Chapter 1

Getting to Know XML

In This Chapter

- ▶ Introducing XML
- ▶ Examining the many uses of XML
- ▶ Deciphering what XML is and what XML isn't
- ▶ Building an XML document

Have you ever needed a document format that you could use to exchange data — either across the Internet or across an intranet? Well, eXtensible Markup Language (XML) may be just the solution. In fact, many different industries have discovered the wonders of XML — and use it extensively to help organize and classify their data.

XML is a *markup language* — it uses tags to label, categorize, and organize information in a specific way. *Markup* describes document or data structure and organization. *Content,* such as text, images, and data, is that part of the code that the markup tags contain; it's also what's of greatest interest to most everyday humans who read or interact with data or documents. XML isn't limited to a particular set of markup — you create your own markup to suit your data and document needs. The flexibility of XML has led to its widespread use for exchanging data in a multitude of forms.

And that's not all! With XML, you can send the same information to various locations — say, to a person using a mobile phone and a person using a Web browser — at the same time. In addition, you can customize the information sent out so it's displayed appropriately on the various devices.

Getting started with XML isn't difficult. Just check out this chapter, and you'll get the skinny on what markup languages are, what XML is, and what you can use XML to do.

XML (eXtreMely cooL)

If you take a close look at the use of XML in today's business world, you soon recognize that pinning down a single, definitive use for XML is nearly impossible. In fact, it is precisely the open-ended nature of XML that makes it so useful for many different things — and so difficult to put into a single, small box. Read on to see what we mean.

Mocking up your own markup

You may be familiar with *H*yper*t*ext *M*arkup *L*anguage (HTML), the markup language used to display information on Web pages. Both XML and HTML are derived from the "mother of all markup languages," Standard Generalized Markup Language (SGML) — but any similarity ends there.

HTML includes a set of predefined tags that format information for display on the Web. XML has no predefined tags — instead, you can create your own XML tags to structure your XML document so its content is in a form that meets your needs. Basically, you design your own custom markup language (actually an XML application) to do data exchange in a way that works for you.

Although XML doesn't include predefined tags, it does include very specific rules about the syntax of an XML document. You'll get a chance to explore those rules (and use said rules to create your own XML document) in Chapter 5.

X*HT*ML is yet another markup language — designed as a transition language between HTML and XML. In a nutshell, XHTML is a version of HTML that follows the strict syntax rules of XML. After you've used it for a while, you're well prepared to use XML. (We uncover the mysteries of XHTML in Chapter 4 — where you also get a chance to create an XHTML file to view on the Web.)

Separating data and context

Among the many benefits of using XML is that it automatically separates data from *context* (presentation). An XML document by itself includes no instructions about how to display the content contained in the document — it only defines the structure of the document. You can then add *styles* — formatting instructions for displaying the content — in a separate document called a *stylesheet.* This separation is actually pretty handy; you can change the display instructions without having to make any changes to your XML document. If the same style sheet is used with more than one document, you can make uniform style changes in all those documents simply by making changes in the stylesheet. All the associated XML documents follow the stylesheet's orders.

XML can be combined with both two different types of stylesheets — Cascading Style Sheets (CSS) and/or Extensible Stylesheet Language Transformations (XSLT) — for extra versatility. This makes it possible to view XML documents on the Web as more than just raw document markup — and you can change this display easily to accommodate different output devices. For example, you can use one stylesheet for display on a PDA and a separate one for printout.

We'll have more to tell about the world of CSS formatting in Chapter 7, where (lucky you) we even show you how to create and link a CSS stylesheet to an XML document. XSLT gets the same treatment in Chapter 12, where you get a chance to explore the power of XSLT stylesheets for formatting the display of an XML document.

Making information portable

XML is all about managing your data — using the best possible format available to you. To talk about how XML can handle your data as discrete bits of information, what better format is there to use than a bulleted list? Check out the following items:

- XML enables you to collect information once and reuse it in a variety of ways.
- XML data is not limited to one application format. You can design an XML document that allows you to collect data online for use in other documents, databases, and spreadsheets.

  For example, suppose your business collects sales information on a group of products by using an XML document to contain the data. The same XML data could be used to create customer purchase records, commission reports, and product-sales graphs.
- Making information portable does require planning and design before the information is collected. (You get a chance to explore the art of developing strategies for data collection in Chapter 3.)

