

XMLEDGE: REGISTER BY MAY 31 TO SAVE!

XML JOURNAL

The World's Leading XML Resource

Volume: 3 Issue:5

XML-JOURNAL.COM



**FULL CONFERENCE PROGRAM
INSIDE ► PAGE 53**

FROM THE EDITOR

The Changing Faces of XML
by Ajit Sagar pg. 5

INDUSTRY COMMENTARY

Grid Computing and Web Services
by Graham Glass pg. 7

PRODUCT REVIEW

jAllora
by HIT Software
Reviewed by John Evdemon pg. 57

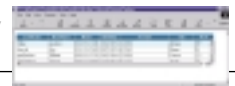
**SYS-CON
MEDIA**



XML Tips: Create Sortable HTML Tables

Sean McMullan

Using XSLT *Full-featured data grids your users will enjoy*



18

Case Study: XSLT on Wall Street

Sam Natarajan

A technical solution to managing a client's large volume of data feeds



22

XML in Transit: Web Services Directions

Simeon Simeonov

Keeping up with all the new initiatives in the Web services space



28

XML Feature: Data Driven Web

Pramod Jain & Satya Komatineni

Graphics with SVG *A declarative approach*



32

Preformatting: Generating Preformatted



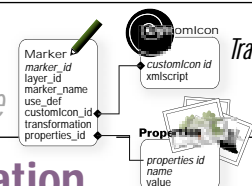
Prasad Joshi

Reports in Excel *Doing it automatically is what makes users happy*

42

Q&A: The RosettaNet Standard...

...and deep into the broader ocean of the XML marketplace



Trace Galloway

50

VoiceXML: Modular Speech Application

Hitesh Seth

Development *Key components for VoiceXML adoption*



52

Sonic Software Corporation

www.sonicsoftware.com

Sonic Software Corporation

www.sonicsoftware.com

Sprint PCS

<http://developer.sprintpcs.com>

EDITORIAL ADVISORY BOARD

JOHN EVDEMON jevdemon@vitria.com
 GRAHAM GLASS graham@themindelectric.com
 COCO JAENICKE cjaenicke@mediaone.net
 SEAN MCGRATH sean.mcgrath@propylon.com
 JP MORGENTHAL jpmorgenthal@tkimbo.com
 SIMEON SIMEONOV simeons@macromedia.com

DEPARTMENT EDITORS

EDITOR-IN-CHIEF: Ajit Sagar
 EDITORIAL DIRECTOR: Jeremy Geelan
 E-BUSINESS EDITOR: Israel Hilerio
 JAVA TECHNOLOGY EDITOR: David Johnson
 PRODUCT REVIEW EDITOR: Jim Milbery
 WEB SERVICES EDITOR: Graham Glass
 EXECUTIVE EDITOR: M'lou Pinkham
 MANAGING EDITOR: Cheryl Van Sise
 EDITOR: Nancy Valentine
 ASSOCIATE EDITORS: Jamie Matusow
 Gail Schultz
 Jean Cassidy

WRITERS IN THIS ISSUE

Tod Emko, John Evdemon, Trace Galloway, Graham Glass,
 Pramod Jain, Prasad Joshi, Satya Komatineni,
 Sean McMullan, Sam Natarajan, Ajit Sagar, Hitesh Seth,
 Simeon Simeonov, Mark Spears

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

For subscriptions and requests for bulk orders,
 please send your letters to Subscription Department

Cover Price: \$6.99/issue
 Domestic: \$77.99/yr (12 issues)
 Canada/Mexico: \$99.99/yr
 all other countries \$129.99/yr
 (U.S. Banks or Money Orders)

EDITORIAL OFFICES

SYS-CON MEDIA

135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
 TELEPHONE: 201 802-3000 FAX: 201 782-9637

XML JOURNAL (ISSN# 1534-9780)
 is published monthly (12 times a year)
 by SYS-CON Publications, Inc.

Periodicals postage pending
 Montvale, NJ 07645 and additional mailing offices.

POSTMASTER: Send address changes to:

XML JOURNAL, SYS-CON Publications, Inc.,
 135 Chestnut Ridge Road, Montvale, NJ 07645.

©COPYRIGHT

Copyright © 2002 by SYS-CON Publications, Inc. All rights reserved.
 No part of this publication may be reproduced or transmitted
 in any form or by any means, electronic or mechanical,
 including photocopy or any information storage and retrieval
 system, without written permission. For promotional reprints,
 contact reprint coordinator. SYS-CON Publications, Inc.,
 reserves the right to revise, republish and authorize its readers
 to use the articles submitted for publication.

All brand and product names used on these pages
 are trade names, service marks, or trademarks of their respective
 companies. SYS-CON Publications, Inc., is not affiliated
 with the companies or products covered in XML Journal.



**SYS-CON
 MEDIA**



WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF

from
 the
 editor

The Changing Faces of XML

The XML landscape has been changing at the speed of light in the last couple of years. The rapid evolutions and new additions to the XML universe have increased in frequency even as the technologies and markets around it have consolidated. Add new paradigms such as Web services to the mix and you see XML blazing a trail that might seem to the nontechnical observer hard to keep track of.

There's also a consolidation around XML and related technologies. XML is an enabling technology that lives only in the context of other programming and application development environments. Environments have emerged around XML and have defined their own space. For example, Web services have emerged as mechanisms for defining business services using XML technologies. Application frameworks have encompassed XML facilities and now offer new tools for application development at a more abstract level. In such a climate the question is where XML and the large community that has evolved around it will end up once the dust settles.

Some of the industry trends have led to consolidation of XML and complementary technologies. Again, the most obvious example is that of Web services. Many XML vendors have redefined themselves as Web service vendors. Take a look at the conferences and events out there. Nowadays you'd be hard-pressed to pick out an event that focuses on XML without offering Web services as its main theme. Or take the examples of events such as tradeshow and conferences. I was at Sun's JavaOne conference in March and there was ample coverage of XML, albeit under the umbrella of the Java programming environment. Similarly, .NET incorporates XML as a building block for its platform.

Is consolidation a good idea for the industry? Consolidation has been a key driver in the computing industry. Good ideas can't stand on their own without strong marketing and sales efforts behind them. Likewise, all the marketing and sales in the world can't help a really bad idea or technology. XML has formally transitioned from the early adopter stage into the mainstream market – the conservative majority. It now needs to live up to its expectations, which is going to take strong sales, services, and support. It's the ongoing consolidations that filter out the bad and allow the higher-quality products and ideas to survive and thrive by providing the infrastructure for continued growth.

The mere fact that large software companies are now competing for IT dollars on the basis of their use of XML – companies like Microsoft, Oracle, IBM, and BEA – demonstrates that it is virtually impossible to isolate XML as a key component in discussing products, applications, and solutions provided by those companies. However, XML is just a portion of those solutions. There is still a great need to focus on the development of XML and its related technologies in an isolated manner, because it is only through understanding the foundation that one can really fathom the reaches of the technology.

So is there still a need for independent environments that cover XML?

Absolutely. XML will continue to drive the integration between different environments. Integration and information flow basically require data formatting and transfer. That is exactly the functionality that XML brings to the table. XML tools will continue to evolve, and the XML developer will acquire newer skills. XML occupies very critical roles in the realms of business processes, messaging, component description and design, presentation, databases, and so on. This role can only grow over time. And concurrently, XML will continue to enable other paradigms. However, because XML is so integral to so many new processes and directions, expect that it will also be a key driver in consolidation.



AJIT @ SYS-CON.COM

AUTHOR BIO

Ajit Sagar is the founding editor and editor-in-chief of XML-J, as well as the J2EE editor of Java Developer's Journal.
 A lead architect with a B2B software solutions firm based in Dallas, Ajit is an expert in Java, Web, and XML technologies.

iWay Software

www.iwaysoftware.com/guaranteed



WRITTEN BY GRAHAM GLASS]

GridComputing andWebServices

Companies have long dreamed of assembling their enterprise systems from a collection of network building blocks. CORBA and DCOM, both early attempts at tackling this problem, never got very far in terms of adoption. But now that Web services have burst onto the scene, it looks like SOAP and WSDL will succeed in becoming the lingua franca for distributed computing, thereby providing the catalyst for a wholesale move toward service-oriented architectures (SOAs).

An SOA implements each part of a system as a Web service. Simple Web services provide low-level features such as access to a particular kind of data, and the ability to send out a piece of e-mail or perform a calculation. Higher-level Web services orchestrate lower-level services to provide more complex behaviors, resulting in a pyramid of increasingly specialized processing.

The big challenge that faces an SOA is how to organize these Web services into a coherent system, especially when they'll probably be implemented using a variety of platforms and languages. An SOA requires features like clustering, fault tolerance, load balancing, security, and management to work in a way that is independent of the services themselves, so application server technology isn't the answer. This is where grid computing, an up-and-coming concept, comes into play.

Grid-computing platforms operate in a manner similar to the national electricity grid, transparently connecting producers and consumers of services while shielding them from issues like failover, load balancing, and clustering. These platforms can link Web services from any vendor, and typically adopt a P2P architecture in order to operate on a large scale.

In the past, grid-computing platforms were positioned primarily as a way to make use of spare CPU cycles. But now that Web services have become popular, they're rapidly being viewed as a promising architecture for supporting large-scale SOAs. IBM's new CEO, Sam Palmisano, recently said, "The grid is the ultimate method whereby you can establish this seamless, open-standard computing model." And Irving Wladawsky-Berger, one of IBM's top strategists, believes that "grid computing is really the natural evolution of the Internet," so there's obviously a lot of cool stuff about to happen in this area.

The fun part, of course, is figuring out good ways to implement a grid-computing platform.

One solution to the service lookup problem is to use a federation of UDDI servers as a kind of DNS for Web services. When a program requires a particular kind of service, it asks a nearby UDDI node for the location of a Web service that implements a specific WSDL interface. If that UDDI node doesn't know the answer, it delegates to a more knowledgeable UDDI node, propagating the request until an answer is found. Answers to Web service lookups can be cached at local nodes to accelerate subsequent lookups for the same kind of service. If a particular service fails, a client could automatically rebind to an equivalent service by repeating the lookup, assuming that the original service is stateless.

There is of course a lot more to a grid-computing platform than just locating and binding to services. For example, how are events reliably distributed to services throughout the grid? How are transactions conducted? And how do you manage an application whose services can be widely distributed among a large number of nodes?

Another deeper question is whether a grid-computing platform should connect producers and consumers of data as well as services. For example, a grid could allow a program to upload a document for use by other programs, automatically replicating the data to ensure performance and reliability, and allowing it to be located easily using a query language. The similarities between sharing of services and sharing of data suggest there might be a way to unify these two features and solve them with a single architecture.

The promise of grid computing is to provide simple, universal access to services and data with the quality of service associated with utilities such as electricity and telephones.

However daunting the technical issues might seem, I'm convinced that commercial service-oriented grid-computing platforms will be available soon. The metaphor is too attractive to ignore, and with the advent of Web services, the timing is perfect. ☎

GRAHAM @ THEMINDELECTRIC.COM

AUTHOR BIO

Graham Glass is CEO, chief architect, and founder of The Mind Electric. He holds a BS in mathematics and computer science from the University of Southampton and an MS in computer science from the University of Texas at Dallas.

**SYS-CON
MEDIA**

135 Chestnut Ridge Rd., Montvale, NJ 07645
TELEPHONE: 201 802-3000 FAX: 201 782-9637

PUBLISHER, PRESIDENT, and CEO
Fuat A. Kircaali fuat@sys-con.com

BUSINESS DEVELOPMENT
VP, Business Development
Grisha Davida grisha@sys-con.com

ADVERTISING
Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com
VP, Sales & Marketing
Miles Silverman miles@sys-con.com
Advertising Director
Robyn Forma robyn@sys-con.com
Advertising Account Manager
Megan Ring megan@sys-con.com
Associate Sales Managers
Carrie Gebert carrie@sys-con.com
Kristin Kuhnle kristin@sys-con.com
Alisa Catalano alisa@sys-con.com
Leah Hittman leah@sys-con.com

EDITORIAL
Executive Editor
M'lou Pinkham mpinkham@sys-con.com
Managing Editor
Cheryl Van Sise cheryl@sys-con.com
Editor
Nancy Valentine nancy@sys-con.com
Associate Editors
Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com
Editorial Intern
Stephanie Williams stephanie@sys-con.com

PRODUCTION
VP, Production & Design
Jim Morgan jim@sys-con.com
Art Director
Alex Botero alex@sys-con.com
Associate Art Directors
Cathryn Burak cathyb@sys-con.com
Louis F. Cuffari louis@sys-con.com
Assistant Art Directors
Richard Silverberg richards@sys-con.com
Aarathi Venkataraman aarathi@sys-con.com
Tami Beatty tami@sys-con.com

SYS-CON EVENTS
VP, Events
Cathy Walters cathyw@sys-con.com
Conference Manager
Michael Lynch mike@sys-con.com
Sales Executives, Exhibits
Michael Pesick michael@sys-con.com
Richard Anderson richarda@sys-con.com

CUSTOMER RELATIONS / JDJ STORE
Manager, Customer Relations / JDJ Store
Anthony D. Spitzer tony@sys-con.com

WEB SERVICES
Webmaster
Robert Diamond robert@sys-con.com
Web Designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
Content Editor
Lin Goetz lin@sys-con.com

ACCOUNTING
Chief Financial Officer
Bruce Kanner bruce@sys-con.com
Assistant Controller
Judith Calnan judith@sys-con.com
Accounts Receivable
Jan Braidech jan@sys-con.com
Accounts Payable
Joan LaRose joan@sys-con.com
Accounting Clerk
Betty White betty@sys-con.com

**SYS-CON
MEDIA**

Happy 4th Birthday, XML!



Here are the last “cards” we opened as part of XML-J’s ongoing celebration of XML’s fourth birthday (actual date: February 4, 1998, the date of the W3C’s XML 1.0 specification). Our question was: What would be the best birthday present anyone could give to XML?

From CHARLES GOLDFARB, the soi-disant “Father of XML technology”:

What to give the girl who has everything for her fourth birthday? Like any beauty pageant queen, XML wants world peace. But though she’s pacified the world of universal interchange, we’re not letting her realize her potential for universal communication.

A generalized markup language is not just a common syntax for messaging or a verbose form of CSV for data exchange. It’s a standardized means of modeling and representing – and therefore sharing – the full universe of information in all its forms, including its semantics, structure, and rendition.

Unfortunately, our major uses of XML today barely even acknowledge that rich model. We merely adapt – even distort – the language to fit familiar parochial data models and processing paradigms. So instead of world peace, we have “data-centric” versus “document-centric” conflicts and schema language warfare.

These distortions and disturbances in which we are all so involved distract us from learning about what’s really important: XML’s powerful information model built on vital distinctions, like type from instance, structure from syntax.

Even our working vocabulary fails to honor them – “element type name” becomes “element name” and “element” becomes interchangeable with “tag” – concealing the very existence of the things we should be focused on.

So what should we give XML on her birthday? Any pageant queen would agree that giving is its own reward. If we gave more attention to understanding XML, we would be rewarded by being able to utilize its gifts to us more effectively.

—Charles Goldfarb

From AJIT SAGAR, XML-Js founding editor and editor in chief:

XML should be recognized as a critical part of enterprise applications. The baby has grown up. It is a bona fide adult now. And it’s spawning babies of its own – look at Web services, for example.

—Ajit Sagar

From JP MORGENTHAL, CTO of IKimbo and author of both *Enterprise Application Integration with XML & Java* and *Manager’s Guide to Distributed Environments*:

Id like to see it get a complete physical exam inclusive of a colonoscopy! I’d like to see a review of the level of real implementation of XML standards within organizations: which ones they’re using, which ones they’re not and why, and which ones users are waiting for before moving ahead with solutions.

Are XML standards the ploy of vendors with a few key customers, or are XML standards being widely adopted and integrated into systems today? Have the members of the W3C overestimated the need for these standards in the real world, are they ahead of the curve, or are they behind the curve and seeking desperately to catch up?

—JP Morgenthal

From SEAN McGRATH, founder/CTO of Propylon and member of XML-Js Editorial Advisory Board:

Ahaircut! The general shape is there, but some judicious trimming at this stage would create a better base for further growth.

—Sean McGrath

From JON BOSAK, Sun’s Distinguished Engineer, who organized and led the working group that created XML, subsequently serving for two years as chair of the XML Coordination Group of the WC3:

For XML’s birthday I’d like to see people stop futzing around with additions to the XML suite that would give Thomas Aquinas a headache and get down to the slow, dirty work of specifying the markup languages that we’ll actually be living with for the next few decades. SGML, the ancestor of XML, was designed to support machine-readable documents with 50-year life cycles.

—Jon Bosak

From ALEXANDER FALK, cofounder, president, CEO of Altova, Inc., and a member of the W3C Advisory Committee and the W3C XML Schema Working Group:

Happy Birthday, XML! May your future be even more successful than your first four years as a technology hatchling. And may you continue to avoid the traps and pitfalls that have caused your older brothers and sisters – SQL and HTML – to be led astray from the path of unity and become highly fragmented by vendors.

And may your parents accept that you live your own life and don’t follow in the footsteps of SGML, and that you are so successful because you embraced the data-centric side as well.

And may you continue to lead the Web to its full potential. And may you contribute to world peace by making information accessible to every human being on the planet.

Enough said, now let’s PARTY! Where’s that hot Spy Girl?...

—Alexander Falk

eXcelon

www.exln.com



Technologies in the Presentat

Written by Mark Spears

There are many different ways to implement XML-based technologies in applications. Understanding the history and background of technologies will aid in becoming an architect of them. Before there was HTML there was a metalanguage called SGML (Standard Generalized Markup Language). As a metalanguage, it was designed to be a language for describing other markup languages. One such markup language created from SGML was HTML. I personally have never programmed in SGML. From what I've heard and read it's a very involved and costly resource to use, but built to last. Either way, it seems that SGML wasn't a language that ordinary developers could easily pick up and use.

This article is intended to help developers derive benefits from using XML technologies in the presentation layer. The samples shown were developed using MSXML 3.0 and IIS 5.0. They've been tested on Internet Explorer 5.0 and UP Browser 4.1. The UP 4.1 simulator can be downloaded from <http://developer.openwave.com>.

XML, written as a lighter version of SGML, was designed as a meta-language that could be used on a wider scale. It's based on the profile of SGML, but not to the same complexity. Where HTML uses SGML, XML is just SGML on a smaller scale. From XML we've seen languages such as XSL (which can be divided into XSLT and XSL-FO), SVG (Scalable Vector Graphics), WML (Wireless Markup Language), and XHTML (Extensible Hypertext Markup Language). I commonly refer to these as the X-based languages. There have been implementations of these X-based languages in the Java and COM world that include considerable compliance with the W3C recommendation with custom extensions.

HTML became widely used very quickly. During its popularity, a lot of nonprogrammers began to create Web pages. A good portion of these nonprogrammers probably had little idea where HTML came from and how to properly write valid HTML. A lot of browsers have been programmed to be very forgiving of malformed HTML. I've heard specula-

tion into WML. Reusable components improve the efficiency of any application.

One issue with XSLT is that what's contained in it must conform to XML (because XSLT is still an XML document). Now consider HTML. The more recent versions aren't XML conformant. So how do we create an XSLT document and in it contain HTML? The answer is simple: XHTML. XHTML was written from XML and is therefore XML compliant. XHTML ensures only that the HTML follows good markup techniques. For example, in HTML "
" elements don't have to be closed. In XHTML we'd show the same element as "

" or as an empty element "
". Most developers using X-based technologies are more likely to use XHTML, even where not explicitly necessary.

Design Considerations

Microsoft has done a wonderful job of keeping bleeding-edge technology for XSLT in the hands of developers while XSL was in working

ion Layer

Adding intelligence to transformations in a multitier environment

tion that an estimated 80% of the code base in Internet Explorer or Netscape Navigator is to render malformed HTML.

With the ever-growing popularity of XML over the last couple of years, many parsers have evolved in the marketplace for the Java and COM environments. One of the first and most popular for the Web has been XSLT, a transformation language that enables transformation of an input XML document into another XML-based output structure such as XHTML. This is a remarkable development for the Web and provides substantial benefits over HTML. HTML includes presentation and data in one document. With XSLT, programmers can use an XML datastore that comes from a database or is created as a standalone document and apply a stylesheet to define how to display the data.

This separation of presentation and data documents allows one XML data source document the ability to have different XSLT documents that can be applied to that one XML document. One XSLT document may transform the data source into XHTML. Another may transform the same data source

