Putting XSL transformations to work

Mark Colan (mcolan@us.ibm.com)
e-business evangelist, IBM Corporation
October 2001

This paper introduces the Extensible Stylesheet Language (XSL) and highlights several real-world business scenarios that benefit from the use of XSL transformations. XML data comes in many forms, so one of the most important technologies needed for XML applications is the ability to translate the data from one form to another and to convert it into document types -- such as HTML and PDF -- that can be rendered visible to end users. XSL defines a powerful mechanism for doing just that.

The Extensible Markup Language (XML) standard is now almost four years old, and a lot of progress has been made since the W3C adopted it as an officially recommended specification. XML.ORG, a registry and repository for XML vocabularies overseen by the Organization for the Advancement of Structured Information Standards (OASIS), now has well over a hundred standard vocabularies for industry-specific usage. With the Electronic Business XML (ebXML) initiative, standards common to all industries -- such as standards for purchase orders, and the like -- will begin to emerge. (See Resources for more information on the XML specification, XML.ORG, OASIS, and ebXML.) Still, compared to the enormous potential of using XML for Web-based applications, these are still the early days.

Some might fear that a large number of vocabularies represent a fragmentation in the standard. On the contrary, XML is intended as a metalanguage for establishing these vocabularies.

XML differs from HTML in that it describes the data but not its presentation. While XML can easily be understood by programmers and programs, there is the need to display the data on Web pages and other page-oriented documents as well. To maximize the flexibility of using this data, the presentation should be specified outside of the XML document, using style sheets, for example, to define its appearance.

The unique business structures that give each company its own competitive edge can be represented in private vocabularies. Companies can organize their departments separately, treating them as individual enterprises with vocabularies that reflect their way of doing business. But ultimately, information in the private definitions must be converted to a public standard for exchange with other organizations.

It is also probable that new versions of vocabularies, even with completely different structures, will replace the old as companies learn better ways to do business.

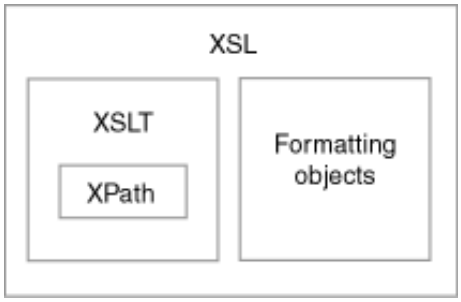
All of this points to a need for automatic conversion from one form of XML to another, from XML to HTML, and from XML to completely different presentation formats, such as PDF. What is needed, then, is a general way to accomplish mechanical translations from XML to all of these different forms.

The solution: XSL transformations

The Extensible Stylesheet Language (XSL) specification describes powerful tools to accomplish the required transformation of XML data (see Figure 1). XSL consists of the XSL Transformations (XSLT) language for transformation, and Formatting Objects (FO), a vocabulary for describing the layout of documents. XSLT uses the XML Path Language (XPath), a separate specification that describes a means

of addressing XML documents and defining simple queries. The XSLT 1.0 and XPath 1.0 specifications are complete, having become W3C recommendations on November 16, 1999. The XSL 1.0 specification (which also describes FO) is expected to reach W3C recommended status soon. (See Resources for more information on the latest versions of XSLT, XPath, and XSL from the W3C.)

