

Repurpose Your Data! The Role of XSL in e-business Solutions

Mark Colan <mcolan@us.ibm.com>

Abstract

The Extensible Markup Language ("XML") sets a standard for the exchange of business data that is completely platform- and vendor-neutral. XML is in increasingly wide use for web applications, especially for business-to-business integration.

Because XML data comes in many forms, the most important technology needed for XML applications is the ability to transform it from one form to another ("vocabulary translation"), and to convert it to visible renderings, for example in HTML or PDF documents.

The Extensible Stylesheet Language ("XSL") specification, from the W3C standards body, defines a powerful language to easily transform XML data from one form to another. This paper introduces XSL and studies several application scenarios that benefit from the use of XSL to solve real-world e-business problems.

1. The Tower of Babel Problem

The Extensible Markup Language standard ("XML") is now over three years old, and a lot of progress has been made since the W3C "recommended" the XML specification. XML.ORG, a registry and repository for XML vocabularies overseen by OASIS, now has many standard vocabularies for industry-specific usage. With the emergence of Web Services technologies -- SOAP, UDDI, and WSDL -- standards appropriate for message-level exchange will begin to emerge. Still, these are the early days of the Information Revolution: the Industrial Revolution was similar in scope and took 30 years; we're on about year 5.

Some might fear that a large number of vocabularies represent a fragmentation in the standard. To the contrary, XML is intended as a meta-language 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, we need to be able to display the data on web pages and page-oriented documents. To maximize the flexibility of using this data, the presentation should be specified outside of the XML document, for example using stylesheets to define its appearance.

We recognize that the unique business structures that give each company its own competitive edge can be represented in private vocabularies. Companies can organize their departments the same as individual enterprises, again 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.

We also expect that new versions of vocabularies, sometimes with completely different structures, are bound to replace the old as we 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 we need, then, is a general way to accomplish mechanical translations from XML to all of these different forms.

2. The Solution: XSL Transformations

The Extensible Stylesheet Language specification, known as XSL, describes powerful tools to accomplish the required transformation of XML data. XSL consists of the XSLT language for transformation, and Formatting Objects ("FO's"), a vocabulary for describing the layout of documents. XSLT uses 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 (see <http://www.w3.org/Style/XSL/>). The XSL 1.0 specification (which describes Formatting Objects) is expected to reach

W3C recommended status soon. XSLT 1.1 exists as a working draft specification to address some minor improvements to the XSLT 1.0 specification, and XSLT 2.0 and XPath 2.0 requirements documents have been created.

There are now several implementations of processors for XSLT. In particular, the Xalan project from Apache Software Foundation (see <http://xml.apache.org>) is a robust and highly-compliant XSLT and XPath implementation. This tool was donated to Apache by IBM; it was developed at Lotus Development Corporation, an IBM company, by Scott Boag and his team. While Boag's team continues to develop Xalan, being part of Apache means Xalan will enjoy contributions from others in the industry. With the XSLT specification in place and with the release of Xalan for Java 2.0 and Xalan for C++ version 1.1 in February 2001, XSLT is now stable and ready for real-world use.

The XSLT language offers powerful means of transforming XML documents into other forms, producing XML, HTML, PDFs (by way of Formatting Objects), or even Ascii formats like comma-separated text. It is capable of sorting, selecting, numbering, and can restructure XML documents for vocabulary translation. You can even use XSLT with non-XML input data, although you need to write a lexical scanner that builds a DOM tree, then XSLT can process from the DOM.

XSLT operates by reading a stylesheet, which consists of one or more templates, then matching the templates as it visits the nodes of the source XML document. The templates can be based on names and patterns. Templates can include literal text that is used in the resulting transform, interspersed with directives to include specific data. This technique thus defines transformations are declared "by example", a simple programming model.

