Chapter 1

The XMLSPY Game

IN THIS CHAPTER

- Understanding XML and XML Schemas
- Examining XML in the industry
- Looking at the Software Development Lifecycle Model
- Finding out how to obtain XMLSPY 5 and how to install it

THE ABILITY TO WORK WITH XML TECHNOLOGIES is an important requirement for programmers, Web developers, and database administrators today. Because of the explosive growth of many different (yet interrelated) XML technologies employed across so many different software application platforms, however, it is a challenge to become well-versed enough to design and program XML-based applications.

This hands-on book is about accelerating the XML learning curve and enabling you to become technically proficient at actually editing and working with all kinds of XML documents with the help of XMLSPY—a powerful XML Integrated Development Environment (IDE) used by a community of over one million developers world-wide. In this chapter, I present an overview of the key XML technologies, how they are used in industry today, and I tell you how to install and get started using XMLSPY 5.

If you are already familiar with XML technologies and know how to obtain and install XMLSPY, you might want to jump to Chapter 2, which covers XML editing with XMLSPY 5.

Got XML?

XML (Extensible Markup Language) in its broadest form can be regarded as a universal language for describing structured information. *Structured information* can be anything that contains both *content* (words, images, tables, and so on) and *markup*, which is the additional information that describes the content and gives it a definite meaning (table header, quantity, price, and so on). XML defines a standard syntax for describing a document's content through the use of *markup tags*, as shown in the following code snippet:

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
<first-name>Larry</first-name>
<last-name>Kim</last-name>
</person>
```

As the preceding code snippet illustrates, XML is unique in comparison to other information storage formats (for example, word processor, text, and spreadsheet files) in that it contains both the content (`Larry Kim`) and additional markup describing the semantics of the document's content (`first-name`, `last-name`). As you can see in the preceding example, the document's content and markup are interwoven within the code.

XML markup resembles HTML syntax to some extent; but unlike HTML, XML has no pre-defined tag set (such as `Title`, `H1`, `Head`, and `Body`). Instead, XML provides a general facility and syntax for defining tags and the structural relationships between them. The idea is that developers can then use this general form to create customized XML-based tag sets (also known as *markup languages* or custom XML *vocabularies*) for describing documents specific to their organization or industry. XML can be used as a flexible way to create common information formats and share content and information on the Internet.

There are many different kinds of XML documents currently used in the software industry. In addition, new XML document standards are being developed by various industry consortiums to address increasingly advanced functionality across virtually any vertical industry.

The vast majority of XML documents (or XML technologies) can be categorized as either a core XML infrastructure technology or as providing an XML-based vocabulary for a particular application or industry. This book focuses on core XML infrastructure technologies: XML Schemas, XSL/XSLT, SOAP, and WSDL, which are introduced in the following sections. Later chapters are entirely dedicated to designing and editing documents in each technology.

The important concept to grasp is that XML Schemas, XSL/XSLT, and SOAP are all simply XML documents. IT professionals, like you, must eventually design and edit these types of documents. Having a solid understanding of the core XML infrastructure technologies can give you a clear picture of what XML is and how the various technologies work together to solve a wide variety of technical challenges.

XML Schemas

An XML Schema is the World Wide Web Consortium's (W3C) official XML document definition language. It addresses many shortcomings associated with Document Type Definitions (DTD) and has industry support from all major software corporations. An XML Schema is an XML document that defines the structure and

allowable permutations that other XML documents can adopt in order to be considered a member of the common family of XML documents. Think of an XML Schema as *metadata* (essentially, data that describes data). As an analogy, think of how, in any object-oriented programming language, a class definition defines a family of objects or a relational database schema defines the data types and constraints to which a dataset must adhere to exist in a particular table. In both analogies, the class definition and relational database schema merely lay out some basic ground rules for restricting structure and data ranges, which in turn can be used in any application.

As previously mentioned, industry consortiums are joining together to develop XML Schemas that define common file formats for describing mathematical formulas, research documents, news articles, credit card transactions, accounting audits, medical prescriptions, and much more. Development of industry-standard XML Schemas enhances software application interoperability through the use of common XML-based file formats to express data and content. Using a common XML Schema, software applications can exchange information as an XML document that conforms to a particular XML Schema.

