XML in Office

Introductory Discussion

- Meet the family! Word, Excel, InfoPath, Access, and FrontPage
- Information capture and reuse
- End-user data connection
- Data-driven application enhancement

Chapter 3

This chapter is an overview of the XML features of Office. We discuss the XML-enabled Office products – Word, Excel, Access, FrontPage and the newly introduced InfoPath – in the context of several information sharing scenarios.

But the products are really just the supporting cast. The true stars are the advances that XML in Office brings to:

- information capture and reuse;
- end-user data connection; and
- data-driven application enhancement.

3.1 | Information capture and reuse

For all the valuable abstract data that is managed in database systems, there is even more that is hidden in rendered word processing documents. That fact represents an enormous intellectual property loss for enterprises, of course, but it also represents a nuisance and a time-waster for the information workers who work with those documents.

Consider the articles written for a company's websites and newsletters. Every one is likely to contain a title, author, and date within it, but more often than not that information has to be retyped, or individually copied and pasted, to get it into a catalog entry. That's because there is no reliable way for a computer to recognize those data items in order to extract them.

3.1.1 *Word processing*

In contrast, look at Figure 3-1, which shows an article being edited in Microsoft Word.

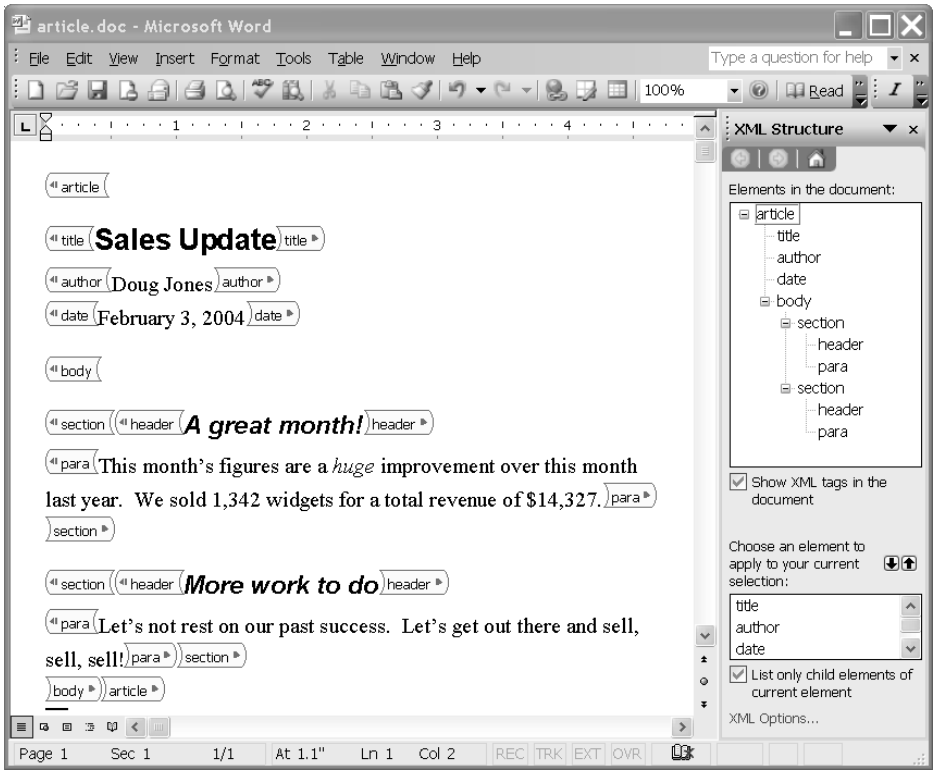

Figure 3-1 Word document showing optional tag icons and task pane with XML structure

The article is actually an XML document that conforms to a schema of the user's choosing, in this case `article`. The user has opted to display icons that represent the start- and end-tags. Note that there are distinct elements for the `title`, `author`, and `date`.

Solution developers can use the XML elements to check and normalize information as it is entered, whether or not the tag icons are displayed. An application, for example, could notify the user if the text entered for a `date` element isn't really a valid date. Or it could automatically supply the current year if none was entered.

