Education: Online courses

XML Tutorial for Programmers

Ralf I. Pfeiffer
IBM XML Technology Group

Tutorial 1: Overview of XML

What is XML? (Yet another markup language?)

We are already familiar with markup tags from HTML. For example, Web authors often use tables to present their product information.

This is an HTML source code example, illustrating a product catalog entry as presented on a corporate Web site:

```
<TABLE BORDER=1>
        <TR>
                <TH>Product ID</TH>
                <TH>Description</TH>
                <TH>Price</TH>
        </TR>
        <TR>
                <TD>12345678-Q</TD>
                <TD>Thinkpad 2000D</TD>
                <TD>$999.99</TD>
        </TR>
</TABLE>
```

Your HTML browser renders the example above like so:

Product ID	Description	Price
12345678-Q	Thinkpad 2000D	$999.99

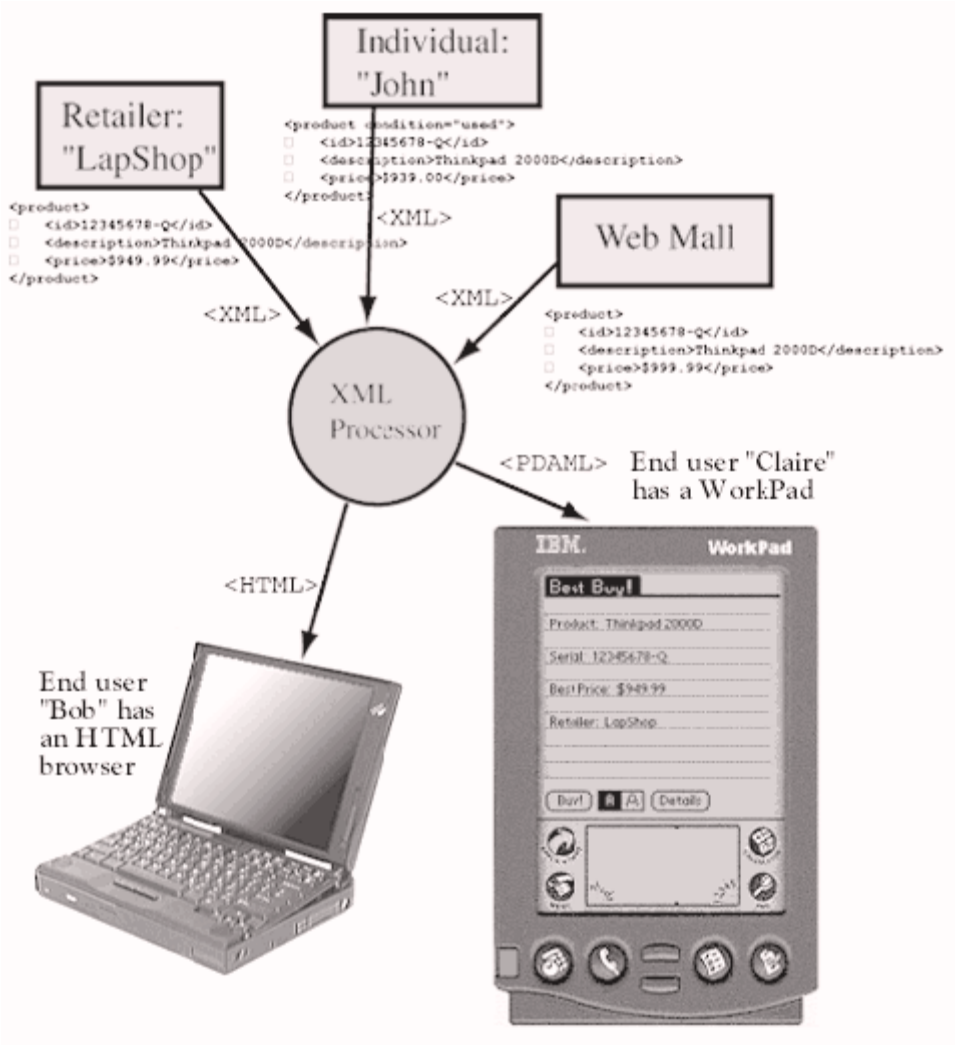
XML uses markup tags as well, but, unlike HTML, XML tags describe the *content*, rather than the *presentation* of that content. So, in the example above instead of using <TABLE>, we would define our own tag called <product>. Now we can find a specific product in all documents that follow this markup convention. We can now distinguish between products and the various data that can be presented as HTML tables.