XSLT can operate on multiple source documents (see `document()` function). It can produce any number of output documents. However, the control of output documents is not spelled out in the XSLT 1.0 Specification, so various implementations have different techniques of managing multiple output files. Xalan, for example, uses an XSLT extension. This problem will be remedied in the XSLT 1.1 Specification, which has specific language details for managing multiple output files. As XSLT implementations become compliant with that future specification, they will handle output files in the same way.

XSLT is not designed to be a general-purpose programming language in the sense of Java or 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 counted "for" or "while" statements are also not available; to some extent, iteration can be accomplished using recursion or lists of nodes.

The limitations in the language definition are intended to support powerful optimization techniques. XSLT supports an extension mechanism to allow processing more easily done in conventional languages: the extension mechanism can call modules developed, for example, in Java or C++, depending on the implementation of the XSLT software.

The most important feature of XSLT 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 like Xerces 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 XSLT.

The rest of this paper suggests ways that XSLT and XSL Formatting Objects can be used in various e-business solutions. These are not intended as design patterns or definitive approaches, but rather as examples of ways it can be employed to solve problems. The purpose is to get you thinking about other ways it can be used for solving problems in your XML solutions. ways that we have not yet invented.

3. XSL Application Scenarios: Rendering XML as HTML

XSLT was originally developed with conversion of XML to HTML in mind, hence "Stylesheet" is its middle name. In this role, XSLT can be run on the client, using a stylesheet either local to the client's system, or on a server.

Using XSLT on the client allows processing to be distributed to each client's computer, however it requires that your clients have a known version of XSLT installed. Unfortunately, the only browser that includes XSLT support -- Microsoft Internet Explorer -- has varying support for the XSLT specification. Versions before

IE5.5 are mostly incompatible with the final XSLT 1.0 Specification; stylesheets require extensive modifications to work.

With IE5.5 and MSXSL 3.0, the Microsoft browser is reasonably (about 98%) spec compliant. You can only consider deploying XSLT transformation on the client if you have control over the installed software configuration for your clients, and if you are sure that the client computers have the required processing power and memory to perform transformations.

For this reason we recommend that most e-business solutions deploy XSLT on middleware servers. IBM's middleware servers, WebSphere Applications Server and Lotus Domino Server, both include XSLT support built-in, making it easy to deploy.

Deploying XSLT on the server simplifies the task of managing computing resources: if more power is needed, the server can be upgraded or supplemented with other servers, for scalability. 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.

XSLT works well on a server. The Apache Xalan release includes a servlet wrapper to make deployment easy. The servlet responds to a client's request by starting XSLT and returning the resulting stream.

Recent studies have suggested that browsing will become the smaller part of internet traffic in the near future. Even though there are uncountable web sites today, and despite the large amount of HTTP traffic to support browsing those sites, a larger use of the internet -- some say ten 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, for example using Web Services.

XSLT can be used for converting internal XML streams to the public formats, simplifying integration with business partners or even in enterprise application integration. In the future, it will probably be used for that purpose more commonly than for rendering data to HTML.

4. XSL Application Scenarios: Transcoding

Translation on demand, whether to HTML, XML, or some other form, is recognized as a common use case on an application server. A transcoding server can be as simple as a servlet that can accept requests for a specified document rendered for a specific device or XML vocabulary, and run XSLT to produce and return the result.

IBM has recently introduced a product called the WebSphere Transcoding Publisher <http://www.software.ibm.com/developer/features/feat-transcoding.html> which automatically provides XSLT translation on demand, and adds graphical translation to support binary graphical formats for diverse devices. As such, it is the logical extension of the server XSLT transformation model discussed in the previous section.

Transcoding can be used to create HTML renderings, or PDF (via XSL Formatting Objects and a formatting objects processor like Apache FOP), thus supporting conventional desktop and laptop clients, as discussed in the previous section.

