CHAPTER 1

Introducing InfoPath

Microsoft introduced InfoPath in 2003 as part of the Office System. While it may appear to be a simple form designer, its apparent simplicity masks an incredibly powerful tool for building the user interface for any number of applications. InfoPath can cover a multitude of situations, from basic workgroup forms that produce XML to enterprise applications that submit data, via web services, to an enterprise application integration (EAI) engine such as BizTalk Server.

This book will examine how InfoPath 2007 along with Microsoft Office SharePoint Server can address each of these scenarios. In addition, since InfoPath is a common thread among them, it eases migration from one scenario to another.

InfoPath

The design interface of InfoPath is very straightforward (see Figure 1-1). A user who wishes to design a form from scratch simply needs to start with a layout table, and then use table structures within the form to establish the look and feel of their form. From there, it's simply a matter of dragging and dropping the various control structures (repeating or optional sections, repeating tables, master/detail sections, etc.), and then the controls needed to establish the data characteristics of the form itself. These simple actions alone can produce a form that can create arbitrary XML or publish to a number of XML-based servers.

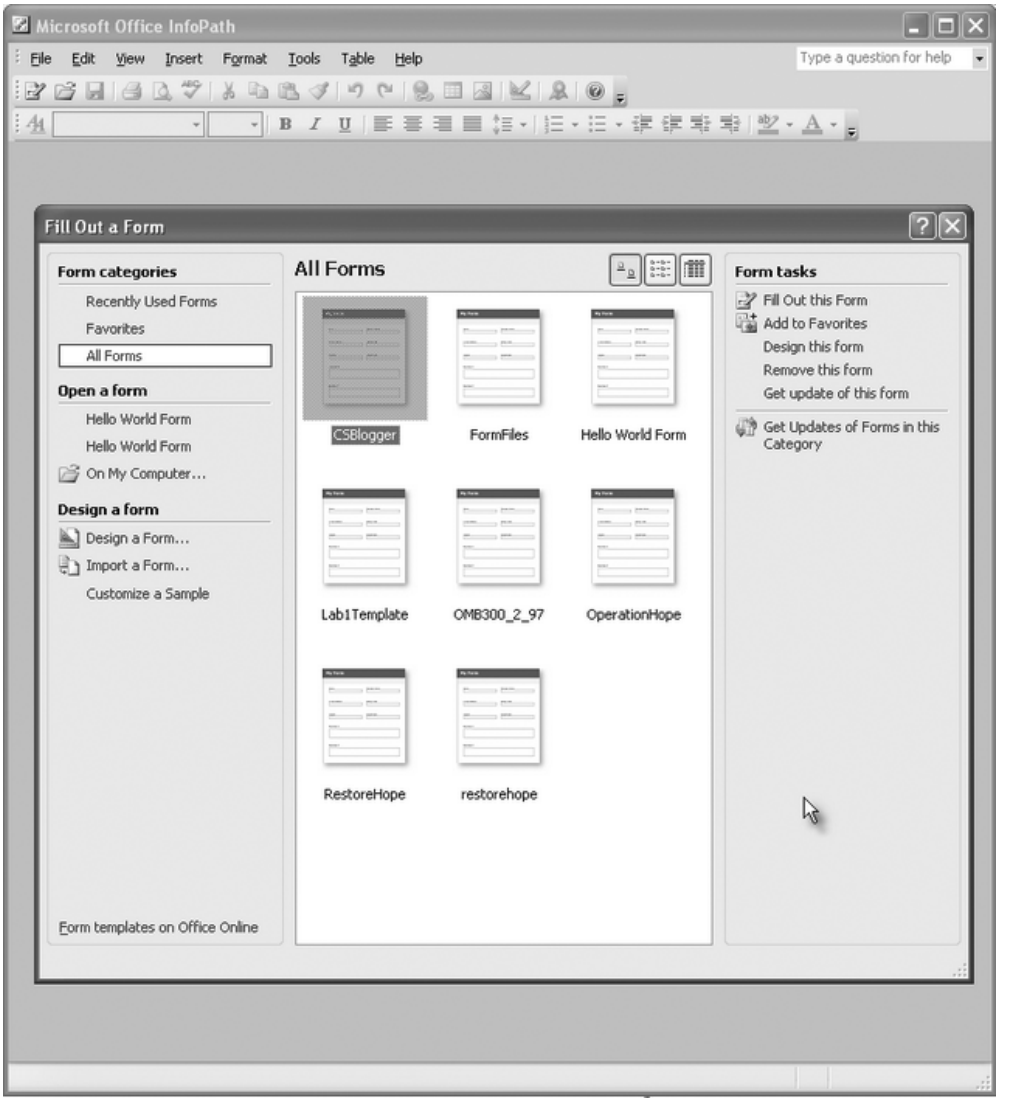


Figure 1-1. *The InfoPath design interface*

One of the major benefits of InfoPath is that it is wholly XML-centric. The form design is based on XML schemas (XSD). Form designers have a choice of either designing a form from scratch (which will result in a schema being built automatically by InfoPath), or building a form based on a preexisting schema. A blank InfoPath form with an attached schema is shown in Figure 1-2.

