Foreword

Leveraging the iSeries with XML

In a relatively short time, XML has earned its place in business and Web applications by providing a common format that allows disparate computer systems to exchange information. Within two years of XML's inception, information technology vendors including Microsoft, IBM, and Netscape have incorporated XML support into their browsers and applications. Even on the IBM iSeries platform, which tends to wait for new technologies to mature before embracing them, XML is beginning to make headway, particularly in the area of Electronic Data Interchange (EDI).

In the past, a lot of information stored on computers became obsolete because the applications used to store and process that information were using proprietary formats understood only by those applications. Those of us who have been using the iSeries and its predecessors, the AS/400 and System/38, understand the value in being able to migrate information to new hardware and applications without conversion. This capability is not common on other platforms; for example, it would be very difficult and costly to extract the information contained in a spreadsheet created just 15 years ago using version 1 of Lotus 1-2-3. XML uses a simple nonproprietary format that transcends these limitations, allowing data to live beyond the applications used to create the files in which it is stored.

XML is proving to be a versatile language, capable of describing and delivering rich, structured data from any source. With the advancement of XML standards and technologies, the benefits of using XML are obvious. The iSeries is the perfect companion for XML; capable of using every technology. Together, they will leverage the platform to new heights.

This book is geared toward those who are new to XML and familiar with the iSeries. It explains the basic concepts of XML and continues to dive into more advanced topics, including XML Schemas, namespaces, security, and Web services. It also takes a look at the available technologies for the iSeries and how they work together in using XML.

By the time you finish this book, you will not only know the basics of XML; you will also be able to validate data and structures, secure XML data, and program for XML documents. All of this will help position you to take advantage of XML technology on the iSeries.

—Steve Bos

1

XML: Where Did It Come From? Where Is It Going?

In this chapter you will learn:

- ✓ What XML is
- ✓ Why XML was created
- ✓ The history of XML
- ✓ How XML is used on the iSeries (formerly AS/400) platform
- ✓ Where XML is going in the future

This chapter provides an introduction to XML, explaining why XML is important, its history, and uses for it. I will also introduce you to some ways in which you can use XML on your iSeries platform to exchange information and create Web applications. Finally, in this chapter, I will give you some insight into where XML is going.

Defining XML

Extensible Markup Language (XML) is a dialect of Standardized Generalized Markup Language (SGML). The World Wide Web Consortium created this new dialect of SGML to provide a simple and easy-to-use alternative to SGML for describing data exchanged between software applications. Although XML is a simplified version of SGML, it is powerful enough to describe almost any data in a format understood by the majority of computers in use today.

In a relatively short time, XML earned its place in business and Web applications by providing a common format that allows disparate computer systems to exchange information. Within two years of XML's inception, computer vendors such as Microsoft, IBM, and Netscape have incorporated XML support into their browsers and applications. Even on the iSeries platform, which tends to wait for new technologies to mature before embracing them, XML is beginning to make headway, particularly in the area of Electronic Data Interchange (EDI).

A lot of information stored on computers in the past has become obsolete because applications used proprietary formats understood only by similar applications. Those of us that have been using the iSeries and its predecessors, the AS/400 and System 38, understand the value in being able to migrate information to new hardware and applications without conversion. This capability is not common on other platforms. For example, it would be very difficult and costly to extract the information contained in a spreadsheet created just 15 years ago using version 1 of Lotus 1-2-3. XML uses a simple, nonproprietary format that transcends these limitations, allowing data to live beyond the applications used to create it.

The main difference between XML and other markup languages is that XML is a *meta*-markup language. In other words, it describes information about the markup, but it does not describe the domain-specific implementation. This characteristic allows XML to be adapted to fit many more needs than other markup languages such as HTML. The ability to adapt to new uses makes XML much more powerful than HTML.

With XML, you define document elements and attributes, which are then used to mark up your information. An element can represent some piece of information such as an address, a telephone number, or a person's name. Attributes are associated with an element and identify information that is typically not printed or displayed. For example, a payment type might be stored as an attribute for a payment element.

Figure 1.1 shows how a payment is defined using XML:

```
<payment type="ET">
   <amount>500.00</amount>
   <unit>USD</unit>
</payment>
```

Figure 1.1: XML defined payment.