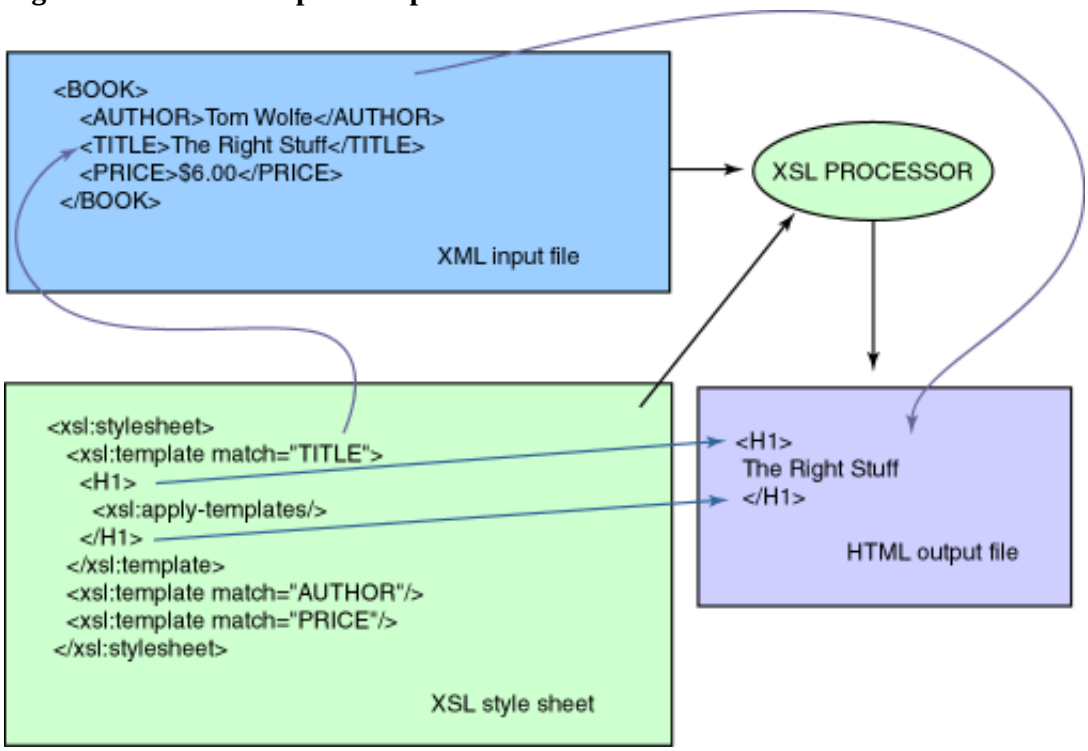
Figure 1. The Extensible Stylesheet Language and its component technologies



There are now several implementations of processors for XSLT. In particular, the Xalan project from Apache Software Foundation (see Resources) is a robust and highly compliant XSLT and XPath implementation. This tool was donated to Apache by IBM; it was developed within Lotus software by Scott Boag and his team. While Boag's team continues to develop Xalan, being part of Apache means it will enjoy contributions from individuals and other companies in the industry. With the XSLT specification in place and with the release of Xalan 1.0 in March 2000, XSLT is now stable and ready for real-world use. Xalan continues to be developed; the current release on xml.apache.org is version 2.1.

The XSLT language offers a powerful means of transforming XML documents into other forms, producing XML, HTML, and other formats. It is capable of sorting, selecting, numbering, and has many other features for transforming XML. It operates by reading a style sheet, which consists of one or more templates, then matching the templates as it visits the nodes of the XML document. The templates can be based on names and patterns. Templates include literal text that is used in the resulting transform interspersed with directives to include specific data. This technique thus defines transformations declared "by example," a simple programming model. Figure 2 illustrates a simple XSLT transformation that pulls a literal string from an XML document and places it, with formatting, into an HTML document.

Figure 2. XSLT: A simple example



XSLT is not a general-purpose programming language like Java and C++. For example, symbolic "variables" cannot be reassigned a new value, so they are really constant definitions. This limitation means that counters and accumulators are not available. Java-like "for" or "while" statements are also not

available; instead, iteration can be accomplished using recursion.

The limitations in the language definition are intended to support powerful optimization techniques. The XSLT language has an extension function that allows a style sheet to call out to modules developed in Java or C++ (depending on the implementation of the XSLT engine). This allows the use of conventional programming languages for problems that are more easily solved that way.

The most important feature of XSL is the ability to develop transformations quickly, with few lines of code. A transformation that could be developed and tested in an hour might take days to write using Java, even when an off-the-shelf XML parser such as Xerces (again, contributed to Apache by IBM) is used. One could write transformations in Perl, using XML4P to add XML parsing and DOM access support, but for many transformations it would be faster to use XSL. (See Resources for more information on Xerces and XML4P.)