It can also reformat the data to WML and other forms suitable for mobile 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. Information from SABRE is transcoded to an appropriate form for the device and sent to the device, at which point the mobile user can make changes to his itinerary as required, using HTTPS to talk to specialized business objects on the server. The flexibility of the transcoding technology allows the system to expand 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 stylesheets 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 our home entertainment system with the home computer, transcoding will play a role.

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

5. XSL Application Scenarios: Enterprise Application Integration

XML is being embraced by every major software vendor. The ability to emit XML, and to incorporate data expressed in XML, is being added to most software products where 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, we may need a quick and mechanical means of converting it from the form we got it into the one we need.

We can 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 between companies, on a smaller scale.

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

6. XSL Application Scenarios: Business Integration

We are seeing a new trend in developing companies where one company specializes in 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 complimentary products or services from other small companies, together offering the complete product or service required by their consumers. This arrangement ranges might be one-time partnership, or may exist in the long term. 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 a new opportunity. When companies expose their services and products as processes represented in XML, it is possible to use XSLT with not much programming to assemble an operating e-business from the partners' component systems. Such companies can be described as "integration-ready".

Prior to the standardization of XML and XSL, building virtual companies from partners could take a long time -- days, weeks, months -- to configure middleware to work together, to write the required business logic. The portability of XML and easy transformation via XSLT make enterprise application business partner integration faster to develop, one step along the way to Dynamic e-business.

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. The picture above shows a company participating in two "virtual companies".

7. XSL Application Scenarios: Portals

Portals like "myYahoo" 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. "MyYahoo" allows the user to request an up-to-date weather forecast for their area, current stock prices they want to watch, news headlines, and the like, gathering data from many sources. 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 "myYahoo" 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, XSLT can be used to automate the transformation required for portals. One can imagine sending HTML streams to sub-objects on the browser as a means of managing regions for display.

8. XSL Application Scenarios: Transcoding Portals

In some ways portals seem to have the opposite function as transcoding servers. They gather information from a variety of sources and present it on one page of some device. Transcoding involves taking information from a source and translating it to support a variety of presentation devices.

What the two models have in common is the personalization of data on behalf of a particular end user. In fact, transcoding is not in conflict with portals; rather, one can view them as complimentary techniques. Using the two together allows gathering data from a variety of sources, selecting and personalizing the data, then converting it to the required device format.

9. Limits of Mechanical Translation

XSLT can solve many problems by translating XML mechanically. However, it is just one tool, and it won't address every need for changing XML documents.

The language itself is not intended as a 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 the XSLT 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>, we cannot be sure what the name means - customer name? company name? or something else?

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 could be expressed as a fixed value, a range, or a fixed value with a percent tolerance. The application which eventually consumes such data may understand only one form.

Both of these problems are best addressed by having the vocabularies be very well defined and agreed 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.

XML Schemas help solve these problems. They allow very rich type specification, allowing the XML parser to automatically validate data against a set of very specific rules to judge its validity.

10. Conclusions

XSL is a powerful transformation facility that provides mechanical translation of XML documents from one form to another. It can convert to HTML, another XML vocabulary, or text which is not XML at all. With XSL Formatting Objects, it is possible to render text to paper or devices with the accuracy of a page-layout program. Many transformations can be designed using only an XSLT processor, and it is possible to add extensions to the processor to support particular requirements that are not easy using only XSL.

We have studied several scenarios where XSL has a role. We recognize that these initial ideas about using XSL represent solutions to certain problems we see today, but that 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.

Biography

Mark Colan

Technical Evangelist
IBM Corporation
Medford
USA
Email: mcolan@us.ibm.com

Mark Colan - Mark Colan is a Technical Evangelist for IBM Corporation. He is also well known as the Lead Architect for the InfoBus Technology (a Java Standard Extension). Mark currently gives presentations on IBM's XML technologies and strategy.

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 has spoken at many leading Java and XML industry conferences and events since 1998.

He has worked for IBM and Lotus since 1984. Mark holds a degree in Computer Science from the Department of Engineering at the University of Illinois in Urbana-Champaign.