In this example there are three elements: payment, amount, and unit. The payment element contains the amount and unit elements and has a type attribute. This example also shows how XML elements are nested. Unlike HTML tags, XML tags must be opened and closed in order. The unit closing tag in this example, which is nested in the payment tag, has to be closed before the payment tag is closed. Another difference between HTML and XML is that XML is case sensitive. A <Unit> tag is not the same as a <unit> tag.

The X in XML

The X in XML stands for the *x* in *extensible*. For this reason "Extensible Markup Language" is frequently written as "eXtensible Markup Language" and helps justify the XML acronym; apparently, e is out and X, as in X-treme, is in. This new technology with the hip acronym is supported by all of the major software vendors and describes data in a widely recognized, platform-independent manner. Although this book focuses on using XML on the iSeries platform, the platform-independent nature of XML is one of its chief strengths.

When comparing XML to Hypertext Markup Language (HTML), you find many similarities. Both are used in Web-based applications, and both use a similar syntax to describe the content of documents. Unlike most other markup languages including HTML, which restrict you to a fixed set of tags, XML allows you to create new tags. Like HTML, XML tags describe the elements within a document, but XML is a meta-markup language, which allows you to use XML to describe your own *domain-specific* elements. Domain-specific elements are elements that are useful in describing content related to a specific area. For example, there is a domain-specific Chemical Markup Language (CML) that allows chemists to describe data in a way that facilitates the exchange of chemical information.

You can create new elements that meet your needs and use them internally, or you can work within your industry to create a universally recognized markup. There are several organizations helping industry develop and publish new markups. The two largest of these are *XML.ORG*, sponsored by the non-profit Organization for the Advancement of Structured Information Standards (OASIS), and *BizTalk*, which is a Microsoft subsidiary. The URLs for these two organizations are *www.xml.org* and *www.bizztalk.org*. The goal of XML.ORG and BizTalk is to accelerate the adoption of XML as the standard way of exchanging data electronically.

As you have seen, XML allows you to create your own markup tags. If you go to the XML.Org or BizTalk Web sites, you will find complete, industry-specific markups. You can use and even extend these markups with your own elements. To prevent the possibility of name collisions, XML uses a concept called a *name-space*. An XML namespace associates elements and attributes with Universal Resource Identifiers (URI). A URI looks a lot like a Universal Resource Locator (URL) and like a URL, a URI describes a universally unique domain.

Describing Document Content

The syntax provided by XML allows you to describe the content of a document. This capability allows you to describe the content of any document and, just as important, to separate the content of a document from the document's presentation. Style is another term for the presentation format.

It is time for you to see a document that has been marked up using XML. The following example compares a recipe encoded with HTML (Figure 1.2) to the same recipe encoded with XML. I could have created my own markup for the XML version of the recipe; instead, I used Recipe Markup Language (recipeML), which is a full-blown XML markup developed to facilitate describing and exchanging recipes. The recipeML Web site is *www.formatdata.com/recipeml/*.

```
<h1>Smores</h1>
<ul>
   <li>2 Graham crackers
   <li>2 Marshmallows
   <li>2 Chocolate squares
</ul>
<p>Place two marshmallows on a graham cracker.<br>
   Place chocolate square on each marshmallow.<br>
   Cover with graham cracker.<br>
   Bake in 300 degree oven for 5 minutes.</p>
```

Figure 1.2: HTML recipe for smores.

If you are familiar with HTML, the example shown in Figure 1.2 should be easy to follow. If you are not, the following description should help. In this document, the <h1> tag describes a top-level heading, the <ul> tag describes an unordered list, <li> tags are list items, the <p> tag begins a paragraph, and <br> is a line

```
<recipe>
   <head>
      <title>Smores</title>
   </head>
   <ingredients>
      <ing>
         <amt><qty>2</qty></amt>
         <item>Graham crackers</item>
      </ing>
      <ing>
         <amt><qty>2</qty></amt>
         <item>Marshmallows</item>
      </ing>
      <ing>
         <amt><qty>2</qty></amt>
         <item>Chocolate squares</item>
      </ing>
   </ingredients>
   <directions>
      <step>Place two marshmallows on a graham cracker.</step>
      <step>Place chocolate square on each marshmallow.</step>
      <step>Cover with graham cracker.</step>
      <step>Bake in 300 degree oven for 5 minutes.</step>
   </directions>
</recipe>
```

Figure 1.3: The same recipe as in Figure 1.2, but marked up using XML.

break. I did not have to include the final </p> tag, because HTML does not require you to close a paragraph; the browser can insert the paragraph close, which can lead to problems and inconsistent results among different browsers.