XSL application scenarios

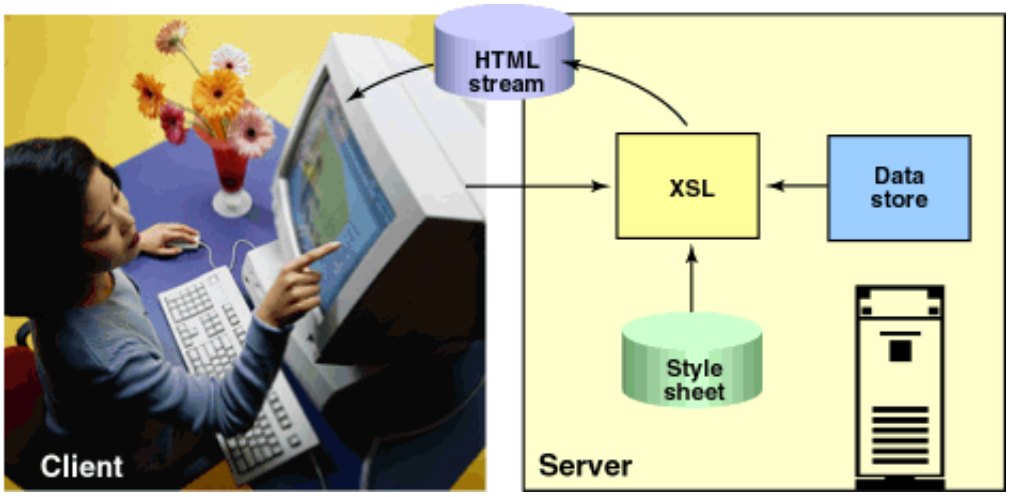
XSL is a new technology and the software industry has only begun to come up with uses for it. In the following sections, you will see some of the ways it is used in these days of its infancy. These scenarios are not intended as design patterns or definitive approaches, but rather as examples of the many ways in which XSL may be employed. The purpose in presenting these approaches is to stimulate your thinking for solving problems by using XSL in ways not yet invented.

Rendering XML as HTML

XSL was originally developed with conversion of XML to HTML in mind, hence "Stylesheet" is its middle name. In this role, XSL can be run on the client, using a style sheet either local to the client's system or stored as a resource on a server. Using XSL on the client allows processing to be distributed to each client's computer.

Most companies find it more convenient to offload processing from client workstations to servers. This simplifies the task of upgrading the power of an entire system; if more power is needed, the server can be upgraded or supplemented with other servers. An important advantage to the use of servers is that applications can be upgraded in only one place -- on the server -- rather than requiring a redeployment of application software on many client machines. A server-based transformation architecture is shown in Figure 3.

Figure 3. Server-side rendering of HTML for client browser use



XSL works well on a server. A common way to provide access is to use servlets that respond to a client's request by starting XSL and returning the resulting stream.

One can even imagine an architecture where XSL is used both on a server and on the client. For example, the server might select records that match a query, and "prune" parts of the tree that contain information not needed by the client to reduce transmission time. The client could then run XSL locally to format the XML data according to the appearance required for viewing.

Recent studies have concluded that browsing will become a small fraction of the total Internet traffic in the coming years. They suggest that even though there are uncountable Web sites today, a larger use of the

Internet (some say 10 times as much) will be in the exchange of information in XML from one server to another, in scenarios that do not include a browser. Thus, business-to-business frequently involves vocabulary translation -- translating from one XML application to another -- rather than transformation of XML to HTML.

Transcoding
Translation on demand, whether to HTML, XML, or some other form, is recognized as a common use case on an application server. IBM's recently announced WebSphere Transcoding Publisher (see Resources) automatically provides XSL translation on demand. It is capable of rendering XML to several different forms. As such, it is the logical extension of the server XSL transformation model discussed in the previous section.

