Chapter 1

Office and XML Technology

IN THIS CHAPTER

◆ Exploring what’s new in Office
◆ Previewing XML’s role in Word
◆ Previewing XML’s role in Excel
◆ Previewing XML’s role in Access
◆ Previewing XML’s role in InfoPath

The latest version of Microsoft Office, called Office 2003, brings many changes and improvements to the desktop. The most important of these changes have to do with the way Office can interact and exchange data with other programs. These new capabilities are implemented by means of a technology called Extensible Markup Language, or XML. This chapter explains why interoperability is so important for today’s computing needs, and provides an overview of the related features in Office. Chapter 2 provides you with a basic look at XML and how it works.

Why XML?

Office applications have always had the ability to exchange data with other Office applications. These capabilities were very useful, and at the time quite impressive. Aside from the obvious and trivial use of the Windows clipboard for “cut-and-paste” operations, you could always do things such as inserting a slide from a PowerPoint presentation into a Word document or embedding a Word document in an Excel worksheet. There was even some data exchange possible with programs outside of the Office suite, although these capabilities were rather limited.

As computing has evolved from a single program operating in isolation on a single computer, to various software components running on a corporate LAN, to applications that use components in different cities or different countries via the Internet, the need for smooth interoperability has increased. Components need to communicate with each other. This was much easier, of course, when the entire program ran on one computer, or even ran on different components on a single
network under the control of one Information Systems (IS) department that could enforce the required compatibility. But now, a worker using an application in the San Francisco office might be interacting with components located on systems in New York and Paris, where different applications and even different operating systems might be in use. At the same time that it became more important to maintain compatibility, it became far more difficult to do so.

Simultaneously, the very concept of an “application” was becoming less meaningful. Developers and systems integrators tend to think more in terms of business processes—capabilities or actions that a business or other organization needs. For example, think of a hospital, the information it needs to keep track of, and the various uses that information is put to. On the “input” side of things, the following is needed (this is surely a simplification, but still serves well as an example):

- Personal information about a patient
- Insurance and/or Medicare information
- Details of procedures that were performed: X-rays, lab tests, surgery, physical therapy, and so forth
- Accounting of supplies used: prescription drugs, dressings, intravenous solutions, and so on
- Records of visits from consulting physicians and other specialists

Then think of the multiple uses to which this information may be put:

- The Billing department uses the information to submit insurance claims and prepare patient bills.
- The Ordering department uses the information to keep track of inventory of supplies and to place orders as needed.
- The Records department keeps track of all information as part of each patient’s medical record.
- The physicians and nurses need access to the information to keep track of each patient’s progress.

When designing a computerized solution to fill needs such as this, the focus is on the tasks that need to be done rather than on individual application programs. The fact is, however, that in order to be potentially useful in a business solution, an individual program should have as much flexibility as possible when it comes to exchanging data with other parts of the solution.

The answer to this problem clearly lay in the widespread adoption of a common standard for data transfer. Any proprietary technology, under the control of a
single organization was unacceptable. As a public and freely available technology, XML was, as they say, “just the ticket.”

**XML in Office 2003**

Previous versions of Office, such as Office 2000 and Office XP, integrated XML to some degree into the various applications. For example, Excel XP could open and save XML files, and Access XP could import and export XML data. But those features are kitten’s play in comparison to the extent to which XML is integrated with Office 2003.

The deeper integration of XML technology into Office 2003 brings a host of important enhancements to the suite. These enhancements are not the type that are obvious to the user right away. XML does not provide a snazzy new user interface, new formatting commands in Word, better charts in Excel, or automated data entry in Access. For the most part the XML-related improvements in Office 2003 have to do with how the Office applications can exchange data with other programs. This includes data exchange between Office programs, but much of the emphasis is on exchange with non-Office programs. What other programs? It doesn’t matter—that’s the beauty of XML. By supporting the XML standard, Office can interact with any other program that also supports XML.

XML support is not spread throughout all of the Office applications. When speaking about XML and Office, the only traditional Office applications that are included are Word, Excel, and Access, plus the new application InfoPath. FrontPage, the Web site development application, has some new XML features, as well.

XML support permits Office applications to communicate with any other software that also supports XML, regardless of the system it is running on. Some of the consequences of this are:

- Office apps can exchange data with complex back-end data stores.
- Data can be retrieved from and sent to disparate and otherwise incompatible systems.
- Information can be reused and repurposed without the need to re-key or recode.
Information of various kinds can be structured in a way that makes it easier to search and organize.

Because the structure of XML data is independent of its display, the same information can be presented in different formats and on different devices as needs dictate.

A central aspect of XML in Office is support for schemas, which are also called data models. A schema is like a database template in that it describes the types and relationships of data. You can work with your own business-specific XML schema, using Office applications to access and reuse important information that may have been hidden away in documents sitting on file servers or on hard-to-access backend systems.

Some schemas will be specially designed for use within an organization. In other cases, it makes more sense to use one of the many published schemas that are designed for various tasks. One example is the Extensible Business Reporting Language (XBRL), an open specification that uses an XML schema to describe financial information. Another example is H7, which was designed for the healthcare industry. By utilizing such standard schemas, different organizations can easily share information even if they are using technologies from different vendors on different platforms.

Office provides several of its own schemas. The XML Spreadsheet Schema is designed for saving spreadsheet data in XML format. Word has its own XML schema, called WordML, that lets you save a document along with its formatting and other information as an XML document. The choice of your own custom schema, an industry standard schema, or Office’s schemas provides great flexibility.