XML means business

XML provides an easy way for businesses to manage and share information. Although XML was originally created by the World Wide Web Consortium (W3C) as a way to disseminate complex, structured data and documents over the Web, its use has expanded. Now no longer a Web-only format, XML is right at home on the business desktop.

Microsoft Office 2003 is one notable application package that includes XML tools for office applications. Using Office 2003, office documents can be created in XML format and information tagged and collected for re-use in other office applications as well as on the Web. We highlight some uses of XML in Office 2003 throughout this book.

Figuring Out What XML Is Good For

Case studies of XML never fail to mention new and exciting possibilities where XML adds value to existing environments — or solves previously intractable problems. That's probably why XML applications are widely used for everything from displaying chemical formulas to setting up a family tree. So how can *you* use the power of XML?

Classifying information

One of the most useful functions of XML involves classifying information. To see how this would work, imagine yourself in the business of selling books.

Books can be classified in many ways, but we kind of like the following classification scheme:

- Title
- Author
- Publisher
- Price
- Content Type (Fiction, Nonfiction)
- Format (Paperback, Hardback)
- ISBN

Using XML, you can create tags to classify this information. The following code shows a possible XML format for one book:

```
<book>
 <title>Night Fall</title>
 <author>Demille, Nelson</author>
 <publisher>Warner</publisher>
 <price>$26.95</price>
 <contentType>Fiction</contentType>
 <format>Hardback</format>
 <isbn>0446576638</isbn>
</book>
```

Giving your tags meaningful names that actually reflect the content makes it easier to work with the information.

Classifying the information as shown here makes it possible for you to search for — and retrieve — any item with ease. For example, after the information on all the books for your imaginary book business is collected and tucked away in XML format, you can create a list of all the authors — or authors and titles, or titles and ISBNs, whatever information you want to access. (Talk about power at your fingertips!)

We go over all the gory details of classifying information in Chapter 3, but do keep this imaginary book business in mind as you make your way through the other chapters of this book: For the sake of illustration, you get to become the next giant (imaginary) bookstore chain. We expand the book-business example in later chapters to demonstrate how you can use XML to collect and use information about inventory, customers, stores, and sales, however massive a success you become.

Enforcing rules on your data

XML excels at allowing you to create rules for the format of your data. Using either Document Type Definitions (DTDs) or XML Schemas to validate your data gives you two immediate advantages:

- It helps ensure the accuracy of the information you collect.
- It helps ensure that the information gathered is in the most usable format for your business needs.

Not sure what a DTD is? Check out the "Getting to know markup-language lingo" sidebar, later in this chapter.

Taking another look at the XML we came up with in the previous section for your imaginary book business, you can see several items for which you might want to include rules to govern how the data is formatted, such as

- A currency format for the price
- A number format for the ISBN
- A restricted selection for content type (`Fiction` or `Nonfiction`)
- A restricted selection for format (`Paperback` or `Hardback`)

You get a detailed look at creating and using DTDs and XML Schemas in Part III of this book.

Getting to know markup-language lingo

You don't have to be a markup pro to read this book or to use XML. If you're new to the markup world (or if you need to brush up on your vocabulary), the following list should help you out.

These terms are the most common ones you run into in the XML world. As you get to know them, you also get a handle on markup languages in general (including XML):

- **Attribute:** In XML, a property associated with an XML element that's also a named characteristic of the element. An attribute also provides additional data about an element, independent of element content. For example:

  ```
  <book location="GatewayMall">Whiteout
  </book>
  ```

