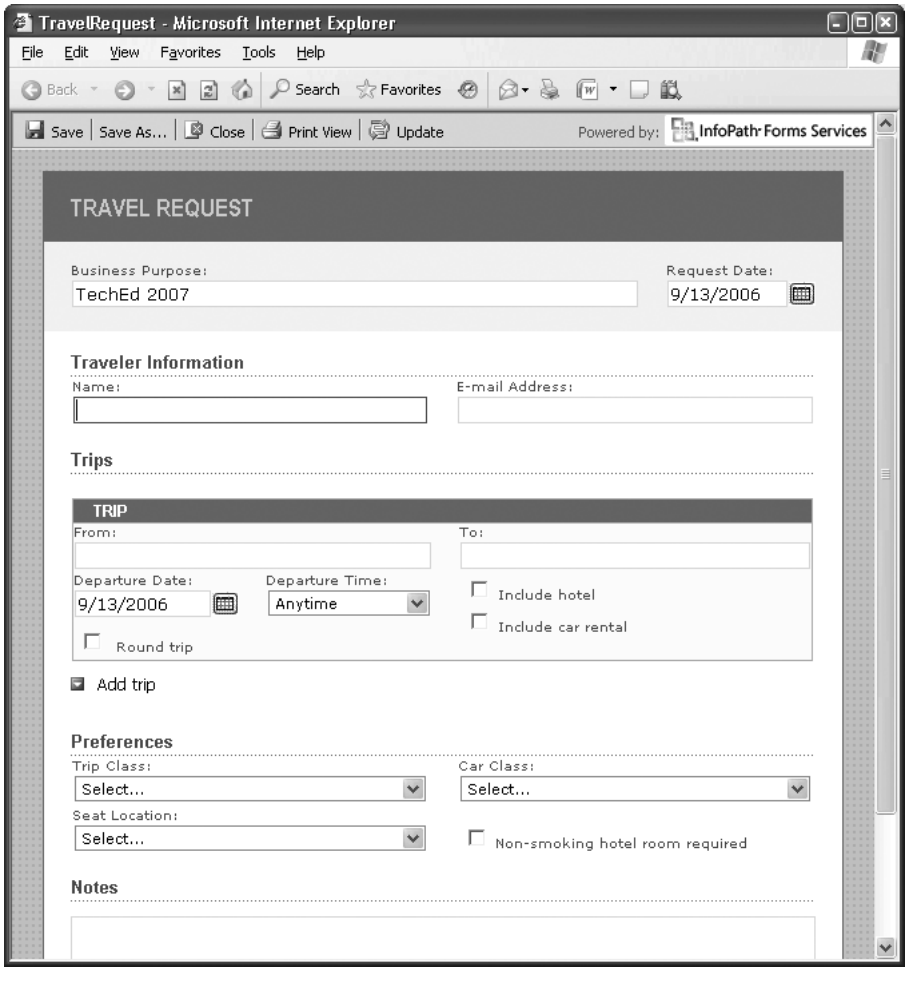


FIGURE 1.6: Filling out a browser-enabled form template in Internet Explorer

many similarities between the InfoPath Forms Services and the InfoPath client application that it's a more interesting conversation to discuss what the InfoPath Forms Services in the 2007 version *does not* support.

Due to the InfoPath form architecture and the tremendous flexibility in form configuration, some of the more advanced features of the InfoPath client are not supported. For example, forms with script are not supported. Managed form code works as expected, but showing UI, such as a message box, by calling methods in the InfoPath or Windows Forms object models, is not. Another major difference between InfoPath and Forms Services is

the behavior of data connections. The security model of data connections is more restrictive with browser forms. Many of the differences between the two types of forms have been addressed in the 2010 release, but the later chapters will cover in more detail which features are still unavailable in the browser.

InfoPath Forms Services is built on top of the rich capabilities of the SharePoint Server 2007 platform, which includes Windows SharePoint Services. You can harness the full power of SharePoint sites, document libraries, lists, and other SharePoint-specific features. The SharePoint Server package adds portal features such as Workflow Services and Enterprise Search. Partially filled forms in the browser can be saved to a form library and ultimately submitted to a SharePoint document library, for example. Administrators can configure InfoPath Forms Services settings from the command line, but the SharePoint Central Administration site offers a more user-friendly option for doing so.

A favorite feature for Web designers is the ability to host a browser-enabled form as a control in a Web page. Since Forms Services is implemented as an ASP.NET control, Web design gurus familiar with ASP.NET will have no problem incorporating forms into their existing Web infrastructure. Various events exposed from the form control serve as liaisons between the control and the hosting page. This helps tie your form intimately into the overall design and experience of your Web page.