into WML. Reusable components improve the efficiency of any application. Hence, the namespace of earlier versions of the Microsoft parser using XSL contained the letters *WD*, for working draft, as follows:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
```

The W3C eventually decided to split XSL into two areas – XSLT for transformations and XSL-FO for formatting objects. Microsoft's new namespace for XSLT supports virtually all of the recommendation from the W3C and no longer includes the letters *WD*, as shown below:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Using the correct namespace in XSLT is crucial to the XSL stylesheet's performing properly. It's probably the No. 1 response to many new users' first technical question on how to troubleshoot code that doesn't perform properly.

A word about Microsoft's MSXML parser: when they released MSXML 3.0, they provided the option to install it in side-by-side mode or replace mode. The former would install the new parser like any normal installation, but replace mode would install the new parser and trick the computer into redirecting any older versions of the parser being called to the newer one. The average user may not find the mixed mode harmful (incidentally, it appears that IE 6 does install MSXML 3.0 in replace mode), but if you're a developer, it isn't recommended. It counteracts normal coding practices of dll instantiation and usage.

Transformations take place on either the server or the client. The code is essentially the same; the major difference is where it's executed. Following are some considerations for using client versus server transformations for your Web site:

- Do the clients have a compatible parser installed?
- Can we force the clients to install the proper parser?
- How much time will it take to download data islands to the client?
- What size are the data islands that could be downloaded to the client?
- How will our server resources be affected for server transformations?

The full source code for the following examples is available at www.sys-con.com/xml/sourcec.cfm.

Client-Side Transformations

Transforming XML data on the client is obviously better for freeing up server resources. The code to transform the data is rather simple in nature, as seen in Listing 1. There are two main options to use for client-side transformations. One is to embed data islands (DataIslandsEmbedded.htm); the other is to link them (DataIslandsLinked.htm). These are the same as the choices for embedding or linking code for client-side scripting. Also, if you view the source on the HTML page, you'll notice that the page displayed in the browser is different from the source. This is most simply illustrated in DataIslandsLinked.htm, which is included with the source code of this article. One of the major obstacles to successful client-side transformations occurs when the client's browser doesn't have the correct version of the XML parser. If the client isn't running the proper version, the transformation will fail. Ensuring that the proper XML parser has been installed is easily achieved through a sniffer. Many MSXML sniffers are written in JavaScript and are easily edited or assimilated into your own custom sniffers.

Server-Side Transformations

If the Web site isn't deemed appropriate for client-side transformations, you'll have to make some minor adjustments to your code. The basic coding technique behind server-side transformations, as demonstrated in Listing 2 and ServerTransformation.asp, is still very simple. This sample loads persisted XML documents into the DomDocument. You just have to instantiate



FIGURE 1 Multitime.asp in HTML browser

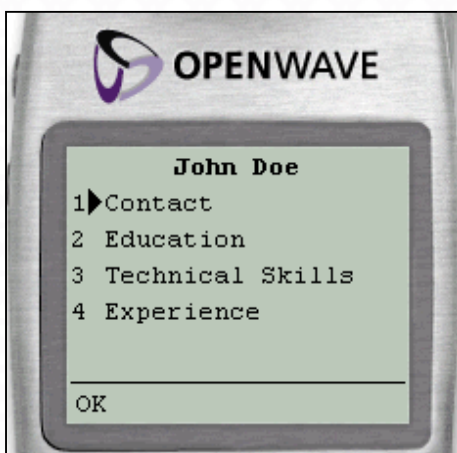


FIGURE 2 Multitime.asp in WML browser

the MSXML2.DomDocument for the XML and XSL documents, load the XML document for each, and then catch the result of the transformation. You must remember to use the Server.MapPath command or hard-code the full path to an XML document. If you view the results in the browser, the source code looks exactly as it appears in the browser window. If your Web site is going to include data from some RDBMS such as SQL Server or Oracle, the XML document server transformation process is slightly changed. The following scenario is likely: in ServerTransformation-2.asp, an XML recordset returned from a database is simulated with the SimulateXMLRecordset function. The code simply loads an XML document. But remember, we're acting as if that function returns a string of XML data acquired from a database. Note that to load the string of XML received from SimulateXMLRecordset, the command in MSXML isn't "load", but "loadXML".

(One thing missing from the previous listings is error handling. I'll cover that later in the article. The error-handling code for Listing 2 is in the downloadable source code.)

I hope that by now you're seeing the benefits of separating the presentation from the data. We could have saved routines to bring back data from the RDBMS as XML. While this data isn't tied to any specific visualization, that isn't the case with the traditional use of HTML.

With this separation we could transform the same XML data source in multiple ways using XSL.

Picture a reporting model. On many occasions reports require the same underlying recordset. You could simply provide multiple XSLT documents to gain greater perspective on the data without having to make another trip to the database. To support even further reusability, let's think about the revolutionary WML language. In the United States our phones are no rival for those in Europe or Japan. We've barely begun to scratch the surface of m-commerce. Multitime.asp will preview in HTML or WML 1.1 browsers (see Figures 1 and 2).

Using ASP, we check the MIME type of the browser requesting the page and then apply the appropriate XSLT to use in a transformation. Now we have one source XML document and multiple XSLT documents to display the data. Note that the XSLT documents are only slightly different. All the XSLT documents so far have simply been a body transformation. The ones for Multitime.asp return a complete page from top to bottom.

WAP-enabling your Web site may or may not be in the scope of your application. It's definitely a technology to keep an eye on. After all, now we're using the same source documents. WAP is just another interface to your Web site for a user. One issue to consider is *how* to integrate different types of MIMEs into your architecture. Some sites simply sniff the client and do a redirect to another directory containing all the decks for WAP. That doesn't provide a whole lot of immediate reuse, but it does provide a safe initial implementation to begin with. A more advanced way to implement a Web site that provides different MIME and media types would be a sniff of the client and a more advanced menu system that understood what the data source and XSLT documents were for a given and allowable media type.

Macromedia

www.macromedia.com/downloads

I know, I know. This is beginning to sound a lot like an enterprise application integration – and it is! I can't think of a better tool to use for EAI than XML. Remember, if you begin integrating your applications with growth, expansion, and acquisitions in mind, it should be easier to do. Also, it's important to diagram and lay out a path for the application. Driving without a road map usually gets you lost. You'll probably end up where you planned, but it may take a lot longer to get there. It's the same when writing software without a plan.

Tools for Developing

There are many tools to use for creating, validating, and parsing XML technologies and applications. Generally, hand-coding in a regular text editor is discouraged because a normal text editor isn't XML savvy and won't understand such issues as parse errors. For most XML developers a product such as XML Spy from Altova will more than suffice as an integrated development environment. The product, by default, uses the MSXML version 3.0 parser, but can be pointed to different XSLT parsers. In its newer releases the product provides full support for the XSL-FO standard using the Apache FOP parser add-in, which is easily downloaded and installed from the Altova Web site. XML Spy is a general XML technologies tool. The integrated development allows you to create XML, SVG, XSLT, XSL-FO, and WML documents, to name just a few. The DTD and schema support is excellent in that you can autogenerate a DTD or schema from an XML document that you may have been writing. Also, if there's a DTD or schema associated with an XML document, you can autogenerate sample data to populate the document with. It allows you to view documents as text, graphically, or in a browser. With the click of a button you can perform an XSLT or XSL-FO transformation and view the result in a new window. The product is available for a free 30-day trial download at www.xmlspy.com.

A great tool for modeling and developing WAP sites, the <alt> Web SDK: WAP Edition, is now in version 3.0. This tool, which is extremely graphical and easy (and available for a free 30-day download from www.alt-mobile.com), was developed specifically to be smart for WAP technologies and is fully WAP 1.1 compliant. Developers just beginning with WAP will get a greater understanding of the Wireless Markup Language from this tool. It excels as an educational tool and in creating static WAP sites.

One of the problems with WAP is that the browser on the mobile device, when it calls to a Web site, is really calling wirelessly to the "carrier," which in most cases is a company such as Sprint, Cingular, or Verizon. When the carrier receives the wireless request from the mobile phone, it will have an HTTP conversation with the requested Web site, then return the results wirelessly to the mobile browser (see Figure 3).

Understandably, the various browsers have implemented WAP with their own custom extensions to provide enhanced support to the device. This sets up a roadblock. If you create a WAP site that uses custom extensions for a browser, the phones from different carriers may not interpret or understand the other browser's custom extensions. This is referred to as the triplet: device dimensions, browser capabilities, and carrier support for the WAP DTD. There are literally over a hundred possible combinations. The <alt> Web SDK: WAP Edition is especially helpful in that it understands and supports the custom extensions from the various browsers and will validate the WAP site. Of the reports that can be drawn, one will tell you which browsers on mobile phones going through the carriers will be viewable on your WAP site.

Techniques

In this section we'll explore specific techniques for using XML and XSLT technologies. Foremost in XML application development is a strong

Document Type Declaration. DTDs are used to define the elements allowed in an XML document, in what order and how many times they can appear, and if the node values are parsable or not. Our document, Resume.xml, widely used throughout this article, utilizes a DTD called CVML that you can acquire for free from XML.org, which has a repository of DTDs that can be perused and used. Defining standard documents to be used across industries is a large piece of the ebXML movement. In fact, XML.org is an OASIS initiative. XML.org's mission, according to general manager Leo Kraunelis, is "to accelerate the global utilization and adoption of XML in industry by providing an open and public industry portal that brings together members of the XML community at large."

While there may be no requirement to use a specific DTD internal to your business, it will greatly aid the ebXML movement to begin using published DTDs or to publish your own. In the foreseeable future more companies exchanging business data will require vendors or suppliers to give and receive data in a specific DTD format. In the final phases of a project it's desirable to generate DTDs that don't exist for XML data documents so as to reduce the risk of the associated XSLT document's finding unexpected elements in the document that would increase the chance of errors in the transformation. Resume.xml has a DOCTYPE declaration that refer-

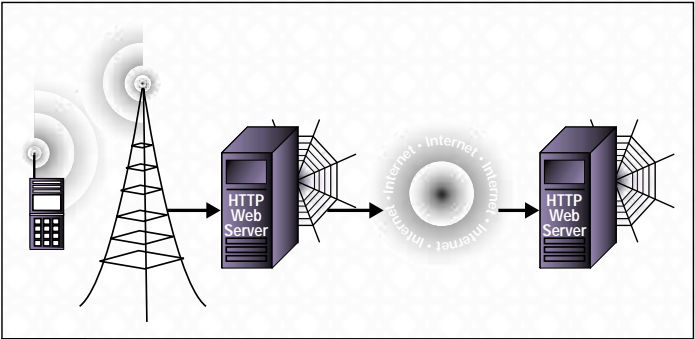


FIGURE 3 How wireless browsers access a Web site

| POPULAR CODING TECHNIQUES | VBSSCRIPT EXAMPLE | XSLT EXAMPLE |
|-----------------------------|---|---|
| For...each statements | For each x in y ... Next | <xsl:for-each select="x"> ... </xsl:for-each> |
| If statements | If x then ... End if | <xsl:if test="x"> ... </xsl:if> |
| Select case statements | Select Case x Case y: ... Case z: ... Case else: ... End select | <xsl:choose> <xsl:when test="y">...</xsl:when> <xsl:when test="z">...</xsl:when> <xsl:otherwise>...</xsl:otherwise> </xsl:choose> |
| Variables | Dim x x = y | <xsl:variable name="x" select="y"/> |
| Subroutines with parameters | Sub MySub (x) ... End sub | <xsl:template name="MySub"> <xsl:param name="x"/> </xsl:template> |
| Calling a subroutine | MySub (y) | <xsl:call-template name="MySub"> <xsl:with param name="x" select="y" /> </xsl:call-template> |

TABLE 1 Comparing XSLT functionality

THE INSIDER INTELLIGENCE YOU NEED...

TO KEEP AHEAD OF THE CURVE

FREE E-Newsletters
SIGN UP TODAY!

Go to www.SYS-CON.com

The most innovative products, new releases, interviews, industry developments, and plenty of solid *i*-technology news can be found in SYS-CON Media's Industry Newsletters. Targeted to meet your professional needs, each e-mail is informative, insightful, and to the point. They're free, and your subscription is just a mouse-click away at www.sys-con.com.

**SELECT THE INDUSTRY NEWSLETTERS THAT MATCH YOUR NEEDS!
CHOOSE ONE – OR TRY THEM ALL!**



Don't Delay!
Subscribe
for **FREE!**

at www.sys-con.com

Exclusively from the World's Leading *i*-Technology Publisher



ences CVML.dtd. Once an XML document has the DOCTYPE declaration, it is then tied into the language structure of the associated DTD.

In our client-side transformation examples, `DataIslandsLinked.htm` and `DataIslandsEmbedded.htm`, we barely scratched the surface of data islands. They have many alternative uses and benefits. One of the most important is to maintain a form's state. In the past, Web sites have commonly used hidden input boxes to keep a form's prior state. State can also be maintained by sending a data island down to the HTML page and accessing the DOM with a client-side script. Virtually any data can be persisted through a client-side data island.

Understandably, error handling is a normal part of the programmer's coding cycle. Some of the more common runtime errors: the XSLT processor encountered an error in an unescaped character from a node value or couldn't find the node that the template's match attribute required, or the XML document isn't valid according to the DTD. Error handling in MSXML is made easy with the exposed properties of the "parseError" property, available from the "DomDocument" class, which exposes within itself properties such as error code, reason, line number, and line position. Viewing the DomDocument class in the Object Browser will provide a greater understanding of what's available for this component. The VBScript procedure called "errHandler" in `ServerTransformation.asp` displays a sample showing how to handle parse errors. This function could easily be packaged neatly in an include file for optimal reuse.

Stylesheet Functionality

If you don't know XSLT intimately, this section explains more about the functionality available within a stylesheet and hopefully will help you draw parallels to some popular technologies (see Table 1). When you were in school and had to learn a foreign language, the teacher may have started at a reference point, something you knew, like English. The teacher would then take different pieces of the English language (verb conjugation, punctuation, etc.) and show you their counterparts in the foreign language. Many spoken languages have close to the same concepts. Programming languages are no different. For example, there are many similarities between JavaScript and VBScript. The same is true for XSLT. XSLT draws on many of the same powerful concepts as other programming or scripting languages. At its most fundamental level, XSLT is about transformations. The root of an XSLT document is called the stylesheet. Within the stylesheet element are templates. Templates have either a reference (or match) to an element in the source XML document or a name they can be called by. More common to programmers on the Microsoft platform is the use of named templates. When a named template is "called," it will push out whatever is contained in it. In the templates you can include many of the "control-of-flow" features that most languages have. Some of these features are for-each loops, if statements, and case statements.

Variables may have either a global scope for the stylesheet or a local scope for the template. With this understanding, each template may be thought of as a subroutine and the stylesheet as a class. The subroutines can be called with parameters. The template may have default parameter values if none are passed in. Stylesheets can "include" files, similarly to ASP scripting.

Stylesheets may also inherit or "import" another stylesheet's interface. Hopefully, a lot of the terminologies in the previous few sentences are starting to sound familiar. A good resource for XSLT terminology is www.mulberrytech.com/quickref/XSLTquickref.pdf. It would be helpful to keep this easily printable page near your workstation.

Taking a closer look at a stripped version of `Resume.xml` in Listing 3, you can see that the templates are broken down by logical sections. Breaking

down the templates this way can help you understand the kinds of transformations the various templates perform in the stylesheet. The first template, a "match" template, shows us where to start transforming the XML data source from – a requirement for a successful stylesheet. If you want to process the entire data document, the first template should match the "root" denoted by "/". In this case it tells the XSLT processor to look for our root element in `Resume.xml`, which is the "<CVML>" tag. The template that matches the root is acting as a wrapper that calls other templates by name. This is a simple model that can be used in many other stylesheets.

Within the templates a fundamental concept that can frustrate beginning or advanced developers is how to use XPath. XPath is like driving directions. While the stylesheet is navigating the DOM of the XML document, it's very easy to get lost. Taking a closer look, XPath looks somewhat similar to DOS, as seen in the excerpt of `Resume.xml` in Listing 4. The listing shows a for-each loop in XSLT. The select attribute of the for-each loop is navigating the DOM of the source document. DOS is a way of navigating through the operating system, whereas XPath navigates an XML document. Maybe there's an error in your transformation and some conditional processing, like a for-each loop, isn't outputting to the result tree. The answer usually lies in XPath. It's likely that the XSLT processor was given incorrect driving directions by whoever coded the stylesheet. The driving directions for XPath usually originate with the initial template that matches the root node of the XML data document. In the sample of a for-each loop, the XPath of the nodes to process is in the select attribute. A quick review of the calling templates and where they drove the processor will help to identify the correct revision to the select attribute of the for-each loop and output the results desired.

A sign of a maturing XSLT developer is the use of externalized templates. External templates can be divided into two groups, included and imported. These are a must for ultimate reuse and optimization of workflow. `Resume.xml` "includes" a template called "Format.xml". This external template provides scripting functionality to our already versatile stylesheets. The calling document includes the external stylesheet by writing:

```
<xsl:include href='Format.xml' />
```

where `Format.xml` could be the name of any valid stylesheet. Using the MSXML namespace in the stylesheet enables the templates to use the power of either VBScript or JavaScript. Its name describes how this external stylesheet is used. The function "formatPhoneNumber" simply takes in the phone number from the XML document and outputs a neatly formatted one. When a stylesheet is included, it's used as if the templates and scripts were written directly in the calling document. In the template named "Header" in `Resume.xml`, the code is written as follows:

```
<xsl:value-of select='user:formatPhoneNumber(string($Phone))' />
```

This is simply saying that the value of this tag is performed in the "user:" namespace, a strict requirement of the MSXML namespace. The functions cannot be performed in the namespace that holds them, in this case "MSXML:". The user namespace calls the "formatPhoneNumber" function and passes it the string value of the variable "\$Phone". Variables are easy to use and spot as they have the "\$" sign preceding them. It's important to note that a node is neither numeric nor string. VBScript interprets everything as a variant, but it doesn't even understand a node value. It's safer to use the XSLT "string()" function to ensure that the script can understand what it's receiving.

The other way to use external templates is to import them. Importing templates will allow inheritance within your stylesheet templates. Importing a stylesheet looks a lot like including a stylesheet. The code is “<xsl:import href=“/”>” where href gives the location and name of the stylesheet to import. The former have lower priority than templates in the calling stylesheet. The imported stylesheet’s templates can be used to override the calling stylesheet’s templates by setting the priority attribute of the template or having the calling stylesheet use “<xsl:apply-imports/>”.

One last technique for the XSL developer is the use of parameters in templates. Again, this is comparable to a subroutine with parameters. To use parameters, after the template is begun, type “<xsl:param name=“>Default Value</xsl:param>”. The name attribute will in essence become and be used exactly like a variable. The parameter is referenced with a “\$” preceding the name attribute. The “Default Value”, if one is necessary, would be placed as the node value of the parameter element. Calling a template with parameters is simple. The calling template would simply write the following:

```
<xsl:call-template name=“”>
  <xsl:with-param name=“”>Value to give parameter
</xsl:with-param>
</xsl:call-template>
```

LISTING 1 Client transformation code sample

```
<HTML>
<HEAD>
<TITLE>Data Islands Sample Transformation</TITLE>
</HEAD>
<BODY>
<div id="idTransform"></div>

<!--
  These data islands are embedded
-->
<xml id="idData" src="XMLData.xml"></xml>
<xml id="idStyle" src="XMLTransform.xml"></xml>

<script type="text/vbscript">

  sub window_onload()

    idTransform.innerHTML = idData.transformNode
      (idStyle.xmlDocument)

  end sub

</script>
</BODY>
</HTML>
```

LISTING 2 Server transformation code sample

```
<%
  function getBody

    dim docXML, docXSL

    'Create the DomDocument and set async property
    set docXML = server.CreateObject
      ("MSXML2.DomDocument")
    docXML.async = false
    'load the xml file
    docXML.load server.MapPath("xml/Resume.xml")

    'Create the DomDocument and set async property
    set docXSL = Server.CreateObject
      ("MSXML2.DomDocument")
    docXSL.async = false
    'load the xsl file
    docXSL.load server.MapPath("xsl/Resume.xsl")
```

• • •

Using XML with XSLT gives great control to the developer. Separation of the presentation from the data allows a developer to give different presentations of the same data, including support for multiple MIME types. The great reusability of these technologies can be adapted to many existing structures and can only enhance a standard Web site. Newer versions of Oracle and Microsoft SQL Server provide enhanced support for these technologies. Another great benefit for this duo is the loss of “provider dependency” in recordsets. An XML data document is truly just a text document. The only requirement for using it is to have a parser compliant with the W3C recommendation. Also, unless the stylesheet is using custom extensions, the document is reusable in another W3C-compliant parser environment, that is, XML developers can move, for example, between the IIS and Apache servers, giving a developer great market advantage and possible exposure to environments that he or she may never have had the opportunity to work in before. The combination of these technologies will allow companies and systems to reach more clients on different devices and browser types, allowing for a broader means of distributing information and generating sales. ☛

AUTHOR BIO

Mark Spears is a leading XML expert and independent consultant in the Southern California area.

M A R K @ M A R K S P E A R S . N E T

```
'perform the transformation
getBody = docXML.transformNode(docXSL)

end function

%>
<HTML>
<HEAD>
<title>Server Side Transformation</title>
</HEAD>
<BODY>

<% =getBody%>

</BODY>
</HTML>
```

LISTING 3 Stripped version of Resume.xsl

```
<xsl:stylesheet version="1.0">
  <xsl:template match="/">
    <xsl:call-template name="Header"/>
    <xsl:call-template name="TechSkills"/>
    <xsl:call-template name="Education"/>
    <xsl:call-template name="Experience"/>
  </xsl:template>
  <xsl:template name="Header"></xsl:template>
  <xsl:template name="TechSkills"></xsl:template>
  <xsl:template name="Education"></xsl:template>
  <xsl:template name="Experience"></xsl:template>
</xsl:stylesheet>
```

LISTING 4 XPath sample from Resume.xsl

```
<!--Loop through skills-->
<xsl:for-each select="CVML/SKILLS/SKILL">
<!--
  The select attribute above
  references a node set from the DOM of the source
  document. The code below outputs specific
  child nodes of the above select attribute.
-->
<tr>
  <td><xsl:value-of select="skill_name"/></td>
  <td><xsl:value-of select="extra"/></td>
</tr>
</xsl:for-each>
```

DOWNLOAD THE CODE @
www.sys-con.com/xml



Full-featured data grids your users will enjoy

Create Sortable HTML Tables Using XSLT

In developing Web applications, the most common method for displaying multiple rows of data on a Web page has been to use the HTML `<TABLE>` element. Presenting the user with a static table of data might be fine in some instances, but if you're trying to develop a more user-friendly application, there are a few features missing that you'd find in a typical rich-client data grid.

If a user wants to change the order of the rows, another call to the server must be made to get the data from a database in a different order, and the HTML table with data must be sent to the browser again, which isn't very efficient. Another problem is that if there are many rows of data, the user must scroll the window, thus losing sight of the column headings.

If you're developing a spreadsheet application or an application in which the user is trying to find a specific row of data quickly, then the static table isn't very user-friendly. The addition of an XML parser to a Web browser, such as Microsoft's XML3 parser in Internet Explorer 6.0, allows developers to leverage the power of XSLT transformations in the browser to quickly re-sort and display rows of data without going back to the server. I'll show you how to make sortable tables using this technique, as well as a few other tricks to build full-featured data grids.

Browser-Side Transformations

The trick to browser-side transformations is to get both the XML data and XSLT presentation logic into the browser where it can be transformed via JavaScript. By using XML data islands, you can cache both the data and the presentation on the client machine. Since XSLT is well-formed XML, we can put the presentation logic on the client by placing the stylesheet between `<XML>` tags. Once that's done, the structure can be accessed as an XML DOM object in JavaScript by using the `XMLDocument` property of the tag ID. Listing 1 shows an example reference as:

```
document.all.tableXSL.XMLDocument
```

To do a transformation you need two XML data islands: one that contains the data, another that contains the XSLT that will create an HTML table from the data. You can then use JavaScript to access both XML DOM objects and do the transformation with the `transformNode` method.

```
newHTML = oXMLDOM  
.transformNode(oXSLDOM)
```

The HTML that results from the transform can be placed on a page by using the `innerHTML` property of a `DIV` tag.

In Figure 1 there are actually two tables that make up the grid. The top one contains the column headings; the bottom one, the rows of data. The column headings are in a separate table so just the rows of data scroll and the column headings are always visible at the top of the grid – a nice feature if you have lots of rows and your columns aren't self-explanatory. The table with the data rows is placed within a `<DIV>` tag to allow scrolling of the rows. Every time the table is re-sorted, we transform the XML with the XSLT and change the `innerHTML` of the `<DIV>` to the newly sorted table data. Before I explain the actual sorting algorithm, let's look at how the table of data is actually created using XSLT.

Creating Tables with XSLT

The algorithm for creating HTML tables with XSLT is fairly simple: for every row of data in the XML, create one row in the HTML table. If you look at Listing 2, you'll see that the stylesheet starts with the obligatory document element `<xsl:stylesheet>` tag. The result of the transformation will be HTML, so you need the `<xsl:output>` tag with the method attribute set to "html". Then add the `<TABLE>` tag. Now use the `<xsl:for-`

`each>` tag to create a loop that will select each row of XML data.

```
<xsl:for-each select="//ROW">
```

We want to sort the rows in the XML by some default field when the table is first displayed.

```
<xsl:sort select="@/AcctNo" order=  
"ascending" datatype="number"/>
```

The value of the `select` attribute is a valid XPath statement that tells the parser how to sort the data. In this case we're sorting on an attribute (as indicated by the "@" character) within the current `ROW` element named "AcctNo". The valid values for the `order` attribute are "ascending" or "descending." The `datatype` attribute tells the parser to do either a number or text sort, which can be tricky if you want to sort on something like date, but I'll touch on this later.

Now create cells for each field you want to display in your table and add `<xsl:value-of>` tags with the proper XPath select statement to plug in the values from the XML. The XSLT in Listing 2 contains the only presentation for the HTML table with the data in it. The rest of the page, including the column headers, is built with the HTML in Listing 3.

Sorting

After the table is displayed, we want to enable the user to click on a column heading and have the table re-sort on that column. The sorting of the rows is handled by the `<xsl:sort/>` tag within the XSLT. When a user clicks on a column heading, we want to (1) change the value of the `select` attribute to the attribute in the XML (see Listing 4) that matches this column, and (2) set the `order` attribute to "ascending". If the user clicks on the column again, we need to change the `order` attribute to "descending".

After the changes are made to the sort tag, we can retransform the XSLT style-

AUTHOR BIO

Sean McMullan is a Web application developer with ALLTEL Information Services, Inc. He received his bachelor's degree in computer science from Siena College.

sheet with the data in the XML data island and insert the new HTML table into the scrollable <DIV>. This all happens within the browser – there's no need to go to the server every time the user changes the sort order. To change the attributes on the sort tag, we'll use the API for the XML DOM to select the <xsl:sort/> using XPath and then update the attributes. In Listing 3 notice that the onClick event on the table headers calls the sort function in Listing 1, passing it the name of the attribute and datatype to sort on. The sort function uses the selectSingleNode method on the XML DOM object to get a reference to the select attribute within the xsl:sort node.

```
var objSelect = XSLIsland
    .selectSingleNode
    ("//xsl:sort/@select")
```

The select attribute is then set to the new attribute from the XML we want to sort on. Next we check the current sort order and change it to the opposite order. Finally, we set the datatype attribute.

Another attribute you can add to the xsl:sort that isn't shown in the source is the "case-order" attribute, which can be set to either "upper-first" or "lower-first". If you don't include this attribute, the default will be the former – for example, "VanStueben" would come before "vanStueben."

The XSLT engine can sort by text or number by setting the datatype attribute. If all your columns have strings or numbers, you won't run into any problems. One of the biggest shortfalls is that you can't sort by date very easily, which is often a requirement of a data grid. Like most problems, there are a couple of workarounds. The first requires you to change the way you display dates. Most applications designed for the U.S. market display their dates in mm/dd/yyyy format – if you try to sort columns by text or number, 08/01/2001 will come before 08/31/2000. But if you change the date format to most significant to least significant, that is, yyyy/mm/dd, your date sorting will work fine as long as you display months and days that are less than 10 as 01, 02, and so on.

Another trick is to have an attribute within your XML that has the date in the yyyy/mm/dd format that isn't displayed in the table. Sort on it, but display the data in the traditional mm/dd/yyyy. So your XML might look like:

```
<row birthday="08/15/1954" sortdate="1954/08/15"/>
```

The column would show the birthday, but the onClick event in the TH tag would look like:

| LastName | FirstName | Phone | Birthday | Account | City | State |
|------------|-----------|--------------|------------|---------|--------|-------|
| Oeller | Andrew | 800-555-1234 | 10/02/1952 | 123456 | Albany | NY |
| Lucich | Alie | 800-123-5643 | 08/05/1954 | 637663 | Miami | FL |
| MacCharles | William | 888-555-3213 | 02/02/1946 | 123456 | Denver | CO |
| VanStueben | Robert | 888-555-7008 | 10/28/1956 | 831233 | Albany | NY |

FIGURE 1 Scrollable HTML Table

```
sort('./@sortdate','number')
```

Selecting Rows

You might also want to allow your users to select a row within the grid and have all the data from that row passed to another Web page. Most grid controls in rich-client applications allow the user to press the up and down arrows to highlight a row and then press the enter key or double-click the mouse to select the row. Listing 1 shows how the arrows are captured from the keyboard and how we keep the current selected row highlighted with the keyCheck and selectRow functions. In the sample code, for a selected row, we want to return all the data to the server as an XML string for processing. We'll do this by taking the value from the hidden key column and using XPath to select the data row from the XML data island and return it to the server. If you look at the XSLT in Listing 2, you'll see that the first column of the table contains an ID that isn't displayed to the user. When a user chooses a row, we'll get the ID by grabbing the innerText of this table cell and passing it to the selectSingleNode method on the data XML DOM object to get a reference to the node the user selected. We can then send the XML string of that node to the server by placing it into a FORM element and submitting it to the server via the POST method. In the function selectCurrentRow (see Listing 1) I just alert the XML string, so I'll leave it to you to create the FORM.

Where to Go from Here

There are some other features you might consider adding to make the table more like a spreadsheet. You could add an indicator in the column heading that shows which column the table is sorted by and in what order. Modify the sort function so that it shows the proper icon within the TH tag.

You could also add drag-and-drop features to the table headers so users could change the order of the column headers, similar to a spreadsheet. When the user drags one column and drops it onto another, you could manipulate the

order of the column nodes in the XSLT and then retransform it with the XML. Remove the dragged node from the XSLT by using the removeChild method, then insert it before the dropped TR tag by using the insertBefore method. For example, to move the AcctNo column before the LastName column execute:

```
dragNode = XSLIsland.selectSingleNode
    ("//TD[@id='accountCol']")
dropNode = XSLIsland.selectSingleNode
    ("//TD[@id='lastnameCol']")
trNode = XSLIsland.selectSingleNode
    ("//TR[@id='trnode']")
trNode.insertBefore
    (trNode.removeChild
    (dragNode),dropNode)
```

Now do the transformation again and replace the old HTML table with the new HTML table.

These are just a few ideas to take your HTML tables to the next level. Because of the power of XSLT transformations and the MSXML3 parser, you can really build full-featured data grids that your users will enjoy. ☛

SEANFMC MULLAN @ HOTMAIL.COM

LISTING 1 TableUtils.js

```
var currentRow = null;
var currentSortBy = null;

function transformTableData(){
    oXMLDOM = document.all.XMLdata.XMLDocument;
    oXSLDOM = document.all.tableXSL.XMLDocument;
    newHTML = oXMLDOM.transformNode(oXSLDOM);
    document.all.tableBody.innerHTML = newHTML;
}

function sort(sortBy,dataType){
    XSLIsland = document.all.tableXSL.XMLDocument;
    var objSelect =

    XSLIsland.selectSingleNode("//xsl:sort/@select");

    objSelect.nodeValue = sortBy;

    var objOrder =
    XSLIsland.selectSingleNode("//xsl:sort/@order");
    if (currentSortBy == sortBy){
        if (objOrder.nodeValue == "ascending") {
            objOrder.nodeValue = "descending";
        } else {
            objOrder.nodeValue = "ascending";
        }
    }
}
```



```

    } else {
        objOrder.nodeValue = "ascending";
    }

    currentSortBy = sortBy;
    var objDataType =
        XSLIsland.selectSingleNode("//xsl:sort/@datatype");
    objDataType.nodeValue = dataType;

    transformTableData();
    selectRow(document.getElementsByTagName('TR').item(1));
}

function keyCheck(keyCode){
    switch (keyCode)
    {
        //enter
        case 13:
            selectCurrentRow();
            break;
        //up arrow
        case 38:
            try{
                selectRow(currentRow.previousSibling);
            } catch(e){}
            break;
        //down arrow
        case 40:
            try{
                selectRow(currentRow.nextSibling);
            } catch(e){}
            break;
    }
}

function selectRow(row){
    if(row == null){
        try{
            currentRow.style.backgroundColor = "#ccffff";
        } catch(e){}
        try{
            currentRow.nextSibling.style.backgroundColor = "#ffffff";
        } catch(e){}
        try{
            currentRow.previousSibling.style.backgroundColor =
                "#ffffff";
        } catch(e){}
    } else {
        try{
            row.style.backgroundColor = "#ccffff";
        } catch(e){}
        try{
            currentRow.style.backgroundColor = "#ffffff";
        } catch(e){}
        parent.currentRow = row;
        currentRow = row;
    }
}

function selectCurrentRow(){
    XMLIsland = document.all.XMLdata.XMLDocument;
    alert("You selected " + XMLIsland.selectSingleNode(
        ("//row[@id='" + currentRow.childNodes(0)
        .innerText + "']").xml);
}

```

LISTING 2 Table.xsl