An XML Schema is most commonly used by an XML processor to validate XML documents. *Validation* is the process of verifying that an XML document conforms to the rules defined within the XML Schema. An XML processor that can perform XML Schema-based document validation (that is, an XML Schema validator) enables a developer to offload the burden of code validation from the application to the XML processor.

I discuss DTDs in Chapter 3. DTDs, however, have clearly been marked for obsolescence by the W3C.

A complete discussion of applied XML Schema design is provided in Chapters 4 and 5.

XSL/XSLT

The Extensible Stylesheet Language (XSL) and the Extensible Stylesheet Language Transformations (XSLT) are standardized XML-based vocabularies (markup languages) for changing the content and data stored in an XML document into a different output form. Using XSL, you can take content saved in an XML format and transform it into any output media (HTML, WML, PDF, PostScript, plain text) by applying a special XML *stylesheet* document written using XSL or XSLT. The XSLT transformation process is illustrated in Figure 1-1.

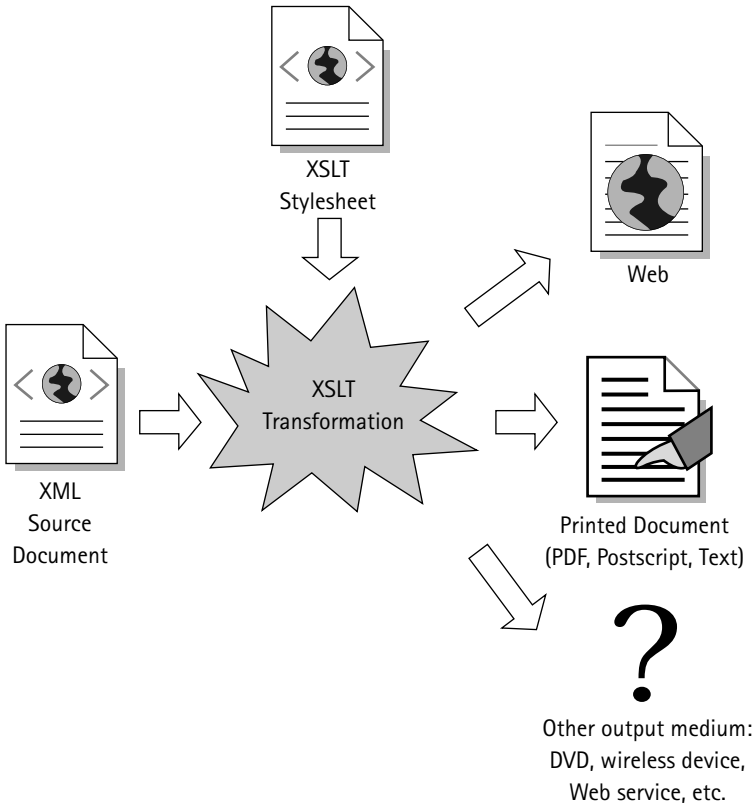

Figure 1-1: Transforming XML into a variety of output formats using XSLT.

Simple Object Access Protocol

Any networked computing environment must be able to invoke methods across both process and physical machine boundaries. This capability requires a means for locating a remote process, serializing method parameters, invoking the remote method, and deserializing the return value – all in a secure manner. Many protocols currently provide this functionality including DCE, Microsoft's DCOM, CORBA's IIOP, Java RMI and EJB, and many others. However, they all require proprietary class libraries to be loaded on both the client and remote host in order to inter-communicate. This requirement has greatly restricted distributed application interoperability.

Simple Object Access Protocol (SOAP) is a natural evolution of existing distributed technologies. SOAP is an encoding of a remote method invocation (method parameters, return type, and error codes) within an XML document that can be easily understood by any client using an XML parser. (XML parsers are freely available in every programming language and platform combination.) The SOAP document is typically sent over HTTP, the core protocol of the Web. HTTP enables greater interoperability because firewalls typically don't block HTTP traffic, and it

provides greater security because encryption is readily available through the use of Secure Sockets Layer (SSL).

XML in Today's Software Industry

The adoption of XML technologies is primarily being driven by *enterprise* applications, which are loosely defined as mission-critical, business-class software applications, as opposed to desktop applications geared for home users. Now that I've introduced the core technologies, this section provides an overview of the various industry focus areas that are being overhauled thanks to new XML technologies.