  In this case, the element (`book`) content is `Whiteout`, but the attribute (`location`) provides additional data (`GatewayMall`).
- **Document Type Definition (DTD):** This is a statement of rules for an XML document — based on SGML (the ancestor of XML) — that specifies which *elements* (markup tags) and *attributes* (names and values associated with specific elements) are allowed in your documents. A DTD also governs the order in which the elements and attributes may appear — or (if you want to get strict) must appear.
- **Element:** A section of a document defined by start and end tags (or an empty tag), including any associated content.
- **Metalanguage:** A language used to communicate information about a language itself; many experts consider both SGML and XML to be metalanguages because they can be used to define other markup languages.
- **Nesting:** An ordering of elements that opens and closes a child element before its parent element is closed. (*Child elements* nest within *parent elements*.)
- **Schema:** An XML-based statement of rules that represents how an XML document models its data and defines its elements (or objects), their attributes (or properties), and relationships between elements.
- **Syntax:** The rules that govern the correct construction of intelligible statements in a markup language.
- **Tag; empty tag:** The markup used to enclose an element's content. An *empty element* employs a single tag; a *regular element* (which isn't empty) has an opening and a closing tag.
- **Valid:** Said of a document if it adheres to the rules outlined in an associated DTD or schema document.
- **Well formed:** Said of a markup-language document that adheres to the syntax rules for XML — which are explicitly designed to make documents easy for a computer to interpret.

Outputting information in a variety of ways

Outputting your data means releasing it from its storage locker — presumably somewhere inside the guts of your computer — and getting it to some other place where it can be a bit more useful. The great thing about XML documents is that they're not limited to any particular form of output; they can end up in a variety of different places, in whatever form is appropriate — for example, in a database, a computer monitor, a printer, or a PDA.

XML documents are at home in a wide range of processes. The phrase *post-processing* was practically tailor-made for XML; it means taking information from a document and using it in some other process or program. For example, suppose you receive a purchase order in the form of an XML document. An application that understands XML purchase orders can use that data to determine which items (and in what quantities) have been ordered — and can even send instructions to another piece of software to generate a pick list so the order can be picked, packed, and shipped from the warehouse. (Now, *that's* our kind of post-processing!)

In many cases, XML documents are used with stylesheets to provide high-quality output on-screen. You can use the same data, however, to send information to a speech-synthesis program that reads the text to a person who is vision impaired. Alternatively, that same data might also create output on a Braille reader. The same document with a layout program and a stylesheet also might be used for high-quality printouts. (Figure 1-1 gives you an idea of the infinite variety of output choices that XML makes available to you.)

The beauty of this concept is that you never need to fuss and fidget with the XML data to create output for different devices. You need only use different pieces of software that can read XML and can provide the output for a particular format or output device.

Using the same data across platforms

The good news looks, at first, like no news: XML documents are not specific to any particular platform or programming language. Okay, why is that something to e-mail home about? Think versatility. Suppose you want to exchange database information across the Web — say, use a Web browser to send information from a user questionnaire back to a Web server. To accomplish this task (and many others), you need a document format that is

- **Extensible:** An *extensible* format is one that can be tailored or customized for specific applications.
- **Open:** It's well documented and widely available.
- **Nonproprietary:** It's expressed in an accepted or standard form of notation that isn't the exclusive property of some individual, company, or organization.

These characteristics enable the document to adapt to changing conditions, to take best advantage of the work of others, and to avoid incurring extra expense or legal liability.

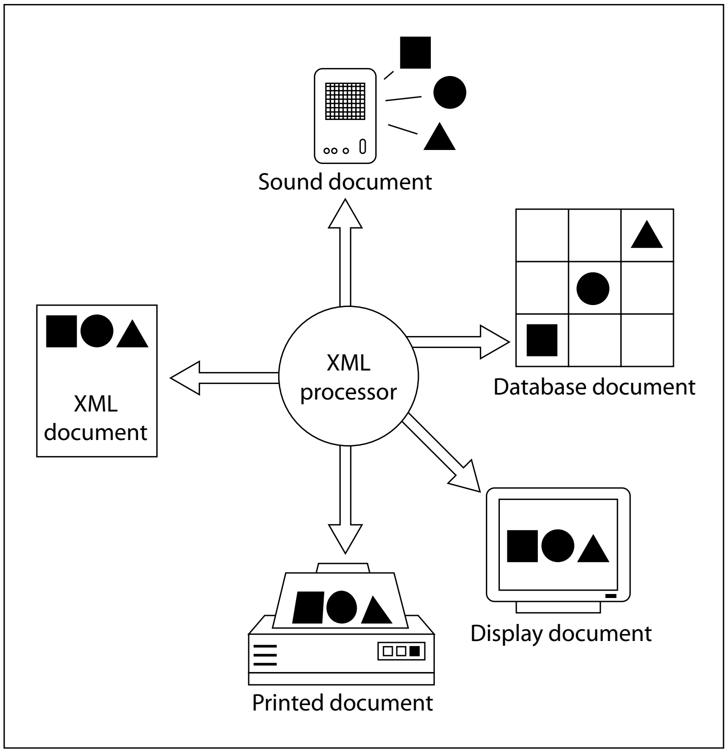


Figure 1-1: Use XML for different outputs.