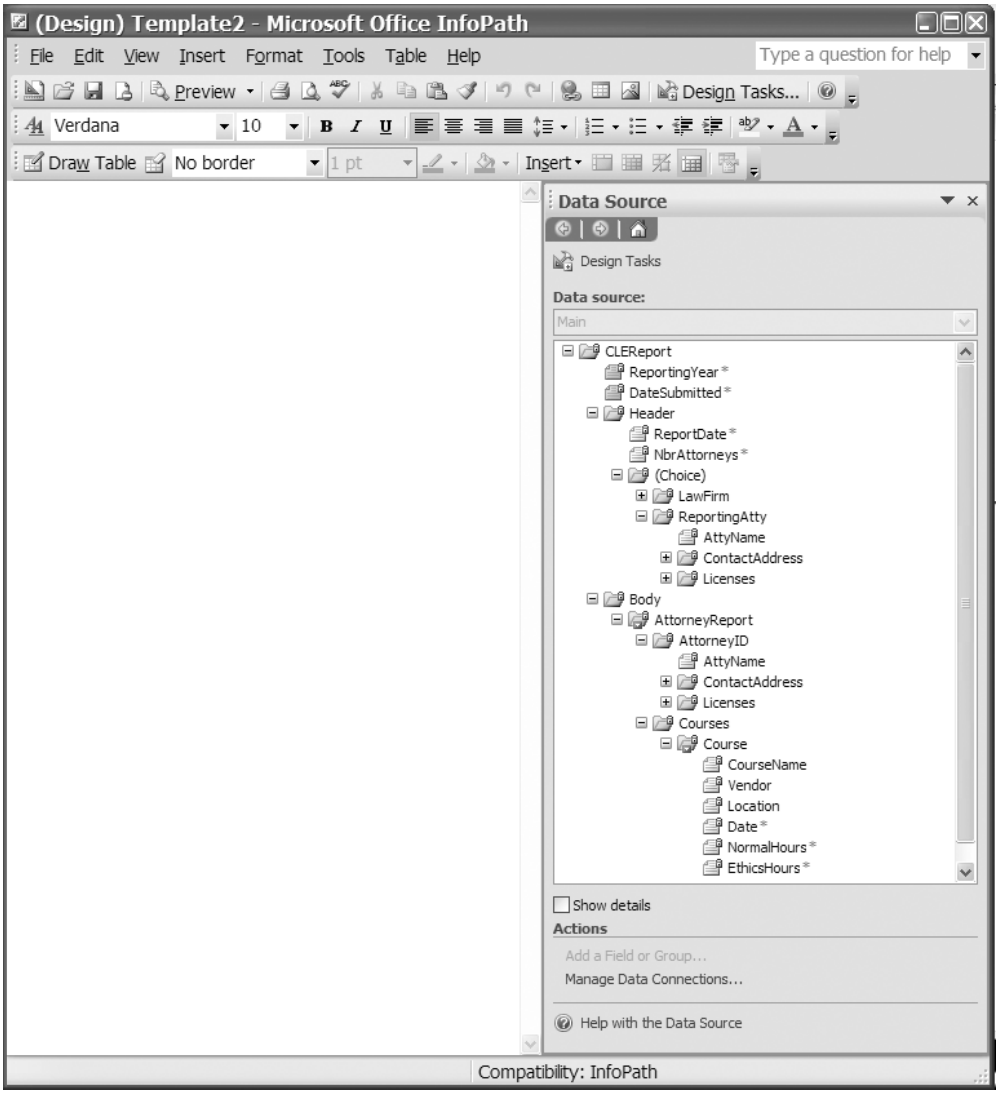


Figure 1-2. *XML schema in an InfoPath form*

InfoPath views are based on XSL transforms (XSLT). Form developers can build various views of their data, and those views are rendered by XSLT automatically generated by InfoPath. Some examples of the uses for views include the following:

- A personnel review in which the manager's comments aren't visible to the employee being reviewed.
- A routed approval form in which previous approvals are read-only for subsequent approvers.
- A multipurpose form (like those at DMV offices) where certain fields are shown to the user based on the form's purpose.
- A user-friendly data entry interface as well as a formal printed view.

Figure 1-3 shows two views of an asset tracking form: one summary view and one that shows the details of the asset.