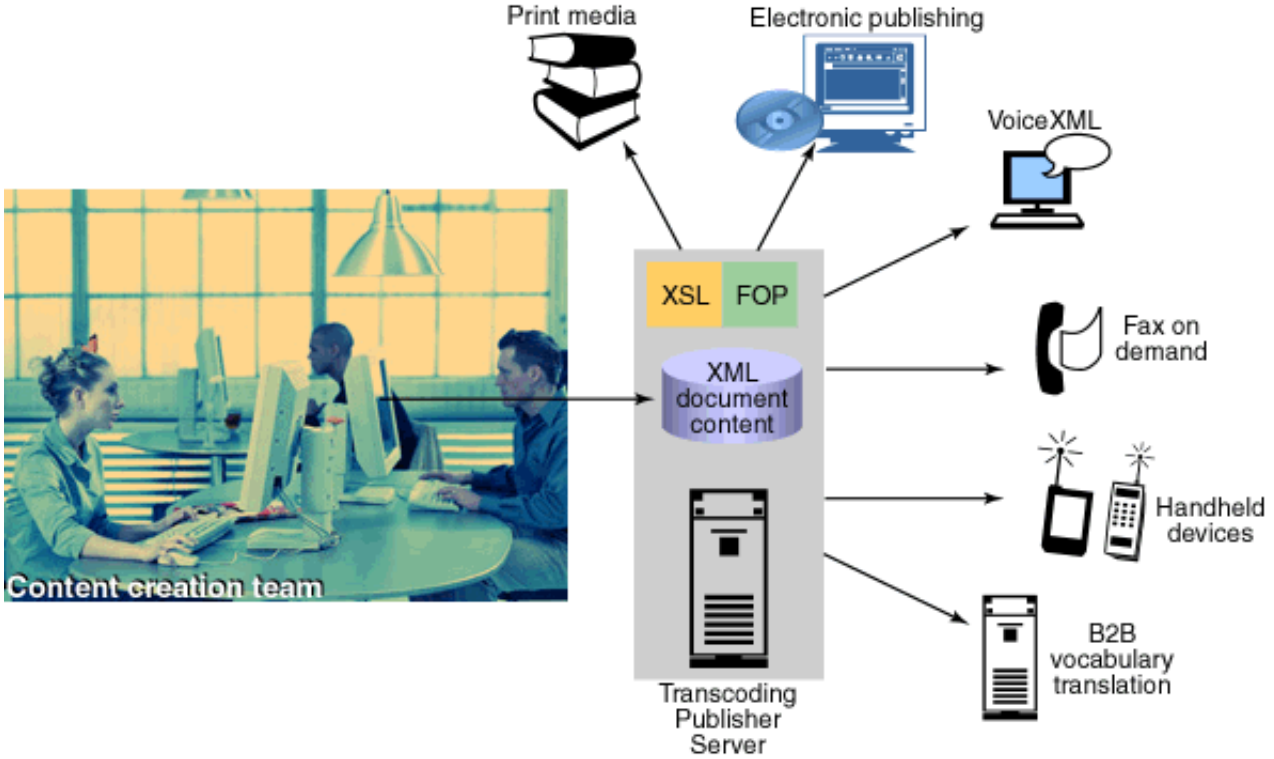
Transcoding can be used to create HTML renderings or PDF (via XSL FO and an FO processor such as Apache FOP, see Resources), thus supporting conventional desktop and laptop clients.

Transcoding can also reformat the data to Wireless Markup Language (WML, see Resources) and other forms suitable for handheld devices. Doing so often requires pruning the data to a simpler form, as well as adapting it to the device requirements for handhelds. In the Copernicus project, IBM used transcoding technology to build a system with SABRE's travel management system coupled to Nokia intelligent telephones (see Resources). Information from SABRE is transcoded to an appropriate form for the device, and then sent to the device. At that point the mobile user can make changes to his or her itinerary as required, using HTTPS to talk to specialized business objects on the server. The flexibility of the transcoding technology expands the system to support many other types of handheld devices, even when they involve vocabularies other than WML.

Finally, aside from converting XML to devices for direct client use, the Transcoding Publisher can be used for automated vocabulary translation, such as may be required for business-to-business transactions.

The major advantage of the transcoding server model is that it can start with support for a few devices, then add style sheets to support others as the need arises. In addition to applications listed above, it could be used to support traditional print media -- newspapers, magazines, books -- as well as Web publishing, or even the new e-books offline readers. It could support a fax-on-demand system. Cars will eventually be able to connect to the network, and transcoding can be set up to send information in the form they require. As set-top boxes integrate home entertainment systems with home computers, transcoding will also play a role. Figure 4 illustrates a number of possible uses for a transcoding server.

Figure 4. XSL used in the Transcoding Publisher Server



IBM's Transcoding Publisher runs the XSL processor from a servlet that handles requests. It also supports caching of transformed data, so that multiple requests for the same transformation do not require running XSL for each request.