Guess what? XML meets all three requirements for a document format for exchanging data — it's open, extensible, and nonproprietary. No surprise, then, that XML is the best choice for data exchange; those three magic characteristics make it a handy, consistent way to hand data around among multiple applications and multiple platforms with the most efficiency and least hassle.

Check out Chapter 2 for additional information and examples of the many uses of XML, as well as an introduction to the world of XML technologies.

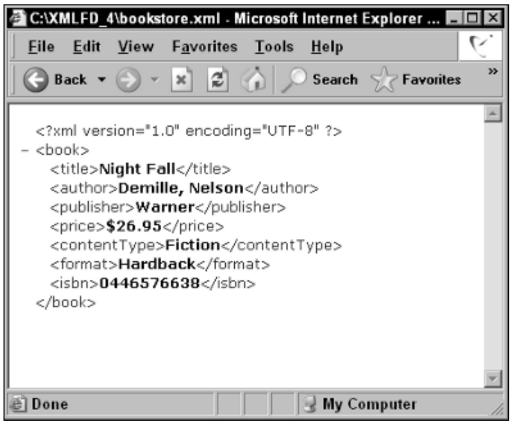
Beyond the Hype: What XML Isn't

The previous section spells out what XML is — an extensible markup language that allows you to create your own tags to develop XML applications. Now it's time to clarify what XML is *not.*

It's not just for Web pages anymore

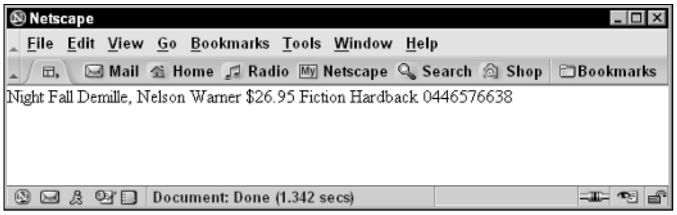
Although the World Wide Web Consortium (W3C) developed XML, it's not specifically designed only for Web pages. In fact, if you display an XML document on the Web in its raw form (without adding styles to format the display), all you'll see is the XML markup itself. Figure 1-2 shows an XML file in Internet Explorer — not much to look at! And there's even less to see when this same file is displayed in Netscape Navigator, as shown in Figure 1-3.

So banish this Web-only idea from your thoughts. XML is a markup language that allows you to organize information by creating tags to construct a specific document structure. XML documents can be viewed on the Web, but unlike HTML documents, they're not limited to the Web.


C:\XMLFD_4\bookstore.xml - Microsoft Internet Explorer ...
File Edit View Favorites Tools Help
Back
Search
Favorites
Done
My Computer

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <book>
    <title>Night Fall</title>
    <author>Demille, Nelson</author>
    <publisher>Warner</publisher>
    <price>$26.95</price>
    <contentType>Fiction</contentType>
    <format>Hardback</format>
    <isbn>0446576638</isbn>
  </book>
```

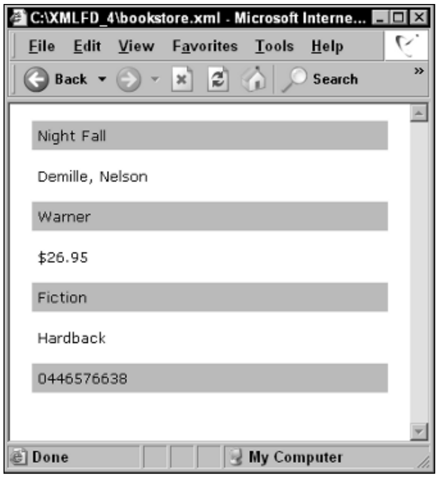
Figure 1-2: An XML file as it looks in Internet Explorer.


Netscape
File Edit View Go Bookmarks Tools Window Help
Mail Home Radio Netscape Search Shop Bookmarks
Document: Done (1.342 secs)