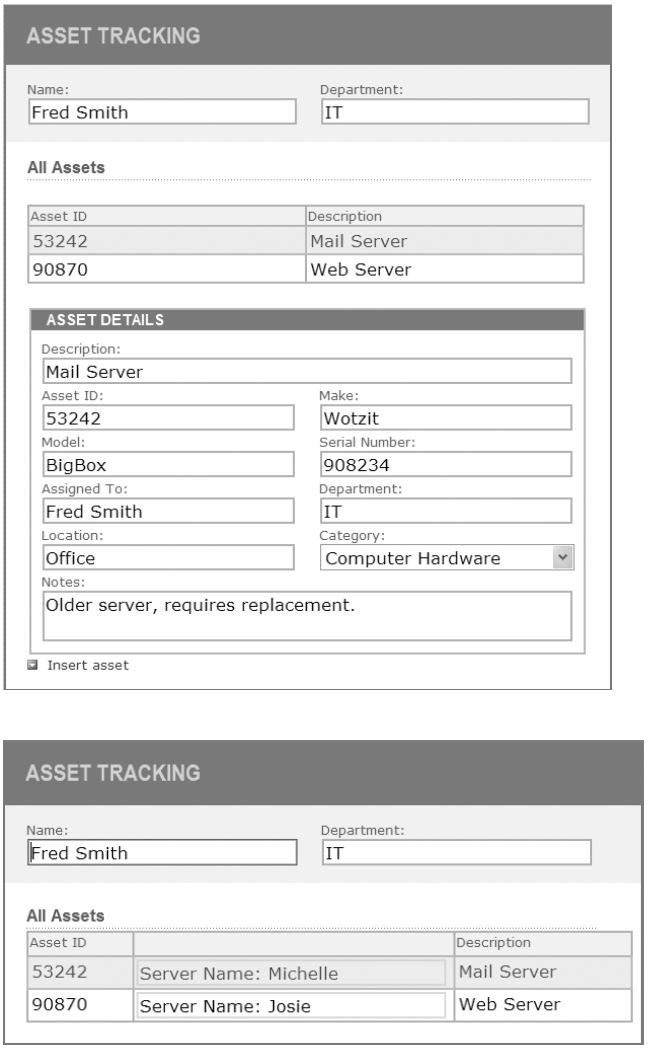


Figure 1-3. *Two views within an InfoPath form*

Finally, all InfoPath data is saved as XML. This enables InfoPath to coexist with other industry standard tools and processes in an XML-oriented solution. The ubiquity and power of XML tools also means that InfoPath can serve as a form UI (user interface) in an environment that isn't XML-based.

For example, consider a document-centric environment for processing invoices. Invoices are all processed as XML documents (as shown in Figure 1-4), and there is an established schema for the invoices within the organization. However, working with the invoices is painful—there are some XML-editing tools, but most people simply work in a text editor to deal with the invoices.