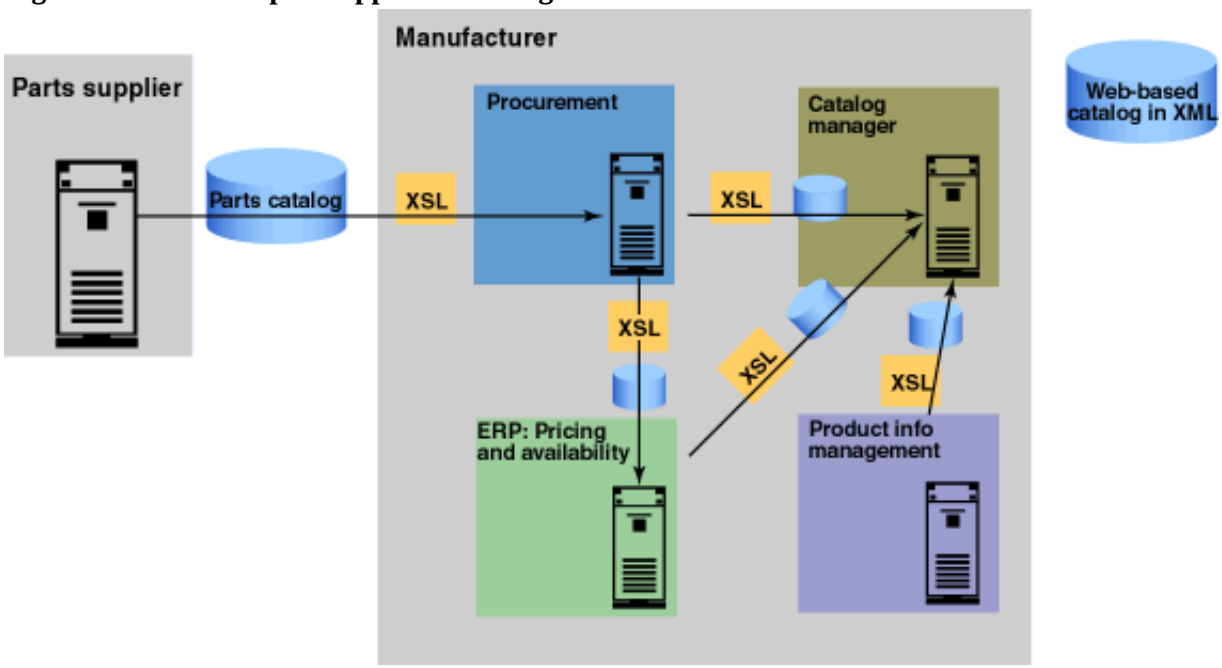
Enterprise application integration
XML is being embraced by every major software vendor. The ability to both emit XML and incorporate data expressed in XML is being added to most software products for which it makes sense.

Because XML is a common and portable data format that is, or will be, available in these products, there is a tremendous opportunity to use XML data to integrate software into a complete system. However, because the XML data may be in a variety of vocabularies, a company may need a quick and mechanical means of converting it from the form received into the one the company needs.

It is also possible to imagine that a company's internal structure might evolve into a series of entities with well-defined interfaces, and XML vocabularies that reflect their function. In this sense, the company's structure begins to resemble the structure of business-to-business relationships between companies, but on a smaller scale.

In Figure 5, XML is the exchange medium between departments of a company, and XSL is used to transform data from the private form favored by one department into a form needed for processing in another.

Figure 5. Intra-enterprise application integration



The same model can be applied to the exchange of information between companies.