In XML, you can also define your own attributes for tags. So the same example above could be coded in XML as:

```
<product>
        <id>12345678-Q</id>
        <description>Thinkpad 2000D</description>
        <price>$999.99</price>
</product>
```

Ironically, by avoiding formatting tags in the data, but marking the *meaning* of the data itself with <product>, we actually make it easier for a client to search various individual servers for a product and receive a product list tailored to the preferences of the user.

Figure 1 - XML makes data accessible from any device.



In Figure 1 above, note how meaningful searches can be applied to XML data, and the result can be rendered differently, depending on the destination device. Note also that the XML processor can exist on the server, the client, or both.

Using XML tags to define what your data means using the natural vocabulary of your data's domain is the key motivation for XML's invention and the basis of its usefulness.

Where does XML come from? What is its status?

XML is a simplified subset of the Standard Generalized Markup Language (SGML). SGML was standardized in 1986, based on the Generalized Markup Language invented by IBM in 1969. XML was simplified for more general use on the Web and as a data interchange format. The simplifications don't detract from XML's extensibility, but make it easier for anyone to write valid XML. It has also been simplified so that a parser can easily and quickly verify that documents are well-formed and valid.

The XML specification is an accepted recommendation awaiting formal acceptance as a standard by the W3C. The sister DOM specification, for programmatic access to XML documents is a proposed recommendation of the W3C. The DOM specification has been more recently developed, but all or most XML parsers like the IBM XML for Java (XML4J) Parser have been keeping up with and tracking the latest DOM specification.

As a technology, XML is in the unique position of being embraced by all of the leaders in the computer industry. Also, many vertical industries are embracing XML for its ability to expedite the availability of their domain-specific information for internal and external use.

There are a number of W3C-proposed *extensions* to XML. Most of the proposed extensions use the XML language, which minimizes the differences in syntax that must be learned. These extensions are beyond the scope of this tutorial, but are listed in our Appendix A - Proposed Extensions to XML.

Why does my manager want XML?

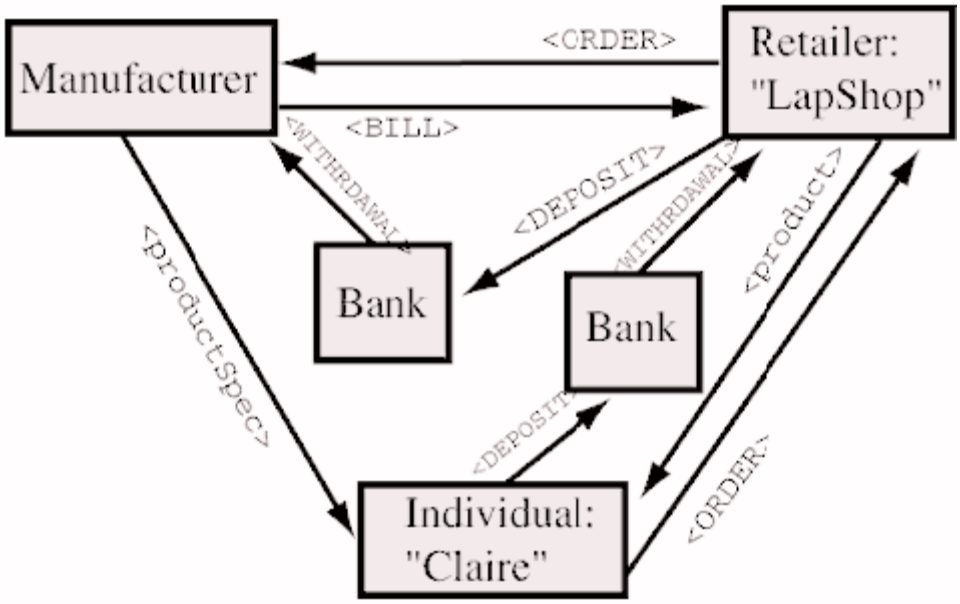
From the business perspective, almost any type of data can be represented as XML, with a grammar to describe its structure.