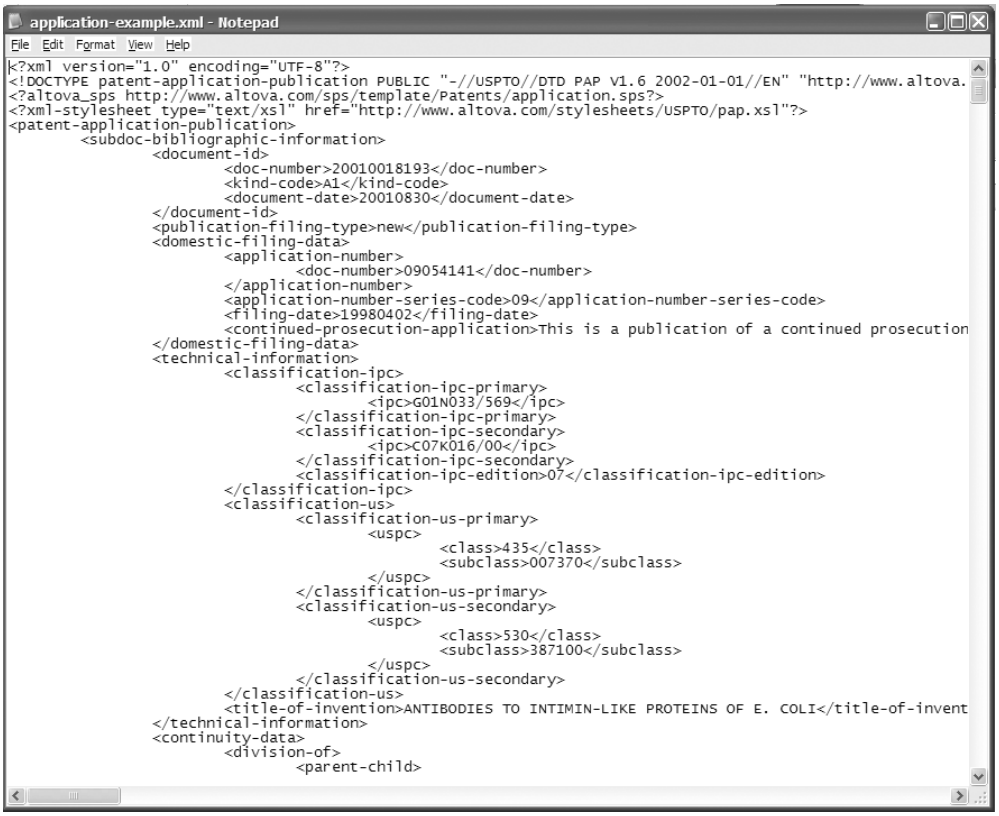


Figure 1-4. *XML data for an invoice*

With the XML schema at your disposal, it would be short work to point InfoPath at the schema and create a user-friendly form, with validation, that hides the complexity of the XML documents and allows users to interact with something that looks properly like an invoice, such as the one shown in Figure 1-5.