```

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="yes"/>
<xsl:template match="/">

<TABLE width="775" border="1" cellspacing="0"
bgcolor="white">
<xsl:for-each select="//row">
<xsl:sort select="./@lastname" order="ascending"
datatype="text" case-order="lower-first"/>
<TR id="trnode">
<xsl:attribute name="onclick">selectRow(this)
</xsl:attribute>
<xsl:attribute name="ondblclick">
selectRow(this);selectCurrentRow()
</xsl:attribute>
<TD style="display:none">
<xsl:value-of select="./@id"/>
</TD>
<TD id="lastnameCol" width="125">
<xsl:value-of select="./@lastname"/>
</TD>
<TD width="100">
<xsl:value-of select="./@firstname"/>
</TD>

```

```

<TD width="100">
<xsl:value-of select="./@phone"/>
</TD>
<TD width="75">
<xsl:value-of select="./@birthdate"/>
</TD>
<TD id="accountCol" width="200">
<xsl:value-of select="./@account"/>
</TD>
<TD width="100">
<xsl:value-of select="./@city"/>
</TD>
<TD width="75">
<xsl:value-of select="./@state"/>
</TD>
</TR>
</xsl:for-each>
</TABLE>
</xsl:template>
</xsl:stylesheet>

```

LISTING 3 SortableTable.html

```

<SCRIPT src="TableUtils.js"></SCRIPT>
<SCRIPT>
function initializeTable(){
    currentSortBy = " ./@lastname";
    transformTableData();
    // select first row in the table
    selectRow(
        document.getElementsByTagName('TR').item(1));
}
</SCRIPT>
<HTML>
<STYLE>
TH{background-color:#006699;
color:white;cursor:hand;}
</STYLE>
<BODY onload="initializeTable()"
onkeydown="keyCheck(event.keyCode);">
<DIV id="tableHeader">
<TABLE width="775" border="1" cellspacing="0">
<TR>
<TH width="125" onClick="sort(' ./@lastname','text');">
LastName
</TH>
<TH width="100" onClick="sort(' ./@firstname','text');">
FirstName
</TH>
<TH width="100" onClick="sort(' ./@phone','number');">
Phone
</TH>
<TH width="75" onClick="sort(' ./@sortdate','number');">
Birthday
</TH>
<TH width="200" onClick="sort(' ./@account','text');">
Account
</TH>
<TH width="100" onClick="sort(' ./@city','text');">
City
</TH>
<TH width="75" onClick="sort(' ./@state','text');">
State
</TH>
</TR>
</TABLE>
</DIV>

```

```

<DIV id="tableBody"
style="width:775px;height:100px;overflow:scroll;">
</DIV>
<XML id="tableXSL" src="Table.xsl"></XML>
<XML id="XMLdata" src="XMLData.xml"></XML>

```

LISTING 4 XMLData.xml

```

<?xml version="1.0"?>
<rows>
    <row id="1234" lastname="Geller"
        firstname="Andrew" phone="800-555-1234"
        birthdate="10/02/1952" sortdate="19521002"
        account="123456" city="Albany" state="NY"/>
    <row id="7622" lastname="Lincoln"
        firstname="Abe" phone="800-123-5643"
        birthdate="08/05/1954" sortdate="19540805"
        account="637663" city="Miami" state="FL"/>
    <row id="3434" lastname="vanStueben"
        firstname="William" phone="888-555-3213"
        birthdate="02/02/1946" sortdate="19460202"
        account="123456" city="Denver" state="CO"/>
    <row id="3132" lastname="VanStueben"
        firstname="Robert" phone="888-555-0000"
        birthdate="02/28/1926" sortdate="19260228"
        account="031233" city="Austin" state="TX"/>
</rows>

```


Missed an issue?

We've got 'em all for you on CD!

JAVA DEVELOPERS JOURNAL

The most complete library of exclusive **JDJ** articles on one CD!

Check out over 500 articles covering topics such as...

Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, Libraries, Embedded Java, XML, Wireless, IDEs, and much more!

JDJ The Complete Works
Reg \$119.99

Buy Online
Only **\$71.99**

XML JOURNAL

The most complete library of exclusive **XML-J** articles on one CD!

Check out over 150 articles covering topics such as...

XML in Transit, XML B2B, Java & XML, The XML Files, XML & WML, Voice XML, SYS-CON Radio, XML & XSLT, XML & XSL, XML & XHTML, 2B or Not 2B, XML Industry Insider, XML Script, <e-BizML>, XML & Business, XML Demystified, XML & E-Commerce, XML Middleware, and much more!

XML-J The Complete Works
Reg \$59.99

Buy Online
Only **\$53.99**

COLDFUSION Developer's Journal

The most complete library of exclusive **CFDJ** articles!

Check out over 250 articles covering topics such as...

Custom Tags, ColdFusion and Java, Finding a Web Host, Conference Reports, Server Stability, Site Performance, SYS-CON Radio, ColdFusion Tips and Techniques, Using XML and XSLT with ColdFusion, Fusebox, Building E-Business Apps, Application Frameworks, Error Handling, and more!

CFDJ The Complete Works
Reg \$79.99

Buy Online
Only **\$71.99**

CF Advisor

The most complete library of exclusive **CFA** articles!

Check out over 200 articles covering topics such as...

E-Commerce, Interviews, Custom Tags, Fusebox, Editorials, Databases, News, CF & Java, CFBasics, Reviews, Scalability, Enterprise CF, CF Applications, CF Tips & Techniques, Programming Techniques, Forms, Object-Oriented CF, WDDX, Upgrading CF, Programming Tips, Wireless, Verity, Source Code, and more!

CFA The Complete Works
Reg \$79.99

Buy Online
Only **\$71.99**

SPECIAL OFFER:
Buy CFDJ & CFA
The Complete Works
For Only **\$129.99**



JDJStore.com

Order Online and Save 10% or More!

WWW.**JDJSTORE**.com

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

COLLECT
FOR ALL
ONLY \$**229.99**
JDJ, XML-J
CFDJ & CFA **4**



A technical solution to managing a client's large volume of data feeds

XSLT on Wall Street

Since its inception, XML has gained a strong foothold on Wall Street, but the use of XSLT for financial applications has been selective. This article reviews a real-world case study involving XSLT (and other "new" technology tools) that led to impressive business results.

Our technical solution included the development of a complete rules engine in XSLT, XML parsing in Java using DOM/SAX, a persistent store of XML data within a relational database (Oracle), a customizable report writer (Cognos), and a complete runtime environment in Unix. A two-person team completed the project work over a period of 12 weeks – an incredible feat, and one that would never have been possible with "traditional" programming paradigms. XML, XSLT, and Java technologies are worthy of your attention; read further to see how they helped us to deliver an innovative solution for one of our important clients on Wall Street.

Project Overview

A large Wall Street financial services firm developed a powerful quantitative model for valuing convertible bonds for use by its traders, risk managers, and accounting department. The convertible bond asset class is a hybrid of fixed-income and equity instruments and presents a challenge for data manipulation. Convertible bonds can have hundreds of fields of reference data (e.g., coupon, maturity, conversion price, conversion rate, currency, par amount), with many embedded time elements (e.g., call schedule, put schedule).

Although many vendors supply convertible bond reference data (e.g., Bloomberg, Reuters), our client was unwilling to accept any single vendor's data feed as the sole basis for its model inputs. The firm needed to normalize the structure of the convertible bond data elements and account for differences by each vendor in feed formats and in nomenclature for describing convertible bond reference data (i.e., field names). By transforming the data ele-

ments from each vendor into a standardized database structure, the firm created a consistent framework to select inputs for its valuation engine.

Prior to our solution, the client was trying to manage the large volume of data feeds through a manual comparison process – an onerous task given the quantity of data. The comparison logic for the data had to be heuristic and learn to suppress previously tagged discrepancies so that daily reporting wouldn't include repeat offenders.

Our project was intended to fulfill several key business objectives:

- Process data feeds from multiple vendors.
- Normalize the incoming data feeds to a common schema using a complex set of rules.
- Enable users to transform source data using complex and changing business rules – without involving subsequent IT programming resources.
- Store normalized data into a persistent data store.
- Enable users to create reports that compared data from multiple sources.
- Automate the manual data entry process.

We developed many shared classes and utilities, leveraging the solution to meet additional client requests to transform related data, generate feeds, and provide automated ways to populate their internal databases.

This article provides an overview of the project and describes the mechanics of that effort, including business and functional requirements, the detailed technical solution, and its rationale. I've included some sample code to highlight key methods we employed to meet these critical business needs for our client.

Background on Convertible Bonds

Bonds are fixed-income securities that represent the debt of domestic and international governments, corporations, banks, institutions, or municipalities. When purchasing a fixed-income security, an investor is lending money to the issuer for a specified period of time. In return, the investor (i.e., lender) receives regular interest payments (i.e., the coupon) and later the bond face value on the maturity date. The fixed-income market offers a wide range of asset classes with varying degrees of risk. The credit ratings from independent rating groups like Moody's Investors Service and Standard & Poor's are intended to rank the relative credit worthiness of these securities.

Convertible bonds are hybrid instruments that offer a fixed-income component (i.e., a coupon) and an option to convert the bond into the issuer's underlying equity at a predetermined conversion price and ratio. Convertible bonds are influenced by interest rates, credit risk, underlying equity price, market volatility (due to the embedded equity option), and other market factors. Sometimes even New York City subway noise can affect convertible bond valuation! The convertible bond asset class is an exemplary illustration of Wall Street innovation, particularly as bonds are currently issued with many interesting features. Hedge funds and other investors have played an important role in influencing the valuation and hedging of convertible bonds.

Solution Overview

Briefly, our solution consisted of the following modules:

- Job scheduling framework in Unix to invoke the XSLT rules engine upon delivery of vendor feeds
- Utility classes to transform non-XML feeds to XML format
- XSLT business rules definitions for transforming incoming data feeds into normalized schema

AUTHOR BIO

Sam Natarajan is the founder and CEO of Harvest Technology Corporation (www.HarvestTechnology.com), a leading software solutions provider. Sam has 20+ years of experience on Wall Street as an IT professional, quantitative analyst, and derivatives trader. He holds an MS in computer science and an MBA in finance, and is also a chartered financial analyst.

- Java programs to parse data feeds and invoke XSLT business rules
- Java programs to store transformed data feeds into Oracle
- Oracle database to store normalized convertible bond data elements for each vendor
- Cognos reporting framework for creating customized reports using the Oracle database
- Utility classes used as extensions in XSLT transformations

System Design Considerations

The functional and reporting framework requirements drove the main system design considerations.

Functional Requirements

- Rules engine should be extensible, accommodating input file changes automatically and consolidating data from multiple files per vendor.
- Rules engine should be independent of nomenclature used by the feed provider.
- Business rules grammar should be easy for the end user to learn, develop, test, and deploy into the production environment.
- Persistent data store should enable extraction of data in XML or other formats.

Reporting Requirements

- Define reports to compare convertible bond data from different sources.
- Define reports to identify the history of changes in data fields from a single-source vendor.
- Create a report to identify new or recently terminated/expired convertible bond issues.
- Allow downloading of report data into various formats (Excel, XML, and text files).

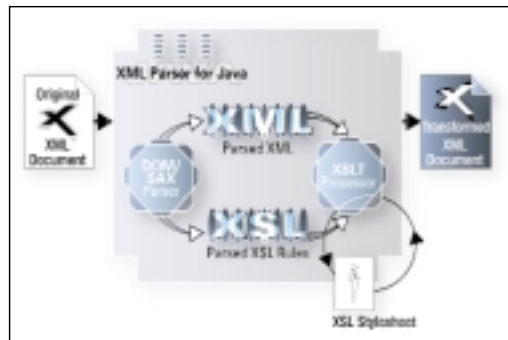


FIGURE 1 Oracle XML parser used in the rules engine

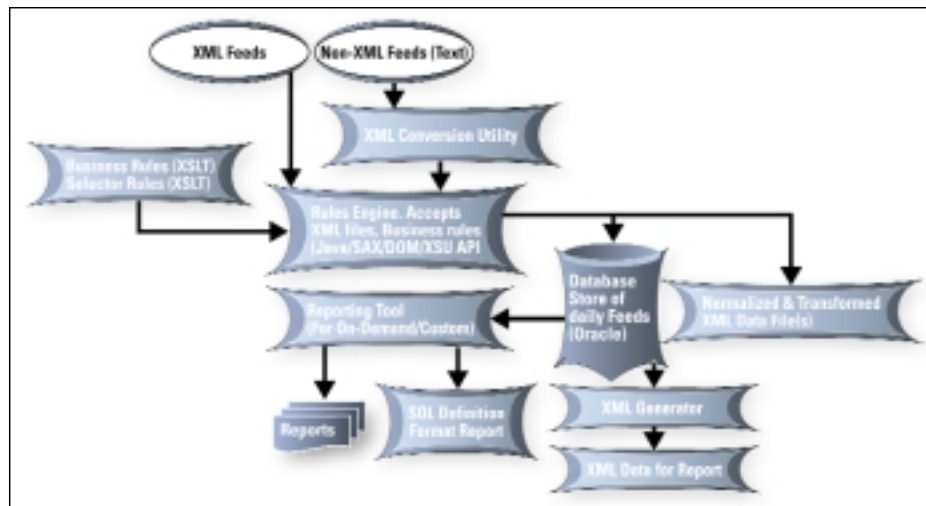


FIGURE 2 Rules engine architecture/flow diagram

EnginData Presents

2002 Developer Market Survey Reports



engindata.com

Our comprehensive reports offer insight and strategy to guide your most critical business decisions in today's fastest growing technologies...

- ✓ Establish your product and marketing strategy
- ✓ Understand your customer's needs
- ✓ Evaluate Technology & Trends

engindata **RESEARCH**

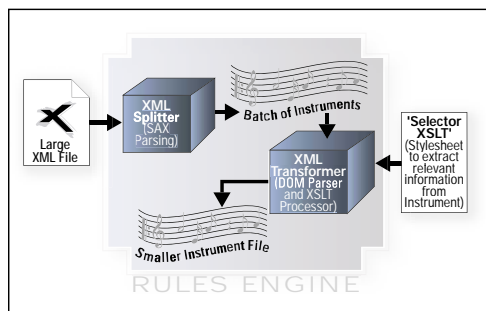


FIGURE 3 Handling large XML files

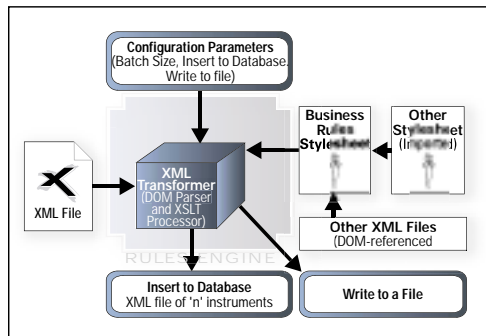


FIGURE 4 Applying the business transformation rules

Why XML? With the flexibility expected from the rules engine, XML is the obvious choice. XML doesn't have the baggage of fixed formatted files. By handling XML, the rules engine is ignorant of both source feed formats and the nomenclature used to define indicative data. (Also, we needed an excuse to get published in *XML-Journal*!)

Why XSLT? One of the core requirements of the rules engine was to allow the end user to maintain the rules engine. This required a grammar that was easy to learn and at the same time had the flexibility to handle data spanned in numerous data files with different nomenclature for attributes. XSLT was the hands-down choice; it's a declarative language (like SQL) and provides mechanisms for handling XML data spanning multiple files. XSLT also allows inclusion of Java functions in defining rules (a very powerful feature to handle more complex rules).

Why Java? We chose Java as the programming language for its platform independence and the availability of XML public domain parsers.

Why Oracle? We chose the Oracle database for its rich XML/XSLT parser API (see Figure 1) and its database-level tools for extracting XML data. The XSLT engine from Oracle has the flexibility and support for using extended Java functions in XSLT rules. Oracle's XML-SQL utility API provides easy ways to store and extract XML documents from a relational database structure. The "oraxsl" tool is especially useful for quick testing of business rules.

Why Cognos Impromptu? For the

reporting tool, we selected COGNOS Impromptu, which provides an interactive reporting framework with numerous built-in functions and flexibility to save reports in multiple formats (e.g., CSV, HTML, PDF, XLS). The Cognos reporting tool also contains a job scheduler and allows the report to be saved as an SQL query, which can be used to extract the contents of the report as well-formed XML data.

Rules Engine Architecture

Figure 2 provides a high-level architecture and data flow diagram of the rules engine. The rules engine consists of (1) numerous Java modules to parse XML files using a combination of SAX/DOM parsers, (2) an XSLT transformation engine to apply business rules in XSLT to transform feeds into normalized XML Schema, and (3) an Oracle XML SQL utility to move transformed data into a persistent data store.

The rules engine provides the option of storing the output to a database, a file, or both. The engine's ability to store the output to a file is exploited in handling large feed files (explained later). In addition, the rules engine can be configured to process a subset of convertible bond instruments rather than all the available instruments on the feed file (an important feature in using memory-constrained DOM parsing).

Once the rules engine stored the normalized data into a relational structure, the business user analyzed the quality of the data and selected the "best" data source for use in the financial models. The XML generator utility provided additional tools for creating XML output from saved SQL queries and/or text files.

Key System Modules

Converting Text Files to XML

We developed a utility class to convert the data contained in delimited text files to XML format. The premise of this utility was simple: the first line in the text file would contain the field names to be used as attribute tag names, while all the other lines contained the values for those fields. The utility built each record in the text file as a row element, with all data from each row as attributes of the row element. The utility provided flexibility in defining the root node and rows of the XML file as parameters within a configuration file. The text file delimiter was also defined as a configuration file parameter, which meant that a file with any delimiter could be processed. We also created two helper classes: one that abstracted out the parser and the other that had utility functions for creating/manipulating XML messages.

Handling Large XML Files

The most important technical aspect

of the rules engine is its ability to handle very large XML files – over 100MB. These large files typically contained data for more than 10,000 convertible bond instruments, with each instrument having over 200 attributes. We adopted the divide and conquer strategy to address the constraints of loading the entire file into memory for DOM parsing. The XML splitter (based on the SAX parser) was used to obtain a subset of instruments to build a document object and subsequent XSLT transformation. To reduce the file size, we extracted only relevant field attributes from the input file using selector XSLT rules. (Figure 3 provides a schematic representation of this process.)

Transforming and Saving Results to a Persistent Store

The XML transformer served as the main module, coordinating the activities of the rules engine and creating an instance of the XML splitter to parse the XML file into smaller subsets. With each extracted subset, the transformer used the XSLT processor to apply the business rules. Upon transformation, the parsed XML data was saved into the database using the Oracle XML SQL utility API. (Figure 4 illustrates this process.) With careful monitoring of the performance of the system, we determined an optimum batch size for each of the feeds. We created intermediary XML documents for each of the input files from the vendor and used XPATH to access content from these imported documents in the main module.

Business Rules – XSLT

After carefully reviewing the various data and working with the business users, we created an initial set of business rules for each data source. From the outset we organized the XSLT rules in layers to promote modularity and to enable the reuse of generic rules common to multiple sources of data. We created a template for each rule with guidelines for variable names, formatting, and rules logic.

The following are examples of the business rules:

- *Rule to translate the coupon frequency:* Replace numeric data 1, 2, 4, and 12 with "A" for annual, "S" for semi-annual, "Q" for quarterly, and "M" for monthly, respectively. There were many such "code/decode" rules in the engine (see Listing 1).
- *Rule to set the maturity date:* Here the maturity date should be set to "01/01/2001" for perpetual instruments (instruments for which the marketSector="Pfids") where the maturity date on the incoming file is null/blank. The ability to change a field

Now in More than 5,000 bookstores worldwide

subscribe **Now!**

FOR FAST
DELIVERY

Go
Online
and
Subscribe
Today!

The World's Leading Independent WebLogic Developer Resource

FOR WLS DEVELOPERS BY WLS DEVELOPERS
BEA WebLogic
DEVELOPER'S JOURNAL

WebLogic

Journal.com

SYS-CON Media, the world's leading publisher of *i*-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebLogic. *Only \$149 for 1 year (12 issues) regular price \$180.

**SYS-CON
MEDIA**

Helping
you enable
inter-company
collaboration
on a global scale

- Product Reviews
- Case Studies
- Tips, Tricks and more!

SPECIAL
INTRODUCTORY OFFER
SAVE \$31*
HURRY, DON'T DELAY! OFFER EXPIRES JUNE 30, 2002

SAVE UP TO \$100 ON MULTIPLE SUBSCRIPTIONS

**Special
online offers**

Pick **4** or **5** and Subscribe
for one **special low** price

**RECEIVE YOUR
DIGITAL EDITION
ACCESS CODE
INSTANTLY
WITH YOUR PAID
SUBSCRIPTION**



Wireless Business & Technology • Java Developer's Journal

Web Services Journal • WebLogic Developer's Journal

XML-Journal • WebSphere Developer's Journal

ColdFusion Developer's Journal • PowerBuilder Developer's Journal

**SYS-CON
MEDIA**

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

WWW.SYS-CON.COM/SUBOFFER.CFM

based on the content of another field was another common rules definition (see Listing 2).

- **Rule for par:** Par is calculated by dividing the legacy par amount by FX rate in case of old Euro currencies. We have an XML file containing old European currency codes and the exchange rate for Euro currency. We had many such rules that referenced legacy data for rules processing (see Listing 3).
- **Rule to obtain the number of days between two dates:** Here we use Java functions to obtain the number of days between two dates. We had many rules involving date manipulation (see Listing 4).
- **Rule to extract time series elements:** Here we show how to extract schedule data from an imported file. The ability to manipulate time series data from

schedules was an important feature of the XSLT rules definitions (see Listing 5).

Reporting Tool

With active participation from the business user, we defined report templates, allowing the end user to create reports with minimal effort. In addition to the above report templates, we provided the capability to suppress known differences/exceptions in the daily comparison of data elements. Using the report/job scheduler, the user could save daily production reports in XLS/PDF formats and subsequently transfer the files to a shared network drive for distribution to other users.

Lessons Learned

The system has been well received by our client and continues to enjoy heavy

use in their daily processing of convertible bond data. We achieved user satisfaction by empowering the business user with the ability to write and control the rules engine.

From a technical perspective, this project helped remind us of several key points:

- XSLT is a powerful framework for manipulating XML data.
- XSLT can be written in simple, easy-to-understand declarative rules.
- XSLT stylesheets can be used effectively to manage rules definitions.
- Using a combination of SAX and DOM parsing with large files is effective and efficient.
- The paradigm of modular design extends well to the organization of business rules. 🌐

SAM @ HARVESTTECHNOLOGY.COM

LISTING 1 Rule to translate coupon frequency

```
<xsl:variable name="vCouponFrequency">
  <xsl:choose>
    <xsl:when test="number($vCouponPaymentFrequency) = 1">
      <xsl:value-of select="'A'"/>
    </xsl:when>
    <xsl:when test="number($vCouponPaymentFrequency) = 2">
      <xsl:value-of select="'S'"/>
    </xsl:when>
    <xsl:when test="number($vCouponPaymentFrequency) = 4">
      <xsl:value-of select="'Q'"/>
    </xsl:when>
    <xsl:when test="number($vCouponPaymentFrequency) = 12">
      <xsl:value-of select="'M'"/>
    </xsl:when>
  </xsl:choose>
</xsl:variable>
```

LISTING 2 Rule to set maturity date

```
<xsl:variable name="vCalMaturityDate">
  <xsl:choose>
    <xsl:when test="$vmarketSector='Pfd' and string($vMaturityDate)=''">
      <xsl:value-of select="'2001-01-01'"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$vMaturityDate"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

LISTING 3 Rule for par

```
<!-- Document containing currency conversion information. -->
<xsl:variable name="docCurrencyConv" select="document('./cur_conversion.xml')"/>

<!-- Lookup Currency Conversion table and get the fx rate & the to Currency code -->
<xsl:variable name="vToCurrency" select="$docCurrencyConv/Root/Conversion[@FromCurrency=$vCurrencyCode]/@ToCurrency" />
<xsl:variable name="vFXRate" select="$docCurrencyConv/Root/Conversion[@FromCurrency=$vCurrencyCode]/@FXRate" />

<!-- Rule to build par value by dividing the LegacyPar with the FXRate-->

<xsl:variable name="vPar">
  <xsl:choose>
    <xsl:when test="string($vToCurrency) != ''">
      <xsl:value-of select="($vLegacyPar div $vFXRate)" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$vLegacyPar" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

LISTING 4 Rule to obtain number of days between two dates

```
<xsl:templatename="GetNumberOfDays"
  xmlns:date="http://www.oracle.com/XSL/Transform/java/java.sql.Date"
  <xsl:param name="pFromDate" />
  <xsl:param name="pToDate" />
  <xsl:variable name="vNumDays"
    select="(Date:getTime(Date:valueOf($pToDate)) - Date:getTime(Date:valueOf($pFromDate))) / (1000*60*60*24)" />
  <xsl:value-of select="$vNumDays" />
</xsl:template>
```

LISTING 5 Rule to extract time series elements

```
<!-- Rule to get Last schedule date from a repeating element group of certain type of schedule -->

<xsl:variable name="vLastType1ScheduleDate">
  <xsl:choose>
    <xsl:when test="schedule[@Type='Type1']">
      <xsl:value-of select="schedule[@Type='Type1']/scheduleEvent[last()]/scheduleDate" />
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="'0000-00-00'"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>

<xsl:for-each select="schedule[@Type='Type2']/scheduleEvent">
  <xsl:variable name="vScheduleDate" select="scheduleDate" />
  <xsl:variable name="vSchedulePrice" select="schedulePrice" />

  <!-- Ignore Type2 Schedule events before the last schedule date for Type1 schedule date -->

  <xsl:if test="$vLastType1ScheduleDate != '' and concat(substring($vScheduleDate,1,4), substring($vScheduleDate,6,2), substring($vScheduleDate,9,2)) > concat(substring($vLastType1ScheduleDate,1,4), substring($vLastType1ScheduleDate,6,2), substring($vLastType1ScheduleDate,9,2))">