These are a few business areas that will benefit immediately from XML use:

e-commerce

In the business domain there will soon be specific XML languages to describe orders, transactions, inventory, and billing. These open XML languages will allow manufacturers, retailers, and consumers, even banking and accounting systems to share the same data.

Figure 2 - Claire starts a whole chain of XML transactions.



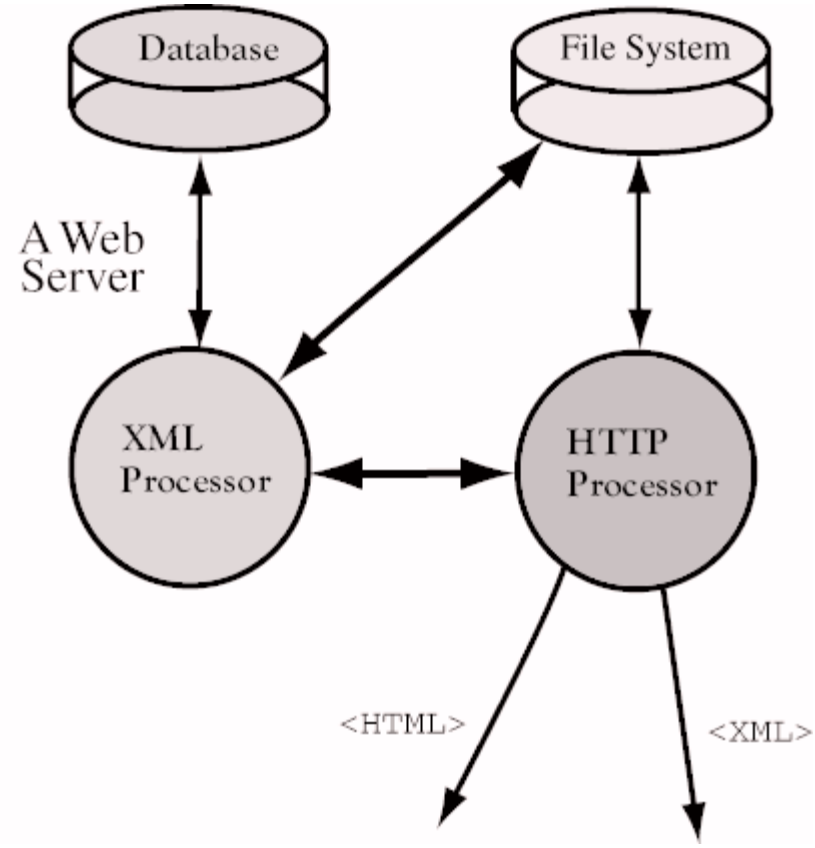
Meaningful searches, platform independence

If Web data is encoded in XML, describing the meaning of the actual content, suddenly customers can find your product or service, and purchase it, without regard for the platform that serves the data, or the platform that the client is using.

Data accessibility

Indeed, some proposed XML extensions like DCD allow XML documents to encode the datatype information and relationships in existing databases. Other XML extensions such as XQL have been proposed for specifying queries within XML. Database information can be immediately accessible as XML on corporate intranets or the Web.