The XML representation of the smores recipe shown in Figure 1.3 is quite a bit longer than the HTML representation shown in Figure 1.2. The extra information helps describe the data in a more meaningful way.

These two examples show how XML focuses on describing content rather than presentation. The recipeML example has a distinct structure and uses the descriptive <ing> tag to identify an ingredient. The HTML example uses generic tags, such as <li> to describe list items without regard to whether they are ingredients or some other element, and has no discernable structure. Also, XML follows a stricter set of rules. All tags in XML must be properly nested and have tags, whereas HTML tags can appear in any order, and in some cases closing tags are optional.

The XML example does not contain the heading and style tags that help a browser determine how to display this recipe. I will be covering these items in detail (and provide examples) when we get to syntax in Chapter 2 and style in Chapter 8.

Ensuring Documents' Integrity

One important feature of XML is that it provides built-in assurances that the form and content of information is correct and reliable. There are several ways that XML provides this assurance:

- Documents must be well-formed, adhering to XML's syntax.
- Documents can conform to a Document Type Definition (DTD) or a Schema.
- XML is prohibited from attempting to fix or understand malformed documents.

Although XML gives you the flexibility to create new tags and content as necessary, all XML documents, as well as your extensions must conform to XML's rules. There are two sets of rules; the first set ensures that the basic syntax and structure of a document are correct, the second set of rules is provided by a DTD or an XML Schema and applies domain-specific validation.

When a document follows certain XML rules, the document is considered to be well-formed. Programs that process XML documents check for conformance to these basic rules and are allowed to identify errors, but the XML specification specifically prohibits the correction of errors or interpretation of any document that is not well-formed. In addition, programs that process XML documents cannot ignore errors.

A well-formed document begins with an XML declaration and contains elements that are properly nested. The document must also have one and only one root element. In addition, attribute values must be surrounded by double quotes ("), and the characters < and & may be used only at the beginning of tags and entity references.

DTDs provide another type of validation. A DTD provides rules ensuring that the actual tags, attributes, entities, and content meet certain criteria. The rules applied by a DTD can enforce the use and order of an XML document's components. For example, the DTD for recipeML specifies that a recipe must have ingredients but nutritional information is optional. Furthermore, the recipeML DTD specifies that ingredients must appear before directions and nutritional information. Chapter 3 describes DTDs in detail.

A Schema is similar to a DTD in the validations it can perform, but it goes a step further in supporting type checking. Unlike a DTD, which can check the structure and order of tags, a Schema can make sure that the data contained within an element conforms to certain rule. For example, with a Schema you can validate a date element to be a valid date in a pre-defined format. One drawback to using a Schema over a DTD is that it takes a lot more typing to build a Schema.

Comparing XML and HTML

Many of the same people contributed to the development of both XML and HTML. The languages are also both subsets of Standardized Generalized

Markup Language (SGML). Because of this, their syntax and structure are similar; where they differ is in purpose. HTML's primary purpose was the exchange of information over the Web. XML is designed for the exchange of information by any means.

Because both HTML and XML use similar syntax and structure to mark up documents, anyone familiar with HTML has a head start in understanding XML. HTML and XML also use the same Universal Character Set (UCS) specified by the Unicode Consortium. The use of this character set makes it easier to translate documents between languages. Finally, both HTML and XML can use Cascading Style Sheets (CSS) to apply a consistent look and feel, or style, to documents. Cascading Style Sheets are a simple mechanism that associates display formatting (such as fonts, colors, and spacing) with document tags.

Now let's look at some of the differences. First and foremost is *purpose*. HTML was designed to "mark up" information displayed using a Web browser. XML is designed to structure information and does not care how the information is used. HTML directly supported formatting using tags like <font>, <emphasis>, and <bold>, whereas XML does not. With XML, you must use CSS to format a document for display using a browser.

Browsers have ignored many HTML errors (for example, missing end tags). When these errors are encountered, the browser would do its best to display what it could. A parser interpreting an XML document, on the other hand, can report errors but is prohibited from processing any document with errors.

You may have notice that I refer to HTML in the past tense. The reason is that XML is replacing HTML. A new hybrid version of HTML that is part HTML and part XML began replacing HTML back in 1999. This reformulated version of HTML known as XHTML retains HTML's functionality while conforming to XML's rigid syntax. Documents marked up using XHTML are validated using one of three DTDs. These DTDs define the HTML-like elements of an XHTML document and ensure compliance.