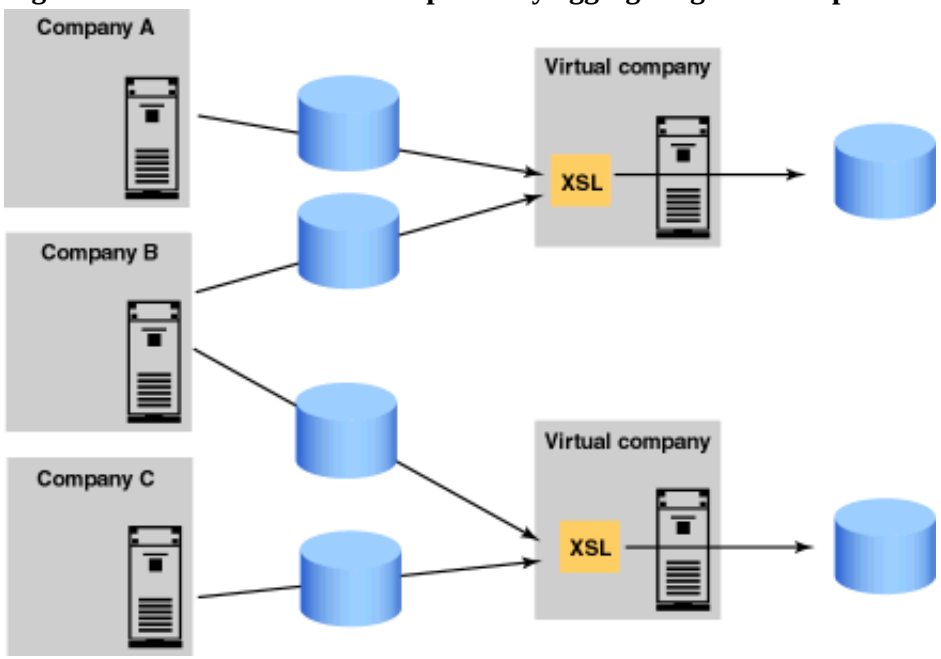
Business integration
There is a new trend in developing companies such that one company specializes in only one aspect of a complete business cycle. Such companies optimize their processes to be cost effective. Since on their own they may not be able to provide certain products or services, they may seek complementary products or services from other small companies, together offering the complete product or service required by their customers. This arrangement might be a one-time partnership, or it may exist for a longer period. For all intents and purposes, a "soft merger" of this type begins to look like a virtual company. Indeed, the virtual company may have a name different from the partners involved in creating the service or product.

In the new economy, this kind of business aggregation requires the ability to respond quickly to new opportunities. When companies expose their services and products as processes represented in XML, it is possible to use XSL with not much programming to assemble an operating e-business from the partners' component systems, as shown in Figure 6. Such companies can be described as "integration-ready."

Prior to the standardization of XML and XSL, building virtual companies from partners -- configuring

middleware to work together and writing the required business logic -- could take days, weeks, or months. While XML and XSL do not eliminate these requirements, they do provide a quickly implemented and efficient means of aggregating the partners' business data.

Figure 6. Creation of virtual companies by aggregating data and processes



In most cases there is no requirement that a company be involved in only one such partnership. One could easily imagine a company that specializes in, say, warehousing and fulfillment, providing the same service to a large number of partnerships.