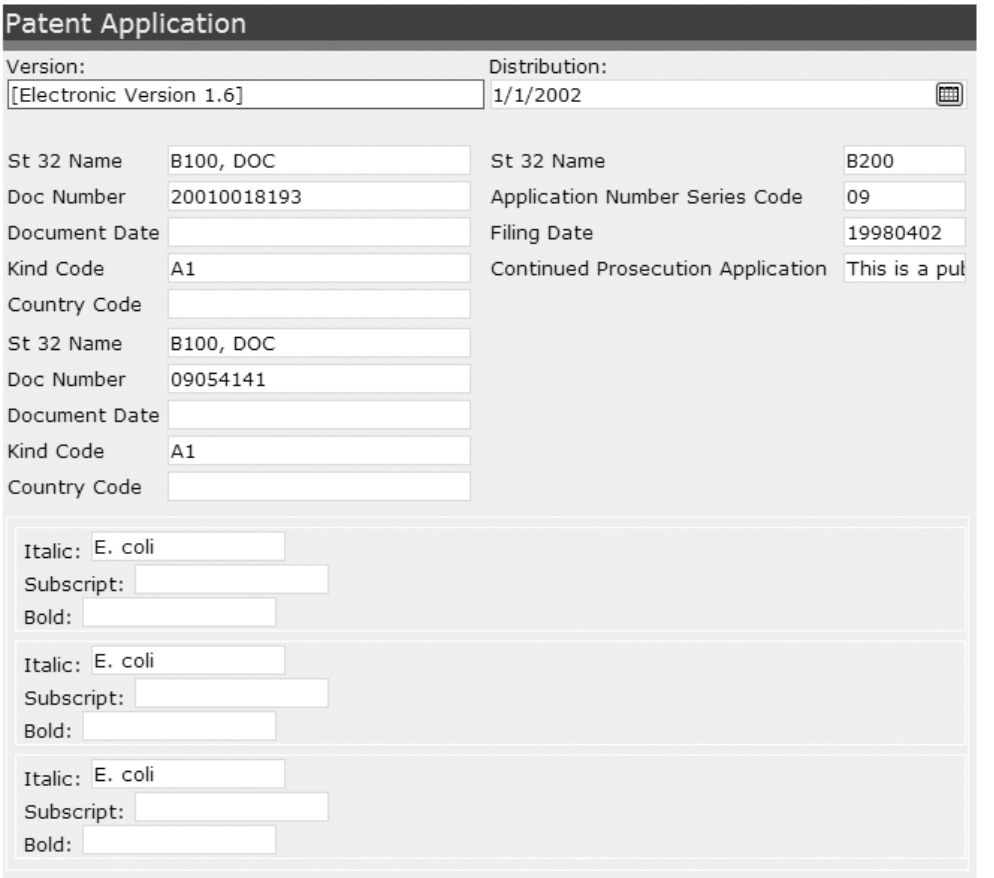


Figure 1-5. *Invoice form in InfoPath*

InfoPath As a Smart Client

An additional benefit with InfoPath is that since it has a rich client for filling out forms, a user that has InfoPath installed doesn't need to be online to fill out a form. For example, let's say a government official needs to fill out a project justification and financials package, which is about 12 pages of detailed data. He may fill it out in one sitting, but it's more likely that he will have to stop in the middle and put it aside for various reasons: interference of other work, the need to research some aspect of the project to properly fill in the form, or simply the lack of enough time in one day.

With a web-based or desktop custom form, the ability to save a user's progress and return to it requires additional coding. If the official wants to travel with the form (on a laptop, for example), that may require even more code (and with a web form, it's simply not possible).

InfoPath, on the other hand, has built-in capability for a user to save form data locally to be opened later. In addition, even if the form template were hosted on a server when the user opened the form, InfoPath caches the form template locally so that the user can continue working on the form even if the original template location (web server or file share) is no longer available.

InfoPath also has a rich collection of controls: a date picker, repeating sections, check boxes, radio buttons, drop-down lists—all the client controls forms designers have come to expect. In addition, InfoPath provides spell checking out of the box.

Finally, for organizations that have a public key infrastructure (PKI) in place, InfoPath provides the ability to digitally sign forms. Enabling this capability simply requires selecting a single option in the form options when designing a form. Once the digital signature option is enabled, users can digitally sign their forms, providing authentication and nonrepudiation of the data in the form. Additional options provide the capability of signing independent sections of the data, as well as co-signing (parallel signatures) and counter-signing (signing a previous signature).

A major limitation with InfoPath 2003 was that users were required to have InfoPath installed on their desktops to fill in InfoPath forms. This limited usage to internal/intranet scenarios—forms for users outside the organization (e.g., the general public, customers, or users from other companies) still had to be implemented by other means. A further limitation was that in the area in which it was best suited—intranet applications—a forms solution generally required a workflow solution. Unfortunately, there was no readily available workflow solution that could interact with InfoPath with the same ease at which users could create forms.