Web services

By using SOAP, XML Schema, and other related technologies (collectively referred to as *Web services*), companies can expose programmatic access to business logic over the Web. This business logic can subsequently be accessed by any device, remote process, desktop application, or Web application. Web services are transforming the World Wide Web from simple business-to-consumer applications, which require human interaction, to a distributed federation of loosely coupled services. A key area for growth will be enhancing business-to-business (B2B) application infrastructure, enabling the creation of virtual marketplaces, as well as streamlined order processing and back-office operations.

The World Wide Web

The Web in its current form is growing at an astounding rate, with an estimated base of 3 billion HTML documents distributed across the world. These documents are primarily intended to be read by people through a browser. Because it could take thousands of years to manually read through these documents, it becomes increasingly important to preserve a document's semantics. The *semantics* provide the context or meaning of a document, allowing you to better understand it. Contrast this to brute force search engines that determine a document's relevancy to a particular subject simply by calculating the number of times a keyword occurs.

Although search engines such as Google.com and Alltheweb.com have developed impressive algorithms for making sense of the vast amount of data on the Web, computers in general have quite a tough time deciphering the billions of documents out there. The challenge comes from all the miscellaneous things that clutter the actual document content: navigation bars, graphics, advertisements, applets, Flash files, and other things meant to enhance the human user experience but that don't count as actual page content as far as a Web-bot is concerned.

XSL/XSLT stylesheets are commonly used by Web developers to separate data from presentation markup on a Web page. This separation can greatly simplify the indexing, sharing, and retrieving of data on the Web by both people and Web-bots. XSLT also enables the internationalization and localization of Web sites and the

delivery of personalized Web site content to Web-enabled mobile devices. XSLT has the potential to radically change Web development. It's likely to become a critical skill of future Web developers.

XML publishing and document management

The publishing and news industries regularly work with volumes of documents, typically published in multiple output forms, most commonly in print and Web-based media. The goal has long been a single document source from which all derivative output could be generated. XML has many benefits as a storage format for the rich, structured content represented in printed publications and Web articles. Industry standard XML vocabularies such as DocBook (an XML vocabulary for describing technical publications) and NewsML (an XML vocabulary for describing news articles) facilitate the preservation of the semantics and context of information and allow for efficient retrieval and repurposing of content. Using XSLT, an XML document can be transformed into several XML-based document-layout languages including PDF, PostScript, Scalable Vector Graphics (SVG), and XHTML.

Document management refers to storing a company's documents in a document repository, thereby preserving the knowledge of a company. Document management systems have been around for a while – long before the relatively recent standardization of the XML specification. Historically, these systems have been both proprietary and costly to implement. Today, XML technologies make document management systems far easier to implement through the use of one or more industry-standard XML languages (or tag sets) for storing a particular type of information, an XML editor, and a database or XML server capable of storing XML documents. This standards-based approach to document management has the potential to unlock proprietary content management systems.

Database and application integration

The back-end processing systems of large companies are a heterogeneous mix of various distributed application platforms (J2EE, CORBA, DCOM, and so on). These applications are written in different programming languages, run on different operating systems, and use different data repositories. XML is being used in many areas to integrate enterprise applications. Most commonly, an XML document is employed as an intermediary format (or adaptor) between two or more systems. For example, an Electronic Data Interchange (EDI) message may be encoded into an XML format and then sent off to another application or database that processes the XML message. Software vendors such as Microsoft and Oracle have been adding support to their database product offerings to deal with such scenarios.

Microsoft .NET Framework

Microsoft, the world's largest software company, produces hundreds of products, Web-based services, and server applications. The challenge for the recently released .NET Framework is to make all these pieces work together and expose the combined

functionality through Web services. The Microsoft .NET product vision encompasses various application servers, SQL Server 2000, the Windows operating system, multiple programming languages, mobile devices, and more. SOAP and XML Schema bring all the pieces together, allowing tremendous application interoperability. XML development skills are likely to become an essential requirement for Microsoft developers wanting to access the various .NET products and services.