    <xsl:variable name="vProvPrice" select="'0'"/>
    <xsl:variable name="vPrice" value="$vSchedulePrice"/>
    <xsl:call-template name="createScheduleElement">
      <xsl:with-param name="pSourceID" select="$vInstrumentID" />
      <xsl:with-param name="pType" select="'Type2 Schedule'" />
      <xsl:with-param name="pScheduleDate" select="$vScheduleDate" />
      <xsl:with-param name="pPrice" select="$vPrice" />
      <xsl:with-param name="pProvPrice" select="$vProvPrice" />
    </xsl:call-template>
  </xsl:if>
</xsl:for-each>
```

DOWNLOAD THE CODE @
www.sys-con.com/xml

THE LARGEST INDEPENDENT

JAVA

DEVELOPER CONFERENCE IN THE WORLD!

**WIN A
\$35,000
LUXURY CAR!**



ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING
TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

JAVA IN JUNE

ESPECIALLY IN NEW YORK

REGISTER ONLINE TODAY

FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!

VISIT **WWW.SYS-CON.COM**

A Sampling of Java-Focused Sessions

- JAVA 1.4: WHAT'S NEW?
- BUILDING TRULY PORTABLE J2EE APPLICATIONS
- JAVA TOOLS FOR EXTREME PROGRAMMING
- BUILDING ASYNCHRONOUS APPLICATIONS USING JAVA MESSAGING
- .NET VS. J2EE
- J2EE: SETTING UP THE DEVELOPMENT ENVIRONMENT
- BUILDING WEB SERVICES WITH J2EE
- DETECTING, DIAGNOSING, AND OVERCOMING THE FIVE MOST COMMON J2EE APPLICATION PERFORMANCE OBSTACLES



ALAN WILLIAMSON
JAVA CHAIR • EDITOR-IN-CHIEF
JAWA DEVELOPER'S JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 J-TECHNOLOGY PROFESSIONALS

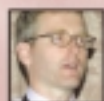
Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- J-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



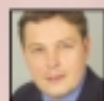
TYLER JEWELL
PRINCIPAL TECH
EVANGELIST • BEA



GREGG KIEBLING
CEO
SITRAKA



DAVID LITWACK
CEO
SILVERSTREAM



BARRY MORRIS
CEO
IDIA



GREGG O'CONNOR
PRESIDENT
SONIC SOFTWARE



RICK ROSS
FOUNDER
JAWA LOBBY



JAMES DUNCAN
DAVIDSON
FATHER OF ANT

For Exhibit Information

CONTACT: MICHAEL PESICK
135 CHESTNUT RIDGE RD.
MONTVILLE, NJ 08045
201 842-3057
MICHAEL@SYS-CON.COM

JDJEDGE
conference & expo

JUNE 24-27

JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3

SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

SPONSORED BY:

IONA | END 2 ANYWHERE™

ADOS

bea

TogetherSoft

SilverStream

Actional

MERANT

Altaworks

PolarLake
Enterprise Strength XML

ALTOVA sm3 app

MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

CE Advisor

WebServicesMail

WIRE

XML.ORG

WROX

port

SDTimes

JAVA SOURCE

wireless

XML

WebLogic

WebServices

WebSphere

COLDFUSION

*How far have we gone?*

WebServicesDirections

It's been a long time since the last **XML in Transit** column. Did you miss my musing on Web services? I doubt it. More likely, you were busy keeping up with all the new initiatives in the Web services space. Those of you with corporate responsibilities were probably wondering how to get some real ROI out of Web services. The more entrepreneurial of you were thinking how to make money in this new world.

I too have been trying to organize my thoughts. This is going to be the first in a series of articles that go beyond the basic SOAP/WSDL/UDDI Web services stack to cover the innovation and market dynamics in adjacent areas that are of key significance to the evolution and ultimate success or failure of Web services as the new new thing in distributed application development, assembly, and integration.

The Basic Technology Stack

Figure 1 shows the basic Web services interoperability stack the way most people would think of it up to last year. Going from the bottom up, in a typical Web services scenario we have HTTP as the communication protocol, XML for data representation, XML Schema for data format specification, SOAP for service invocation, WSDL for service description, and UDDI for service discovery. This technology stack enabled the basic Web services workflow described in Figure 2. We've covered these technologies and the publish-find-bind processes in detail in past issues of **XML in Transit**.

What we get from the combination of these technologies is the ability to connect disparate systems. Okay, I'm simplifying things a bit, but I think the basic message holds true. The potential is there for doing much more than simply connecting applications. We could expose enterprise-quality Web services. We could orchestrate atomic processes into meaningful B2B applications. We could assemble entire new applications (front end to back end) from Web services.

We could, but as a whole our industry is quite far from this better world. We're held back by the combination of three factors: technological reality, standards friction, and market dynamics. I'll cover some of the technologies we need to put in place in this issue. In the next installment I'll fin-

ish off the discussion with a look at standards friction and market dynamics.

Technological Reality

Let's do a quick check of the various technological capabilities we'd need to have to achieve what's promised by the Web services hype machine: enterprise-quality seamless dynamic assembly and orchestration of Web services.

We can start by looking at what it means to have enterprise-quality Web services. Since enterprise requirements do vary, we'll focus on a relatively stringent requirement set.

Probably the most basic thing to think about is reliable transport. The overwhelming majority of Web services nowadays run on top of HTTP, a protocol with pretty poor reliability. The delivery of requests and responses isn't guaranteed. Certain exceptional conditions in the HTTP stack can leave a system in an undetermined state. We can use SOAP over any protocol; however, the industry as a whole hasn't agreed on how to do this. Therefore, if you want to use reliable HTTP (HTTPR) or guaranteed messaging, you can't count on interoperability. This is the main problem with the current state of Web services – pretty much anything is possible, but little beyond the basic technology stack is common practice.

In addition to reliable transport, enterprise-quality Web services often need to be transactional. This means that the systems exposed through Web services need to be able to participate in a transaction under the coordination of a transaction manager (TM, also known as a distributed transaction coordinator, or DTC). Distributed transactions used to be rocket science. With the proliferation of transactable DBMSs and managed components such as EJBs, we're getting into a world that can deal reasonably well with

transactions. However, transactions pose at least two problems for Web services:

1. SOAP messages need to carry transaction identifiers (TIDs) that associate the message with a currently executing transaction.
2. Distributed transaction management requires that the systems participating in transactions have a back channel of communication about the transaction itself, independent of the actual Web service operation.

There are some approaches for Web services transaction management. The leader is probably the Transaction Authority Markup Language (XAML) but the industry buy-in is not overwhelming. There also are hybrid approaches that use the divide and conquer pattern: put TIDs in SOAP messages but do transaction coordination out-of-band without the burden of Web services. The Transaction Internet Protocol (TIP) could enable this approach.

In environments where Web services are used for automating business processes, system implementers face another problem – long-running transactions. Without going into implementation details, suffice it to say that the basic approaches for managing transactions that take on the order of a few seconds to a few minutes do not scale to handle transactions that take days or weeks to complete. Long-running transactions have more complicated management cycles. This affects Web services by requiring agreement and standardization of additional APIs, SOAP headers, and WSDL extensions. OASIS has done some interesting work on the Business Transaction Protocol (BTP), which could be a good base for broad standardization in this space because it covers both cross-enterprise transactions and long-lived transactions.

Another requirement for flexible business process management is the customizable routing of messages. Therefore, Web services routing is another area that needs standardization. Microsoft's WS-

AUTHOR BIO

Simeon Simeonov, VP, emerging technologies, at Macromedia, Inc., is a member of the W3C working group on XML Protocol and the J2EE expert groups on XML business messaging and XML data binding.

THE LARGEST INTERNATIONAL

WEB SERVICES CONFERENCE & EXPO IN THE WORLD!

**WIN A
\$35,000
LUXURY CAR!**



ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

WEB SERVICES SKILLS, STRATEGY, AND VISION

REGISTER ONLINE TODAY

**FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!**

VISIT WWW.SYS-CON.COM

Focus on Web Services

Web Services, the next generation technology that will enable the Internet to work for you and your business, and finally provide that ROI you have been after, will be put under a microscope at Web Services Edge East 2002.

Information-packed sessions, exhibits, and tutorials will examine Web Services from every angle and will provide cutting-edge solutions and a glimpse at current and future implementations. Hear from the innovators and thought leaders in Web Services. Enjoy a highly interactive CEO Keynote panel that will debate and discuss the realities and promises of Web Services.



A Sampling of Web Services-Focused Sessions

- PRACTICAL EXPERIENCES WITH WEB SERVICES AND J2EE
- STATE OF THE WEB SERVICES INDUSTRY
- THE ODD COUPLE: MAKING .NET AND J2EE WORK TOGETHER
- EXPLORING THE .NET MY SERVICES INITIATIVE
- STANDARDS WATCH
- GUARDING THE CASTLE: SECURITY & WEB SERVICES



SEAN PADDY
CONFERENCE TECH CHAIR
WEB SERVICES TRACK CHAIR
EDITOR-IN-CHIEF
WEB SERVICES JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 J-TECHNOLOGY PROFESSIONALS

Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- J-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



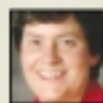
JEAN-FRANCOIS ABRAMATIC
SENIOR VP R&D, ILOG
FORMER CHAIRMAN, W3C



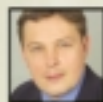
DAVE CHAPPELL
IP CHIEF TECHNOLOGY EVANGELIST
SONIC SOFTWARE



GREG KRESSLING
CEO
SITRAKA



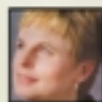
ANNE THOMAS MANES
CEO
SYSTEMET



BARRY MORRIS
CEO
IONA



ERIC RUDDER
SENIOR VP DEVELOPER
AND PLATFORM EVANGELISM
MICROSOFT



PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit Information

CONTACT: MICHAEL PESICK
115 CHESTNUT RIDGE RD.
MONTVILLE, NJ 08045
201 882-3057
MICHAEL@SYS-CON.COM

web services **EDGE**
conference & expo

JUNE 24-27

JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3

SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

SPONSORED BY:

IONA | END 2 ANYWHERE™

ADOS

bea

TogetherSoft

SilverStream

Actional

MERANT

Altaworks

PolarLake
End-to-end Strength XML

ALTOVA

MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

CE Advisor

WebServicesMail

WIRE

XML.org

WROX

iCert

SDTimes

BASIS

wireless

JAVA

WebLogic

XML

WebSphere

WebServices

WebSphere

COLDFUSION

OWNED AND PRODUCED BY

SYS-CON MEDIA | SYS-CON EVENTS

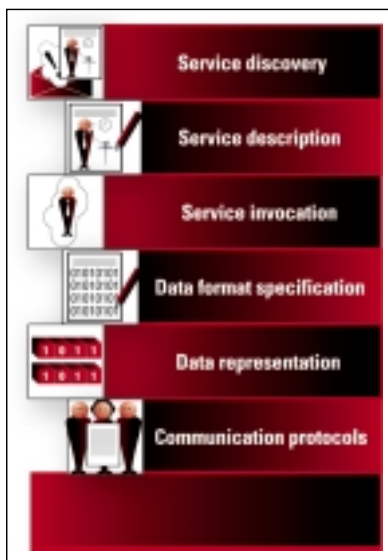


FIGURE 1 Web services interoperability stack

Routing is a step in the right direction, but it lacks industry support.

Before I get off the topic of business process automation, I have to point out that if business processes are exposed via Web services, then we need a set of standards defining the required APIs of these services together with a complete description of a process in an XML format so that composite business processes can themselves be exposed as Web services.

The good news is that much work has been done in this area. Microsoft developed XLang for BizTalk Server. IBM developed Web Services Flow Language (WSFL). The bad news is that the merger of the two specifications into one (something that IBM and MS did well in the case of NASSL and SCL/SDL that became WSDL) is not in sight.

Wait, I haven't had the chance to talk about security and privacy. At a basic

level we need at least three capabilities:

1. Associate credentials (username/password, X.509 certificate, a Kerberos ticket, an authenticated session identifier) with Web services messages.
2. Maintain message integrity to guarantee that nobody has tampered with the message content.
3. Maintain message confidentiality by preventing anyone other than the author and the target recipient from seeing the contents of the message.

These basic capabilities are well understood in the security space, and we've built (over and over and over again) infrastructure to provide them. Web services pose two challenges: (1) the need for widely adopted standards and (2) the complications of intermediaries that may need to see some parts of SOAP messages but not others. All the pieces are in place for the industry to agree on a solution to this problem; we just haven't gotten around to doing it.

In fact, the various companies and industry groups have produced a number of specs and are busy working on others. The W3C has XML Signature, XML Encryption, and XML Key Management Specification (XKMS, consisting of X-KRSS [XML Key Registration Service Specification] and X-KISS [XML Key Information Service Specification]). Microsoft has combined some of these into WS-Security, a reasonably complete approach to Web services security covering credentials association as well as message integrity and confidentiality.

Meanwhile, OASIS is working on a security assertions markup language (SAML) that will help describe trust assertions for user identities and authori-

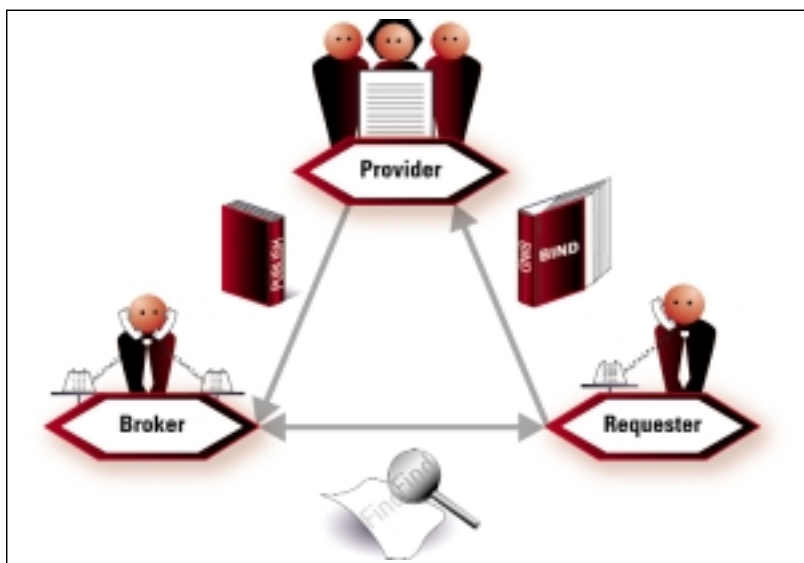


FIGURE 2 Publish-Find-Bind workflow

RECEIVE \$150
DISCOUNT OFF FULL CONFERENCE
WEB SERVICES EDGE REGISTRATION

web services **EDGE**
world tour 2002

**Learn How to Develop
SOAP Web Services NOW!**
at a One-Day Seminar...Coming to a City Near You!

zations, something that becomes important in assembling aggregate services and in automating business processes via Web services. Needless to say, the fact that so many specifications are in development and that they are split between different companies and standards organizations will delay adoption of more universal standards.

Another huge topic of interest is quality of service (QoS) and the requisite monitoring and management capabilities that enable QoS metrics and tuning. This is a large area with unfortunately too little coordinated action related to the Web services space. There are two types of QoS that we need to be concerned with:

1. *Technical QoS*, which encompasses concepts such as availability, accessibility, integrity, performance, and reliability
2. *Business-level QoS* (called by some quality of business [QoB]), which focuses on metrics related to the business processes supporting a service – for example, inventory levels, shipping guarantees, and insurance

Providing targeted QoS significantly complicates the architectures and workflows of using Web services. Consider the following that need to be in place:

- Common vocabularies for defining service-level agreements (SLAs) as collections of QoS assertions.
- Extensions to UDDI to include QoS parameters in service discovery.
- Agreed-upon patterns for QoS negotiation between service requesters and service providers.
- Pervasive monitoring of all components (technology- or business-level) taking part in a transaction. This requires common APIs and message

formats that will likely be exposed via Web services.

- Management infrastructure that can alert interested parties when QoS parameters change.

This is a hot area where we can expect significant innovation and much competition for market dominance between the established leaders in the Web services space.

The Future Technology Stack

Putting all of this together, it seems that the Web services technology stack of the future looks much more like what is shown in Figure 3. It's somewhat difficult

to represent the dynamics of the Web services space in two dimensions, which is why I resorted to the use of color as a third dimension. The green zones denote calmer areas where the standards are already well established and/or vendors and standards bodies are collaborating reasonably well. The yellow zones are areas where standards are still being fleshed out and there's more uncertainty. The red zones define areas where interoperability is a long way away because standards are incomplete and there is high likelihood of platform battles. The blue zones are areas yet to be explored in an organized way.

If history offers any lessons from the software

space, we must expect at least a couple of turbulent years full of standards flux. While the basic message of Web services is that of interoperability across disparate systems, if you're building enterprise-quality Web services, you should prepare to fund a lot of custom integration work between various types of Web services. We're starting to see the beginnings of this period: Web services produced by some of the major vendors' enterprise systems are not interoperable. You know, on some gloomier days I think that God is a systems integration consultant. ☹

SIMEONS @ MACROMEDIA.COM

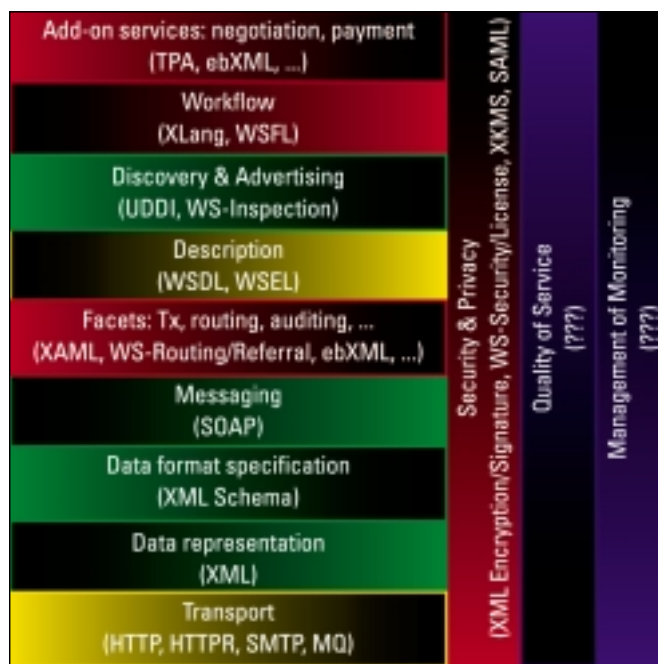
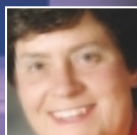


FIGURE 3 Web services technology stack of the future

Jump-start your Web Services knowledge. Get ready for Web Services Edge East and West!

AIMED AT THE JAVA DEVELOPER COMMUNITY AND DESIGNED TO EQUIP ATTENDEES WITH ALL THE TOOLS AND INFORMATION TO BEGIN IMMEDIATELY CREATING, DEPLOYING, AND USING WEB SERVICES.

EXPERT PRACTITIONERS TAKING AN APPLIED APPROACH WILL PRESENT TOPICS INCLUDING BASE TECHNOLOGIES SUCH AS SOAP, WSDL, UDDI, AND XML, AND MORE ADVANCED ISSUES SUCH AS SECURITY, EXPOSING LEGACY SYSTEMS, AND REMOTE REFERENCES.



PRESENTERS...

Anne Thomas Manes, Systinet CTO, is a widely recognized industry expert who has published extensively on Web Services and service-based computing. She is a participant on standards development efforts at JCP, W3C, and UDDI, and was recently listed among the Power 100 IT Leaders by Enterprise Systems, which praised her "uncanny ability to apply technology to create new solutions."



Zdenek Svoboda is a Lead Architect for Systinet's WASP Web Services platform and has worked for various companies designing and developing Java and XML-based products.

EXCLUSIVELY SPONSORED BY



systinet

Register at www.sys-con.com or Call 201 802-3069

BOSTON, MA (Boston Marriott Newton) **SOLD OUT!** JANUARY 29

WASHINGTON, DC (Tysons Corner Marriott) **SOLD OUT!** FEBRUARY 26


NEW YORK, NY (Doubletree Guest Suites) **NEW DATE!** APRIL 19

SAN FRANCISCO, CA (Marriott San Francisco) MAY 13

REGISTER WITH A COLLEAGUE AND SAVE 15% OFF THE \$495 REGISTRATION FEE.

**i-TECHNOLOGY
SYS-CON EDUCATION**

Data Driven Web Graphics with SVG: A Declarative



Scalable Vector Graphics is likely to revolutionize the way Web graphics are rendered, stored, manipulated, and associated with content. SVG, a 2D portable Web graphics standard recommended by the W3C in September 2001, is an XML-based standard for specifying both graphics and content. SVG replaces server-side image file creation or applet-based graphics with client/Web browser-based rendering of images.

This article focuses on Web annotation applications in which users draw on a Web browser and associate properties with the drawing. The graphical and nongraphical (business) properties of the annotations created by users are stored in a database.

The appeal of SVG is that it's data driven, not bitmap driven, meaning that the graphical content of an image is described in SVG as a sequence of commands to draw a line from one point to another, draw a circle with a specified center and radius, and so on. Thus the entire SVG image is created, stored, and rendered using commands and data in an XML format.

This article presents a declarative approach to creating a middle tier for SVG applications. The J2EE/XML-based middle tier:

- Updates data sources (relational databases, EJB, and/or Java objects) with data in SVG files
- Delivers SVG XML files with graphics and business data extracted from data sources.

The declarative approach requires no coding in Java or VB. Instead, the SVG application is created by declaratively specifying:

- Data sources (relational databases, EJB, Java objects, and others)
- Data methods (SQL, stored procedures, Java objects) for retrieving data from and updating data into data sources

- Tag-based JSP or XSL-based transformations for placing collected data into an SVG XML file

The intent of this article is to provide insight into the rapid development of SVG-based Web applications that contain both graphical and business content. Applications of this approach include Web simulation, Web charting, Web reporting, and Web-based image annotation.

The Appeal of SVG

Although currently popular browsers like Internet Explorer and Netscape don't render SVG directly, plug-ins to the browsers from Adobe (SVG Viewer) and others will render an SVG document. We believe that because SVG is a W3C standard, it will be supported by future releases of browsers.

SVG has a number of characteristics that make it appealing for Web graphics – for example, vector graphics. All the graphical information is stored in a sequence of commands to draw lines, shapes, and other objects. This information is eventually converted to a display application-specific bitmap, also called raster graphics. SVG is converted to a raster image by a vendor-specific program.

SVG is also scalable. Zooming into an SVG image doesn't distort the image (by producing jagged edges) because redrawing instructions are sent to the rendering program, rather than pixel values in a bitmap.

Being XML-based, SVG is data driven. It specifies an XML format for vector graphics commands. In addition, the SVG XML file can include business data tied to the graphical objects. Listing 1 is a simple SVG code for creating a ball object by first creating a graphical element – a circle – and associating properties like weight, material, and price with it. (To try out the example, download the SVG

SVG Approach

Written by Pramod Jain & Satya Komatineni

plug-in for your browser [one source for the plug-in is www.adobe.com]; create a new file, circle.svg; copy the code into the file; open it from the Web browser. All the examples have been tested on IE5.5 with Adobe's SVGViewer.)

Data driven in this case means that the center of the circle and the radius are specified explicitly. SVG makes Web graphics declarative: an SVG XML file declaratively specifies the properties of a drawing element. This is substantially different from bitmap-based graphics like GIFs or JPEGs.

SVG is like HTML in that the source code for what's displayed in a browser can be viewed, making it easier to learn.

`<g>` is a group tag. The three properties of the ball are accessible through JavaScript by walking through the DOM (Document Object Model) of SVG. For example, it can be programmed in JavaScript to display the three values in three input text boxes when a user clicks on the circle, and it can update the values in DOM when a user changes values in the text boxes.

The SVG specifications contain the DOM, an API for accessing, adding, deleting, and changing the contents of the SVG-XML document. For instance, interactivity and animation can be controlled by JavaScript. The example in Listing 1 is modified to illustrate this point:

```
<g id="ball" onmousedown="DoOMD(evt)" onmouseup="DoOMU(evt)">
  <circle cx="200" cy="200" r="100"/>
</g>
```

In this example, when the user clicks down on the ball, an event is generated. This event is processed by a user-defined JavaScript function, DoOMD().

For animation consider the following, where an onmouse click increases the radius of the ball from 100 to 200 in one second:

```
<svg width="800" height="440">
  <g id="ball">
    <circle cx="200" cy="200" r="100">
      <animate begin="click" attributeName="r" from="100" to="200"
        dur="1s" fill="freeze"/>
    </circle>
  </g>
</svg>
```

Furthermore, using SVG DOM, mouse events captured by JavaScript can be used to draw. This allows drawing and annotation on the Web browser. For examples of such tools see SVG Editor by Adobe at www.adobe.com/svg/demos/svgDraw/svgDraw/index.html.

The focus of most SVG applications thus far has been on creating better Web pages using SVG. The focus of this article is data-driven business applications with an emphasis on creating graphical and business content from a browser.

Declarative Framework for SVG Applications

Floor Plan Example

The remainder of this article will use the floor plan of a building for illustrative purposes. Users should pull up a digital image (GIF or JPEG file) of the floor plan and review existing and create new annotations for locations of electrical outlets, phone jacks, and Ethernet jacks, then save the changes into a database. Annotations will take the form of rectangles: red for electrical outlets, blue for phone jacks, and green for Ethernet jacks.

In addition to the graphics, the user has to specify the number of electrical outlets and phone and Ethernet jacks at each location, as well as the IP address.

In the example in Listing 2, the appeal of SVG is that on a Web browser annotation markings (AM) can be created on the floor plan using drawing tools, and annotation properties (AP) can be associated with the markings. While AM and AP are created on the browser, JavaScript populates the DOM with AM and AP data. When the user

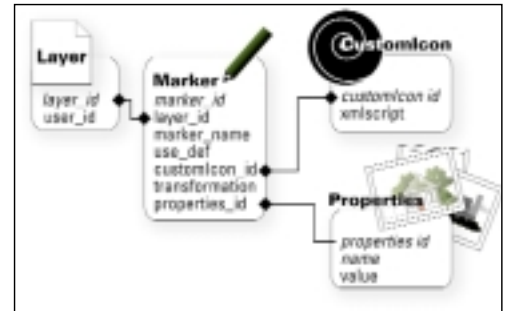


FIGURE 1 Database schema for Annotation Markings and Properties

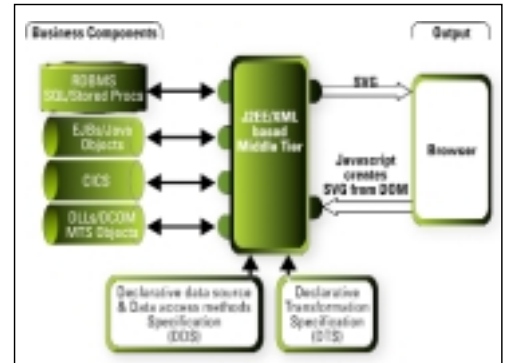


FIGURE 2 Components and flow of information of the Declarative approach

saves the annotation, JavaScript takes the content of the DOM, creates an XML file, and sends it to a middle-tier application. The middle-tier application then stores the data in the database. In the other direction the middle tier creates the SVG XML file by extracting data from the database and delivers it to the browser.

The example in Listing 2 will be used to illustrate how the declarative nature of SVG can be combined with the declarative nature of the middle tier to rapidly deliver scalable applications. With this combination there is little or no Java coding to build SVG-based enterprise Web applications.

The example in Listing 2 is used in two ways:

1. *To create Annotation Markings and Properties (AMP):* When a user enters the floor plan example for the first time, the drawing canvas will be empty. The user will see a background image of floorplan.jpg. As the user creates AMP, the contents inside the drawing canvas will be created in the DOM. When the user saves, JavaScript will create the SVG XML file in Listing 2 from the DOM and submit it to the middle tier, which extracts the AMP data and inserts it into the database.
2. *To display AMP with data extracted from data sources:* When the user asks to see AMP created in the previous step, the middle tier executes data access methods on the database or other data sources and converts relational and nonrelational data returned by data access methods to an infoset (a hierarchical data model for XML). Next, the middle tier transforms the generated infoset into the SVG XML file using

one of the available transformations – XSL, JSP, or tags – and delivers the file to the browser

The database schema in Figure 1 illustrates the example. A layer is like a transparency placed over an image on which a user draws annotations. It contains multiple markers, each either a “hand-drawn” image (custom icon) or a standard predefined icon (the elements defined inside the <defs> in Listing 2). Each marker is associated with multiple properties.

Overall Process

The internals of the middle tier are described later; however, the tier is completely controlled by declarative specifications contained in two files, DDS and DTS (see Figure 2). A programmer constructs an application by creating a Declarative data source and a Data access and update methods Specification (DDS) along with a Declarative Transformation Specification (DTS). DDS specifies data sources, access methods, and update methods. Using the API of the middle tier, plug-in modules can be written to interface with other data sources. An example of DDS is provided in later sections of this article.

A user request from a browser contains a request name that points to a section of the DDS XML file containing data access or update methods executed on the database. The returned data is held in the middle tier as XML infosets (an example of the data format is given later). The data is then

THE LARGEST INTERNATIONAL

XML

CONFERENCE & EXPO IN THE WORLD!

**WIN A
\$35,000
LUXURY CAR!**



ATTENDEES WILL BE INVITED TO TAKE A GOLF SWING
TO WIN AND RIDE OFF IN A \$35,000 LUXURY CAR!

XML-NEXT G OF ENTERPRISE DEPLOYMENT

REGISTER ONLINE TODAY

**FOR LOWEST CONFERENCE RATES
EARLY SELL-OUT GUARANTEED!**

VISIT WWW.SYS-CON.COM

Focus on XML

XML is today's essential technology to develop and deploy Web services. Here's your chance to learn from expert practitioners how XML is making the next phase of the Web a reality.

Focus on standards, interoperability, content management, and today's new quest for more efficient and cost-effective Internet and intranet-enabled business process integration.

Focus on XML during information-packed days while you enjoy an XML/Web Services keynote panel, comprehensive conference sessions, and an unparalleled opportunity to exchange ideas with peers and industry leaders.



A Sampling of XML-Focused Sessions

- XML STANDARDS - AN OVERVIEW
- OASIS STANDARDS UPDATE
- UBL - A UNIVERSAL BUSINESS LANGUAGE
- BRINGING XML TO PKI
- LINK MANAGEMENT WITH XLINK
- PRACTICAL XSLT AND XPATH
- ENTERPRISE CONTENT MANAGEMENT WITH XML
- XML IN THE ENTERPRISE AND INTER-ENTERPRISE WORLD



NORBERT MIGULA
XML CHAIR
BOARD OF DIRECTORS, OASIS
INDUSTRY EDITOR
WEB SERVICES JOURNAL

Featuring...

- UNMATCHED KEYNOTES AND FACULTY
- THE LARGEST INDEPENDENT JAVA, WEB SERVICES, AND XML EXPOS
- AN UNPARALLELED OPPORTUNITY TO NETWORK WITH OVER 5,000 I-TECHNOLOGY PROFESSIONALS

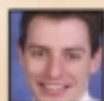
Who Should Attend...

- DEVELOPERS, PROGRAMMERS, ENGINEERS
- I-TECHNOLOGY PROFESSIONALS
- SENIOR BUSINESS MANAGEMENT
- SENIOR IT/IS MANAGEMENT/C LEVEL EXECUTIVES
- ANALYSTS, CONSULTANTS

Hear these thought leaders in interactive, cutting-edge keynote addresses and panels...



JEAN-FRANCOIS ABRAMATIC
SENIOR VP R&D, ILOG
FORMER CHAIRMAN, W3C



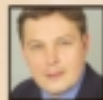
TYLER JEWELL
PRINCIPAL TECH
EVANGELIST • BEA



DAVID LITWAK
CEO
SILVERSTREAM



ANNE THOMAS MANES
CFO
SYSTEMS



BARRY MORRIS
CEO
IONA



RICK ROSS
FOUNDER
JAVALOBBY



PATRICIA SEYBOLD
FOUNDER & CEO
SEYBOLD

For Exhibit Information

CONTACT: RICHARD ANDERSON
115 CHESTNUT RIDGE RD.
MONTVILLE, NJ 08055
201 842-3054
RAND@SYS-CON.COM

XMLEDGE
conference & expo

JUNE 24-27

JACOB JAVITS
CONVENTION CENTER
NEW YORK, NY

OCTOBER 1-3

SAN JOSE
CONVENTION CENTER
SAN JOSE, CA

SPONSORED BY:

IONA | END 2 ANYWHERE™

ADOS

bea

TogetherSoft

SilverStream

Actional

MERANT

Altaworks

PolarLake
Enterprise Strength XML

ALTOVA

MEDIA SPONSORS

Federal Computer Week

WebServices.org

XML TIMES.com

Java Skyline

CE Advisor

WebServicesMall

WIRE

XML.org

WROX

XML.org

SDTimes

JAVA

wireless

XML

WebLogic

WebServices

WebSphere

COLDFUSION

OWNED AND PRODUCED BY

SYS-CON MEDIA | SYS-CON EVENTS

transformed using the transformation method (XSL, JSP, and tags) and an associated file specified in DTS.

Approaches to Updating the Database

Three declarative approaches are discussed in this section with the goal of simplifying the saving into and extracting from data sources while enhancing maintainability, that is, making it easy to change and enhance the application.

In the *simplistic approach* JavaScript submits the SVG XML file and individual data fields to be put in the database. The middle tier then inserts the entire XML file in a CLOB field of the database, and the individual data elements are updated into the database.

Following is an example of how this is done declaratively:

```
Submit URL = http://ip/servlets/UpdateServlet
?request_name=updateAnnotation
&layer_id=layerid
&xmlfile={...}
&field1=value1
&field2=value2
```

Entry into the DDS file for the request named updateAnnotation will be data source-specific.

For example, if the data source is Oracle, a call to the stored procedure updateAnnotation may look like this:

```
call updateAnnotation({xmlfile},{field1},{field2}).
```

For the *XPATH approach* we created a concept called Transparent Data Pipelines that has been used successfully to bridge the gap between the HTML pages and relational databases. In this approach an incoming HTML document is processed by calling a series of data access and update procedures in a transactionally secure manner. These procedures are declared in an XML/properties file. The necessary parameters for the procedures are culled from the incoming HTML form.

This approach is extended for incoming XML documents so parameter values can be culled

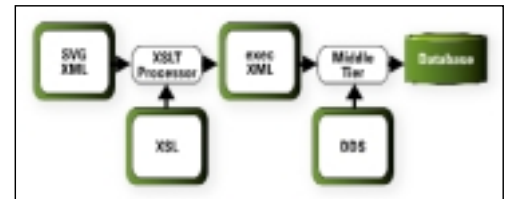


FIGURE 3 XSLT approach to updating AMP

directly from the XML document, using XPATH expressions. This gives the added flexibility to get at values that are hierarchical. Multiple repeated nodes can be accessed by specifying an XPATH repeated node attribute on the business procedure (in the DDS file) and repeating that procedure for each instance of the node.

One of the pros of this approach is its ability to call multiple procedures declaratively using the incoming XML document as input. In addition, XPATH allows for a rich set of selections for parameters. It is a declarative approach with no Java or JDBC code, which results in fewer bugs. It also provides transparent transport of XML data back and forth from data sources.

Some of the cons are that the XPATH approach can't cover all the ground that custom Java code can, complexity increases when multivalued items and conditional storage have to be taken into account, and error handling is less sophisticated.

We're exploring the *XSLT approach* in greater detail because it provides a complete approach to processing the incoming SVG XML file and updating all the data sources. This keeps the benefits of the previous approach while addressing some of the drawbacks. XSLT will be employed on the incoming SVG XML document to generate a set of procedures that will then be executed. XSLT (see Listing 3) will process the incoming SVG XML document and create an exec-XML (see Listing 4). The exec-XML contains all the procedures to be executed and arguments to these procedures. The overall process (diagrammed in Figure 3) is as follows:

1. A complete set of procedures is defined declaratively in the DDS file.
2. An XSL document is created for the incoming XML file.
3. Embedded in the XSL document are references to procedures specified in step 1.
4. When the XSL document is processed, it creates an exec-XML document (in-memory) containing the procedures to be called, along with their parameters.
5. This exec-XML document is processed and all procedures mentioned are executed to update the database with AMP.

While Figure 3 illustrates the process, the examples in the following listings provide details of the relevant inputs and outputs. Input XML is the floor plan example cited earlier. Listing 3 is an XSL transformational file that generates the "exec-XML" executable file in Listing 4.

The middle tier processes the exec-XML file and executes each of the execProc statements. RequestName from the execProc is used as a look-up key into the DDS. The remainder of the argument string is called execProcArgs. The code snippet in the next section is the structure and content of the DDS file that supports the execProcs.

DDS File

Data sources, update methods defined declaratively

Following is an example of the Oracle database specification in the DDS file:

JDJSTORE.COM GUARANTEED! LOWEST PRICES!

Guaranteed Best Prices

JDJ Store Guarantees the Best Prices. If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.

Terms of offer:

- Offer good through June 30, 2002
- Only applicable to pricing on current versions of software
- Offer does not apply toward errors in competitors' printed prices
- Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

Attention Software Vendors:

To include your product in JDJStore.com, please contact tony@sys-con.com

WEBGAIN

Visual Café Expert 4.5.1



JDJStore.com **\$899⁰⁰**



\$79⁰⁰

SYS-CON MEDIA
**WebServices Journal –
XML-Journal Resource CD**

INTERLINK

Remotepoint RF Wireless Handheld w/ Software



JDJStore.com **\$169⁹⁹**

INTERLINK

RemotePoint RF Combo Keyboard & Remote Control

Radio Frequency Wireless Technology - up to 100 feet of cordless freedom with no line-of-sight limitation. Streamlined full-featured keyboard with mouse. Integrated laser pointer.



RemotePoint RF Combo Keyboard & Remote **\$234⁹⁹**

MACROMEDIA

Homesite 5.0

Macromedia HomeSite 5 delivers advanced hand-coding features, productivity-enhancing wizards, and a customizable user interface to efficiently build, deploy and manage your entire Web site.



Homesite 5.0 **\$93⁹⁹**

MICROSOFT

Visual Studio .NET 2002 Professional

Visual Studio .NET enables developers to rapidly build next-generation Internet applications that target any device and integrate with any platform. By providing the most modern and feature-rich development environment, Visual Studio .NET gives developers the tools for integrating solutions across operating systems and languages.



Visual Studio .NET 2002 **\$948⁹⁹**

XML WHIZ

XML Certification Test Simulator and Tutorial

5 Mock Tests (285 Questions) - It comes with a test engine and a question bank of 285 questions. Not only you can test your knowledge of XML but also enhance it by going through the exhaustive explanation given for each and every question.



XML Certification test Simulator and Tutorial **\$49⁰⁰**

SYBASE

SQL Anywhere Studio (base with 1 user) v 7.0

Sybase® SQL Anywhere Studio is a comprehensive package that provides data management and enterprise synchronization to enable the rapid development and deployment of distributed e-Business solutions.



Visual Café Enterprise Suite **\$348⁹⁹**

KODAK

DX3900 Zoom Digital Camera

The DX3900 digital camera, with 3.1 megapixels gives you the performance you want, the quality you demand and the simplicity of EasyShare. You'll get excellent picture quality for prints up to 11" x 14". Its 6X zoom (2X optical and 3X digital) lets you get closer to your subject for crisp detail. If you use the KODAK EasyShare Camera Dock, (sold separately) sharing is just as easy.



DX3900 Zoom Digital Camera **\$349⁹⁹**

ORDER TODAY!

888-303-JAVA

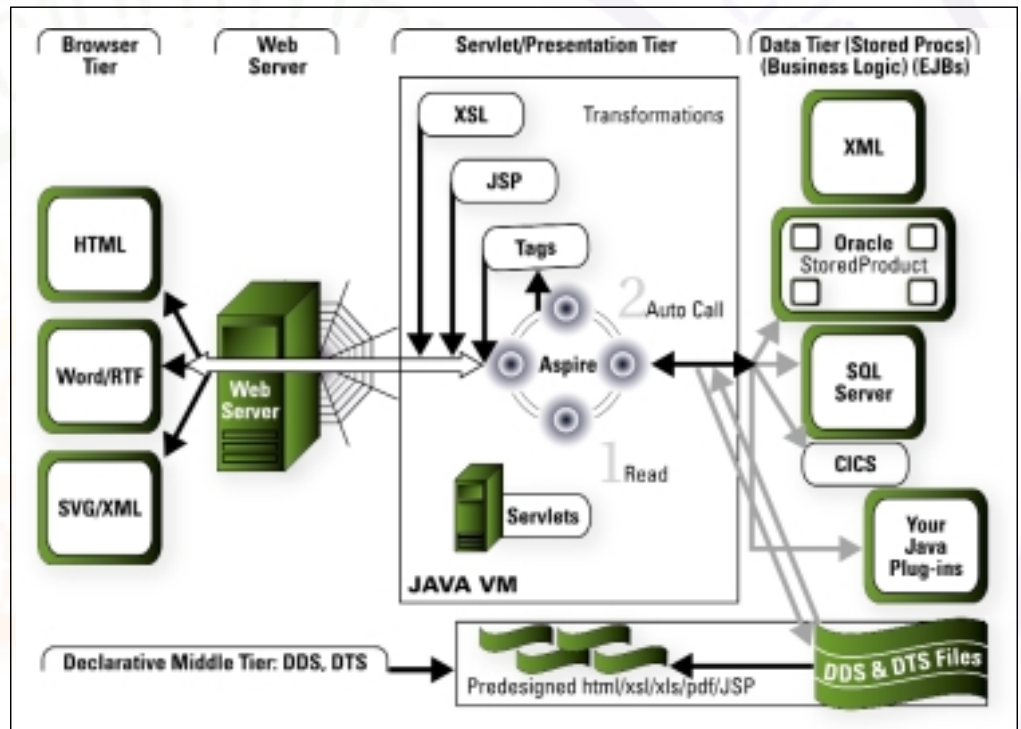


FIGURE 4 Aspire/J2EE single JVM high-level architecture

```
<!-- section 1 of dds.xml -->
<datasource>
  <name>MyDataBase</name>
  <jdbc_url>jdbc:oracle:127.0.0.1:1521@eawp</ip>
  <user>ME</user>
  <pass>ANY</pass>
</datasource>
```

This definition is used every time “MyDataBase” is referenced in code. Note that the DDS file contains the declarative specification of the data update method. In the example below, `sp_updateUse` is a stored procedure that will parse the arguments passed through `execProcArgs` and update the database.

```
<!--section 2 of dds.xml -->
<UpdateExec name="updateUse">
  <request>
    <javaclass>com.ai.db.DBStoredProc</javaclass>
    <datasource>MyDataBase</datasource>
    <stmt>call sp_updateUse(?,{execProcArgs})</stmt>
  </request>
</UpdateExec>
```

The entire `execXML` file is executed in a single transactional context.

It's possible to apply a variety of transactional modes to support the requirements of the transaction.

In summary, the approach presented here takes an SVG XML file submitted by the browser, applies an XSL transform to extract the data, and creates an execution XML file. This file is then run by the middle tier to update the database. XSL and stored procedure are the two programs that have to be created to accomplish this task.

Approaches to Creating SVG XML Files from Data

Creating SVG XML data from data sources follows the same declarative approach advocated above. The process is as follows:

1. A series of procedures in DDS is defined to retrieve the data.
2. The middle tier generates an internal XML document with a predefined structure.
3. A transformation (like XSLT, tags, or JSP) on this internal XML data structure is then used to arrive at the target SVG XML.

The three examples in Listings 5–7 illustrate the nature of transformations. The first two are tag based; the third, JSP based.

Template SVG file

The first example is of tag-based transformation. The input tag file is in Listing 5.

Compare the template file to the desired SVG file. All the drawing ele-

ments in the canvas have been replaced with a loop tag, loop1. Inside the loop tag is a replacement tag {{gElements}}. The middle tier transforms the template file by replacing loop1 and the {{gElements}} with rows of data obtained from data sources. In this example the entire XML code for the element is obtained from the database.

Next, examine the data access specifications to gather data from the data source mentioned above. The userid is passed in from the HTML page, which returns one row from the layer table with the layerid. This value is used in loop1 to get the entire SVG segment for a marker (see Listing 6).

An alternate method is to return individual data elements from the database and replace them in the template file. The template file in Listing 7 is much larger, with all the XML code (interspersed with replacement tags) for the drawing elements in the canvas. The associated DDS are similar to the one in Listing 6 but contain two loops.

JSP-Based Transformation

Programmers familiar with Java may want to use JSP to transform the retrieved data to SVG XML. The example in Listing 8 shows how this is accomplished. The JSP sections are highlighted for clarity. The associated DDS are the same as in the previous example.

The process is as follows:

1. Retrieve a Java interface representing the entire data set: IFormHandler.
2. Report any errors if this object is null.
3. Retrieve a Java interface representing a loop of data: IControlHandler3.
4. Iterate through the loop: gotoNextRow().
5. Retrieve values from the current row: getValue(key).
6. Complete the document.

XSLT to Generate the Target SVG-XML

The data set retrieved in the middle tier is XML, and the target SVG is XML. It's certainly straightforward to write XSLT to convert from one XML to the other. As a result, the XSLT sample code is not shown here.

Format of the Retrieved Data Set

The relationship between the tag elements and named procedures in the DDS is accomplished through a conceptual XML data model (info-set) of the retrieved data from named procedures. This XML data structure plays an important role in tags, XSLT, and JSP transformations. This well-defined data model is the cornerstone for allowing pluggable data sets in the back end from any sort of data source. The middle tier is responsible for generating and maintaining an efficient version of this data model and making it available for transformations (see Listing 9).

The data set is essentially a collection of key/value pairs and loops derived from the key/value pairs. The loops themselves are a collection of rows. Each row again contains a collection of key/value pairs and more derived loops.

Specification of Transformations to Create SVG

DTS is responsible for specifying which of the preceding transformations is actually used to obtain the target SVG XML. Listing 10 is one such example specification.

Note that "getMarkers" is the same name used in section 2 of dds.xml. After the middle tier creates an XML document using the getMarkers specification, DTS.xml transforms the XML using the getMarkers speci-

fication. As seen in Listing 10, depending on the Java class and the template transformation file (floorplanTemplate.svg), the source data set is transformed to the appropriate output (floorplan.svg).

General Architecture

In depicting how browsers are connected to the middle tier, and hence to the data sources via DTS (Transformations) and DDS (Data), Figure 4 puts in perspective how such a middle tier can be architected.

We borrowed this architecture from one of our implementations of it – Aspire, a J2EE/XML-based middle tier (for details see www.indent.org/aspire.htm). The architecture allows:

- Multiple output formats (HTML, XML, text)
- Multiple choices for transformations (JSP, XSL, tags)
- Use of standard J2EE principles
- Data gathering and transformation, specified declaratively
- Access to any data source through well-defined plug-ins

Summary

We've presented a declarative approach to creating SVG-based applications for creating, editing, and displaying graphics, and associating properties with the graphical elements. The approach is standards based, widely deployable, and doesn't require Java programming (when working with relational databases as the data source). All the data and transformation-related specifications are put in external files, and not hard-coded into the middle tier.

SVG References

- *SVG specifications*: www.w3c.org/TR/SVG/index.html.
- Watt, A.H. (2001). *Designing SVG Web Graphics*. New Riders Publishing.
- Eisenberg, J.D. (2002). *SVG Essentials*. O'Reilly XML.
- Jain, P. (2002). "Data-Driven SVG Apps: A Rapid Development Approach," February: www.onjava.com/pub/a/onjava/2002/02/13/svg.html.
- *SVG tutorials, sample programs, SVG Viewer, others*: www.adobe.com/svg.
- *PopCharts ImageServerPro*: www.corda.com. Commercial SVG-based product for graphs and charts; Web-based data-driven charting solution, using charting templates.
- Komatineni, S. (2001). "Applying Java/XML/XSLT Technology: Developing Web Applications Powered by Relational Databases." *XML-J*, Vol. 2, issues 4.5. Using TDP to convert relational data to XML data on the fly.
- —. "A JSP Architecture for Oracle Stored Procedures." *Java Report*, July. Stored procedures as plug-in adapters developed for declarative middle tier. ☛

AUTHOR BIO

Pramod Jain is founder (in 1995) and president of INDENT, Inc., which provides tools and consulting for rapid development of J2EE/XML solutions. He holds a PhD in mechanical engineering from the University of California, Berkeley.

Satya Komatineni, CTO of INDENT, is the author of a Java-based RAD framework for developing J2EE-based HTML applications. He earned an MS in electrical engineering from the Indian Institute of Technology, New Delhi.

P R A M O D @ I N D E N T . O R G
S A T Y A @ I N D E N T . O R G

LISTING 1 Sample SVG file with graphical and business properties

```
<?xml version="1.0"?>
<svg width="800" height="440">
  <g id="ball">
    <circle cx="200" cy="200" r="100"/>
  <g style="visibility:hidden" desc="ObjectProperty">
    <text>
      <tspan desc="weight">33.0 ounce</tspan>
      <tspan desc="material">rubber</tspan>
      <tspan desc="price">$0.50</tspan>
    </text>
  </g>
```

```
</g>
</svg>
```

LISTING 2 Example of a desired SVG file

```
<!-- floorplan.svg -->
<svg>
  <defs> <!--definitions of graphic elements that will be
reused-->
    <g id="power_def">
      <rect width="10" height="8" style="fill:red"/>
    </g>
    <g id="phone_def">
      <rect width="10" height="8" style="fill:blue"/>
    </g>
```



```

</g>
<g id="ethernet_def">
  <rect width="10" height="8" style="fill:green"/>
</g>
</defs>
<image src="/floorplan.jpg"/>
<g id="drawingCanvas"> <!--Start of drawing canvas -->
  <g id="power1" transform="translate(410 212)"> <!--element 1-->
    <use xlink:href="#power_def"/> <!--reuse of defs -->
    <g style="visibility:hidden;" desc="objectProperty">
      <text>
        <tspan desc="number">2</tspan>
      </text>
    </g>
  </g>
  <g id="phone1" transform="translate(430 212)"> <!--element 2-->
    <use xlink:href="#phone_def"/>
    <g style="visibility:hidden;" desc="objectProperty">
      <text>
        <tspan desc="number">1</tspan>
        <tspan desc="phone_number">904-555-1212</tspan>
      </text>
    </g>
  </g>
  <g id="ethernet1" transform="translate(450 212)">
    <!-- element 3-->
    <use xlink:href="#ethernet_def"/>
    <g style="visibility:hidden;" desc="objectProperty">
      <text>
        <tspan desc="number">1</tspan>
        <tspan desc="ip_address">192.168.1.5</tspan>
      </text>
    </g>
  </g>
  <g id="path_to_closet"> <!-- element 4-->
    <path d="M450,212H50V100H-100V100"/>
  </g>
</g>
</svg>

```

LISTING 3 XSL file for the floor plan example

```

<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl=
  "http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml"/>
<xsl:template match="/">
  <execXML>
<xsl:for-each select="/svg/g/g">
  <execProc>
<xsl:variable name="argStr">
  <xsl:for-each select="@*">
    <xsl:value-of select="name(.)"/>
    <xsl:text>=</xsl:text>
    <xsl:value-of select="."/>
    <xsl:text>&amp;&amp;</xsl:text>
  </xsl:for-each>
<xsl:for-each select="*">
  <xsl:if test="name(.)='use'">
    <xsl:text>requestName=updateUse</xsl:text>
    <xsl:for-each select="@*">
      <xsl:value-of select="name(.)"/>
      <xsl:text>=</xsl:text>
      <xsl:value-of select="."/>
      <xsl:text>&amp;&amp;</xsl:text>
    </xsl:for-each>
  </xsl:if>
  <xsl:if test="name(.)='path'">
    <xsl:text>requestName=updatePath</xsl:text>
    <xsl:for-each select="@*">
      <xsl:value-of select="name(.)"/>
      <xsl:text>=</xsl:text>
      <xsl:value-of select="."/>
      <xsl:text>&amp;&amp;</xsl:text>
    </xsl:for-each>
  </xsl:if>
  <xsl:if test="./@desc='objectProperty'">
    <xsl:for-each select="./text/tspan">
      <xsl:value-of select="@desc"/>
      <xsl:text>=</xsl:text>
      <xsl:value-of select="."/>
      <xsl:text>&amp;&amp;</xsl:text>
    </xsl:for-each>
  </xsl:if>
</xsl:for-each>
</xsl:variable>
<xsl:value-of select="$argStr"/>

```

```

</execProc>
</xsl:for-each>
</execXML>
</xsl:template>
</xsl:stylesheet>

```

LISTING 4 Output exec-XML file

```

<?xml version="1.0"?>
<execXML>
  <execProc>
requestName=updateUse&id=power1&transform=translate(410,212)
&href=#power_def&number=2
  </execProc>
  <execProc>
...
  </execProc>
  <execProc>
    requestName=updatePath&id=path_to_closet&d=M450,
      212H50V100H-100V100
  </execProc>
</execXML>

```

LISTING 5 Simple tag-based SVG XML template file

```

<!-- floorplanTemplate.svg Example 1-->
<svg>
<defs> <!--definitions of graphic elements that will be
  reused -->
...
</defs>
<g id="drawingCanvas"> <!--Start of drawing canvas -->
  <!--RLF-TAG BGN Loop loop1 -->
    {{gElements}}
  <!--RLF-TAG END Loop loop1 -->
</g>
</svg>

```

LISTING 6 DDS file for Listing 5

```

<!--section 2 of dds.xml -->
<dataset>
  <name>getMarkers</name>
  <maindataset>
    <request>
      <javaclass>com.ai.db.DBRequestExecutor1</javaclass>
      <datasource>MyDataBase</datasource>
      <stmt> select layerid from layers where user_id=
        {userid}</stmt>
    </request>
  </maindataset>
  <loopdata_request>
    <name>loop1</name>
    <request>
      <javaclass>com.ai.db.DBRequestExecutor1</javaclass>
      <datasource>MyDataBase</datasource>
      <stmt> select gelements from marker where layer_id =
        {layerid}
    </stmt>
    </request>
  </loopdata_request>
</dataset>

```

LISTING 7 Complex tag-based SVG XML template file