```
Night Fall Demille, Nelson Warner $26.95 Fiction Hardback 0446576638
```

Figure 1-3: An XML file as it looks in Netscape Navigator.

Browser support for XML is limited and variable. Hopefully this will change in the next generation of browsers, but for now XML works well in Web pages only when combined with another language (CSS) or XML technology (XSLT) to format the display of the XML information. Figure 1-4 shows our XML file when it's combined with simple CSS style instructions — now, that's more like it!


C:\XMLFD_4\bookstore.xml - Microsoft Interne...
File Edit View Favorites Tools Help
Back Search
Done
My Computer

Night Fall

Demille, Nelson

Warner

$26.95

Fiction

Hardback

0446576638

Figure 1-4: An XML file with an attached CSS stylesheet, shown in Internet Explorer.

It's not a database

Whether XML "is" a database depends on your definition of *database.* If you're defining a database as a collection of data, then yes, XML qualifies as a database. If you're defining a database as a Database Management System (DBMS) program, such as Microsoft Access, XML has some DBMS features (storage, queries, programming interfaces) but doesn't have others (queries across multiple documents, security, indexes). So, okay, you *could* use XML as a database for a small amount of data — but it wouldn't be efficient to use XML as a database for large amounts of data. (Why would you want to, when DBMS programs are designed to do exactly that?)

That's not to say XML is in any way database unfriendly. XML documents work well for both input and output, going to and from a database — and you can also use them to display database information in print or on the Web. (You get a closer look at how to use XML effectively with databases in Chapter 17.)

It's not a programming language

One of the most common misconceptions about XML is that it's a programming language. Although XML can be used with programming languages for certain types of application development, it's a markup language, not a programming language. A markup language is essentially descriptive; a programming language is for issuing logical commands. Programming languages include (for example) variables, datatypes, operators, loops, functions, and conditional statements. XML doesn't include any of these features, so it's no programming language.

Part of the confusion here is that some XML document types do include some features *found* in programming languages. For example, XML Schemas (which are themselves XML documents) include several built-in datatypes and also allow user-defined datatypes. But wait a minute: Although XML Schema documents can include datatypes — one feature of programming languages — that doesn't make them full-fledged programming languages with all the features just listed here. They remain XML documents — with an XML document structure, created with a markup language (XML). You can get XML to describe how a document will look; you can't get it to dim your house lights or start your car — at least, not without some help from an actual programming language.

Building XML Documents

When it comes to actually getting your XML tags in a row, regular old-fashioned text editors (such as Notepad) can do the job if you're just getting your feet wet with XML. If you're using Windows, you can access Notepad by choosing Start➪Programs➪Accessories➪Notepad. A new Notepad window opens. You can save the files just as you would in a word processor — and do simple functions such as copy and paste. Aside from that, though, Notepad is a pretty bare-bones program — you must insert all the markup yourself when you use a text editor such as Notepad.

Avoid using the WordPad text editor to create an XML document; it won't let you save a file with the `.xml` extension.

If the bare-bones approach just isn't good enough, you may want to check out text editors that are built specifically for XML. (We think they are definitely the way to go if you plan on using XML regularly.) These editors often look like a blend of traditional word processors and HTML editors. In fact, most XML editors work so much like word processors that you could easily forget you're working with XML.

XML editors can make your job easier and help keep those creative juices flowing! (Tracking tags and cleaning up structures can interrupt — even completely destroy — the creative train of thought.) XML editors have two distinct features that are essential for creating good XML documents:

- **Ease of markup:** XML editors, such as XMLSpy, Turbo XML, and XML Pro, can add markup to text as simply as you can turn text **bold** in today's word processors. All XML editors provide the capability to select text with a cursor and choose which markup you want to apply from a menu of selections. (See Chapter 19 for more on XMLSpy, Turbo XML, XML Pro, and other XML-authoring tools.)

- **Automatic enforcement of XML document rules:** For many applications, XML editors can determine which element types can appear in certain contexts. In this way, the editor helps you avoid making syntax or structure mistakes. For example, if you specify that the `ChapterTitle` element is valid only at the beginning of a chapter and never within an ordinary paragraph, the editor can make sure that your rule is enforced if you accidentally break it.

XML is a subset of SGML, so many authoring tools and editors previously used for SGML have been recast and are now ready to take on XML.