Figure 3 - Databases can be presented as XML



Future Web servers will incorporate XML processors to access data from databases and the file system, and return it as HTML or XML.

Application simplification

Today's memory- and disk-consuming applications suffer from file-format bloat. That is, typical image or text processing applications are often forced to read and write many different formats. XML's open and extensible nature allows us to represent all of the functionality of different formats in one XML domain-specific grammar. An application could simply read and write XML for its domain.

XML will greatly enable the sharing of data over the Internet. Since the actual data is encoded, and not its presentation, it may be presented on any output device from the PC with a browser to the cell-phone and small PDA. Indeed, the prevalence of XML data will drive browsers and applications to be XML aware.

Also, note that XML opens up end-user access to data, and the ability to make intelligent choices. XML will further accelerate the connection between business and consumer.

Why should a programmer learn XML?

XML enables the separation of the meaning or semantics of the data from the way it is used by an application or rendered on the screen or output device. This is analogous to the practice of model/view separation in good object-oriented design. An example of model/view separation is Java's JFC Swing components. Two different instances of JFC presentation components can share the same model.

Model/view separation allows for multiple different presentations of the same type of XML data. The different presentations can be tailored to the output platform capability, or even to user preference. The simultaneous presentation of data in tabular and graphical ways is often helpful to understanding the data.

An XML document can be thought of as an instance of its grammar or DTD, similar to the way a Java object is an instance of a Class or Interface. Like a Java object or JavaBean, XML is self-declaring! Just as the Reflection capability in Java allows you do ask a class/bean about its methods and behavior at run-time, XML provides DOM, a programmatic API to access the logical structure and content of any XML document.

XML Parsers that expose the DOM API, like the XML for Java Parser will soon be ubiquitous, in Java and natively on various platforms. The benefits to the programmer are many:

- The programming task of writing multiple parsers for the data an application must read is reduced to coding against an API that presents the logical structure of the data itself!
- Domain-specific frameworks will be written on top of DOM, to present the DOM tree structure in ways natural to that domain.
- Since the DOM spec is a set of language-independent Interfaces, expressing a *logical* tree structure, translators from existing formats can easily be created by parsing existing formats, and building a DOM structure. Then you can use DOM Parsers, which not only parse XML, but can generate XML from DOM (like the XML for Java Parser) to write XML corresponding to an existing format.

In essence it is much easier for the programmer to tag his data in XML, and code against DOM Interfaces to access application data.

Summary

XML is a simple, cross-platform, and extensible way to mark up data. Like HTML, XML lives on the Web as a first-class Web format. Clients will access XML data as easily as they access HTML today.

Commerce and the software industry are gearing up to use XML because of its benefits to users. Currently there are many proprietary ways of accessing databases, catalogs, weather, and stock data. When this same data is presented as XML, search engines and agents can finally present the user with exactly what he or she is interested in. Also, since the data itself is tagged, rather than the presentation, XML can be custom-rendered for whatever device the user might have at the moment, whether it is a browser or cell-phone.

Unlike HTML, the built-in validity checking of XML allows users to trust the data. Validity checking makes XML appropriate for transactions, electronic commerce, and inventory management.

And finally, XML solves both technical and strategic problems of application programming. On the technical side, the programmer simply codes against the DOM, a set of interfaces to access the structure and content of a parsed XML document. Freed from writing parsers, the programmer can thus concentrate on the expertise and value-add of the application domain. Strategically, an open data format like XML allows applications to compete based on their feature sets, rather than compete by locking out other applications due to secret proprietary formats.

What's next?

You are probably ready to start writing XML documents. If you know a little about HTML, you will probably find that XML is not much of a leap. The next tutorial will teach you to start writing XML by comparing it to HTML and highlighting the subtle differences. You will learn coding techniques through interactive XML examples--which you can edit and parse, and correct and re-parse--from within the tutorial.

If the next tutorial, "Writing XML Documents," isn't available as you read this, please check back very soon.