Java and XML

The Java platform enables platform interoperability at a binary level. Java programs are compiled into an intermediate language and subsequently executed on any operating system through a native Java Virtual Machine. The combination of Java and XML has the potential to improve interoperability by further decoupling the application from the underlying data storage format and opening up the application's communication protocol; these are important milestones in realizing true application portability. At the time of this writing, Sun has just recently released several powerful new standards for Web services, XML bindings, and XML messaging, which will greatly improve application interoperability.

XML technologies are interrelated and are pervasive across a wide spectrum of industry applications. Figure 1-2 graphically summarizes some of the most common uses and their relationships.

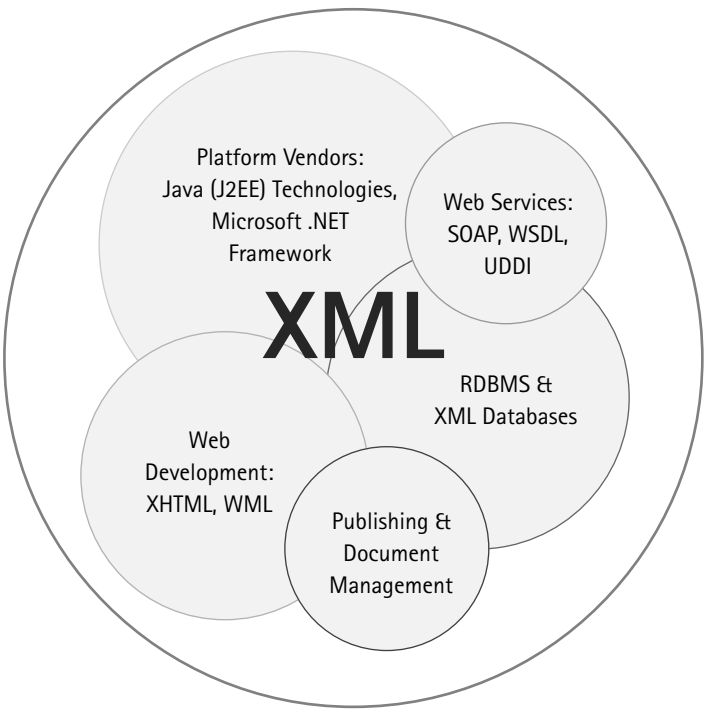

Figure 1-2: Common use of XML technologies in the enterprise.

It's safe to say that any IT professional writing any kind of code has had (or will eventually have) the need to effectively edit and work with XML documents at some level. Clearly the ability to develop using XML is a critical skill in today's job market!

The Software Development Lifecycle Model

This section compares the classic software development process and the XML software development process to explore the overall effect of the introduction of XML technologies on the software development process.

Classic software development (before XML)

Classic software development refers to the process of developing applications using procedural programming languages such as C/C++, Java, C#, COBOL, and Perl. The classic software development process typically begins with a high-level architecting process that includes modeling the software objects and their interactions. Next, an editor is used to write the source code according to the proper syntax, and the compiler is invoked to translate and link the software to an executable binary format. Finally, a debugger is used to catch any errors, thereby ensuring correct program behavior.

Classic IDEs, such as Microsoft's Visual Studio or Borland's JBuilder, have revolutionized the software development process by providing enhanced tool support for editing source code, as well as modeling and debugging tools that have enabled developers to produce higher quality software while simultaneously reducing the required effort.

Modern XML software development

XML technologies differ significantly from classic procedural programming languages in structure, syntax, and nature. Therefore, it's reasonable to expect that XML application development is also different from classic software development. You begin XML application development by developing the XML Schema, which defines a family of XML documents to be used in the application. Next, you edit and validate XML documents according to the XML Schema. Finally, a language binding must be programmed to enable the XML document to be consumed or processed by some XML-enabled framework. As previously discussed, XML document operations typically include transforming the XML document to another format, saving the XML document to a database, or transmitting the XML document to a remote process. The XML software development process is illustrated in Figure 1-3.