The InfoPath Forms Services package is meant not as a replacement but rather as a complement to filling out forms with the InfoPath client. The relationship between Forms Services and the InfoPath client application can be compared to the relationship between Outlook Web Access (OWA) and the Outlook client application. OWA tries to do the job of the Outlook client on the Web but offers fewer features than the Outlook client application; for example, OWA lacks offline and client-only mail rules. As with OWA and Outlook, when the InfoPath client is available, you would prefer to use it over the server due to the fact that the InfoPath client provides a richer set of features. Even with the advancements in browser and Web technology, it is nearly impossible to do the same things in a Web browser as you would in the InfoPath client. However, most people don't really notice what is missing when they use a browser-enabled form. Limitations

aside, InfoPath Forms Services is a truly remarkable advancement for creating ubiquitous InfoPath forms.

In addition to meeting the number one customer request with Microsoft InfoPath Forms Services, the InfoPath team also responded to feedback about the InfoPath client application by adding many new features.

A *Getting Started* dialog was introduced to give easy access to recently used and created forms. The *Design a Form Template* dialog gives a choice of form types and data sources to simplify the design process. You can customize existing templates or templates downloaded from Office Online. There is an option to import forms from Word or Excel to make use of any existing company forms.

InfoPath 2007 added new controls—the Combo Box, the Horizontal Repeating Table, the Multiple-Selection List Box, and the Horizontal Region. However, the most exciting addition in terms of controls is the ability to create custom controls with no code. InfoPath **template parts** allow you to create custom controls that are aggregates of existing controls already built into the InfoPath design mode. In addition, you can add rules, calculations, data validation, and even secondary data sources to template parts you design, thus creating reusable components with no code required. Another appealing addition to InfoPath 2007 is integration with Outlook. **InfoPath e-mail forms,** as the feature is called, enables users to fill out forms in the body of an e-mail instead of having to open the InfoPath client application.

Some of the other additions to the InfoPath client application are not as obvious. Many performance improvements enhance the usability of InfoPath. Several long operations now offer progress dialogs that can be canceled. Also, cases where multiple dialogs used to be displayed now show only one.

A few more of the many feature additions include Information Rights Management (IRM), date calculations, support for read-only views, bound fields in headers and footers, and better support for offline forms.

Features were also added specifically for developers including integration of the InfoPath design mode into Visual Studio Tools for Office (VSTO). But what if you just want to enhance your forms a bit without installing Visual Studio or resorting to script? You can do that with Visual

Studio Tools for Applications (VSTA). With VSTA, you can use managed code anywhere you would have used script in InfoPath 2003 and SP1.

Another cool feature addition geared toward developers in InfoPath 2007 is support for add-ins. Developers can write their own add-in components using the Component Object Model (COM) or managed code. These add-ins enable you to control or modify InfoPath's behavior and add functionality that does not exist in the core InfoPath client application. Since add-ins are supported by other Microsoft Office applications, developers can write code once that will work across applications. Many developers have asked for the ability to host InfoPath since InfoPath 2003 was released. With InfoPath 2007, it became possible to add rich form editing to your own applications by reusing InfoPath. This is such a popular scenario that other Microsoft Office applications have taken advantage of this feature. Word, Excel, and PowerPoint now show their document properties in a **Document Information Panel** that is actually an InfoPath form. The Document Information Panel includes all the features you have come to expect from an InfoPath form. In the advanced section of the book, we will show you how you can add these features to your own application.

InfoPath 2010

The next wave of Office products, Office 2010, includes InfoPath 2010. Unlike the previous versions, the InfoPath client is split into two programs in this release—InfoPath Designer and InfoPath Filler. This creates a clearer distinction between the process of creating a form and the process of filling one out. There is also a change in focus from InfoPath being a standalone product to being the primary design tool for SharePoint forms. This builds on the browser-based forms capabilities that were introduced in the 2007 release with easier customization of SharePoint data collection elements for lists and workflows.

When opening either product, those familiar with earlier versions of InfoPath will notice instantly the change in user interface. The fluent user interface, sometimes referred to as the ribbon UI, which was introduced in many Office 2007 products, has now extended to InfoPath. The idea of the ribbon UI is to make it easier to find features by grouping them into

contextual tabs that appear when relevant, such as the *Table Layout* tab (see Figure 1.7), appearing when you start to edit a table.

When the ribbon UI was first introduced, a strong piece of customer feedback was regarding the lack of a *File* menu. So, in the 2010 release of Office, a new backstage tab was introduced, which allows access to those functions traditionally associated with the *File* menu, such as *Save* and *Open*, as well as functionality around publishing and compatibility of the form. When InfoPath Designer is first opened, it opens in this backstage view, automatically selecting the section for designing a new form. This replaces the *Design a Form Template* dialog from InfoPath 2007, but many of the template types, such as Blank Form and Web Service, remain the same. The main differences in form template types are around the tighter integration with Microsoft SharePoint Server 2010. There are the options to create forms based on a SharePoint List, a SharePoint Form Library, and a form that sets the document properties of files in a SharePoint document library.