InfoPath 2007 addresses these shortcomings very nicely. In the first case, the new Microsoft Office InfoPath Forms Services (part of Microsoft Office SharePoint Server) provides a capability to publish InfoPath 2007 forms to browser-based forms. As a result, a developer (or power user) can design a form in InfoPath, leverage the InfoPath rich UI capabilities inside the firewall, and publish the same form to an Office SharePoint server for consumption by users outside the firewall.

With respect to workflow, Microsoft Office SharePoint Server is built on Windows SharePoint Services, and provides all the collaboration capabilities inherent in that platform. In addition, the newest version of Windows SharePoint Services provides powerful new workflow capabilities based on the Windows Workflow Foundation (WF). Windows SharePoint Services version 3 provides out-of-the-box workflow capabilities that can be configured by power users and administrators. In addition, the WF development model means that .NET developers can expand and extend the workflow capabilities to provide for any possible circumstance.

■ **Note** Lest you think that there is a glaring typo in the preceding paragraph, the proper abbreviation for Windows Workflow Foundation is indeed "WF." It seems that after the Worldwide Wrestling Federation lost a legal battle with the World Wildlife Fund over the initials WWF, Microsoft decided discretion was the better part of valor.

These are just two of the exciting new capabilities in InfoPath 2007. Other new benefits include the following:

- Control templates, with which power users or developers can create templates of controls in frequently used layouts or combinations
- Integration with Outlook for offline folder capabilities
- The design checker, which is used to validate your form designs
- The multi-select list box, which is now a standard control in the toolbox
- Ability to publish a form directly to an installable MSI file
- Data connection libraries in Microsoft Office 2007 Server
- Built-in ability to publish to PDF or Microsoft's new XPS format
- The Trust Center, for managing trusted forms
- Additional form events
- Offline caching of lookup data
- InfoPath forms hosted in the designer when working in Visual Studio

We'll look deeper into these capabilities in the next chapter.

E-forms

Every company or agency has a problem with forms. They have paper forms in filing cabinets, submitted by customers or constituents, in folders routing around the office, and so on. The problems these forms cause are well known—routing takes time and frequently requires rekeying of data (potentially introducing mistakes). Data is difficult to find once the process is complete, and metrics are nearly impossible to generate.

Hosting the forms electronically has long been recognized as a solution to these problems, but that simple concept opens a whole new Pandora's box: how to design the forms; how to design workflow; how to ensure that forms are available internally, externally, online, offline, and in print; how to process forms or publish data into other business systems; and so on.

Part of the problem is the desire to find a "one size fits all" solution—something that can solve the problem for

- Enterprise intranet applications (time cards, purchase requests, etc.)
- Workgroup applications (configuration management, status reports, etc.)
- Ad hoc applications (shift scheduling, equipment tracking, etc.)
- Internet applications (customer requests, constituent form submission, etc.)

Traditionally, these have been solved by myriad solutions—heavyweight code development, web development, Access databases, Excel spreadsheets, and so on. Each serves its purpose, but each has varying degrees of supportability. In addition, the use of different platforms for each style of application makes migrating an application from one "slot" to another difficult (e.g., taking a grassroots workgroup application and creating an enterprise application from it).

InfoPath for Forms Solutions

Workgroups everywhere have a horde of data-handling requirements that aren't being met. They have a need to collect structured and semi-structured data, aggregate it, and search and report on the results. These requirements often find a home in Excel, Access databases, or other small custom applications. Consider a hypothetical office that is required to track certain types of training for regulatory requirements (continuing legal education for lawyers, accountants, auditors, etc.). Those records probably started on ledger paper (see Figure 1-6).