The right-hand pane is called the *task pane*; it can be used for various purposes. In the figure, the top of the task pane shows the XML structure of the document. At the bottom is a list of the types of element that are valid at the current point in the document, according to the `article` schema.

The document is also a normal Word document, so Word's formatting features can be used in the usual way.

There are three ways to save this document as XML:

WordML

WordML is Word's native XML file format. It preserves the Word document just as the DOC format would, including formatting and hyperlinks. However, it doesn't include any of the `article` markup, so we won't discuss this option further here. (We cover it in Chapter 5, "Rendering and presenting XML documents", on page 86.)

custom XML

The document can be saved as an XML document conforming to a custom schema; in this case, `article`. A custom schema would normally be defined by an enterprise, or by a committee set up by an industry to which the enterprise belongs. For that reason, it would be designed to preserve the abstract data needed for the user's applications. For example, the `title`, `author`, and `date` can easily be identified by software and extracted for use in a catalog of articles.

mixed XML

The saved document could contain both WordML and the `article` markup, since the two are in different namespaces. This option preserves the formatting applied by the user, while still

preserving the abstract data and distinguishing it from the rendition information.

In our example, the article is the entire Word document, but that isn't a requirement. It is possible to intersperse short XML documents within a larger Word document. For example, a travel guide might include multiple XML structures that describe hotels, with subelements for the name, address, number of rooms, rates, etc.

Using XML with Word documents enables companies to capture more of the intellectual property that is created informally by individuals and work groups, and that typically remains inaccessible to enterprise information systems. As XML, that property becomes a portable asset that can be reused as needed.[1]

3.1.2 *Forms*

For many purposes, a data entry form is more suitable for information capture than a typically larger and less constrained word processing document. InfoPath lets you design and use forms that are really XML documents that conform to your own custom schemas.[2]

Figure 3-2 shows the layout of an order form in InfoPath's design mode. The structure of the `order` schema is shown in the task pane on the right, from which element types can be dragged onto the form.

Note that there is only one `item` line in the form design. Because the `order` schema allows `item` elements to be repeated, a user entering data will be able to add `item` lines as needed. Had `customer` elements been repeatable, the form would expand to allow insertion of the group of customer information fields.

Unlike Word, InfoPath generates an XSLT stylesheet to control the rendering of the form. The formatting can even be based on the data entered in the form. For example, the dialog box in Figure 3-3 specifies that negative prices should be shown in a different color.

1. We cover the details in Chapter 4, "Creating and editing XML documents", on page 60.
2. InfoPath is available in the Office Professional Enterprise Edition and individually.