Why XML was Created

Documents marked up with XML have many benefits over documents that are stored as plain text or are marked up using a less capable markup like HTML. The following list describes some of those benefits:

- Data marked up with XML is self-describing.
- Organizations can exchange data in a common predefined format.
- Search engines can search documents intelligently.
- XML can be extended to new domains.
- Document presentation is independent of a document's data.
- Most browsers in use today support XML directly.
- Widespread support ensures tools availability.

XML simplifies many Web development tasks that are difficult to do in HTML. One example of this is the ability to search a document intelligently. With XML, you can search for specific content rather than trying to discern meaning from the words contained within an HTML document. For example, you could search a Web catalog site for blue 18-speed mountain bikes priced between $500 and $1,000. There is little danger this search will get a hit for Blue Mountain coffee priced at $18 per pound.

When you need to exchange information over the Internet, XML is often the best answer. With XML, you do not need specialized import and export support to handle the conversion of data from one format to another. It is easy to transform data described using XML from one format to another format. This ability to transform data is one of XML's most important features and helps facilitate the exchange of information between different databases running on more than one platform.

Another benefit of XML is that it is more rigidly structured. Because XML editors check documents to ensure that their structure is valid, browsers do not have to waste time trying to figure out the intention of a document that contains unmatched or invalid tags. The requirement that documents be valid helps ensure that browsers from different vendors display data more consistently.

Building Perspective: The History of XML

Thousands of people have contributed to the development of XML. Many of those people came up with similar ideas at similar times. Because of this, it is impossible to give a single unified history of XML, and it is likely that I have not given credit to some important contributors and their accomplishments. Keep this in mind as you read the following and remember that XML is the result of the dedicated work of many and not one person's brainchild. The following timeline shows the sequence of events leading up to today's XML:

▪ 1945	Bush describes foundation of hypertext markups
▪ 1965	Nelson coins term *hypertext*
▪ 1967	Tunnicliffe evangelizes use of generalized markup
▪ 1967	Rice develops GenCode
▪ 1969	IBM researchers led by Goldfarb develop GML
▪ 1986	SGML recognized as an ANSI standard
▪ 1987	Apple introduces HyperCard
▪ 1990	World Wide Web proposed
▪ 1993	Mosaic browser released
▪ 1996	XML's initial working draft released by W3C
▪ 1998	XML 1.0 working draft becomes recommendation
▪ 1999	XML new working groups formed, namespaces recommended

The concept of markup languages began with Vannevar Bush's July 1945 *Atlantic Monthly* article "As We May Think," which described his vision of a hypertext system he called memex. Later, around 1965, Theodor Holm Nelson coined the terms *hypertext* and *hypermedia*. Later he wrote the book *Computer*

Lib/Dream Machines, published in 1974, which helped influence the development of the World Wide Web.

Several independent groups laid the foundation for XML late in the 1960s, when they began using descriptive markup tags to supplement written information. Before this, specialized codes were used that did not describe the type of data, but rather a specific typeface, font, or style. One person who recognized the value in separating document content from format was William Tunnicliffe, who evangelized this concept in a presentation that he gave to the Canadian Government Printing Office in 1967.

Shortly after Tunnicliffe's speech, a group led by book designer Stanley Rice developed GenCode, which defined a set of generic markup tags. Norman Scharpf recognized the significance of GenCode and created the Gencode Committee to oversee its development. This committee went on to contribute to the development of Standardized General Markup Language (SGML).

At about the same time, another research group at IBM led by Charles Goldfarb, with Edward Mosher and Raymond Lorie developed the Generalized Markup Language (GML). If you are ever going to be tested on this, remember that this acronym also happens to reflect their initials. One significant feature of this new markup is that it introduced the concept of a nested element structure.

Combining the concepts of GenCode and GML, the American National Standards Institute (ANSI) presented a working draft of the first version of Standardized Generalized Markup Language (SGML) in 1980. This final version of SGML became a recognized international standard in 1986 with the publication of ISO 8879.

Following Bush and Nelson's initial activity, hypertext development faded until 1987, when Apple introduced HyperCard. Developers quickly saw that hyperlinked text was a much better way to organize and navigate text stored on computers. Apple and Microsoft incorporated this feature of hypertext links into both the Macintosh and Windows help systems.