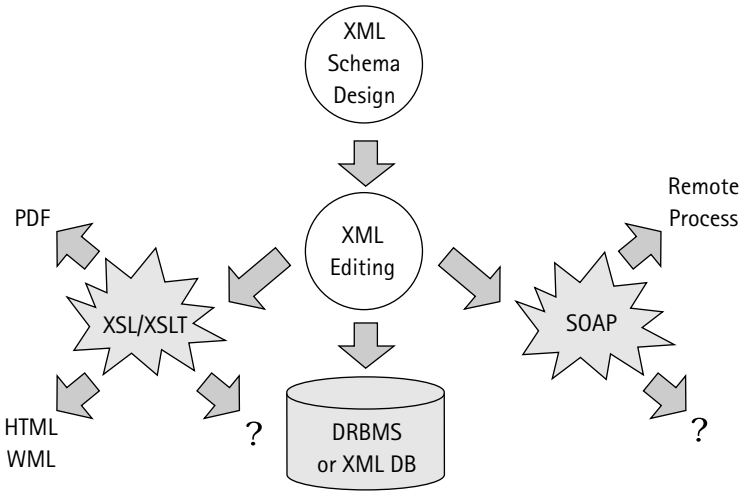

Figure 1-3: The XML software development process.

The important point here is that you must develop proficiency with XML development tools. These important tools, all covered in this book, include XML Schema modeling tools to define XML content; XML editing and validation tools to author XML documents; XSL/XSLT development and debugging tools for transforming XML; XML-to-database mapping tools for database integration applications; and SOAP development and debugging tools for building Web services.

The Spy Who Loved XML

Once there was a spy on a very important assignment. The mission: To build advanced XML and Web services applications. In order to help ensure a successfully completed mission, Spy Headquarters, which typically equips field agents with state-of-the-art gadgets like cars with ejection seats and wrist watches with laser beams, has provided the spy with an XML integrated development environment, codename: XMLSPY.

An XML IDE is a collection of tools that provides support for the development of critical XML technologies: XML Schemas and DTDs, XSL/XSLT, SOAP, and Web services, as well as XML editing and validation. XMLSPY 5 is an XML IDE. It's not meant to replace an existing classic software programming IDE, Web-development tool, or database programming/administration tool. Instead, XMLSPY 5 complements and enhances an external developer tool by providing comprehensive support for the XML development component of any potential application. XMLSPY 5 also provides tools and features to help cross the boundary from a pure XML technology to a particular language binding, server runtime environment, or database. Figure 1-4 illustrates how an XML IDE complements existing software development tools.

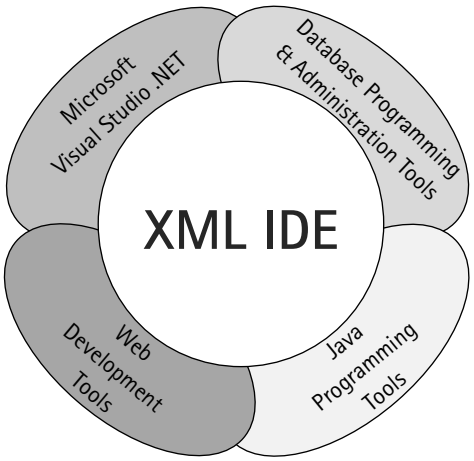

Figure 1-4: An XML IDE, such as XMLSPY 5, provides XML development support and complements other software development tools.

Just as classic IDEs have revolutionized the software development process over the past two decades, XML IDEs – in particular, XMLSPY 5 – are modernizing XML and Web services development by providing editing support for all XML technologies. As for the spy's secret identity and the outcome of the mission? You are the spy, and your mission will be accomplished with the help of XMLSPY and the *XMLSPY Handbook* as your secret guide to the intricacies of XML.

License to XML

XMLSPY 5 is developed by Altova, Inc. (`www.altova.com`) and is available in Enterprise, Professional, and Home Editions that vary in price and feature-set. The CD-ROM that accompanies this book includes a special 90-day trial version of the XMLSPY 5 Enterprise Edition. This special version differs from the version that you can download from the XMLSPY Web site (`www.altova.com`) in that it enables you to obtain a 90-day evaluation key code to ensure that you have more than enough time to complete all the exercises in this book. The regular downloadable version from the Altova site operates as a 30-day evaluation copy.

XMLSPY 5 system requirements

To install and use XMLSPY 5, your system should have the following minimum specifications:

- Windows 95/98/ME/NT4/2000/XP
- 64MB of memory
- 30MB free disk space
- Internet Explorer 5.5 or higher

Additional memory resources will allow you to parse and process larger documents.