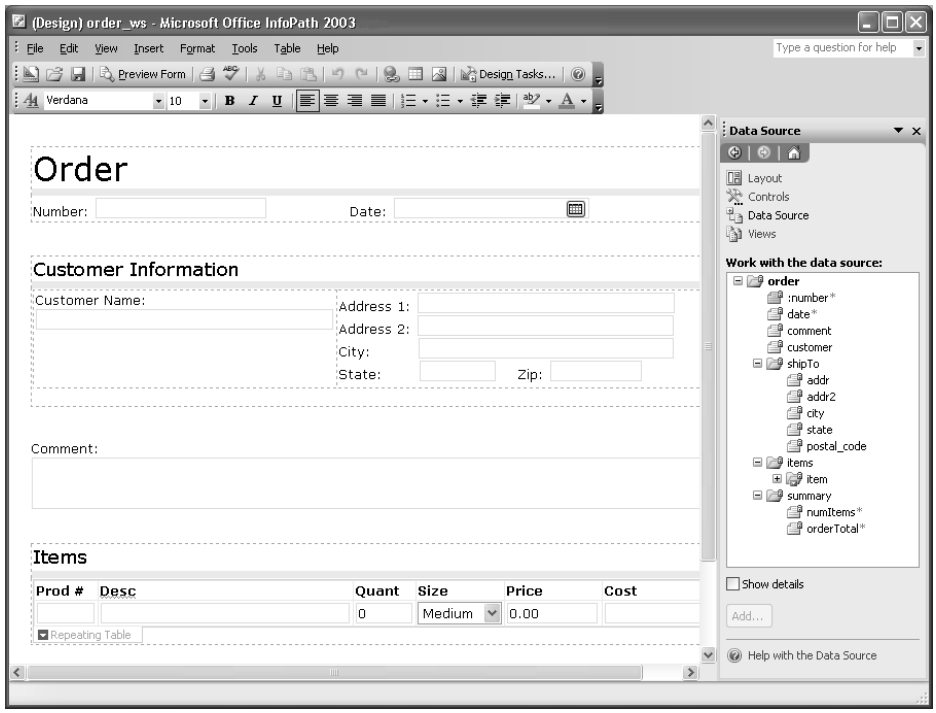

Figure 3-2 InfoPath design interface with data source in task pane

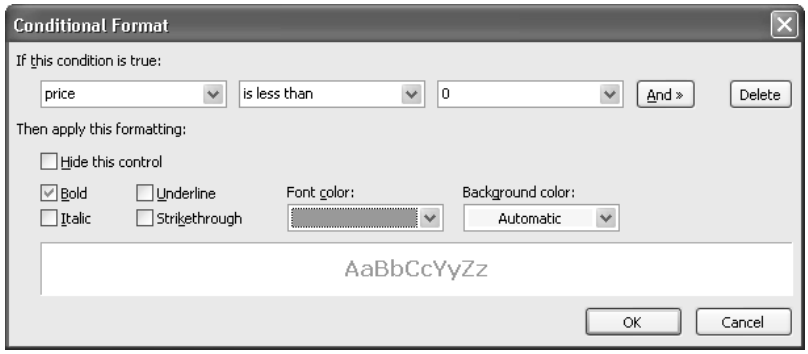

Figure 3-3 InfoPath conditional formatting dialog

InfoPath is described in detail in Chapter 9, "Designing and using forms", on page 180.

3.1.3 *Relational data*

XML elements, whether captured in Word or Excel or InfoPath (or any other way, for that matter), are as well-defined and predictable as the columns and tables of a database. XML documents of all kinds are therefore a source of information as rich as any other operational data store. Companies can aggregate, parse, search, manage, and reuse the data in documents in the same way they do the transactional data that is typically captured for relational databases.

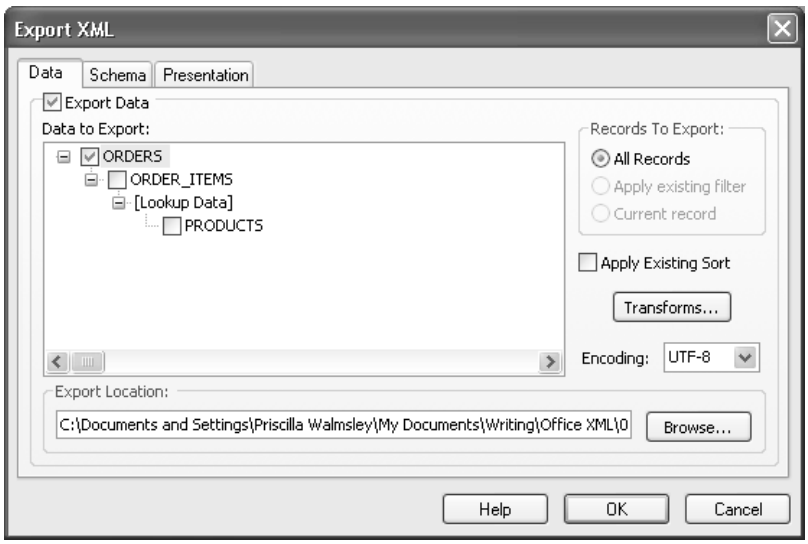

Figure 3-4 Access dialog for exporting data as XML

They can also import the document data into a database and use it in conjunction with data from other sources. In addition, they can export DBMS data as XML documents.