In the late 1980s, Tim Berners-Lee, working with Robert Cailliau at the European Particle Physics Laboratory (CERN), began work on a distributed information system. In 1990, they proposed a distributed information system based on hypertext that they called the World Wide Web. They also created the first Web browser and introduced the initial version of Hypertext Markup Language (HTML). In 1993, Marc Andreessen, working for the National Center for Supercomputing Applications at the University of Illinois, released the first easy-to-install and easy-to-use Web browser, Mosaic for X.

David Raggett at Hewlett-Packard Laboratories created HTML+ in 1993 to support forms, tables, and figures. In 1995, the HTML Working Group proposed HTML 2.0, which was the first version of HTML described by a formal specification. Late in 1999, the World Wide Web Consortium (W3C) recommended version 4.01 of HTML. This was the final version of HTML; XHTML, an XML-compliant HTML derivative, is HTML's replacement.

In 1996, the W3C Generic SGML Editorial Review Board, chaired by Jon Bosak of Sun Microsystems, with support from the W3C's Generic SGML Working Group, began the first phase of XML and developed the original XML specifications. Soon after this, two new XML groups were formed: an XML Working Group and an XML Special Interest Group.

The first phase of XML culminated on February 10, 1998, when the W3C issued the XML 1.0 Recommendation. Following this recommendation, the second phase of work began. By late 1998 the W3C had restructured the XML effort and directed it toward a new XML Coordination Group. The actual definition was broken down and handled by five new working groups: the XML Schema, XML Fragment, XML Linking, XML Information Set, and XML Syntax Working Groups. These groups are part of the W3Cs Architecture domain

The second phase of XML resulted in the Namespaces in XML Recommendation, issued in January 1999, and the Style Sheet Linking Recommendation, issued later that year in June. The third phase of development continues the work of the second phase and adds a new working XML Query Working Group.

Setting Standards

Like most Internet technologies, standards define XML. Several groups contribute to the development of Internet standards, including the World Wide Web Consortium (W3C), the Internet Engineering Task Force (IETF), and the Organization for the Advancement of Structured Information Standards (OASIS).

XML's Creator, the W3C

The group credited with creating XML is the W3C. Their mission is to provide an open forum for the discussion and promotion of interoperability standards. In addition to the original XML standard, recommendations submitted by the W3 include Mathematical Markup Language (mathML), XHTML, versions of the Document Object Model (DOM), and XSL Transformations (XSLT).

The W3C strives to achieve the following goals:

- To promote technologies that provide for universal access to the Web
- To develop a software environment conducive to effective Web use
- To guide Web development to ensure that legal, commercial, and social need are met

The membership of the W3C works within a set of guidelines. The membership first approves proposals for new initiatives, known as activity proposals, and assigns the activity to a W3C Working Group. The W3C organizes these activities under four domains: the Architecture Domain, the User Interface Domain, the Technology and Society Domain, and the Web Accessibility Initiative.

The first stage of a new proposal in the W3C is a Working Draft. During the Working Draft stage, changes are expected. Following the Working Draft, a proposal evolves to become a Candidate Recommendation, then a Proposed Recommendation, and finally a Recommendation.

Guiding Internet Development at the IETF

The IETF is a loosely self-organized group that helps guide the development of the Internet and its technologies. This group is the principal source of new Internet standards and specifications. The mission statement of this group includes the following items:

- To identify and propose solutions and protocols that address Internet problems
- To make standards recommendations
- To facilitate technology transfer from researchers to the Internet community
- To provide a forum for the exchange of Internet information

The IETF is open to anyone, and its participation has grown considerably from twenty-one back in 1986 to thousands in more recent years. Each year they hold three conferences to discuss and propose Internet standards, as well as working group meetings and online discussion. Most standards start out as Internet Drafts that, after discussion and approval, are published as Requests for Comment (RFCs) and are refined until their recommendations are completed. The processes for proposing and accepting these standards are very structured; this structure reduces obstacles that might sidetrack and delay standards adoption.

OASIS: Promoting Interoperable Standards

Another group that has contributed to the development of XML and its related standards is OASIS. This nonprofit consortium creates interoperable industry specifications based on public standards such as XML, SGML, and HTML. Members of OASIS are companies and individuals with a personal stake in furthering the use of interoperability standards.