Installing XMLSPY 5

Insert the CD-ROM into your disk drive and the InstallShield application is automatically launched. If for some reason the installer does not automatically launch, manually execute the `XMLSPYHandbook.exe` file on the CD-ROM. Select the Install XMLSPY 5 Enterprise Edition option and follow the installation instructions, choosing all the default values, which will place an XMLSPY shortcut button on the Windows taskbar and on the desktop. After installation, when you open XMLSPY 5 for the first time (by clicking either the shortcut or the program icon), a dialog box appears and prompts you to enter a valid license key-code, as shown in Figure 1-5.

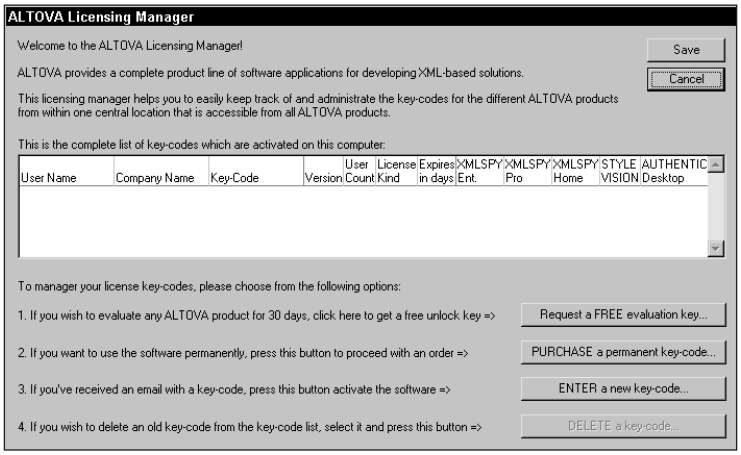

Figure 1-5: The Altova Licensing Manager.

To obtain a free key-code, click the Request a FREE Evaluation Key button and fill out the dialog box shown in Figure 1-6. Click the Request Now! button, and an e-mail containing your key-code will be sent to the address you provided. Please note that requesting and obtaining a key-code requires a working Internet connection. Copy and paste your key-code into the Key-Code field and click OK. You have now completed installing and registering XMLSPY.

Ignore the message that says `... to evaluate XMLSPY for 30 days for free press this button ...`". The key-code that will be e-mailed to you is valid for 90 days.

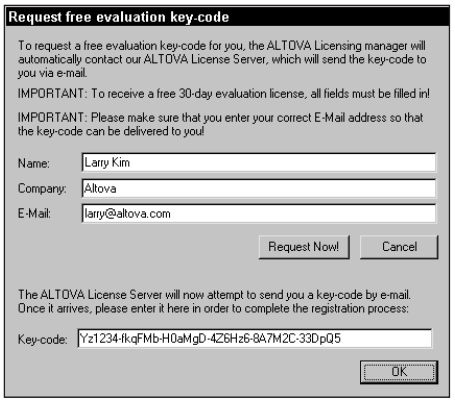

Figure 1-6: Requesting a free 90-day evaluation code requires a working Internet connection.

Updating the XMLSPY 5 Suite

Altova, Inc. periodically issues service packs containing bug fixes and other updates. The software updates are put on the Web for free download at `www.altova.com/download`. Be sure to visit this site periodically to check for updates. Valid evaluation key-code holders or customers who have purchased an XMLSPY license may upgrade to the newest version for free by uninstalling XMLSPY (using the Windows Control Panel and choosing Add/Remove Programs) and reinstalling the new version. An XMLSPY 5 key-code works for any XMLSPY service pack update, provided that the key-code has not expired.

Summary

In this chapter, I introduced XML as a language that describes structured documents in a such a way that the document is easily understood by any person or computer. This chapter covered these points:

- XML is a standard syntax for creating tag-based vocabularies, such as XML Schema, XSL/XSLT, and SOAP. These vocabularies provide a way to express a document's structure, as well as to transform and transmit the information contained within that XML document.

- XML technologies are employed in a wide variety of software development applications, yet they are all expressed using XML syntax in the form of XML documents, which in turn need to be designed and edited.
- XMLSPY 5 is an XML IDE that makes editing XML documents easy, including XML Schemas, XSL/XSLT stylesheets, and SOAP documents.
- How to obtain and install XMLSPY 5.

In the next chapter, you find out how to edit XML documents in XMLSPY 5.