Figure 3-4, for example, shows the options Access offers when exporting data as XML. You can specify which tables and records to export and how to sort and/or transform them.

Figure 3-5 shows the options for exporting a schema as XML. You can choose whether or not to export the schema, and whether it should be

exported within the data document or as an independent schema document.[3]

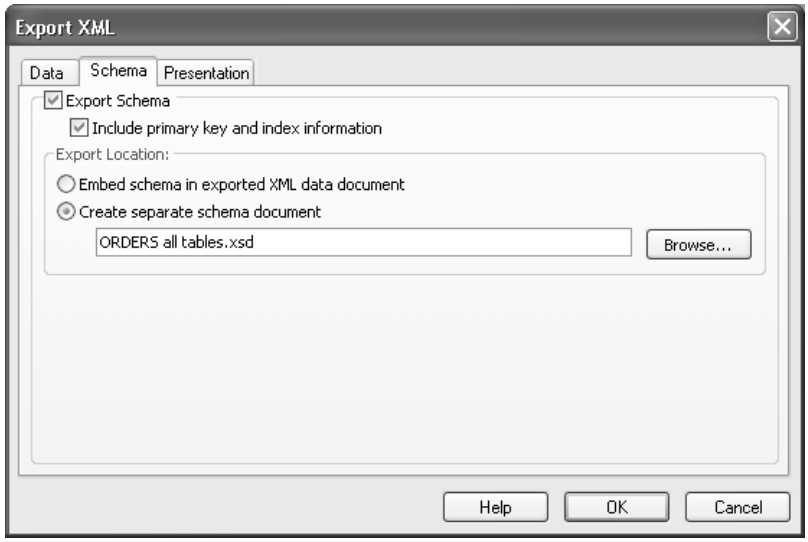

Figure 3-5 Access dialog for exporting schema as XML

3.2 | End-user data connection

Including custom XML elements in Office documents presents companies with new opportunities for business process integration.

For example, an end-user can connect directly to enterprise systems and data sources using a Web services interface. The products do the heavy lifting: natively, in the case of InfoPath, and via VBA or an external extension, in the case of Word and Excel.

The data from the Web service can be cached by the product, which can then disconnect from the server. The user still maintains the ability to work with that data, even while disconnected. For this reason, Microsoft refers to

3. See Chapter 12, "Access databases and XML", on page 266 for details on Access.

Word, InfoPath, and Excel as *smart clients*.[4] Once reconnected to the corporate server, the smart client can update the data sources.

The individual end-user also benefits from this ability to search for specific information and to aggregate information from multiple sources. It eliminates such time-consuming, error-prone tasks as:

- opening and closing files to find information;
- cutting and pasting information between documents; and
- searching for labels to combine data in like fields.

3.2.1 *Spreadsheets*

Consider how a smart client can assist in the creation and processing of expense reports.

Prior to leaving on a trip, during which she won't have access to the corporate network, Ellen opens the Excel worksheet shown in Figure 3-6. Its cells are mapped to the element types of the `expenseReport` schema, as shown in the task pane.

Each mapped element type corresponds to an area of the worksheet, either a single cell or a column of cells. The mapping allows XML data to be imported into, and exported from, the appropriate areas of the worksheet.

Ellen enters her employee number and the business purpose of her trip. The other cells that are visible in the example are populated automatically from the enterprise data store.

- Her name comes from the human resources records, based on the employee number that she entered.
- The lodging and airfare amounts come from the bookings made by the travel department.
- The per diem is based on the location in the hotel booking.
- The mileage is the calculated distance between the airport and Ellen's home address, also taken from the human resources records.

4. What Microsoft calls a "smart client" is essentially what the industry calls a "rich client". Perhaps Microsoft believes that if you're smart, you ought to be rich. Our readers – smart by definition – should agree!