OASIS does not create and recommend new standards as the W3C and IETF do. Instead, OASIS focuses on furthering the adoptions of standards. One function of OASIS is the recommendation of specific application strategies that meet certain interoperability goals. The XML Cover Pages sponsored by OASIS provides a comprehensive online reference for XML, SGML, and their related technologies. The XML Cover Pages Web site is located at *xml.coverpages.org*. Another resource sponsored by OASIS is XML.ORG, which provides an independent resource for the use of XML in industrial and commercial settings. The XML.ORG Web site is located at *www.xml.org*.

Corporate and individual membership in OASIS is not cheap. Individual membership starts at $250 USD per year, and a corporate membership starts at $2,500 USD for companies with fewer than ten employees.

Each of the standards groups mentioned here maintains a Web site. Their Web sites are located at *www.w3.org*, *www.ietf.org*, and *www.oasis-open.org*. Each of these Web sites provides a wealth of information related to XML and the Internet. Visit these Web sites to read the original standards specifications or to follow the development of Internet standards.

Transform and Style with XSL

Almost every computer program takes some sort of input and converts it to another form of output. Sometimes the support for formatting the output is embedded in the language, as in the venerable RPG cycle; added to the language, like Java servlets; or defined by an open standard, such as the Common Gateway Interface (CGI). Extensible Style Language (XSL) gives XML the ability to add style to documents or transform information from one format to another.

Two specifications describe XSL. The first specification, which was the first released and is the most widely adopted, is XSL Transformation (XSLT) language. The second specification describes an XML vocabulary for specifying formatting semantics, known as XSL Formatting Language. It is somewhat confusing, but both XSL and the XSL Formatting Language share the same abbreviation (XSL). Chapter 8 describes XSLT and XSL Formatting Language in more detail.

The initial 1.0 version of XSLT became a W3C Recommendation in November of 1999. Before becoming a Recommendation, XSLT was stable, so the preliminary support that vendors provided for XSLT closely matched the Recommendation.

Because of this, early adopters of XSL have used the transformation language. The most common use of the transformation language is to take an XML document and transform it to HTML for display on the Web.

Transforming XML to HTML is a workaround for the limited support of XML in browser clients. Performing the transformation on a server and sending out the result to a client gives consistent results and is a bridge between the new XML technology and the majority of Web users, who do not run the latest browser. I suspect that this may become a long-term strategy, especially for open Internet applications, because it helps transcend the subtle differences among various vendors' browsers.

XSLT does not restrict you to converting XML to HTML. I have seen this technology used in very creative ways. For example, I have seen XSLT used to convert database files described by XML to Structured Query Language (SQL) statements, which actually created the tables, indexes, and views that the XML data described. You can also use XSLT to transform data to different Electronic Data Interchange (EDI) formats from a common XML format.

The second XSL specification, XSL Formatting Language, has not enjoyed the same stability that XSLT has had. However, after several false starts, XSL Formatting Language is finally on track to widespread adoption. The formatting language describes how to present a document, and is much more capable than a style sheet.The XSL Formatting Language uses formatting objects to take elements described in an abstract tree-like representation of a document and renders those elements in the document's actual presentation. The formatting language and formatting objects are extremely powerful, supporting features such as sorting, expressions, and complex document elements. With this power comes complexity. XML is supposed to be a simple alternative to SGML; you may want to put off adopting XSL Formatting Language until XML and CSS no longer meet your needs.

Tools of the Trade

For many people, a new project, whether Java, XML, or home improvement, is an excuse to acquire and use new tools. I am one of these people; in my case a home improvement project usually involves a couple of hours, some supplies, and a new tool—a tape measure, handsaw, or if I am really lucky, a power tool.

XML is a tool lover's dream. There are hundreds of cool new products to add to your toolbox, which support every aspect of XML. I don't understand why, but the majority of these new tools are also free. The Internet is the best place to find these XML tools and resources. Here are some sites with useful tools to get you started:

www.alphaworks.ibm.com: This Web site provides access to IBM's emerging alpha-code technologies. At this site you will find dozens of XML tools, including XML Interface for RPG, XML Lightweight Extractor, XML Parser for Java (XMLJ), XML and Web Services Development Environment, Xeena, and the XSL by Demo WebSphere Studio plug-in.

msdn.microsoft.com/xml/: Microsoft's XML site contains XML fixes, updates, and product previews for Internet Explorer (look for the Xmlinst.exe Replace Mode Tool). On the downloads page you will find tools such as Internet Explorer Tools for Validating XML and Viewing XSLT as well as XML Notepad.