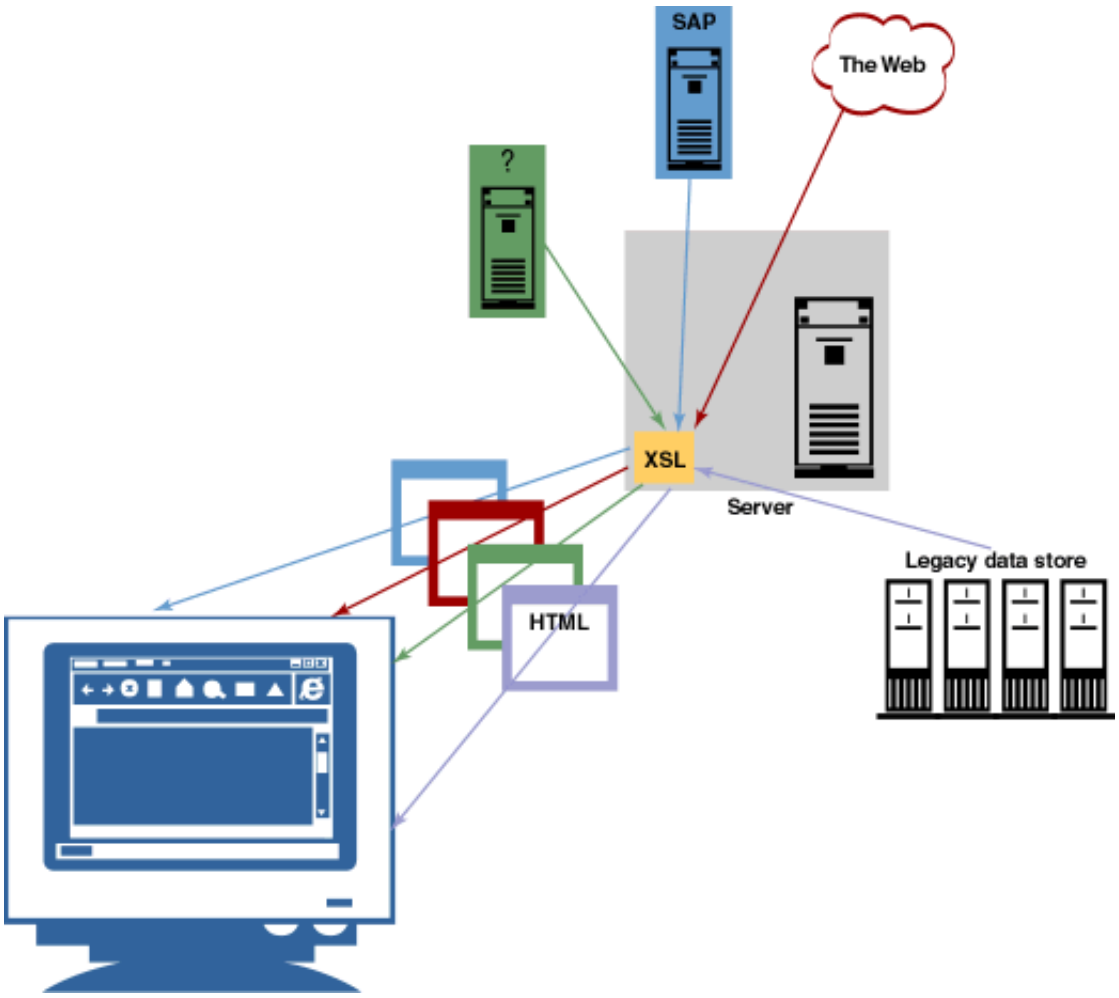
Portals

Portals such as My Yahoo! are familiar to many Web users. They allow the client to design a custom home page with live, updated information according to the user's wishes. My Yahoo! gathers data from many sources to let users request an up-to-date weather forecast for their area, current stock prices, news headlines, and the like. This information is combined into a single Web page that has different parts of the screen allocated to presenting each part of the customized report.

This model can also benefit a business worker. Suppose a clerk is employed to manage the supply of a particular line of parts needed for his company's manufacturing process. A portal could be designed to display prices or availability for certain critical components from various vendors. Information from the company's ERP system, such as inventory and forecasted demand, can be incorporated on the same page. The similarity with the My Yahoo!-type portal is the ability to gather data from a variety of resources, select according to a user's profile, and format the data for a particular screen.

When the sources of such data can provide it in XML, XSL can be used to automate the transformation required for portals. One can imagine sending HTML streams to subobjects on the browser as a means of managing regions for display. Figure 7 shows an example of pulling data from multiple sources and formatting it into a single portal screen.

Figure 7. A portal used for aggregating information from diverse sources



Code generation

In all of the examples above, XML is treated as data to be converted from one form to another, either for consumption by a client or by another server. Yet another way of using XML is to generate procedural code based on specifications described by XML data.

For example, in configuring a complex product such as a personal computer, the information about available options might be exported into XML. XSL could convert it into HTML for forms filled out by the end user. If the user chooses a SCSI adapter, a refresh of the form from the same XML might include SCSI device options that were not available until the adapter was selected. By writing a new style sheet, the same XML source document could generate forms definition in other computer languages -- Java code that instantiates and initializes controls, Windows resource definitions, even forms for older 3270 terminal-based systems.

All of this is possible because XML describes only the content; the presentation is defined using XSL style sheets. By designing various style sheets for various form systems, the same XML can be used for different kinds of applications.

The sections above list just a few application categories where XSL can be gainfully employed. Expect many other uses to emerge as the technology is embraced by creative developers around the world.

Limits of mechanical translation

XSL can solve many problems by translating XML mechanically, but a few caveats apply. It is just one tool, and it will not address every need for changing XML documents.