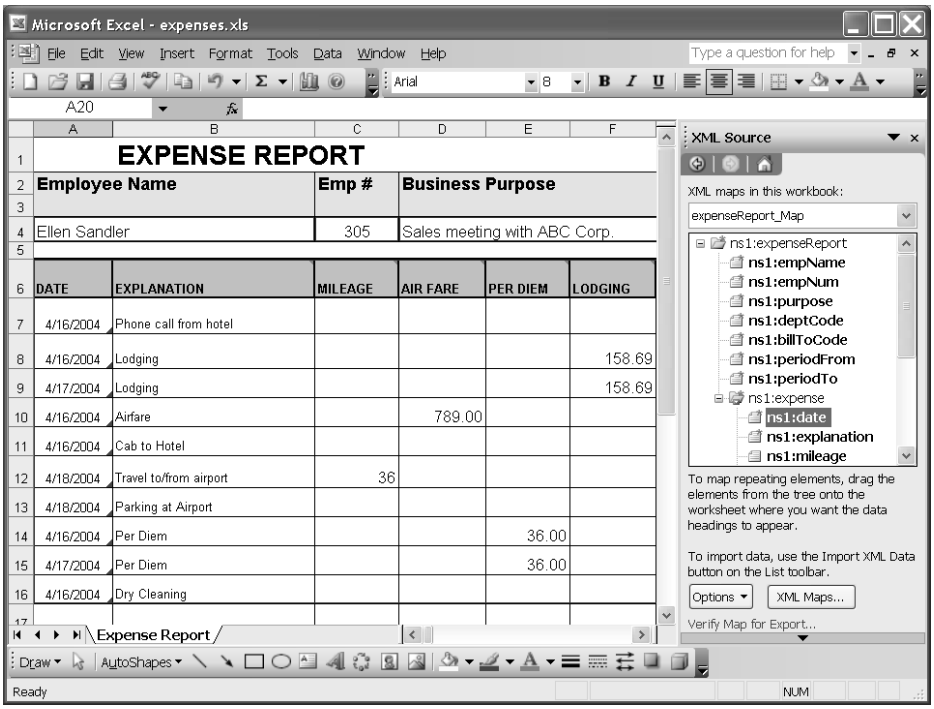

Figure 3-6 Excel worksheet and task pane with XML source map

Ellen adds more items to the worksheet during the trip, even without the network connection. When she returns, she completes the report and exports the mapped cells as an XML document conforming to the `expenseReport` schema.[5]

3.2.2 *Web pages*

Ellen next wants to submit her report for management approval. It needs five levels of sign-off, so she decides to post it to an internal website where all the managers can read it.[6]

5. We cover Excel's XML features in Chapter 7, "Using XML data in spreadsheets", on page 132.
6. Yes, in most organizations five levels of management would be, er, over the top!

Figure 3-7 shows how FrontPage is used to design a Web page based on an XML document. Elements can be dragged from the `expenseReport` structure in the task pane onto the main page, and styles and other formatting options can be used to present the data.