**XML and Word**

Word 2003 has its own XML schema called WordML. When you save a document as an XML file using this schema, all of the formatting and layout information is preserved along with the document text. WordML does not provide semantic markup, so it gives no information about the meaning of the document contents. Such meaning can be provided by another schema. This gives you a great deal of flexibility because the WordML schema preserves layout and formatting information, while a custom schema can simultaneously provide semantic structure to the document.

The support for XML in Word 2003 creates a new way of looking at documents. In previous versions of Word, a Word document was really nothing more than a combination of raw text data with formatting. Searching the document or attempting to retrieve information from it was limited to a regular text search. There were at best very limited ways for the document to denote what its contents meant. With XML, a Word document can take on a dual identity, as both document (text with formatting and layout) and a data store (structured information). For example, Figure 1-1 shows an XML file open in Word with the XML tags visible. You could hide the tags and apply formatting to the data, but the tags would still be present and providing structure to the data.
Figure 1-1: Word can display XML data and retain the structure provided by the tags.

Here's an illustration: Suppose that your company requires prospective employees to submit a resume as a Word document. This is fine for printing and viewing on-screen, but suppose you are asked to see if any of the several hundred applicants have a degree in economics and speaks French? In the past, the only way to do this would be for someone to examine each resume looking for the relevant information. With XML, however, the resume documents could be structured in such a way that locating the relevant information would be a simple automated process.

Word also supports XSLT (XML Stylesheet Language for Transformations), a language for defining transformations to XML data. When a Word document uses a custom schema, you can create an XSLT transform, which takes the original document as input and creates a new document based on applying the transform rules to the original document contents. There are few limitations to what you can accomplish using XSLT. Here are some examples of what you could do:

- Extract parts of the document and output them as an HTML (Hypertext Markup Language) document for publishing on the Web
- Perform calculations and create summaries based on data contained in tables within the document
- Embed commands for outputting the document to a typesetter, text-to-speech converter, or other specialized presentation device
- Create a table of contents or an index
You can learn more about Word and XML in Chapter 7, “Word and XML,” and Chapter 11, “Connecting Word and InfoPath.”

**XML and Excel**

Excel has its own XML schema, XML Spreadsheet Schema (XMLSS), and can read and save data using this schema. In addition, Excel can read XML data based on any other schema without any need for reformatting. This means that the powerful presentation and analysis features of Excel can be brought to bear on essentially any data as long as the original source of that data has the ability to save in XML format. Manipulation of external XML data is simplified by Excel’s *Field Chooser*, which lets the user select data elements from an external schema and simply drag them to the worksheet for inclusion. The link between an Excel worksheet and external XML data is dynamic. Tables and charts in Excel will be updated in real time when the underlying XML data changes.

The Field Chooser acts like a visual mapping tool. When you open an XML file, it presents a visual representation of the data elements. This can be based on the file’s schema or, if there is no schema, Excel can generate one based on the file’s internal structure. Figure 1-2 shows an example; the hierarchical tree under “sampleData” shows the structure of the XML data. Any of these elements can be dragged to the desired location in the worksheet.

![Figure 1-2: The Field Chooser lets you map elements of an XML file to your worksheet.](image-url)
The Field Chooser greatly simplifies many tasks that in the past have required programming. For example: Map XML data to existing worksheet structure for data import.

Design dynamic workbooks load XML data, display it, and write it out in any format.

Create information repositories are based on existing Excel workbooks.

You can learn more about Excel and XML in Chapter 8, “Excel and XML,” and Chapter 12, “Connecting Excel and InfoPath.”

**XML and Access**

Access is a database management program designed for organizing, structuring, and manipulating data. As such it has a natural relationship with XML. In fact, in earlier versions of Office it was Access that first received the capability to work with XML data.

Access can work with XML data, importing data into any one of the various types of databases that Access supports. When you import XML data, you can select which parts of the XML file to import, as is shown in Figure 1-3.

![Figure 1-3: Access can import data from XML data files.](image)

Access can also export data from an existing database into an XML document. You have the option of applying an XSLT transform during the import process to convert the XML data into a format that the database can accept.

Access can also work with XML schemas. During the importing of data, a schema can be used to ensure that the data being imported adheres to a certain structure. You can also choose to export the structure of an Access database as an
XSD (XML Schema Definition) schema. The same is true when exporting Access data to XML. XSLT transforms can be applied during the data exporting process.

ReportML is a custom XML schema that is supported by Access. It permits exporting to go beyond just the data so that you can export the details of an Access datasheet, report, form, query, or table. The resulting XML file contains the associated presentation and connection information.

You can learn more about Access and XML in Chapter 9, “Access and XML,” and Chapter 13, “Connecting Access and InfoPath.”

XML and InfoPath

InfoPath is a new application in the Office suite. On the surface, InfoPath is a forms designer that lets you create forms for data entry and editing. Beneath the surface, InfoPath provides much more. Its forms are dynamic and can be associated with a schema to ensure that the form and the data that is entered meet the schema’s data model. InfoPath forms are based on XML technology and can be integrated with back-end databases and other applications that also support XML. For example, a form can be designed so its data is saved as an XML file, submitted to a Web service, or submitted to a database. The ability to integrate script into forms provides additional power and flexibility. Figure 1-4 shows an example of an InfoPath form.

Figure 1-4: An InfoPath form.
InfoPath provides for both the design of forms and the use of forms. Forms can be used offline as needed.

You start exploring this exciting new application in Chapter 3, “Introduction to InfoPath,” and learn to design InfoPath forms in Chapter 4, “Designing InfoPath Forms, Part 1” and Chapter 5, “Designing InfoPath Forms, Part 2.” Chapter 6, “Scripting with InfoPath,” shows you how to add scripts to your forms for additional functionality. Then, you see how InfoPath works with other Office applications and with Web publishing in Part IV.