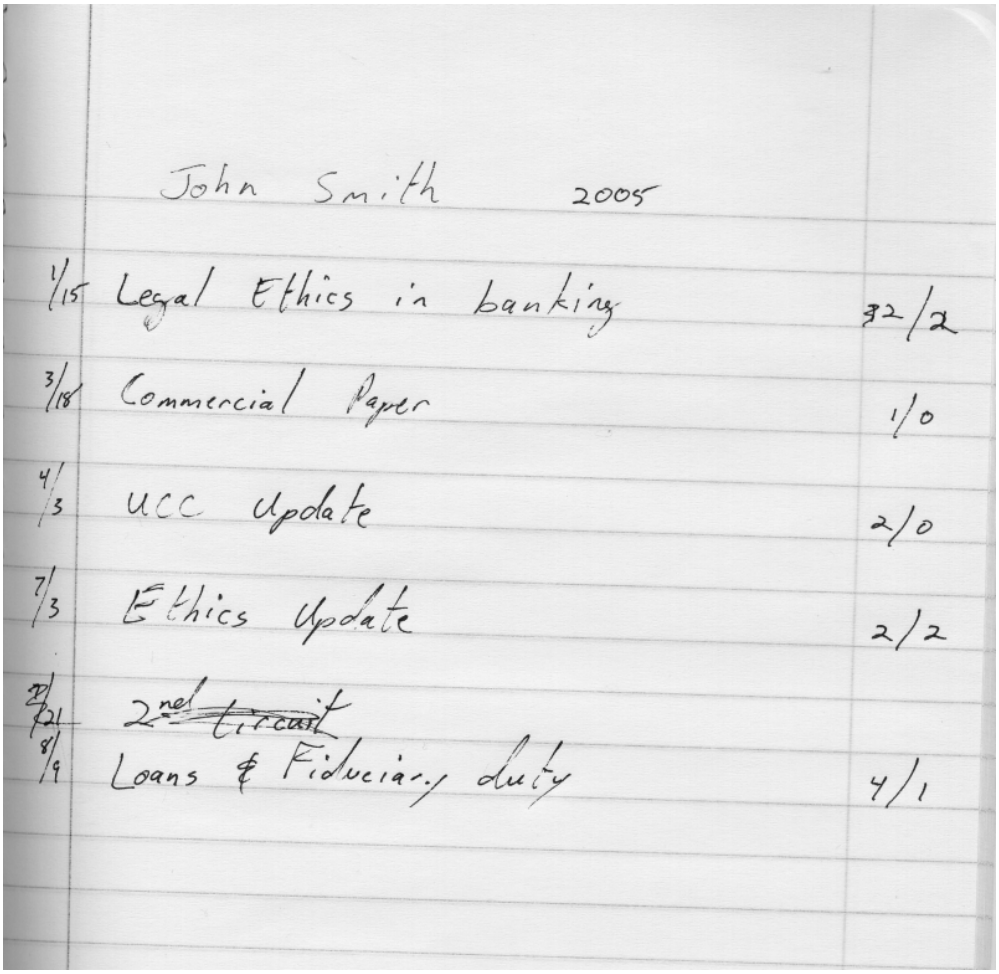
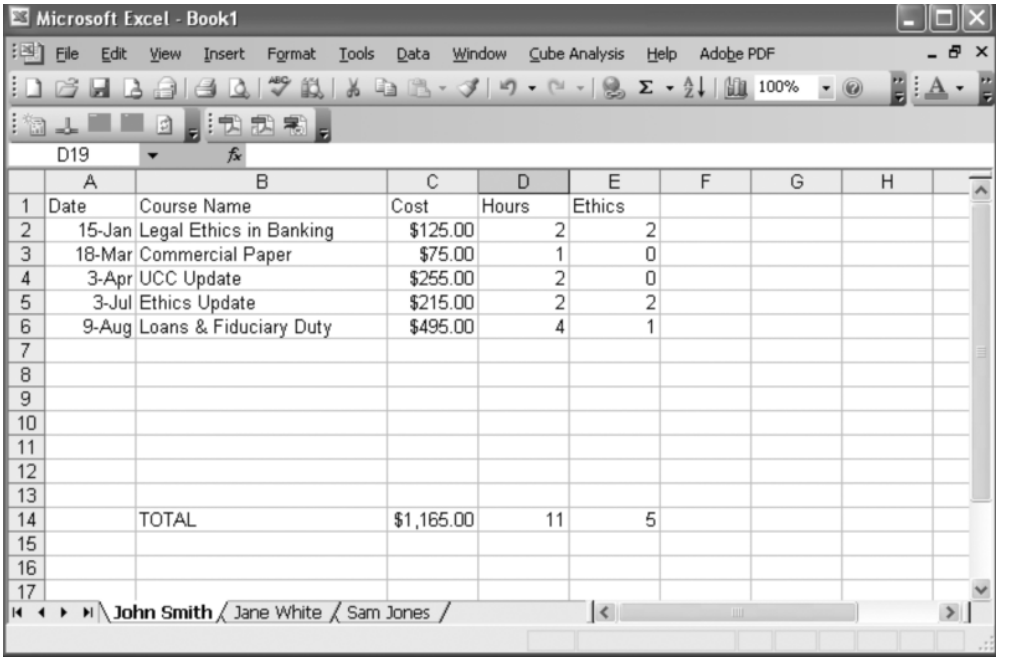