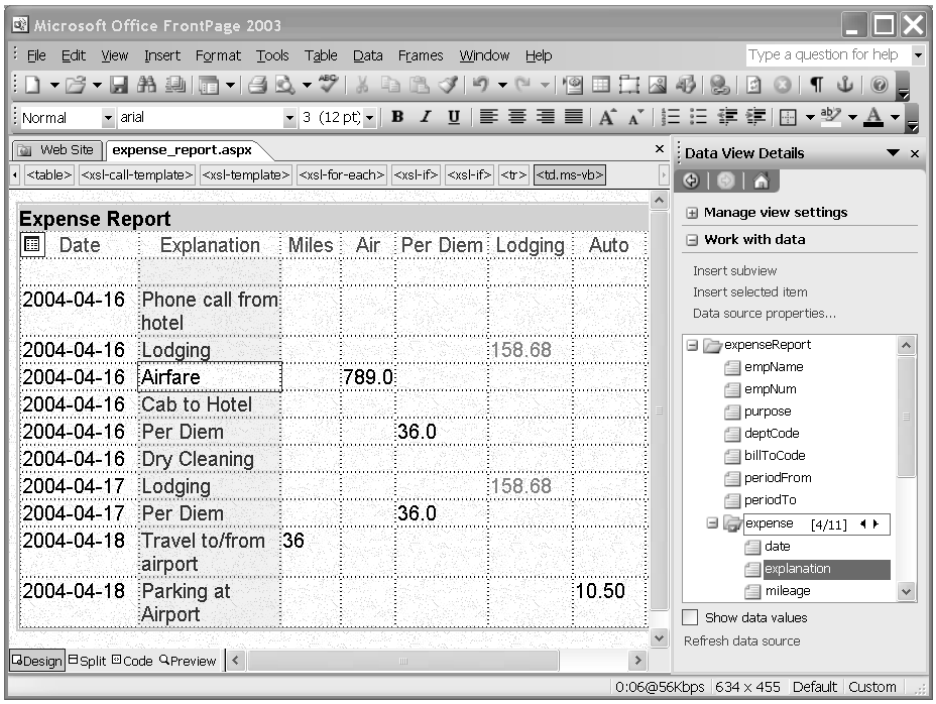

Figure 3-7 FrontPage website view with data view details in task pane

Any XML document could be used as the data source, as could databases and Web services. Sorting, grouping, filtering, and conditional formatting of the data are supported.

FrontPage generates an XSLT stylesheet from the WYSIWYG display.[7]

After Ellen's expense report is approved, the XML document is sent to the accounting department's system, which deposits the reimbursement in her bank account.

7. The FrontPage XML features are covered in Chapter 13, "Publishing XML to the Web with FrontPage", on page 294.

3.3 | Data-driven application enhancement

There are ways to enhance Ellen's experience with her expense report. Raising the per diem would undoubtedly provide the most satisfaction, but we have to stick to improvements enabled by mapping XML to the spreadsheet.

Solution developers have several ways to take advantage of the document knowledge that XML gives them: custom renditions, smart tags, and smart documents.

3.3.1 *Custom renditions*

The classic technique for data-driven application enhancement dates back to the dawn of markup languages, when GML first separated abstract data from presentation. It is to render the same data in different ways for different tasks or users.

In our scenario, for example, a report to the accounting department might contain only the summary totals from Ellen's spreadsheet, while her management gets to see every detail.

3.3.2 *Smart tags*

Office has a facility called *smart tags* that allows actions to be associated with words and phrases in a document. The "tags" don't have to be delimited, as XML tags are. Instead, they are defined by a program or by a lookup table that contains character strings and their associated actions. The product recognizes the matching strings in the document.[8]

Usually, an icon is displayed when the cursor is over a smart tag. When the user clicks on it, the associated list of actions pops up.

8. Instead of a specific character string, recognition could be based on a kind of pattern for strings that computer scientists call a *regular expression*. In this case, any character string in the document that matched the pattern would be recognized and considered a smart tag.

It is also possible to define smart tags so that an action will take place automatically when the tag is recognized. For example, recognition of the employee number in Ellen's worksheet invoked an action to send the Web service request for her employee name, which was entered into the appropriate worksheet cell.

3.3.3 *Smart documents*

The ultimate data-driven application enhancement is to respond intelligently to user input, offering context-sensitive actions and guidance, suggesting content, and providing supporting data or links to related information.

The XML facilities we've looked at so far can be used in combination to approach that goal. Add a customized task pane and even more can be done. You have what in Office-speak is called a *smart document solution*.

For example, as a user moves the cursor to different elements in a document, the task pane could display help details, related data, tools to work with the document, or related graphics. Ellen could click in a lodging cell and see the hotel contact information displayed in the task pane. She could then click on the hotel's email address and send a quick note advising the hotel of her arrival time.[9]

3.3.4 *Using the Office tools*

While all these features sound useful, there is some setup required. Fortunately, you have this book to guide you through the process. We have identified typical implementation tasks, each of which is explained in a chapter. In the context of these tasks, we present the XML features of each of the Office products. Finally, in the last part, we cover in detail some of the technologies, such as schemas and stylesheets, you may use to get there.

9. See Chapter 14, "Developing Office XML applications", on page 318 for more on smart tags and smart documents.