As stated above, the language itself is not intended for general-purpose programming. Unlike Java or C++, for example, variables can be set only once; they are really more like symbolic constants in that respect. They cannot be incremented, so loop counting is not possible. If there is a need to parse a "lastname, firstname" string into separate components, it can be done in XSL, but not easily. Such situations may call for the use of extensions plugged into XSL. With the Java version of Xalan, Java classes can be used to extend the power of an XSL processor.

Mechanical translation must be done with care. When converting from one vocabulary to another, it is important to consider the meaning of the data between tags, not just the tag name. Even with a common tag name like `<name>` (customer name? company name?), it is hard to be sure what the name means.

In addition to the meaning of the data, the format of the data must be understood. When combining listings from two catalogs of electronic parts, for example, the specifications of particular components must be expressed in a similar standard. The working voltage of a capacitor, say, could be expressed as a fixed value, a range of values, or a fixed value with a percent tolerance. The application that eventually consumes such data may understand only one form.

Both of these problems are best addressed by having very well-defined vocabularies that are agreed upon between companies. XML.ORG oversees the definition and development of such vocabularies within an industry, and it is important that the specifications reflect the input of all companies that will be using the vocabulary for e-business.

Conclusions

XSL is a powerful transformation facility that provides mechanical translation of XML documents from one form to another. It can convert to HTML, to another XML vocabulary, or to text that is not XML at all. Many transformations can be designed using only an XSL processor, and it is possible to add extensions to the processor to support particular requirements that are not easy using only XSL.

You have seen several scenarios where XSL plays a role. These initial ideas about using XSL represent solutions to certain problems seen today, but XSL can be used in many ways that have yet to be invented.

Finally, XSL by itself cannot address all incompatibilities between XML documents. When vocabularies are not well defined, either by the exact meaning of a tag or the exact format of the data associated with it, mechanical translation will not solve the problem. This underscores the importance of developing well-defined standard vocabularies for e-business usage under the auspices of a neutral standards organization such as XML.ORG.

Resources

- Participate in the discussion forum on this article by clicking **Discuss** at the top or bottom of the article.
- The Extensible Markup Language (XML) is an officially recommended standard of the W3C. For the most up-to-date information on XML, go to the W3C's XML page.
- To stay on top of current XML developments, visit XML.ORG, The XML Industry Portal.
- OASIS, the Organization for the Advancement of Structured Information Standards, is a consortium that creates interoperable specifications based on XML.
- The Electronic Business XML Initiative home page is the source for ebXML specifications, technical reports, reference materials, and news.
- Review the specification documents for XSL Transformations (XSLT), Version 1.0, XML Path Language (XPath), Version 1.0, and Extensible Stylesheet Language (XSL), Version 1.0, from the W3C.
- Get more information on Xalan, a robust XSLT and XPath implementation from the Apache Software Foundation. Apache also provides Xerces, an XML parser, XML4P, a DOM parser for Perl, and FOP, an FO-based print formatter.
- The IBM WebSphere Transcoding Publisher provides automatic XSL translation and is capable of rendering XML into several different forms.
- For a discussion of Wireless Markup Language (WML) and an extensive list of resources, read "WAP Wireless Markup Language Specification (WML)" in *The XML Cover Pages*.
- Read a news release about the SABRE/Nokia project.
- What kind of language is XSLT? by Michael Kay puts XSLT in perspective.
- Transforming XML documents by Doug Tidwell, is a three-part tutorial on how to transform XML documents into various formats, including HTML, Scalable Vector Graphics (SVG), and PDF.

About the author

Mark Colan is IBM's lead e-business technology evangelist for IBM Corporation. He gives technical, keynote, and customer presentations on Web Services and XML technologies and strategy, and has spoken at most XML conferences in 2000 and 2001, as well as Java One '98 and '99. PDFs of Mark's current presentations can be downloaded from http://ibm.com/developerworks/speakers/colan.

Before joining IBM, Mark worked at Lotus Development Corporation for 12 years, and helped to develop several commercial products. With over 20 years experience in designing and implementing commercial software products and technologies, Mark is well versed in component software strategies, operating systems, and software tools. He served as the Lead Architect for the InfoBus Technology, a Java Standard Extension developed at Lotus. You can contact Mark Colan at mcolan@us.ibm.com.