xml.apache.org: This Web site, which is related to the well-known Apache Web server, contains quite a few open-source tools, including Xerces, Xalan, Crimson, and Xang. Although the names are odd, these tools are some of the best available; visit the Web site to find out whether Xerces is a markup language for Greek mythology, or an XML parser written in Java.

java.sun.com/xml/: You will find several Java-related XML tools at Sun's XML site, including Java API for XML Processing (JAXP) and Java API for XML Messaging (JAXM).

www.w3.org/People/Raggett/tidy/: XML Tidy is a handy freeware utility by Dave Raggett that converts HTML to XHTML. Be sure to read the Getting Started guide. If you prefer a Windows interface, download the TidyGUI version, which is also on this page.

You can use any text editor, such as Code/400 or Notepad, to edit your XML documents, but eventually you will want to get an editor that helps you with the job. Quite a few are available; some are even free or have free trials. IBM's Alphaworks Web site has several editors, including Xeena. You can go to Microsoft's Web site and download XML Notepad; the direct URL is *msdn.microsoft.com/xml/notepad/intro.asp*. If you want an open-source editor, Conglomerate provides support for both Windows and Linux; their Web site is *www.conglomerate.org*.

iSeries XML Solutions

The final section of this book describes two complete iSeries (formerly AS/400) solutions. These solutions show how to use XML on the iSeries platform and may give you some ideas on how XML can help you in your environment. In addition to XML, these solutions use RPG IV, Java, servlets, and the Apache Web server running on the iSeries. I include enough information to set up and get started with these solutions. Where appropriate, I have listed resources describing where you can get more details.

Building an XML File Viewer

The first solution describes an XML file viewer that uses XML to describe database files in a manner similar to Data Description Specifications (DDS). This solution allows you to create database files from this definition. After creating a file, you can add some data to the file and then view the formatted data with a Web browser. To view the data, a Java application builds an XML data stream from the data and then displays the results in a browser.

This solution contains two parts. The first part shows how to describe database files using XML. The first step creates an XML document that describes a database file. A DTD ensures that the XML document is valid and conforms to several rules. Next, a process converts the XML description to a DB2 database file residing on your iSeries system. The actual conversion uses XSL Transformations (XSLT) to convert the XML definition to the SQL statements used to create the table.

The second part is a file viewer that uses an XML database definition to display database file information in a browser. First, information contained in a database file is converted to an XML document using IBM's XML Lightweight Extractor (XLE). A Cascading Style Sheet helps to format the resulting XML document for display using a JavaServer Page (JSP) in a browser.

The browser-based file viewer uses a simple Java servlet that runs on your iSeries server. In order to focus on the XML parts of the application and not be sidetracked by WebSphere's complexities, I used the very capable and easy-to-configure Jakarta servlet engine. The Jakarta servlet engine is an open-source project supported by the Apache organization. There is nothing in the example that is specific to Jakarta, so it will run just fine if you already have WebSphere running and want to use WebSphere.

Building an XML Data Interchange

One application that many iSeries shops run is Electronic Data Interchange (EDI). This solution describes how to create an industry-specific markup language and DTD. This new markup language is used to exchange information in a common format. The Timber Exchange (TIMEX) markup language is specific to the timber and logging industry, but the concepts are valid for any industry-specific solution.

I begin this example by describing the elements of the TIMEX markup language. Following that, I show you how to create the TIMEX DTD, which ensures TIMEX documents adhere to some rules. Trading partners use the TIMEX DTD to ensure validity of information before sending the information.

On the receiving end, I show you how to use the XML Interface for RPG, which is a free tool that validates and parses XML documents. After determining that the information is correct, I show you how to interpret and write the XML-formatted data to a DB2 database file.

This example walks you through the steps necessary to create a new markup that supports the timber and logging industry, but these techniques apply to all industry-specific markups. If someone has already defined a widely accepted markup in your industry, use it; if none is available, now is your chance to become an industry leader and define your own markup.

Chapter Highlights

In this chapter, you learned about the problems that XML solves as well as some XML history and background.

- XML is extensible because it is a meta-markup language.
- With XML, content is separated from style.
- Well-formed documents comply with XML's syntax and structure.
- An XML document can use a DTD or Schema for additional validation.
- Search engines search XML documents more effectively.
- XHTML is now replacing HTML.
- XML uses CSS, XSL, and XSLT to provide style.

In the next chapter, I will start to describe the parts and structure of an XML document as well as basic XML syntax.