Figure 1-6. *Twentieth-century record keeping*

The problems here are obvious—reporting has to be completed by hand, correcting mistakes requires good old basic "line out and rewrite," and the potential for both data entry and compilation errors is immense. Data entry is tedious, and if someone wants to verify their status, they have to either look at the paper records themselves (lack of security) or get the auditor responsible for tracking coursework to look it up for them (wasting personnel resources).

Basically, this process gets reduced to "publish one warning report four weeks before the end of the year, spend a week reconciling updates, and then publish again at year's end." There *might* be quarterly updates if management (and the bookkeeper) has the time to collate and publish them.

The quickest and most obvious fix would be to address the constant retotaling by hand, and that could be done in Excel (Figure 1-7).



	A	B	C	D	E
1	Date	Course Name	Cost	Hours	Ethics
2	15-Jan	Legal Ethics in Banking	$125.00	2	2
3	18-Mar	Commercial Paper	$75.00	1	0
4	3-Apr	UCC Update	$255.00	2	0
5	3-Jul	Ethics Update	$215.00	2	2
6	9-Aug	Loans & Fiduciary Duty	$495.00	4	1
14		TOTAL	$1,165.00	11	5

Figure 1-7. *Using Excel for record keeping*

Of course, the problems here are similar—you have very little control in terms of security, so most likely someone would have to be disturbed for an individual to update their file or get their status. Compiling reports is also still problematic—someone has to go through and compile the data by hand. The math is already done, but generating the report isn't.

At some point, an enterprising individual in the office would have started adding macros to the spreadsheet to implement some business rules, and the truly industrious would have started building a training tracking application. Over time, this application would evolve, and the bible of business rules and validations would grow along the lines of the following:

- Any course more than $500 must be approved by a senior partner.
- Ethics hours cannot be more than total course hours.
- Course dates cannot be after today's date (coursework can't be pre-entered).
- Courses cannot be more than $75 per hour of training.
- Due to the load in tax filing work, no training may be entered for April.

Then there are reporting requirements—year end reports, delinquency reports, quarterly updates, accounting reports, spending per attorney, and so on.

What happens after this generally varies, but the worst case is that the application grows in popularity and is adopted by another department. Once the application becomes a serious dependency for the organization, it will need to be migrated to a more robust architecture. This may (probably will) require a relational database back end, forms with the same validations, a reporting solution, and other improvements.

One major task in such a migration is ensuring that the new UI (the forms) meets all the user requirements of the old solution while keeping the same program logic and validations. The existing forms must be picked apart, and every rule or criterion documented to be reimplemented in the new solution.

InfoPath can ease these types of migrations because while it can act as a front-end tool for power users, it can also act as the front end of a server-based enterprise solution. Since InfoPath is XML-based and can hook natively to web services, InfoPath forms cover the full spectrum, from ad hoc workgroup "micro-applications" to enterprise solutions.

Summary

This book will cover InfoPath 2007 from the ground up, using a solution-based approach. It will cover the basics of the forms package, and how you can use InfoPath to build a pretty powerful form just by dragging controls to a design surface, publishing to Windows SharePoint Services, and using the native forms libraries in Windows SharePoint Services for aggregating data and providing a basic reporting capability.

You'll then move on to more advanced concepts: using data connections, workflow, publishing to browser-based forms, and custom code for those things that simply can't be done through the InfoPath UI.

By the end, you'll see how InfoPath can ease a lot of application development pain.