```

<!-- floorplan.svg Example 2 -->
<svg>
<defs> <!--definitions of graphic elements that will be reused-->
...
</defs>
<g id="drawingCanvas"> <!--Start of drawing canvas -->
  <!--RLF_TAG BGN_LOOP loop1 -->
    <g id="{{floor_id}}" transform="{{transform}}"> <!--elements-->
      <!--RLF_TAG BGN_IF svg_type=path if1-->
        <path d="{{path_string}}"/>
      <!--RLF_TAG END_IF svg_type=path if1 -->

      <!--RLF_TAG BGN_IF svg_type=use if2-->
        <use xlink:href="{{use_id}}"/>
      <!--RLF_TAG END_IF svg_type=use if2 -->

      <g style="visibility:hidden;" desc="objectProperty">
        <text>
          <!--RLF_TAG BGN_LOOP svg_text -->
            <tspan desc="{{name}}">{{value}}</tspan>
          <!--RLF_TAG END_LOOP svg_text -->
        </text>
      </g>
    <!--RLF_TAG END_LOOP loop1 -->

```

```
</g>
</g>
</svg>
```

LISTING 8 Retrieved data transformed via JSP to SVG XML

```
<!-- floorplan.svg -->
<%@ page import="com.ai.htmlgen.*" %>
<%@ page import="com.ai.application.utils.*" %>
<%
IFormHandler svgData =
(IFormHandler)request.getAttribute("Aspire.formHandler");
if (svgData == null)
{
// Report no data found and return
}
%>
<svg>
<defs> <!--definitions of graphic elements that will be
reused-->
...
</defs>
<g id="drawingCanvas"> <!--Start of drawing canvas -->
<% IControlHandler3 floorDef = (IControlHandler3)
svgData.getControlHandler("svg_def_floor_handler");
if (floorDef.isDataAvailable() == true)
{
while(floorDef.gotoNextRow() == true)
{
<g id="<%= floorDef.getValue("floor_id") %%"
transform="<%= floorDef.getValue("transform") %%">
<% if(floorDef.getValue("svg_type").equals("use"))
{
<use xlink:href="#<%=floorDef.getValue("use_id")
%"/>
<% }
else if(floorDef.getValue
("svg_type").equals("path"))
{ %>
<path d="<%= floorDef.getValue("path_string")
%"/>
<% } %>
}
```

```
<g style="visibility:hidden;" desc=
"objectProperty">
<text>
<tspan desc="number">
<%= floorDef.getValue("objectText") %></tspan>
</text>
</g>
</g>
<% } //end while %>
</g>
<% } //end if floorDef has data %>
</svg>
```

LISTING 9 Aspire-infosets: internal representation of data extracted from database

```
<!--Illustration of middle-tier XML output format for
example 1-->
<dataset>
<image_id>8829</ image_id>
<user_id>22</user_id>
<loop name="loop1">
<row>
<gelements><g id="power1"><use>...</use></g></gelements>
</row>
<row>
<gelements><g id="phone1"><use>...</use></g></gelements>
</row>
<row>
<gelements><g id="ethernet1"><use>...</use></g></gelements>
</row>
</loop>
</dataset>
```

LISTING 10 DTS file for Listing 5

```
<!--section 3 of dts.xml -->
<target_dataset>
<source_dataset>
getMarkers
</source_dataset>
<transform>
<javaclass>com.ai.xml.TagTransform</javaclass>
<templatefile>floorplanTemplate.svg</templatefile>
</transform>
</target_dataset>
```

DOWNLOAD THE CODE @
www.sys-con.com/xml

SYS-CON Media, the world's leading publisher of i-technology magazines for developers, software architects, and e-commerce professionals, brings you the most comprehensive coverage of WebSphere.



WebSphereDevelopersJournal.com

WebSphere
DEVELOPER'S JOURNAL



**Introductory
Charter Subscription**

**SUBSCRIBE NOW AND SAVE \$31.00
OFF THE ANNUAL NEWSSTAND RATE**

ONLY \$149 FOR 1 YEAR (12 ISSUES) REGULAR RATE \$180

OFFER SUBJECT TO CHANGE WITHOUT NOTICE

**Do You Have
Access to the
Internet?**

**Then
Subscribe
Online and
Save \$31!**

It's that easy

The
World's
Leading
Independent
WebSphere
Developer
Resource



Doing it automatically is what makes users happy

Generating Preformatted Reports in Excel

It's quite common for developers to present data from the database (or elsewhere) for analysis by more than one team in an organization. Over the years, Microsoft's Excel has been one of the preferred tools for spreadsheet-type reports. Now, in combination with XML technologies, we can control the formatting of the spreadsheets.

With Web-enabled applications so popular, it's customary to give end users the ability to generate the reports themselves. This article discusses the automatic generation of preformatted reports. We'll build an application that has a menu of the available reports or report-generation criteria and the ability to generate preformatted reports in Excel using XML and XSLT.

Interim Solution

Given that Excel can read from a character-delimited file, we could pull out the data from the database and save it in a TAB DELIMITED text file with a .xls extension. The file could be saved on a Web server and we could provide a link to it for any interested teams. This would enable the teams to get the reports whenever they want, but this solution presents some serious concerns.

Drawbacks of the Interim Solution

For example, the report isn't preformatted. The column widths are the same, so the user has to change the widths manually to get a better view. Features that could be incorporated are missing, such as titles in bold, important data marked in different colors, text wrapping for bigger text data, headers, and footers.

In essence, it is simply unformatted data that could be viewed in an Excel sheet. The end user has to go through all the formatting before the report can be printed. And since these reports have to be dynamic, the formatting process has to be repeated every time the reports are generated.

Another drawback is that all the teams see the data in the same format. If one of the teams expects the data in a different

format, they have to change it manually or we have to create different applications to suit the different requirements.

XML Feature of Excel 2002 (Office XP)

Office XP includes Excel 2002, which provides support for XML. In other words, Excel 2002 can save data as an XML document as well as read data from an XML document.

But Excel can't read from or write to just any XML document; it requires the XML data in a specific format – XML spreadsheet format.

How Can We Use This Feature?

First we get the raw data for the report as an XML document. We then create an XSL stylesheet to convert the data to another XML document (the XML spreadsheet format) that can be read by Excel 2002. Our report is now an XML document in Excel's XML spreadsheet format, and will have all the data and related formatting information.

In order to do this we have to understand Excel's XML spreadsheet format. Let's take a look at some basic techniques that Excel uses to save the data and formatting information in XML spreadsheet format.

How Excel 2002 Saves/Reads Data from XML

Excel 2002 saves the data using the XML Spreadsheet Schema (XML-SS). So an Excel document that's saved as an XML spreadsheet has all its data and formatting information stored as XML elements that use the XML-SS.

The <Workbook> element is the parent of all. It contains the <Worksheet> elements for each worksheet. Each sheet has an associated <Table> element. Within the table each row is stored as a <Row> element, the cell is stored as a <Cell> element, the data in the cell is stored as a <Data> element, and so on. This structure takes care of storing all the data.

To store the formatting information, Excel's XML spreadsheet uses the <Style> element. A unique ID is generated for every unique combination of formatting information. The styling information is stored within this <Style> element. This style ID is referenced whenever this particular type of formatting is required.

Here's an example of a style element having the following formatting characteristics:

- Alignment set to Center
- Text size set to 16
- Font set to Courier
- Text style set to Bold

```
<Style ss:ID="s26">
  <Alignment ss:Horizontal=
    "Center" ss:Vertical="Bottom" />
  <Font ss:FontName="Courier" x:Family=
    "Modern" ss:Size="16" ss:Bold="1" />
</Style>
```

Note that this particular style element has its ID set to s26.

Whenever we require the formatting characteristics listed above, a reference will be made to the style element shown above. Here is an example of the usage (note the reference to StyleID s26):

```
<Cell ss:Index="2" ss:StyleID="s26">
  <Data ss:Type="String">Report Title
</Data>
</Cell>
```

AUTHOR BIO

Prasad Joshi is a Syntel, Inc., consultant working for Nekema, Inc., based in New Jersey. He has some four years of experience in designing and implementing Web-based applications using Java and XML technologies. Previously he worked on Microsoft technologies such as ASP and Visual Basic.

Simplex Knowledge

www.skc.com

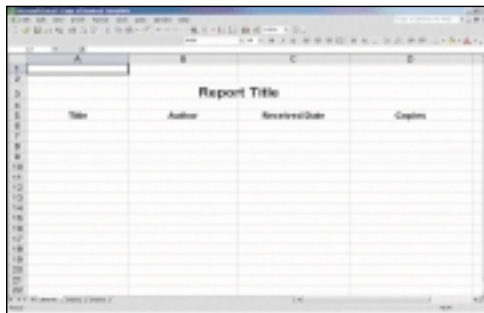


FIGURE 1 Report template in Excel 2002

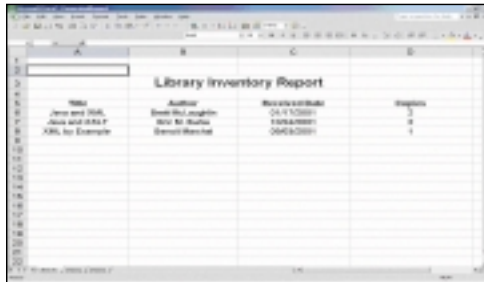


FIGURE 2 Generated preformatted report

As a result of this, the data Report Title will be displayed on the Excel sheet with the styling information listed earlier.

While we can create the `<Style>` elements ourselves, an easier option would be to save an Excel sheet with the desired formatting as an XML spreadsheet and pick up the `<Style>` elements from there. We'll discuss this trick in detail later.

Thus the `<Style>` elements take care of the formatting information.

The Excel sheet in the XML spreadsheet format is basically data as a value of specific XML elements; the attributes of XML elements store the formatting information.

Other features of Excel are stored in the XML spreadsheet format using a similar technique involving various other XML elements.

To get more information about how Excel saves the data in an XML spreadsheet format, please follow the link to the Microsoft XML Spreadsheet Schema given in the "References" section.

I'll illustrate the report-generation process with a simple example of a preformatted library inventory report.

Desired Excel Template

First we create a template for our report. The template helps us to know about the expected formatting requirements (see Figure 1).

Raw Data in XML

As mentioned earlier, the raw data has to be available in an XML document because we want to convert it to another XML document via XSLT. This raw data in XML format is the input to the XSLT process.

If the raw data is in a database or file, it can be converted to XML using various techniques. One of the techniques would be to use Java and DOM, but that is beyond the scope of this discussion. In this case we'll just assume that the data is available to us in the XML format.

The data in our example is a short list of books that are available in a fictitious library. Listing 1 shows the data in XML.

Creating the XSL

Creating the XSL is the major task in our process. We have to convert the data in the XML document to another XML document that Excel can read from. This new XML document will have all the data along with the formatting information. We achieve this with XSLT (XSL Transformation).

For this we have to create an XSL document, a onetime process. Once created, the XSL can be used to process any data in the XML document. (Please follow the links in the "References" section for related information on XSLT.) A lot of tools are available to generate the XSL documents. I used XML Spy.

In the XSL document we use XPath to navigate to various parts of the input XML document, that is, to locate a particular element or attribute in the input XML document. Please follow the links in the "References" section for related information on XPath.

Listing 2 gives part (template) of the XSL document. (The entire document can be found in the final code listing.)

Let's take a look at the XSL template. The first `<Row>` element in this XSL is for the title of the report. We associate the ReportTitle attribute of the Report element in the input XML document with this row. This association will display the value of the ReportTitle attribute in that particular cell. Note that we start from the row with Index=3, which gives us two blank rows in the beginning.

Note also the use of the `ss:MergeAcross` attribute; this is used to represent merging cells. You can learn more about the Microsoft Spreadsheet Schema (XML-SS) by following the links in the "References" section.

Moving along, the next `<Row>` element with Index=5 is for the column headings. After the column headings row, we display the contents of each `<Record>` element from the input XML document.

Note the use of different StyleIDs associated with the `<Cell>` elements. A simple trick will get all these `<Style>` elements created for you automatically.

The Trick

The Excel 2002 report template with all our desired formatting features can be

created and saved in the XML spreadsheet format. It will have `<Style>` elements (with unique style IDs) for all the unique combinations of formatting requirements. We can now use these `<Style>` elements in our XSL stylesheet. All we have to do then is to associate a particular `<Cell>` element in the XSL stylesheet with the appropriate `<Style>` elements.

This procedure can also be used to learn how several other features of Excel are stored in the XML spreadsheet format and we can use them in our XSL stylesheet.

Getting the Formatted Report (XSL Transformation)

Once the XSL is ready, we have to perform XSL transformation in which the XSL document is used to process the input XML document and create another XML document as output. There are a number of XSL processors available to do this. I used IBM's LotusXSL processor.

Listing 3 is a very simple Java code used to achieve the transformation. It accepts the XML and XSL file names as inputs and generates a new XML document as output. This output file can be saved as an XML document and can be opened with Excel 2002.

We can save the generated XML file on the Web server (with a .xls extension) and provide a link to this file for users. Figure 2 is how the report looks when opened in Excel 2002. (Please refer to "Issues and Possible Workarounds" if you want to view the report in other versions of Excel.)

Possible Enhancements

My example illustrates the use of very basic formatting features. Other features of Excel – headers, footers, page numbers, page breaks, formulas, inserting current date, etc. – could be included as well. Using the trick discussed earlier, we can include these features in our XSL document.

We can also have different XSL sheets to present the same data in different formats. For example, one team might not want the "ReceivedDate" column in their report, so what we do is to create a different XSL sheet and that team would get a report in another format. Thus we can give a choice of format on our report-generation menu page.

Issues and Possible Workarounds

If we Web-enabled this report application (so the end user could open it in a browser), then all the users should have Office XP (Excel 2002) installed to view it. Since we can't force them to do this, a possible workaround would be to save the document to an earlier Excel version and

then provide a link to the report. This could be done easily with Visual Basic 6.0 (using the Excel 2002 Objects). All we have to do here is execute a simple Save As (old format) command. In this case we'd have to install Office XP only on the server.

References

- Microsoft XML Spreadsheet Schema(XML:SS) reference: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnexcel2k2/html/odc_xmlss.asp
- XSL transformations: www.w3.org/TR/xslt
- XPath: www.w3.org/TR/xpath
- LotusXSL processor: www.alphaworks.ibm.com
- XML Spy: www.xmlspy.com/

P R A S A D V J O S H I @ H O T M A I L . C O M

LISTING 1 XML data

```
<?xml version="1.0" encoding="UTF-8"?>
<Report ReportTitle="Library Inventory Report">
  <Record>
    <Title>Java and XML</Title>
    <Author>Brett McLaughlin</Author>
    <ReceivedDate>01/17/2001</ReceivedDate>
    <Copies>2</Copies>
  </Record>
  <Record>
    <Title>Java and XSLT</Title>
    <Author>Eric M. Burke</Author>
    <ReceivedDate>10/24/2001</ReceivedDate>
    <Copies>2</Copies>
  </Record>
  <Record>
    <Title>XML by Example </Title>
    <Author>Benoit Marchal</Author>
    <ReceivedDate>09/03/2001</ReceivedDate>
    <Copies>1</Copies>
  </Record>
</Report>
```

LISTING 2 XSL stylesheet

```
<xsl:template match="Report">
  <Row ss:Index="3" ss:AutoFitHeight="0" ss:Height="20.25">
    <Cell ss:Index="2" ss:MergeAcross="1" ss:StyleID="s22">
      <Data ss:Type="String"><xsl:value-of select=
        "@ReportTitle"/></Data>
    </Cell>
  </Row>
  <Row ss:Index="5">
    <Cell ss:StyleID="s23">
      <Data ss:Type="String">Title</Data>
    </Cell>
    <Cell ss:StyleID="s23">
      <Data ss:Type="String">Author</Data>
    </Cell>
    <Cell ss:StyleID="s23">
      <Data ss:Type="String">Received Date</Data>
    </Cell>
    <Cell ss:StyleID="s23">
      <Data ss:Type="String">Copies</Data>
    </Cell>
  </Row>

  <xsl:for-each select="Record">
    <Row>
      <Cell ss:StyleID="s24">
        <Data ss:Type="String"><xsl:value-of select="Title"/></Data>
      </Cell>
      <Cell ss:StyleID="s24">
        <Data ss:Type="String"><xsl:value-of select=
          "Author"/></Data>
      </Cell>
      <Cell ss:StyleID="s24">
        <Data ss:Type="String"><xsl:value-of select=
          "ReceivedDate"/></Data>
      </Cell>
      <Cell ss:StyleID="s24">
        <Data ss:Type="Number"><xsl:value-of select=
          "Copies"/></Data>
      </Cell>
    </Row>
  </xsl:for-each>
</xsl:template>
```

Once you're in it...



...reprint it!

- Wireless Business & Technology
- Java Developer's Journal
- XML Journal
- ColdFusion Developer's Journal
- PowerBuilder Developer's Journal

Contact Carrie Gebert
201 802-3026
carrieg@sys-con.com



Re Prints

LISTING 3 Java code for XSL transformation

```
import org.xml.sax.SAXException;
import org.apache.xalan.xslt.*;

public class XSLProcessor
{
    public static void main(String[] args)
    {
        try
        {
            // create an instance of the XSLT processor.
            XSLTProcessor processor = XSLTProcessorFactory.getProcessor();
            // perform the XSL transformation giving the "XML file name"
            // and "XSL file name" as input
            processor.process
            (new XSLTInputSource(new
            java.io.FileInputStream(args[0])),
            new XSLTInputSource(new java.io.FileInputStream
            (args[1])),
            new XSLTResultTarget(System.out));
        }
        catch( Exception e)
        {
            System.out.println("Exception in XSL processing" + e);
        }
    }
}
```

LISTING 4 Entire XSL stylesheet

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform" xmlns:fo=
"http://www.w3.org/1999/XSL/Format" xmlns:x="urn:schemas-
microsoft-com:office:excel"
xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet">

<xsl:output method="xml"/>
<xsl:template match="/">

<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
xmlns:o="urn:schemas-microsoft-com:office:office"
xmlns:x="urn:schemas-microsoft-com:office:excel"
xmlns:ss="urn:schemas-microsoft-com:office:spreadsheet"
xmlns:html="http://www.w3.org/TR/REC-html40">

<DocumentProperties xmlns="urn:schemas-microsoft-com:
office:office">
<Author>Prasad</Author>
<LastAuthor>Prasad</LastAuthor>
<Created>2001-12-24T15:03:24Z</Created>
<Company></Company>
<Version>10.2625</Version>
</DocumentProperties>
<OfficeDocumentSettings xmlns="urn:schemas-microsoft-com:
office:office">
<DownloadComponents/>
<LocationOfComponents HRef="file:///D:/"/>
</OfficeDocumentSettings>
<ExcelWorkbook xmlns="urn:schemas-microsoft-com:
office:excel">
<WindowHeight>8835</WindowHeight>
<WindowWidth>11340</WindowWidth>
<WindowTopX>480</WindowTopX>
<WindowTopY>120</WindowTopY>
<ProtectStructure>False</ProtectStructure>
<ProtectWindows>False</ProtectWindows>
</ExcelWorkbook>
<Styles>
<Style ss:ID="Default" ss:Name="Normal">
<Alignment ss:Vertical="Bottom"/>
<Borders/>
<Font/>
<Interior/>
<NumberFormat/>
<Protection/>
</Style>
<Style ss:ID="s22">
<Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
<Font x:Family="Swiss" ss:Size="16" ss:Bold="1"/>
</Style>
<Style ss:ID="s23">
<Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
<Font x:Family="Swiss" ss:Bold="1"/>
</Style>
<Style ss:ID="s24">
<Alignment ss:Horizontal="Center" ss:Vertical="Bottom"/>
<Font x:Family="Swiss"/>
</Style>
</Styles>
<Worksheet ss:Name="Sheet1">
```

```
<Table ss:ExpandedColumnCount="4" ss:ExpandedRowCount=
"100" x:FullColumns="1" x:FullRows="1">
<Column ss:AutoFitWidth="0" ss:Width="111.75"/>
<Column ss:AutoFitWidth="0" ss:Width="120"/>
<Column ss:AutoFitWidth="0" ss:Width="121.5"/>
<Column ss:AutoFitWidth="0" ss:Width="135"/>
```

```
<xsl:apply-templates/>
```

```
</Table>
<WorksheetOptions xmlns="urn:schemas-microsoft-com:
office:excel">
<Print>
<ValidPrinterInfo/>
<HorizontalResolution>600</HorizontalResolution>
<VerticalResolution>600</VerticalResolution>
</Print>
<Selected/>
<Panes>
<Pane>
<Number>3</Number>
<ActiveRow>4</ActiveRow>
<ActiveCol>2</ActiveCol>
</Pane>
</Panes>
<ProtectObjects>False</ProtectObjects>
<ProtectScenarios>False</ProtectScenarios>
</WorksheetOptions>
</Worksheet>
<Worksheet ss:Name="Sheet2">
<WorksheetOptions xmlns="urn:schemas-microsoft-com:
office:excel">
<ProtectObjects>False</ProtectObjects>
<ProtectScenarios>False</ProtectScenarios>
</WorksheetOptions>
</Worksheet>
<Worksheet ss:Name="Sheet3">
<WorksheetOptions xmlns="urn:schemas-microsoft-com:
office:excel">
<ProtectObjects>False</ProtectObjects>
<ProtectScenarios>False</ProtectScenarios>
</WorksheetOptions>
</Worksheet>
</Workbook>
```

```
</xsl:template>
```

```
<xsl:template match="Report">
<Row ss:Index="3" ss:AutoFitHeight="0" ss:Height="20.25">
<Cell ss:Index="2" ss:MergeAcross="1" ss:StyleID="s22">
<Data ss:Type="String"><xsl:value-of select=
"ReportTitle"/></Data>
</Cell>
</Row>
<Row ss:Index="5">
<Cell ss:StyleID="s23">
<Data ss:Type="String">Title</Data>
</Cell>
<Cell ss:StyleID="s23">
<Data ss:Type="String">Author</Data>
</Cell>
<Cell ss:StyleID="s23">
<Data ss:Type="String">Received Date</Data>
</Cell>
<Cell ss:StyleID="s23">
<Data ss:Type="String">Copies</Data>
</Cell>
</Row>
<xsl:for-each select="Record">
<Row>
<Cell ss:StyleID="s24">
<Data ss:Type="String"><xsl:value-of select=
"Title"/></Data>
</Cell>
<Cell ss:StyleID="s24">
<Data ss:Type="String"><xsl:value-of select=
"Author"/></Data>
</Cell>
<Cell ss:StyleID="s24">
<Data ss:Type="String"><xsl:value-of select=
"ReceivedDate"/></Data>
</Cell>
<Cell ss:StyleID="s24">
<Data ss:Type="Number"><xsl:value-of select=
"Copies"/></Data>
</Cell>
</Row>
</xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

Dynamic Buyer

www.ibm.com/smallbusiness/dynamicbuyer

WebServices
JOURNAL

XML JOURNAL

THE FIRST & ONLY
WEB SERVICES
RESOURCE CD

WEB SERVICES RESOURCE CD

THE SECRETS OF THE WEB SERVICES MASTERS

INCLUDES EXCLUSIVE .NET ARTICLES

MORE THAN

400
EXCLUSIVE

WEB SERVICES
& XML
ARTICLES



EDITED BY
SEAN RHODY



EVERY ISSUE OF WSJ & XML-J EVER PUBLISHED

THE MOST COMPLETE LIBRARY OF EXCLUSIVE WSJ & XML-J ARTICLES ON ONE CD!

"The Secrets of the Web Services Masters"

CD is edited by well-known WSJ Editor-in-Chief
Sean Rhody and organized into more than 40 chapters
containing more than 400 exclusive WSJ & XML-J articles.

Easy-to-navigate HTML format!

Bonus:

Full .PDF versions of every WSJ & XML-J published
since the first issue

XML in Transit
XML B2B
Java & XML
The XML Files
XML & WML
Voice XML
SYS-CON Radio
XML & XSLT
XML & XSL
XML & XHTML
2B or Not 2B
XML Script

XML Industry
Insider
<e-BizML>
XML & Business
XML Demystified
XML &
E-Commerce
XML Middleware
XML Modeling
CORBA & XML
Data Transition
XML @ Work

XML &
Databases
Electronic Data
Interchange
Ubiquitous
Computing
Information
Management
Objects & XML
XML Pros & Cons
Java Servlets
XML Filter

UML
Integration
WSDL
Beginning
Web Services
Web Services
Tips & Techniques
Frameworks
Management
Security
UDDI
.NET

3
YEARS
25
ISSUES
400
ARTICLES
ONE CD



Now
Shipping

\$79

ONLINE
ORDER AT
JDJSTORE.COM
SAVE
\$40



What does the future hold?

The RosettaNet Standard

This month's focus is on RosettaNet-related questions. As my company is a serious player in this market, I've received quite a few inquiries about the so-called lingua franca of supply chain software. I'd like to alert my readers, however, that many of the issues discussed stretch way beyond the smaller domain of RosettaNet and deep into the broader ocean we call the XML marketplace.

Let me begin with a quick paragraph on RosettaNet and then I'll respond to questions concerning schemas, DTDs, and all things PIP (Partner Interface Processes) related.

RosettaNet is a consortium of roughly 400 large supply chain companies from three major industries (semiconductor manufacturing, electronic components, and information technology). Their goal is to "create and implement industry-wide, open e-business process standards. These standards form a common e-business language, aligning processes between supply chain partners on a global basis" (from the RosettaNet Web site at www.rosettanet.org/rosettanet/Rooms/DisplayPages/LayoutInitial).

In a nutshell, RosettaNet is striving to create a common set of standardized processes for the electronic sharing of information over the Internet or other electronic medium. Now, on to the questions...

AUTHOR BIO

Trace Galloway is the chief evangelist for Infoteria Corporation (www.infoteria.com), an XML software development company based in Beverly, MA. Trace has an MCSE plus Internet certification from Microsoft and was a contributor to the third edition of the XML Handbook. An expert in XML and XSLT technologies, he specializes in RosettaNet B2B implementations.

Q: HOW DO YOU THINK PIPS WILL EVOLVE?

A: PIPs are where the RosettaNet standard meets the road. PIPs are the implementable building blocks of RosettaNet-based transactions. In its simplest form a PIP represents the electronic version of a traditionally paper-based transaction. For example, purchase orders have traditionally existed as paper-based documents either mailed or faxed directly from buyer to seller. PIP 3A4 is an XML-

based representation of that same paper document in electronic format. PIP purchase orders can be sent electronically over HTTP, SMTP, FTP, and so on.

One thing I'm certain of is that PIPs are here to stay. Since PIPs represent the most implementable part of any RosettaNet system, replacing them entirely is out of the question. What's going to happen in the future is that the specifications used to define a PIP are going to change.

Currently, PIPs are defined using two very different documents, and in some cases multiple documents, as is the case with PIP 2A9. For this discussion, however, I'll focus only on the two documents. The first is a DTD (Document Type Definition). DTDs are older carry-over documents that were prevalent in the SGML world and have been quite useful in the newer XML world. The PIP DTD defines the overall document structure of the PIP XML document as well as the cardinality, hierarchy, and mandatory/nonmandatory elements that make up the PIP. A DTD is similar to an XML-formatted document.

The second document, the Message Guideline (MGL), is an HTML file that goes one step further in defining our PIP document. The MGL defines not only structure, but also the allowable values for given elements within the PIP document. For example, in PIP 3A4 R02.00 PO Request, the DTD defines an element called "GlobalPartnerRoleClassificationCode" as being a mandatory child element of "PartnerRoleDescription". In addition to knowing that we

must include that element in our PIP 3A4 document, the MGL also defines the allowable value(s) for that element. In the case of "GlobalPartnerRoleClassificationCode" for PIP 3A4, the only allowable value is "Buyer". In this way the DTD and the MGL together play an important part in the overall makeup of a PIP document.

In the future, the DTD and the MGL will be replaced by a single XML Schema (XSD) document that will offer advantages over the current DTD and MGL in several areas. A single XSD file will provide solution providers and systems integrators with a single source to define the PIP. An XSD file is machine readable, which will increase the time necessary for solution providers to support new PIP standards. Finally, the current DTD and MGL model presents some challenges when there are discrepancies between the DTD and the MGL.

Because the MGL file isn't machine readable, many vendors simply use the DTD as their authoritative source for the definition of the PIP. The drawback to that approach is that when discrepancies occur, RosettaNet has stated that the MGL should be referenced as the authoritative document. The move to XSD will eliminate this problem.

Note: In addition to the DTD and the MGL, RosettaNet also defines several local and global dictionaries for use with PIPs. These documents are important, but a full discussion of their value and future is outside the scope of this article.

Q: HOW HARD ARE PIPS TO WORK WITH?

A: In my experiences so far, the PIPs have been the easiest parts of an implementation.

Most of the implementations we've done to date have involved the RosettaNet Basics PIPs (PIP 3A4, 3A7, 3A8, 3A9, and 3B2). In some cases – PIP 3A4, for example – multiple versions of the same PIP are available from RosettaNet. The partner companies we most often work with are connecting into an already established RosettaNet business process at their trading partner. The partner with the existing RosettaNet infrastructure usually dictates which version of the PIP they'd like to use. Once the PIP and PIP version have been identified, the real integration work begins. The PIP message itself is simply an XML-formatted document that provides placeholders for your business data.

Most business data resides in some type of relational back-end system like MS SQL or Oracle. The process of "mapping" your data from the back-end system to the PIP document is often the most time-consuming part of any integration effort. Tools and SDKs available from a multitude of vendors help to reduce this time.

Q: IS THE TECHNOLOGY BEHIND ROSETTANET SOLID?

A: In my opinion, absolutely. When you think about the RosettaNet standard, it's best to think of it in two parts. Part 1 is the PIP. The defining specifications for PIPs are DTDs and MGLs, with XSD specifications just around the corner. XML has now been in the marketplace for four years and has gained widespread adoption. Any specifications leveraging XML as their underpinning, especially data exchange specifications, are certainly headed in the right direction. The movement from the legacy DTD format to the newer, pure XML-based XSD format is also a clear indication of RosettaNet's forward-thinking vision.

Part 2 is RNIF (RosettaNet Implementation Framework). RNIF leverages several existing standards, including HTTP, HTTPS, SMTP, MIME, XML, and SOAP (coming in RNIF 3.0). In addition, because the RosettaNet architecture is decoupled into PIP and RNIF components, the replacement of RNIF with an equivalent ebXML or other standard is relatively easy and straightforward. RNIF-level services can even be augmented by services provided by other frameworks (RNIF 3.0 will take advantage of ebXML's

BPSS – Business Process Specification Schema). When you combine XML, HTTP, MIME, and SOAP with the ability to incorporate emerging standards into the existing framework, you've built one very solid foundation for the future.

Q: HOW HARD IS ROSETTANET TO IMPLEMENT? CAN YOU GIVE US SOME ISSUES TO BE PREPARED FOR?

A: Conducting RosettaNet-based transactions between two or more companies used to be a very challenging endeavor. In some instances it could take up to six months for a trading partner to implement just one PIP. With the rapid adoption of RosettaNet by many large supply chain companies, and with the steady improvements of B2B software offerings, implementing RosettaNet-based transactions has become easier than ever before. Nonetheless, there are still challenges to be overcome:

- **Interoperability:** Solution providers who currently provide RosettaNet functionality in their offerings have come together and begun an initiative, together with RosettaNet, to perform interoperability trials. These trials will ensure that each vendor's RosettaNet implementation can communicate with every other vendor's solution at the RNIF level. In addition, the RosettaNet compliance program is aimed at providing interoperability at the PIP level. Together, these two programs will help to ensure that the B2B software solution you choose can communicate efficiently with your trading partner's system.

- **Trading Partner Agreement:** Most companies that transact business with partners have some form of TPA currently in place. Anything from a handshake to a legally binding document represents an agreement between companies as to the particulars of their business interactions. An electronic form of this TPA is required for RosettaNet-based transactions. In addition to the specific roles that companies define for their interactions (buyer, seller, shipper, etc.), other RosettaNet-specific details must be addressed in the electronic TPA.

For example, RosettaNet transactions define default values for Time to Acknowledge and Time to Perform operations. The former represents the time period that you will allow your partner to respond to a message you've sent. The default is two hours for most PIPs, but you and your partner may decide that the pace of your business interactions is time sensitive and that you need to reduce that time so the transactions meet your business needs.

In addition to these settings, you and your partner also need to define the communication protocol (HTTP, HTTPS, SMTP) and other variables for your electronic TPA to be complete. Time spent ahead of time defining these parameters will greatly reduce your time to implement.

- **Partner's Implementation:** An understanding of the RosettaNet B2B solution that your partner uses, as well as an understanding of their internal processes, is often of great value. While the interoperability trials I mentioned earlier are still going on, many solution providers are compiling their own internal DBs of known

issues with certain vendors. The ability to tap into that knowledge will help to reduce integration efforts significantly.

Understanding how your partner does business from a process standpoint is also important. Do invoices normally take two to three business days to process? Do they typically respond to purchase order requests the same day? Given that most companies already have an existing relationship in place, this step should be fairly easy. The more knowledge you have about how the process works today, the better off you'll be when the time comes to do your RosettaNet implementation. ☺





Reusability and modularity are key components for VoiceXML adoption

Modular Speech Application Development Using VoiceXML

One thing we've learned from Web-based application development is that tools are useful only if they can reuse components and third-party libraries and make it easy to assemble applications. This article reviews how we can build modular speech applications using VoiceXML. The focus will be on the language constructs that VoiceXML provides for modularization and reusability and on vendor-specific approaches toward creation of a library of reusable dialogs for speech applications.

As a language, VoiceXML is designed for reusability and modularity. Similar to the Web paradigm, VoiceXML supports modularization of the application components through document-to-document navigation (through menus, form submits, subdialogs etc.). The "src" attribute associated with a number of VoiceXML elements allows URL-based loosely coupled integration.

Since we're used to a hyperlinked world today, we take this for granted, but if we compare it with speech applications developed earlier, it's really like comparing a huge compiled application with a loosely coupled yet integrated application. We're used to separating Web application functionality into multiple sets of scripts (Perl scripts, PHP pages, ASP.NET pages, JavaServer Pages), images, and other media. This capability has been leveraged by VoiceXML as the language supports division of applications into VoiceXML documents (which can be dynamically generated through

server-side programming), grammars, prompts, and subdialogs.

However, we'd be kidding ourselves if we didn't acknowledge that development of rich, high-quality speech applications is complex. VoiceXML does make it really simple to assemble a speech application, but we still need to create complex grammars and dialogs that can handle the complex conversations that the application needs to support.

This is where we can utilize the skills available within the speech recognition industry in the form of dialog and grammar experts. And this is where reusable components come to the rescue. Reusable components represent best practices and frequently used dialogs/grammars that can be used by an application developer (or, rather, an assembler) to create a high-quality speech application.

For instance, if we were to develop a stock-trading application, we'd need the ability to recognize the various compa-

nies traded on the stock exchange. If application developers were to include this capability as part of their own applications, they wouldn't be able to focus on the end application and would be lost in the functionality of recognizing all the possible companies. On the other hand, if a third party has already built such a dialog/grammar, it can be readily used, allowing developers to focus on the valued-added business scenarios rather than dealing with the complexity of recognizing all the companies.

Reusability Requirements

As speech application developers, what we need is a set of reusable dialogs and grammars that we can incorporate in our applications. We need grammars to recognize city names, streets, companies, businesses airlines, and airports as well as dialogs that use these complex grammars to recognize addresses, driving directions, credit card information, and the like.

In a nutshell, we need a set of published, reusable voice components. Apart from the components themselves, we need a methodology for creating and utilizing reusable components. Also required is the integration of reusable

AUTHOR BIO

Hitesh Seth is chief technology evangelist for Silverline Technologies, a global e-business and mobile solutions consulting and integration services firm. He has extensive experience in the technologies associated with Internet application development. Hitesh holds a bachelor's degree from the Indian Institute of Technology Kanpur (IITK), India.



components with development tools and VoiceXML gateways and/or hosting providers so that applications built on top of the components can be readily built/assembled and deployed.

VoiceXML Subdialogs

Probably the most adopted approach for creating reusable dialogs is through VoiceXML's `<subdialog>` element. From a definition perspective a subdialog element invokes a called dialog identified by the `src` element in the calling dialog. The subdialog is then created and executed in its own temporary execution context and proceeds with execution until it encounters a `<return>` element, at which point any information is returned to the calling dialog. Subdialogs can be parameterized through the `param` elements and are fundamental in building a set of reusable components for use within VoiceXML applications.

The following code snippet defines a subdialog that collects users' phone numbers and PINs and returns the information to the calling environment:

```
<vxml version="2.0">
  <form>
    <field name="phone" type="phone">
      <prompt>What is your phone
        number?</prompt>
    </field>
    <field name="pin" type="digits">
      ...
    <return namelist="phone pin"/>
    ...
  </form>
</vxml>
```

The subdialog can then be used by another application to authenticate a user, as shown by the following snippet:

```
<vxml version="2.0">
  <var name="phone"/>
  <var name="pin"/>
  <subdialog name="GetUserInfo" src=
    "GetUserInfo.vxml">
    <filled>
      <assign name="phone" expr=
        "GetUserInfo.phone">
        ...
      </filled>
    </subdialog>
  </vxml>
```

VoiceXML subdialogs behave somewhat differently from traditional programming languages. Subdialogs execute in a "temporary" execution context. They can take parameters, but don't follow the event percolation model (events must be handled within the subdialog itself or be explicitly returned). Unlike

external objects, subdialogs do limit what can be built as a component as they themselves also have to be VoiceXML code. However, subdialogs are also the only portable mechanism today to create reusable dialog components for VoiceXML applications. As we'll see later, subdialogs are the basis for the majority of vendor-specific reusable components as well.

VoiceXML `<object>` Tag

The motivation behind the `<object>` tag was to keep the VoiceXML-based implement extensible and to package advanced functionality that hasn't been introduced into the standard. A VoiceXML implementation platform (tool, gateway, and/or service provider) may expose additional functionality that's currently not available through the VoiceXML language itself.

A good example of this is detailed caller ID information service, which can be used to screen/identify/initialize the callers. Another example from the mobile world is a cell phone location service to provide emergency services. Boiling it down, components available through the object tag are entirely platform-dependent. The object element can pass parameters to the underlying subdialog through the `param` elements, which return an ECMAScript object that can be used to get the values returned.

```
<object
  name="location"
  classid="component://co.location
    /Location/GetDetails"
  data="LocationComponent.jar"/>
  <param name="region" expr="'US'">
</object>
```

Due to its component-oriented nature, the object tag lends itself to providing a methodology for creation of a reusable component. The key benefit of `<object>` tag-based components is that, unlike subdialogs, they aren't limited to implementation in VoiceXML itself and hence can be implemented in languages such as Java and C++. Another inherent benefit is that the object tag can help to create components that keep the intellectual property in the hands of the provider, as components can be delivered in binary format and you don't have to expose all the source code to the application developer.

Modularized VoiceXML

Complementary to the idea of creating of a set of reusable components is perhaps the modularity of the language itself. XHTML+Voice, acknowledged by

SUBSCRIBE AND SAVE

XML JOURNAL

Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|---------------------------|
| \$83.88 | |
| YOU PAY | |
| \$77.99 | |
| YOU SAVE | |
| \$5.89 | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of **XML-Journal** for only \$77.99! That's a savings of \$5.89 off the annual newsstand rate.

Sign up online at www.sys-con.com or call 1 800 513-7111 and subscribe today!

Here's what you'll find in every issue of XML-Journal:

- Exclusive feature articles
- Interviews with the hottest names in XML
- Latest XML product reviews
- Industry watch



SAVE 30% off the annual newsstand rate

JAVA DEVELOPER'S JOURNAL

Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|------------------------|
| \$71.88 | |
| YOU PAY | |
| \$49.99 | |
| YOU SAVE | |
| 30% | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of **Java Developer's Journal** for only \$49.99! That's a savings of \$21.89 off the cover price. Sign up online at www.sys-con.com or call 1 800 513-7111 and subscribe today!

In May *JDJ*:

Programming Neural Networks in Java

This article shows a simple, yet particle, neural network that can recognize handwritten letters and describes the implementation of a neural network in a small sample program.

Manifest Destiny

This article presents some of the issues involved with packaging Java code.

Using the Java Native Interface Productively

Although we try to make our applications pure Java, outside forces sometimes make this impossible. This article discusses supporting a C/C++ API in Java to enable a Java application to use it.



W3C, attempts to modularize the core VoiceXML language itself into a set of modules. These modules are then combined with XHTML 1.1 to build applications that support a combination of visual and speech interactions. The specification modularizes the VoiceXML 2.0 specification into about 19 different modules, such as events, executable statements, flow control, dialogs, menus, objects, telephony control, audio output and speech synthesis, resources, and so on. This allows different applications and environments to support specific scenarios that don't need the complete language. For instance, a multimodal PDA (without any telephony interface) could support a VoiceXML browser with features like simplified speech recognition and speech synthesis/audio playback but doesn't need to support the telephony control module.

As an example, let's look at Listing 1, the code snippet of a simple XHTML document that uses the proposed Voice Profile modules. Whereas a traditional telephony-based VoiceXML browser could understand it by playing the Hello World prompt, a multimodal device would render a page with the paragraph. However, the paragraph can be "listened to" by clicking on it as well.

It will be interesting to see how the modularity of VoiceXML itself (as proposed by the XHTML+Voice) will work as it will be instrumental in modularizing the language itself so that it can be applied in multiple scenarios.

The remainder of this article will focus on understanding a couple of vendor-specific reusable component initiatives.

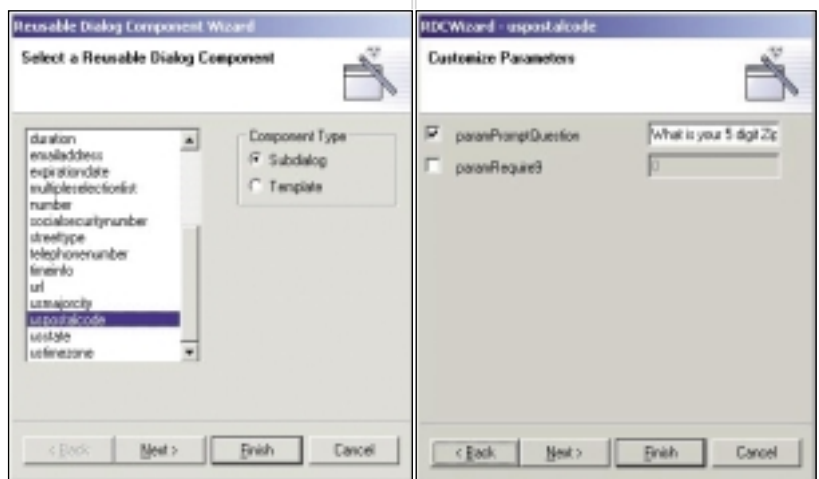
IBM Reusable Dialog Components

IBM Reusable Dialog Components are based on the VoiceXML subdialog element. They're built on top of grammars based on ECMAScript, VoiceXML,

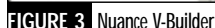
and JSGF (Java Speech Grammar Format) and are freely available at www3.ibm.com/software/speech/enterprise/wvs-rdc.html. The package includes the entire source code of the various dialog components including JSGF-based grammars, VoiceXML dialog code, and samples for using the dialogs. The availability of the entire source code allows developers to further customize these dialogs based on individual requirements for, for example, a regional application in which you'd like to recognize the cities in your region. This can be done by customizing just the grammars for the cities and using the dialog component for "US Major City". IBM provides reusable dialog components for confirmation dialogs, selection lists, getting user input for alphanumeric strings, credit card information, currency, date, direction, duration, e-mail addresses, numbers, SSN, addresses, telephone numbers, URLs, major cities/states, and so on.

IBM provides a methodology for creating custom reusable dialog components as well. Reusable dialog components are based on VoiceXML 1.0 and have been tested for IBM Voice products including IBM's WebSphere Voice Toolkit and Voice Server. A key highlight of reusable dialog components is that they're tightly integrated with IBM's VoiceXML development toolkit (see February 2002 *XML-J*, "Tools for Developing VoiceXML Applications," Vol. 3, issue 2). A couple of simple wizards (see Figures 1 and 2) allow the application developer to select a dialog component, customize it using the defined parameters, and integrate it with the VoiceXML application.

Listing 2 shows how a VoiceXML application generated through the toolkit uses the reusable dialog component "uspostalcode".



FIGURES 1 AND 2 IBM WebSphere Voice Toolkit wizards for inserting a reusable dialog component



Unlike IBM Reusable Dialog Components, SpeechObjects from Nuance utilize the <object> tag to provide reusable functionality. SpeechObjects, however, are self-contained and include prompts, dialog logic, and grammars. SpeechObjects are written using JavaBeans. Nuance also provides a methodology and a set of interfaces (as Java classes and APIs) for developers to write custom SpeechObjects. Currently available SpeechObjects from Nuance include confirmation dialogs, menus, browsable and actionable lists, audio recorder, dialogs for getting date, time, U.S. currency, time zone, SSN, zip code, telephone numbers, alphanumeric strings, credit card information, and the like as well as speech objects to integrate with Nuance products: speaker verification and enrollment and voice phrase enrollment.

SpeechObjects are also integrated with the Nuance development tool V-Builder (see Figure 3). In addition, BeVocal Café, a hosted VoiceXML development environment, gives developers the ability to test and use SpeechObjects as part of the VoiceXML applications.

Like IBM Reusable Dialog Components, DialogModules also use the subdialog approach to create a set of reusable speech components. However, they aren't shipped as static vxml, script, or grammar source files but as server-side J2EE/JSP application modules. Available DialogModules include confir-

A partial list of DialogModules is supported by VoiceGenie, a VoiceXML gateway and development tools provider. These DialogModules are available on VoiceGenie's SpeechWorks-centric developer site, SpeechGenie Developer Workshop (<http://speechgenie.voicegenie.com>), a hosted VoiceXML development and testing environment. The following code snippet shows the Yes/No dialog module used as part of the VoiceXML application.

```
<?xml version="1.0"?>
<vxml version="1.0">
  <form>
    <subdialog src=
      "http://dm.voicegenie.com/-
      osdm-core/jsp/yesno/wrapper.-
      jsp"
      name="yesno_1" method=
        "post"
      enctype="application/
        x-www-form-urlencoded">
      ...
    </subdialog>
  </form>
</vxml>
```

Let's face it, interactive speech-based application development is complex. As an open standard, VoiceXML truly did leverage developments in Interactive Voice Response (IVR), Advanced Speech Recognition (ASR), Text-to-Speech (TTS), and telephony integration and made it simpler for application developers to develop speech applications. However, this doesn't preclude the fact that we need dialog and speech recognition experts to create reusable and modular components that an application developer can reuse and build upon. Reusability and modularity are key components for VoiceXML adoption by the enterprises and independent software vendor community.

VoiceXML today does provide basic mechanisms for enabling reusability using the `<object>` tag and the ability to create Subdialogs. However, there is ample scope for an established standard/approach around reusable dialog and other speech components. What is also needed is a published library of such dialog components, so that as application and software developers we enrich the library and don't rebuild (for instance, there are a number of common dialog

SAVE 30% off the annual newsstand rate

wireless

BUSINESS & TECHNOLOGY

Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|------------------------|
| \$71.88 | |
| YOU PAY | |
| \$49.99 | |
| YOU SAVE | |
| 30% | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of **Wireless Business & Technology** for only \$49.99! That's a savings of 30% off the cover price. Sign up online at www.sys-con.com or call 1 800 513-7111 and subscribe today!

In May **WBT**:

Securing the WLAN 'Maginot Line'

The current state of security in most WLANs is no more than a Maginot Line.

GPS, Cellphones, and the Enterprise

Global Positioning Systems that work in cellphones, and indoors, will soon be ubiquitous and bring many benefits to enterprise resource management.

Trust in Wireless

NTRU and Texas Instruments explore a multifaceted approach to achieving wireless system-wide security and trust through the channel and down to each user's mobile device.

The Windup Phone

How can you recharge your phone when you're on the move?

SUBSCRIBE AND SAVE

WebServices JOURNAL

Offer subject to change without notice

| ANNUAL NEWSSTAND RATE | |
|-----------------------|------------------------|
| \$83.88 | |
| YOU PAY | |
| \$69.99 | |
| YOU SAVE | |
| \$13.89 | Off the Newsstand Rate |

DON'T MISS AN ISSUE!

Receive 12 issues of **Web Services Journal** for only **\$69.99!** That's a savings of **\$13.89** off the annual newsstand rate. Sign up online at www.sys-con.com or call **1 800 513-7111** and subscribe today!

In May **WSJ**:

Web Services for Enterprise Application Integration

What solutions are offered for enterprises today?

Powering Web Services Through Integration Technology

Agreement on standards will lead to a new era

J2EE, EAI, and Web Services

New challenges for your information systems

Will Web Services Mean the End for EAI?

Complementary approaches for peaceful coexistence

Data: A Key Part of Web Services

Keep the fundamentals for current environments and tools



components among the three vendors reviewed). However, until a standard approach and methodology is established and adopted, we do have vendor-specific approaches and reusable components that we can reuse today.

References

- **VoiceXML 2.0 (W3C Working Draft):** www.w3.org/TR/voicexml20/
- **IBM Reusable Dialog Components:** www-3.ibm.com/software/speech/enterprise/wvs-rdc.html
- **Nuance SpeechObjects:** www.nuance.com/products/speechobjects.html

- **BeVocal Café:** <http://cafe.bevocal.com>
- **SpeechWorks OpenSpeech Dialog Modules:** www.speechworks.com/products/speechrec/openspeechdialog-mod.cfm
- **SpeechGenie Developer Workshop:** <http://speechgenie.voicegenie.com>
- **W3C Reusable Dialog Requirements:** www.w3.org/TR/reusable-dialog-reqs
- **XHTML+Voice Profile 1.0 (W3C Note):** www.w3.org/TR/xhtml+voice/
- **XHTML 1.1 (W3C Working Draft):** www.w3.org/TR/xhtml11/

HKS @ HITESHSETH.COM

LISTING 1

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C/DTD HTML+Voice//EN"
"xhrml+voice10.dtd"> <html
xmlns="http://www.w3.org/1999/xhtml"
xmlns:vxml="http://www.w3.org/2001/voicexml20"
xmlns:ev="http://www.w3.org/2001/xml-events"
>
<head>
<title>Hello World</title>
<vxml:form id="SayHello">
  <block>
    <prompt>Hello World</prompt>
  </block>
</vxml:form>
</head>
<body>
<p ev:event="onclick" ev:handler="#SayHello">
Hello World
</p>
</body>
</html>
```

LISTING 2

```
<?xml version="1.0"?>
<vxml version="1.0">
<script src=
"reusable_comp/subdialogs/uspostalcode/en_US/
uspostalcode.es">
  var objUspostalcode = new USPostalCode();
</script>
<form>
  <subdialog name="uspostalcode" src=
"reusable_comp/subdialogs/uspostalcode/en_US/
uspostalcode.vxml">
    <param name="paramSubdialogObj"
      expr="objUspostalcode"/>
    <param name="paramPromptQuestion"
      expr="'What is your ZipCode?'" />
  </subdialog>
  ...
</form>
</vxml>
```

LISTING 3

```
<?xml version="1.0" ?>
<!DOCTYPE vxml PUBLIC "-//BeVocal Inc//VoiceXML 1.0//EN"
"http://cafe.bevocal.com/libraries/dtd/vxml1-0-bevocal.dtd">
<vxml version="1.0">
<form id="getStock">
  <object classid="speechobject://bevocal.cafe.SOPickStock"
    name="stock"/>
  <filled>
    <prompt>
      <value expr="stock"/> is trading at $N.
    </prompt>
  </filled>
</form>
</vxml>
```

DOWNLOAD THE CODE @
www.sys-con.com/xml

jAllora

by HiTSoftware

HiT Software provides a line of standards-based XML and SQL middleware for application development and systems integration. HiT's Allora product line provides bidirectional XML access to databases and supports standard DOM/SAX parsers and, optionally, is accessible via SOAP interfaces (JMS support is also available).

jAllora is the Java-based version of HiT's Allora middleware product and provides an easy-to-use and -understand approach to serializing relational data into an XML format (a version of Allora, winAllora, is also available for the Windows platform).

Pros

- Easy to install and use; APIs are well documented.
- Conforms to existing standards (supports DTD, XSD, SAX, DOM).
- Integrates into popular Java IDEs (Sun's Forte and Borland's JBuilder).
- The graphical mapper product made generating XML representations of relational data relatively simple.
- Excellent support for W3C XML Schema. I had no problems importing or working with schemas that contained well over 10,000 lines of code.
- Supports popular databases including (but not limited to) Oracle, DB2, MS SQL Server, Sybase.
- Support for Web services (includes WSDL templates and code samples).
- Low cost.

Cons

- Mapper can be a bit difficult to use. Mappings aren't displayed until you click on the fields being mapped.
- Can be used only by Java developers (obviously).
- Some of the features are included in the latest versions of some relational databases. jAllora faces the risk of becoming a commodity as database-vendor support for XML matures.

Overall

Recommended for large enterprises that don't want to implement competing RDBMS vendor XML integration methodologies. Also recommended for small to mid-sized enterprises that lack the time/skill sets to provide XML-based representations of their existing systems. Large-scale organizations may also want to implement a custom solution or use a full-featured integration server with support for nonrelational data adapters.

Background

In Italy the word *allora* is used as a sort of verbal warm-up for conversation. For example, a waiter may ask, "Allora...caffè, dolce?" ("So...coffee or dessert?"). While this use provides a jump-start for conversations, jAllora can help enterprises jump-start their support for XML.

jAllora includes several wizards for mapping relational structures into XML, including:

- A graphical drag-and-drop mapper for mapping relational tables, views, or custom queries/views into XML. The mapper creates files that describe the transformations between the database and XML.
- Source code wizards for jump-starting development.
- Object interfaces (including support for SOAP and Web services).
- Several well-documented code samples demonstrating marshaling/unmarshaling techniques, data binding, and message queue interface support.