The integration improvements between InfoPath and SharePoint include the ability to customize the SharePoint list entry form using InfoPath and more easily create forms to initialize workflows and execute workflow tasks. One of the improvements around browser-enabled forms is the introduction of an InfoPath Form Web Part to SharePoint, which makes it much easier to include an InfoPath form as part of a SharePoint page. Combining InfoPath with SharePoint will be discussed in more detail in later chapters of this book.

The 2010 release of InfoPath improves the browser-enabled form capabilities provided by InfoPath Forms Services. The gap of functionality has narrowed between what is possible in the InfoPath client and what is possible in the browser. The majority of features and controls now work exactly the same in the browser as the client. When designing a new form, the default setting is for a Web browser form, whereas in InfoPath 2007 the default was for a client form.

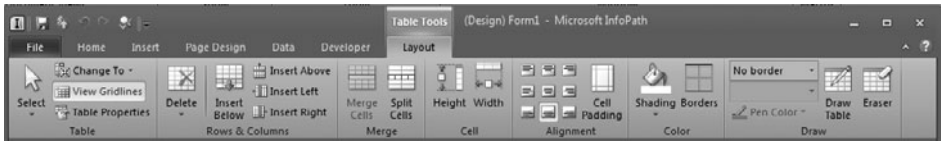


Figure 1.7: The Table Layout tab

InfoPath 2010 has introduced some new controls—Person/Group Picker, External Item Picker, Picture Button, Hyperlink, and Signature Line.

There are other significant improvements around the management of rules. There is now a *Rules* task pane (see Figure 1.8), which can be used to organize the rules associated with the controls on the form. This makes it easier to add and manage multiple rules—for example, conditional formatting or setting field values—to an individual control. There is also a new option, which is to copy a rule or rules from one control and paste them on another. If you have two text fields and want to create the same multiple conditional formatting rules on both, you can now create the rules once, use the *Copy All* button, and then simply select the other field and select to *Paste Rules*. In previous versions of InfoPath, you would have been forced to create the rules separately for each control.

The developer experience for InfoPath 2010 has had a few changes. As with InfoPath 2007, you can use Visual Studio Tools for Applications

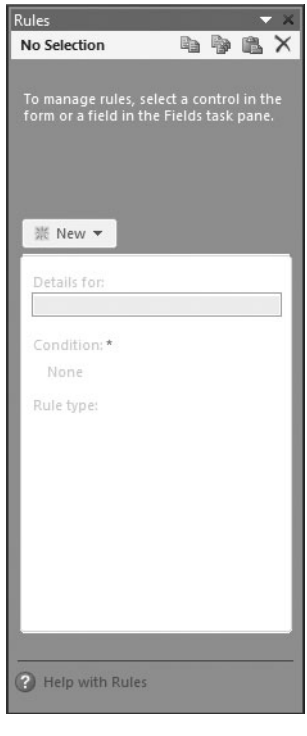


FIGURE 1.8: The Rules task pane prior to rules being created

to add small pieces of code to a form. The ribbon UI includes a *Developer* tab. This allows you to choose from a selection of event templates to easily associate your code to specific actions within the form. For example, selecting *Form Loading Event* allows you to write code when the form is loaded and *Changed Event* creates code associated to a selected field that will run when that field changes. This makes it easier to organize small pieces of code attached to a form.

There are other improvements that are less obvious around performance. Some processes that used to take many dialog boxes now involve less. There is one significant improvement around form publishing, which is the introduction of the *Quick Publish* button. As with earlier versions, when you have finished designing your form, you must publish it before it can be filled out. The various options for publishing will be discussed in Chapter 9. In InfoPath 2007, every time a change is made, the form must be republished by going through the whole publishing process. In InfoPath 2010, the *Quick Publish* button allows you to publish your updated form template while keeping the rest of the settings the same.

As you can probably tell just from this introduction, InfoPath contains a very rich set of features for creating forms for SharePoint Server 2010. In addition, InfoPath remains a powerful tool for creating standalone forms.

What's Next?

You may be thinking, "With the many features available in InfoPath, where do I start?" That's a good question. The answer obviously depends on your level of expertise. If you have no experience with InfoPath, start at the beginning. Part I is all about designing form templates from scratch and assumes you have no prior knowledge of InfoPath. Starting with Chapter 2, we will walk you through very basic form design. Each chapter in Part I may assume knowledge of and build on previous chapters. By the time you finish Part I, you can consider yourself an expert form designer and will be able to create complex form templates from scratch with no code, both for InfoPath Filler and for Microsoft SharePoint Server 2010.

If you are an application developer and already know a bit about InfoPath, feel free to skim over Part I and then jump right into Part II, where

we'll talk about advanced form design. We'll discuss adding code to your InfoPath form templates (both script and managed code), using advanced features of InfoPath Forms Services, creating add-ins, hosting InfoPath, creating custom components with ActiveX, and much more.

So, now that you know a little about InfoPath and where to go next, let's get started.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z