Changes to the underlying database structure won't affect applications using jAllora, since the mapping files can be updated easily to reflect the changes (the mapper files used XML Schema to describe the mappings from relational structures into XML).

There are three options for deploying/using jAllora:

1. Using the APIs, developers can use jAllora to transfer data between XML and a relational database (or from a relational database into XML). jAllora's APIs are organized into the following packages:
 - **com.hitsw.xml.databinding:** Used to marshal/unmarshal data into (or out of) a relational database. Options include bulk operations (useful for transferring entire tables into XML), SQL-level operations (for record-specific processing), and data binding operations (enables the manipulation of tables and rows using Java objects). jAllora's data binding is based on the W3C DOM model, enabling developers to create objects representing database records while retaining the XML formatting structures.
 - **com.hitsw.xml.jms:** Provides support for JMS (Java Message Service), enabling XML to be sent or retrieved from a JMS queue.
 - **com.hitsw.xml.mapping:** Enables developers to generate Java objects for working with jAllora mapping files. These files are generated by the mapper and are used to translate XML elements and attributes into database columns or database columns into XML elements and attributes.
2. jAllora can be used as an automated XML data-binding engine using jAllora classes. Developers can use these classes directly in their applications to manipulate the underlying data. The objects can be self-populating from either a database table or an XML stream.

[REVIEWED BY JOHN EVDEMON]



Bio

John Evdemon has been working with XML, Java, and other programming languages and platforms far longer than he cares to remember. He has been a CTO at both XMLSolutions and Vitria and has contributed to several books on XML. A frequent speaker at industry conferences, he assisted in the development of the W3C XML Technical Recommendation. John is currently an independent consultant, helping enterprises develop and execute their XML and e-business strategies.

J EVDEMON @ A C M . O R G



HiT Software, Inc.

4020 Moorpark Ave., Suite 100
San Jose, CA 95117

Phone: 408 345-4001

E-Mail: info@hitsw.com

Web: hitsw.com/dsheets/jallora-dsf.htm

Test Environment

OS: Windows 2000 (Service Pack 2)
Processor: Pentium III, 250 MB RAM
Software: Java v1.3.1_01, Oracle 8i/Enterprise Edition, Apache Tomcat v3.2/SOAP v2.2, Forte for Java v3.0

3. jAllora also provides a SOAP interface. Developers can use the Apache SOAP API to invoke jAllora Web services from virtually any platform or programming language. Several well-documented code samples are provided, enabling new/inexperienced developers to get a reference implementation up and running with little effort. The Web services option, which requires purchase of a server license from HiT Software, uses JSP pages to manage data sources, group IDs, and passwords (it's not compatible with Netscape 6, but HiT is reportedly addressing this issue).

Installing

You can download an evaluation copy of jAllora at www.hitsw.com (in either zip or compressed tar formats). You have to register with a "real" e-mail address, since the license is sent in a separate e-mail. Licensing fees start at \$2,995.

Installation is fairly simple. Simply unzip the product and set a number of environment variables in the mapper's startup script. While the directions for modifying the script are fairly straightforward, packaging the product with an installation utility would make installation/configuration a bit easier for less technical users. In many companies the data administrators/metadata experts and software developers have very different backgrounds – the person defining the maps may not be familiar with the concept of setting up Java classpaths and environment variables.

Using

As already stated, useful, fully working code examples are provided, enabling developers to quickly understand and begin using the jAllora APIs. The APIs are surprisingly easy to use. Once an XML mapping file has been set up (described below), instantiating an XML recordset requires only a few lines of code:

```
recordset = XMLRecordsetBuilder.newInstance();
mapper = XMLSchemaMapperBuilder.newInstance(name/location of xml mapper file);
recordset.setMapping(mapper);
recordset.marshalXMLStream(jdbc/output information);
```

While the mapper provided a familiar user interface (see Figure 1), fields and elements mapped to one another had to be clicked individually to see the actual mapping. This could be confusing if you're working with very large schemas (such as an XML representation of an X12 purchase order – as seen in the figure) or modifying a mapper file constructed by another person. Luckily, the mapper files are generated in an XML format – it would be fairly easy to build a stylesheet to document the mappings from the database into XML (perhaps HiT might consider adding one to the product distribution).

The mapper also provides the ability to map fields/elements to literals or expressions. Expressions are built by combining elements, attributes, fields, operators, and functions in the mapper's Expression Builder wizard. While jAllora provides many built-in functions, there doesn't appear to be a way to define custom functions (though HiT plans to add scripting support for its next release).

jAllora's Web services implementation can be implemented using:

- **Standard Web service:** Pure SOAP communications; no client-side jAllora libraries needed.
- **Client-side convenience classes:** Requires installation of a jAllora JAR on the client. This avoids having to learn the SOAP API (calls are translated into SOAP for you).
- **Client-side jAllora API:** Requires installation of two jAllora-specific JARs on the client. This automates many tasks that must be

coded in the first and second options and provides the most granular (record-level) control over data marshaling and unmarshaling.

I used Tomcat to set up jAllora's Web services options. The documentation provided detailed, step-by-step instructions that ensured my modifications would work the first time.

While all the options appeared to work quite well, companies with experienced Java/XML developers would probably opt for the first one (since the client in the first option doesn't require any jAllora-specific footprint). A number of WSDL templates are provided to help kick-start the development process. jAllora interfaces can also be accessed using UDDI. While I didn't have time to sufficiently exercise the jAllora SOAP implementation (running SOAPBuilder's test cases would have been an interesting exercise), the sample code ran flawlessly and provided sufficient proof for most simple SOAP implementations. The sample jAllora Web services code also contains numerous comments, enabling new users to understand and quickly write their own services.

Bottom Line

jAllora provides a lightweight, easily implemented solution to relational database/XML transformations. The XML-based mapping files help isolate database schema changes, avoiding potentially expensive code rewrites. jAllora's approach to SOAP and Web services supports both highly experienced developers (via the standard Web service) and less experienced programmers (via the client-side convenience classes and client-side jAllora APIs). If you're looking for a quick and easy solution that can transform your legacy data into XML, jAllora should be on your short list. ☺

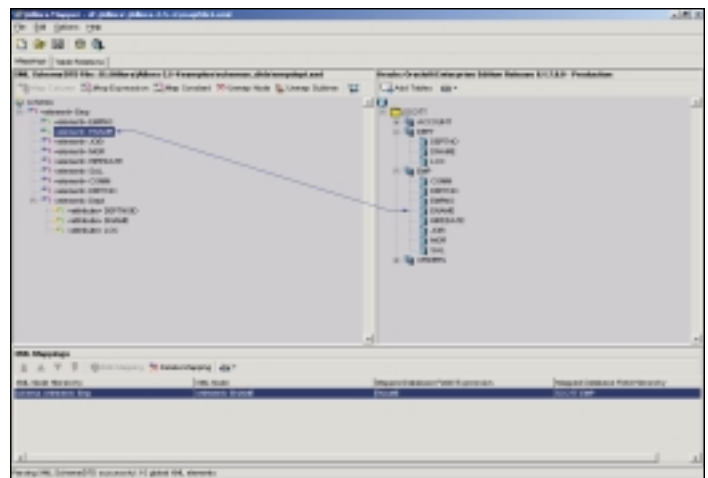
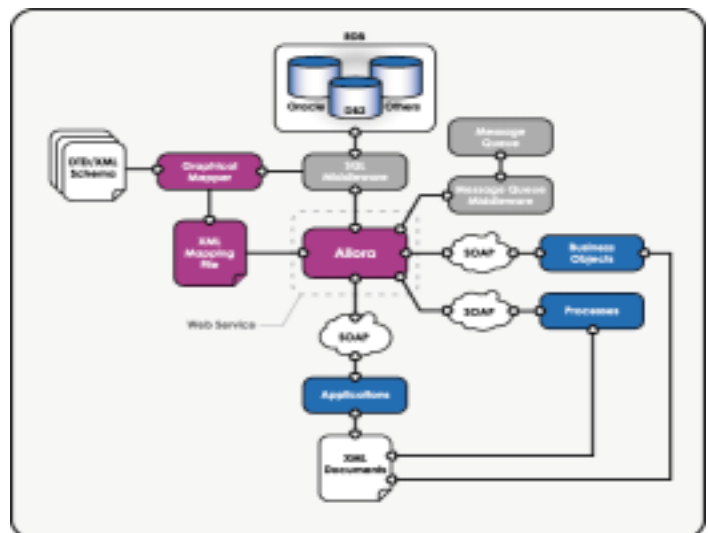


FIGURE 1 XML representation of X12 purchase order



XML-J ADVERTISER INDEX

| ADVERTISER | URL | PHONE | PAGE |
|-------------------------------------|--|------------------------|----------------|
| Altova | www.altova.com | | 64 |
| engindata Research | www.engindata.com | | 23 |
| eXcelon | www.exln.com | 800-962-9620 | 9 |
| Dynamic Buyer | www.ibm.com/smallbusiness/dynamicbuyer | 800-426-7235 ext. 5014 | 47 |
| iWay Software | www.iwayssoftware.com/guaranteed | 866-297-4929 | 6 |
| Java Developer's Journal | www.sys-con.com | 800-513-7111 | 54 |
| JDJ EDGE Conference & Expo | www.sys-con.com | 201-802-3069 | 27 |
| JDJ Store | www.jdjstore.com | 888-303-JAVA | 21, 37 |
| Simplex Knowledge | www.skc.com | 845-620-3700 | 43 |
| Sonic Software Corporation | www.sonicsoftware.com | | 2, 3 |
| Sprint PCS | http://developer.sprintpcs.com | | 4 |
| SYS-CON Media | www.sys-con.com | 800-513-7111 | 15, 25, 48, 49 |
| SYS-CON Reprints | www.sys-con.com | 201-802-3026 | 45 |
| Web Services Edge Conference & Expo | www.sys-con.com | 201-802-3069 | 29-31 |
| Web Services Journal | www.sys-con.com | 800-513-7111 | 56 |
| WebLogic Developer's Journal | www.weblogicdevelopersjournal.com | 800-513-7111 | 25 |
| WebSphere Developer's Journal | www.sys-con.com/websphere | 800-513-7111 | 41 |
| Wireless Business & Technology | www.sys-con.com/wireless | 800-513-7111 | 55 |
| XML EDGE Conference & Expo | www.sys-con.com | 201-802-3056 | 35 |
| XML Global Technologies, Inc. | www.xmlglobal.com/yourinformation | 800-201-1848 ext.210 | 63 |
| XML-Journal | www.sys-con.com | 800-513-7111 | 53 |

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of *XML-Journal*. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in *XML-Journal*. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.






www.wbt2.com

SUBSCRIBE NOW

www.javadevelopersjournal.com

TO THE

www.sys-con.com/xml

FINEST

www.coldfusionjournal.com

TECHNICAL

www.sys-con.com/pbdj

JOURNALS

www.webspheredevlopersjournal.com

IN THE

www.wldj.com

INDUSTRY!

www.wsj2.com






subscribe online www.sys-con.com or call 800 513-7111



wireless | java | xml | coldfusion | powerbuilder | websphere | weblogic | web services

Wait till you see June...

XML JOURNAL

Mapping XML Technology to Business


SECTION-SPECIFIC COVERAGE:

Content Management • Data Management • Enterprise Solutions • XML Lab


ENHANCED industry and technical coverage...

Expanded BUSINESS focus...

Dynamic new EDITORIAL TEAM

Ipedo Releases Database 3.0 (Redwood City, CA) – Ipedo, Inc., has introduced the newest version of its flagship product, featuring a fully W3C-compliant XML Query implementation, new XML Schema evolution capabilities, unique XML document versioning, improved SOAP support, and other performance and scalability improvements. 
www.ipedo.com

New Software Module for Real-Time Business Visibility (San Mateo, CA) – Persistence Software, Inc., is now shipping EdgeXtend DirectAlert 1.0, a data- and event-monitoring tool for a real-time view into daily business operations.

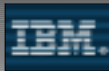
Based on Persistence's distributed dynamic caching technology, DirectAlert is an application server add-on that allows businesses to build applications that provide up-to-the-minute information regarding critical facets of their business and distribute changing information directly to decision makers, wherever they are. 
www.persistence.com

OASIS Committee to Define XML Standard for Biometrics (Boston) – The OASIS membership has formed the OASIS XML Common Biometric Format (XCBF) Technical Committee to provide a standard XML schema for biometrics. XCBF will describe information that verifies identity based on human characteristics such as DNA, fingerprints, iris scans, and hand geometry. It will be used in biometric applications that measure attendance, grant access control to documents or other resources, and facilitate nonrepudiation in commerce, particularly over open networks.

Participation in the OASIS XCBF Technical Committee remains open to all organizations

IBM Offers Another First: Production Printer Support for XML

(Boulder, CO) – IBM has announced Print Services Facility v3.3 for OS/390 and z/OS to enable printing directly from XML applications to high-speed production printers while guar-



anteeing output fidelity, print completion, and charge-back capability. PSF v3.3 is available as a no-charge upgrade for PSF v3 customers. www.ibm.com/printers

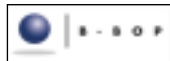
and individuals interested in advancing a standard XML schema for biometrics. OASIS will host an open mail list for public comment on XCBF, and completed work will be freely available to the public without licensing or other fees.



Information on joining OASIS can be found on www.oasis-open.org/join.

XMLCities, B-Bop to Offer Publishing Solution

(Milpitas, CA / Burlingame, CA) – XMLCities, Inc. and B-Bop Associates, Inc., have forged an alliance to jointly market a comprehensive solution for scalable XML-based content publishing.



XMLCities' XML conversion software, along with B-Bop's Xfinity XML data management server, offers the enterprise an easy-to-use solution to fully automate the conversion, creation, and delivery of large volumes of reusable XML content for e-learning, multichannel publishing, and online catalogs. Businesses can now convert their documents cost-effectively from proprietary formats to dynamic, Web-ready content.

www.b-bop.com
www.XMLCities.com

Cyclone Delivers Certified Support for ebXML (Scottsdale, AZ) – Cyclone Commerce claims to be the first

company to deliver a commercially available electronic business XML solution.

ebXML represents the leading effort to bring consistency and unification to business processes and transactions that previously industry-wide standards initiatives have not yet adequately addressed.




Through Cyclone's new Open Business Connections suite, customers can now engage in ebXML-based initiatives while simultaneously ramping communities of partners using other open, secure messaging standards. www.cyclonecommerce.com

Reliance Forms Partnership with IXIASOFT

(Boston / Montreal) – IXIASOFT and Reliance Technology have formed a strategic partnership involving solution development, reselling, and technical integration between Reliance Technology's Any2XML and IXIASOFT's TEXTML Server. The alliance also encompasses joint sales and marketing activities. www.ixiasoft.com
www.goreliance.com



Manning Publications Releases How-to XML Guide (Greenwich, CT) – J2EE and XML Development by Kurt Gabrick and Dave Weiss is a concise guide that teaches how, where, and why to use XML in each layer of a J2EE

application. The book categorizes and explains many recent Java and XML technologies and the ways in which a J2EE application can best use them. It untangles the web of Java APIs for XML, including the JAX family, as well as other popular emerging standards like JDOM, explaining each in terms of its functionality and illustrating its intended use through examples. 
www.manning.com

Docucorp Updates Document Management Suite

(Dallas) – Docucorp International has released Documanage 6.0, a new version of its document management and archive solution suite. Highlights include integrated XML, expanded database support, integrated viewing with annotation for all file types, a revamped user interface, and extended document properties.

Other new features are improved application administration and logon, automated and centralized printer resources, future-proofed sequential volume storage, and enhanced document migration and retention utilities. www.docucorp.com



Ford Motor Company CIO Joins NeoCore Board

(Colorado Springs, CO) – NeoCore has added Marvin W. Adams, vice president and chief information officer of Ford Motor Company, to its board of directors.

Adams became interested in NeoCore via the company's major investor, Baker Capital of New York, based on his interest in the problem of how to effectively store an increasing volume of data transmitted as XML. He holds a bachelor's degree in electrical engineering from Michigan State University. www.neoconet.com



XML NEWS

Media Fusion Announces Xweaver Plug-in

(San Jose, CA) – Media Fusion USA, Inc., has released Xweaver, an XSLT/XML plug-in for Macromedia's Dreamweaver, for professional Web page development.



Xweaver increases development productivity and produces rich, in-style XSLT/XML Web pages instantly.

www.mediafusion-usa.com

ContentGuard Offers XrML to Oasis

(Bethesda, MD) – ContentGuard, Inc., a leading provider of digital rights language technology, is contributing the eXtensible rights Markup Language (XrML) to OASIS, the XML interoperability standards consortium, for long-term development and governance of the rights language. This contribution is in support of the OASIS Rights Language Technical Committee, which was formed to advance a common XML digital rights language standard for the digital rights management marketplace.



www.contentguard.com

ACOM Updates E-Commerce Management Solution

(Long Beach, CA) – ACOM Solutions has released version 6.5 of its modular EZConnect EDI-XML B2B e-commerce management solution. It incorporates a graphical user interface redesign for greater ease of use and smoother application flow; context-sensitive, hypertext-based online help



that's accessible from any point in the application; report redesigns that assure compatibility with latest versions of major back-end solutions; and new database and EDI mapping functions.

www.acom.com

Tek-Tools Ships Storage Profiler 2.0

(Dallas) – Tek-Tools, Inc., is shipping Storage Profiler 2.0, a high-performance, Web-based storage resource management (SRM) solution with enhanced support for EMC Corporation's Symmetrix enterprise storage systems.

Tek-Tools' storage profiler is a cross-vendor, enterprise-level solution providing automated usage-trend reporting, capacity planning, and real-time monitoring for storage networks. Version 2.0 introduces support for operator-friendly, location-independent monitoring and reporting for



www.tek-tools.com

Insurance Data and Service Suite from e-Nable

(Westwood, MA) – e-Nable Corporation has released version 1.8 of its Insurance e-Nable suite of Internet-based insurance data and service offerings. The product allows users access to exclusive real-time MIB data, e-Nable prescription profile pharmaceutical data, motor vehicle records, paramed orders, APS, and inspection reports.

Version 1.8 also introduces e-Nable's MIB simplified interface, which provides, on an ASP basis, the ability to take a simple HTTP post of the required data elements to search the MIB data-



base and return either XML- or custom-formatted data.

www.e-nable.com

New Version of WebIntelligence from Business Objects

(San Jose, CA) – The newest version of Business Objects' query, reporting, and analysis tool, WebIntelligence 2.7, provides Web access to all major relational and online analytical processing (OLAP) sources, intelligent navigation between OLAP and relational data, and additional query options.



WebIntelligence 2.7 uses a new XML interface and workflow for an open, metadata-based approach to transferring queries between sources, thereby ensuring the correct contextual translation between the summarized OLAP data and the generated detailed relational report.

www.businessobjects.com

enumerate Releases Numerator! Publish

(McLean, VA) – enumerate Solutions Inc. reportedly has a first-of-its kind software package for posting data on the Web in an interactive format that speeds and simplifies tasks commonly performed when analyzing data (e.g., making charts, formatting spreadsheets, adjusting data).

Visitors to Web sites featuring Numerator! Publish can create charts and tables simply by clicking their mouse. From a standard Web browser users can add or remove data from their displays, apply equations instantly, transform data views, compare data from different

data sets, and share their analysis with colleagues.

www.enumerate.com

Shana Offers Next-Gen Management Solution

(Edmonton, Canada) – Shana Corporation, a leading provider of e-forms-driven e-business solutions, is offering its next-generation Web-based e-forms management solution.

Informed Quadra is an online solution that delivers intelligent e-forms to an organization's internal users as well as public users on the Internet. Informed Quadra features Windows XP and Mac OS X design environments, rich XML e-forms data and templates, and a browser-centric e-forms distribution center.



www.shana.com

Pramati Technologies, CARNOT in Partnership

(Mesa, AZ) – Pramati Technologies and CARNOT Inc. have announced a technology partnership to provide system integrators and ISVs with a more complete offering of J2EE/XML application development tools.

The partnership centers around Pramati Server and CARNOT's Global eProcess Management



Solutions (GeMS), the respective core J2EE products of Pramati and CARNOT, Inc. CARNOT's GeMS has been ported to run on Pramati's application server, and joint marketing activities will be developed and executed.

www.pramati.com

www.carnot-usa.com



SilverStream Releases eXtend Composer 3.5



(Billerica, MA) –

SilverStream Software, Inc., has announced the general availability of SilverStream eXtend Composer 3.5, an XML integration server that enables organizations to maximize the potential of new and existing business systems by transforming them into Web services. A cornerstone of

the SilverStream

eXtend product suite, the software includes the first commercially available process manager designed to build and orchestrate complex Web services interactions based on IBM's Web Services Flow Language (WSFL) specification.

www.silverstream.com



XML Can Help Keep Track of Your Life...

...Old and new



Graduating Guy was president of the Students Against Sweatshops organization at NYU. A strong voice for equal rights, he spoke at many gay rights rallies and almost any other social reform event. Newly graduated, he now works at Andersen Consulting and plans to vote Republican in the next election.

BY A COLLEGE-GRADUATING LIBERAL GUY TURNING REPUBLICAN

Wow, have you seen these tax rates? They're outrageous! I just graduated from college this month, and with my first real paycheck I feel righteous for having protested against the government during all my years of higher learning. I mean, I was told I'd be paid a certain amount a week, but thanks to taxes I don't even come close to that amount! Next election, whoever gets into office better do something about this. That's why I'm using XML to track the candidates and the tax programs each voted for.

Sure, I could use almost any database program to do it, but I want to get the word out on this, so I use XML. I've created a Web site where I've put this XML-based database document. Now anyone, regardless of platform or program, can view the data. That's because I've also put up an XSL stylesheet that users can download. The stylesheet determines whether to display the database information for Microsoft Excel (which I encourage others to use, since this software diversity thing is a pain for me to keep up with), Corel Paradox database software, or even Macintosh office programs. I haven't created stylesheet conditions to deal with open source StarOffice databases, though. That open source stuff is a little too much trouble to deal with.

People tell me that I've abandoned my "save the world" stance. Not true at all. I'm sensitive to the world's needs, now more than ever. In fact, especially now, since I've met the woman of my dreams, Amanda, the legal department girl who looks out for our business practices. We're

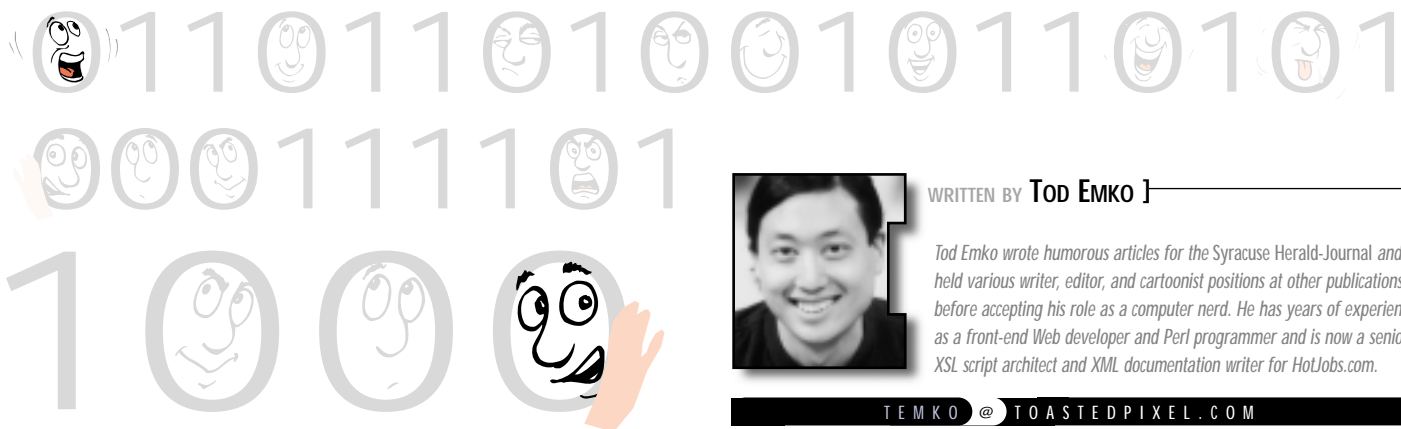
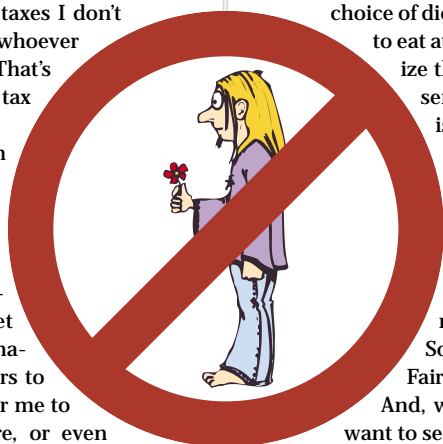
getting married in Aruba, and I want the world we live in to be a great and peaceful one. That's why I've begun to support candidates who want more military spending. After all, we can no longer be naïve enough to think everyone in the world likes us.

I'm also using XML to track all my donations to the Salvation Army. Excel is the official database program everyone I know uses, but I find it unwieldy sometimes. That's when I use XML. For my Salvation Army donations I simply add descriptive attributes to my database elements. That's useful when I need to insert a hard-to-categorize description. I still feel good about giving to the less fortunate, and I can get a real savings on next year's taxes by keeping track of all this stuff. It's also good to get rid of all those tie-dyes and band T-shirts I don't wear anymore.

Come to think of it, I've been making a lot of changes lately, like my choice of diet. I was a vegetarian in college, but now I can afford to eat at nice places like Chanterelle or La Bernardin. I realize that I can go to those restaurants as a liberal representative. I need to show that expensive French food isn't just for stuffy, old, unethical people. I even informed the restaurant owners of my XML Web site so they can check out next election's candidates. That'll help win them over to our humanitarian causes.

Once I got my XML databases going, I decided to use them for everything, like organizing my music. I used to listen to a lot of folk bands like Jill Sobule, Brenda Kahn, and, well, any band from Lilith Fair. Now I listen to Hans Zimmer and modern rock. And, with XML, it's a snap to keep track of which CDs I want to sell and how much I can expect to get for them.

So, in a nutshell, with graduation comes change, sure. And XML can document the changes I make, easily and cross-platform, for the world to see. But I'm still the same caring person I was before. I'm still as thoughtful and self-aware as I was in college. Take my voting, for instance. I still don't vote along party lines; I still vote by issue. And in the next election, if Bush doesn't offer another tax refund, I'm not voting for him. ☹



WRITTEN BY TOD EMKO }

Tod Emko wrote humorous articles for the Syracuse Herald-Journal and held various writer, editor, and cartoonist positions at other publications before accepting his role as a computer nerd. He has years of experience as a front-end Web developer and Perl programmer and is now a senior XSL script architect and XML documentation writer for HotJobs.com.

TEMKO @ TOASTEDPIXEL.COM

XML Global Technologies, Inc.

www.xmlglobal.com/yourinformation

Altova

www.altova.com